Stat 521A
Lecture 1
Introduction; directed graphical models

Outline

- Administrivia
- Overview
- Local markov property, factorization (3.2)
- Global markov property (3.3)
- Deriving graphs from distributions (3.4)

Administrivia

- Class web page

  www.cs.ubc.ca/~murphyk/Teaching/Stat521A-spring08
- Join groups.google.com/group/stat521a-spring09
- Office hours: Fri 10-11 am
- Final project due Fri Apr 24th
- Weekly homeworks
- Grading
  - Final project: 60%
  - Weekly Assignments: 40%

Auditing

- If you want to ‘sit in’ on the class, please register for it as ‘pass/fail’; you will automatically pass as long as you show up for (most of) the class (no other requirements!)
- If you take it for real credit, you will likely learn more…

Homeworks

Weekly homeworks, out on Tue, due back on Tue

- Collaboration policy:
  - You can collaborate on homeworks if you write the name of your collaborators on what you hand in; however, you must understand everything you write, and be able to do it on your own
- Sickness policy:
  - If you cannot do an assignment, you must come see me in person; a doctor's note (or equivalent) will be required.

Workload

- This class will be quite time consuming.
- Attending lectures: 3h.
- Weekly homeworks: about 3h.
- Weekly reading: about 10h.
- Total: 16h/week.

Pre-requisites

- You should know
  - Basic applied math (calculus, linear algebra)
  - Basic probability/ statistics e.g. what is a covariance matrix, linear/logistic regression, PCA, etc
  - Basic data structures and algorithms (e.g., trees, lists, sorting, dynamic programming, etc)
  - Prior exposure to machine learning (eg CS540) and/or multivariate statistics is strongly recommended

Textbooks

- “Probabilistic graphical models: principles and techniques”, Daphne Koller and Nir Friedman (MIT Press 2009, in press).
- We will endeavour to cover the first 900 (of 1100) pages!
- Copies available at Copiesmart copy center in the village (next to McDonalds) from Thursday
- I may hand out some chapters from Michael Jordan’s draft book, “Probabilistic graphical models”
- I am writing my own book “Machine learning: a probabilistic approach”; I may hand out some chapters from this during the semester.

Matlab

- Matlab is a mathematical scripting language widely used for machine learning (and engineering and numerical computation in general).
- Everyone should have access to Matlab via their CS or Stats account.
- You can buy a student version for $170 from the UBC bookstore. Please make sure it has the Stats toolbox.
- Matt Dunham has written an excellent Matlab tutorial which is on the class web site – please study it carefully!

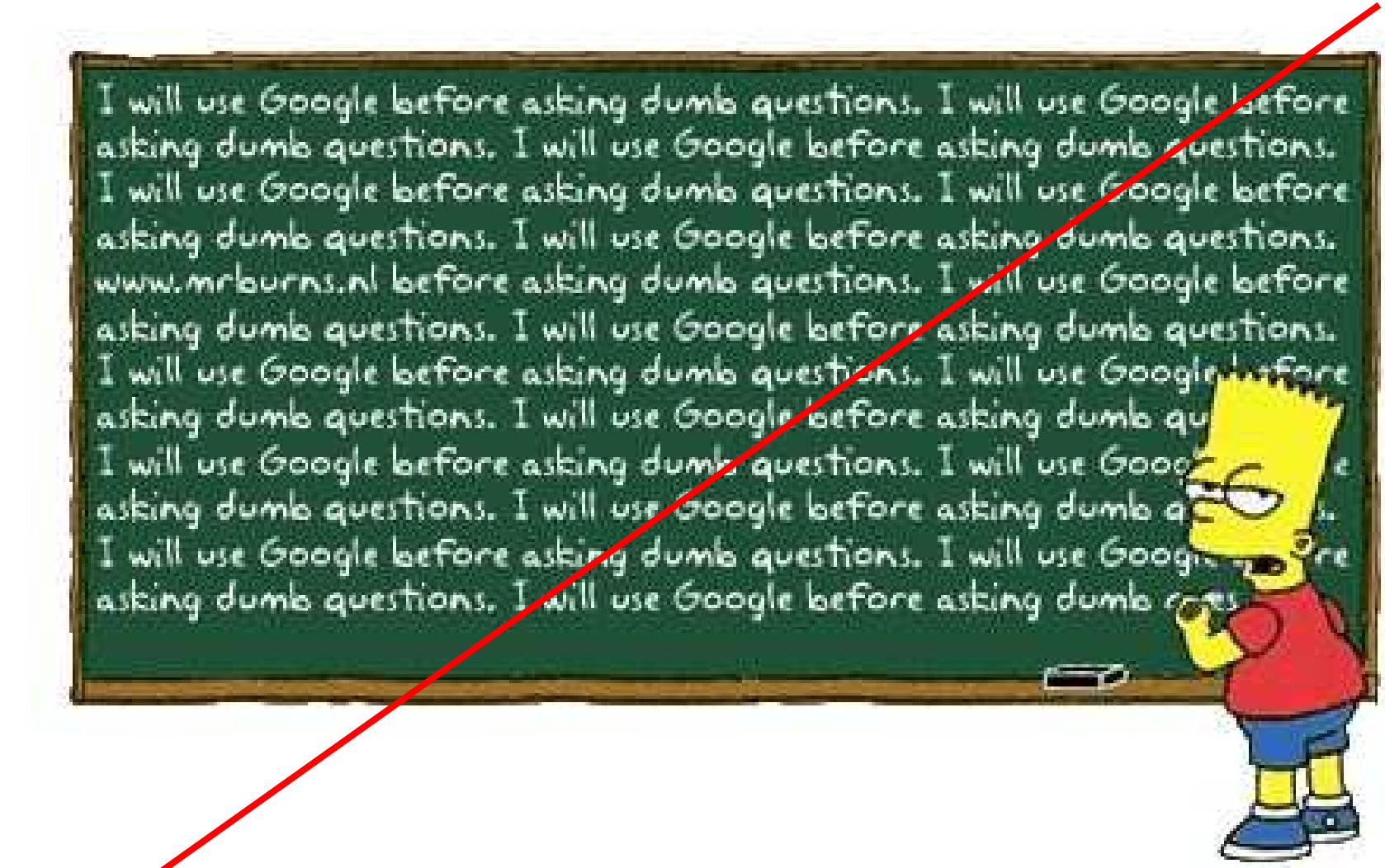
PMTK

- Probabilistic Modeling Toolkit is a Matlab package I am currently developing to go along with my book.
- It uses the latest object oriented features of Matlab 2008a and will not run on older versions.
- It is designed to replace my earlier ‘Bayes net toolbox’.
- PMTK will form the basis of some of the homeworks, and may also be useful for projects. (Currently support for GMs is very limited.)
- http://www.cs.ubc.ca/~murphyk/pmtk/

Learning objectives

- By the end of this class, you should be able to
  - Understand basic principles and techniques of probabilistic graphical models
  - Create suitable models for any given problem
  - Derive the algorithm (equations, data structures etc) needed to apply the model to data
  - Implement the algorithm in reasonably efficient Matlab
  - Demonstrate your skills by doing a reasonably challenging project

Ask questions early and often!

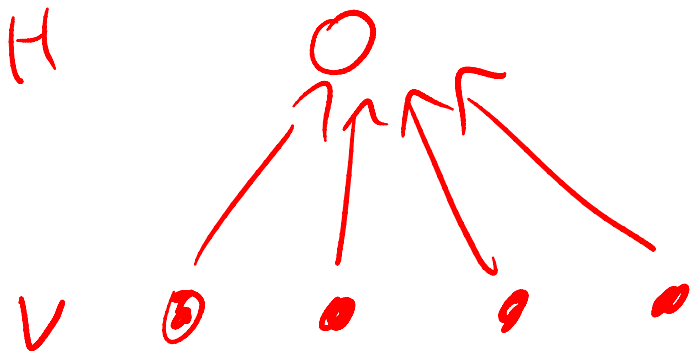


Outline

- Administrivia
- Overview
- Local markov property, factorization (3.2)
- Global markov property (3.3)
- Deriving graphs from distributions (3.4)

Supervised learning

- Predict output given inputs, ie compute p(h|v)
- Regression: h in R
- Classification: h in {1,…,C}



Structured output learning

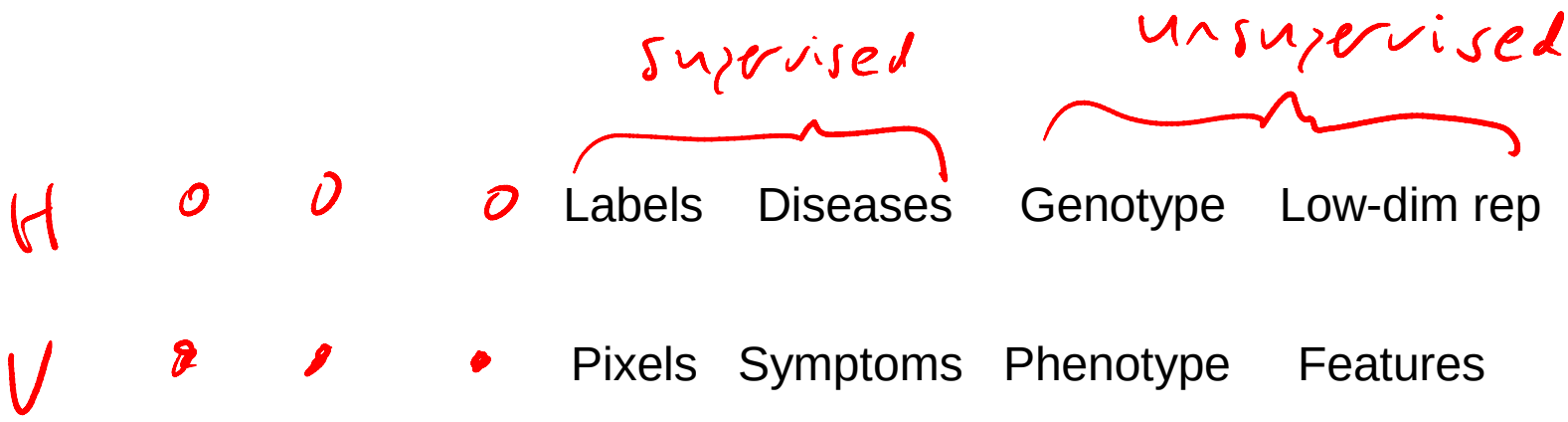
- Model *joint* density of p(h,v) (or maybe p(h|v))
- Then infer p(h|v) - state estimation
- MAP estimation (posterior mode)

$$\mathbf{h}^* = \arg\max_{h_1}, \ldots, \arg\max_{h_n} p(\mathbf{h}|\mathbf{v}, \boldsymbol{\theta})$$

- Posterior marginals

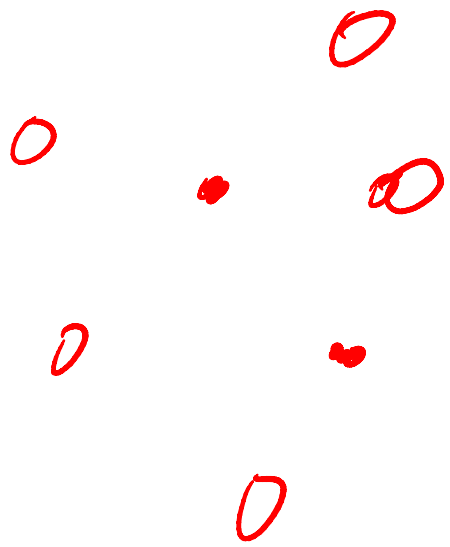
$$h_1^* = \sum_{h_2} \ldots, \sum_{h_n} p(\mathbf{h}|\mathbf{v}, \boldsymbol{\theta})$$

- Also need to estimate parameters and structure

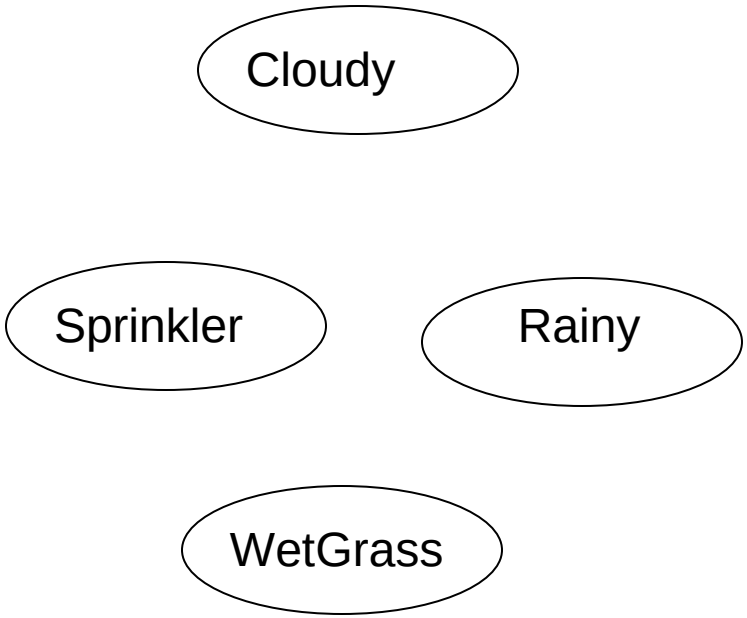


Density estimation

- Model joint density of all variables
- No distinction between inputs and outputs: different subsets of variables can be observed at different times (eg for missing data imputation)
- Can run model in any ‘direction’



Water sprinkler joint distribution



$$p(C, S, R, W)$$

c	s	r	w	prob
0	0	0	0	0.200
0	0	0	1	0.000
0	0	1	0	0.005
0	0	1	1	0.045
0	1	0	0	0.020
0	1	0	1	0.180
0	1	1	0	0.001
0	1	1	1	0.050
1	0	0	0	0.090
1	0	0	1	0.000
1	0	1	0	0.036
1	0	1	1	0.324
1	1	0	0	0.001
1	1	0	1	0.009
1	1	1	0	0.000
1	1	1	1	0.040

Inference

- Prior that sprinkler is on

$$p(S=1) = \sum_{c=0}^{1}\sum_{r=0}^{1}\sum_{w=0}^{1} p(C=c, S=1, R=r, W=w) = 0.3$$

- Posterior that sprinkler is on given that grass is wet

$$p(S=1|W=1) = \frac{p(S=1, W=1)}{p(W=1)} = 0.43$$

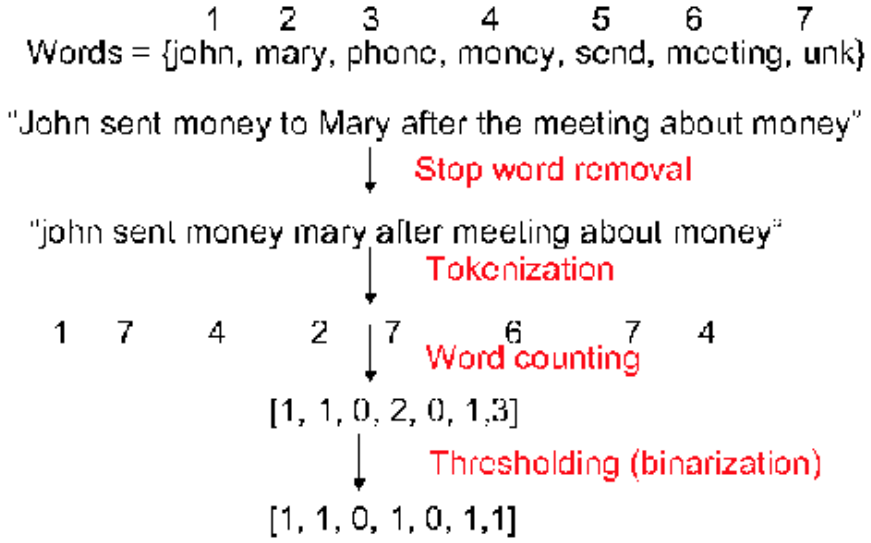
- Posterior that sprinkler is on given that grass is wet and it is raining

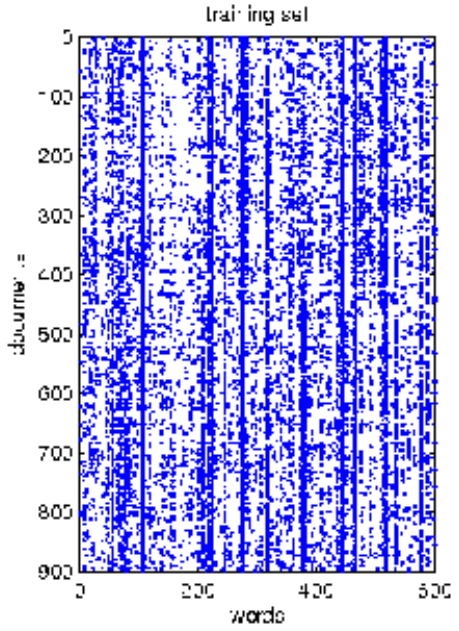
$$p(S=1|W=1, R=1) = \frac{p(S=1, W=1, R=1)}{p(W=1, R=1)} = 0.19$$

Explaining away

Bag of words model

- bag-of-words representation of text documents
- Xi=1 iff word i occurs in document
- Define a joint distribution over bit vectors, p(x1,…,xn)



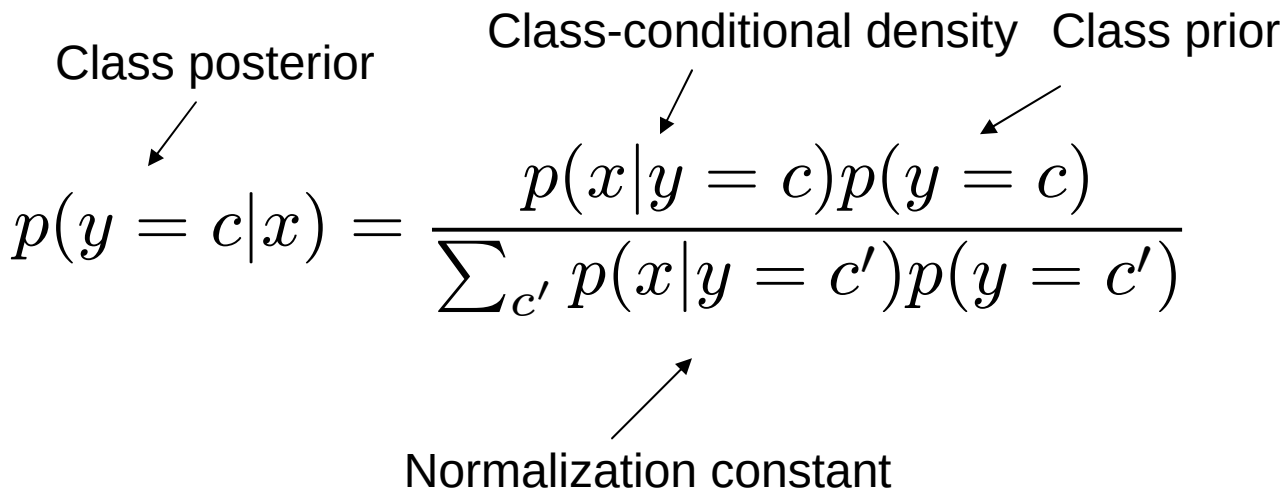


Inference

- Given word Xi occurs, which other words are likely to co-occur?
- What is the probability of any particular bit vector?
- Sample (generate) documents from joint p(x)

Bayesian classifiers

- Define joint p(y,x) = p(x|y) p(y) on document class label and bit vectors
- Can infer class label using Bayes rule



- If y is hidden, we can use this to cluster documents.
- In both cases, we need to define p(x|y=c)

Naïve Bayes assumption

- The simplest approach is to assume each feature is conditionally independent given the class/cluster Y

$$X_i \perp X_j | Y = c$$

- In this case, we can write

$$p(\mathbf{x}|y=c) = \prod_{j=1}^{d} p(x_j|y=c)$$

- The number of parameters is reduced from $O(C\,K^d)$ to $O(C\,K\,d)$, assuming C classes and K-ary features

Conditional independence

- In general, making CI assumptions is one of the most useful tools in representing joint probability distributions in terms of low-dimensional quantities, which are easier to estimate from data
- **Graphical models are a way to represent CI assumptions using graphs**
- The graphs provide an intuitive representation, and enable the derivation of efficient algorithms

Graphical models

- There are many kinds of graphical models
- Directed Acyclic graphs – “Bayesian networks”
- Undirected graphs – “Markov networks”
- Directed cyclic graphs – “dependency networks”
- Partially directed acyclic graphs (PDAGs) – “chain graphs”
- Factor graphs
- Mixed ancestral graphs
- Etc
- Today we will focus on DAG models

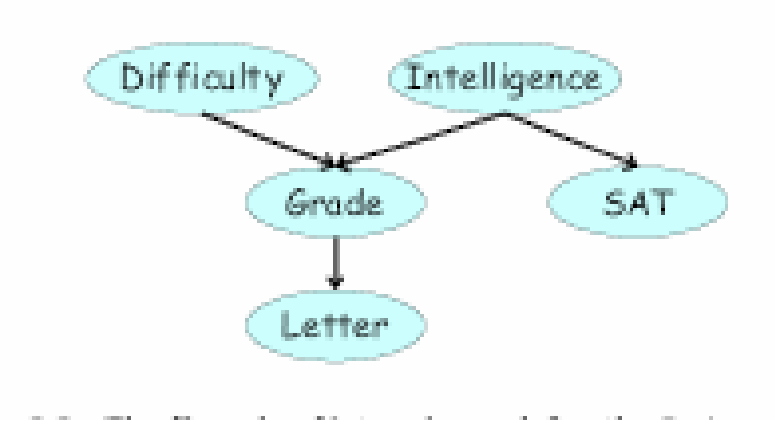
Outline

- Administrivia
- Overview
- → Local markov property, factorization (3.2)
- Global markov property (3.3)
- Deriving graphs from distributions (3.4)

CI properties of DAGs

- Defn 3.2.1. A BN *structure* G is a DAG whose nodes represent rvs $X_1,\ldots,X_n$. Let $Pa(X_i)$ be the parents of X_i, and $Nd(X_i)$ be the non-descendants of Xi. Then G encodes the following directed local Markov assumptions:

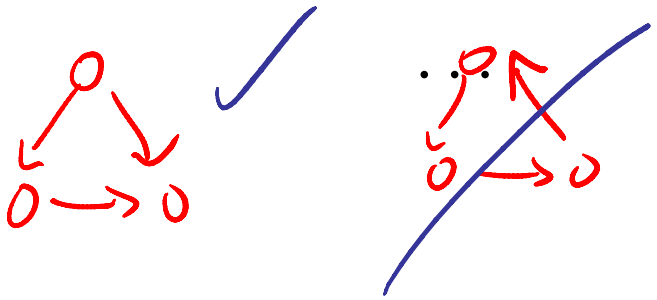
$$I_\ell(G) = \{X_i \perp Nd(X_i) | Pa(X_i)\}$$



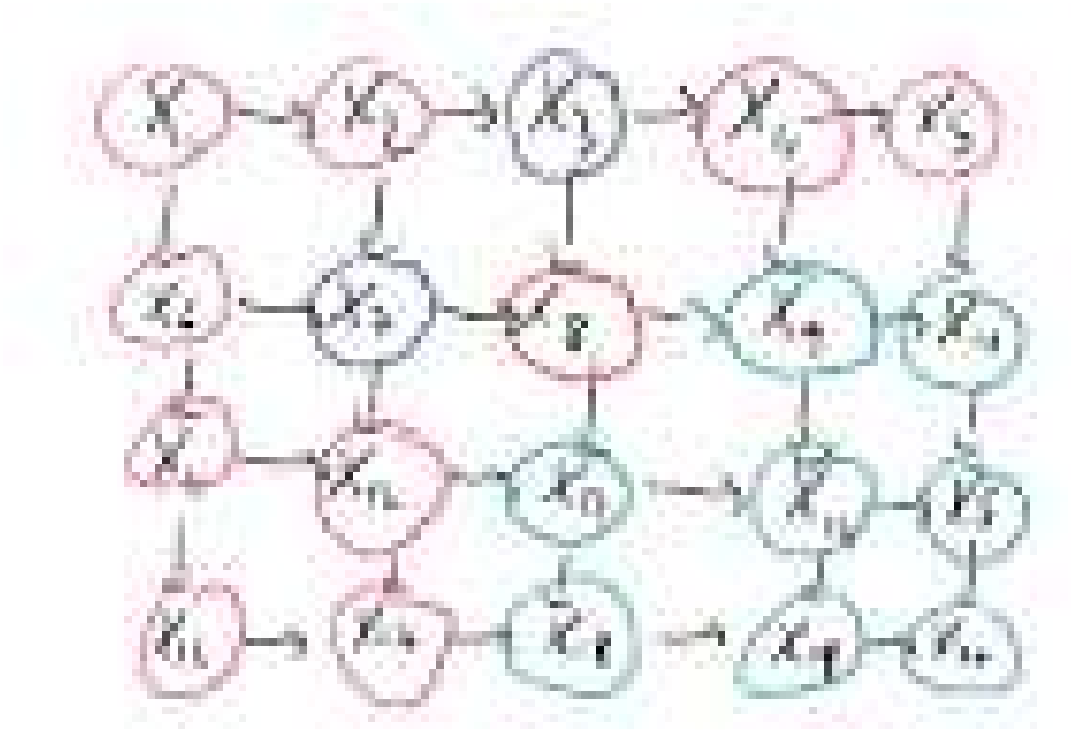
Student network

$$G \perp S | D, I$$

$$I \perp D$$



Another Example



Red (X8) ⊥ pink | blue

I-maps

- Def 3.2.2. Let I(P) be the set of independence assertions of the form X $\perp$ Y | Z that hold in P

$$P \models X \perp Y | Z$$

- Def 3.2.3. We say G is an I-map for set I if I(G) $\subseteq$ I

  (hence the graph does not make any false independence assumptions)

I-maps: examples

- Examples 3.2.4, 3.2.5

X	Y	$P(X,Y)$
x^0	y^0	0.08
x^0	y^1	0.32
x^1	y^0	0.12
x^1	y^1	0.48

X	Y	$P(X,Y)$
x^0	y^0	0.4
x^0	y^1	0.3
x^1	y^0	0.2
x^1	y^1	0.1

$$\begin{pmatrix} 0.08 & 0.32 \\ 0.12 & 0.43 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.6 \end{pmatrix} \begin{pmatrix} 0.2 & 0.8 \end{pmatrix}$$

$P(X,Y) = P(X)\,P(Y)$

$P \models X \perp Y$

Imaps = X Y, X -> Y, X <- Y

$P \not\models X \perp Y$

Imaps = X -> Y, X <- Y

I-map to factorization

- Def 3.2.5. A distribution P factorizes over a DAG G if it can be written in the form

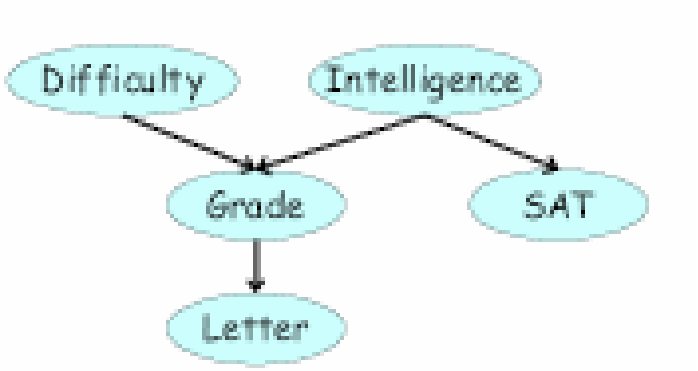
$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i | Pa(X_i))$$

- Thm 3.2.7. If G is an I-map for P, then P factorizes according to G.
- Proof: by the chain rule, we can always write

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i | X_{1:i-1})$$

- By the local markov assumption, we can drop all the ancestors except the parents. QED.

Student network



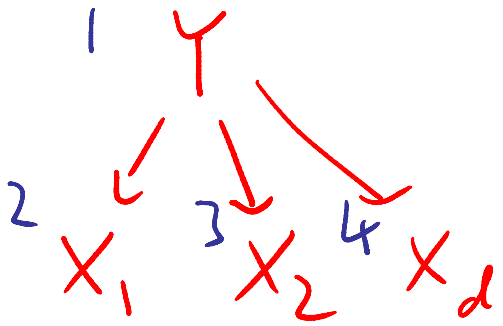
$$p(I, D, G, S, L) =$$

$$p(I)p(D|I)p(G|I, D)p(S|I, D, G)p(L|I, D, G, S)$$

$$= \quad p(I)p(D|)p(G|I, D)p(S|I)p(L|S)$$

$p(L|G)$

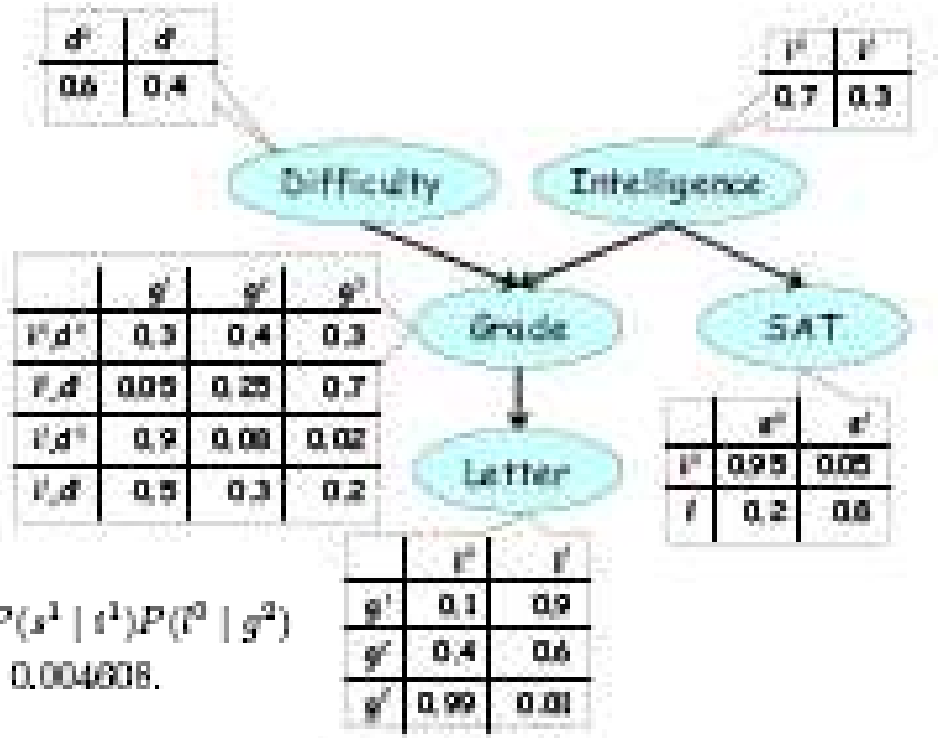
Naïve Bayes classifier



$$p(y, \mathbf{x}) = p(y) \prod_{j=1}^{d} p(x_j|y)$$

Bayes net = DAG + CPD

- A DAG defines a family of distributions, namely all those that factorize in the specified way.
- Def 3.2.6. A Bayes net is a DAG G together with a set of local Conditional Probability Distributions p(X_i|Pa(X_i)).

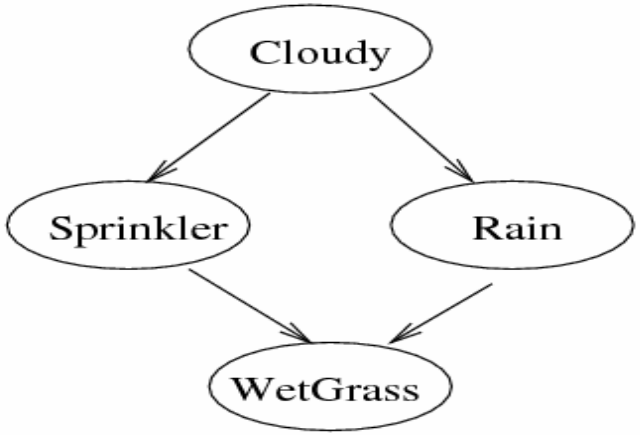


CPTs:
Each row is a different multinomial distribution,
One per parent combination

$$P(i^1, d^0, g^2, s^1, l^0) = P(i^1)P(d^0)P(g^2 \mid i^1, d^0)P(s^1 \mid i^1)P(l^0 \mid g^2)$$
$$= 0.3 \cdot 0.6 \cdot 0.08 \cdot 0.8 \cdot 0.4 = 0.004608.$$

Water sprinkler BN

P(C=F)	P(C=T)
0.5	0.5



C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

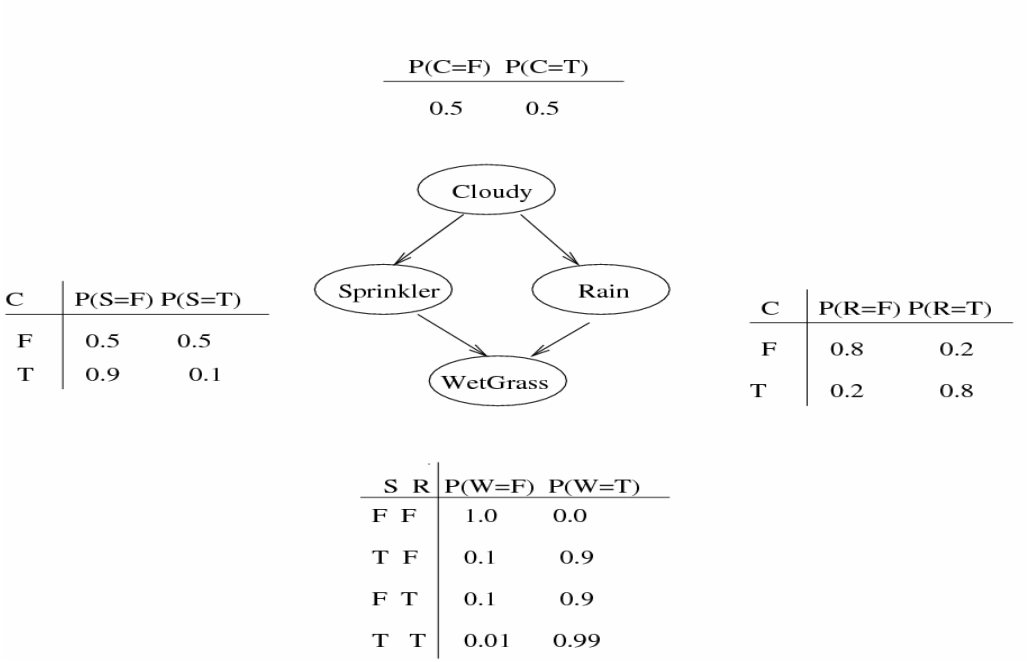
C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

$$p(C, S, R, W) = p(C)p(S|C)p(R|C)p(W|S, R)$$

Joint distribution for sprinkler network

$$p(C, S, R, W) = p(C)p(S|C)p(R|C)p(W|S, R)$$



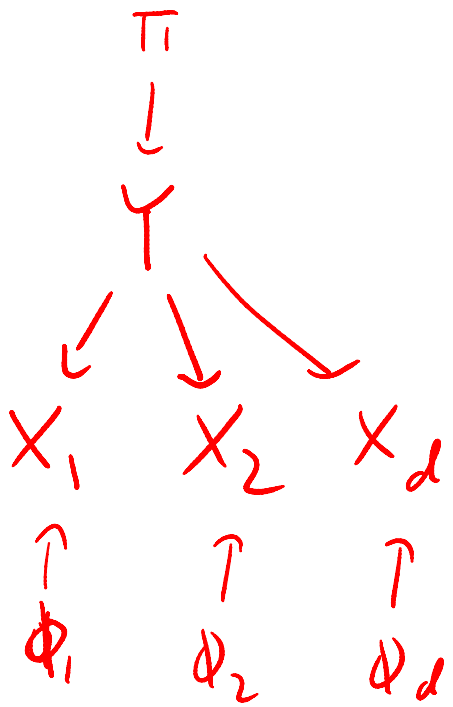
c	s	r	w	prob
0	0	0	0	0.200
0	0	0	1	0.000
0	0	1	0	0.005
0	0	1	1	0.045
0	1	0	0	0.020
0	1	0	1	0.180
0	1	1	0	0.001
0	1	1	1	0.050
1	0	0	0	0.090
1	0	0	1	0.000
1	0	1	0	0.036
1	0	1	1	0.324
1	1	0	0	0.001
1	1	0	1	0.009
1	1	1	0	0.000
1	1	1	1	0.040

CPDs

- CPDs can be any conditional distribution p(X_i|Pa(X_i))
- If Xi has no parents, this is an unconditional distribution
- For discrete variables, it is common to use tables (conditional multinomials)
- However, CPTs have $O(K^{|pa|})$ parameters; we will consider more parsimonious representations (such as logistic regression) – see ch 5
- For continuous variables, it is common to use linear regression to define CPDs (see ch 7)

$$p(X_i|Pa(X_i) = \mathbf{u}, \boldsymbol{\theta}_i) = \mathcal{N}(X_i|\mathbf{u}^T\boldsymbol{\theta}_i, \sigma_i^2)$$

Representing parameters as nodes



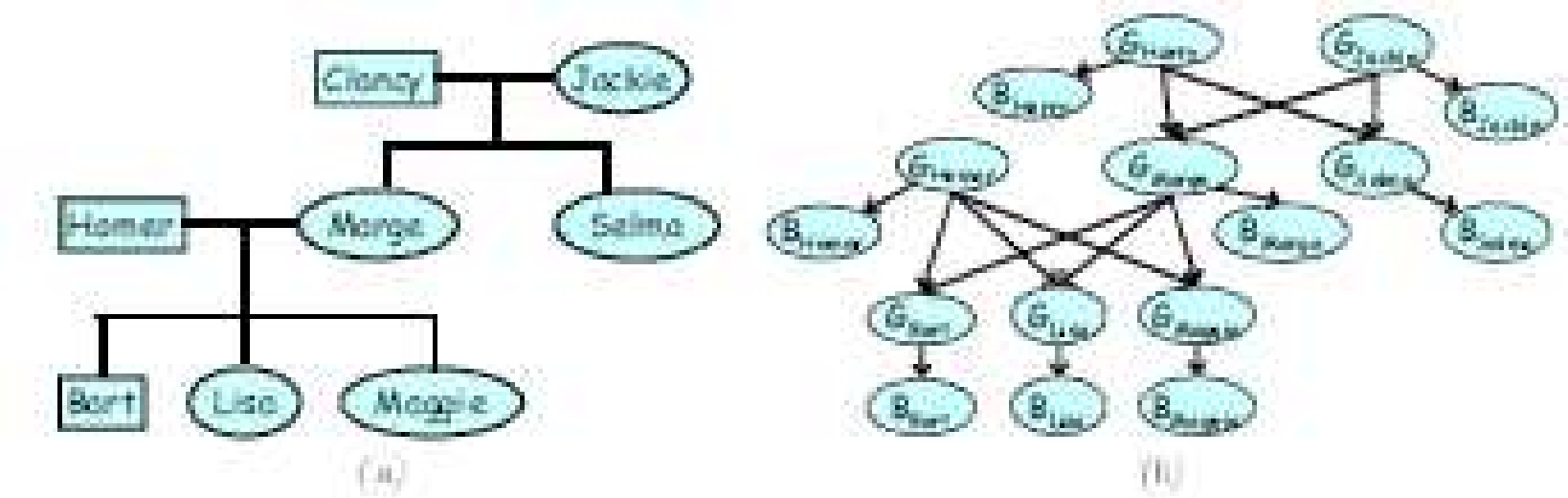
$$p(y, \mathbf{x}, \boldsymbol{\theta}) = p(y|\boldsymbol{\pi})p(\boldsymbol{\pi}) \prod_{j=1}^{d} p(x_j|y, \boldsymbol{\phi}_j)p(\boldsymbol{\phi}_j)$$

We will return to this representation when we discuss parameter estimation

DAGs are widely used for Hierarchical Bayesian models

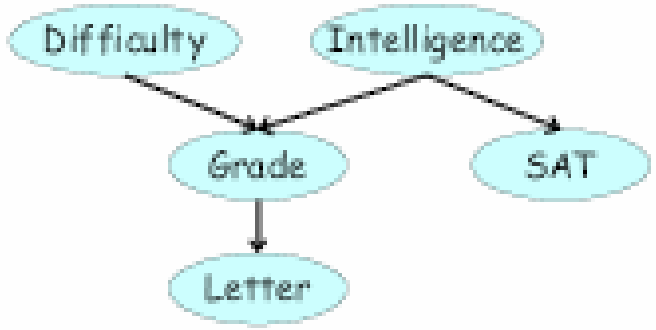
Genetic inheritance

- G(x) = genotype (allele) of person x at given locus, say {A,B,O} x {A,B,O}
- B(x) = phenotype (blood group) in {A,B,O}
- P(B(c)|G(c)) = penetrance model
- P(G(c)|G(p),G(m)) = transmission model
- P(G(c)) = priors for founder nodes



Factorization to I-map

- Thm 3.2.9. If P factorizes over G, then G is an I-map for P.
- Proof (by example)
- We need to show all the local Markov properties hold in P eg. RTP



$$p(S|I,D,G,L) = p(S|I)$$

- By factorization and elementary probability,

$$\begin{aligned} p(S|I,D,G,L) &= \frac{p(S,I,D,G,L)}{p(I,D,G,L)} \\ &= \frac{p(I)p(D)p(G|I,D)p(L|G)p(S|I)}{p(I)p(D)p(G|I,D)p(L|G)} = p(S|I) \end{aligned}$$

Outline

- Administrivia
- Overview
- Local markov property, factorization (3.2)
- Global markov property (3.3)
- Deriving graphs from distributions (3.4)

Global Markov properties

- The DAG defines local markov properties

$$I_\ell(G) = \{X_i \perp Nd(X_i)|Pa(X_i)\}$$

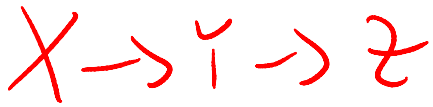
- We would like to be able to determine global markov properties, i.e., statements of the form

$$I(G) = \{X \perp Y|Z : f(X, Y, Z, G)\}$$

  for some function f.
- There are several equivalent ways to define f:
- Bayes ball
- d-separation
- Ancestral separation (ch 4)

Chains

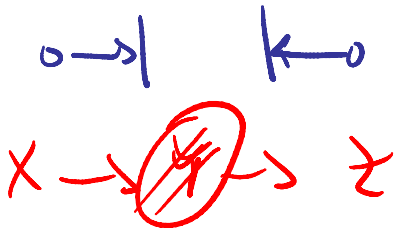
- Consider the chain



$$p(x, y, z) = p(x)p(y|x)p(z|y)$$

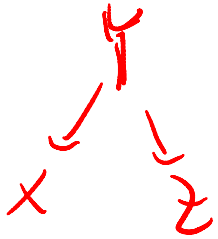
- If we condition and y, x and z are independent

$$\begin{aligned} p(x, z|y) &= \frac{p(x)p(y|x)p(z|y)}{p(y)} \\ &= \frac{p(x, y)p(z|y)}{p(y)} \\ &= p(x|y)p(z|y) \end{aligned}$$



Common cause

- Consider the "tent"

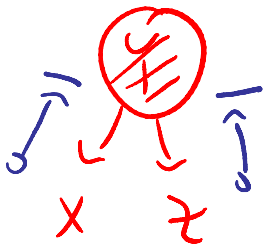


$$p(x, y, z) = p(y)p(x|y)p(z|y)$$

- Conditioning on Y makes X and Z independent

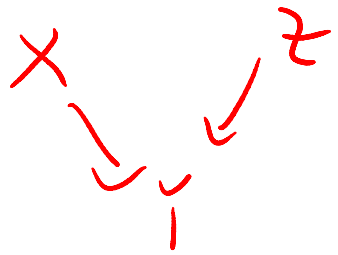
$$p(x, z|y) = \frac{p(x, y, z)}{p(y)}$$

$$= \frac{p(y)p(x|y)p(z|y)}{p(y)} = p(x|y)p(z|y)$$



V-structure (common effect)

- Consider the v-structure



$$p(x, y, z) = p(x)p(z)p(y|x, z)$$

- X and Z are unconditionally independent

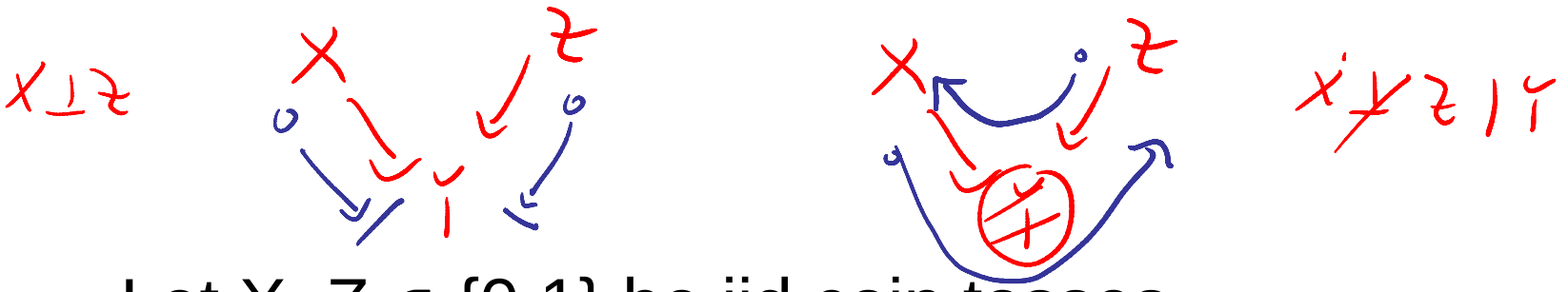
$$p(x, z) = \sum_y p(x, y, z) = \sum_y p(x)p(z)p(y|x, z) = p(x)p(z)$$

but are conditionally dependent

$$p(x, z|y) = \frac{p(x)p(z)p(y|x, z)}{p(y)} \neq f(x)g(z)$$

Explaining away

- Consider the v-structure

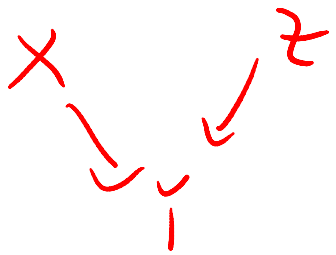


- Let X, Z $\in$ {0,1} be iid coin tosses.
- Let Y = X + Z.
- If we observe Y, X and Z are coupled.

X	Z	Y
0	0	0
0	1	1
1	0	1
1	1	2

Explaining away

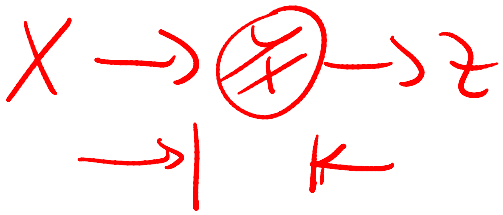
- Let Y = 1 iff burglar alarm goes off,
- X=1 iff burglar breaks in
- Z=1 iff earthquake occurred

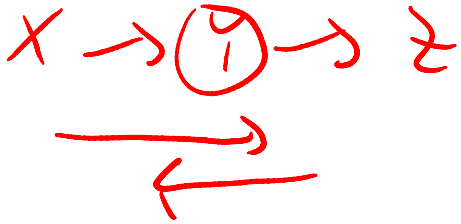


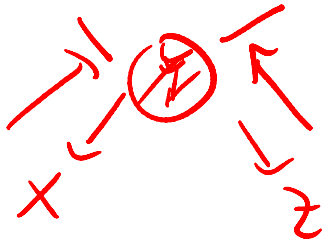
- X and Z compete to explain Y, and hence become dependent
- Intuitively, p(X=1|Y=1) > p(X=1|Y=1,Z=1)

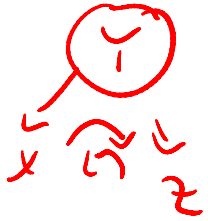
Bayes Ball Algorithm

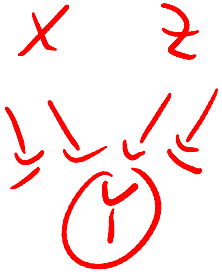
- $X_A \perp X_B \mid X_C$ if we cannot get a ball from any node in A to any node in B when we shade the variables in C. Balls can get blocked as follows.



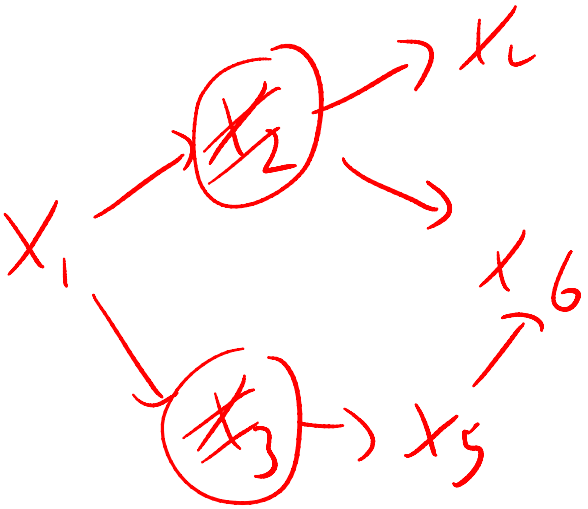






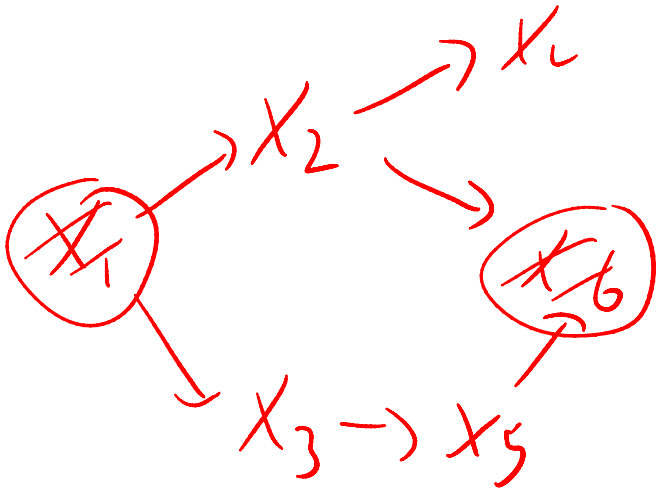


Example



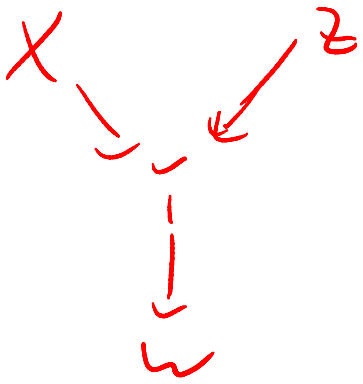
$X_1 \perp X_6 \mid X_2, X_3$?

Example



$X_2 \perp X_3 \mid X_1, X_6$?

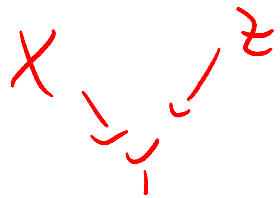
Observing descendant of a v-structure

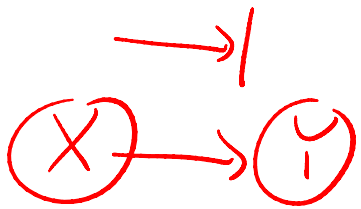


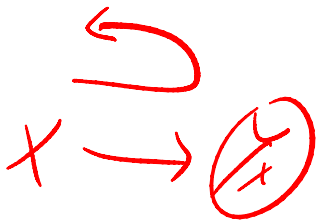
Eg if W=Y (deterministic)

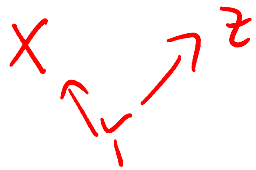
Current rules do not give this. X -> Y -> Z is blocked.
What happens when ball goes X -> Y -> W?
We want the ball to “bounce back” and then go W -> Y -> Z.

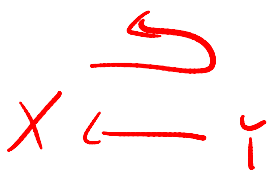
Boundary conditions (source X = destn Z)

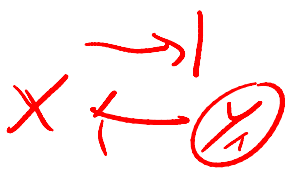




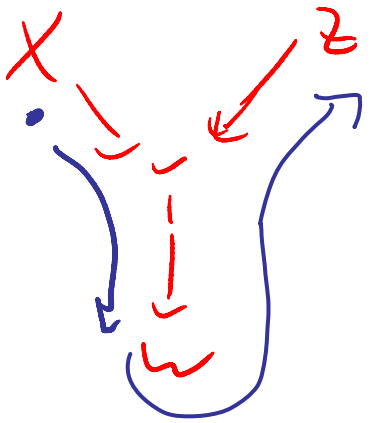








Observing descendant of a v-structure

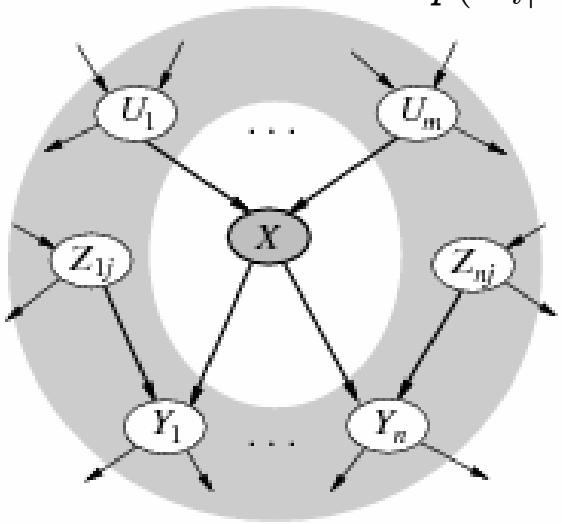


Markov blankets for DAGs

- The Markov blanket of a node is the set that renders it independent of the rest of the graph.

$$MB(X) = \text{minimal set } U s.t. X \perp \mathcal{X} \setminus \{X\} \setminus U | U$$

- This is the parents, children and co-parents.

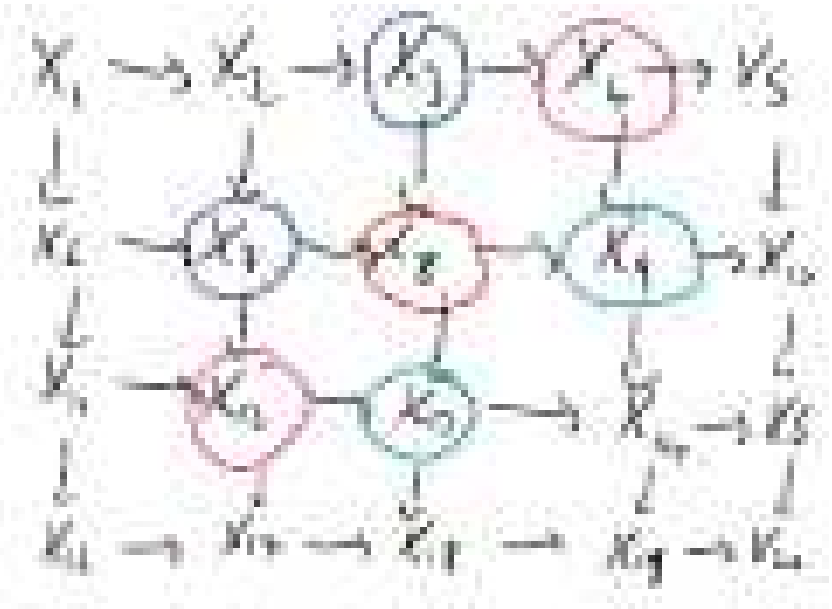


$$\begin{aligned}
p(X_i|X_{-i}) &= \frac{p(X_i, X_{-i})}{\sum_x p(X_i, X_{-i})} \\
&= \frac{p(X_i, U_{1:n}, Y_{1:m}, Z_{1:m}, R)}{\sum_x p(x, U_{1:n}, Y_{1:m}, Z_{1:m}, R)} \\
&= \frac{p(X_i|U_{1:n})[\prod_j p(Y_j|X_i, Z_j)]P(U_{1:n}, Z_{1:m}, R)}{\sum_x p(X_i = x|U_{1:n})[\prod_j p(Y_j|X_i = x, Z_j)]P(U_{1:n}, Z_{1:m}, R)} \\
&= \frac{p(X_i|U_{1:n})[\prod_j p(Y_j|X_i, Z_j)]}{\sum_x p(X_i = x|U_{1:n})[\prod_j p(Y_j|X_i = x, Z_j)]}
\end{aligned}$$

$$p(X_i|X_{-i}) \propto p(X_i|Pa(X_i)) \prod_{Y_j \in ch(X_i)} p(Y_j|Pa(Y_j)$$

Useful for Gibbs sampling

Another example



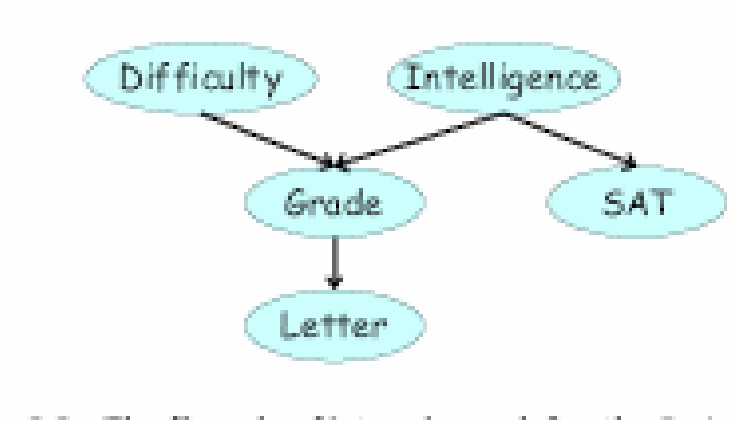
Red node (X8) indep of rest (black) given MB (blue parents, green children, pink co-parents)

Active trails

- Whenever influence can flow from to Y via Z, we say that the trail X <-> Y <-> Z is active.
- Causal trail: X -> Z -> Y. Active iff Z not obs.
- Evidential trail: X <- Z <- Y. Active iff Z not obs
- Common cause: X <- Z -> Y. Active iff Z not obs
- Common effect; X -> Z <- Y. Active iff either Z or one of its descendants is observed.
- Def 3.3.1. Let G be a BN structure, and X1 <-> … <-> Xn be a trail in G. Let E be a subset of nodes. The trail is active given E if
- Whenever we have a v-structure X_{i-1} -> X_i <- X_{i+1}, then X_i or one of its desc is in E
- No other node along the trail is in E

Example

- D-> G <- I ->S not active for E={}
- D-> G <- I ->S is active for E={L}
- D-> G <- I ->S not active for E={L,I}
- Non-monotonic



d-separation

- Def 3.3.2, We say X and Y are d-separated given Z, denoted d-sep_G(X;Y|Z), if there is no active trail between any node in X to any node in Y, given Z. The set of such independencies is denoted

$$I(G) = \{X \perp Y | Z : \text{dsep}_G(X; Y|Z)\}$$

- Thm 3.3.3. (Soundness of dsep). If P factorizes according to G, then I(G) $\subseteq$ I(P).
- False thm (completeness of dsep). For any P that factorizes according to G, if X $\perp$ Y | Z in I(P), then desp$_G$(X;Y|Z) (i.e., P is faithful to G)

Stat 521A
Lecture 2

Outline

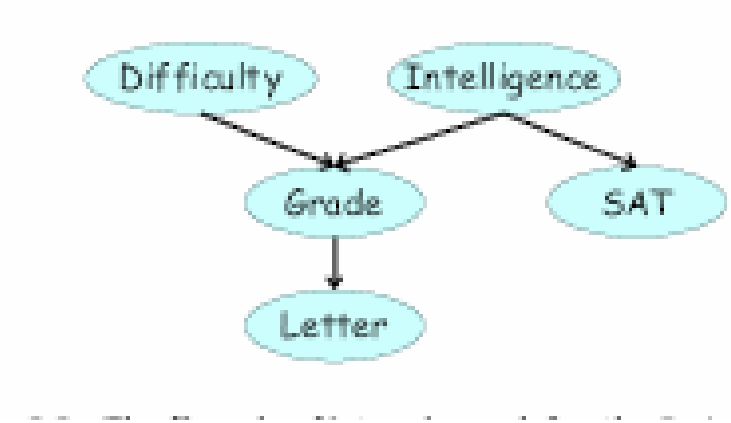
- DAGs
  - global Markov (3.3)
  - deriving graphs from distributions (3.4)
- UGs
  - Global Markov property (4.3.1)
  - Parameterization (4.2)
  - Gibbs distributions, energy based models (4.4.1)
  - Local and pairwise Markov properties (4.3.2)
  - From distributions to graphs (4.3.3)

Active trails

- Whenever influence can flow from to Y via Z, we say that the trail X <-> Y <-> Z is active.
- Causal trail: X -> Z -> Y. Active iff Z not obs.
- Evidential trail: X <- Z <- Y. Active iff Z not obs
- Common cause: X <- Z -> Y. Active iff Z not obs
- Common effect; X -> Z <- Y. Active iff either Z or one of its descendants is observed.
- Def 3.3.1. Let G be a BN structure, and X1 <-> … <-> Xn be a trail in G. Let E be a subset of nodes. The trail is active given E if
- Whenever we have a v-structure X_{i-1} -> X_i <- X_{i+1}, then X_i or one of its desc is in E
- No other node along the trail is in E

Example

- D-> G <- I ->S not active for E={}
- D-> G <- I ->S is active for E={L}
- D-> G <- I ->S not active for E={L,I}
- Non-monotonic



d-separation

- Def 3.3.2, We say X and Y are d-separated given Z, denoted d-sep_G(X;Y|Z), if there is no active trail between any node in X to any node in Y, given Z. The set of such independencies is denoted

$$I(G) = \{X \perp Y | Z : \text{dsep}_G(X; Y|Z)\}$$

- Thm 3.3.3. (Soundness of dsep). If P factorizes according to G, then I(G) $\subseteq$ I(P).
- False thm (completeness of dsep). For any P that factorizes according to G, if X $\perp$ Y | Z in I(P), then desp$_G$(X;Y|Z) (i.e., P is faithful to G)

Faithfulness

- Def 3.3.4. A distribution P is faithful to G if, whenever X ⊥ Y | Z in I(P), we have dsep_G(X;Y|Z) i.e., there are no "non-graphical" independencies buried in the parameters
- A simple unfaithful distribution, with Imap A->B:
-

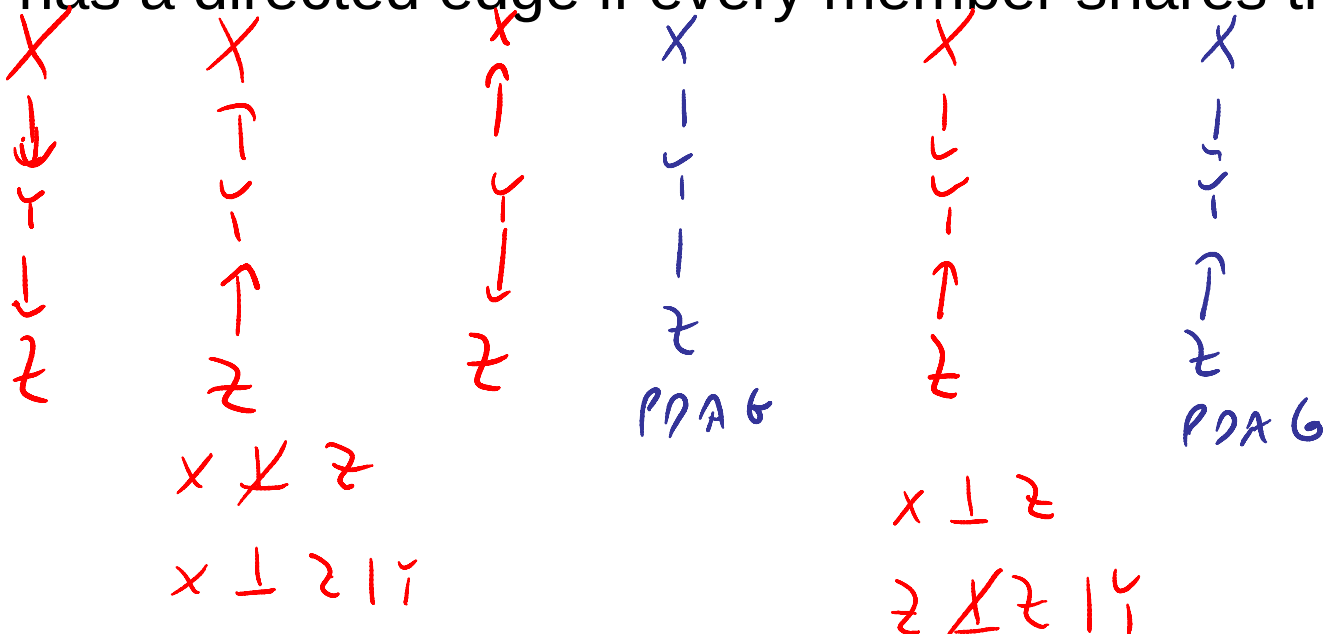
	b^0	b^1
a^0	0.4	0.6
a^1	0.4	0.6

Such distributions are "rare"

- Thm 3.3.7. For almost all distributions P that factorize over G (ie except for a set of measure zero in the space of CPD parameterizations), we have that I(P)=I(G)

Markov equivalence

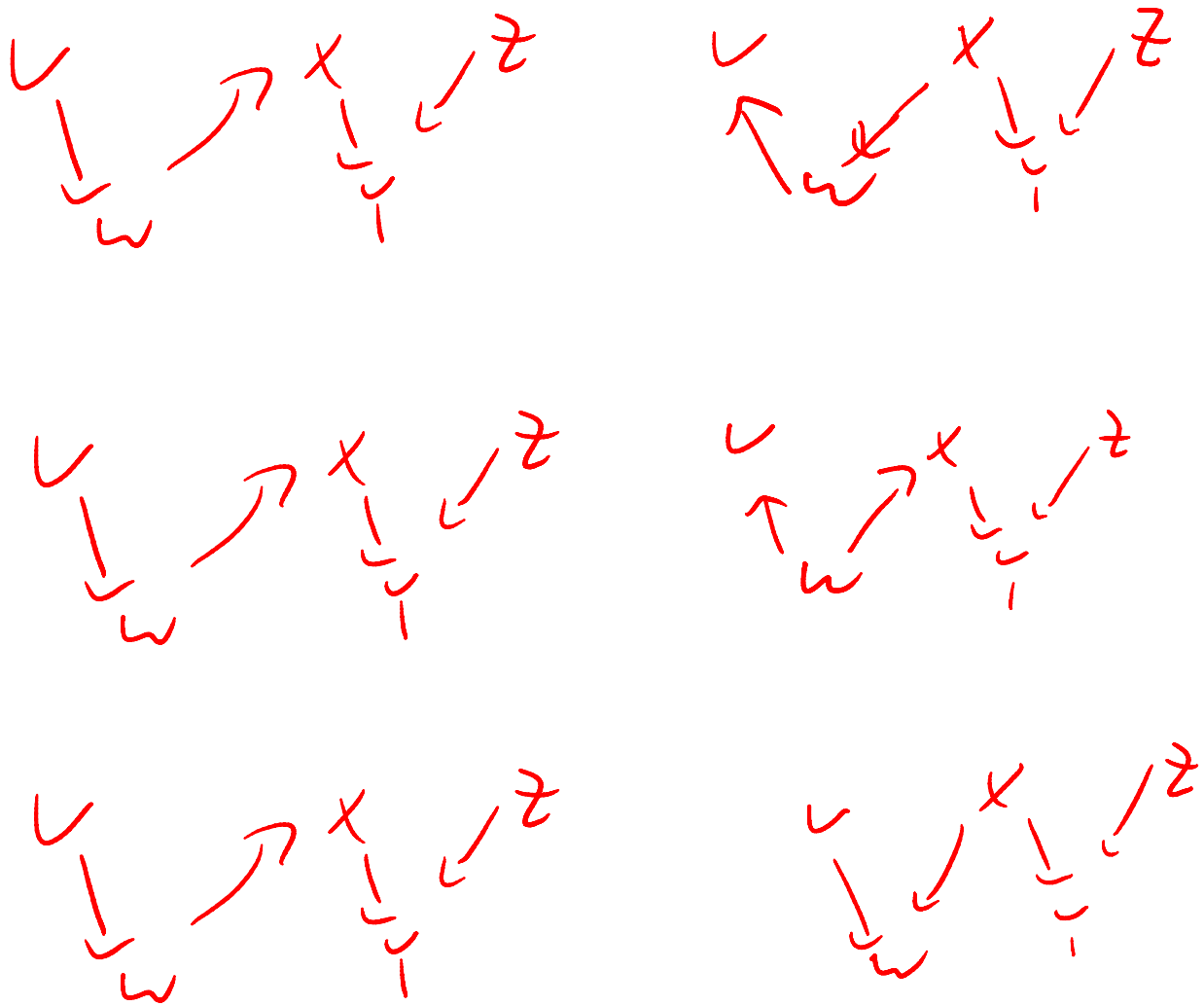
- A DAG defines a set of distributions. Different DAGs may encode the same set and hence are indistinguishable given observational data.
- Def 3.3.10. DAGs G1 and G2 are I-equivalent if I(G1)=I(G2). The set of all DAGs can be partitioned into I-equivalence classes.
- Def 3.4.11. Each can be represented by a class PDAG: only has a directed edge if every member shares that edge.



Identifying I-equivalence

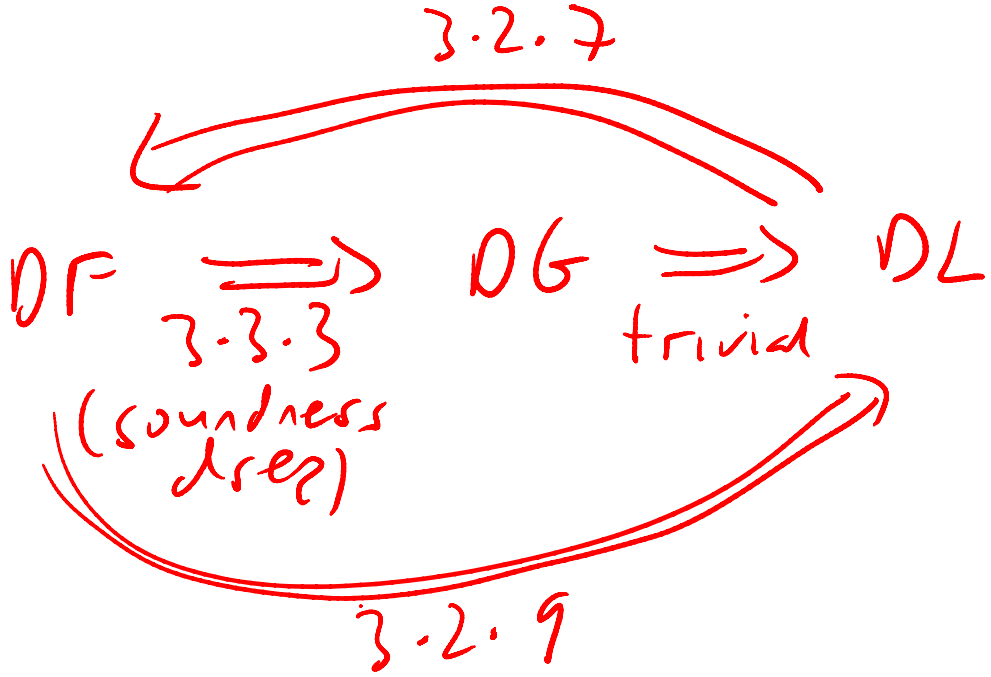
- Def 3.3.11. The skeleton of a DAG is an undirected graph obtained by dropping the arrows.
- Thm 3.3.12. If G1 and G2 have the same skeleton and the same v-structures, they are I-equivalent.
- However, there are structures that are I-equiv but do not have same v-structures (eg fully connected DAG).
- Def 3.3.13. A v-structure X->Z<-Y is an immorality if there is no edge between X and Y (unmarried parents who have a child)
- Thm 3.3.14. G1 and G2 have the same skeleton and set of immoralities iff they are I-equiv.

Examples



Markov properties of DAGs

- DF: F factorizes over G
- DG: $I(G) \subseteq I(P)$
- DL: $I_l(G) \subseteq I(P)$



Based on Jordan ch 4

Deriving graphs from distributions

- So far, we have discussed how to derive distributions from graphs.
- But how do we get the DAG?
- Assume we have access to the true distribution P, and can answer questions of the form

$$P \models X \perp Y | Z$$

- For finite data samples, we can approximate this oracle with a CI test – the frequentist approach to graph structure learning (see ch 18)
- What DAG can be used to represent P?

Minimal I-map

- The complete DAG is an I-map for any distribution (since it encodes no CI relations)
- Def 3.4.1. A graph K is a minimal I-map for a set of independencies I if it is an I-map for I, and if the removal of even a single edge from K renders it not an I-map.
- To derive a minimal I-map, we pick an arbitrary node ordering, and then find some minimal subset U to be X_i's parents, where

$$X_i \perp \{X_1, \ldots, X_{i-1}\} \setminus U | U$$

- (K2 algorithm replace this CI test with a Bayesian scoring metric: sec 18.4.2).

Effect of node ordering

- "Bad" node orderings can result in dense, unintuitive graphs.
- Eg L,S,G,I,D. Add L. Add S: must add L as parent, since $P \not\models L \perp S$. Add G: must add L,S as parents.

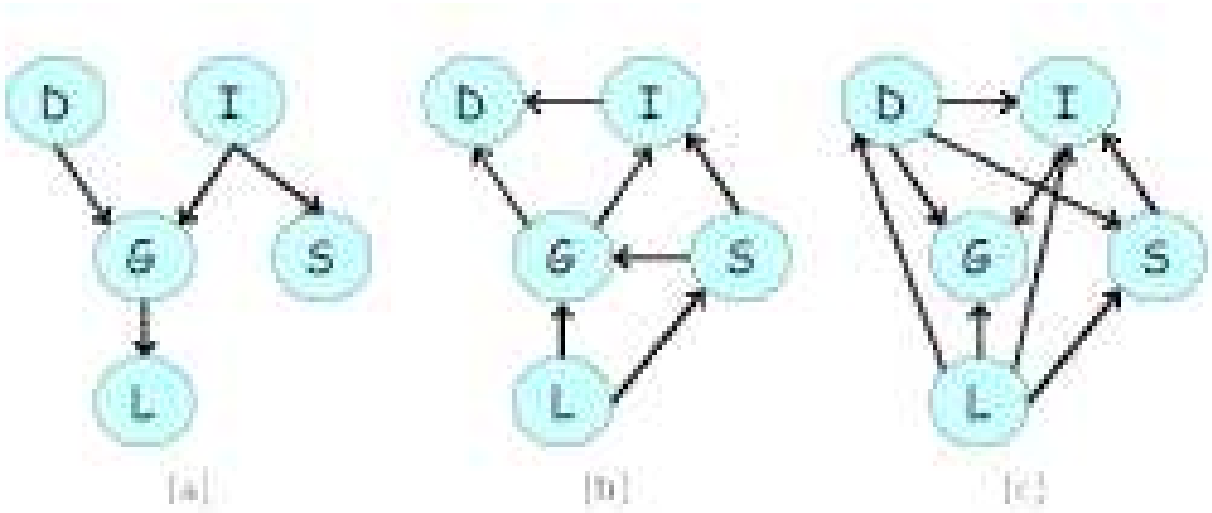
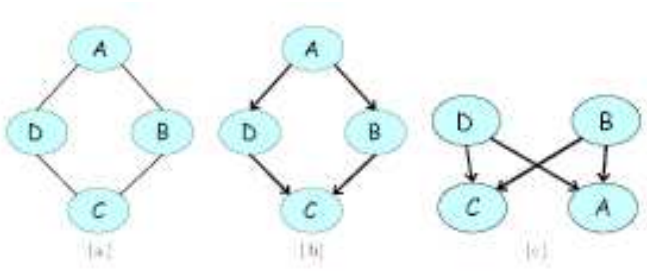


Figure 3.8 Three minimal I-maps for $P_{\{Student\}}$ induced by different orderings: (a) D, I, S, G, L (b) L, S, G, I, D (c) L, D, S, I, G

Perfect maps

- Minimal I-maps can have superfluous edges.
- Def 3.4.2. Graph K is a perfect map for a set of independencies I if I(K)=I. K is a perfect map for P if I(K)=I(P).
- Not all distributions can be perfectly represented by a DAG.
- Eg let Z = xor(X,Y) and use some independent prior on X, Y. Minimal I-map is X -> Z <- Y. However, X $\perp$ Z in I(P), but not in I(G).
- Eg. A $\perp$ C | {B,D} and B $\perp$ D | {A,C}, A dep | B,C, etc



Finding perfect maps

- If P has a perfect map, we can find it in polynomial time, using an oracle for the CI tests.
- We can only identify the graph up to I-equivalence, so we return the PDAG that represents the corresponding equivalence class.
- The method* has 3 steps (see sec 3.4.3)
  - Identify undirected skeleton
  - Identify immoralities
  - Compute eclass (compelled edges)
- This algorithm has been used to claim one can infer causal models from observational data, but this claim is controversial

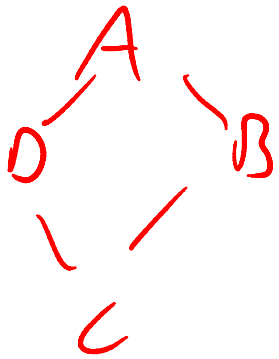
Algorithm due to Verma & Pearl 1991, Spirtes, Glymour, Scheines 1993, Meek 1995

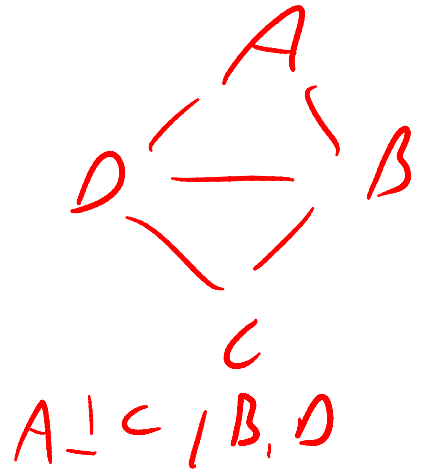
Global Markov property of UGs

- Def 4.3.1. The path X_1 - … - X_k is active given E if none of the nodes on the path are in E.
- Def 4.3.2. The global Markov assumptions associated with a UG H are

$$I(H) = \{X \perp Y | Z : \text{sep}_H(X; Y | Z)\}$$

•eg. A ⊥ C | {B,D} and B ⊥ D | {A,C}





Monotonic, unlike d-separation

$$\text{sep}_H(X; Y | Z) \Rightarrow \text{sep}_H(X; Y | Z') \forall Z \subset Z'$$

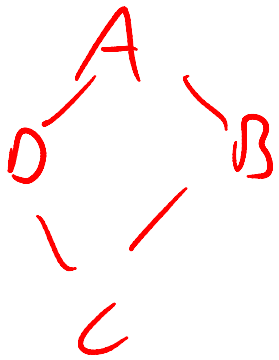
Parameterization

- To specify a specific distribution, we need to associate parameters (local distributions) with the graph.
- CPDs cannot be used because they are not symmetric, and the chain rule need not apply.
- Marginals cannot be used because a product of marginals does not define a consistent joint.
- Instead we multiply a product of **factors (potentials),** one per maximal clique, and then compute a global normalization constant Z (partition function)

$$P(A,B,C,D) = 1/Z\ \phi(A,B,D)\ \phi(B,C,D)$$

$$Z = \sum\_\{A,B,C,D\}\ \phi(A,B,D)\ \phi(B,C,D)$$

Misconception network



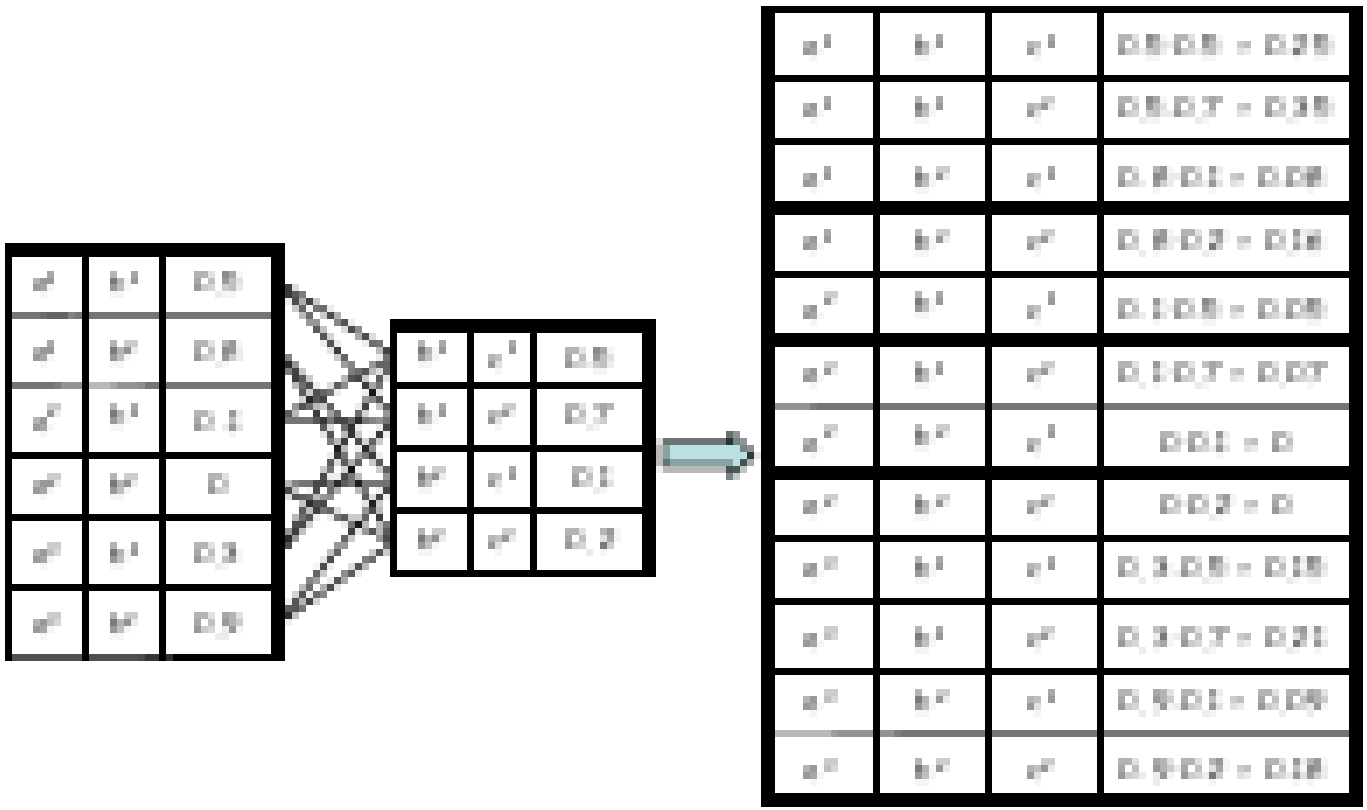
$\phi_1[A,B]$			$\phi_2[B,C]$			$\phi_3[C,D]$			$\phi_4[D,A]$		
a^0	b^0	30	b^0	c^0	100	c^0	d^0	1	d^0	a^0	100
a^0	b^1	5	b^0	c^1	1	c^0	d^1	100	d^0	a^1	1
a^1	b^0	1	b^1	c^0	1	c^1	d^0	100	d^1	a^0	1
a^1	b^1	10	b^1	c^1	100	c^1	d^1	1	d^1	a^1	100

Assignment				Unnormalized	Normalized
a^0	b^0	c^0	d^0	300000	0.04
a^0	b^0	c^0	d^1	300000	0.04
a^0	b^0	c^1	d^0	300000	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	5000000	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	1000000	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	100000	0.014
a^1	b^1	c^1	d^0	100000	0.014
a^1	b^1	c^1	d^1	100000	0.014

P(A,B,C,D) = 1/Z ϕ(A,B) ϕ(A,D) ϕ(C,D) ϕ(C,B)

Multiplying factors

- Def 4.2.2. We multiply factors by matching up corresponding dimensions
  $\Psi(X,Y,Z) = \phi_1(X,Y) \cdot \phi_2(Y,Z)$



Factors are not marginals

- In the misconception network, the marginal on A,B is

a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04

- But the local clique potential is

a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

- Factors are local affinities or preferences, but get combined with other terms in a non-local way

Factorization and I-maps

- Thm 4.3.3. If P factorizes over H, then H is an I-map for P, ie. I(H) $\subseteq$ I(P). (Soundness of separation.)
- Proof. Suppose Z separates X from Y. Then we can partition the factors such that

$$p(\mathbf{x}) = (1/Z) f(X, Z) g(Y, Z)$$

  QED.
- Def 2.1.11. A distribution is positive if P(x)>0 for all x.
- Thm 4.3.4 (Hammersley Clifford). If P is positive, and H is an I-map for P, then P factorizes over H:

$$p(\mathbf{x}) = (1/Z) \prod_c \phi_c(\mathbf{x}_c)$$

Gibbs distributions

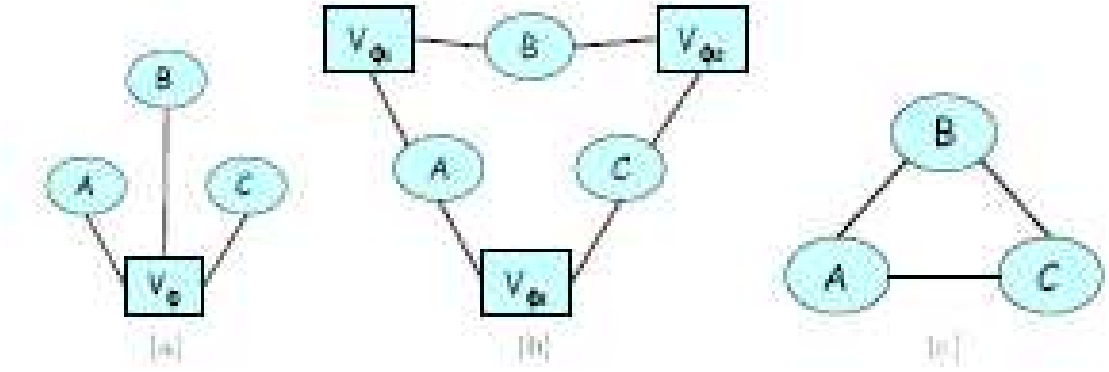
- Def 4.2.3. A Gibbs distribution is defined as

$$p(X_1, \ldots, X_n) = \frac{1}{Z}\phi_1(D_1) \times \cdots \times \phi_m(D_m)$$

- The D_i are the domains or scopes of the factors. We can infer the graph by connecting up all nodes in the same domain. If the D_i are on pairs of nodes (edges), we call it a pairwise Markov random field.
- For a complete graph, we could have one factor per edge or a single clique potential for the whole graph.

Factor graphs

- For a complete graph, we could have one factor per edge or a single clique potential for the whole graph.
- Factor graphs can distinguish these cases.
- Def 4.4.1. Square nodes = factors, ovals = rv's.



Energy based models

- It is common to work with energies = negative log factors/ potentials (low energy = more probable)

$$\phi(D) = \exp(-\epsilon(D)) \qquad p(x_1, \ldots, x_n) = 1/Z \exp[-\sum_{i=1}^{m} \epsilon_i(D_i)]$$

$\epsilon_1[A, B]$

a^0	b^0	-3.4
a^0	b^1	-1.61
a^1	b^0	0
a^1	b^1	-2.3

$\epsilon_2[B, C]$

b^0	c^0	-4.61
b^0	c^1	0
b^1	c^0	0
b^1	c^1	-4.61

$\epsilon_3[C, D]$

c^0	d^0	0
c^0	d^1	-4.61
c^1	d^0	-4.61
c^1	d^1	0

$\epsilon_4[D, A]$

d^0	a^0	-4.61
d^0	a^1	0
d^1	a^0	0
d^1	a^1	-4.61

$\phi_1[A, B]$

a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

$\phi_2[B, C]$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

$\phi_3[C, D]$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_4[D, A]$

d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

Ising model

- X_i = +1 if atom is spin up, X_i = -1 if spin down
- Define edge energy as

$$\epsilon_{i,j}(x_i, x_j) = -w_{i,j} x_i x_j$$

- If spins equal (aligned), product is +1, else -1.
- w_{i,j} = 0.5 (E(anti-aligned)-E(aligned)). If +ve, model aligns atoms (ferromagnetic). If –ve, spins should be different (anti-ferromagnetic).
- Define local node energy (external field) as

$$\epsilon_i(x_i) = -u_i x_i$$

- Overall distribution

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \exp\left(\sum_{i<j} w_{i,j} x_i x_j + \sum_i u_i x_i \right)$$

Ising models capture pairwise correlation

- Energy can be written as

$$
\begin{aligned}
\epsilon(\mathbf{x}) &= -\sum_{i<j} w_{i,j} x_i x_j - \sum_i u_i x_i \\
&= -\tfrac{1}{2}\mathbf{x}^T \mathbf{W} \mathbf{x} - \mathbf{u}^T \mathbf{x} \\
&= -\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{W} (\mathbf{x} - \boldsymbol{\mu}) + c \\
\boldsymbol{\mu} &= -\mathbf{W}^{-1}\mathbf{u} \\
c &= \tfrac{1}{2}\boldsymbol{\mu}^T \mathbf{W} \mu
\end{aligned}
$$

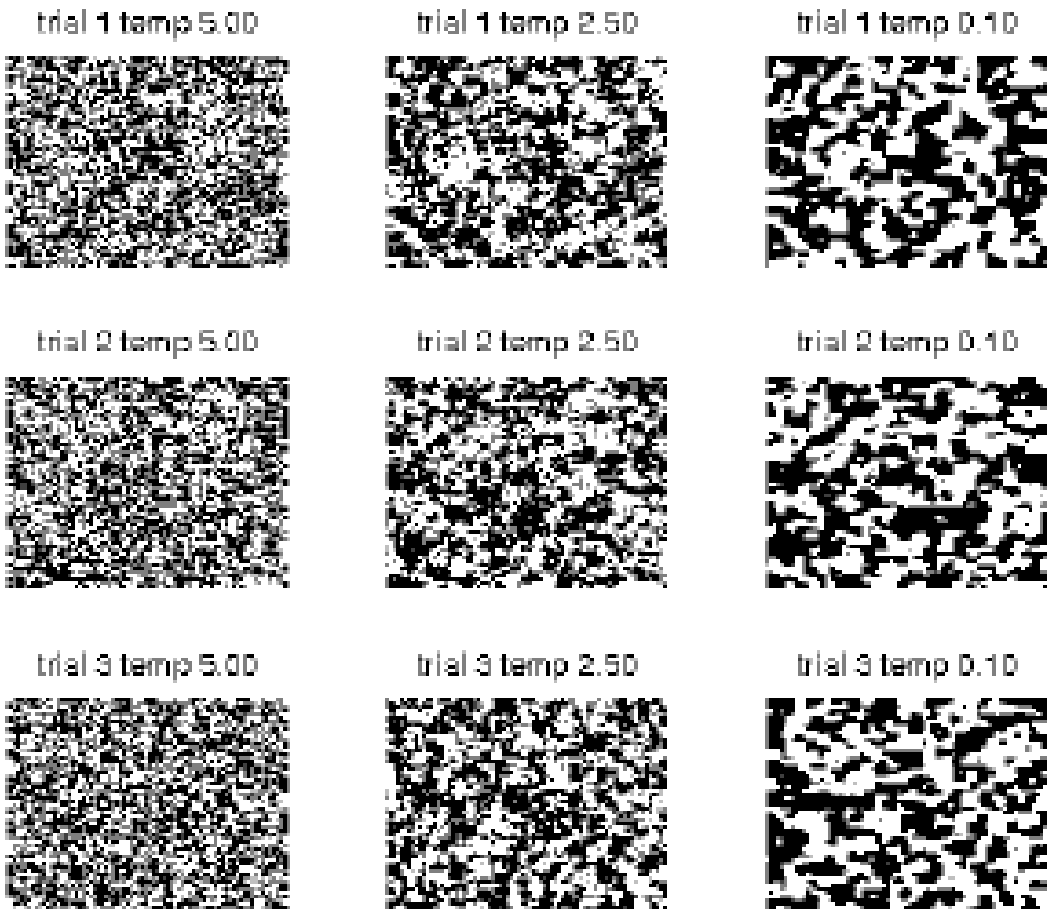
Phase transition

- The strength of the interactions is modulated by a global temperature parameter T

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(-\epsilon(\mathbf{x})/T\right)$$

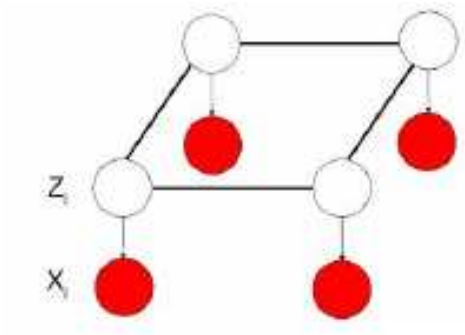
- Large temperature “flattens” the energy landscape and makes the uniform distribution most probable
- Small temperature makes the distribution “peaky”
- One can compute the density of pure vs mixed state configurations as a function of T (as the number of atoms -> ∞). There is often a phase transition: as T exceeds a critical temperature, there is a sudden regime change.
- This has computational analogs in the mixing time of Markov chains.

Samples from an Ising model



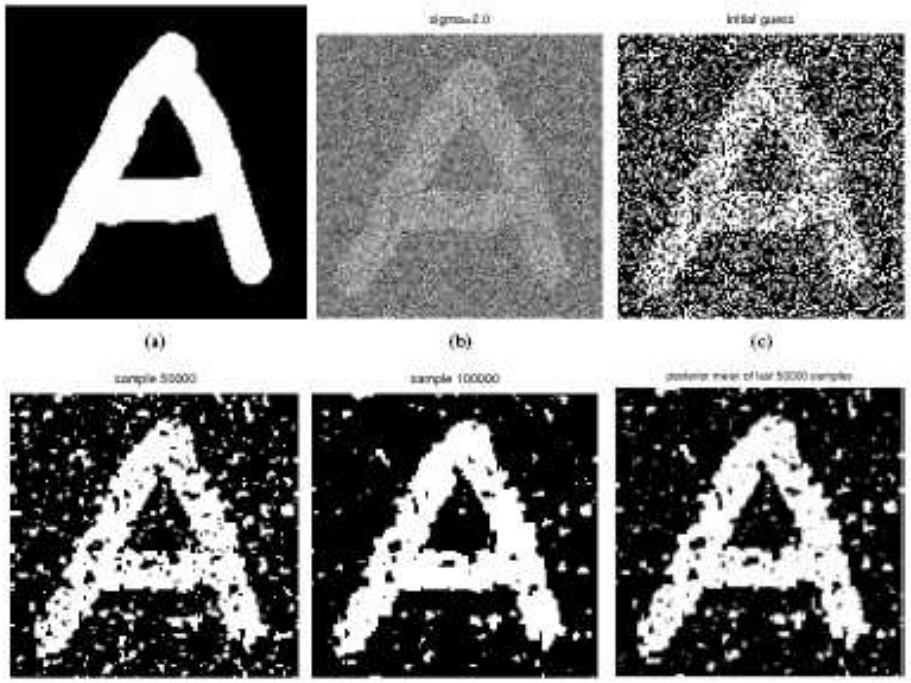
See GibbsDemoIsing in PMTK/bookCode

Image denoising



$$p(\mathbf{x},\mathbf{y}) \quad = \quad p(\mathbf{x})p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{<ij>}\phi_{ij}(x_i,x_j)\prod_i p(y_i|x_i)$$

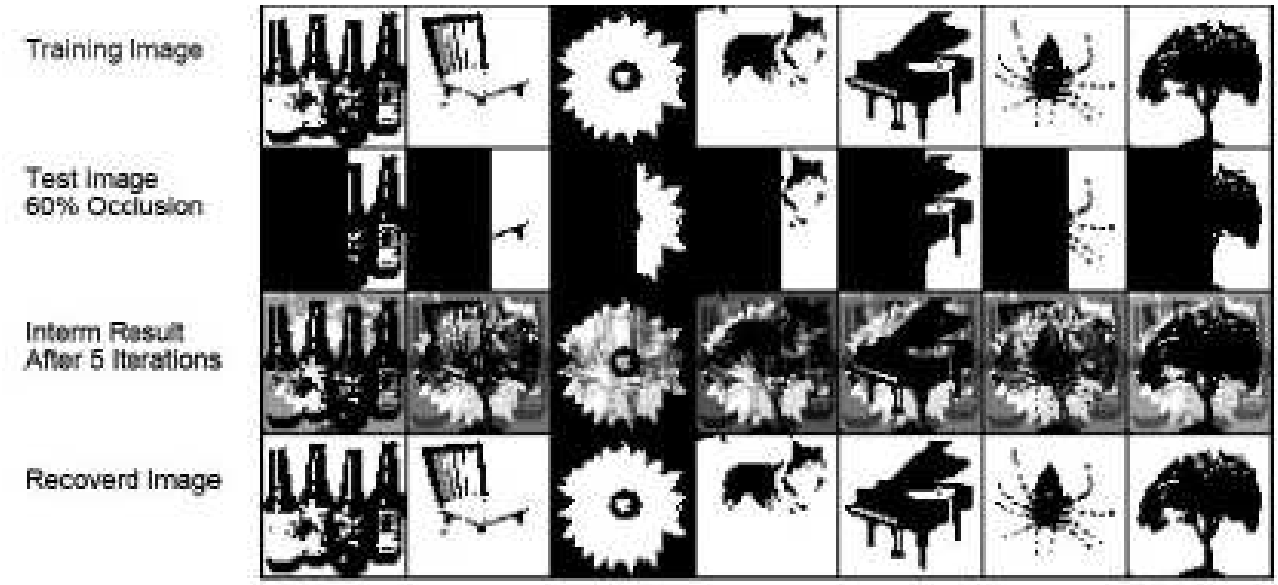
argmax_x P(x|y) is best guess of denoised image



See GibbsDemoIsing in PMTK/bookCode

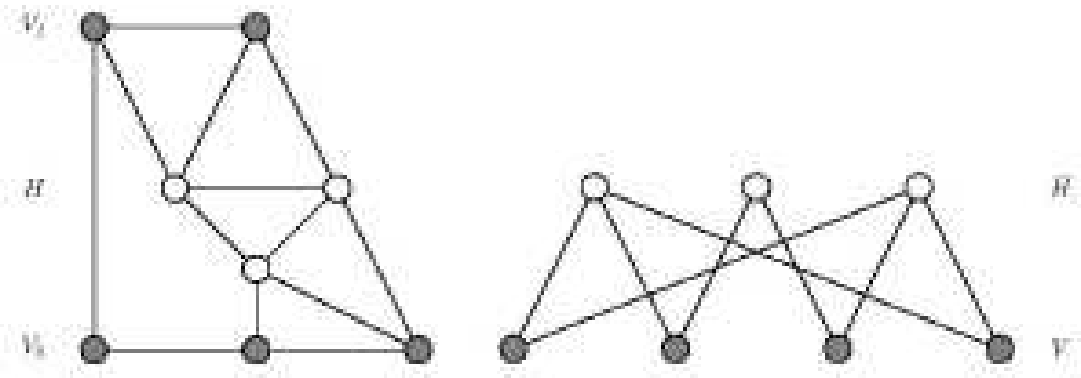
Hopfield network

- A Hopfield network is a stochastic, recurrent neural network.
- It is equivalent to a fully connected Ising model.
- Weights are learned.
- Often used for associative memory/ pattern completion.



Boltzmann machine

- A Boltzmann machine is a Hopfield network (Ising model) with hidden nodes.
- A restricted Boltzmann machine (RBM) is a bipartite BM. This supports efficient block Gibbs sampling (see ch 12).

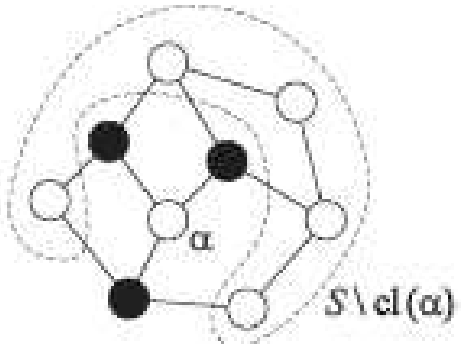


Local Markov assumption

- So far, we have defined the global Markov assumptions using simple graph separation.
- We now consider some variants.
- The boundary of a node α, bd(α), is all nodes which are directly connected to it.
- The closure is cl(α) = bd(α)U α.
- Def 4.3.9. The local Markov properties of H are

$$I_l(H) = \{\alpha \perp S \setminus cl(\alpha) | bd(\alpha)\}$$

- i.e. a is indep of rest given its Markov blanket bd(a).



Pairwise Markov assumption

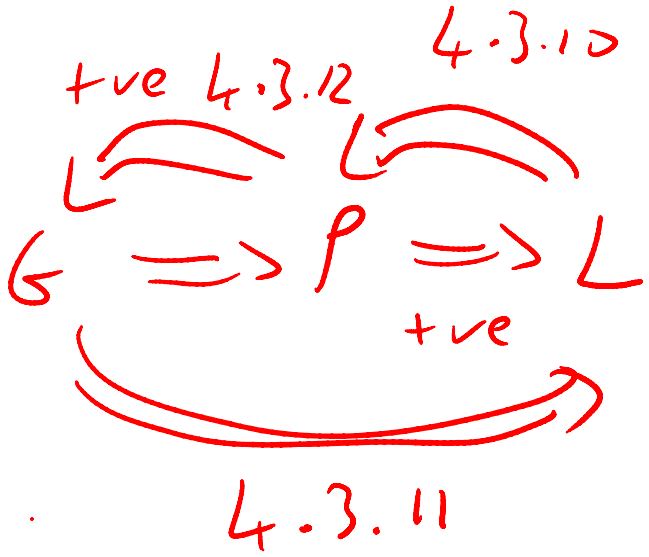
- Def 4.3.7. The pairwise Markov independencies associated with H are

$$I_p(H) = \{\alpha \perp \beta | S \setminus \{\alpha, \beta\} : \alpha - \beta \notin H\}$$

- i.e., a is independent of b given rest if not directly connected.

Markov properties

- G: $I(G) \subseteq I(P)$
- L: $I_l(G) \subseteq I(P)$
- P: $I_p(G) \subseteq I(P)$
- If P is positive, all are equivalent.



Based on Jordan ch 4, thm numbers refer to Koller&Friedman

Problems caused by determinism

- If the distribution is not positive, pairwise indep does not imply local or global indep.
- Ex 4.3.15. Let P be any distribution over $(X_1,\ldots,X_n)$. Make 3 identical copies of each variable, X_i, X_i', X_i''. Let H be the empty MRF on this expanded state space. This satisfies the pairwise Markov properties eg X_i and X_i' are independent, because the remaining nodes contain X_i''. Also, X_i and X_j are independent, because the remaining nodes contain X_i'. However, H does not satisfy local or global indep.

From distributions to graphs

- How do we derive a graph from a distribution?
- For positive distributions, there are two approaches, based on pairwise and local prop.
- Thm 4.3.17. Let P be a +ve dist. Let H be an MRF in which we add an edge X-Y for all X,Y which cannot be made independent when conditioned on any other set:

$$P \not\models (X \perp Y | \mathcal{X} \setminus \{X, Y\})$$

Then H is the unique minimal I-map for P.

From distributions to graphs

- Thm 4.3.18. Let P be a +ve dist. For each node X, let MB_P(X) be a minimal set of nodes U rendering X indep of the rest:

$$X \perp \mathcal{X} \setminus \{X\} \setminus U | U \in I(P)$$

Add an edge X-Y for all Y in MB_P(X). Then H is a unique minimal I-map for P.

Stat 521A
Lecture 3

Announcements

- Hw1 out today, due back Thur 22nd (in class)
- How many people will do hw?
- Matlab access?
- Join groups.google.com/group/stat521a-spring09 email list
- How many people have bought/ will buy book?
- Auditors: please fill out form

Outline

- Hammersley Clifford theorem (4.4.2)
- Log-linear models (Wasserman ch 19)
- Directed vs undirected graphs (4.5)
- Conditional random fields, chain graphs (4.6)

Hammersley Clifford Thm

- Thm 4.4.9. If $I_P(H) \subseteq I(P)$, and P is strictly positive, then P factorizes over the max cliques of H, i.e.

$$P(\mathbf{x}) = \frac{1}{Z} \prod_c \psi_c(\mathbf{x}_c)$$

Mobius inversion lemma

- Let V be a finite set with elements. Let Ψ and Φ be functions defined over all possible subsets. Then

$$\Psi(a) = \sum_{b:b\subseteq a} \Phi(b)$$

is equivalent to the statement

$$\Phi(a) = \sum_{b:b\subseteq a} (-1)^{|a\setminus b|}\Psi(b)$$

Mobius inversion lemma

- Proof
- RTP $\Phi(b) = \sum_{c:c\subseteq b} (-1)^{|b\setminus c|}\Psi(c) \Rightarrow \Psi(a) = \sum_{b:b\subseteq a} \Phi(b)$ and vice versa.

$$
\begin{aligned}
\sum_{b:b\subseteq a} \Phi(b) &= \sum_{b:b\subseteq a} \sum_{c:c\subseteq b} (-1)^{|b\setminus c|}\Psi(c) \\
&= \sum_{c:c\subseteq a} \left\{ \sum_{b:c\subseteq b\subseteq a} (-1)^{|b\setminus c|} \right\} \Psi(c) \\
&= \sum_{c:c\subseteq a} \left\{ \sum_{h:h\subseteq a\setminus c} (-1)^{|h|} \right\} \Psi(c) \\
&= \Psi(a)
\end{aligned}
$$

Since the inner sum is zero unless a\c={} (ie c=a),
since number of subsets of even cardinality is equal to the number of subsets of odd cardinality.

Over parameterization

- A standard parameterization of an MRF is over parameterized eg the information about B is stored in both cliques {A,B} and {B,C}.
- We can shift probability mass from one factor to another without affecting the overall distribution.

$\phi_1[A,B]$			$\phi_2[B,C]$		
a^0	b^0	−3.4	b^0	c^0	−4.61
a^0	b^1	−1.61	b^0	c^1	0
a^1	b^0	0	b^1	c^0	0
a^1	b^1	−2.3	b^1	c^1	−4.61

$\phi'_1[A,B]$			$\phi'_2[B,C]$		
a^0	b^0	−4.4	b^0	c^0	−3.61
a^0	b^1	1.61	b^0	c^1	+1
a^1	b^0	−1	b^1	c^0	0
a^1	b^1	2.3	b^1	c^1	4.61

Canonical parameterization

- Choose a distinguished setting of the variables, $S^* = (s_1^*, \ldots s_n^*)$. Augment any partial setting by filling in the rest with these default values.

$$\hat{S}_a = (S_a, S^*_{-a})$$

- Define the log probability of a partial assignment

$$H_a(S) = \log P(\hat{S}_a)$$

- Define the canonical energy function as

$$\phi_a(S) = \sum_{b:b\subseteq a} (-1)^{|a\setminus b|} H_b(S)$$

- This defines energy for {A,B,C}, subtracts off influence of {A,B}, {B,C}, {C,A}, adds back influence of singletons, subtracts off baseline

Misconception : canonical params

$\phi_1^*[A,B]$

a^0	b^0	0
a^0	b^1	0
a^1	b^0	0
a^1	b^1	4.09

$\phi_2^*[B,C]$

b^0	c^0	0
b^0	c^1	0
b^1	c^0	0
b^1	c^1	9.21

$\phi_3^*[C,D]$

c^0	d^0	0
c^0	d^1	9.21
c^1	d^0	0
c^1	d^1	0

$\phi_4^*[D,A]$

d^0	a^0	0
d^0	a^1	9.21
d^1	a^0	0
d^1	a^1	18.4

$\phi_5^*[A]$

a^0	0
a^1	-8.01

$\phi_6^*[B]$

b^0	0
b^1	-6.4

$\phi_7^*[C]$

c^0	0
c^1	0

$\phi_8^*[D]$

d^0	0
d^1	-9.21

$\phi_9^*[\emptyset]$

-3.18

$$
\begin{aligned}
\phi_{A,B}(a^1,b^1,c,d) &= \sum_{z\in\{A,B\},\{A\},\{B\},\{\}} (-1)^{|\{A,B\}\setminus z|} H_z(a^1,b^1,c,d) \\
&= (-1)^0 H(a^1,b^1,c^*,d^*) + (-1)^1 H(a^1,b^*,c^*,d^*) \\
&\quad +(-1)^1 H(a^*,b^1,c^*,d^*) + (-1)^2 H(a^*,b^*,c^*,d^*) \\
&= (-13.49) - (-11.18) - (-9.58) + (-3.18) = 4.09
\end{aligned}
$$

Hammersley Clifford Thm

- Thm 4.4.9. If $I_P(H) \subseteq I(P)$, and P is positive, then P factorizes over the max cliques of H.
- Proof.
- Define

$$H_a(S) = \log P(\hat{S}_a) \qquad \phi_a(S) = \sum_{b:b\subseteq a} (-1)^{|a\setminus b|} H_b(S)$$

- By Mobius inversion lemma

$$H_S(S) = \sum_{a:a\subseteq S} \phi_a(S)$$

- Define $\psi_a(S_a) = \exp \phi_a(S_a)$. Then

$$P(S) = \exp H_S(S) = \exp \sum_{a:a\subseteq} \phi_a(S) = \prod_{a:a\subseteq S} \psi_a(S)$$

Hammersley Clifford proof II

- We have shown the distribution can be written as a product of potential functions

$$P(S) = \prod_{a:a\subseteq S} \psi_a(S)$$

- The final step is to show that $\phi_a(S)$ is zero unless a is a maximal clique.
- Let $\alpha,\beta \in$ a be 2 nodes without an edge, and let c = a \ {α,β}. Let H_a = H_a(S). Then

$$\phi_a(S) = \sum_{b:b\subseteq c} (-1)^{|c\setminus b|} \left(H_b - H_{b\cup\alpha} - H_{b\cup\beta} + H_{b\cup\{\alpha,\beta\}}\right)$$

Hammersley Clifford proof III

- Define d=S \ {α,β}. By the pairwise Markov property, $\alpha \perp \beta \mid d$ and hence

$$
\begin{aligned}
H_{b\cup\{\alpha,\beta\}} - H_{b\cup\alpha} &= \log \frac{p(S_b, S_\alpha, S_\beta, S^*_{d\setminus b})}{p(S_b, S_\alpha, S^*_\beta, S^*_{d\setminus b})} \\
&= \log \frac{p(S_\alpha|S_b, S^*_{d\setminus b})p(S_\beta, S_b, S^*_{b\setminus d})}{p(S_\alpha|S_b, S^*_{d\setminus b})p(S^*_\beta, S_b, S^*_{b\setminus d})} \\
&= \log \frac{p(S^*_\alpha|S_b, S^*_{d\setminus b})p(S_\beta, S_b, S^*_{b\setminus d})}{p(S^*_\alpha|S_b, S^*_{d\setminus b})p(S^*_\beta, S_b, S^*_{b\setminus d})} \\
&= \log \frac{p(S_b, S^*_\alpha, S_\beta, S^*_{d\setminus b})}{p(S_b, S^*_\alpha, S^*_\beta, S^*_{d\setminus b})} \\
&= H_{b\cup\beta} - H_b
\end{aligned}
$$

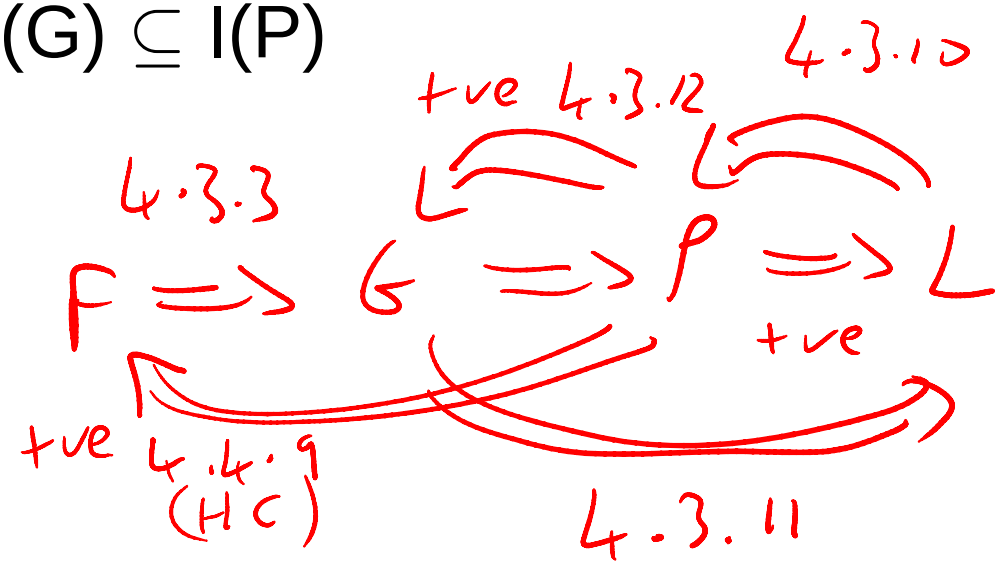
Hence all the terms on the RHS below vanish whenever we can find an α and β that are not connected within a.

$$\phi_a(S) = \sum_{b:b\subseteq c} (-1)^{|c\setminus b|} \left(H_b - H_{b\cup\alpha} - H_{b\cup\beta} + H_{b\cup\{\alpha,\beta\}}\right)$$

QED

Summary

- F: P factorizes over G
- G: $I(G) \subseteq I(P)$
- L: $I_l(G) \subseteq I(P)$
- P: $I_p(G) \subseteq I(P)$



Based on Jordan ch 4, thm numbers refer to Koller&Friedman

Log-linear models

- Let X=(X1,…,Xm), Xj in {1,…,rj}. The joint density is a multinomial with N=r_1 x … r_m states. (Contigency table.) Let S={1,..,m}.
- Thm (Wasserman p292), The pmf can be written as

$$\log p(\mathbf{x}) = \sum_{A \subset S} \psi_A(\mathbf{x})$$

where the ψ satisfy:

$\psi_0(x)$ is a constant

$\psi_A(x)$ only depends on x_A, not other bits

- If i in A, and x_i=0, then $\psi_A(x)$=0

Based on Wasserman's "All of statistics", ch 19

Example: Bernoulli

- Let X ~ Ber(θ), 0 < θ < 1. Then

$$p(x) = \theta^x (1-\theta)^{1-x} = p_1^x p_2^{1-x}$$

$$\begin{aligned} \log p(x) &= \psi_\emptyset(x) + \psi_1(x) \\ \psi_\emptyset(x) &= \log(p_2) \\ \psi_1(x) &= x \log\left(\frac{p_1}{p_2}\right) \end{aligned}$$

- Multinom. Param space

$$\{(p_1, p_2) : p_j \geq 0, p_1 + p_2 = 1\}$$

- Log-linear param space

$$\{(\beta_0 = \log(p_2), \beta_1 = \log(p_1/p_2)) : e^{\beta_0+\beta_1} + e^{\beta_0} = 1\}$$

Example: 2 way contingency

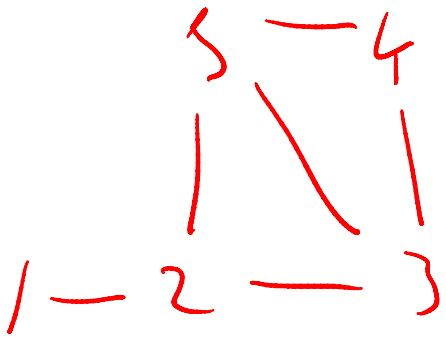
- X_1 in {0,1}, X_2 in {0,1,2}.

$$
\begin{aligned}
\log p(x) &= \psi_\emptyset(x) + \psi_1(x) + \psi_2(x) + \psi_{12}(x) \\
\psi_\emptyset(x) &= \log p_{00} \\
\psi_1(x) &= x_1 \log\left(\frac{p_{10}}{p_{00}}\right) \\
\psi_2(x) &= I(x_2 = 1)\log\left(\frac{p_{01}}{p_{00}}\right) + I(x_2 = 2)\log\left(\frac{p_{02}}{p_{01}}\right) \\
\psi_{12}(x) &= I(x_1 = 1, x_2 = 1)\log\left(\frac{p_{11}p_{00}}{p_{01}p_{10}}\right) + I(x_1 = 1, x_2 = 2)\log\left(\frac{p_{12}p_{00}}{p_{02}p_{10}}\right)
\end{aligned}
$$

Graphical log-linear models

- Def. A log-linear model is graphical if $\psi_A(x)=0$ iff $\{i,j\} \subset A$ and (i,j) is not an edge, i.e., if you can add a term to the model and the graph does not change, the model is not graphical.
- Eg this is graphical, since eg edge 1-5 is missing, and all terms containing (1,5) are 0

$$\begin{aligned}\log p(x) = \; & \psi_0 + \psi_1(x) + \psi_2(x) + \psi_3(x) + \psi_4(x) + \psi_5(x) \\ & + \psi_{12}(x) + \psi_{23}(x) + \psi_{25}(x) + \psi_{34}(x) + \psi_{35}(x) + \psi_{45}(x) \\ & + \psi_{235}(x) + \psi_{345}(x)\end{aligned}$$



If we remove the 3-way terms, the graph is the same. This would not be graphical.

Hierarchical log-linear models

- Def. A log-linear model is hierarchical if $\psi(A)=0$ and $A \subset B$ implies $\psi(B)=0$
- Thm: graphical implies hierarchical but not necessarily the reverse.
- Eg. Hierarchical but not graphical

$$\log p(x) = \psi_0(x) + \psi_1(x) + \psi_2(x) + \psi_3(x) + \psi_{12}(x) + \psi_{13}(x) + \psi_{23}(x)$$

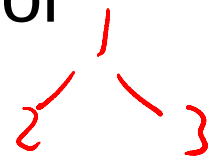
- Eg. Not hierarchical.

$$\log p(x) = \psi_0(x) + \psi_3(x) + \psi_{12}(x)$$

Model generators

- Hierarchical models can be written succinctly using generators.
- Eg. X=(X_1,X_2,X_3). M = 1.2 + 1.3 stands for

$$\log p = \psi_0 + \psi_1 + \psi_2 + \psi_3 + \psi_{12} + \psi_{13}$$



- M = 1.2.3 is the saturated model (complete graph)

$$\log p = \psi_0 + \psi_1 + \psi_2 + \psi_3 + \psi_{12} + \psi_{13} + \psi_{23} + \psi_{123}$$

- M = 1 + 2 + 3 is the empty graph

$$\log p = \psi_0 + \psi_1 + \psi_2 + \psi_3$$

1 2 3

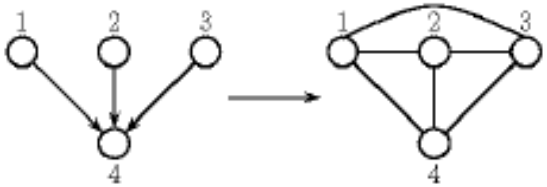
- M = 1.2 represents

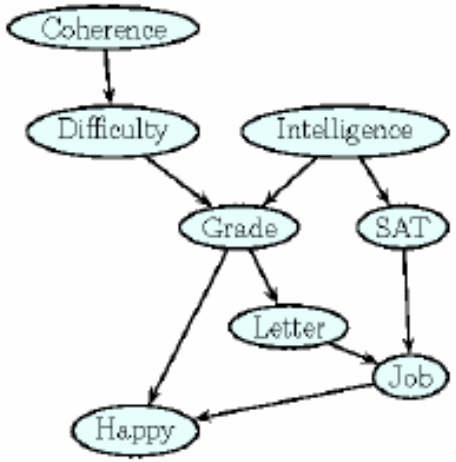
$$\log p = \psi_0 + \psi_1 + \psi_2 + \psi_{12}$$

1–2

Moralization

- Def 4.5.2. The moral graph M(G) of a DAG is obtained by adding an undirected edge between X and Y if both are unmarried parents of a common child, and then dropping all the arrows.





DAG to UG

- Intuitively, connecting the parents prevents any unwanted independence assumptions: given Z, X and Y are dependent (explaining away)

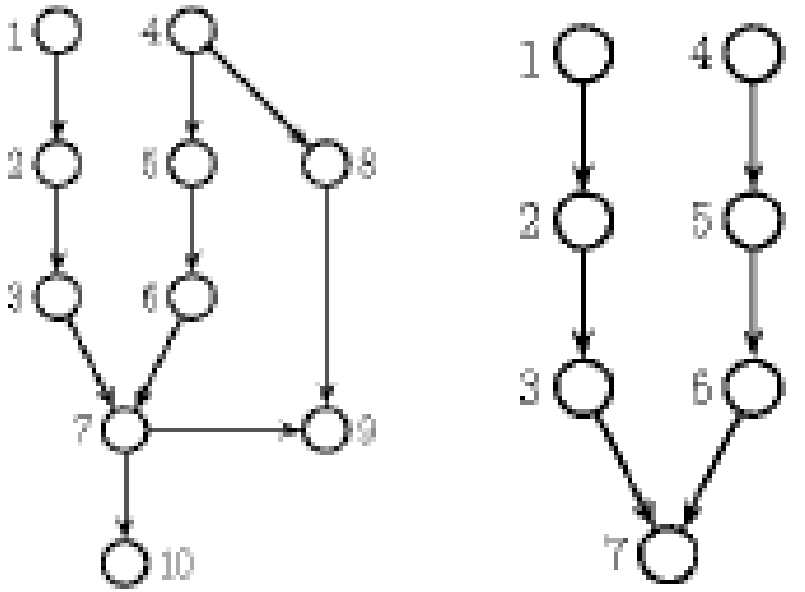
- Thm 4.5.4. The moralization M(G) is a minimal I-map for DAG G.

An alternative to d-separation

- Suppose we want to determine if G := X ⊥ Y | Z
- It is tempting to think that simple separation in the moral graph will yield d-separation. However, we should not connect unmarried parents unless their children or descendants are observed.
- Therefore we remove all nodes except for U=X,Y,Z and their ancestors (ancestral graph) - (Any loops back to U via descendants must be via v-structures, and are therefore blocked.)
- We then moralize the resulting graph and use simple separation.

Ancestral graph example

U={1,4,5,7} - remove 8,9,10



Ancestral + Moralization examples

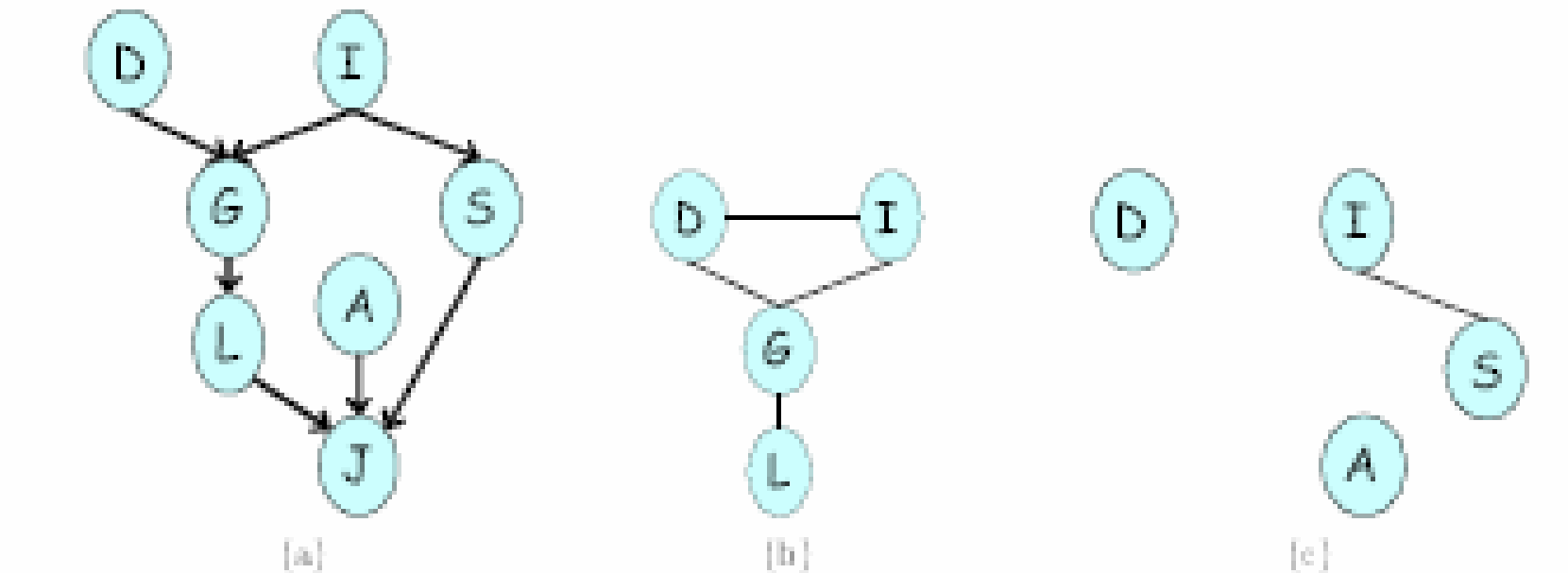


Figure 4.12 Example of alternative definition of d-separation based on Markov networks. (a) A Bayesian network $\mathcal{G}$ (b) The Markov network $\mathcal{M}[\mathcal{K}^+[D, I, L]]$ (c) The Markov network $\mathcal{M}[\mathcal{K}^+[D, I, A, S]]$

d-sep(D; I |L): false

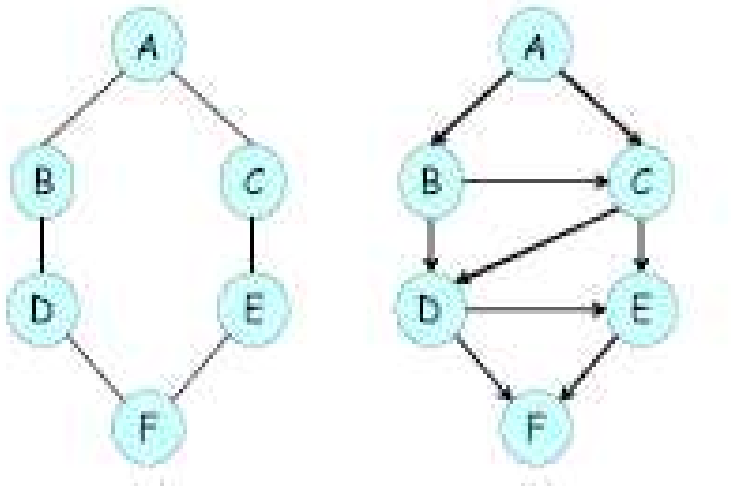
d-sep(D; I | S, A): true

d-Separation revisited

- Thm 4.5.6. Let U = X u Y u Z. Let G' = G[U] be the induced BN over U and Ancestors(U). Let H = moral(G'). Then d-sep$_G$(X;Y|Z) iff sep$_H$(X;Y|Z).

UG to DAG

- We may have to add many new edges



- Add nodes in alphabetical order.
- When add C, not B $\perp$ C | A hence add edge

Chordal graphs

- Def 2.2.15. Let X_1 – X_2 … - X_1 be a loop in a graph (ignoring edge directions). A chord is an edge connecting Xi, Xj for two non-consecutive nodes. A graph is chordal (triangulated) if every loop of length k>=4 has a chord.

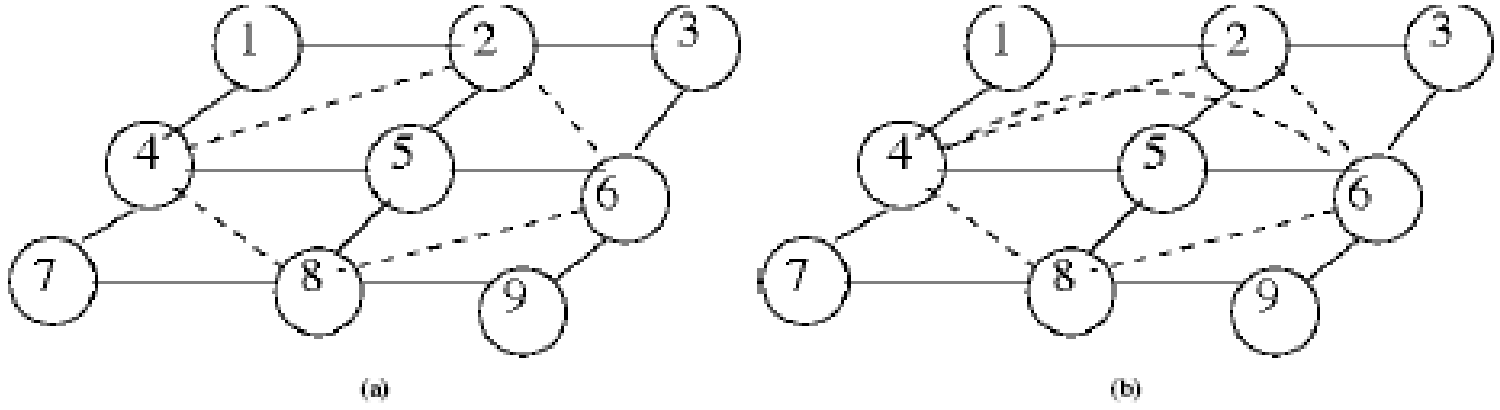
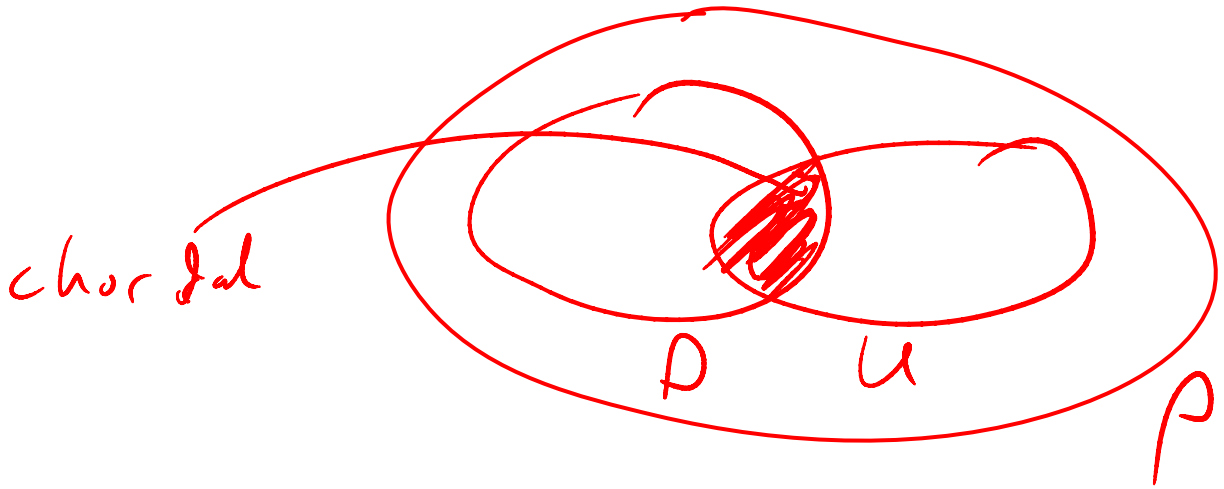


Figure 24.7: Left: this graph is not triangulated, even though it contains lots of little triangles. Note the dotted 2-4-6-8 loop is a chordless 4-cycle. Right: one possible triangulation, by adding the 4-6 fill-in edge. Alternatively we could add the 2-8 edge.

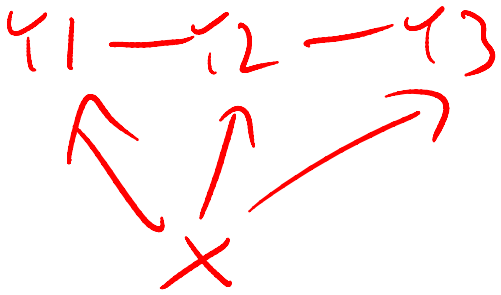
UG to DAG

- Thm 4.5.11. Let H be an MRF and G be any minimal I-map for H. Then G is necessarily chordal.
- The process of converting UG to DAG, and DAG to UG, adds edges.
- When does this conversion not add edges?
- Thm 4.5.16. I(H)=I(G) iff H and G are both chordal.



CRFs

- A Markov random field (MRF) is an unconditional density p(y) represented by an UG
- A conditional random field is a conditional density p(y|x) represented by an UG, where each y node is conditioned on (potentially) all the x nodes.
- Discriminative; no need to model input features.

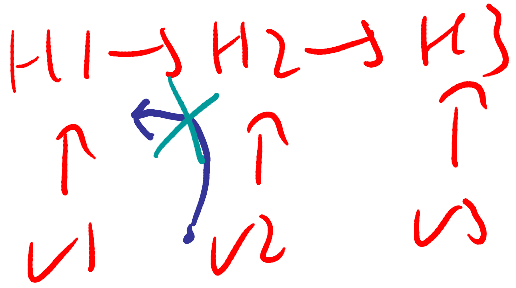


Cf logistic regression with multiple output nodes

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_c \phi(\mathbf{y}_c; \mathbf{x})$$

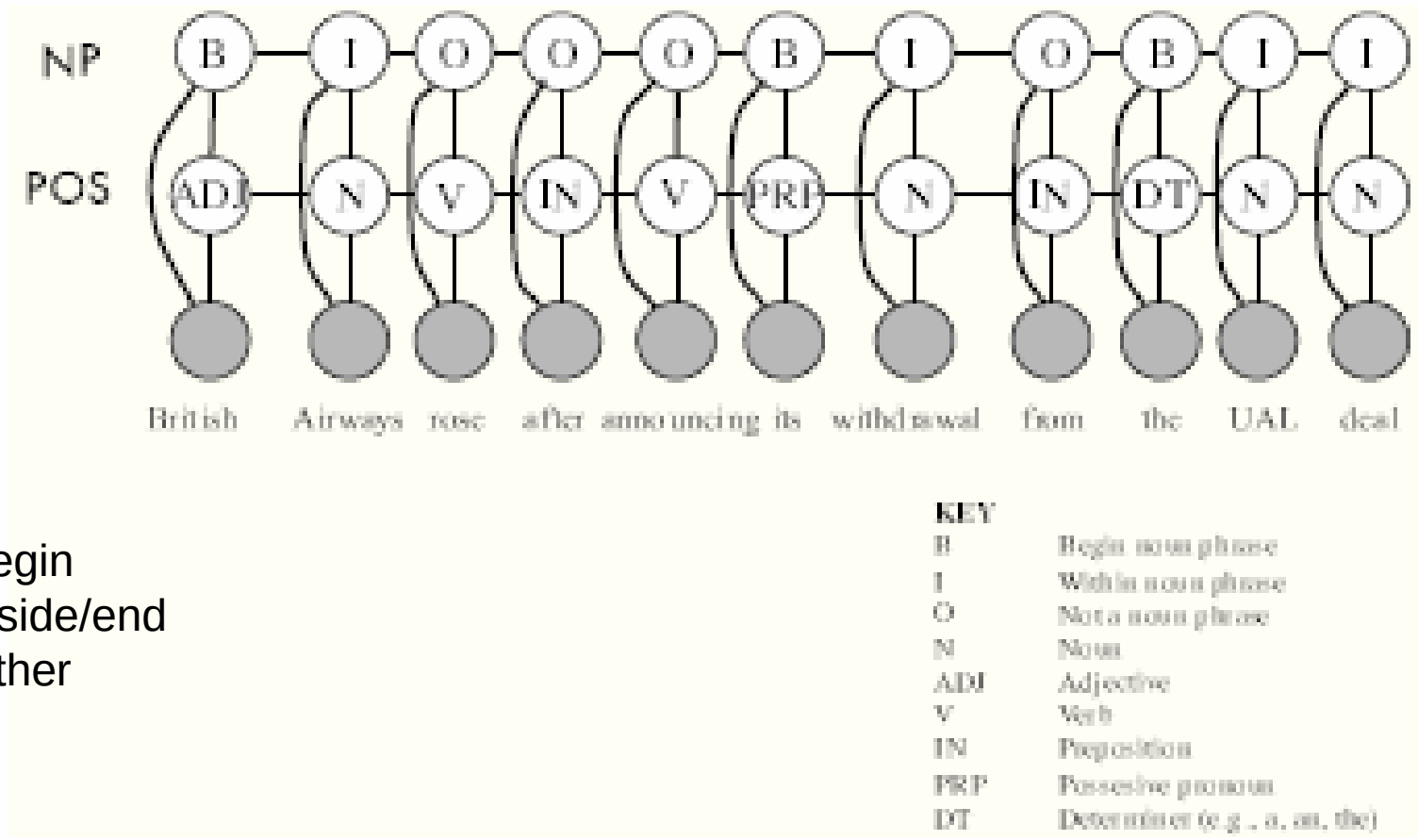
CRFs vs CBNs

- A conditional DAG can sometimes be inferior to a conditional UG due to blocking of info flow
- Eg consider a conditional chain (aka Maximum Entropy Markov Model): H1 $\perp$ V2 – no backwards information flow ("label bias problem") due to v-structure/ local normalization



$$p(\mathbf{h}|\mathbf{v}) = \prod_t p(h_t|h_{t-1}, \mathbf{v}_t)$$

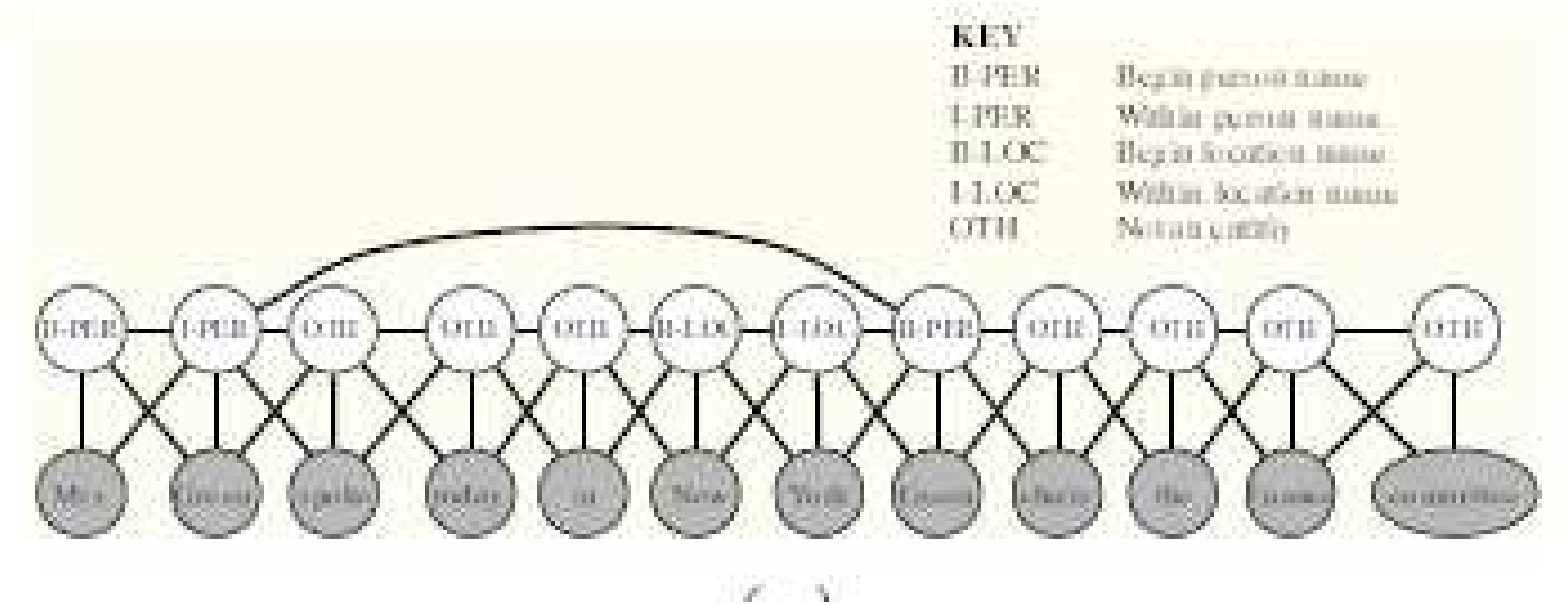
CRFs for text analysis



Begin
Inside/end
Other

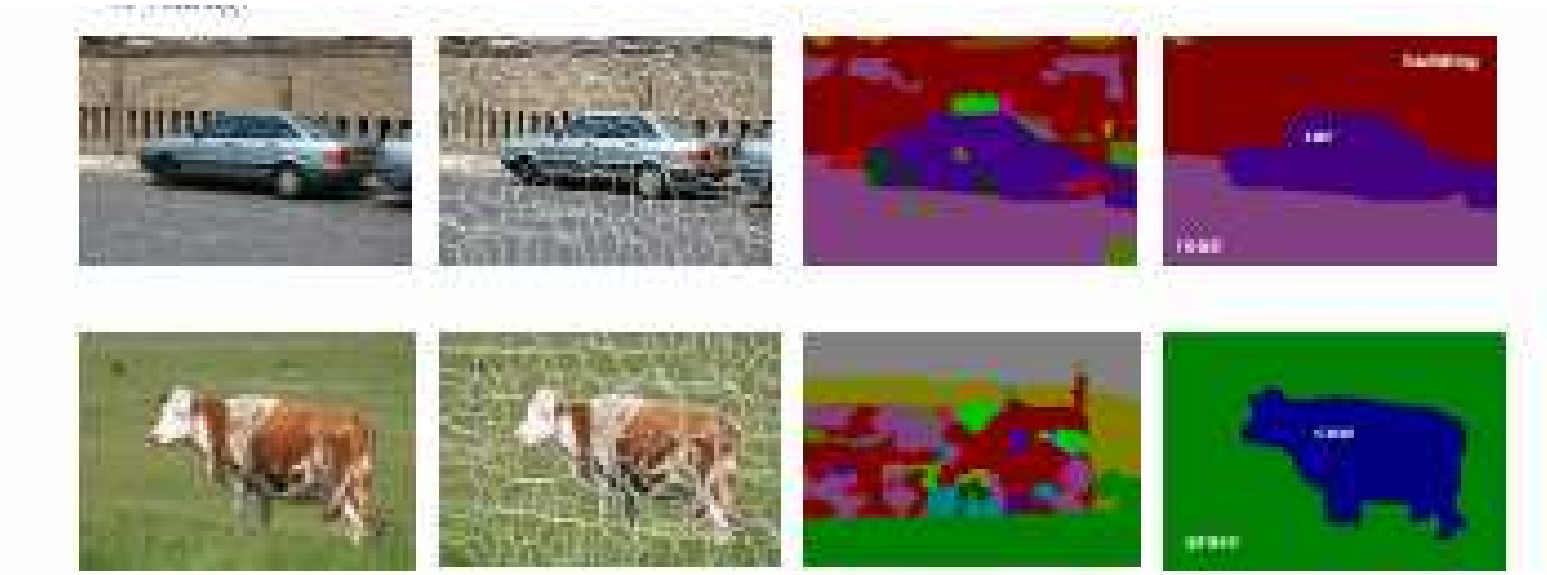
Part of speech tagging and noun-phrase segmentation

Skip-chain CRFs



Mrs Green spoke today in New York. Green chairs the finance committee.

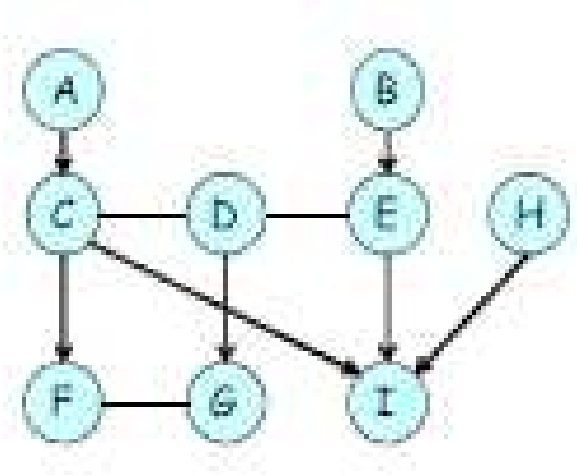
CRFs for low-level vision



Super-pixels Local classifier CRF

Chain graphs

- Each chain component is an UG, connected together in a DAG.



$$
\begin{aligned}
p(A,...,I) &= p(A)p(B)p(C,D,E|A,B)p(F,G|C,D)p(I|C,H)p(H) \\
p(C,D,E|A,B) &= \frac{1}{Z(A,B)}\phi_A(A,C)\phi_2(B,E)\phi_3(C,D)\phi_4(D,E) \\
p(F,G|C,D) &= ... \\
p(I|C,H) &= ...
\end{aligned}
$$

Stat 521A
Lecture 4

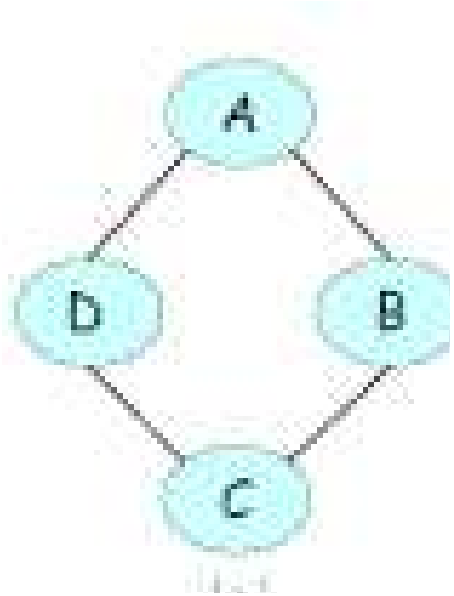
Admin

- CS auditors: please turn in your form to Joyce Poon, who will pass it to Laks for signing

Outline

- Aside on canonical parameterization (ex 4.4.14)
- Structured factors (4.4.1.2)
- Structured CPDs (5.2-5.6)
- Temporal models (6.2)

Degrees of freedom of a UGM



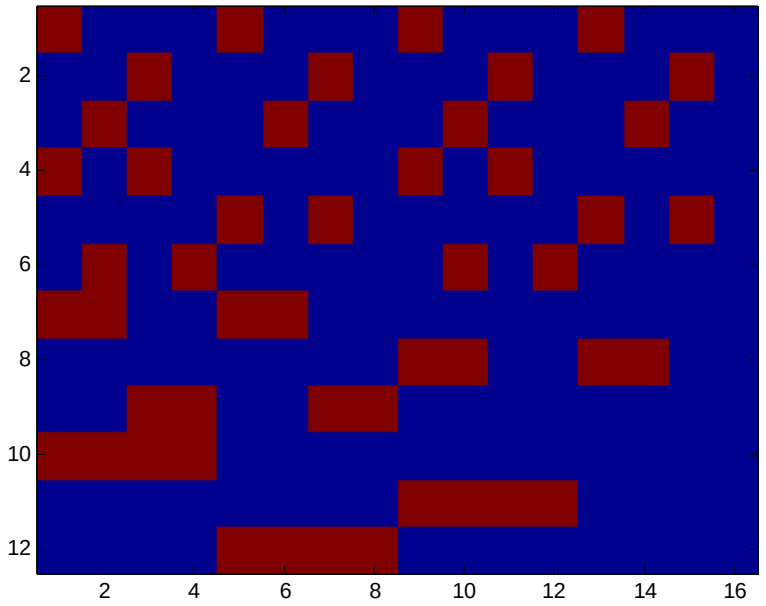
Why do we just need 8 numbers to uniquely parameterize the distribution?

Eg a^1, b^1, c^1, d^1, (a^1,b^1), (b^1,c^1), (c^1,d^1), (a^1,d^1)

Num params = rank of feature matrix

- Let F(n, i)=1 iff i'th bit vector turns on n'th feature
- Each feature specifies a value for every pair of nodes connected by an edge, and hence is a vector in R^{16}. 4 edges, 3 unique settings = 12 rows.

Rank = 8



Eg a^1, b^1, c^1, d^1, (a^1,b^1), (b^1,c^1), (c^1,d^1), (a^1,d^1)

Rank of feature matrix

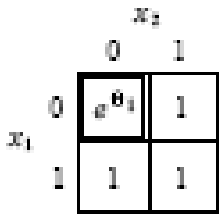
```
edges = {[1 2], [1 3], [2 4], [3 4]};
ndx  = 1;
F = zeros(0, 2^4);
for e=1:length(edges)
  s = edges{e}(1); t = edges{e}(2);
  for j=1:2
    for k=1:2
      if j==2 && k==2, continue; end
      for x=1:16
        xv= ind2subv([2 2 2 2], x);
        if xv(s)==j && xv(t)==k
          F(ndx,x)=1;
        end
      end
      ndx = ndx + 1;
    end
  end
end
rank(F)
```

Log-linear factors

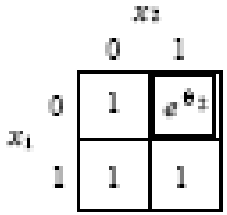
- A factor defined on m discrete rv's with K states needs K^m parameters.
- Imagine a factor on triples of letters. Instead of having 26^3 numbers, we can define binary features that only turn on for certain values, eg $f_{ing}(x) = 1$ iff x_1='I',x_2='n',x_3='g'. This has weight ω_{ing}. We define

$$\phi_c(\mathbf{x}_c) = \exp(\sum_{i=1}^{k} w_{c,i} f_{c,i}(\mathbf{x}_c))$$

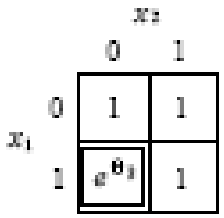
Tables are a special case



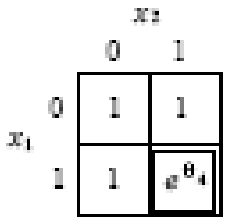
$$f_1(x_1, x_2) = \delta(x_1 = 0, x_2 = 0)$$



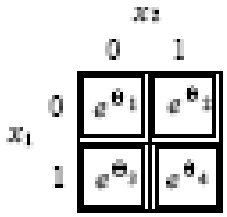
$$f_2(x_1, x_2) = \delta(x_1 = 0, x_2 = 1)$$



$$f_3(x_1, x_2) = \delta(x_1 = 1, x_2 = 0)$$



$$f_4(x_1, x_2) = \delta(x_1 = 1, x_2 = 1)$$



$$q_{\theta}(x_1, x_2) = e^{\theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4}$$

Jordan, fig 19.1

CRF features

- Typical features used in a CRF model for language processing (X=words, Y=labels)
- $F_1(Y_t,X_t,X_{t-1},X_t+1) = I(X_{t-1}=\text{“New”}, X_t=\text{“York”}, X_{t+1}=\text{“Times”}, Y_t=\text{“Object”})$
- $F_2(Y_t,X_t,X_{t-1},X_t+1) = I(X_{t-1}=\text{“New”}, X_t=\text{“York”}, X_{t+1} \neq \text{“Times”}, Y_t=\text{“Place”})$
- Models often have ~100k manually specified features.
- Common to use L1 regularization to sparsify.
- Can also perform feature induction, by eg greedily creating conjunctions or disjunctions

Exponential family (maxent) models

- Combining all the local potentials

$$p(\mathbf{x}) = \frac{1}{Z}\prod_c \phi_c(\mathbf{x}_c)$$

$$\phi_c(\mathbf{x}_c) = \exp(\sum_{i=1}^{k} w_{c,i} f_{c,i}(\mathbf{x}_c))$$

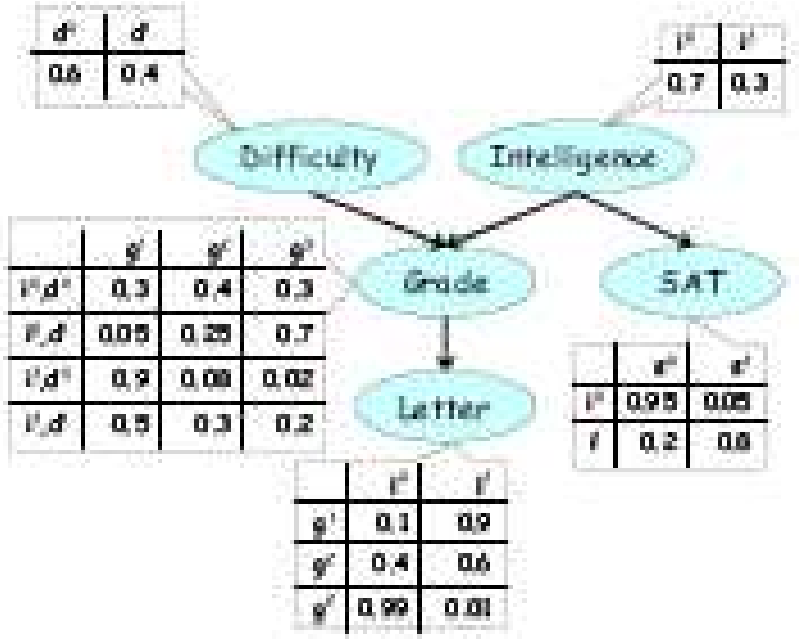
$$p(\mathbf{x}) = \frac{1}{Z}\exp(\sum_i w_i f_i(\mathbf{x}_{c_i}))$$

DAGs are a special case where each $\phi_c(x_c) = p(X_i|Pa(X_i))$ sums to 1, so Z=1

See ch 8

Tabular CPDs

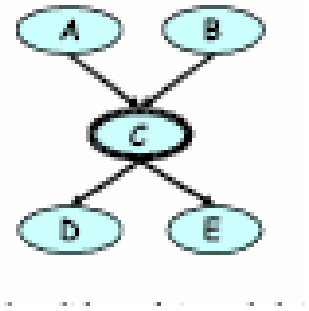
- If all nodes are discrete and have K values, we can represent p(X_i|Pa(X_i)) as a table, with one row per conditioning case (K^#pa), and K columns which sum to 1
- If K and/or #pa is large, this is too many parameters, so we seek more parsimonious representations.



Deterministic CPDs

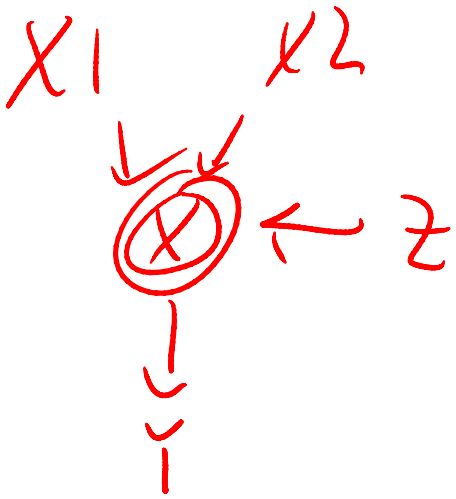
- In some cases, the child is a deterministic function of the parents, eg bloodtype is determined by the 2 alleles
- Deterministic nodes often denoted by double-ringed oval.
- Determinism can imply additional (non-graphical) independencies
- Eg $D \perp E \mid A,B$ since $C = fn(A,B)$

Det-sep



Context specific independence (CSI)

- Sometimes, the set of edges which are “active” depends on the value of the nodes
- Eg Y is a noisy observation of object X1, or X2. Z specifies the identity of the measurement. Let X =multiplexer(X1,X2, Z). Then $X2 \perp Y \mid Z=1$. So our posterior on X2 is not affected by the measurement. (Data association ambiguity)

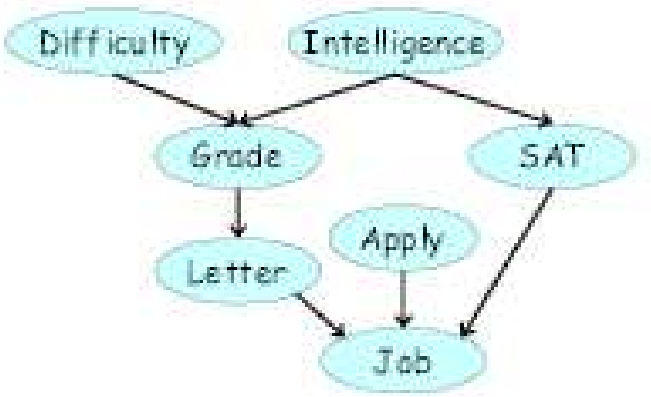


Contingently acyclic BNs

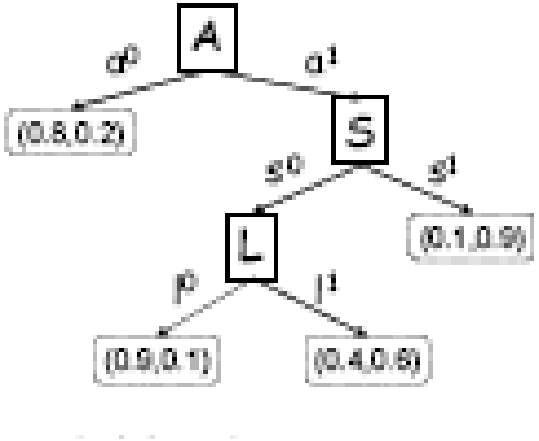
- Sometimes we can define a directed graph with cycles, but where some of the edges are not active for a given setting of certain variables C.
- If we can guarantee that the graph is a DAG for each context C=c, the result is a mixture of differently structured BNs.
- This is called a Bayesian multinet.

Tree-structured CPDs

- Different parents can be rendered irrelevant, depending on the values



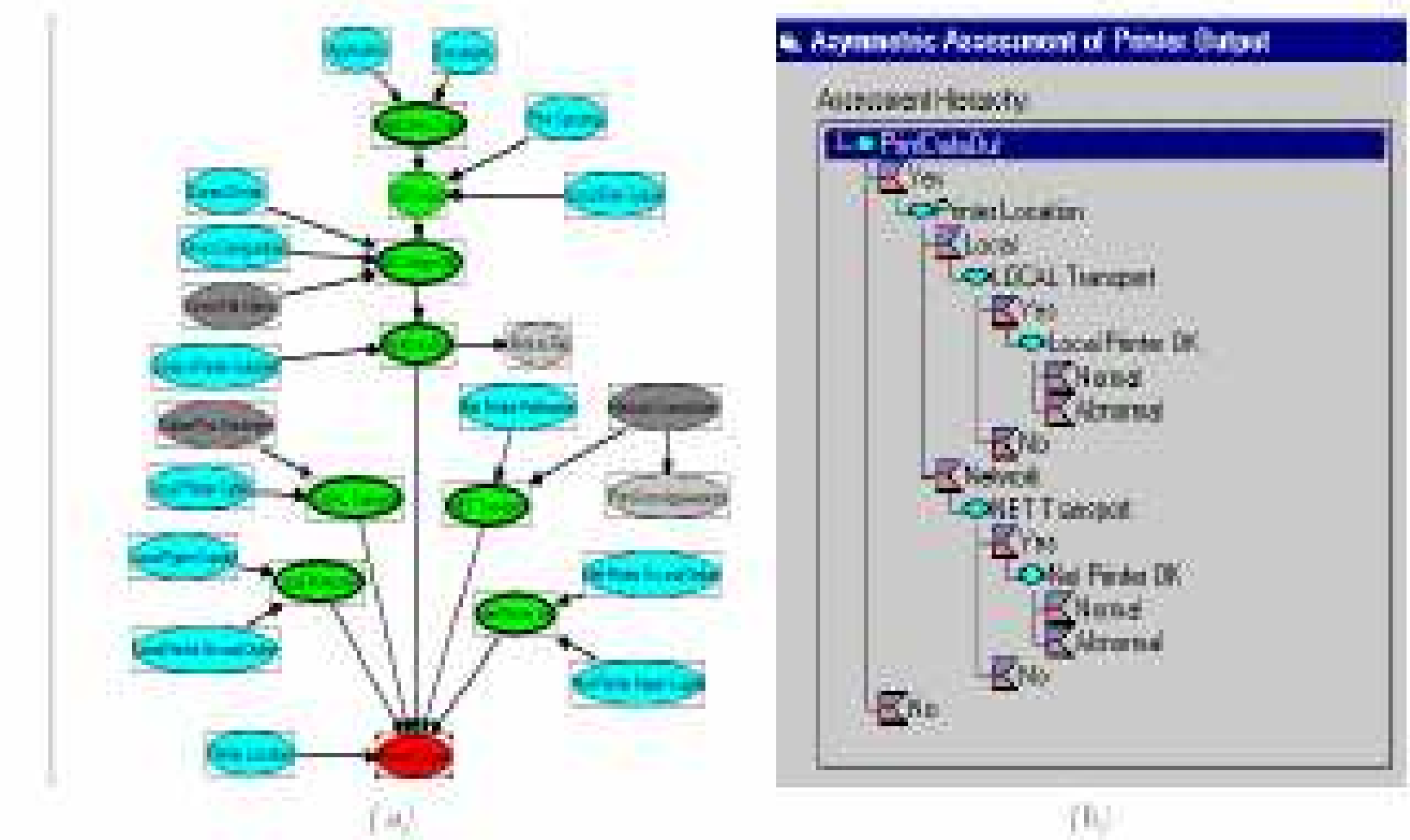
P(J|A,S,L)



Eg. J | S,L if A=0 since we go down left branch of tree

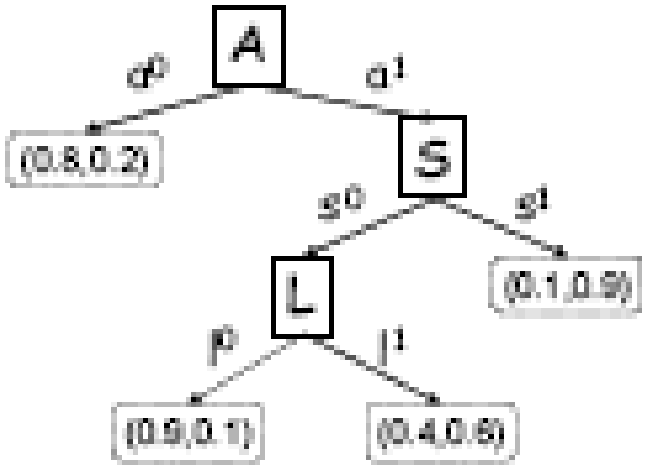
Printer fault diagnosis in MS windows

- Uses tree structured CPDs, since different sets of variables are relevant in different contexts



Rule-structured CPDs

- Specify a pattern and a value

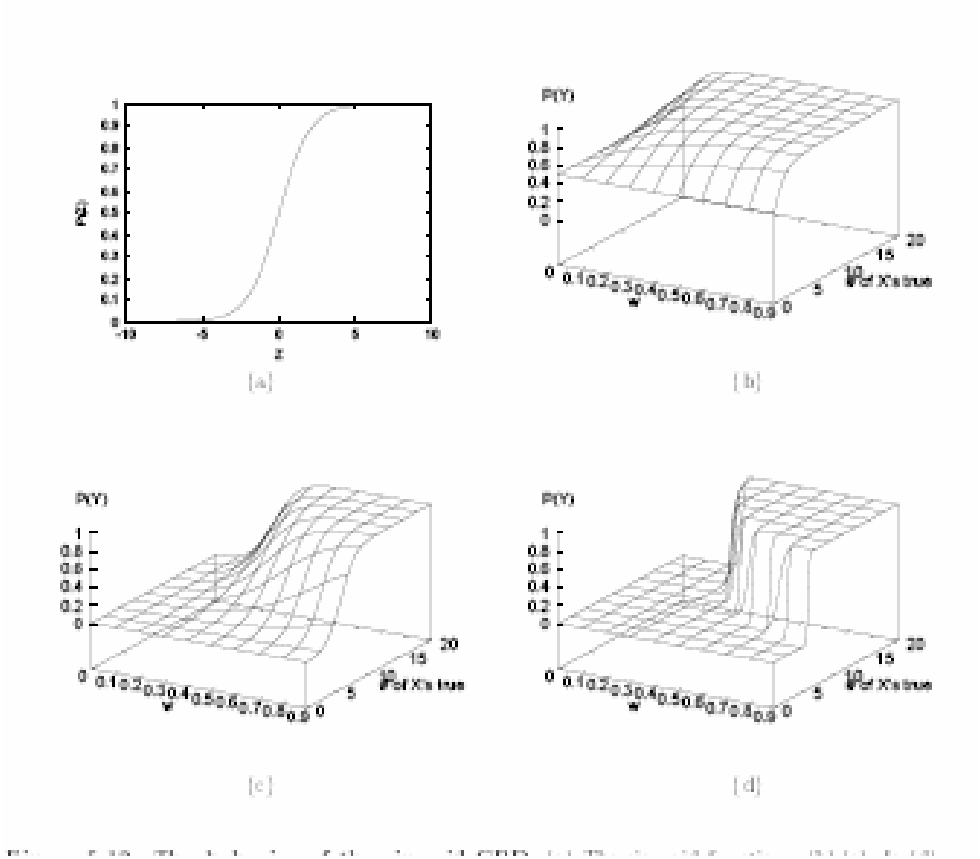


$$\left\{ \begin{array}{l} \rho_1 : \{a^0, j^0; 0.8\} \\ \rho_2 : \{a^0, j^1; 0.2\} \\ \rho_3 : \{a^1, s^0, l^0, j^0; 0.9\} \\ \rho_4 : \{a^1, s^0, l^0, j^1; 0.1\} \\ \rho_5 : \{a^1, s^0, l^1, j^0; 0.4\} \\ \rho_6 : \{a^1, s^0, l^1, j^1; 0.6\} \\ \rho_7 : \{a^1, s^1, j^0; 0.1\} \\ \rho_8 : \{a^1, s^1, j^1; 0.9\} \end{array} \right\}$$

Logistic regression (sigmoid BNs)

- Suppose all nodes are binary. We can use logreg CPDs

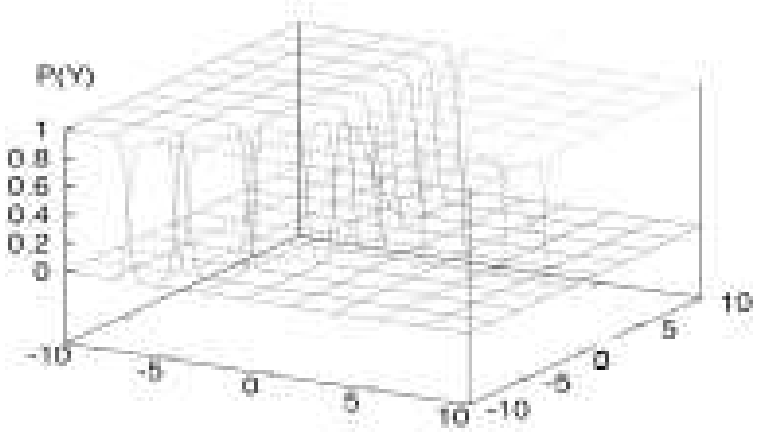
$$p(y=1|\mathbf{x}) = \sigma(w_0 + \sum_{i=1}^{k} w_i x_i) \quad \sigma(u) = \frac{1}{1+e^{-u}}$$



Multinomial logreg

- If Y is K-ary, and the parents are binary or cts, we can use a softmax function

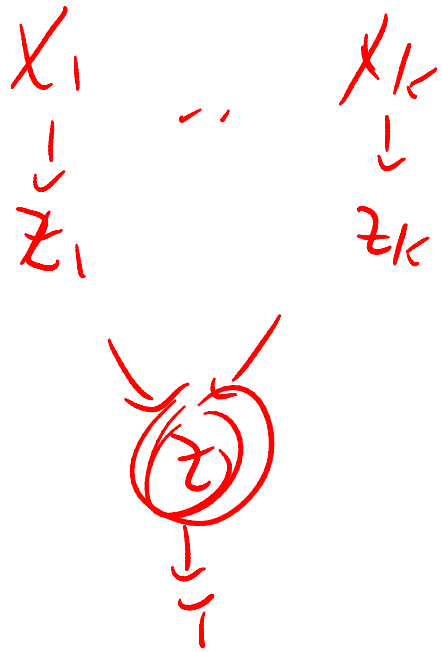
$$p(y=j|\mathbf{x}) = \frac{\exp(\mathbf{w}_j^T \mathbf{x})}{\sum_{j'=1}^{K} \exp(\mathbf{w}_{j'}^T \mathbf{x})}$$



For K-ary parents, use 1-of-K encoding

Independence of causal influence

- We can model the effects of many parents by assuming that each parent is corrupted by independent noise, and the results are deterministically combined via a simple function such as OR or MAX



Noisy-or model

- Each Xi in {0,1} gets passed through a noisy wire to produce Zi in {0,1}. 0 maps to 0, 1 maps to 0 wp w_i (failure probability). λ_i=1-w_i is the prob. that Xi alone turns on Y.
- The Zi's are combined in an OR to produce Z. Then Y=Z.
- The only way Y can be off is if all Zi's are off, which means all the wires for Xi st Xi=1 independently failed:

$$p(y=0|\mathbf{x}) = \prod_{i:x_i=1} w_i = \prod_{i=1}^{k} w_i^{x_i}$$

$$p(y=1|\mathbf{x}) = 1 - p(y=0|\mathbf{x})$$

Popular in cogsci models of causality

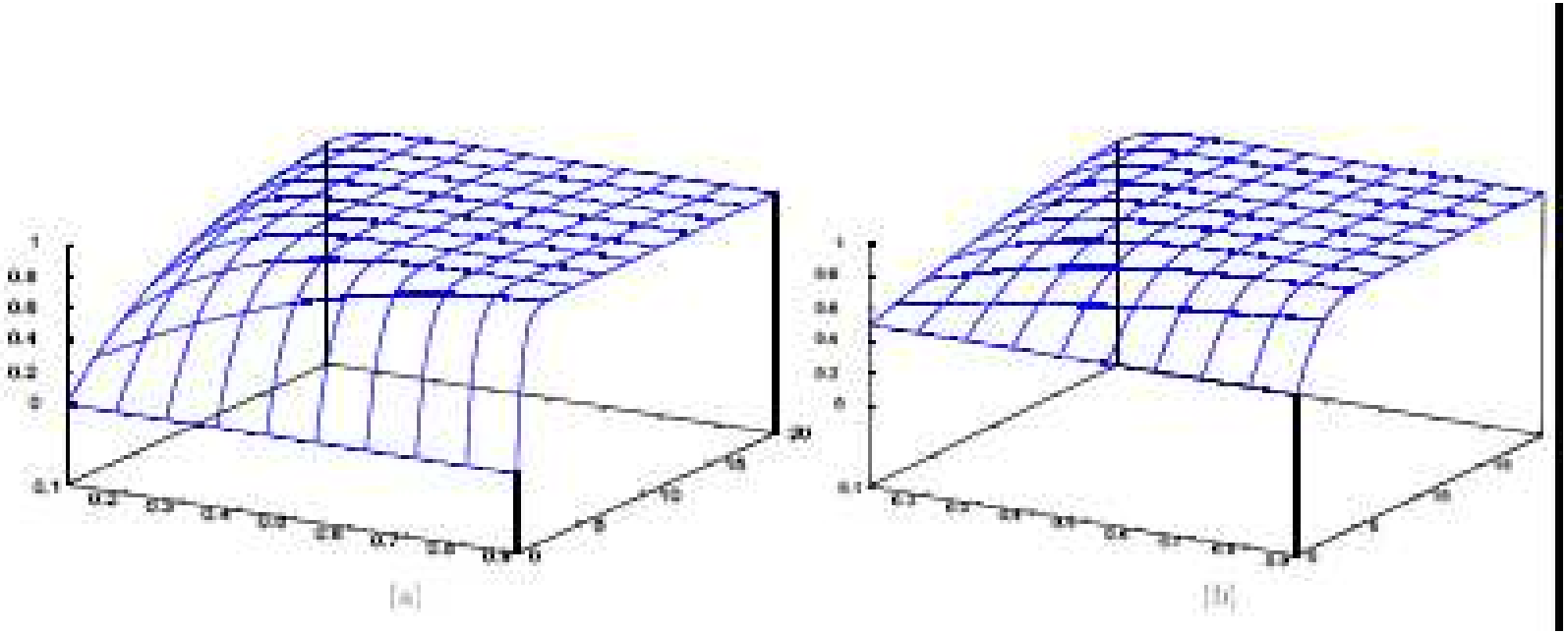
Example

- P(fever=0|cold=1, flu=0, malaria=0)=0.6
- P(fever=0|cold=0, flu=1, malaria=0)=0.2
- P(fever=0|cold=0, flu=0, malaria=1)=0.1

Cold	Flu	Malaria	p(Fever=1)	p(Fever=0)
0	0	0	0.0	1.0
0	0	1	0.0	0.1
0	1	0	0.8	0.2
0	1	1	0.98	$0.02 = 0.2 \times 0.1$
1	0	0	0.4	0.6
1	0	1	0.94	$0.0.6 = 0.6 \times 0.1$
1	1	0	0.88	$0.12 = 0.6 \times 0.2$
1	1	1	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

Leak nodes

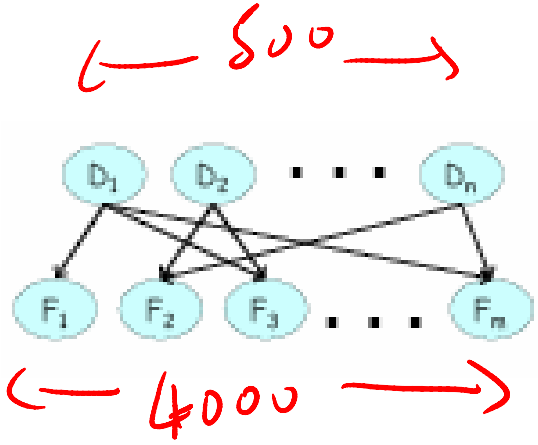
- If Y=0 and all Xi=0, the CPD assigns 0 probability to this event. To prevent this, we add a leak node, X0=1, which is always on, to model “any other cause”. The leak can fail wp w0.



W0=1 W0=0.5

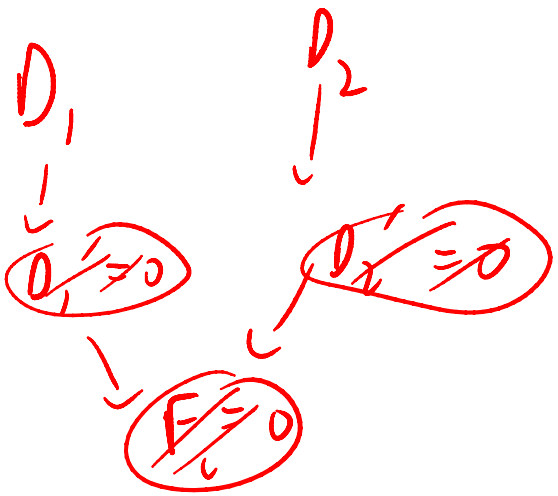
BN20 networks

- In medical diagnosis, it is common to construct 2 layered bipartite networks of binary nodes, mapping diseases to symptoms (findings).
- Because of the large number of parents, the child nodes use noisy-or.
- Conditional on F, the diseases D are correlated.
- The QMR-DT network is a standard testbed for evaluating approximate inference algorithms.



Negative findings

- If Fi=1, the disease parents fight to explain the finding. Hence they become fully correlated.
- But if Fi=0, the parents are independent! Hence the p(Fi=0|Pa(Fi)) likelihood fully factorizes, and does not make inference harder (homework).



$D_1 \perp D_2 \mid F_i = 0$

Conditional linear Gaussian CPDs

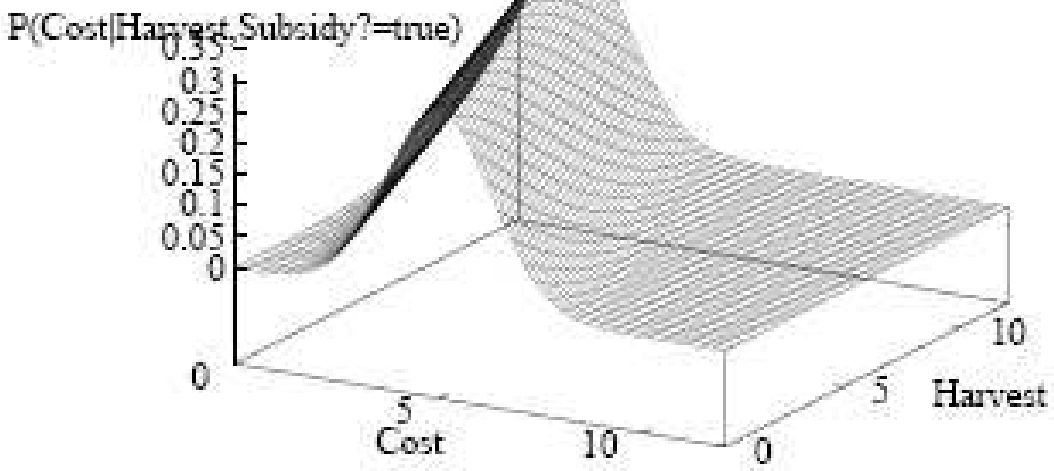
- If Y is continuous and all the parents are cts we can define

$$p(y|\mathbf{x}) = \mathcal{N}(y|\mathbf{x}^T\mathbf{w}, \sigma^2)$$

- Networks of linear Gaussian CPDs define a joint multivariate Gaussian (see ch 7)
- For discrete parents u, we can use 1-of-K and LG, or we can use a different set of parameters for each discrete setting (CLG). The resulting distribution is a mixture of Gaussians, where each discrete setting defines a mixture component.

$$p(y|\mathbf{x}, \mathbf{u} = k) = \mathcal{N}(y|\mathbf{x}^T\mathbf{w}_k, \sigma_k^2)$$

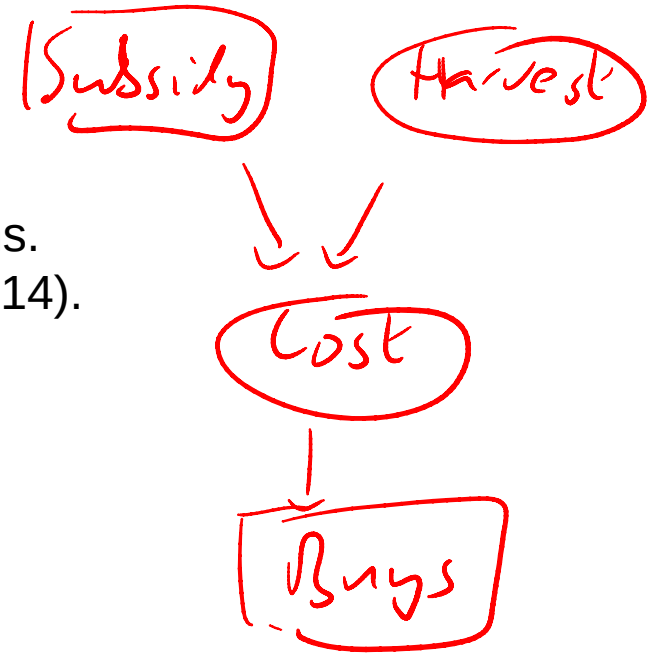
Example of CLG network



Russell & Norvig, p502

Hybrid network

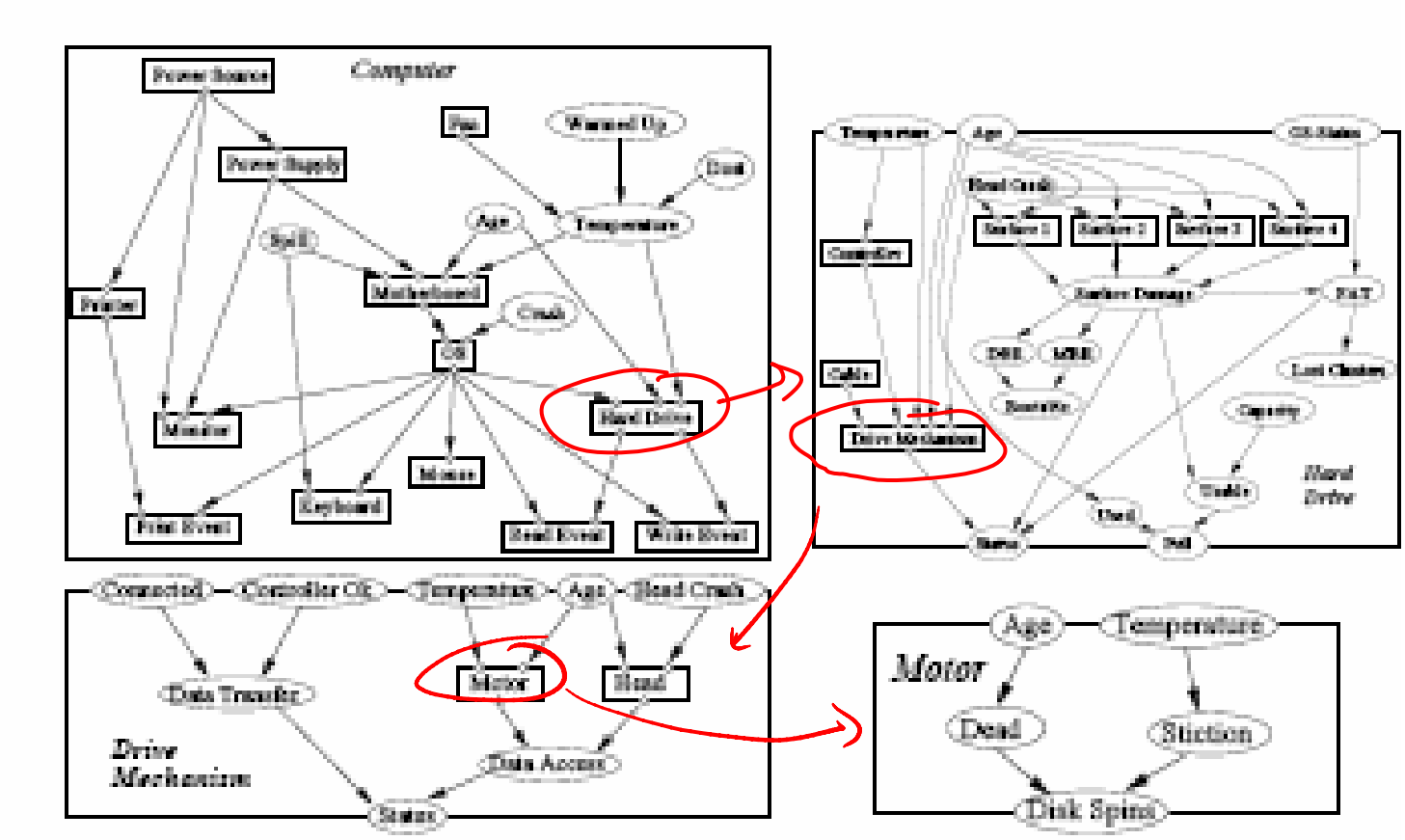
P(buys=1|cost) = logreg or probit.
Joint distribution is no longer mixture of Gaussians.
Closed-form inference no longer possible (see ch14).



Russell & Norvig, p502

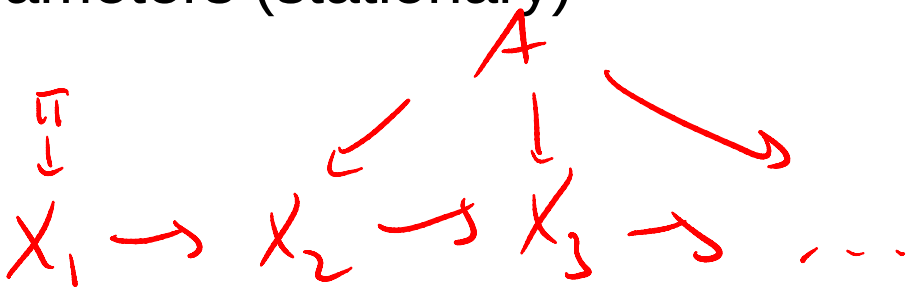
Encapsulated BNs

- We can embed a BN inside a CPD, and “hide” the internal nodes using an interface layer.
- This, combined with parameter tying, yields OOBN.



Markov chains

- We can define a distribution over a semi-infinite sequence X_1, X_2, … by using a discrete-time Markov chain with tied parameters (stationary)

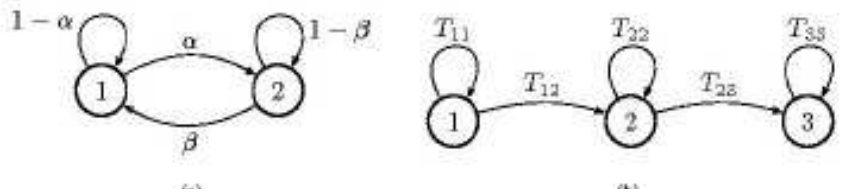


$$p(\mathbf{x}|\boldsymbol{\theta}) = p(x_1|\pi)\prod_{t=T}^{\infty} p(X_t|X_{t-1}, A)$$

$$A(i,j) = p(X_t = j|X_{t-1} = i)$$

State transition diagram

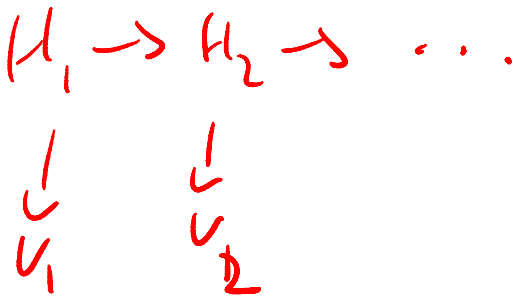
Picture of the stochastic finite state automaton



$$T = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \qquad T = \begin{pmatrix} T_{11} & T_{12} & 0 \\ 0 & T_{22} & T_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

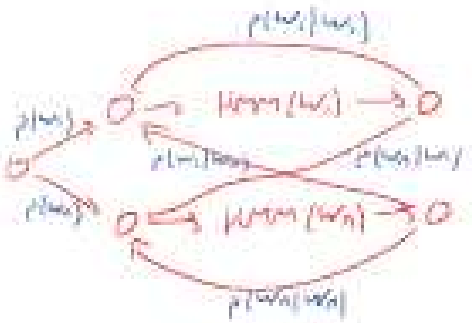
Hidden Markov Models

- An HMM is a function of a Markov chain.
- We observe V_t, hidden state is H_t in {1,…,K}
- $P(H_t=j|H_{t-1}=i)$ is the transition model
- $P(V_t|H_t=j)$ is the observation model (eg mixture of Gaussians)



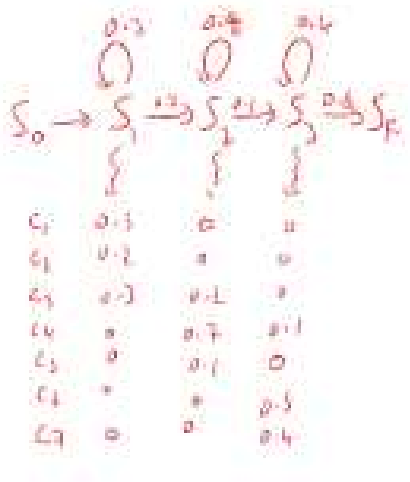
HMMs for speech recognition

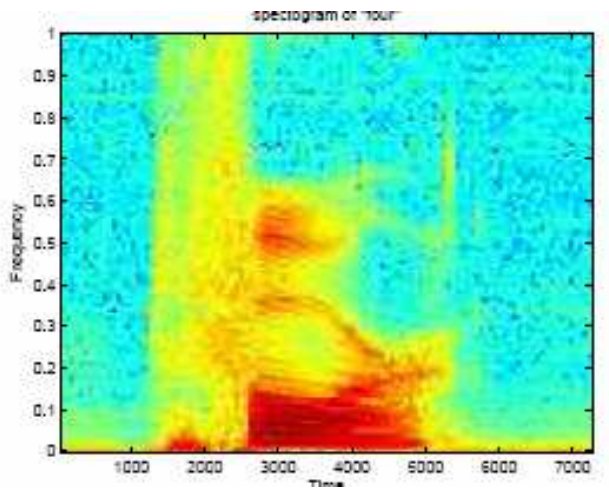
Bigram model of words



Pronunciation model : word -> phonemes

Acoustic model: phonemes -> observations





State space models

- Same graph (CI assumptions) as HMM, but now X and Y are real-valued vectors
- Special case: linear dynamical system (LDS)

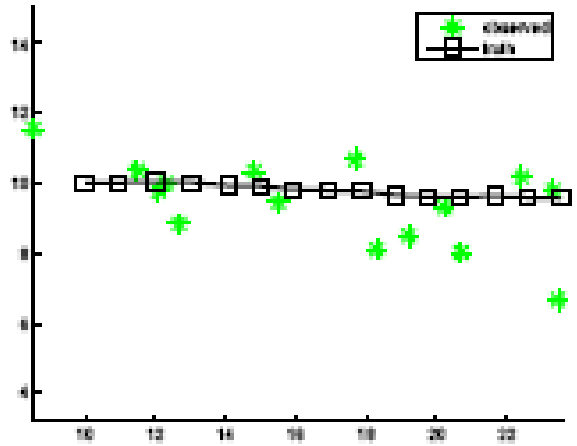
$$
\begin{aligned}
p(\mathbf{x}_t|\mathbf{x}_{t-1}) &= \mathcal{N}(\mathbf{x}_t|\mathbf{A}\mathbf{x}_{t-1}, \mathbf{Q}) \\
p(\mathbf{y}_t|\mathbf{x}_t) &= \mathcal{N}(\mathbf{y}_t|\mathbf{H}\mathbf{x}_t, \mathbf{R})
\end{aligned}
$$

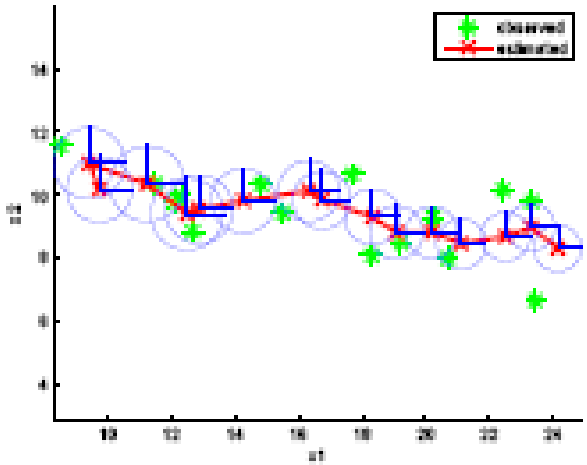
$$
\begin{aligned}
\mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \mathcal{N}(\mathbf{0}, \mathbf{Q}) \\
\mathbf{y}_t &= \mathbf{H}\mathbf{x}_t + \mathcal{N}(\mathbf{0}, \mathbf{R})
\end{aligned}
$$

Example: tracking in 2D

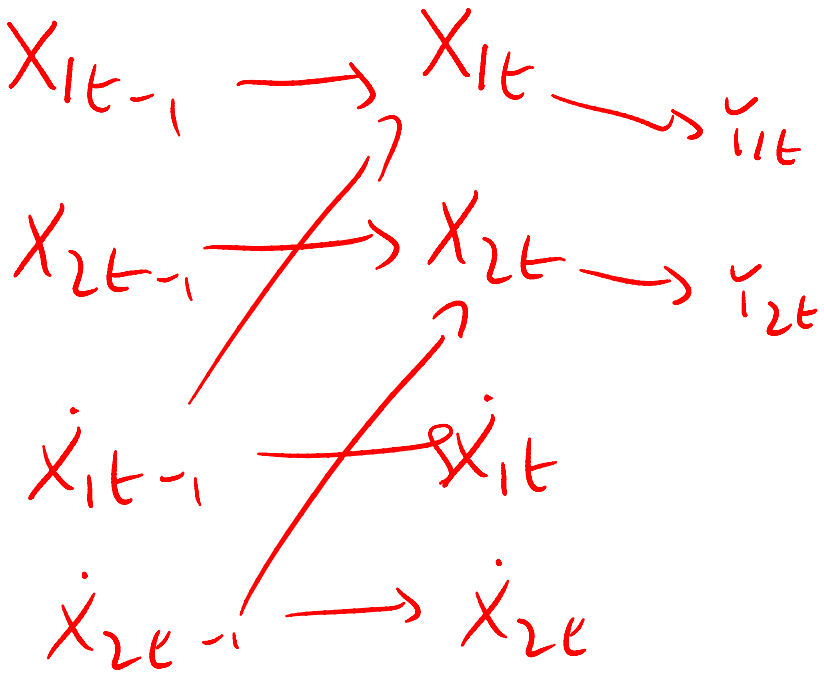
$$\begin{pmatrix} x_{1t} \\ x_{2t} \\ \dot{x}_{1t} \\ \dot{x}_{2t} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x_{1t-1} \\ x_{2t-1} \\ \dot{x}_{1t-1} \\ \dot{x}_{2t-1} \end{pmatrix} + \begin{pmatrix} w_{1t} \\ w_{2t} \\ w_{3t} \\ w_{4t} \end{pmatrix}$$

$$\begin{pmatrix} y_{1t} \\ y_{2t} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} x_{1t} \\ x_{2t} \\ \dot{x}_{1t} \\ \dot{x}_{2t} \end{pmatrix} + \begin{pmatrix} v_{1t} \\ v_{2t} \\ v_{3t} \\ v_{4t} \end{pmatrix}$$





LDS as DGM

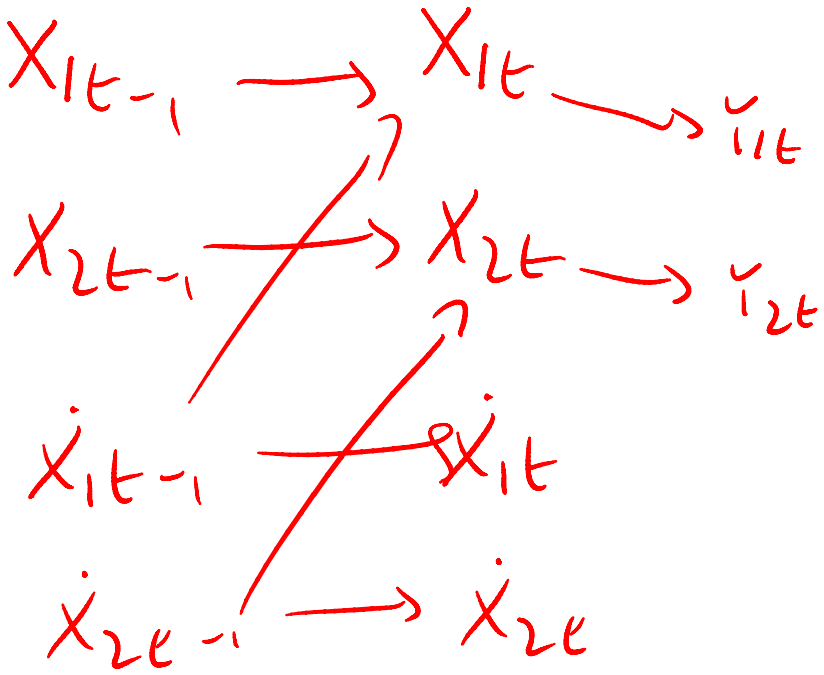


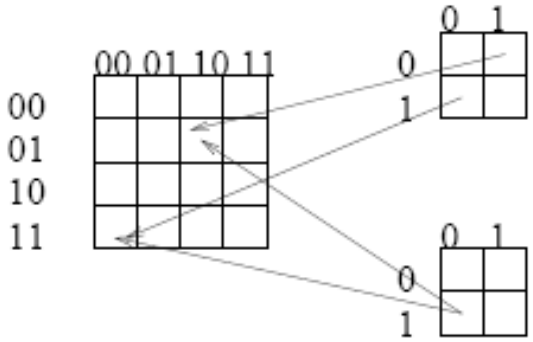
$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

For linear Gaussian systems, sparse matrices = sparse graphs

Dynamic Bayes Nets





P(X1(t),X2(t) | X1(t-1),X2(t-1)

If the variables are discrete, the transition matrix of the compound model (all 4 variables) is not sparse or structured. So the graph structure is crucial.

See ch 15

Stat 521A
Lecture 5

Outline

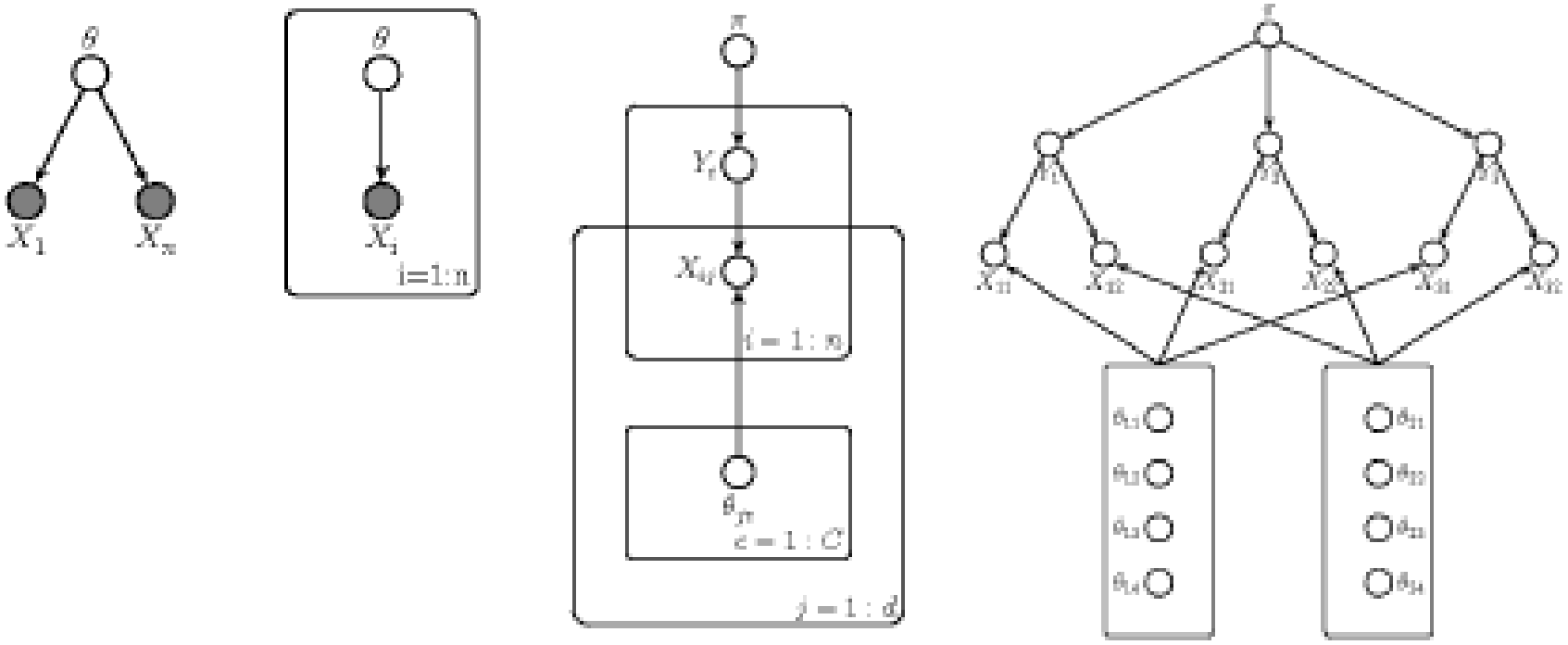
- Template models (6.3-6.5)
- Structural uncertainty (6.6)
- Multivariate Gaussians (7.1)
- Gaussian DAGs (7.2)
- Gaussian MRFs (7.3)

Parameter tying

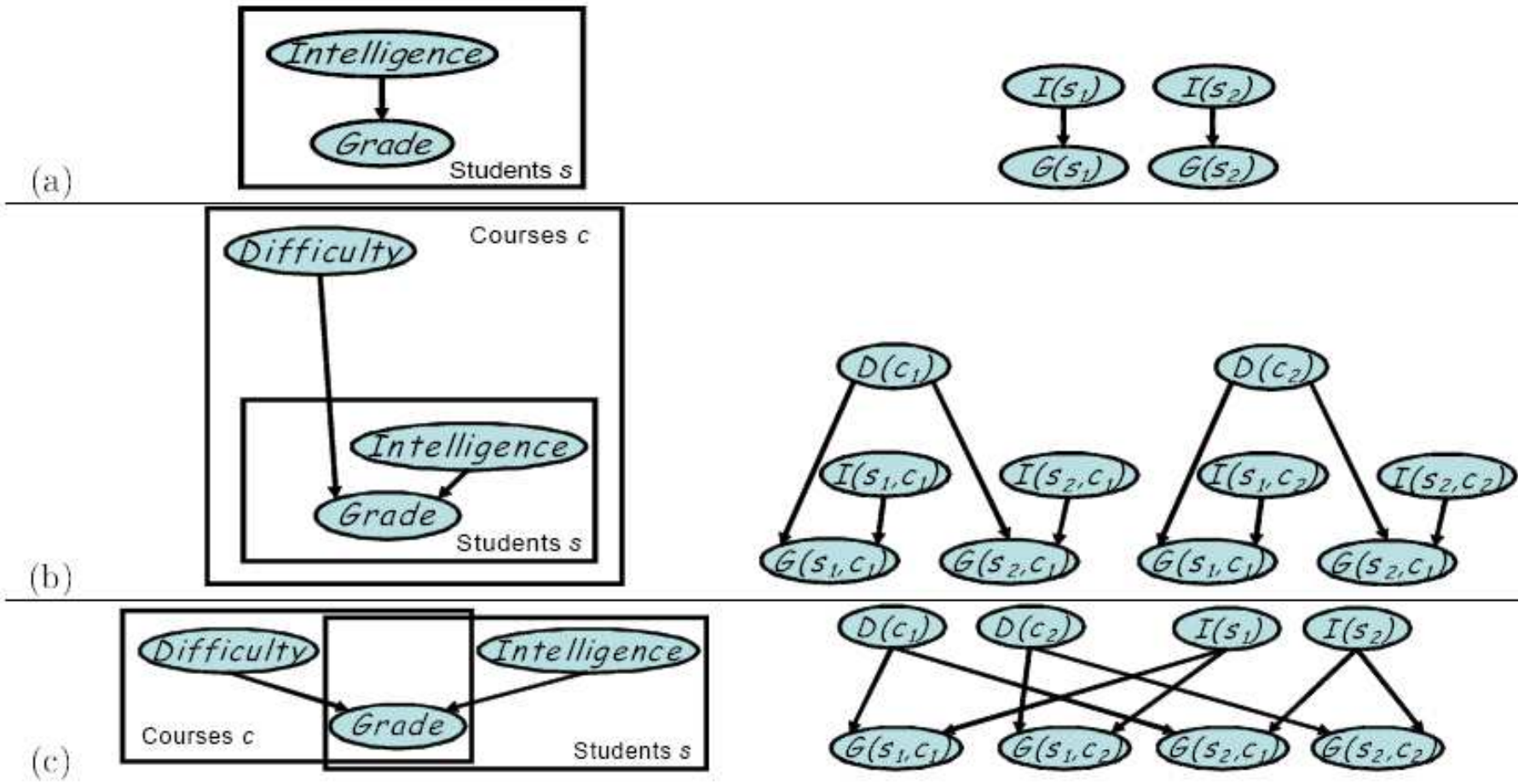
- A DBN defines a distribution over an unboundedly large number of variables by assuming that they all share the same CPDs.
- This is called parameter tying (weight sharing).
- It is useful even for fixed sized models in order to help learning (pool the sufficient statistics).
- We now discuss notational conventions (“syntactic sugar”) for representing large “unrolled” networks with shared parameters.

Plates

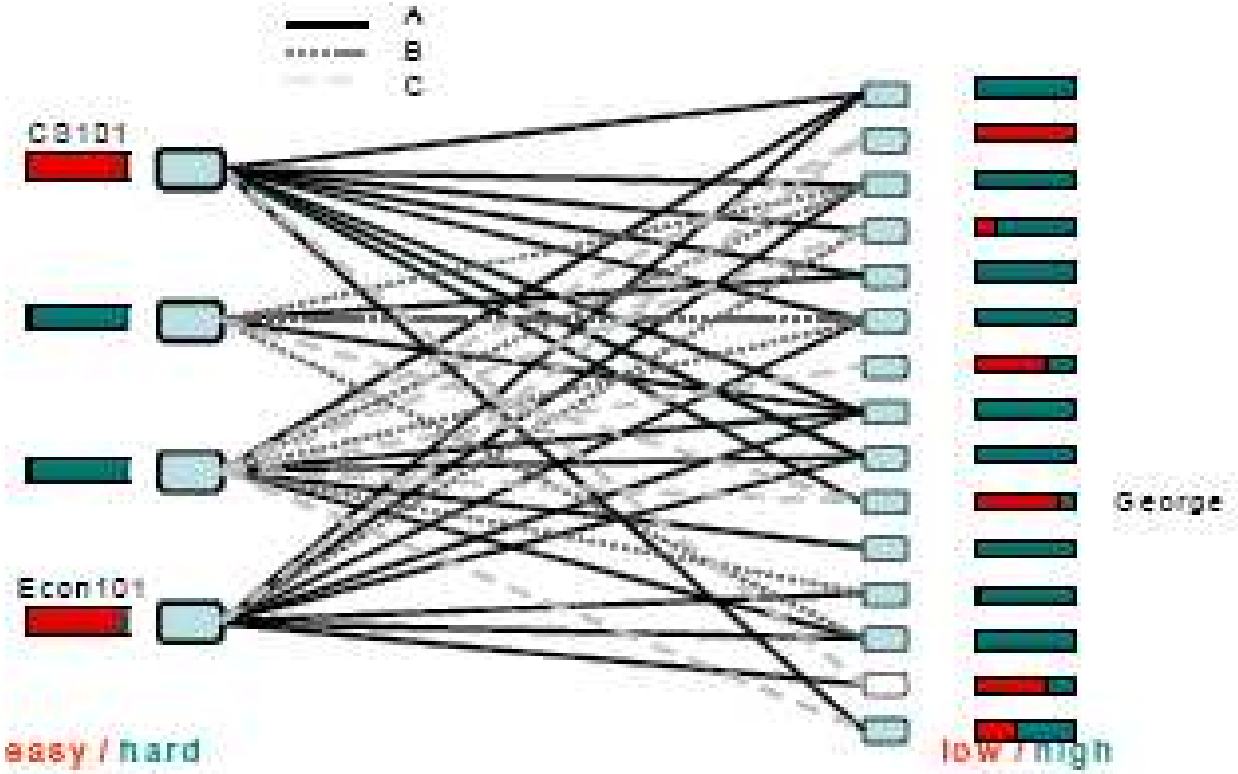
- Plates are useful for specifying simple repetitive patterns, as frequently arise in hierarchical Bayesian models



Plates


(a)
Intelligence
Grade
Students s
I(s1)
I(s2)
G(s1)
G(s2)
(b)
Difficulty
Courses c
Intelligence
Grade
Students s
D(c1)
D(c2)
I(s1,c1)
I(s2,c1)
I(s1,c2)
I(s2,c2)
G(s1,c1)
G(s2,c1)
G(s1,c1)
G(s2,c1)
(c)
Difficulty
Intelligence
Courses c
Grade
Students s
D(c1)
D(c2)
I(s1)
I(s2)
G(s1,c1)
G(s1,c2)
G(s2,c1)
G(s2,c2)

Unrolled network



Grade(s,c) in {A,B,C} is encoded on edges.
Cf discrete probabilistic matrix factorization

$$\begin{matrix} & D_1 \cdots D_4 \\ I_1 \\ \vdots \\ I_{14} \end{matrix} \begin{pmatrix} & \\ & G \\ & \end{pmatrix}$$

Limitations of plates

- There are various structures that plates cannot represent
- Eg DBNs
- Eg genotype(x1) depends on genotype(x2), where x2=parent(x1)
- We can write programs to generate graphs of specified structure, but we would like a declarative representation language for such repetitive patterns so that no new code has to be written

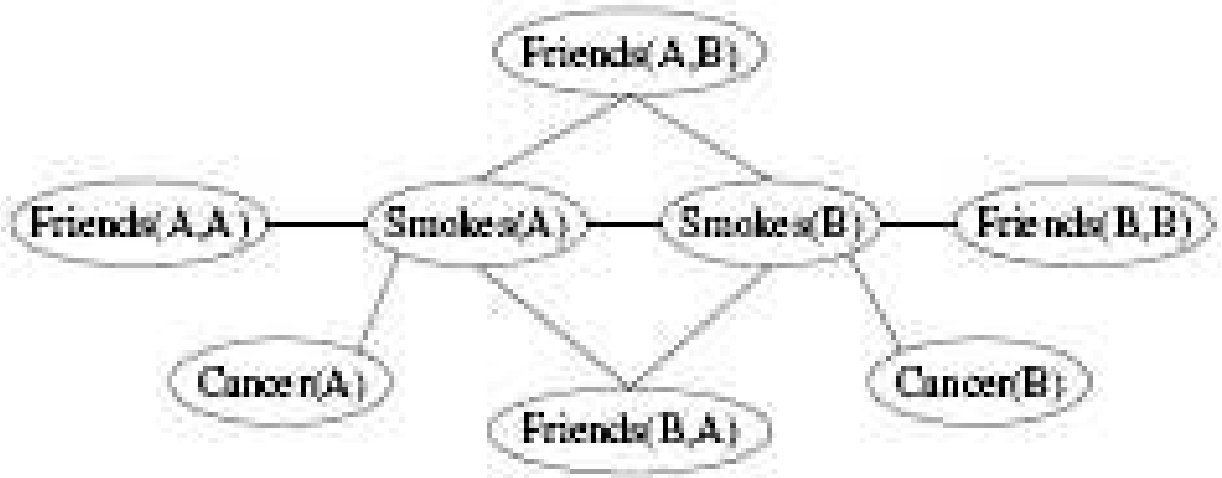
Beyond plates

- Probabilistic Relational Models (PRMs) encode large DAG models with tied CPDs
- Relational Markov Networks encode large MRFs with tied factors
- Markov Logic Networks are like RMNs, except the factors are represented in log-linear form, and the features are represented as logical expressions

Markov Logic Networks

Table I. Example of a first-order knowledge base and MLN. Fr() is short for Friends(), Sm() for Smokes(), and Ca() for Cancer().

English	First-Order Logic	Clausal Form	Weight
Friends of friends are friends.	$\forall x \forall y \forall z\ \mathrm{Fr}(x,y) \wedge \mathrm{Fr}(y,z) \Rightarrow \mathrm{Fr}(x,z)$	$\neg\mathrm{Fr}(x,y) \vee \neg\mathrm{Fr}(y,z) \vee \mathrm{Fr}(x,z)$	0.7
Friendless people smoke.	$\forall x\ (\neg(\exists y\ \mathrm{Fr}(x,y)) \Rightarrow \mathrm{Sm}(x))$	$\mathrm{Fr}(x,g(x)) \vee \mathrm{Sm}(x)$	2.3
Smoking causes cancer.	$\forall x\ \mathrm{Sm}(x) \Rightarrow \mathrm{Ca}(x)$	$\neg\mathrm{Sm}(x) \vee \mathrm{Ca}(x)$	1.5
If two people are friends, either both smoke or neither does.	$\forall x \forall y\ \mathrm{Fr}(x,y) \Rightarrow (\mathrm{Sm}(x) \Leftrightarrow \mathrm{Sm}(y))$	$\neg\mathrm{Fr}(x,y) \vee \mathrm{Sm}(x) \vee \neg\mathrm{Sm}(y)$, $\neg\mathrm{Fr}(x,y) \vee \neg\mathrm{Sm}(x) \vee \mathrm{Sm}(y)$	1.1 1.1



Directed vs undirected models

- Undirected models are simpler: no need to worry about cycles, lots of freedom in defining factors
- However, in a UG, the probability of a node depends on the *size* of the graph and/or its connectivity, even if all the other nodes are hidden.
- This may not be desirable.

$X1 \rightarrow X2 \rightarrow X3$

$X1 - X2 - X3$

$X1 \rightarrow X2 \rightarrow \ldots \rightarrow X10$

$X1 - X2 - \cdots - X10$

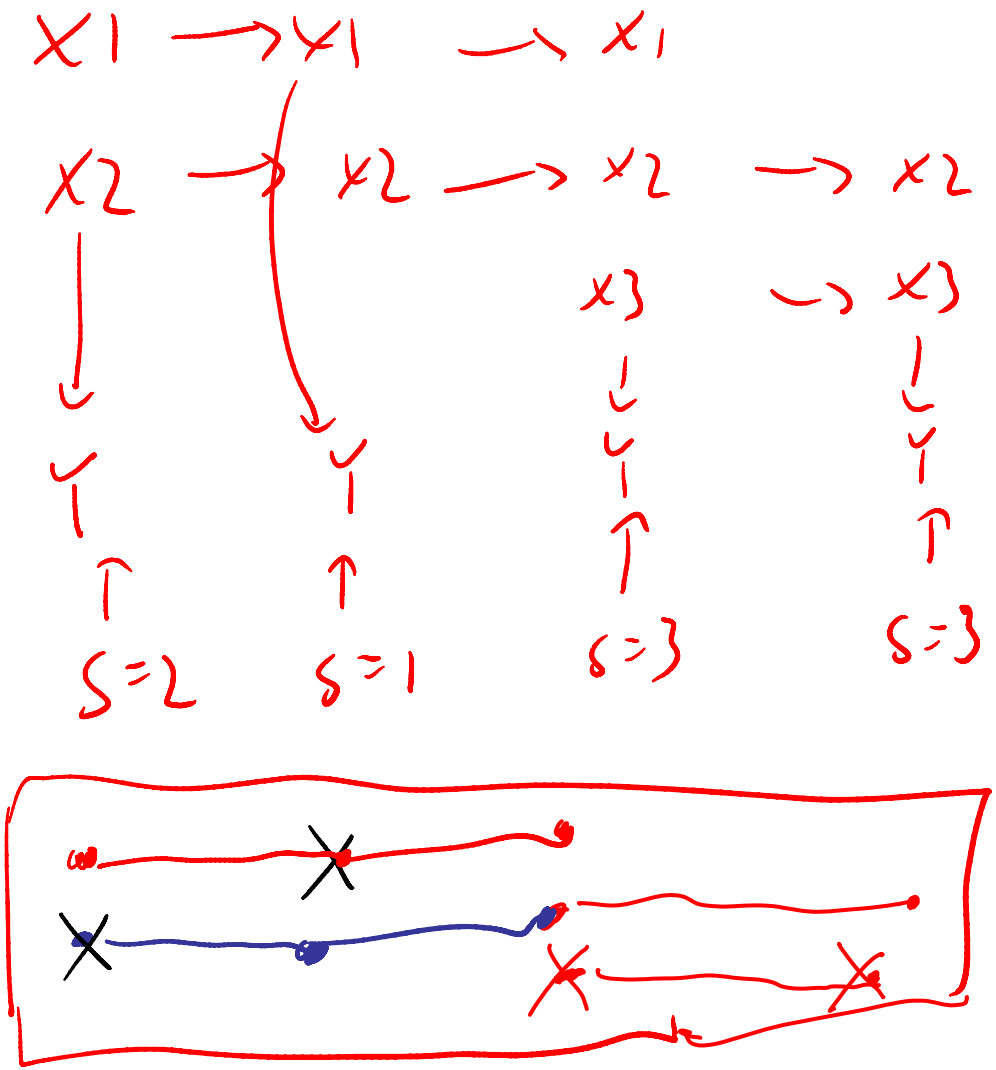
$p(X_2)$ same

$p(X2)$ different

Structural uncertainty

- For a fixed domain, if we do not know the graph structure, we may estimate it using model selection.
- But for relational domains, the structure may change depending on the values of the nodes
- Eg. Genotype(x1) -> genotype(x2) is only active if parent(x1,x2)=true
- In addition, we may be uncertain about how many objects exist in the world
- Eg. In tracking, 3 blips on the radar is consistent with {0,1,…, infty} objects in the world!

Data association ambiguity



Citation matching

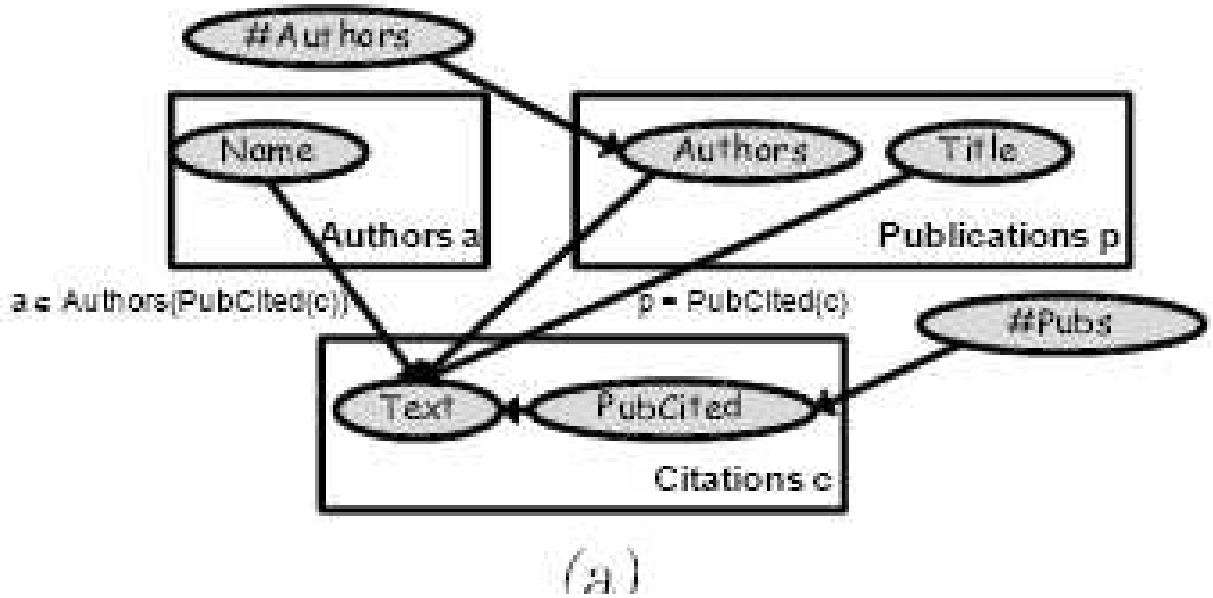
Are these the same article?
Huge industry concerned with database merging

Elston R, Stewart A. A General Model for the Genetic Analysis of Pedigree Data. Hum. Hered. 1971;21:523-542.

Elston RC, Stewart J (1971): A general model for the analysis of pedigree data. Hum Hered 21523-542.

DAG model

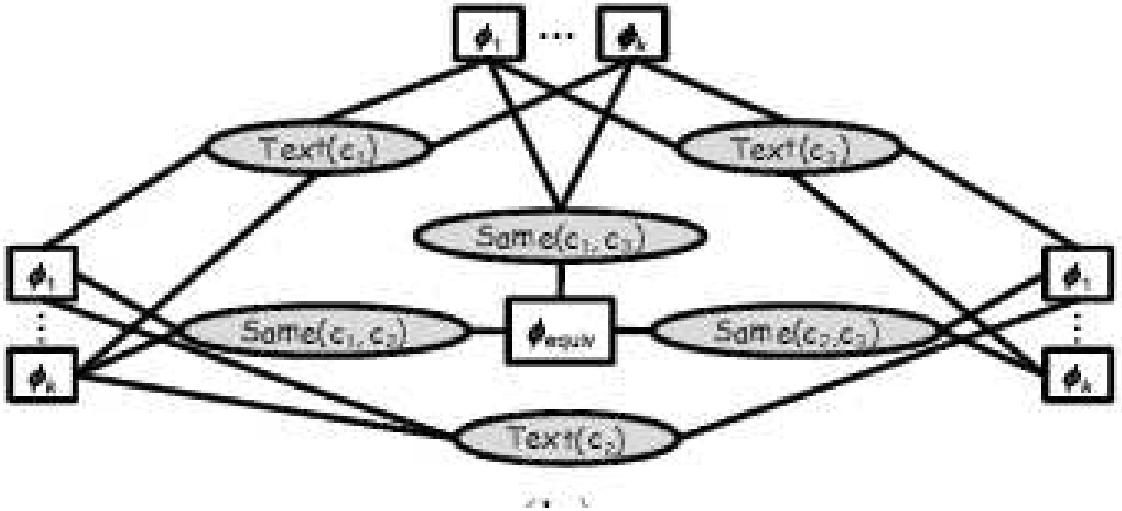
- Assumes there is an unknown number of authors and papers, which generates the observed set of citation strings.



(a)

UG model

- No unknown objects. Just enforce that citations are the same.
- Need 3 way factor to encode transitivity of sameness relation: S(c1,c2), and S(c2,c3) => S(c1,c3)
- And if 2 docs are same, text should be similar: Factor(s(c1,c2), T(c1), T(c2))



MVN: 2 parameterizations

- Moment form

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) \stackrel{\text{def}}{=} \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp[-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})]$$

- Information (canonical) form

$$\begin{aligned}
\boldsymbol{\Lambda} &\stackrel{\text{def}}{=} \boldsymbol{\Sigma}^{-1} \qquad \text{precision (information) matrix} \\
\boldsymbol{\eta} &\stackrel{\text{def}}{=} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\
\mathcal{N}(\mathbf{x}|\boldsymbol{\eta},\boldsymbol{\Lambda}) &= \frac{|\boldsymbol{\Lambda}|^{1/2}}{(2\pi)^{d/2}} \exp[-\tfrac{1}{2}(\mathbf{x}^T\boldsymbol{\Lambda}\mathbf{x} + \boldsymbol{\eta}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{\eta} - 2\mathbf{x}^T\boldsymbol{\eta})] \\
&= \exp[c - \tfrac{1}{2}\mathbf{x}^T\boldsymbol{\Lambda}\mathbf{x} + \mathbf{x}^T\boldsymbol{\eta}]
\end{aligned}$$

Moment and anonical form

- Canonical form is denoted

$$\mathbf{x} \sim \mathcal{N}_C(\mathbf{b}, \mathbf{Q}) \iff p(\mathbf{x}) \propto \exp\left(-\tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{b}^T\mathbf{x}\right)$$

- Moment form

$$\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1}) = \mathcal{N}_C(\mathbf{Q}\boldsymbol{\mu}, \mathbf{Q})$$

Independencies in MVN

- Thm 7.1.3. Let X ~ MVN. $X_i \perp X_j$ iff $\Sigma_{i,j}=0$
- Thm 7.1.4. let X ~ MVN with info matrix J. Then $J_{i,j}=0$ iff $X_i \perp X_j \mid X_{-ij}$
- Factorization thm.

$$\mathbf{x} \perp \mathbf{y} | \mathbf{z} \iff p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}, vz) g(\mathbf{y}, vz)$$

Indep => uncorrelated

- Ex 7.2.1. For any p(X,Y), if X $\perp$ Y then Cov[X,Y]=0.

$$
\begin{aligned}
\text{Cov}[x, y] &= \int\int p(x, y)(x - \overline{x})(y - \overline{y})dxdy \\
&= (\int p(x)(x - \overline{x})dx)(\int p(y)(y - \overline{y})dy) \\
&= (\overline{x} - \overline{x})(\overline{y} - \overline{y}) = 0
\end{aligned}
$$

Uncorrelated & MVN => indep

- Ex 7.2.2. If p(X,Y) is Gaussian, and Cov[X,Y]=0, then $X \perp Y$.
- Pf. The bivariate Gaussian can be written as

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp[-\frac{1}{2(1-\rho^2)}(\frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} - 2\rho\frac{(x_1-\mu_1)}{\sigma_1}\frac{(x_2-\mu_2)}{\sigma_2})]$$

- If \rho=0, then

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2} \exp[-\frac{1}{2}(\frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2})] = f(x_1)g(x_2)$$

- Hence by factorization thm, x1 \perp x2.

Uncorrelated not imply independent

- Ex 7.2.3. Find an example where Cov[X,Y]=0 yet not $X \perp Y$.
- Let X ~ U(-1,1) and Y=X^2. Clearly Y is dependent on X yet one can show (exercise) that Cov(X,Y)=0.
- Let X ~ N(0,1) and Y= W X, p(W=-1)=p(W=1)=0.5. Clearly Y is dependent on X, yet one can show (exercise) that Y ~ N(0,1) and Cov[X,Y]=0.

Independencies in MVN

- Thm 7.1.3. Let X ~ MVN. $X_i \perp X_j$ iff $\Sigma_{i,j}=0$
- Pf. By ex 7.2.1, we have => direction.
- By ex 7.2.2, we have that <= direction.
- By ex 7.2.3, we have that X ~ MVN is necessary for <= direction to work.

Conditional Independencies in MVN

- Thm 7.1.4. let X ~ MVN with info matrix J. Then $J_{i,j}=0$ iff $X_i \perp X_j \mid X_{-ij}$
- Pf. Let mu=0.

$$p(x_i, x_j, \mathbf{x}_{-ij}) \propto \exp(-\frac{1}{2}\sum_{k,l} x_k Q_{kl} x_l)$$

$$\propto \exp\left(-\frac{1}{2}x_i x_j (Q_{ij}+Q_{ji}) - \frac{1}{2}\sum_{\{k,l\}\neq\{i,j\}} x_k Q_{kl} x_l\right)$$

- The second term does not involve x_i x_j, and nor does the first iff $Q_{ij}=0$. Hence this factorizes into $f(x_i,x_{-ij})$ $g(x_j,x_{-ij})$ iff $Q_{ij}=0$. QED.

Structural zeros

- Zeros in the precision matrix correspond to missing edges in the UGM

$$\boldsymbol{\Sigma} = \begin{pmatrix} 4 & 2 & -2 \\ 2 & 5 & -5 \\ -2 & -5 & 8 \end{pmatrix}, \quad \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} 0.3125 & -0.125 & 0 \\ -0.125 & 0.5833 & 0.3333 \\ 0 & 0.3333 & 0.3333 \end{pmatrix}$$

$X_1 - X_2 - X_3$

Marginals and conditionals

	Marginal $p(\mathbf{x}_2)$
Moment	$\mathcal{N}(\mathbf{x}_2 \vert \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$
Info	$\mathcal{N}(\mathbf{x}_2 \vert \boldsymbol{\eta}_2 - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\eta}_1, \boldsymbol{\Lambda}_{22} - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12})$

	Conditional $p(\mathbf{x}_2 \vert \mathbf{x}_1)$
Moment	$\mathcal{N}(\mathbf{x}_1 \vert \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21})$
Info	$\mathcal{N}(\mathbf{x}_2 \vert \boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\mathbf{x}_2, \boldsymbol{\Lambda}_{11})$

Marginalization easy in moment form.
Conditioning easy in canonical form.

Conditioning in canonical form

- Thm (Conditioning).

$$\mathbf{x} \sim \mathcal{N}_C(\mathbf{b}, \mathbf{Q}) \Rightarrow \quad \mathbf{x}_A|\mathbf{x}_B \sim \mathcal{N}_C(\mathbf{b}_A - \mathbf{Q}_{AB}\mathbf{x}_B, \mathbf{Q}_{AA})$$

- Thm (soft conditioning) .

$$\mathbf{x} \sim \mathcal{N}_C(\mathbf{b}, \mathbf{Q}) \quad \text{and} \quad \mathbf{y}|\mathbf{x} \sim \mathcal{N}(\mathbf{x}, \mathbf{P}^{-1})$$

$$\mathbf{x}|\mathbf{y} \sim \mathcal{N}_C(\mathbf{b} + \mathbf{P}\mathbf{y}, \mathbf{Q} + \mathbf{P})$$

Precisions add

- We can accumulate evidence by addition of matrix-vector products, and then compute posterior mean at end by solving Qb = mu.

Partial correlation coefficient

- Let X ~ Mvn with precision matrix

$$\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \omega_{11} & \dots & \omega_{1d} \\ \vdots & \ddots & \ddots \\ \omega_{d1} & \dots & \omega_{dd} \end{pmatrix}$$

- The conditional distribution p(x1,x2|x3,…,xd) is bivariate Gaussian with covariance

$$\begin{pmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \end{pmatrix}^{-1} = \frac{1}{\omega_{11}\omega_{22} - (\omega_{12})^2} \begin{pmatrix} \omega_{22} & -\omega_{12} \\ -\omega_{21} & \omega_{11} \end{pmatrix}$$

- The partial correlation coefficient is given by

$$\rho_{1,2|3,\dots,d} \stackrel{\text{def}}{=} \frac{Cov[X_1, X_2|X_{3:d}]}{\sqrt{\text{Var}\left[X_1|X_{3:d}\right]\text{Var}\left[X_2|X_{3:d}\right]}} = \frac{-\omega_{21}}{\sqrt{\omega_{11}\omega_{22}}}$$

Conditioning in moment form

- Thm (Rue&Held p26).

$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1}) \Rightarrow \\ \mathbf{x}_A|\mathbf{x}_B &\sim \mathcal{N}(\boldsymbol{\mu}_{A|B}, \mathbf{Q}_{AA}^{-1}) \\ \boldsymbol{\mu}_{A|B} &= \boldsymbol{\mu}_A - \mathbf{Q}_{AA}^{-1}\mathbf{Q}_{AB}(\mathbf{x}_B - \boldsymbol{\mu}_B) \end{aligned}$$

- Thus to find the mean we need to solve the linear system

$$\mathbf{Q}_{AA}\boldsymbol{\mu}_{A|B} = \mathbf{Q}_{AA}\boldsymbol{\mu}_A - \mathbf{Q}_{AB}\mathbf{x}_B + \mathbf{Q}_{AB}\boldsymbol{\mu}_B$$

- Eg if A={i} we have

$$\begin{aligned} E[x_i|\mathbf{x}_{-i}] &= \mu_i - \frac{1}{Q_{ii}} \sum_{j:j\neq i} Q_{ij}(x_j - \mu_j) \\ \text{prec}(x_i|\mathbf{x}_{-i}) &= Q_{ii} \end{aligned}$$

Proof

- Assume \mu=0 for simplicity. Then

$$
\begin{aligned}
p(\mathbf{x}_A|\mathbf{x}_B) &\propto \exp\left(-\tfrac{1}{2}\begin{pmatrix}\mathbf{x}_A & \mathbf{x}_B\end{pmatrix}\begin{pmatrix}\mathbf{Q}_{AA} & \mathbf{Q}_{AB}\\ \mathbf{Q}_{BA} & \mathbf{Q}_{BB}\end{pmatrix}\begin{pmatrix}\mathbf{x}_A\\ \mathbf{x}_B\end{pmatrix}\right)\\
&\propto \exp\left(-\tfrac{1}{2}\mathbf{x}_A^T\mathbf{Q}_{AA}\mathbf{x}_A - (\mathbf{Q}_{AB}\mathbf{x}_B)^T\mathbf{x}_A\right)
\end{aligned}
$$

- Compare this to a Gaussian with precision K and mean m

$$
p(\mathbf{z}) \propto \exp\left(-\tfrac{1}{2}\mathbf{z}^T\mathbf{K}\mathbf{z} + (\mathbf{K}\mathbf{m})^T\mathbf{z}\right)
$$

- We see that Q_{AA} is the conditional precision and the conditional mean is given by

$$
\mathbf{Q}_{AA}\boldsymbol{\mu}_{A|B} = -\mathbf{Q}_{AB}\mathbf{x}_B
$$

QED

Soft conditioning in moment form

$$
\begin{aligned}
\mathbf{x} &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
\mathbf{y}|\mathbf{x} &\sim \mathcal{N}(\mathbf{x}, \mathbf{S}) \\
\mathbf{x}|\mathbf{y} &\sim \mathcal{N}(\boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y}) \\
\boldsymbol{\Sigma}_{x|y}^{-1} &= \boldsymbol{\Sigma}^{-1} + \mathbf{S}^{-1} \\
\boldsymbol{\Sigma}_{x|y}^{-1}\boldsymbol{\mu}_{x|y} &= \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \mathbf{S}^{-1}\mathbf{y}
\end{aligned}
$$

Bayes rule for linear Gaussian systems

Linear Gaussian DGMs

- A CPD is linear Gaussian if

$$p(x_i|x_{\pi_i}) = \mathcal{N}(x_i|\sum_{j\in\pi_i} w_{ij}x_j + b_i, v_i)$$

- A DGM is linear Gaussian if all CPDs are LG.
- Such networks define a joint Gaussian. Each node is given by

$$x_i = \sum_{j\in\pi_i} w_{ij}x_j + b_i + \sqrt{v_i}\epsilon_i$$

  where $\varepsilon_i \sim N(0,1)$ and $E[\varepsilon_i\, \varepsilon_j] = I_{i,j}$.

- W is lower triangular matrix: w_{i,j} = weights into i from j.

LG DGM to MVN

- We can compute the global mean and covariance recursively, in topological order

$$x_i = \sum_{j \in \pi_i} w_{ij} x_j + b_i + \sqrt{v_i}\epsilon_i$$

$$E[x_i] = \sum_{j \in \pi_i} w_{ij} E[x_j] + b_i$$

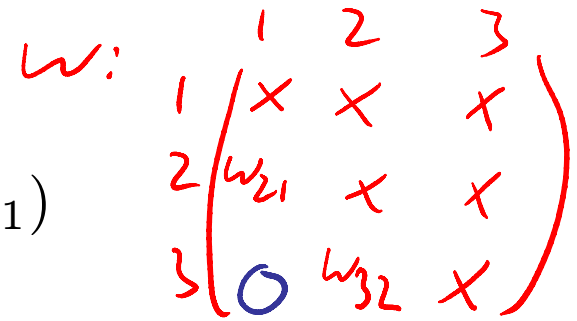
$$\begin{aligned}
\text{Cov}[x_i, x_j] &= E[(x_i - E[x_i])(x_j - E[x_j])] \\
&= E\left[(x_i - E[x_i])\left\{\sum_{k \in \pi_j} w_{jk}(x_k - E[x_k]) + \sqrt{v_j}\epsilon_j\right\}\right] \\
&= \sum_{k \in \pi_j} w_{jk}\text{Cov}[x_i, x_k] + I_{i,j} v_j
\end{aligned}$$

Bishop p371

LG DGM to MVN

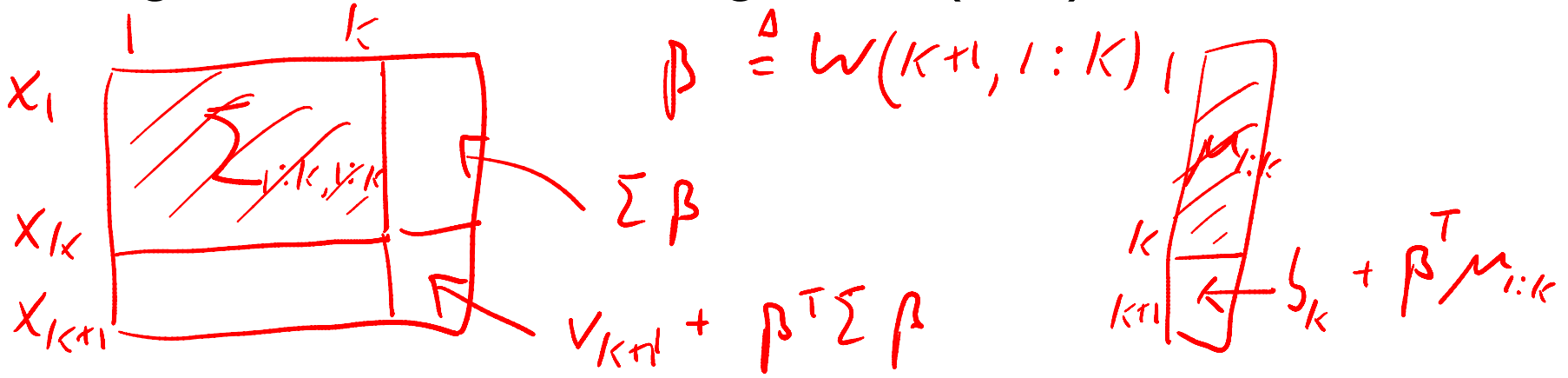
- Consider a chain x1 -> x2 -> x3

$$\boldsymbol{\mu} = (b_1, b_2 + w_{21} b_1, b_3 + w_{32} b_2 + w_{32} w_{21} b_1)$$



$$\boldsymbol{\Sigma} = \begin{pmatrix} v_1 & w_{21} v_1 & w_{32} w_{31} v_1 \\ w_{21} v_1 & v_2 + w_{21}^2 v_1 & w_{32}(v_2 + w_{21}^2 v_1) \\ w_{32} w_{21} v_1 & w_{32}(v_2 + w_{21}^2 v_1) & v_3 + w_{32}^2 (v_2 + w_{21}^w v_1) \end{pmatrix}$$

- In general, when adding node (k+1)



K&F Thm 7.2.2

Alternative parameterization

- The results are much “prettier” if we write

$$X_j = \mu_j + \sum_{k \in \pi_j} w_{jk}(X_k - \mu_k) + \sqrt{v_j} Z_j$$

where the offset is given by

$$w_j^{(0)} = \mu_j - \sum_{k \in \pi_j} w_{jk} \mu_k$$

- Then we have

$$\begin{aligned} (\mathbf{x} - \boldsymbol{\mu}) &= \mathbf{W}(\mathbf{x} - \boldsymbol{\mu}) + \mathbf{S}^T \mathbf{z} = \mathbf{W}(\mathbf{x} - \boldsymbol{\mu}) + \mathbf{e} \\ \mathbf{e} &= \mathbf{S}^T \mathbf{z} = (\mathbf{I} - \mathbf{W})(\mathbf{x} - \boldsymbol{\mu}) \end{aligned}$$

$$\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_d \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ -w_{21} & 1 & & & \\ -w_{32} & -w_{31} & 1 & & \\ \vdots & & & \ddots & \\ -w_{d1} & -w_{d2} & \dots & -w_{d,d-1} & 1 \end{pmatrix} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \\ \vdots \\ x_d - \mu_d \end{pmatrix}$$

DAG weights = Cholesky Decomposition

$$
\begin{aligned}
\mathbf{x}-\boldsymbol{\mu} &= (\mathbf{I}-\mathbf{W})^{-1}\mathbf{e} \stackrel{\text{def}}{=} \mathbf{U}\mathbf{e} = \mathbf{U}\mathbf{S}^T\mathbf{z} \stackrel{\text{def}}{=} \mathbf{A}^T\mathbf{z} \\
\boldsymbol{\Sigma} &= \mathsf{Var}\left[\mathbf{x}\right] = \mathsf{Var}\left[\mathbf{x}-\boldsymbol{\mu}\right] \\
&= \mathsf{Var}\left[\mathbf{A}^T\mathbf{z}\right] = \mathbf{A}^T\mathsf{Var}\left[\mathbf{z}\right]\mathbf{A} = \mathbf{A}^T\mathbf{A} \\
&= \mathbf{U}\mathbf{S}^T\mathbf{S}\mathbf{U}^T = \mathbf{U}\mathbf{D}\mathbf{U}^T \\
\boldsymbol{\Sigma}^{-1} &= \mathbf{U}^{-T}\mathbf{D}^{-1}\mathbf{U}^{-1} = (\mathbf{I}-\mathbf{W})^T\mathbf{D}^{-1}(\mathbf{I}-\mathbf{W}) \stackrel{\text{def}}{=} \mathbf{T}^T\mathbf{D}^{-1}\mathbf{T}
\end{aligned}
$$

$$
\mathbf{T} = \begin{pmatrix}
1 & & & & \\
-w_{21} & 1 & & & \\
-w_{32} & -w_{31} & 1 & & \\
\vdots & & & \ddots & \\
-w_{d1} & -w_{d2} & \dots & -w_{d,d-1} & 1
\end{pmatrix}
$$

Chains

- Consider a chain X1 -> X2 -> … -> X5
- The DAG and UG are both sparse (same CI)

```
n = 5;
w=randn(n,1);
W = spdiags([w zeros(n,1) zeros(n,1)], -1:1, n, n);
T = eye(n)-W;
D = diag(ones(n,1));
K = T'*D*T;

>> full(W)
ans =
         0         0         0         0         0
    1.1909         0         0         0         0
         0    1.1892         0         0         0
         0         0   -0.0376         0         0
         0         0         0    0.3273         0
>> K
K =
    2.4183   -1.1909         0         0         0
   -1.1909    2.4141   -1.1892         0         0
         0   -1.1892    1.0014    0.0376         0
         0         0    0.0376    1.1071   -0.3273
         0         0         0   -0.3273    1.0000
```

Diamond

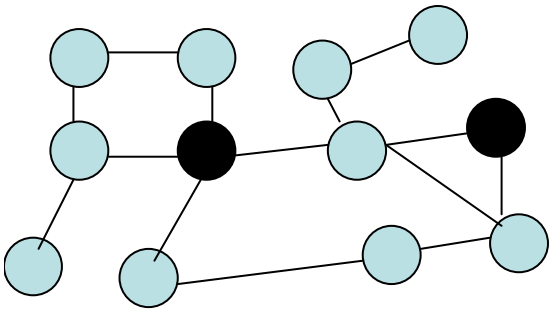
- DAG is sparse, Sigma and SigmaInv are dense


```
W =
         0         0         0         0
    0.5488         0         0         0
    0.7152         0         0         0
         0    0.6028    0.5449         0
>> K
K =
    1.8127   -0.5488   -0.7152         0
   -0.5488    1.3633    0.3284   -0.6028
   -0.7152    0.3284    1.2969   -0.5449
         0   -0.6028   -0.5449    1.0000
>> inv(K)
ans =
    1.0000    0.5488    0.7152    0.7205
    0.5488    1.3012    0.3925    0.9982
    0.7152    0.3925    1.5115    1.0602
    0.7205    0.9982    1.0602    2.1793
```

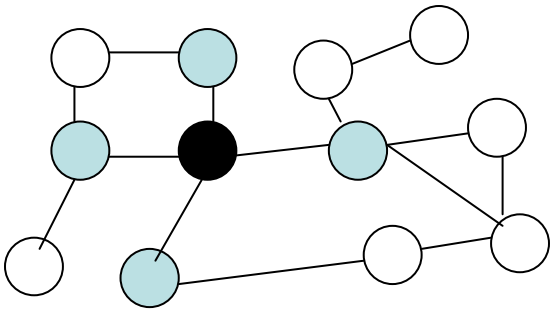
Gaussian MRFs

- Defn. A GMRF is a Gaussian of the form $N(\mu, Q^{-1})$ where $Q_{ij} \neq 0$ iff $G_{ij} \neq 0$ (Q=precision matrix)
- Thm. For a GMRF, the following properties are equivalent.
- Pairwise Markov: $x_i \perp x_j | \mathbf{x}_{-ij}$ if $G_{i,j} = 0$ and $i \neq j$
- Local Markov: $x_i \perp \mathbf{x}_{-i,ne(i)} | \mathbf{x}_{ne(i)}$
- Global Markov: $x_A \perp x_B | x_C$

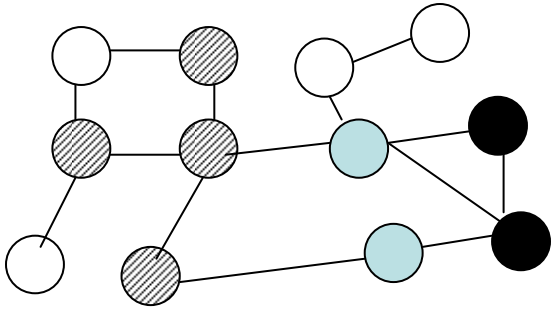
Rue&Held p25



Blacks indep given gray



Black indep of white given gray



Black indep striped given gray

MVN to Gaussian UGM

- We can convert any MVN into a UGM with pairwise potentials which are quadratics

$$
\begin{aligned}
\mathbf{J} &\stackrel{\text{def}}{=} \boldsymbol{\Sigma}^{-1} \\
\mathbf{h} &\stackrel{\text{def}}{=} \mathbf{J}\boldsymbol{\mu} \\
\mathcal{N}(\mathbf{x}|\mathbf{h},\mathbf{J}) &= \exp[c - \tfrac{1}{2}\mathbf{x}^T\mathbf{J}\mathbf{x} + \mathbf{x}^T\mathbf{h}] \\
\log p(\mathbf{x}) &= c - \tfrac{1}{2}\sum_i [J_{i,i}x_i^2 + h_i x_i] - \tfrac{1}{2}\sum_i \sum_j J_{i,j} x_i x_j \\
&= c + \sum_i \phi_i(x_i) + \sum_i \sum_{j>i} \phi_{i,j}(x_i, x_j) \\
\phi_i(x_i) &= -\tfrac{1}{2}[J_{i,i}x_i^2 + h_i x_i] \\
\phi_{i,j}(x_i, x_j) &= -J_{i,j} x_i x_j
\end{aligned}
$$

Pairwise UGM to MVN

- Consider a UGM in which the node and edge potentials are quadratics

$$\begin{aligned} \epsilon_i(x_i) &= d_0^i + d_1^i x_1 + d_2^i x_i^2 \\ \epsilon_{ij}(x_i, x_j) &= a_{00}^{i,j} + a_{01}^{i,j} x_i + a_{10}^{ij} x_j + a_{11}^{ij} x_i x_j + a_{02}^{ij} x_i^2 + a_{20}^{ij} x_j^2 \end{aligned}$$

- We can always rewrite the corresponding unnormalized distribution as

$$p'(\mathbf{x}) = \exp[-\tfrac{1}{2}\mathbf{x}^T \mathbf{J}\mathbf{x} + \mathbf{x}^T \mathbf{h}]$$

- But the normalization constant Z will only be finite if J is positive definite.

Sufficient conditions on info matrix

- Def 7.3.1. A matrix J is attractive if, for all i \neq j, we have that all partial correlations are non-neg

$$-\frac{J_{i,j}}{\sqrt{J_{i,i}J_{j,j}}} \geq 0$$

- Thm 7.3.2. If J is attractive, then p is a valid MVN.
- Def 7.3.1b. A matrix J is diagonally dominant if, for all rows i, $J_{ii} > \sum_{j \neq i} |J_{i,j}|$
- Thm 7.3.2b. If J is diagonally dominant, then p is a valid MVN.

Pairwise normalizable

- Def 7.3.3. A pairwise MRF with energies of the form

$$\begin{aligned} \epsilon_i(x_i) &= d_0^i + d_1^i x_1 + d_2^i x_i^2 \\ \epsilon_{ij}(x_i, x_j) &= a_{00}^{i,j} + a_{01}^{i,j} x_i + a_{10}^{ij} x_j + a_{11}^{ij} x_i x_j + a_{02}^{ij} x_i^2 + a_{20}^{ij} x_j^2 \end{aligned}$$

  is called pairwise normalizable if

$d_2^i > 0, \forall i$ and $\begin{pmatrix} a_{02}^{ij} & a_{11}^{ij}/2 \\ a_{11}^{ij}/2 & a_{20}^{ij} \end{pmatrix}$ is psd for all i,j

- Thm 7.3.4. If the MRF is pairwise normalizable, then it defines a valid Gaussian.
- Sufficient but not necessary eg.

$$\begin{pmatrix} 1 & 0.6 & 0.6 \\ 0.6 & 1 & 0.6 \\ 0.6 & 0.6 & 1 \end{pmatrix}$$

May be able to reparameterize the node/ edge potentials to ensure pairwise normalized.

Conditional autoregressions (CAR)

- We can parameterize a GMRF in terms of its full conditionals

$$E[x_i|\mathbf{x}_{-i}] = \mu_i - \sum_{j:j\sim i} \beta_{ij}(x_j - \mu_j)$$

$$\text{prec}[x_i|\mathbf{x}_{-i}] = \kappa_i > 0$$

- From before, we have

$$E[x_i|\mathbf{x}_{-i}] = \mu_i - \frac{1}{Q_{ii}} \sum_{j:j\neq i} Q_{ij}(x_j - \mu_j)$$

$$\text{prec}(x_i|\mathbf{x}_{-i}) = Q_{ii}$$

- To be a valid MVN we must set

$$\kappa_i = Q_{ii}, \beta_{ij} = \frac{Q_{ij}}{\kappa_i}, \kappa_i\beta_{ij} = \kappa_j\beta_{ji}$$

$$\mathbf{Q} = \text{diag}(\boldsymbol{\kappa})(\mathbf{I} + \boldsymbol{\beta})$$

Rue&Held p29

Stat 521A
Lecture 6

Outline

- Exponential family: what?(8.2)
- Why? (Extra)
- Connection with GMs (8.3)
- Entropy (8.4)
- Projections (8.5)
- Querying a distribution (“inference”) – 2.1.5
- Worst case complexity of exact inference (9.1)

Exponential family

- Def 8.2.2. The exponential family is a set of distributions of the form

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} h(\mathbf{x}) \exp\left(\mathbf{t}(\boldsymbol{\theta})^T \mathbf{T}(\mathbf{x})\right)$$

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{x} \in S} h(\mathbf{x}) \exp\left(\mathbf{t}(\boldsymbol{\theta})^T \mathbf{T}(\mathbf{x})\right)$$

Where $x \in X$ are the variables, h(x) defines the support (must not depend on θ), $T(x) \in R^K$ are the sufficient statistics, $\theta \in \Theta \subseteq R^M$ are the parameters, $t(\theta)$ in R^K are the natural parameters, and $Z(\theta) \in R^+$ is the partition function.

We would like Θ to be a convex open subset of R^M, and to be non-redundant (iff $t(\theta)$ is invertible).

Examples

- X ~ Ber(θ).

$$\begin{aligned} \mathbf{T}(x) &= [I(x=0), I(x=1)] \\ \mathbf{t}(\boldsymbol{\theta}) &= [\log\theta, \log(1-\theta)] \\ Z(\theta) &= 1 \\ p(x) &= \exp\left(\mathbf{T}(x)^T \mathbf{t}(\boldsymbol{\theta})\right) \end{aligned} \qquad \Theta = [0,1], \mathcal{X} = \{0,1\}$$

- X ~ N(μ,σ²).

$$\begin{aligned} p(x) &= \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2) \\ \mathbf{T}(x) &= [x, x^2] \\ \mathbf{t}(\mu,\sigma^2) &= [\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}] \\ Z(\mu,\sigma^2) &= \sqrt{2\pi}\sigma \exp(\frac{\mu^2}{2\sigma^2}) \end{aligned} \qquad \Theta = \mathbb{R}\times\mathbb{R}^+, \mathcal{X} = \mathbb{R}$$

Non-examples

- Let X ~ Unif(a,b). Then

$$p(x|\boldsymbol{\theta}) \quad = \quad \frac{1}{b-a} I(a \leq x \leq b) = \exp(\log \frac{1}{b-a})) I(a \leq x \leq b)$$

- Support depends on \theta.
- Let X ~ $\sum_k \pi_k$ f(x,ϕ_k) – mixture model. Cannot be written in required form.

Linear exponential family

- Consider the set

$$\Theta = \{\boldsymbol{\theta} \in \mathbb{R}^K : \int \exp(\boldsymbol{\theta}^T \mathbf{T}(\mathbf{x}))d\mathbf{x} < \infty\}$$

- If Θ is open and convex, and t(θ)=θ, we say it is a linear exponential family.
- We write

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\eta}) &= \frac{1}{Z(\boldsymbol{\eta})} h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(\mathbf{x})] \\ Z(\boldsymbol{\eta}) &= \int h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(\mathbf{x})] d\mathbf{x} \end{aligned}$$

- Or

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\eta}) &= h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(\mathbf{x}) - A(\boldsymbol{\eta})] \\ A(\boldsymbol{\eta}) &= \log Z(\boldsymbol{\eta}) \end{aligned}$$

Bernoulli try 1

- .

$$\begin{aligned} \mathbf{T}(x) &= [I(x=0), I(x=1)] \\ \boldsymbol{\eta} &= [\log\theta, \log(1-\theta)] \\ p(x) &= \exp\left(\boldsymbol{\eta}^T \mathbf{T}(x)\right) \end{aligned}$$

- However, (log \theta, log (1-\theta)) is a curve, not a convex subset. Also, it is redundant.

Bernoulli try 2

- Define

$$
\begin{aligned}
T(x) &= [I(x=1)] \\
\eta &= \log \frac{\theta}{1-\theta} \qquad \Theta = \mathbb{R} \\
Z(\eta) &= 1 + \frac{\theta}{1-\theta} = \frac{1}{1-\theta} \\
p(x) &= \frac{1}{Z(\eta)} \exp(\eta T(x)) = (1-\theta) \exp(x \log \frac{\theta}{1-\theta}) \\
p(x=0) &= (1-\theta) \\
p(x=1) &= (1-\theta)\frac{\theta}{1-\theta} = \theta
\end{aligned}
$$

Gaussian – natural params

$$\begin{aligned} \boldsymbol{\eta} &= [\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}] \\ \mathbf{T}(x) &= [x, x^2] \end{aligned}$$

The natural parameter space is $\mathbb{R} \times \mathbb{R}^-$

Finite sufficient statistics

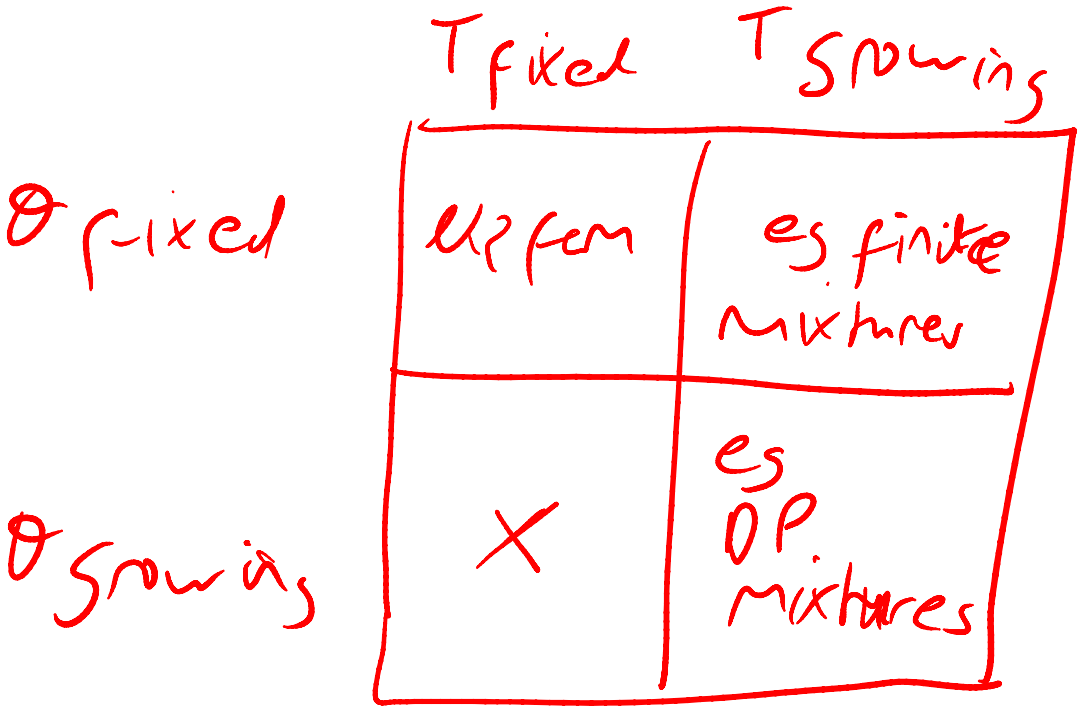
- Defn. A statistic is a function of the data, T(D), where D=(x1,…,xn). A sufficient statistic is one that contains all the information in the data. More formally, T is sufficient for θ if θ -> T(D) -> D.
- Let Xi ~ ExpFam. The likelihood is given by

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})^n}[\prod_i h(\mathbf{x}_i)] \exp(\mathbf{t}(\boldsymbol{\theta})^T \sum_{i=1}^{n} \mathbf{T}(\mathbf{x}_i))$$

- Hence the distribution has sufficient statistics of size K, independent of n $\mathbf{T}(D) = \sum_{i=1}^{n} \mathbf{T}(\mathbf{x}_i))$
- Thm (**Pitman-Koopman-Darmois).** The expfam is the only family (amongst those where support is indep of theta) with fixed sized suff stat.

Non-parametric models

- Parametric = fixed sized theta
- Exp fam = fixed size suff stat



LogZ is MGF

- Consider a linear expfam

$$p(\mathbf{x}|\boldsymbol{\eta}) = \frac{1}{Z(\boldsymbol{\eta})} h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(\mathbf{x})]$$

- Define

$$\frac{1}{g(\boldsymbol{\eta})} \stackrel{\text{def}}{=} Z(\boldsymbol{\eta}) = \int h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(\mathbf{x})] d\mathbf{x}$$

- Then

$$\begin{aligned} 1 &= g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(x)] d\mathbf{x} \\ 0 &= \nabla g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(x)] d\mathbf{x} \\ &\quad + g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(\mathbf{x})] \mathbf{T}(\mathbf{x}) d\mathbf{x} \\ \int p(\mathbf{x}|\boldsymbol{\eta}) \mathbf{T}(\mathbf{x}) d\mathbf{x} &= -\nabla g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(x)] d\mathbf{x} \end{aligned}$$

LogZ is MGF

$$\begin{aligned}
\int p(\mathbf{x}|\boldsymbol{\eta})\mathbf{T}(\mathbf{x})d\mathbf{x} &= -\nabla g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(x)]d\mathbf{x} \\
-\nabla \log g(\boldsymbol{\eta}) &= -\frac{\nabla g(\boldsymbol{\eta})}{g(\boldsymbol{\eta})} = -(\nabla g(\boldsymbol{\eta}))(\int h(\mathbf{x}) \exp[\boldsymbol{\eta}^T \mathbf{T}(\mathbf{x})]d\mathbf{x}) \\
E[\mathbf{T}(\mathbf{X})] &= -\nabla \log g(\boldsymbol{\eta}) = \nabla \log Z(\boldsymbol{\eta})
\end{aligned}$$

MLE is moment matching

- Proof

$$\begin{aligned}
\log p(\mathcal{D}|\boldsymbol{\theta}) &= -n \log Z(\boldsymbol{\theta}) + \boldsymbol{\theta}^T \mathbf{T}(\mathcal{D}) \\
\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) &= -n \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta}) + \mathbf{T}(\mathcal{D}) = \mathbf{0} \\
E\mathbf{T}(\mathbf{X}) &= \frac{1}{n} \mathbf{T}(\mathcal{D})
\end{aligned}$$

- Example. Gaussian, T(X) = (X, X^2).

$$\begin{aligned}
E[X] &= \mu = \frac{1}{n} \sum_i x_i \\
\text{Var}\,[X] &= (EX^2) - (EX)^2 \\
E[X^2] &= \sigma^2 + \mu^2 = \frac{1}{n} \sum_i x_i^2 \\
\sigma^2 &= \frac{1}{n} \sum_i x_i^2 - \mu^2
\end{aligned}$$

Conjugate priors

- Defn. A prior $p(\theta) \in F$ is conjugate to a likelihood $p(D|\theta)$ if the posterior satistifes $p(\theta|D) \in F$, i.e., has the same functional form as the prior.
- Thm. All dist in expfam have conj prior.
- Most distrib with conj prior are in exp fam.

Maximum entropy principle

- Defn. The entropy of a pmf is

$$H(p) \stackrel{\text{def}}{=} -\sum_x p(x) \log p(x), H(p) \geq 0$$

- The differential entropy of a pdf can be –ve

$$h(p) \stackrel{\text{def}}{=} -\int_S p(x) \log p(x) dx$$

- The relative entropy, or KL divergence, from p to q is given by

$$KL(p,q) \stackrel{\text{def}}{=} \sum_x p(x) \log \frac{p(x)}{q(x)}$$

- KL is always >= 0, even for pdf's.

Maxent principle

- Suppose we want to pick the most uncertain distribution (principle of least commitment) subject to the constraints that

$$\sum_x f_k(x)p(x) = F_k$$

- Optimize the Lagrangian

$$\begin{aligned}
J(p) &= -\sum_x p(x)\log p(x) + \lambda_0(1 - \sum_x p(x)) + \sum_k \lambda_k(F_k - \sum_x p(x) f_k(x)) \\
\frac{\partial J}{\partial p(x)} &= -1 - \log p(x) - \lambda_0 - \sum_k \lambda_k f_k(x) = 0 \\
p(x) &= \frac{1}{Z}\exp(-\sum_k \lambda_k f_k(x)) \\
Z &= e^{1+\lambda_0} \\
1 &= \sum_x p(x) = \frac{1}{Z}\sum_x \exp(-\sum_k \lambda_k f_k(x)) \\
Z &= Z(\boldsymbol{\lambda}) = \sum_x \exp(-\sum_k \lambda_k f_k(x))
\end{aligned}$$

Gaussian maximizes entropy

- MVN is in expfam.

$$p(\mathbf{x}) = \frac{1}{Z}\exp(-\tfrac{1}{2}\mathbf{x}^T\mathbf{K}\mathbf{x}) = \frac{1}{Z}\exp(\sum_k \lambda_k f_k(\mathbf{x}))$$

$$f_{ij}(\mathbf{x}) = x_i x_j,\ \lambda_{ij} = \tfrac{1}{2}K_{ij}$$

Theorem 0.1. *Let $g(\mathbf{x})$ be any density satisfying $\int g(\mathbf{x})x_i x_j = \Sigma_{ij}$. Let $\phi = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. Then $h(g) \leq h(\phi)$.*

Proof. (From (**?**, p234).) We have

$$\begin{aligned}
0 &\leq KL(g||\phi) && (1)\\
&= \int g(\mathbf{x}) \log \frac{g(\mathbf{x})}{\phi(\mathbf{x})} d\mathbf{x} && (2)\\
&= -h(g) - \int g(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} && (3)\\
&= -h(g) - \int \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} \text{ (**)} && (4)\\
&= -h(g) + h(\phi) && (5)
\end{aligned}$$

where the line marked (**) follows since g and ϕ yield the same moments for the quadratic form $\log \phi(\mathbf{x})$. ∎

Some GMs are expfam models

- We showed earlier that many +ve UGM can be represented as an expfam

$$p(\mathbf{x}) = \frac{1}{Z}\exp(\sum_i \boldsymbol{\theta}_i^T f_i(\mathbf{x}))$$

- Most CPDs can be represented as expfam
- Eg table p(X|U). T(X,U)=[I(X=x), I(U=u)], t(\theta) = [\log p(x|u)].
- Eg lingauss.

$$p(x|\mathbf{u}) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-(w_0+w_1u_1+\cdots+w_ku_k))^2\right)$$
$$\mathbf{T}(x,\mathbf{u}) = [1, x, u_1, \ldots, u_k, xu_1, \ldots, xu_k, u_1^2, u_1u_2, \ldots, u_k^2]$$

- Product of expfam is expfam.

DGMs are curved expfam

- In general, the fact that CPDs sum to 1 locally means that they are not linear expfam
- See p248 of K&F
- Geiger'01 shows that DGMs are curved expfam models (curved means the params are not linearly indep, so \theta is smaller than t(\theta)).
- Geiger'01 also shows that GMs with hidden variables are stratified exponential families (SEFs) - a finite union of CEFs of various dimensions satisfying some regularity conditions.

Stratified exponential families: Graphical models and model selection
Dan Geiger, David Heckerman, Henry King, and Christopher Meek
Source: Ann. Statist. Volume 29, Number 2 (2001), 505-529.

Entropy of an expfam model

- Thm 8.4.1. If X ~ ExpFam(theta), then

$$H(P_{\boldsymbol{\theta}}(\mathbf{x})) = \log Z(\boldsymbol{\theta}) - E[\mathbf{T}(\mathbf{x})^T \mathbf{t}(\boldsymbol{\theta})]$$

- Ex 8.4.2. Gaussian.

$$\begin{aligned}
p(x) &= \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2) \\
\mathbf{T}(x) &= [x, x^2] \\
\mathbf{t}(\mu, \sigma^2) &= [\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}] \\
Z(\mu, \sigma^2) &= \sqrt{2\pi}\sigma \exp(\frac{\mu^2}{2\sigma^2}) \\
H &= \tfrac{1}{2}\ln(2\pi\sigma^2) + \frac{\mu^2}{2\sigma^2} - \frac{\mu}{\sigma^2}E[x] + \frac{1}{2\sigma^2}E[x^2] \\
&= \tfrac{1}{2}\ln(2\pi\sigma^2) + \frac{\mu^2}{2\sigma^2} - \frac{2\mu^2}{2\sigma^2} + \frac{1}{2\sigma^2}(\mu^2 + \sigma^2) \\
&= \tfrac{1}{2}\ln(2\pi\sigma^2) + \tfrac{1}{2} = \tfrac{1}{2}\ln(2\pi\sigma^2) + \tfrac{1}{2}\ln e = \tfrac{1}{2}\ln(2\pi\sigma^2 e)
\end{aligned}$$

Entropy of a GM

- Thm 8.4.3. If P(X) = 1/Z $\prod_c \phi_c(X)$ is a UGM, then

$$H(P_{\boldsymbol{\theta}}(\mathbf{x})) = \log Z(\boldsymbol{\theta}) + \sum_c E[-\ln \phi_c(\mathbf{x}_c)]$$

- Thm 8.4.5. If P(X) is a DGM, then

$$H(P(\mathbf{X})) = \sum_i H(P(X_i|X_{\pi_i}))$$

- Pf.

$$\begin{aligned} H(P(\mathbf{X})) &= E[-\log p(\mathbf{X})] = E[-\sum_i \log p(X_i|\mathbf{X}_{\pi_i})] \\ &= \sum_i E[-\log p(X_i|\mathbf{X}_{\pi_i})] = \sum_i H(P(X_i|\mathbf{X}_{\pi_i})) \\ &= \sum_i \sum_{\mathbf{X}_{\pi_i}} p(\mathbf{x}_{\pi_i}) H(P(X_i|\mathbf{x}_{\pi_i})) \end{aligned}$$

- Thm 8.4.6. If P(X) is a DGM, then

$$\sum_i \min_{\mathbf{X}_{\pi_i}} H(P(X_i|\mathbf{x}_{\pi_i})) \leq H(P(\mathbf{X})) \leq \sum_i \max_{\mathbf{X}_{\pi_i}} H(P(X_i|\mathbf{x}_{\pi_i}))$$

Projections

- Def 8.5.1. Let P b a distribution and Q a convex set of distributions.
- The I-projection (information) is

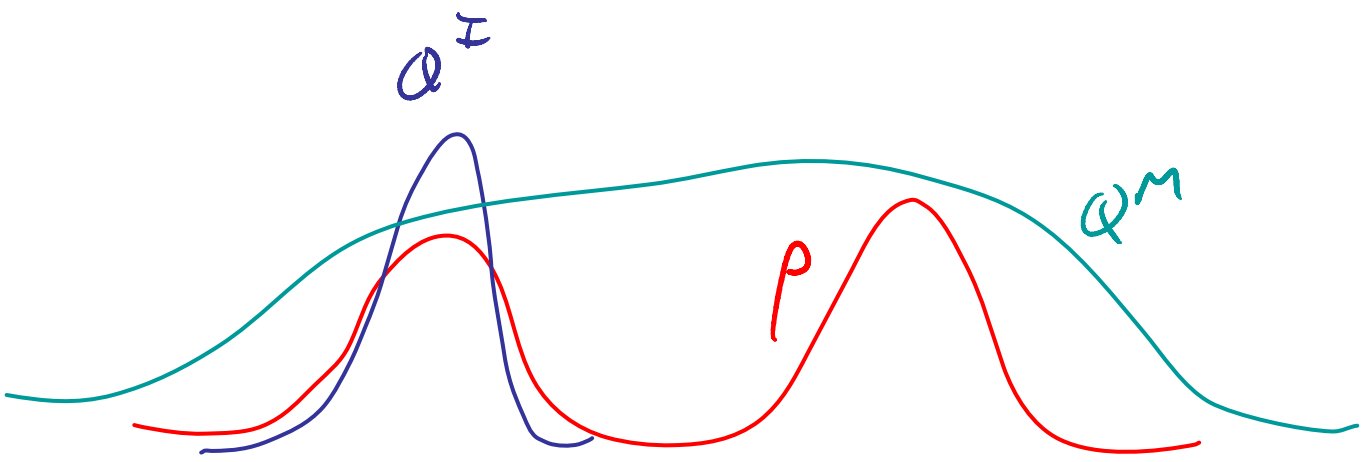
$$Q^I = \arg\min_{Q \in \mathcal{Q}} D(Q||P)$$

Zero forcing: P=0 => Q=0 Mode seeking

- The M-projection (moment) is

$$Q^M = \arg\min_{Q \in \mathcal{Q}} D(P||Q)$$

Q=0 => P=0 High variance



M-projection is moment matching

- Thm 8.5.5. Let P be any distrib over X, and let Q be expfam. If there is a set of params θ st $E_Q(\theta)[\tau(X)] = E_P[\tau(X)]$, then the M-projection of P onto Q is Q_θ.
- Ex. Let Q = fully factorized distribution. Then Q^M is given by product of marginals.

$$Q^M(\mathbf{x}) = p(X_1) \dots p(X_d)$$

- Ex. Let P = mix Gaussians, Q = single Gaussian.

$$\begin{aligned} p(\mathbf{x}) &= \sum_k \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ Q^M(\mathbf{x}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_Q, \boldsymbol{\Sigma}_Q) \\ \boldsymbol{\mu}_Q &= \sum_k \pi_k \boldsymbol{\mu}_k \\ \boldsymbol{\Sigma}_Q &= \sum_k \pi_k (\boldsymbol{\Sigma}_k + (\boldsymbol{\mu}_k - \boldsymbol{\mu}_Q)(\boldsymbol{\mu}_k - \boldsymbol{\mu}_Q)^T) \end{aligned}$$

I-projection

- I-projection requires computing expectations of log(P) – which often factorizes - wrt Q, and the entropy of Q.

$$Q^I = \arg\min_{Q \in \mathcal{Q}} D(Q||P) = \arg\min \sum_x Q(x) \log \frac{Q(x)}{P(x)}$$

- We can choose Q to be “simple”, so that it is easy to compute these expectations and entropy terms.
- This is the basis of variational inference.
- By contrast, M-projections require expectations wrt P. Usually this can only be done locally, as in expectation propagation.

Querying a distribution ("inference")

- Suppose we have a joint $p(X_1,\ldots,X_d)$. Partition the variables into E (evidence), Q (query), and H (hidden/ nuisance). We might pose the following queries
- Conditional probability (posterior):

$$p(\mathbf{X}_Q|\mathbf{x}_E) \propto \sum_{\mathbf{x}_H} p(\mathbf{X}_Q, \mathbf{x}_E, \mathbf{x}_H)$$

- MAP estimate (H=∅) (posterior mode)

$$\mathbf{x}_Q^* = \arg\max_{\mathbf{x}_Q} p(\mathbf{x}_Q|\mathbf{x}_E) = \arg\max_{\mathbf{x}_Q} p(\mathbf{x}_Q, \mathbf{x}_E)$$

- Marginal MAP estimate (mode of marginal post):

$$\mathbf{x}_Q^* = \arg\max_{\mathbf{x}_Q} p(\mathbf{x}_Q|\mathbf{x}_E) = \arg\max_{\mathbf{x}_Q} \sum_{\mathbf{x}_H} p(\mathbf{x}_Q, \mathbf{x}_E, \mathbf{x}_H)$$

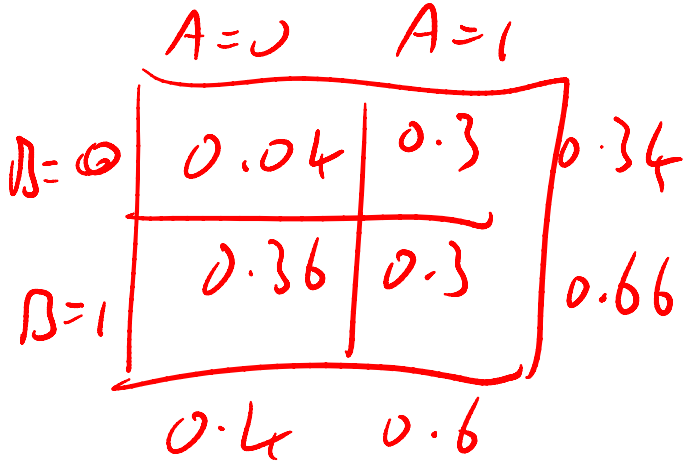
MAP vs marginal MAP

- Max max ≠ max sum
- Ex 2.1.12. Joint is

$$a^* = \arg\max_a \sum_b p(a,b) = 1$$

$$b^* = \arg\max_b \sum_a p(a,b) = 1$$

$$(a,b)^* = \arg\max_{a,b} p(a,b) = (0,1)$$



- One can show that max sum is strictly computationally harder than sum, which is in turn harder than max

Speech recognition

- Eg speech recognition. Let Q=words, H = pronunciation (phonemes sequence), E = signal.
- We often make the following approximation, which lets us use the Viterbi algorithm

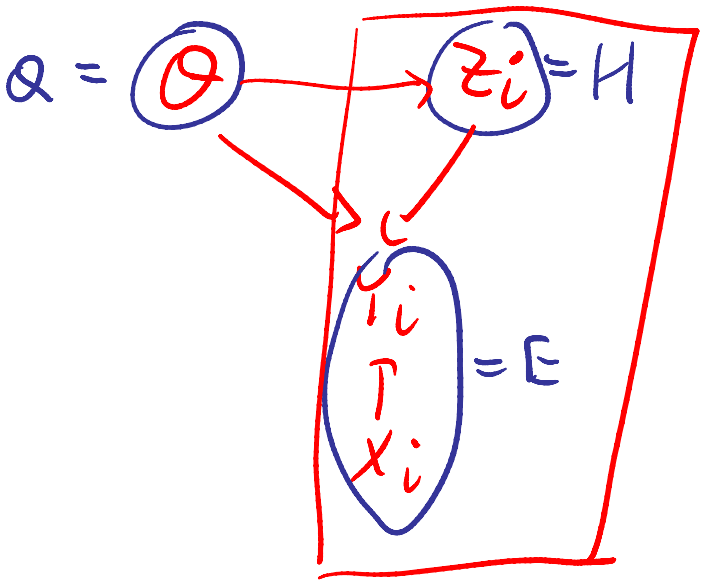
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{\mathbf{h}} p(\mathbf{w}, \mathbf{h}|\mathbf{e}) \approx \arg\max_{\mathbf{w}} \max_{\mathbf{h}} p(\mathbf{w}, \mathbf{h}|\mathbf{e})$$

- Eg. Consider W1="a back", vs W2="aback". There might be 10 alternative state sequences for W1, each with prob 0.03, but just one sequence for W2, with prob 0.2. Viterbi would choose W2, but W1 is actually more likely.

Bayesian statistics

- Bayesian statistics amounts to defining a single joint distribution for both “variables” – latent and observed - and “parameters” (often fixed in number), and then querying the parameters.

$$p(\boldsymbol{\theta}|\mathbf{X},\mathbf{Y}) \quad \propto \quad p(\boldsymbol{\theta}) \prod_i \int p(\mathbf{z}_i|\boldsymbol{\theta}) p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{z}_i,\boldsymbol{\theta}) d\mathbf{z}_i$$



Probability of evidence

- To compute conditional queries, we need to evaluate $p(x_E)$

$$p(\mathbf{X}_Q|\mathbf{x}_E) = \frac{\sum_{\mathbf{x}_H} p(\mathbf{X}_Q, \mathbf{x}_E, \mathbf{x}_H)}{p(\mathbf{x}_E)}$$

$$p(\mathbf{x}_E) = \sum_{\mathbf{x}_Q} \sum_{\mathbf{x}_H} p(\mathbf{x}_Q, \mathbf{x}_E, \mathbf{x}_H)$$

- This may be a high dimensional integral

$$p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Y}) = \frac{p(\boldsymbol{\theta}) \prod_i \int p(\mathbf{z}_i|\boldsymbol{\theta}) p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta}) d\mathbf{z}_i}{p(\mathbf{X}, \mathbf{Y})}$$

$$p(\mathbf{X}, \mathbf{Y}) = \int p(\boldsymbol{\theta}) \left[\prod_i \int p(\mathbf{z}_i|\boldsymbol{\theta}) p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta}) d\mathbf{z}_i \right] d\boldsymbol{\theta}$$

- $p(x_E)$ can be used to decide how likely x_E is to have come from this model (classification and model selection)

Sampling

- Often the posterior is too big to even store explicitly.
- Marginals and MAP estimates are one summary, but may be unrepresentative.
- Samples may provide a better summary.
- eg Attractive Ising model has 2 modes, all 0 and all 1. The marginals are [0.5, 0.5].
- We want to be able to sample from p(xQ|xE)
- Sometimes we can do this even if we cannot evaluate p(xE) – this is the key idea behind MCMC

Monte Carlo integration

- Sometimes we want to E[f(xQ)|xE], where f() depends on global properties of Q, so we cannot use marginal distributions.
- However, if we sample from p(XQ|xE), we can use

$$E[f(\mathbf{X}_Q)|\mathbf{x}_E] = \int f(\mathbf{x}_Q)p(\mathbf{x}_Q|\mathbf{x}_E)d\mathbf{x}_Q \approx \frac{1}{N}\sum_{i=1}^{n} f(\mathbf{x}_Q^i)$$

Inference in discrete state spaces

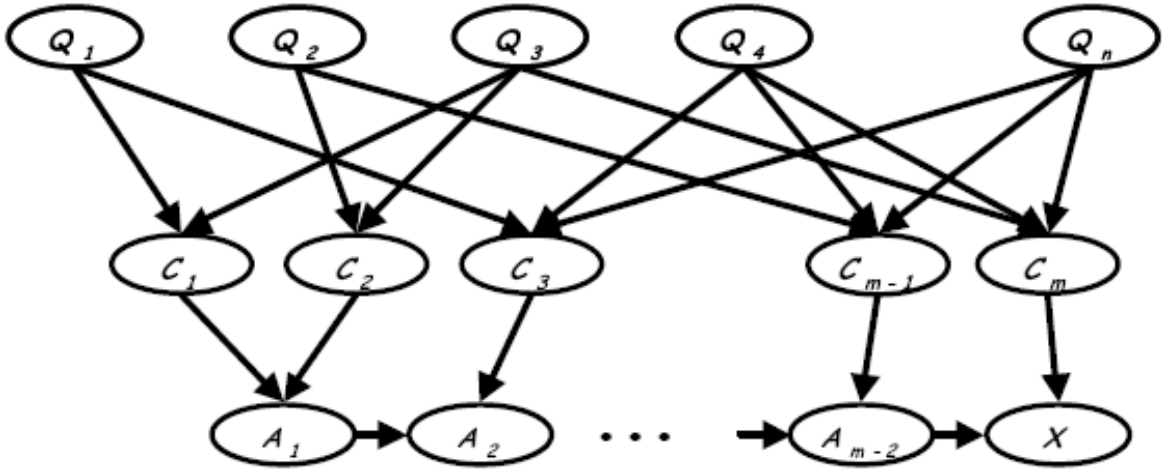
- We will mostly focus on the case where Q and H are discrete rv's (E can be cts or discrete).
- Thus everything amounts to computing a large number of sums as quickly as possible.
- We will also consider the case where Q, H and E are all jointly Gaussian, where exact answers can also be obtained.
- For general distributions (eg for applications in Bayesian statistics), exact inference is usually not possible (except 1 layer of parameters with conjugate priors and no latent variables).

Complexity of inference

- Consider computing p(X_Q), p(X_Q|x_E), or p(x_E) for a discrete state space.
- Later we will show that if P is representable by a GM, then we can compute these quantities efficiently, if the graph has special properties.
- However, in general, the problem is computationally expensive.

Complexity of exact inference

- Thm 9.1.1. Given a DGM, deciding if p(X=x)>0 is NP-complete.
- Pf. Easy to see is in NP (linear time to check if p(x)>0.) Can show is NP-hard by showing how to reduce 3-SAT to a poly-sized DGM.



$$X = (Q_1 \vee \neg Q_2 \vee Q_3) \wedge (Q_2 \vee Q_5 \vee Q_3) \cdots$$

P(X=1) = #satisfying assignments/ 2^n

Complexity of exact inference

- Defn. NP is the class of problems of the form “are there any solutions x such that f(x) is true”. #P is the class of problems “Count the number of solutions x st f(x) is true”.
- Thm 9.1.2. Given a DGM, computing p(X=x) is #P-complete.

Complexity of approximate inference

- Def 9.1.3. A estimate ρ has absolute error ε if

$$|p(\mathbf{x}_Q|\mathbf{x}_e) - \rho| \leq \epsilon$$

- Def 9.1.4. An estimate ρ has relative error ε if

$$\frac{\rho}{1+\epsilon} \leq p(\mathbf{x}_Q|\mathbf{x}_e) \leq \rho(1+\epsilon)$$

- Thm 9.1.5. Given a DGM, finding a number ρ which as relative error ε for p(X=x) is NP-hard.
- Thm 9.1.6. Given a DGM, finding a number ρ that has absolute error ε for p(X|e) is NP-hard for any $0 \leq \varepsilon \leq 0.5$.

Stat 521A
Lecture 7

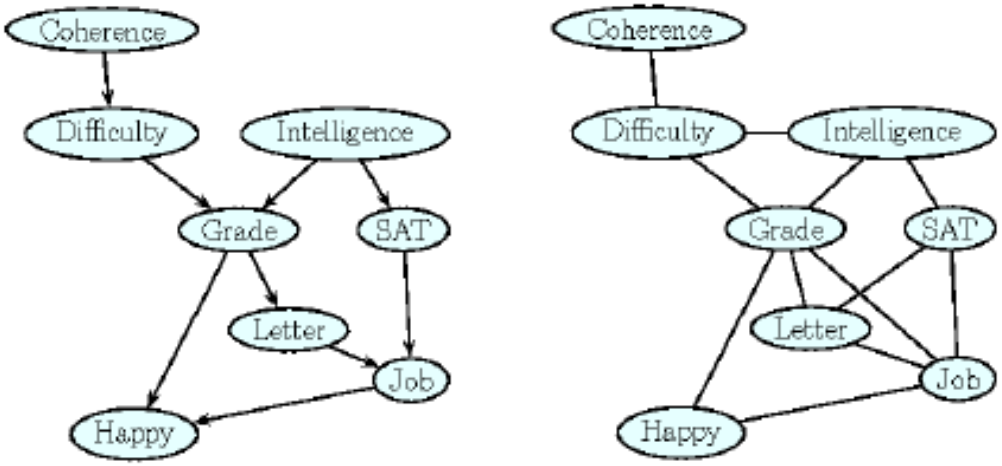
Outline

- Variable elimination (9.2-9.3)
- Complexity of VE (9.4)
- Conditioning (9.5)
- From VE to clique trees (10.1)
- Message passing on clique trees (10.2-10.3)
- Creating clique trees (10.4)

Inference

- Consider the following distribution

$$P(C, D, I, G, S, L, J, H)$$
$$= P(C)P(D|C)P(I)P(G|I, D)P(S|I)P(L|G)P(J|L, S)P(H|G, .$$
$$P(C, D, I, G, S, L, J, H)$$
$$= \psi_C(C)\psi_D(D, C)\psi_I(I)\psi_G(G, I, D)\psi_S(S, I)$$
$$\psi_L(L, G)\psi_J(J, L, S)\psi_H(H, G, J)$$



Brute force enumeration

- Compute marginal probability someone has a job

$$P(J) = \sum_{L}\sum_{S}\sum_{G}\sum_{H}\sum_{I}\sum_{D}\sum_{C} P(C, D, I, G, S, L, J, H)$$

Variable elimination 1

- Push sums inside products (distributive law)

$$
\begin{aligned}
P(J) &= \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C P(C, D, I, G, S, L, J, H) \\
&= \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C \psi_C(C)\psi_D(D, C)\psi_I(I)\psi_G(G, I, D)\psi_S(S, I) \\
&\quad \psi_L(L, G)\psi_J(J, L, S)\psi_H(H, G, J) \\
&= \sum_L \sum_S \psi_J(J, L, S) \sum_G \psi_L(L, G) \sum_H \psi_H(H, G, J) \sum_I \psi_S(S, I)\psi_I(I) \\
&\quad \sum_D \psi_G(G, I, D) \sum_C \psi_C(C)\psi_D(D, C)
\end{aligned}
$$

VE 2: work right to left

$$
\begin{aligned}
P(J) &= \sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \sum_D \psi_G(G,I,D) \underbrace{\sum_C \psi_C(C)\psi_D(D,C)}_{\tau_1(D)} \\
&= \sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \underbrace{\sum_D \psi_G(G,I,D)\tau_1(D)}_{\tau_2(G,I)} \\
&= \sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \underbrace{\sum_I \psi_S(S,I)\psi_I(I)\tau_2(G,I)}_{\tau_3(G,S)} \\
&= \sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \underbrace{\sum_H \psi_H(H,G,J)}_{\tau_4(G,J)} \tau_3(G,S) \\
&= \sum_L \sum_S \psi_J(J,L,S) \underbrace{\sum_G \psi_L(L,G)\tau_4(G,J)\tau_3(G,S)}_{\tau_5(J,L,S)} \\
&= \sum_L \underbrace{\sum_S \psi_J(J,L,S)\tau_5(J,L,S)}_{\tau_6(J,L)} \\
&= \underbrace{\sum_L \tau_6(J,L)}_{\tau_7(J)}
\end{aligned}
$$

Variable elimination
Bucket elimination
Peeling
Non-serial dynamic programming

Pseudocode

Algorithm 9.1 Sum-Product Variable Elimination algorithm

```
    Procedure Sum-Product-Variable-Elimination (
      Φ,    // Set of factors
      Z,    // Set of variables to be eliminated
      ≺     // Ordering on Z
    )
1     Let Z_1, ..., Z_k be an ordering of Z such that
2       Z_i ≺ Z_j iff i < j
3     for i = 1, ..., k
4       Φ ← Sum-Product-Eliminate-Var(Φ, Z_i)
5     φ* ← ∏_{φ∈Φ} φ
6     return φ*

    Procedure Sum-Product-Eliminate-Var (
      Φ,    // Set of factors
      Z     // Variable to be eliminated
    )
1     Φ' ← {φ ∈ Φ : Z ∈ Scope[φ]}
2     Φ'' ← Φ − Φ'
3     ψ ← ∏_{φ∈Φ'} φ
4     τ ← ∑_Z ψ
5     return Φ'' ∪ {τ}
```

Dealing with evidence

- Conditional prob is ratio of uncond prob

$$P(J|I=1,H=0) = \frac{P(J,I=1,H=0)}{P(I=1,H=0)}$$

- Soft/ virtual evidence: $\phi_i(X_i) = p(y_i|X_i)$

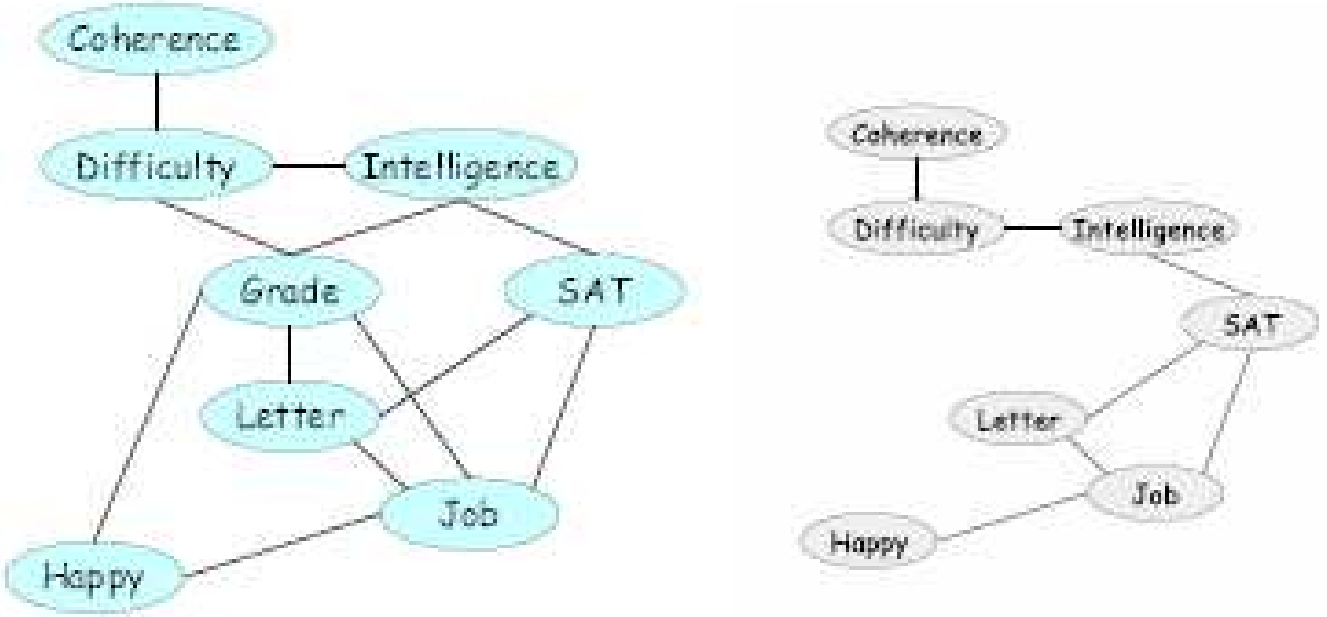
$$P(J,I=1,H=0) =$$
$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J)\phi_{\mathbf{H}}(\mathbf{H}) \sum_I \psi_S(S,I)\psi_I(I)\phi_{\mathbf{I}}(\mathbf{I})$$
$$\sum_D \psi_G(G,I,D) \sum_C \psi_C(C)\psi_D(D,C)$$

- Hard evidence: $\phi_i(X_i) = I(X_i=x_i^*)$

$$P(J,I=1,H=0) =$$
$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G)\psi_H(H=0,G,J)\psi_S(S,I=1)\psi_I(I=1)$$
$$\sum_D \psi_G(G,I,D) \sum_C \psi_C(C)\psi_D(D,C)$$

Reduced graph

- If nodes are instantiated (fully observed), we can remove them and their edges and absorb their effect by updating all the other factors that reference them
- Eg if G is observed



VE with hard evidence

Procedure Cond Prob VE (
  $\mathcal{K}$, // A network over $\mathcal{X}$
  $\boldsymbol{Y}$, // Set of query variables
  $\boldsymbol{E} = \boldsymbol{e}$ // Evidence
)

1 $\Phi \leftarrow$ Factors parameterizing $\mathcal{K}$
2 Replace each $\phi \in \Phi$ by $\phi[\boldsymbol{E} = \boldsymbol{e}]$
3 Select an elimination ordering $\prec$
4 $\boldsymbol{Z} \leftarrow = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$
5 $\phi^* \leftarrow$ Sum Product Variable Elimination$(\Phi, \prec, \boldsymbol{Z})$
6 $\alpha \leftarrow \sum_{\boldsymbol{y} \in Val(\boldsymbol{Y})} \phi^*(\boldsymbol{y})$
7 **return** α, ϕ^*

Complexity analysis of VE

- At step i, we multiply all factors involving x_i into a large factor, then sum out x_i to get τ_i.
- Let N_i be number of entries in factor ψ_i.
- The total number of factors is m+n, where m = original number of factors in model ($m \geq n$), and n=num. vars.
- Each factor gets multiplied into something bigger once. Hence #mult is at most

$$(n+m)N_i \leq (n+m)N_{max} = O(mN_{max})$$

- When we sum out a node from a factor, we touch each entry once, so #adds is at most

$$nN_{max}$$

Complexity analysis of VE

- If each variable has v values, and factor ψ_i involves k_i variables, then $N_i \leq v^k$
- So complexity is exponential in the size of the largest factor.

Different elimination ordering

$$
\begin{aligned}
P(J) &= \sum_D \sum_C \psi_D(D,C) \sum_H \sum_L \sum_S \psi_J(J,L,S) \sum_I \psi_I(I)\psi_S(S,I) \underbrace{\sum_G \psi_G(G,I,D)\psi_L(L,)\psi_H(H,G,J)}_{\tau_1(I,D,L,J,H)} \\
&= \sum_D \sum_C \psi_D(D,C) \sum_H \sum_L \sum_S \psi_J(J,L,S) \underbrace{\sum_I \psi_I(I)\psi_S(S,I)\tau_1(I,D,L,J,H)}_{\tau_2(D,L,S,J,H)} \\
&= \sum_D \sum_C \psi_D(D,C) \sum_H \sum_L \underbrace{\sum_S \psi_J(J,L,S)\tau_2(D,L,S,J,H)}_{\tau_3(D,L,J,H)} \\
&= \sum_D \sum_C \psi_D(D,C) \sum_H \underbrace{\sum_L \tau_3(D,L,J,H)}_{\tau_4(D,J,H)} \\
&= \sum_D \sum_C \psi_D(D,C) \underbrace{\sum_H \tau_4(D,J,H)}_{\tau_5(D,J)} \\
&= \sum_D \underbrace{\sum_C \psi_D(D,C)\tau_5(D,J)}_{\tau_6(D,J)} \\
&= \underbrace{\sum_D \tau_6(D,J)}_{\tau_7(J)}
\end{aligned}
$$

Effect of ordering

- A bad ordering can create larger intermediate factors, and therefore is slower

Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$. $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$. $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$. $\phi_S(S,I)$. $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$. $\tau_3(G,S)$. $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$. $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

4

Table 9.1 A run of variable elimination for the query $P(J)$.

Step	Variable eliminated	Factors used	Variables involved	New factor
1	G	$\phi_G(G,I,D)$. $\phi_L(L,G)$. $\phi_H(H,G,J)$	G,I,D,L,J,H	$\tau_1(I,D,L,J,H)$
2	I	$\phi_I(I)$. $\phi_S(S,I)$. $\tau_1(I,D,L,S,J,H)$	S,I,D,L,J,H	$\tau_2(D,L,S,J,H)$
3	S	$\phi_J(J,L,S)$. $\tau_2(D,L,S,J,H)$	D,L,S,J,H	$\tau_3(D,L,J,H)$
4	L	$\tau_3(D,L,J,H)$	D,L,J,H	$\tau_4(D,J,H)$
5	H	$\tau_4(D,J,H)$	D,J,H	$\tau_5(D,J)$
6	C	$\tau_5(D,J)$. $\phi_D(D,C)$	D,J,C	$\tau_6(D,J)$
7	D	$\tau_6(D,J)$	D,J	$\tau_7(J)$

6
6

Table 9.2 A different run of variable elimination for the query $P(J)$.

Graph theoretic analysis

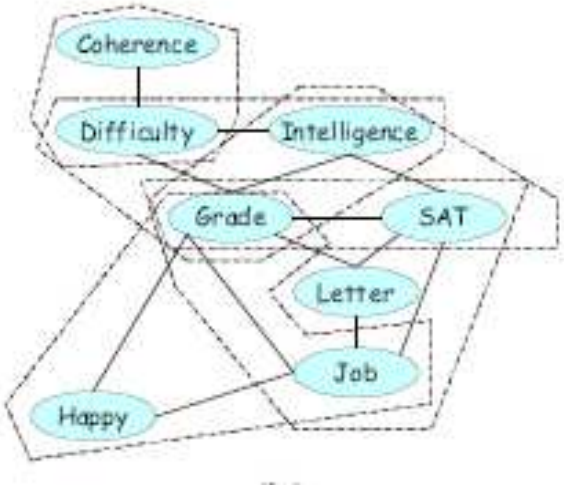
- Every time we eliminate a node, we build a new factor which combines variables that may have previously been in separate factors. Let us add an edge (fill-in edge) between such nodes to create the induced graph

When we eliminate I, we add a fill-in between G and S

$$\tau_3(G,S) = \sum_I \psi_S(S,I)\psi_I(I)\tau_2(G,I)$$

Induced graph

- Def 9.4.3. Let I(G,<) represent the graph induced by applying VE with order < to graph G.
- Thm 9.4.4.Every factor generated by VE is a clique in I(G,<). Also, every maximal clique in I(G,<) corresponds to some intermediate factor.



Variables involved
C, D
G, I, D
G, S, I
H, G, J
G, J, L, S
J, L, S
J, L

$$\{C, D\}, \{D, I, G\}, \{G, L, S, J\}, \{G, J, H\}, \{G, I, S\}$$

Treewidth

- Def 9.4.5. The width of an induced graph is the number of nodes in the largest clique minus 1. The minimal induced width of a graph, aka the treewidth, is defined as

$$W_G = \min_{\prec} \max_i |\tau_i| - 1$$

- The treewidth of a tree is 1, since the max clique (edge) in the original graph has size 2, and the optimal elimination order (eliminate all the leaves, then the root) adds no fill-in edges.

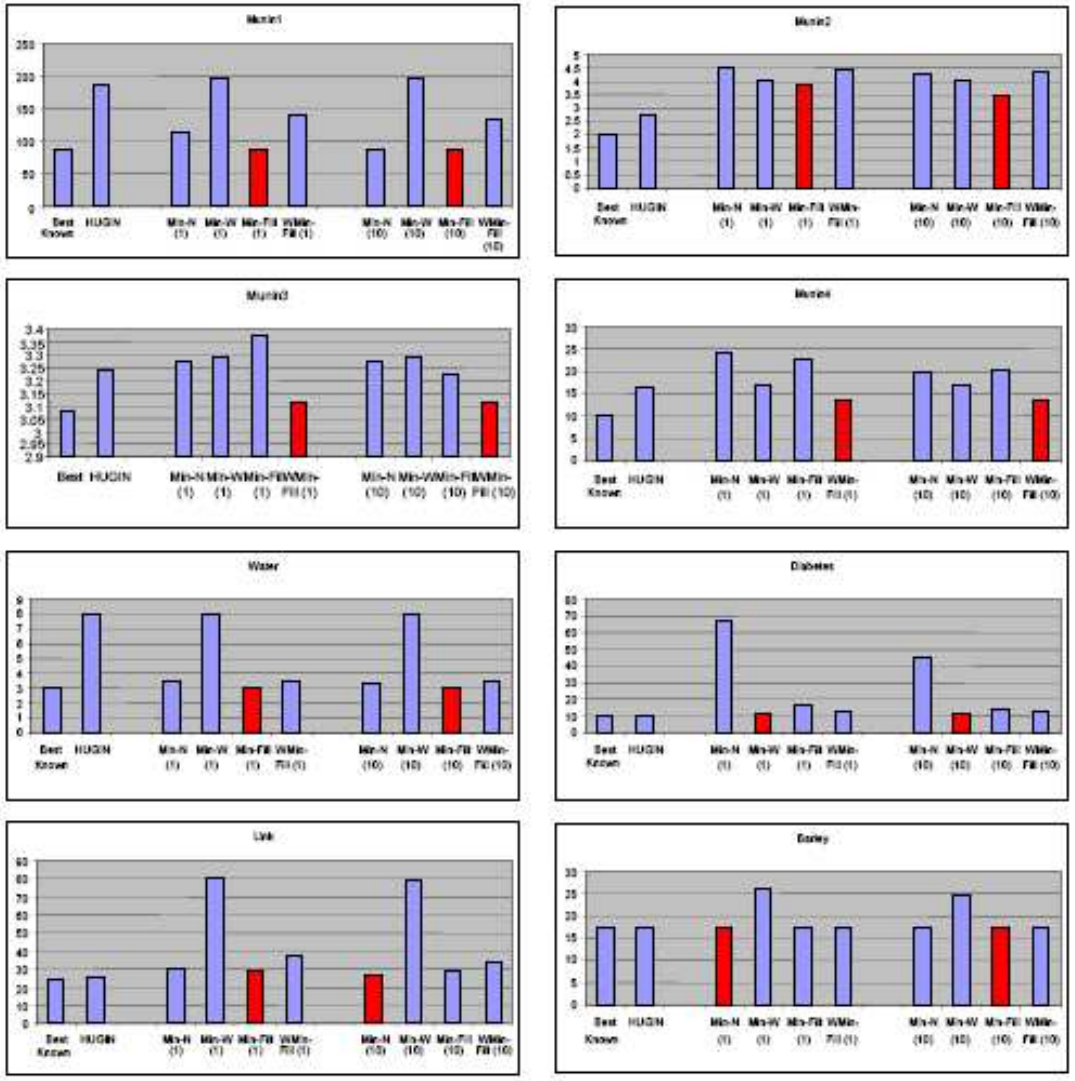
$$1,2,3 \quad : \quad \sum_{x_3} \sum_{x_2} \phi(x_3, x_2) \sum_{x_1} \phi(x_3, x_1)$$

$$3,2,1 \quad : \quad \sum_{x_1} \sum_{x_2} \sum_{x_3} \phi(x_3, x_1) \phi(x_3, x_2)$$

Finding an elim order

- Thm 9.4.6. Finding the optimal elimination order (which minimizes induced width) is NP-hard.
- Typical approach: greedy search, where at each step, we eliminate the node that minimizes some cost function
- Min-fill heuristic: the cost of a node is the number of fill-in edges that would be added.
- Min-weight heuristic: the cost of a node is the number of states in the factor that would be created (product of cardinalities).

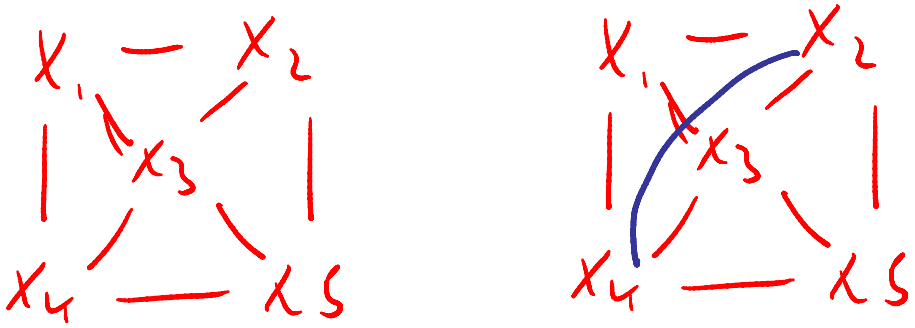
Empirical comparison of heuristics



Min-fill is often close to best known ordering (computed with simAnneal)

Chordal graphs

- Def 2.2.15. Let $X_1 – X_2$ - … $X_k – X_1$ be a loop in a graph. A chord is an edge connecting X_i and X_j for two nonconsecutive nodes. An undirected graph is chordal (triangulated) if every loop of length k >= 4 has a chord.

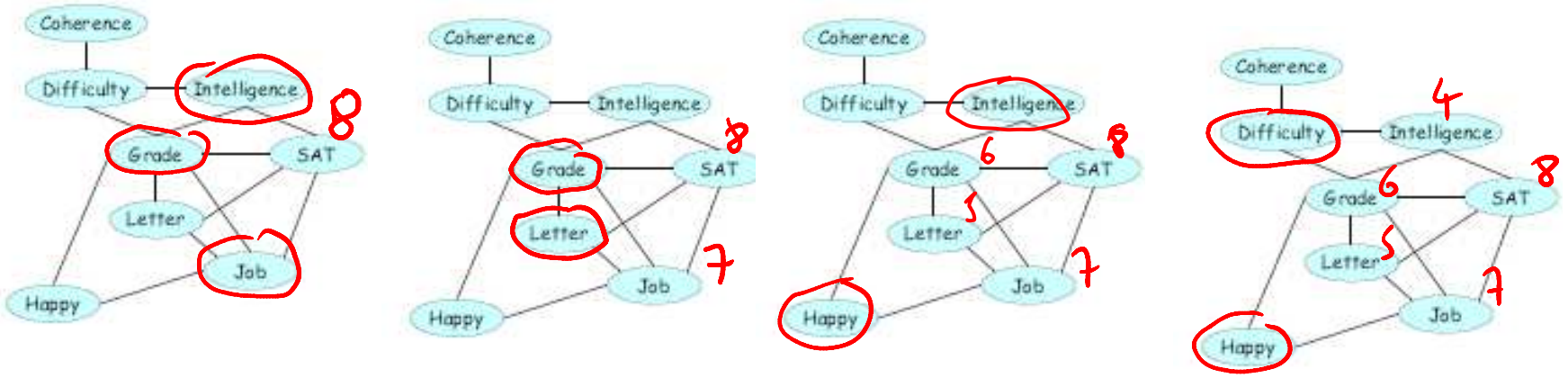


- Thm 9.4.7. Every induced graph is chordal.
- Thm 9.4.8. Any chordal graph admits a perfect elimination order which does not introduce any fill-in edges.

Finding perfect elim order

- The max cardinality search algorithm will find a perfect elimination ordering for a chordal graph.

```
Procedure Max-Cardinality (
    H    // An undirected graph over X
)
1    Initialize all nodes in X as unmarked
2    for k = |X| ... 1
3        X ← unmarked variable in X with largest number of marked neighbors
4        π(X) ← k
5        Mark X
6    return π
```



For non-chordal graphs, the MCS ordering often results in large induced width

Conditioning

- We can condition on a variable to break the graph into smaller pieces, run VE on each piece, and then add up the results. We also need the probability of each conditioning case.

$$\tilde{P}(\mathbf{Y}) = \sum_{\mathbf{u}} \tilde{P}(\mathbf{Y}, \mathbf{u}) \qquad Z = \sum_{\mathbf{u}} Z(\mathbf{u})$$

Evidence G=g

Condition on S

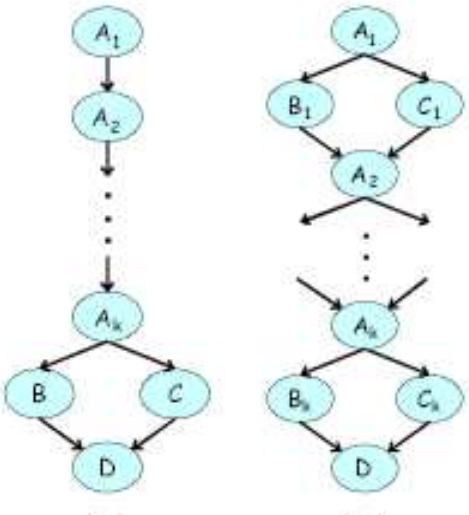
Conditioning + VE

Procedure Sum-Product-Conditioning (
  Φ, // Set of factors, possibly reduced by evidence
  $\boldsymbol{Y}$, // Set of query variables
  $\boldsymbol{U}$ // Set of variables on which to condition
)

1 **for** each $\boldsymbol{u} \in Val(\boldsymbol{U})$
2 $\quad \Phi_{\boldsymbol{u}} \leftarrow \{\phi[\boldsymbol{U} = \boldsymbol{u}] : \phi \in \Phi\}$
3 $\quad$ Construct $\mathcal{H}_{\Phi_{\boldsymbol{u}}}$
4 $\quad (\alpha_{\boldsymbol{u}}, \phi_{\boldsymbol{u}}(\boldsymbol{Y})) \leftarrow$ Cond-Prob-VE$(\mathcal{H}_{\Phi_{\boldsymbol{u}}}, \boldsymbol{Y}, \emptyset)$
5 $\phi^*(\boldsymbol{Y}) \leftarrow \frac{\sum_{\boldsymbol{u}} \phi_{\boldsymbol{u}}(\boldsymbol{Y})}{\sum_{\boldsymbol{u}} \alpha_{\boldsymbol{u}}}$
6 Return $\phi^*(\boldsymbol{Y})$

Cutset conditioning

- If we instantiate a set of nodes such that the resulting network is a tree, we can apply a simple message passing algorithm on the tree (see later).
- This is called cutset conditioning.
- Thm 9.5.2. Conditioning + VE is never more efficient than VE.

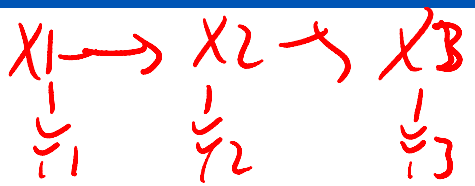


Left: condition on A_k. Repeatedly re-eliminate $A_1 \ldots A_{k-1}$ instead of reusing computation (as in DP/VE).

Right: condition on A_k, k odd. Exponential in k. But induced width is only 2.

Space-time tradeoff.

VE on chain = forwards algorithm



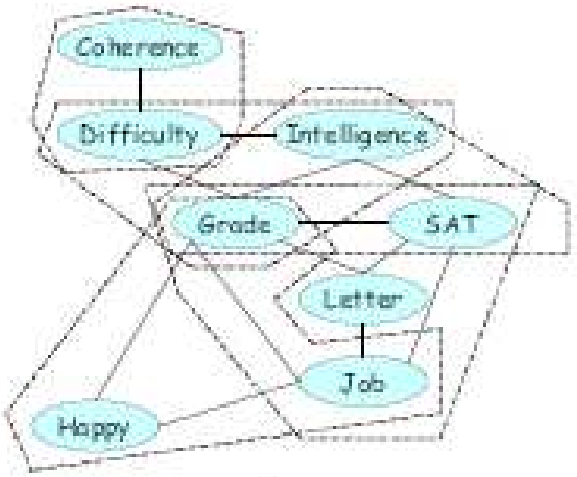
$$
\begin{aligned}
p(x_1, x_2, x_3 | y_1, y_2, y_3) &\propto \phi_1(x_1)\psi(x_1, x_2)\phi_2(x_2)\psi(x_2, x_3)\phi_3(x_3) \\
\phi_1(x_1) &= \pi_1(x_1)p(y_1|x_1) \\
\phi_t(x_t) &= p(y_t|x_t), t > 1 \\
\psi(x_{t-1}, x_t) &= p(x_t|x_{t-1}) \\
p(x_3|y_{1:3}) &\propto \phi_3(x_3) \sum_{x_2} \phi_2(x_2)\psi(x_2, x_3) \sum_{x_1} \phi_1(x_1)\psi(x_1, x_2) \\
\alpha_1(x_1) &\propto \phi_1(x_1) \\
\alpha_2(x_2) &\propto \phi_2(x_2) \sum_{x_1} \alpha_1(x_1)\psi(x_1, x_2) \\
\alpha_3(x_3) &\propto \phi_3(x_3) \sum_{x_2} \alpha_2(x_2)\psi(x_2, x_3)
\end{aligned}
$$

What's wrong with VE?

- Consider a chain X1 – X2 - .. – XT, where the local evidence has been absorbed into the node factors.
- If we use VE to compute p(XT|y(1:T)), it is equivalent to the forwards algorithm for HMMs, and takes $O(T K^2)$ time, where K = #states.
- Suppose we also want to compute p(X(T-1)|y(1:T)). We could rerun the algorithm for an additional $O(T K^2)$ time.
- We now discuss how to reuse most of the computation we have already done in eliminating X(1:T-2). We can then compute all marginals in $O(2 K^2 T)$ time (FB algorithm).

Cluster graphs

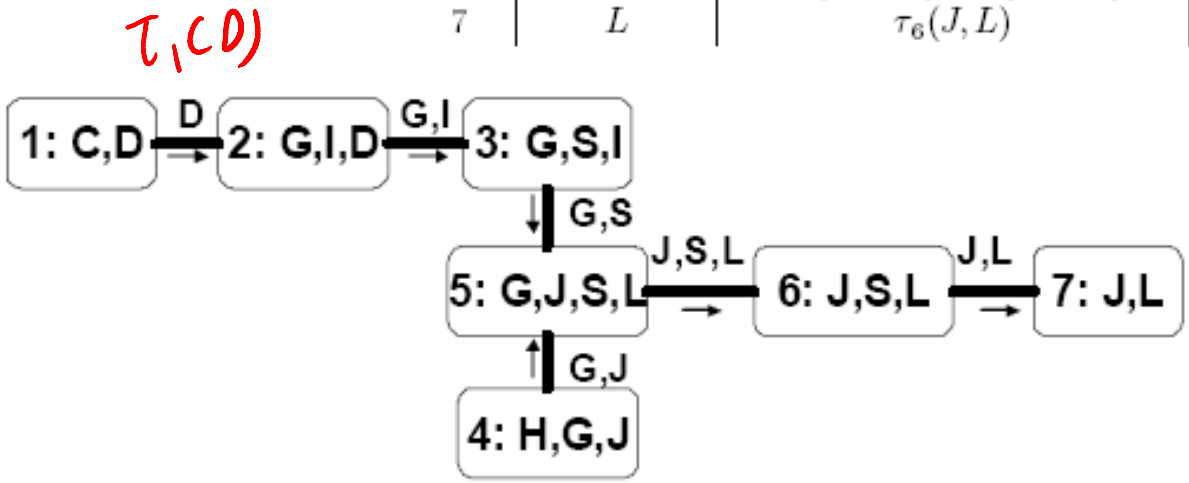
- Def 10.1.1. A cluster graph for a set of factors on X is an undirected graph, each of whose nodes I is associated with a set $C_i \subseteq X$. Each factor is contained in precisely one cluster. Each edge between a pair of clusters C_i, C_j is associated with a sepset (separating set) S_{ij}. $S_{ij} = C_i \cap C_j$



Cluster graph from VE

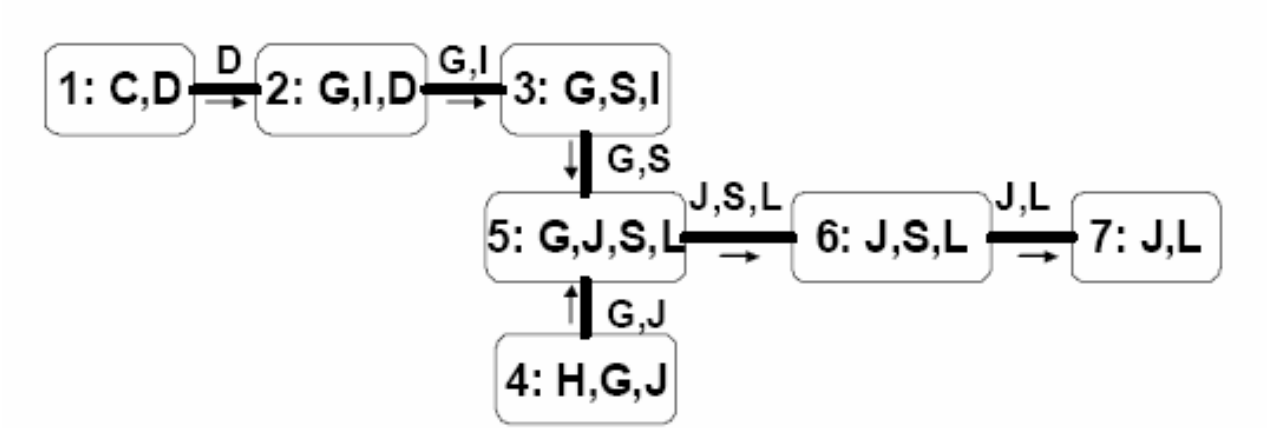
- We can create a cluster graph to represent the process of VE. Before we marginalize out x_i, we create factor ψ_i (its bucket potential); make this a cluster. When we marginalize out x_i, we create factor τ_i which is stored in bucket j; think of this as a message from i to j. Draw an edge $C_i - C_j$.

Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C), \phi_D(D,C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G,I,D), \tau_1(D)$	G, I, D	$\tau_2(G,I)$
3	I	$\phi_I(I), \phi_S(S,I), \tau_2(G,I)$	G, S, I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H, G, J	$\tau_4(G,J)$
5	G	$\tau_4(G,J), \tau_3(G,S), \phi_L(L,G)$	G, J, L, S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S), \phi_J(J,L,S)$	J, L, S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J, L	$\tau_7(J)$



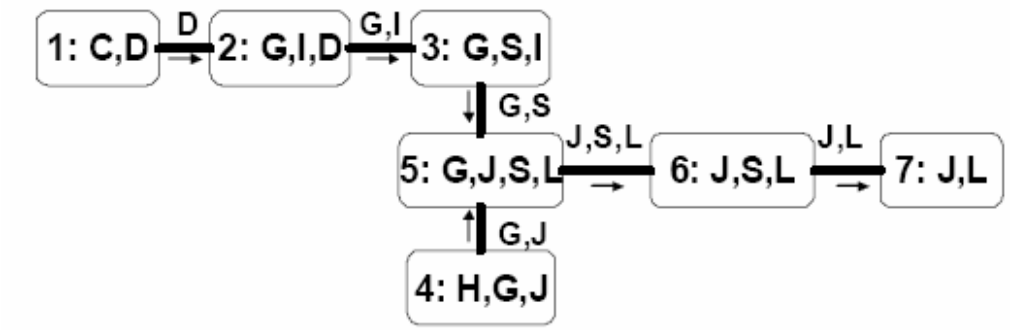
Properties of VE cluster graph

- The VE cluster graph is a tree, since each message gets sent to a single bucket (so each cluster connects to at most one other cluster)
- Def 10.1.3. Let T be a cluster tree. T has the **running intersection property** if , whenever X in Ci and X in Cj, then X is also in every cluster on the unique path from Ci to Cj.
- Thm 10.1.5. The VE CG has RIP.
- Pf (sketch). A variable appears in every factor from the moment it is introduced to when it is summed out.



Messages

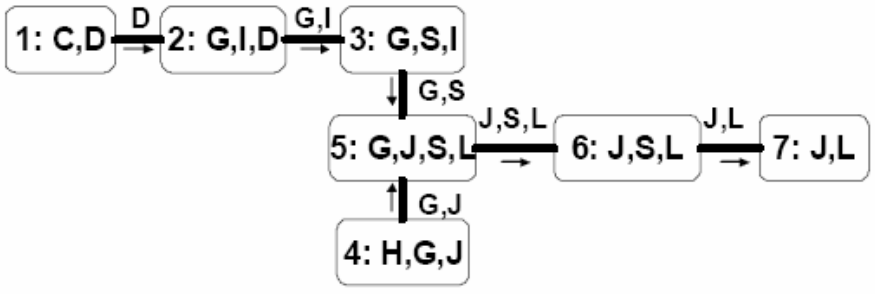
- Thm 10.1.6. The scope of the msg τ_i from C_i to C_i is $S_{i,j}$.

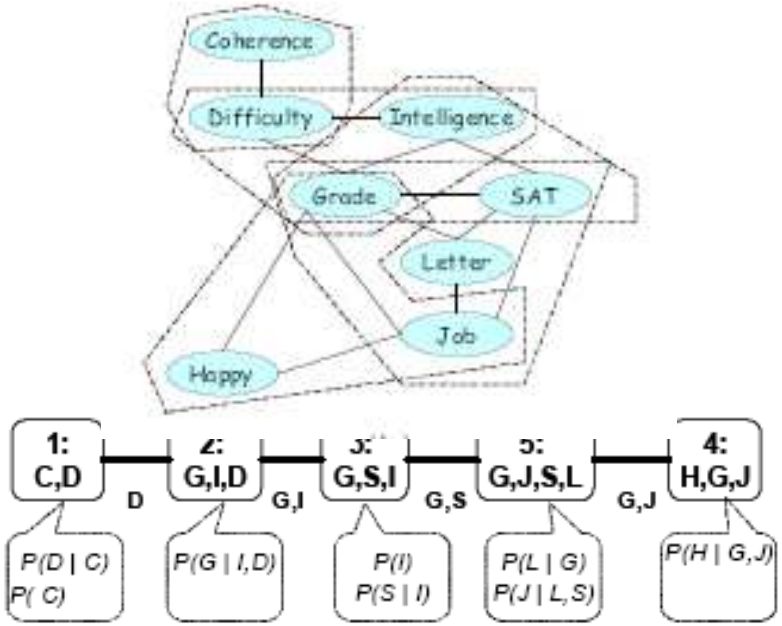


- Def. For any sepset S_{ij}, let $W_{<ij}$ be the variables in the scope of the clusters on the C_i side, and $W_{<ji}$ be the vars on the C_j side.
- Thm 10.1.8. T satisfies RIP iff for every S_{ij}, $W_{<ij} \perp W_{<ji} \mid S_{ij}$.
- Hence msg from C_i to C_j is sufficient statistic for all info to left of C_i – C_j.
- RIP ensures local communication => global consistency.

Clique trees

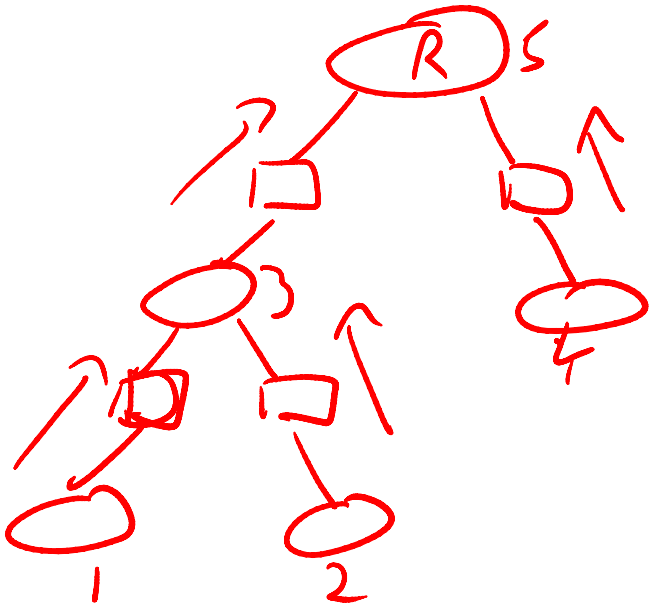
- Def 10.1.7. A cluster tree that satisfies RIP is called a clique tree or join tree or junction tree.
- Thm 4.5.15. A graph has a Jtree (where the clusters are the maxcliques) iff it is chordal.
- Thm 10.4.1. We can always remove non maximal cliques from a Jtree without violating RIP.





Message passing on a clique tree

- To compute p(X_i), find a clique that contains X_i, make it the root, and send messages to it from all other nodes.
- A clique cannot send a node to its parent until it is ready, ie. Has received msgs from all its children.
- Hence we send from leaves to root.

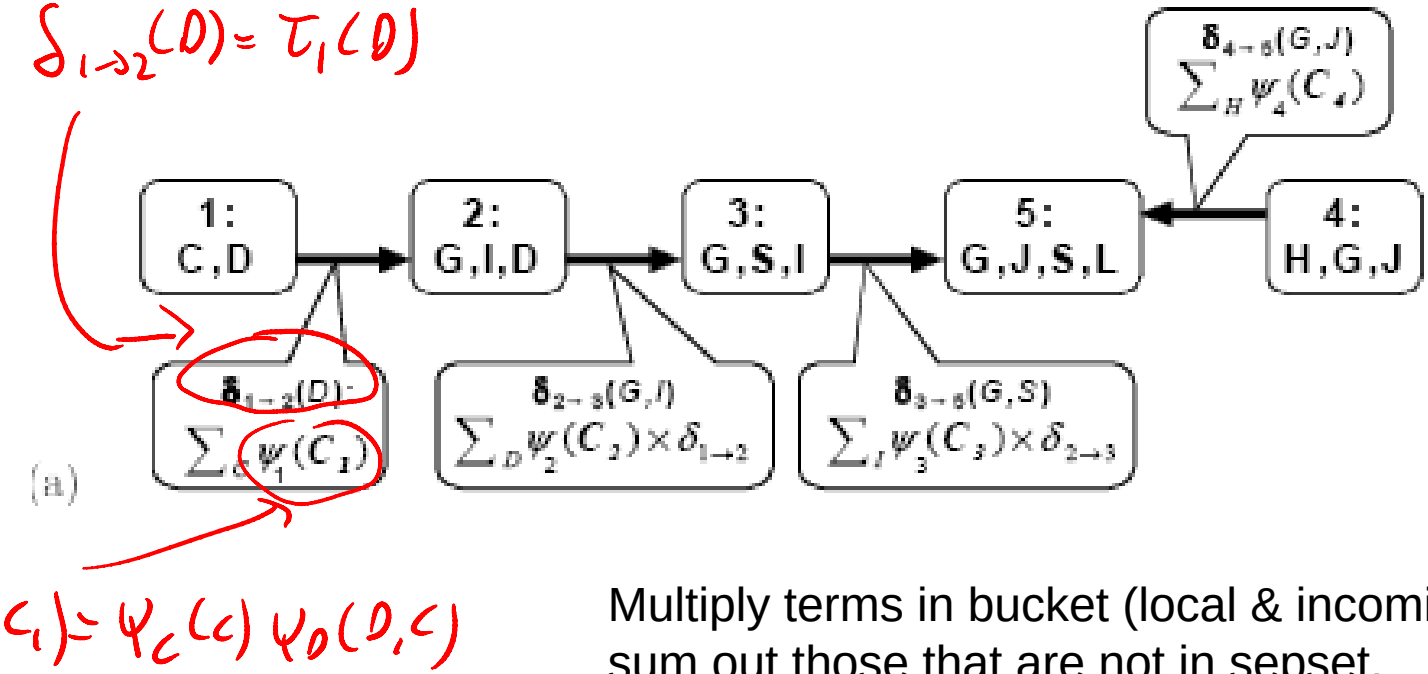


Message passing on a clique tree

- .

$$P(J) = \sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \sum_D \psi_G(G,I,D) \underbrace{\sum_C \psi_C(C)\psi_D(D,C)}_{\tau_1(D)}$$

$$= \sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \underbrace{\sum_D \psi_G(G,I,D)\tau_1(D)}_{\tau_2(G,I)}$$



Multiply terms in bucket (local & incoming), sum out those that are not in sepset, send to nbr upstream

Upwards pass (collect to root)

Procedure CTree Sum Product Up (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  C_r // Some selected root clique
)

1 Initialize Cliques
2 **while** C_r is not ready
3 Let C_i be a ready clique
4 $\delta_{i \to p_r(i)}(S_{i,p_r(i)}) \leftarrow$ SP Message$(i, p_r(i))$
5 $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \text{Nb}_{C_r}} \delta_{k \to r}$
6 **return** β_r

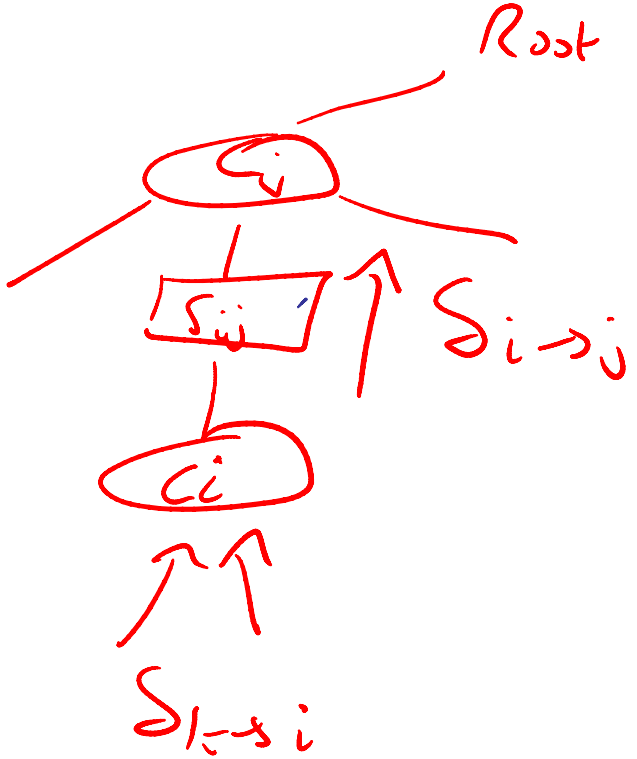
Procedure Initialize Cliques (

)

1 **for** each clique C_i
2 $\psi_i[C_i] \leftarrow \prod_{\phi_j : \alpha(\phi_j)=i} \phi$
3

Procedure SP Message (
  i, // sending clique
  j // receiving clique
)

1 $\psi(C_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
2 $\tau(S_{i,j}) \leftarrow \sum_{C_i - S_{i,j}} \psi(C_i)$
3 **return** $\tau(S_{i,j})$

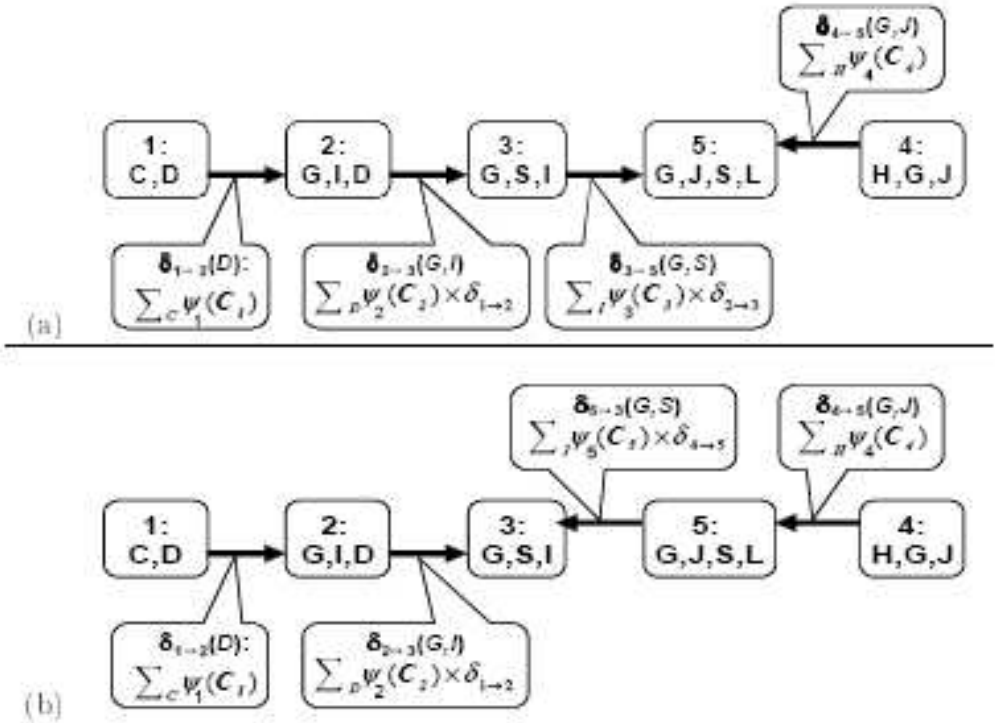


$$\beta_i(C_i) = \phi_i(C_i) \prod_{k \in n_i, k \neq j} \delta_{k \to i}(S_{k,i})$$

$$\delta_{i \to j}(S_{ij}) = \sum_{C_i \setminus S_{ij}} \beta_i(C_i)$$

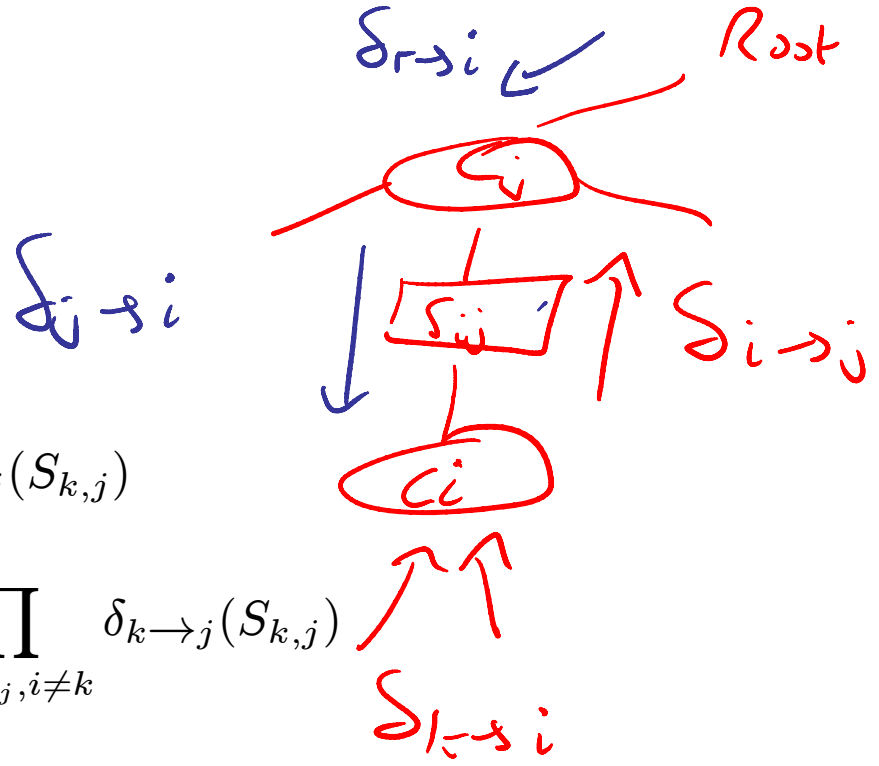
Message passing to a different root

- If we send messages to a different root, many of them will be the same
- Hence if we send messages to all the cliques, we can reuse the messages- dynamic programming!



Downwards pass (distribute from root)

- At the end of the upwards pass, the root has seen all the evidence.
- We send back down from root to leaves.



$$\beta_j(C_j) = \phi_j(C_j) \prod_{k \in n_j} \delta_{k \to j}(S_{k,j})$$

$$\delta_{j \to i}(S_{ij}) = \sum_{C_j \setminus S_{ij}} \phi_j(C_j) \prod_{k \in n_j, i \neq k} \delta_{k \to j}(S_{k,j})$$

$$= \sum_{C_j \setminus S_{ij}} \frac{\beta_j(C_j)}{\delta_{i \to j}(S_{ij})}$$

Use division operator to avoid double counting

Beliefs

- Thm 10.2.7. After collect/distribute, each clique potential represents a marginal probability (conditioned on the evidence)

$$\beta_i(C_i) = \sum_{\mathbf{x}\, C_i} \tilde{P}(\mathbf{x})$$

- If we get new evidence on X_i, we can multiply it in to any clique containing i, and then distribute messages outwards from that clique to restore consistency.

MAP configuration

- We can generalize the Viterbi algorithm to find a MAP configuration as follows.
- On the upwards pass, replace sum with max.
- At the root, find the most probable joint setting and send this as evidence to the root's children.
- Each child finds its most probable setting and sends this to its children.
- The jtree property ensures that when the state of a variable is fixed in one clique, that variable assumes the same state in all other cliques.

Samples

- We can generalize forwards-filtering backwards-sampling to draw exact samples from the joint as follows.
- Do a collect pass to the root as usual.
- Sample xR from the root marginal, and then enter it as evidence in all the children.
- Each child then samples itself from its updated local distribution and sends this to its children.

Calibrated clique tree

- Def 102.8. A clique tree is calibrated if, for all pairs of neighboring cliques, we have

$$\sum_{C_i \setminus S_{i,j}} \beta_i(C_i) = \sum_{C_j \setminus S_{i,j}} \beta_j(C_j) = \mu_{i,j}(S_{i,j})$$

- Eg. A-B-C clq tree AB – [B] – BC. We require

$$\sum_a \beta_{ab}(a,b) = \sum_c \beta_{bc}(b,c)$$

- Thm. After collect/distribute, all cliques are calibrated.
- Thm 10.2.12. A calibrated tree defines a joint distribution as follows $p(x) = \frac{\prod_i \beta_i(C_i)}{\prod_{<ij>} \mu_{i,j}(S_{ij})}$

eg $p(A,B,C) = \frac{p(A,B)p(B,C)}{p(C)} = p(A,B)p(C|B) = p(A|B)p(B,C)$

Clique tree invariant

- Suppose at every step, clique i sends a msg to clique j, and stores it in $\mu_{i,j}$:

Procedure Send-BU-Msg (
  i, // sending clique
  j // receiving clique
)

1 $\sigma_{i \to j} \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i$
2 // marginalize the clique over the sepset
3 $\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i \to j}}{\mu_{i,j}}$
4 $\mu_{i,j} \leftarrow \sigma_{i \to j}$

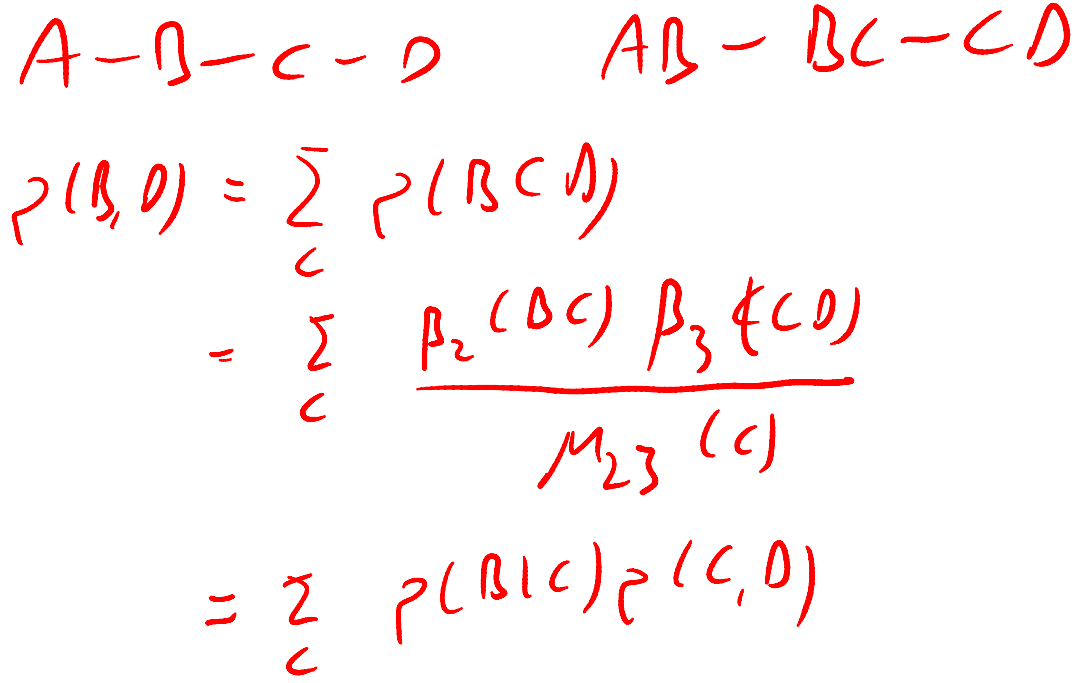
- Initially $\mu_{i,j}=1$ and $\beta_i = \prod_{f:\ f \text{ ass to } i} \phi_f$. Hence the following holds.

$$p(x) = \frac{\prod_i \beta_i(C_i)}{\prod_{<ij>} \mu_{i,j}(S_{ij})}$$

- Thm 10.3.4. This property holds after every belief updating operation.

Out of clique queries

- We can compute the distribution on any set of variables inside a clique. But suppose we want the joint on variables in different cliques. We can run VE on the calibrated subtree
- eg

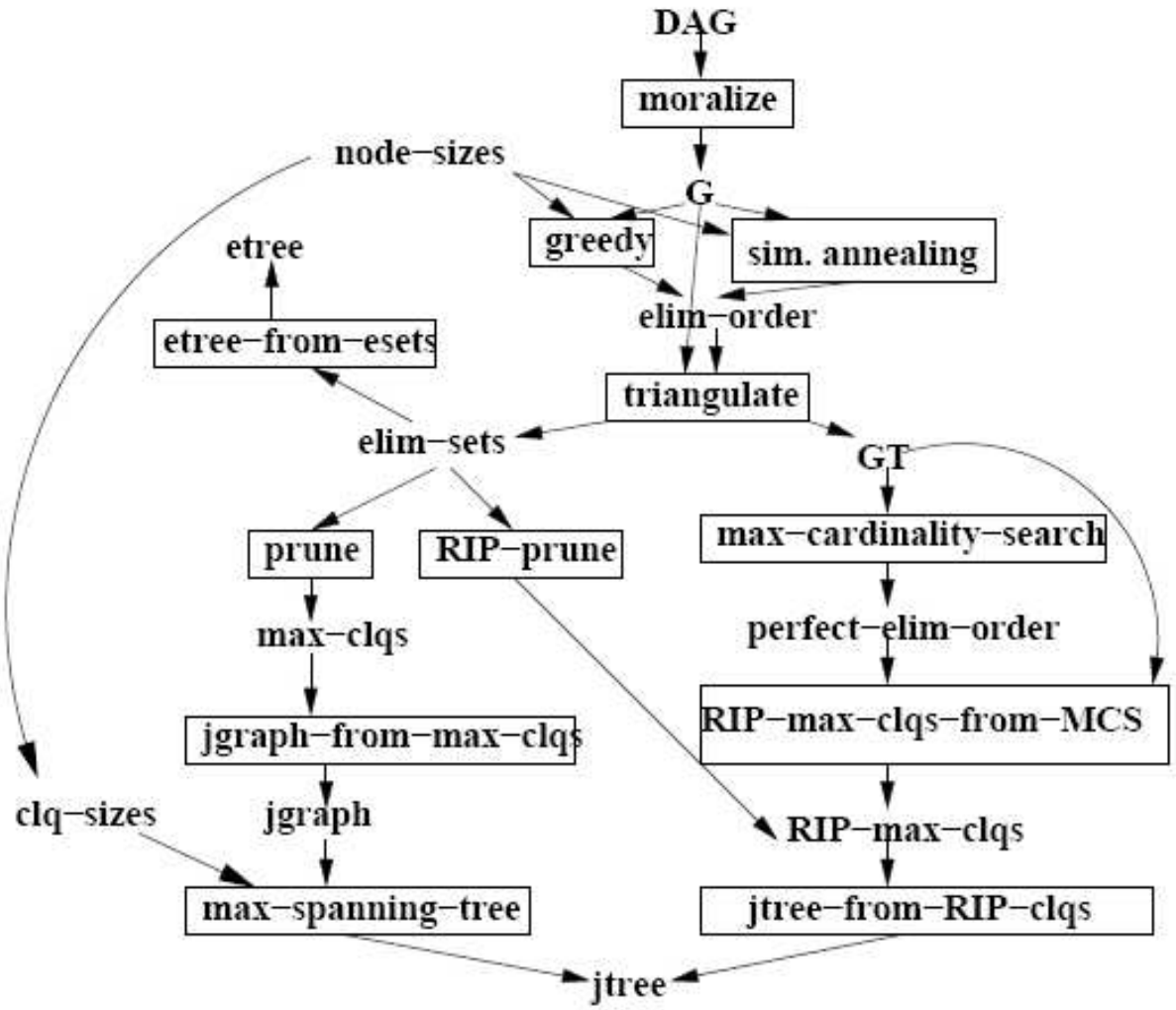


Out of clique inference

Procedure CTree-Query (
  $\mathcal{T}$, // Clique tree over Φ
  $\{\beta_i\}, \{\mu_{i,j}\}$, // Calibrated clique and sepset beliefs for $\mathcal{T}$
  $\mathbf{Y}$ // A query
)

Let $\mathcal{T}'$ be a subtree of $\mathcal{T}$ such that $\mathbf{Y} \subseteq Scope[\mathcal{T}']$
Select a clique $r \in \mathcal{V}_{\mathcal{T}'}$ to be the root
$\Phi \leftarrow \beta_r$
for each $i \in \mathcal{V}_{\mathcal{T}}$
  $\phi \leftarrow \frac{\beta_i}{\mu_{i,Pr(i)}}$
  $\Phi \leftarrow \Phi \cup \{\phi\}$
$\mathbf{Z} \leftarrow Scope[\mathcal{T}'] - \mathbf{Y}$
Let $\prec$ be some ordering over $\mathbf{Z}$
return Sum-Product-Variable-Elimination$(\Phi, \mathbf{Z}, \prec)$

Creating a Jtree



Murphy PhD thesis (2002) p140

Max cliques from a chordal graph

- Triangulate the graph according to some ordering.
  - Start with all vertices unnumbered, set counter $i := N$.
  - While there are still some unnumbered vertices:
    - Let $v_i = \pi(i)$.
    - Form the set C_i consisting of v_i and its (unnumbered/ uneliminated) neighbors.
    - Fill in edges between all pairs of vertices in C_i.
    - Eliminate v_i and decrement i by 1.
- At each step, keep track of the clique that is created; if it is a subset of any previously created clique, discard it (since non maximal).

Cliques to Jtree

- Build a weighted graph where
  $W_{ij} = |C_i \text{ intersect } C_j|$
- Find max weight spanning tree. This is a jtree.

Stat 521A
Lecture 8

Outline

- Forwards backwards on chains
- FB on trees
- FB on clique chains
- FB on clique trees
- Message passing on clique trees (10.2-10.3)
- Creating clique trees (10.4)

Forwards algorithm

1. predict: compute the the **one-step-ahead predictive density** $p(S_t|\mathbf{x}_{1:t-1})$ as follows:

$$p(S_t = j|\mathbf{x}_{1:t-1}) = \sum_i p(S_t = j, S_{t-1} = i|\mathbf{x}_{1:t-1}) \quad (1)$$

$$= \sum_i p(S_t = j|S_{t-1} = i)p(S_{t-1} = i|\mathbf{x}_{1:t-1}) \quad (2)$$

   In the second step we used the fact that $S_t \perp X_{1:t-1}|S_{t-1}$.

2. update: compute $p(S_t|\mathbf{x}_t, \mathbf{x}_{1:t-1})$ using Bayes rule, where we use $p(S_t|\mathbf{x}_{1:t-1})$ as the prior:

$$p(S_t = j|\mathbf{x}_{1:t}) = \frac{1}{c_t} p(\mathbf{x}_t|S_t = j)p(S_t = j|\mathbf{x}_{1:t-1}) \quad (3)$$

   where we used the fact that $X_t \perp X_{1:t-1}|S_t$. The normalizing constant c_t is given by

$$c_t = p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = \sum_j p(\mathbf{x}_t|S_t = j)p(S_t = j|\mathbf{x}_{1:t-1}) \quad (4)$$

The base case is

$$p(S_1 = j|\mathbf{x}_1) \propto p(S_1 = j)p(\mathbf{x}_1|S_1 = j) = \pi_j p(\mathbf{x}_1|S_1 = j) \quad (5)$$

Matrix vector form

$$\alpha_t(j) = p(S_t = j|\mathbf{x}_{1:t}) \quad (1)$$
$$b_t(j) = p(\mathbf{x}_t|S_t = j) \quad (2)$$
$$A(i,j) = p(S_t = j|S_{t-1} = i) \quad (3)$$

Hence the recursion step is

$$\alpha_t(j) \propto b_t(j) \sum_i A_{ij} \alpha_{t-1}(i) \quad (4)$$

This can be rewritten in matrix-vector notation as

$$\boldsymbol{\alpha}_t \propto \text{diag}(\mathbf{b}_t)\mathbf{A}^T \boldsymbol{\alpha}_{t-1} \quad (5)$$

It is somewhat clearer if we use Matlab-style notation, and use $.*$ to denote elementwise multiplication by a vector:

$$\boldsymbol{\alpha}_t \propto \mathbf{b}_t .* (\mathbf{A}^T \boldsymbol{\alpha}_{t-1}) \quad (6)$$

The log-likelihood of the data sequence can be computed from the normalizing constants as follows:

$$\log p(\mathbf{x}_{1:T}) = \sum_{t=1}^{T} \log p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = \sum_{c=1}^{T} \log c_t \quad (7)$$

Matlab

Listing 1: Listing of `hmmFilter`

```
function [alpha, loglik] = hmmFilter(initDist, transmat, obslik)
% initDist(i) = Pr(Q(1) = i)
% transmat(i,j) = Pr(Q(t) = j | Q(t-1)=i)
% obslik(i,t) = Pr(Y(t)| Q(t)=i)
[K T] = size(obslik);
alpha = zeros(K,T);
[alpha(:,1), scale(1)] = normalize(initDist(:) .* obslik(:,1));
for t=2:T
  [alpha(:,t), scale(t)] = normalize((transmat' * alpha(:,t-1)) .* obslik(:,t));
end
loglik = sum(log(scale+eps));
```

Listing 2: Listing of `makeLocalEvidence`

```
function localEvidence = makeLocalEvidence(model,obs)
% localEvidence(i,t) = p(Y(t) | Z(t)=i)
localEvidence = zeros(model.nstates,size(obs,2));
for i = 1:model.nstates
 localEvidence(i,:) = exp(logprob(model.emissionDist{i},obs'));
end
```

Offline estimation: goals

- Single slice marginals:

$$\gamma_t(j) \stackrel{\text{def}}{=} p(S_t = j|\mathbf{x}_{1:T}, \boldsymbol{\theta}) \tag{1}$$

  for all $1 \leq t \leq T$. This can be computed via the **forwards backwards** algorithm, as we discuss in Section **??**.

- Two-slice marginals

$$\xi_{t-1,t}(i,j) \stackrel{\text{def}}{=} p(S_{t-1} = i, S_t = j|\mathbf{x}_{1:T}, \boldsymbol{\theta}) \tag{2}$$

  These are needed for parameter estimation, as described in Section **??**. These quantities are easy to compute using forwards-backwards, as we describe in Section **??**.

- The posterior mode, or most probable path:

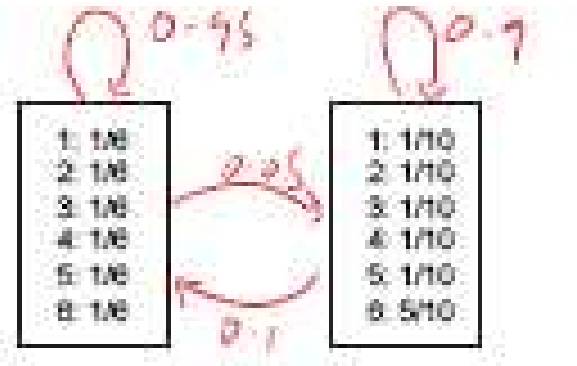
$$\mathbf{s}^*_{1:T} = \arg\max_{\mathbf{s}_{1:T}} p(\mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta}) \tag{3}$$

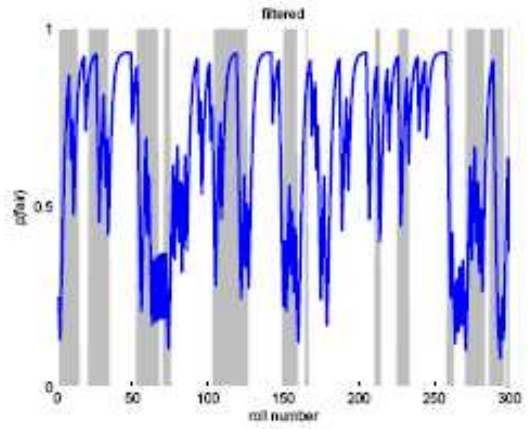
  This can be computed by the **Viterbi algorithm**, as we describe in Section **??**.

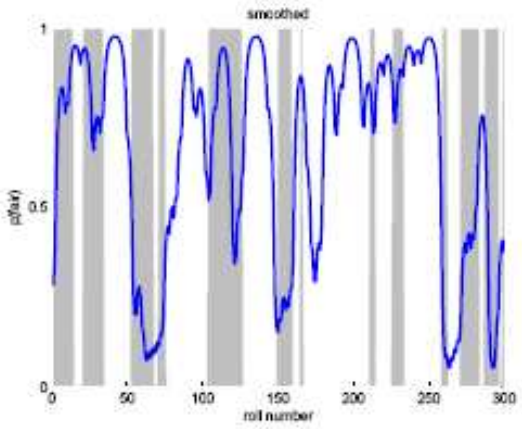
- Samples from the posterior

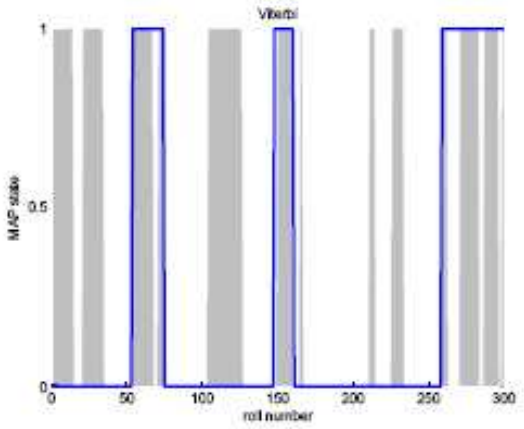
$$\mathbf{s}_{1:T} \sim p(\mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta}) \tag{4}$$

Filtering vs smoothing vs Viterbi

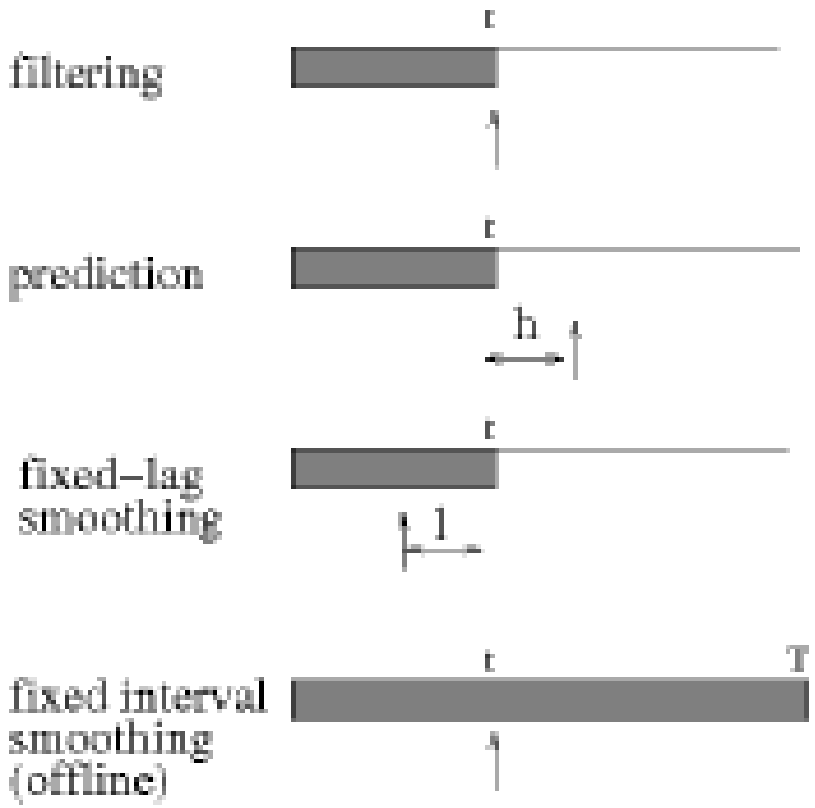






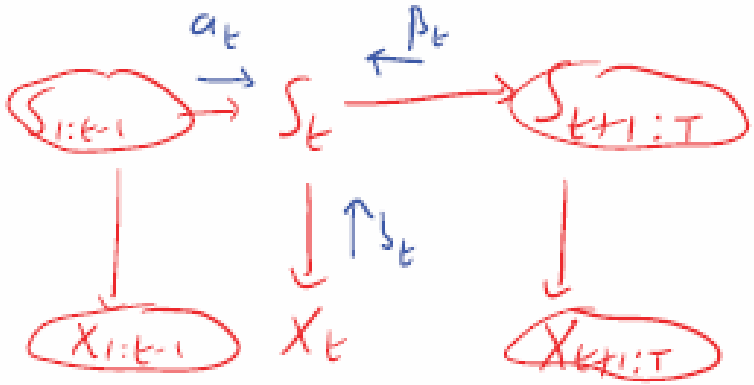


Fixed lag smoothing



nds of inference for state-space models. The shaded region is the interval for

FB



$$p(S_t|\mathbf{x}_{1:T}) \propto \sum_{\mathbf{s}_{1:t-1}} \sum_{\mathbf{s}_{t+1:T}} p(\mathbf{s}_{1:t-1}, \mathbf{x}_{1:t-1}, S_t, \mathbf{x}_t, \mathbf{s}_{t+1:T}, \mathbf{x}_{t+1:T}) \quad (1)$$

$$= \sum_{\mathbf{s}_{1:t-1}} \sum_{\mathbf{s}_{t+1:T}} p(\mathbf{s}_{1:t-1}, \mathbf{x}_{1:t-1}) p(S_t|s_{t-1}) p(\mathbf{x}_t|S_t) p(\mathbf{s}_{t+1:T}, \mathbf{x}_{t+1:T}|S_t) \quad (2)$$

$$= \sum_{s_{t-1}} p(s_{t-1}, \mathbf{x}_{1:t-1}) p(S_t|s_{t-1}) p(\mathbf{x}_t|S_t) p(\mathbf{x}_{t+1:T}|S_t) \quad (3)$$

$$\propto \sum_{s_{t-1}} p(s_{t-1}|\mathbf{x}_{1:t-1}) p(S_t|s_{t-1}) p(\mathbf{x}_t|S_t) p(\mathbf{x}_{t+1:T}|S_t) \quad (4)$$

Matrix vector form

Let us define the following notation

$$\alpha_t(j) \stackrel{\text{def}}{=} p(S_t = j|\mathbf{x}_{1:t}) \quad (1)$$

$$\beta_t(j) \stackrel{\text{def}}{=} p(\mathbf{x}_{t+1:T}|S_t = j) \quad (2)$$

$$\gamma_t(j) \stackrel{\text{def}}{=} p(S_t = j|\mathbf{x}_{1:T}) \quad (3)$$

Then we can rewrite the above equation as

$$\gamma_t(j) \propto \sum_i \alpha_{t-1}(i) A_{ij} b_t(j) \beta_t(j) \quad (4)$$

Furthermore, let us define the one-step ahead predictive density

$$a_t(j) \stackrel{\text{def}}{=} p(S_t = j|\mathbf{x}_{1:t-1}) = \sum_i \alpha_{t-1}(i) A_{ij} \quad (5)$$

Then we can rewrite the above equation as

$$\gamma_t(j) \propto a_t(j) b_t(j) \beta_t(j) \quad (6)$$

Backwards algorithm

$$
\begin{aligned}
\beta_{t-1}(i) &= p(\mathbf{x}_{t+1:T}|S_{t-1}=i) && (1)\\
&= \sum_j p(S_t=j, \mathbf{x}_t, \mathbf{x}_{t+1:T}|S_{t-1}=i) && (2)\\
&= \sum_j p(S_t=j|S_{t-1}=i)p(\mathbf{x}_t|S_t=j, S_{t-1}=i)p(\mathbf{x}_{t+1:T}|S_t=j, S_{t-1}=i) && (3)\\
&= \sum_j p(S_t=j|S_{t-1}=i)p(\mathbf{x}_t|S_t=j)p(\mathbf{x}_{t+1:T}|S_t=j) && (4)\\
&= \sum_j A_{ij} b_t(j)\beta_t(j) && (5)
\end{aligned}
$$

where Equation **??** is justi┐ed since $X_t \perp X_{t+1:T}|S_t$ and Equation **??** is justi┐ed since $X_t \perp S_{t-1}|S_t$ and $X_{t+1:T} \perp S_{t-1}|S_t$. We can write the resulting equation in matrix-vector form as

$$\boldsymbol{\beta}_{t-1} = \mathbf{A}(\mathbf{b}_t . * \boldsymbol{\beta}_t) \quad (6)$$

The base case is

$$\beta_T(i) = p(\mathbf{x}_{T+1:T}|S_T=i) = p(\emptyset|S_T=i) = 1 \quad (7)$$

Matlab

Listing 1: Listing of `hmmBackwards`

```
function [beta] = hmmBackwards(transmat, obslik)
% beta(i,t) propto p(y(t+1:T) | Q(t)=i)
[K T] = size(obslik);
beta = zeros(K,T);
beta(:,T) = ones(K,1);
for t=T-1:-1:1
 beta(:,t) = normalize(transmat * (beta(:,t+1) .* obslik(:,t+1)));
end
\end{codeCap

\begin{codeCap}{Listing of \codename{hmmFwdBack}}
function [gamma, alpha, beta, loglik] = hmmFwdBack(initDist, transmat, obslik)
% gamma(i,t) = p(Q(t)=i | y(1:T))
[alpha, loglik] = hmmFilter(initDist, transmat, obslik);
beta = hmmBackwards(transmat, obslik);
gamma = normalize(alpha .* beta, 1);% make each colum sum to 1
```

Avoiding underflow

$$\alpha_t(j) = p(S_t = j|\mathbf{x}_{1:T}) = \frac{1}{c_t} b_t(j) \sum_i A_{ij} \alpha_{t-1}(i) \quad (1)$$

$$c_t = \sum_j b_t(j) \sum_i A_{ij} \alpha_{t-1}(i) \quad (2)$$

$$\hat{\beta}_{t-1}(i) = \frac{1}{d_{t-1}} \sum_j A_{ij} b_t(j) \hat{\beta}_t(j) \quad (3)$$

$$d_{t-1} = \sum_i A_{ij} b_t(j) \hat{\beta}_t(j) \quad (4)$$

$$p(S_t = j, \mathbf{x}_{1:t}) = p(S_t = j|\mathbf{x}_{1:t}) p(\mathbf{x}_{1:t}) = \alpha_t(j) (\prod_{\tau=1}^{t} c_\tau) \quad (5)$$

$$p(\mathbf{x}_{t+1:T}|S_t = j) = \hat{\beta}_t(j) (\prod_{\tau=t}^{T} d_\tau) \quad (6)$$

Avoiding underflow

$$\gamma_t(j) = p(S_t = j|x_{1:T}) \quad (1)$$

$$= \frac{p(x_{t+1:T}|S_t = j)p(S_t = j, x_{1:t})}{p(x_{1:T})} \quad (2)$$

$$= \frac{(\prod_{\tau=t}^{T} d_\tau)\hat{\beta}_t(j)(\prod_{\tau=1}^{t} c_\tau)\alpha_t(j)}{\sum_{j'}(\prod_{\tau=t}^{T} d_\tau)\hat{\beta}_t(j')(\prod_{\tau=1}^{t} c_\tau)\alpha_t(j')} \quad (3)$$

$$= \frac{\beta_t(j)\alpha_t(j)}{\sum_{j'} \hat{\beta}_t(j')\alpha_t(j')} \quad (4)$$

Two-slice marginals

$$N_{ij} = \sum_{t=1}^{T-1} E[I(S_t = i, S_{t+1} = j)|\mathbf{x}_{1:T}] = \sum_{t=1}^{T-1} p(S_t = i, S_{t+1} = j|\mathbf{x}_{1:T}) \quad (1)$$

$$\begin{aligned}
\xi_{t-1,t}(i,j) &\stackrel{\text{def}}{=} p(S_{t-1} = i, S_t = j|x_{1:T}) \\
&\propto p(S_{t-1} = i|\mathbf{x}_{1:t-2})p(\mathbf{x}_{t-1}|S_{t-1} = i)p(S_t = j|S_{t-1} = i)p(\mathbf{x}_t|S_t = j)p(\mathbf{x}_{t+1:T}|S_t = j) \\
&= a_{t-1}(i)b_{t-1}(i)A_{ij}b_t(j)\beta_t(j)
\end{aligned}$$

$$\boldsymbol{\xi}_{t-1,t} \propto \mathbf{A}. * (\boldsymbol{\alpha}_{t-1} * (\mathbf{b}_t. * \boldsymbol{\beta}_t)^T) \quad (2)$$

Time and space complexity

- O(T K b) time, b = branching factor
- In discretization of cts space,
  O(T K log K) or O(T K) – Felzenswalb & Huttenlocher
- O(T K) space, O(T K^2) time
- O(K log T) space, O(T log T K^2) time (island algorithm)

Viterbi

MAP path

$$s^*_{1:T} = \arg\max_{s_{1:T}} p(s_{1:T}|x_{1:T}) \tag{1}$$

Max marginals

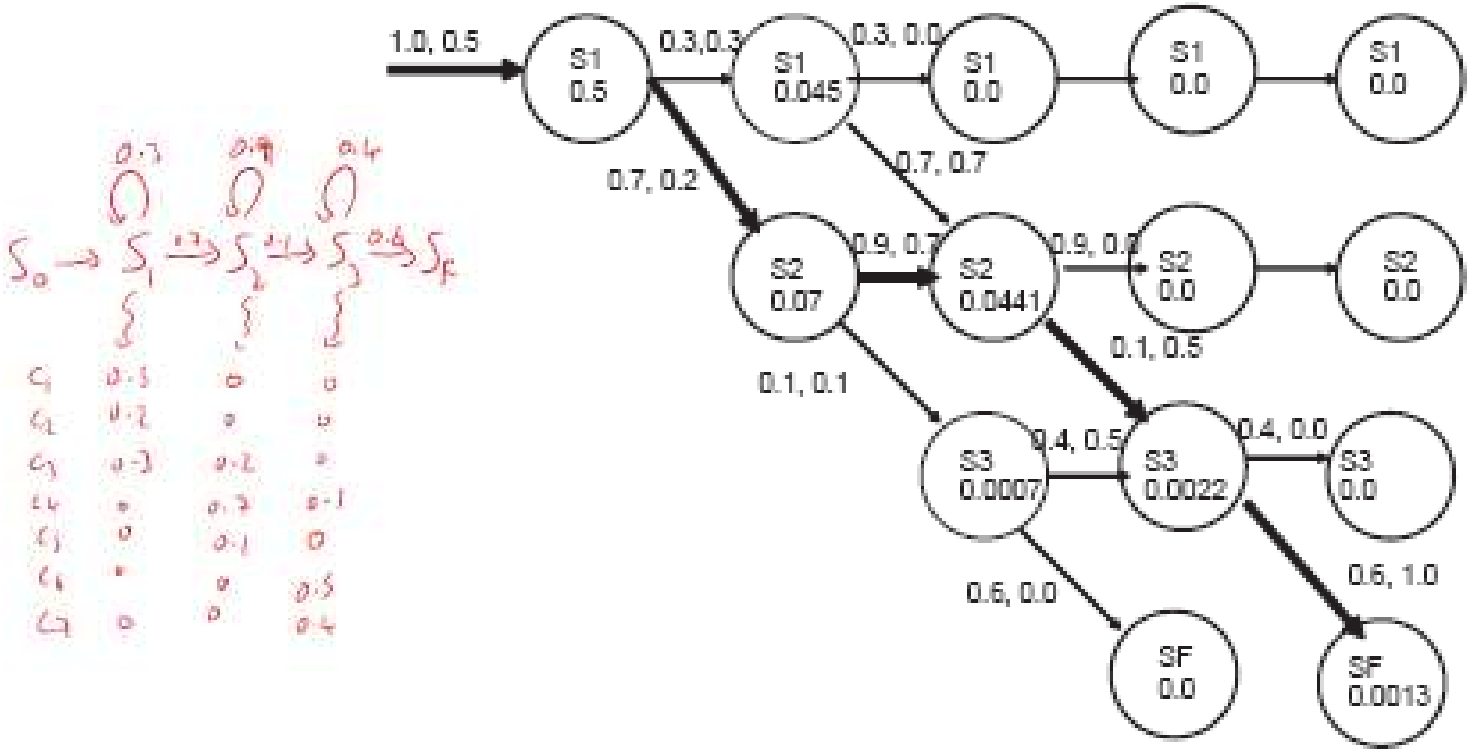
$$s^*_t = \arg\max_i p(S_t = i|\mathbf{x}_{1:T}) = \arg\max_i \sum_{\mathbf{s}_{-t}} p(S_t = i, \mathbf{s}_{-t}|\mathbf{x}_{1:T}) \tag{2}$$

$$\begin{aligned}
\delta_t(i) &\stackrel{\text{def}}{=} \max_{s_1,\ldots,s_{t-1}} p(\mathbf{s}_{1:t-1}, s_t = i, \mathbf{x}_{1:t}|\boldsymbol{\theta}) \\
\delta_{t+1}(j) &= \max_i \delta_t(i) A_{ij} b_{t+1}(j) \\
\psi_{t+1}(j) &= \arg\max_i \delta_t(i) A_{ij} b_{t+1}(j) \\
\delta_1(j) &= \pi_j b_1(j)
\end{aligned}$$

Traceback

$$\begin{aligned}
S^*_T &= \arg\max_i \delta_T(i) \\
S^*_t &= \psi_{t+1}(s^*_{t+1})
\end{aligned}$$

Viterbi example



$$\delta_1(1) = 0.5$$
$$\delta_2(1) = \delta_1(1)A_{11}b_2(1) = 0.5 \cdot 0.3 \cdot 0.3 = 0.045$$
$$\delta_2(2) = \delta_1(1)A_{12}b_2(2) = 0.5 \cdot 0.7 \cdot 0.2 = 0.07$$

Top N list
Discrim. reranking

Fwd filtering, back sampling

$$s^*_{1:T} \sim p(\mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\theta}) \quad (1)$$

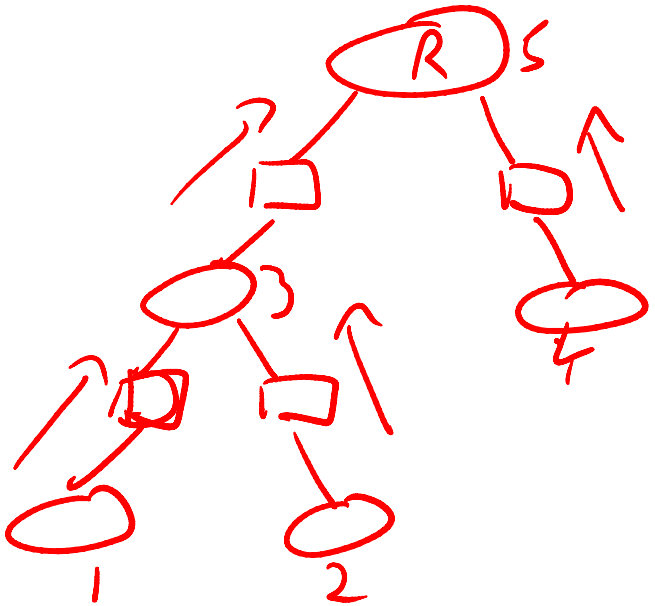
$$\begin{aligned}
s^*_t &\sim p(S_t|s^*_{t+1:T}, \mathbf{x}_{1:T}) && (2)\\
&\propto p(S_t|s^*_{t+1}, \mathbf{x}_{1:t}) && (3)\\
p(S_t = i|S_{t+1} = j, x_{1:t}) &= p(S_t = i|S_{t+1} = j, x_{1:t}, x_{t+1}) && (4)\\
&= \frac{p(S_t = i, S_{t+1} = j|x_{1:t+1})}{p(S_{t+1} = j|x_{1:t+1})} && (5)\\
&= \frac{p(\mathbf{x}_t|S_t = j)p(S_t = j|S_{t-1} = i)p(S_{t-1} = i|\mathbf{x}_{1:t-1})}{p(S_{t+1} = j|x_{1:t+1})} && (6)\\
&= \frac{A_{ij}\alpha_t(i)b_{t+1}(j)}{\alpha_{t+1}(j)} && (7)
\end{aligned}$$

Listing 1: Listing of `hmmSamplePost`

```
function [samples] = hmmSamplePost(initDist, transmat, obslik, nsamples)
% samples(t,s) = value of S(t) in sample s
[K T] = size(obslik);
alpha = hmmFilter(initDist, transmat, obslik);
samples = zeros(T, nsamples);
dist = normalize(alpha(:,T));
samples(T,:) = sample(dist, nsamples);
for t=T-1:-1:1
  tmp = obslik(:,t+1) ./ (alpha(:,t+1)+eps); % b_{t+1}(j) / alpha_{t+1}(j)
  xi_filtered = transmat .* (alpha(:,t) * tmp');
  for n=1:nsamples
    dist = xi_filtered(:,samples(t+1,n));
    samples(t,n) = sample(dist);
  end
end
```

Message passing on a clique tree

- To compute p(X_i), find a clique that contains X_i, make it the root, and send messages to it from all other nodes.
- A clique cannot send a node to its parent until it is ready, ie. Has received msgs from all its children.
- Hence we send from leaves to root.

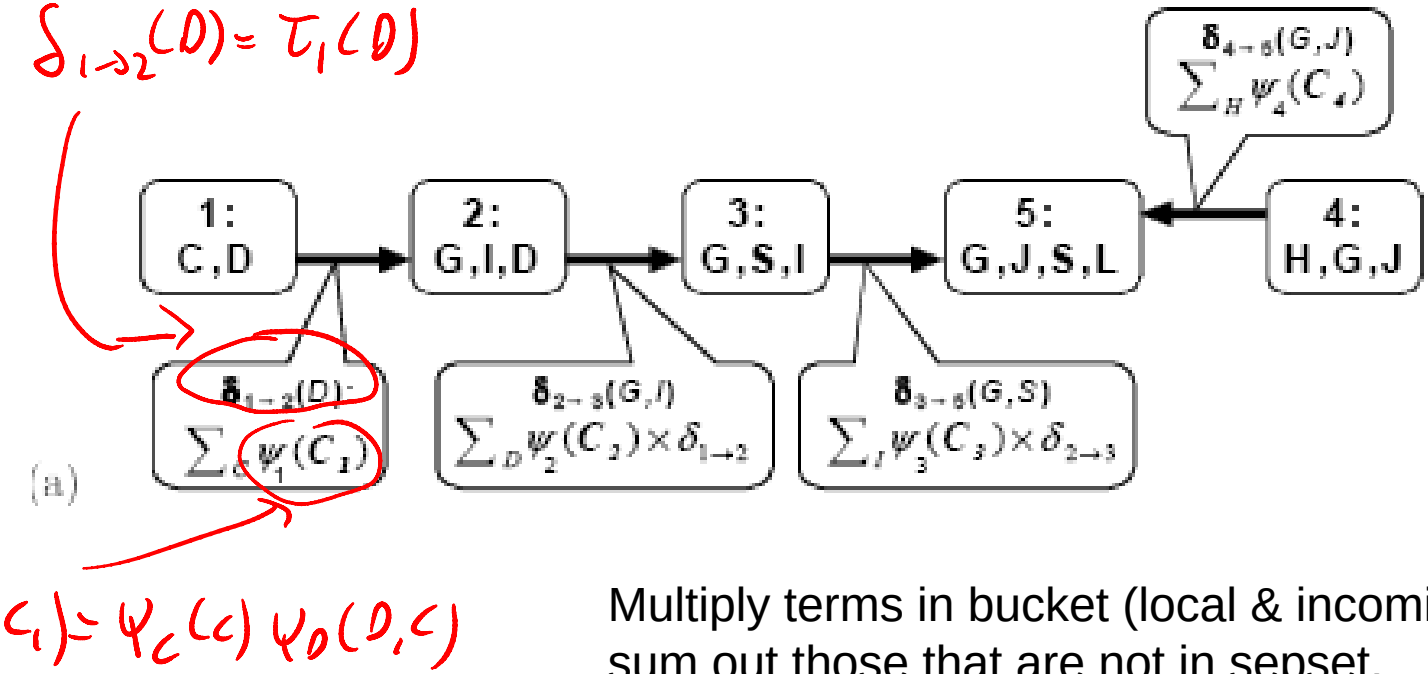


Message passing on a clique tree

- .

$$P(J) = \sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \sum_D \psi_G(G,I,D) \underbrace{\sum_C \psi_C(C)\psi_D(D,C)}_{\tau_1(D)}$$

$$= \sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \underbrace{\sum_D \psi_G(G,I,D)\tau_1(D)}_{\tau_2(G,I)}$$



Multiply terms in bucket (local & incoming), sum out those that are not in sepset, send to nbr upstream

Upwards pass (collect to root)

Procedure CTree Sum Product Up (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  C_r // Some selected root clique
)

1 Initialize Cliques
2 **while** C_r is not ready
3 Let C_i be a ready clique
4 $\delta_{i \to p_r(i)}(S_{i,p_r(i)}) \leftarrow$ SP Message$(i, p_r(i))$
5 $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \text{Nb}_{C_r}} \delta_{k \to r}$
6 **return** β_r

Procedure Initialize Cliques (

)

1 **for** each clique C_i
2 $\psi_i[C_i] \leftarrow \prod_{\phi_j : \alpha(\phi_j) = i} \phi$
3

Procedure SP Message (
  i, // sending clique
  j // receiving clique
)

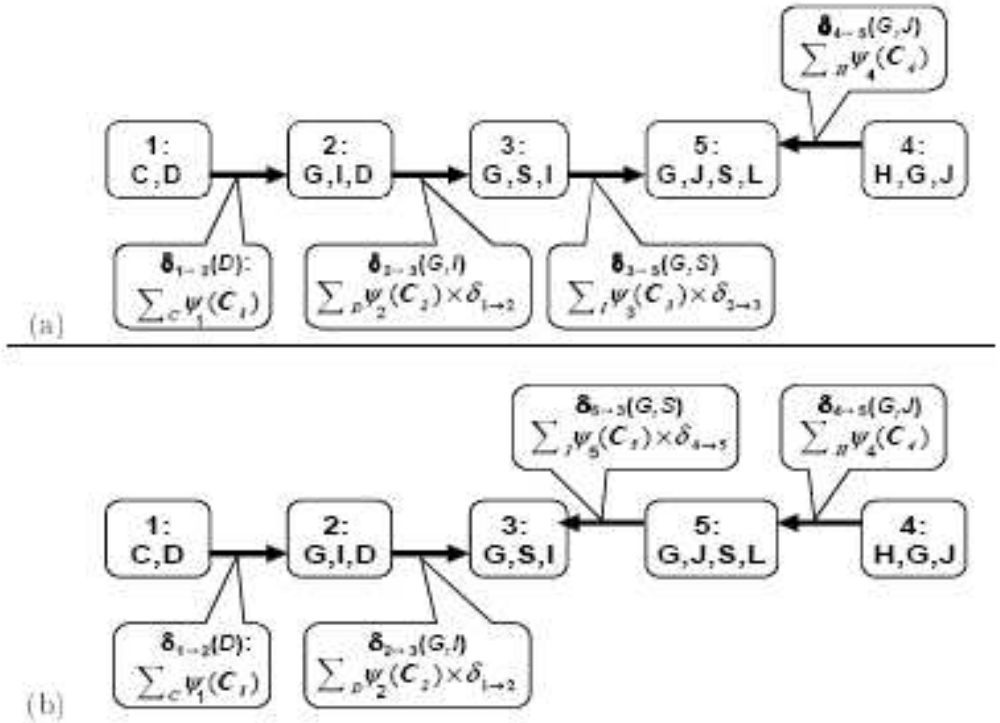
1 $\psi(C_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
2 $\tau(S_{i,j}) \leftarrow \sum_{C_i - S_{i,j}} \psi(C_i)$
3 **return** $\tau(S_{i,j})$

$$\beta_i(C_i) = \phi_i(C_i) \prod_{k \in n_i, k \neq j} \delta_{k \to i}(S_{k,i})$$

$$\delta_{i \to j}(S_{ij}) = \sum_{C_i \setminus S_{ij}} \beta_i(C_i)$$

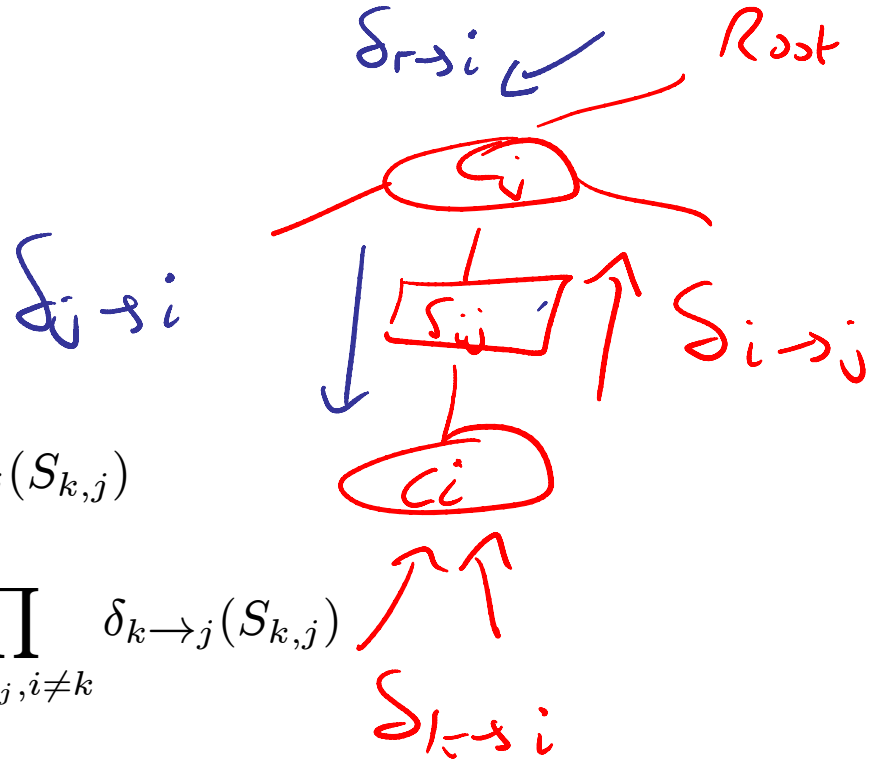
Message passing to a different root

- If we send messages to a different root, many of them will be the same
- Hence if we send messages to all the cliques, we can reuse the messages- dynamic programming!



Downwards pass (distribute from root)

- At the end of the upwards pass, the root has seen all the evidence.
- We send back down from root to leaves.



$$\beta_j(C_j) = \phi_j(C_j) \prod_{k \in n_j} \delta_{k \to j}(S_{k,j})$$

$$\delta_{j \to i}(S_{ij}) = \sum_{C_j \setminus S_{ij}} \phi_j(C_j) \prod_{k \in n_j, i \neq k} \delta_{k \to j}(S_{k,j})$$

$$= \sum_{C_j \setminus S_{ij}} \frac{\beta_j(C_j)}{\delta_{i \to j}(S_{ij})}$$

Use division operator to avoid double counting

Beliefs

- Thm 10.2.7. After collect/distribute, each clique potential represents a marginal probability (conditioned on the evidence)

$$\beta_i(C_i) = \sum_{\mathbf{x}\, C_i} \tilde{P}(\mathbf{x})$$

- If we get new evidence on X_i, we can multiply it in to any clique containing i, and then distribute messages outwards from that clique to restore consistency.

MAP configuration

- We can generalize the Viterbi algorithm to find a MAP configuration as follows.
- On the upwards pass, replace sum with max.
- At the root, find the most probable joint setting and send this as evidence to the root's children.
- Each child finds its most probable setting and sends this to its children.
- The jtree property ensures that when the state of a variable is fixed in one clique, that variable assumes the same state in all other cliques.

Samples

- We can generalize forwards-filtering backwards-sampling to draw exact samples from the joint as follows.
- Do a collect pass to the root as usual.
- Sample xR from the root marginal, and then enter it as evidence in all the children.
- Each child then samples itself from its updated local distribution and sends this to its children.

Calibrated clique tree

- Def 102.8. A clique tree is calibrated if, for all pairs of neighboring cliques, we have

$$\sum_{C_i \setminus S_{i,j}} \beta_i(C_i) = \sum_{C_j \setminus S_{i,j}} \beta_j(C_j) = \mu_{i,j}(S_{i,j})$$

- Eg. A-B-C clq tree AB – [B] – BC. We require

$$\sum_a \beta_{ab}(a,b) = \sum_c \beta_{bc}(b,c)$$

- Thm. After collect/distribute, all cliques are calibrated.
- Thm 10.2.12. A calibrated tree defines a joint distribution as follows $p(x) = \frac{\prod_i \beta_i(C_i)}{\prod_{<ij>} \mu_{i,j}(S_{ij})}$

eg $p(A,B,C) = \frac{p(A,B)p(B,C)}{p(C)} = p(A,B)p(C|B) = p(A|B)p(B,C)$

Clique tree invariant

- Suppose at every step, clique i sends a msg to clique j, and stores it in $\mu_{i,j}$:

Procedure Send-BU-Msg (
  i, // sending clique
  j // receiving clique
)

1 $\sigma_{i \to j} \leftarrow \sum_{C_i - S_{i,j}} \beta_i$
2 // marginalize the clique over the sepset
3 $\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i \to j}}{\mu_{i,j}}$
4 $\mu_{i,j} \leftarrow \sigma_{i \to j}$

- Initially $\mu_{i,j}=1$ and $\beta_i = \prod_{f:\ f \text{ ass to } i} \phi_f$. Hence the following holds.

$$p(x) = \frac{\prod_i \beta_i(C_i)}{\prod_{<ij>} \mu_{i,j}(S_{ij})}$$

- Thm 10.3.4. This property holds after every belief updating operation.

Out of clique queries

- We can compute the distribution on any set of variables inside a clique. But suppose we want the joint on variables in different cliques. We can run VE on the calibrated subtree
- eg

$A - B - C - D \qquad AB - BC - CD$

$$P(B,D) = \sum_C P(B,C,D)$$

$$= \sum_C \frac{\beta_2(B,C)\,\beta_3(C,D)}{\mu_{23}(C)}$$

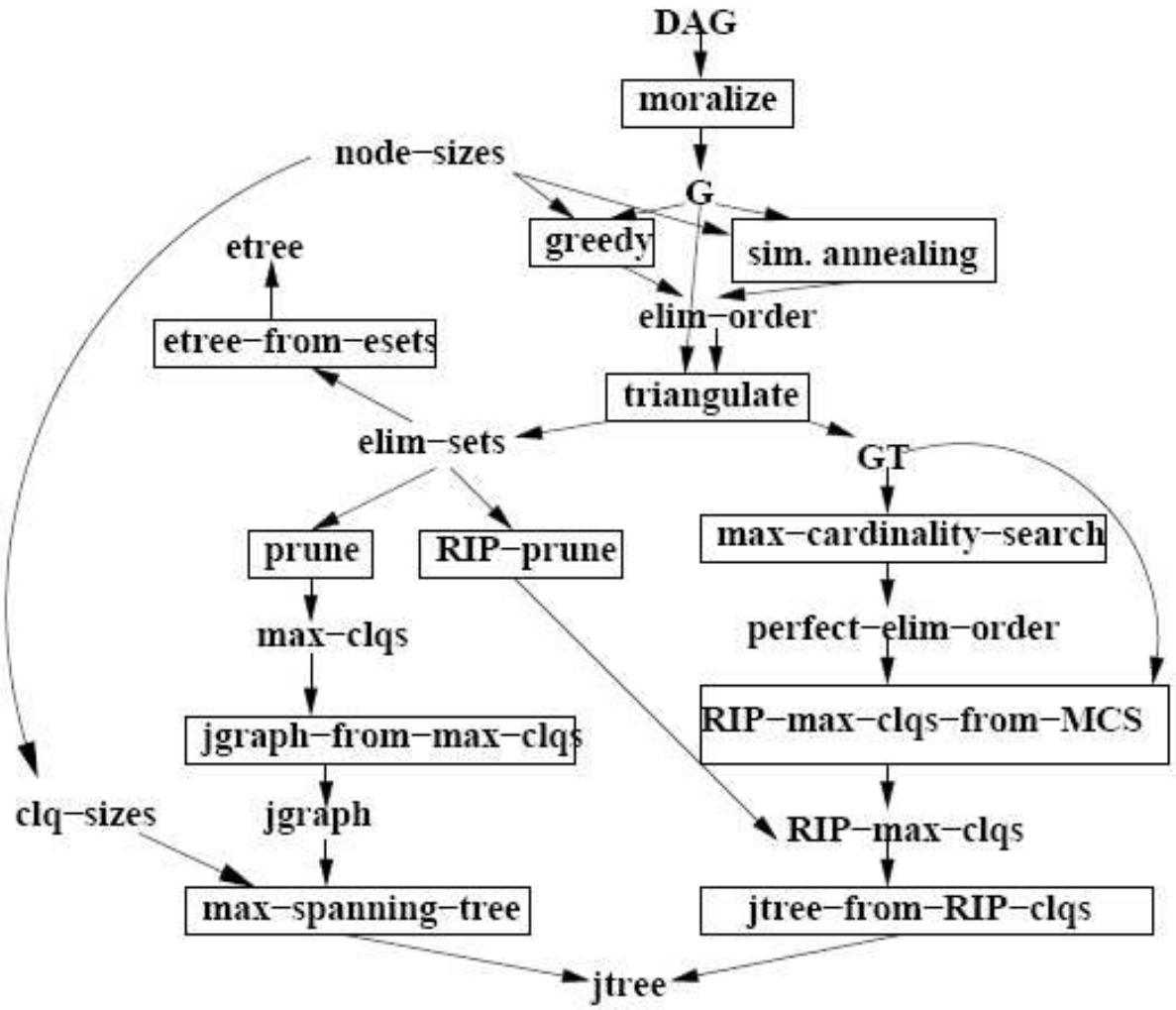
$$= \sum_C P(B|C)\,P(C,D)$$

Out of clique inference

Procedure CTree-Query (
  $\mathcal{T}$, // Clique tree over Φ
  $\{\beta_i\}, \{\mu_{i,j}\}$, // Calibrated clique and sepset beliefs for $\mathcal{T}$
  $\mathbf{Y}$ // A query
)

Let $\mathcal{T}'$ be a subtree of $\mathcal{T}$ such that $\mathbf{Y} \subseteq Scope[\mathcal{T}']$
Select a clique $r \in \mathcal{V}_{\mathcal{T}'}$ to be the root
$\Phi \leftarrow \beta_r$
for each $i \in \mathcal{V}_{\mathcal{T}}$
  $\phi \leftarrow \frac{\beta_i}{\mu_{i,Pr(i)}}$
  $\Phi \leftarrow \Phi \cup \{\phi\}$
$\mathbf{Z} \leftarrow Scope[\mathcal{T}'] - \mathbf{Y}$
Let $\prec$ be some ordering over $\mathbf{Z}$
return Sum-Product-Variable-Elimination$(\Phi, \mathbf{Z}, \prec)$

Creating a Jtree



Murphy PhD thesis (2002) p140

Max cliques from a chordal graph

- Triangulate the graph according to some ordering.
  - Start with all vertices unnumbered, set counter $i := N$.
  - While there are still some unnumbered vertices:
    - Let $v_i = \pi(i)$.
    - Form the set C_i consisting of v_i and its (unnumbered/ uneliminated) neighbors.
    - Fill in edges between all pairs of vertices in C_i.
    - Eliminate v_i and decrement i by 1.
- At each step, keep track of the clique that is created; if it is a subset of any previously created clique, discard it (since non maximal).

Cliques to Jtree

- Build a weighted graph where
  $W_{ij} = |C_i \text{ intersect } C_j|$
- Find max weight spanning tree. This is a jtree.

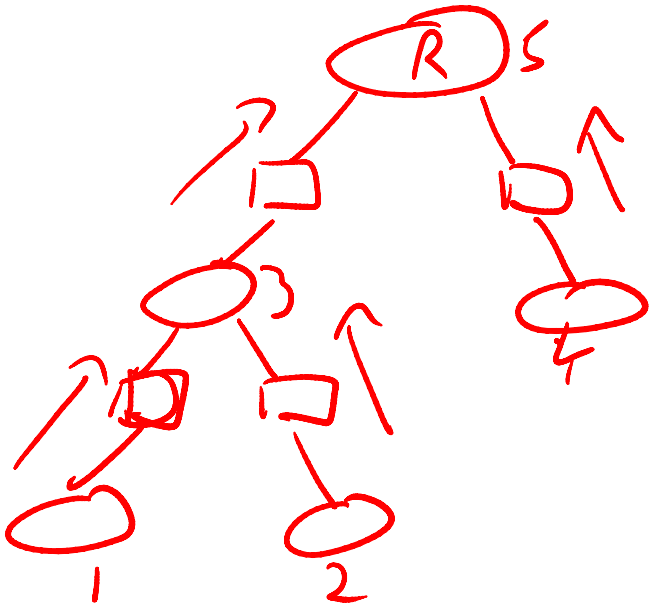
Stat 521A
Lecture 9

Outline

- Exact inference in clique trees (10.2, 10.3)
- Approximate inference – overview
- Loopy belief propagation (11.3)

Message passing on a clique tree

- To compute $p(X_i)$, find a clique that contains X_i, make it the root, and send messages to it from all other nodes.
- A clique cannot send a node to its parent until it is ready, ie. Has received msgs from all its children.
- Hence we send from leaves to root.



Upwards pass (collect to root)

Procedure CTree Sum Product Up (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  C_r // Some selected root clique
)

1 Initialize Cliques
2 **while** C_r is not ready
3 Let C_i be a ready clique
4 $\delta_{i \to p_r(i)}(S_{i,p_r(i)}) \leftarrow$ SP Message$(i, p_r(i))$
5 $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \text{Nb}_{C_r}} \delta_{k \to r}$
6 **return** β_r

Procedure Initialize Cliques (

)

1 **for** each clique C_i
2 $\psi_i[C_i] \leftarrow \prod_{\phi_j : \alpha(\phi_j)=i} \phi$
3

Procedure SP Message (
  i, // sending clique
  j // receiving clique
)

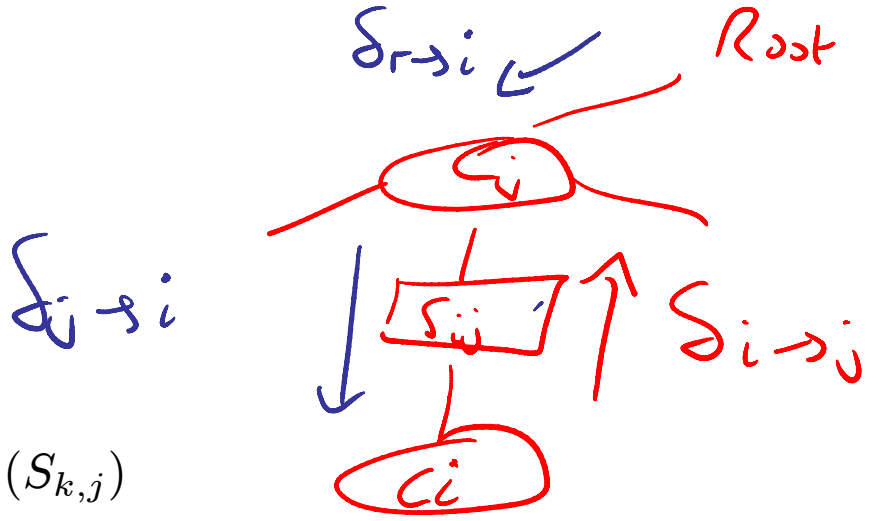
1 $\psi(C_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
2 $\tau(S_{i,j}) \leftarrow \sum_{C_i - S_{i,j}} \psi(C_i)$
3 **return** $\tau(S_{i,j})$

$$\beta_i(C_i) = \phi_i(C_i) \prod_{k \in n_i, k \neq j} \delta_{k \to i}(S_{k,i})$$

$$\delta_{i \to j}(S_{ij}) = \sum_{C_i \setminus S_{ij}} \beta_i(C_i)$$

Downwards pass (distribute from root)

- At the end of the upwards pass, the root has seen all the evidence.
- We send back down from root to leaves.



$$\beta_j(C_j) = \phi_j(C_j) \prod_{k \in n_j} \delta_{k \to j}(S_{k,j})$$

$$\delta_{j \to i}(S_{ij}) = \sum_{C_j \setminus S_{ij}} \phi_j(C_j) \prod_{k \in n_j, i \neq k} \delta_{k \to j}(S_{k,j})$$

$$= \sum_{C_j \setminus S_{ij}} \frac{\beta_j(C_j)}{\delta_{i \to j}(S_{ij})}$$

Use division operator to avoid double counting

Beliefs

- Thm 10.2.7. After collect/distribute, each clique potential represents a marginal probability (conditioned on the evidence)

$$\beta_i(C_i) = \sum_{\mathbf{x} \setminus C_i} \tilde{P}(\mathbf{x})$$

- If we get new evidence on X_i, we can multiply it in to any clique containing i, and then distribute messages outwards from that clique to restore consistency.

MAP configuration

- We can generalize the Viterbi algorithm from HMMs to find a MAP configuration of a general graph as follows.
- On the upwards pass, replace sum with max.
- At the root, find the most probable joint setting and send this as evidence to the root's children.
- Each child finds its most probable setting and sends this to its children.
- The jtree property ensures that when the state of a variable is fixed in one clique, that variable assumes the same state in all other cliques.

Samples

- We can generalize forwards-filtering backwards-sampling to draw exact samples from any GM as follows.
- Do a collect pass to the root as usual.
- Sample xR from the root marginal, and then enter it as evidence in all the children.
- Each child then samples itself from its updated local distribution and sends this to its children.

Calibrated clique tree

- Def 102.8. A clique tree is calibrated if, for all pairs of neighboring cliques, we have

$$\sum_{C_i \setminus S_{i,j}} \beta_i(C_i) = \sum_{C_j \setminus S_{i,j}} \beta_j(C_j) = \mu_{i,j}(S_{i,j})$$

- Eg. A-B-C clq tree AB – [B] – BC. We require

$$\sum_a \beta_{ab}(a,b) = \sum_c \beta_{bc}(b,c)$$

- Def 10.2.11. The measure defined by a calibrated tree is defined as

$$\beta_T(x) = \frac{\prod_i \beta_i(C_i)}{\prod_{<ij>} \mu_{i,j}(S_{ij})}$$

Calibrated clique tree

- Thm 10.2.12. For a calibrated clique tree, $p(x) \propto \beta_T(x)$ iff $\beta_i(C_i) \propto p(C_i)$
- Pf (sketch). Assume A-B-C. Then

$$p(A,B,C) = \frac{p(A,B)p(B,C)}{p(B)} = p(A,B)p(C|B) = p(A|B)p(B,C)$$

Clique tree invariant

- Suppose at every step, clique i sends a msg to clique j, and stores it in $\mu_{i,j}$:

Procedure Send-BU-Msg (
  i, // sending clique
  j // receiving clique
)

1 $\sigma_{i \to j} \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i$
2 // marginalize the clique over the sepset
3 $\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i \to j}}{\mu_{i,j}}$
4 $\mu_{i,j} \leftarrow \sigma_{i \to j}$

- Initially $\mu_{i,j}=1$ and $\beta_i = \prod_{f:\ f \text{ ass to } i} \phi_f$. Hence the following holds.

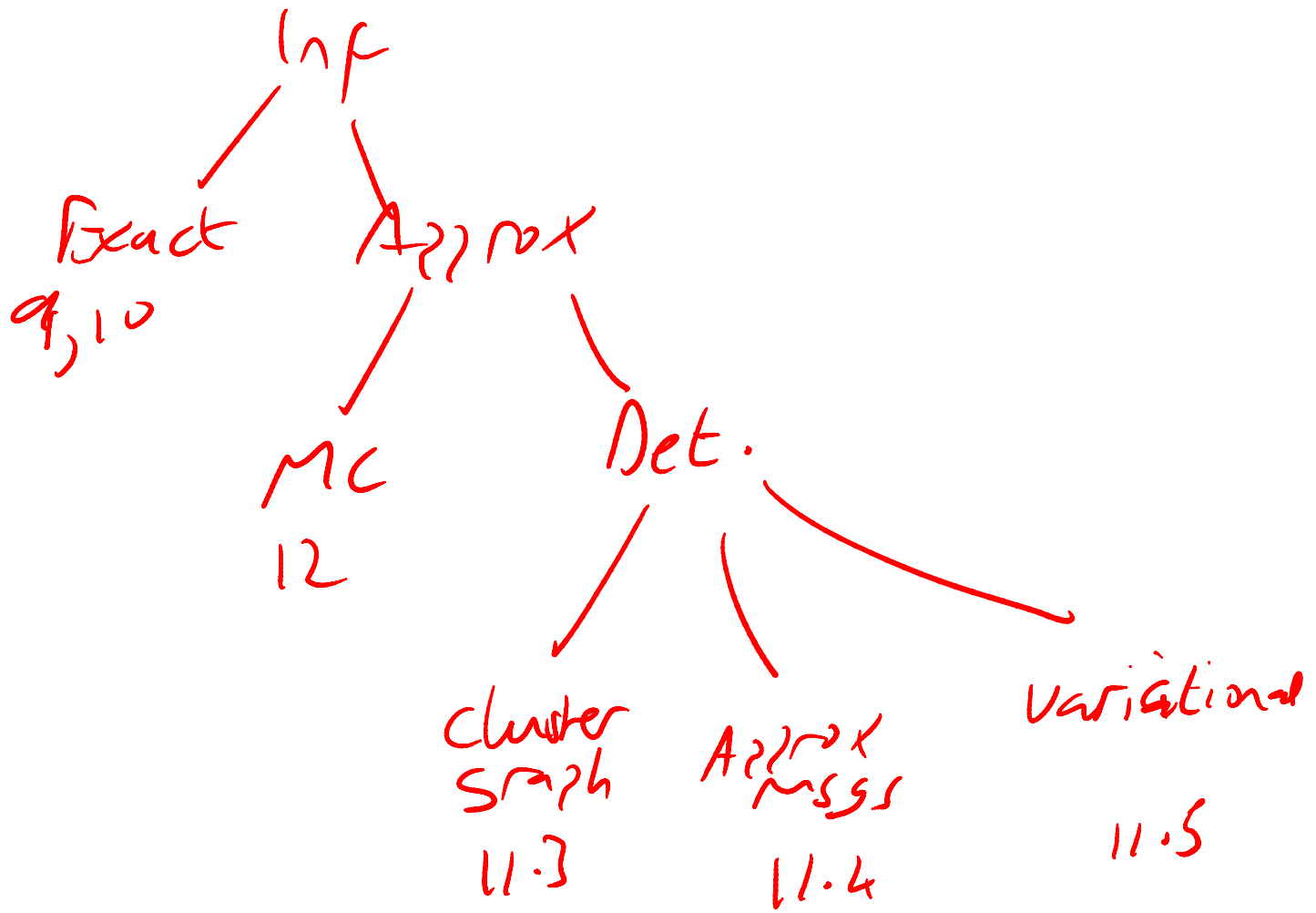
$$p(x) = \frac{\prod_i \beta_i(C_i)}{\prod_{<ij>} \mu_{i,j}(S_{ij})}$$

- Thm 10.3.4. This property holds after every belief updating operation. (But only when fully calibrated do clq pots = marginals.)

Summary of exact inference

- Build clique tree
  - eliminate nodes in some order
  - collect maximal cliques
  - Build a weighted graph where $W_{ij} = |C_i \text{ intersect } C_j|$
  - Find max weight spanning tree
- Initialize clique potentials with model potentials and evidence
- Do message passing on the tree

Approximate inference



Inference as optimization (11.1)

- Goal: find $\min_Q D(Q||P)$

- Thm 11.1.2

$$\begin{aligned}\min_Q D(Q||P) &= D(Q||\frac{1}{Z}\tilde{P}) = \sum_x Q(x)\log Q(X) - Q(x)\ln\tilde{P}(x) + \ln Z \\ &= lnZ - F(\tilde{P},Q) \\ F(\tilde{P},Q) &= H_Q(x) + \sum_c E_{x_c\sim Q}\ln\phi(x_c)\end{aligned}$$

  where F is the energy functional, and –F is the Helmholtz free energy

- Since D(Q||P) >=0, ln Z >= F(P,Q). We will maximize a lower bound on the log likelihood wrt Q.

Factored energy functional

- Consider a Q based on a cluster graph

$$Q = \{\beta_i : i \in \mathcal{V}\} \cup \{\mu_{i,j} : (i,j) \in \mathcal{E}\}$$

- Def 11.2.1. The factored energy functional is given by the following, where we approximate the entropy of Q

$$\tilde{F}(\tilde{P}, Q) = \sum_i E_{C_i \sim \beta_i} \ln \psi_i + \sum_i H_{\beta_i}(C_i) - \sum_{<ij>} H_{\mu_{i,j}}(S_{i,j})$$

- Thm 11.2.2. If Q is a set of calibrated beliefs for a tree, and Q has the form $Q(x) = \frac{\prod_i \beta_i(C_i)}{\prod_{<ij>} \mu_{i,j}(S_{ij})}$ then

$$\tilde{F}(\tilde{P}, Q) = F(\tilde{P}, Q)$$

Exact inference as optimization

- Define the local consistency polytope as (p381) the set of distributions

$$Q = \{\beta_i : i \in \mathcal{V}\} \cup \{\mu_{i,j} : (i,j) \in \mathcal{E}\}$$

which satisfy

$$\begin{aligned} \mu_{i,j}(S_{i,j}) &= \sum_{C_i \setminus S_{i,j}} \beta_i(C_i) \\ \sum_{c_i} \beta_i(c_i) &= 1 \\ \beta_i(c_i) &\geq 0 \end{aligned}$$

- Thm 11.1.1 If Q is a calibrated clique tree, which is an I-map for P, then

$$max_{Q \in \text{Local}} \tilde{F}(\tilde{P}, Q)$$

has a unique global optimum, in which Q=P

Constrained optimization

CTree-Optimize

Find $Q = \{\beta_i : i \in \mathcal{V}_T\} \cup \{\mu_{i,j} : (i\ j) \in \mathcal{E}_T\}$

that maximize $\tilde{F}[\tilde{P}_\Phi, Q]$

$$\mu_{i,j}[s_{i,j}] = \sum_{\boldsymbol{C_i - S_{i,j}}} \beta_i[c_i]$$

subject to $\forall (i\ j) \in \mathcal{E}_T, \forall s_{i,j} \in Val(\boldsymbol{S}_{i,j})$

$$\sum_{c_i} \beta_i[c_i] = 1 \qquad \forall i \in \mathcal{V}_T$$

$$\beta_i[c_i] \geq 0 \qquad \forall i \in \mathcal{V}_T, c_i \in Val(C_i)$$

$$\begin{aligned}\mathcal{J} = {} & \tilde{F}[\tilde{P}_\Phi, Q] \\ & - \sum_{i \in \mathcal{V}_T} \lambda_i \left(\sum_{c_i} \beta_i[c_i] - 1 \right) \\ & - \sum_i \sum_{j \in \mathrm{Nb}_i} \sum_{s_{i,j}} \lambda_{j \to i}[s_{i,j}] \left(\sum_{c_i \sim s_{i,j}} \beta_i[c_i] - \mu_{i,j}[s_{i,j}] \right),\end{aligned}$$

Msgs = Lagrange multipliers

$$\begin{aligned}\mathcal{J} &= \tilde{F}[\tilde{P}_\Phi, Q] \\ &\quad - \sum_{i \in \mathcal{V}_T} \lambda_i \left(\sum_{\boldsymbol{c}_i} \beta_i[\boldsymbol{c}_i] - 1 \right) \\ &\quad - \sum_i \sum_{j \in \mathrm{Nb}_i} \sum_{\boldsymbol{s}_{i,j}} \lambda_{j \to i}[\boldsymbol{s}_{i,j}] \left(\sum_{\boldsymbol{c}_i \sim \boldsymbol{s}_{i,j}} \beta_i[\boldsymbol{c}_i] - \mu_{i,j}[\boldsymbol{s}_{i,j}] \right),\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial \beta_i[\boldsymbol{c}_i]} \mathcal{J} &= \ln \psi_i[\boldsymbol{c}_i] - \ln \beta_i[\boldsymbol{c}_i] - 1 - \lambda_i - \sum_{j \in \mathrm{Nb}_i} \lambda_{j \to i}[\boldsymbol{s}_{i,j}] \\ \frac{\partial}{\partial \mu_{i,j}[\boldsymbol{s}_{i,j}]} \mathcal{J} &= \ln \mu_{i,j}[\boldsymbol{s}_{i,j}] + 1 + \lambda_{i \to j}[\boldsymbol{s}_{i,j}] + \lambda_{j \to i}[\boldsymbol{s}_{i,j}].\end{aligned}$$

$$\begin{aligned}\beta_i[\boldsymbol{c}_i] &= \exp\{-1 - \lambda_i\} \psi_i[\boldsymbol{c}_i] \prod_{j \in \mathrm{Nb}_i} \exp\{-\lambda_{j \to i}[\boldsymbol{s}_{i,j}]\} \\ \mu_{i,j}[\boldsymbol{s}_{i,j}] &= \exp\{-1\} \exp\{-\lambda_{i \to j}[\boldsymbol{s}_{i,j}]\} \exp\{-\lambda_{j \to i}[\boldsymbol{s}_{i,j}]\}.\end{aligned}$$

$$\delta_{i \to j}[\boldsymbol{s}_{i,j}] \triangleq \exp\left\{ -\lambda_{i \to j}[\boldsymbol{s}_{i,j}] - \frac{1}{2} \right\}.$$

Msgs = Lagrange multipliers

- Thm 11.2.3. A set of beliefs Q is a stationary point of CTreeOptimize iff there exist a set of messages such that

$$\delta_{i\to j} \propto \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \left(\prod_{k \in \mathrm{Nb}_i - \{j\}} \delta_{k\to i} \right)$$

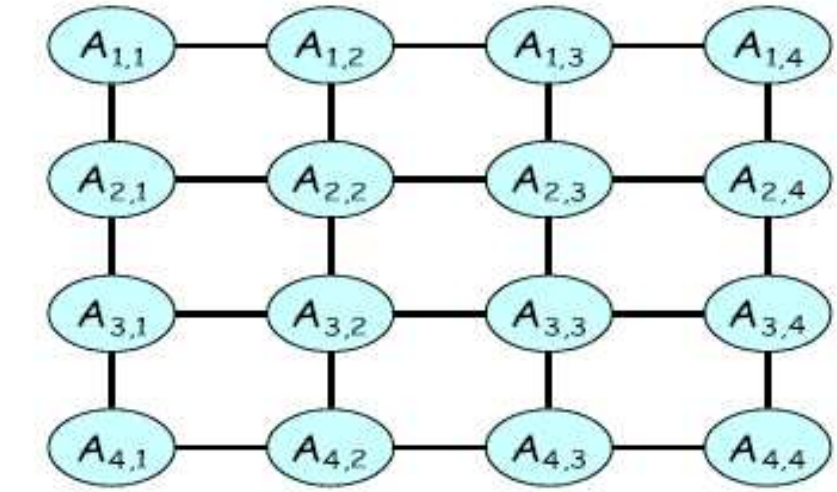
and moreover, we have that

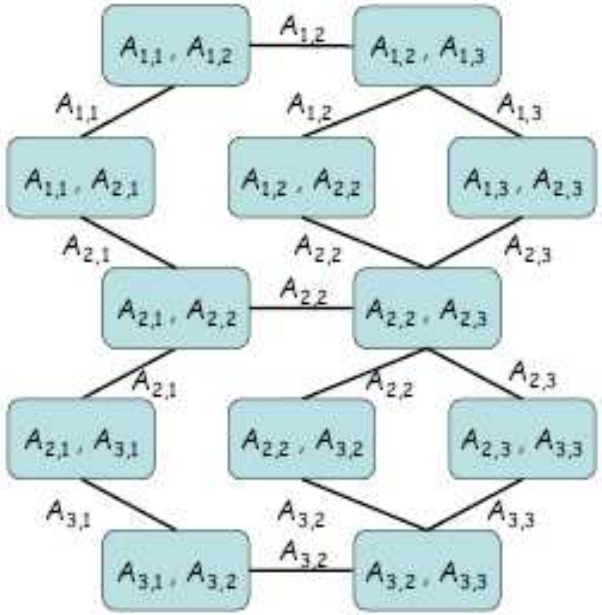
$$\begin{aligned} \beta_i &\propto \psi_i \left(\prod_{j \in \mathrm{Nb}_i} \delta_{j\to i} \right) \\ \mu_{i,j} &= \delta_{j\to i} \cdot \delta_{i\to j}. \end{aligned}$$

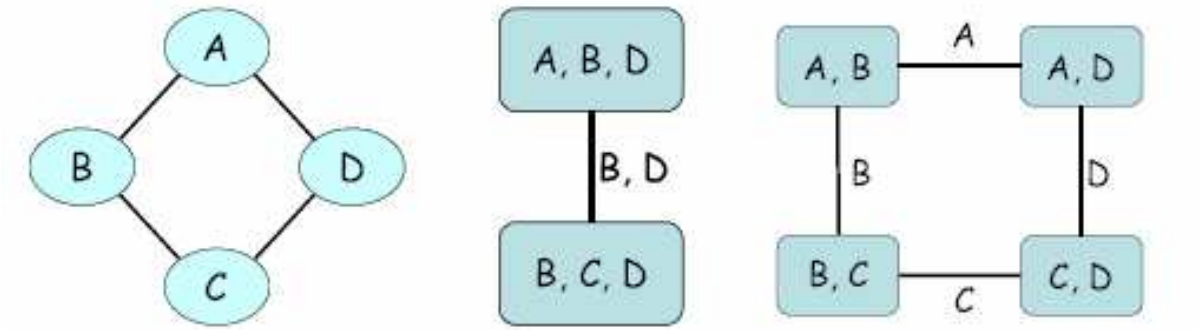
Cluster graphs

- If the cluster graph is a cluster tree with RIP, then the factored energy is equal to the energy, and enforcing local consistency is equivalent to enforcing global consistency.
- However, the cliques may be too big.
- Let us consider general CGs which only have to satisfy the RIP constraint.
- Hence all edges associated with some node X form a tree and all clusters agree on the marginal for each X. However, they may not agree on higher order marginals.

Examples

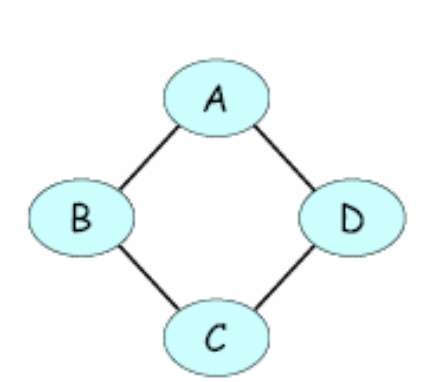

$A_{1,1}$ $A_{1,2}$ $A_{1,3}$ $A_{1,4}$
$A_{2,1}$ $A_{2,2}$ $A_{2,3}$ $A_{2,4}$
$A_{3,1}$ $A_{3,2}$ $A_{3,3}$ $A_{3,4}$
$A_{4,1}$ $A_{4,2}$ $A_{4,3}$ $A_{4,4}$

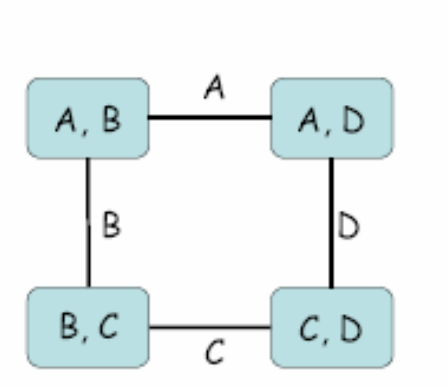

$A_{1,1}$, $A_{1,2}$
$A_{1,2}$
$A_{1,2}$, $A_{1,3}$
$A_{1,1}$
$A_{1,2}$
$A_{1,3}$
$A_{1,1}$, $A_{2,1}$
$A_{1,2}$, $A_{2,2}$
$A_{1,3}$, $A_{2,3}$
$A_{2,1}$
$A_{2,2}$
$A_{2,3}$
$A_{2,1}$, $A_{2,2}$
$A_{2,2}$
$A_{2,2}$, $A_{2,3}$
$A_{2,1}$
$A_{2,2}$
$A_{2,3}$
$A_{2,1}$, $A_{3,1}$
$A_{2,2}$, $A_{3,2}$
$A_{2,3}$, $A_{3,3}$
$A_{3,1}$
$A_{3,2}$
$A_{3,3}$
$A_{3,1}$, $A_{3,2}$
$A_{3,2}$
$A_{3,2}$, $A_{3,3}$

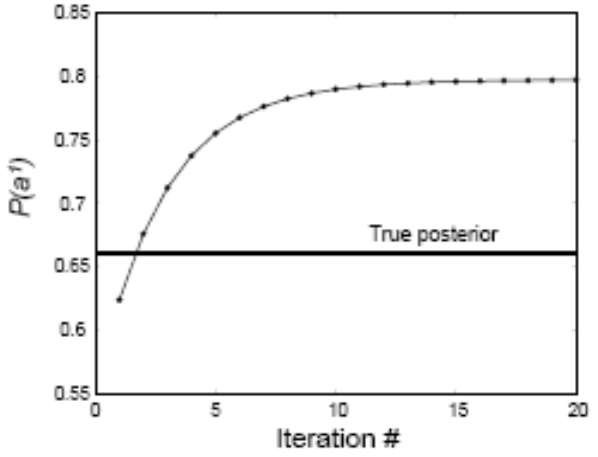

A
B
D
C
A, B, D
B, D
B, C, D
A, B
A
A, D
B
D
B, C
C
C, D

Belief prop on a cluster graph

- We can run the BP algorithm on a CG even if it is not a tree. This is called loopy BP.
- This can fail to converge and give the wrong answers due to double counting of evidence.



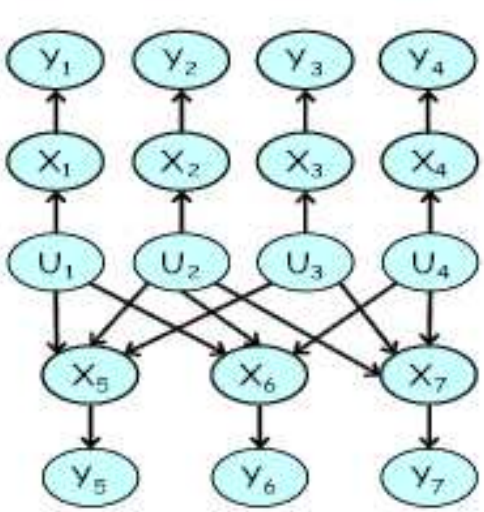




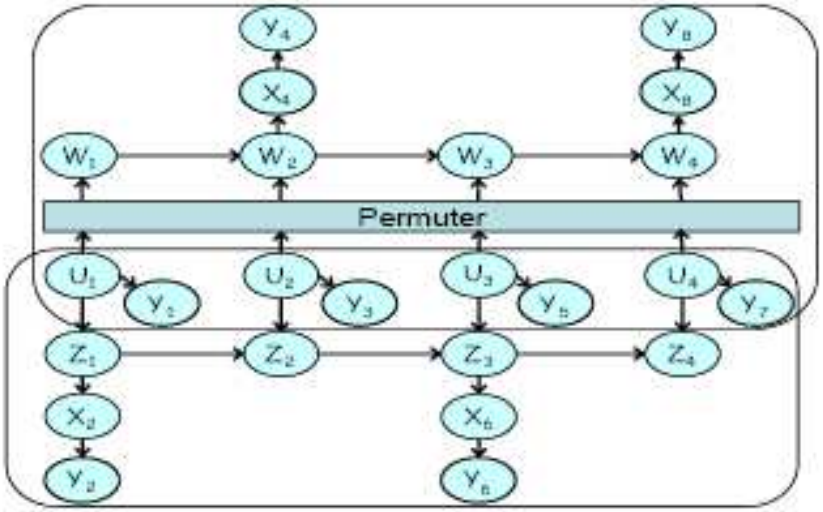
Turbocodes

- Channel coding is a way of encoding msgs that makes them resistant to noise, and hence easier to decode.
- Let us send a k-bit msg u(1:k) using n bits, x(1:n) eg x = 3 copies of u. We receive y(1:n) and estimate u. The rate of the code is k/n.
- Shannon's thm characterizes the best rate one can achieve for a given error rate and noise level.
- Turbodecoding is a method to approximately estimate u from y which achieves near-optimal rate. It is equivalent to loopy BP in a particular DGM.

Turbocodes



(a)



(b)

K=4,n=7 parity check

K=4,n=8 turbocode

Convergence (11.3.4)

- For discrete networks, one can show that a sufficient (but not necessary) condition for LBP to converge is if the connections are not “too deterministic”.
- Eg for Ising model, sufficient condition is

$$\max_i \max_{j \in \mathrm{Nb}_i} \sum_{k \in \mathrm{Nb}_i - \{j\}} \tanh |\epsilon_{k,i}| < 1.$$

- Similar conditions exist for Gaussian networks.
- Special case analysis has been derived for turbocodes.

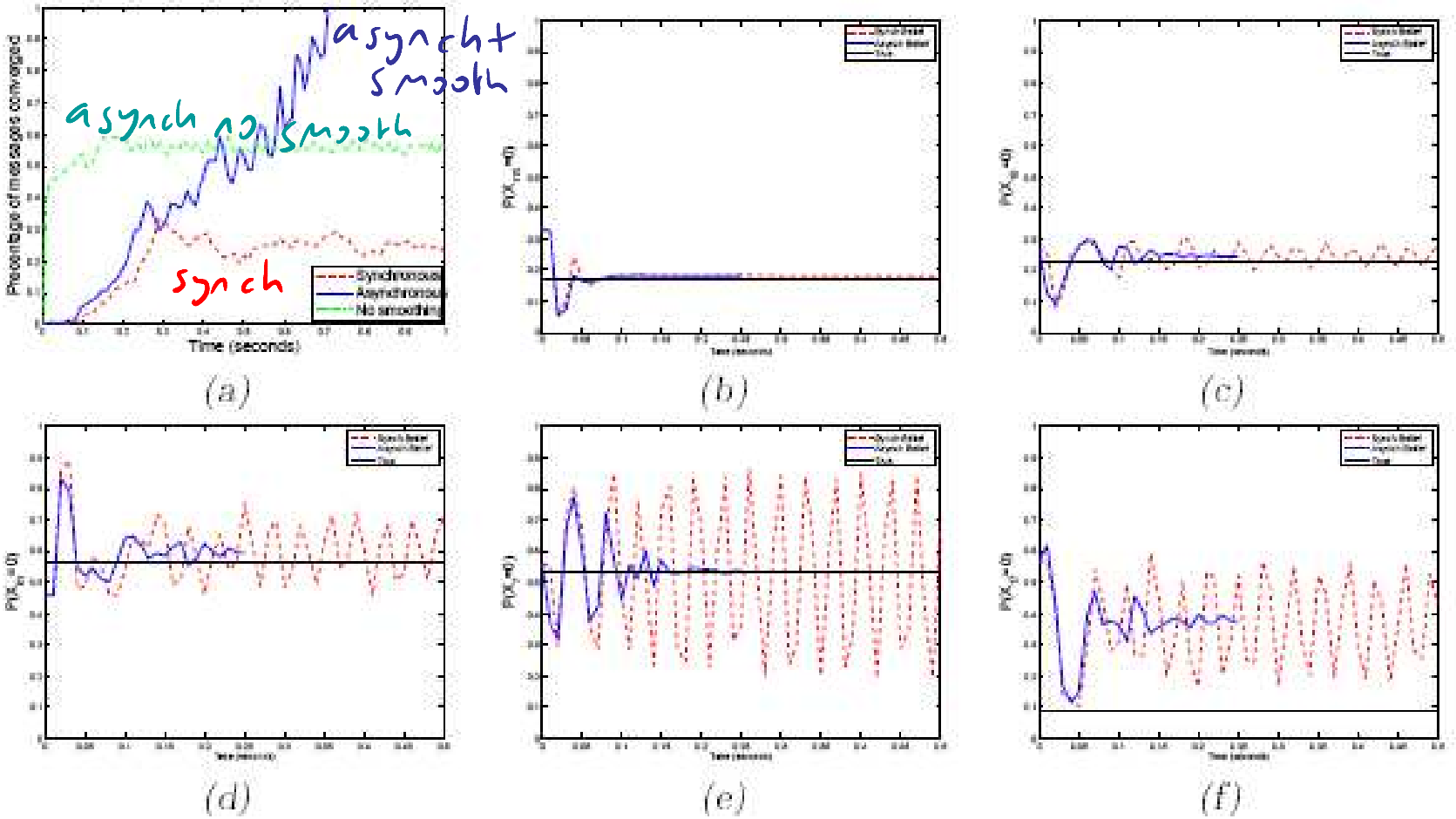
Encouraging convergence

- One can use damped updates

$$\delta_{i\to j} \leftarrow \sum_{C_i - S_{i,j}} \prod_{k\neq j} \overrightarrow{\delta_{k\to i}} \qquad \delta_{i\to j} \leftarrow \lambda \left(\sum_{C_i - S_{i,j}} \prod_{k\neq j} \delta_{k\to i} \right) + (1-\lambda)\delta_{i\to j}^{\text{old}},$$

- Asychronous updates work better than sychronous.
- Tree reparameterization (TRP) selects a set of trees, each of which spans a large number of clusters, and whose union covers all the edges. It then selects a tree at rnd and calibrates it, treating all other messages as local evidence.
- Priority-queue based msg scheduling also works very well.

Example: 11x11 Ising

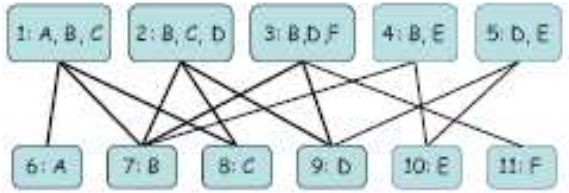


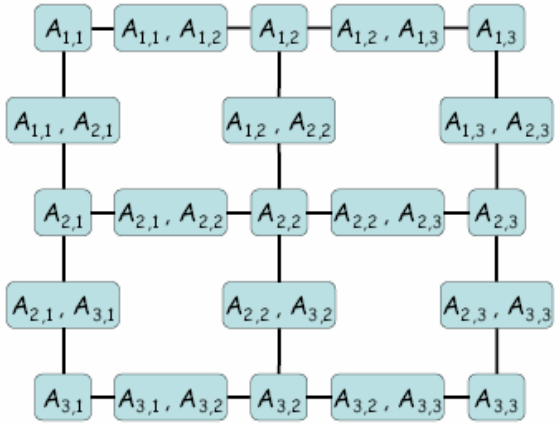
Accuracy

- In general, it is hard to characterize the accuracy of approximate solutions. Often the most probable state is locally correct but is over confident.
- For Gaussian networks, Weiss et al showed that, if the method converges, the means are exact, but the variances are too small.

Bethe cluster graphs

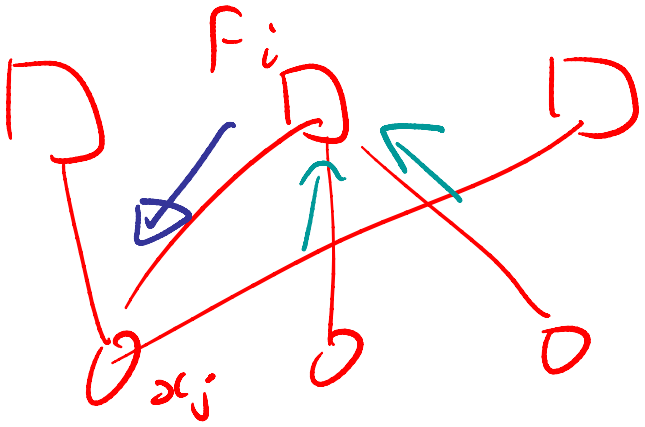
- Suppose we create one cluster for each original factor, and one cluster for each node.





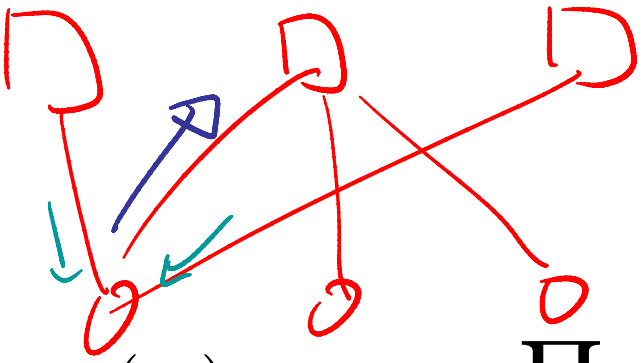
- Then for a pairwise MRF, propagating $C_i – C_{ij} – C_j$ is equivalent to sending msgs from node i to node j via edge ij.
- In general, BP on the Bethe CG = BP on the factor graph.

BP on factor graphs



Bishop p406

$$\mu_{f_i \to x_j}(x_j) = \sum_{c_i \setminus x_j} f(c_i) \prod_{k \in nb(f_i) \setminus x_j} \mu_{x_k \to f_i}(x_k)$$



$$\mu_{x_i \to f_j}(x_i) = \prod_{k \in nb(x_i) \setminus f_j} \mu_{f_k \to x_i}(x_i)$$

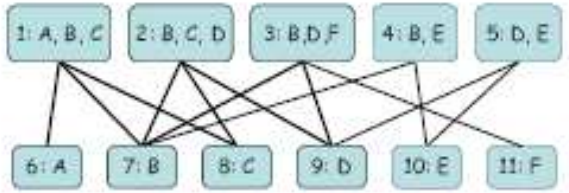
Stat 521A
Lecture 10

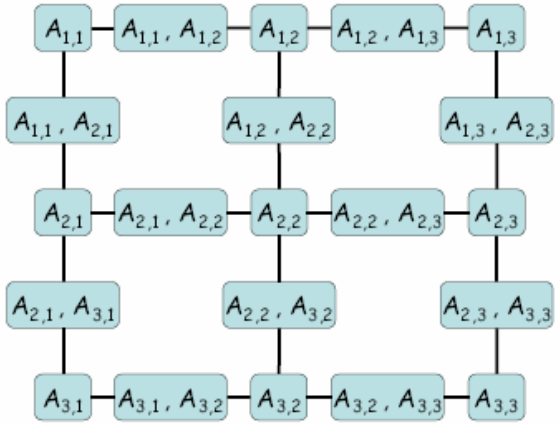
Outline

- Belief propagation: entropy approximations (11.3.7)
- Expectation propagation (11.4)
- Mean field (11.5.1)
- Variational EM/ Bayes (Bishop 10.1-10.2)
- Structured variational (11.5.2)

Bethe cluster graphs

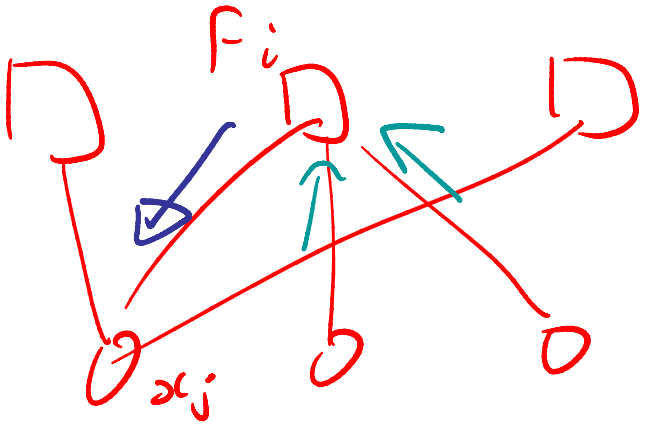
- Suppose we create one cluster for each original factor, and one cluster for each node.





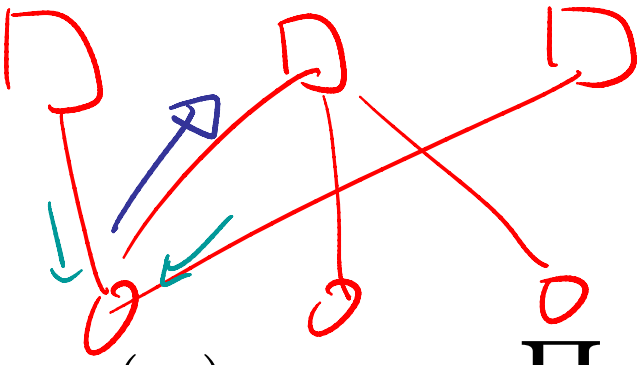
- Then for a pairwise MRF, propagating $C_i – C_{ij} – C_j$ is equivalent to sending msgs from node i to node j via edge ij.
- In general, BP on the Bethe CG = BP on the factor graph.

BP on factor graphs



Bishop p406

$$\mu_{f_i \to x_j}(x_j) = \sum_{c_i \setminus x_j} f(c_i) \prod_{k \in nb(f_i) \setminus x_j} \mu_{x_k \to f_i}(x_k)$$



$$\mu_{x_i \to f_j}(x_i) = \prod_{k \in nb(x_i) \setminus f_j} \mu_{f_k \to x_i}(x_i)$$

Bethe approximation to entropy

- Thm 11.3.10. If Q is a calibrated set of beliefs for a Bethe approximation CG then the factored energy is given by

$$\begin{aligned}\tilde{F}(\tilde{P}, Q) &\stackrel{\text{def}}{=} \sum_{\phi} E_{\beta_\phi} \ln \phi + \sum_{\phi} H_{\beta_\phi}(C_\phi) - \sum_{s} H_{\mu_s}(S_s) \\ &= \sum_{\phi} E_{\beta_\phi} \ln \phi + \sum_{\phi} H_{\beta_\phi}(C_\phi) - \sum_{i} (d_i - 1) H_{\beta_i}(X_i)\end{aligned}$$

  where d_i = #factors that contain Xi.

- If Xi appears in di factors, by RIP, it appears in (di-1) sepsets. Hence we count the entropy of each Xi once in total.

Weighted approximation to entropy

- Consider a cluster graph, each of whose clusters (regions) has a counting number μ_r. Define the weighted approximate entropy as

$$H_Q^{\mu}(X) = \sum_r \mu_r H_{\beta_r}(C_r)$$

- For a Bethe-structured CG, we set

$$\mu_i = 1 - \sum_{r \in nb_i} \mu_r$$

- If we set μ_r=1, we recover the Bethe approximation.
- Let us consider more general weightings.

Convex approximation to entropy

- Def 11.3.13. We say that μ_r are convex counting numbers if there exist non-negative numbers ν_r, ν_i, $\nu_{r,i}$ st

$$\begin{aligned} \mu_r &= \nu_r + \sum_{i\,:\,X_i \in \boldsymbol{C}_r} \nu_{r,i} \quad for\ all\ r \\ \mu_i &= \nu_i - \sum_{r\,:\,X_i \in \boldsymbol{C}_r} \nu_{r,i} \quad for\ all\ i \end{aligned}$$

- Then

$$\sum_r \mu_r \mathbb{H}_{\beta_r}(\boldsymbol{C}_r) + \sum_i \mu_i \boldsymbol{H}_{\beta_i}(X_i) = \sum_r \nu_r \mathbb{H}_{\beta_r}(\boldsymbol{C}_r) + \sum_{r, X_i \in \boldsymbol{C}_r} \nu_{r,i} (\mathbb{H}_{\beta_r}(\boldsymbol{C}_r) - \boldsymbol{H}_{\beta_i}(X_i)) \sum_i \nu_i \boldsymbol{H}_{\beta_i}(X_i$$

- Thm 11.3.14. The above eqn is concave for any set of beliefs Q which satisfy marginal consistency constraints.

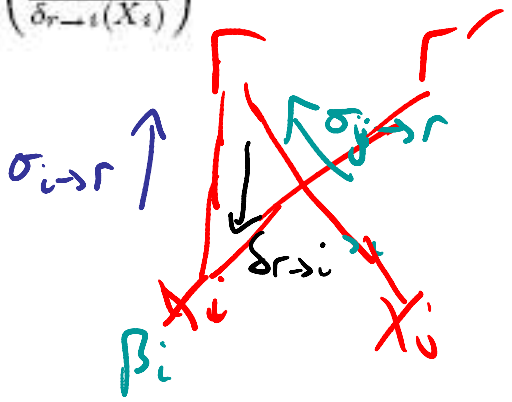
Convex BP

Algorithm 11.2 Convergent message passing algorithm for Bethe-structured region graphs with convex counting numbers

Procedure Convex-BP-Msg (
  $\psi_r[C_r]$ // set of initial potentials
  $\sigma_{i \to r}(C_r)$ // Current node-to-region messages
)

1 **for** $i = 1, \ldots, n$
2 // Compute incoming messages from neighboring regions to X_i **for** $r \in \mathrm{Nb}_i$
3 $\delta_{r \to i}(X_i) \leftarrow \sum_{C_r - X_i} \left(\psi_r[C_r] \prod_{j \in \mathrm{Nb}_r - \{i\}} \sigma_{j \to r}(C_r) \right)^{\frac{1}{\hat{\nu}_{i,r}}}$
4 // Compute beliefs for X_i renormalizing to avoid numerical underflows
5 $\beta_i[X_i] \leftarrow \propto \prod_{r \in \mathrm{Nb}_i} (\delta_{r \to i}(X_i))^{\hat{\nu}_{i,r}/\hat{\nu}_i}$
6 // Compute outgoing messages from X_i to neighboring regions **for** $r \in \mathrm{Nb}_i$
7 $\sigma_{i \to r}(C_r) \leftarrow \left(\psi_r[C_r] \prod_{j \in \mathrm{Nb}_r - \{i\}} \sigma_{j \to r}(C_r) \right)^{-\frac{\nu_{i,r}}{\hat{\nu}_{i,r}}} \left(\frac{\beta_i[X_i]}{\delta_{r \to i}(X_i)} \right)^{\nu_r}$
8 **return** $\{\sigma_{i \to r}(C_r)\}_{i, r \in \mathrm{Nb}_i}$

$$\hat{\nu}_i = \nu_i + \sum_{r \in \mathrm{Nb}_i} \nu_r; \qquad \hat{\nu}_{i,r} = \nu_r + \nu_{i,r}.$$



TRW

- Tree reweighting algorithm (TRW) uses the following convex counting numbers, given a distribution over trees T st each edge in the pairwise network is present in at least 1 tree

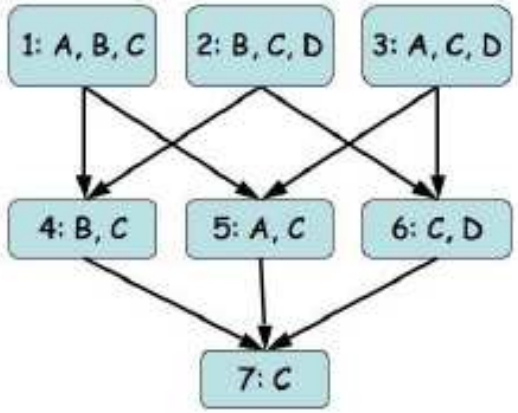
$$\begin{aligned} \mu_i &= -\sum_{T \ni X_i} \rho(T) \\ \mu_{i,j} &= \sum_{T \ni (X_i, X_j)} \rho(T) \end{aligned}$$

Convex or not?

- When standard BP converges, the Bethe approximation to the entropy is often more accurate than the convex approximation.
- However, it is desirable to have a convex inference engine in the inner loop of learning.
- If you train with a convex approximation, there are some arguments you should use the same convex approx at test time for decoding.

Region graphs (11.3.7.3)

- One can use more general CGs than the Bethe construction, which lets you model higher order interactions which are intermediate between the original factors and singletons.
- Resulting algorithm is complex.

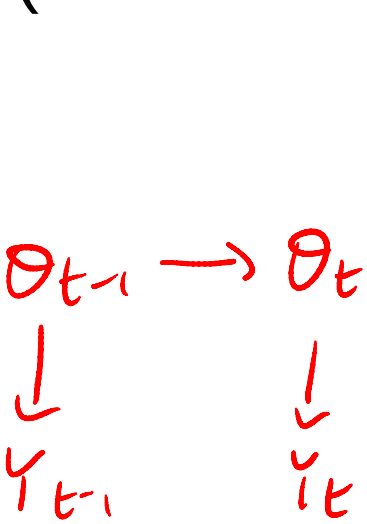


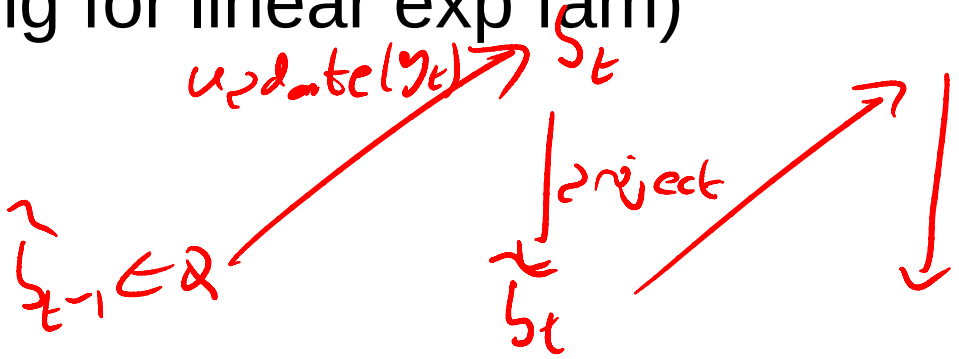
Approximate messages

- Suppose we use a cluster tree (or graph), but approximate the messages eg. to prevent them becoming “too fat”
- If the clusters have internal structure, we can efficiently combine factored incoming messages with factored clusters to get factored outgoing messages
- We can also use this to combine non conjugate distributions: eg we approximate a non-conjugate likelihood by a simple form (eg MVN) and combine with a simple cluster potential (eg MVN) to get a simple posterior for the next step

Assumed density filtering (ADF)

- Consider sequential Bayesian updating in which we assume the prior $p(\theta_{t-1}|y_{1:t-1})$ lives in some tractable family Q (eg MVN).
- At each step, we do 1 step of Bayesian updating to get the posterior $p(\theta_t|y_{1:t})$ and then do an M-projection to get the best approximation within Q (=moment matching for linear exp fam)





Eg $y_t|\theta_t$ = logistic(y_t x_t^T θ_t), $\theta_t=\theta_{t-1}$

Eg. $y_t|\theta_t$ = Gauss, θ_t | θ_{t-1} = mix Gauss

ADF cont'd

- We combine msg from past (prior) with local evidence (likelihood), project, then compute new msg

$$
\begin{aligned}
b_{t-1,t} &\propto \phi_{t-1,t}\mu_{t-1} \\
\tilde{b}_{t-1,t} &= \text{proj}(b_{t-1,t}, Q) \\
\mu_t &= \sum_t \tilde{b}_{t-1,t}
\end{aligned}
$$

Expectation propagation

- For batch problems, ADF is suboptimal, and depends on order of data.
- EP idea: add backwards pass

$$
\begin{aligned}
b^*_{t,t+1} &= \frac{\tilde{b}_{t,t+1}}{\mu_{t+1}} \mu^*_{t+1} \\
\tilde{b}^*_{t,t+1} &= \text{proj}(b^*_{t,t+1}, Q) \\
\mu^*_t &= \sum_{t+1} \tilde{b}^*_{t,t+1}
\end{aligned}
$$

- Since msgs no longer exact, need to iterate

Division = subtraction of natural params

- Assume all beliefs and msgs are linear exponential families. Then

$$\tilde{\delta}_{i \to j} = \frac{\tilde{\sigma}_{i \to j}}{\tilde{\delta}_{j \to i}} \propto \exp\left\{ \langle (\theta_{\tilde{\sigma}_{i \to j}} - \theta_{\tilde{\delta}_{j \to i}}), \tau_{i,j}(s_{i,j}) \rangle \right\}$$

- This can result in negative values for the natural params (eg Gaussians with –ve variance).
- But undirected GMs with tabular potentials are in the linear exp family and can always be used to represent valid beliefs/msgs

Projection

- How compute natural parameters of a msg?
- Compute the expected statistics of the separator, according to the current approximate beliefs

$$\theta_{\tilde{\delta}_{i\to j}} \leftarrow \text{M-project}_{i,j}(\boldsymbol{E}_{\boldsymbol{S}_{i,j}\sim\tilde{\beta}_i}[\tau_{i,j}(\boldsymbol{S}_{i,j})]) - \theta_{\tilde{\delta}_{j\to i}}.$$

- Computing the expectation can be made tractable if β_i has factored structure.
- In general, the M projection can be hard.
- But if we have discrete variables, and Q is fully factorized, it amounts to computing a product of marginals.

Variational analysis

- We optimize the same (approximate) objective as before (factored free energy), but relax the local consistency conditions so we only match statistics (eg marginals) instead of full distributions

EP-Optimize

Find Q

that maximize $\tilde{F}[\tilde{P}_\Phi, Q]$

subject to

$$E_{\boldsymbol{s}_{i,j} \sim \mu_{i,j}}[\tau_{i,j}] = E_{\boldsymbol{s}_{i,j} \sim \beta_{ij}}[\tau_{i,j}] \quad \forall (i\ j) \in \mathcal{E}_{\mathcal{I}}$$

$$\sum_{\boldsymbol{c}_i} \beta_i[\boldsymbol{c}_i] = 1 \qquad \forall i \in \mathcal{V}_{\mathcal{I}}$$

$$\sum_{\boldsymbol{s}_{i,j}} \mu_{i,j}[\boldsymbol{s}_{i,j}] = 1 \qquad \forall (i\ j) \in \mathcal{E}_{\mathcal{I}}$$

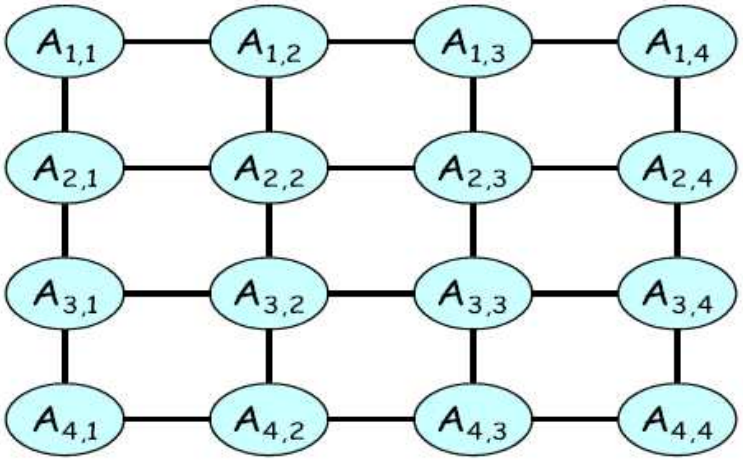
$$\beta_i[\boldsymbol{c}_i] \geq 0 \qquad \forall i \in \mathcal{V}_{\mathcal{I}}, \boldsymbol{c}_i \in Val(\boldsymbol{C}_i)$$

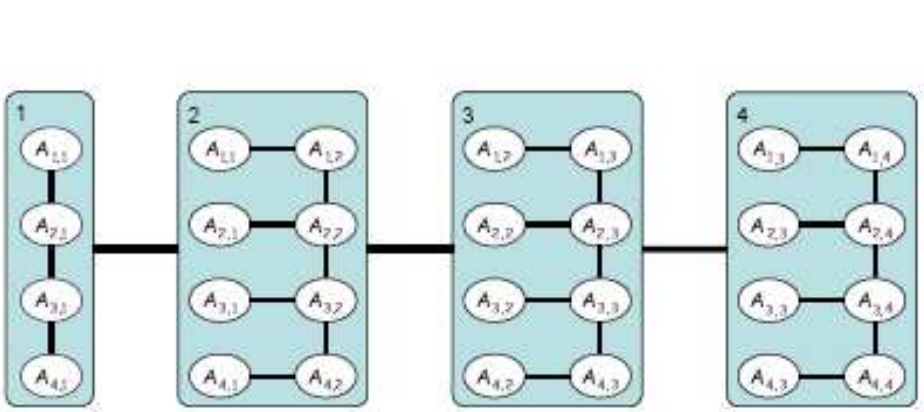
EP msg passing

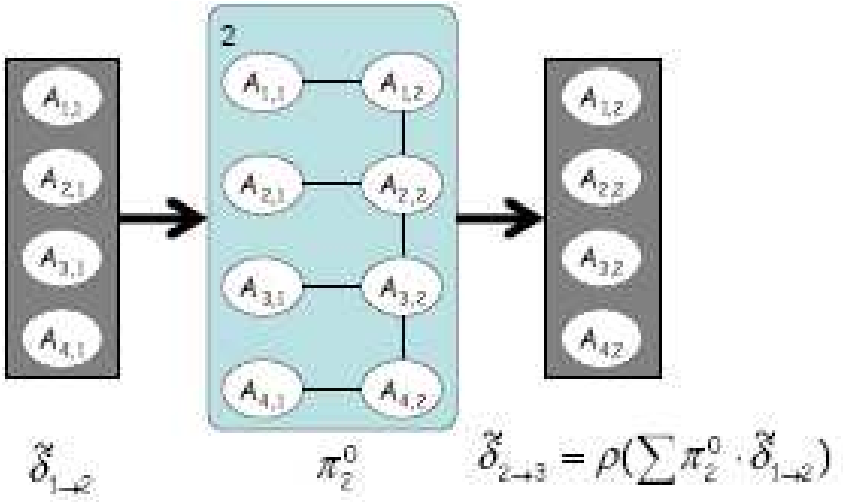
- Thm 11.4.5. Let Q be a set of beliefs st μ_{ij} is in the exp family Q_{ij}. Let M-project-distr$_{i,j}$ marginalize onto $S_{i,j}$ and then project onto Q_{ij}. Then Q is a stationary point of EP-optimize iff there exist auxiliary beliefs δ such that

$$\begin{aligned}
\delta_{i \to j} &= \frac{\text{M-project-distr}_{i,j}(\beta_i)}{\delta_{j \to i}} \\
\beta_i &\propto \psi_i \cdot \prod_{j \in \text{Nb}_i} \delta_{j \to i} \\
\mu_{i,j} &\propto \delta_{j \to i} \cdot \delta_{i \to j}.
\end{aligned}$$

Example





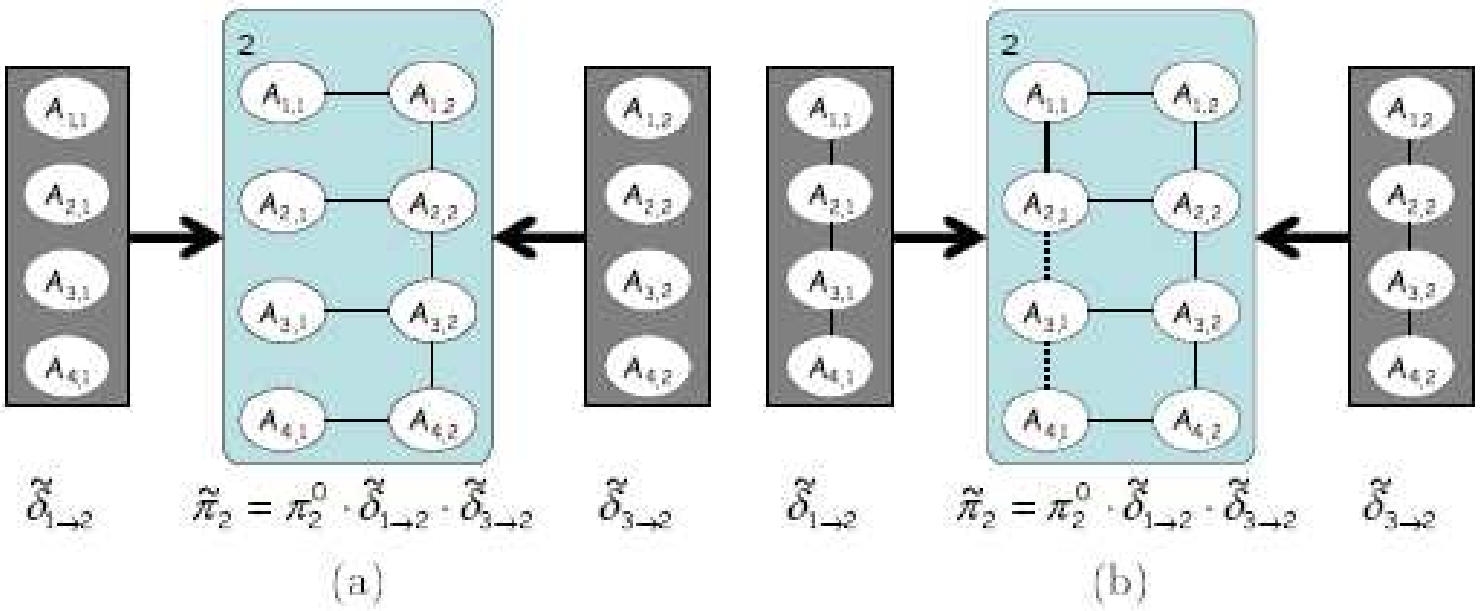


Cluster graph

Fully factored Q_{ij}

Structured messages

- The Q distribution (onto which we project) can be any structure that makes computing marginals efficient, eg a chain or clique tree.



Summary so far

- Let us summarize the BP and EP methods, and then introduce a new class of variational methods

BP on Cluster graphs

- In CGBP, we made 2 approximations
- 1. Optimize a bound on D(Q||P)

$$D(Q||P) = \ln Z - F(\tilde{P}, Q)$$
$$F(\tilde{P}, Q) \stackrel{\text{def}}{=} H_Q(x) + \sum_i E_{C_i \sim Q} \ln \psi_i(C_i)$$

Thm 11.1.2

- 2. Use pseudo marginals β_I, μ_{ij} and thus approximated the entropy H(Q) and hence used the approximate bound

$$\tilde{F}(\tilde{P}, Q) \stackrel{\text{def}}{=} \sum_i E_{C_i \sim \beta_i} \ln \psi_i + \sum_i H_{\beta_i}(C_i) - \sum_{<ij>} H_{\mu_{i,j}}(S_{i,j})$$

- We then optimize the approximate bound subject to local consistency constraints over some cluster graph

CGBP objective

Find $Q = \{\beta_i : i \in \mathcal{V}_T\} \cup \{\mu_{i,j} : (i\ j) \in \mathcal{E}_T\}$

that maximize $\tilde{F}[\tilde{P}_\Phi, Q]$

$$\mu_{i,j}[s_{i,j}] = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i[\boldsymbol{c}_i]$$

subject to $\forall (i\ j) \in \mathcal{E}_T, \forall \boldsymbol{s}_{i,j} \in Val(\boldsymbol{S}_{i,j})$

$$\sum_{\boldsymbol{c}_i} \beta_i[\boldsymbol{c}_i] = 1 \qquad \forall i \in \mathcal{V}_T$$

$$\beta_i[\boldsymbol{c}_i] \geq 0 \qquad \forall i \in \mathcal{V}_T, \boldsymbol{c}_i \in Val(\boldsymbol{C}_i)$$

If the cluster graph is a cluster tree, this is exact

EP

- In EP, we make the same 2 approximations as in CGBP, but we also relax the local consistency constraint so that now cliques only have to agree on their expected sufficient statistics, not on their distributions

$$
\begin{array}{lll}
\text{Find} & Q & \\
\text{that maximize} & \tilde{F}[\tilde{P}_\Phi, Q] & \\
\text{subject to} & E_{\boldsymbol{S}_{i,j}\sim\mu_{i,j}}[\tau_{i,j}] = E_{\boldsymbol{S}_{i,j}\sim\beta_j}[\tau_{i,j}] \quad \forall (i\ \ j) \in \mathcal{E}_T & (11.41) \\
& \sum_{\boldsymbol{c}_i} \beta_i[\boldsymbol{c}_i] = 1 \qquad \forall i \in \mathcal{V}_T & (11.42) \\
& \sum_{\boldsymbol{s}_{i,j}} \mu_{i,j}[\boldsymbol{s}_{i,j}] = 1 \quad \forall (i\ \ j) \in \mathcal{E}_T & (11.43) \\
& \beta_i[\boldsymbol{c}_i] \geq 0 \qquad \forall i \in \mathcal{V}_T, \boldsymbol{c}_i \in Val(\boldsymbol{C}_i) & (11.44)
\end{array}
$$

Even if the CG is a tree, this is no longer exact (in general)

Variational methods

- The problems with BP and EP are
  - They do not monotonically increase a lower bound on ln Z
  - They may not converge (except convex BP)
- Let us now require Q to be a coherent probability distribution (of tractable form). Hence we can now compute the exact entropy and optimize the exact objective

$$
\begin{aligned}
D(Q||P) &= \ln Z - F(\tilde{P}, Q) \\
F(\tilde{P}, Q) &\stackrel{\text{def}}{=} H_Q(x) + \sum_i E_{C_i \sim Q} \ln \psi_i(C_i)
\end{aligned}
$$

- This always increases the lower bound and will always converge

Mean field approximation

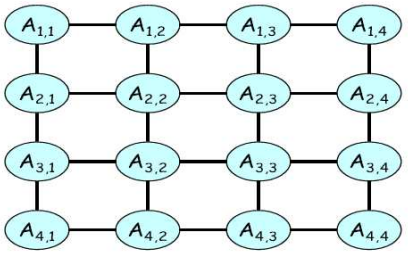
- Let us assume the approximate posterior is fully factorized

$$Q(x) = \prod_i Q_i(x_i)$$

- Then the objective (negative free energy) is

$$\begin{aligned} F(\tilde{P}, Q) &\stackrel{\text{def}}{=} H_Q(x) + \sum_c E_{X_c \sim Q} \ln \phi_c(X_c) \\ &= \sum_i H(Q_i) + \sum_c \sum_{x_c} (\prod_{i \in c} Q_i(x_{c,i})) \ln \phi_c(x_c) \end{aligned}$$

- Eg 4x4 grid $O(n_e K^2)$ for energy, $O(n_e K)$ for H



$$\begin{aligned} F[\tilde{P}_\Phi, Q] &= E_{\{A_{1,1}, A_{2,1}\} \sim Q}[\ln \phi(A_{1,1}, A_{2,1})] + E_{\{A_{2,1}, A_{3,1}\} \sim Q}[\ln \phi(A_{2,1}, A_{3,1})] + E_{\{A_{3,1}, A_{4,1}\} \sim Q}[\\ &\quad \ldots \\ &\quad E_Q[\ln \phi(A_{1,1}, A_{1,2})] + E_Q[\ln \phi(A_{1,2}, A_{1,3})] + E_Q[\ln \phi(A_{1,3}, A_{1,4})] + \\ &\quad \ldots \\ &\quad H_Q(A_{1,1}) + H_Q(A_{1,2}) + H_Q(A_{1,3}) + H_Q(A_{1,4}) + \\ &\quad \ldots \\ &\quad H_Q(A_{4,1}) + H_Q(A_{4,2}) + H_Q(A_{4,3}) + H_Q(A_{4,4}) \end{aligned}$$

Convexity

- Objective is concave in each arg (entropy is concave in each Q_i, expected energy is linear in Q_i)

$$F(\tilde{P}, Q) \quad = \quad \sum_i H(Q_i) + \sum_c \sum_{x_c} (\prod_{i \in c} Q_i(x_{c,i})) \ln \phi_c(x_c)$$

- The set of completely factorized distributions is not convex

$$Q^3(x) = \lambda \prod_i Q^1(x_i) + (1 - \lambda) \prod_i Q_i^2(x_i)$$

Not factorized

- Hence we are optimizing the objective over a non-convex space, and will be subject to local maxima
- Let us derive equations that characterize the fixed points. These could correspond to saddle points or local minima, but such points are unstable and unlikely to be the result of our iterative update scheme.

Notation

- Define

$$\langle f(x_h)\rangle \stackrel{\text{def}}{=} \sum_{x_h}\left[\prod_{i\in h} Q_i(x_i)\right] f(x_h)$$

$$\langle f(x_h)\rangle_{j,k} \stackrel{\text{def}}{=} \sum_{x_h\setminus x_j}\left[\prod_{i\in h, i\neq j} Q_i(x_i)\right] f(x_h|x_j=k)$$

$$\langle f(x_h)\rangle = \sum_k Q_j(x_j=k)\langle f(x_h)\rangle_{j,k}$$

$$\ln p(x_v) \geq \sum_c \langle \ln \phi_c(x_c)\rangle + \sum_i H(Q_i)$$

$$= \sum_k Q_j(k) \sum_c \langle \ln \phi_c(x_c)\rangle_{j,k} + H(Q_j) + \sum_{i\neq j} H(Q_i)$$

We mostly follow Tommi Jaakkola's notation rather than Daphne Koller's

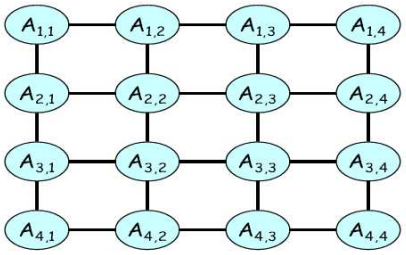
Mean field equations

$$
\begin{aligned}
\ln p(x_v) &\geq \sum_k Q_j(k) \sum_c \langle \ln \phi_c(x_c) \rangle_{j,k} + H(Q_j) + \sum_{i \neq j} H(Q_i) \\
&\stackrel{\text{def}}{=} L(Q_j) \\
S_{j,k} &\stackrel{\text{def}}{=} \sum_{c: j \in c} \langle \ln \phi_c(x_c) \rangle_{j,k} \\
L(Q_j) &= \sum_k Q_j(k)(S_{j,k} - \ln Q_j(k)) + C \\
L(Q_j, \lambda) &\stackrel{\text{def}}{=} L(Q_j) + \lambda(\sum_{k'} Q_j(k') - 1) \\
\frac{\partial}{\partial Q_j(k)} L(Q_j, \lambda) &= S_{j,k} - \ln Q_j(k) - 1 + \lambda = 0 \\
Q_j(k) &= \exp(S_{j,k}) \exp(\lambda - 1) \\
&= \frac{1}{Z_j} \exp(\sum_c \langle \ln \phi_c(x_c) \rangle_{j,k})
\end{aligned}
$$

Example: grid

$$Q(x_i) = \frac{1}{Z_i} \exp\left\{ \sum_{\phi: X_i \in Scope[\phi]} E_{(\boldsymbol{U}_\phi - \{X_i\}) \sim Q}[\ln \phi(\boldsymbol{U}_\phi, x_i)] \right\}$$

$$Q(a_{i,j}) = \frac{1}{Z_{i,j}} \exp\left\{ \begin{array}{l} \sum_{a_{i-1,j}} Q(a_{i-1,j}) \ln(\phi(a_{i-1,j}, a_{i,j})) + \\ \sum_{a_{i,j-1}} Q(a_{i,j-1}) \ln(\phi(a_{i,j-1}, a_{i,j})) + \\ \sum_{a_{i+1,j}} Q(a_{i+1,j}) \ln(\phi(a_{i,j}, a_{i+1,j})) + \\ \sum_{a_{i,j+1}} Q(a_{i,j+1}) \ln(\phi(a_{i,j}, a_{i,j+1})) \end{array} \right\}.$$



EM

- Suppose we want to find a MAP estimate

$$\max_{\theta} \log p(\theta) + \sum_{n} \log p(x_n|\theta)$$

- If we have latent variables Z we can use EM
- E step: compute expected complete data log joint

$$f(\theta, \theta_{old}) \quad = \quad \log p(\theta) + \sum_{n=1}^{N} \sum_{z} p(z|x_n, \theta_{old}) \log p(z, x_n|\theta)$$

- M step: set

$$\theta_{new} = \arg\max f(\theta, \theta_{old})$$

Variational EM

- Consider the negative free energy

$$F(x, Q, \theta) = \sum_z Q(z) \log p(x, z|\theta) + H(Q)$$

- Earlier we showed this is a lower bound on the log-likelihood

$$\begin{aligned} F(x, Q, \theta) &= \ln Z(x, \theta) - D(Q||p(z|x, \theta)) \\ \log p(x|\theta) &= \ln Z = \max_Q F(x, Q, \theta) = F(x, Q^*, \theta) \geq F(x, Q, \theta) \end{aligned}$$

- Where the bound is tight if $Q^*(z) = p(z|x, \theta)$
- E step: find $Q_n(z)$ that maximize

$$F(x_n, Q_n, \theta_{old})$$

- M step: find \theta that maximize

$$\log p(\theta) + \sum_n F(x_n, Q_n, \theta)$$

Variational EM

- An exact E step is equivalent to setting

$$Q_n(z) = p(z|x_n, \theta_{old})$$

- The corresponding M step maximizes

$$\begin{aligned}\sum_n F(x_n, Q_n, \theta) &= \sum_n [\sum_z p(z|x_n, \theta_{old}) \log p(z, x_n|\theta)] + H(Q_n) \\ &= f(\theta, \theta_{old}) + \sum_n H(Q_n)\end{aligned}$$

- Since $H(Q_n)$ is independent of θ, this reduces to the standard EM algorithm.
- Generalized EM merely increases (not maximizes) θ in the M step.
- Similarly we can simply improve Q_n in the E step

Neal and Hinton, "A new view of the EM algorithm", 1998

Variational Bayes

- We can replace the point estimate of θ with a distribution and try to minimize

$$D(Q(z_{1:N}, \theta | x_{1:N}) || p(z_{1:N}, \theta | x_{1:N}))$$

- The distinction between E and M vanishes: we are just doing sequential updates of $Q(Z_n)$ and $Q(\theta)$
- This gives us the benefits of being Bayesian for the same computational speed as EM

VB for univariate Gaussian

$$q_j^\star(\mathbf{Z}_j) = \frac{\exp\left(\mathbb{E}_{i\neq j}[\ln p(\mathbf{X},\mathbf{Z})]\right)}{\int \exp\left(\mathbb{E}_{i\neq j}[\ln p(\mathbf{X},\mathbf{Z})]\right)\,\mathrm{d}\mathbf{Z}_j}. \qquad \ln q_j^\star(\mathbf{Z}_j) = \mathbb{E}_{i\neq j}[\ln p(\mathbf{X},\mathbf{Z})] + \text{const.}$$

$$p(\mathcal{D}|\mu,\tau) = \left(\frac{\tau}{2\pi}\right)^{N/2} \exp\left\{-\frac{\tau}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}. \qquad \begin{aligned} p(\mu|\tau) &= \mathcal{N}\left(\mu|\mu_0,(\lambda_0\tau)^{-1}\right) \\ p(\tau) &= \mathrm{Gam}(\tau|a_0,b_0) \end{aligned}$$

$$q(\mu,\tau) = q_\mu(\mu)q_\tau(\tau).$$

Gaussian

$$\begin{aligned} \ln q_\mu^\star(\mu) &= \mathbb{E}_\tau\left[\ln p(\mathcal{D}|\mu,\tau) + \ln p(\mu|\tau)\right] + \text{const.} \\ &= -\frac{\mathbb{E}[\tau]}{2}\left\{\lambda_0(\mu-\mu_0)^2 + \sum_{n=1}^{N}(x_n-\mu)^2\right\} + \text{const.} \end{aligned}$$

$$\begin{aligned} \mu_N &= \frac{\lambda_0\mu_0 + N\overline{x}}{\lambda_0+N} \\ \lambda_N &= (\lambda_0+N)\mathbb{E}[\tau]. \end{aligned}$$

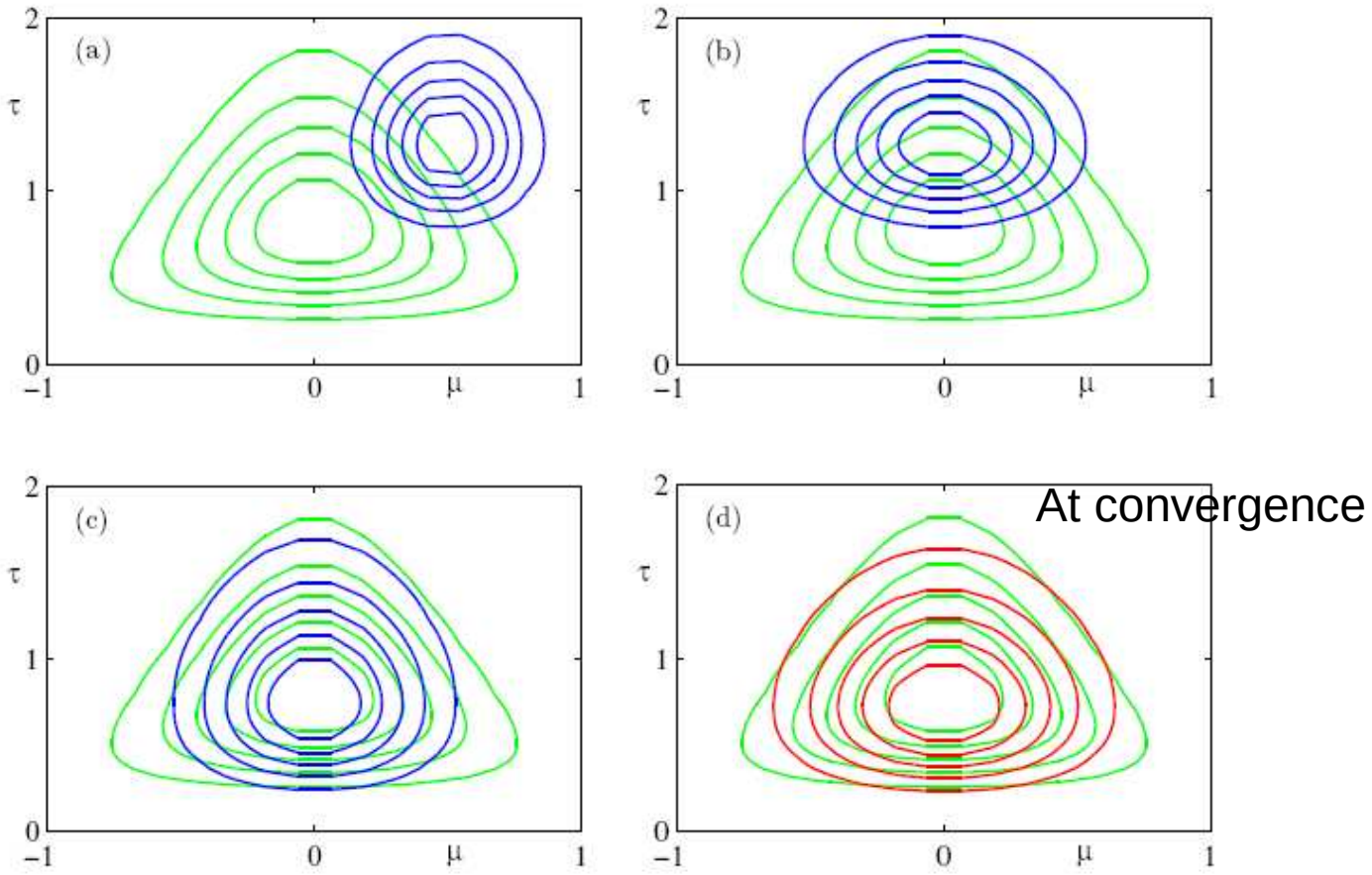
Gamma

$$\begin{aligned} \ln q_\tau^\star(\tau) &= \mathbb{E}_\mu\left[\ln p(\mathcal{D}|\mu,\tau) + \ln p(\mu|\tau)\right] + \ln p(\tau) + \text{const.} \\ &= (a_0-1)\ln\tau - b_0\tau + \frac{N+1}{2}\ln\tau \\ &\quad -\frac{\tau}{2}\mathbb{E}_\mu\left[\sum_{n=1}^{N}(x_n-\mu)^2 + \lambda_0(\mu-\mu_0)^2\right] + \text{const.} \end{aligned}$$

$$\begin{aligned} a_N &= a_0 + \frac{N+1}{2} \\ b_N &= b_0 + \frac{1}{2}\mathbb{E}_\mu\left[\sum_n (x_n-\mu)^2 + \lambda_0(\mu-\mu_0)^2\right]. \end{aligned}$$

Bishop p471

VB for univariate Gaussian



At convergence

Green = exact posterior (NormalGamma), blue = factorized approximation

VB for mixtures of Gaussians

Inference

$$q(\mathbf{Z}, \pi, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\mathbf{Z}) q(\pi, \boldsymbol{\mu}, \boldsymbol{\Lambda}).$$

$$\ln q^\star(\mathbf{Z}) = \mathbb{E}_{\pi, \boldsymbol{\mu}, \boldsymbol{\Lambda}}[\ln p(\mathbf{X}, \mathbf{Z}, \pi, \boldsymbol{\mu}, \boldsymbol{\Lambda})] + \text{const.}$$

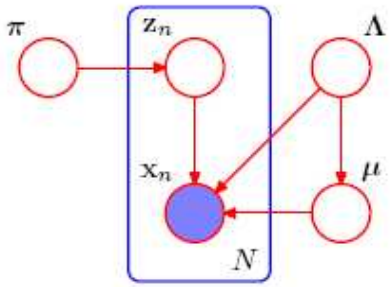
$$\ln q^\star(\mathbf{Z}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \ln \rho_{nk} + \text{const.}$$

$$\begin{aligned} \ln \rho_{nk} &= \mathbb{E}[\ln \pi_k] + \frac{1}{2}\mathbb{E}\left[\ln |\boldsymbol{\Lambda}_k|\right] - \frac{D}{2}\ln(2\pi) \\ &\quad - \frac{1}{2}\mathbb{E}_{\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k}\left[(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Lambda}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)\right] \end{aligned}$$

$$q^\star(\mathbf{Z}) \propto \prod_{n=1}^{N} \prod_{k=1}^{K} \rho_{nk}^{z_{nk}}.$$

Multinomial (soft responsibilities), as in EM,
except we used expected parameters rather than plug-in

Model



$$p(\mathbf{X}, \mathbf{Z}, \pi, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\mathbf{Z}|\pi) p(\pi) p(\boldsymbol{\mu}, \boldsymbol{\Lambda})$$

$$p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}\right)^{z_{nk}}$$

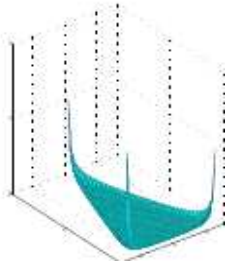
$$p(\mathbf{Z}|\pi) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}}.$$

$$p(\pi) = \mathrm{Dir}(\pi|\boldsymbol{\alpha}_0) = C(\boldsymbol{\alpha}_0) \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1}$$

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{\mu}_k | \mathbf{m}_0, (\beta_0 \boldsymbol{\Lambda}_k)^{-1}\right) \mathcal{W}(\boldsymbol{\Lambda}_k | \mathbf{W}_0, \nu_0)$$

Automatic model selection

- Recall $\pi \sim \mathrm{Dir}(\alpha)$. If $\alpha << 1$, we prefers skewed π and hence sparse z.



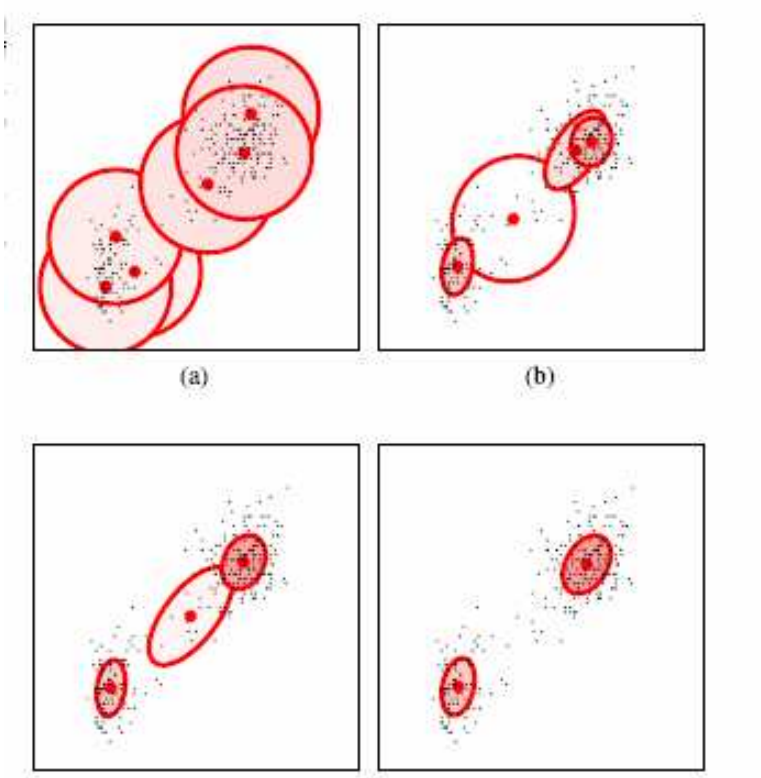
- MAP estimate from regular EM is

$$\hat{\pi}_k = \frac{\sum_n r_{nk} + \alpha_k - 1}{\sum_k (r_{nk} + \alpha_k - 1)} = \frac{N_k + \alpha - 1}{N + K\alpha - K}$$

- Posterior mean estimate from VB is

$$\hat{\pi}_k = \frac{\sum_n r_{nk} + \alpha_k}{\sum_k (r_{nk} + \alpha_k)} = \frac{N_k + \alpha}{N + K\alpha} \rightarrow \frac{\alpha}{N + K\alpha} \rightarrow 0$$

Selecting K with one run of VB



Variational message passing

- Consider a DAG model

$$p(\mathbf{x}) = \prod_i p(\mathbf{x}_i|\mathrm{pa}_i)$$

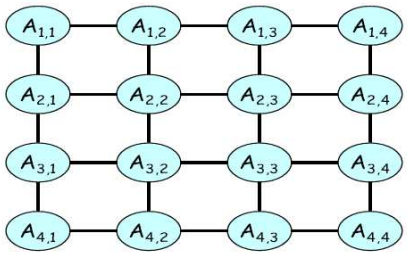
- The mean field equations are

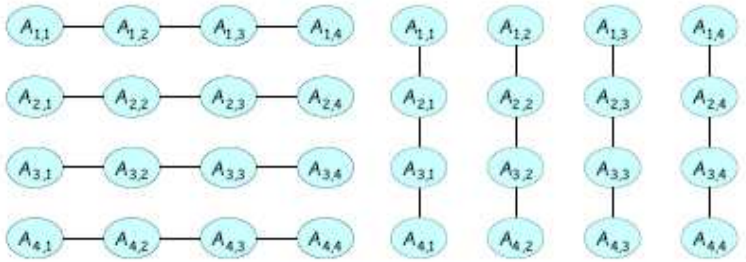
$$\ln q_j^*(\mathbf{x}_j) = \mathbf{E}_{i \neq j}\left[\sum_i \ln p(\mathbf{x}_i|\mathrm{pa}_i)\right] + \text{const.}$$

- The only terms that depend on x_j are in x_j's Markov blanket
- If all CPDs have conjugate-exponential form, the VB updates can be converted into a msg passing algorithm
- VIBES software (John Winn)

Structured variational approx

- Rather than assuming Q is fully factorized, we can use any structure for which computing the expectations of ln ϕ_c and the entropy is tractable





$$Q(\mathcal{X}) = \frac{1}{Z_Q} \prod_{j=1}^{J} \psi_j$$

ϕ = model, ψ = approx

Corollary 11.5.13: *If* $Q(\mathcal{X}) = \frac{1}{Z_Q} \prod_j \psi_j$, *then the potential* ψ_j *is a stationary point of the energy functional if and only if:*

$$\psi_j(\boldsymbol{c}_j) \propto \exp\left\{ \mathbb{E}_Q\left[\ln \tilde{P}_\Phi \mid \boldsymbol{c}_j\right] - \sum_{k \neq j} \boldsymbol{E}_Q[\ln \psi_k \mid \boldsymbol{c}_j] \right\}. \qquad (11.59)$$

Stat 521A
Lecture 11

Outline

- Forward sampling (12.1)
- Importance sampling (12.2)
- MCMC (12.3)
- Collapsed particles (12.4)
- Deterministic search (12.5)

Monte Carlo integration

- The goal is to approximate E[f(X)] for some function f eg f(X) = I(X_i=k), so E[f(X)] = p(X_i=k)
- Usually we take expectations wrt p(X|e), where e is the evidence
- If we can draw samples X ~ p(X|e), we can evaluate the expectation thus:

$$E_P[f] \approx \frac{1}{M} \sum_{m=1}^{M} f(x[m]).$$

Error analysis

Let $\mu = E[h(X)]$ be the exact expected value, and $\hat{f}_S$ a Monte Carlo approximation based on S samples. One can show a central-limit type theorem

$$(\hat{f}_S - \mu) \rightarrow \mathcal{N}(0, \frac{\sigma^2}{S}) \tag{16.6}$$

where $\sigma^2 = \text{Var}\,[h(X)]$. The latter quantity can itself be estimated by MC:

$$\hat{\sigma}^2 = \frac{1}{S}\sum_{s=1}^{S}(f(\theta^s) - \hat{f}_S)^2 \tag{16.7}$$

Then we have

$$P\left\{\mu - 1.96\frac{\hat{\sigma}^2}{\sqrt{S}} \leq \hat{f}_s \leq \mu + 1.96\frac{\hat{\sigma}^2}{\sqrt{S}}\right\} \approx 0.95 \tag{16.8}$$

The term $\sqrt{\frac{\hat{\sigma}^2}{S}}$ is called the (numerical or empirical) **standard error**. Thus we see that the error in our MC estimate goes down at a rate of $1/\sqrt{S}$.

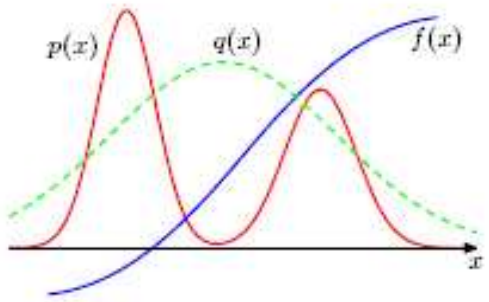
Forward sampling

- To sample from the prior p(x) of a DGM is easy: just sample each node in topological order, conditional on its parents
- To sample from the prior of a UGM is much harder
- Usually we want to sample from the posterior p(x|e)
- We can use forwards sampling and throw away all samples that are inconsistent with e; this is called **rejection sampling** ("logic sampling" in the context of discrete DGMs) and is very inefficient

Unnormalized importance sampling

- Often sampling from P is hard
- Suppose we sample from a proposal distribution Q instead. All we require is that P(x)>0 => Q(x)>0

$$
\begin{aligned}
\boldsymbol{E}_{Q(\boldsymbol{X})}\left[f(\boldsymbol{X})\frac{P(\boldsymbol{X})}{Q(\boldsymbol{X})}\right] &= \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) f(\boldsymbol{x}) \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} \\
&= \sum_{\boldsymbol{x}} f(\boldsymbol{x}) P(\boldsymbol{x}) \\
&= \boldsymbol{E}_{P(\boldsymbol{X})}[f(\boldsymbol{X})]
\end{aligned}
$$



$$
\hat{\boldsymbol{E}}_{\mathcal{D}}(f) = \frac{1}{M}\sum_{m=1}^{M} f(\boldsymbol{x}[m])\frac{P(\boldsymbol{x}[m])}{Q(\boldsymbol{x}[m])}.
$$

Unbiased estimator

Variance

- Variance of estimator given by

$$\begin{aligned}\sigma_Q^2 &= E_{Q(\boldsymbol{X})}\left[(f(\boldsymbol{X})w(\boldsymbol{X}))^2\right] - E_{Q(\boldsymbol{X})}\left[(f(\boldsymbol{X})w(\boldsymbol{X}))\right]^2 \\ &= E_{Q(\boldsymbol{X})}\left[(f(\boldsymbol{X})w(\boldsymbol{X}))^2\right] - (E_{P(\boldsymbol{X})}[f(\boldsymbol{X})])^2.\end{aligned}$$

- Let f(X)=1. Then variance is variance of P(X)/Q(X)

$$E_{Q(\boldsymbol{X})}\left[\left(\frac{P(\boldsymbol{X})}{Q(\boldsymbol{X})}\right)^2\right] - \left(E_{Q(\boldsymbol{X})}\left[\frac{P(\boldsymbol{X})}{Q(\boldsymbol{X})}\right]\right)^2,$$

- Variance will be large if Q(x) << P(x) f(x)

Normalized importance sampling

- Often we only know P'(x) = α P(x) with unknown α
- Define

$$w(\boldsymbol{X}) = \frac{\tilde{P}(\boldsymbol{X})}{Q(\boldsymbol{X})}.$$

- Then

$$E_{Q(\boldsymbol{X})}[w(\boldsymbol{X})] = \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) \frac{\tilde{P}(\boldsymbol{x})}{Q(\boldsymbol{x})} = \sum_{\boldsymbol{x}} \tilde{P}(\boldsymbol{x}) = \alpha.$$

$$\begin{aligned}
E_{P(\boldsymbol{X})}[f(\boldsymbol{X})] &= \sum_{\boldsymbol{x}} P(\boldsymbol{x}) f(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) f(\boldsymbol{x}) \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} \\
&= \frac{1}{\alpha} \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) f(\boldsymbol{x}) \frac{\tilde{P}(\boldsymbol{x})}{Q(\boldsymbol{x})} \\
&= \frac{1}{\alpha} E_{Q(\boldsymbol{X})}[f(\boldsymbol{X}) w(\boldsymbol{X})] \\
&= \frac{E_{Q(\boldsymbol{X})}[f(\boldsymbol{X}) w(\boldsymbol{X})]}{E_{Q(\boldsymbol{X})}[w(\boldsymbol{X})]}
\end{aligned}$$

$$\hat{E}_{\mathcal{D}}(f) = \frac{\sum_{m=1}^{M} f(\boldsymbol{x}[m]) w(\boldsymbol{x}[m])}{\sum_{m=1}^{M} w(\boldsymbol{x}[m])}$$

Bias

- Biased estimator

$$\hat{E}_{\mathcal{D}}(f) = \frac{\sum_{m=1}^{M} f(\boldsymbol{x}[m])w(\boldsymbol{x}[m])}{\sum_{m=1}^{M} w(\boldsymbol{x}[m])}$$

- Eg M=1. x[1] ~ Q has wrong mean

$$\frac{f(\boldsymbol{x}[1])w(\boldsymbol{x}[1])}{w(\boldsymbol{x}[1])} = f(\boldsymbol{x}[1]).$$

- But bias -> 0 as 1/M since numerator and denominator are both unbiased

Variance

- Variance ->0 as 1/M

$$\mathbb{V}\mathrm{ar}_P\left[\hat{E}_{\mathcal{D}}(f(\boldsymbol{X}))\right] \approx \frac{1}{M}\mathbb{V}\mathrm{ar}_P[f(\boldsymbol{X})](1 + \mathbb{V}\mathrm{ar}_Q[w(\boldsymbol{X})]),$$

- Variance of optimal estimator is $\hat{\mathbb{V}}\mathrm{ar}_P[f(\boldsymbol{X})]/M$
- Ratio is $\dfrac{1}{1 + \mathbb{V}\mathrm{ar}_Q[w(\boldsymbol{x})]}$.

- Effective sample size

$$\begin{aligned} M_{\text{eff}} &= \frac{M}{1 + \mathbb{V}\mathrm{ar}[\mathcal{D}]} \\ \mathbb{V}\mathrm{ar}[\mathcal{D}] &= \sum_{m=1}^{M} w(\boldsymbol{x}[m])^2 - (\sum_{m=1}^{M} w(\boldsymbol{x}[m]))^2. \end{aligned}$$

Likelihood weighting

- Let us apply importance sampling to a DGM where the proposal is as follows: do forwards in the mutilated DGM where observed nodes are clamped to Z=z
- Prop 12.2.5. Weights are $w(\xi) = \frac{P_{\mathcal{B}}(\xi)}{P_{\mathcal{B}_{\boldsymbol{Z}=\boldsymbol{z}}}(\xi)}$.

Algorithm 12.2 Likelihood Weighted Particle Generation

```
Procedure LW-sample (
  B,     // Bayesian network over X
  Z = z  // Event in the network
)
1   Let X_1, ..., X_n be a topological ordering of X
2   w ← 1
3   for i = 1, ..., n
4     u_i ← x⟨Pa_{X_i}⟩   // Assignment to Pa_{X_i} in x_1, ..., x_{i-1}
5     if X_i ∉ Z then
6       Sample x_i from P(X_i | u_i)
7     else
8       x_i ← z⟨X_i⟩   // Assignment to X_i in z
9       w ← w · P(x_i | u_i)   // Multiply weight by probability of desired value
10  return (x_1, ..., x_n), w
```

Using LW weights

- Recall that E[w(X)] = α = p(Z=z)
- Ratio likelihood weighting: run LW twice for each y

$$\hat{P}_{\mathcal{D}}(y \mid e) = \frac{\hat{P}_{\mathcal{D}}(y, e)}{\hat{P}_{\mathcal{D}'}(e)} = \frac{1/M \sum_{m=1}^{M} w[m]}{1/M' \sum_{m=1}^{M'} w'[m]}.$$

- Normalized likelihood weighting: run LW once, and use samples to evaluate any query

$$\hat{P}_{\mathcal{D}}(y \mid e) = \frac{\sum_{m=1}^{M} w[m] \mathbf{1}\{y[m] = y\}}{\sum_{m=1}^{M} w[m]}.$$

= p(y,z)/ p(z)

Efficiency

- Although LW does not “throw away” samples that are inconsistent with e, it down weights them
- If the evidence is at the leaves, the samples are drawn from the prior and may be assigned low weight
- Backward importance sampling (evidence reversal): if X->Y=y, sample from $Q(X) \propto p(Y=y|X)$
- Importance sampling does not scale well to high dimensions, because hard to make Q match P

MCMC

- Markov Chain Monte Carlo constructs a Markov chain whose stationary distribution is equal to the posterior p(x|e).
- Metropolis Hastings: only need proposal Q(x'|x) and ability to evaluate $\pi(x) = p(x,e) \propto p(x|e)$
- Gibbs: only need ability to sample full conditionals p(xi|x(-i),e)

Metropolis Hastings algorithm

- We propose q(x'|x) and evaluate $\alpha=\pi(x')/\pi(x)$
- If α >= 1, we accept, otherwise we accept wp r
- Always accept uphill move, occasionally accept downhill move
- If proposal is asymmetric, need Hastings correction

$$\begin{aligned} \alpha &= \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)} = \frac{\pi(x')/q(x'|x)}{\pi(x)/q(x|x')} \\ r &= \min(1,\alpha) \end{aligned}$$

MH pseudocode

1 Initialize x^0
2 **for** $s = 0, 1, 2, \ldots$ **do**
3 Sample $x' \sim q(x'|x)$
4 Compute acceptance probability

$$\alpha = \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)} = \frac{\pi(x')/q(x'|x)}{\pi(x)/q(x|x')}$$

5 Compute $r = \min(1, \alpha)$
6 Set new sample to

$$x^{s+1} = \begin{cases} x' & \text{with probability } r \\ x^s & \text{with probability } 1 - r \end{cases}$$

Why MH works

- MH generates a MC with this transition matrix

$$p(x'|x) = \begin{cases} q(x'|x)r(x'|x) & \text{if } x' \neq x \\ q(x|x) + \sum_{x' \neq x} q(x'|x)(1 - r(x'|x)) & \text{otherwise} \end{cases} \quad (16.21)$$

Theorem 16.2.1. *If the transition matrix defined by the MH algorithm (given by Equation 16.21) is ergodic and irreducible, then π is its unique limiting distribution.*

Proof. Consider two states x and x'. Either

$$\pi(x)q(x'|x) < \pi(x')q(x|x') \quad (16.22)$$

or

$$\pi(x)q(x'|x) > \pi(x')q(x|x') \quad (16.23)$$

We will ignore ties (which occur with probability zero for continuous distributions). Without loss of generality, assume that $\pi(x)q(x'|x) > \pi(x')q(x|x')$. Hence

$$\alpha(x'|x) = \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)} < 1 \quad (16.24)$$

Proof cont'd

Hence we have $r(x'|x) = \alpha(x'|x)$ and $r(x|x') = 1$.

Now to move from x to x' we must first propose x' and then accept it. Hence

$$p(x'|x) = q(x'|x)r(x'|x) = q(x'|x)\frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)} = \frac{\pi(x')}{\pi(x)}q(x|x') \tag{16.25}$$

Hence

$$\pi(x)p(x'|x) = \pi(x')q(x|x') \tag{16.26}$$

The backwards probability is

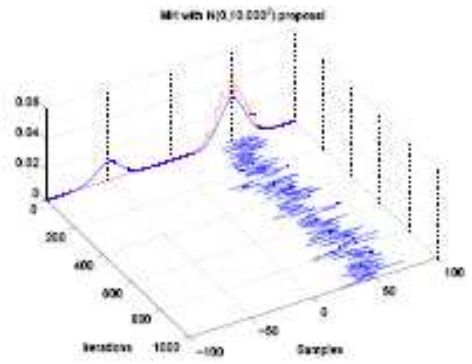
$$p(x|x') = q(x|x')r(x|x') = q(x|x') \tag{16.27}$$

since $r(x|x') = 1$. Inserting this into Equation 16.26 we get

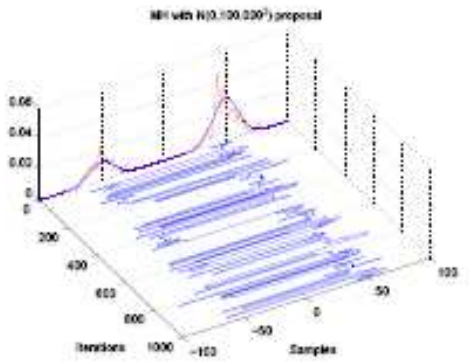
$$\pi(x)p(x'|x) = \pi(x')p(x|x') \tag{16.28}$$

so detailed balance holds. Hence, from Theorem ??, π is the stationary distribution. ∎

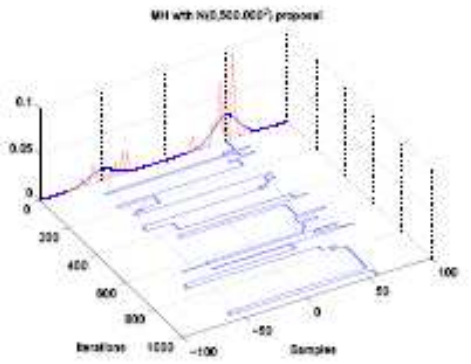
Proposal distributions



(a)

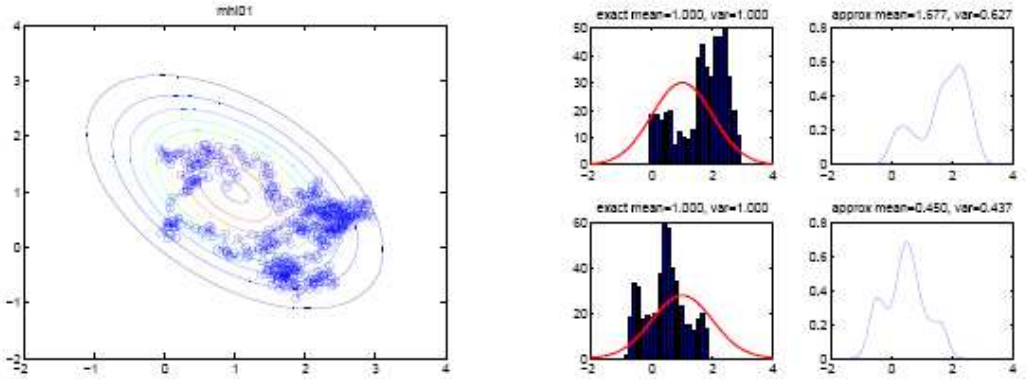


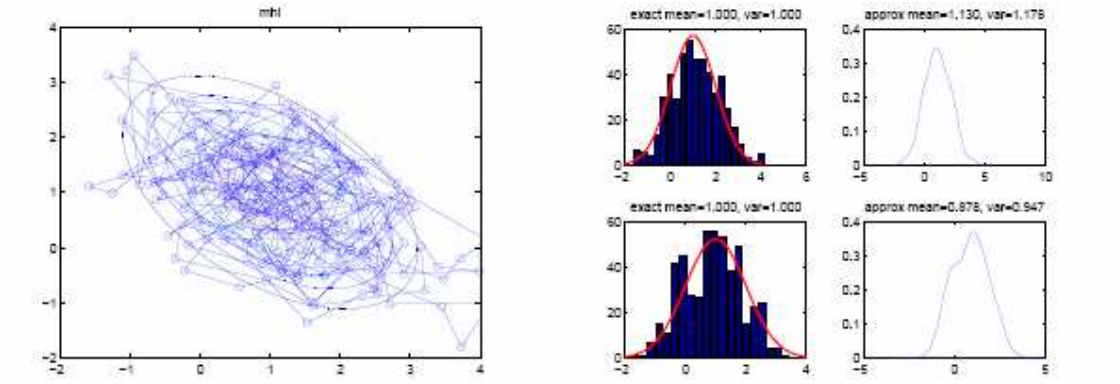
(b)



(c)

Proposal distributions





Methods for choosing proposals

- Initialize chain at a local mode (found with an optimizer)
- Gaussian random walk, with covariance = Hessian
- Mixture of base kernels, corresponding to different heuristic algorithms

$$q(x'|x) = \sum_{k=1}^{K} w_k q_k(x'|x)$$

- Adaptive MCMC: modify Gaussian covariance online

Gibbs sampling

- Sample each node given all others, from its full conditional

1. $x_1^{s+1} \sim p(x_1|x_2^s, \ldots, x_d^s)$
2. $x_2^{s+1} \sim p(x_2|x_1^{s+1}, x_3^s, \ldots, x_d^s)$
3. $x_i^{s+1} \sim p(x_i|x_{1:i-1}^{s+1}, x_{i+1:d}^s)$
4. $x_d^{s+1} \sim p(x_d|x_1^{s+1}, \ldots, x_{d-1}^{s+1})$

- This is MH with the following proposal

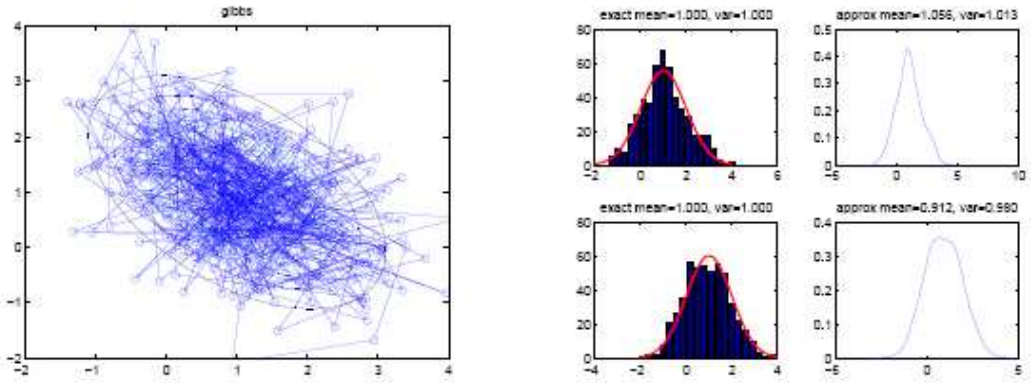
$$q((x_i', \mathbf{x}_{-i})|(x_i, \mathbf{x}_{-i})) = p(x_i'|\mathbf{x}_{-i})$$

- Acceptance rate is 100%

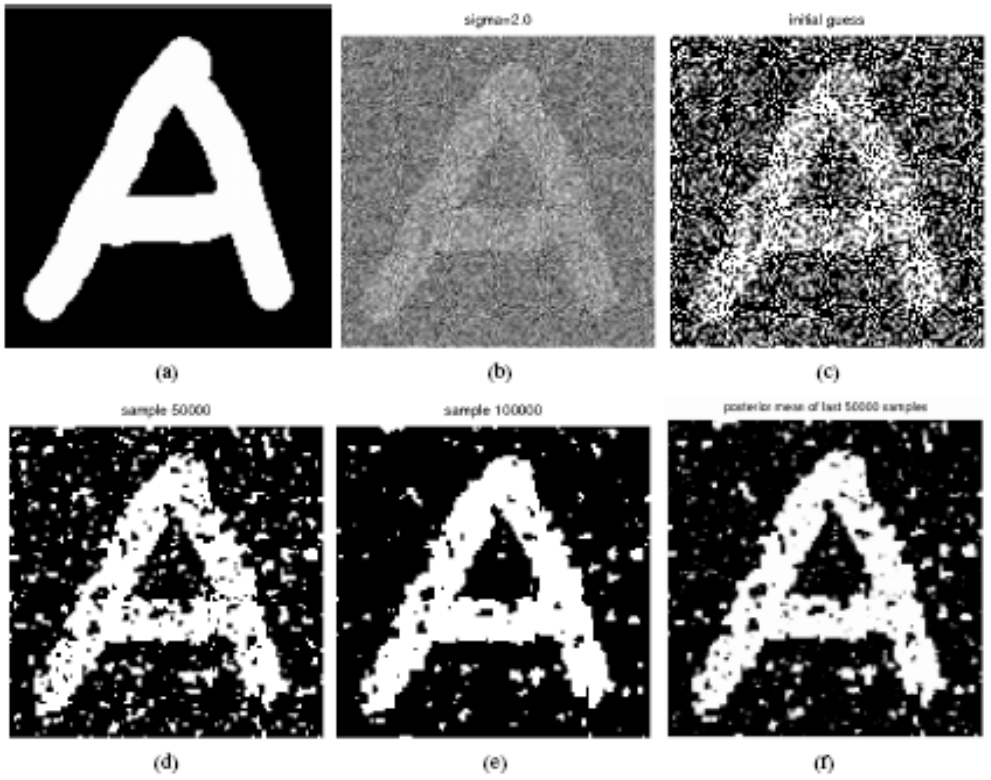
$$\alpha = \frac{p(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p(\mathbf{x})q(\mathbf{x}'|\mathbf{x})} = \frac{p(x_i'|\mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x_i|\mathbf{x}_{-i})}{p(x_i|\mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x_i'|\mathbf{x}_{-i})} = 1$$

Gibbs for bivariate Gaussian

$$
\begin{aligned}
p(x_1|x_2) &= \mathcal{N}(x_1|\mu_{1|2}, \Sigma_{1|2}) \\
\mu_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \\
\Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}
\end{aligned}
$$



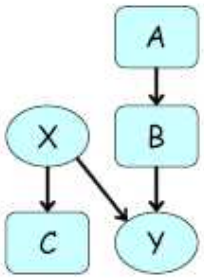
Gibbs for Ising



$$p(x_t = +1 | \mathrm{x}_{-t}, \mathrm{y}, \theta) = \frac{\exp[J w_t] \phi_t(+1, y_t)}{\exp[J w_t] \phi_t(+1, y_t) + \exp[-J w_t] \phi_t(-1, y_t)}$$
$$= \sigma(2J \log \frac{\phi_t(+1)}{\phi_t(-1)})$$

BUGS

- Bayesian Updating using Gibbs Sampling

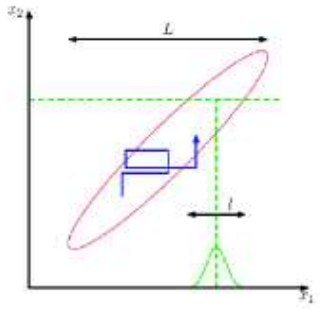



```
var
   A, B, C, X, Y, mu, tau, p[2,3], q;

p = ...
A ~ dbern(0.3)
B ~ dcat(p[A,1:3])
X ~ dnorm(-1,0.25)
mu <- 3*X+B^2
tau <- 1/X^2
Y ~ dnorm(mu,tau)
logit(q) <- 4*X + 2
C ~ dbern(q)
```

Single vs block updates

- Gibbs does single site updating which can move slowly, or even get stuck (eg XOR)
- Blocked Gibbs sampling samples multiple variables at once



Accuracy

- Even though the samples are correlated, we have a CLT-type result

$$(\mu - \hat{\mu}) \to \mathcal{N}(0, \sigma^2)$$

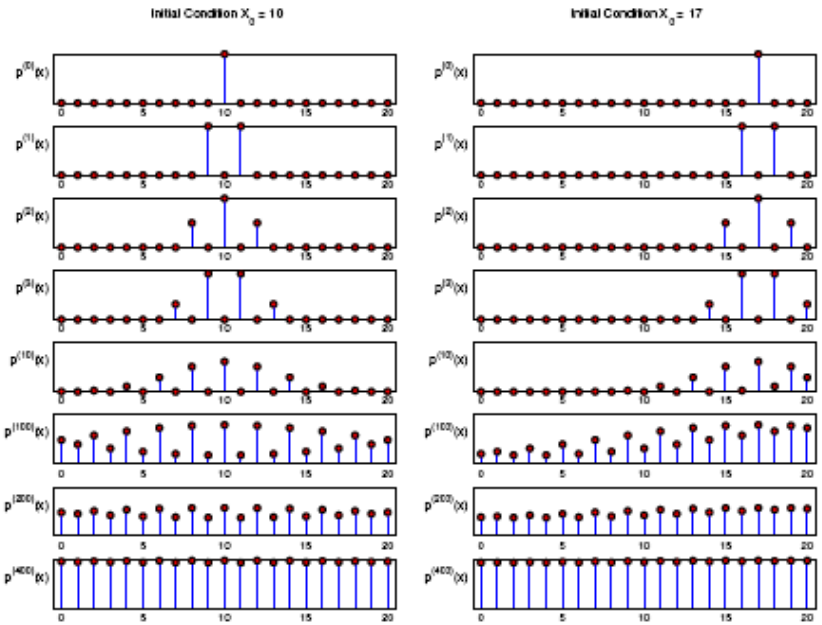
$$\sigma^2 = \text{Var}\left[f(X)\right] + 2\sum_{\ell=0}^{\infty} \text{Cov}[f(X_t), f(X_{t+\ell})]$$

- Autocorrelation function

$$\rho(\ell) = \frac{\text{Cov}[f(X_t), f(X_{t+\ell})]}{\sigma^2}$$

Mixing time

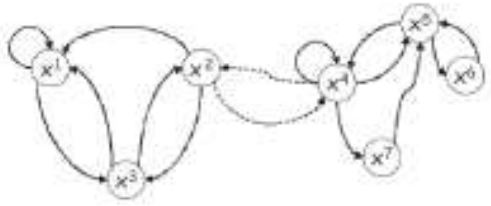
- Mixing time is time to reach stationary distribution



Samples drawn before convergence (during burnin phase) should be discarded

Conductance

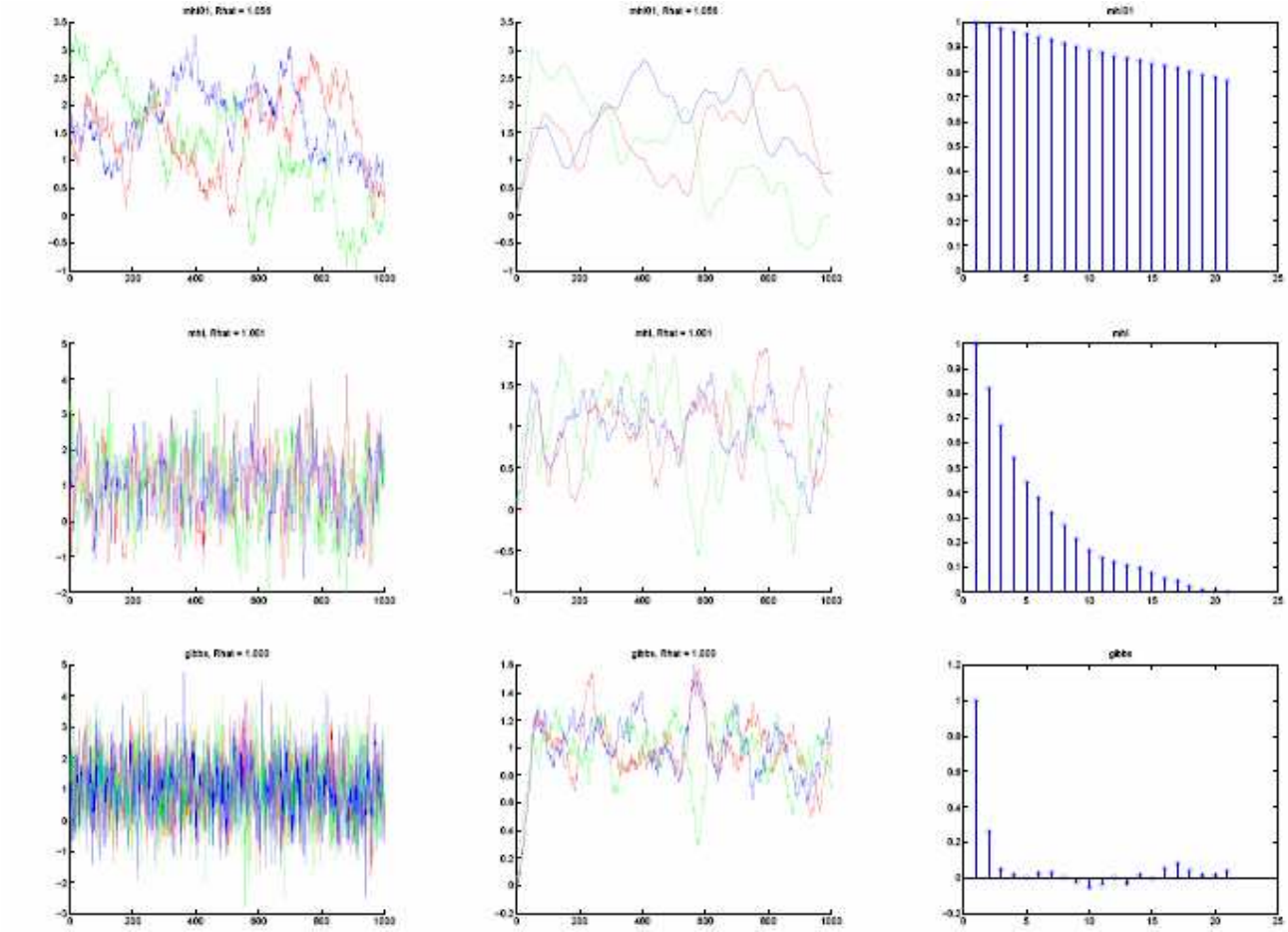
- Mixing time depends on eigengap, $\gamma=\lambda_1-\lambda_2$
- Hard to compute
- Can develop bounds based on the conductance (which is low if there are narrow bottlenecks in the state space)



Convergence

- 2 issues
  - Speeding up convergence
  - Determining if convergence has happened
- Speedups: various tricks, see later
- Determining: various heuristics

Traceplots and ACF



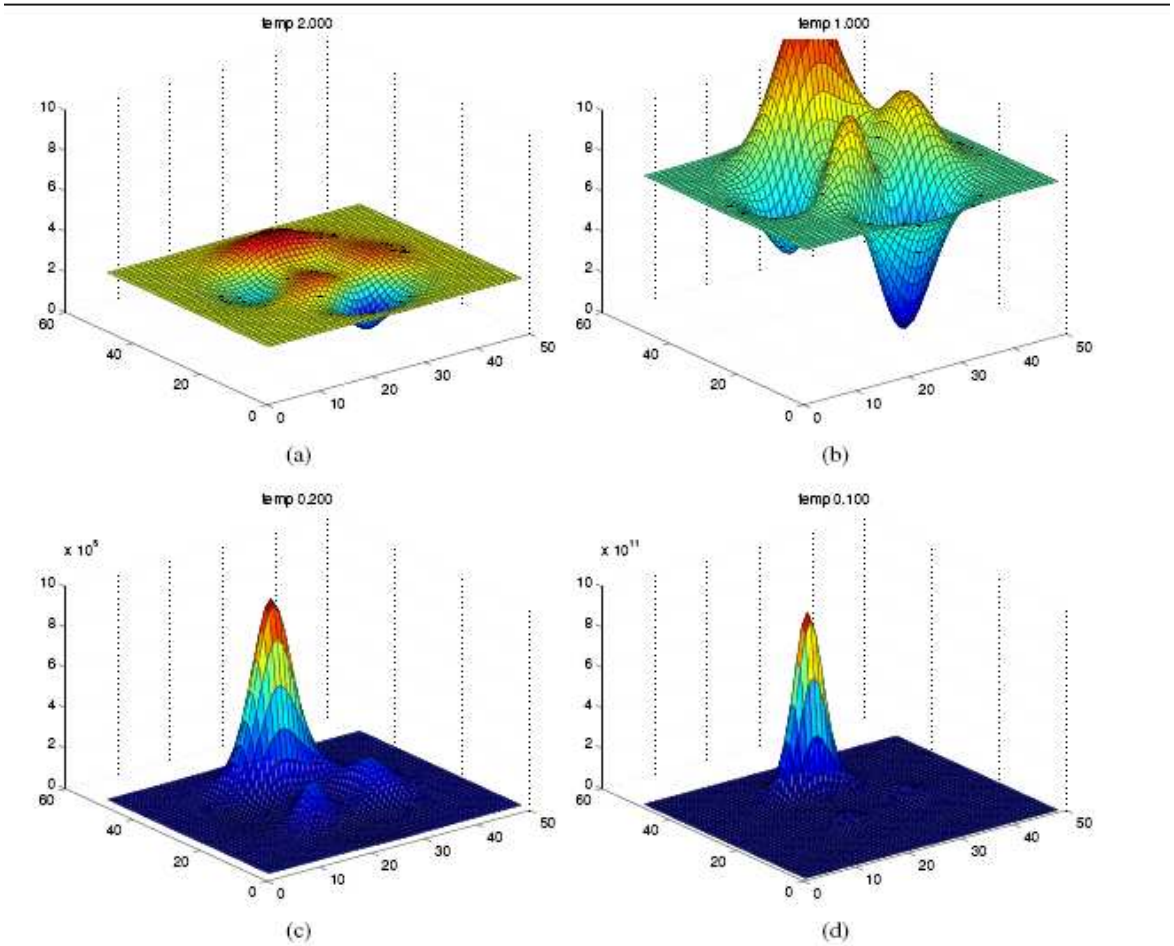
EPSR

- Start 3 chains from different states, run them for a while, check if variance within a chain is comparable to variance between chains
- Can be formalized using the Rhat statistic (estimated potential scale reduction).
- If Rhat ~ 1.0 for a specific f(X), then it suggest that the chain has converged.
- Can compute Rhat for multiple features f(X).

Simulated annealing

- Global optimization method
- Raise surface to a temperature to smooth it out/ kill off the non-peaks

$$\pi_s(x) = \pi(x)^{1/T_s}$$



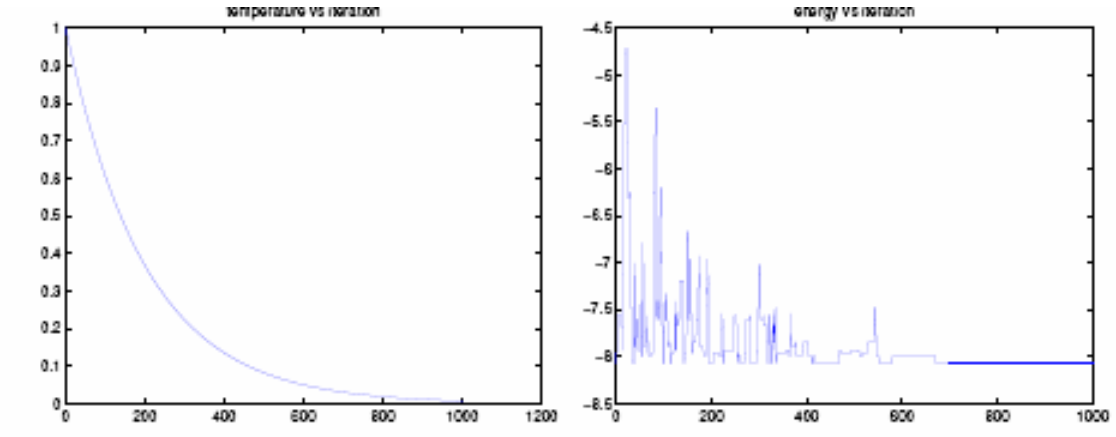
Simulated annealing

- $\pi(x) = \exp(-E(x))$, E(x)=energy (+ve or –ve)

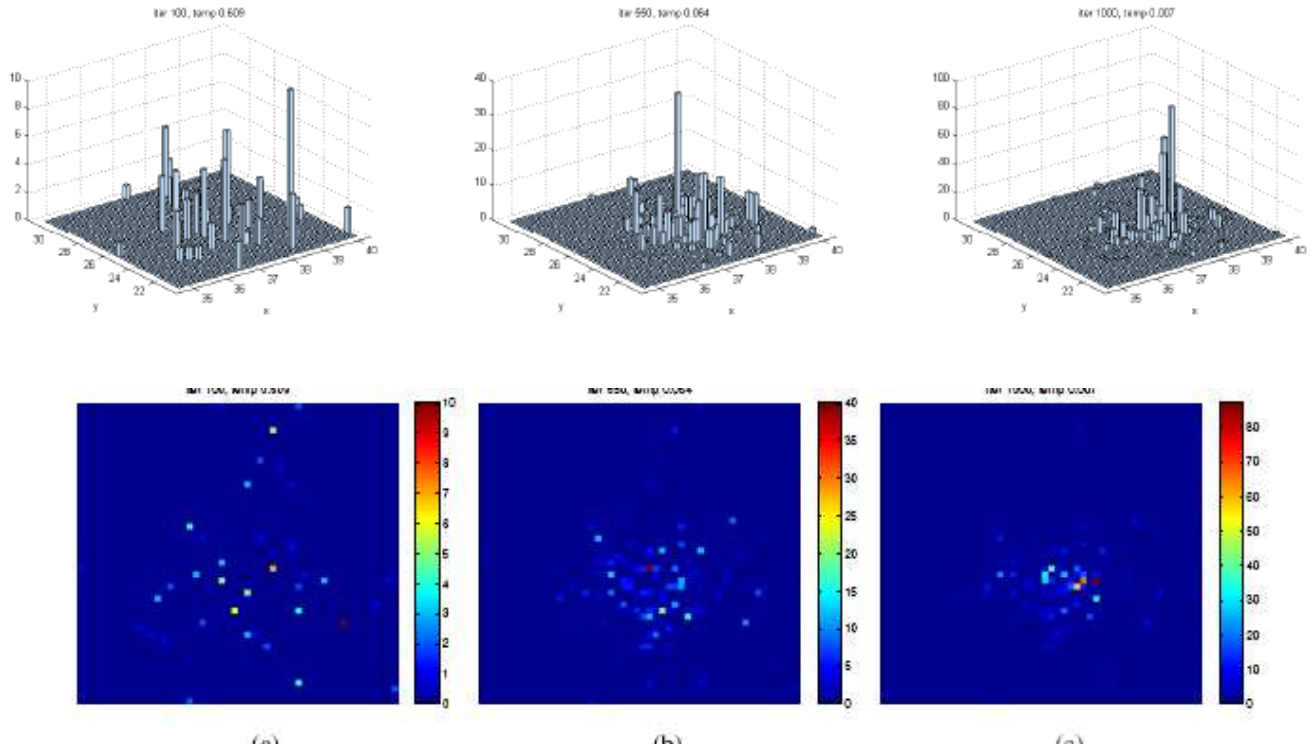
$$\begin{aligned}\alpha &= \frac{\pi(x')^{1/T_s}}{\pi(x)^{1/T_s}} \\ &= \frac{\exp(-E(x'))^{1/T_s}}{\exp(-E(x))^{1/T_s}} \\ &= \exp\left((E(x) - E(x'))/T_s\right)\end{aligned}$$

- Cooling schedule

$$T_s = T_0 C^s$$



Samples from SA



Parallel tempering

- Run multiple chains at different temperatures
- Let them swap samples
- Lowest chain at temp=1 is used to return samples to user; other chains encourage global moves
- Good for multi-model posteriors

Evolutionary Monte Carlo

- Combine ideas from genetic algorithms with MCMC
- Population is the new state space; propose moves that swap pieces of particles.

GMs and MCMC

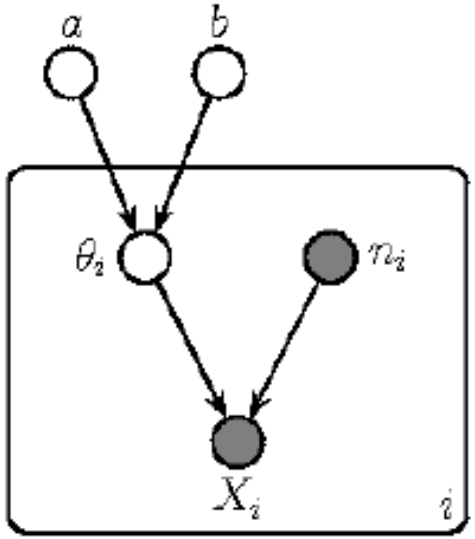
- MCMC can benefit from GMs
  - To define Markov blanket for Gibbs
  - To efficiently evaluate $\pi(x')/\pi(x)$ for MH
- GMs need MCMC for
  - State estimation (Inference)
  - Parameter estimation (Learnign)
  - Model selection (structure learning)

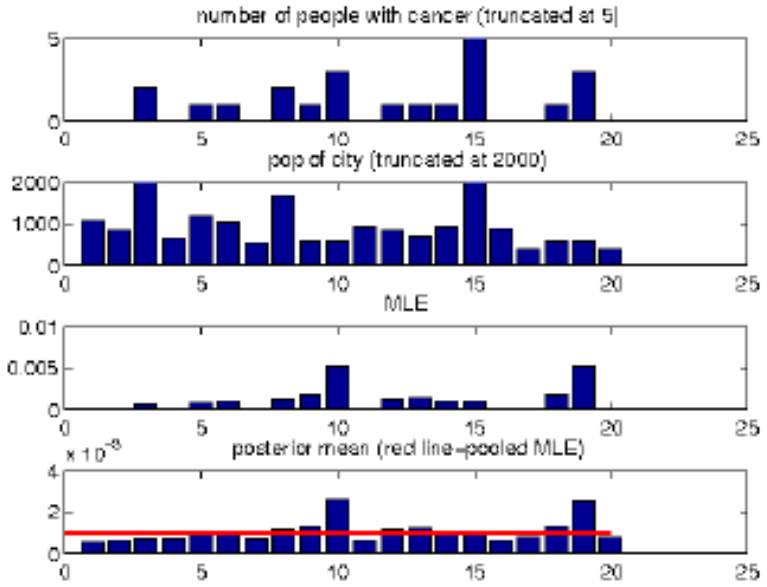
Collapsed samplers

- A collapsed sampler means analytically integrating out some variables and sampling the rest
- Aka Rao-Blackwellization
- Later we will see an interesting example when we consider RB for particle filtering
- Today, a simpler example, which will form the basis of a homework exercise

Hierarchical Bayesian modeling

- Model related cancer incidence rates



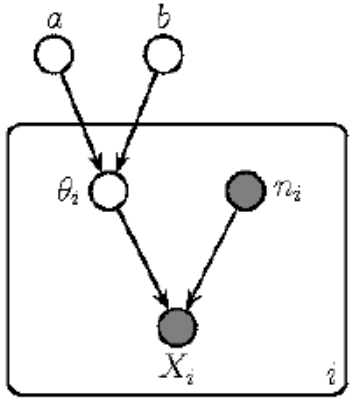


$$p(\mathbf{x}, \mathbf{n}, \boldsymbol{\theta}, a, b) = \prod_{i=1}^{n} p(x_i|n_i, \theta_i)p(\theta_i|a, b)p(a, b) \quad (1)$$

$$= \prod_{i=1}^{n} \text{Bin}(x_i|n_i, \theta_i)\text{Beta}(\theta_i|a, b)p(a, b) \quad (2)$$

Inference

- Gibbs sampling $p(a,b,\theta_i|D)$ - homework
- MH $p(a,b|D)$ – sample a,b, integrate out theta



$$p(\boldsymbol{\alpha}|\mathcal{D}) \propto p(\boldsymbol{\alpha}) \prod_i \int p(x_i|n_i,\theta_i) p(\theta_i|a,b) d\theta_i$$
$$= p(\boldsymbol{\alpha}) \prod_i \frac{B(a+x_i, b+n_i-x_i)}{B(a,b)}$$

$$E[\theta_i|\mathcal{D}] = E\left[E[\theta_i|\boldsymbol{\alpha},\mathcal{D}]|\mathcal{D}\right] \approx \frac{1}{S}\sum_{s=1}^{S} E[\theta_i|\boldsymbol{\alpha}^s]$$

- Empirical Bayes (a^*,b^*)=arg max $p(a,b|D)$, then $E[\text{theta}_i|a^*,b^*]$

MH for Missouri cancer problem

- We use mean m=a/(a+b) and K=a+b
- Beta prior on m, noninformative prior on K

$$p(m, K|\mathcal{D}) \propto \frac{m^{a_m-1}(1-m)^{b_m-1}}{(1+K)^2} \prod_i \frac{B(Km+x_i, K(1-m)+n_i-y_i)}{B(Km, K(1-m))}$$

- Transform to unconstrained params

$$\theta_1 = \log \frac{m}{1-m}, \quad \theta_2 = \log K$$

- MH with diagonal Gaussian proposal

Inference in discrete state spaces

- For a cts state space, $\pi(x)$ is a pdf, so we represent high probability values by repeating them many times
- For a discrete state space (eg model search, or after integrating out cts), the posterior is a pmf, so we can evaluate p(x|e) up to a normalization constant. There is no need to repeat a discrete state to represent its probability.
- Hence it is better to rapidly visit as many states as possible, and *never revisit a state*
- Hence use stochastic/ deterministic, local/ global search not MCMC

Deterministic search

- There are many (exact or approx) methods from the AI/ OR communities to find the top K values of a discrete distribution
- We approximate P(Z=z) by counting how many instantiations are compatible with Z=z, weighted by their probability

$$\sum_{m=1}^{M} \mathbf{1}\{x[m] = z\}\tilde{P}(\xi[m]),$$

- More precisely, we have bounds on p(Z=z)

$$\sum_{m=1}^{M} \mathbf{1}\{x[m] = z\}\tilde{P}(\xi[m]) \leq \tilde{P}(Z = z) \leq \left(1 - \sum_{m=1}^{M} \mathbf{1}\{x[m] \neq z\}\tilde{P}(\xi[m])\right).$$

Bounds on conditional probabilities

- We have

$$\begin{array}{ccccc} \ell_{\boldsymbol{y},\boldsymbol{e}} & \leq & P(y,e) & \leq & u_{\boldsymbol{y},\boldsymbol{e}} \\ \ell_{\boldsymbol{e}} & \leq & P(e) & \leq & u_{\boldsymbol{e}} \end{array}$$

$$\frac{\ell_{\boldsymbol{y},\boldsymbol{e}}}{u_{\boldsymbol{e}}} \leq P(y \mid e) \leq \frac{u_{\boldsymbol{y},\boldsymbol{e}}}{\ell_{\boldsymbol{e}}}.$$

Stat 521A
Lecture 17

Outline

- MAP estimation (13.1)
- Exact methods (13.2-13.3)
- Approx method based on clq graph (13.4)
- Linear programming relaxation (13.5)
- Graph cuts (13.6)
- Search (13.7)

Querying a distribution ("inference")

- Suppose we have a joint $p(X_1,\ldots,X_d)$. Partition the variables into E (evidence), Q (query), and H (hidden/ nuisance). We might pose the following queries
- Conditional probability (posterior):

$$p(\mathbf{X}_Q|\mathbf{x}_E) \propto \sum_{\mathbf{x}_H} p(\mathbf{X}_Q, \mathbf{x}_E, \mathbf{x}_H)$$

- MAP estimate (H=∅) (posterior mode)

$$\mathbf{x}_Q^* = \arg\max_{\mathbf{x}_Q} p(\mathbf{x}_Q|\mathbf{x}_E) = \arg\max_{\mathbf{x}_Q} p(\mathbf{x}_Q, \mathbf{x}_E)$$

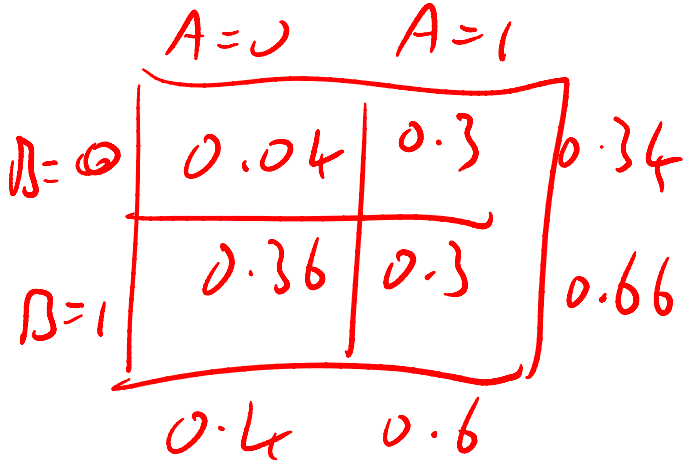
- Marginal MAP estimate (mode of marginal post):

$$\mathbf{x}_Q^* = \arg\max_{\mathbf{x}_Q} p(\mathbf{x}_Q|\mathbf{x}_E) = \arg\max_{\mathbf{x}_Q} \sum_{\mathbf{x}_H} p(\mathbf{x}_Q, \mathbf{x}_E, \mathbf{x}_H)$$

MAP vs marginal MAP

- Max max ≠ max sum
- Ex 2.1.12. Joint is

$$
\begin{aligned}
a^* &= \arg\max_a \sum_b p(a,b) = 1 \\
b^* &= \arg\max_b \sum_a p(a,b) = 1 \\
(a,b)^* &= \arg\max_{a,b} p(a,b) = (0,1)
\end{aligned}
$$

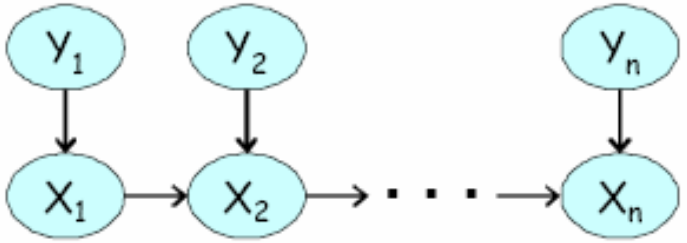


- Sequence of most probable states <> most probable sequence of states.

MMAP harder than MAP

- Thm 13.1.1. MAP for BNs is NP-hard.
- Thm 13.1.3. MMAP for BNs is complete for NP^{PP}.
- Thm 13.1.4. MMAP for tree structured GMs is NP-hard.
- Pf. Must sum out X before max out Y.

$$y^{p-map} = \arg \max_{Y_1,\ldots,Y_n} \sum_{X_1,\ldots,X_n} P(Y_1,\ldots,Y_n,X_1,\ldots,X_n).$$



VarElim for MAP

- Since max distributes over products, we can trivially modify the VE algorithm to compute the *scalar* max_x p(x).
- To find the assignment which achieves this MAP probability, we must do a traceback, analogous to the Viterbi traceback algorithm
- For the MMAP case, we can use the same algorithm, but with a constrained elim order (sum before max), which can make the problem harder

Clq Trees for MAP

- VE is inherently sequential: it is hard to imagine how to make a parallel/ distributed version of the traceback operator
- However, we can easily compute the max-marginals in parallel, replacing sum-product messages with max-product

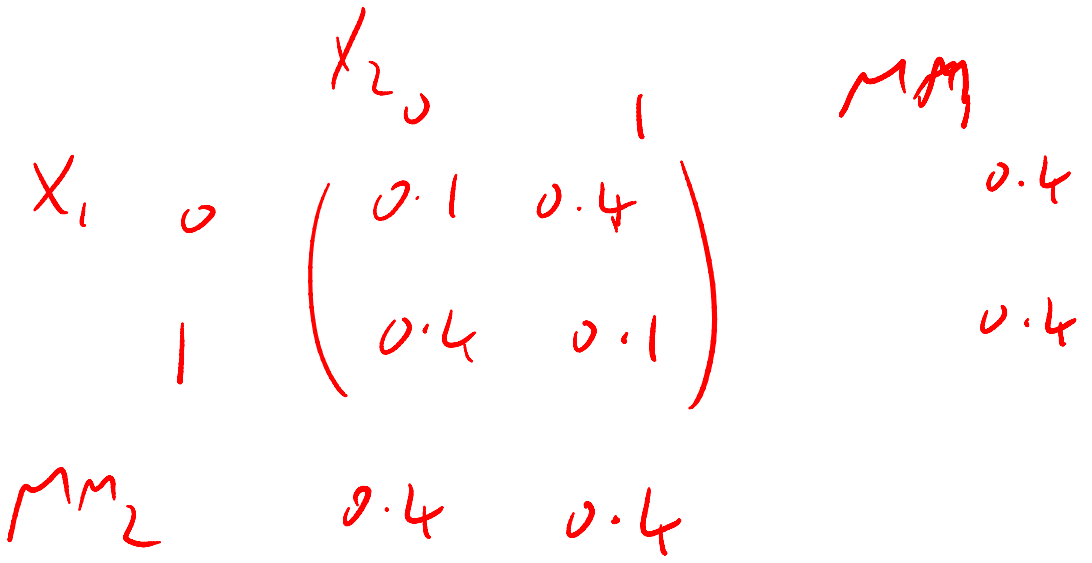
$$\text{MaxMarg}(x_i) = \max_{\mathbf{x}_{-i}} \tilde{p}(\mathbf{x}_{-i}, x_i)$$

- But how do we decode the corresponding assignment? Easy if each MM is unambiguous.

$$\exists \text{unique} x_i^* = \arg\max_{x_i} \text{MaxMarg}(x_i)$$

Problems of ambiguity

- Ex 13.3.7



- If we pick $x_1^*=1$ and $x_2^*=2$, we don't get $(x_1,x_2)^*$
- Must break ties consistently – requires global traceback.

Max-product in loopy cluster graphs

- We can change the sum-product algorithm to max product and run it on clique graphs that are not trees. The result is a set of pseudo max marginals which are max-calibrated

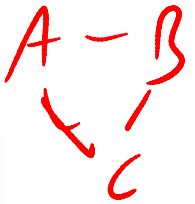
$$\max_{C_i - S_{i,j}} \beta_i = \max_{C_j - S_{i,j}} \beta_j = \mu_{i,j}(S_{i,j}).$$

Decoding pseudo max marginals

- Def 13.3.9. Let β_c be the max marginals in a clique tree/graph. An assignment x^* is locally optimal if

$$\mathbf{x}^*(c) \in \arg\max_{\mathbf{x}_c} \beta_c(\mathbf{x}_c)$$

- We can label each local assignment as equal to the local optimum (1) or not (0). We then need to solve a constraint satisfaction problem (CSP).
- Ex 13.4.2. Consider these "beliefs":



	a^1	a^0
b^1	1	2
b^0	2	1

	b^1	b^0
c^1	1	2
c^0	2	1

	a^1	a^0
c^1	1	2
c^0	2	1

Max-calibrated but not locally optimal; no solution exists

Quality of approximate solution

- Suppose the solution is locally optimal, so CSP can find a satisfying assignment. This is an exact MAP iff the clique graph is a tree with RIP.
- Suppose it is a general loopy graph. We can show (thm 13.4.6) that the solution is a “strong” local optimum, meaning that any change wrt to a large set of legal moves will decrease the probability.
- The legal moves including flipping states of any embedded subtree or single loops.

Max product TRW

- Suppose we replace “vanilla” max-product with a counting number version

$$\delta_{i\to j} = \max_{x_i} \left[\left(\psi_i[x_i] \prod_{k \in \mathrm{Nb}_i} \delta_{k\to i}(x_i) \right)^{\frac{\mu_{i,j}}{\mu_i}} \frac{1}{\delta_{j\to i}(x_i)} \psi_{i,j}[x_i, x_j] \right].$$

- Tree reweighting algorithm (TRW) uses the following convex counting numbers, given a distribution over trees T st each edge in the pairwise network is present in at least 1 tree

$$\begin{aligned} \mu_i &= -\textstyle\sum_{T \ni X_i} \rho(T) \\ \mu_{i,j} &= \textstyle\sum_{T \ni (X_i, X_j)} \rho(T) \end{aligned}$$

- Thm 13.4.8. If this algorithm finds a locally optimal solution, it is also globally optimal. (For sum-product, TRW is just convergent.)

Image completion

Priority-BP [Komodakis '06]

- In this case BP has an intolerable computational cost:
  - Just the basic operation of updating messages from node p to node q takes $O(|\mathcal{L}|^2)$ time
  - $|\mathcal{L}|^2$ SSD calculations between patches thus needed **(recall that |L| is huge in our case!)**
- Two extensions over standard-BP to reduce computation cost:
  - **"Dynamic label pruning"** and
  - **"Priority-based message scheduling"**

- **Labels L** = all wxh patches from source region S
- **MRF nodes** = all lattice points whose neighborhood intersects target region T
- **potential $V_p(x_p)$** = how well source patch x_p agrees with source region around p
- **potential $V_{pq}(x_p, x_q)$** = how well source patches x_p, x_q agree on their overlapping region

$$\mathcal{F}(\hat{x}) = \sum_{p \in \mathcal{V}} V_p(\hat{x}_p) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(\hat{x}_p, \hat{x}_q)$$

MAP as integer program

- Let $q(x_r^j)=1$ if clique r is in state j.
- Let $\eta_r^j = \log \phi_r(j)$.
- MAP problem:

$$\text{maximize}_{\boldsymbol{q}} \sum_{r \in \mathbf{R}} \sum_{j=1}^{n_r} \eta_r^j q(x_r^j),$$

- Integer constraint:

$$q(x_r^j) \in \{0,1\} \qquad \text{For all } r \in \mathbf{R};\ j \in \{1,\ldots,n_r\}$$

- Mutual exclusion constraint:

$$\sum_{j=1}^{n_r} q(x_r^j) = 1 \qquad \text{For all } r \in \mathbf{R}.$$

- Consistency constraint:

$$\sum_{j\ :\ \boldsymbol{c}_r^j \sim \boldsymbol{s}_{r,r'}} q(x_r^j) = \sum_{l\ :\ \boldsymbol{c}_{r'}^l \sim \boldsymbol{s}_{r,r'}} q(x_{r'}^l).$$

LP relaxation

- Let $q(x_r^j) \geq 0$ instead of {0,1}.

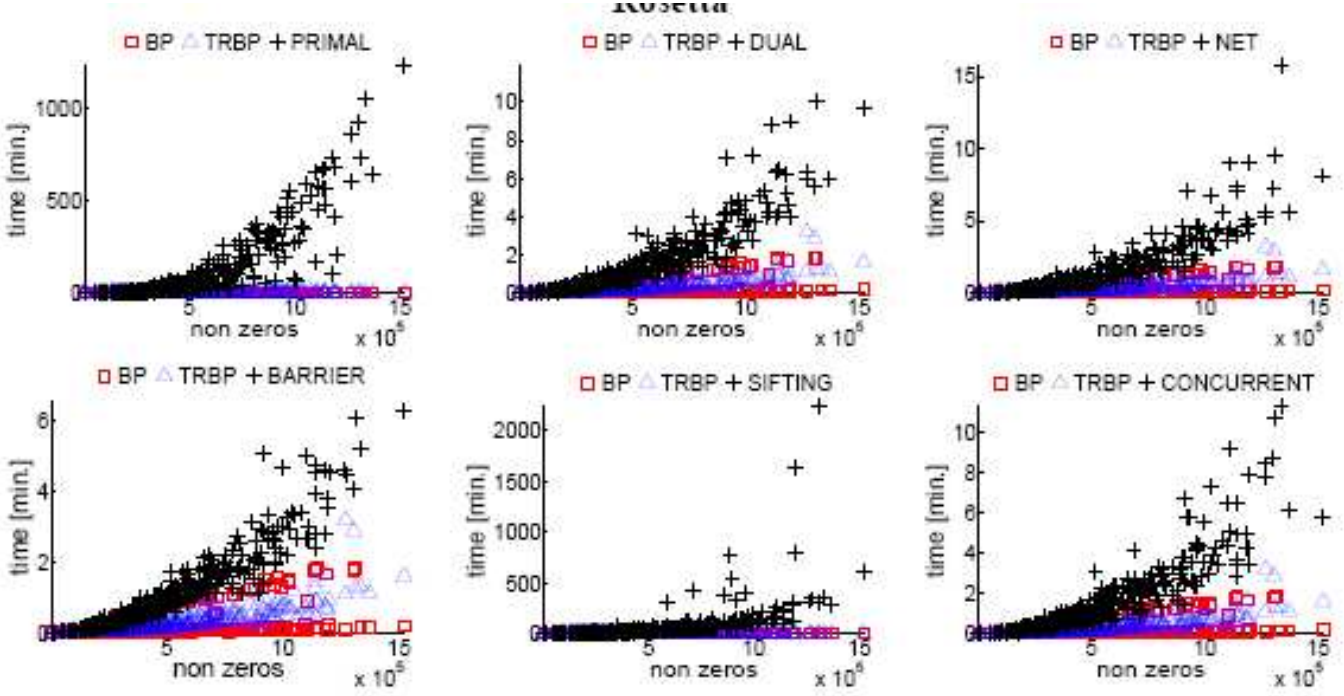
$$\begin{aligned}
&\textbf{Find} && \{q(x_{\mathbf{R}}^j) \ : \ r \in \mathbf{R}; \mathbf{j} = 1, \ldots, \mathbf{n_r}\} \\
&\textbf{that maximize} && \eta^\top q \\
&\textbf{subject to} && \sum_{j=1}^{n_r} q(x_r^j) = 1 \qquad r \in \mathbf{R} && (13.25) \\
& && \sum_{j \,:\, \mathbf{c}_r^j \sim \boldsymbol{s}_{r,r'}} q(x_r^j) = \sum_{l \,:\, \mathbf{c}_{r'}^l \sim \boldsymbol{s}_{r,r'}} q(x_{r'}^l) \qquad \begin{matrix} r, r' \in \mathbf{R} \\ s_{r,r'} \in Val(C_r \cap C_{r'}) \end{matrix} && (13.26) \\
& && q \geq \mathbf{0} && (13.27)
\end{aligned}$$

Convex BP is solving the dual of this LP.
If the solution is integer, and there are are no ties,
then fixed points of this are exact MAP estimates.

MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies
Yair Weiss, Chen Yanover, Talya Meltzer, UAI 2007

BP beats CPLEX

- Convex max-product is 100-1000 times faster than CPLEX at finding the exact solution to certain MAP problems in computer vision and protein folding.



Linear Programming Relaxations and Belief Propagation - an Empirical Study
Chen Yanover, Talya Meltzer, Yair Weiss, JMLR 2006

Submodularity

- Let L = {0,1,…,K} be an ordered set.
- Let g: LxL -> R be a function.
- We say g is submodular iff

$$\forall x, y \in \mathcal{L} \;\; g(x \vee y) + g(x \wedge y) \leq g(x) + g(y)$$

$$(x \vee y)_i = \min(x_i, y_i), \quad (x \wedge y)_i = \max(x_i, y_i)$$

- Submodularity ~ convexity for discrete opt.
- Eg L = {0,1}, g is submodular iff

$$g(0,0) + g(1,1) \leq g(0,1) + g(1,0)$$

$$[(0,1) \vee (1,0)] = [\min(0,1), \min(1,0)] = [0,0],$$

$$[(0,1) \wedge (1,0)] = [\max(0,1), \max(1,0)] = [1,1]$$

Submodular potentials

- Defn 13.6.2. A pairwise energy term on binary nodes is submodular if

$$\epsilon(1,1) + \epsilon(0,0) \leq \epsilon(1,0) + \epsilon(0,1)$$

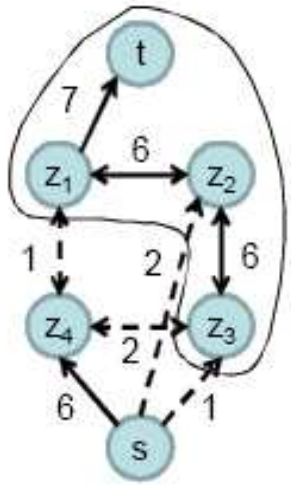
- Example: Ising model with attractive potential

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & \lambda \\ 1 & \lambda & 0 \end{array} \quad \lambda \geq 0$$

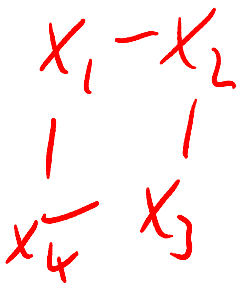
- For any binary MRF with submodular potentials, we can find the exact MAP in polynomial time

Graph cuts for Ising model

- Create a source and sink node, s, t.
- Add edge X_i->t with weight $\varepsilon_i[0]$.
- Add edge X_i->s with weight $\varepsilon_i[1]$
- Add X_i – X_j with λ_{ij}.
- Find minimal cut. All nodes on t-side of cut are in state 1.



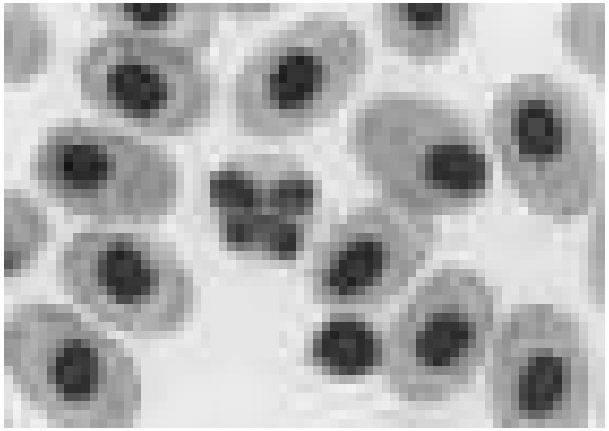
$$\epsilon_1[0] = 7 \quad \epsilon_2[1] = 2 \quad \epsilon_3[1] = 1 \quad \epsilon_4[1] = 6$$
$$\lambda_{1,2} = 6 \quad \lambda_{2,3} = 6 \quad \lambda_{3,4} = 2 \quad \lambda_{1,4} = 1.$$

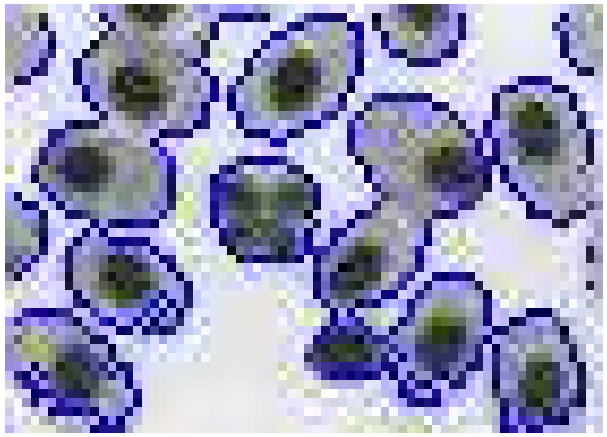


$X^* = 1, 1, 1, 0$

$X_{maxmarg} = 1, 0, 0, 0$ but strong constraints

Segmentation using binary MRF





$$P(I(i); p_i = 0) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp(-\frac{(I(i)-\mu_b)^2}{2\sigma_b^2}),$$

$$P(I(i); p_i = 1) = \frac{0.5}{\sqrt{2\pi}\sigma_{f,1}} \exp(-\frac{(I(i)-\mu_{f,1})^2}{2\sigma_{f,1}^2}) + \frac{0.5}{\sqrt{2\pi}\sigma_{f,2}} \exp(-\frac{(I(i)-\mu_{f,2})^2}{2\sigma_{f,2}^2}),$$

Slide by Nilanjan Ray, from Google search

Metric MRFs

- A metric MRF is one with K states and pairwise potentials of the form

$$\epsilon_{i,j}(v_k, v_l) = \mu(v_k, v_l) \geq 0$$

where \mu is a metric:

$\mu : \mathcal{V} \times \mathcal{V} \mapsto [0, \infty)$ *is a* metric *if it satisfies:*

- **Reflexivity:** $\mu(v_k, v_l) = 0$ *if and only if* $k = l$;
- **Symmetry:** $\mu(v_k, v_l) = \mu(v_l, v_k)$;
- **Triangle Inequality:** $\mu(v_k, v_l) + \mu(v_l, v_m) \geq \mu(v_k, v_m)$.

- Hence for any v we have submodularity:

$$\epsilon_{i,j}[x_i, x_j] + \epsilon_{i,j}[v, v] \not\geq \epsilon_{i,j}[x_i, v] + \epsilon_{i,j}[v, x_j].$$

Δ ineq

Functions of label differences

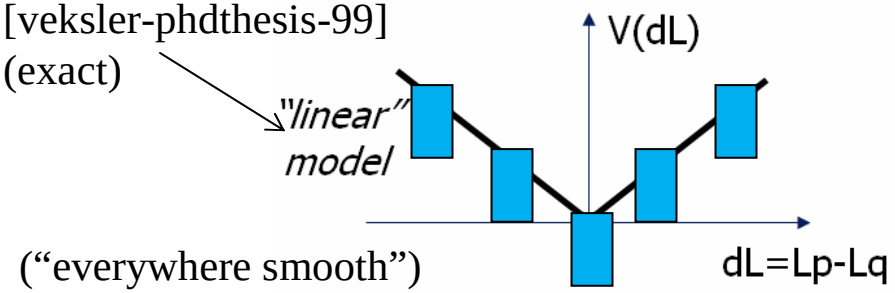
- V(*p*,*q*) is 2nd order potential of the difference in the labels of pixels *p* and *q*
  - These functions penalize big difference in label values between neighboring data
    - Image restoration: want to maintain similar intensities with neighbors

Convex interactions
(minimize is P)

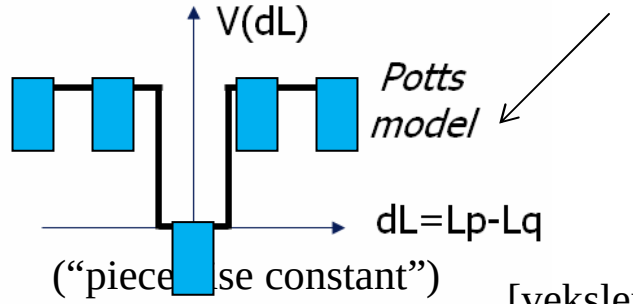
Robust or "discontinuity-preserving" interactions
(minimize is NP-complete)

[boykov-pami-01]
(approximate)

[veksler-phdthesis-99]
(exact)

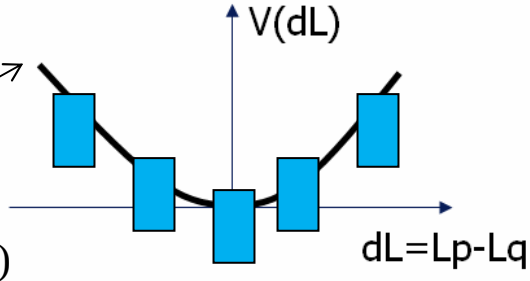


("everywhere smooth")



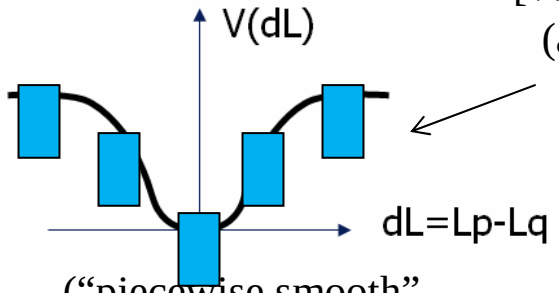
("piecewise constant")

[ishikawa-pami-03]
(exact)



("convex")

[veksler-cvpr-07]
(approximate)



("piecewise smooth",
"truncated convex")

$$\epsilon[x_i, x_j] = \min(c\|x_i - x_j\|_p, \mathrm{dist}_{\max}),$$

Source: Daniel Munoz

GC for non-binary submodular

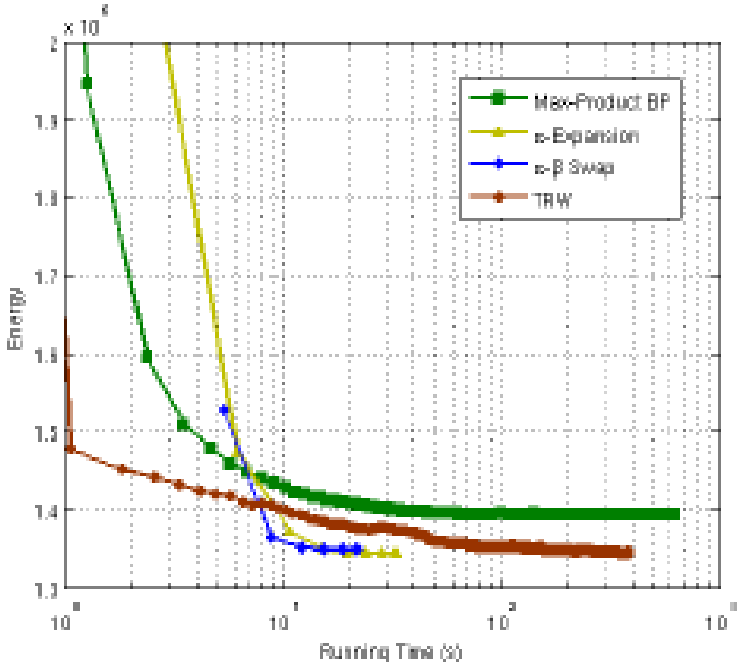
- For non-binary models, MAP estimation is NP-hard.
- But if the potential is submodular for any pair of states (eg metric MRF) then we can use a greedy algorithm in which we make large moves
- Alpha expansion: consider setting each node to its current state or to state α (2-optimal).
- Alpha-beta swap: consider swapping any two states; energy function only need be semi-metric (triangle inequality not required).

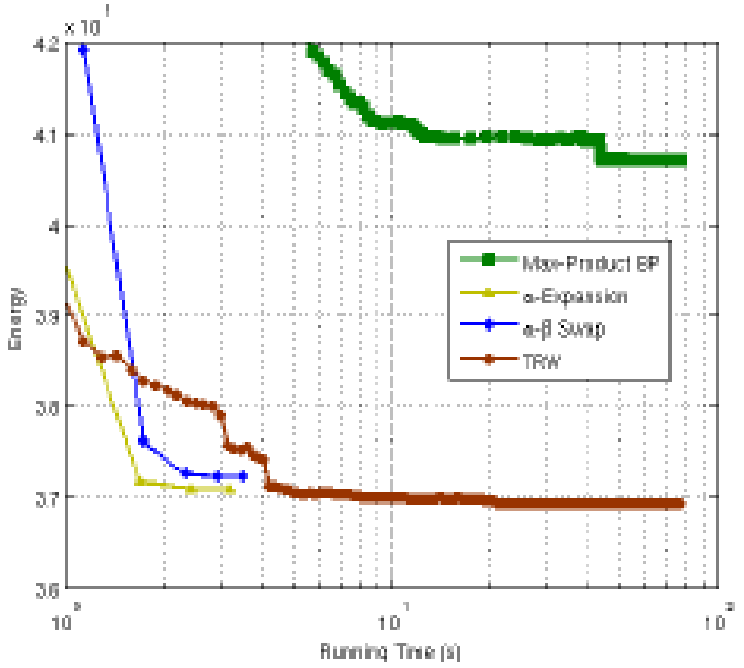
Expansion

Swap

(Boykov'01)

Stereo reconstruction





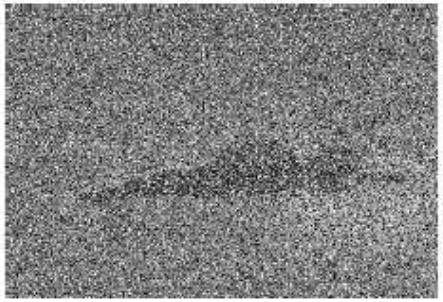
Total variation norm

$$TV(u) = \int |\nabla u| du$$

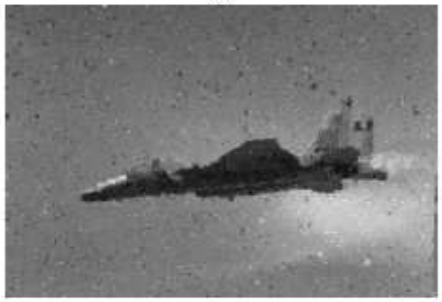
$$g_{pq}(u_p, u_q) = \beta |u_p - u_q|$$

Graph cuts on the level sets

(a)



(b)



(c)

Global Optimization for First Order Markov Random Fields with Submodular Priors
Jerome Darbon,
Discrete Applied Mathematics, 2009

Additional info

Good tutorial at ECCV'08: "**MAP Estimation in Computer Vision**"
Kumar, Kolhi, Zisserman, Torr
http://www.robots.ox.ac.uk/~pawan/eccv08_tutorial/index.html

"A Linear Programming Approach to Max-sum Problem: A Review",
Tomas Werner, PAMI 2007

Search (A.4)

- Systematic tree search – partial assignments
  - Branch and bound: prune off trajectory if lower bound of extension higher than current best
  - Particle filtering: stochastically grow partial solutions
- Local search – complete assignments
  - Hill climbing, Tabu search, Beam search, simulated annealing
  - See Holger Hoos's class in CS
- Search methods for Marginal MAP
  - Search over max, compute sum using VE (cf Rao-Blackwellize). Use unconstrained elim order to get upper bound.

Greedy hill climbing

Algorithm A.5 Greedy local search algorithm with search operators.

```
    Procedure Greedy-Local-Search (
      σ0,    // initial candidate solution
      score,    // Score function
      O,    // Set of search operators
    )
1     σbest ← σ0
2     do
3       σ ← σbest
4       Progress ← false
5       for each operator o ∈ O
6         σo ← o(σ)    // Result of applying o on σ
7         if σo is legal solution then
8           if score(σo) > score(σbest) then
9             σbest ← σo
10            Progress ← true
11    while Progress
12
13    return σbest
```

Instead of looking amongst all neighbors O, we can pick the first improving one (first-ascent or best first search).
Converges to local maximum or plateau.

Tabu search

- Once we get to a plateau, allow selection of ‘neutral’ move to a state that hasn’t been visited before .
- Requires lots of memory. Instead, prevent picking a move that would undo a recently applied operator.

```
1   σ_best ← σ_0
2   σ ← σ_best
3   t ← 1
4   LastImprovement ← 0
5   while LastImprovement < N
6     o^(t) ← ε    // Set current operator to be uninitialized
7     for each operator o ∈ O    // Search for best allowed operator
8       if LegalOp(o, {o^(t−L), ..., o^(t−1)}) then
9         σ_o ← o(σ)
10        if σ_o is legal solution then
11          if o^(t) = ε or score(σ_o) > score(σ_o^t) then
12            o^(t) ← o
13    σ ← σ_o^t
14    if score(σ) > score(σ_best) then
15      σ_best ← σ_o
16      LastImprovement ← 0
17    else
18      LastImprovement ← LastImprovement + 1
19    t ← t + 1
20
21  return σ_best
```

Stat 521A
Lecture 18

Outline

- Cts and discrete variables (14.1)
- Gaussian networks (14.2)
- Conditional Gaussian networks (14.3)
- Non-linear Gaussian networks (14.4)
- Sampling (14.5)

Hybrid networks

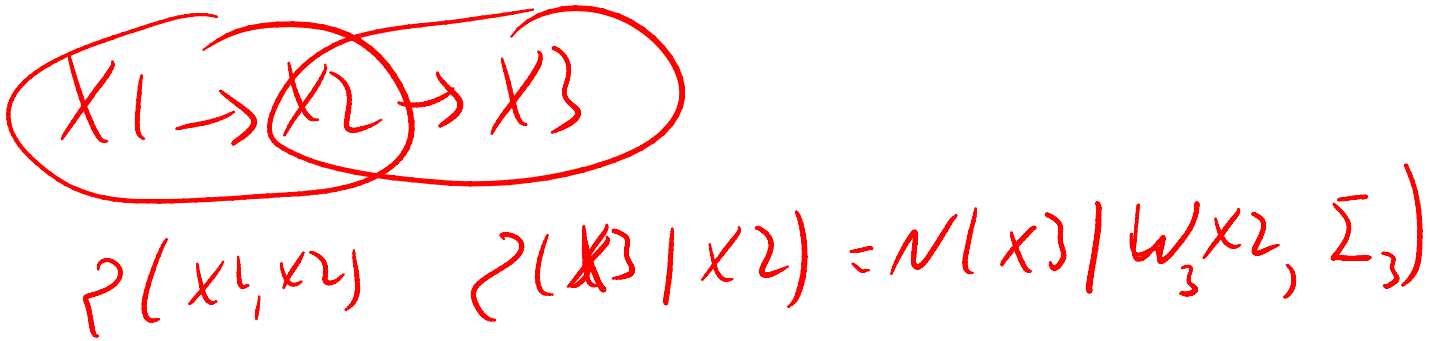
- A “hybrid” GM contains discrete and cts variables
- Except in the case that everything is all discrete or all Gaussian, exact inference is rarely possible
- The reason is that the basic operations of multiplication, marginalization and conditioning are not closed except for tables and MVNs

Gaussian networks

- We can always convert a Gaussian DGM or UGM to an MVN and do exact inference in $O(d^2)$ space and $O(d^3)$ time
- However, d can be large (eg 1000x1000 image)
- We seek methods that exploit the graph structure, that will take $O(d\ w^2)$ space and $O(d\ w^3)$ time, where w is the tree width
- In cases where w is too large, we can use loopy belief propagation, which takes O(1) space and O(d) time

Canonical potentials

- When performing VarElim or ClqTree propagation, we have to represent factors \phi(x). These may not be Gaussians, but can always be represented as exponentials of quadratics



$$\mathcal{C}(\boldsymbol{X}; K, \boldsymbol{h}, g) = \exp\left(-\frac{1}{2}\boldsymbol{X}^T K \boldsymbol{X} + \boldsymbol{h}^T \boldsymbol{X} + g\right).$$

Thus, $\mathcal{N}(\boldsymbol{\mu}; \Sigma) = \mathcal{C}(K, \boldsymbol{h}, g)$ where:

$$\begin{aligned} K &= \Sigma^{-1} \\ \boldsymbol{h} &= \Sigma^{-1}\boldsymbol{\mu} \\ g &= -\frac{1}{2}\boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{\mu} - \log\left((2\pi)^{n/2}|\Sigma|^{1/2}\right). \end{aligned}$$

Operations on canonical potentials

- Multiplication

$$\mathcal{C}(K_1, h_1, g_1) \cdot \mathcal{C}(K_2, h_2, g_2) = \mathcal{C}(K_1 + K_2, h_1 + h_2, g_1 + g_2)$$

$$\phi_1(X,Y) = \mathcal{C}\left(X, Y; \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, -3\right) \quad * \quad \phi_2(Y,Z) = \mathcal{C}\left(Y, Z; \begin{bmatrix} 3 & -2 \\ -2 & 4 \end{bmatrix}, \begin{pmatrix} 5 \\ -1 \end{pmatrix}, 1\right).$$

$$= \mathcal{C}\left(X, Y, Z; \begin{bmatrix} 1 & -1 & 0 \\ -1 & 4 & -2 \\ 0 & -2 & 4 \end{bmatrix}, \begin{pmatrix} 1 \\ 4 \\ -1 \end{pmatrix}, -2\right)$$

- Division

$$\frac{\mathcal{C}(K_1, h_1, g_1)}{\mathcal{C}(K_2, h_2, g_2)} = \mathcal{C}(K_1 - K_2, h_1 - h_2, g_1 - g_2)$$

Operations on canonical potentials

Marginalization (requires KYY be pd)

$$\int \mathcal{C}(X, Y; K, h, g)\, dY.$$

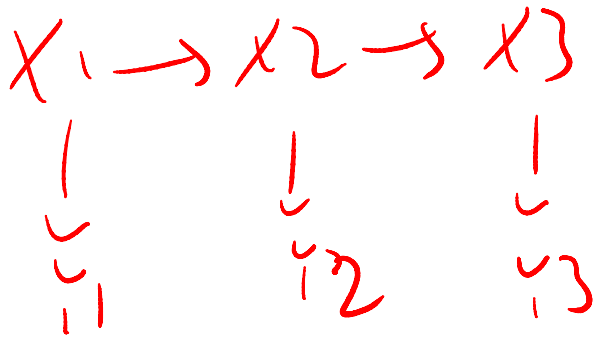
$$\begin{aligned} K' &= K_{\boldsymbol{XX}} - K_{\boldsymbol{XY}} K_{\boldsymbol{YY}}^{-1} K_{\boldsymbol{YX}} \\ h' &= h_{\boldsymbol{X}} - K_{\boldsymbol{XY}} K_{\boldsymbol{YY}}^{-1} h_{\boldsymbol{Y}} \\ g' &= g + \tfrac{1}{2}\left(|Y|\log(2\pi) - \log|K_{\boldsymbol{YY}}| + h_{\boldsymbol{Y}}^T K_{\boldsymbol{YY}} h_{\boldsymbol{Y}}\right). \end{aligned}$$

Conditioning (Y=y)

$$\begin{aligned} K' &= K_{\boldsymbol{XX}} \\ h' &= h_{\boldsymbol{X}} - K_{\boldsymbol{XY}} y \\ g' &= g + h_{\boldsymbol{Y}}^T y - \frac{1}{2} y^T K_{\boldsymbol{YY}} y. \end{aligned}$$

Kalman filter- smoother

- If you apply the FB algorithm with these new operators, you get the same results as the RTS smoother



Gaussian LBP

- If the treewidth is too large, we can pass messages on the original (pairwise) graph
- We just apply the regular BP rules with the new operators. Once can show this is equivalent to the following:

$$p(X_1, \ldots, X_n) \propto \left(-\frac{1}{2} \boldsymbol{X}^T J \boldsymbol{X} + h^T \boldsymbol{X} \right).$$

$$\delta_{i \to j}(x_j) = \exp\left(-\frac{1}{2} J_{i \to j} x_j^2 + h_{i \to j} x_j \right).$$

$$\begin{aligned} \hat{J}_{i \setminus j} &= J_{ii} + \textstyle\sum_{k \in \mathrm{Nb}_i - \{j\}} J_{k \to i} \\ \hat{h}_{i \setminus j} &= h_i + \textstyle\sum_{k \in \mathrm{Nb}_i - \{j\}} h_{k \to i}. \end{aligned}$$

$$\begin{aligned} J_{i \to j} &= -J_{ji} \hat{J}_{i \setminus j}^{-1} J_{ji} \\ h_{i \to j} &= -J_{ji} \hat{J}_{i \setminus j}^{-1} \hat{h}_{i \setminus j}. \end{aligned}$$

$$\begin{aligned} \hat{J}_i &= J_{ii} + \sum_{k \in \mathrm{Nb}_i} J_{k \to i} \\ \hat{h}_i &= h_i + \sum_{k \in \mathrm{Nb}_i} h_{k \to i}. \end{aligned}$$

$$\begin{aligned} \hat{\mu}_i &= (\hat{J}_i)^{-1} \hat{h}_i \\ \hat{\sigma}_i^2 &= (\hat{J}_i)^{-1} \end{aligned}$$

Gaussian LBP

- Thm 14.2.4. If LBP converges, then the means are exact, but the variances are too small (overconfident)
- Thm. A sufficient condition for convergence is that the potentials are pairwise normalizable
- Any attractive model (all +ve correlations) is pairwise normalizable
- The method for computing the means is similar to solving a set of linear equations

Pairwise normalizable

- Def 7.3.3. A pairwise MRF with energies of the form

$$\begin{aligned} \epsilon_i(x_i) &= d_0^i + d_1^i x_1 + d_2^i x_i^2 \\ \epsilon_{ij}(x_i, x_j) &= a_{00}^{i,j} + a_{01}^{i,j} x_i + a_{10}^{ij} x_j + a_{11}^{ij} x_i x_j + a_{02}^{ij} x_i^2 + a_{20}^{ij} x_j^2 \end{aligned}$$

  is called pairwise normalizable if

$d_2^i > 0, \forall i$ and $\begin{pmatrix} a_{02}^{ij} & a_{11}^{ij}/2 \\ a_{11}^{ij}/2 & a_{20}^{ij} \end{pmatrix}$ is psd for all i,j

- Thm 7.3.4. If the MRF is pairwise normalizable, then it defines a valid Gaussian.
- Sufficient but not necessary eg.

$$\begin{pmatrix} 1 & 0.6 & 0.6 \\ 0.6 & 1 & 0.6 \\ 0.6 & 0.6 & 1 \end{pmatrix}$$

May be able to reparameterize the node/ edge potentials to ensure pairwise normalized.

Conditional linear Gaussian networks

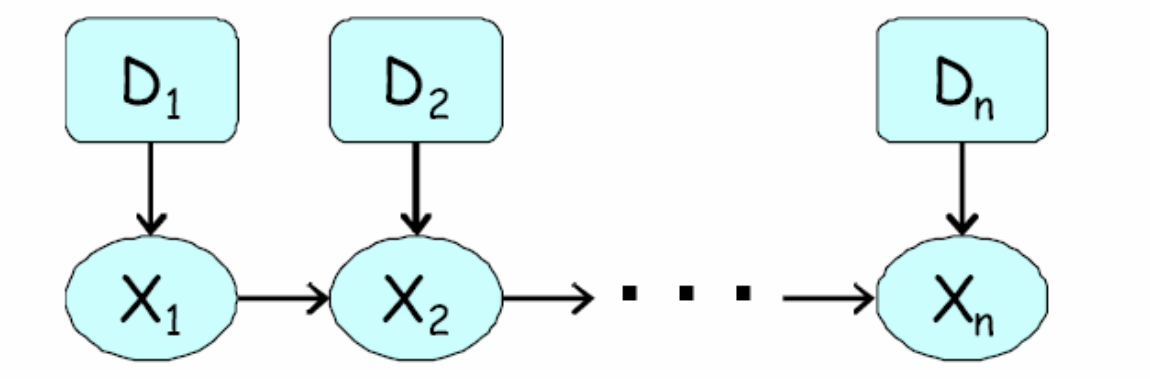
- Suppose all discrete nodes only have discrete parents, and all cts nodes either have discrete parents, cts parents, or no parents.
- Further, assume all cts CPDs have the form

$$p(X = x|C = \mathbf{c}, D = k) = \mathcal{N}(x|\mathbf{w}_k^T \mathbf{c}, \sigma_k^2)$$

- This is called a CLG network. It is equivalent to a mixture of MVNs, where the distribution over discrete indicators has structure, as does each covariance matrix.
- We create a canonical factor for each discrete setting of the variables in a clique.

Inference in CLG networks

- Thm 14.3.1. Inference in CLG networks is NP-hard, even if they are polytrees.
- Pf (sketch). Consider the network below. When we sum out D_1, $p(X_1)$ is a mixture of 2 Gaussians. In general, $p(X_i)$ is a mixture of 2^i Gaussians.



$$p(X_2) = \sum_{D_2} P(D_2) \int_{X_1} p(X_2 \mid X_1, D_2) \sum_{D_1} p(X_1 \mid D_1).$$

Weak marginalization

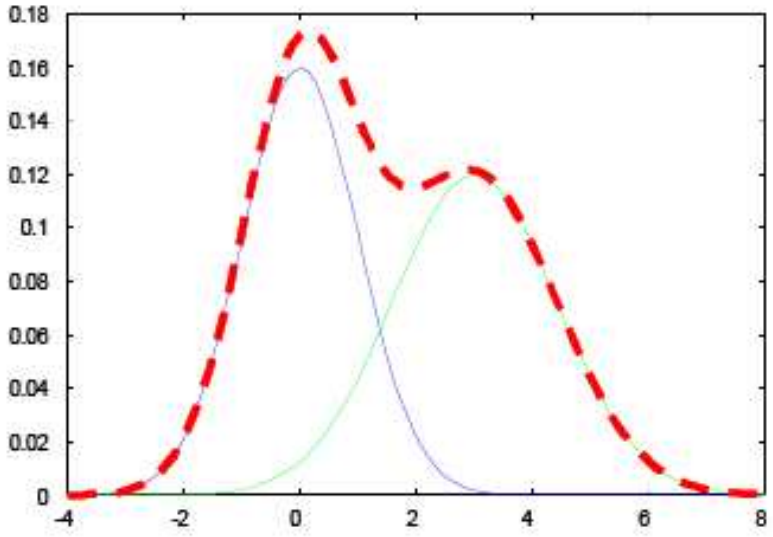
- To prevent the blowup in the number of mixture components, we can project back to the class of single mixtures at each step, as in EP
- Prop 14.3.6. argmin_q KL(p|q) where q is a Gaussian has parameters (

$$\begin{aligned} \mu_i &= E_p[X_i] \\ \Sigma_{i,j} &= Cov_p[X_i; X_j] \end{aligned}$$

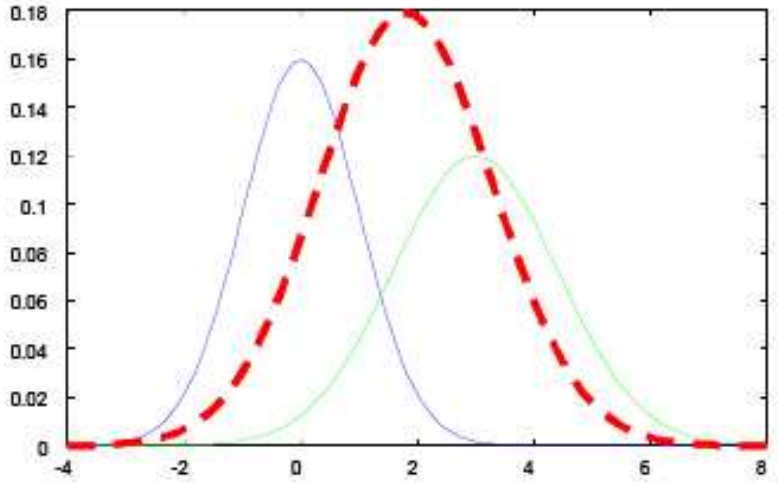
- Prop 14.3.7. argmin_q KL(p,q) where p is a mixture of Gaussians is a single Gaussian with params

$$\begin{aligned} \mu &= \sum_{i=1}^{k} w_i \mu_i \\ \Sigma &= \sum_{i=1}^{k} w_i \Sigma_i + \sum_{i=1}^{k} w_i (\mu_i - \mu)(\mu_i - \mu)^T. \end{aligned}$$

Weak marginalization



(a)



(b)

Canonical vs moment form

- Weak marginalization is defined in terms of moment form
- To convert a canonical factor to moment form, we require that it represent a valid joint density
- This typically requires we pass messages from parents to children.
- Once we have initialized all factors, they can be converted to moment form.
- However, division in the backwards pass may cause some variances to become negative! (see Ex 14.3.13)
- EP is hairy!

Strong marginalization

- By using a constrained elimination order, in which we integrate out before summing out, we can ensure that the upwards pass never needs to perform weak marginalization.
- Furthermore, one can show that the downwards pass results in exact results for the discrete variables and exact 1^{st} and 2^{nd} moments for the cts variables (Lauritzen's "strong jtree" algorithm)
- However, the constrained elim order usually results in large discrete cliques, making this often impractical.

Non linear dependencies

- In a linear Gaussian network, the mean is a linear function of its parents.
- Now assume $X_i = f(U_i, Z_i)$, where $Z_i \sim N(0,I)$

auxiliary variables into the variables of interest. For a vector of functions $\vec{f} = (f_1, \ldots, f_d)$ and a Gaussian distribution p_0, we use the notation $p(X_1, \ldots, X_d) = (p_0 \bigoplus \vec{f})$ to refer to the distribution that has $p(f_1(Z), \ldots, f_d(Z)) = p_0(Z)$ and 0 elsewhere.

- Examples

Example 14.4.1: . *For example, consider a multi-variate Gaussian* $p(X_1, \ldots, X_d) = \mathcal{N}(X; \mu, \Sigma)$. *We define a matrix* A *to be a* $d \times d$ *matrix such that* $AA^T = \Sigma$; A *is often called the* square root *of* Σ, *and is guaranteed to exist whenever* Σ *is positive definite. In this case we can show (see Exercise 14.6) that we can redefine* p *as:*

$$p(X) = p_0(W) \bigoplus (AW + \mu), \tag{14.14}$$

where $p_0(W) = \mathcal{N}(W; 0, I)$, *for* I *the identity matrix.* ■

Example 14.4.2: *As another example, consider the non-linear CPD* $X \sim \mathcal{N}\left(\sqrt{Y_1^2 + Y_2^2}; \sigma^2\right)$. *We can reformulate this CPD in terms of a deterministic, non-linear function, as follows: We introduce a new exogenous variable* W *that captures the stochasticity in the CPD. We then define* $X = f(Y_1, Y_2, W)$ *where* $f(Y_1, Y_2, W) = \sqrt{Y_1^2 + Y_2^2} + \sigma W$. ■

Taylor series approx

- We can linearize f and then fit a Gaussian (basis of the EKF algorithm)

As we know, if $p_0(Z)$ is a Gaussian distribution and $X = f(Z)$ is a linear function, then $p(X) = p(f(Z))$ is also a Gaussian distribution. Thus, one very simple and commonly used approach is to approximate f as a linear function $\hat{f}$, and then define $\hat{p}$ in terms of $\hat{f}$.

The most standard linear approximation for $f(Z)$ is the Taylor series expansion around the mean of $p_0(Z)$:

$$\hat{f}(Z) = f(\mu) + \nabla f|_{\mu} Z. \qquad (14.15)$$

Can be bad if f not linear near mu

Although the Taylor series expansion provides us with the optimal linear approximation to f, the Gaussian $\hat{p}(X) = p_0(Z) \oplus \hat{f}(Z)$ may not be the optimal Gaussian approximation to $p(X) = p_0(Z) \oplus f(Z)$.

Example 14.4.4: *Consider the function $X = Z^2$, and assume that $p(Z) = \mathcal{N}(Z; 0, 1)$. The mean of X is simply $\mathbf{E}_p[X] = \mathbf{E}_p[Z^2] = 1$. The variance of X is*

$$\mathbf{Var}_p[X] = \mathbf{E}_p[Z^2] - \mathbf{E}_p[Z]^2 = \mathbf{E}_p[Z^4] - \mathbf{E}_p[Z^2]^2 = 3 - 1^2 = 2.$$

On the other hand, the first order Taylor series approximation of f at the mean value $Z = 0$ is:

$$\hat{f}(Z) = 0^2 + (2Z)_{U=0} Z \equiv 0.$$

Thus, $\hat{p}(X)$ will simply be a delta function where all the mass is located at $X = 0$, a very poor approximation to p. ■

M projection using quadrature

- Best Gaussian approx has these moments

$$\begin{aligned} \mathbb{E}_p[X_i] &= \int_{-\infty}^{\infty} f_i(z) p_0(z) dz \\ \mathbb{E}_p[X_i X_j] &= \int_{-\infty}^{\infty} f_i(z) f_j(z) p_0(z) dz. \end{aligned}$$

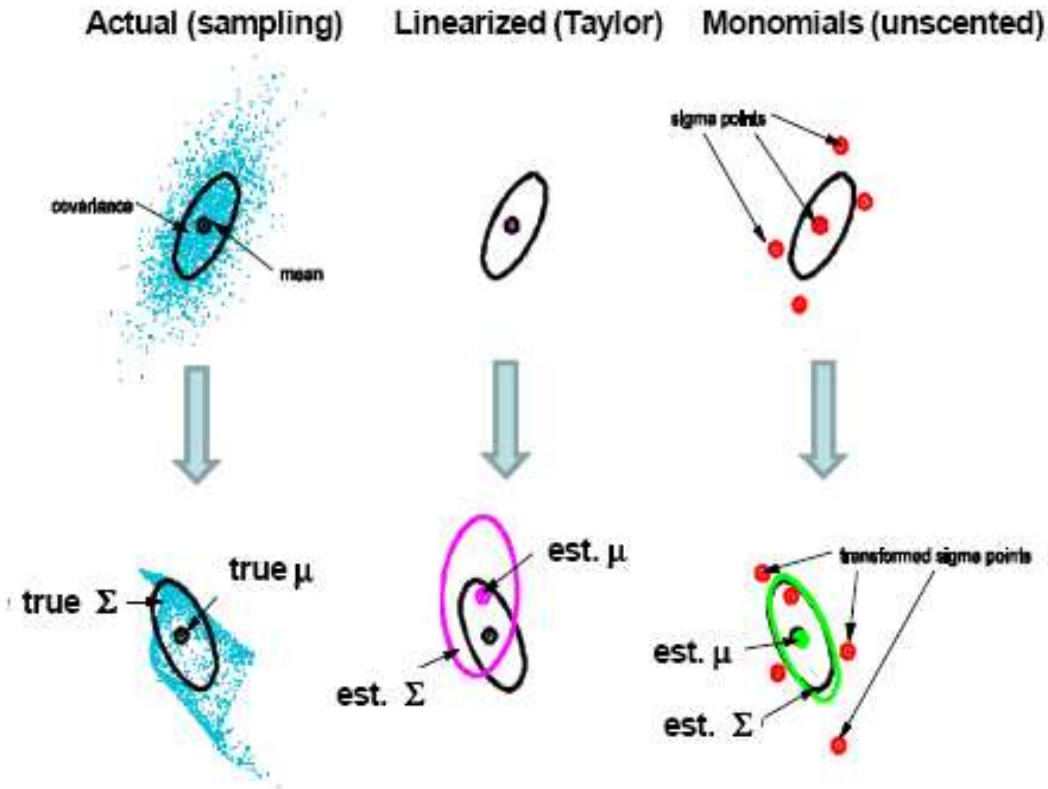
- Gaussian quadrature computes this integral for any W(z)>0 (here, Gaussian)

$$\int_a^b W(z) f(z) dz \approx \sum_{j=1}^{m} w_j f(z_j).$$

Unscented transform

- Pass mean and +- std in each dim through transform, and then fit Gaussian to transformed points

$$\int_{-\infty}^{\infty} W(z)f(z)dz \approx \left(1-\frac{d}{\lambda^2}\right) f(\mathbf{0}) + \sum_{i=1}^{d} \frac{1}{2\lambda^2} f(\lambda z_i^+) + \sum_{i=1}^{d} \frac{1}{2\lambda^2} f(\lambda z_i^-).$$



Nonlinear GMs

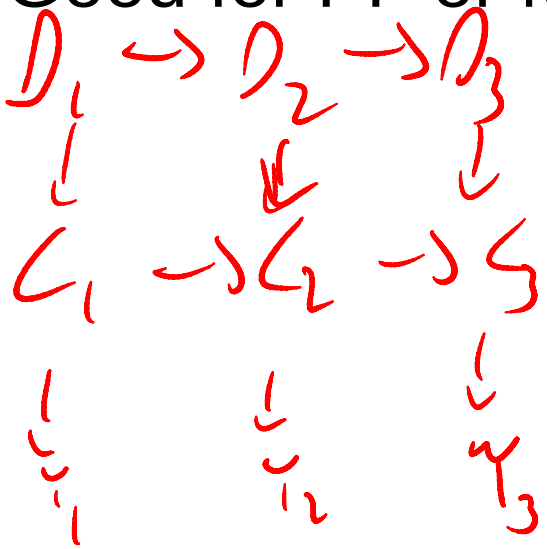
- We approximate nonlinear factors by approximating them by Gaussians
- The above methods require a joint Gaussian factor, not a canonical factor – we have to pass messages in topological order, and introduce variables one at a time to use the above tricks
- Linearization is done relative to current \mu. In EP, we iterate, and re-approximate each factor in the context of its incoming messages, which provides a better approx. to the posterior.
- Pretty hairy.

Discrete children, cts parents

- C -> D arcs are useful eg thermostat turns on/off depending on temperature
- We can approximate Gaussian * logistic by a Gaussian (variational approx)
- We can combine these Gaussian factors with the other factors as usual.

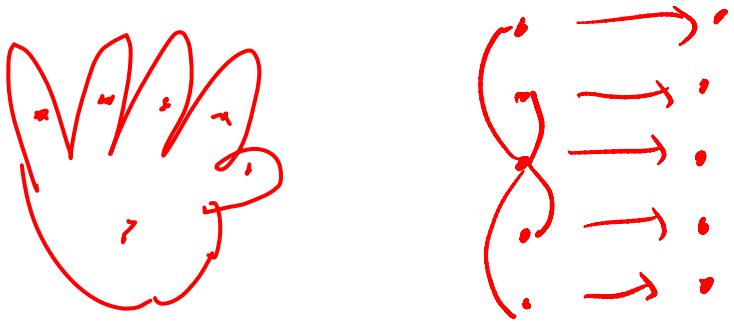
Sampling

- Sampling is the easiest way to handle cts and mixed variables
- "Collapsed particles" (Rao-Blackwellisation): sample the discretes, integrate out cts analytically. Each particle has a value for D and a Gaussian over C. Good for PF or MCMC.



Non-parametric BP

- We can combine sampling and msg passing.
- We approximate factors/ msgs by samples.
- Factors are lower dimensional than full joints.
- Eg hand-pose tracking



Adaptive discretization

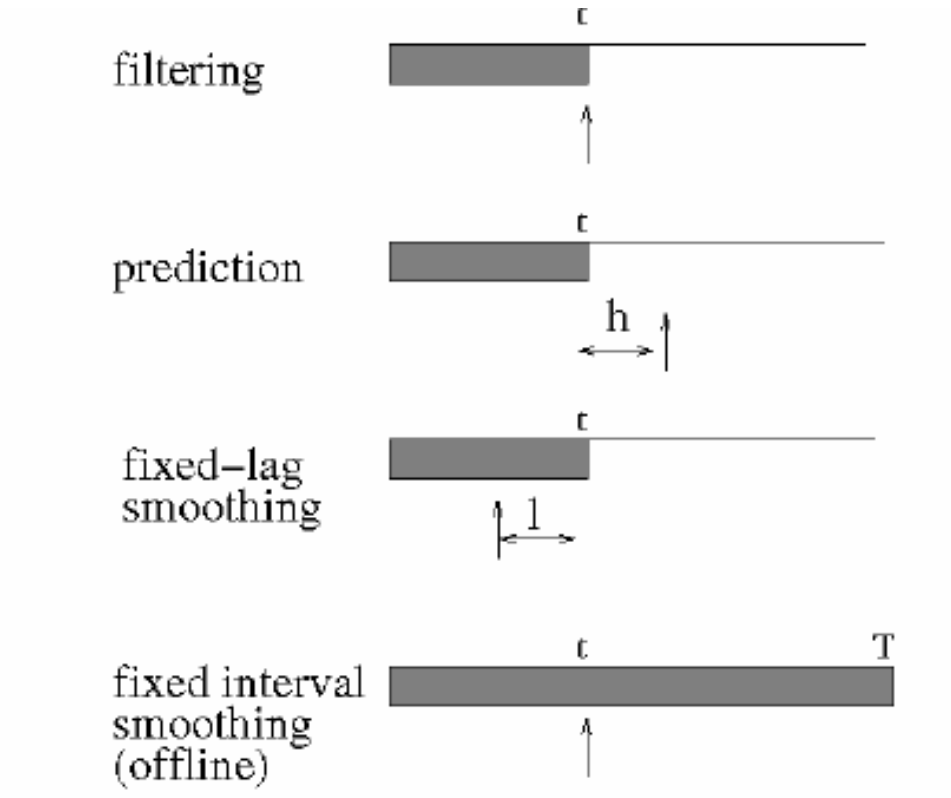
- We can discretize all the cts variables, then use a method for discrete vars.
- To increase accuracy, we expand the grid resolution for variables whose posterior entropy is high.
- Can use such approximations as proposal distributions for MH.

Stat 521A
Lecture 19

Outline

- Inference goals (15.1)
- Exact inference in DBNs (15.2)
- Factored belief sates (15.3.2)
- Particle filtering (15.3.3)
- Switching LDS (15.4.2)

Inference goals

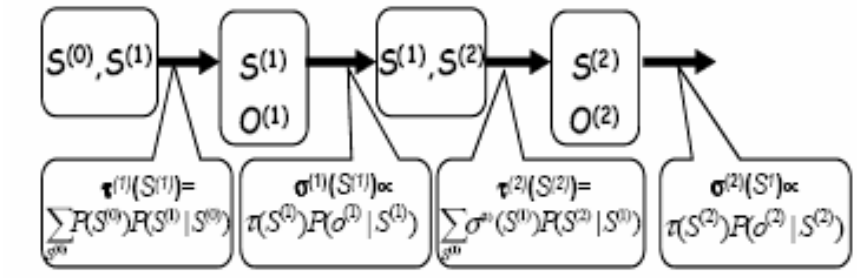


Exact filtering in HMMs

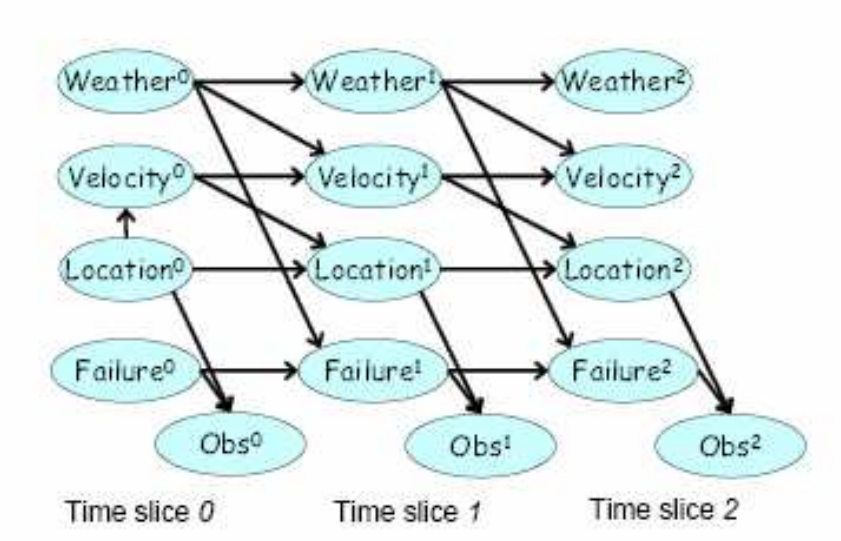
- We can apply the predict-update equations to any dynamical model

$$\begin{aligned}
\sigma^{(\cdot t+1)}(\boldsymbol{X}^{(t+1)}) &\triangleq P(\boldsymbol{X}^{(t+1)} \mid o^{(1:t)}) \\
&= \sum_{\boldsymbol{X}^{(t)}} P(\boldsymbol{X}^{(t+1)} \mid \boldsymbol{X}^{(t)}, o^{(1:t)}) P(\boldsymbol{X}^{(t)} \mid o^{(1:t)}) \\
&= \sum_{\boldsymbol{X}^{(t)}} P(\boldsymbol{X}^{(t+1)} \mid \boldsymbol{X}^{(t)}) \sigma^{(t)}(\boldsymbol{X}^{(t)}).
\end{aligned}$$

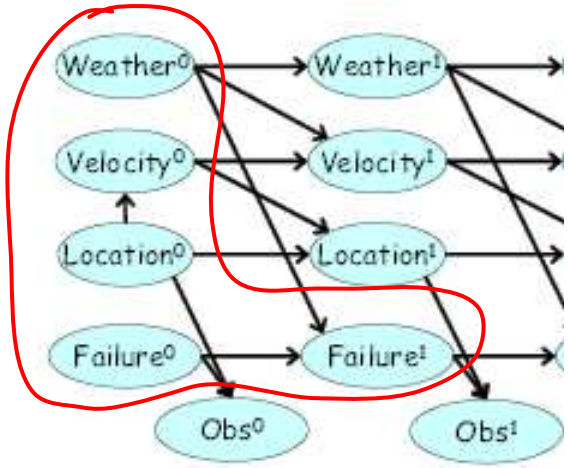
$$\begin{aligned}
\sigma^{(t+1)}(\boldsymbol{X}^{(t+1)}) &= P(\boldsymbol{X}^{(t+1)} \mid o^{(1:t)}, o^{(t+1)}) \\
&= \frac{P(o^{(t+1)} \mid \boldsymbol{X}^{(t+1)} \mid o^{(1:t)}) P(\boldsymbol{X}^{(t+1)} \mid o^{(1:t)})}{P(o^{(t+1)} \mid o^{(1:t)})} \\
&= \frac{P(o^{(t+1)} \mid \boldsymbol{X}^{(t+1)}) \sigma^{(\cdot t+1)}(\boldsymbol{X}^{(t+1)})}{P(o^{(t+1)} \mid o^{(1:t)})}.
\end{aligned}$$

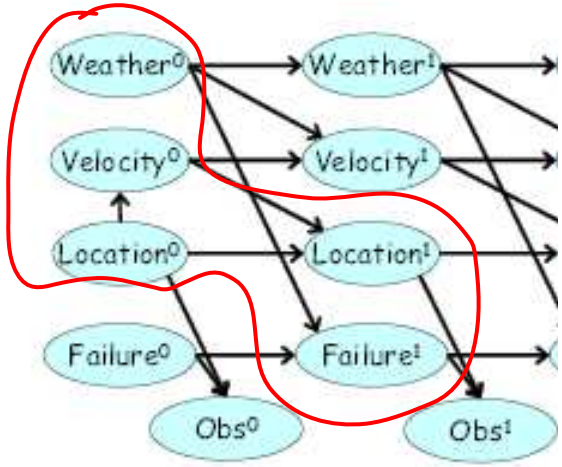


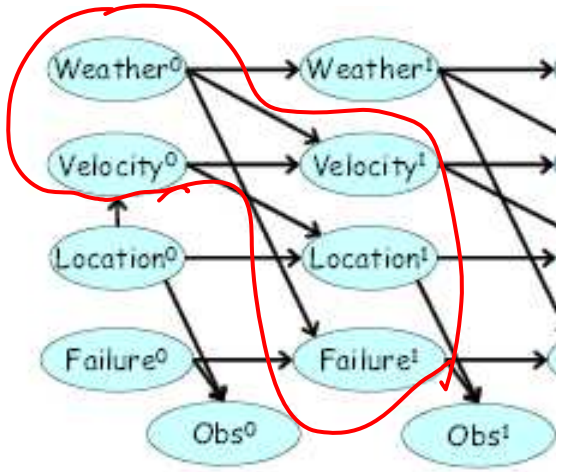
Entanglement

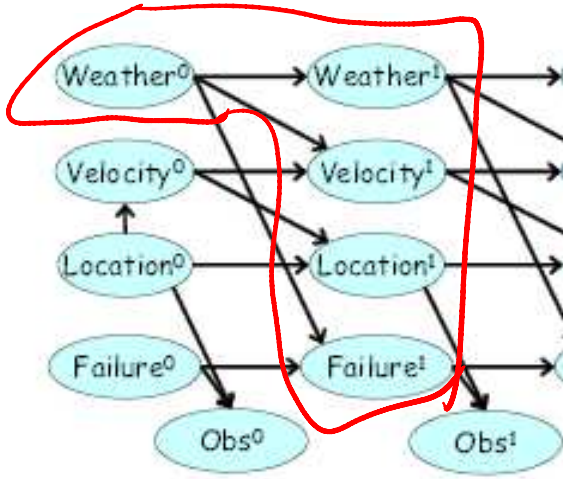


In the unrolled network, all the persistent nodes become correlated. Hence the belief state does not admit any factorization.









Frontier algorithm

- We need cliques that can store the interface variables

Factored frontier algorithm

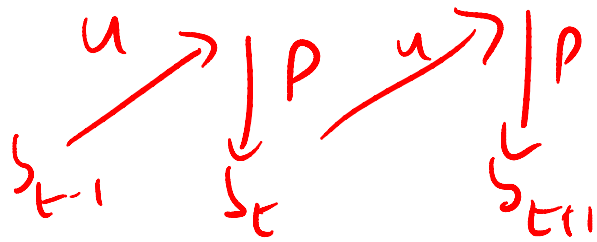
- Represent incoming belief state as a product of marginals

$$\hat{\sigma}^{(t)}(\mathcal{X}^{(t)}) = \prod_r (\beta_r^{(t)}[\boldsymbol{X}_r^{(t)}])^{\mu_r}.$$

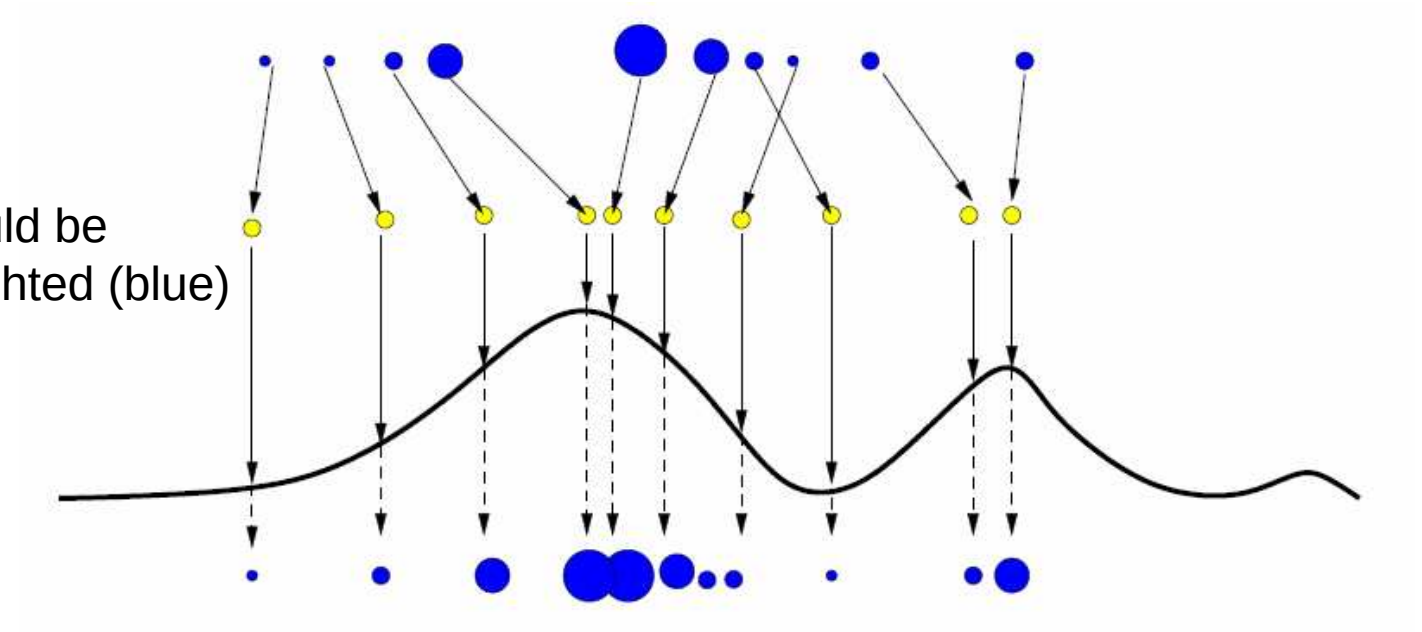
- Perform calibration in the 2-slice jtree
- Compute posterior marginals (M projection onto factored distribution)
- Can also use conditionally factored belief states

$$\left(\beta_g^{(t)}[\boldsymbol{Z}^{(t)}]\right)^{-(k-1)} \prod_{i=1}^{k} \beta_i^{(t)}[\boldsymbol{Z}^{(t)}, \boldsymbol{Y}_i^{(t)}],$$

- This is like EP without the backwards pass, aka ADF



Importance sampling

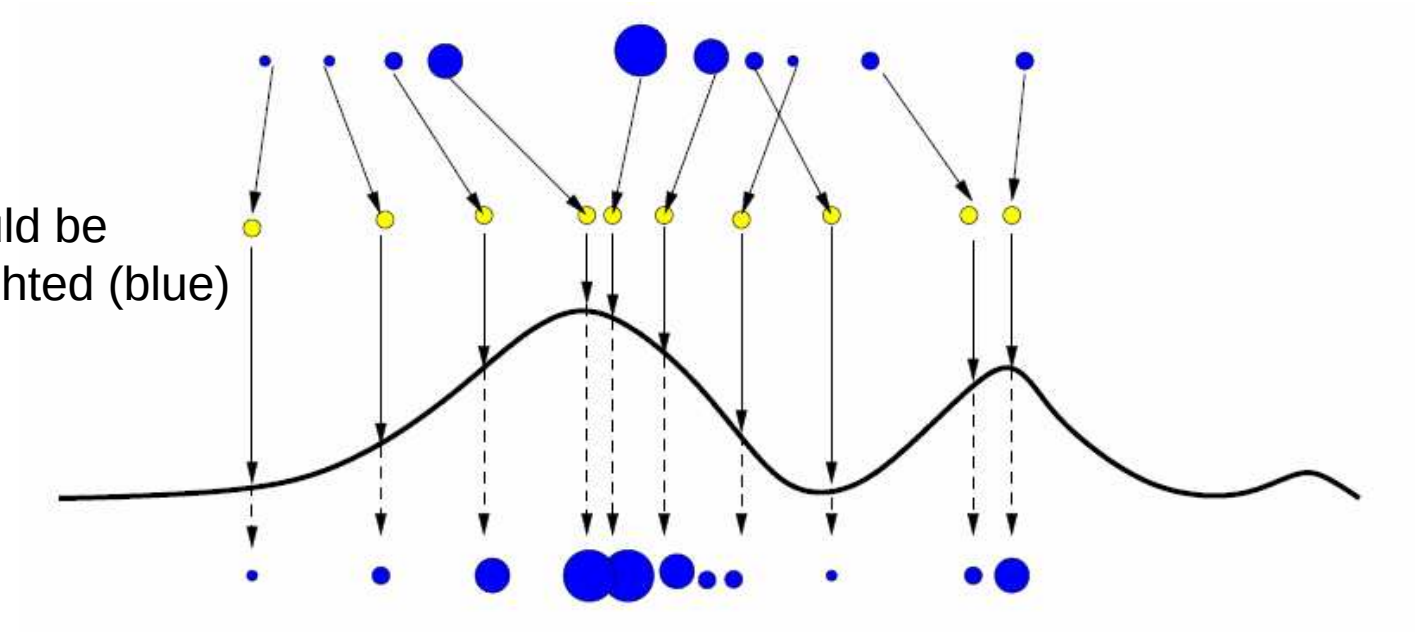


$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \approx \frac{1}{N}\sum_{i=1}^{N}\tilde{w}_t^{(i)} I(\mathbf{x}_{0:t} = \mathbf{x}_{0:t}^{(i)}) \quad (1)$$

$$\tilde{w}_t^{(i)} \stackrel{\text{def}}{=} \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}} \quad (2)$$

$$w_t^{(i)} \stackrel{\text{def}}{=} \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}^{(i)})} \quad (3)$$

Sequential Importance Sampling



Should be Weighted (blue)

Propose

Weight

Markov proposal

$$\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \pi(\mathbf{x}_0)\prod_{k=1}^{t}\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1},\mathbf{y}_{1:t})$$

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)}\frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)},\mathbf{y}_{1:t})}$$

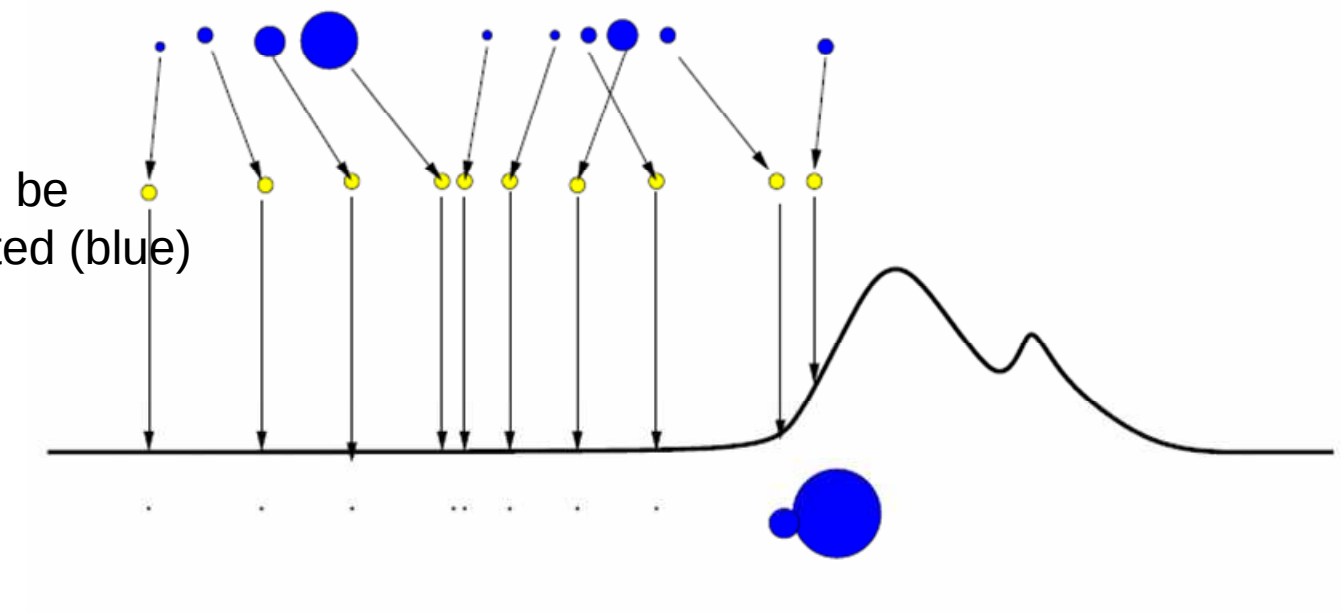
Propose from dynamical prior

$$\pi(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)},\mathbf{y}_{1:t}) = p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})$$

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)}p(\mathbf{y}_t|\mathbf{x}_t^{(i)})$$

Problem with SIS

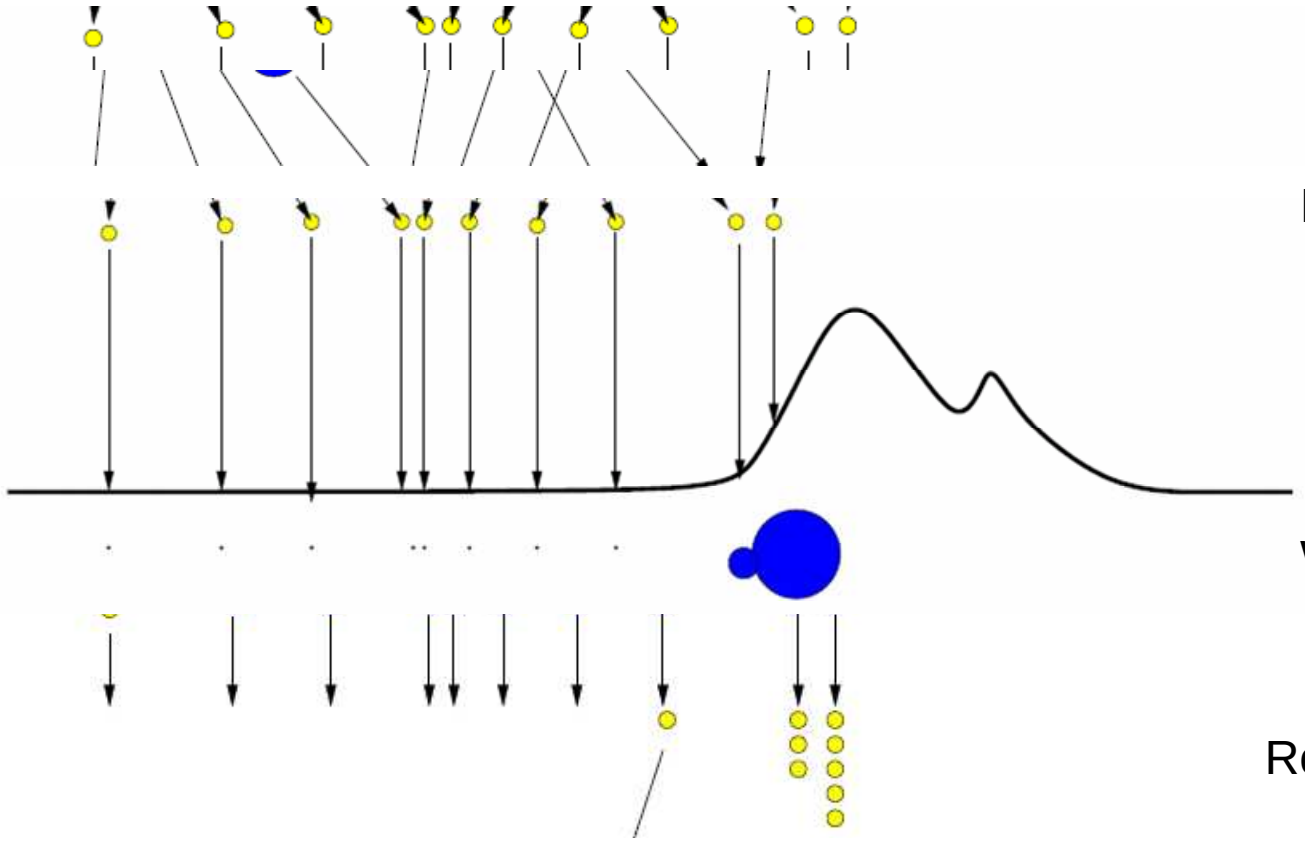
Should be
Weighted (blue)



Unlikely evidence “kills off” most particles
(Particle impoverishment) resulting in high variance estimate

SIR/ PF/ SOF/ SMC

Should be in diff locns



PF

1. Sequential importance sampling step

   - For $i = 1, \ldots, N$, sample

   $$\left(\widehat{\mathbf{x}}_t^{(i)}\right) \sim q(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t})$$

   and set

   $$\left(\widehat{\mathbf{x}}_{1:t}^{(i)}\right) \triangleq \left(\widehat{\mathbf{x}}_t^{(i)}, \mathbf{x}_{1:t-1}^{(i)}\right)$$

   - For $i = 1, \ldots, N$, evaluate the importance weights up to a normalising constant:

   $$w_t^{(i)} = \frac{p\left(y_t | \mathbf{x}_t^{(i)}\right) p\left(\widehat{\mathbf{x}}_t^{(i)} \middle| \widehat{\mathbf{x}}_{t-1}^{(i)}\right)}{q\left(\widehat{\mathbf{x}}_t | \widehat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{y}_{1:t}\right)}$$

   - For $i = 1, \ldots, N$, normalise the importance weights:

   $$\widetilde{w}_t^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_t^{(j)}\right]^{-1}$$

2. Selection step

   - Resample the discrete weighted measure $\{(\widehat{\mathbf{x}}_{1:t}^{(i)}, \tilde{w}_t^{(i)})\}_{i=1}^N$ to get an unweighted measure $\{(\mathbf{x}_{1:t}^{(i)}, \frac{1}{N})\}_{i=1}^N$

Example from Nando de Freitas

$$x_t = \frac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + v_t$$

$$y_t = \frac{x_t^2}{20} + w_t$$

where $x_0 \sim \mathcal{N}(0, \sigma_1^2)$, v_t and w_t are mutually independent white Gaussian noises, $v_t \sim \mathcal{N}(0, \sigma_v^2)$ and $w_t \sim \mathcal{N}(0, \sigma_w^2)$

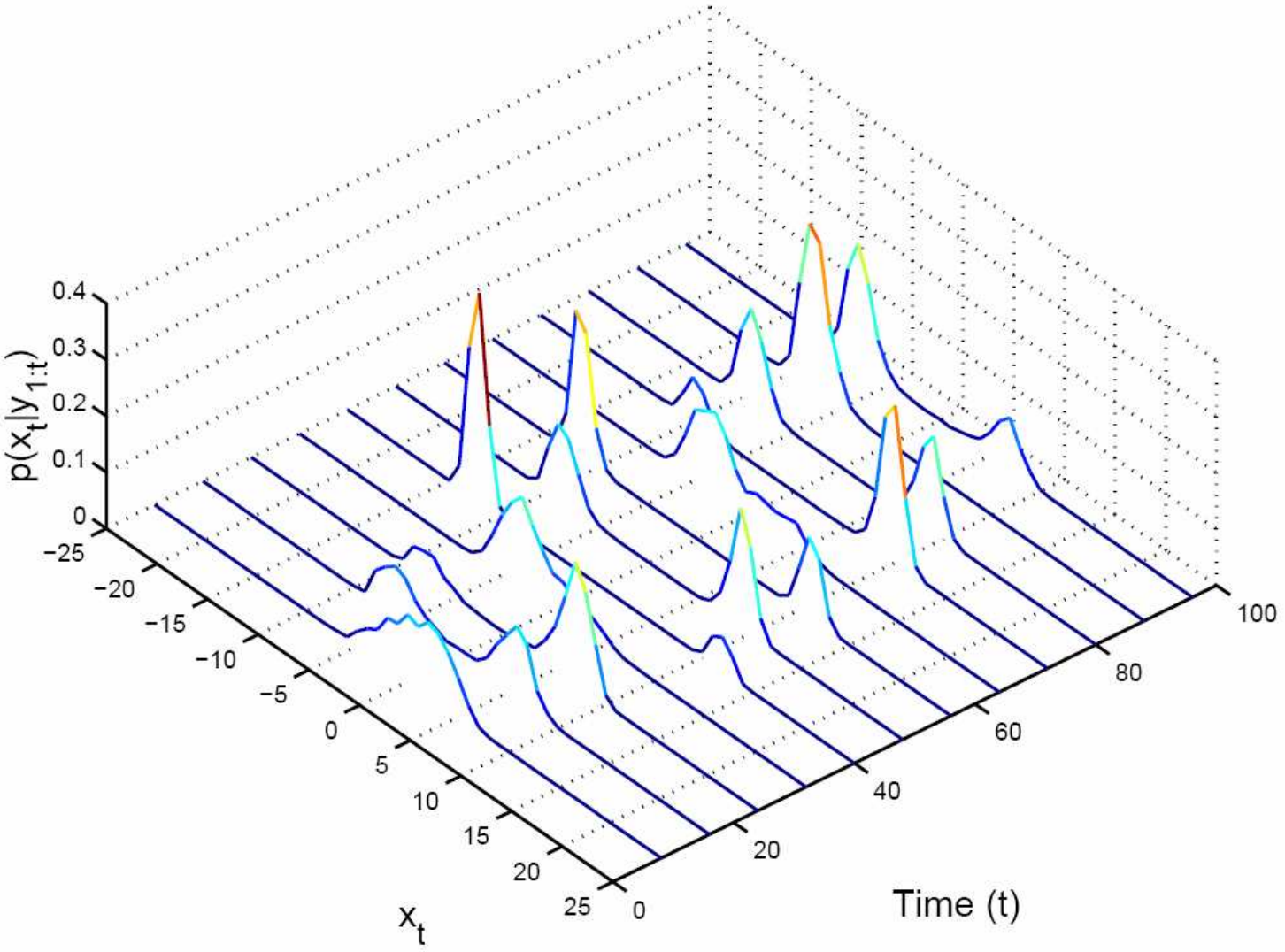
- For $i = 1, \ldots, N$, sample $\mathbf{x}_0^{(i)} \sim \mathcal{N}\left(0, \sigma_1^2\right)$
- For $i = 1, \ldots, N$, sample

$$x_t^{(i)} = \frac{1}{2} x_{t-1}^{(i)} + 25 \frac{x_{t-1}^{(i)}}{1 + x_{t-1}^{2(i)}} + 8 \cos\left(1.2t\right) + \mathcal{N}\left(0, \sigma_v^2\right)$$

- For $i = 1, \ldots, N$, evaluate the importance weights

$$\widetilde{w}_t^{(i)} = \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{1}{2\sigma_w^2}\left(y - \frac{x_t^{2(i)}}{20}\right)^2}$$

- Normalise the importance weights.
- Resample fittest samples (black-box).


0.4
0.3
0.2
0.1
0
$p(x_t|y_{1:t})$
−25
−20
−15
−10
−5
0
5
10
15
20
25
x_t
0
20
40
60
80
100
Time (t)

PF for DBNs

PF

```
1   for m = 1, ..., M
2     Sample x̄^(0)[m] from B_0
3     w^(0)[m] ← 1/M
4   for t = 1, 2, ...
5     for m = 1, ..., M
6       Sample x̄^(0:t−1) from the distribution P̂_D(t−1).
7         // Select sample for propagation
8       (x̄^(0:t)[m], w^(t)[m]) ← LW-2TBN(B_→, x̄^(0:t−1), o^(t))
9         // Generate time t sample and weight from selected sample
          x̄^(t−1)
10      D^(t) ← {(x̄^(0:t)[m], w^(t)[m]) : m = 1, ..., M}
11    σ̂^(t)(x) ← P_D^(t)
```

LW-2TBN

```
1   Let X'_1, ..., X'_n be a topological ordering of X' in B_→
2   w ← 1
3   for i = 1, ..., n
4     u_i ← (ξ, x')⟨Pa_X'_i⟩
5       // Assignment to Pa_X'_i in x_1, ..., x_n, x'_1, ..., x'_i−1
6     if X'_i ∉ O^(t) then
7       Sample x'_i from P(X'_i | u_i)
8     else
9       x'_i ← o^(t)⟨X'_i⟩    // Assignment to X'_i in o^(t)
10      w ← w · P(x'_i | u_i)    // Multiply weight by probability of desired value
11  return (x'_1, ..., x'_n), w
```

Condensation algorithm

Isard & Blake (ICCV98)

Monte Carlo Localization

Fox, Burgard, Dellaert, Thrun, AAAI'99

Optimal proposal distribution

- Optimal proposal is the posterior

$$\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t}) \quad = \quad p(\mathbf{x}_t|\mathbf{y}_t,\mathbf{x}_{t-1})$$

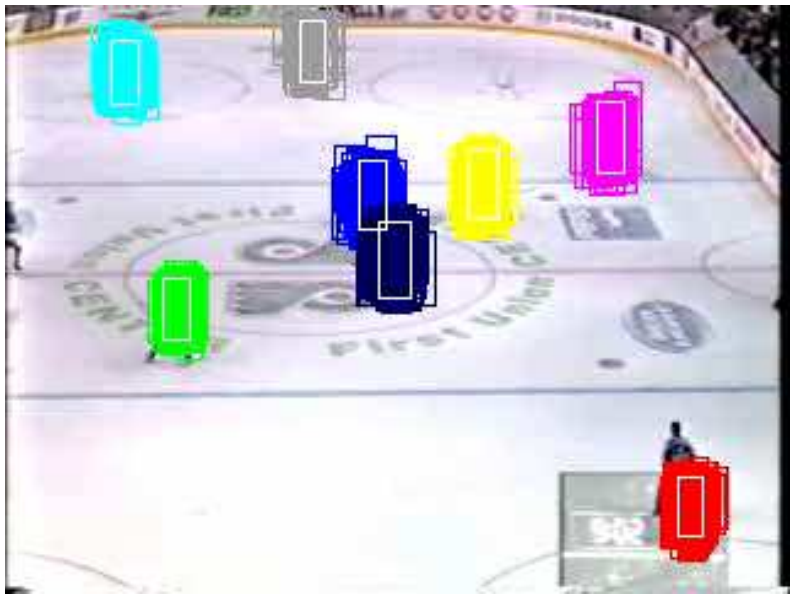
- Incremental weights are one-step-ahead predictive density

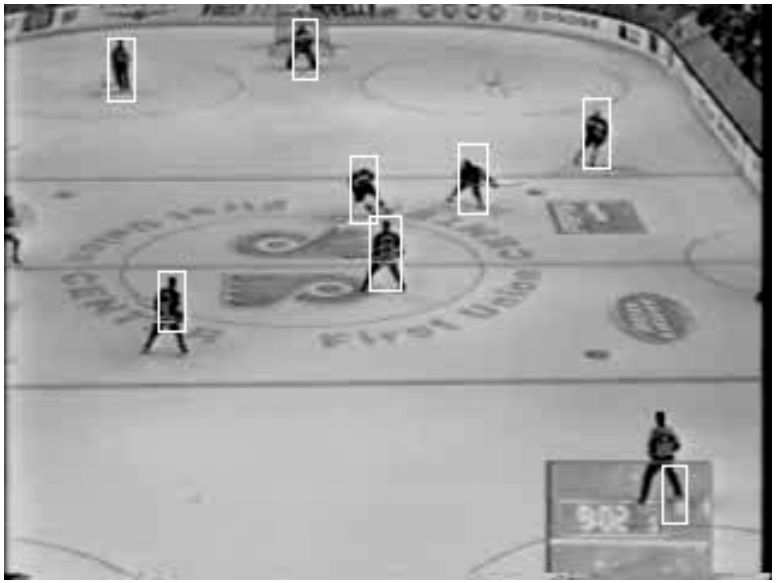
$$\begin{aligned}
\tilde{w}_t^{(i)} &\propto \tilde{w}_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{p(\mathbf{x}_t^{(i)}|\mathbf{y}_t,\mathbf{x}_{t-1}^{(i)})} \\
&= \tilde{w}_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_{t-1}^{(i)}) \\
p(\mathbf{y}_t|\mathbf{x}_{t-1}^{(i)}) &= \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})d\mathbf{x}_t
\end{aligned}$$

- Can approximate this using EKF, UKF, etc.

Boosted particle filter

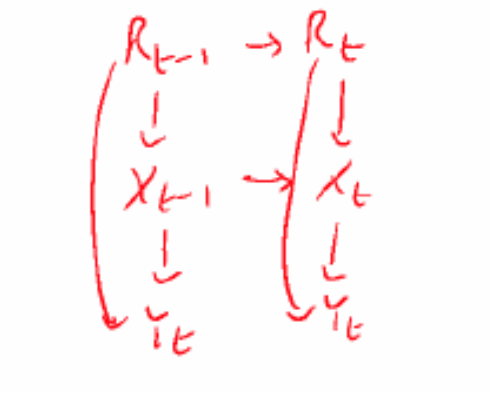
- Run a classifier, trained using boosting, to detect people, and use this as a proposal
- Okuma, Taleghani, de Freitas, Little, Lowe, ECCV04





RBPF

- Rao-Blackwellisation: integrate out X, sample R



- Distributional particles

$$\alpha^i_{t-1|t-1}(\mathbf{x}_{t-1}) \stackrel{\text{def}}{=} p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}, r^{(i)}_{1:t-1})$$

RBPF high level

Generic RBPF

1. Sequential importance sampling step

   - For $i = 1, \ldots, N$, sample

     $$\left(\widehat{r}_t^{(i)}\right) \sim q(r_t; r_{1:t-1}^{(i)}, y_{1:t})$$

     and set

     $$\left(\widehat{r}_{1:t}^{(i)}\right) \triangleq (\widehat{r}_t^{(i)}, r_{1:t-1}^{(i)})$$

   - For $i = 1, \ldots, N$, evaluate the importance weights up to a normalising constant:

     $$w_t^{(i)} = \frac{p\left(y_t | y_{1:t-1}, \widehat{r}_{1:t}^{(i)}\right) p\left(\widehat{r}_t^{(i)} \middle| \widehat{r}_{1:t-1}^{(i)}, y_{1:t-1}\right)}{q\left(\widehat{r}_t; \widehat{r}_{1:t-1}^{(i)}, y_{1:t}\right)}$$

   - For $i = 1, \ldots, N$, normalise the importance weights:

     $$\widehat{w}_t^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_t^{(j)}\right]^{-1}$$

2. Selection step

   - Resample the discrete weighted measure $\{(\widehat{r}_{1:t}^{(i)}, \widehat{w}_t^{(i)})\}_{i=1}^N$ to get an unweighted measure $\{(r_{1:t}^{(i)}, \frac{1}{N})\}_{i=1}^N$

3. Exact step

   - Update $p(\mathbf{X}_t | y_{1:t}, r_{1:t}^{(i)})$ given $p(\mathbf{X}_{t-1} | y_{1:t-1}, r_{1:t-1}^{(i)})$, $r_t^{(i)}$, and y_t.

RBPF updates

Then, for each possible value of r_t, we perform a predict-update cycle:

$$\alpha^i_{t|t-1}(\mathbf{x}_t, r_t) \stackrel{\text{def}}{=} p(\mathbf{x}_t, r_t|\mathbf{y}_{1:t-1}, r^{(i)}_{1:t-1}, r_t) = \int p(\mathbf{x}_t|r_t, \mathbf{x}_{t-1}) p(r_t|r^{(i)}_{1:t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}, r_t) d\mathbf{x}_{t-1} \quad (21.2$$

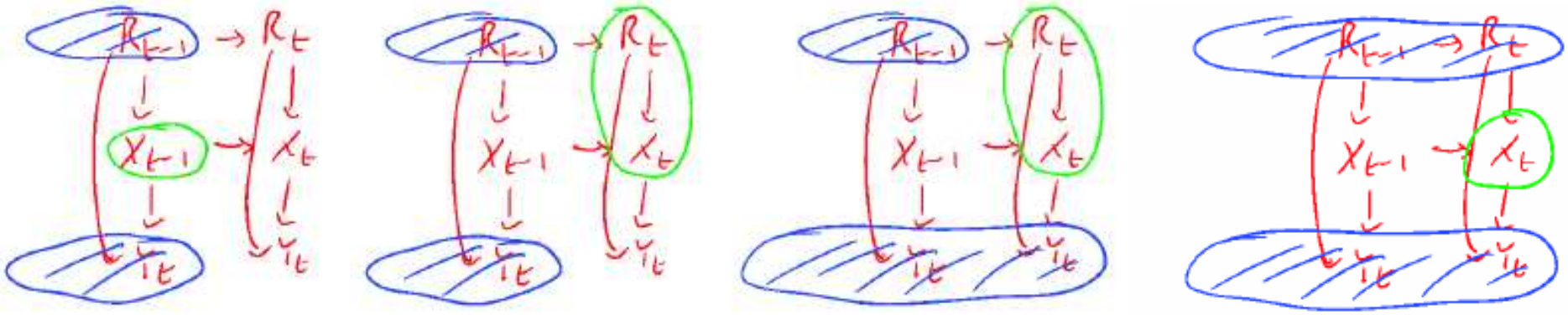
$$\alpha^i_{t|t}(\mathbf{x}_t, r_t) \stackrel{\text{def}}{=} p(\mathbf{x}_t, r_t|\mathbf{y}_{1:t-1}, \mathbf{y}_t, r^{(i)}_{1:t-1}, r_t) = \frac{p(\mathbf{y}_t|\mathbf{x}_t, r_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}, r^{(i)}_{1:t-1}, r_t)}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, r^{(i)}_{1:t-1}, r_t)} \quad (21.2$$

where

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, r^{(i)}_{1:t-1}, r_t) = \int p(\mathbf{y}_t|\mathbf{x}_t, r_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}, r^{(i)}_{1:t-1}, r_t) d\mathbf{x}_t \quad (21.3$$

Finally, once we have chosen $r^{(i)}_t$, we pick the corresponding updated distribution:

$$\alpha^i_{t|t}(\mathbf{x}_t) = \alpha^i_{t|t}(\mathbf{x}_t, r^{(i)}_t) \quad (21.3$$



(a) (b) (c)

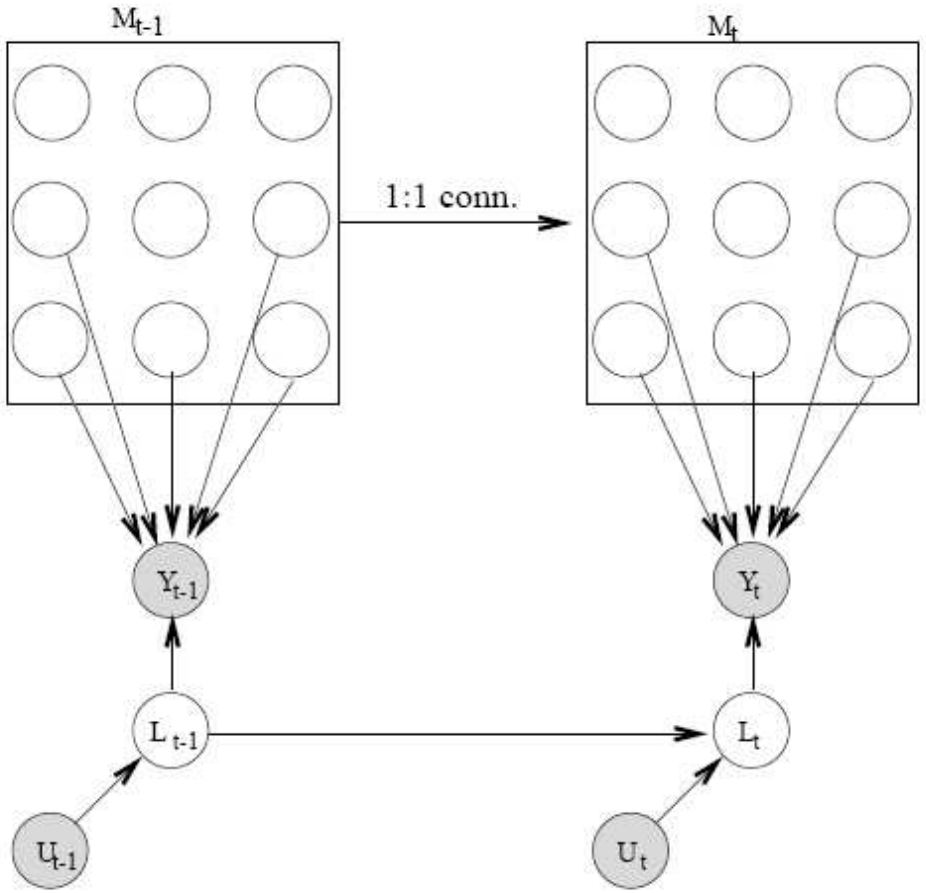
RBPF for Switching LDS

Algorithm 7: One step of the Mixture Kalman filter algorithm

for $i = 1 : N$ **do**
  $r_{t,i} \sim p(r|r_{t-1,i})$
  $(\boldsymbol{\mu}_{t,i}, \boldsymbol{\Sigma}_{t,i}, w_{t,i}) = \text{KFupdate}(\boldsymbol{\mu}_{t-1,i}, \boldsymbol{\Sigma}_{t-1,i}, \mathbf{y}_t, r_{t,i})$
for $i = 1 : N$ **do**
  $\tilde{w}_{t,i} = w_{t,i} \left[\sum_j w_{t,j}\right]^{-1}$
$\pi = \text{resample}(\tilde{w}_{t,1:N})$
return $(r_{t,\pi}, \boldsymbol{\mu}_{t,\pi}, \boldsymbol{\Sigma}_{t,\pi})$

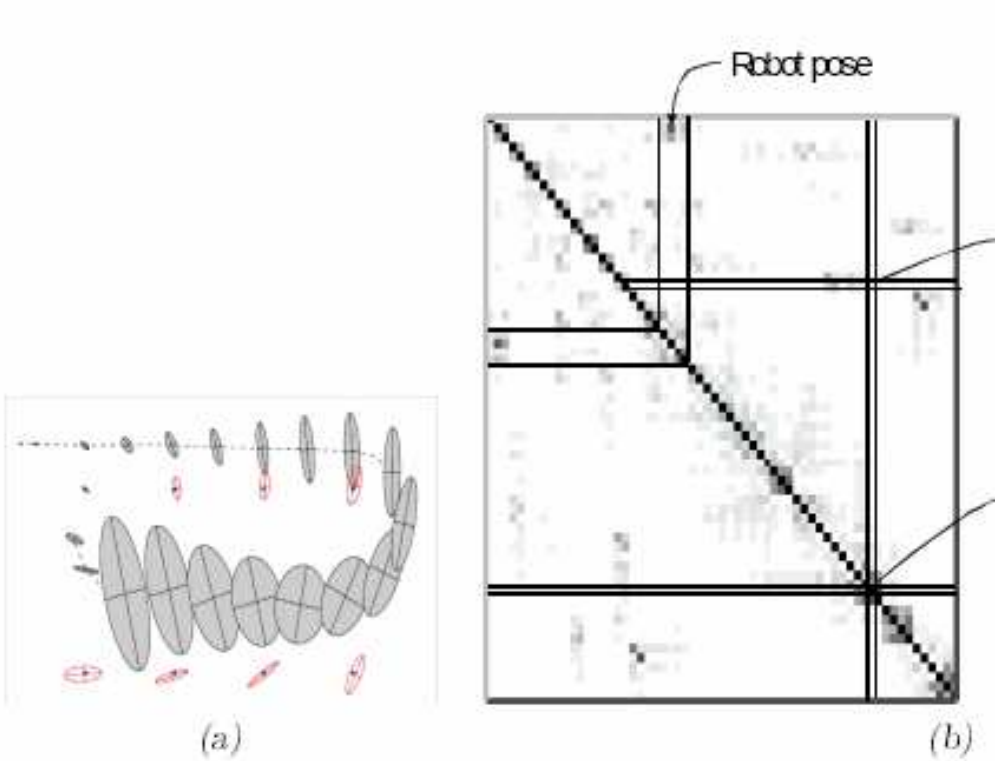
RBPF for SLAM

- Simultaneous Localization and Mapping
- Occupancy grid version (Murphy, NIPS'00)



FastSLAM

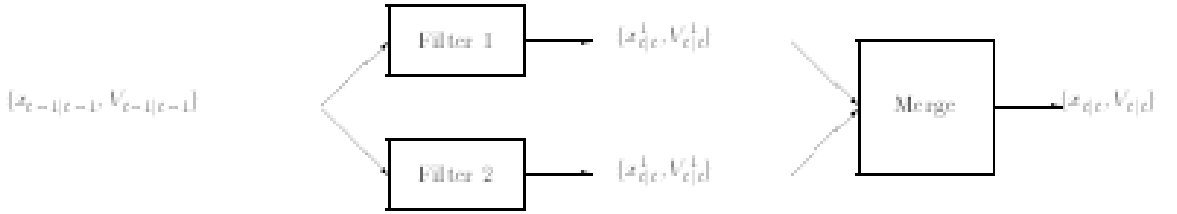
- Kalman filter version: replace covmat of size $(2K+2)^2$ with $P*K*2^2$ covmats, P=#particles, K=#num landmarks



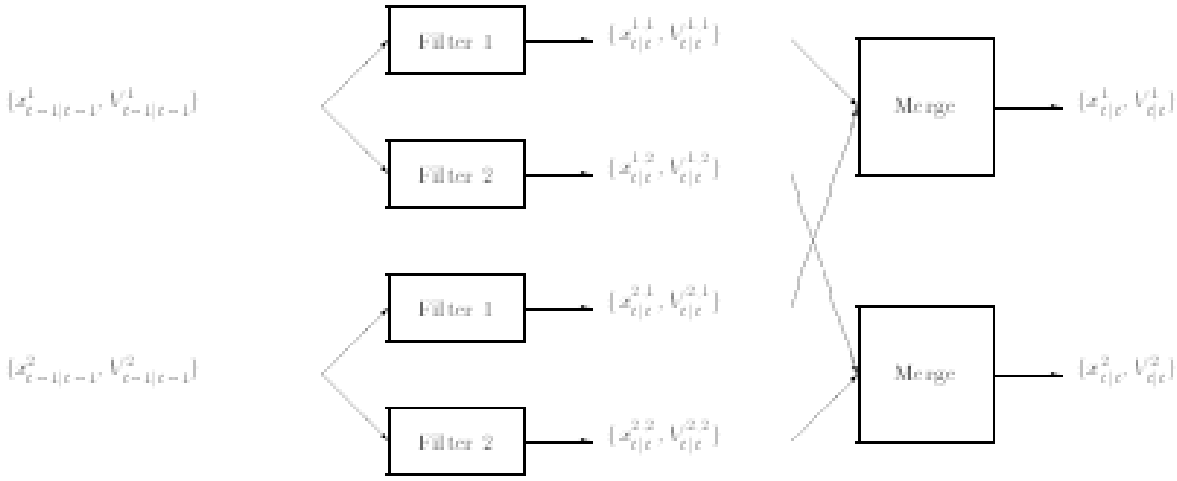
Montemerlo, Thrun, Koller, Wegbreit, AAAI'02

Switching LDS

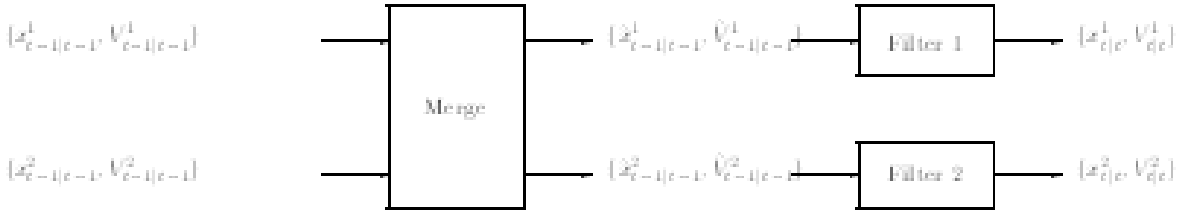
GBP1



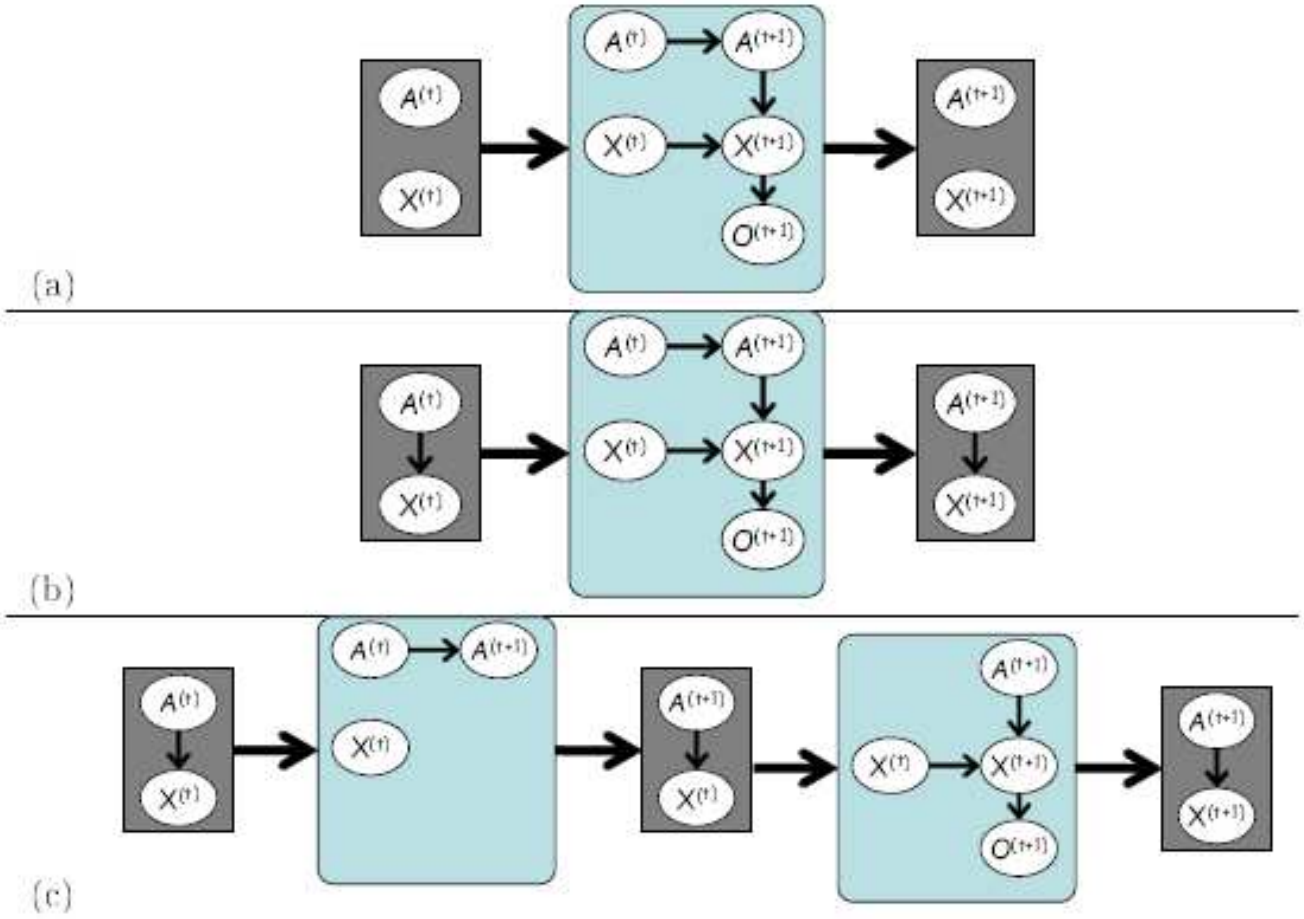
GBP2



IMM



EP approximations



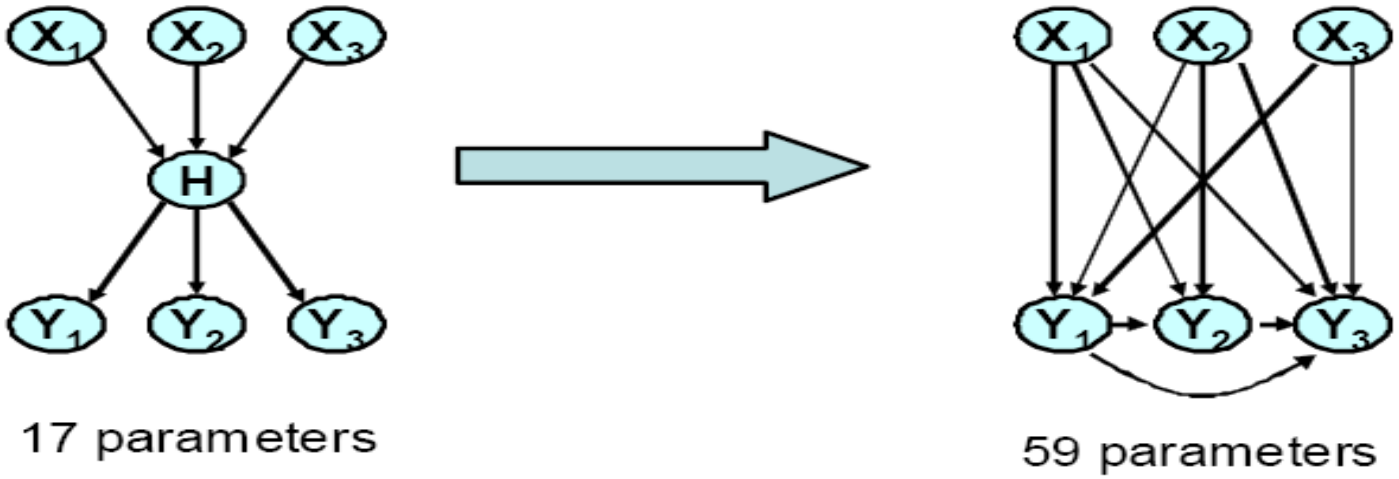
Stat 521A
Lecture 20

Outline

- Overview of learning (ch 16)
- MLE for DGMs (17.2)
- Bayesian parameter estimation for DGMs (17.4)
- Parameter tying (17.5)
- Hierarchical Bayes (17.5.4)
- PAC analysis (17.6)

Overview of learning

- Learn parameters or structure
- Observe all variables, or have missing values, or have known hidden variables, or have unknown hidden variables
- Hidden variables can simplify the model (fewer params)



Overview of learning

[t]

Learning Bayesian networks

	Complete data	Missing data	Hidden variables
Known structure	Closed form solution	• Iterated optimization to local maximum, • Inference on network multiple times	• Symmetrical solutions • Infinite # of solutions
Unknown structure, known variables	• Combinatorial optimization over structures • score has closed form	• Inference over multiple different network structures • no closed form for score	
Unknown vars	N/A	N/A	• Infinite number of possible solutions

Learning Markov networks

	Complete data	Missing data	Hidden variables
Known structure	• Convex optimization problem solved optimally via numerical optimization • Inference on network multiple times	• Non-convex problem • Iterated optimization to local maximum • Inference on network multiple times	• Symmetrical solutions • Infinite # of solutions
Unknown structure, known variables	• Combinatorial and numerical formulations • Can be solved via convex optimization • Inference over multiple different network structures		
Unknown vars	N/A	N/A	• Infinite number of possible solutions

Rest of ch 16

- Overfitting
- Cross validation
- Empirical risk minimization
- PAC bounds (see later)
- Generative vs discriminative
- Bias/variance tradeoff
- Prediction vs density estimation vs knowledge discovery

MLE for DGMs

- Assume DAG is known and variables are fully observed
- The likelihood factorizes into a product of local likelihoods, so we can optimize each CPD independently

$$
\begin{aligned}
p(\mathcal{D}|\boldsymbol{\theta}) &= \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\theta}) \\
&= \prod_{n=1}^{N}\prod_{i=1}^{D} p(x_{in}|\mathbf{x}_{\pi_i,n},\boldsymbol{\theta}_i) \\
&= \prod_{i=1}^{D}[\prod_{n=1}^{N} p(x_{in}|\mathbf{x}_{\pi_i,n},\boldsymbol{\theta}_i)] \\
&= \prod_{i=1}^{D} p(\mathcal{D}_i|\boldsymbol{\theta}_i)
\end{aligned}
$$

Tabular CPDs

$$\theta_{ijk} \stackrel{\text{def}}{=} p(X_i = k | \mathbf{X}_{\pi_i} = j)$$

$$\prod_{n=1}^{N} p(x_{in} | \mathbf{x}_{\pi_i,n}, \boldsymbol{\theta}_i) = \prod_{n=1}^{N} \prod_{j=1}^{r_i} \prod_{k=1}^{q_i} \theta_{ijk}^{I(x_{i,n}=k, \mathbf{x}_{\pi_i,n}=j)}$$

$$= \prod_{j} \prod_{k} \theta_{ijk}^{N_{ijk}}$$

$$N_{ijk} \stackrel{\text{def}}{=} \sum_{n=1}^{N} I(x_{i,n} = k, \mathbf{x}_{\pi_i,n} = j)$$

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{\sum_{j'=1}^{r_i} N_{ij'k}}$$

$$\hat{\theta}_{x|u} = \frac{M[u, x]}{M[u]},$$

MLE for linear Gaussian CPDs

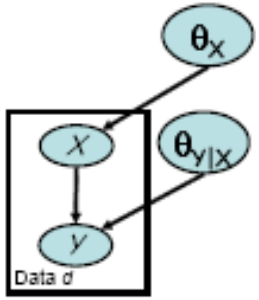
- Use usual linear regression equations

$$p(x_i|\mathbf{x}_{\pi_i}, \boldsymbol{\theta}_i) \quad = \quad \mathcal{N}(x_i|\mathbf{w}_i^T \mathbf{x}_{\pi_i}, \sigma_i^2)$$

Bayesian parameter estimation

- Global parameter independence

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{D} p(\boldsymbol{\theta}_i)$$



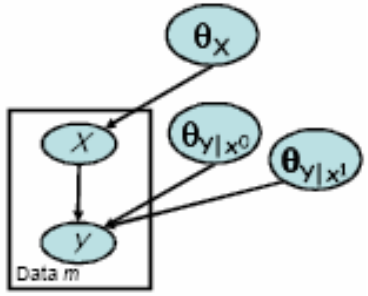
- Implies factorized posterior

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto \prod_{i} p(\boldsymbol{\theta}_i) p(\mathcal{D}_i|\boldsymbol{\theta}_i)$$

- For multinomials, let us assume local param indep

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{n} \prod_{j=1}^{r_i} p(\boldsymbol{\theta}_{ij})$$

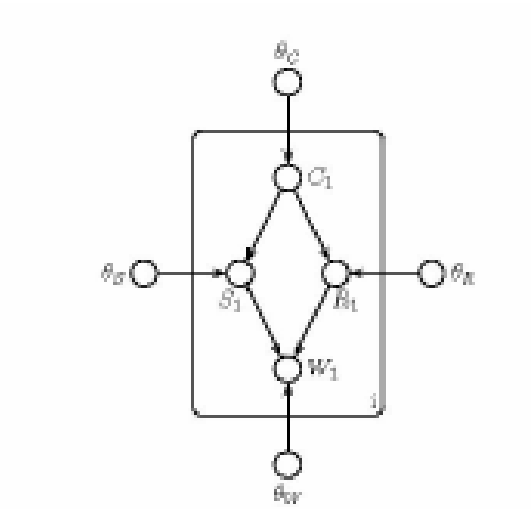
- Geiger & Heckerman showed this implies theta_{ij} must have a Dirichlet prior



Tabular CPDs

- We have

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto \prod_{i=1}^{D}\prod_{j=1}^{r_i} \text{Dir}(\boldsymbol{\theta}_{ij}|\boldsymbol{\alpha}_{ij}+\mathbf{N}_{ij})$$

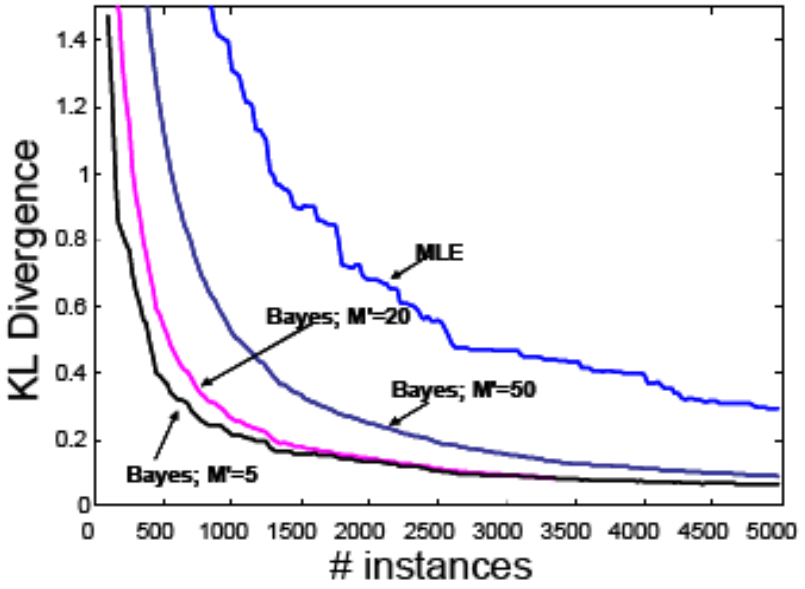


i	C	S	R	W
1	0	0	0	0
2	0	0	1	1
3	1	1	1	1

p(θC)		p(θR | C = 0)		p(θR | C = 1)	
1	1	1	1	1	1
2	1	2	1	1	1
3	1	2	2	1	1
3	2	2	2	1	2

Example

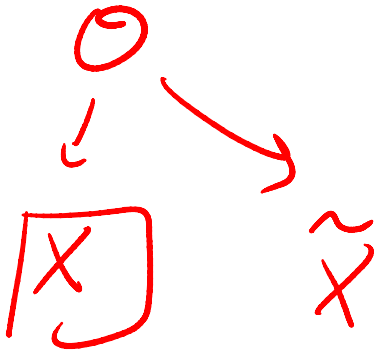
- ICU alarm network, 37 nodes, 504 params
- Compute theta-hat using MLE or posterior mean. Then compute KL(p(X|theta*), p(X|theta-hat)) as a function of sample size.



Posterior predictive density

- We can predict future variables by integrating out the params

$$p(\tilde{X}|\mathcal{D}) = \int p(\tilde{X}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$$



- In the case of Dirichlet-multinomial model, this is equivalent to plugging in the posterior mean

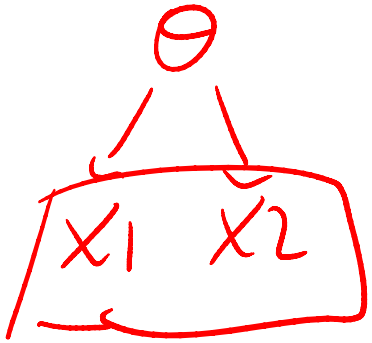
$$\begin{aligned} p(\tilde{X} = k|\mathcal{D}) &= \int \theta_k p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} \\ &= \int \theta_k p(\boldsymbol{\theta}_k|\mathcal{D})d\boldsymbol{\theta} \\ &= \overline{\theta}_k = \frac{\alpha_k + N_k}{\sum_{k'} \alpha_{k'} + N_{k'}} \end{aligned}$$

MAP estimation

- Since in general computing the posterior is difficult, a compromise is to compute a MAP estimate
- However, the result is not invariant to parameterization – change of variables formula changes the prior density (Box 17.D)
- Reparameterizing the likelihood does not change the MLE, since the lik is not a density function
- Reparameterizing the posterior does not change anything, since we integrate over params

Parameter tying

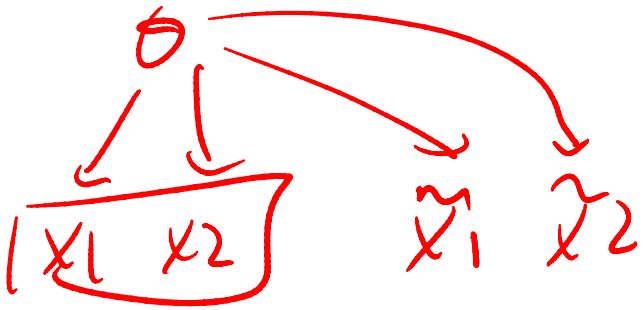
- We just pool the sufficient statistics from the nodes that share the same params



$$p(\boldsymbol{\theta}|\mathcal{D}) = \text{Dir}(\alpha_k + \sum_n I(X_{1n} = j) + \sum_n I(X_{2n} = j)$$

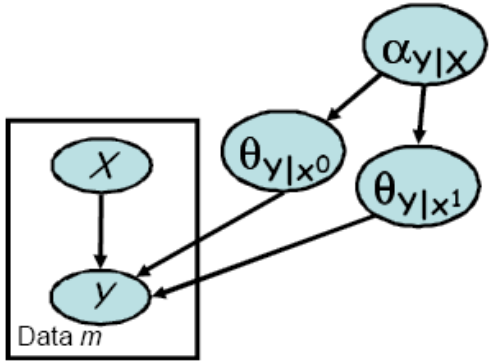
Prediction with tied params

- A subtlety arises when computing the posterior predictive density with tied params
- When we observe Xtilde1, we learn something about theta that helps us predict Xtilde2. So we cannot just multiply the postpred for each node separately, but need to use the formula for a batch of data

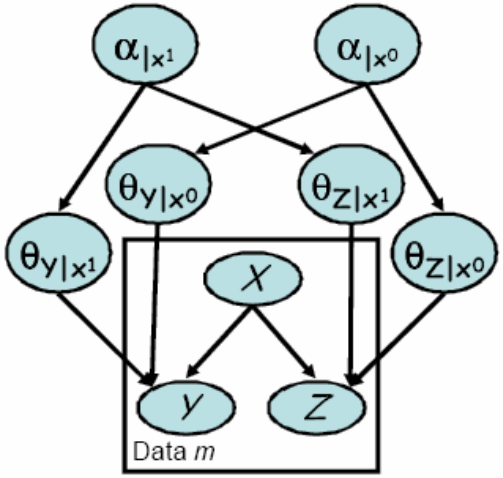


Hierarchical priors

- Encourage params to be similar across conditioning contexts (rows) within 1 CPD

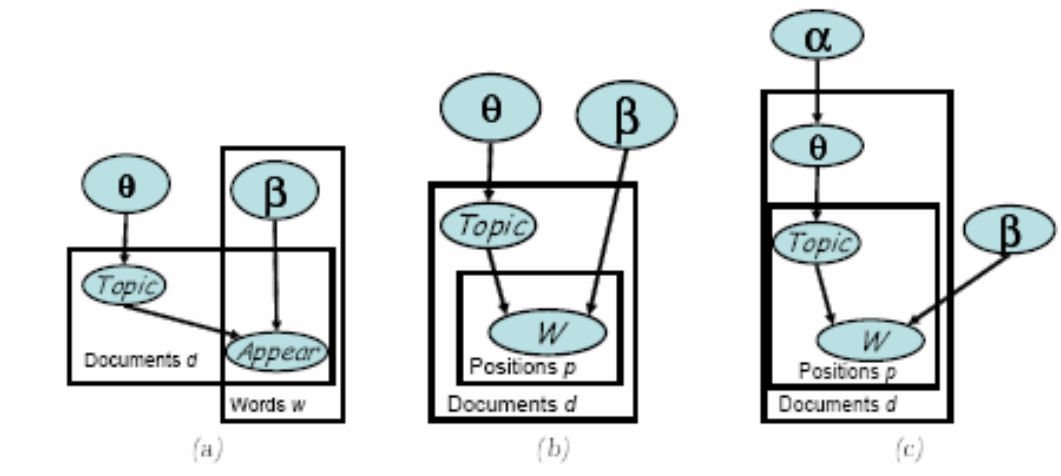


- Encourage params to be similar across response for each conditionign context



Text classification

- Let T(d) = t be topic of document d, ~Mun(θ)
- Product of Bernoullis, A(w,d) in {0,1} ~ Ber($\beta_{w,t}$), w =1:K
- Product of multinoullis, W(p,d) in {1,…,K} ~ Mun(β_t), p=1:len(d)
- Latent Dirichlet Allocation: θ(d) = distribution over topics, ~Dir(α), T(p,d) = t, W(p,d) ~ Mun(β_t)



PAC analysis

- Probably approximately correct
- Let P*M be distribution over datasets of size M drawn from P*
- $P_{ML(D)}$ be distribution over X given by model M learned using algo L on data D
- We want to prove that

Let $\epsilon > 0$ be our approximation parameter and $\delta > 0$ our confidence parameter. Then, for M "large enough", we have that

$$P_M^*(\{\mathcal{D} : D(P^* \| P_{\mathcal{M}_{L(\mathcal{D})}}) \leq \epsilon\}) \geq 1 - \delta.$$

- Frequentist analysis of estimator; bounds on deviation from 'truth'

Excess risk

- Minimizing KL(P*,P) may be hard if P* is not in the model class of P
- Define best achievable param in class as

$$\theta^{\mathrm{opt}} = \arg \min_{\theta \in \Theta[\mathcal{G}]} D(P^* | P_\theta).$$

- Define excess risk as

$$D(P^* \| P_\theta) - D(P^* \| P_{\theta^{\mathrm{opt}}}) :$$

DGM param learning: PAC bounds

Theorem 17.6.8: *Let $\mathcal{G}$ be a network structure, and P^* a distribution consistent with some network $\mathcal{G}^*$ such that $P^*(x_i \mid \mathrm{pa}_i^{\mathcal{G}^*}) \geq \lambda$ for all i, x_i, and $\mathrm{pa}_i^{\mathcal{G}^*}$. If P is the distribution learned by maximum likelihood estimate for $\mathcal{G}$, then*

$$P\left(\boldsymbol{D}(P^* \| P) - \boldsymbol{D}(P^* \| P_{\boldsymbol{\theta}^{\mathrm{opt}}}) > n\epsilon\right) \leq nK^{d+1} e^{-2M\lambda^{2(d+1)}\epsilon^2 \frac{1}{(1+\epsilon)^2}}$$

where K is the maximal variable cardinality and d is the maximum number of parents in $\mathcal{G}$.

Corollary 17.6.9: *Under the conditions of Theorem 17.6.8, if*

$$M \geq \frac{1}{2} \frac{1}{\lambda^{2(d+1)}} \frac{(1+\epsilon)^2}{\epsilon^2} \log \frac{nK^{d+1}}{\delta},$$

then

$$P\left(\boldsymbol{D}(P^* \| P) - \boldsymbol{D}(P^* \| P_{\boldsymbol{\theta}^{\mathrm{opt}}}) < n\epsilon\right) > 1 - \delta.$$

Stat 521A
Lecture 21

Outline

- Overview of structure learning
- Constraint based approach (18.2)
- Scoring functions (18.3)

Overview of structure learning

- Goals: density estimation and knowledge discovery
- Can only learn graph up to Markov equivalence
- 2 main approaches:
- Find PDAG which is an I-map of the empirical distribution, using conditional independence test (eg \chi^2) at the 5% level in lieu of oracle
- Find MAP DAG by defining a scoring and search through DAG space
- Can also do Bayes model averaging over DAGs to get posterior of features of interest eg predictive density, edge/path marginals, etc

Assumptions behind constraint based

- Each node has a fan-in of at most d
- We have a CI oracle $X \perp Y \mid Z$ that gives correct results for conditioning sets up to size 2d+2
- P* is faithful to G*
- Def 3.3.4. A distribution P is faithful to G if, whenever $X \perp Y \mid Z$ in I(P), we have dsep_G(X;Y|Z)

  i.e., there are no “non-graphical” independencies buried in the parameters

Deriving graphs from distributions

- Sec 3.4, from Lecture 2
- So far, we have discussed how to derive distributions from graphs.
- But how do we get the DAG?
- Assume we have access to the true distribution P, and can answer questions of the form

$$P \models X \perp Y | Z$$

- For finite data samples, we can approximate this oracle with a CI test – the frequentist approach to graph structure learning (see ch 18)
- What DAG can be used to represent P?

Minimal I-map

- The complete DAG is an I-map for any distribution (since it encodes no CI relations)
- Def 3.4.1. A graph K is a minimal I-map for a set of independencies I if it is an I-map for I, and if the removal of even a single edge from K renders it not an I-map.
- To derive a minimal I-map, we pick an arbitrary node ordering, and then find some minimal subset U to be X_i's parents, where

$$X_i \perp \{X_1, \ldots, X_{i-1}\} \setminus U | U$$

- (K2 algorithm replace this CI test with a Bayesian scoring metric: sec 18.4.2).

Constructing I-map given ordering

Algorithm 3.2 Procedure to build a minimal I-map given an ordering

```
    Procedure Build-Minimal-I-Map (
      X_1, ..., X_n    // an ordering of random variables in 𝒳
      ℐ    // Set of independencies
    )
1     Set 𝒢 to an empty graph over 𝒳
2     for i = 1, ..., n
3       U ← {X_1, ..., X_{i-1}}    // U is the current candidate for parents of X_i
4       for U' ⊆ {X_1, ..., X_{i-1}}
5         if U' ⊂ U and (X_i ⊥ {X_1, ..., X_{i-1}} − U' | U') ∈ ℐ then
6           U ← U'
7         // At this stage U is a minimal set satisfying (X_i ⊥
            {X_1, ..., X_{i-1}} − U | U)
8         // Now set U to be the parents of X_i
9       for X_j ∈ U
10        Add X_j → X_i to 𝒢
11    return 𝒢
```

Effect of node ordering

- "Bad" node orderings can result in dense, unintuitive graphs.
- Eg L,S,G,I,D. Add L. Add S: must add L as parent, since $P \not\models L \perp S$. Add G: must add L,S as parents.

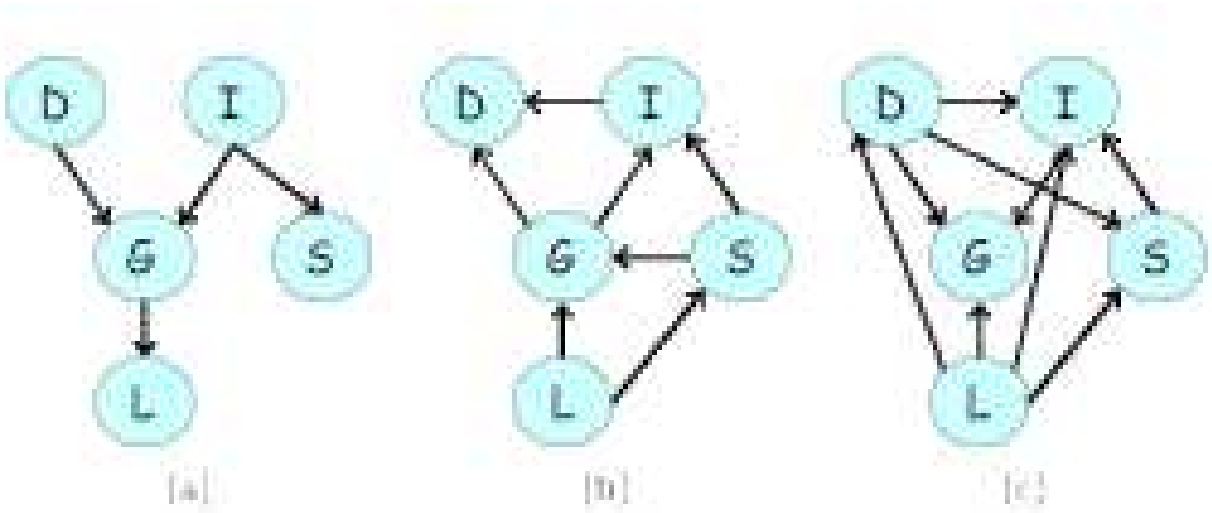


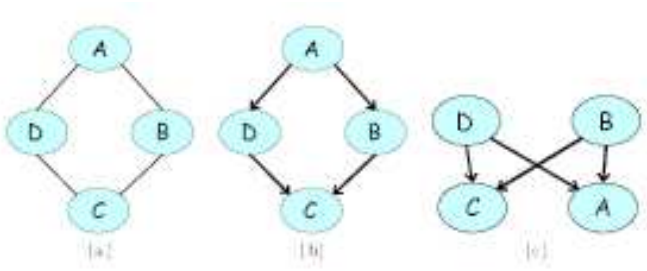
Figure 3.8 Three minimal I-maps for $P_{\{Student\}}$, induced by different orderings: (a) D, I, S, G, L (b) L, S, G, I, D (c) L, D, S, I, G

Dealing with node ordering

- Search over orders
- Work with PDAGs

Perfect maps

- Minimal I-maps can have superfluous edges.
- Def 3.4.2. Graph K is a perfect map for a set of independencies I if I(K)=I. K is a perfect map for P if I(K)=I(P).
- Not all distributions can be perfectly represented by a DAG.
- Eg let Z = xor(X,Y) and use some independent prior on X, Y. Minimal I-map is X -> Z <- Y. However, X $\perp$ Z in I(P), but not in I(G).
- Eg. A $\perp$ C | {B,D} and B $\perp$ D | {A,C}, A dep | B,C, etc



Finding perfect maps

- If P has a perfect map, we can find it in polynomial time, using an oracle for the CI tests.
- We can only identify the graph up to I-equivalence, so we return the PDAG that represents the corresponding equivalence class.
- The method has 3 steps (see sec 3.4.3)
  - Identify undirected skeleton
  - Identify immoralities
  - Compute eclass (compelled edges)
- This algorithm has been used to claim one can infer causal models from observational data, but this claim is controversial

Algorithm due to Verma & Pearl 1991, Spirtes, Glymour, Scheines 1993, Meek 1995

Identifying the undirected skeleton

- Initially connect all node pairs
- Remove an edge if we find a U st Xi ⊥ Xj | U

Lemma 3.4.8: *Let $\mathcal{G}^*$ be an I-map of a distribution P, and let X and Y be two variables that are not adjacent in $\mathcal{G}^*$. Then either $P \models (X \perp Y \mid \mathrm{Pa}_X^{\mathcal{G}^*})$ or $P \models (X \perp Y \mid \mathrm{Pa}_Y^{\mathcal{G}^*})$.*

- Hence we can restrict our search for witnesses U to the sets and

$$U \subseteq \mathcal{X} - \{X_i, X_j\} - \mathrm{Nb}_{X_i}^{\mathcal{H}},$$

$$U \subseteq \mathcal{X} - \{X_i, X_j\} - \mathrm{Nb}_{X_j}^{\mathcal{H}}.$$

Identifying the undirected skeleton

Algorithm 3.3 Algorithm for recovering undirected a distribution P for which $\mathcal{G}^*$ is a P-map

```
    Procedure Build-PMap-Skeleton (
      X = {X_1, ..., X_n},    // Set of random variables
      P,                       //      Distribution over X
    // Bound on witness set
    )
1     Let H be the complete undirected graph over X
2     for X_i, X_j in X
3       U_{X_i,X_j} ← ∅
4       for U ∈ Witnesses(X_i, X_j, H, d)
5             // Consider U as a witness set for X_i, X_j
6         if P ⊨ (X_i ⊥ X_j | U) then
7           U_{X_i,X_j} ← U
8           Remove X_i — X_j from H
9           break
10      return (H, {U_{X_i,X_j} : i, j ∈ {1, ..., n}})
```

Complexity

This algorithm will recover the correct skeleton given that $\mathcal{G}^*$ is a P-map of P and has bounded indegree d. If P does not have a P-map, then the algorithm can fail; see Exercise 3.22. This algorithm has complexity of $O(n^{d+2})$ since we consider $O(n^2)$ pairs, and for each perform $O((n-2)^d)$ independence tests. We greatly reduce the number of independence tests by ordering potential witnesses accordingly, and by aborting the inner loop once we find a witness for a pair (after line 9). However, for pairs of variables that are directly connected in the skeleton, we still need to evaluate all potential witnesses.

Identifying immoralities

Proposition 3.4.9: *Let $\mathcal{G}^*$ be a P-map of a distribution P, and let X, Y and Z be variables that form an immorality $X \rightarrow Z \leftarrow Y$. Then, $P \not\models (X \perp Y \mid \boldsymbol{U})$ for any set $\boldsymbol{U}$ that contains Z.*

Proposition 3.4.10: *Let $\mathcal{G}^*$ be a P-map of a distribution P, and let the triplet X, Y, Z be a potential immorality in the skeleton of $\mathcal{G}^*$, such that $X \rightarrow Z \leftarrow Y$ is not in $\mathcal{G}^*$. If $\boldsymbol{U}$ is such that $P \models (X \perp Y \mid \boldsymbol{U})$, then $Z \in \boldsymbol{U}$.*

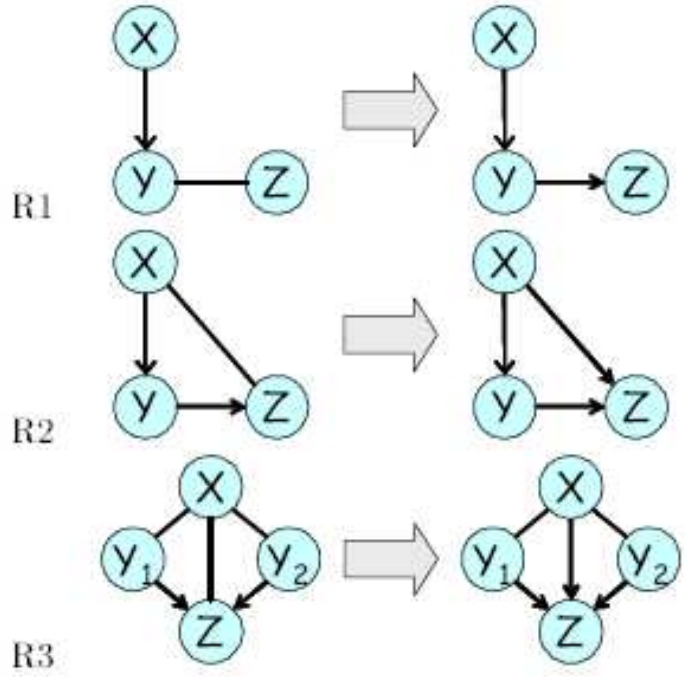
Combining these two results, we see that a potential immorality $X - Z - Y$ is an immorality if and only if Z is not in the witness set(s) for X and Y. That is, if $X - Z - Y$ is an immorality, then Proposition 3.4.9 shows that Z is not in any witness set $\boldsymbol{U}$; conversely, if $X - Z - Y$ is not an immorality, the Z must be in every witness set $\boldsymbol{U}$. Thus, we can use the specific witness set $\boldsymbol{U}_{X,Y}$ that we recorded for X, Y in order to determine whether this triplet is an immorality or not: we simply check whether $Z \in \boldsymbol{U}_{X,Y}$. If $Z \notin \boldsymbol{U}_{X,Y}$, then we declare the triplet an immorality. Otherwise, we declare that it is not an immorality. The Mark-Immoralities procedure shown in Algorithm 3.4 summarizes this process.

```
)
1    K ← S
2    for X_i, X_j, X_k such that X_i - X_j - X_k ∈ S and X_i - X_k ∉ S
3        // X_i—X_j—X_k is a potential immorality
4      if X_j ∉ U_{X_i,X_k} then
5        Add the orientations X_i → X_j and X_j ← X_k to K
6    return K
```

Compute PDAG

- Skeleton plus immoralities defines equiv class
- But we might want to orient as many edges as possible, not just those in immoralities

Definition 3.4.11: *Let $\mathcal{G}$ be a DAG. A chain graph $\mathcal{K}$ is a* class PDAG *of the equivalence class of $\mathcal{G}$ if shares the same skeleton as $\mathcal{G}$, and contains a directed edge $X \rightarrow Y$ if and only if all $\mathcal{G}'$ that are I-equivalent to $\mathcal{G}$ contain the edge $X \rightarrow Y$.*[8] ■



Overall PC algorithm

Algorithm 3.5 Procedure for finding the class PDAG that characterizes the P-map of a distribution P.

```
Procedure Build-PDAG (
  X = {X_1, ..., X_n}   // A specification of the random variables
  P   // Distribution of interest
)
1   S, {U_{X_i,X_j}} ← Build-PMap-Skeleton(X, P)
2   K ← Find-Immoralities(X, S, {U_{X_i,X_j}})
3   while not converged
4     Find a subgraph in K matching the left-hand side of a rule R1  R3
5     Replace the subgraph with the right-hand side of the rule
6   return K
```

Theorem 3.4.14: *Let P be a distribution that has a P-map $\mathcal{G}^*$, and let $\mathcal{K}$ be the PDAG returned by* Build-PDAG$(\mathcal{X}, P)$. *Then, $\mathcal{K}$ is a class PDAG of $\mathcal{G}^*$.*

n=#nodes, d=fanin, complexity = O(n^{d+2})
One error in a CI test can propagate through whole structure – not robust
Can choose thresholds to control the FDR

Recent developments

Kalisch, M. and Bühlmann, P. (2007). Estimating high-dimensional directed acyclic graphs with the PC-algorithm. Journal of Machine Learning Research 8, 613-636.
[Proves uniform consistency in the Gaussian case]

Kalisch, M. and Bühlmann, P. (2008). Robustification of the PC-algorithm for directed acyclic graphs. Journal of Computational and Graphical Statistics 17, 773-789.
[Uses robust estimate of covariance matrix]

Maathuis, M.H., Kalisch, M. and Bühlmann, P. (2008). Estimating high-dimensional intervention effects from observational data. To appear in the Annals of Statistics.
[Causal DAGs]

Bühlmann, P., Kalisch, M. and Maathuis, M.H. (2009). Variable selection for high-dimensional models: partially faithful distributions and the PC-simple algorithm.
[Lasso-type methods]

Score functions

- We can treat model selection as an optimization problem: arg max score(G,D)
- ML score: $\text{score}_L(\mathcal{G} : \mathcal{D}) = \ell(\langle \mathcal{G}, \hat{\theta}_{\mathcal{G}} \rangle : \mathcal{D})$
- Obviously this will prefer the fully connected graph
- But if we limit the fan-in (eg restrict attention to simple trees), this can be useful

ML score and Mutual information

- Consider G0: X, Y and G1: X->Y

$$\text{score}_L(\mathcal{G}_0 \ : \ \mathcal{D}) = \sum_m \log \hat{\theta}_{x[m]} + \log \hat{\theta}_{y[m]}$$

$$\text{score}_L(\mathcal{G}_1 \ : \ \mathcal{D}) = \sum_m \log \hat{\theta}_{x[m]} + \log \hat{\theta}_{y[m]|x[m]}$$

$$\text{score}_L(\mathcal{G}_1 \ : \ \mathcal{D}) - \text{score}_L(\mathcal{G}_0 \ : \ \mathcal{D}) = \sum_m \log \hat{\theta}_{y[m]|x[m]} - \log \hat{\theta}_{y[m]}$$

$$\text{score}_L(\mathcal{G}_1 \ : \ \mathcal{D}) - \text{score}_L(\mathcal{G}_0 \ : \ \mathcal{D}) = \sum_{x,y} M[x,y] \log \hat{\theta}_{y|x} - \sum_y M[y] \log \hat{\theta}_y$$

$$\text{score}_L(\mathcal{G}_1 \ : \ \mathcal{D}) - \text{score}_L(\mathcal{G}_0 \ : \ \mathcal{D}) = M \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(y \mid x)}{\hat{P}(y)} = M \cdot \mathbb{I}_{\hat{P}}(X;Y)$$

Proposition 18.3.1: *The likelihood score* decomposes *as follows:*

$$\text{score}_L(\mathcal{G} \ : \ \mathcal{D}) = M \sum_{i=1}^{n} \boldsymbol{I}_{\hat{P}}(X_i; \text{Pa}_{X_i}^{\mathcal{G}}) - M \sum_{i=1}^{n} \mathbb{H}_{\hat{P}}(X_i)$$

Bayesian score

Defined as log marginal likelihood plus log prior
Log p(G) is constant whereas log p(D|G) grows linearly with nsamples
Log p(D|G) offers automatic complexity control – Bayesian Occam's razor

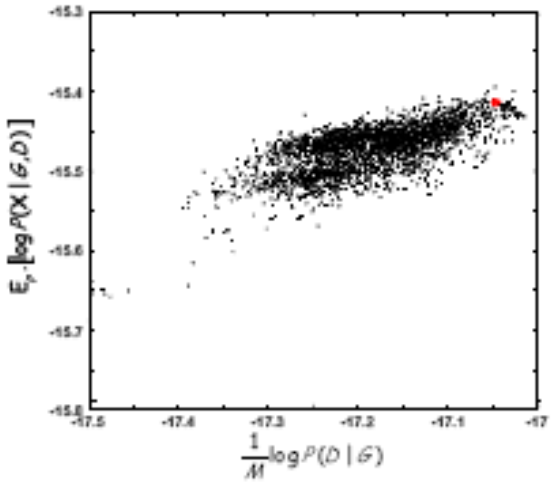
$$\text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

$$P(\mathcal{D} \mid \mathcal{G}) = \int_{\Theta_\mathcal{G}} P(\mathcal{D} \mid \boldsymbol{\theta}_\mathcal{G}, \mathcal{G}) P(\boldsymbol{\theta}_\mathcal{G} \mid \mathcal{G}) d\boldsymbol{\theta}_\mathcal{G}$$

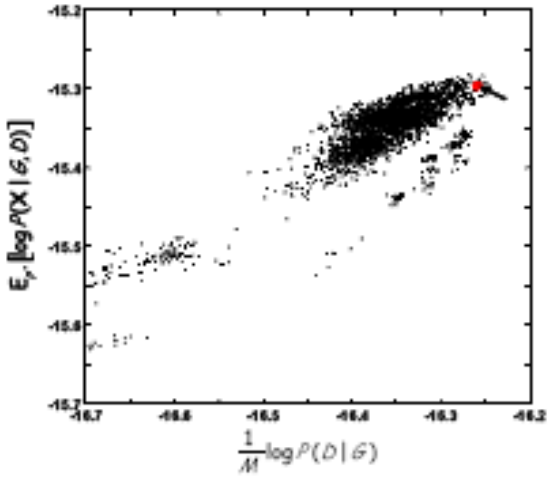
$$P(\mathcal{D} \mid \mathcal{G}) = \prod_{m=1}^{M} P(\xi[m] \mid \xi[1], \ldots, \xi[m-1], \mathcal{G})$$

$$\frac{1}{M} \log P(\mathcal{D} \mid \mathcal{G}) \approx \boldsymbol{E}_{P^*}[\log P(\mathcal{X} \mid \mathcal{G}, \mathcal{D})]$$

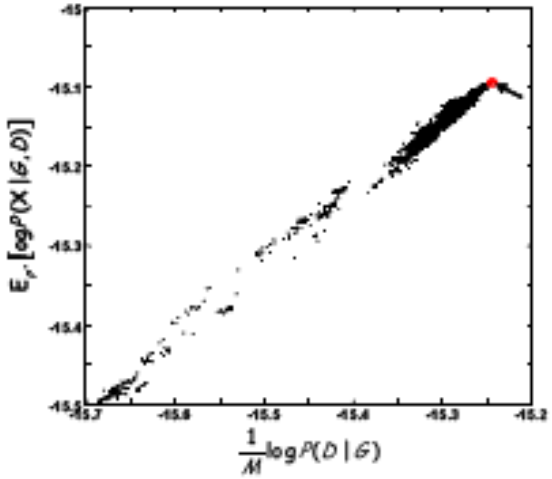
Expected log pred lik vs avg log marg lik



500 instances



1000 instances



10000 instances

Computation of marginal likelihood

- For a Dirichlet-multinomial we have

$$P(x[1],\ldots,x[M]) = \frac{\Gamma(\alpha)}{\Gamma(\alpha+M)} \cdot \prod_{i=1}^{k} \frac{\Gamma(\alpha_i + M[x^i])}{\Gamma(\alpha_i)}.$$

- For a DAG X->Y we have

$$\begin{aligned} P(\mathcal{D} \mid \mathcal{G}_{X\to Y}) &= \left(\int_{\Theta_X} P(\theta_X \mid \mathcal{G}_{X\to Y}) \prod_m P(x[m] \mid \theta_X, \mathcal{G}_{X\to Y}) d\theta_X\right) \\ &\quad \left(\int_{\Theta_{Y|x^0}} P(\theta_{Y|x^0} \mid \mathcal{G}_{X\to Y}) \prod_{m:x[m]=x^0} P(y[m] \mid \theta_{Y|x^0}, \mathcal{G}_{X\to Y}) d\theta_{Y|x^0}\right) \\ &\quad \left(\int_{\Theta_{Y|x^1}} P(\theta_{Y|x^1} \mid \mathcal{G}_{X\to Y}) \prod_{m:x[m]=x^1} P(y[m] \mid \theta_{Y|x^1}, \mathcal{G}_{X\to Y}) d\theta_{Y|x^1}\right) \end{aligned}$$

- For CPTs with dirichlet priors:BDe score

$$P(\mathcal{D} \mid \mathcal{G}) = \prod_i \prod_{\boldsymbol{u}_i \in Val(\mathrm{Pa}_{X_i}^{\mathcal{G}})} \frac{\Gamma(\alpha_{X_i|\boldsymbol{u}_i}^{\mathcal{G}})}{\Gamma(\alpha_{X_i|\boldsymbol{u}_i}^{\mathcal{G}} + M[\boldsymbol{u}_i])} \prod_{x_i^j \in Val(X_i)} \left[\frac{\Gamma(\alpha_{x_i^j|\boldsymbol{u}_i}^{\mathcal{G}} + M[x_i^j, \boldsymbol{u}_i])}{\Gamma(\alpha_{x_i^j|\boldsymbol{u}_i}^{\mathcal{G}})}\right]$$

Asymptotic approximations to Bayesian score

- We have

Theorem 18.3.4: *If we use a Dirichlet parameter prior for all parameters in our network, then, as $M \to \infty$, we have that:*

$$\log P(\mathcal{D} \mid \mathcal{G}) = \ell(\hat{\boldsymbol{\theta}}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}] + O(1)$$

where $\mathrm{Dim}[\mathcal{G}]$ *is the number of independent parameters in* $\mathcal{G}$.

$$\mathrm{score}_{BIC}(\mathcal{G} : \mathcal{D}) = \ell(\hat{\boldsymbol{\theta}}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}]$$

$$\mathrm{score}_{BIC}(\mathcal{G} : \mathcal{D}) = M\sum_{i=1}^{n} \boldsymbol{I}_{\hat{P}}(X_i; \mathrm{Pa}_{X_i}) - M\sum_{i=1}^{n} \boldsymbol{H}_{\hat{P}}(X_i) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}]$$

MDL = BIC

Thm 18.3.6. BIC, MDL and Bayesian score are consistent (so score(G)=score(G*) iff G is I-equivlent to G*)

Structure priors

- P(G) only matters in small sample setting

- Penalized number of edges $P(\mathcal{G}) \propto c^{|\mathcal{G}|}$
- Penalize deviation from fixed prior structure

Decomposable score

- When we make local changes to a graph, we want to evaluate the score change in constant time

Definition 18.3.8: *A structure score function* score *is* decomposable *if the score of a structure $\mathcal{G}$ can be written as*

$$\text{score}(\mathcal{G} : \mathcal{D}) = \sum_i \text{FamScore}(X_i \mid \text{Pa}_i^{\mathcal{G}} : \mathcal{D})$$

- BIC score is decomposable

Definition 18.3.9: *Let $\{P(\boldsymbol{\theta}_{\mathcal{G}} \mid \mathcal{G}) : \mathcal{G} \in \mathbf{G}\}$ be a set of parameter priors that satisfy global parameter independence. The prior satisfies* Parameter modularity *if for each $\mathcal{G}, \mathcal{G}'$ such that $\text{Pa}_{X_i}^{\mathcal{G}} = \text{Pa}_{X_i}^{\mathcal{G}'} = \boldsymbol{U}$, then $P(\boldsymbol{\theta}_{X_i|\boldsymbol{U}} \mid \mathcal{G}) = P(\boldsymbol{\theta}_{X_i|\boldsymbol{U}} \mid \mathcal{G}')$.* ■

- Thm 18.3.10. parameter modularity => BDe score is decomposable
- Defn: Structural modularity if p(G) decomposes
- Thm 18.3.10. param & struct modularity => Bayesian score decomposable

Score equivalence

- Def 18.3.11. Score() is score equiv if score(G)=score(G') if G, G' are I-equiv
- Thm 18.3.12. Likelihood and BIC scores are score equiv.
- BDe score is only score equivalent if we set the Dirichlet hyper-parameters as follows

$$\alpha_{x_i|\mathrm{pa}_i} = \alpha \cdot P'(x_i, \mathrm{pa}_i).$$

- Eg if P' is a uniform prior network, then

$$\begin{aligned} \theta_{ijk} &\stackrel{\text{def}}{=} p(X_i = k | X_{\pi_i} = j) \\ \theta_{ijk} &\sim \mathrm{Dir}(\alpha_{ijk}) \\ \alpha_{ijk} &= \alpha \frac{1}{q_i r_i} \end{aligned}$$

$\alpha_{\iota\varphi\kappa}$=1 (K2 prior) is not score equiv

thetaY ~ Dir(1,1)
thetaY|X=1 ~ Dir(1,1)
thetaY|X=0 ~ Dir(1,1)

Decomposable score

- When we make local changes to a graph, we want to evaluate the score change in constant time

Definition 18.3.8: *A structure score function* score *is* decomposable *if the score of a structure $\mathcal{G}$ can be written as*

$$\text{score}(\mathcal{G} : \mathcal{D}) = \sum_i \text{FamScore}(X_i \mid \text{Pa}_i^{\mathcal{G}} : \mathcal{D})$$

- BIC score is decomposable
- We say a prior satisfies structural modularity if

$$P(\mathcal{G}) \propto \prod_i P(\text{Pa}_{X_i} = \text{Pa}_{X_i}^{\mathcal{G}})$$

Definition 18.3.9: *Let $\{P(\boldsymbol{\theta}_{\mathcal{G}} \mid \mathcal{G}) : \mathcal{G} \in \mathcal{G}\}$ be a set of parameter priors that satisfy global parameter independence. The prior satisfies* Parameter modularity *if for each $\mathcal{G}, \mathcal{G}'$ such that $\text{Pa}_{X_i}^{\mathcal{G}} = \text{Pa}_{X_i}^{\mathcal{G}'} = \boldsymbol{U}$, then $P(\boldsymbol{\theta}_{X_i|\boldsymbol{U}} \mid \mathcal{G}) = P(\boldsymbol{\theta}_{X_i|\boldsymbol{U}} \mid \mathcal{G}')$.* ■

- Thm 18.3.10. Structural & parameter modularity => Bayesian score is decomposable

Stat 521A
Lecture 22

Outline

- Algorithms for finding MAP structure (18.4)
- MCMC over DAG structure (18.5)
- Dynamic programming + MH
- Stochastic search

Computationally intractable

- There are $O(d!2^{\binom{d}{2}})$DAGs on d nodes

d	2	3	4	5	6	7	8	9
#G(d)	3	25	543	29,281	3,781,503	1.1e9	7.8e11	1.2e15

Trees

- Can learn optimal tree using MST algo in $O(n^2 \log n + n^2 M)$ time, n=#num nodes, M=#cases

Definition 18.4.1: *A network structure $\mathcal{G}$ is called* tree-structured *if each variable X has at most one parent in $\mathcal{G}$, i.e., $|\mathrm{Pa}_X^{\mathcal{G}}| \leq 1$.* ■

$$\Delta(\mathcal{G}) = \mathrm{score}(\mathcal{G} : \mathcal{D}) - \mathrm{score}(\mathcal{G}_\emptyset : \mathcal{D})$$

$$\Delta(\mathcal{G}) = \sum_{i, \mathrm{Pa}_i^{\mathcal{G}} \neq \emptyset} \left(\mathrm{FamScore}(X_i \mid \mathrm{Pa}_i^{\mathcal{G}} : \mathcal{D}) - \mathrm{FamScore}(X_i : \mathcal{D})\right)$$

$$w_{X_j \to X_i} = \mathrm{FamScore}(X_i \mid X_j : \mathcal{D}) - \mathrm{FamScore}(X_i : \mathcal{D})$$

$$\Delta(\mathcal{G}) = \sum_{X_j \to X_i \in \mathcal{G}} w_{X_j \to X_i}$$

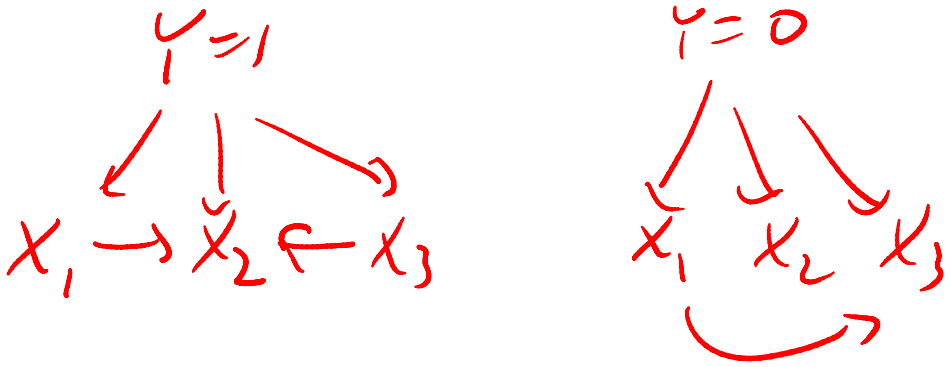
Undirected max weight spanning tree

Score equivalence => $w_{X_i \to X_j} = w_{X_j \to X_i}$.

$$\mathrm{score}_L(\mathcal{G}_1 : \mathcal{D}) - \mathrm{score}_L(\mathcal{G}_0 : \mathcal{D}) = M \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(y \mid x)}{\hat{P}(y)} = M \cdot I_{\hat{P}}(X;Y)$$

TAN classifiers

- Tree-augmented naïve Bayes
- Can learn tree structure for each class conditional density



Mixtures of trees

- We can fit mixtures of trees using EM: just run MST algorithm in M step
- Analogous to mixture of diagonal Gaussians

DAG with known order

- Can find optimal set of parents for each node independently

Proposition 18.4.2: *Let $\prec$ be an ordering over $\mathcal{X}$, and let* $\text{score}(\mathcal{G} : \mathcal{D})$ *be a decomposable score. If we choose $\mathcal{G}$ to be the network where*

$$\text{Pa}_i^{\mathcal{G}} = \arg \max_{\boldsymbol{U} \subseteq \{X_j : X_j \prec X_i\}} \text{FamScore}(X_i \mid \boldsymbol{U}_i : \mathcal{D})$$

for each i, then $\mathcal{G}$ maximizes the score among the structures consistent with $\prec$.

- If at most d parents, last node Xn must select from

$$X_n \text{ is } 1 + \binom{n-1}{2} + \ldots + \binom{n-1}{d} = O(d\binom{n-1}{d})$$

- If CPDs are GLMs, can use lasso to find parents
- If order unknown, can search over orders.

Dependency networks

- A depnet is a set of full conditionals p(Xi|X(-i)) learned independently.
- There may not be any joint which is consistent with these conditionals.
- However, one can define a (non-unique) joint by using an ordered Gibbs sampler.
- If the conditionals are learned from (lots of) data, they are likely to be consistent.
- By performing variable selection at each node independently, we get a sparse graph.
- Provides a fast way to visualize dependencies.

Collaborative filtering

- One succesful application of depnets is CF.
- Xi=1 if item i has been bought, Xi=0 otherwise
- Assume S=set of bought items, Sbar = not bought items, i = target item. Compute p(Xi|S=1,Sbar=0).
- In a depnet, this is a simple lookup – all other nodes are observed.
- In a DGM, this is also fairly simple – product of CPDs in the Markov blanket.
- Both techniques have similar predictive accuracy, but depnet is much faster to learn.
- Ships in Microsoft's ecommerce package.

DAG with unknown order

- Thm 18.4.3 It is NP-hard to find the optimal DAG with d >= 2 parents.
- Standard approach: heuristic local search (eg hill climbing), using add/ delete/ reverse edge (n^2 neighbors to each DAG).
- Diameter of space is $O(n^2)$: to get from G1 to G2, delete all edges of G1 then add all edges of G2.
- If too many neighbors, use first best instead of evaluating all of them.
- Often there will be large plateaus of I-equivalent DAGs. Can use tabu search to escape these.
- Multiple restarts or data perturbation.

Data perturbation

- Can be used to escape local minima for many ML algorithms, where score = sum_i score(D_i)
- Idea: use weights w_i, and perturb them at random (or more cleverly – rather like boosting)

```
1    G ← Search(G_∅, D, score, O)
2    G_best ← G
3    t ← t_0
4    for i = 1, ... until convergence
5      D' ← Perturb(D, t)
6      G ← Search(G, D', score, O)
7      if score(G : D) > score(G_best : D) then
8        G_best ← G
9      t ← γ · t
10
11   return G_best
```

Efficient scoring of proposed new graph

$$\delta(\mathcal{G} : o) = \text{score}(o(\mathcal{G}) : \mathcal{D}) - \text{score}(\mathcal{G} : \mathcal{D})$$

to be the change of score associated with applying o on $\mathcal{G}$. Using score decomposition, we can compute this quantity relatively efficiently.

Proposition 18.4.4: *Let $\mathcal{G}$ be a network structure, and* score *be a decomposable score.*

- *If o is "Add $X \rightarrow Y$", and $X \rightarrow Y \notin \mathcal{G}$, then*

$$\delta(\mathcal{G} : o) = \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} \cup \{X\} : \mathcal{D}) - \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} : \mathcal{D})$$

- *If o is "Delete $X \rightarrow Y$" and $X \rightarrow Y \in \mathcal{G}$, then*

$$\delta(\mathcal{G} : o) = \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} - \{X\} : \mathcal{D}) - \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} : \mathcal{D})$$

- *If o is "Reverse $X \rightarrow Y$" and $X \rightarrow Y \in \mathcal{G}$, then*

$$\begin{aligned}\delta(\mathcal{G} : o) = {} & \text{FamScore}(X, \text{Pa}_X^{\mathcal{G}} \cup \{Y\} : \mathcal{D}) + \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} - \{X\} : \mathcal{D}) \\ & - \text{FamScore}(X, \text{Pa}_X^{\mathcal{G}} : \mathcal{D}) - \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} : \mathcal{D})\end{aligned}$$

Efficient update of cached scores

- After accepting change, only have to update scores of affected families - O(n) operators

Proposition 18.4.5: *Let $\mathcal{G}$ and $\mathcal{G}'$ be two network structures, and* score *be a decomposable score.*

- *If o is either "Add $X \rightarrow Y$" or "Delete $X \rightarrow Y$" and $\mathrm{Pa}_Y^{\mathcal{G}} = \mathrm{Pa}_Y^{\mathcal{G}'}$, then $\delta(\mathcal{G} : o) = \delta(\mathcal{G}' : o)$.*

- *If o is "Reverse $X \rightarrow Y$", $\mathrm{Pa}_Y^{\mathcal{G}} = \mathrm{Pa}_Y^{\mathcal{G}'}$, and $\mathrm{Pa}_X^{\mathcal{G}} = \mathrm{Pa}_X^{\mathcal{G}'}$, then $\delta(\mathcal{G} : o) = \delta(\mathcal{G}' : o)$.*

Sufficient statistics

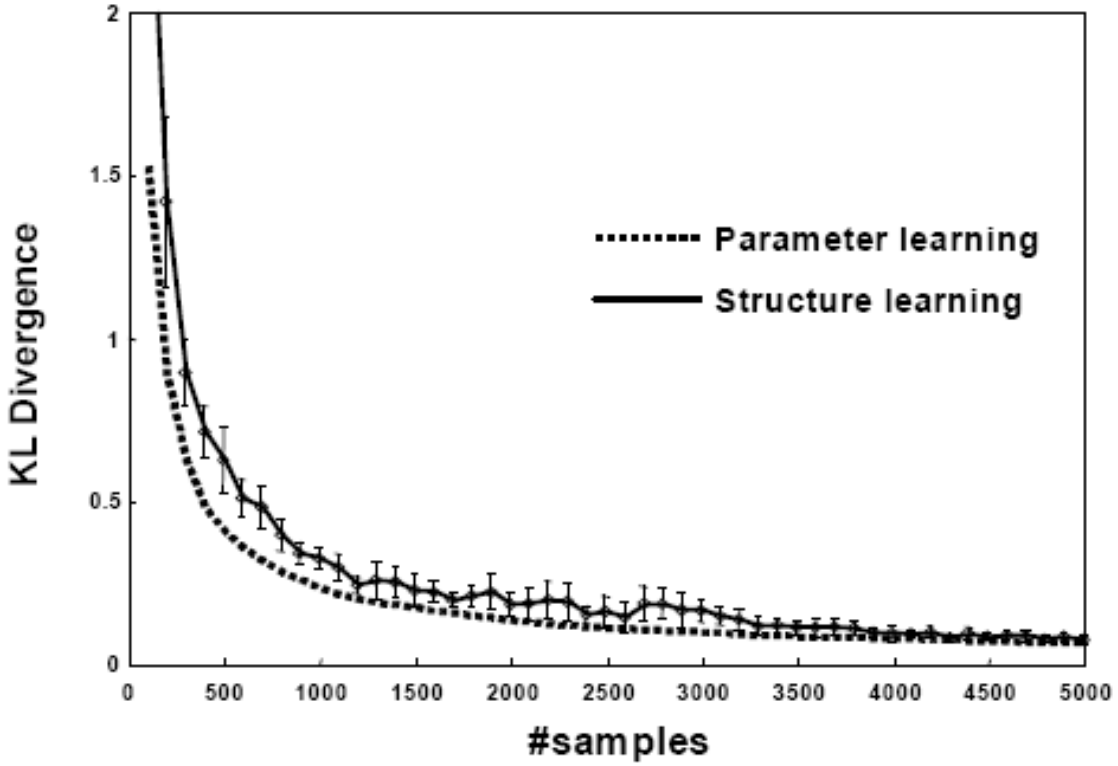
- Need to walk over M rows for all the columns in a given family
- If we need to update n operators, this is O(nM) time
- Can use AD-trees for discrete data, or KD-trees for cts data, to do this more efficiently (possibly subject to approximation error)

Heaps

- Need to search over O(n^2) operators to find best at each step
- Can use a heap to find the best in O(1) time if we do O(n log n) time to update it when scores change

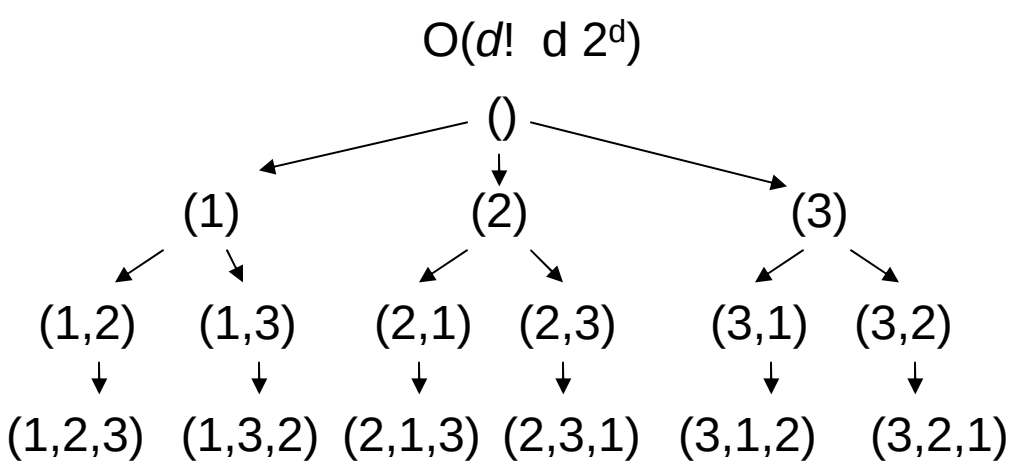
Learning params vs structure

- ICU Alarm network

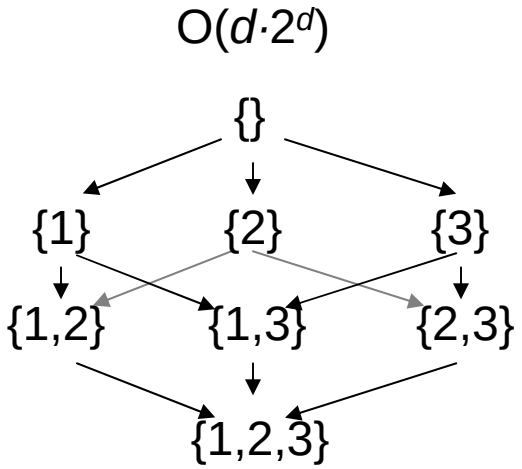


Dynamic programming (DP)

- Can analytically marginalize over *d*! orderings and all possible subsets in O(d 2^d) time/ space using DP and fast Mobius transform
- Since order of parents does not matter, eg $p(X_1|X_2,X_3) == p(X_1|X_3,X_2)$, we can share work
- Can find exact global MAP DAG



Ordered permutation tree



Unordered permutation lattice

Equivalence classes

- Can search through PDAG space - smaller than DAG space, and fewer (if any) plateau
- To evaluate score of a PDAG, convert to a DAG then use score for DAG
- To find neighbors: convert PDAG to DAG, add or delete edge; this changes skeleton hence moves to a new PDAG
- Greedy Equivalence Search: start with empty PDAG, add best edge until local max, then delete best edge till local max. If M->infty, this will provably find optimal PDAG given any consistent scoring fn.
- Performing local updates to score of a PDAG is harder.

Bayes model averaging

- When the sample size is small, the posterior p(G|D) gives support to multiple (non equivalent) models
- We should perform BMA when performing prediction

$$P(\xi[M+1] \mid \mathcal{D}) = \sum_{\mathcal{G}} P(\xi[M+1] \mid \mathcal{D}, \mathcal{G}) P(\mathcal{G} \mid \mathcal{D})$$

- And when computing E[f(G)|D], where f(G) is some feature, eg f(G)=there is an edge X->Y in G, average path length in G

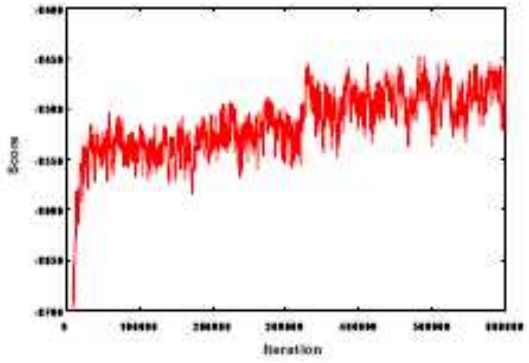
$$\mathbb{E}_{P(\mathcal{G}|\mathcal{D})}[f(\mathcal{G})] = \sum_{\mathcal{G}} f(\mathcal{G}) P(\mathcal{G} \mid \mathcal{D}).$$

MC3

- Markov Chain Monte Carlo Model Composition
- Use MH in space of DAGs, with proposal = uniform over neighbors (add/delete/reverse edge)
- Does not mix well in more than ~10 dimensions. Also, posterior gets more peaky as sample size increases (can use parallel tempering).

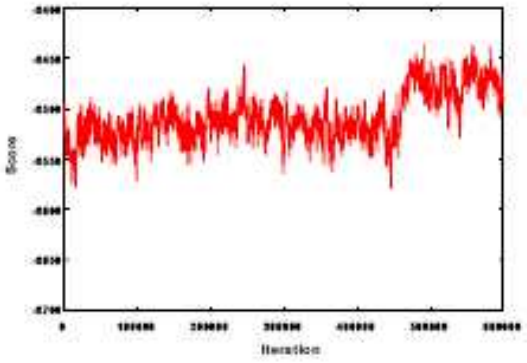
MH on Alarm

Init empty

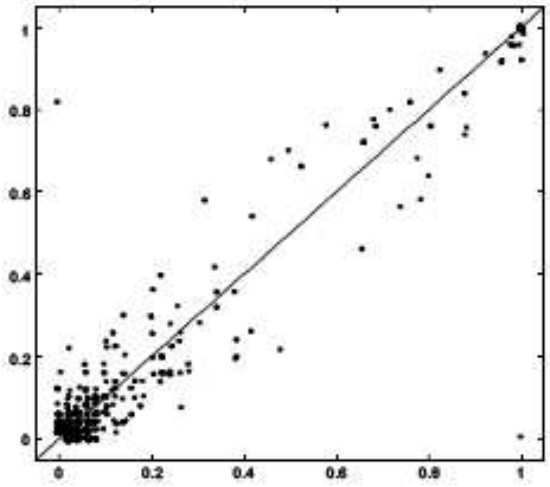


(a)

Init local search

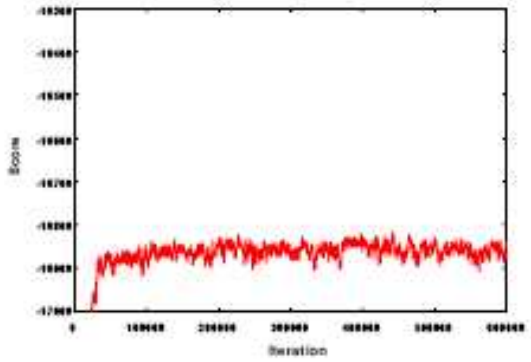


(b)

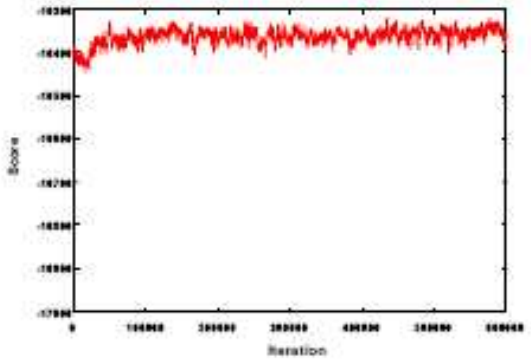


(c)

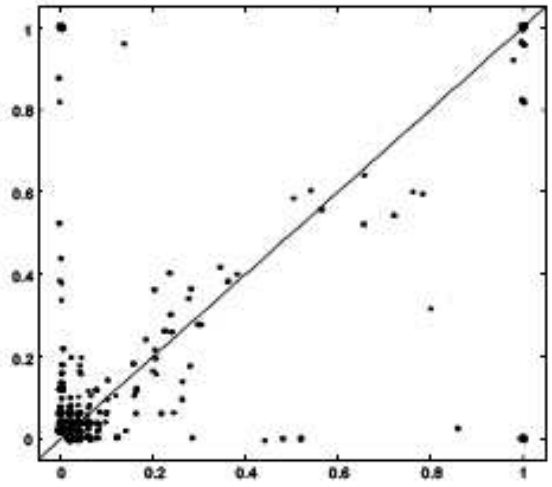
N=500



(a)



(b)



Edge marginals

N=1000

MH in order space

- Given a known order, we can integrate over all possible graphs by summing over all parents sets
- Hence use MH to sample over orders, using traveling-salesman like moves

$$(X_{i_1} \dots X_{i_j} \dots X_{i_d} \dots X_{i_n}) \mapsto (X_{i_1} \dots X_{i_d} \dots X_{i_j} \dots X_{i_n}).$$

$$\min\left[1, \frac{P(\prec', \mathcal{D}) T^{\mathcal{Q}}(\prec' \to \prec)}{P(\prec, \mathcal{D}) T^{\mathcal{Q}}(\mathcal{G} \to \prec')}\right]$$

Target distribution

$$\begin{aligned} P(D \mid \prec) &= \sum_{\mathcal{G} \in \mathcal{G}_{d,\prec}} \prod_i \exp\{\text{FamScore}_B(X_i \mid \text{Pa}_{X_i}^{\mathcal{G}} : \mathcal{D})\} \\ &= \prod_i \sum_{\boldsymbol{U}_i \in \mathcal{U}_{i,\prec}} \exp\{\text{FamScore}_B(X_i \mid \boldsymbol{U}_i : \mathcal{D})\} \end{aligned}$$

$$\mathcal{U}_{i,\prec} = \{\boldsymbol{U} : \boldsymbol{U} \prec X_i, |\boldsymbol{U}| \leq k\}.$$

Posterior features

- Given samples from p(<|D) we compute

$$P(f \mid \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^{T} P(f \mid \mathcal{D}, \prec_t).$$

- Parent features

Proposition 18.5.1:

$$P(\mathrm{Pa}_{X_i}^{\mathcal{G}} = \boldsymbol{U} \mid D, \prec) = \frac{\exp\{\mathrm{FamScore}_B(X_i \mid \boldsymbol{U} : \mathcal{D})\}}{\sum_{\boldsymbol{U}' \in \mathcal{U}_{i,\prec}} \exp\{\mathrm{FamScore}_B(X_i \mid \boldsymbol{U}' : \mathcal{D})\}}.$$

- Edge features

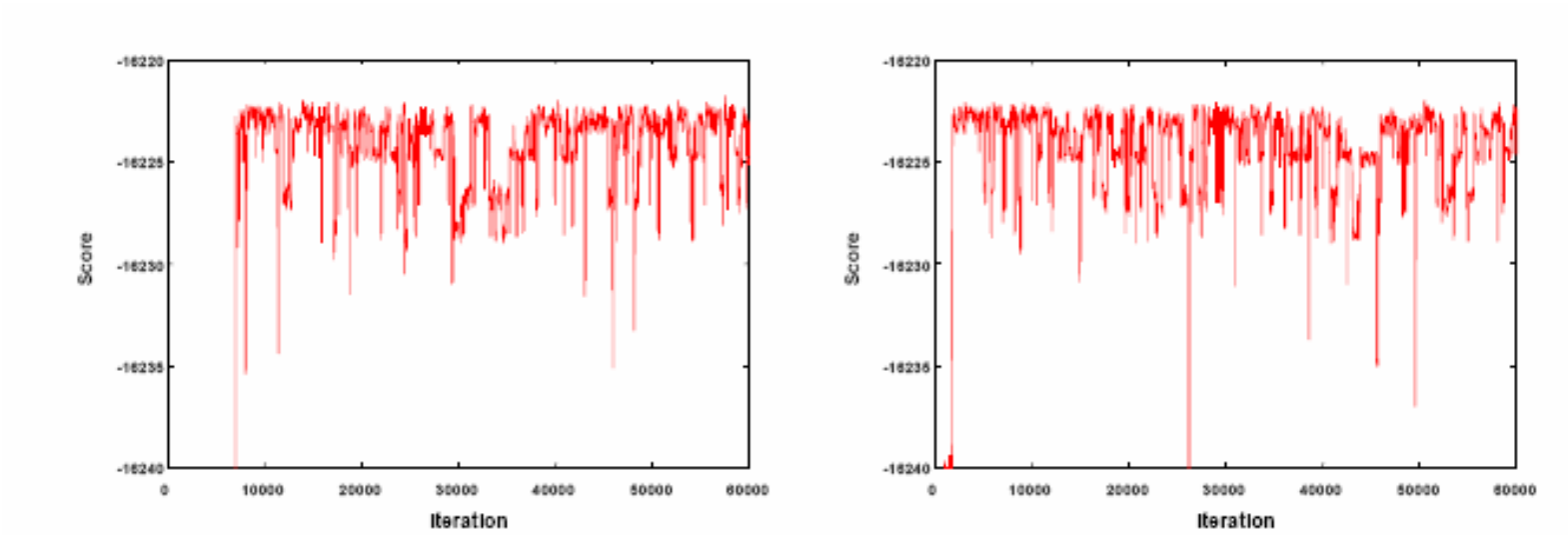
Proposition 18.5.2:

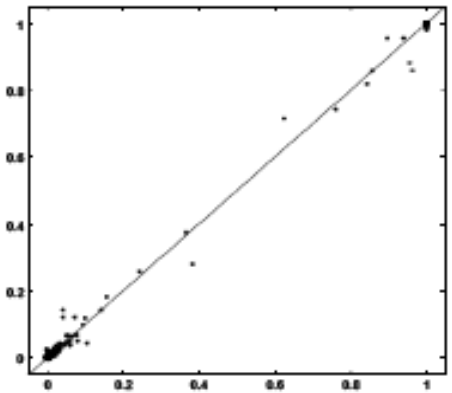
$$P(X_j \in \mathrm{Pa}_{X_i}^{\mathcal{G}} \mid \prec, \mathcal{D}) = \frac{\sum_{\{\boldsymbol{U} \in \mathcal{U}_{i,\prec} \,:\, X_j \in \boldsymbol{U}\}} \exp\{\mathrm{FamScore}_B(X_i \mid \boldsymbol{U} : \mathcal{D})\}}{\sum_{\boldsymbol{U} \in \mathcal{U}_{i,\prec}} \exp\{\mathrm{FamScore}_B(X_i \mid \boldsymbol{U} : \mathcal{D})\}}$$

- General features: sample G given <, then use

$$P(f \mid \prec, \mathcal{D}) = \frac{P(f, \mathcal{D} \mid \prec)}{P(\mathcal{D} \mid \prec)}. \qquad P(f, \mathcal{D} \mid \prec) = \sum_{\mathcal{G} \in \boldsymbol{\mathcal{G}}_{d,\prec}} f(\mathcal{G}) P(\mathcal{G} \mid \prec) P(\mathcal{D} \mid \mathcal{G})$$

RB MH on Alarm





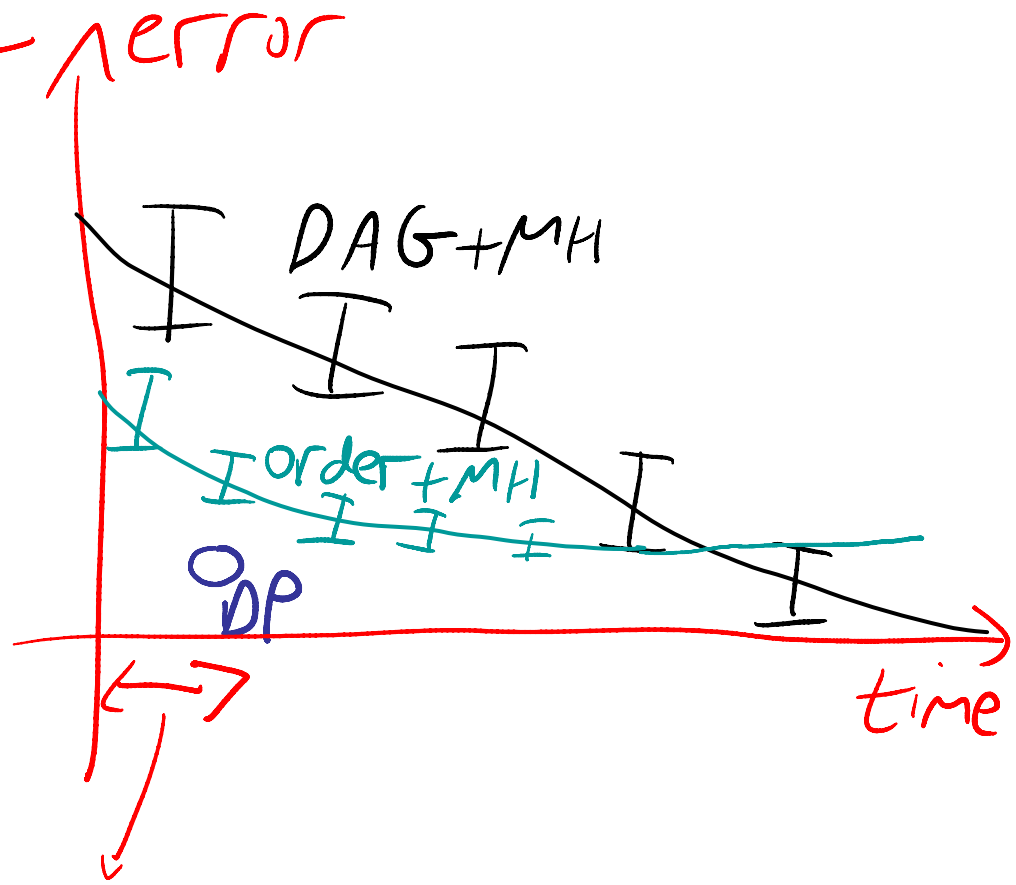
(c)

Dynamic programming

- Koivisto & Sood showed how to compute all edge marginals p(Gij=1|D) exactly in O(n 2^n) time
- Requires special (“modular”) prior p(G) which can be unnatural (see later)

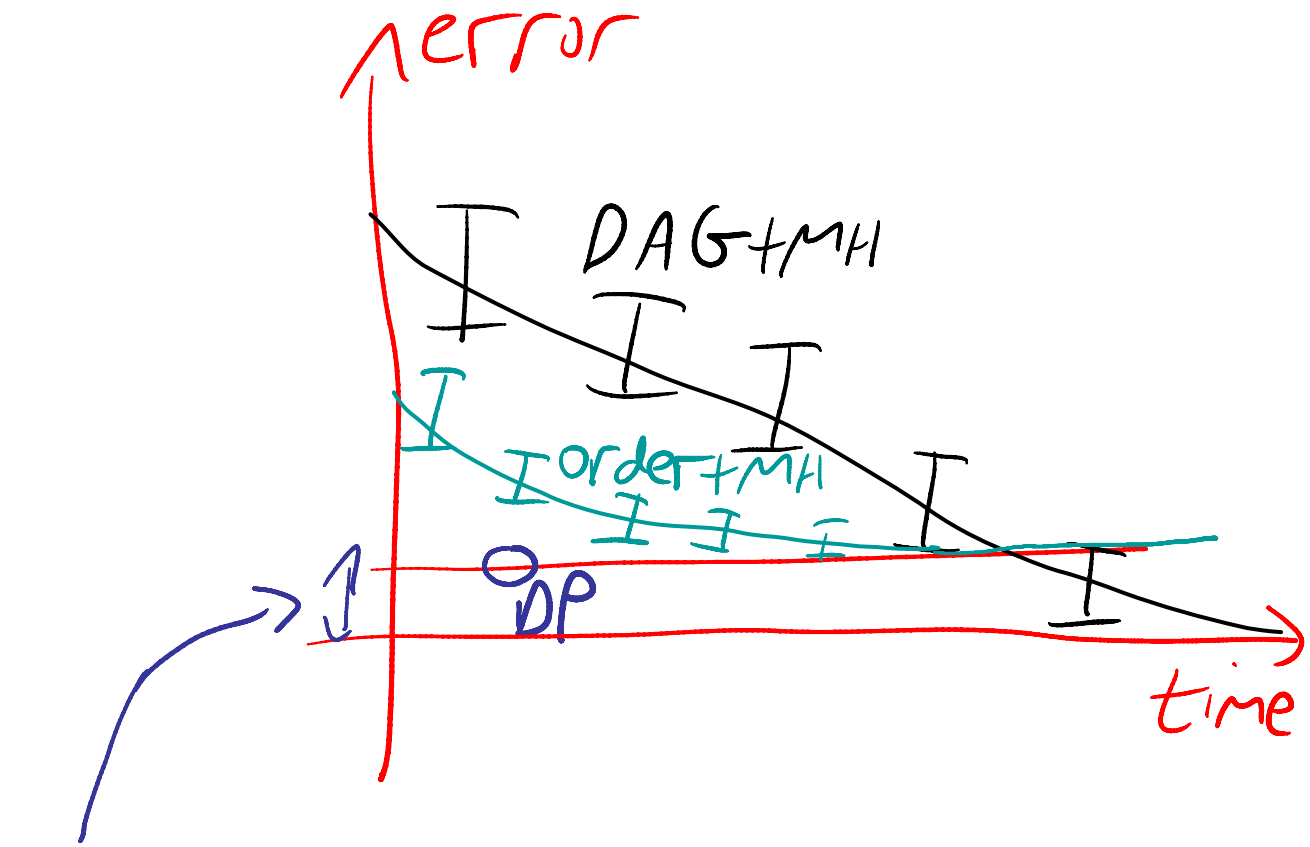
Comparison of approaches

$$\sum_{ij} |\hat{P}(G_{ij} = 1|D) - P(G_{ij} = 1|D)|$$



5 sec (d=10) to 5 mins (d=20)

Error floor due to wrong p(G)



Error due to wrong p(G)

p(G) needed by DP and MH+order

- Joint ("modular") prior on G and $\prec$

$$p(G, \prec) = \frac{1}{Z} \prod_{i=1}^{d} \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})$$

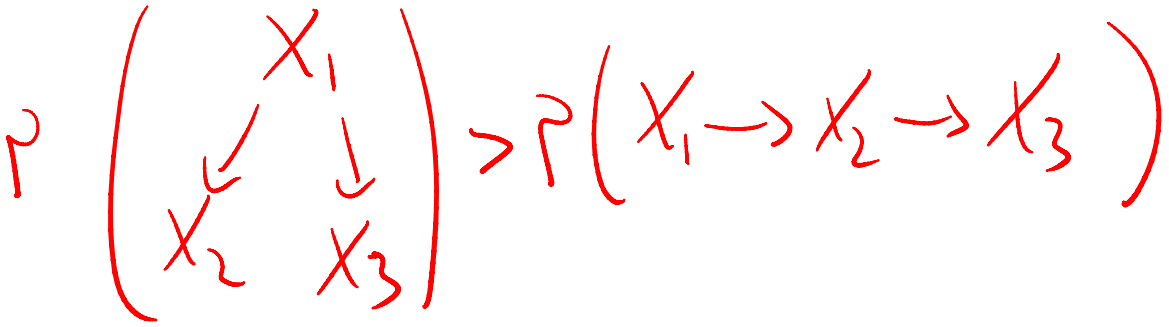
unordered set of parents

ordering of predecessors

- Induced prior on p(G)

$$p(G) = \sum_{\prec} p(G, \prec)$$

Graphs consistent with more orderings are more probable



- Effect will not get erased even with infinite data, since both models are likelihood equivalent

Problems with induced p(G)

- Prior is highly non-uniform
- Effect will not get erased even with infinite data
- Cannot encode arbitrary prior knowledge in p(G)

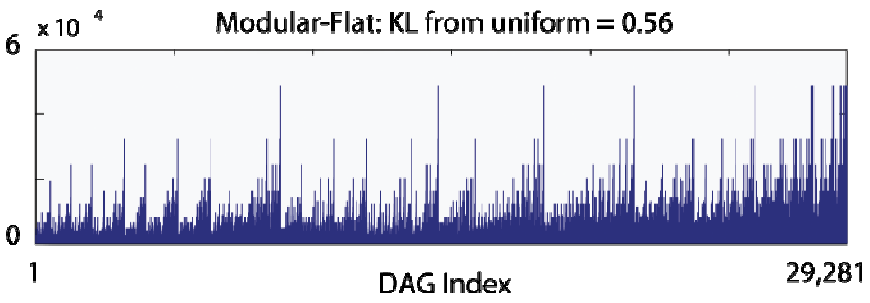
$$p(G) = \sum_{\prec} \frac{1}{Z} \prod_{i=1}^{d} \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})$$

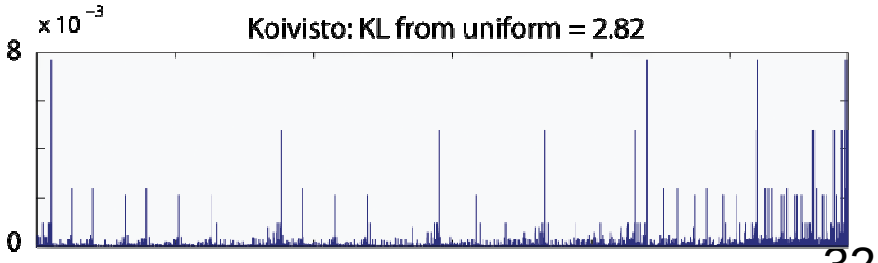
unordered set of parents

ordering of predecessors, $q_i \propto 1$

$$\rho_i(G_i) = 1$$

$$\rho_i(G_i) = \binom{d-1}{|G_i|}^{-1}$$





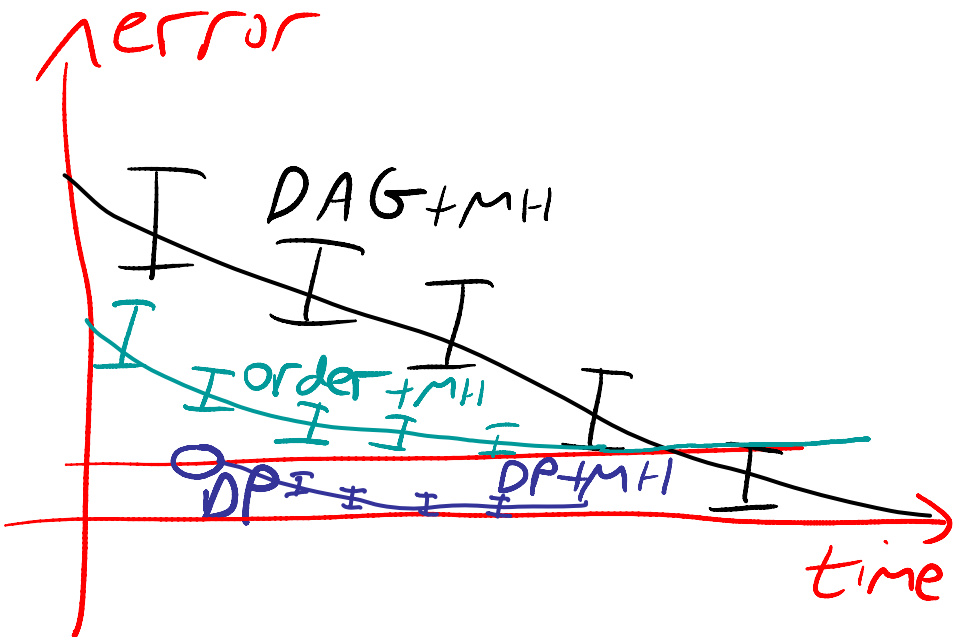
Solutions to p(G) problem

- Importance sampling -- Ellis & Wong ’06
  - Use MH+order as proposal
  - #P-hard to compute exact IS weights

$$
\begin{aligned}
w(G) &= \frac{p^*(G)}{p(G)} = \frac{\frac{1}{Z}\prod_i \rho_i^*(G_i)}{\sum_{\prec} \frac{1}{Z}\prod_{i=1}^{d} \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})} \\
&= \frac{1}{\sum_{\prec} I(\text{consistent}(\prec, G))}
\end{aligned}
$$

Solutions to p(G) problem

- Importance sampling -- Ellis & Wong ’06
- Metropolis Hastings -- this paper
  - Use DP marginals as proposal for MH



MH with DP+local proposal

- Compute $p_{ij}=p(G_{ij}=1|D)$ offline using DP
- wp β, we use a standard local move
- wp 1-β, sample a new graph ~ p_{ij}
- If β=0 (global) independence sampler
- If β=1 (local) standard proposal

Why MH?

- DP alone has 3 problems
  1. Modular prior p(G)
  2. Cannot compute prob. of “long range” features (e.g., path from i to j), only edge features.
  3. Very slow to compute predictive density
     $p(x|D) = \sum_G p(x|G)\, p(G|D)$

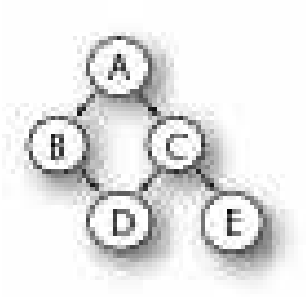
MH allows any $p(G)$

- Propose using $q(G'|G)$
- Accept wp α

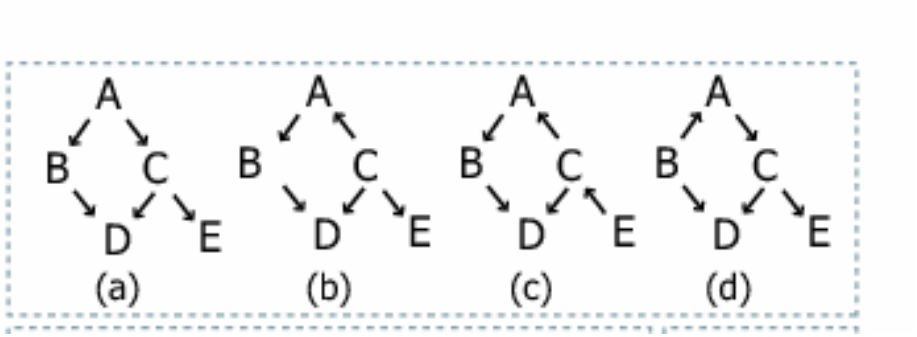
$$\alpha = \min\left(1, \frac{p(D|G')p(G')}{p(D|G)p(G)} \frac{q(G|G')}{q(G'|G)}\right)$$

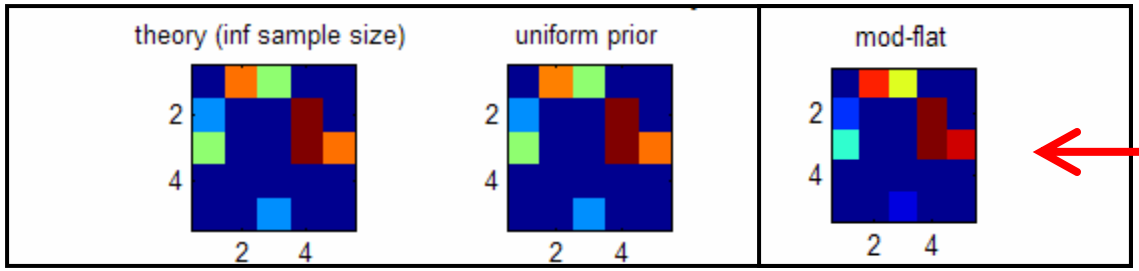
Modular vs uniform p(G)

5 node "cancer" network

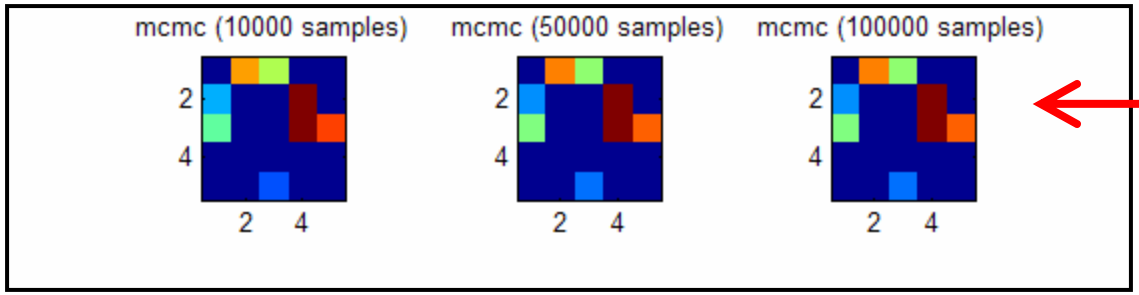


Markov equivalence class





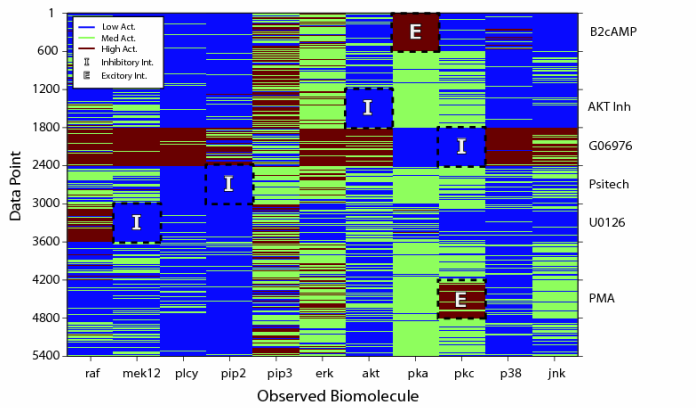
Modular prior biases posterior even as $|D| \rightarrow \infty$



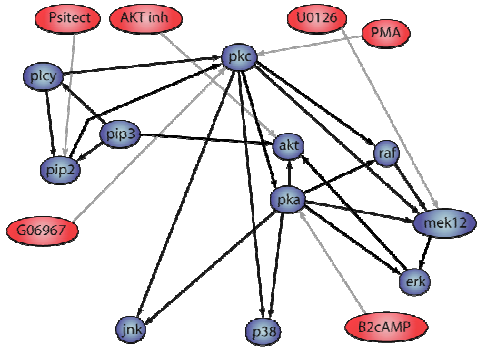
MH fixes bias

T-cell signaling network

Protein phosphorylation (d=11, N=5400)

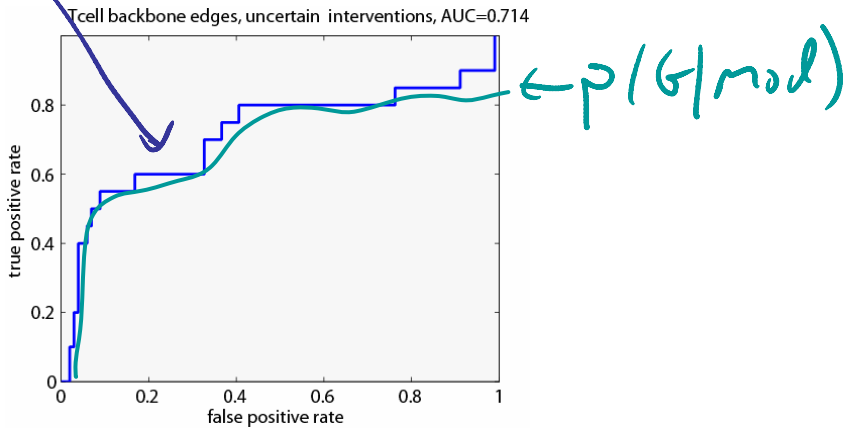


Ground truth DAG



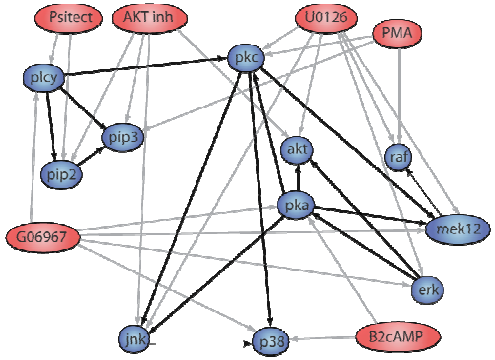
p(G|mij)

ROC



←p(G|mod)

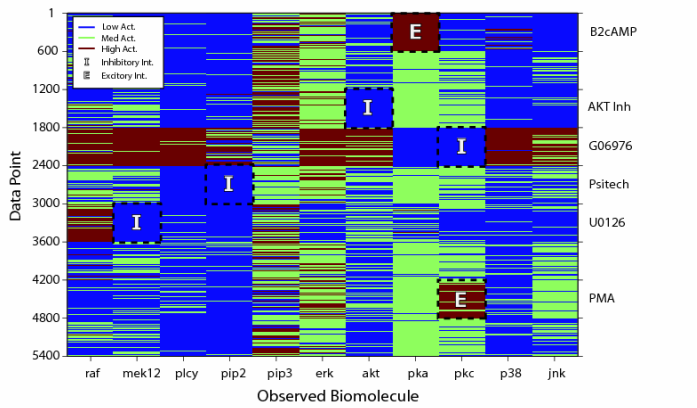
Exact $P(G_{ij}=1|D)$



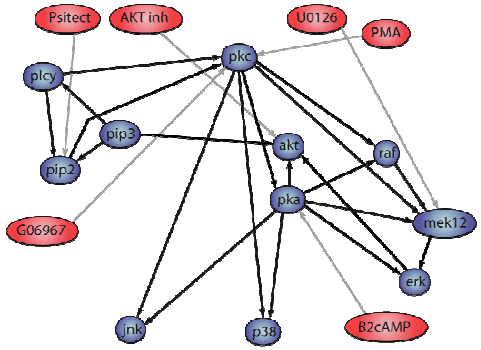
"Causal Protein-Signaling Networks derived from Multiparameter Single-Cell Data", Sachs, Perez, Pe'er, Lauffenberger, Nolan, Science 2005

Informative p(G)

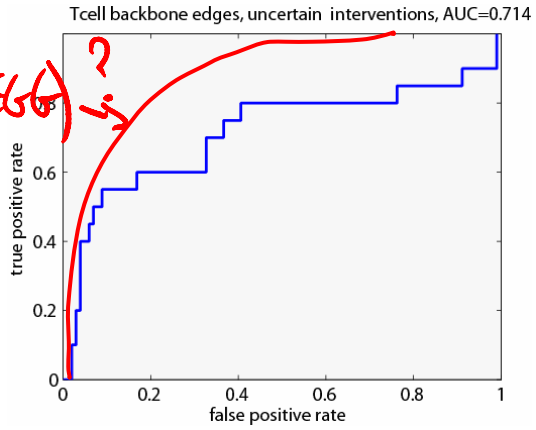
Protein phosphorylation (d=11, N=5400)



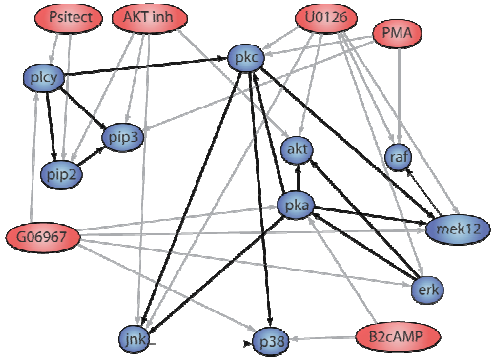
Ground truth DAG



ROC



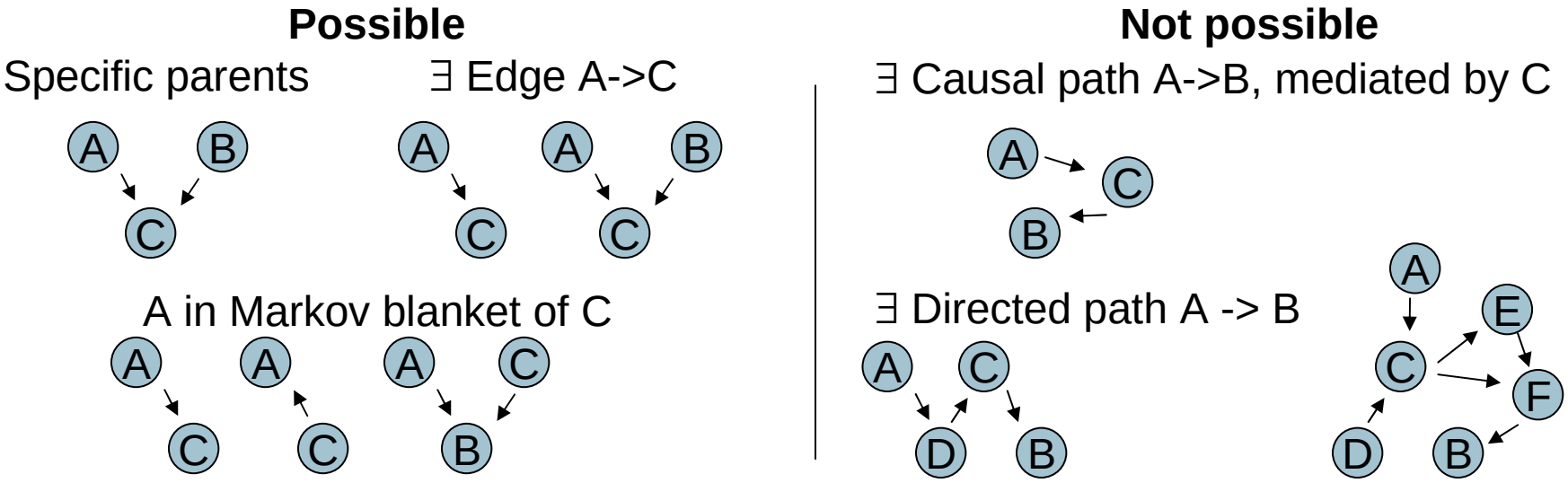
$P(G_{ij}=1|D)$



"Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge", Werhli & Husmeier, 2007

Sampling G allows any features

- DP can only compute posterior of features that are functions of a local family topology

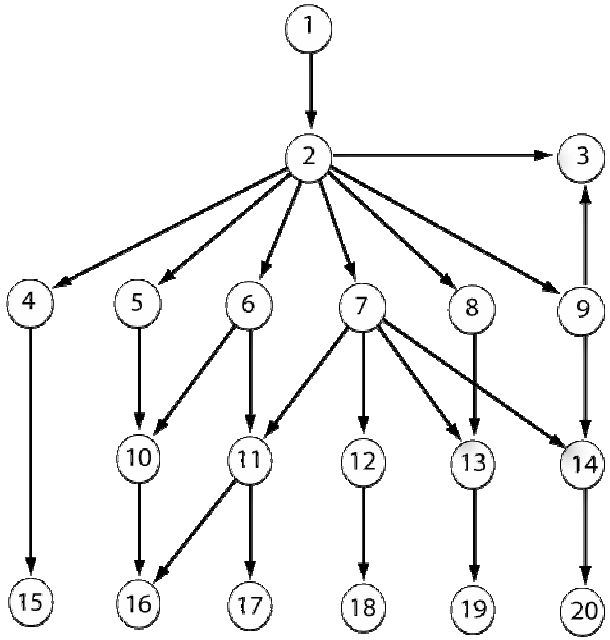


- By sampling DAGs, we can compute E[f(G)] for arbitrary features f

Posterior features

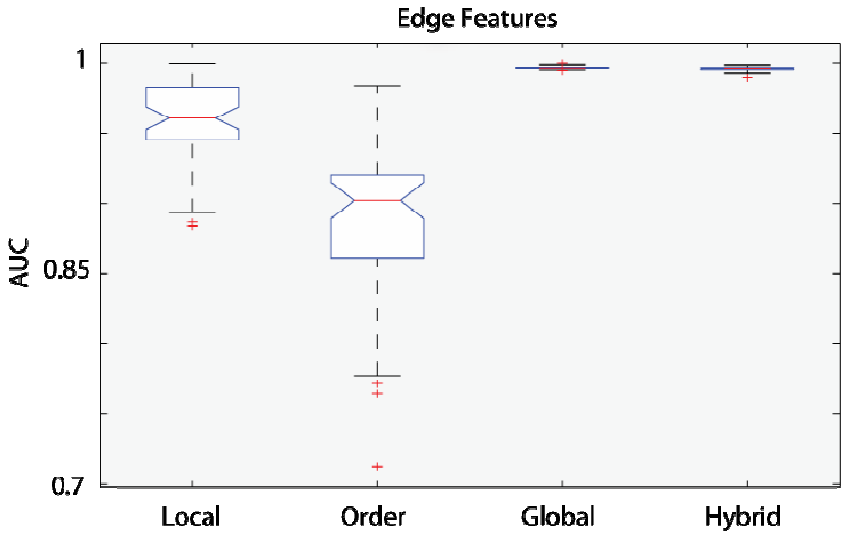
- We sampled N=10k data from d=20 node graph with random CPTs
- Compute p(edge i->j|D) and p(path i->j|D)

"child" network

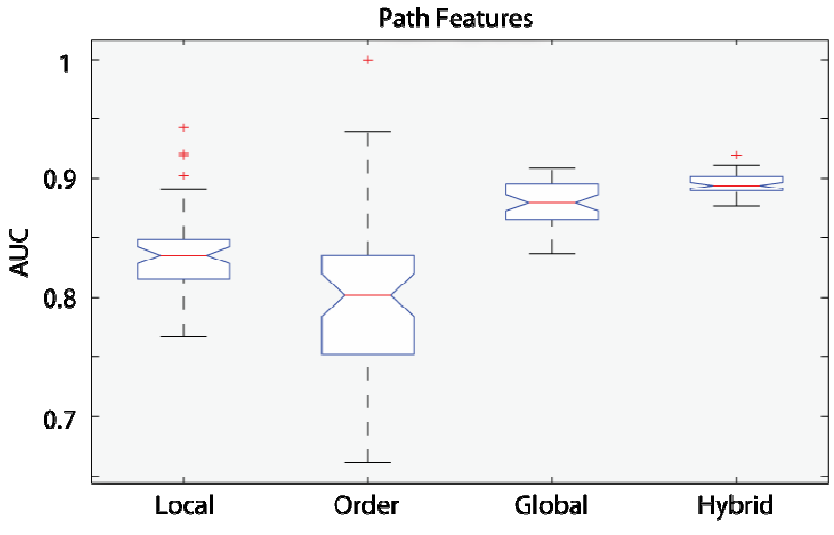


AUC for p(feature=1|D)

Area under the ROC curve after 200 seconds of wall clock time*



β=1 β=0 β=0.1



β=1 β=0 β=0.1

All algorithms were implemented in Matlab/C and run on a standard desktop

Sampling G allows fast prediction

- DP can compute the marginal likelihood of data p(D)
- Hence can compute the predictive likelihood of a test point x:

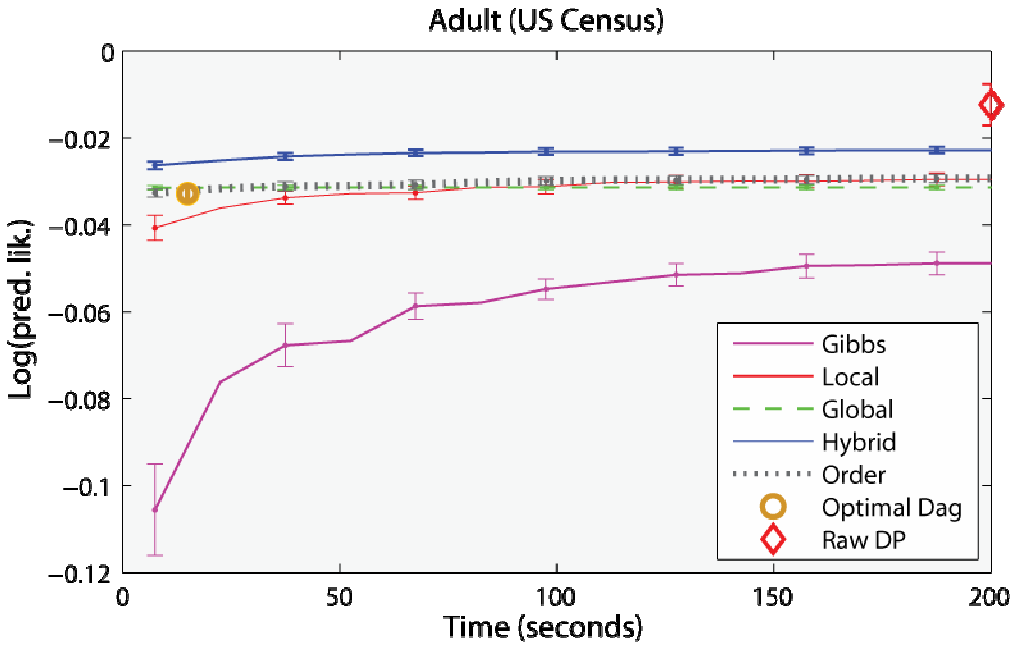
$$p(x|D) = \frac{p(x,D)}{p(D)}$$

- Since DP integrates out G, we have to keep D, and re-run algorithm for each x, which is very slow

- Our approach: keep a sample of $G^s \sim p(G|D)$ and compute posterior mean parameters $\overline{\theta}^s$ for each G^s

$$p(x|D) = \sum_G \int_\theta p(x|G,\theta)p(\theta|G,D)p(G|D) \approx \frac{1}{M}\sum_{s=1}^{M} p(x|G^s,\overline{\theta}^s)$$

US census data (d=15,N=49k)



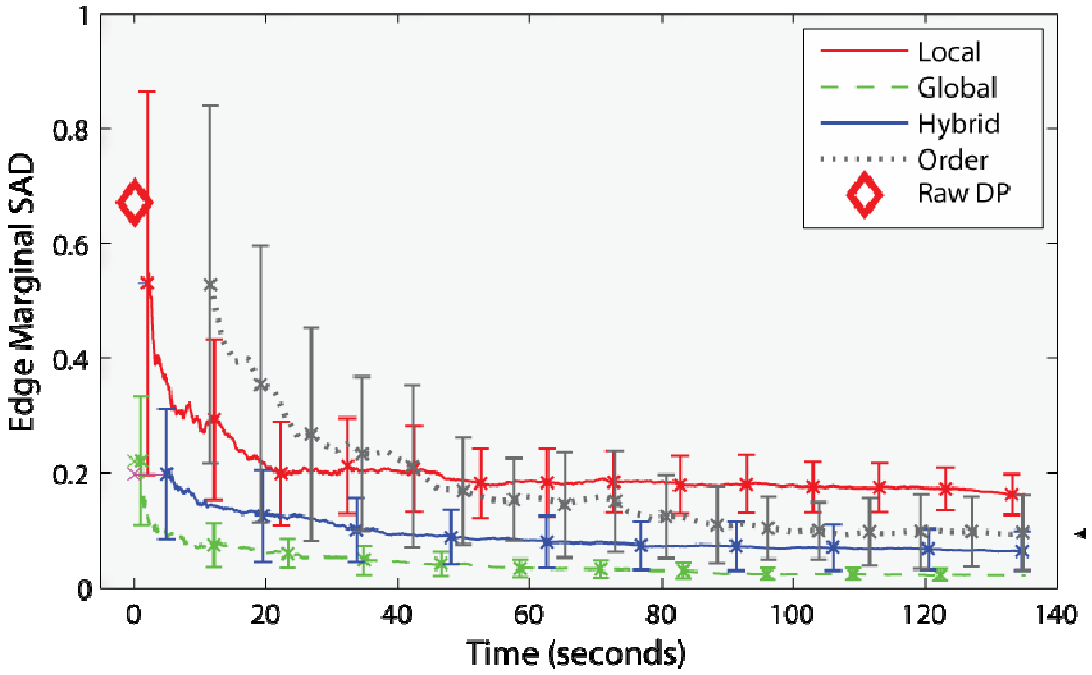
1. Exact BMA (but takes 350h!)
2. MH-DP hybrid β=0.1
3. Plug-in MAP-optimal DAG, MH-DP global β=0, MH-order
4. MH-local β=1
5. Gibbs

Why MH+DP?

- MH + DP mixes faster than MH + other

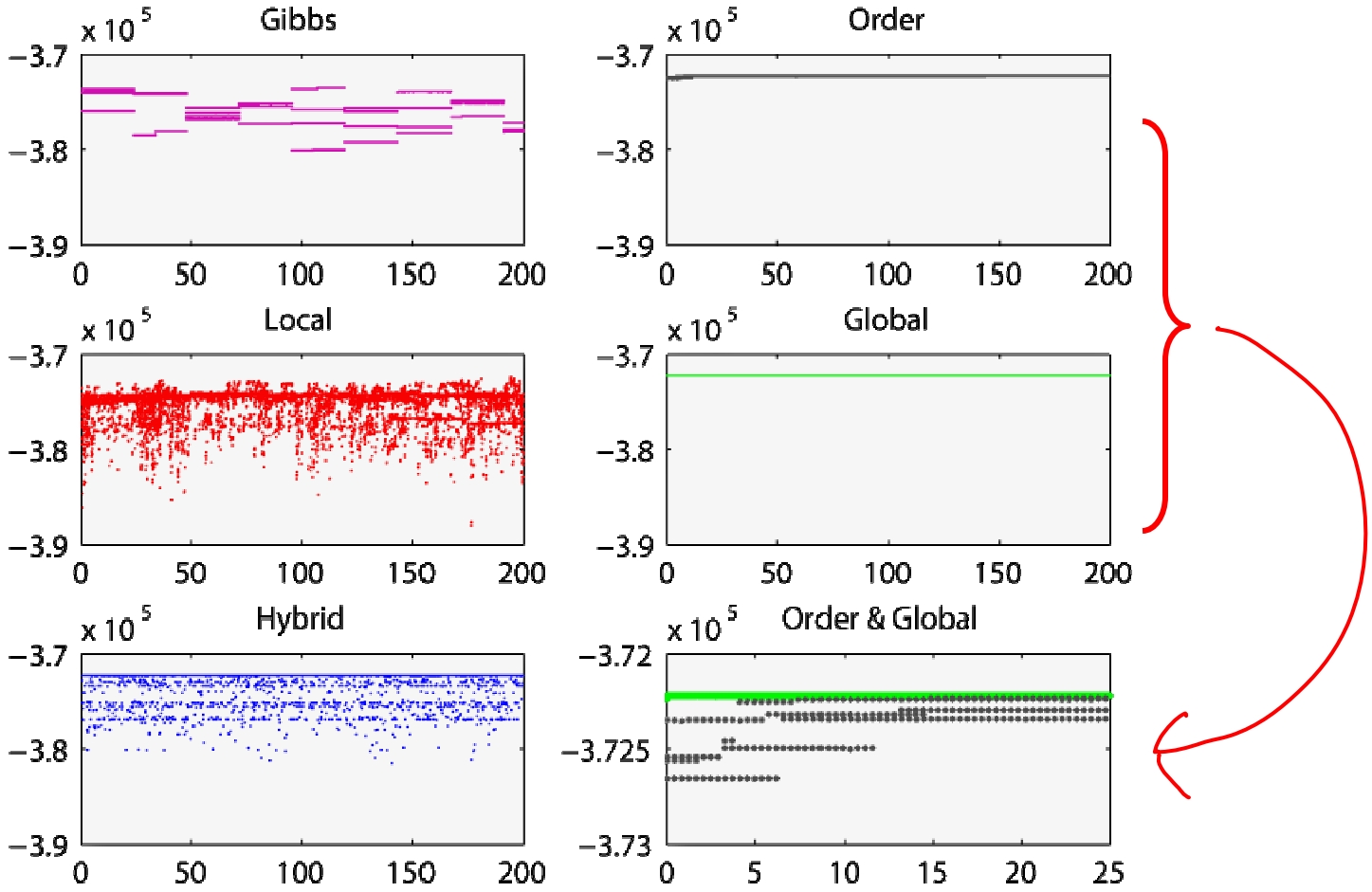
Edge marginal error vs time

$$\sum_{ij} |p(G_{ij} = 1|D) - \hat{p}_t(G_{ij} = 1|D)|$$



d=5 cancer network

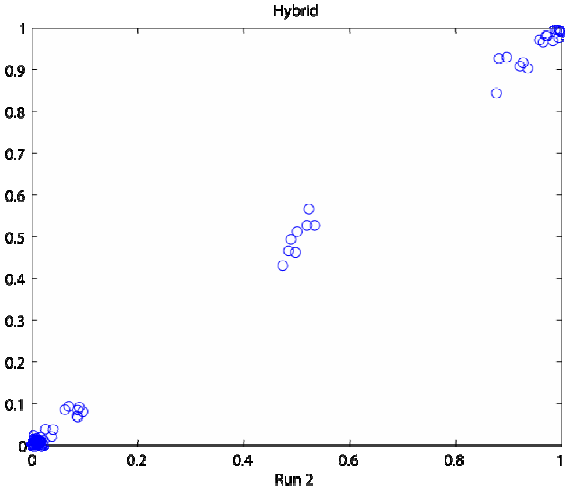
Traceplots of log p(G,D)



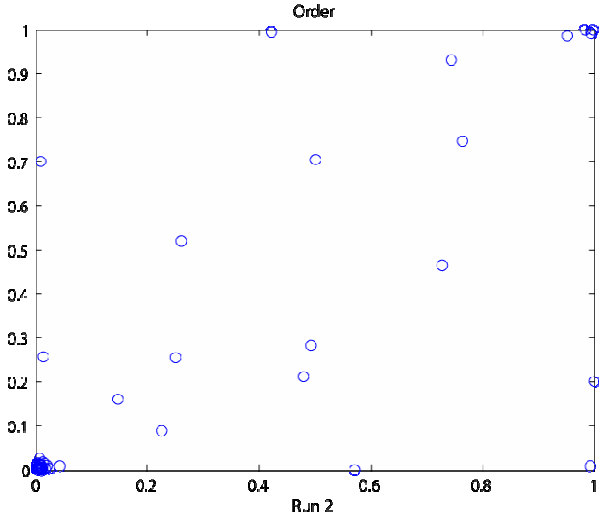
US census (d=14, N=49k)

Repeatability

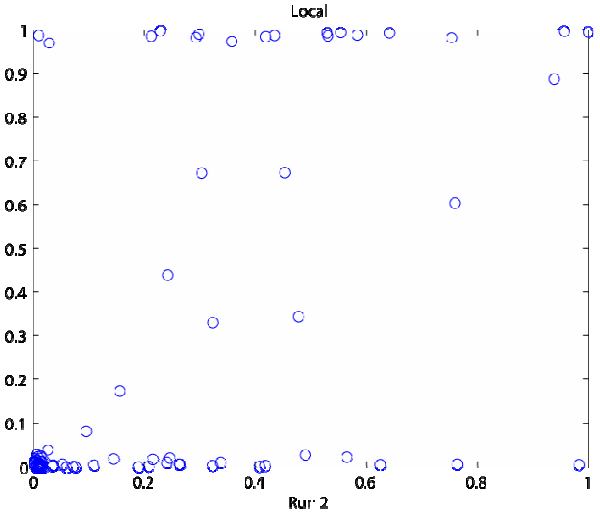
MH+DP (hybrid)



MH-order



MH-local



US census (d=15, N=49k)

We plot edge marginals after two runs from different random starting points

Stochastic search

- MCMC approximates p(M=m|D) by counting how many samples are equal to m.
- Since we can compute p(m,D) exactly, we don't need to visit m more than once. We can approximate

$$p(m|D) \approx \frac{p(m,D)}{\sum_{m' in S} p(m',D)}$$

- It is better to rapidly move through model space, covering as much posterior mass as possible.
- Shotgun stochastic search (SSS), mode oriented stochastic search (MOSS)

Occam's window

- Goal: compute level set of the posterior

$$C(\alpha) = \{m : p(m|D) \geq \alpha p(m^*|D)\}$$

- M^* is unknown, so approximate this by

$$\hat{C}(\alpha) = \{m : p(m|D) \geq \alpha p(\hat{m}^*|D)\}$$

$$\hat{m}^* = \arg\max_{m \in S} p(m|D)$$

- Can find this by beam search, throwing out models that are worse than α time the current best (Raftery, Dobra)

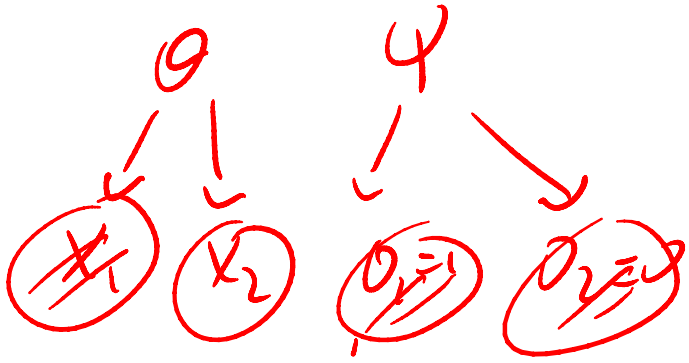
Stat 521A
Lecture 23

Outline

- Basic issues (19.1)
- Gradient ascent for DGMs (19.2.1)
- EM for DGMs (19.2.2)
- Variational EM (19.2.4)
- MCMC for param inf in DGMs (19.3.2)
- Variational Bayes (19.3.3)

MCAR

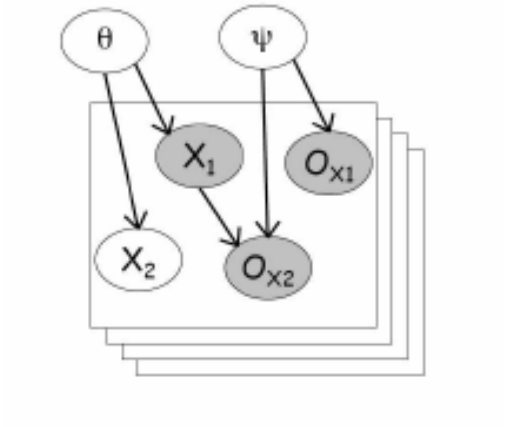
- Let Xi be the true value of variable I, and Oi in {0,1} be whether it is observed or not. Yi(Oi) = Xi or ?.
- Defn 19.1.6. Missing completely at random (MCAR) means X \perp O.
- Given MCAR, we can safely ignore the missing variables (for which Ox=1), since they tell us nothing about theta



$$p(\theta, \psi | Y_1, Y_2) = p(\theta | X_1) p(\psi | O_1, O_2)$$

Missing at random

- Defn 19.1.8. Let H be hidden vars, V be visible vars, and O be observation status. Missing at random means O \perp H | V.



- Intuitively, although O may depend on some of the variables Xv, since we observe Xv, we do not learn anything new about Xh.

Benefits of MAR

- Thm 19.1.9. Given MAR, and a factored prior, $p(\theta,\psi|D) = p(\theta|Xv)\, p(\psi|Xv,O)$
- Pf.

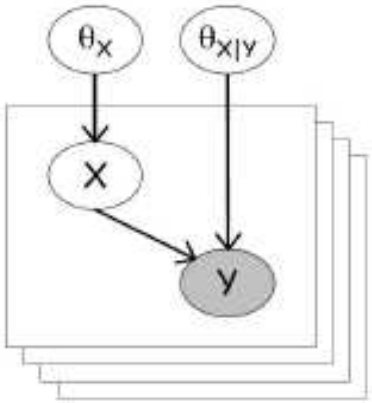
$$
\begin{aligned}
P_{missing}(\boldsymbol{y}) &= \sum_{\boldsymbol{x}^{\boldsymbol{y}}_{\text{hidden}}} [P(\boldsymbol{x}^{\boldsymbol{y}}_{\text{obs}}, \boldsymbol{x}^{\boldsymbol{y}}_{\text{hidden}}) P_{missing}(o_{\boldsymbol{X}} \mid \boldsymbol{x}^{\boldsymbol{y}}_{\text{hidden}}, \boldsymbol{x}^{\boldsymbol{y}}_{\text{obs}})] \\
&= \sum_{\boldsymbol{x}^{\boldsymbol{y}}_{\text{hidden}}} [P(\boldsymbol{x}^{\boldsymbol{y}}_{\text{obs}}, \boldsymbol{x}^{\boldsymbol{y}}_{\text{hidden}}) P_{missing}(o_{\boldsymbol{X}} \mid \boldsymbol{x}^{\boldsymbol{y}}_{\text{obs}})] \\
&= P_{missing}(o_{\boldsymbol{X}} \mid \boldsymbol{x}^{\boldsymbol{y}}_{\text{obs}}) \sum_{\boldsymbol{x}^{\boldsymbol{y}}_{\text{hidden}}} P(\boldsymbol{x}^{\boldsymbol{y}}_{\text{obs}}, \boldsymbol{x}^{\boldsymbol{y}}_{\text{hidden}}) \\
&= P_{missing}(o_{\boldsymbol{X}} \mid \boldsymbol{x}^{\boldsymbol{y}}_{\text{obs}}) P(\boldsymbol{x}^{\boldsymbol{y}}_{\text{obs}}).
\end{aligned}
$$

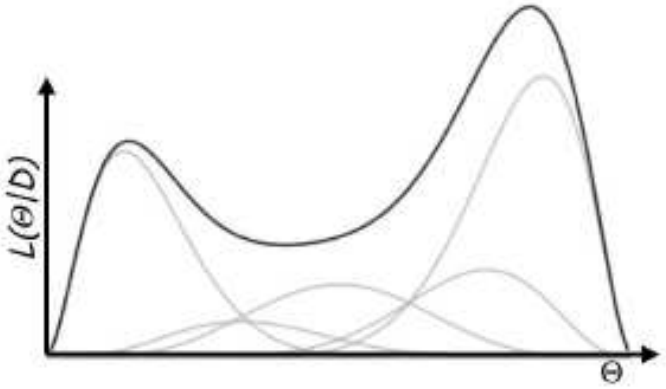
Counter examples to MAR

- Collaborative filtering: people are more likely to rate movies they strongly like or dislike.
- Medicine: if a patient does not have a check mark in the “had X-ray” field, they probably don’t have any bone problems. However, if we explicitly write the “primary complaint” as the cause of which tests are performed, MAR is restored (since we observe why O(Xray)=0).
- Henceforth we will assume MAR.

Multimodality

- For fully observed DGMs, likelihood is convex (assuming each CPD is convex), and hence has a single global maximum.
- When we have missing data, the likelihood is a mixture of up to K^n modes, corresponding to every possible completion pattern



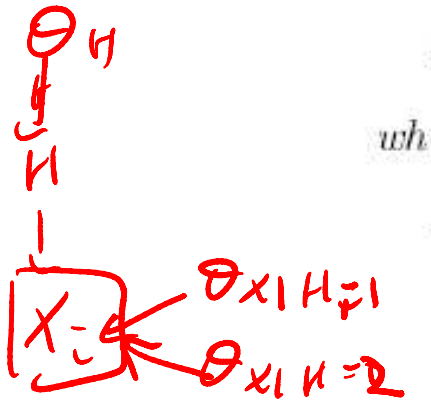


Proposition 19.1.10: *Assuming i.i.d. data, the likelihood can be written as*

$$L(\boldsymbol{\theta} : \mathcal{D}) = \prod_m P(\boldsymbol{o}[m] \mid \boldsymbol{\theta}) = \prod_m \sum_{\boldsymbol{h}[m]} P(\boldsymbol{o}[m], \boldsymbol{h}[m] \mid \boldsymbol{\theta}).$$

Identifiability

- Sometimes we cannot uniquely identify the parameters, even given infinite data
- Eg The experimenter either tosses coin 1 or coin 2, but we don't know which. The model is



$$L(\boldsymbol{\theta} : \mathcal{D}) = P(X = Heads)^{M[Heads]}(1 - P(X = Heads))^{M[Tails]},$$

where

$$P(X = Heads) = \theta_H \theta_{X|h^1} + (1 - \theta_H)\theta_{X|h^2}.$$

- We have eg. $p(D|\theta_H=0.5, \theta_1=0.5, \theta_2=0.5) = p(D|\theta_H=0.5, \theta_1=0.8, \theta_2=0.2)$. The problem is underconstrained.

Identifiability

- Defn 19.1.13. A parameter θ is identifiable if there is no $\theta' \neq \theta$ st $p(X|\theta)=p(X|\theta')$. A model is identifiable if all θ are identifiable.
- A mixture model cannot be identifiable since we can always arbitrarily permute the hidden labels, and the corresponding parameters.
- Hence we should not ask things like “what is the prob. Xi belongs to cluster k” but rather “what is the prob Xi and Xi belong to the same cluster”.

Gradient descent for DGMs

- We can find a local maximum using gradient based methods.
- Consider tabular CPDs.
- Thm 19.2.1.

$$\frac{\partial}{\partial \theta_{ijk}} p(\mathbf{e}) = \frac{p(x_i = k, \mathbf{x}_{\pi_i} = j, \mathbf{e})}{\theta_{ijk}}$$

- Pf.

$$\begin{aligned} \frac{\partial}{\partial \theta_{ijk}} \prod_{i'} \theta_{i', \mathbf{x}_{i'}, x_{i'}} &= \prod_{i' \neq i} \theta_{i', \mathbf{x}_{i'}, x_{i'}} I(\mathbf{x}_i = j, x_i = k) \\ &= \frac{p(\mathbf{e})}{\theta_{ijk}} I(\mathbf{x}_i = j, x_i = k) \end{aligned}$$

Gradient descent for DGMs

- Pf contd

$$\begin{aligned}\frac{\partial}{\partial P(x \mid \boldsymbol{u})} P(\boldsymbol{e}) &= \sum_{\xi:\xi\langle \boldsymbol{E}\rangle = \boldsymbol{e}} \frac{\partial}{\partial P(x \mid \boldsymbol{u})} P(\xi) \\ &= \sum_{\xi:\xi\langle \boldsymbol{E}\rangle = \boldsymbol{e}, \xi\langle X, \mathrm{Pa}_X\rangle = \langle x, \boldsymbol{u}\rangle} \frac{1}{P(x \mid \boldsymbol{u})} P(\xi) \\ &= \frac{1}{P(x \mid \boldsymbol{u})} P(x, \boldsymbol{u}, \boldsymbol{e}).\end{aligned}$$

- Thm 19.2.2.

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(x \mid \boldsymbol{u})} = \frac{1}{P(x \mid \boldsymbol{u})} \sum_{m=1}^{M} P(x, \boldsymbol{u} \mid \boldsymbol{o}[m], \boldsymbol{\theta}).$$

- Chain rule for non-tabular CPDs.

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial \theta} = \sum_{x, \boldsymbol{u}} \frac{\partial \ell(P_{\boldsymbol{\theta}} : \mathcal{D})}{\partial P(x \mid \boldsymbol{u})} \frac{\partial P(x \mid \boldsymbol{u})}{\partial \theta},$$

Gradient algorithms

- Gradient requires inference to compute family marginals.
- Need to enforce positivity and sum-to-one constraints (for discrete) eg reparameterize to unconstrained form

$$P(x \mid u) = \frac{e^{\lambda_{x|u}}}{\sum_{x' \in Val(X)} e^{\lambda_{x'|u}}}.$$

- Need to enforce positive definite – optimize wrt the cholesky factors.
- Have to specify step-size and search direction (use black-box algorithm).
- EM is much easier…

EM for DGMs

- Key intuition: if we knew the values of H, we could compute the MLEs/MAP estimates for θ easily. So we infer H|θ and then estimate θ|H. For the latter, we just need the expected sufficient statistics. For tabular CPDs, this is just a table of expcted counts
- E step

$$\bar{M}_{\boldsymbol{\theta}^t}[x, \boldsymbol{u}] = \sum_m P(x, \boldsymbol{u} \mid \boldsymbol{o}[m], \boldsymbol{\theta}^t).$$

- M step
-

$$\theta^{t+1}_{x|\boldsymbol{u}} = \frac{\bar{M}_{\boldsymbol{\theta}^t}[x, \boldsymbol{u}]}{\bar{M}_{\boldsymbol{\theta}^t}[\boldsymbol{u}]}$$

Pseudocode

```
1   for each t = 0, 1 ..., until convergence
2       // E-step
3     {M̄_t[x_i, u_i]} ← Compute-Expected-Sufficient-Statistics(G, θ^t, D)
4       // M-step
5     for each i = 1, ..., n
6       for each x_i, u_i ∈ Val(X_i, Pa^G_{X_i})
7         θ^{t+1}_{x_i|u_i} ← M̄_t[x_i, u_i] / M̄_t[u_i]
8   return θ^t
```

```
1       // Initialize data structures
2     for each i = 1, ..., n
3       for each x_i, u_i ∈ Val(X_i, Pa^G_{X_i})
4         M̄[x_i, u_i] ← 0
5       // Collect probabilities from all instances
6     for each m = 1 ... M
7       Run inference on ⟨G, θ⟩ using evidence o[m]
8       for each i = 1, ..., n
9         for each x_i, u_i ∈ Val(X_i, Pa^G_{X_i})
10          M̄[x_i, u_i] ← M̄[x_i, u_i] + P(x_i, u_i | o[m])
11    return {M̄[x_i, u_i] : ∀i = 1, ..., n, ∀x_i, u_i ∈ Val(X_i, Pa^G_{X_i})}
```

ECDLL

- Define expected complete data log likelihood, wrt Q distribution over H|D

$$\mathbf{E}_Q[\ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle)] = \sum_{\mathcal{H}} Q(\mathcal{H}) \ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle)$$

- For tabular CPDs, we have

$$\ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle) = \sum_{i=1}^{n} \sum_{(x_i, \boldsymbol{u}_i) \in Val(X_i, \mathrm{Pa}_{X_i})} M_{\langle \mathcal{D}, \mathcal{H} \rangle}[x_i, \boldsymbol{u}_i] \log \theta_{x_i | \boldsymbol{u}_i}.$$

$$\mathbf{E}_Q[\ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle)] = \sum_{i=1}^{n} \sum_{(x_i, \boldsymbol{u}_i) \in Val(X_i, \mathrm{Pa}_{X_i})} \mathbf{E}_Q \left[M_{\langle \mathcal{D}, \mathcal{H} \rangle}[x_i, \boldsymbol{u}_i] \right] \log \theta_{x_i | \boldsymbol{u}_i}.$$

$$\mathbf{E}_Q[\ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle)] = \sum_{i=1}^{n} \sum_{(x_i, \boldsymbol{u}_i) \in Val(X_i, \mathrm{Pa}_{X_i})} \bar{M}_Q[x_i, \boldsymbol{u}_i] \log \theta_{x_i | \boldsymbol{u}_i}.$$

ECDLL for exp fam

- The key to making EM simple for expfam models is that the log-likelihood is linear in the sufficient statistics

$$P(\xi \mid \theta) = \frac{1}{Z(\theta)} A(\xi) \exp\{\langle \mathrm{t}(\theta), \tau(\xi) \rangle\}, \qquad \tau(\langle \mathcal{D}, \mathcal{H} \rangle) = \sum_m \tau(\boldsymbol{o}[m], \boldsymbol{h}[m]).$$

$$\ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle) = \langle \mathrm{t}(\boldsymbol{\theta}), \tau(\langle \mathcal{D}, \mathcal{H} \rangle) \rangle + \sum_m A(\boldsymbol{o}[m], \boldsymbol{h}[m]) - \log Z(\boldsymbol{\theta}).$$

$$\boldsymbol{E}_Q[\ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle)] = \langle \mathrm{t}(\boldsymbol{\theta}), \boldsymbol{E}_Q[\tau(\langle \mathcal{D}, \mathcal{H} \rangle)] \rangle + \sum_m \boldsymbol{E}_Q[A(\boldsymbol{o}[m], \boldsymbol{h}[m])] - M \log Z(\boldsymbol{\theta}).$$

const

Choosing Q (for E step)

- Define

$$\Phi_{\mathcal{D}}[\theta, Q] = \mathbb{E}_Q[\ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle)] + \mathbb{H}_Q(\mathcal{H}).$$

- Thm 19.2.5.

Corollary 19.2.5: *For any Q,*

$$\begin{aligned} \ell(\boldsymbol{\theta} : \mathcal{D}) &= \Phi_{\mathcal{D}}[\boldsymbol{\theta}, Q] + \mathbb{D}(Q(\mathcal{H}) \| P(\mathcal{H} \mid \mathcal{D}, \boldsymbol{\theta})) \\ &= \mathbb{E}_Q[\ell(\boldsymbol{\theta} : \langle \mathcal{D}, \mathcal{H} \rangle)] + \mathbb{H}_Q(\mathcal{H}) + \mathbb{D}(Q(\mathcal{H}) \| P(\mathcal{H} \mid \mathcal{D}, \boldsymbol{\theta})). \end{aligned}$$

- From (2), ECDLL is lower bound on LL.
- From (1), if Q=p(H|D,\theta), then bound is tight.
- EM alternates between optimizing Q and optimizing \theta. Can do partial updates.

Convergence

- Thm 19.2.6. If we do exact EM (so Q=p(H|D,theta)), then the LL never decreases

Theorem 19.2.6: *During iterations of the EM procedure of Algorithm 19.2, we have that*

$$\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}) - \ell(\boldsymbol{\theta}^{t} : \mathcal{D}) \geq \mathbf{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}, \mathcal{H})\right] - \mathbf{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta}^{t} : \mathcal{D}, \mathcal{H})\right].$$

As a consequence, we obtain that:

$$\ell(\boldsymbol{\theta}^{t} : \mathcal{D}) \leq \ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}).$$

PROOF We begin with the first statement. Using Corollary 19.2.5, with the distribution $Q^t(\mathcal{H}) = P(\mathcal{H} \mid \mathcal{D}, \boldsymbol{\theta}^t)$ we have that

$$\begin{aligned}
\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}) &= \mathbf{E}_{Q^t}\left[\ell(\boldsymbol{\theta}^{t+1} : \langle \mathcal{D}, \mathcal{H} \rangle)\right] + \mathbf{H}_{Q^t}(\mathcal{H}) + \mathbf{D}(Q^t(\mathcal{H}) \| P(\mathcal{H} \mid \mathcal{D}, \boldsymbol{\theta}^{t+1})) \\
\ell(\boldsymbol{\theta}^{t} : \mathcal{D}) &= \mathbf{E}_{Q^t}\left[\ell(\boldsymbol{\theta}^{t} : \langle \mathcal{D}, \mathcal{H} \rangle)\right] + \mathbf{H}_{Q^t}(\mathcal{H}) + \mathbf{D}(Q^t(\mathcal{H}) \| P(\mathcal{H} \mid \mathcal{D}, \boldsymbol{\theta}^{t})) \\
&= \mathbf{E}_{Q^t}\left[\ell(\boldsymbol{\theta}^{t} : \langle \mathcal{D}, \mathcal{H} \rangle)\right] + \mathbf{H}_{Q^t}(\mathcal{H})
\end{aligned}$$

The last step is justified by our choice of $Q^t(\mathcal{H}) = P(\mathcal{H} \mid \mathcal{D}, \boldsymbol{\theta}^t)$. Subtracting these two terms, we have that

$$\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}) - \ell(\boldsymbol{\theta}^{t} : \mathcal{D}) = \mathbf{E}_{Q^t}\left[\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}, \mathcal{H})\right] - \mathbf{E}_{Q^t}\left[\ell(\boldsymbol{\theta}^{t} : \mathcal{D}, \mathcal{H})\right] + \mathbf{D}(Q^t(\mathcal{H}) \| P(\mathcal{H} \mid \mathcal{D}, \boldsymbol{\theta}^{t+1}))$$

As the last term is non-negative, we get the desired inequality.

Convergence cont'd

Theorem 19.2.6: *During iterations of the EM procedure of Algorithm 19.2, we have that*

$$\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}) - \ell(\boldsymbol{\theta}^{t} : \mathcal{D}) \geq \mathbf{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}, \mathcal{H})\right] - \mathbf{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}\left[\ell(\boldsymbol{\theta}^{t} : \mathcal{D}, \mathcal{H})\right].$$

As a consequence, we obtain that:

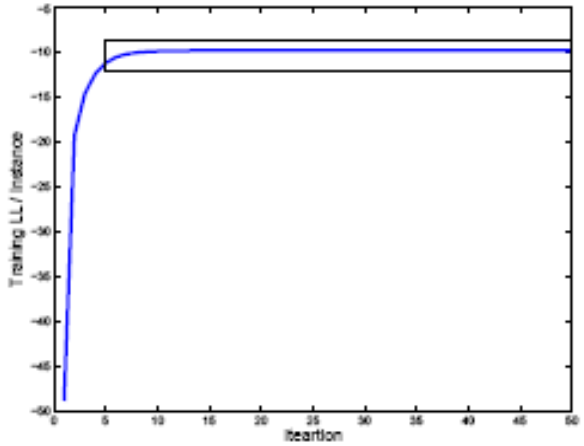
$$\ell(\boldsymbol{\theta}^{t} : \mathcal{D}) \leq \ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}).$$

To prove the second statement of the theorem, we note that $\boldsymbol{\theta}^{t+1}$ is the value of $\boldsymbol{\theta}$ that maximizes $\mathbf{E}_{P(\mathcal{H}|\mathcal{D},\boldsymbol{\theta}^t)}[\ell(\boldsymbol{\theta} : \mathcal{D}, \mathcal{H})]$. Hence the value obtained for this expression for $\boldsymbol{\theta}^{t+1}$ is at least at large as the value obtained for any other set of parameters, including $\boldsymbol{\theta}^t$. We conclude that the right-hand side of the inequality is non-negative, which implies the first statement. ■

Theorem 19.2.7: *Suppose that $\boldsymbol{\theta}^t$ is such that $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t$ during EM, and $\boldsymbol{\theta}^t$ is also an interior point of the allowed parameter space. Then $\boldsymbol{\theta}^t$ is a stationary point of the log-likelihood function.*

Rate of convergence

- Initially fast, then very slow; can switch over to conjugate gradient near optimum



- EM has linear convergence rate

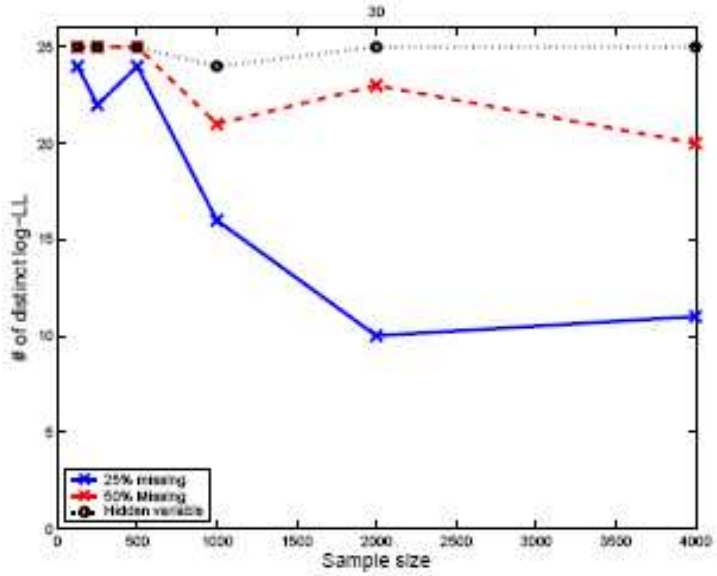
$$\epsilon_t = \ell^* - \ell_t$$

Although we do not go through the proof, one can show that EM has *linear convergence rate*. This means that for each domain there exists a t_0 and $\alpha < 1$ such that for all $t \geq t_0$

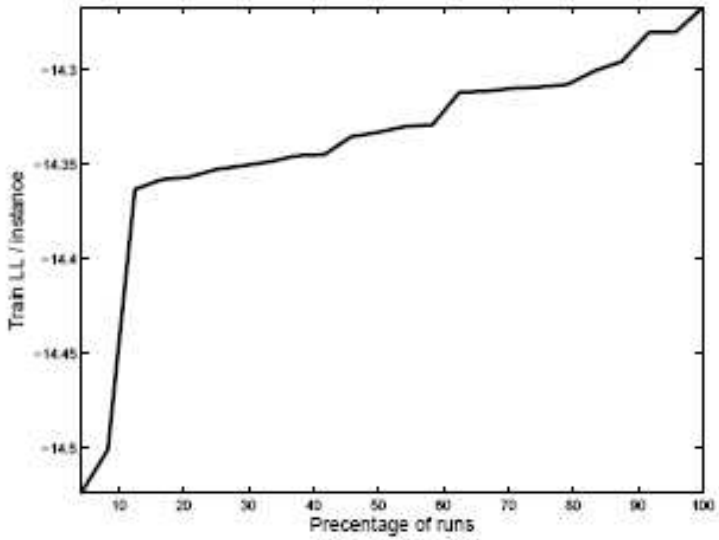
$$\epsilon_{t+1} \leq \alpha \epsilon_t.$$

Local maxima

- Maxima can differ a lot in quality.
- Can do multiple restart, killing off some runs early if they look bad (as in beam search).



(a)



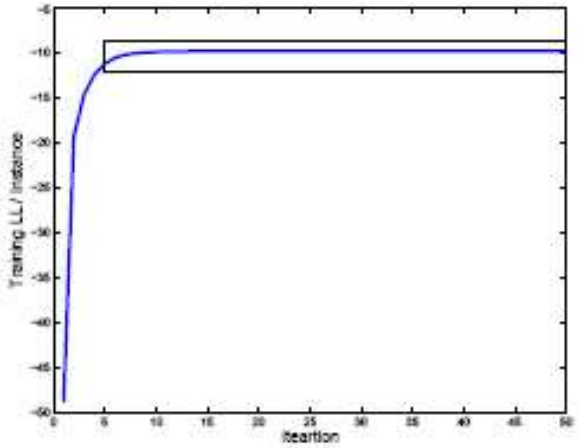
(b)

Assessing convergence

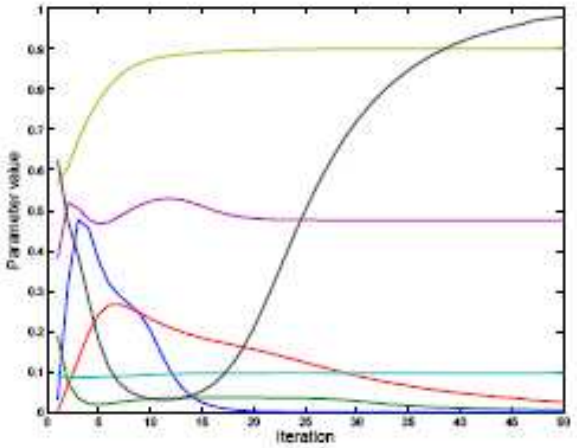
- Can check whether parameters stop changing or LL stops changing. Can be quite different.
- Recall

$$\frac{\partial \ell(\theta : \mathcal{D})}{\partial P(x \mid u)} = \frac{1}{P(x \mid u)} \sum_{m=1}^{M} P(x, u \mid o[m], \theta).$$

- If p(x,u|o[m]) small, gradient is small, else O(M)
- Hence effects of param on LL can be small or large.



(a)

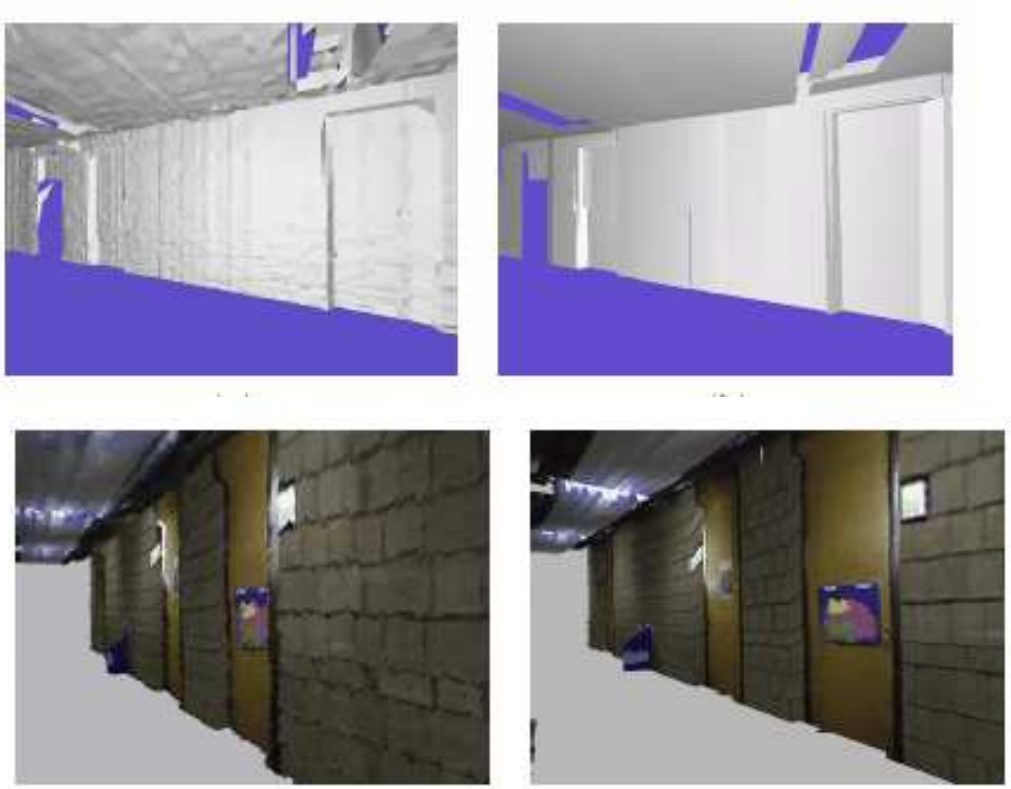


(b)

Accelerating convergence

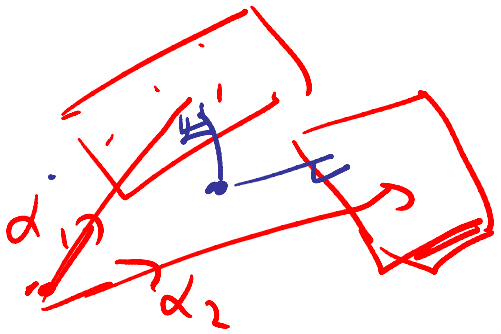
- Hard assignment EM (eg Kmeans). E step is searching over discrete assignments; this tends to converge faster (but to a worse solution).
- Hybrid EM/CG
- Over relaxation: step size > 1.
- Stochastic EM: since Q(H) = prod_m Q(hm|om), we can do inference on only a subset of the datacases (mini-batch) and then do an M step
- (Monte Carlo EM: sampling in the E step)

Example: fitting planes to 3d point clouds



$P(\boldsymbol{X}_m \mid C_m = k : \boldsymbol{\theta}_k)$ *to be* $\propto \mathcal{N}\left(d(x, p_k); 0, \sigma^2\right)$.

$d(\boldsymbol{x}, p_k) = |\alpha_k \boldsymbol{x} - \beta_k|$.



Variational EM

- Restrict Q distribution in E step to a tractable family, rather than p(H|D,theta)

$$\max_{\theta} \max_{Q \in \mathcal{Q}} \Phi_{\mathcal{D}}[\theta, Q]$$

- Eg do mean-field in the E step, then regular M step
- Maximizes a lower bound on the LL

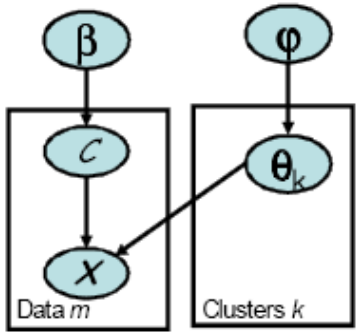
$$\ell(\theta : \mathcal{D}) = \max_{Q} \Phi_{\mathcal{D}}[\theta, Q] \geq \max_{Q \in \mathcal{Q}} \Phi_{\mathcal{D}}[\theta, Q].$$

MCMC

- Can compute $p(\theta,H|D)$ using standard algorithms
- Parameter collapsed particles: sample θ, compute $p(H|D)$ analytically
- Data completion collapsed particles: sample H, compute $p(\theta|H,D)$ analytically

Marginalizing out H

- Bayesian Mixture model

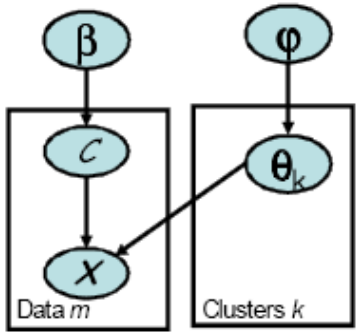


$$P(\mathcal{D} \mid \theta_1, \ldots, \theta_K, \lambda) = \prod_{m=1}^{M} \left(\sum_{k=1}^{K} P(C[m] = c^k \mid \lambda) P(\boldsymbol{x}[m] \mid C[m] = c^k, \theta_k) \right)$$

$$P(\theta_k \mid \lambda, \theta_{-k}, \mathcal{D}) \propto P(\mathcal{D} \mid \theta_1, \ldots, \theta_K, \lambda).$$

Have to use MH

Marginalizing out \theta



Collapsed Gibbs sampling
Cf DP mixtures

$$P(\boldsymbol{\lambda} \mid \boldsymbol{c}) = Dirichlet(\alpha_0/K + |I_1(\boldsymbol{c})|, \ldots, \alpha_0/K + |I_K(\boldsymbol{c})|).$$

$$P(\boldsymbol{\theta}_k \mid \boldsymbol{c}, \mathcal{D}, \phi) = Q(\boldsymbol{\theta}_k \mid \mathcal{D}_{I_k(\boldsymbol{c})}, \phi) \propto P(\boldsymbol{\theta}_k \mid \phi) \prod_{m \in I_k(\boldsymbol{c})} P(\boldsymbol{x}[m] \mid \boldsymbol{\theta}_k),$$

$$P(C[m'] = k \mid \boldsymbol{c}_{-m'}, \mathcal{D}, \phi) \propto P(C[m'] = k \mid \boldsymbol{\lambda}, \boldsymbol{c}_{-m'}) P(\boldsymbol{x}[m'] \mid C[m'] = k, \boldsymbol{x}[I_k(\boldsymbol{c}_{-m'})], \phi).$$

$$P(C[m'] = k \mid \boldsymbol{c}_{-m'}, \mathcal{D}, \phi) \propto (|I_k(\boldsymbol{c}_{-m'})| + \alpha_0/K) Q(\boldsymbol{X} \mid \mathcal{D}_{I_k(\boldsymbol{c}_{-m'})}, \phi).$$

Variational Bayes

- Min KL(Q|P) where we assume

$$Q(\boldsymbol{\theta}, \mathcal{H}) = Q(\boldsymbol{\theta})Q(\mathcal{H}).$$

- Thm 19.3.6. If we have global param independence and Q(θ,H)=Q(θ) Q(H) then

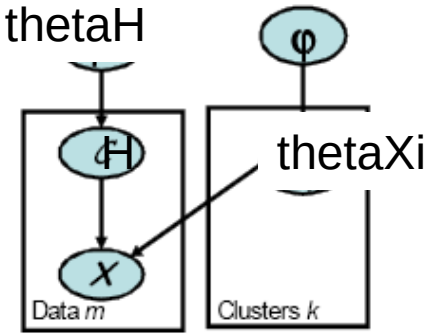
$$Q(\boldsymbol{\theta}, \mathcal{H}) = \prod_i Q(\boldsymbol{\theta}_{X_i|\boldsymbol{U}_i}) \prod_m Q(\boldsymbol{h}[m]).$$

- Hence we can optimize each Q(h[m]) separately – just like inference per case in the E step – and then optimize each Q(\theta_i) separately – just like optimizing each family in the M step
- E step: we do inference with expected params
- M step: we fit a distribution

VB for MixBernoulli

$$Q = Q(\boldsymbol{\theta}_H)\left[\prod_i Q(\boldsymbol{\theta}_{X_i|H})\right]\left[\prod_m Q(H[m])\right].$$

Beta Beta Bernoulli



$$\begin{aligned}
Q(\boldsymbol{\theta}_H) &\propto \exp\left\{\ln P(\boldsymbol{\theta}_H) + \sum_m E_{Q(H[m])}[\ln P(H[m] \mid \boldsymbol{\theta}_H)]\right\} \\
Q(\boldsymbol{\theta}_{X_i|H}) &\propto \exp\left\{\ln P(\boldsymbol{\theta}_{X_i|H}) + \sum_m E_{Q(H[m])}\left[\ln P(x_i[m] \mid H[m], \boldsymbol{\theta}_{X_i|H})\right]\right\} \\
Q(H[m]) &\propto \exp\left\{E_{Q(\boldsymbol{\theta}_H)}[\ln P(H[m] \mid \boldsymbol{\theta}_H)]\right. \\
&\qquad \left. + \sum_i E_{Q(\boldsymbol{\theta}_{X_i|H})}\left[\ln P(x_i[m] \mid H[m], \boldsymbol{\theta}_{X_i|H})\right]\right\}.
\end{aligned}$$

$$\ln P(\boldsymbol{\theta}_H = \langle \theta_{h^0}, \theta_{h^1} \rangle) = \ln c + (\alpha_{h^0} - 1)\ln\theta_{h^0} + (\alpha_{h^1} - 1)\ln\theta_{h^1}.$$

$$E_{Q(H[m])}[\ln P(H[m] \mid \boldsymbol{\theta}_H = \langle \theta_{h^0}, \theta_{h^1} \rangle)] = Q(H[m] = h^0)\ln\theta_{h^0} + Q(H[m] = h^1)\ln\theta_{h^1}.$$

$$\begin{aligned}
Q(\boldsymbol{\theta}_H = \langle \theta_{h^0}, \theta_{h^1} \rangle) &\propto \exp\left\{\left(\alpha_{h^0} + \sum_m Q(H[m] = h^0) - 1\right)\ln\theta_{h^0} + \right. \\
&\qquad \left.\left(\alpha_{h^1} + \sum_m Q(H[m] = h^1) - 1\right)\ln\theta_{h^1}\right\} \\
&= \theta_{h^0}^{\alpha_{h^0} + \sum_m Q(H[m]=h^0) - 1}\theta_{h^1}^{\alpha_{h^1} + \sum_m Q(H[m]=h^1) - 1}.
\end{aligned}$$

$$\begin{aligned}
\alpha'_{h^0} &= \alpha_{h^0} + \sum_m Q(H[m] = h^0) \\
\alpha'_{h^1} &= \alpha_{h^1} + \sum_m Q(H[m] = h^1).
\end{aligned}$$

VB update for H[m]

Regular E step

$$P(H[m] \mid x_1[m], \ldots x_n[m]) \propto P(H[m] \mid \boldsymbol{\theta}_H) \prod_i P(x_i[m] \mid H[m], \boldsymbol{\theta}_{X_i|H}).$$

VB version

$$\boldsymbol{E}_{Q(\boldsymbol{\theta}_{X_i|H})}\left[\ln P(x_i \mid H[m], \boldsymbol{\theta}_{X_i|H})\right] = \int_0^1 Q(\theta_{x_i|H[m]}) \ln \theta_{x_i|H[m]} d\theta_{x_i|H[m]}.$$

$$\boldsymbol{E}_{Q(\boldsymbol{\theta}_{X_i|H})}\left[\ln P(x_i \mid H[m], \boldsymbol{\theta}_{X_i|H})\right] = \varphi(\alpha'_{x_i|h}) - \varphi(\sum_{x'_i} \alpha'_{x'_i|h})$$

where α' are the hyperparameters of the posterior approximation in $Q(\boldsymbol{\theta}_{X_i|H})$ and $\varphi(z) = (\ln \Gamma(z))' = \frac{\Gamma'(z)}{\Gamma(z)}$ is the digamma *function, which is equal to $\ln(z)$ plus a polynomial function of $\frac{1}{z}$. And so, for $z \gg 1$, $\varphi(z) \approx \ln(z)$. Using this approximation, we see that*

$$\boldsymbol{E}_{Q(\boldsymbol{\theta}_{X_i|H})}\left[\ln P(x_i \mid H[m], \boldsymbol{\theta}_{X_i|H})\right] \approx \ln \frac{\alpha'_{x_i|h}}{\sum_{x'_i} \alpha'_{x'_i|h}},$$

Variational methods

- From Lecture 10:
- Minimize

$$
\begin{aligned}
D(Q||P) &= \ln Z - F(\tilde{P}, Q) \\
F(\tilde{P}, Q) &\stackrel{\text{def}}{=} H_Q(x) + \sum_i E_{C_i \sim Q} \ln \psi_i(C_i)
\end{aligned}
$$

- This always increases the lower bound and will always converge

Mean field approximation

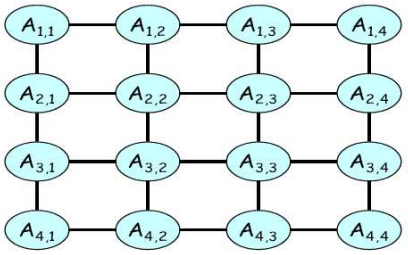
- Let us assume the approximate posterior is fully factorized

$$Q(x) = \prod_i Q_i(x_i)$$

- Then the objective (negative free energy) is

$$\begin{aligned} F(\tilde{P}, Q) &\stackrel{\text{def}}{=} H_Q(x) + \sum_c E_{X_c \sim Q} \ln \phi_c(X_c) \\ &= \sum_i H(Q_i) + \sum_c \sum_{x_c} (\prod_{i \in c} Q_i(x_{c,i})) \ln \phi_c(x_c) \end{aligned}$$

- Eg 4x4 grid $O(n_e K^2)$ for energy, $O(n_e K)$ for H



$$\begin{aligned} F[\tilde{P}_\Phi, Q] &= E_{\{A_{1,1}, A_{2,1}\} \sim Q}[\ln \phi(A_{1,1}, A_{2,1})] + E_{\{A_{2,1}, A_{3,1}\} \sim Q}[\ln \phi(A_{2,1}, A_{3,1})] + E_{\{A_{3,1}, A_{4,1}\} \sim Q}[\\ &\quad \ldots \\ &\quad E_Q[\ln \phi(A_{1,1}, A_{1,2})] + E_Q[\ln \phi(A_{1,2}, A_{1,3})] + E_Q[\ln \phi(A_{1,3}, A_{1,4})] + \\ &\quad \ldots \\ &\quad H_Q(A_{1,1}) + H_Q(A_{1,2}) + H_Q(A_{1,3}) + H_Q(A_{1,4}) + \\ &\quad \ldots \\ &\quad H_Q(A_{4,1}) + H_Q(A_{4,2}) + H_Q(A_{4,3}) + H_Q(A_{4,4}) \end{aligned}$$

Convexity

- Objective is concave in each arg (entropy is concave in each Q_i, expected energy is linear in Q_i)

$$F(\tilde{P}, Q) = \sum_i H(Q_i) + \sum_c \sum_{x_c} (\prod_{i \in c} Q_i(x_{c,i})) \ln \phi_c(x_c)$$

- The set of completely factorized distributions is not convex

$$Q^3(x) = \lambda \prod_i Q^1(x_i) + (1-\lambda) \prod_i Q_i^2(x_i)$$

Not factorized

- Hence we are optimizing the objective over a non-convex space, and will be subject to local maxima
- Let us derive equations that characterize the fixed points. These could correspond to saddle points or local minima, but such points are unstable and unlikely to be the result of our iterative update scheme.

Notation

- Define

$$\langle f(x_h)\rangle \stackrel{\text{def}}{=} \sum_{x_h}\left[\prod_{i\in h} Q_i(x_i)\right] f(x_h)$$

$$\langle f(x_h)\rangle_{j,k} \stackrel{\text{def}}{=} \sum_{x_h\setminus x_j}\left[\prod_{i\in h, i\neq j} Q_i(x_i)\right] f(x_h|x_j=k)$$

$$\langle f(x_h)\rangle = \sum_k Q_j(x_j=k)\langle f(x_h)\rangle_{j,k}$$

$$\ln p(x_v) \geq \sum_c \langle \ln \phi_c(x_c)\rangle + \sum_i H(Q_i)$$

$$= \sum_k Q_j(k) \sum_c \langle \ln \phi_c(x_c)\rangle_{j,k} + H(Q_j) + \sum_{i\neq j} H(Q_i)$$

We mostly follow Tommi Jaakkola's notation rather than Daphne Koller's

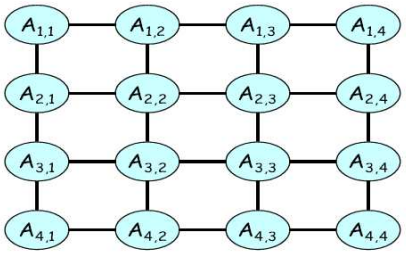
Mean field equations

$$
\begin{aligned}
\ln p(x_v) &\geq \sum_k Q_j(k) \sum_c \langle \ln \phi_c(x_c) \rangle_{j,k} + H(Q_j) + \sum_{i \neq j} H(Q_i) \\
&\stackrel{\text{def}}{=} L(Q_j) \\
S_{j,k} &\stackrel{\text{def}}{=} \sum_{c: j \in c} \langle \ln \phi_c(x_c) \rangle_{j,k} \\
L(Q_j) &= \sum_k Q_j(k)(S_{j,k} - \ln Q_j(k)) + C \\
L(Q_j, \lambda) &\stackrel{\text{def}}{=} L(Q_j) + \lambda(\sum_{k'} Q_j(k') - 1) \\
\frac{\partial}{\partial Q_j(k)} L(Q_j, \lambda) &= S_{j,k} - \ln Q_j(k) - 1 + \lambda = 0 \\
Q_j(k) &= \exp(S_{j,k}) \exp(\lambda - 1) \\
&= \frac{1}{Z_j} \exp(\sum_c \langle \ln \phi_c(x_c) \rangle_{j,k})
\end{aligned}
$$

Example: grid

$$Q(x_i) = \frac{1}{Z_i} \exp\left\{ \sum_{\phi: X_i \in Scope[\phi]} E_{(\boldsymbol{U}_\phi - \{X_i\}) \sim Q}[\ln \phi(\boldsymbol{U}_\phi, x_i)] \right\}$$

$$Q(a_{i,j}) = \frac{1}{Z_{i,j}} \exp\left\{ \begin{array}{l} \sum_{a_{i-1,j}} Q(a_{i-1,j}) \ln(\phi(a_{i-1,j}, a_{i,j})) + \\ \sum_{a_{i,j-1}} Q(a_{i,j-1}) \ln(\phi(a_{i,j-1}, a_{i,j})) + \\ \sum_{a_{i+1,j}} Q(a_{i+1,j}) \ln(\phi(a_{i,j}, a_{i+1,j})) + \\ \sum_{a_{i,j+1}} Q(a_{i,j+1}) \ln(\phi(a_{i,j}, a_{i,j+1})) \end{array} \right\}.$$



EM

- Suppose we want to find a MAP estimate

$$\max_{\theta} \log p(\theta) + \sum_{n} \log p(x_n|\theta)$$

- If we have latent variables Z we can use EM
- E step: compute expected complete data log joint

$$f(\theta, \theta_{old}) \quad = \quad \log p(\theta) + \sum_{n=1}^{N} \sum_{z} p(z|x_n, \theta_{old}) \log p(z, x_n|\theta)$$

- M step: set

$$\theta_{new} = \arg\max f(\theta, \theta_{old})$$

Variational EM

- Consider the negative free energy

$$F(x, Q, \theta) = \sum_z Q(z) \log p(x, z|\theta) + H(Q)$$

- Earlier we showed this is a lower bound on the log-likelihood

$$\begin{aligned} F(x, Q, \theta) &= \ln Z(x, \theta) - D(Q||p(z|x, \theta)) \\ \log p(x|\theta) &= \ln Z = \max_Q F(x, Q, \theta) = F(x, Q^*, \theta) \geq F(x, Q, \theta) \end{aligned}$$

- Where the bound is tight if $Q^*(z) = p(z|x, \theta)$
- E step: find $Q_n(z)$ that maximize

$$F(x_n, Q_n, \theta_{old})$$

- M step: find \theta that maximize

$$\log p(\theta) + \sum_n F(x_n, Q_n, \theta)$$

Variational EM

- An exact E step is equivalent to setting

$$Q_n(z) = p(z|x_n, \theta_{old})$$

- The corresponding M step maximizes

$$\begin{aligned}\sum_n F(x_n, Q_n, \theta) &= \sum_n [\sum_z p(z|x_n, \theta_{old}) \log p(z, x_n|\theta)] + H(Q_n) \\ &= f(\theta, \theta_{old}) + \sum_n H(Q_n)\end{aligned}$$

- Since $H(Q_n)$ is independent of θ, this reduces to the standard EM algorithm.
- Generalized EM merely increases (not maximizes) θ in the M step.
- Similarly we can simply improve Q_n in the E step

Neal and Hinton, "A new view of the EM algorithm", 1998

Variational Bayes

- We can replace the point estimate of θ with a distribution and try to minimize

$$D(Q(z_{1:N}, \theta | x_{1:N}) || p(z_{1:N}, \theta | x_{1:N}))$$

- The distinction between E and M vanishes: we are just doing sequential updates of $Q(Z_n)$ and $Q(\theta)$
- This gives us the benefits of being Bayesian for the same computational speed as EM

VB for univariate Gaussian

$$q_j^\star(\mathbf{Z}_j) = \frac{\exp\left(\mathbb{E}_{i\neq j}[\ln p(\mathbf{X},\mathbf{Z})]\right)}{\int \exp\left(\mathbb{E}_{i\neq j}[\ln p(\mathbf{X},\mathbf{Z})]\right)\,\mathrm{d}\mathbf{Z}_j}. \qquad \ln q_j^\star(\mathbf{Z}_j) = \mathbb{E}_{i\neq j}[\ln p(\mathbf{X},\mathbf{Z})] + \text{const.}$$

$$p(\mathcal{D}|\mu,\tau) = \left(\frac{\tau}{2\pi}\right)^{N/2} \exp\left\{-\frac{\tau}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}.$$

$$\begin{aligned} p(\mu|\tau) &= \mathcal{N}\left(\mu|\mu_0,(\lambda_0\tau)^{-1}\right) \\ p(\tau) &= \mathrm{Gam}(\tau|a_0,b_0) \end{aligned}$$

$$q(\mu,\tau) = q_\mu(\mu)q_\tau(\tau).$$

Gaussian

$$\begin{aligned} \ln q_\mu^\star(\mu) &= \mathbb{E}_\tau\left[\ln p(\mathcal{D}|\mu,\tau) + \ln p(\mu|\tau)\right] + \text{const.} \\ &= -\frac{\mathbb{E}[\tau]}{2}\left\{\lambda_0(\mu-\mu_0)^2 + \sum_{n=1}^{N}(x_n-\mu)^2\right\} + \text{const.} \end{aligned}$$

$$\begin{aligned} \mu_N &= \frac{\lambda_0\mu_0 + N\overline{x}}{\lambda_0+N} \\ \lambda_N &= (\lambda_0+N)\mathbb{E}[\tau]. \end{aligned}$$

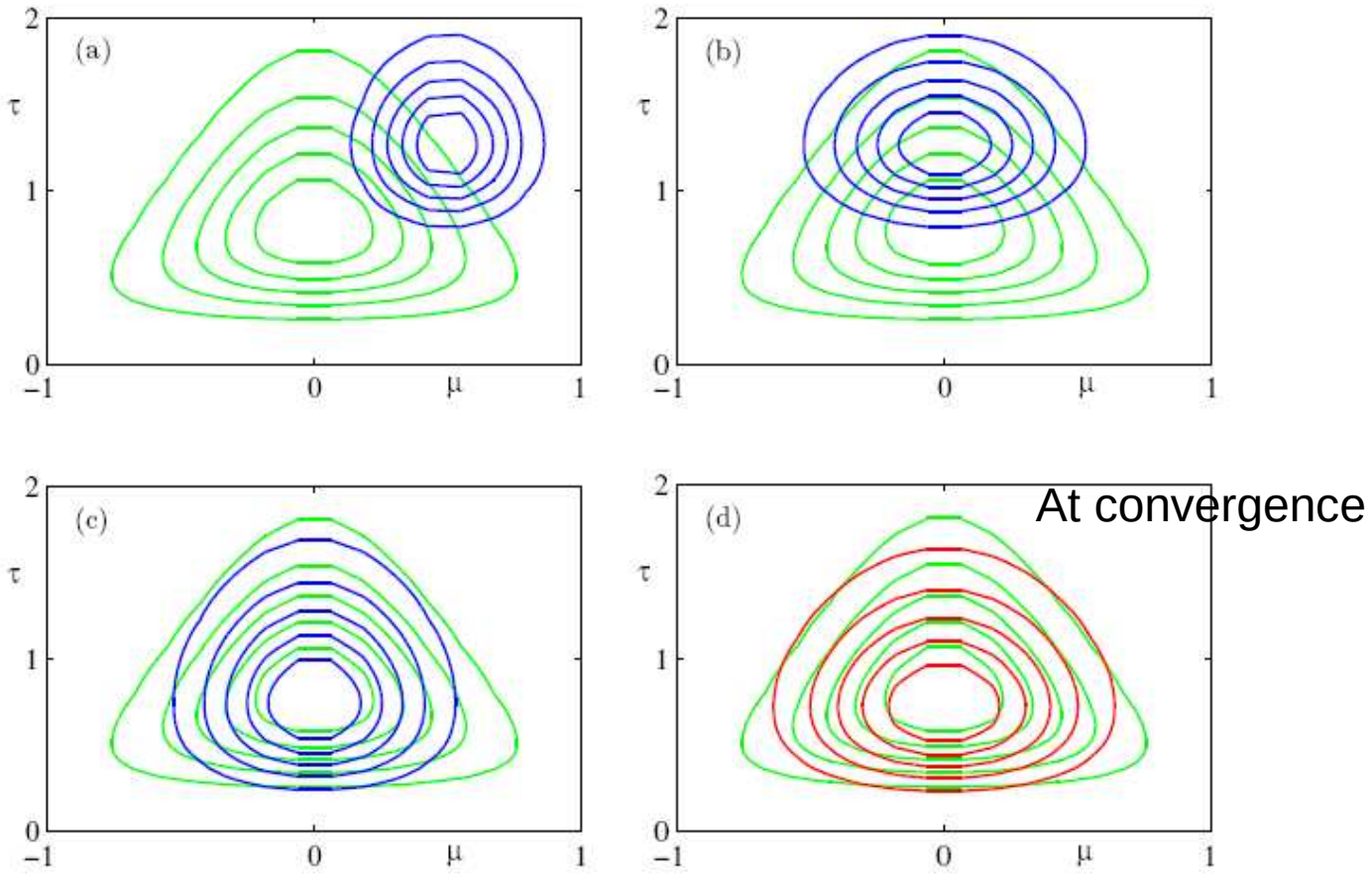
Gamma

$$\begin{aligned} \ln q_\tau^\star(\tau) &= \mathbb{E}_\mu\left[\ln p(\mathcal{D}|\mu,\tau) + \ln p(\mu|\tau)\right] + \ln p(\tau) + \text{const.} \\ &= (a_0-1)\ln\tau - b_0\tau + \frac{N+1}{2}\ln\tau \\ &\quad -\frac{\tau}{2}\mathbb{E}_\mu\left[\sum_{n=1}^{N}(x_n-\mu)^2 + \lambda_0(\mu-\mu_0)^2\right] + \text{const.} \end{aligned}$$

$$\begin{aligned} a_N &= a_0 + \frac{N+1}{2} \\ b_N &= b_0 + \frac{1}{2}\mathbb{E}_\mu\left[\sum_n (x_n-\mu)^2 + \lambda_0(\mu-\mu_0)^2\right]. \end{aligned}$$

Bishop p471

VB for univariate Gaussian



At convergence

Green = exact posterior (NormalGamma), blue = factorized approximation

VB for mixtures of Gaussians

Inference

$$q(\mathbf{Z}, \pi, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\mathbf{Z}) q(\pi, \boldsymbol{\mu}, \boldsymbol{\Lambda}).$$

$$\ln q^\star(\mathbf{Z}) = \mathbb{E}_{\pi, \boldsymbol{\mu}, \boldsymbol{\Lambda}}[\ln p(\mathbf{X}, \mathbf{Z}, \pi, \boldsymbol{\mu}, \boldsymbol{\Lambda})] + \text{const.}$$

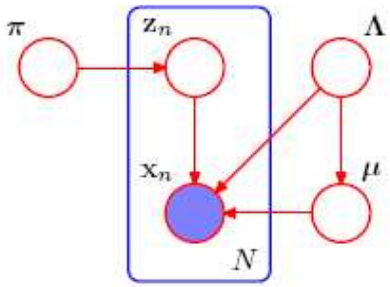
$$\ln q^\star(\mathbf{Z}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \ln \rho_{nk} + \text{const.}$$

$$\begin{aligned} \ln \rho_{nk} &= \mathbb{E}[\ln \pi_k] + \frac{1}{2}\mathbb{E}\left[\ln |\boldsymbol{\Lambda}_k|\right] - \frac{D}{2}\ln(2\pi) \\ &\quad - \frac{1}{2}\mathbb{E}_{\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k}\left[(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Lambda}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)\right] \end{aligned}$$

$$q^\star(\mathbf{Z}) \propto \prod_{n=1}^{N} \prod_{k=1}^{K} \rho_{nk}^{z_{nk}}.$$

Multinomial (soft responsibilities), as in EM,
except we used expected parameters rather than plug-in

Model



$$p(\mathbf{X}, \mathbf{Z}, \pi, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\mathbf{Z}|\pi) p(\pi) p(\boldsymbol{\mu}, \boldsymbol{\Lambda})$$

$$p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}\right)^{z_{nk}}$$

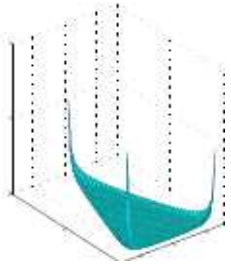
$$p(\mathbf{Z}|\pi) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}}.$$

$$p(\pi) = \mathrm{Dir}(\pi|\boldsymbol{\alpha}_0) = C(\boldsymbol{\alpha}_0) \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1}$$

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{\mu}_k | \mathbf{m}_0, (\beta_0 \boldsymbol{\Lambda}_k)^{-1}\right) \mathcal{W}(\boldsymbol{\Lambda}_k | \mathbf{W}_0, \nu_0)$$

Automatic model selection

- Recall $\pi \sim \mathrm{Dir}(\alpha)$. If $\alpha << 1$, we prefers skewed π and hence sparse z.



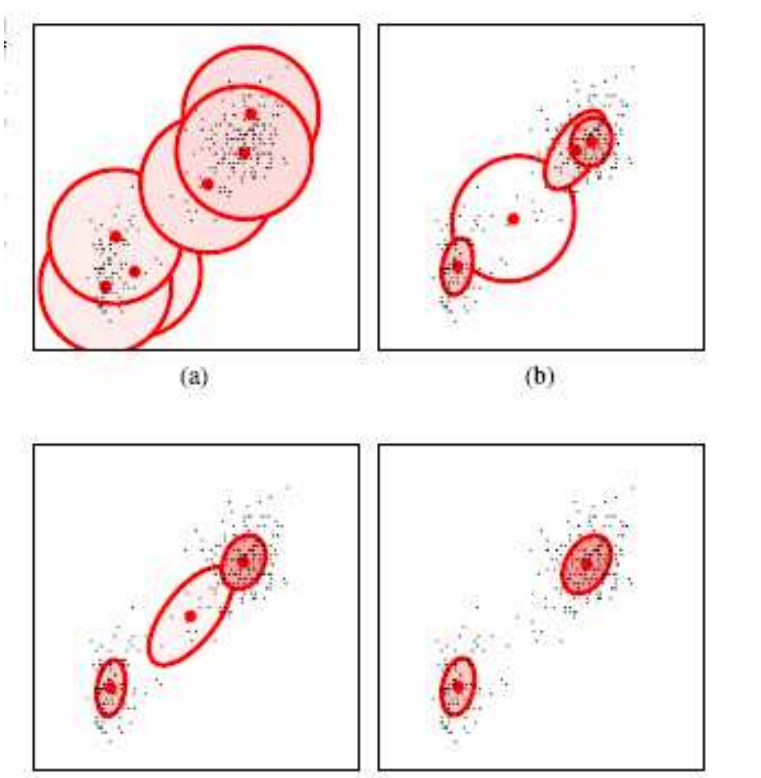
- MAP estimate from regular EM is

$$\hat{\pi}_k = \frac{\sum_n r_{nk} + \alpha_k - 1}{\sum_k (r_{nk} + \alpha_k - 1)} = \frac{N_k + \alpha - 1}{N + K\alpha - K}$$

- Posterior mean estimate from VB is

$$\hat{\pi}_k = \frac{\sum_n r_{nk} + \alpha_k}{\sum_k (r_{nk} + \alpha_k)} = \frac{N_k + \alpha}{N + K\alpha} \rightarrow \frac{\alpha}{N + K\alpha} \rightarrow 0$$

Selecting K with one run of VB



Variational message passing

- Consider a DAG model

$$p(\mathbf{x}) = \prod_i p(\mathbf{x}_i|\mathrm{pa}_i)$$

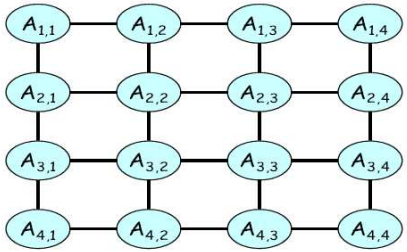
- The mean field equations are

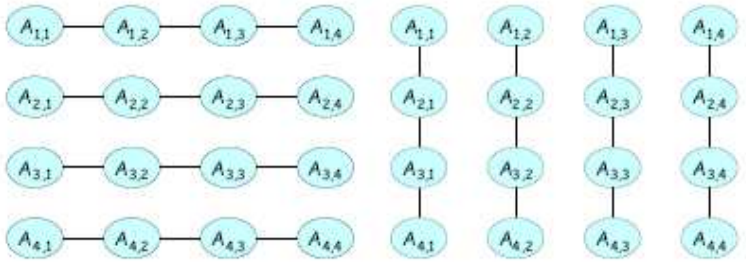
$$\ln q_j^*(\mathbf{x}_j) = \mathbf{E}_{i \neq j}\left[\sum_i \ln p(\mathbf{x}_i|\mathrm{pa}_i)\right] + \text{const.}$$

- The only terms that depend on x_j are in x_j's Markov blanket
- If all CPDs have conjugate-exponential form, the VB updates can be converted into a msg passing algorithm
- VIBES software (John Winn)

Structured variational approx

- Rather than assuming Q is fully factorized, we can use any structure for which computing the expectations of ln ϕ_c and the entropy is tractable





$$Q(\mathcal{X}) = \frac{1}{Z_Q} \prod_{j=1}^{J} \psi_j$$

ϕ = model, ψ = approx

Corollary 11.5.13: *If* $Q(\mathcal{X}) = \frac{1}{Z_Q} \prod_j \psi_j$, *then the potential* ψ_j *is a stationary point of the energy functional if and only if:*

$$\psi_j(\boldsymbol{c}_j) \propto \exp\left\{ \mathbb{E}_Q\left[\ln \tilde{P}_\Phi \mid \boldsymbol{c}_j\right] - \sum_{k \neq j} \boldsymbol{E}_Q[\ln \psi_k \mid \boldsymbol{c}_j] \right\}. \qquad (11.59)$$

Stat 521A
Lecture 24

Outline

- Scoring functions for DAGs with hidden vars (19.4.1)
- Structure search (19.4.2)
- Structural EM (19.4.3)
- Inventing hidden variables in DGMs (19.5)

Bayesian score

- Need a way to measure model quality; orthogonal to issue of how we search through space of models
- Bayesian score hard to compute since posterior is an exponential number of modes

$$\text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

$$p(D|G) = \int \prod_m p(\mathbf{o}[m]|\boldsymbol{\theta}, G) p(\boldsymbol{\theta}|G) d\boldsymbol{\theta}$$

$$p(\mathbf{o}[m]|\boldsymbol{\theta}, G) = \sum_{\mathbf{h}} p(\mathbf{o}[m], \mathbf{h}|\boldsymbol{\theta}, G)$$

- Approximations: asymptotic, variational, MCMC

Chib's candidate method

- Approximate p(D|G) using output of a standard MCMC run. For any θ (eg MAP) compute

$$P(\mathcal{D} \mid \mathcal{G}) = \frac{P(\mathcal{D} \mid \theta, \mathcal{G}) P(\theta \mid \mathcal{G})}{P(\theta \mid \mathcal{D}, \mathcal{G})}.$$

- Requires that p(θ|D,G) cover chosen θ.
- This requires that MCMC mix over all posterior modes, even if symmetrical. If not, it will underestimate p(D|G). See rejected letter to editor by Radford Neal.*

* http://www.cs.utoronto.ca/~radford/ftp/chib-letter.pdf

RJMCMC

- Instead of doing discrete search, and integrating out params at each point, let us jointly sample in graph and param space
- Since the size of the cts space is changing, we need to use a change of measure when we move between dimensionalities
- This results in reversible jump MCMC
- Getting it working is delicate…

Laplace approximation

Box 19.F Concept: Laplace Approximation. *The Laplace approximation can be applied to any function of the form* $f(\boldsymbol{w}) = e^{g(\boldsymbol{w})}$ *for some vector* $\boldsymbol{w}$. *Our task is to compute the integral*

$$F = \int f(\boldsymbol{w}) d\boldsymbol{w}$$

Using Taylor's expansion, we can expand an approximation of g *around a point* $\boldsymbol{w}_0$

$$g(\boldsymbol{w}) \approx g(\boldsymbol{w}_0) + \left[\frac{\partial g(\boldsymbol{w})}{\partial x_i}\right] |_{\boldsymbol{w}=\boldsymbol{w}_0} (\boldsymbol{w} - \boldsymbol{w}_0) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \left[\frac{\partial \partial g(\boldsymbol{w})}{\partial x_i \partial x_j}\right] |_{\boldsymbol{w}=\boldsymbol{w}_0} (\boldsymbol{w} - \boldsymbol{w}_0).$$

where $\left[\frac{\partial g(\boldsymbol{w})}{\partial x_i}\right] |_{\boldsymbol{w}=\boldsymbol{w}_0}$ *denotes the vector of first derivatives and* $\left[\frac{\partial \partial g(\boldsymbol{w})}{\partial x_i \partial x_j}\right] |_{\boldsymbol{w}=\boldsymbol{w}_0}$ *denotes the* Hessian *the matrix of second derivatives.*

If $\boldsymbol{w}_0$ *is the maximum of* $g(\boldsymbol{w})$, *then the second term disappears. We now set*

$$\boldsymbol{C} = -\left[\frac{\partial^2 g(\boldsymbol{w})}{\partial x_i \partial x_j}\right] |_{\boldsymbol{w}=\boldsymbol{w}_0}$$

to be the negative of the matrix of second derivatives of $g(\boldsymbol{w})$ *at* $\boldsymbol{w}_0$. *Since* $\boldsymbol{w}_0$ *is a maximum, this matrix is positive semi-definitive. Thus, we get the approximation*

$$g(\boldsymbol{w}) \approx g(\boldsymbol{w}_0) - \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \boldsymbol{C} (\boldsymbol{w} - \boldsymbol{w}_0).$$

Plugging this approximation into the definition of $f(\boldsymbol{x})$, *we can write*

$$\int f(\boldsymbol{w}) d\boldsymbol{w} \approx f(\boldsymbol{w}_0) \int e^{-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}_0)^T \boldsymbol{C} (\boldsymbol{w}-\boldsymbol{w}_0)} d\boldsymbol{w}.$$

The integral is identical to the integral of an unnormalized Gaussian distribution with covariance matrix $\Sigma = C^{-1}$. *We can therefore solve this integral analytically and obtain:*

$$\int f(\boldsymbol{w}) d\boldsymbol{w} \approx f(\boldsymbol{w}_0) |\boldsymbol{C}|^{-\frac{1}{2}} (2\pi)^{\frac{1}{2}\dim(\boldsymbol{C})}$$

where $\dim(C)$ *is the dimension of the matrix* $\boldsymbol{C}$.

Laplace approximation cont'd

- Let g(w) = log p(D,w|G).
- Laplace approximation to p(D,G) is

$$\text{score}_{Laplace}(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{G}) + \log P(\mathcal{D} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}) + \frac{\dim(C)}{2} \log 2\pi - \frac{1}{2} \log |C|,$$

C is negative Hessian: requires inference on xi, xj, ui, uj

$$- \left. \frac{\partial^2 \log P(\mathcal{D} \mid \theta, \mathcal{G})}{\partial \theta_{x_i|\mathbf{u}_i} \partial \theta_{x_j|\mathbf{u}_j}} \right|_{\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} = - \sum_m \left. \frac{\partial^2 \log P(\boldsymbol{o}[m] \mid \theta, \mathcal{G})}{\partial \theta_{x_i|\mathbf{u}_i} \partial \theta_{x_j|\mathbf{u}_j}} \right|_{\tilde{\boldsymbol{\theta}}_{\mathcal{G}}},$$

BIC score

- BIC is the limit of Laplace as M->inf.

$$C = \sum_{m=1}^{M} C_m \qquad C = M\frac{1}{M}\sum_{m=1}^{M} C_m.$$

$$\det(C) = M^{\dim(C)} \det\left(\frac{1}{M}\sum_{m=1}^{M} C_m\right) \approx M^{\dim(C)} \det\left(E_{P^*}[C_o]\right).$$

$$\log\det(C) \approx \dim(C)\log M + \log\det\left(E_{P^*}[C_o]\right).$$

Theorem 19.4.1: *As $M \to \infty$, we have that:*

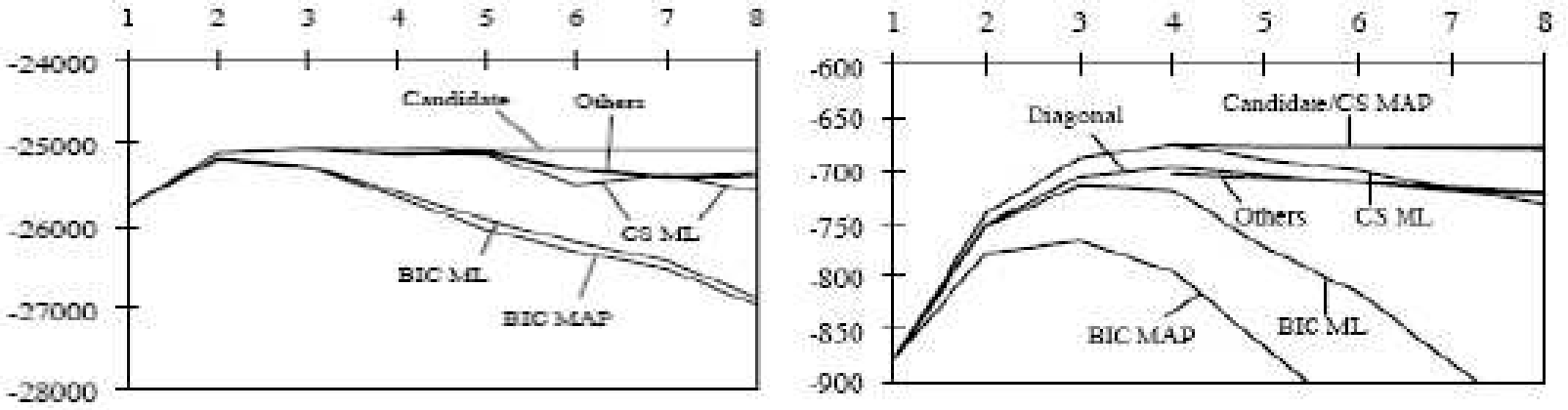
$$\text{score}_{Laplace}(\mathcal{G} : \mathcal{D}) = \text{score}_{BIC}(\mathcal{G} : \mathcal{D}) + O(1)$$

where $\text{score}_{BIC}(\mathcal{G} : \mathcal{D})$ *is the BIC score*

$$\text{score}_{BIC}(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}) - \frac{\log M}{2}\text{Dim}[\mathcal{G}] + \log P(\mathcal{G}) + \log P(\tilde{\theta}_{\mathcal{G}} \mid \mathcal{G}).$$

Cheeseman-Stutz approximation

- CS approx to log p(D|G) is more accurate than BIC, yet faster than Laplace
- Matt Beal's thesis proves CS is a lower bound
- Example: we plot log p(D|K) vs K for a mixture of Bernoullis for different methods; 'candidate' is a 'gold standard' MCMC method



CS approx

- Idea 1: If D* is complete, p(D*|G) just relies on sufficient statistics, so use ESS instead

$$P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_\mathcal{G}} \mid \mathcal{G}) = \int p(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_\mathcal{G}} \mid \boldsymbol{\theta}, \mathcal{G}) P(\boldsymbol{\theta} \mid \mathcal{G}) d\boldsymbol{\theta}$$

- Unfortunately this does not work well, since it sums over 1 (imputed) dataset whereas p(D|G) sums over an exponential number

$$P(\mathcal{D} \mid \mathcal{G}) = \int \sum_{\mathcal{H}} p(\mathcal{D}, \mathcal{H} \mid \boldsymbol{\theta}, \mathcal{G}) P(\boldsymbol{\theta} \mid \mathcal{G}) d\boldsymbol{\theta} = \sum_{\mathcal{H}} \int p(\mathcal{D}, \mathcal{H} \mid \boldsymbol{\theta}, \mathcal{G}) P(\boldsymbol{\theta} \mid \mathcal{G}), d\boldsymbol{\theta}.$$

- Idea 2: add an approximate correction term

$$\log P(\mathcal{D} \mid \mathcal{G}) \quad = \quad \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_\mathcal{G}} \mid \mathcal{G}) + \log P(\mathcal{D} \mid \mathcal{G}) - \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_\mathcal{G}} \mid \mathcal{G})$$

Approximate with BIC

CS approx

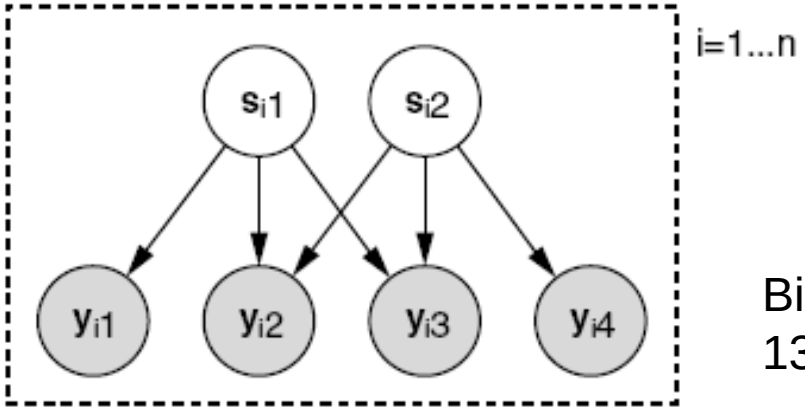
$$\begin{aligned}\log P(\mathcal{D} \mid \mathcal{G}) - \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \mathcal{G}) &\approx \left[\log P(\mathcal{D} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}) - \frac{1}{2}\mathrm{Dim}[\tilde{\theta}_{\mathcal{G}}] \log M\right] \\ &\quad - \left[\log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}) - \frac{1}{2}\mathrm{Dim}[\tilde{\theta}_{\mathcal{G}}] \log M\right] \\ &= \log P(\mathcal{D} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}) - \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}).\end{aligned}$$

$$\begin{aligned}\log P(\mathcal{D} \mid \mathcal{G}) &= \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \mathcal{G}) + \log P(\mathcal{D} \mid \mathcal{G}) - \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \mathcal{G}) \\ &\approx \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \mathcal{G}) + \log P(\mathcal{D} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}) - \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}).\end{aligned}$$

$$\mathrm{score}_{CS}(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{G}) + \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \mathcal{G}) + \log P(\mathcal{D} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G}) - \log P(\mathcal{D}^*_{\mathcal{G},\tilde{\boldsymbol{\theta}}_{\mathcal{G}}} \mid \tilde{\theta}_{\mathcal{G}}, \mathcal{G})$$

Variational lower bound

EM for MAP estimation	Variational Bayesian EM
Goal: maximise $p(\theta \mid \mathbf{y}, m)$ w.r.t. θ	**Goal:** lower bound $p(\mathbf{y} \mid m)$
E Step: compute $q_{\mathbf{x}}^{(t+1)}(\mathbf{x}) = p(\mathbf{x} \mid \mathbf{y}, \theta^{(t)})$	**VBE Step:** compute $q_{\mathbf{x}}^{(t+1)}(\mathbf{x}) = p(\mathbf{x} \mid \mathbf{y}, \overline{\phi}^{(t)})$
M Step: $\theta^{(t+1)} = \arg\max_{\theta} \int d\mathbf{x}\; q_{\mathbf{x}}^{(t+1)}(\mathbf{x}) \ln p(\mathbf{x}, \mathbf{y}, \theta)$	**VBM Step:** $q_{\theta}^{(t+1)}(\theta) \propto \exp \int d\mathbf{x}\; q_{\mathbf{x}}^{(t+1)}(\mathbf{x}) \ln p(\mathbf{x}, \mathbf{y}, \theta)$

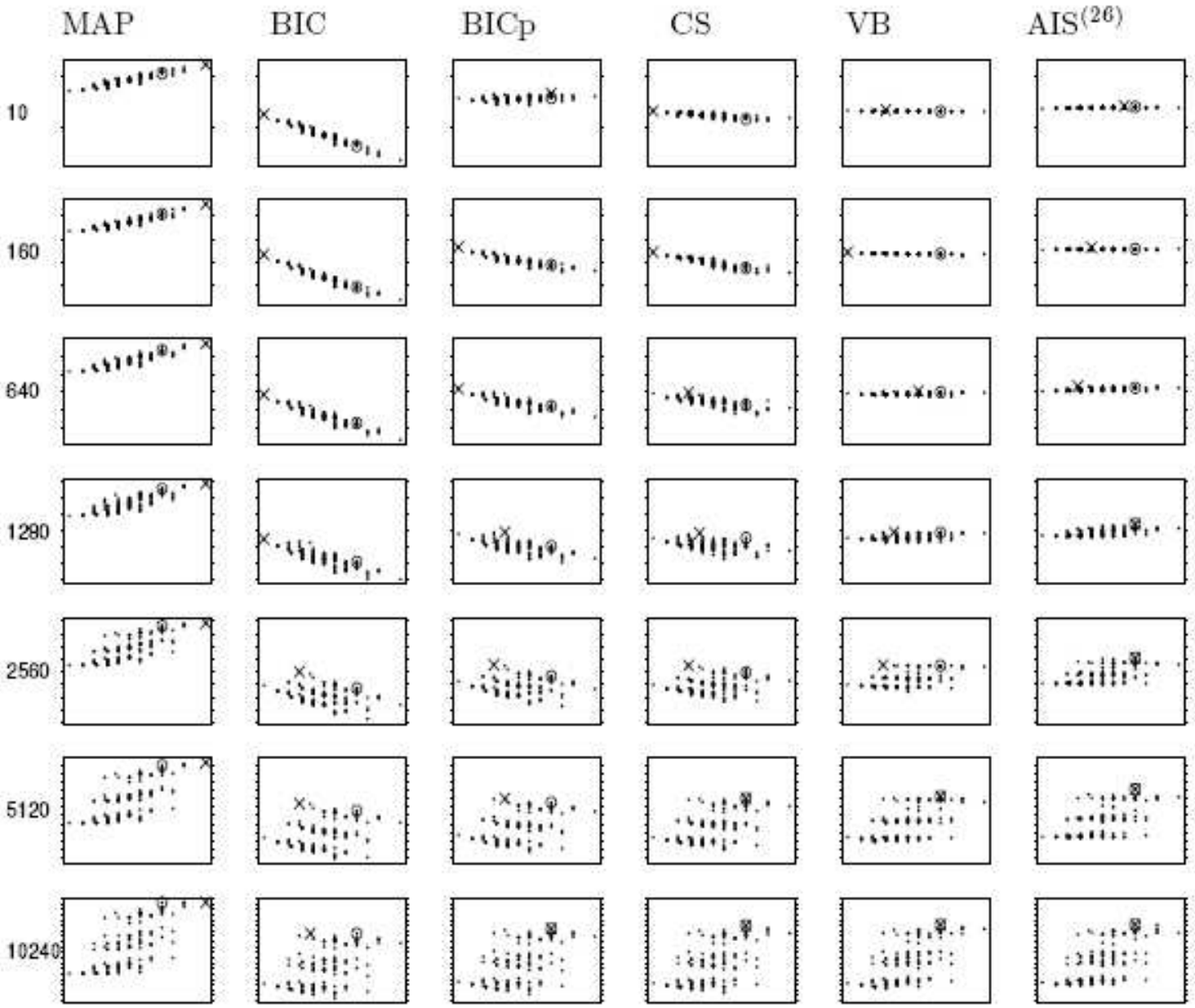


VB provably tighter lower Bound than CS

Binary hidden nodes, 5-ary obs nodes
136 distinct DAGs

Beal, M.J. and Ghahramani, Z.
Variational Bayesian Learning of Directed Graphical Models with Hidden Variables
Bayesian Analysis **1**(4), 2006.

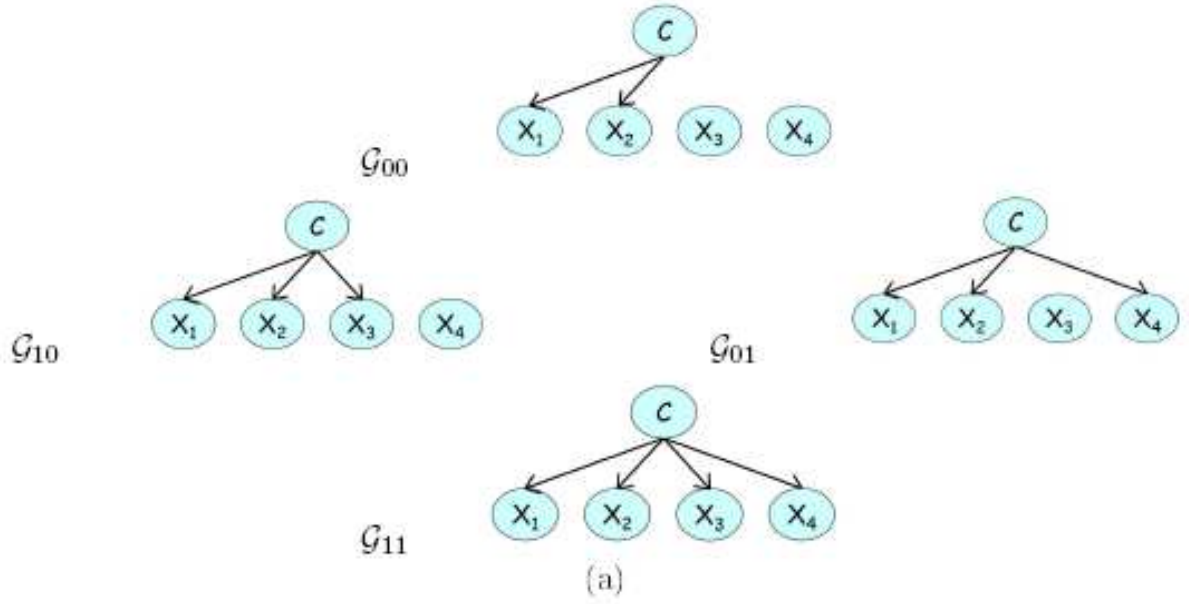
Log p(D|G) vs dof(G)



Structure search

- P(D|G) does not factorize across families, unlike the fully observed case
- Cannot find (easily) optimal tree or optimal DAG given ordering.
- For local search, evaluating score of neighbors is expensive – score does not decompose, so need to find MAP estimate for each graph just to compute its BIC score

Illustration of non-decomposability



(a)

{1,2} and 3: weak corr
3 and 4: strong corr

Network	ΔLL	ΔCS
$\mathcal{G}_{10}$ (add $C \rightarrow X_3$)	+3	−0.4
$\mathcal{G}_{01}$ (add $C \rightarrow X_4$)	+10.6	+7.2
$\mathcal{G}_{11}$ (add $C \rightarrow X_3$, $C \rightarrow X_4$)	+24.1	+17.4

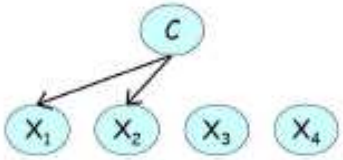
Structural EM

- Given current graph Gt and MAP params theta(t), compute ESS for all possible families (potentially in a lazy fashion – may need out-of-clq queries)
- Evaluate BIC score for G(t+1) using ESS|G(t)
- Thm: increasing expected BIC score increases true BIC score

Theorem 19.4.3: *Let $\mathcal{G}_0$ be a graph structure and $\tilde{\boldsymbol{\theta}}_0$ be the MAP parameters for $\mathcal{G}_0$ given a dataset $\mathcal{D}$. Then for any graph structure $\mathcal{G}$:*

$$\text{score}_{BIC}(\mathcal{G} : \mathcal{D}^*_{\mathcal{G}_0,\tilde{\boldsymbol{\theta}}_0}) - \text{score}_{BIC}(\mathcal{G}_0 : \mathcal{D}^*_{\mathcal{G}_0,\tilde{\boldsymbol{\theta}}_0}) \leq \text{score}_{BIC}(\mathcal{G} : \mathcal{D}) - \text{score}_{BIC}(\mathcal{G}_0 : \mathcal{D}).$$

Sparse mixture model



$\mathcal{G}_{00}$

- Run parameter estimation (such as EM or gradient ascent) to learn parameters $\tilde{\theta}_t$ for $\mathcal{G}_t$.
- Construct a new structure $\mathcal{G}_{t+1}$ so that $\mathcal{G}_{t+1}$ contains the edge $C \rightarrow X_i$ if

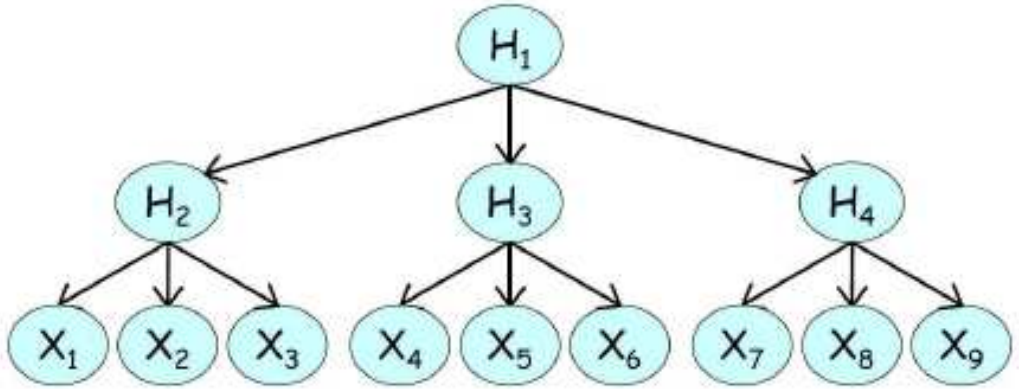
$$\text{FamScore}(X_i, \{C\} \ : \ \mathcal{D}^*_{\mathcal{G}_t, \tilde{\boldsymbol{\theta}}_t}) > \text{FamScore}(X_i, \emptyset \ : \ \mathcal{D}^*_{\mathcal{G}_t, \tilde{\boldsymbol{\theta}}_t}).$$

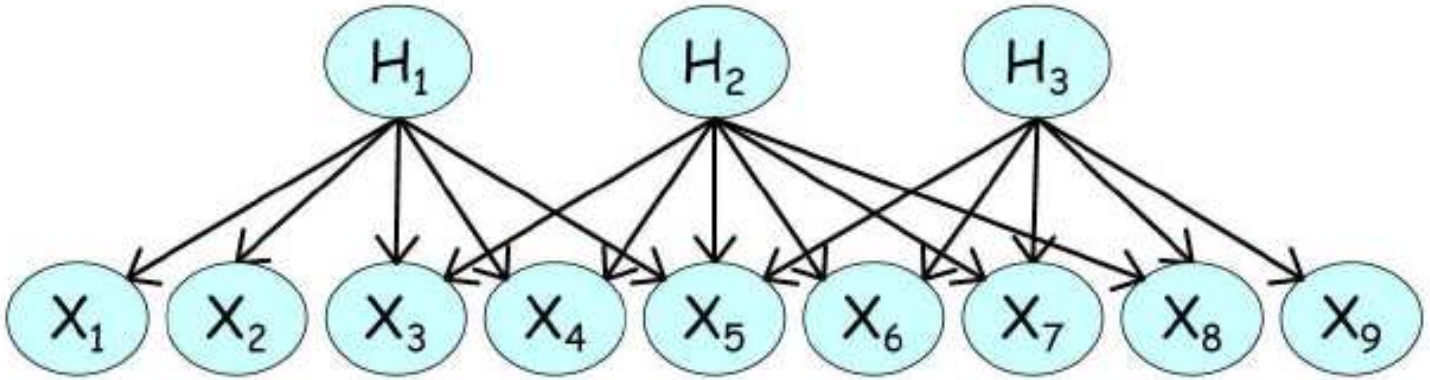
$$\begin{aligned} \bar{M}_{\mathcal{D}^*_{\mathcal{G}_t, \tilde{\boldsymbol{\theta}}_t}}[x_i, c] &= \sum_m P(C[m] = c, X_i[m] = x_i \mid \boldsymbol{o}[m], \mathcal{G}_t, \tilde{\theta}_t) \\ &= \sum_{m, X_i[m] = x_i} P(C[m] = c \mid \boldsymbol{o}[m], \mathcal{G}_t, \tilde{\boldsymbol{\theta}}_t). \end{aligned}$$

Initialization: if start from no children, will never add any! So start from all Children or random subset.

Inventing hidden variables

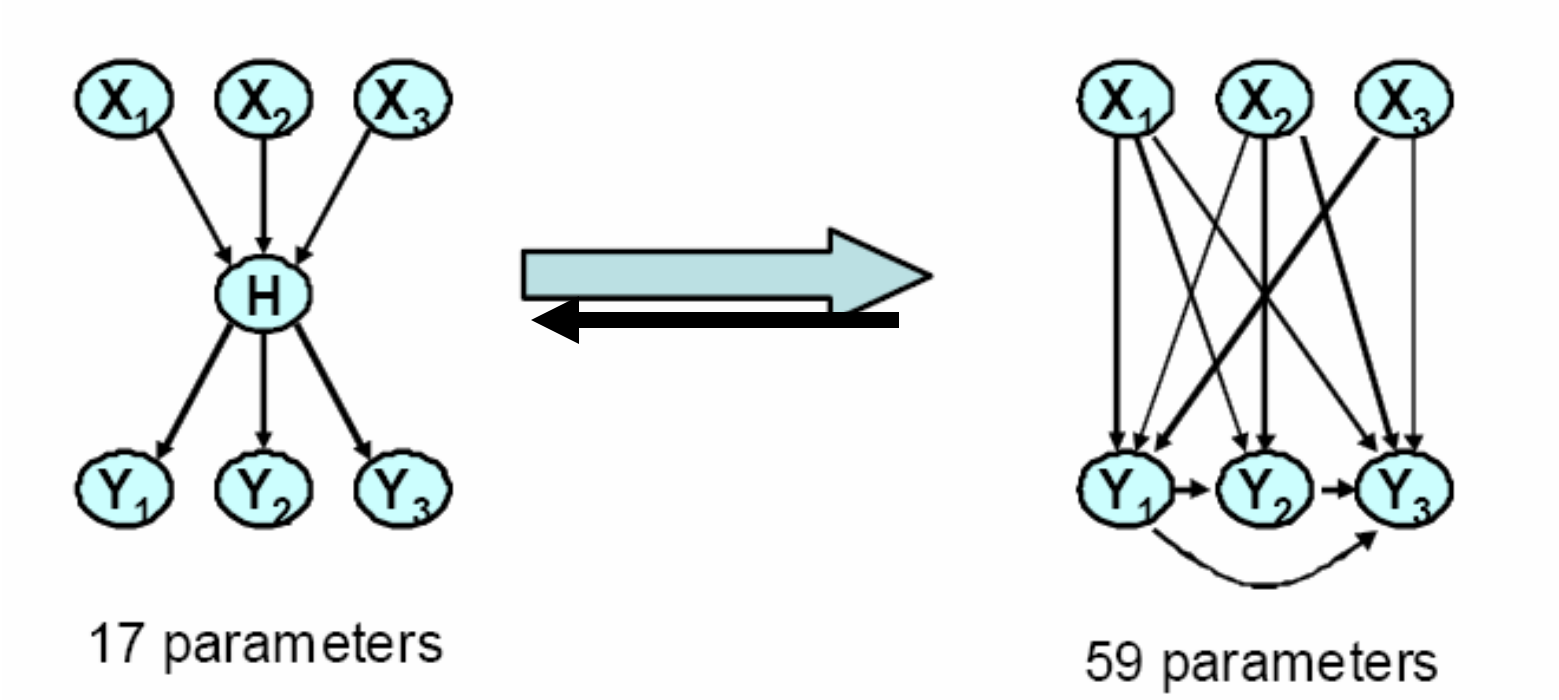
- Can add hidden variables in ‘canonical’ places





Structural signatures

- Can learn structure with no hidden vars, then look for ‘semi-cliques’.
- Unfortunately original model discourages nodes with high fan-in.

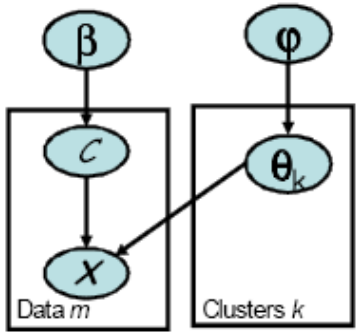


Can also look for signatures in the data - eg FCI* algorithm

Cardinality of hidden nodes

- Need to choose number of states.
- Can use an “infinite” number using Dirichlet processes.
- Let us first consider DP mixture models.

Marginalizing out θ



Collapsed Gibbs sampling
Cf DP mixtures

$$P(\boldsymbol{\lambda} \mid \boldsymbol{c}) = Dirichlet(\alpha_0/K + |I_1(\boldsymbol{c})|, \ldots, \alpha_0/K + |I_K(\boldsymbol{c})|).$$

$$P(\boldsymbol{\theta}_k \mid \boldsymbol{c}, \mathcal{D}, \phi) = Q(\boldsymbol{\theta}_k \mid \mathcal{D}_{I_k(\boldsymbol{c})}, \phi) \propto P(\boldsymbol{\theta}_k \mid \phi) \prod_{m \in I_k(\boldsymbol{c})} P(\boldsymbol{x}[m] \mid \boldsymbol{\theta}_k),$$

$$P(C[m'] = k \mid \boldsymbol{c}_{-m'}, \mathcal{D}, \phi) \propto P(C[m'] = k \mid \boldsymbol{\lambda}, \boldsymbol{c}_{-m'}) P(\boldsymbol{x}[m'] \mid C[m'] = k, \boldsymbol{x}[I_k(\boldsymbol{c}_{-m'})], \phi).$$

$$P(C[m'] = k \mid \boldsymbol{c}_{-m'}, \mathcal{D}, \phi) \propto (|I_k(\boldsymbol{c}_{-m'})| + \alpha_0/K) Q(\boldsymbol{X} \mid \mathcal{D}_{I_k(\boldsymbol{c}_{-m'})}, \phi).$$

O(M K) per iter

DP mixture model (p865)

- Identity of clusters does not matter. Let σ={I1,...,IL} be a partition, Ic=cases in cluster c. For case m', either join existing cluster or create new one O(ML) per iter

$$P(I \leftarrow I \cup \{m'\} \mid \sigma_{-m'}, \mathcal{D}, \phi) \propto \left(|I| + \frac{\alpha_0}{K}\right) Q(\boldsymbol{x}[m'] \mid \mathcal{D}_I, \phi)$$
$$P(\sigma \leftarrow \sigma \cup \{\{m'\}\} \mid \sigma_{-m'}, \mathcal{D}, \phi) \propto (K - L)\frac{\alpha_0}{K} Q(\boldsymbol{x}[m'] \mid \phi),$$

- Now let K->inf.

$$P(I \leftarrow I \cup \{m'\} \mid \sigma_{-m'}, \mathcal{D}, \phi) \propto |I| \cdot Q(\boldsymbol{x}[m'] \mid \mathcal{D}_I, \phi)$$
$$P(\sigma \leftarrow \sigma \cup \{\{m'\}\} \mid \sigma_{-m'}, \mathcal{D}, \phi) \propto \alpha_0 \cdot Q(\boldsymbol{x}[m'] \mid \phi).$$

- More likely to join a cluster if it is already crowded.
- Chinese Restaurant process.

Stat 521A
Lecture 25

Outline

- MAP param estimation for UGMs (20.1-20.4)
- Learning using approximate inference (20.5)
- Alternative objectives (20.6)

Likelihood fn for UGMs

- Log-linear model

$$P(X_1, \ldots, X_n : \theta) = \frac{1}{Z(\theta)} \exp\left\{\sum_{i=1}^{k} \theta_i f_i[D_i]\right\}.$$

$$\ln Z(\theta) = \ln \sum_{\xi} \exp\left\{\sum_{i} \theta_i f_i[\xi]\right\}.$$

Concave

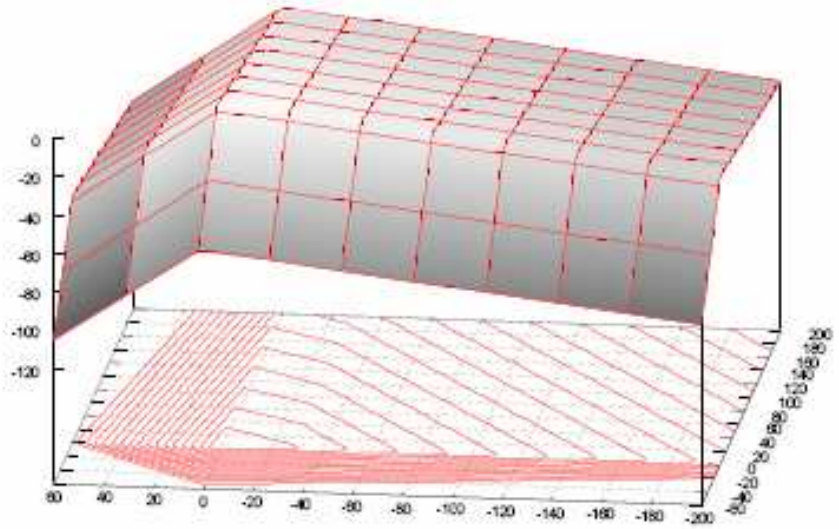


Figure 20.1 **Log-likelihood surface for the Markov network A—B—C, as a function of $\ln \phi_1[a^1, b^1]$ (x-axis) and $\ln \phi_2[b^1, c^1]$ (y-axis)**: all other parameters in both potentials are set to 1. The data set $\mathcal{D}$ has $M = 100$ instances, for which $M[a^1, b^1] = 90$ and $M[b^1, c^1] = 15$. (The other sufficient statistics are irrelevant, as all of the other log-parameters are 0.)

LogZ: first deriv

Proposition 20.2.3: *Let Φ be a set of features. Then,*

$$\begin{aligned}\frac{\partial}{\partial\theta_i}\ln Z(\theta) &= \boldsymbol{E}_{\boldsymbol{\theta}}[f_i] \\ \frac{\partial^2}{\partial\theta_i\partial\theta_j}\ln Z(\theta) &= \boldsymbol{C}ov_{\boldsymbol{\theta}}[f_i; f_j],\end{aligned}$$

where $\boldsymbol{E}_{\boldsymbol{\theta}}[f_i]$ is a shorthand for $\boldsymbol{E}_{P(\mathcal{X};\boldsymbol{\theta})}[f_i]$.

$$\begin{aligned}\frac{\partial}{\partial\theta_i}\ln Z(\theta) &= \frac{1}{Z(\theta)}\sum_{\xi}\frac{\partial}{\partial\theta_i}\exp\left\{\sum_j \theta_j f_j[\xi]\right\} \\ &= \frac{1}{Z(\theta)}\sum_{\xi} f_i[\xi]\exp\left\{\sum_j \theta_j f_j[\xi]\right\} \\ &= \boldsymbol{E}_{\boldsymbol{\theta}}[f_i].\end{aligned}$$

logZ: second deriv

$$
\begin{aligned}
\frac{\partial^2}{\partial\theta_j\partial\theta_i}\ln Z(\boldsymbol{\theta}) &= \frac{\partial}{\partial\theta_j}\left[\frac{1}{Z(\boldsymbol{\theta})}\sum_{\xi} f_i[\xi]\exp\left\{\sum_k \theta_k f_k[\xi]\right\}\right] \\
&= -\frac{1}{Z(\boldsymbol{\theta})^2}\left(\frac{\partial}{\partial\theta_j}Z(\boldsymbol{\theta})\right)\sum_{\xi} f_i[\xi]\exp\left\{\sum_k \theta_k f_k[\xi]\right\} \\
&\quad + \frac{1}{Z(\boldsymbol{\theta})}\sum_{\xi} f_i[\xi]f_j[\xi]\exp\left\{\sum_k \theta_k f_k[\xi]\right\} \\
&= -\frac{1}{Z(\boldsymbol{\theta})^2}Z(\boldsymbol{\theta})\boldsymbol{E}_{\boldsymbol{\theta}}[f_i]\sum_{\xi} f_i[\xi]\tilde{P}(\xi:\boldsymbol{\theta}) \\
&\quad + \frac{1}{Z(\boldsymbol{\theta})}\sum_{\xi} f_i[\xi]f_j[\xi]\tilde{P}(\xi:\boldsymbol{\theta}) \\
&= \boldsymbol{E}_{\boldsymbol{\theta}}[f_i]\sum_{\xi} f_i[\xi]P(\xi:\boldsymbol{\theta}) \\
&\quad + \sum_{\xi} f_i[\xi]f_j[\xi]P(\xi:\boldsymbol{\theta}) \\
&= \boldsymbol{E}_{\boldsymbol{\theta}}[f_i f_j] - \boldsymbol{E}_{\boldsymbol{\theta}}[f_i]\boldsymbol{E}_{\boldsymbol{\theta}}[f_j] \\
&= \boldsymbol{Cov}_{\boldsymbol{\theta}}[f_i; f_j].
\end{aligned}
$$

Finding the MLE

At optimum, model moments = empirical moments

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\boldsymbol{\theta} : \mathcal{D}) = \boldsymbol{E}_{\mathcal{D}}[f_i[\mathcal{X}]] - \boldsymbol{E}_{\boldsymbol{\theta}}[f_i]. \tag{20.4}$$

This analysis provides us with a precise characterization of the maximum likelihood parameters $\hat{\boldsymbol{\theta}}$:

Theorem 20.3.1: *Let Φ be a set of features. Then, $\boldsymbol{\theta}$ is a maximal likelihood parameter assignment if and only if $\boldsymbol{E}_{\mathcal{D}}[f_i[\mathcal{X}]] = \boldsymbol{E}_{\boldsymbol{\theta}}[f_i]$ for all i.*

Must perform inference once per gradient

Just do gradient based optimization, eg stochastic gradient descent.
Expensive to compute Hessian explicitly, so use Quasi-Newton.

$$\frac{\partial}{\partial \theta_i \partial \theta_j} \ell(\boldsymbol{\theta} : \mathcal{D}) = -M \mathbb{C}\mathrm{ov}_{\boldsymbol{\theta}}[f_i; f_j].$$

CRFs

- Conditional density models

$$\ell_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \ln P(\boldsymbol{y}[1, \ldots, M] \mid \boldsymbol{x}[1, \ldots, M], \boldsymbol{\theta}) = \sum_{m=1}^{M} \ln P(\boldsymbol{y}[m] \mid \boldsymbol{x}[m], \boldsymbol{\theta}).$$

$$\frac{\partial}{\partial \theta_i} \ell_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \left[f_i[\boldsymbol{y}[m], \boldsymbol{x}[m]] - \boldsymbol{E}_{\boldsymbol{\theta}}[f_i \mid \boldsymbol{x}[m]] \right].$$

Must perform inference M times per gradient

MRFs with hidden variables

- Must perform inference M times per gradient

$$\frac{1}{M}\ln P(\mathcal{D}\mid\theta) = \frac{1}{M}\ln\left(\sum_{m=1}^{M}\sum_{\boldsymbol{h}[m]} P(\boldsymbol{o}[m],\boldsymbol{h}[m]\mid\theta)\right)$$
$$= \frac{1}{M}\ln\left(\sum_{m=1}^{M}\sum_{\boldsymbol{h}[m]} \tilde{P}(\boldsymbol{o}[m],\boldsymbol{h}[m]\mid\theta)\right) - \ln Z.$$

$$\frac{\partial}{\partial\theta_i}\ln\sum_{\boldsymbol{h}[m]}\tilde{P}(\boldsymbol{o}[m],\boldsymbol{h}[m]\mid\theta) = \boldsymbol{E}_{\boldsymbol{h}[m]\sim P(\mathcal{H}[m]\mid\boldsymbol{o}[m],\boldsymbol{\theta})}[f_i],$$

Proposition 20.3.3: *For a data set $\mathcal{D}$*

$$\frac{\partial}{\partial\theta_i}\frac{1}{M}\ell(\boldsymbol{\theta}:\mathcal{D}) = \frac{1}{M}\left[\sum_{m=1}^{M}\boldsymbol{E}_{\boldsymbol{h}[m]\sim P(\mathcal{H}[m]\mid\boldsymbol{o}[m],\boldsymbol{\theta})}[f_i]\right] - \boldsymbol{E}_{\boldsymbol{\theta}}[f_i].$$

clamped unclamped

CRFs with hidden variables

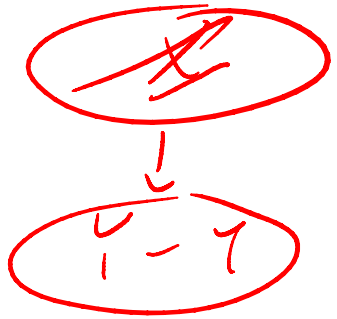
- Training is similar to MRFs with hidden variables, except expectations condition on x_n, so need to be redone for each case

Summary

MRF

$$\nabla = \sum_i f(x_i) - ME_X[f(X)]$$

CRF

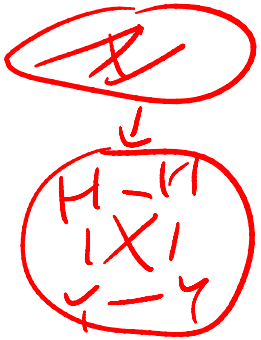


$$\nabla = \sum_i f(x_i, y_i) - \sum_i E_Y[f(x_i, Y)]$$

MRF + H

$$\nabla = \sum_i E_H f(H_i, x_i) - ME_{H,X}[f(H, X)]$$

CRF + H



$$\nabla = \sum_i E_H f(x_i, y_i, H) - \sum_i E_{H,Y}[f(x_i, Y, H)]$$

ML and MaxEnt

- MLE in the expfam is equivalent to MaxEnt subject to moment constraints

Maximum-Entropy

Find $Q(\mathcal{X})$
that maximize $H_Q(\mathcal{X})$

subject to
$$E_Q[f_i] = E_{\mathcal{D}}[f_i] \quad i = 1, \ldots, k$$

Theorem 20.3.4: *The distribution Q^* is the maximum entropy distribution satisfying Eq. (20.10) if and only if $Q^* = P_{\hat{\boldsymbol{\theta}}}$, where*

$$P_{\hat{\boldsymbol{\theta}}}(\mathcal{X}) = \frac{1}{Z(\hat{\boldsymbol{\theta}})} \exp\left\{\sum_i \hat{\theta}_i f_i[\mathcal{X}]\right\}$$

and $\hat{\boldsymbol{\theta}}$ is the maximum likelihood parameterization relative to $\mathcal{D}$.

Proof

PROOF For notational simplicity, let $P = P_{\hat{\boldsymbol{\theta}}}$. From Theorem 20.3.1, it follows that $\boldsymbol{E}_P[f_i] = \boldsymbol{E}_{\mathcal{D}}[f_i[\mathcal{X}]]$ for $i = 1, \ldots, k$, and hence that P satisfies the constraints of Eq. (20.10). Therefore, to prove that $P = Q^*$, we need only show that $\boldsymbol{H}_P(\mathcal{X}) \geq \boldsymbol{H}_Q(\mathcal{X})$ for all other distributions Q that satisfy these constraints. Consider any such distribution Q.

From Theorem 8.4.1, it follows that:

$$\boldsymbol{H}_P(\mathcal{X}) = -\sum_i \hat{\theta}_i \boldsymbol{E}_P[f_i] + \ln Z(\boldsymbol{\theta}). \tag{20.11}$$

Thus,

$$\begin{aligned}
\boldsymbol{H}_P(\mathcal{X}) - \boldsymbol{H}_Q(\mathcal{X}) &= -\left[\sum_i \theta_i \boldsymbol{E}_P[f_i[\mathcal{X}]]\right] + \ln Z_P - \boldsymbol{E}_Q[-\ln Q(\mathcal{X})] \\
(i) &= -\left[\sum_i \theta_i \boldsymbol{E}_Q[f_i[\mathcal{X}]]\right] + \ln Z_P + \boldsymbol{E}_Q[\ln Q(\mathcal{X})] \\
&= \boldsymbol{E}_Q[-\ln P(\mathcal{X})] + \boldsymbol{E}_Q[\ln Q(\mathcal{X})] \\
&= \mathbb{D}(Q|P) \geq 0,
\end{aligned}$$

where (i) follows from the fact that both $P_{\hat{\boldsymbol{\theta}}}$ and Q satisfy the constraints, so that $\boldsymbol{E}_{P_{\hat{\boldsymbol{\theta}}}}[f_i] = \boldsymbol{E}_Q[f_i]$ for all i.

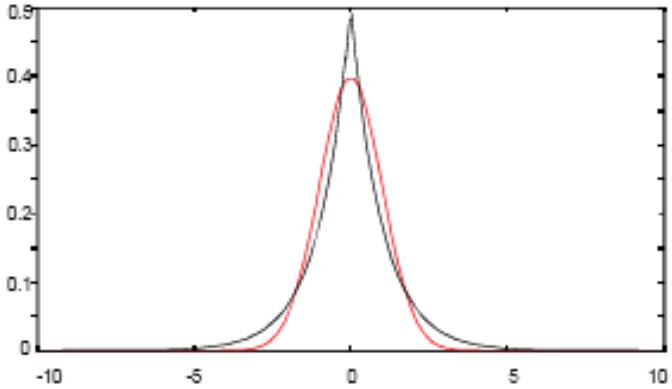
We conclude that $\boldsymbol{H}_{P_{\hat{\boldsymbol{\theta}}}}(\mathcal{X}) \geq \boldsymbol{H}_Q(\mathcal{X})$ with equality if and only if $P_{\hat{\boldsymbol{\theta}}} = Q$. Thus, the maximum entropy distribution Q^* is necessarily equal to $P_{\hat{\boldsymbol{\theta}}}$, proving the result. ■

MAP estimation

- Convex prior + convex likelihood makes objective strictly convex (unique soln)
- Also helps prevent overfitting
- L2 and L1

$$P(\theta \mid \sigma^2) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{\theta_i^2}{2\sigma^2}\right\},$$

$$P_{Laplacian}(\theta \mid \beta) = \frac{1}{2\beta} \exp\left\{-\frac{|\theta|}{\beta}\right\}.$$



$$\ln \frac{P(\xi)}{P(\xi')} = \sum_{i=1}^{k} \theta_i f_i[\xi] - \sum_{i=1}^{k} \theta_i f_i[\xi']$$
$$= \sum_{i=1}^{k} \theta_i (f_i[\xi] - f_i[\xi']).$$

Learning with approximate inference

- Recall that the gradient requires model expectation over the features

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\boldsymbol{\theta} : \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[f_i[\mathcal{X}]] - \mathbb{E}_{\boldsymbol{\theta}}[f_i]. \tag{20.4}$$

- We can use approximate inference to approximate the expectation, but approximate gradients can cause learning to diverge

Pseudo moment matching

- At the optimum, the pseudo marginals must satisfy

$$\mathbb{E}_{\beta_i[\boldsymbol{C}_i]}[f_{\boldsymbol{C}_i}] = \mathbb{E}_{\mathcal{D}}[f_i[\boldsymbol{C}_i]].$$

- Suppose we use tabular features. Then

$$\beta_i[\boldsymbol{c}_i^j] = \hat{P}(\boldsymbol{c}_i^j).$$

- Hence we don't need to run inference. There are multiple potentials that can generate these beliefs. We can uniquely recover one set using (for any ordering i<j)

$$\phi_i \leftarrow \frac{\beta_i}{\mu_{i,j}}.$$

Unified inference and learning

- Pseudo moment matching only works for unconditional, tabular potentials with no tying and no regularizer
- To combine BP with param optimization, we can optimize

Approx-Maximum-Entropy

$$\begin{array}{ll} \textbf{Find} & Q \\ \textbf{that maximize} & \sum_{C_i \in \mathcal{U}} H_{\beta_i}(C_i) - \sum_{(C_i \quad C_j) \in \mathcal{U}} H_{\mu_{i,j}}(S_{i,j}) \\ \\ \textbf{subject to} & \mathbb{E}_{\beta_i}[f_i] = \mathbb{E}_{\mathcal{D}}[f_i] \quad i = 1, \ldots, k \\ & Q \in Local[\mathcal{U}] \end{array}$$

The model parameters theta are the Lagrange multiplers for E[f]
And the messages are the Lagrange multipliers for the local consistency

Example

A	B	C
0	0	0
0	1	0
1	0	0

Tied features

$f_{00}(x,y)=1$ iff $x=y=0$

$f_{11}(x,y)=1$ iff $x=y=1$

$E_0\ f_{00} = I(A_1=B_1) + I(A_1=C_1) + I(B_1=C_1) + \ldots = 5$

Find Q

that maximize $H_{\beta_1}(A,B) + H_{\beta_2}(B,C) + H_{\beta_3}(A,C) - H_{\mu_{1,2}}(B) - H_{\mu_{2,3}}(C) - H_{\mu_{2,3}}(A)$

subject to

$$\sum_i E_{\beta_i}[f_{00}] = 2 \quad (20.17)$$

$$\sum_i E_{\beta_i}[f_{11}] = 0 \quad (20.18)$$

$$\sum_a [\beta_1[a,b]] - \sum_c [\beta_2[b,c]] = 0 \quad (20.19)$$

$$\sum_b [\beta_2[b,c]] - \sum_a [\beta_3[a,c]] = 0 \quad (20.20)$$

$$\sum_c [\beta_3[a,c]] - \sum_b [\beta_1[a,b]] = 0 \quad (20.21)$$

$$\sum_{\mathbf{c}_i} \beta_i[\mathbf{c}_i] = 1 \qquad i = 1,2,3 \quad (20.22)$$

$$\beta_i \geq 0 \qquad i = 1,2,3 \quad (20.23)$$

Double loop algorithm

- Inner loop optimizes δ_{ij} by iterating the fixed point eqns
- Outer loop optimizes θ eg using gradient descent

Approximating Z

- Loglik

$$\ell(\boldsymbol{\theta} : \xi) = \ln \tilde{P}(\xi \mid \boldsymbol{\theta}) - \ln Z(\boldsymbol{\theta})$$

$$\ln \tilde{P}(\xi \mid \boldsymbol{\theta}) - \ln \left(\sum_{\xi'} \tilde{P}(\xi' \mid \boldsymbol{\theta}) \right).$$

- We can approximate the sum in different ways

Pseudolikeliood

- Define

$$P(\xi) = \prod_{j=1}^{n} P(x_j \mid x_1, \ldots, x_{j-1})$$

$$P(\xi) \approx \prod_{j} P(x_j \mid x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$$

$$\ell_{\text{pseudo}}(\boldsymbol{\theta} : \mathcal{D}) = \frac{1}{M} \sum_{m} \sum_{j} \ln P(x_j[m] \mid \boldsymbol{x}_{-j}[m], \boldsymbol{\theta})$$

$$P(x_j \mid \boldsymbol{x}_{-j}) = \frac{P(x_j, \boldsymbol{x}_{-j})}{P(\boldsymbol{x}_{-j})} = \frac{\tilde{P}(x_j, \boldsymbol{x}_{-j})}{\tilde{P}(\boldsymbol{x}_{-j})} = \frac{\tilde{P}(x_j, \boldsymbol{x}_{-j})}{\sum_{x'_j} \tilde{P}(x'_j, \boldsymbol{x}_{-j})}.$$

Gradient of PL

$$\ln P(x_j \mid \boldsymbol{x}_{-j}) = \left(\sum_{i \,:\, Scope[f_i] \ni X_j} \theta_i f_i[x_j, \boldsymbol{u}_j] \right) - \ln \left(\sum_{x'_j} \exp \left\{ \sum_{i \,:\, Scope[f_i] \ni X_j} \theta_i f_i[x'_j, \boldsymbol{u}_j] \right\} \right).$$

Convex

$$\frac{\partial}{\partial \theta_i} \ln P(x_j \mid \boldsymbol{x}_{-j}) = f_i[x_j, \boldsymbol{x}_{-j}] - \mathbb{E}_{x'_j \sim P_{\boldsymbol{\theta}}(X_j \mid \boldsymbol{x}_{-j})} \left[f_i[x'_j, \boldsymbol{x}_{-j}] \right].$$

Proposition 20.6.1:

$$\frac{\partial}{\partial \theta_i} \ell_{\text{pseudo}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{j : X_j \in Scope[f_i]} \left(\frac{1}{M} \sum_m f_i[\xi[m]] - E_{x'_j \sim P_{\boldsymbol{\theta}}(X_j \mid \boldsymbol{x}_{-j}[m])} \left[f_i[x'_j, \boldsymbol{x}_{-j}[m]] \right] \right). \tag{20.31}$$

Consistency of PL

- Thm 20.6.2 (Besag). If data is generated from our model with params θ^*, then as M->inf, argmax PL(θ) -> θ^*.
- Pf. The empirical approaches P(θ^*). Hence

$$\frac{1}{M}\sum_m f_i[\xi[m]] \longrightarrow \mathbb{E}_{\xi\sim P_{\theta^*}(\mathcal{X})}[f_i[\xi]].$$

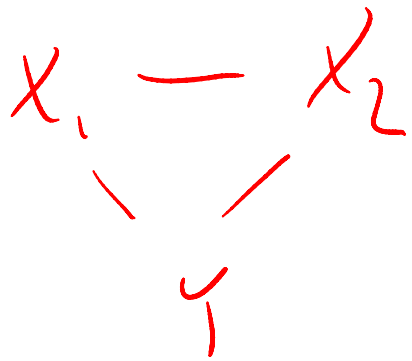
- And

$$\begin{aligned}\frac{1}{M}\sum_m \mathbb{E}_{x'_j\sim P_{\theta^*}(X_j|\boldsymbol{x}_{-j}[m])}[f_i[x'_j, x_{-j}[m]]] &= \sum_{\boldsymbol{x}} P_{\mathcal{D}}(x_{-j})\sum_{x'} P_{\theta^*}(x'_j \mid x_{-j}) f_i[x'_j, x_{-j}] \\ &\longrightarrow \sum_{\boldsymbol{x}_j} P_{\theta^*}(x_{-j})\sum_{x'_j} P_{\theta^*}(x'_j \mid x_{-j}) f_i[x'_j, x_{-j}] \\ &= \mathbb{E}_{\xi\sim P_{\theta^*}}[f_i[\xi]].\end{aligned}$$

- Hence gradient of PL is zero at θ^*.

Problem with PL

- Ex 20.6.3 (cf Hinton's greek vase)



Assume X1, X2 are strongly correlated (eg mirror images),
And X1,Y and X2,Y are less strongly correlated.
PL will learn that X1 can be predicted from X2, and will ignore Y.
At test time, if we observe Y and want to predict X1, we are hosed.

Sample based learning

- Recall loglik is $\frac{1}{M}\ell(\theta : \mathcal{D}) = \sum_i \theta_i \mathbf{E}_{\mathcal{D}}[f_i[d_i]] - \ln Z(\theta).$

$$
\begin{aligned}
Z(\theta) &= \sum_{\xi} \exp\left\{\sum_i \theta_i f_i[\xi]\right\} \\
&= \sum_{\xi} \frac{Q(\xi)}{Q(\xi)} \exp\left\{\sum_i \theta_i f_i[\xi]\right\} \\
&= \boldsymbol{E}_Q\left[\frac{1}{Q(\mathcal{X})} \exp\left\{\sum_i \theta_i f_i[\mathcal{X}]\right\}\right].
\end{aligned}
$$

Sample K x's given θ
Compute ln Z(θ)
Update θ
Repeat

$$
\begin{aligned}
Z(\theta) &= \boldsymbol{E}_{P_{\boldsymbol{\theta}^0}}\left[\frac{Z(\boldsymbol{\theta}^0)\exp\{\sum_i \theta_i f_i[\mathcal{X}]\}}{\exp\{\sum_i \theta_i^0 f_i[\mathcal{X}]\}}\right] \\
&= Z(\boldsymbol{\theta}^0)\boldsymbol{E}_{P_{\boldsymbol{\theta}^0}}\left[\exp\left\{\sum_i (\theta_i - \theta_i^0) f_i(\mathcal{X})\right\}\right].
\end{aligned}
$$

$$
\ln Z(\boldsymbol{\theta}) \approx \ln\left(\frac{1}{K}\sum_{k=1}^{K} \exp\left\{\sum_i (\theta_i - \theta_i^0) f_i(\xi^k)\right\}\right) + \ln Z(\boldsymbol{\theta}^0).
$$

Contrastive divergence

- Might need to sample many x's to accurately approximate Z, but this is slow
- So define a set D- of randomly perturbed neighbors of D, and use

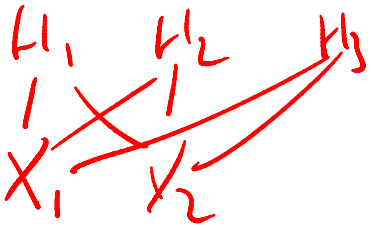
$$\ell_{\mathrm{CD}}(\boldsymbol{\theta} : \mathcal{D} \| \mathcal{D}^-) = \left[\mathbb{E}_{\xi \sim \hat{P}_{\mathcal{D}}} [\ln P_{\boldsymbol{\theta}}(\xi)] - \mathbb{E}_{\xi \sim \hat{P}_{\mathcal{D}^-}} [\ln P_{\boldsymbol{\theta}}(\xi)] \right],$$

$$\frac{\partial}{\partial \theta_i} \ell_{\mathrm{CD}}(\boldsymbol{\theta} : \mathcal{D} \| \mathcal{D}^-) = \mathbb{E}_{\hat{P}_{\mathcal{D}}} [f_i[\mathcal{X}]] - \mathbb{E}_{\hat{P}_{\mathcal{D}^-}} [f_i].$$

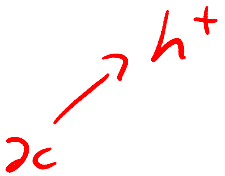
- Often x_i- is generated by applying 1 step of Gibbs sampling to x_i

CD for RBMs

- RBMs have 1 layer of hidden variables, so we need an additional expectation

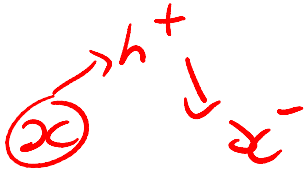


$$
\begin{aligned}
\nabla_i &= E_{\mathbf{x}\sim D} E_{\mathbf{h}} f_i(\mathbf{x},\mathbf{h}) - E_{\mathbf{x}\sim D^-} E_{\mathbf{h}} f_i(\mathbf{x},\mathbf{h}) \\
&\approx \frac{1}{N}\sum_n E_{\mathbf{h}_n} f_i(\mathbf{x}_n,\mathbf{h}_n) - E_{\mathbf{h}_n} f_i(\mathbf{x}_n^-,\mathbf{h}_n) \\
&\approx \frac{1}{N}\sum_n f_i(\mathbf{x}_n,\mathbf{h}_n^+) - f_i(\mathbf{x}_n^-,\mathbf{h}_n^-)
\end{aligned}
$$

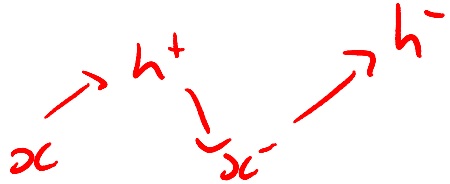


Stop learning when your dreams match reality

$\mathbf{h}_n^+ \sim p(\mathbf{h}|\mathbf{x}_n,\boldsymbol{\theta})$ Interpretation of data



$\mathbf{x}_n^- \sim p(\mathbf{x}|\mathbf{h}_n^+,\boldsymbol{\theta})$ Reconstruction/ Fantasy data



$\mathbf{h}_n^- \sim p(\mathbf{h}|\mathbf{x}_n^-,\boldsymbol{\theta})$ Interpretation of your fantasies

MAP approximation (perceptron training)

- Let us approximate Z (sum over all X) by the MAP estimate. Objective becomes

$$\frac{1}{M}\ell(\boldsymbol{\theta}:\mathcal{D}) - \ln P(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}), \qquad \frac{1}{M}\sum_{m=1}^{M} \ln \tilde{P}(\xi[m] \mid \boldsymbol{\theta}) - \ln \tilde{P}(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}).$$

- For a single data term

$$\begin{aligned}
&\ln P(\xi[m] \mid \boldsymbol{\theta}) - \ln P(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}) \\
&\quad = [\ln \tilde{P}(\xi[m] \mid \boldsymbol{\theta}) - \ln Z(\boldsymbol{\theta})] - [\ln \tilde{P}(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}) - \ln Z(\boldsymbol{\theta})] \\
&\quad = \ln \tilde{P}(\xi[m] \mid \boldsymbol{\theta}) - \ln \tilde{P}(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}) \\
&\quad = \sum_i \theta_i [f_i[\xi[m]] - f_i[\xi^{\mathrm{MAP}}(\boldsymbol{\theta})]].
\end{aligned}$$

- Hence gradient is

$$E_{\mathcal{D}}[f_i[\mathcal{X}]] - f_i[\xi^{\mathrm{MAP}}(\boldsymbol{\theta})].$$

Problem with MAP approximation

- The objective is always negative or 0 since

$$\frac{1}{M}\ell(\boldsymbol{\theta} : \mathcal{D}) - \ln P(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}),$$

$$\ln P(\xi[m] \mid \boldsymbol{\theta}) \leq \ln P(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}).$$

- We can always achieve the maximum of 0 by setting \theta=0

$$\begin{aligned}
\ln P(\xi[m] \mid \boldsymbol{\theta}) - \ln P(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}) \\
&= [\ln \tilde{P}(\xi[m] \mid \boldsymbol{\theta}) - \ln Z(\boldsymbol{\theta})] - [\ln \tilde{P}(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}) - \ln Z(\boldsymbol{\theta})] \\
&= \ln \tilde{P}(\xi[m] \mid \boldsymbol{\theta}) - \ln \tilde{P}(\xi^{\mathrm{MAP}}(\boldsymbol{\theta}) \mid \boldsymbol{\theta}) \\
&= \sum_i \theta_i [f_i[\xi[m]] - f_i[\xi^{\mathrm{MAP}}(\boldsymbol{\theta})]].
\end{aligned}$$

- “collapsing” problem

Max-margin training

- For conditional density models, we can change the objective to the following, which prevents collapsing

$$\ln P_{\boldsymbol{\theta}}(\boldsymbol{y}[m] \mid \boldsymbol{x}[m]) - \left[\max_{\boldsymbol{y} \neq \boldsymbol{y}[m]} \ln P_{\boldsymbol{\theta}}(y \mid \boldsymbol{x}[m])\right].$$

Find	$\gamma, \boldsymbol{\theta}$
that maximize	γ
subject to	$\ln P_{\boldsymbol{\theta}}(\boldsymbol{y}[m] \mid \boldsymbol{x}[m]) - \ln P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}[m]) \geq \gamma$ for all m, $\boldsymbol{y} \neq \boldsymbol{y}[m]$

$$\boldsymbol{\theta}^T(\boldsymbol{f}(\boldsymbol{y}[m], \boldsymbol{x}[m]) - \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}[m])) \geq \gamma.$$

To prevent margin blowing up we bound \theta

Simple-Max-Margin

Find	$\boldsymbol{\theta}$
that minimize	$\|\boldsymbol{\theta}\|_2^2$
subject to	$\boldsymbol{\theta}^T(\boldsymbol{f}(\boldsymbol{y}[m], \boldsymbol{x}[m]) - \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}[m])) \geq 1$ for all m, $\boldsymbol{y} \neq \boldsymbol{y}[m]$

QP: quad obj+linear constraints

Slack variables

- We want to minimize ||w||^2 st

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1.$$

- But we may not be able to achieve this gap, so we introduce slack variables (results in a Hidden Markov Support Vector Machine)

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t. \;\; \forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1 - \xi_i, \;\; \forall_i \; \xi_i \geq 0$$

Thanks to Mark Schmidt

Margin rescaling

- Intuitively if Yi' is similar to Yi, we don't mind if their probabilities are similar, but if they are very different, we want the gap to grow
- This gives max-margin markov network (M3N) aka structural SVM

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i') - \xi_i, \quad \forall_i \xi_i \geq 0,$$

Thanks to Mark Schmidt

Unconstrained form

- We can eliminate the slack vars to get

$$\min_w \sum_i \max_{Y_i' \neq Y_i} (\Delta(Y_i, Y_i') - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i))^+ + \lambda ||w||_2^2,$$

- Requires 2nd best decoding. But since $\Delta(Y_i, Y_i)=0$ we can write

$$\min_w \sum_i \max_{Y_i'} (\Delta(Y_i, Y_i') + \log p(Y_i'|w, X_i)) - \log p(Y_i|w, X_i) + \lambda ||w||_2^2,$$

- This can use generic MAP decoders that just change the local evidence potentials on Y'.
- For associative markov nets, globally optimal.

Thanks to Mark Schmidt

Cutting plane optimization

- Many possible optimization methods
- Simple approach for QP is cutting planes:
- Maximize quad objective with empty set of constraints – this is an upper bound.
- Add a violated constraint (*)
- Repeat until no violations.
- Thm: only need to add a poly num constraints.
- To find if constraints are violated: define

$$\boldsymbol{y}^{map} = \arg \max_{\boldsymbol{y} \neq \boldsymbol{y}[m]} \tilde{P}(\boldsymbol{y}, \boldsymbol{x}[m]).$$

- If P(y[m],x[m]) < p(ymap,x[m]) +1, add this violation. Else all constraints for m'th case are ok

$$\tilde{P}(\boldsymbol{y}[m], \boldsymbol{x}[m]) > \tilde{P}(\boldsymbol{y}^{map}, \boldsymbol{x}[m]) + 1 \geq \tilde{P}(\boldsymbol{y}, \boldsymbol{x}[m]) + 1,$$

Stat 521A
Lecture 26

Structure learning in UGMs

- Dependency networks
- Gaussian UGMs
- Discrete UGMs

Dependency networks

- A simple way to learn a graph is to regress each node on all others, p(x_i | x_{-i})
- If the full conditionals are sparse, this gives rise to a sparse graph
- Heckerman et al used classification trees to do variable selection
- Meinshausen & Buhlman proved that if you use lasso, the method is a consistent estimator of graph structure
- Wainwright et al extended the proof to L1 penalized logistic regression

Problem with depnets

- Although one can recover the structure, the params of the full conditionals need not correspond to any consistent joint
- To estimate params given the graph can be computationally hard (esp for discrete variables)
- Only give a point estimate of the structure*

* Parent fusion project

Bayesian inference for GGMs

- If we use decomposable graphical models, we can use the hyper inverse wishart as a conjugate prior, and hence compute p(D|G) analytically
- Problem reduces to discrete search
- Can use MCMC, MOSS, etc
- For non-decomposable models, have to approximate p(D|G) eg by BIC. Have to compute MLE for every neighboring graph! *
- See work by Adrian Dobra.

* Derive analog of structural EM to speed this up – nips project, anyone?

Graphical lasso

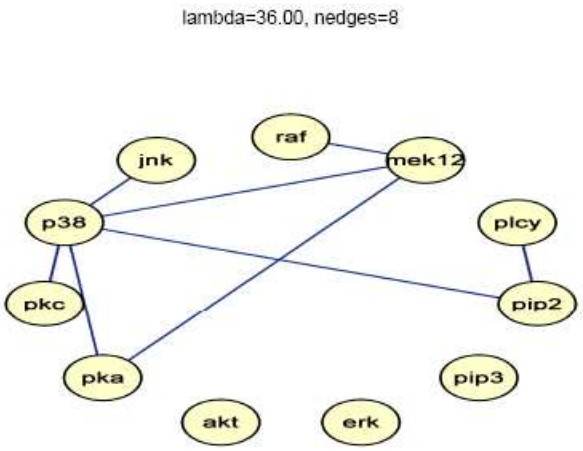
- We can estimate parameters and structure for GGMs simultaneously by optimizing

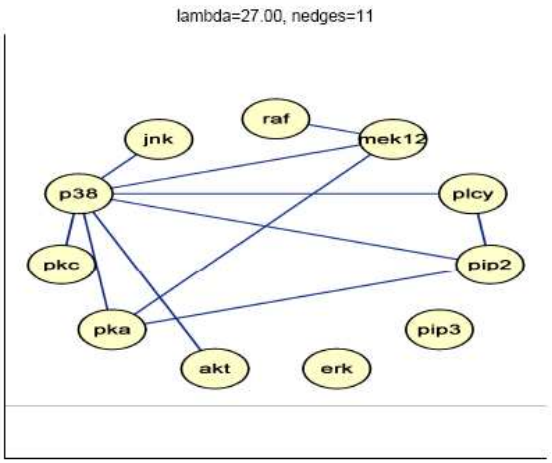
$$f(\mathbf{\Omega}) = \log\det\mathbf{\Omega} - \mathrm{tr}(\mathbf{S\Omega}) - \lambda||\mathbf{\Omega}||_1$$

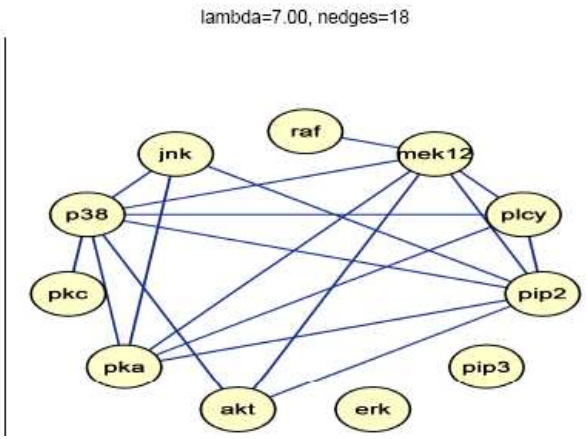
e $||\mathbf{\Omega}||_1 = \sum_{j,k}|\omega_{jk}|$

- Convex
- Can solve in O(#iter d^4) time by solving a sequence of lasso subproblems

Example


lambda=36.00, nedges=8
raf
jnk
mek12
p38
plcy
pkc
pip2
pka
pip3
akt
erk


lambda=27.00, nedges=11
raf
jnk
mek12
p38
plcy
pkc
pip2
pka
pip3
akt
erk


lambda=7.00, nedges=18
raf
jnk
mek12
p38
plcy
pkc
pip2
pka
pip3
akt
erk

MLE params for GGM

- Consider first the problem of estimating Ω given known zeros (absent edges)

$$\ell_C(\Omega) = \log\det\Omega - \mathrm{tr}(S\Omega) - \sum_{(j,k)\notin E(G)} \gamma_{jk}\Omega_{jk}$$

- Setting gradient to zero gives

$$\Omega^{-1} - S - \Gamma = 0 \qquad w_{12} - s_{12} - \gamma_{12} = 0$$

Let j be a specific node in group 1. Then if $G_{j2} \neq 0$, then $\gamma_{j2} = 0$, so $w_{j2} = s_{j2}$. In other words, edges that are not constrained to be zero must have an MLE covariance equal to the empirical covariance.

- Consider this partition

$$\begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix} \begin{pmatrix} \Omega_{11} & \omega_{12} \\ \omega_{12}^T & \omega_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0^T & 1 \end{pmatrix}$$

—

$$w_{12} = -W_{11}\omega_{12}/\omega_{22} = W_{11}\beta$$

e $\beta \stackrel{\text{def}}{=} -\omega_{12}/\omega_{22}$.

$$W_{11}\beta - s_{12} - \gamma_{12} = 0$$

Cont'd

- We have $\mathbf{W}_{11}\beta - \mathbf{s}_{12} - \gamma_{12} = \mathbf{0}$
- Dropping the zeros $\mathbf{W}_{11}^{*}\beta^{*} - \mathbf{s}_{12}^{*} = \mathbf{0}$
- Can recover Ω from weights using $\omega_{12} = -\beta_{12}\omega_{22}$
- To find w_22, use block inversion lemma

$$\omega_{22} = (\mathbf{W}/\mathbf{W}_{11})^{-1} = (w_{22} - \mathbf{w}_{12}^{T}\mathbf{W}_{11}^{-1}\mathbf{w}_{12})^{-1}$$

Now $\mathbf{W}_{11}^{-1}\mathbf{w}_{12} = (\mathbf{W}_{11}^{*})^{-1}\mathbf{s}_{12}^{*} = (\beta, 0)$, since $\mathbf{w}_{12} = \mathbf{s}_{12}$ in all locations that are not constrained to be zero. Similarly, $w_{22} = s_{22}$. Hence

$$\frac{1}{\omega_{22}} = s_{22} - \mathbf{w}_{12}^{T}\beta \tag{3.82}$$

code

```
W = S; % W = inv(precMat)
precMat = zeros(p,p);
beta = zeros(p-1,1);
iter = 1;
converged = false;
normW = norm(W);
while ~converged
  for i = 1:p
    % partition W & S for i
    noti = [1:i-1 i+1:p];
    W11 = W(noti,noti);
    w12 = W(noti,i);
    s22 = S(i,i);
    s12 = S(noti,i);

    % find G's non-zero index in W11
    idx = find(G(noti,i));  % non-zeros in G11
    beta(:) = 0;
    beta(idx) = W11(idx,idx) \ s12(idx);

    % update W
    w12 = W11 * beta;
    W(noti,i) = w12 ;
    W(i,noti) = w12';

    % update precMat (technically only needed on last iteration)
    p22 = max([0  1/(s22 - w12'*beta)]);  % must be non-neg
    p12 = -beta * p22;
    precMat(noti,i) = p12 ;
    precMat(i,noti) = p12';
    precMat(i,i) = p22;
  end
  converged =  convergenceTest(norm(W), normW) || (iter > maxIter);
  normW = norm(W);
  iter = iter + 1;
end
```

ggmFitHtf in pmtk (by Baback Moghaddam)

Example

Let us now give a worked example of this algorithm. Let the input be the following adjacency matrix, representing the cyclic structure, $X_1 - X_2 - X_3 - X_4 - X_1$, and empirical covariance matrix:

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} 10 & 1 & 5 & 4 \\ 1 & 10 & 2 & 6 \\ 5 & 2 & 10 & 3 \\ 4 & 6 & 3 & 10 \end{pmatrix} \tag{3.83}$$

After 3 iterations we converge to the following MLE:

$$\mathbf{\Sigma} = \begin{pmatrix} 10.00 & 1.00 & \mathbf{1.31} & 4.00 \\ 1.00 & 10.00 & 2.00 & \mathbf{0.87} \\ \mathbf{1.31} & 2.00 & 10.00 & 3.00 \\ 4.00 & \mathbf{0.87} & 3.00 & 10.00 \end{pmatrix}, \quad \mathbf{\Omega} = \begin{pmatrix} 0.12 & -0.01 & \mathbf{0} & -0.05 \\ -0.01 & 0.11 & -0.02 & \mathbf{0} \\ \mathbf{0} & -0.02 & 0.11 & -0.03 \\ -0.05 & \mathbf{0} & -0.03 & 0.13 \end{pmatrix} \tag{3.84}$$

Graphical lasso

$$f(\boldsymbol{\Omega}) = \log\det\boldsymbol{\Omega} - \mathrm{tr}(\mathbf{S}\boldsymbol{\Omega}) - \lambda||\boldsymbol{\Omega}||_1 \qquad \lambda_{jj} \geq 0, \lambda_{jk}^{max} = |\hat{\Sigma}_{jk}|$$

The basic idea is very similar to the method in Section 3.3.7, except we replace the least squares subproblem with a lasso subproblem. The analog of the gradient equation (3.75) is the following:

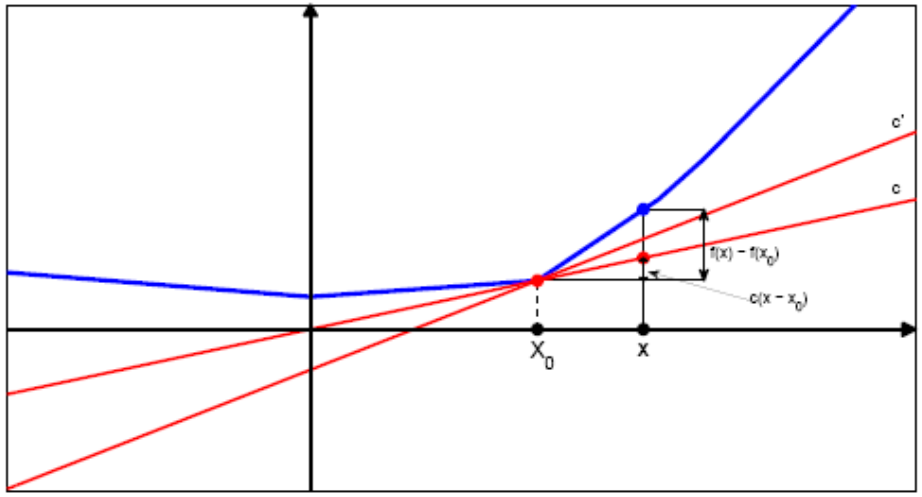
$$\boldsymbol{\Omega}^{-1} - \mathbf{S} - \lambda\mathrm{Sign}(\boldsymbol{\Omega}) = \mathbf{0} \tag{3.86}$$

As discussed in Section ??, we must replace the gradient with the subgradient, due to the non differentiable penalty term. So we define $\mathrm{Sign}(\omega_{jk}) = \mathrm{sign}(\omega_{jk})$ if $\omega_{jk} \neq 0$, and $\mathrm{Sign}(\omega_{jk}) \in [-1, 1]$ otherwise. The analogous result to Equation 3.79 is

$$\mathbf{W}_{11}\boldsymbol{\beta} - \mathbf{s}_{12} + \lambda\mathrm{Sign}(\boldsymbol{\beta}) = \mathbf{0} \tag{3.87}$$

since $\boldsymbol{\beta}$ and $\boldsymbol{\omega}_{12}$ have opposite signs.

Subgradients



We can generalize the notion of derivative to handle this case as follows. We define a **subderivative** of a function $f : \mathcal{I} \to \mathbb{R}$ at a point x_0 to be a scalar c such that

$$f(x) - f(x_0) \geq c(x - x_0) \forall x \in \mathcal{I} \tag{29.84}$$

where $\mathcal{I}$ is some interval containing x_0. See Figure 29.16. We define the *set* of subderivatives as the interval $[a, b]$ where a and b are the one-sided limits

$$a = \lim_{x \to x_0^-} \frac{f(x) - f(x_0)}{x - x_0}, \quad b = \lim_{x \to x_0^+} \frac{f(x) - f(x_0)}{x - x_0} \tag{29.85}$$

The set $[a, b]$ of all subderivatives is called the **subdifferential** of the function f at x_0 and is denoted $\partial f(x)|_{x_0}$. For example, the subdifferential of the absolute value function $f(x) = |x|$ is

$$\partial f(x) = \begin{cases} \{-1\} & \text{if } x < 0 \\ [-1, 1] & \text{if } x = 0 \\ \{+1\} & \text{if } x > 0 \end{cases} \tag{29.86}$$

If the function is everywhere differentiable, then $\partial f(x) = \{\frac{df(x)}{dx}\}$. By analogy to the standard calculus result, one can show that the point $\hat{x}$ is a local minimum of f iff $0 \in \partial f(x)$.

Graphical lasso

$$f(\mathbf{\Omega}) = \log\det\mathbf{\Omega} - \mathrm{tr}(\mathbf{S\Omega}) - \lambda||\mathbf{\Omega}||_1$$

The basic idea is very similar to the method in Section 3.3.7, except we replace the least squares subproblem with a lasso subproblem. The analog of the gradient equation (3.75) is the following:

$$\mathbf{\Omega}^{-1} - \mathbf{S} - \lambda\mathrm{Sign}(\mathbf{\Omega}) = \mathbf{0} \tag{3.86}$$

As discussed in Section ??, we must replace the gradient with the subgradient, due to the non differentiable penalty term. So we define $\mathrm{Sign}(\omega_{jk}) = \mathrm{sign}(\omega_{jk})$ if $\omega_{jk} \neq 0$, and $\mathrm{Sign}(\omega_{jk}) \in [-1, 1]$ otherwise. The analogous result to Equation 3.79 is

$$\mathbf{W}_{11}\boldsymbol{\beta} - \mathbf{s}_{12} + \lambda\mathrm{Sign}(\boldsymbol{\beta}) = \mathbf{0} \tag{3.87}$$

since $\boldsymbol{\beta}$ and $\boldsymbol{\omega}_{12}$ have opposite signs.

This is equivalent to a lasso problem. To see this, consider the objective

$$J(\boldsymbol{\beta}) = \tfrac{1}{2}(\mathbf{y} - \mathbf{Z}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{Z}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1 \tag{3.88}$$

Setting the gradient to zero we get

$$\mathbf{Z}^T\mathbf{Z}\boldsymbol{\beta} - \mathbf{Z}^T\mathbf{y} + \lambda\mathrm{Sign}(\boldsymbol{\beta}) = \mathbf{0} \tag{3.89}$$

We see that $\mathbf{Z}^T\mathbf{y}$ is similar to $\mathbf{s}_{12}$ (namely an estimate of the covariance between target and inputs), and that $\mathbf{Z}^T\mathbf{Z}$ gets replaced by $\mathbf{W}_{11}$, which represents correlation amongst the current inputs.

Shooting (coord desc for lasso)

We now present a **coordinate descent** algorithm called **shooting** [Fu98] for solving the unconstrained lasso problem:

$$J(\mathbf{w}, \lambda) = RSS(\mathbf{w}) + \lambda \sum_{j=1}^{d} |w_j| \tag{17.36}$$

Besides being simple and fast, this method yields additional insight into why an L1 regularizer results in a sparse solution.

We can compute the partial derivative of the lasso objective function wrt a particular parameter, say w_k as follows. One can show (Exercise 17) that

$$\frac{\partial}{\partial w_k} RSS(\mathbf{w}) = a_k w_k - c_k \tag{17.37}$$

$$a_k = 2 \sum_{i=1}^{n} x_{ik}^2 \tag{17.38}$$

$$c_k = 2 \sum_{i=1}^{n} x_{ik}(y_i - \mathbf{w}_{-k}^T \mathbf{x}_{i,-k}) \tag{17.39}$$

$$= 2 \sum_{i=1}^{n} \left[x_{ik} y_i - x_{ik} \mathbf{w}^T \mathbf{x}_i + w_k x_{ik}^2 \right] \tag{17.40}$$

where $\mathbf{w}_{-k} = \mathbf{w}$ without component k, and similarly for $\mathbf{x}_{i,-k}$. We see that c_k is (proportional to) the correlation between the k'th feature $\mathbf{x}_{:,k}$ and the residual due to the other features, $\mathbf{r}_{-k} = \mathbf{y} - \mathbf{X}_{:,-k}\mathbf{w}_{-k}$; if this correlation is zero, then feature k would be orthogonal to the residual, and we couldn't reduce the RSS by updating w_k. Hence the magnitude of c_k is an indication of how relevant feature k is for predicting $\mathbf{y}$ (relative to the other features and the current parameters).

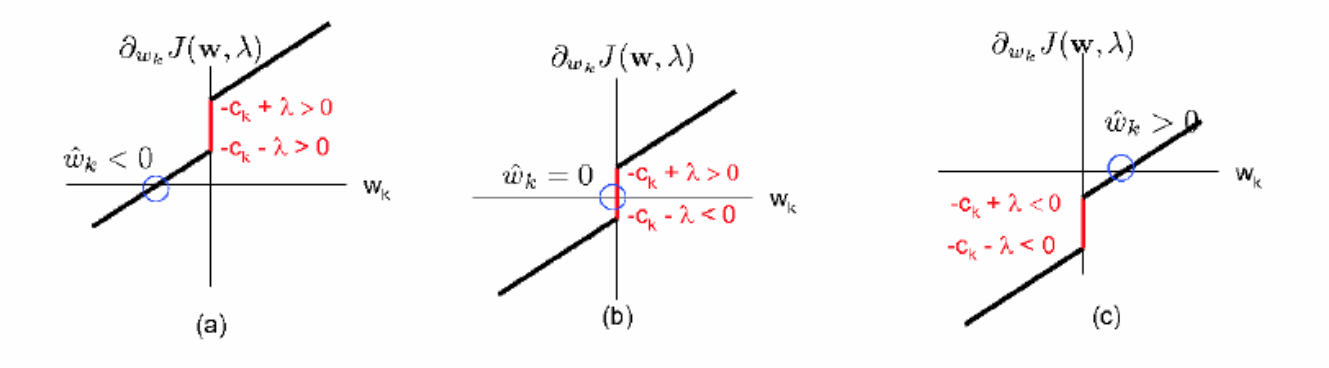
Shooting cont'd

The L1 penalty function is not differentiable, so we need to compute the **subdifferential** (see Section 29.6.1) rather than the standard differential. This is given by

$$\partial_{w_k} J(\mathbf{w}, \lambda) = (a_k w_k - c_k) + \lambda \partial_{w_k} ||\mathbf{w}||_1 \tag{17.41}$$

$$= \begin{cases} \{a_k w_k - c_k - \lambda\} & \text{if } w_k < 0 \\ [-c_k - \lambda, -c_k + \lambda] & \text{if } w_k = 0 \\ \{a_k w_k - c_k + \lambda\} & \text{if } w_k > 0 \end{cases} \tag{17.42}$$

This subdifferential is a piecewise linear function of w_k. Since $a_k > 0$, it is sloping up and to the right, except it has a vertical "kink" in it at $w_k = 0$, spanning the range $[-c_k - \lambda, -c_k + \lambda]$: see Figure 17.6. Depending on the value of c_k, the solution to $\partial_{w_k} J(\mathbf{w}, \lambda) = 0$ can occur at 3 different values of w_k, as follows:

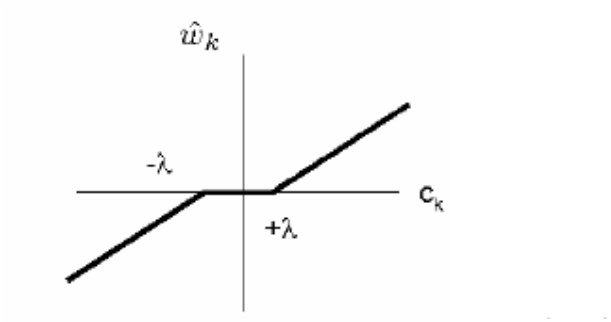


Soft thresholding

1. $c_k < -\lambda$, so the feature is strongly negatively correlated with the residual. In this case, the subgradient is zero at $\hat{w}_k = \frac{c_k+\lambda}{a_k} < 0$.
2. $c_k \in [-\lambda, \lambda]$, so the feature is only weakly correlated with the residual. In this case, the subgradient is zero at $\hat{w}_k = 0$. Thus if the correlation is not less than λ, we set the corresponding coefficient to 0.
3. $c_k > \lambda$, so the feature is strongly positively correlated with the residual. In this case, the subgradient is zero at $\hat{w}_k = \frac{c_k-\lambda}{a_k} > 0$.

In summary, we have

$$\hat{w}_k(c_k) = \begin{cases} (c_k + \lambda)/a_k & \text{if } c_k < -\lambda \\ 0 & \text{if } c_k \in [-\lambda, \lambda] \\ (c_k - \lambda)/a_k & \text{if } c_k > \lambda \end{cases} \tag{17.43}$$



$$\hat{w}_k = \text{soft}(\frac{c_k}{a_k}; \frac{\lambda}{a_k})$$

$$\text{soft}(a; \delta) = \text{sign}(a) \max\{0, |a| - \delta\} = \text{sign}(a) (|a| - \delta)_+$$

Lasso vs ridge vs subset selection

For orthonormal features, we have explicit solns

the lasso solution as follows (using the fact that $a_k = 2$ and $\hat{w}_k^{OLS} = c_k/2$)

$$\hat{w}_k^{lasso} = \text{sign}(\hat{w}_k^{OLS}) \left(|\hat{w}_k^{OLS}| - \frac{\lambda}{2} \right)_+ \tag{17.46}$$

By contrast, the ridge estimate would be

$$\hat{w}_k^{ridge} = \frac{\hat{w}_k^{OLS}}{1+\lambda} \tag{17.47}$$

which does not force sparsity. If we pick the best K features using subset selection, the parameter estimate is as follows

$$\hat{w}_k^{SS} = \begin{cases} \hat{w}_k^{OLS} & \text{if rank}(|w_k|) \leq K \\ 0 & \text{otherwise} \end{cases} \tag{17.48}$$

Graphical lasso with shooting

$$f(\boldsymbol{\Omega}) = \log\det\boldsymbol{\Omega} - \mathrm{tr}(\mathbf{S}\boldsymbol{\Omega}) - \lambda||\boldsymbol{\Omega}||_1$$

The basic idea is very similar to the method in Section 3.3.7, except we replace the least squares subproblem with a lasso subproblem. The analog of the gradient equation (3.75) is the following:

$$\boldsymbol{\Omega}^{-1} - \mathbf{S} - \lambda\mathrm{Sign}(\boldsymbol{\Omega}) = \mathbf{0} \tag{3.86}$$

As discussed in Section ??, we must replace the gradient with the subgradient, due to the non differentiable penalty term. So we define $\mathrm{Sign}(\omega_{jk}) = \mathrm{sign}(\omega_{jk})$ if $\omega_{jk} \neq 0$, and $\mathrm{Sign}(\omega_{jk}) \in [-1, 1]$ otherwise. The analogous result to Equation 3.79 is

$$\mathbf{W}_{11}\boldsymbol{\beta} - \mathbf{s}_{12} + \lambda\mathrm{Sign}(\boldsymbol{\beta}) = \mathbf{0} \tag{3.87}$$

since $\boldsymbol{\beta}$ and $\boldsymbol{\omega}_{12}$ have opposite signs.

This is equivalent to a lasso problem. To see this, consider the objective

$$J(\boldsymbol{\beta}) = \tfrac{1}{2}(\mathbf{y} - \mathbf{Z}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{Z}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1 \tag{3.88}$$

Setting the gradient to zero we get

$$\mathbf{Z}^T\mathbf{Z}\boldsymbol{\beta} - \mathbf{Z}^T\mathbf{y} + \lambda\mathrm{Sign}(\boldsymbol{\beta}) = \mathbf{0} \tag{3.89}$$

We see that $\mathbf{Z}^T\mathbf{y}$ is similar to $\mathbf{s}_{12}$ (namely an estimate of the covariance between target and inputs), and that $\mathbf{Z}^T\mathbf{Z}$ gets replaced by $\mathbf{W}_{11}$, which represents correlation amongst the current inputs.

One simple way to solve this lasso problem is to use coordinate descent, known as the **shooting algorithm** (see Section ??). To apply this to the current problem, let $\mathbf{V} = \mathbf{W}_{11}$. (Recall $\mathbf{W} = \boldsymbol{\Sigma}$.) Then the update for $\boldsymbol{\beta}$ becomes

$$\beta_j := S_\lambda\left(s_{12j} - \sum_{k\neq j} V_{kj}\beta_k\right)/V_{jj} \tag{3.90}$$

where S is the soft-threshold operator

$$S_t(x) = \mathrm{sign}(x)\max(0, |x| - t) \tag{3.91}$$

We can implement this in a way which is very similar to Listing ??. The only change is to replace the line `beta(idx) = W11(idx,idx) \ s12(idx)` with the code shown below.

Discrete UGMs

- Computing Z and hence the likelihood is intractable unless the graph is decomposable
- Hence Bayesian methods “never” used
- Even search and score is inefficient

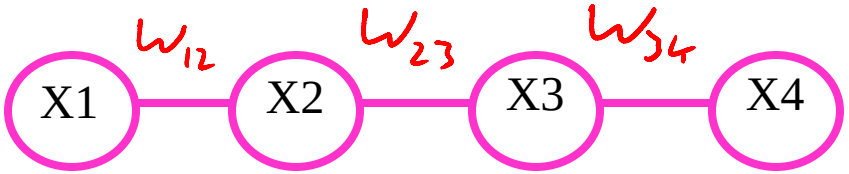
Ising models

- Analogous to GGM for binary data

$$\mathcal{N}(\mathbf{x}|\mathbf{K}) = \frac{1}{Z(\mathbf{K})} \exp(-\tfrac{1}{2} \sum_{j,k} K_{j,k} x_j x_k), \quad x_j \in \mathbb{R}$$

$$p(\mathbf{x}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp(\sum_{j,k} W_{jk} x_j x_k), \quad x_j \in \{-1, +1\}$$

$$\mathbf{W} = \begin{pmatrix} W_{11} & W_{12} & 0 & 0 \\ W_{21} & W_{22} & W_{23} & 0 \\ 0 & W_{32} & W_{33} & W_{34} \\ 0 & 0 & W_{43} & W_{44} \end{pmatrix}$$



$w_{jk} \geq 0$ attractive (ferro magnet)

$w_{jk} \leq 0$ repuslive (anti ferro magnetic)

w_{jk} mixed sign frustrated system

$$X_j \perp X_{-j} | X_{N_j}$$

Markov property

Besag, Hammersley & Clifford, Geman & Geman

Glasso for Ising models (Banerjee)

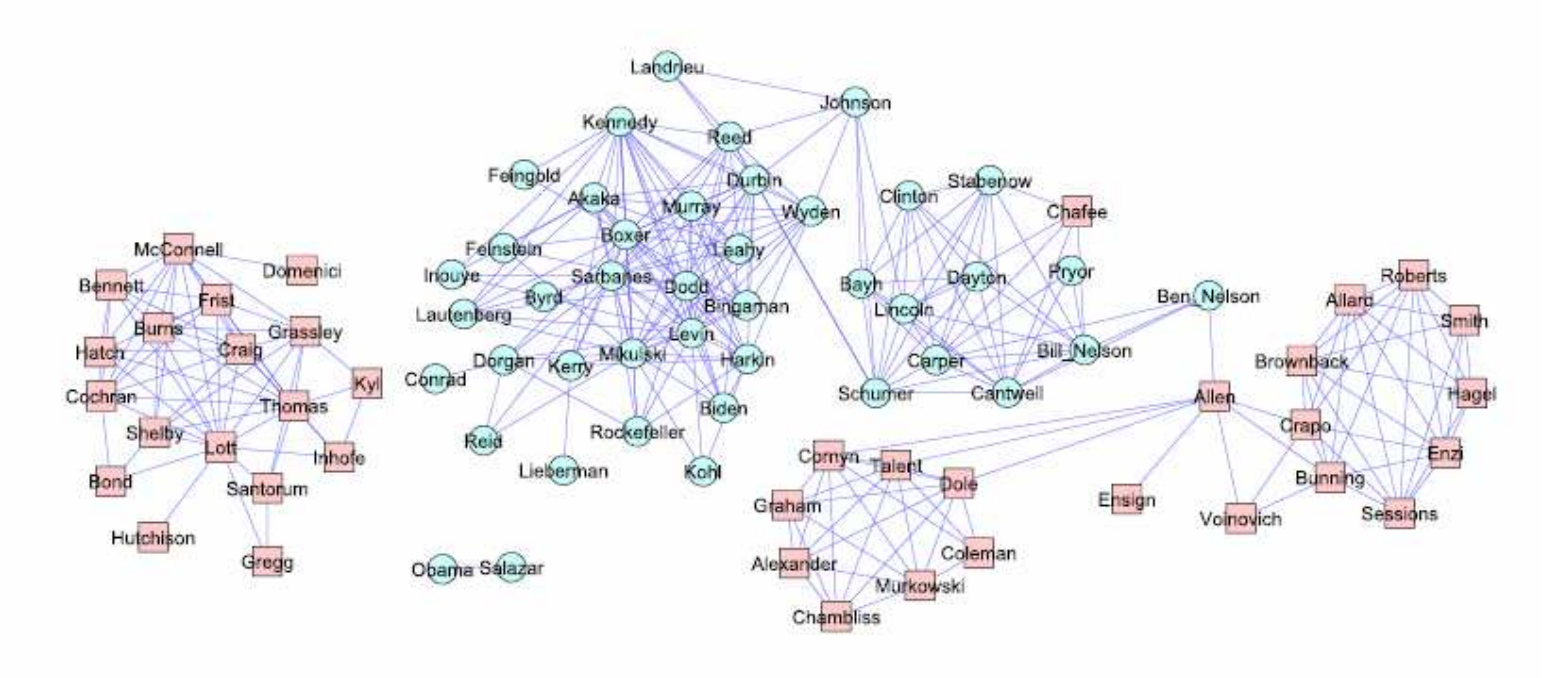
$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z}\exp[\sum_{i=1}^{d-1}\sum_{j=i+1}^{d} W_{ij}x_i x_j]$$

$$Z = \sum_{\mathbf{x}\in\{-1,+1\}^d} \exp[\sum_{i=1}^{d-1}\sum_{j=i+1}^{d} W_{ij}x_i x_j]$$

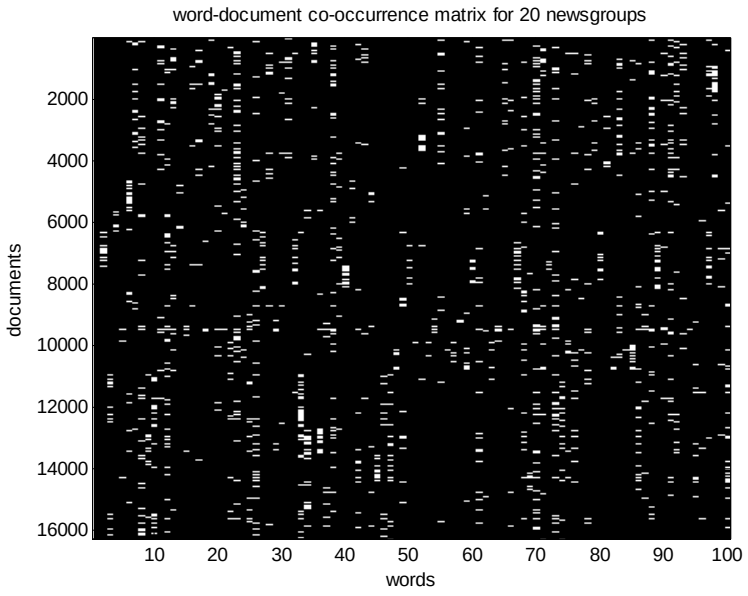
Convex relaxation of matrix permanent to matrix determinant

$$\hat{\mathbf{W}} = \text{graphicalLasso}(\text{Cov}(\mathbf{X}) - \lambda\mathbf{I} + \frac{1}{3}\mathbf{I}, \quad \lambda)$$

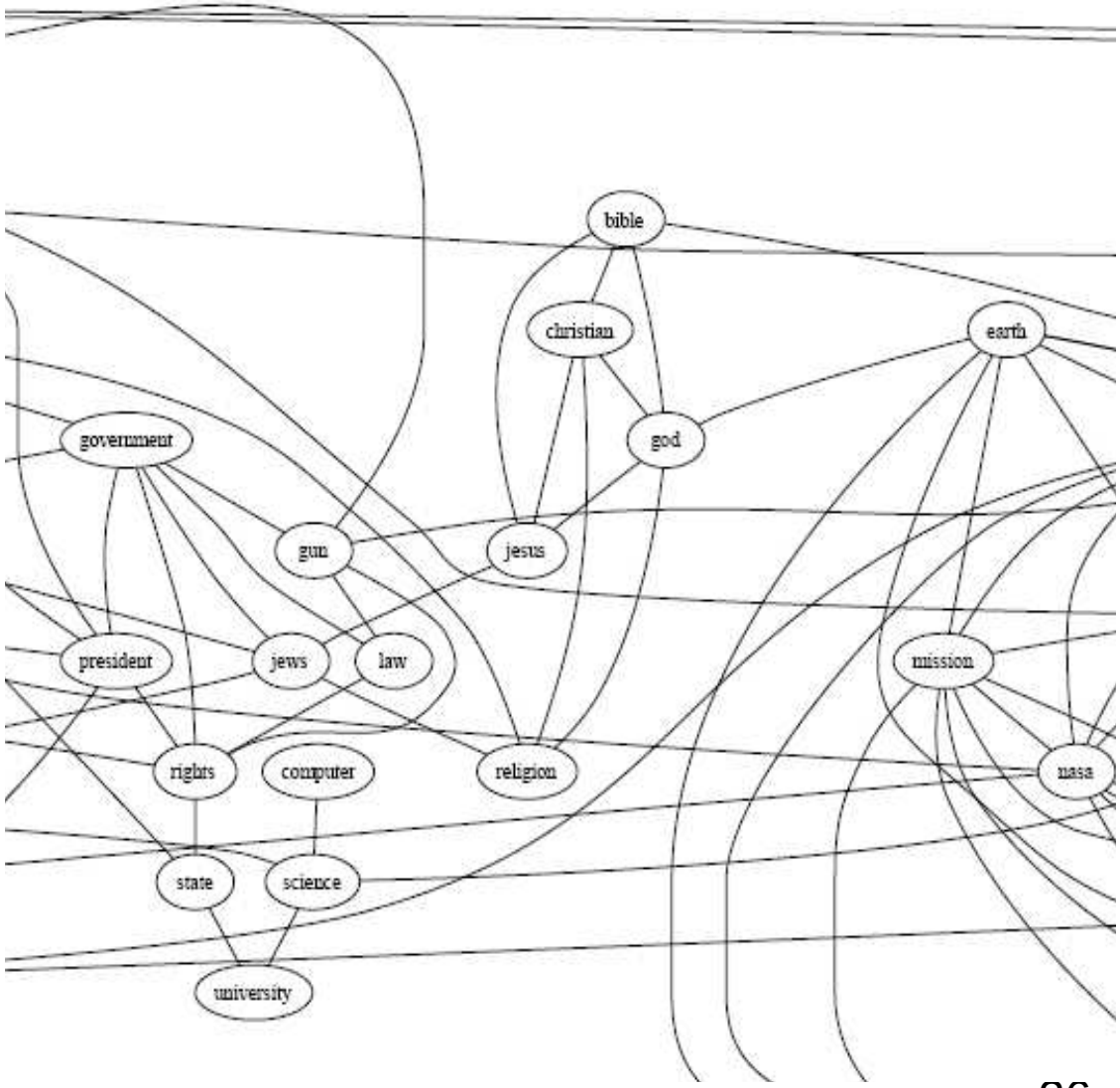
Senate voting data



20 newsgroups



n=16,000, d=100



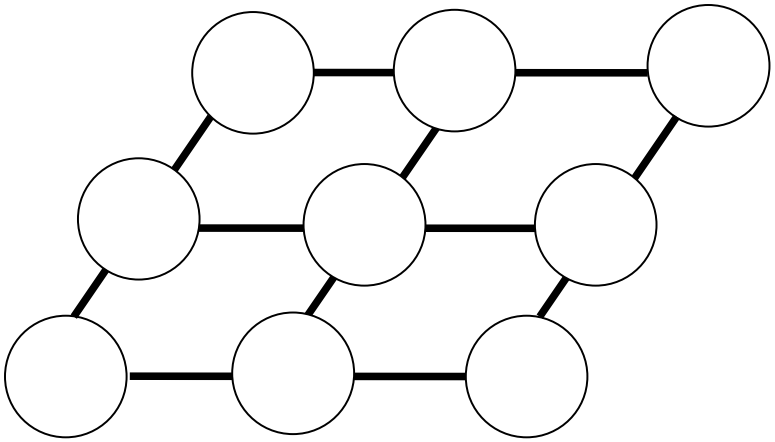
Courtesy Mark Schmidt

Markov random fields

- Markov random fields for $y_j \in \{1,\ldots,K\}$

$$p(\mathbf{y}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp(\sum_{j,k} \mathbf{w}_{jk}^T \mathbf{f}_{jk}(y_j, y_k)) \propto \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}))$$

y_j	y_k	$\mathbf{f}_{jk}(y_j, y_k)$
1	1	$(1,0,0,0,0,0,0,0,0)$
1	2	$(0,1,0,0,0,0,0,0,0)$
1	3	$(0,0,1,0,0,0,0,0,0)$
2	1	$(0,0,0,1,0,0,0,0,0)$
...		
3	3	$(0,0,0,0,0,0,0,0,1)$



Parameter *vector* on each edge

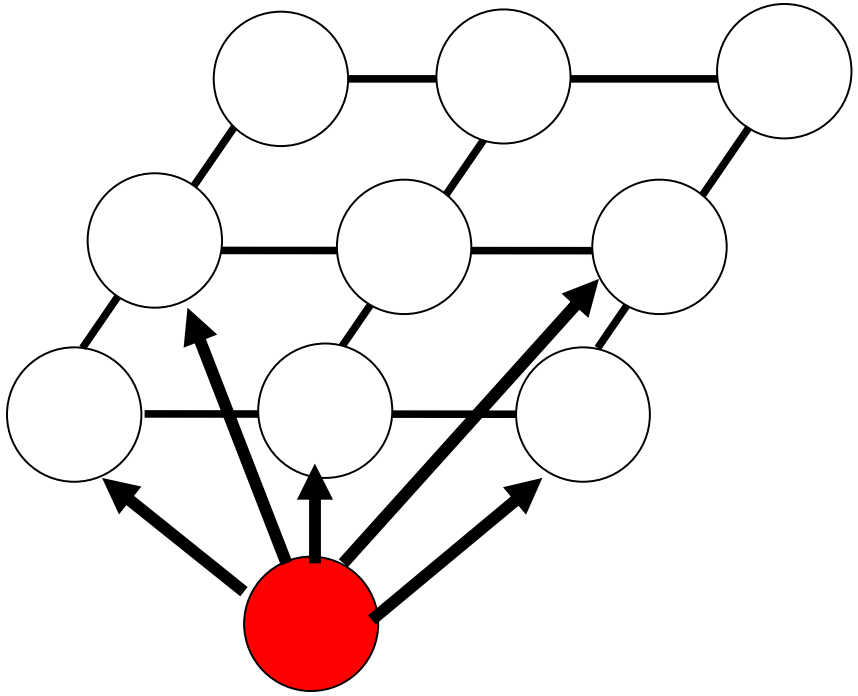
- No longer a 1:1 mapping between G and W

Conditional random fields

- CRFs are a conditional density model

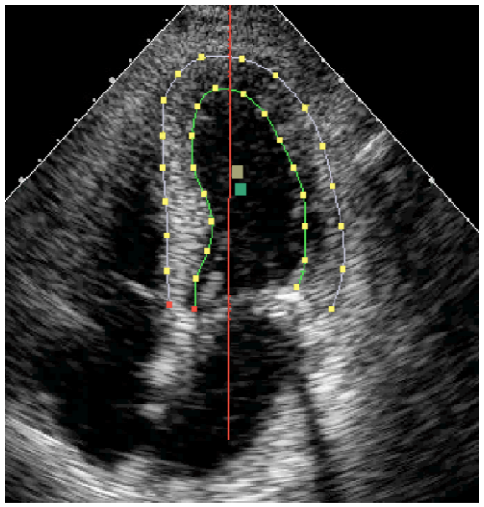
$$p(\mathbf{y}|\mathbf{x},\mathbf{W},\mathbf{V}) = \frac{1}{Z(\mathbf{W},\mathbf{V},\mathbf{x})}\exp(\sum_{j,k}\mathbf{w}_{j,k}^T\mathbf{f}_{jk}(y_j,y_k,\mathbf{x}) + \sum_j \mathbf{v}_j^T\mathbf{g}_j(y_j,\mathbf{x}))$$

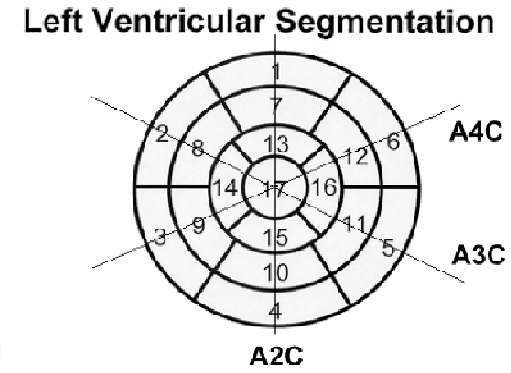
- No longer a 1:1 mapping between G and W

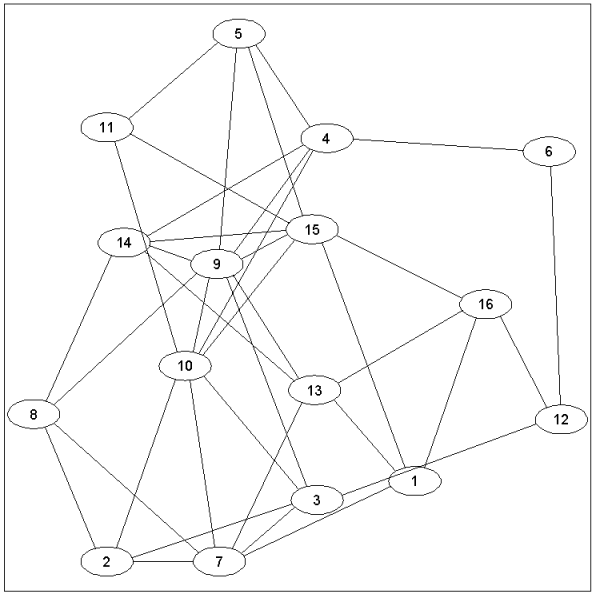


Heart wall abnormality data

- d=16, n=345, $y_j \in \{0,1\}$ representing normal or abnormal segment, x_j in R^{100} representing features derived from image processing







"Structure Learning in Random Fields for Heart Motion Abnormality Detection" Mark Schmidt, Kevin Murphy, Glenn Fung, Romer Rosales. CVPR 2008.

Group L1 regularization

- Solution: penalize groups of parameters, one group per edge

$$
\begin{aligned}
J(\mathbf{w}, \mathbf{v}) &= -\log \sum_i p(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}, \mathbf{v}) + \lambda_2 ||\mathbf{v}||_2^2 + \lambda_1 \sum_g ||\mathbf{w}_g||_p \\
||\mathbf{w}||_2 &= \sqrt{\sum_k w_k^2} \\
||\mathbf{w}||_\infty &= \max_k |w_k|
\end{aligned}
$$

Group lasso

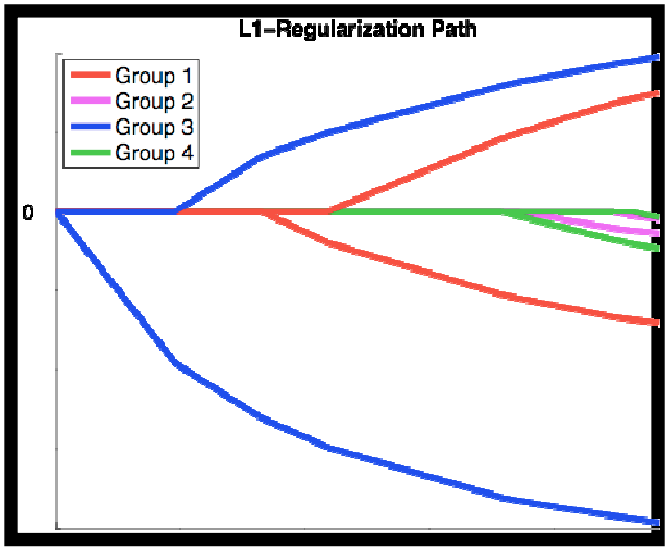
- Sometimes we want to select groups of parameters together (e.g., when encoding categorical inputs)

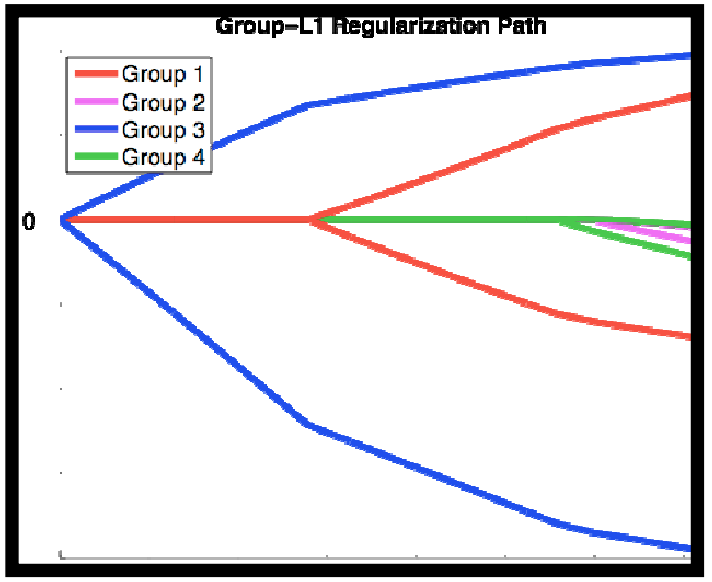
$$\hat{\mathbf{w}} = \arg\min RSS(\mathbf{w}) + \lambda R(\mathbf{w})$$

$$R(\mathbf{w}) = \sum_g ||\mathbf{w}_g||_2 = \sum_g \sqrt{\sum_{j \in g} w_{gj}^2}$$

$$R(\mathbf{w}) = \sum_g ||\mathbf{w}_g||_\infty = \sum_g \max_{j \in g} |w_{gj}|$$

Still convex, but much harder to optimize…





Group L1 for graphs

- Penalize groups of parameters, one group per edge

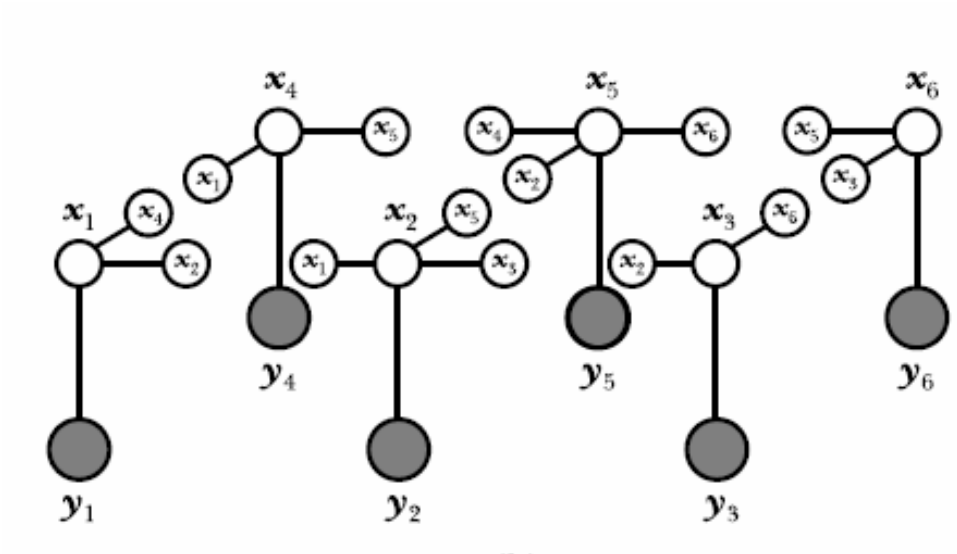
$$
\begin{aligned}
J(\mathbf{w}, \mathbf{v}) &= -\log \sum_i p(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}, \mathbf{v}) + \lambda_2 ||\mathbf{v}||_2^2 + \lambda_1 \sum_g ||\mathbf{w}_g||_p \\
||\mathbf{w}||_2 &= \sqrt{\sum_k w_k^2} \\
||\mathbf{w}||_\infty &= \max_k |w_k|
\end{aligned}
$$

- Issues
  - How deal with intractable log-likelihood? Use PL (Schmidt) or LBP (Lee & Koller)
  - How handle non-smooth penalty functions? (Projected gradient or projected quasi newton)

Pseudo likelihood

- PL is locally normalized

$$
\begin{aligned}
L(\mathbf{W}) &= \prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{W}) = \prod_{i=1}^{n} \frac{1}{Z(\mathbf{W})} \exp(\sum_j \sum_k x_{ij} W_{jk} x_{ik}) \\
PL(\mathbf{W}) &= \prod_{i=1}^{n} \prod_{j=1}^{d} p(x_{ij}|\mathbf{x}_{i,n_i}, \mathbf{w}_{j,:}) \\
&= \prod_j \prod_i \frac{1}{Z(\mathbf{w}_j, \mathbf{x}_{i,N_j})} \exp(x_{ij} \sum_k W_{jk} x_{ik}) \\
Z(\mathbf{w}_j, \mathbf{x}_{N_j}) &= \sum_{x_j \in \{-1,+1\}} \exp(x_j \sum_{k \in N_j} W_{jk} x_k)
\end{aligned}
$$



Constrained formulation

- Convert penalized negative log pseudo likelihood

$$
\begin{aligned}
f(\mathbf{w},\mathbf{v}) &= -\log \sum_i PL(\mathbf{y}_i|\mathbf{x}_i,\mathbf{v},\mathbf{w}) + \lambda_2 ||\mathbf{v}||_2^2 \\
\min_{\mathbf{w},\mathbf{v}} &= f(\mathbf{w},\mathbf{v}) + \lambda_1 \sum_g ||\mathbf{w}_g||_p
\end{aligned}
$$

- into constrained form

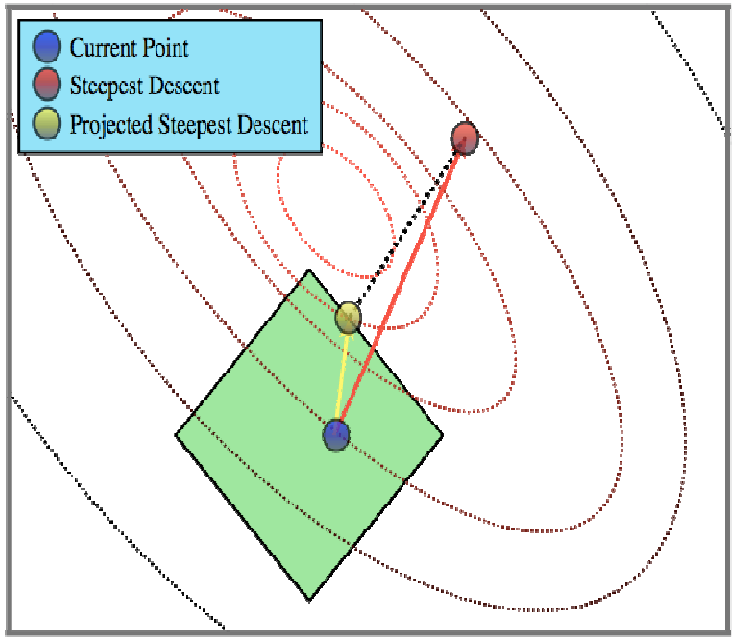
$$
\begin{aligned}
L(\boldsymbol{\alpha},\mathbf{w},\mathbf{v}) &= f(\mathbf{w},\mathbf{v}) + \lambda_1 \sum_g \alpha_g \\
\min_{\boldsymbol{\alpha},\mathbf{w},\mathbf{v}} &= L(\boldsymbol{\alpha},\mathbf{w},\mathbf{v}) \text{ st } \forall g.\alpha_g \geq ||\mathbf{w}_g||_p
\end{aligned}
$$

Desiderata for an optimizer

- Must handle $\binom{d}{2}$ groups (d = 16 in our application, so 120 groups)
- Must handle 100s features per group
- Cannot use second-order information (Hessian too expensive to compute or store) – so interior point is out
- Must converge quickly

Projected gradient method

- At each step, we perform an efficient projection onto the convex constraint set



$$
\begin{aligned}
\mathbf{x}_k &= (\boldsymbol{\alpha}, \mathbf{w})_k \\
\mathbf{x}_{k+1} &= t\Pi_{S_p}(\mathbf{x}_k - \beta \mathbf{g}_k) \\
\mathbf{g}_k &= \nabla f(\mathbf{x})_{\mathbf{x}_k} \\
\Pi_{\mathcal{S}}(\mathbf{x}) &= \arg\min_{\mathbf{x}^* \in \mathcal{S}} ||\mathbf{x} - \mathbf{x}^*||_2 \\
\mathcal{S}_p &= \{\mathbf{x} : \forall g . \alpha_g \geq ||\mathbf{w}_g||_p\}
\end{aligned}
$$

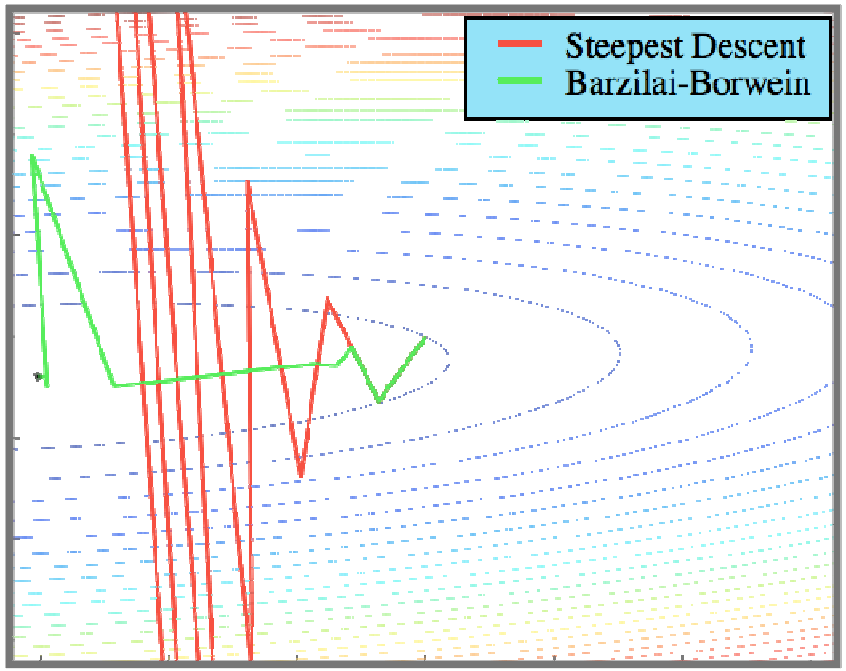
Project each group separately.
Takes O(N) time for p=2,
O(N log N) time for p=∞,

Where N = #params per group.

van der Berg, Schmidt, Friedlander, Murphy; Duchi et al.

Spectral step size

- Gradient descent can be slow
- Barzilai and Borwein proposed the following stepsize, which in some cases enjoys super-linear convergence rates



$$\mathbf{x}_{k+1} = t\Pi(\mathbf{x}_k - \beta_k \mathbf{g}_k)$$

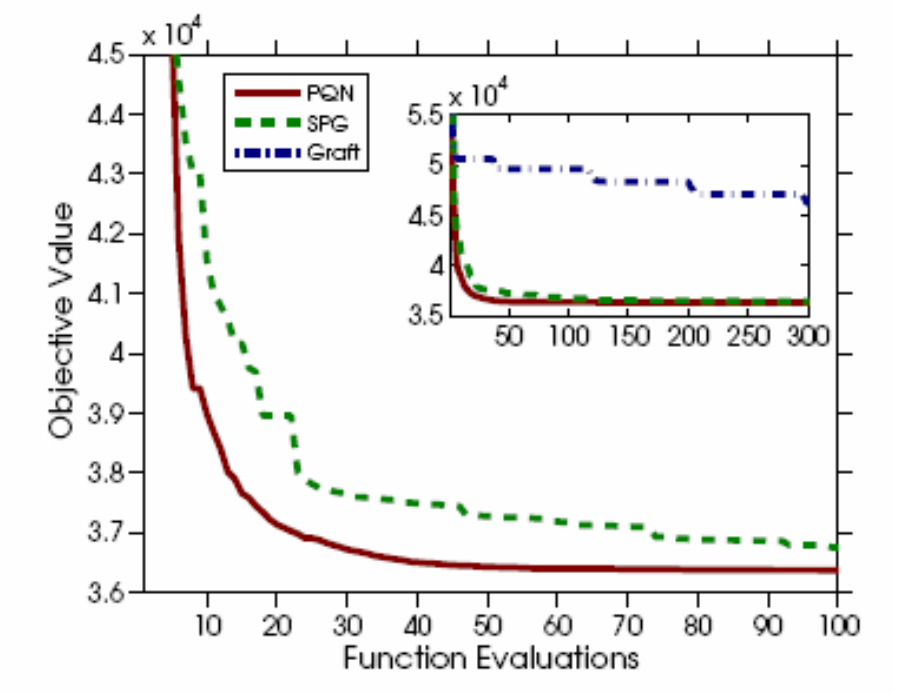
$$\mathbf{g}_k = \nabla f(\mathbf{x})|_{\mathbf{x}_k}$$

$$\beta_{k+1} = \frac{(\mathbf{x}_k - \mathbf{x}_{k-1})^T(\mathbf{x}_k - \mathbf{x}_{k-1})}{(\mathbf{x}_k - \mathbf{x}_{k-1})^T(\mathbf{g}_k - \mathbf{g}_{k-1})}$$

t chosen using non-monotone Armijo line search

Projected quasi Newton

- Use LBFGS in outer loop to create a constrained quadratic approximation to objective
- Use spectral projected gradient in inner loop to solve subproblem



"Optimizing Costly Functions with Simple Constraints:
A Limited-Memory Projected Quasi-Newton Algorithm",
Mark Schmidt, Ewout van den Berg, Michael P. Friedlander, and Kevin Murphy,
AI/Stats 2009

Experiments

- We compared classification accuracy on synthetic 10-node CRF and real 16-node CRF.
- For each node, we compute the max of marginal using exact inference

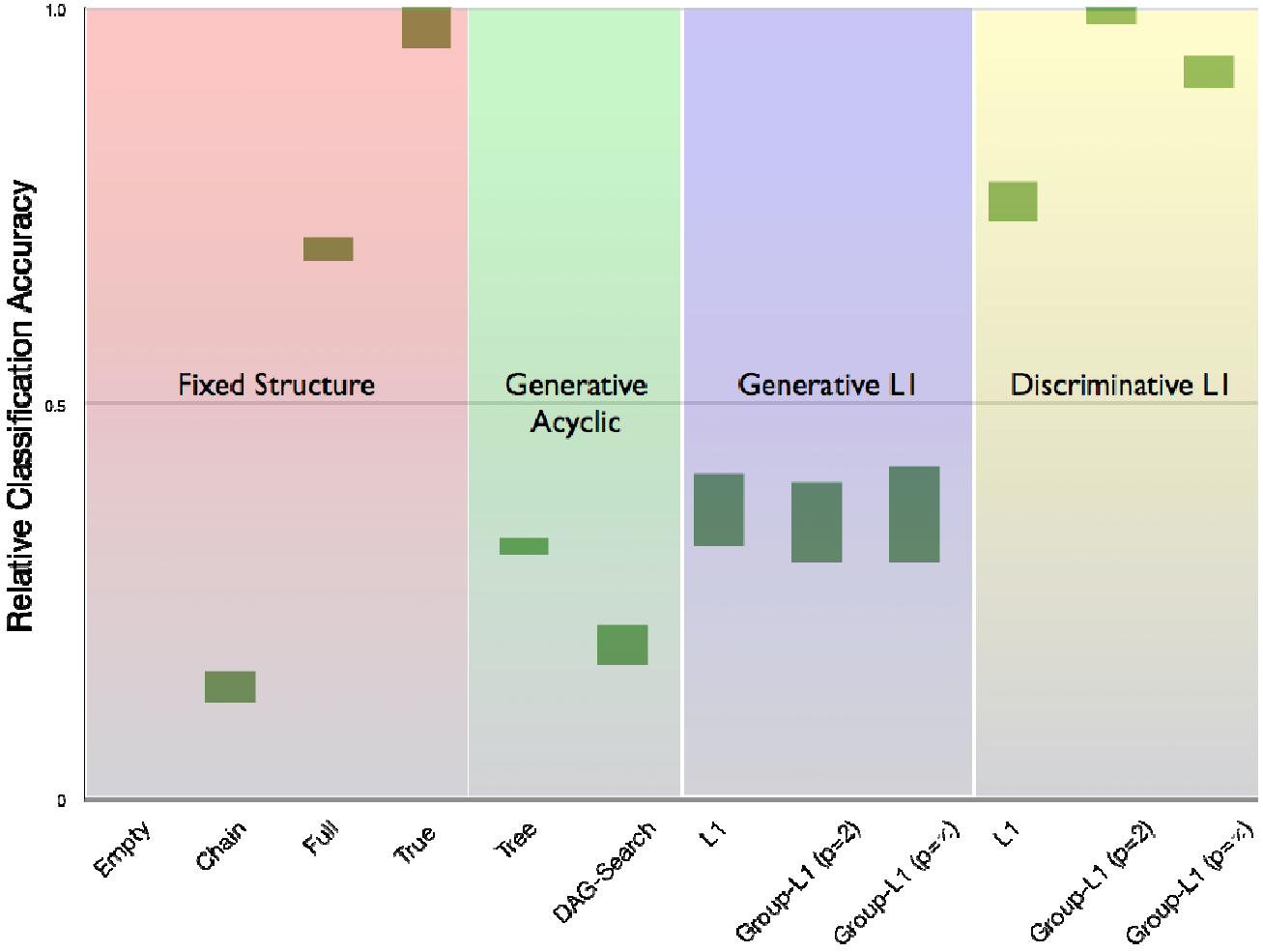
$$\hat{y}_j \quad = \quad \arg\max p(y_j|\mathbf{x}, \mathbf{w}, G)$$

- First learn (or fix) G, then learn w given G
  - Empty, chain, full, true
  - Best DAG (greedy search), best tree (Chow-Liu)
  - max p(y|w) $||w||_1$, $||w||_2$, $||w||_\infty$
- Jointly learn G and w
  - Max p(y|x,w,v) $||w||_1$, $||w||_2$, $||w||_\infty$

Results on synthetic data

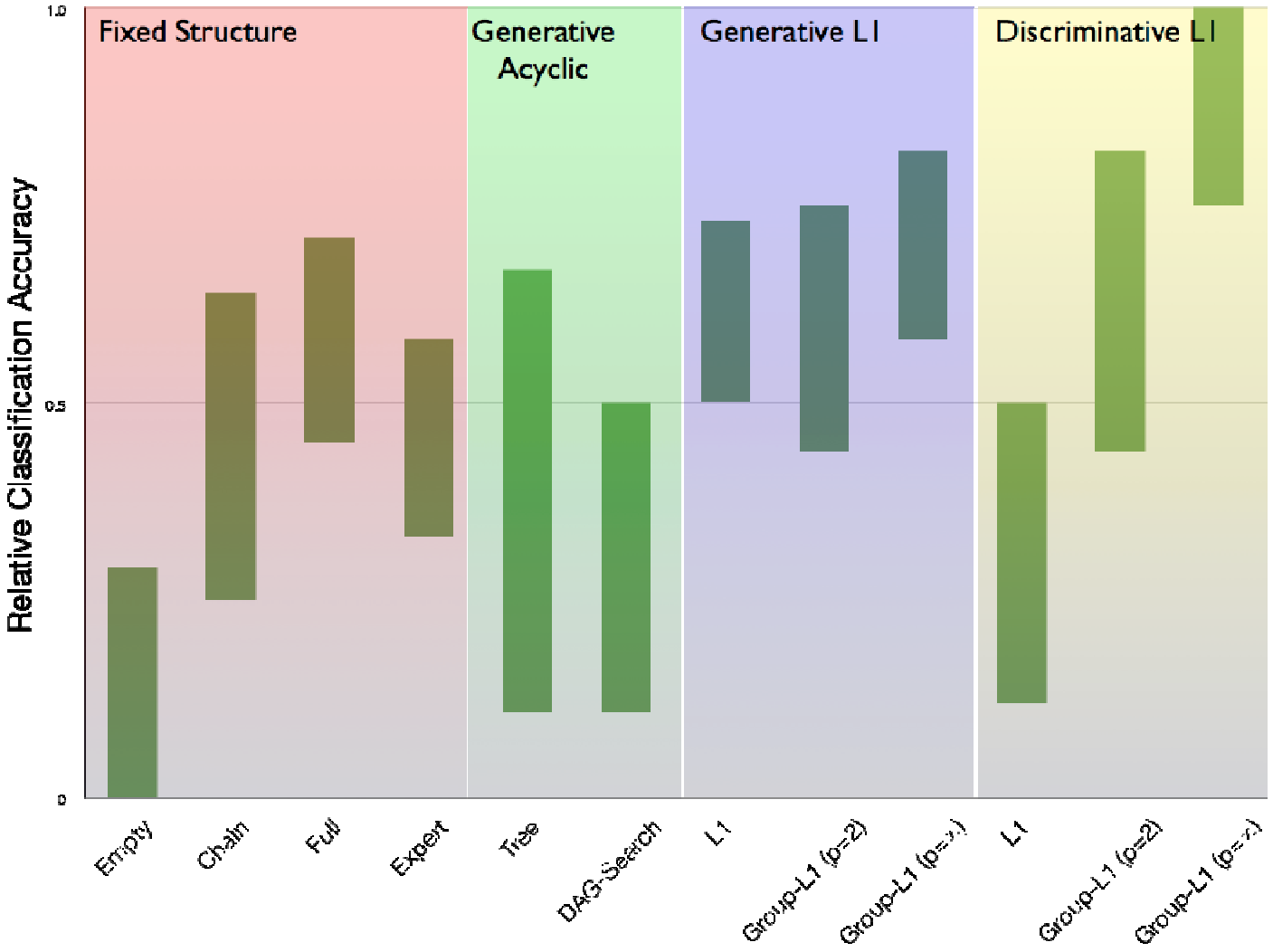
- d=10, n=500 train, 1000 test

90% confidence interval derived from 10 random trials



Results on heart data

90% confidence interval derived from 10-fold cross validation



Incremental feature addition

- Lee, Ganapathi & Koller compute gradient and expectations using LBP instead of PL
- They greedily add features according to their expected gain (change in penalized loglik)
- Initially the graph is sparse so LBP is accurate, but degrades over time

Della Pietra

Can use Gibbs sampling + IS corrections
Della Pietra, Della Pietra, Lafferty, PAMI 1997

```
m, r, xevo, ijjiir, b, to, jz, gsr, wq, vf, x, ga,
msmGh, pcp, d, oziVlal, hzagh, yzop, io, advzmxnv,
ijv_bolft, x, emx, kayerf, mlj, rawzyb, jp, ag,
ctdnnnbg, wgdw, t, kguv, cy, spxcq, uzflbbf,
dxtkkn, cxwx, jpd, ztzh, lv, zhpkvnu, l^, r, qee,
nynrx, atze4n, ik, se, w, lrh, hp+, yrqyka'h,
zcngotcnx, igcump, zjcjs, lqpWiqu, cefmfhc, o, lb,
fdcY, tzby, yopxmvk, by, fz,, t, govyccm,
ijyiduwfzo, 6xr, duh, ejv, pk, pjw, l, fl, w
```

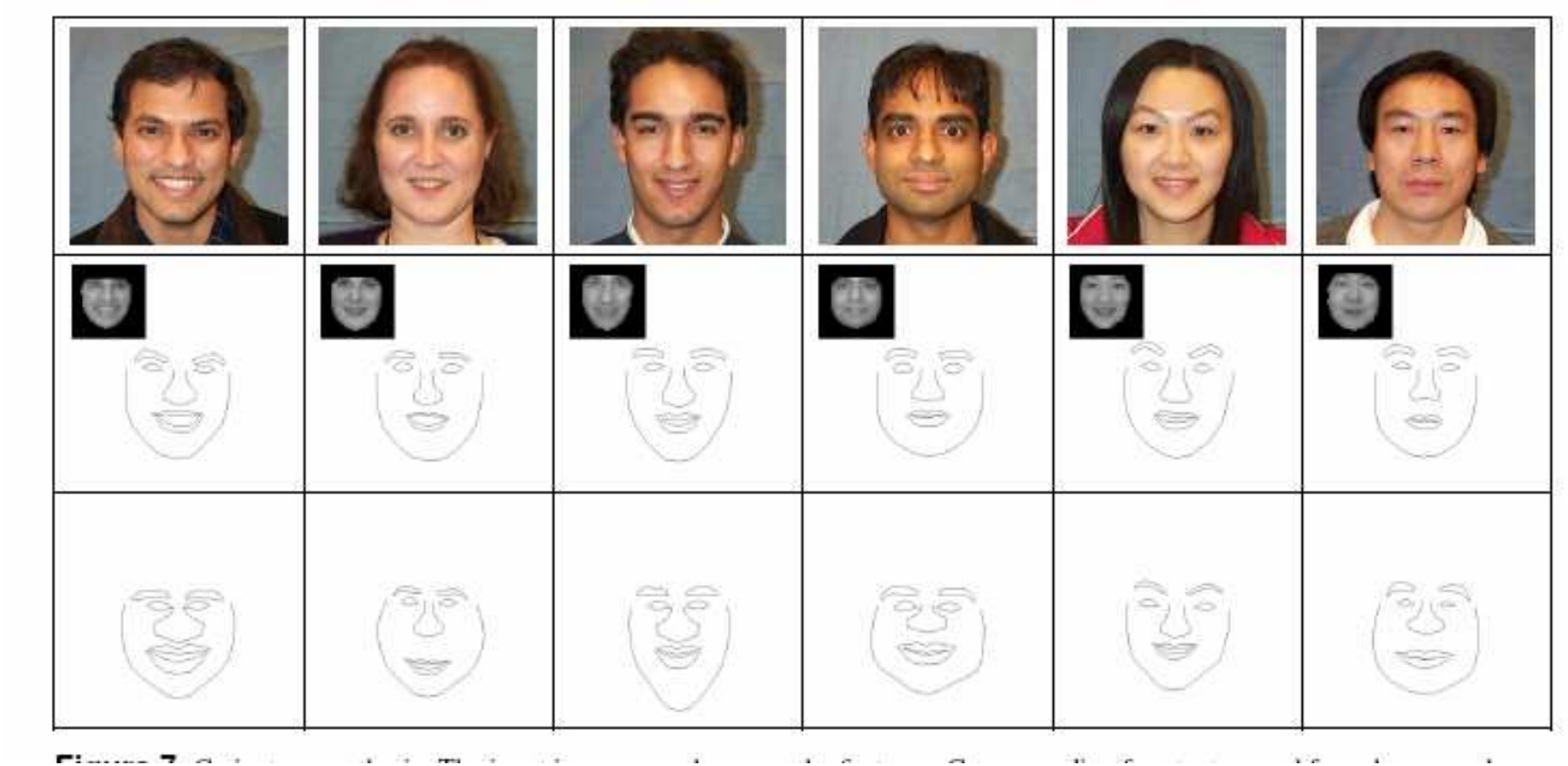
The second most important feature, according to the algorithm, i
that two adjacent lower-case characters are extremely common
The second-order field now becomes

$$p(\omega) = \frac{1}{Z} e^{\sum_{i,j} \lambda_{[a-z][a-z]} \chi_{[a-z][a-z]}(\omega_{ij}) + \sum_i \lambda_{[a-z]} \chi_{[a-z]}(\omega_i)}$$

The first 1000 features that the algorithm induces include the strings `s>`, `<re`, `ly>`, and `ing>`, where the character "<" denotes beginning-of-string and the character ">" denotes end-of-string. In addition, the first 1000 features include the regular expressions `[0-9] [0-9]` (with weight 9.15) and `[a-z] [A-Z]` (with weight −5.81) in addition to the first two features `[a-z]` and `[a-z] [a-z]`. A set of strings obtained by Gibbs sampling from the resulting field is shown here:

```
was, reaser, in, there, to, will, ,, was, by,
homes, thing, be, reloverated, ther, which,
conists, at, fores, anditing, with, Mr., proveral,
the, ,, ***, on't, prolling, prothere, ,, mento,
at, yaou, 1, chestraing, for, have, to, intrally,
of, qut, ., best, compers, ***, cluseliment, uster,
of, is, deveral, this, thise, of, offect, inatever,
thifer, constranded, stater, vill, in, thase, in,
youse, menttering, and, ., of, in, verate, of, to
```

Maxent models of faces



Use importance sampling to reweight the Gibbs samples when evaluating feature gain

C. Liu and S.C. Zhu and H.Y. Shum, ICCV 2001