Directed Acyclic Graphical Models

CPSC 440/550: Advanced Machine Learning

`cs.ubc.ca/~dsuth/440/23w2`

University of British Columbia, on unceded Musqueam land

2023-24 Winter Term 2 (Jan–Apr 2024)

Higher-Order Markov Models

- Markov models use a density of the form

$$p(x) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3) \cdots p(x_d \mid x_{d-1}).$$

- They support efficient computation but Markov assumption is strong

- A more flexible model would be a second-order Markov model,

$$p(x) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2, x_1)p(x_4 \mid x_3, x_2) \cdots p(x_d \mid x_{d-1}, x_{d-2})$$

  or even higher-order models

- General case is called directed acyclic graphical (DAG) models:
  - They allow dependence on any subset of previous features

DAG Models

- As in Markov chains, DAG models use the chain rule to write

$$p(x_1, x_2, \ldots, x_d) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_d \mid x_1, x_2, \ldots, x_{d-1})$$

- We can alternately write this as:

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

- In Markov chains, we assumed x_j only depends on previous x_{j-1} given past

- In DAGs, x_j can depend on any subset of the past $x_1, x_2, \ldots, x_{j-1}$

DAG Models

- We often write joint probability in DAG models as

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

  where $\text{pa}(j)$ are the "parents" of feature j
  - For Markov chains, the only parent of j is $j - 1$
  - If everything is binary, a variable with k parents needs (up to) 2^{k+1} parameters

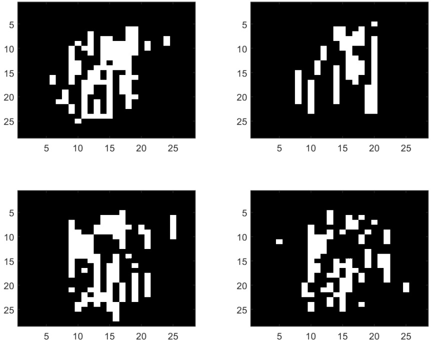
- This corresponds to a set of conditional independence assumptions:

$$p(x_j \mid x_{1:j-1}) = p(x_j \mid x_{\text{pa}(j)})$$

- Variables are independent of previous non-parents, given the parents

MNIST Digits with Markov Chains

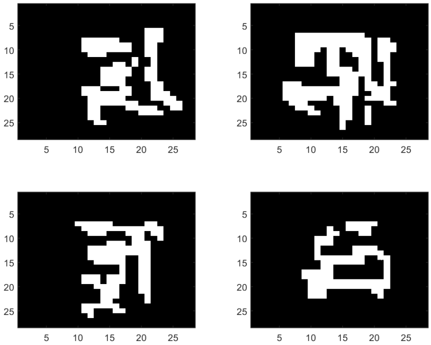
- Recall trying to model digits using an inhomogeneous Markov chain:



- Only models dependence on pixel above, not on 2 pixels above nor across columns

MNIST Digits with DAG Model (Sparse Parents)

- Samples from a DAG model with 8 parents per feature:



- Parents of (i, j) are 8 other pixels in the neighbourhood ("up by 2, left by 2"):

$$\{(i{-}2, j{-}2), (i{-}1, j{-}2), (i, j{-}2), (i{-}2, j{-}1), (i{-}1, j{-}1), (i, j{-}1), (i{-}2, j), (i{-}1, j)\}$$

DAG Models

- "Graphical" name comes from visualizing parents/features as a graph:
  - We have a node for each feature j
  - We place an edge into j from each of its parents
- This graph is not just a visualization tool:
  - Can be used to test arbitrary conditional independences ("d-separation")
  - Graph structure tells us whether message passing is efficient ("treewidth")

Graph Structure Examples

- For a product of independent distributions, we have

$$p(x) = \prod_{j=1}^{d} p(x_j)$$

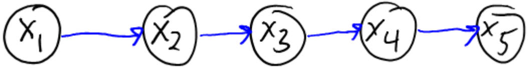
- So, $\mathrm{pa}(j) = \{\}$, and the graph is

Graph Structure Examples

- In a Markov chain, we have

$$p(x) = p(x_1) \prod_{j=2}^{d} p(x_j \mid x_{j-1}),$$

- So, $\text{pa}(j) = \{j-1\}$, and the graph is

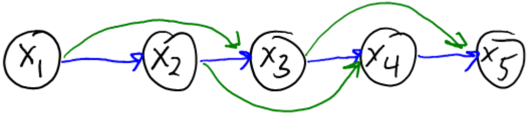


Graph Structure Examples

- In a second-order Markov chain, we have

$$p(x) = p(x_1)p(x_2 \mid x_1)\prod_{j=3}^{d} p(x_j \mid x_{j-1}, x_{j-2}),$$

- So, $\text{pa}(j) = \{j-2, j-1\}$, and the graph is

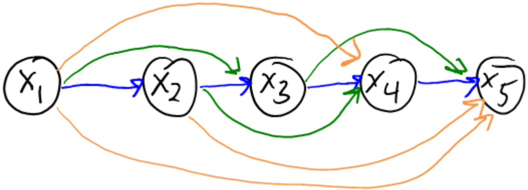


Graph Structure Examples

- With a fully general distribution, we have

$$p(x) = \prod_{j=1}^{d} p(x_j \mid x_{1:j-1})$$

- So, $\mathrm{pa}(j) = \{1, 2, \ldots, j-1\}$, and the graph is

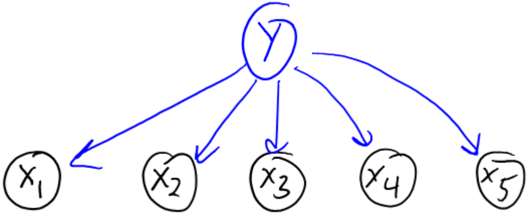


Graph Structure Examples

- In naive Bayes (or GDA with diagonal Σ) we add an extra variable y:

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

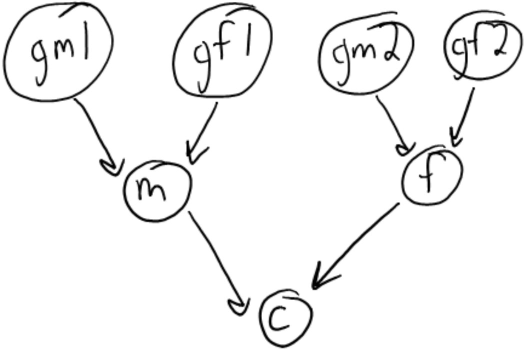
- So, $\text{pa}(y) = \{\}$, $\text{pa}(x_j) = y$:



- Notation inconsistent: both parents of a random variable (x_j) and of index (j)

Graph Structure Examples

- We can consider genetic phylogeny (family trees):



- The "parents" in the graph are an individual's biological parents
  - Independence assumption: only depend on grandparent's genes through parents

First DAG Model

bonus!

- DAGs were first used to analyze inheritance in guinea pigs (1920):

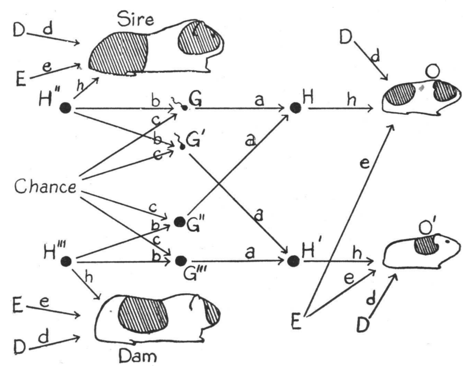


FIG. 5.

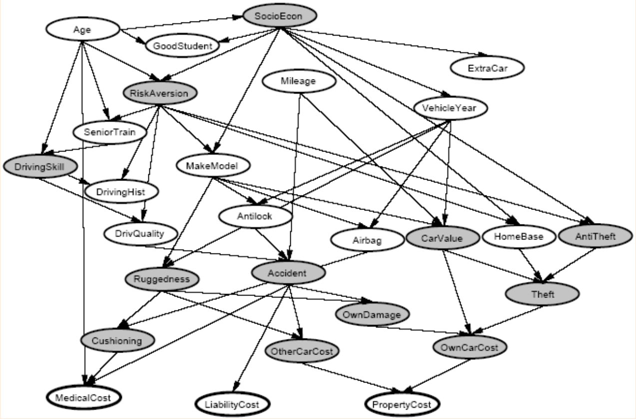
Diagram illustrating the casual relations between litter mates (O, O′) and between each of them and their parents. H, H′, H′′, H,′′′ represent the genetic constitutions of the four individuals, G, G′, G′′, and G′′′ that of four germ cells. E represents such environmental factors as are common to litter mates. D represents other factors, largely ontogenetic irregularity. The small letters stand for the various path coefficients.

https://www.pnas.org/doi/pdf/10.1073/pnas.6.6.320

Example: Vehicle Insurance

bonus!

- Want to predict bottom three "cost" variables, given observed and unobserved values:

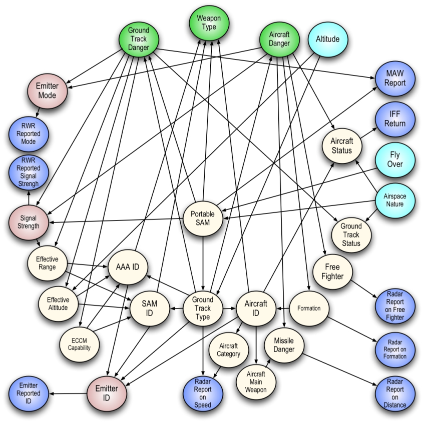


https://www.cs.princeton.edu/courses/archive/fall10/cos402/assignments/bayes

Example: Radar and Aircraft Control

bonus!

- Modeling multiple planes and radar signals:

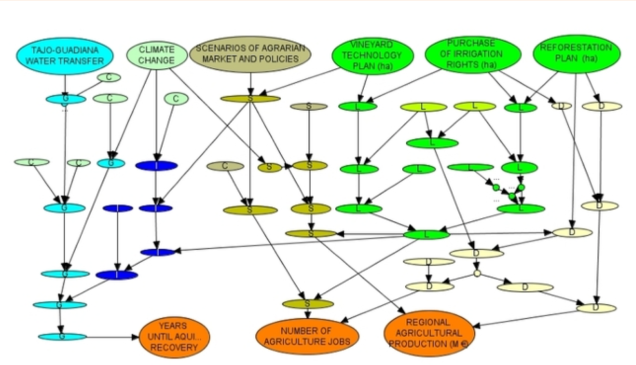


https://pr-owl.org/basics/bn.php

Example: Water Resource Management

bonus!

- Dependencies in environmental monitor and susatainability issues:



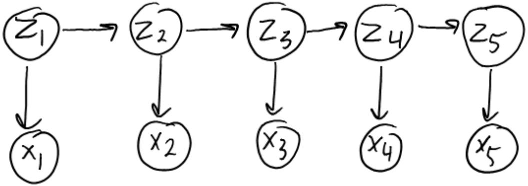
https://www.jstor.org/stable/26268156

Outline

Density Estimators vs. Relationship Visualizers

- In machine learning, DAGs are often used in two different ways:
  1. As a multivariate density estimation method (soon)
  2. As a way to describe the relationships we are modeling
     - All independence assumptions we have used in 340/440 have DAG representation*
     - Includes product of Bernoullis and naive Bayes, but also IID and prior vs. hyper-prior
     - *Except multivariate Gaussians (which can use "undirected" independence)

- For example, we'll talk later about hidden Markov models (HMMs):



- The graph and variable names already give you an idea of what this model does:
  - Hidden variables z_j follow a Markov chain; feature x_j depends on z_j

Extra Conditional Independences in Markov Chains

- Markov assumption in Markov chains: $x_j \perp\!\!\!\perp x_1, x_2, \dots, x_{j-2} \mid x_{j-1}$ for all j

- This implies other independences, like $x_j \perp\!\!\!\perp x_1, x_2, \dots, x_{j-3} \mid x_{j-2}$
  - We didn't assume this directly; it follows from assumptions we made
  - We can use this property to easily compute $p(x_j \mid x_{j-2}, x_{j-3}, \dots, x_1)$:

$$
\begin{aligned}
p(x_j \mid x_{j-2}, x_{j-3}, \dots x_1) &= p(x_j \mid x_{j-2}) \\
&= \sum_{x_{j-1}} p(x_j, x_{j-1} \mid x_{j-2}) \\
&= \sum_{x_{j-1}} p(x_j \mid x_{j-1}, x_{j-2}) p(x_{j-1} \mid x_{j-2}) \\
&= \sum_{x_{j-1}} \underbrace{p(x_j \mid x_{j-1})}_{\text{transition prob}} \underbrace{p(x_{j-1} \mid x_{j-2})}_{\text{transition prob}}
\end{aligned}
$$

- Mathematically showing extra independence assumptions is tedious (see bonus)
- But all conditional independences implied by a DAG can seen in the graph

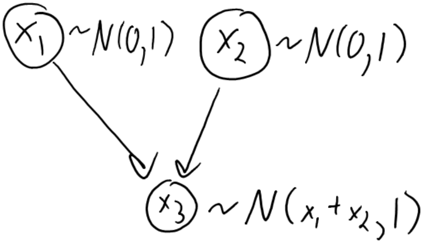
D-Separation: From Graphs to Conditional Independence

- In DAGs: sets of variables A and B are conditionally independent given C if:
  - "D-separation blocks all undirected paths in the graph from any variable in A to any variable in B"
- In the special case of product of independent models our graph is:

- Here there are no paths to block, which implies the variables are independent
- Checking paths in a graph tends to be faster than tedious calculations

D-Separation as Genetic Inheritance

- The rules of d-separation are intuitive in a simple model of gene inheritance:
  - Each node/person has single number, which we'll call a "gene"
  - If you have no parents, your gene is a random number
  - If you have parents, your gene is a sum of your parents plus noise
- For example, think of something like this:



- Graph corresponds to the factorization $p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3 \mid x_1, x_2)$
  - In this model, does $p(x_1, x_2) = p(x_1)p(x_2)$? (Are x_1 and x_2 independent?)

D-Separation as Genetic Inheritance

- Genes of people are independent if knowing one says nothing about the other
- Your gene is dependent on your parents:
  - If I know your parent's gene, I know something about yours
- Your gene is independent of your (unrelated) friends:
  - If you know your friend's gene, it doesn't tell me anything about you
- Genes of people can be conditionally independent given a third person:
  - Knowing your grandparent's gene tells you something about your gene
  - But grandparent's gene isn't useful if you know parent's gene
    - You're conditionally independent of grandparent, given parent

D-Separation Case 0 (No Paths and Direct Links)

Are genes in person x independent of the genes in person y?

- No path: x and y are not related (independent)

  We have $x \perp y$: there are no paths to be blocked

- Direct link: X is the parent of y

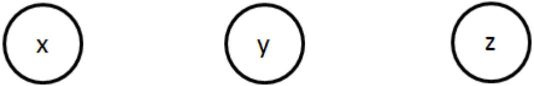
  We have $x \not\perp y$: knowing x tells you about y (direct paths aren't blockable)

  - And similarly, knowing y tells you about x

D-Separation Case 0 (No Paths and Direct Links)

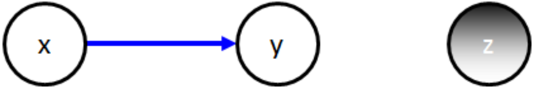
Neither case changes if we have a third independent person z:

- No path: If x and y are independent,



  We have $x \perp y$: adding z doesn't make a path.

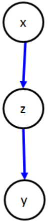
- Direct link: x is the parent of y,



  We have $x \not\perp y \mid z$: adding z doesn't block path
  - We'll use **black or shaded** nodes to denote values we condition on (in this case Z)
    - We sometimes also call the nodes that we condition on the "observations"

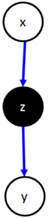
D-Separation Case 1: Chain

- Case 1: x is the grandparent of y
  - If z is the parent we have:



We have $x \not\perp y$: knowing x would give information about y because of z

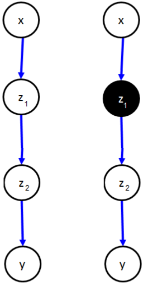
  - But if z is *observed*:



In this case $x \perp y \mid z$: knowing z "breaks" dependence between x and y

D-Separation Case 1: Chain

- The same logic holds for great-grandparents:

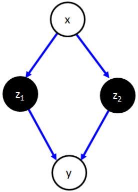


- We have $x \not\perp\!\!\!\perp y$ (left), but $x \perp\!\!\!\perp y \mid z_1$ (right).
  - We also have $x \perp\!\!\!\perp y \mid z_2$ and that $x \perp\!\!\!\perp y \mid z_1, z_2$

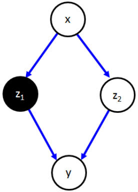
- This case lets you test any independence in Markov chains
  - "Variables are independent conditioned on any variable in betweeen"

D-Separation Case 1: Chain

- Consider weird case where parents z_1 and z_2 share parent x:
  - If z_1 and z_2 are observed:



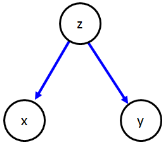
    We have $x \perp y \mid z_1, z_2$: knowing both parents breaks dependency
  - But if only z_1 is *observed*:



    We have $x \not\perp y \mid z_1$: dependence still "flows" through z_2

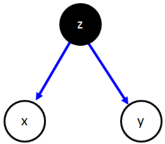
D-Separation Case 2: Common Parent

- Case 2: x and y are siblings
  - If z is a common unobserved parent:



We have $x \not\perp y$: knowing x would give information about y

  - But if z is *observed*:

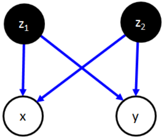


In this case $x \perp y \mid z$: knowing z "breaks" dependence between x and y

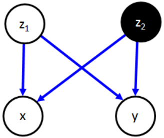
- This is the type of independence used in naive Bayes

D-Separation Case 2: Common Parent

- Case 2: x and y are siblings
  - If z_1 and z_2 are common observed parents:



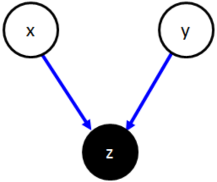
    We have $x \perp\!\!\!\perp y \mid z_1, z_2$: knowing z_1 and z_2 breaks dependence between x and y
  - But if we only observe z_2:



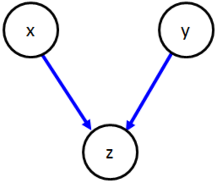
    Then we have $x \not\perp\!\!\!\perp y \mid z_2$: dependence still "flows" through z_1

D-Separation Case 3: Common Child

- Case 3: x and y share a child z:
  - If we observe z then we have:



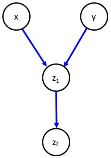
    We have $x \not\perp y \mid z$: if we know z, then knowing x gives us information about Y (Sometimes called "explaining away")
  - But if z is not observed:



    We have $x \perp y$: if you don't observe z then x and y are independent
- Different from Case 1 and Case 2: **not** observing the child blocks the path

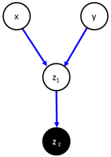
D-Separation Case 3: Common Child

- Case 3: x and y share a child z_1:
  - If there exists an unobserved grandchild z_2:



We have $x \perp y$: the path is still blocked by not knowing z_1 or z_2.

  - But if z_2 is observed:

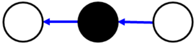


We have $x \not\perp y \mid z_2$: grandchild creates dependence even with unobserved child

- Case 3 needs to consider descendants of child

D-Separation Summary (MEMORIZE)

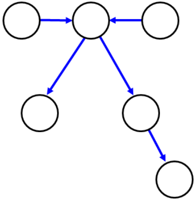
- *Undirected path* from A to B is a path between anything in A and anything in B, ignoring the direction of edges and whether nodes are observed
- A and B are d-separated given C if *all undirected paths* from A to B have (at least) *one* of the following *somewhere* on the path:
  1. P includes a "chain" 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

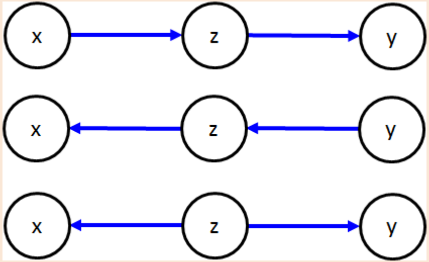
- Instead of using the order $\{1, 2, \ldots, j-1\}$, can have general parent choices
  - So x_2 could be a parent of x_1
- As long the graph is acyclic, there exists some valid ordering

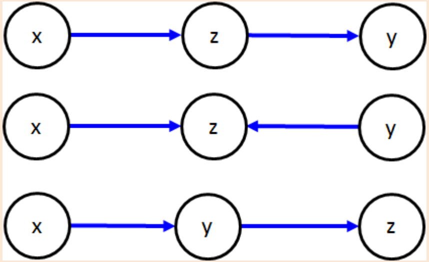
(all DAGs have a "topological order" of variables where parents are before children)

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

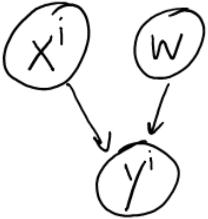
Outline

Tilde Notation as a DAG

- When we write

$$y^{(i)} \sim \mathcal{N}(w^\mathsf{T} x^{(i)}, 1),$$

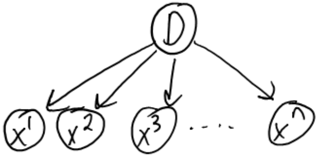
  this can be interpreted as a DAG model:



- "The variables on the right of $\sim$ are the parents of the variables on the left"
  - We can see our standard $x \perp\!\!\!\perp w$ assumption in the graph
  - Common child case: w only depends on x if we know y

IID Assumption as a DAG

- During week 1, our first independence assumption was the IID assumption:

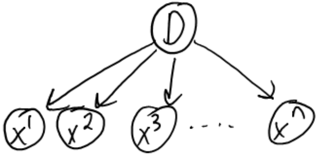


- Training/test examples come independently from data-generating process D
  - e.g. D could be "use a normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$"

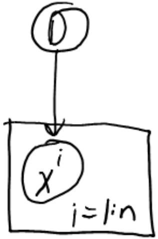
- But D is unobserved, so knowing about some $x^{(i)}$ tells us about the others
  - This why the IID assumptions lets us learn

Plate Notation

- Graphical representation of the IID assumption:

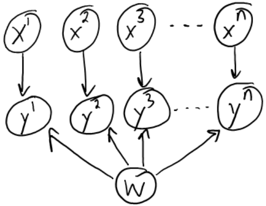


- It's common to represent repeated parts of graphs using plate notation:

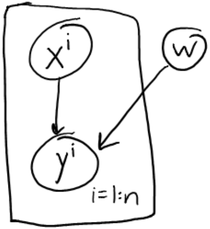


Linear Regresion

- If the $x^{(i)}$ are IID then we can represent linear regression as



or



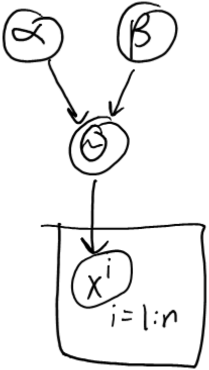
- From d-separation on this graph we have $p(\mathbf{y} \mid \mathbf{X}, w) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}, w)$
  - Our standard assumption that data is independent given parameters

- We often omit the data-generating distribution D
  - But if you want to learn it, then you should remember that it's there

- Discriminative model: here we don't try to model things about $p(x^{(i)})$
- Note that plate reflects parameter tying: that we use same w for all i

IID Bernoulli-Beta Model

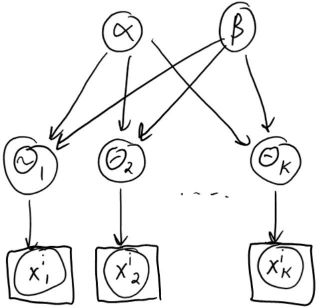
- The Bernoulli-beta model as a DAG (with parameters and hyper-parameters):



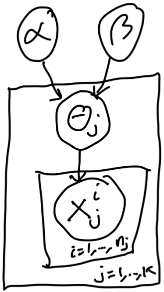
- Notice data is independent of hyper-parameters given parameters
  - This is another of our standard independence assumptions

Non-IID Bernoulli-Beta Model

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



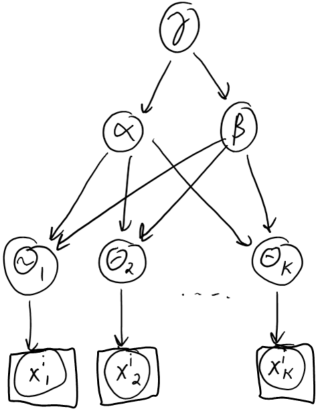
or



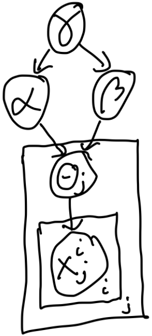
- DAG reflects that we do not tie parameters across all training examples
- Notice that if you fix α and β then you can't learn across groups:
  - The θ_j are d-separated given α and β
- Can also write more succinctly with nested plates

Non-IID Bernoulli-Beta Model

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



or

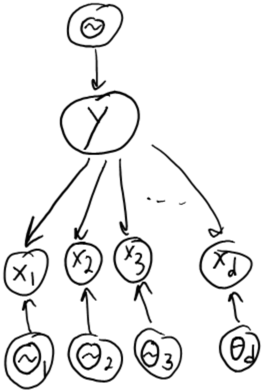


- Needed to avoid degeneracy

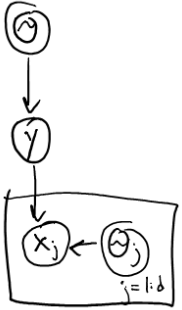
Naive Bayes with DAGs/Plates

- For naive Bayes we have

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



or

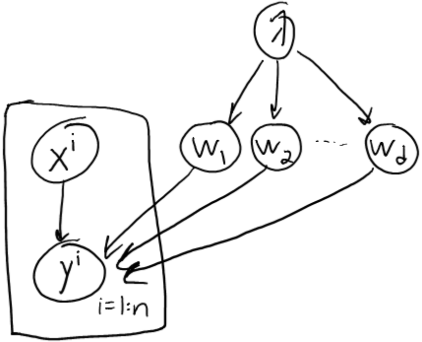


Bayesian Linear Regression as a DAG

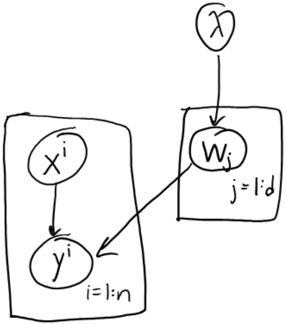
- In Bayesian linear regression we assume

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

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

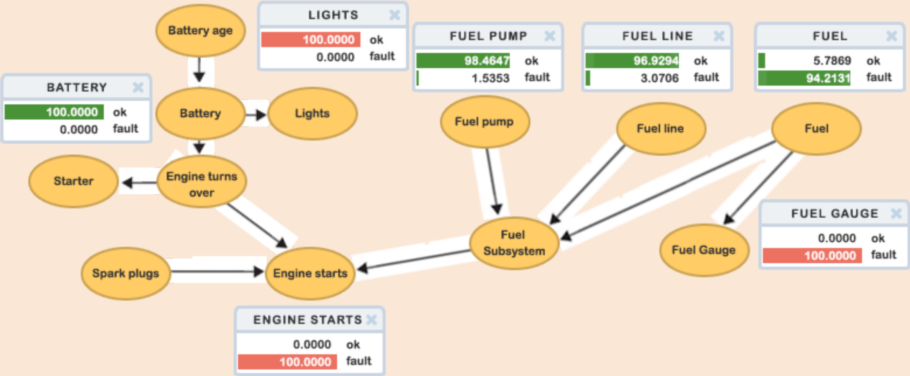
$$p(x_1, x_2, \ldots, x_d) = \prod_{j=1}^{d} p(x_j \mid x_{\mathrm{pa}(j)})$$

- 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): $x_j \sim \mathcal{N}(w^\mathsf{T} x_{\mathrm{pa}(j)}, \sigma^2)$
    - Called a Gaussian belief net; joint distribution becomes a multivariate Gaussian
  - Sigmoid (binary $x_j \in \{-1, +1\}$): $p(x_j \mid x_{j-1}, w) = 1/(1 + \exp(-x_j w^\mathsf{T} x_{\mathrm{pa}(j)}))$
    - 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

bonus!

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



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

DAG Learning and Sampling

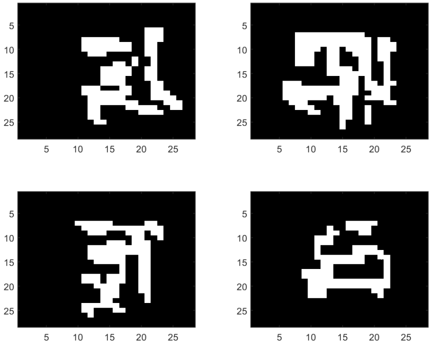
- For $j = 1, 2, \ldots, d$:
  1. Set $\bar{y}^{(i)} = x_j^{(i)}$ and $\bar{x}^{(i)} = x_{\text{pa}(j)}^{(i)}$
  2. Solve a supervised learning problem using $\{\bar{\mathbf{X}}, \bar{\mathbf{y}}\}$
     - Gives you a model of $p(x_j \mid x_{\text{pa}(j)})$

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

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

MNIST Digits with Tabular DAG Model

- Recall our latest MNIST model using a tabular DAG:

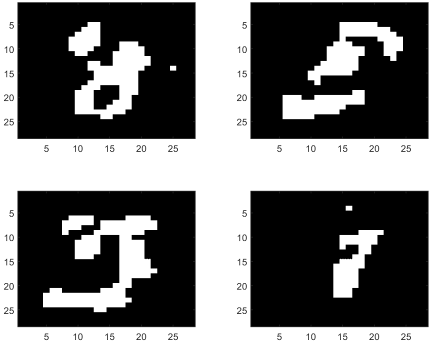


- This model is pretty bad because you only see 8 parents

MNIST Digits with Sigmoid Belief Network

- Samples from sigmoid belief network:

(DAG with logistic regression for each variable)



using all previous pixels as parents (from 0 to 783 parents)

- Models long-range dependencies but has a linear assumption

Exact Inference in DAGs?

bonus!

- 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 $\mathcal{O}(d)$ or $\mathcal{O}(d^3)$

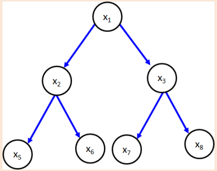
- Continuous-state non-Gaussian DAGs:
  - Inference usually isn't closed-form; need Monte Carlo or variational inference

- 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")

bonus!

- 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
  - Decoding and univariate marginals/conditionals in $\mathcal{O}(dk^2)$
  - Forward-backward applied to tree-structured graphs is called belief propagation
  - Also possible to find the optimal tree given data ("structure learning") – bonus slides
- Less-efficient variant (message passing) on general DAGs: bonus slides

Summary

- DAG models factorize joint distribution into product of conditionals.
  - Usually we assume conditionals depend on small number of "parents".
  - Most models we've seen can be represented as DAGs.
  - Plate notation helps us do this efficiently.
- D-separation allows us to test conditional independences based on graph.
  - Conditional independence follows if all undirected paths are "blocked".
  - Observed values in chain or parent block paths.
  - Unobserved children (with no observed grandchildren) also blocks paths.
- Next time: learning with DAGs.

Extra Conditional Independences in Markov Chains

bonus!

- Proof that x_j is independent of $\{x_1, x_2, \dots, x_{j-3}\}$ given x_{j-2} in Markov chain:

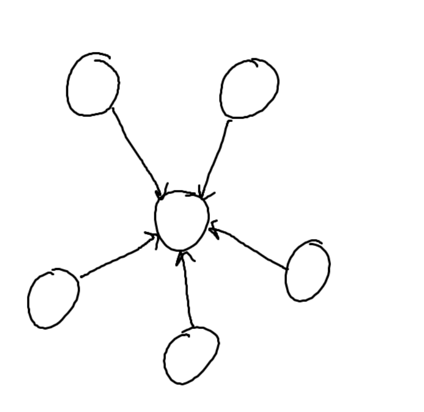
$$
\begin{aligned}
p(x_j \mid x_{j-2}, x_{j-3}, \dots, x_1) &= \frac{p(x_j, x_{j-2}, x_{j-3}, \dots, x_1)}{p(x_{j-2}, x_{j-3}, \dots, x_1)} \quad \text{(def'n cond. prob.)} \\
&= \frac{\sum_{x_{j-1}} p(x_j, x_{j-1}, x_{j-2}, \dots, x_1)}{p(x_{j-2} \mid x_{j-3}, x_{j-4}, \dots, x_1)p(x_{j-3} \mid x_{j-4}, x_{j-5}, \dots, x_1) \cdots p(x_1)} \quad \text{(marg. and chain rule)} \\
&= \frac{\sum_{x_{j-1}} p(x_j \mid x_{j-1}, x_{j-2})p(x_{j-1} \mid x_{j-2}) \dots p(x_2 \mid x_1)p(x_1)}{p(x_{j-2} \mid x_{j-3})p(x_{j-3} \mid x_{j-4}) \cdots p(x_1)} \quad \text{(chain rule and Markov)} \\
&= \frac{p(x_1)p(x_2 \mid x_1) \cdots p(x_{j-2} \mid x_{j-3}) \sum_{x_{j-1}} p(x_j \mid x_{j-1}, x_{j-2})p(x_{j-1} \mid x_{j-2})}{p(x_{j-2} \mid x_{j-3})p(x_{j-3} \mid x_{j-4}) \cdots p(x_1)} \quad \text{(take terms outside} \\
&= \sum_{x_{j-1}} p(x_j \mid x_{j-1}, x_{j-2})p(x_{j-1} \mid x_{j-2}) \quad \text{(cancel out in numerator/denominator)} \\
&= \sum_{x_{j-1}} p(x_j, x_{j-1} \mid x_{j-2}) \quad \text{(product rule)} \\
&= p(x_j \mid x_{j-2}) \quad \text{(marg rule).}
\end{aligned}
$$

- Similar steps could be used to show $X_j \perp\!\!\!\perp X_{j+2} \mid X_{j+1}$, and a variety of other conditional independences like $X_1 \perp\!\!\!\perp X_{10} \mid X_5$.

Conditional Independence in Star Graphs

bonus!

- Consider the following star graph:

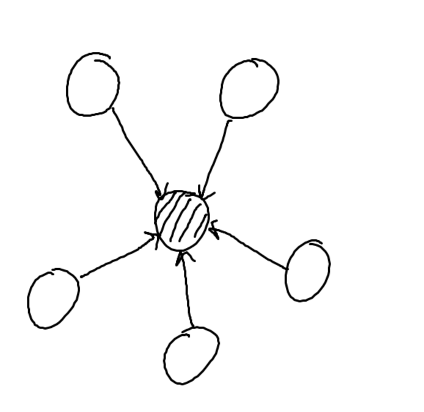


- "5 aliens get together and make a baby alien".
  - Unconditionally, the 5 aliens are independent.

Conditional Independence in Star Graphs

bonus!

- Consider the following star graph:

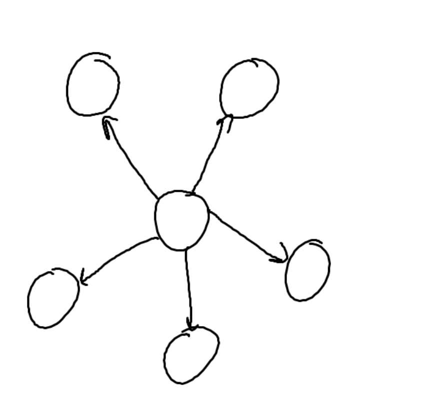


- "5 aliens get together and make a baby alien".
  - Conditioned on the baby, the 5 aliens are dependent.

Conditional Independence in Star Graphs

bonus!

- Consider the following star graph:

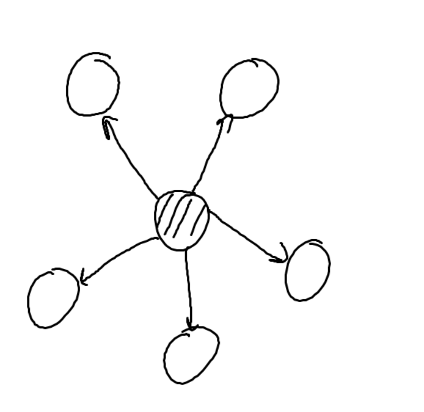


- "An organism produces 5 clones".
  - Unconditionally, the 5 clones are dependent.

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.

Inference in General DAGs

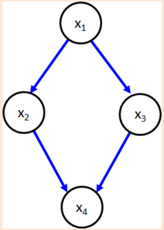
bonus!

- 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

bonus!

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

$$p(x_4) = \sum_{x_3}\sum_{x_2}\sum_{x_1} p(x_1, x_2, x_3, x_4)$$

$$= \sum_{x_3}\sum_{x_2}\sum_{x_1} p(x_4 \mid x_2, x_3) p(x_3 \mid x_1) p(x_2 \mid x_1) p(x_1)$$

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

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

$$p(x_4) = \sum_{x_3}\sum_{x_2} p(x_4 \mid x_2, x_3) M_{23}(x_2, x_3)$$

$$= \sum_{x_3} M_{34}(x_3, x_4),$$

Inference in General DAGs

bonus!

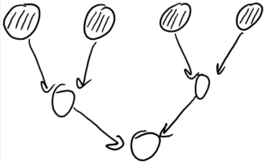
- With 2-variable messages, our cost increases to $O(dk^3)$.
- If we add the edge $x_1 \to x_4$, then the cost is $O(dk^4)$.

  (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.

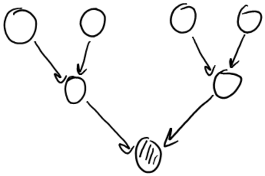
Conditional Sampling in DAGs

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.



DAG Structure Learning

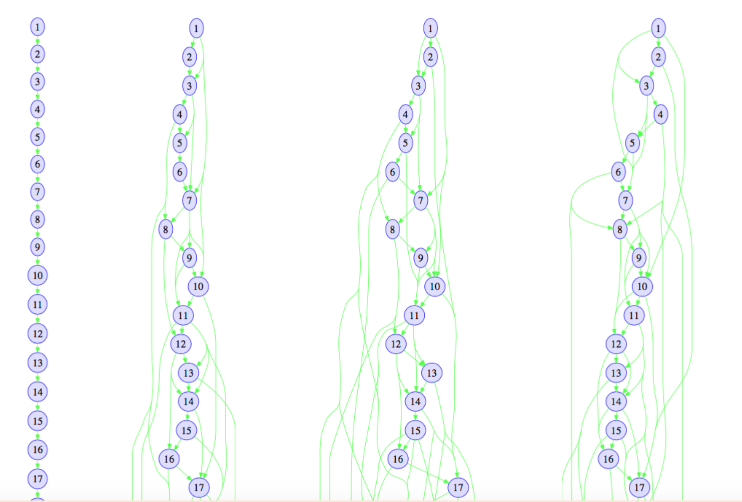
bonus!

- 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 λ 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.

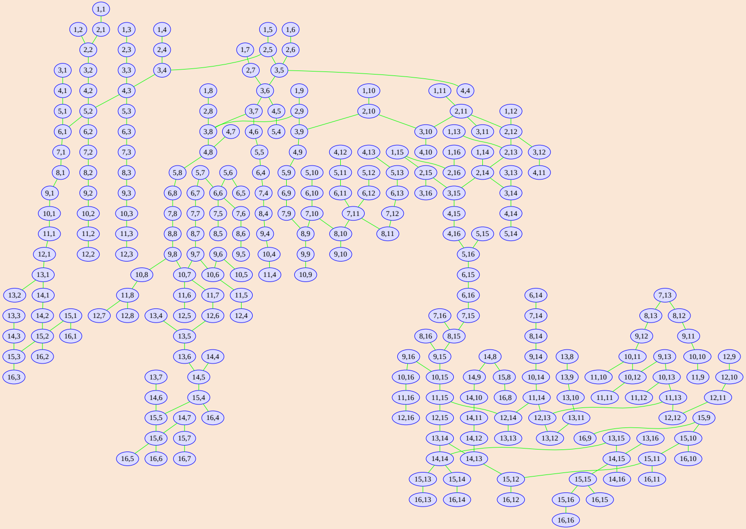
- May have equivalent graphs with the same score (don't trust edge direction).
  - Do not interpret causally a graph learned from data.

- 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:

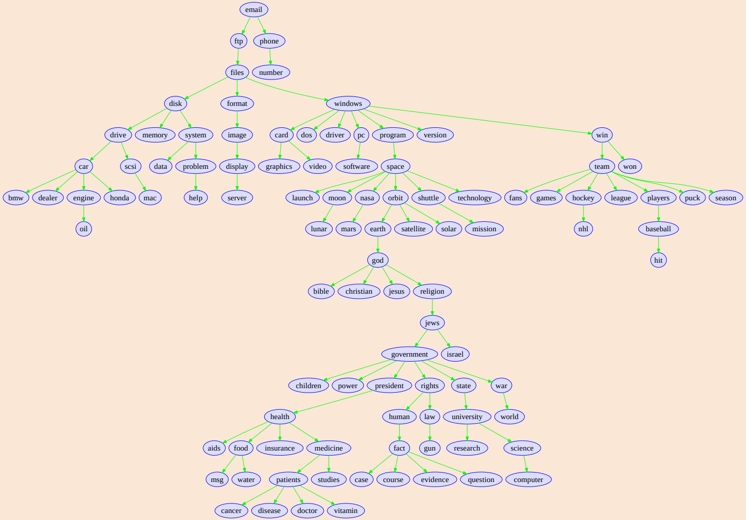
car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

- 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.)