# CPSC 440/540: Advanced Machine Learning

## More DAGs

Mark Schmidt
(using materials by Danica Sutherland (building on materials from Mark Schmidt))

University of British Columbia

Winter 2023

# Reminders

admin

- Project proposals due **Friday**
- Assignment 3 due **Monday**
- Assignment 4 released definitely by Monday, probably sooner

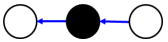
# Last Time: DAG models

- Directed acyclic graphical models: $p(x) = \prod_{j=1}^{d} p(x_j \mid x_{\mathrm{pa}(j)})$
  - $\mathrm{pa}(j) \subseteq \{1, \ldots, j-1\}$ is the set of parents of $j$
  - Generalizes Markov chains (use $\mathrm{pa}(j) = \{j-1\}$)
  - Every possible distribution can be written as one (use $\mathrm{pa}(j) = \{1, \ldots, j-1\}$)

- Defines a graph (one node per $x_j$, edges from parents to children)
- Started $d$-separation to read conditional independences off of that graph

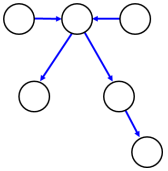
# D-Separation Summary (MEMORIZE)

- Checking whether DAG implies $A$ is independent of $B$ given $C$:
  - Consider each undirected path from any node in any $A$ to any node in $B$.
    - Ignoring directions and observations.
  - Use directions/observations, check if any of below hold somewhere along each path:
    1. $P$ includes a "chain" with an observed middle node (e.g., Markov chain):



    2. $P$ includes a "fork" with an observed parent node (e.g., naive Bayes):

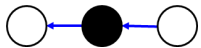
    3. $P$ includes a "v-structure" or "collider" (e.g., genetic inheritance):



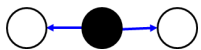
    where the "child" and all its descendants are unobserved.
  - If all paths are blocked by one of above, DAG implies the conditional independence.

# D-Separation Summary (MEMORIZE)

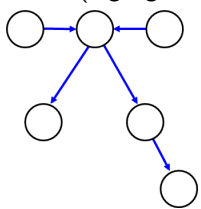
- We say that $A$ and $B$ are d-separated (conditionally independent) given $C$ if *all undirected paths* from $A$ to $B$ are "blocked" because *one* of the following holds *somewhere* on the path:
  1. $P$ includes a "chain" with an observed middle node (e.g., Markov chain):



  2. $P$ includes a "fork" with an observed parent node (e.g., naive Bayes):



  3. $P$ includes a "v-structure" or "collider" (e.g., genetic inheritance):

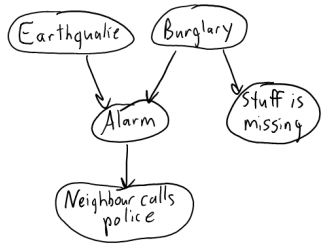


where the "child" and all its descendants are unobserved.

# Alarm Example


- Case 1:
  - Earthquake $\not\perp\!\!\!\perp$ Call.
  - Earthquake $\perp\!\!\!\perp$ Call | Alarm.
- Case 2:
  - Alarm $\not\perp\!\!\!\perp$ Stuff Missing.
  - Alarm $\perp\!\!\!\perp$ Stuff Missing | Burglary.

# Alarm Example




- Case 3:
  - Earthquake $\perp\!\!\!\perp$ Burglary.
  - Earthquake $\not\perp\!\!\!\perp$ Burglary | Alarm.
    - "Explaining away": knowing one parent can make the other less/more likely.
- Multiple Cases:
  - Call $\not\perp\!\!\!\perp$ Stuff Missing.
  - Earthquake $\perp\!\!\!\perp$ Stuff Missing.
  - Earthquake $\not\perp\!\!\!\perp$ Stuff Missing | Call.

# Discussion of D-Separation

- D-separation lets you say if conditional independence is implied by assumptions:

$$(A \text{ and } B \text{ are d-separated given } C) \Rightarrow A \perp\!\!\!\perp B \mid C.$$

- However, there might be extra conditional independences in the distribution:
  - These would depend on specific choices of the DAG parameters.
    - For example, if we set Markov chain parameters so that $p(x_j \mid x_{j-1}) = p(x_j)$.
  - Or some *orderings* of the chain rule may reveal different independences.
  - Lack of d-separation doesn't imply dependence.
    - Just that it's not guaranteed to be independent by the graph structure.

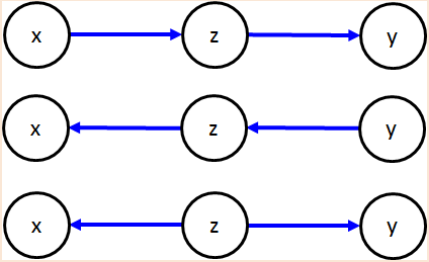
- Instead of restricting to $\{1, 2, \ldots, j-1\}$, can have general parent choices.
  - So $x_2$ could be a parent of $x_1$.
- As long the graph is acyclic, there exists a valid ordering (chain rule makes sense).

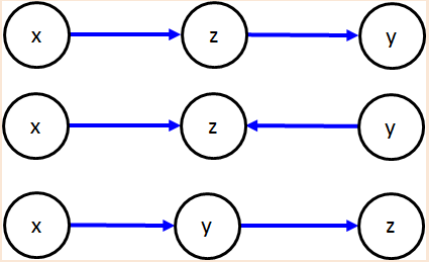
(all DAGs have a "topological order" of variables where parents are before children)

# Non-Uniqueness of Graph and Equivalent Graphs

bonus!

- Note that some graphs imply same conditional independences:
  - Equivalent graphs: same v-structures and other (undirected) edges are the same.
  - Examples of 3 *equivalent* graphs (left) and 3 non-equivalent graphs (right):



# Beware of the "Causal" DAG

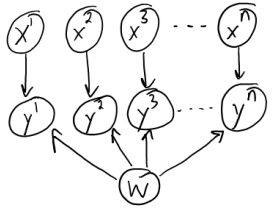
bonus!

- It can be helpful to use the language of causality when reasoning about DAGs.
  - You'll find that they give the correct causal interpretation based on our intuition.
- However, keep in mind that the arrows are not necessarily causal.
  - "$A$ causes $B$" can have the same graph as "$B$ causes $A$"!
- There is work on causal DAGs which add semantics to deal with "interventions".
  - But these require assuming that the arrow directions are causal.
    - Fitting a DAG to observational data doesn't imply anything about causality.

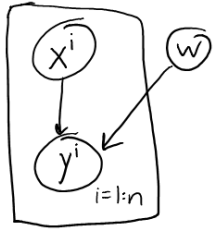
# Outline

# Linear Regression

- As we saw last time, if the $x^i$ are IID, then we can represent linear regression as




or

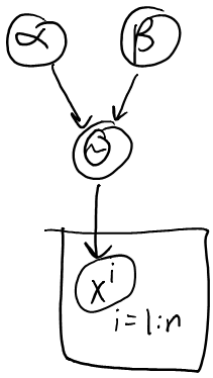



- From $d$-separation on this graph we have $p(\mathbf{y} \mid \mathbf{X}, w) = \prod_{i=1}^{n} p(y^i \mid x^i, w)$.
  - Can see our standard assumption: data is independent given parameters.
  - $y^1 \not\perp\!\!\!\perp y^2$, but $y^1 \perp\!\!\!\perp y^2 \mid w$.
  - $x^1 \perp\!\!\!\perp x^2$, but $x^1 \not\perp\!\!\!\perp x^2 \mid y^1, y^2$.

- Discriminative model: here we don't try to model things about $p(x^i)$.

# IID Bernoulli-Beta Model

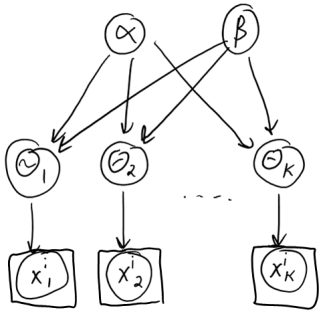
- The Bernoulli-beta model as a DAG (with parameters and hyper-parameters):



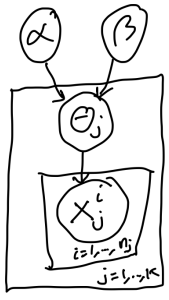

- Notice data is independent of hyper-parameters given parameters.
  - This is another of our standard independence assumptions.

# Non-IID Bernoulli-Beta Model

- The non-IID variant we considered with grouped data:



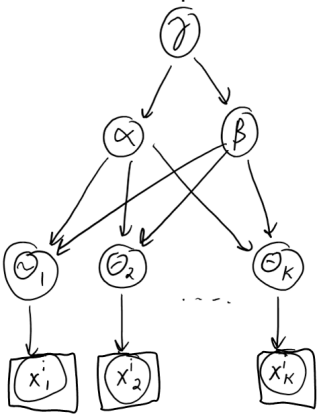

or




- DAG reflects that we do not tie parameters across all training examples.
- Notice that if you fix $\alpha$ and $\beta$ then you can't learn across groups:
  - The $\theta_j$ are d-separated given $\alpha$ and $\beta$.
- Can also write more succinctly with nested plates.

# Non-IID Bernoulli-Beta Model

- Variant of the previous model with a hyper-hyper-parameter:



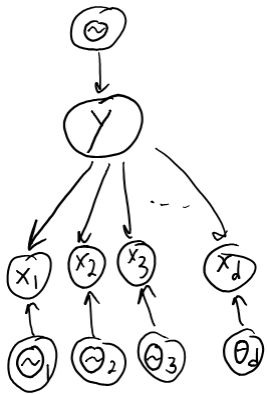

or


- Needed to avoid degeneracy.

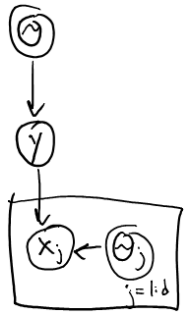
# Naive Bayes with DAGs/Plates

- For naive Bayes we have

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




or

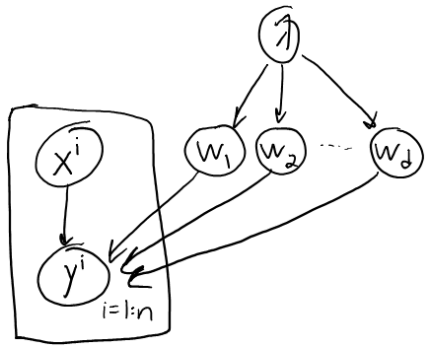


# Bayesian Linear Regression as a DAG

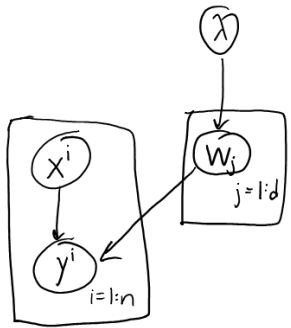
- In Bayesian linear regression we assume

$$y^i \sim \mathcal{N}(w^\mathsf{T} x^i, 1), \quad w_j \sim \mathcal{N}(0, 1/\lambda),$$

which we can write as




or



# Outline

# Density Estimators vs. Relationship Visualizers

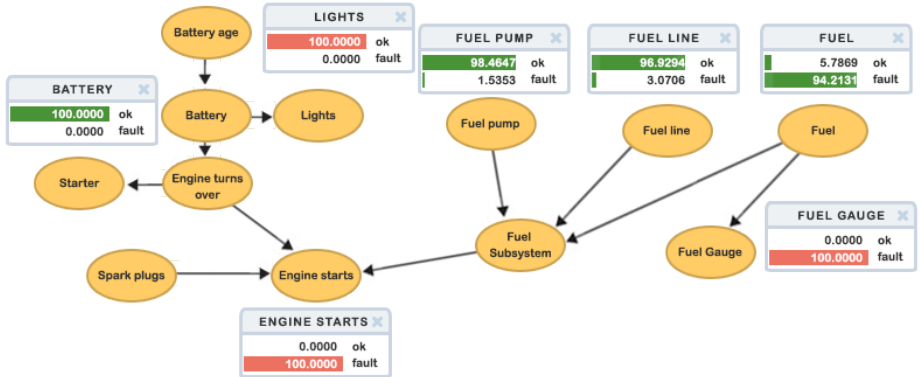
- Besides dependency visualization, we can use DAGs as density estimators.
- Recall that DAGs model joint distribution using

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

- We need to choose a parameterization for these conditional probabilities:
  - Tabular parameterization (discrete $x_j$): can model any joint probability.
    - Common choice; sometimes set parameters from expert knowledge.
  - Gaussian (continuous $x_j$): $x_j \sim \mathcal{N}(w^\top x_{\text{pa}(j)}, \sigma^2)$.
    - Called a Gaussian belief net. Joint distribution becomes a multivariate Gaussian.
  - Sigmoid (binary $x_j \in \{-1, +1\}$): $p(x_j \mid x_{j-1}, w) = 1/(1 + \exp(-x_j w^\top x_{\text{pa}(j)}))$.
    - Called a sigmoid belief net.
  - Could use softmax, probabilistic random forest, neural network, and so on.
    - Our tricks for probabilistic supervised learning can be used for unsuperivsed learning.

# Tabular Parameterization Example

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




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

# DAG Learning and Sampling

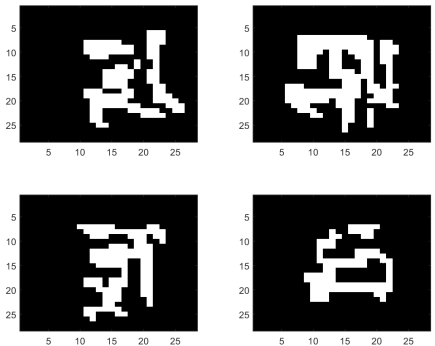
- For $j = 1 : d$:
  1. Set $\bar{y}^i = x_j^i$ and $\bar{x}^i = x^i_{\mathrm{pa}(j)}$.
  2. Solve a supervised learning problem using $\{\bar{X}, \bar{y}\}$.
     - Gives you a model of $p(x_j \mid x_{\mathrm{pa}(j)})$.

- Can sample from DAGs using ancestral sampling:
  - Sample $x_1$ from $p(x_1)$.
  - Sample $x_2$ from $p(x_2 \mid x_{\mathrm{pa}(2)})$.
  - $\vdots$
  - Sample $x_d$ from $p(x_d \mid x_{\mathrm{pa}(d)})$.

- This allows us to do inference with Monte Carlo methods.
  - Conditional sampling can be hard; might need rejection sampling for conditionals.

# MNIST Digits with Tabular DAG Model

- Recall our latest MNIST model using a tabular DAG:

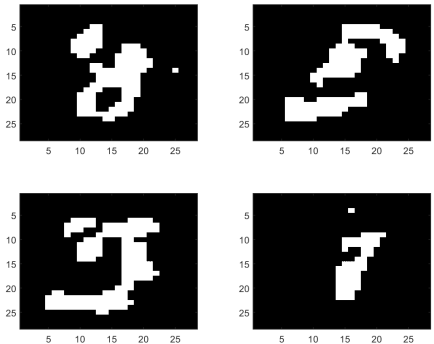



- This model is pretty bad because you only see 8 parents.

# 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).

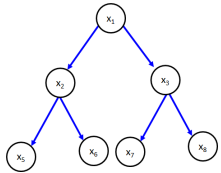
- Models long-range dependencies but has a linear assumption.

# Exact Inference in DAGs?

- Can we do exact inference in DAGs like in Markov chains?
- Continuous-state Gaussian DAGs:
  - Special case of multvariate Gaussian, so inference is tractable.
    - Most operations are $O(d)$ or $O(d^3)$.
- Continuous-state non-Gaussian DAGs:
  - Inference usually isn't closed-form; need Monte Carlo or variational inference.
  - If parents are conjugate, then Gibbs sampling is easy to implement.
- Discrete-state DAGs (whether tabular or sigmoid or other):
  - Inference takes exponential-time in the "treewidth" of the graph.
  - Exact inference is cheap in trees and forests, which have a treewidth of 1.
    - Low-treewidth graphs allow efficient exact inference; otherwise need approximations.

# Inference in Forest DAGs ("Belief Propagation")

- Connected graphs with at most one parent per node are called trees.




  - If not connected, these kinds of graphs are forests; both are "singly-connected."
- We can generalize the CK equations to trees/forests:

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

- Trees/forests allow efficient dynamic programming methods as in Markov chains.
  - In particular, decoding and univariate marginals/conditionals in $O(dk^2)$.
  - Forward-backward applied to tree-structured graphs is called belief propagation.
  - It's also possible to find the optimal tree given data ("structure learning").

# Outline

# Undirected Graphical Models (UGMs)

- Undirected graphical models (UGMs) are another popular graphical model class.
  - Also called Markov random fields.

- UGMs define joint distribution in terms of non-negative potential functions,

$$p(x_1, x_2, \ldots, x_d) \propto \prod_{c \in \mathcal{C}} \phi_c(x_c).$$

- Define a potential $\phi_c$ for each set $c$ where we want to model a direct relationship.

- The most common choice is a pairwise UGM,

$$p(x_1, x_2, \ldots, x_d) \propto \left( \prod_{j=1}^{d} \phi_j(x_j) \right) \left( \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \right).$$

This only has potentials on single variables ($\phi$) and pairs of variables ($\psi$).

- The "edge potentials" $\psi$ are defined on edges of an undirected graph $\mathcal{E}$.

# Pairwise Undirected Graphical Models

- Pairwise undirected graphical models factorize probability using

$$p(x_1, x_2, \ldots, x_d) \propto \left( \prod_{j=1}^{d} \phi_j(x_j) \right) \left( \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \right).$$

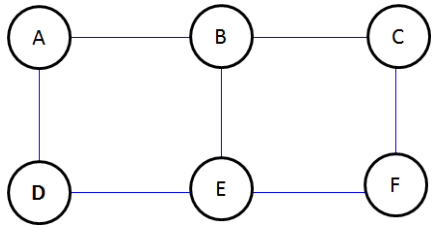
- Special cases:
  - Markov chains: $\mathcal{E}$ only has edges between adjacent nodes.
  - Multivaiate Gaussian: a specific choice of the $\phi$ and $\psi$ functions.
    - Gaussians AKA "Gaussian graphical models" or "Gaussian Markov random fields".
  - Ising model for binary $x_j$ uses

$$\phi_j(x_j) = \exp(x_i w_i), \quad \phi_{ij}(x_i, x_j) = \exp(x_i x_j w_{ij}),$$

    where $w_i$ is the node weight and $w_{ij}$ is the edge weight.
    - If $w_{ij} > 0$ it encourages neighbours to have same value ("attractive").
    - If $w_{ij} < 0$ it encourages neighbours to have different values ("repulsive").

# Conditional Independence in UGMs

- A UGM's independence properties are described by an undirected graph.
  - For pairwise UGMs, the edges are given by the set of edges $\mathcal{E}$.




- If you have 3-variable or higher-order potentials:
  - Add an edge $(i, j)$ if $i$ and $j$ are together in at least one $c$.
- So these two factorizations have the same graph:

$$p(x_1.x_2, x_3) \propto \phi_{12}(x_1, x_2)\phi_{13}(x_1, x_3)\phi_{23}(x_2, x_3), \quad p(x_1, x_2, x_3) \propto \phi_{123}(x_1, x_3, x_3).$$

- UGM implies $A \perp\!\!\!\perp B \mid C$ if $C$ separates all nodes in $A$ from all nodes in $B$.
  - General version of what we did with the graph from Gaussians' precision matrix.

# Multivariate Gaussians as UGMs

bonus!

- Writing a Gaussian as a pairwise UGM:

$$
\begin{aligned}
p(x_1, \ldots, x_d) &\propto \exp\left(-\frac{1}{2}(x-\mu)^\mathsf{T}\Sigma^{-1}(x-\mu)\right) \\
&= \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}(x_i-\mu_i)(\Sigma^{-1})_{ij}(x_j-\mu_j)\right) \\
&= \left(\prod_{j=1}^{d} e^{-\frac{1}{2}(\Sigma^{-1})_{jj}(x_j-\mu_j)^2}\right)\left(\prod_{(i,j):(\Sigma^{-1})_{ij}\neq 0} e^{-\frac{1}{2}(\Sigma^{-1})_{ij}(x_i-\mu_i)(x_j-\mu_j)}\right)
\end{aligned}
$$

- Hence why zeros of the precision $\Sigma^{-1}$ that determine conditional independence.

# DAGs vs. UGMs

- Neither DAGs or UGMs are "more powerful" than the other.
  - Any distribution can be written as a DAG, and as a UGM.
  - But you might need to use a highly connected graph.

- Set of independences in DAG cannot always be written as UGM (and vice versa).
  - UGMs cannot reflect independences in common child graph: $(x) \rightarrow (y) \leftarrow (z)$.
  - DAGs cannot reflect independences in 4-node loop: $(x) - (y) - (z) - (x)$.
  - Independences representable as both DAGs and UGMs are called decomposable.
    - An example is Markov chains: independences are same in DAG and UGM graphs.

- DAGs are often used when it makes sense to work with conditionals, or we have an idea of causal directions.

- UGMs are often used when there are no obvious directions (like MNIST), and are more often used when we want to add features to do supervised learning.

# Tractability of UGMs

- Without using $\propto$, a UGM probability would be

$$p(x) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(x_c),$$

where $Z$ is the constant that makes the probabilites sum up to 1.

$$Z = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \prod_{c \in \mathcal{C}} \phi_c(x_c) \quad \text{or} \quad Z = \int_{x_1} \int_{x_2} \cdots \int_{x_d} \prod_{c \in \mathcal{C}} \phi_c(x_c) \mathrm{d}x_d \mathrm{d}x_{d-1} \cdots \mathrm{d}x_1.$$

- Whether you can compute $Z$ (and do inference) depends on the choice of the $\phi_c$:
  - Gaussian case: $O(d^3)$ in general, but $O(d)$ for forests (no loops).
  - Continuous non-Gaussian: usually requires approximate inference.
  - Discrete case: #P-hard in general, but $O(dk^2)$ for forests (no loops).

# Discrete DAGs vs. Discrete UGMs

- Common inference tasks in graphical models:
  1. Compute $p(x)$ for an assignment to the variables $x$.
  2. Generate a sample $x$ from the distribution.
  3. Compute univariate marginals $p(x_j)$.
  4. Compute decoding $\arg\max_x p(x)$.
  5. Compute univariate conditional $p(x_j \mid x_{j'})$.

- With discrete $x_i$, all of the above are easy in tree-structured graphs.
  - For DAGs, a tree-structured graph has at most one parent.
  - For UGMs, a tree-structured graph has no cycles.

- With discrete $x_i$, the above may be harder for general graphs:
  - In DAGs the first two are easy, the others are NP-hard.
  - In UGMs all of these are NP-hard.

# Inference in UGMs

- The course does not "officially cover" details on inference in graphical models.
- For however long is left today, we'll cover some stuff as bonus slides.
- These include:
  - Inference in non-tree DAGs/UGMs.
  - Learning the graph structure.
  - Treewidth of graphs, and efficient inference with low treewidth.
  - Exact decoding for binary attractive models using graph cuts.
  - ICM and alpha-expansion algorithms for approximate decoding.
  - Block Gibbs sampling in UGMs (UGMs are what Gibbs sampling was invented for).

# Summary

- Independence assumptions about data and parameters can be written as DAGs.
- D-seperation lets us read conditional independences from DAGs.
- 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.
- Sampling in DAGs is easy (ancestral sampling).
- Exact inference in discrete DAGs is easy for trees.
  - But becomes exponential in "treewidth" of graph.
- Undirected graphical models factorize probability into non-negative potentials.
  - Gaussians are a special case, but can place potentials on any subset of variables.
  - Inference is again exponential in "treewidth" of graph.

- Next time: adding graphical models to neural networks.

# Outline

# Inference in General DAGs

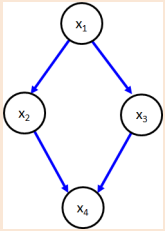
bonus!

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

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

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

- Consider the non-tree graph:



# Inference in General DAGs

bonus!

- 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

bonus!

- With 2-variable messages, our cost increases to $O(dk^3)$.
- If we add the edge $x_1 \to x_4$, then the cost is $O(dk^4)$.

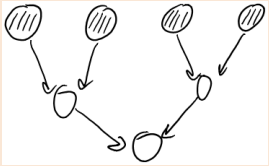
(the same cost as enumerating all possible assignments)

- 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.

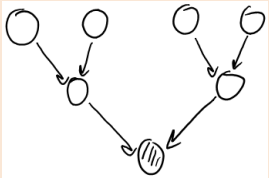
# Conditional Sampling in DAGs

bonus!

- 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.



# Outline

# DAG Structure Learning

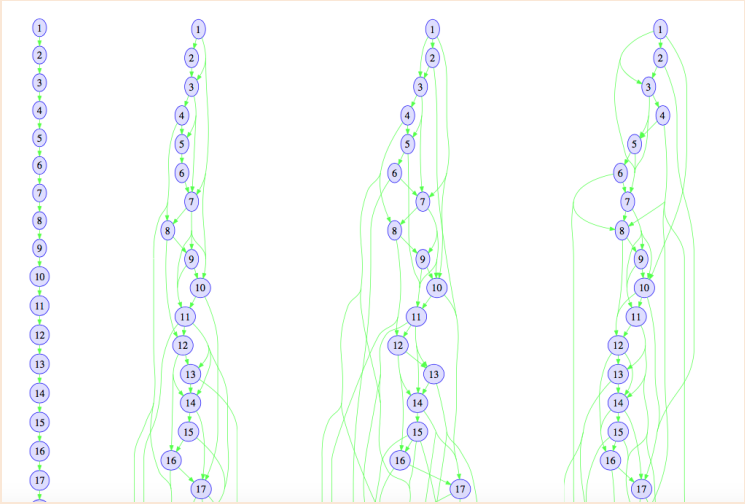
bonus!

- 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

bonus!

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



# DAG Structure Learning without an Ordering

bonus!

- 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.

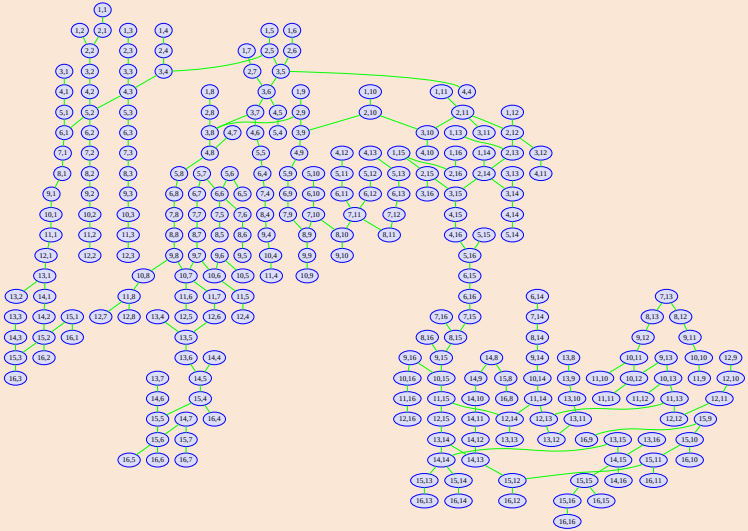
- May have equivalent graphs with the same score (don't trust edge direction).
  - Do not interpret causally a graph learned from data.

- 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").
    - Score is symmetric if score($x_j \to x_{j'}$) is the same as score($x_{j'} \to x_j$).
  - For asymetric scores, can be found by minimum spanning arborescence.

# Structure Learning on USPS Digits

bonus!

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



# 20 Newsgroups Data

bonus!

- Data containing presence of 100 words from newsgroups posts:

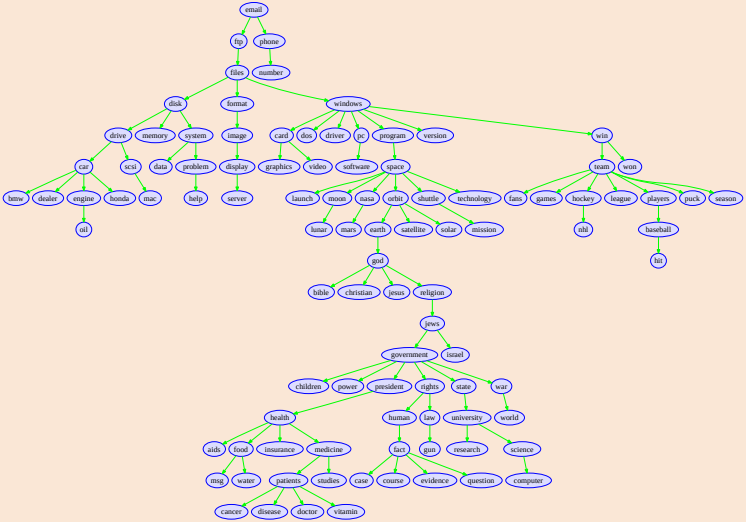
| car | drive | files | hockey | mac | league | pc | win |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

- Structure learning should give some relationship between word occurrences.

# Structure Learning on News Words

bonus!

Optimal tree on newsgroups data:



# "Constraint-Based" DAG Structure Learning

bonus!

- 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.)

# Outline

# Gaussians as Undirected Graphical Models

bonus!

- Multivariate Gaussian can be written as

$$p(x) \propto \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) \propto \exp\left(-\frac{1}{2}x^\top \Sigma^{-1}x + x^\top \underbrace{\Sigma^{-1}\mu}_{v}\right),$$

and writing it in summation notation we can see that it's a pairwise UGM:

$$p(x) \propto \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j (\Sigma^{-1})_{ij} + \sum_{i=1}^{d} x_i v_i\right)$$

$$= \left(\prod_{i=1}^{d}\prod_{j=1}^{d} \underbrace{\exp\left(-\frac{1}{2}x_i x_j (\Sigma^{-1})_{ij}\right)}_{\phi_{ij}(x_i,x_j)}\right)\left(\prod_{i=1}^{d} \underbrace{\exp\left(x_i v_i\right)}_{\phi_i(x_i)}\right)$$

- Above we include all edges. You can "remove" edges by setting $(\Sigma^{-1})_{ij} = 0$.
- "Gaussian graphical model" (GGM) or "Gaussian Markov random field" (GMRF).

# General Pairwise UGM

bonus!

- For general discrete $x_i$ a generalization of Ising models is

$$p(x_1, x_2, \ldots, x_d) = \frac{1}{Z} \exp\left(\sum_{i=1}^{d} w_{i,x_i} + \sum_{(i,j)\in E} w_{i,j,x_i,x_j}\right),$$

  which can represent any "positive" pairwise UGM (meaning $p(x) > 0$ for all $x$).

- Interpretation of weights for this UGM:
  - If $w_{i,1} > w_{i,2}$ then we prefer $x_i = 1$ to $x_i = 2$.
  - If $w_{i,j,1,1} > w_{i,j,2,2}$ then we prefer $(x_i = 1, x_j = 1)$ to $(x_i = 2, x_j = 2)$.

- As before, we can use parameter tieing:
  - We could use the same $w_{i,x_i}$ for all positions $i$.
  - Ising model corresponds to a particular parameter tieing of the $w_{i,j,x_i,x_j}$.

# Label Propagation (Graph-Based Semi-Supervised) as a UGM

bonus!

- Consider modeling the probability of a vector of labels $\bar{y} \in \mathbb{R}^t$ using

$$p(\bar{y}^1, \bar{y}^2, \ldots, \bar{y}^t) \propto \exp\left(-\sum_{i=1}^{n}\sum_{j=1}^{t} w_{ij}(y^i - \bar{y}^i)^2 - \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t} \bar{w}_{ij}(\bar{y}^i - \bar{y}^j)^2\right).$$

- Decoding in this model is the label propagation problem.

- This is a pairwise UGM:

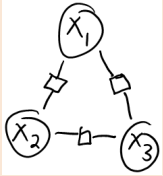
$$\phi_j(\bar{y}^j) = \exp\left(-\sum_{i=1}^{n} w_{ij}(y^i - \bar{y}^j)^2\right), \quad \phi_{ij}(\bar{y}^i, \bar{y}^j) = \exp\left(-\frac{1}{2}\bar{w}_{ij}(\bar{y}^i - \bar{y}^j)^2\right).$$

# Factor Graphs

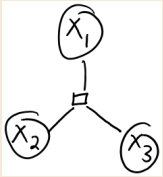
bonus!

- Factor graphs are a way to visualize UGMs that distinguishes different orders.
  - Use circles for variables, squares to represent dependencies.

- Factor graph of $p(x_1, x_2, x_3) \propto \phi_{12}(x_1, x_2)\phi_{13}(x_1, x_3)\phi_{23}(x_2, x_3)$:




- Factor graph of $p(x_1, x_2, x_3) \propto \phi_{123}(x_1, x_2, x_3)$:



# Other Graphical Models

bonus!

- Factor graphs: we use a square between variables that appear in same factor.
  - Can distinguish between a 3-way factor and 3 pairwise factors.
- Chain-graphs: DAGs where each block can be a UGM.
- Ancestral-graph:
  - Generalization of DAGs that is closed under conditioning.
- Structural equation models (SEMs): generalization of DAGs that allows cycles.

# Outline

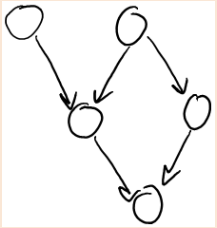
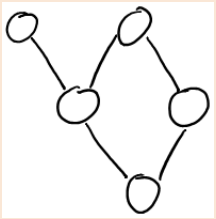
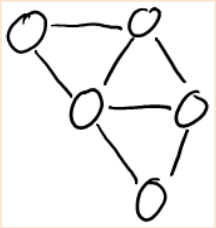
# Moralization: Converting DAGs to UGMs

bonus!

- To address the NP-hard problems, DAGs and UGMs use same techniques.
- We'll focus on UGMs, but we can convert DAGs to UGMs:

$$p(x_1, x_2, \dots, x_d) = \prod_{j=1}^{d} p(x_j | x_{\text{pa}(j)}) = \prod_{j=1}^{d} \underbrace{\phi_j(x_j, x_{\text{pa}(j)})}_{=p(x_j|x_{\text{pa}(j)})},$$

  which is a UGM with $Z = 1$.
- Graphically: we drop directions and "marry" parents (moralization).

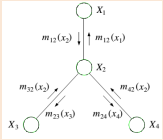
- May no longer see some independences, but doesn't change computational cost.

# Easy Cases: Chains, Trees and Forests

bonus!

- The forward-backward algorithm still works for chain-structured UGMs:
  - We compute the forward messages $M$ and the backwards messages $V$.
  - With both $M$ and $V$ we can [conditionally] decode/marginalize/sample.

- Belief propagation generalizes this to trees (undirected graphs with no cycles):
  - Pick an arbitrary node as the "root", and order the nodes going away from the root.
    - Pass messages starting from the "leaves" going towards the root.
  - "Root" is like the last node in a Markov chain.
    - Backtrack from root to leaves to do decoding/sampling.
    - Send messages from the root going to the leaves to compute all marginals.




https://www.quora.com/

Probabilistic-graphical-models-what-are-the-relationships-between-sum-product-algorithm-belief-propagation-and-junction-tree-

# Easy Cases: Chains, Trees and Forests

bonus!

- Recall the CK equations in Markov chains:

$$M_c(x_c) = \sum_{x_p} p(x_c \mid x_p) M_p(x_p).$$

- For chain-structure UGMs we would have:

$$M_c(x_c) \propto \sum_{x_p} \phi(x_p)\phi(x_p, x_c) M_p(x_p).$$

- In tree-structured UGMs, parent $p$ in the ordering may have multiple parents.
- Message coming from "neighbour" $i$ that itself has neighbours $j$ and $k$ would be

$$M_{ic}(x_c) \propto \sum_{x_i} \phi_i(x_i)\phi_{ic}(x_i, x_c) M_{ji}(x_i) M_{ki}(x_i),$$

- Univariate marginals are proportional to $\phi_i(x_i)$ times all "incoming" messages.
  - The "forward" and "backward" Markov chain messages are a special case.
  - Replace $\sum_{x_i}$ with $\max_{x_i}$ for decoding.
    - "Sum-product" and "max-product" algorithms.

# Exact Inference in UGMs

bonus!

- For general graphs, the cost of message passing depends on
  1. Graph structure.
  2. Variable order.
- To see the effect of the order, consider Markov chain inference with bad ordering:

$$
\begin{aligned}
p(x_5) &= \sum_{x_5}\sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)p(x_5 \mid x_4) \\
&= \sum_{x_5}\sum_{x_1}\sum_{x_4}\sum_{x_3}\sum_{x_2} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)p(x_5 \mid x_4) \\
&= \sum_{x_5}\sum_{x_1} p(x_1) \sum_{x_3}\sum_{x_4} p(x_4 \mid x_3)p(x_5 \mid x_4) \underbrace{\sum_{x_2} p(x_2 \mid x_1)p(x_3 \mid x_2)}_{M_{13}(x_1,x_3)}
\end{aligned}
$$

- So even though we have a chain, we have an $M$ with $k^2$ values instead of $k$.
  - Increases cost to $O(dk^3)$ instead of $O(dk^2)$.
  - Inference can be exponentially more expensive with the wrong ordering.

# Exact Inference in UGMs

bonus!

- For general graphs, the cost of message passing depends on
  1. Graph structure.
  2. Variable order.
- As a non-tree example, consider computing $Z$ in a simple 4-node cycle:

$$
\begin{aligned}
Z &= \sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1} \phi_{12}(x_1,x_2)\phi_{23}(x_2,x_3)\phi_{34}(x_3,x_4)\phi_{14}(x_1,x_4) \\
&= \sum_{x_4}\sum_{x_3} \phi_{34}(x_3,x_4) \sum_{x_2} \phi_{23}(x_2,x_3) \sum_{x_1} \phi_{12}(x_1,x_2)\phi_{14}(x_1,x_4) \\
&= \sum_{x_4}\sum_{x_3} \phi_{34}(x_3,x_4) \sum_{x_2} \phi_{23}(x_2,x_3) M_{24}(x_2,x_4) \\
&= \sum_{x_4}\sum_{x_3} \phi_{34}(x_3,x_4) M_{34}(x_3,x_4) = \sum_{x_4} M_4(x_4).
\end{aligned}
$$

- We again have an $M$ with $k^2$ values instead of $k$.
  - We can do inference tasks with this graph, but it costs $O(dk^3)$ instead of $O(dk^2)$.

# Variable Order and Treewidth

bonus!

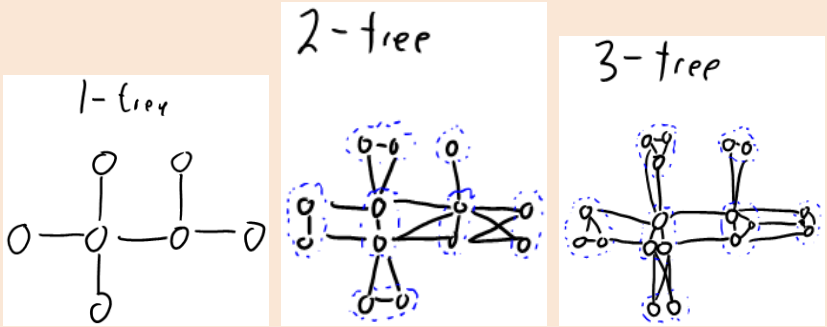
- Cost of message passing in general graphs is given by $O(dk^{\omega+1})$.
  - Here, $\omega$ is the number of dimensions of the largest message.
  - For trees, $\omega = 1$ so we get our usual cost of $O(dk^2)$.

- The minimum value of $\omega$ across orderings for a given graph is called treewidth.
  - In terms of graph: "minimum size of largest clique, minus 1, over all triangulations".
    - Also called "graph dimension" or "$\omega$-tree".
  - Intuitively, you can think of low treewidth as being "close to a tree".
    - Trees have a treewidth of 1, and a single loop has a treewidth of 2.

# Treewidth Examples

bonus!

- Examples of k-trees:

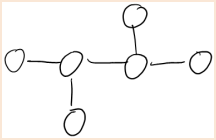



- 2-tree and 3-tree are trees if you use dotted circles to group nodes.

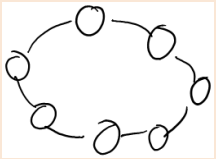
# Treewidth Examples

bonus!

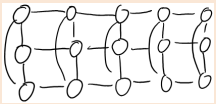
- Trees have $\omega = 1$, so with the right order inference costs $O(dk^2)$.



- A big loop has $\omega = 2$, so cost with the right ordering is $O(dk^3)$.



- The below grid-like structure has $\omega = 3$, so cost is $O(dk^4)$.



# Variable Order and Treewidth

bonus!

- Junction trees generalize belief propagation to general graphs (requires ordering).
  - This is the algorithm that achieves the $O(dk^{\omega+1})$ runtime.
- Computing $\omega$ and the optimal ordering is NP-hard.
  - But various heuristic ordering methods exist.
- An $m_1$ by $m_2$ lattice has $\omega = \min\{m_1, m_2\}$.
  - So you can do exact inference on "wide chains" with Junction tree.
  - But for 28 by 28 MNIST digits it would cost $O(784 \cdot 2^{29})$.
- Some links if you want to read about treewidth:
  - https://www.win.tue.nl/~nikhil/courses/2015/2WO08/treewidth-erickson.pdf
  - https://math.mit.edu/~apost/courses/18.204-2016/18.204_Gerrod_Voigt_final_paper.pdf
- For some graphs $\omega = (d-1)$ so there is no gain over brute-force enumeration.
  - Many graphs have high treewidth so we need approximate inference.

# Outline

# Iterated Conditional Mode (ICM)

bonus!

- The iterated conditional mode (ICM) algorithm for approximate decoding:
  - On each iteration $k$, choose a variable $j_t$.
  - Maximie the joint probability in terms of $x_{j_t}$ (with other variables fixed),

$$x_j^{t+1} \in \arg\max_c p(x_1^t, \ldots, x_{j-1}^t, x_j = c, x_{j+1}^t, \ldots, x_d^t).$$

- Equivalently, iterations correspond to finding mode of conditional $p(x_j \mid x_{-j}^t)$,

$$x_j^{t+1} \in \arg\max_c p(x_j = c \mid x_{-j}^t),$$

where $x_{-j}$ means "$x_i$ for all $i$ except $x_j$": $x_1, x_2, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d$.

# ICM in Action

bonus!

- Start with some initial value: $x^0 = \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix}$.
- Select random $j$ like $j = 3$.
- Set $j$ to maximize $p(x_3 \mid x^0_{-3})$: $x^1 = \begin{bmatrix} 2 & 2 & 1 & 1 \end{bmatrix}$.
- Select random $j$ like $j = 1$.
- Set $j$ to maximize $p(x_1 \mid x^1_{-1})$: $x^2 = \begin{bmatrix} 3 & 2 & 1 & 1 \end{bmatrix}$.
- Select random $j$ like $j = 2$.
- Set $j$ to maximize $p(x_2 \mid x^2_{-2})$: $x^3 = \begin{bmatrix} 3 & 2 & 1 & 1 \end{bmatrix}$.
- ...
- Repeat until you can no longer improve by single-variable changes.
  - Intead of random, could cycle through the variables in order.
  - Or you could greedily choose the variable that increases the probability the most.

# Optimality and Globalization of ICM

bonus!

- Does ICM find the global optimum?
- Decoding is usually non-convex, so doesn't find global optimum.
  - ICM is an approximate decoding method.
- There exist many globalization methods that can improve its performance:
  - Restarting with random initializations.
  - Global optimization methods:
    - Simulated annealing, genetic algorithms, ant colony optimization, GRASP, etc.

# Using the Unnormalized Objective

bonus!

- How can you maximize $p(x)$ in terms of $x_j$ if evaluating it is NP-hard?
- Let's define the unnormalized probability $\tilde{p}$ as

$$\tilde{p}(x) = \prod_{c \in \mathcal{C}} \phi_c(x_c).$$

- So the normalized probability is given by

$$p(x) = \frac{\tilde{p}(x)}{Z}.$$

- In UGMs evaluating $Z$ is hard but evaluating $\tilde{p}(x)$ is easy.
- And for decoding we only need unnormalized probabilities,

$$\arg\max_x p(x) \equiv \arg\max_x \frac{\tilde{p}(x)}{Z} \equiv \arg\max_x \tilde{p}(x),$$

so we can decode based on $\tilde{p}$ without knowing $Z$.

# ICM Iteration Cost

bonus!

- How much does ICM cost?

- Consider a pairwise UGM,

$$\tilde{p}(x) = \left(\prod_{j=1}^{d} \phi_j(x_j)\right)\left(\prod_{(i,j)\in E} \phi_{ij}(x_i, x_j)\right).$$

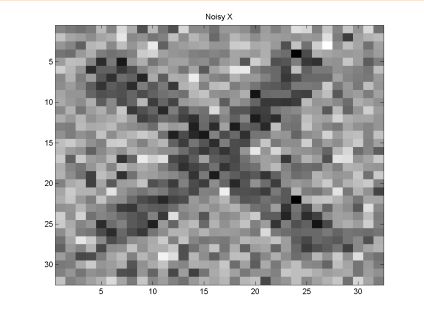
- Each ICM update would:
  1. Set $M_j(x_j = s)$ to product of terms in $\tilde{p}(x)$ involving $x_j$, with $x_j$ set to $s$.
  2. Set $x_j$ to the largest value of $M_j(x_j)$.

- The variable $x_j$ has $k$ values and appears in at most $d$ factors here.
  - You can compute the $k$ values of these $d$ factors in $O(dk)$ to find the largest.
  - If you only have $m$ nodes in "Markov blanket", this reduces to $O(mk)$.
    - We will define "Markov blanket" in a couple slides.

# ICM in Action

bonus!

Consider using a UGM for binary image denoising:




We have

- Unary potentials $\phi_j$ for each position.
- Pairwise potentials $\phi_{ij}$ for neighbours on grid.
- Parameters are trained as CRF (later).

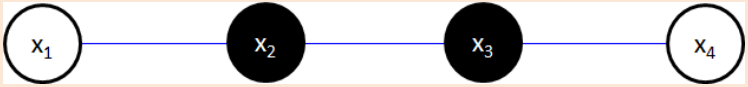
Goal is to produce a noise-free binary image (show video).

# Digression: Closure of UGMs under Conditioning

bonus!

- UGMs are closed under conditioning:
  - If $p(x)$ is a UGM, then $p(x_A \mid x_B)$ can be written as a UGM (for partition $A$ and $B$).

- Conditioning on $x_2$ and $x_3$ in a chain,




  gives a UGM defined on $x_1$ and $x_4$ that is disconnected:


- Graphically, we "erase the black nodes and their edges".

- Notice that inference in the conditional UGM may be mucher easier.

# Digression: Closure of UGMs under Conditioning

bonus!

- Mathematically, a 4-node pairwise UGM with a chain structure assumes

$$p(x_1, x_2, x_3, x_4) \propto \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4).$$

- Conditioning on $x_2$ and $x_3$ gives UGM over $x_1$ and $x_4$.

$$p(x_1, x_4 \mid x_2, x_3) = \frac{1}{Z'}\phi'_1(x_1)\phi'_4(x_4),$$

where new potentials "absorb" the shared potentials with observed nodes:

$$\phi'_1(x_1) = \phi_1(x_1)\phi_{12}(x_1, x_2), \quad \phi'_4(x_4) = \phi_4(x_4)\phi_{34}(x_3, x_4).$$

# Conditioning in UGMs

bonus!

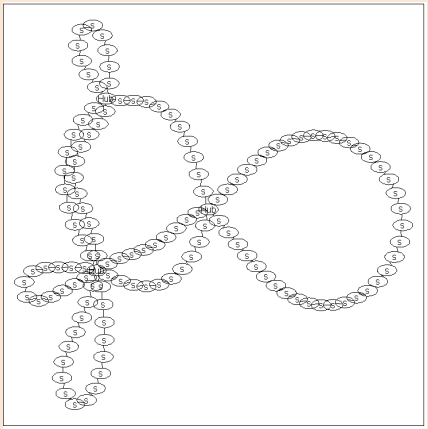
- Conditioning on $x_2$ and $x_3$ in 4-node chain-UGM gives

$$
\begin{aligned}
p(x_1, x_4 | x_2, x_3) &= \frac{p(x_1, x_2, x_3, x_4)}{p(x_2, x_3)} \\
&= \frac{\frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_1(x_1, x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \frac{1}{Z}\phi_1(x_1')\phi_2(x_2)\phi_3(x_3)\phi_4(x_4')\phi_1(x_1', x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4')} \\
&= \frac{\frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_1(x_1, x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4)}{\frac{1}{Z}\phi_2(x_2)\phi_3(x_3)\phi_2(x_2, x_3)\sum_{x_1', x_4'} \phi_1(x_1')\phi_4(x_4')\phi_1(x_1', x_2)\phi_3(x_3, x_4')} \\
&= \frac{\phi_1(x_1)\phi_4(x_4)\phi_1(x_1, x_2)\phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \phi_1(x_1')\phi_4(x_4')\phi_1(x_1', x_2)\phi_3(x_3, x_4')} \\
&= \frac{\phi_1'(x_1)\phi_4'(x_4)}{\sum_{x_1', x_4'} \phi_1'(x_1')\phi_4'(x_4')}
\end{aligned}
$$

# Simpler Inference in Conditional UGMs

bonus!

- Consider the following graph which could describe bus stops:



- If we condition on the "hubs", the graph forms a forest (and inference is easy).
  - Simpler inference after conditioning is used by many approximate inference methods.

# Digression: Local Markov Property and Markov Blanket

bonus!

- Approximate inference methods often use conditional $p(x_j \mid x_{-j})$,
  - where $x_{-j}^k$ means "$x_i^k$ for all $i$ except $x_j^k$": $x_1^k, x_2^k, \ldots, x_{j-1}^k, x_{j+1}^k, \ldots, x_d^k$.

- In UGMs, the conditional simplifies due to conditional independence,

$$p(x_j \mid x_{-j}) = p(x_j \mid x_{\text{nei}(j)}),$$

  this local Markov property means conditional only depends on neighbours.

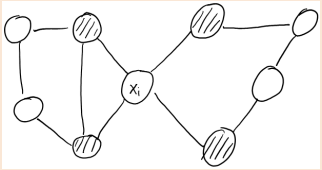
- We say that the neighbours of $x_j$ are its "Markov blanket".

- Markov blanket is the set nodes that make you independent of all other nodes.

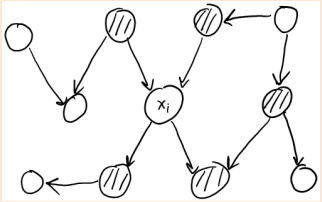
# Digression: Local Markov Property and Markov Blanket

bonus!

- In UGMs the Markov blanket is the neighbours.




- Markov blanket in DAGs: parents, children, co-parents (parents of same children):



# Outline

# Block-Structured Approximate Inference

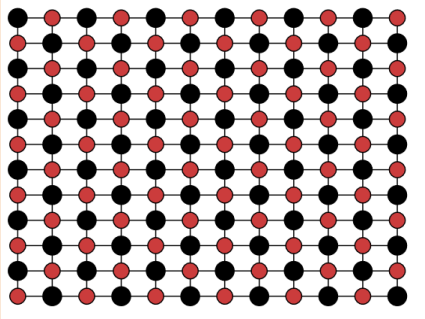
bonus!

- Basic approximate inference methods like ICM and Gibb sampling:
  - Update one $x_j$ at a time.
  - Efficient because conditional UGM is 1 node.
- Better approximate inference methods use block updates:
  - Update a block of $x_j$ values at once.
  - Efficient if conditional UGM allows exact inference.
- If we choose the blocks cleverly, this works substantially better.

# Block-Structured Approximate Inference

bonus!

- Consider a lattice-structure and the following two blocks ("red-black ordering"):



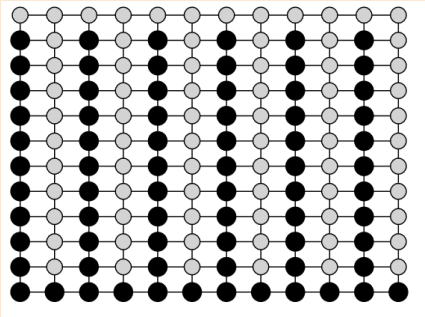
- Given black nodes, conditional UGM on red nodes is a disconnected graph.
  - "I can optimally update the red nodes given the black nodes" (and vice versa).
    - You update $d/2$ nodes at once for cost of this is $O(dk)$, and easy to parallelize.

- Minimum number of blocks to disconnect the graph is graph colouring.

# Block-Structured Approximate Inference

bonus!

- We could also consider general forest-structured blocks:

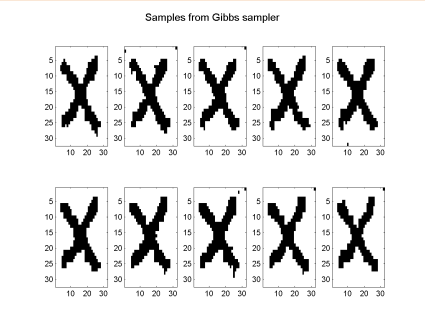


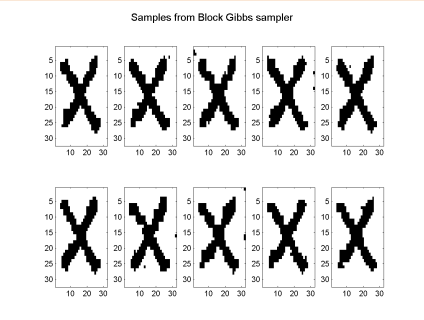
- We can still optimally update the black nodes given the gray nodes in $O(dk^2)$.
  - This works much better than "one at a time".

# Block Gibbs Sampling in Action

bonus!

- Gibbs vs. tree-structured block-Gibbs samples:



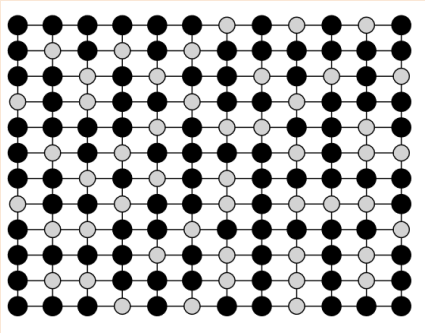




- With block sampling, the samples are far less correlated.
- We can also do tree-structured block ICM.
  - Harder to get stuck if you get to update entire trees.

# Block-Structured Approximate Inference

bonus!

- Or we could define a new tree-structured block on each iteration:



- The above block updates around two thirds of the nodes optimally.

(Here we're updating the black nodes.)

# Block ICM Based on Graph Cuts

bonus!

- Consider a binary pairwise UGM with "attractive" potentials,

$$\log \phi_{ij}(1,1) + \log \phi_{ij}(2,2) \geq \log \phi_{ij}(1,2) + \log \phi_{ij}(2,1).$$

- In words: "neighbours prefer to have similar states".

- In this setting exact decoding can be formulated as a max-flow/min-cut problem.
  - Can be solved in polynomial time.

- This is widely-used computer vision:
  - Want neighbouring pixels/super-pixels/regions to be more likely to get same label.

# Graph Cut Example: "GrabCut"

bonus!

Figure 1: **Three examples of GrabCut.** The user drags a rectangle loosely around an object. The object is then extracted automatically.

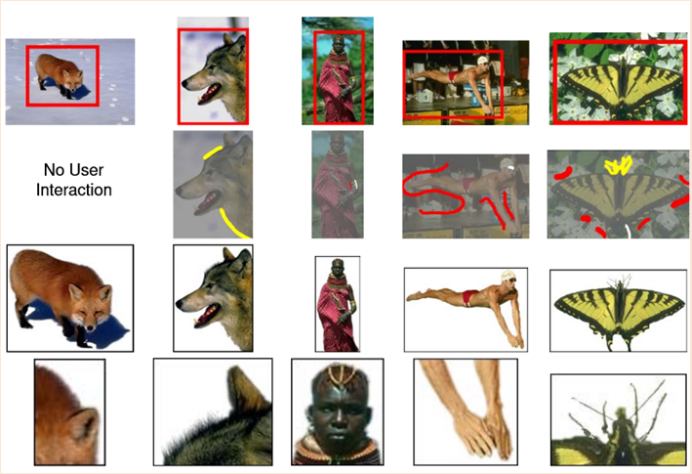
http://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

1. User draws a box around the object they want to segment.
2. Fit Gaussian mixture model to pixels inside the box, and to pixels outside the box.
3. Construct a pairwise UGM using:
   - $\phi_i(x_i)$ set to GMM probability of pixel $i$ being in class $x_i$.
   - $\phi_{ij}(x_i, x_j)$ set to Ising potential times RBF based on spatial/colour distance.
     - Use $w_{ij} > 0$ so the model is "attractive".
4. Perform exact decoding in the binary attractive model using graph cuts.

# Graph Cut Example: "GrabCut"

bonus!

- GrabCut with extra user interaction:




http://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

# Alpha-Beta Swap and Alpha-Expansions: ICM with Graph Cuts

bonus!

- If we have more than 2 states, we can't use graph cuts.
- Alpha-beta swaps are an approximate decoding method for "pairwise attractive",

$$\log \phi_{ij}(\alpha, \alpha) + \log \phi_{ij}(\beta, \beta) \geq \log \phi_{ij}(\alpha, \beta) + \log \phi_{ij}(\beta, \alpha).$$

  - Each step choose an $\alpha$ and $\beta$, optimally "swaps" labels among these nodes.
- Alpha-expansions are another variation based on a slightly stronger assumption,

$$\log \phi_{ij}(\alpha, \alpha) + \log \phi_{ij}(\beta_1, \beta_2) \geq \log \phi_{ij}(\alpha, \beta_1) + \log \phi_{ij}(\beta_2, \alpha).$$

  - Steps choose label $\alpha$, and consider replacing the label of any node not labeled $\alpha$.

# Alpha-Beta Swap and Alpha-Expansions: ICM with Graph Cuts

bonus!

- These don't find global optima in general, but make huge moves:

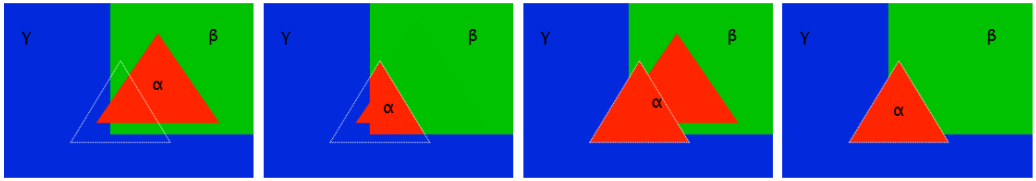



Figure 1: From left to right: Initial labeling, labeling after $\alpha\beta$-swap, labeling after $\alpha$-expansion, labeling after $\alpha$-expansion $\beta$-shrink. The optimal labeling of the $\alpha$ pixels is outlined by a white triangle, and is achieved from the initial labeling by one ~~$\alpha$-expansion $\beta$-shrink move~~. ex-swap move

- A somewhat-related MCMC method is the Swendson-Wang algorithm.

# Example: Photomontage

bonus!

- Photomontage: combining different photos into one photo:

http://vision.middlebury.edu/MRF/pdf/MRF-PAMI.pdf

- Here, $x_i$ corresponds to identity of original image at position $i$.

# Example: Photomontage

bonus!

- Photomontage: combining different photos into one photo:

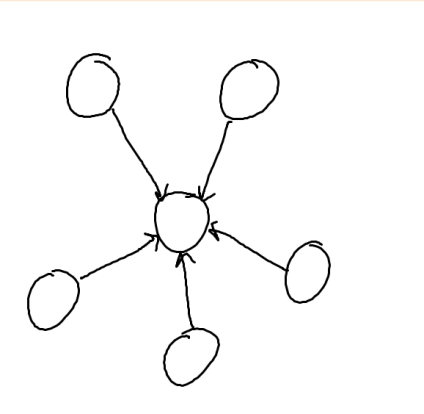
http://vision.middlebury.edu/MRF/pdf/MRF-PAMI.pdf

# Outline

# Conditional Independence in Star Graphs

bonus!

- Consider the following star graph:

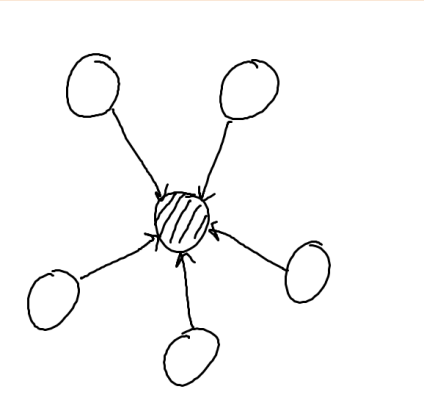


- "5 aliens get together and make a baby alien".
  - Unconditionally, the 5 aliens are independent.

# Conditional Independence in Star Graphs

bonus!

- Consider the following star graph:

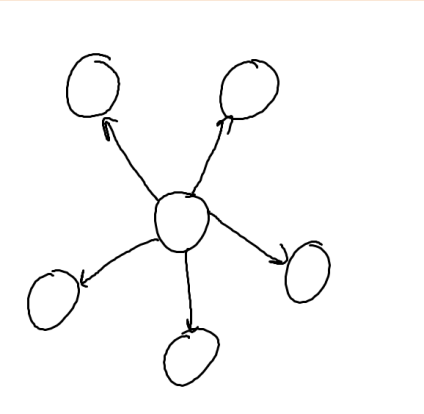


- "5 aliens get together and make a baby alien".
  - Conditioned on the baby, the 5 aliens are dependent.

# Conditional Independence in Star Graphs

bonus!

- Consider the following star graph:

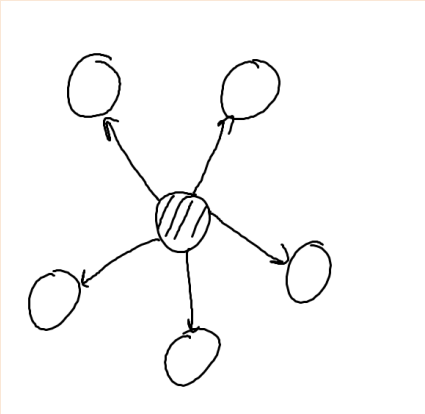


- "An organism produces 5 clones".
  - Unconditionally, the 5 clones are dependent.

# Conditional Independence in Star Graphs

bonus!

- Consider the following star graph:



- "An organism produces 5 clones".
  - Conditioned on the original, the 5 clones are independent.

# Does Semi-Supervised Learning Make Sense?

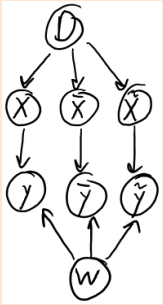
bonus!

- Should unlabeled examples always help supervised learning?
  - No!
- Consider choosing unlabeled features $\bar{x}^i$ uniformly at random.
  - Unlabeled examples collected in this way will not help.
  - By construction, distribution of $\bar{x}^i$ says nothing about $\bar{y}^i$.
- Example where SSL is not possible:
  - Try to detect food allergy by trying random combinations of food:
    - The actual random process isn't important, as long as it isn't affected by labels.
    - You can sample an infinite number of $\bar{x}^i$ values, but they says nothing about labels.
- Example where SSL is possible:
  - Trying to classify images as "cat" vs. "dog.":
    - Unlabeled data would need to be images of cats or dogs (not random images).
    - Unlabeled data contains information about what images of cats and dogs look like.
    - For example, there could be clusters or manifolds in the unlabeled images.

# Does Semi-Supervised Learning Make Sense?

bonus!

- Let's assume our semi-supervised learning model is represented by this DAG:

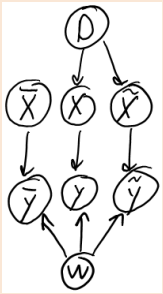



- Assume we observe $\{X, y, \bar{X}\}$ and are interested in test labels $\tilde{y}$:
  - There is a dependency between $y$ and $\tilde{y}$ because of path through $w$.
    - Parameter $w$ is tied between training and test distributions.
  - There is a dependency between $X$ and $\tilde{y}$ because of path through $w$ (given $y$).
    - But note that there is also a second path through $D$ and $\tilde{X}$.
  - There is a dependency between $\bar{X}$ and $\tilde{y}$ because of path through $D$ and $\tilde{X}$.
    - Unlabeled data helps because it tells us about data-generating distribution $D$.

# Does Semi-Supervised Learning Make Sense?

bonus!

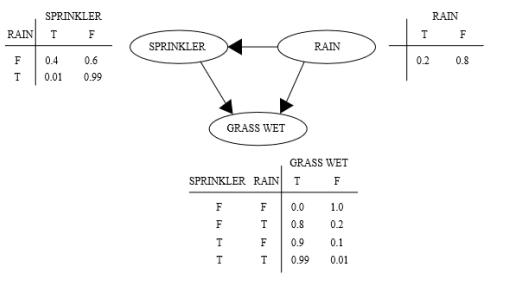
- Now consider generating $\bar{X}$ independent of $D$:




- Assume we observe $\{X, y, \bar{X}\}$ and are interested in test labels $\tilde{y}$:
  - Knowing $X$ and $y$ are useful for the same reasons as before.
  - But knowing $\bar{X}$ is not useful:
    - Without knowing $\bar{y}$, $\bar{X}$ is $d$-separated from $\tilde{y}$ (no dependence).

# Tabular Parameterization Example

bonus!




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

Some quantities can be directly read from the tables:

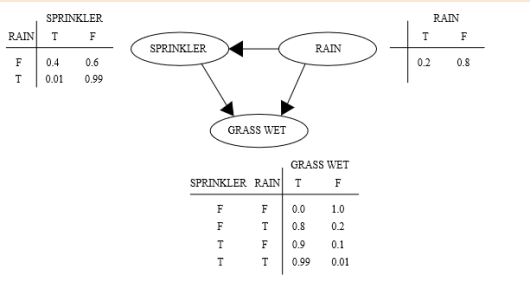
$$p(R = 1) = 0.2.$$

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

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

# Tabular Parameterization Example

bonus!




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

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

$$p(G=1 \mid R=1) = p(G=1, S=0 \mid R=1) + p(G=1, S=1 \mid R=1) \quad \left( p(a \mid c) = \sum_b p(a, b \mid c) \right)$$

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

$$= 0.8(0.99) + 0.99(0.01) = 0.81.$$

# Dynamic Bayesian Networks

bonus!

- Dynamic Bayesian networks combine ideas from DAGs and Markov chains:
  - 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.

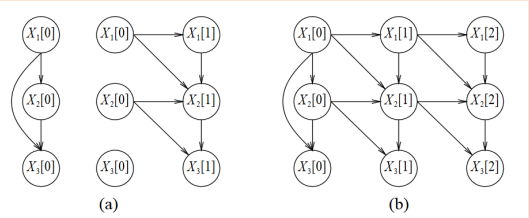



Figure 1: (a) A prior network and transition network defining a DPN for the attributes $X_1$, $X_2$, $X_3$. (b) The corresponding "unrolled" network.

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.