CPSC 540: Machine Learning

More DAGs

Mark Schmidt

University of British Columbia

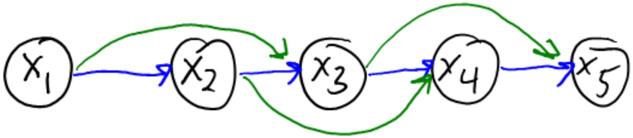
Winter 2020

Last Time: Directed Acyclic Graphical (DAG) Models

- DAG models use a factorization of the joint distribution,

$$p(x_1, x_2, \ldots, x_d) = \prod_{j=1}^{d} p(x_j \mid x_{\text{pa}(j)}),$$

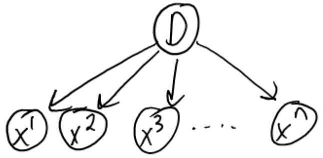
- We visualize the assumptions made by the model as a graph:



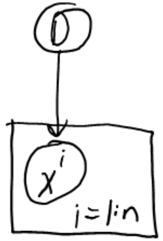
- D-separation can be used to "read" conditional independence from graph.
    - Can be derived by considering DAG as "inheritance of genes".

Plate Notation

- Graphical representation of the IID assumption:

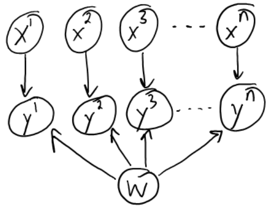


- It's common to represent repeated parts of graphs using plate notation:

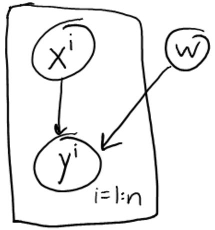


Tilde Notation as a DAG

- If the x^i are IID then we can represent regression as



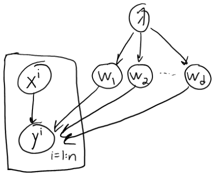
or



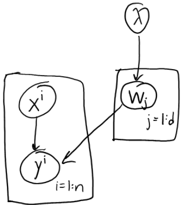
- From d-separation on this graph we have $p(y \mid X, w) = \prod_{i=1}^{n} p(y^i \mid x^i, w)$.
- We often omit the data-generating distribution D.
  - But if you want to learn then you should remember that it's there.
- Note that plate reflects parameter tieing: that we use same w for all i.

Tilde Notation as a DAG

- When we do MAP estimation under the assumptions
$$y^i \sim \mathcal{N}(w^T x^i, 1), \quad w_j \sim \mathcal{N}(0, 1/\lambda),$$
we can interpret it as the DAG model:



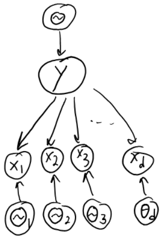
- Or introducing a second plate using:



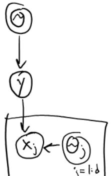
Other Models in DAG/Plate Notation

- For naive Bayes we have

$$y^i \sim \text{Cat}(\theta), \quad x^i \mid y^i = c \sim Cat(\theta_c).$$



- Or in plate notation as



Outline

Parameter Learning in General DAG Models

- The log-likelihood in DAG models is separable in the conditionals,

$$\begin{aligned}\log p(x \mid \Theta) &= \log \prod_{j=1}^{d} p(x_j \mid x_{\mathsf{pa}(j)}, \Theta_j) \\ &= \sum_{j=1}^{d} \log p(x_j \mid x_{\mathsf{pa}(j)}, \Theta_j)\end{aligned}$$

- If each $p(x_j \mid x_{\mathsf{pa}(j)})$ has its own parameters Θ_j, we can fit them independently.
  - We've done this before: naive Bayes, Gaussian discriminant analysis, mixtures, etc.

- Sometimes you want to have tied parameters $(\Theta_j = \Theta_{j'})$
  - Homogeneous Markov chains, Gaussian discriminant analysis with shared covariance.
  - Still easy, but need to fit $p(x_j \mid x_{\mathsf{pa}(j)}, \Theta_j)$ and $p(x_{j'} \mid x_{\mathsf{pa}(j')}, \Theta_j)$ together.

Tabular Parameterization in DAG Models

- To specify distribution, we need to decide on the form of $p(x_j \mid x_{\text{pa}(j)}, \Theta_j)$.
- For discrete data a default choice is the tabular parameterization:

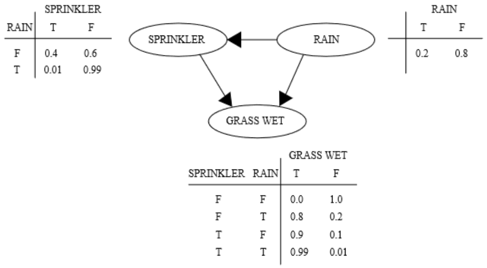
  $$p(x_j \mid x_{\text{pa}(j)}, \Theta_j) = \theta_{x_j, x_{\text{pa}(j)}} \quad \text{(one parameter per child/parent combo)},$$

  as we did for Markov chains (but now with multiple parents).
- Intuitive: just need conditional probabilities of children given parents like

  $$p(\text{“wet grass”} = 1 \mid \text{“sprinkler”} = 1, \text{“rain”} = 0),$$

  and MLE is just counting.

Tabular Parameterization Example



	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

RAIN	
T	F
0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

https://en.wikipedia.org/wiki/Bayesian_network

Some quantities can be directly read from the tables:

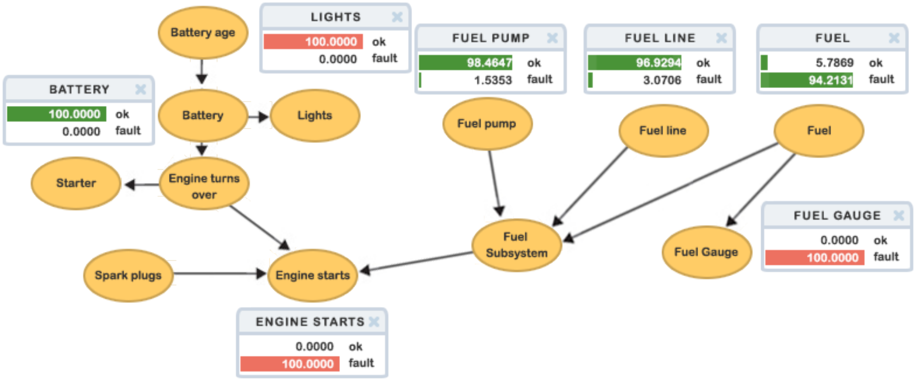
$$p(R = 1) = 0.2.$$

$$p(G = 1 \mid S = 0, R = 1) = 0.8.$$

Can calculate any probabilities using marginalization/product-rule/Bayes-rule (bonus).

Tabular Parameterization Example

Some companies sell software to help companies reason using tabular DAGs:



http://www.hugin.com/index.php/technology

Fitting DAGs using Supervised Learning

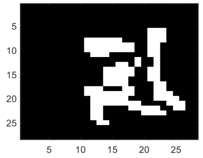
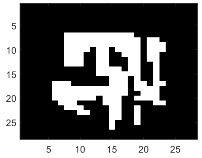
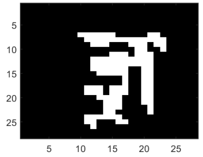
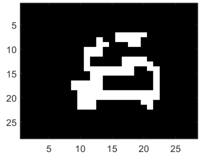
- But tabular parameterization requires too many parameters:
  - With binary states and k parents, need 2^{k+1} parameters.
- One solution is letting users specify a "parsimonious" parameterization:
  - Typically have a linear number of parameters.
  - For example, the "noisy-or" model: $p(x_j \mid x_{\text{pa}(j)}) = 1 - \prod_{k \in \text{pa}(j)} q_k$.
    - "Estimate probability that each symptom leads to disease on its own".
- But if we have data, we can use supervised learning.
  - Write fitting $p(x_j \mid x_{\text{pa}(j)})$ as our usual $p(y \mid x)$ problem.
    - Predicting one column of X given the values of some other columns.

Fitting DAGs using Supervised Learning

- For $j = 1 : d$:
  1. Set $\bar{y}^i = x^i_j$ and $\bar{x}^i = x^i_{\text{pa}(j)}$.
  2. Solve a supervised learning problem using $\{\bar{X}, \bar{y}\}$.
     - Gives you a model of $p(x_j \mid x_{\text{pa}(j)})$.
- Use the d regression/classification models as the density estimator.
- We've turned unsupervised learning into supervised learning.
- We can use our usual tricks:
  - Linear models, non-linear bases, regularization, kernel trick, random forests, etc.
  - With least squares it's called a Gaussian belief network.
  - With logistic regression it's called a sigmoid belief networks.
  - Don't need Markov assumptions to tractably fit these models.

MNIST Digits with Tabular DAG Model

- Recall our latest MNIST model using a tabular DAG:

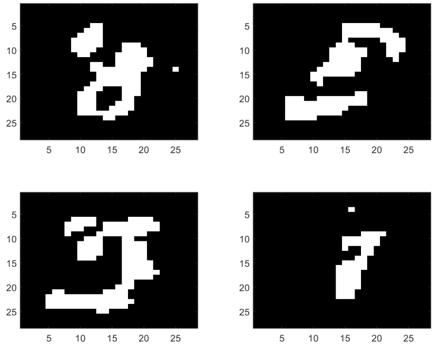





- This model is pretty bad because you only see 8 parents.

MNIST Digits with Sigmoid Belief Network

- Samples from sigmoid belief network:

(DAG with logistic regression for each variable)



where we use all previous pixels as parents (from 0 to 783 parents).

- Models long-range dependencies but has a linear assumption.

DAGs: Big Picture

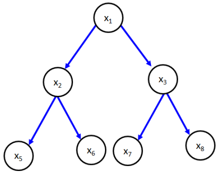
- Setting the parameters of a DAG model:
  - Get the graph from an expert, or learn the graph (later).
  - Given the conditional probabilities from an expert, or learn them from data.
    - Counting or supervised learning, and EM if you have hidden/missing values.
- Inference in DAG models:
  - Can use Monte Carlo approximations with ancestral sampling:
    - Sample x_1 from $p(x_1)$, x_2 from $p(x_2 \mid x_{\text{pa}(2)})$, x_3 from $p(x_3 \mid x_{\text{pa}(3)})$,...
  - Can use dynamic programming for exact inference with discrete x_j.
    - Also works if all $p(x_j \mid x_{\text{pa}(j)})$ are Gaussian.

Inference in Forest DAGs

- If we try to generalize the CK equations to DAGs we obtain

$$p(x_j = s) = \sum_{x_{\text{pa}(j)}} p(x_j = s, x_{\text{pa}(j)}) = \sum_{x_{\text{pa}(j)}} \underbrace{p(x_j = s \mid x_{\text{pa}(j)})}_{\text{given}} p(x_{\text{pa}(j)}).$$

  which works if each node has at most one parent.
  - Such graphs are called trees (connected), or forests (disconnected).
    - Also called "singly-connected".



  - Forests allow efficient dynamic programming methods as in Markov chains.
    - In particular, decoding and univariate marginals/conditionals in $O(dk^2)$.
    - Message passing applied to tree-structured graphs is called belief propagation.

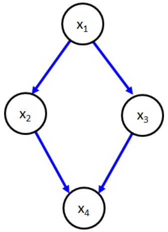
Inference in General DAGs

- If we try to generalize the CK equations to DAGs we obtain

$$p(x_j = s) = \sum_{x_{\text{pa}(j)}} p(x_j = s, x_{\text{pa}(j)}) = \sum_{x_{\text{pa}(j)}} \underbrace{p(x_j = s \mid x_{\text{pa}(j)})}_{\text{given}} p(x_{\text{pa}(j)}).$$

- What goes wrong if nodes have multiple parents?
  - The expression $p(x_{\text{pa}(j)})$ is a joint distribution depending on multiple variables.

- Consider the non-tree graph:



Inference in General DAGs

- We can compute $p(x_4)$ in this non-tree using:

$$\begin{aligned} p(x_4) &= \sum_{x_3}\sum_{x_2}\sum_{x_1} p(x_1, x_2, x_3, x_4) \\ &= \sum_{x_3}\sum_{x_2}\sum_{x_1} p(x_4 \mid x_2, x_3)p(x_3 \mid x_1)p(x_2 \mid x_1)p(x_1) \\ &= \sum_{x_3}\sum_{x_2} p(x_4 \mid x_2, x_3) \underbrace{\sum_{x_1} p(x_3 \mid x_1)p(x_2 \mid x_1)p(x_1)}_{M_{23}(x_2,x_3)} \end{aligned}$$

- Dependencies between $\{x_1, x_2, x_3\}$ mean our message depends on two variables.

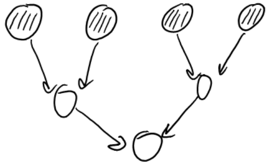
$$\begin{aligned} p(x_4) &= \sum_{x_3}\sum_{x_2} p(x_4 \mid x_2, x_3)M_{23}(x_2, x_3) \\ &= \sum_{x_3} M_{34}(x_3, x_4), \end{aligned}$$

Inference in General DAGs

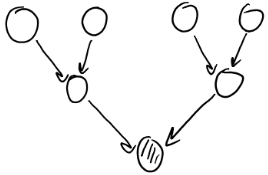
- With 2-variable messages, our cost increases to $O(dk^3)$.
- If we add the edge $x_1 -> x_4$, then the cost is $O(dk^4)$.
  (the same cost as enumerating all possible assignments)
- Unfortunately, cost is not as simple as counting number of parents.
  - Even if each node has 2 parents, we may need huge messages.
  - Decoding is NP-hard and computing marginals is #P-hard in general.
  - We'll see later that maximum message size is "treewidth" of a particular graph.
- On the other hand, ancestral sampling is easy:
  - We can obtain Monte Carlo estimates of solutions to these NP-hard problems.

Conditional Sampling in DAGs

- What about conditional sampling in DAGs?
  - Could be easy or hard depending on what we condition on.
- For example, easy if we condition on the first variables in the order:
  - Just fix these and run ancestral sampling.



- Hard to condition on the last variables in the order:
  - Conditioning on descendent makes ancestors dependent.

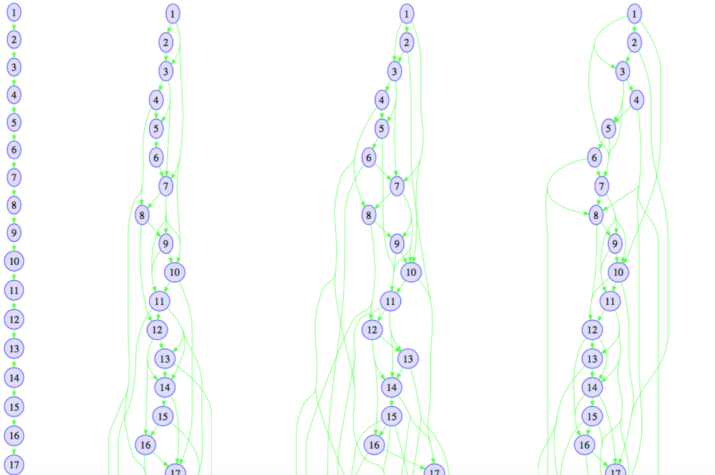


DAG Structure Learning

- Structure learning is the problem of choosing the graph.
  - Input is data X.
  - Output is a graph G.
- The "easy" case is when we're given the ordering of the variables.
  - So the parents of j must be chosen from $\{1, 2, \ldots, j-1\}$.
- Given the ordering, structure learning reduces to feature selection:
  - Select features $\{x_1, x_2, \ldots, x_{j-1}\}$ that best predict "label" x_j.
  - We can use any feature selection method to solve these d problems.

Example: Structure Learning in Rain Data Given Ordering

- Structure learning in rain data using L1-regularized logistic regression.
  - For different λ values, assuming chronological ordering.

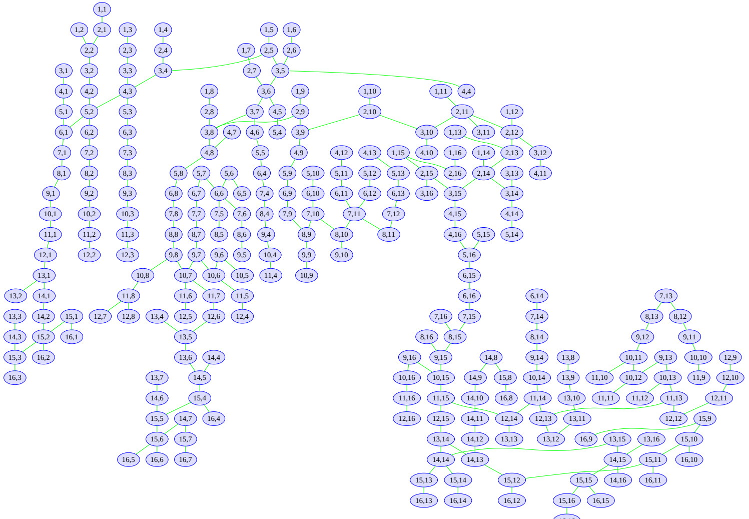


DAG Structure Learning without an Ordering

- Without an ordering, a common approach is "search and score"
  - Define a score for a particular graph structure (like BIC or other L0-regularizers).
  - Search through the space of possible DAGs.
    - "DAG-Search": at each step greedily add, remove, or reverse an edge.
- May have equivalent graphs with the same score (don't trust edge direction).
  - Do not interpret causally a graph learned from data.
- Structure learning is NP-hard in general, but finding the optimal tree is poly-time:
  - For symmetric scores, can be found by minimum spanning tree ("Chow-Liu").
  - For asymetric scores, can be found by minimum spanning arborescence.

Structure Learning on USPS Digits

An optimal tree on USPS digits (16 by 16 images of digits).



20 Newsgroups Data

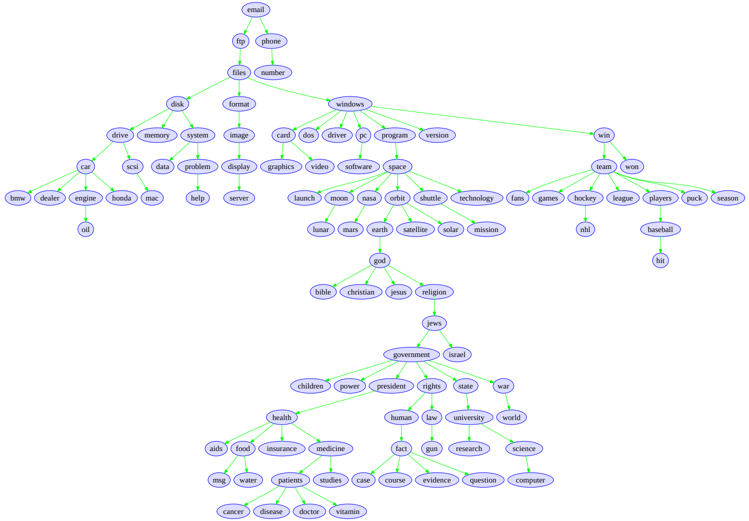
- Data containing presence of 100 words from newsgroups posts:

car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

- Structure learning should give some relationship between word occurrences.

Structure Learning on News Words

Optimal tree on newsgroups data:



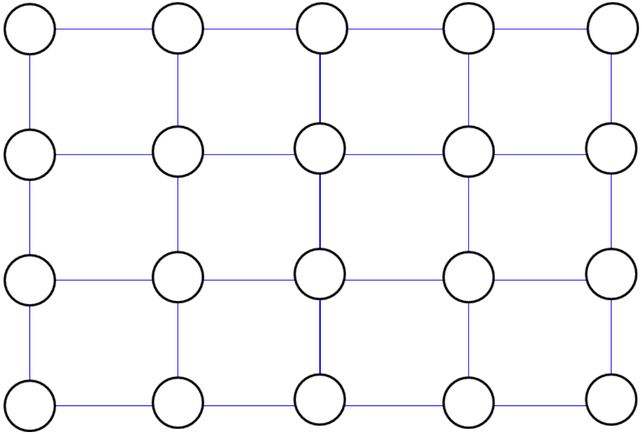
Outline

Directed vs. Undirected Models

- In some applications we have a natural ordering of the x_j.
  - In the “rain” data, the past affects the future.
- In some applications we don’t have a natural order.
  - E.g., pixels in an image.
- In these settings we often use undirected graphical models (UGMs).
  - Also known as Markov random fields (MRFs) and originally from statistical physics.

Directed vs. Undirected Models

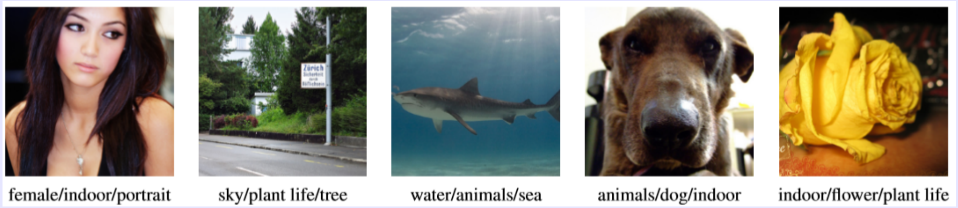
- Undirected graphical models are based on undirected graphs:



- They are a classic way to model dependencies in images:
  - Can capture dependencies between neighbours without imposing an ordering.

Multi-Label Classification

- Consider multi-label classification:

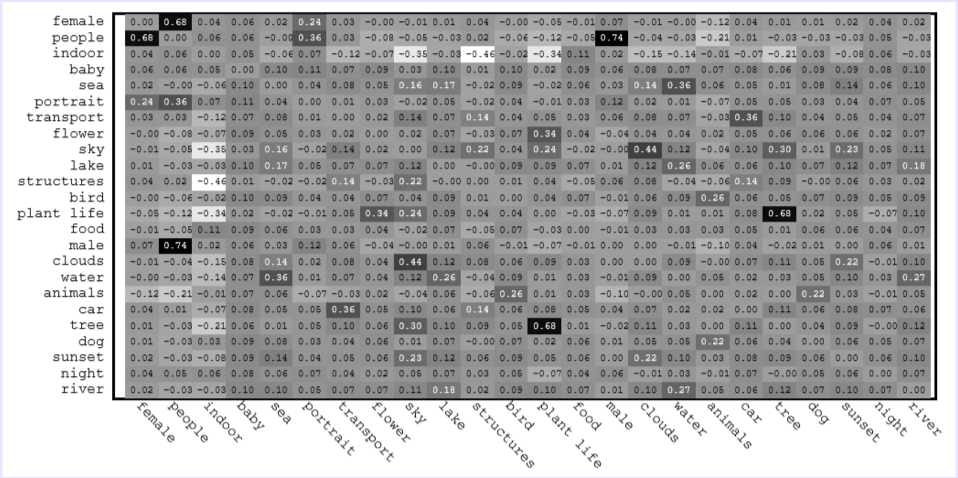


http://proceedings.mlr.press/v37/chenb15.pdf

- Flickr dataset: each image can have multiple labels (out of 38 possibilities).
- Use neural networks to generate "factors" in an undirected model.
  - Decoding undirected model makes predictions accounting for label correlations.
  - We'll discuss how neural networks and density models fit together later.

Multi-Label Classification

- Learned correlation matrix:



	female	people	indoor	baby	sea	portrait	transport	flower	sky	lake	structures	bird	plant life	food	male	clouds	water	animals	car	tree	dog	sunset	night	river
female	0.00	0.68	0.04	0.06	0.02	0.24	0.03	−0.00	−0.01	0.01	0.04	−0.00	−0.05	−0.01	0.07	−0.01	−0.00	−0.12	0.04	0.01	0.01	0.02	0.04	0.02
people	0.68	0.00	0.06	0.06	−0.00	0.36	0.03	−0.08	−0.05	−0.03	0.02	−0.06	−0.12	−0.05	0.74	−0.04	−0.03	−0.21	0.01	−0.03	−0.03	−0.03	0.05	−0.03
indoor	0.04	0.06	0.00	0.05	−0.06	0.07	−0.12	−0.07	−0.35	−0.03	−0.46	−0.02	−0.34	0.11	0.02	−0.15	−0.14	−0.01	−0.07	−0.21	0.03	−0.08	0.06	−0.03
baby	0.06	0.06	0.05	0.00	0.10	0.11	0.07	0.09	0.03	0.10	0.01	0.10	0.02	0.09	0.06	0.08	0.07	0.07	0.08	0.06	0.09	0.09	0.08	0.10
sea	0.02	−0.00	−0.06	0.10	0.00	0.04	0.08	0.05	0.16	0.17	−0.02	0.09	−0.02	0.06	0.03	0.14	0.36	0.06	0.05	0.01	0.08	0.14	0.06	0.10
portrait	0.24	0.36	0.07	0.11	0.04	0.00	0.01	0.03	−0.02	0.05	−0.02	0.04	−0.01	0.03	0.12	0.02	0.01	−0.07	0.05	0.05	0.03	0.04	0.07	0.05
transport	0.03	0.03	−0.12	0.07	0.08	0.01	0.00	0.02	0.14	0.07	0.14	0.04	0.05	0.03	0.06	0.08	0.07	−0.03	0.36	0.10	0.04	0.05	0.04	0.07
flower	−0.00	−0.08	−0.07	0.09	0.05	0.03	0.02	0.00	0.02	0.07	−0.03	0.07	0.34	0.04	−0.04	0.04	0.04	0.02	0.05	0.06	0.06	0.06	0.02	0.07
sky	−0.01	−0.05	−0.35	0.03	0.16	−0.02	0.14	0.02	0.00	0.12	0.22	0.04	0.24	−0.02	−0.00	0.44	0.12	−0.04	0.10	0.30	0.01	0.23	0.05	0.11
lake	0.01	−0.03	−0.03	0.10	0.17	0.05	0.07	0.07	0.12	0.00	−0.00	0.09	0.09	0.07	0.01	0.12	0.26	0.06	0.06	0.10	0.07	0.12	0.07	0.18
structures	0.04	0.02	−0.46	0.01	−0.02	−0.02	0.14	−0.03	0.22	−0.00	0.00	0.01	0.04	−0.05	0.06	0.08	−0.04	−0.06	0.14	0.09	−0.00	0.06	0.03	0.02
bird	−0.00	−0.06	−0.02	0.10	0.09	0.04	0.04	0.07	0.04	0.09	0.01	0.00	0.04	0.07	−0.01	0.06	0.09	0.26	0.06	0.05	0.07	0.09	0.05	0.09
plant life	−0.05	−0.12	−0.34	0.02	−0.02	−0.01	0.05	0.34	0.24	0.09	0.04	0.04	0.00	−0.03	−0.07	0.09	0.01	0.01	0.08	0.68	0.02	0.05	−0.07	0.10
food	−0.01	−0.05	0.11	0.09	0.06	0.03	0.03	0.04	−0.02	0.07	−0.05	0.07	−0.03	0.00	−0.01	0.03	0.03	0.03	0.05	0.01	0.06	0.06	0.04	0.07
male	0.07	0.74	0.02	0.06	0.03	0.12	0.06	−0.04	−0.00	0.01	0.06	−0.01	−0.07	−0.01	0.00	0.00	−0.01	−0.10	0.04	−0.02	0.01	0.00	0.06	0.01
clouds	−0.01	−0.04	−0.15	0.08	0.14	0.02	0.08	0.04	0.44	0.12	0.08	0.06	0.09	0.03	0.00	0.00	0.09	−0.00	0.07	0.11	0.05	0.22	−0.01	0.10
water	−0.00	−0.03	−0.14	0.07	0.36	0.01	0.07	0.04	0.12	0.26	−0.04	0.09	0.01	0.03	−0.01	0.09	0.00	0.05	0.02	0.03	0.05	0.10	0.03	0.27
animals	−0.12	−0.21	−0.01	0.07	0.06	−0.07	−0.03	0.02	−0.04	0.06	−0.06	0.26	0.01	0.03	−0.10	−0.00	0.05	0.00	0.02	0.00	0.22	0.03	−0.01	0.05
car	0.04	0.01	−0.07	0.08	0.05	0.05	0.36	0.05	0.10	0.06	0.14	0.06	0.08	0.05	0.04	0.07	0.02	0.02	0.00	0.11	0.06	0.08	0.07	0.06
tree	0.01	−0.03	−0.21	0.06	0.01	0.05	0.10	0.06	0.30	0.10	0.09	0.05	0.68	0.01	−0.02	0.11	0.03	0.00	0.11	0.00	0.04	0.09	−0.00	0.12
dog	0.01	−0.03	0.03	0.09	0.08	0.03	0.04	0.06	0.01	0.07	−0.00	0.07	0.02	0.06	0.01	0.05	0.05	0.22	0.06	0.04	0.00	0.06	0.05	0.07
sunset	0.02	−0.03	−0.08	0.09	0.14	0.04	0.05	0.06	0.23	0.12	0.06	0.09	0.05	0.06	0.00	0.22	0.10	0.03	0.08	0.09	0.06	0.00	0.06	0.10
night	0.04	0.05	0.06	0.08	0.06	0.07	0.04	0.02	0.05	0.07	0.03	0.05	−0.07	0.04	0.06	−0.01	0.03	−0.01	0.07	−0.00	0.05	0.06	0.00	0.07
river	0.02	−0.03	−0.03	0.10	0.10	0.05	0.07	0.07	0.11	0.18	0.02	0.09	0.10	0.07	0.01	0.10	0.27	0.05	0.06	0.12	0.07	0.10	0.07	0.00

http://proceedings.mlr.press/v37/chenb15.pdf

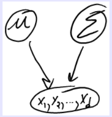
Summary

- Plate Notation lets us compactly draw graphs with repeated patterns.
  - There are fancier versions of plate notation called "probabilistic programming".
- Parameter learning in DAGs:
  - Can fit each $p(x_j \mid x_{\text{pa}(j)})$ independently.
  - Tabular parameterization, or treat as supervised learning.
- Inference in DAGs:
  - Ancestral sampling and Monte Carlo methods work as before.
  - Message-passing message sizes depend on graph structure.
- Structure learning is the problem of learning the graph structure.
  - Hard in general, but easy for trees and L1-regularization gives fast heuristic.
- Next time: undirected models.

Other Models in DAG/Plate Notation

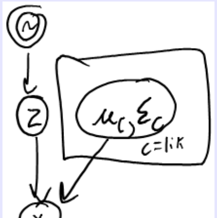
- In a full Gaussian model for a single x we have

$$x^i \sim \mathcal{N}(\mu, \Sigma).$$

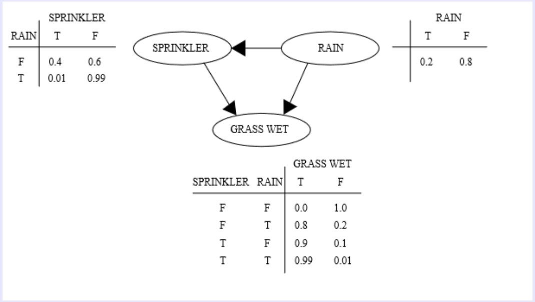


- For mixture of Gaussians we have

$$z^i \sim \text{Cat}(\theta), \quad x^i \mid z^i = c \sim \mathcal{N}(\mu_c, \Sigma_c).$$



Tabular Parameterization Example



	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

RAIN	
T	F
0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

https://en.wikipedia.org/wiki/Bayesian_network

Can calculate any probabilities using marginalization/product-rule/Bayes-rule, for example:

$$\begin{aligned} p(G=1 \mid R=1) &= p(G=1, S=0 \mid R=1) + p(G=1, S=1 \mid R=1) \qquad \left(p(a \mid c) = \sum_b p(a, b \mid c)\right) \\ &= p(G=1 \mid S=0, R=1)p(S=0 \mid R=1) + p(G=1 \mid S=1, R=1)p(S=1 \mid R=1) \\ &= 0.8(0.99) + 0.99(0.01) = 0.81. \end{aligned}$$

Dynamic Bayesian Networks

- Dynamic Bayesian networks are a generalization of Markov chains and DAGs:
  - At each time, we have a set of variables x^t.
  - The initial x^0 comes from an "initial" DAG.
  - Given x^{t-1}, we generate x^t from a "transition" DAG.

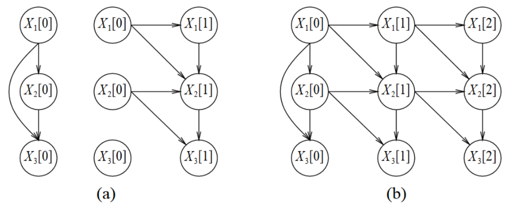


Figure 1: (a) A prior network and transition network defining a DPN for the attributes X_1, X_2, X_3. (b) The corresponding "unrolled" network.

https://www.cs.ubc.ca/~murphyk/Papers/dbnsem_uai98.pdf

- Can be used to model multiple variables over time.
  - Unconditional sampling is easy but inference may be hard.

DAG Structure Learning without an Ordering

- Another common structure learning approach is "constraint-based":
  - Based on performing a sequence of conditional independence tests.
  - Prune edge between x_i and x_j if you find variables S making them independent,

$$x_i \perp x_j \mid x_S.$$

  - Challenge is considering exponential number of sets x_S (heuristic: "PC algorithm").
  - Assumes "faithfulness" (all independences are reflected in graph).
    - Otherwise it's weird (a duplicated feature would be disconnected from everything.)