CSE 150A Notes

1 9/26

1.1 Course Information

Prerequisites:

- Programming Knowledge
- Elementary Probability
  - Random Variables - discrete and continuous
  - Expected Values - sums and integrals
- Multivariable Calculus
- Linear Algebra

HW Released Tuesdays, due Monday 24 hr late policy for HW Quizzes in person every Thursday lecture – Based on HW Point lost on quizzes go to the Final Midterm: 10/31 (Week 5) in class Final: 12/5 (Week 10) in class One sheet of handwritten notes allowed

1.2 Course Overview

- Inference and learning in Bayesian Networks
- Markov decision processes for reinforcement learning

Does not cover:

- Neural architectures
- Purely logical reasoning
- Heuristic search (A*)
- Theorem proving
- Genetic algorithms
- Philosophy of AI

2 10/1

Probability Theory: how knowledge affects belief (Poole and Mackworth)
This view is known as the Bayesian view of probability
Other view is the frequentist view; probability is the limit of the relative frequency of an event
Discrete Random Variables: denoted with capital letters
Domain of possible values for a variable, denoted with lowercase letters
Uncoditional (prior) probability: $P(X = x)$
Axioms of Probability:
$P(X = x) \geq 0$
$\sum_{i=1}^{n} P(X = x_i) = 1$
$P(X = x_i \text{ or } X = x_j) = P(X = x_i) + P(X = x_j)$ iff $x_i \neq x_j$
Conditional Probability: $P(X = x_i | Y = y_j)$
In this case X and Y are dependent
Bayes rule: $P(X = x_i | Y = y_j) = \frac{P(Y=y_j|X=x_i)P(X=x_i)}{P(Y=y_j)}$

Product rule: $P(X = x_i, Y = y_j) = P(X = x_i|Y = y_j)P(Y = y_j) = P(x)P(y|x)$
Marginalization: $P(X = x_i) = \sum_{j=1}^{n} P(X = x_i, Y = y_j)$
Independence: $P(X = x_i, Y = y_j) = P(X = x_i)P(Y = y_j)$

3 10/3

Marginal Independence: $P(X = x_i, Y = y_j) = P(X = x_i)P(Y = y_j)$
$P(X|Y) = P(X)$
$P(Y|X) = P(Y)$
Conditional Independence: $P(X, Y|E) = P(X|E)P(Y|E)$
$P(X|Y, E) = P(X|E)$
$P(Y|X, E) = P(Y|E)$
Suppose $X_i \in \{0, 1\}$; then it requires $O(2^n)$ parameters to represent the joint distribution
Goals: compact representation, efficient inference
Use belief to simplify the joint distribution
Conditional Probability Tables (CPT) represent the conditional probability of a variable given its parents
Any inference can be explained in terms of the joint probability, using product rule and marginalization
To perform inference efficiently:
Visualize models as directed acyclic graphs (DAGs)
Exploit the graph structure to simplify and organize calculations
Absent edges represent assumptions of independence
Visual representation of the joint distribution is called a Bayesian Network or belief network
Nodes represent random variables
Edges represent direct dependencies
CPTs for each node describe how each node depends on its parents
Belief network = DAG + CPTs
It's always true from the product rule that $P(X_1, X_2, ..., X_n) = P(X_1)P(X_2|X_1) \cdots = \Sigma_{i=1}^{n} P(X_i|X_1, X_2, \ldots, X_{i-1})$
But suppose in a particular domain that $P(X_i|X_1, X_2, \ldots, X_{i-1}) = P(X_i|\text{parents}(X_i))$
Where parents is a subset of $X_1, X_2, \ldots, X_{i-1}$

To create a belief network:
Choose random variables of interest, choose ordering of the variables, and:
While there are variables left, add the node X_i to the network, with parents the minimum subset satisfying:
$P(X_1, X_2 \ldots X_n) = \Sigma_{i=1}^{n} P(X_i|\text{parents}(X_i))$
Define CPTs
Best order is to take "root causes", then variables they influence, etc.
Edge does not necessarily represent dependence, especially if a bad order is chosen
DAGs encode qualitative knowledge: assumptions of marginal and conditional independence
CPTs encode quantitative knowledge: numerical influences of some variables on others

4 10/8

Expanding on Bayesian networks: $P(X_i|X_1, \ldots, X_{i-1}) = P(X_i|\text{parents}(X_i))$
where parents(X_i) is a subset of $\{X_1, \ldots, X_{i-1}\}$ and denotes the parents of node X_i
True no matter what CPTs are attached to the nodes in the DAG
Representing CPTs:
How to represent $P(X_i|X_1, X_2, \ldots, X_k)$?
Simplest case: X_i, Y_i are binary random variables, how to represent $P(Y = 1|X_1, X_2, \ldots, X_k)$
Possibilities: tabular, logical/deterministic, noisy-OR, sigmoid
Tabular: Table of all values of $X_1, X_2, \ldots, X_k$
and the probability for Y

Exhaustively enumerates a conditional probability for every distribution of parents, able to model arbitrarily complicated dependence, but takes 2^k rows
Logical circuits: AND/OR gate; compact representation for large k, but no model of uncertainty
Noisy-OR CPT: k numbers $p_i \in [0,1]$ to parameterize all 2^k entries in the CPT

$$P(Y=0|X_1,X_2,\ldots,X_k) = \prod_{i=1}^{k}(1-p_i)^{X_i}$$

$$P(Y=1|X_1,X_2,\ldots,X_k) = 1 - \prod_{i=1}^{k}(1-p_i)^{X_i}$$

Why Noisy-OR?
When all parents are 0, probability is 0. When all parents are 1, probability is 1.
With each parent that becomes 1, the higher the probability
When exactly one parent X_j is equal to one: $P(Y=1|X_1=0,\ldots,X_j=1,\ldots X_k=0)=p_j$
Modeling uncertainty:
Intuitively, $p_i \in [0,1]$ is the probability that $X_i=1$ by itself triggers $Y=1$
Logical OR as special case: We recover a logical OR gate by taking the limit $p_i \to 1$ for all parents $i=1,2,\ldots,k$
Canonical application: the parents $\{X_i\}_{i=1}^k$ are diseases, and the child Y is a symptom. The more diseases, the more likely the symptom
Sigmoid CPT: Use k real numbers $\theta_i \in \mathbb{R}$ to parameterize all 2^k entries in the CPT

$$P(Y=1|X_1,X_2,\ldots,X_k) = \sigma(\Sigma_{i=1}^{k}\theta_i X_i)$$

Sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$
Other uses of sigmoid functions: activation functions in neural networks, inverse of the link function for logistic regression
Properties: If $\theta_i > 0$, then $X_i = 1$ favors $Y=1$
If $\theta_i < 0$, then $X_i = 1$ inhibits $Y=1$
Effects can mix in a sigmoid CPT, unlike noisy-OR
d-separation:
Node X_i is conditionally independnt of its non-parents given its parents
More generally: Let X, Y, E refer to disjoint sets of nodes in a BN
When is X conditionally independent of Y given E?
When is $P(X|E,Y)=P(X|E)$?, $P(Y|E,X)=P(Y|E)$, $P(X,Y|E)=P(X|E)P(Y|E)$
We can relate conditional independence of random variables in a BN to properties of its DAG
Must generalize a highly intuitive and natural property of undirected graphs to a more subtle property of DAGs
Let X, Y, E be disjoint sets of nodes in an undirected graph
X and Y are said to be separated by E if every path from a node in X to a node in Y contains one or more nodes in E
d-separation: direction-dependent separation
How is conditional independence in a BN encoded by the structure of its DAG?
Theorem: $P(X,Y|E)=P(X|E)P(Y|E)$ if and only if every path from a node in X to a node in Y is blocked by E
Path: sequence of nodes connected by edges, no nodes repeat
Blocked path: path π is blocked by a set of nodes E if there exists a node Z on π such that:
1. If $Z \in E$, and edges align (node 1 to Z to node 2 or backwards)
2. If $Z \in E$, and edges diverge (Z to node 1 and node 2)
3. If $Z \notin E$, and edges converge (node 1 to Z and node 2 to Z), and descendants of Z are not in E (Z is a collider node)
Explaining by alarm example: A is the alarm, B is burglary, E is earthquake, J is John calls, M is Mary calls
1. Case where B causes A causes J, if A is known, B doesn't matter

2. Case where A causes M and J - common explanation for both
3. Case where E and B cause A: if you know A, then E and B are no longer independent

5 10/10

Inference: Given a set E of evidence nodes, and a set Q of query nodes, how to compute the posterior distribution $P(Q|E)$?
More precisely: How to express $P(Q|E)$ in terms of the CPTs $P(X_i|pa(X_i))$ of the BN, which are assumed to be given?
Tools: Bayes rule, marginalization, product rule, marginal independence, conditional independence, d-separation
Bayes rule: Express $P(Q|E)$ in terms of conditional probabilities that respect the order of the DAG
Marginalization: Use to introduce nodes on the left side of the conditioning bar when they need to appear as parents
Product rule: Use to express join predictions (over multiple variables) in terms of simpler individual predictions
Marginal independence and conditional independence: Use to remove non-informative variables from the right side of the conditioning bar
In what types of BNs is exact inference (exact probability distribution) tractable?
What makes exact inference tractable in these BNs?
How does it scale with the size of the DAG and CPTs?

Polytrees: singly connected belief network; between any two nodes there is at most one path (undirected)
Alternative definition: belief network with no loops
All trees are polytrees, not vice versa
Let X be a node in a polytree, let $U_1, \ldots, U_m$ denote its parents, let $Y_1, \ldots, Y_n$ denote its children
Subgraphs: Draw a box around each parent U_i and all nodes connected to it not via X
Draw a box around each child Y_i and all nodes connected to it not via X
In any polytree, these subgraphs have no common nodes
Proof: parents and children of X are connected via X, a common node in different boxes would create a loop, but there are no loops by the definition of polytrees
Divide and conquer: for each subgraph, it's a polytree. Suggests recursion
Let E denote a set of evidence nodes. Assume $X \notin E$
How to compute $P(X = x|E)$ in a polytree?

Look at evidence from above:
Let E_X^+ denote the evidence nodes connected to X through its parents
Let E_X^- denote the evidence nodes connected to X through its children
$E = E_X^+ \cup E_X^-$
Use Bayes' rule: $P(X = x|E_X^-, E_X^+) = \frac{P(E_X^-|X=x, E_X^+)P(X=x|E_X^+)}{P(E_X^-|E_X^+)}$
Using d-separation: given X, E_X^- and E_X^+ are independent
Above simplifies to $P(X = x|E) = \frac{P(E_X^-|X=x)P(X=x|E_X^+)}{P(E_X^-|E_X^+)}$
First term in numerator: only involves subgraphs below X
Second term in numerator: only involves subgraphs above X
Denominator: if we know how to compute the numerator for any x, then the denominator follows easily:
$P(E_X^-|E_X^+) = \Sigma_x P(E_X^-|X = x)P(X = x|E_X^+)$
Let $\overrightarrow{U}$ denote the parents $(U_1, \ldots, U_m)$ of X
Let $\overrightarrow{u}$ denote the values $(u_1, \ldots, u_m)$ for $\overrightarrow{U}$
$P(X = x|E_X^+) = \Sigma_{\overrightarrow{u}} P(X = x, \overrightarrow{U} = \overrightarrow{u}|E_X^+)$(marginalization)
$= \Sigma_{\overrightarrow{u}} P(\overrightarrow{U} = \overrightarrow{u}|E_X^+)P(X = x|\overrightarrow{U} = \overrightarrow{u}, E_X^+)$ (product rule)

$= \Sigma_{\overrightarrow{u}} P(\overrightarrow{U} = \overrightarrow{u}|E_X^+)P(X = x|\overrightarrow{U} = \overrightarrow{u})$ (conditional independence)
$P(X = x|\overrightarrow{U} = \overrightarrow{u})$ is the CPT at node X

Let $E_{U_i \setminus X} \subset E_X^+$ denote the evidence in the ith parent's box; it is the evidence connected to U_i but not through X
$E_X^+ = E_{U_1 \setminus X} \cup \ldots \cup E_{U_m \setminus X}$
If node X and its descendants are not observed, then X blocks every path between different boxes by case 3 of d-separation
$P(\overrightarrow{U} = \overrightarrow{u}|E_X^+) = \Pi_{i=1}^m P(U_i = u_i|E_X^+)$ (conditional independence)
$= \Pi_{i=1}^m P(U_i = u_i|E_{U_i \setminus X})$ (conditional independence)
$P(U_i = u_i|E_{U_i \setminus X})$ is a recursive instance of the original problem
Overall: for the parents of X: $P(X = x|E_X^+) = \Sigma_{\overrightarrow{u}} P(X = x|\overrightarrow{U} = \overrightarrow{u})\Pi_{i=1}^m P(U_i = u_i|E_{U_i \setminus X})$
Recursion through chilren:
$P(E_X^-|X = x) = \Pi_{j=1}^n P(E_{Y_j \setminus X}|X = x)$ (conditional independence)
When X is observed, it blocks every path between different boxes by case 2 of d-separation
Let Z_{jk} denote the kth parent of Y_j excluding X itself. The nodes Z_{jk} are sometimes called the spouses of X

Let $\overrightarrow{Z}_j$ denote the other parents of Y_j exculding X; let $\overrightarrow{z}_j$ denote a vector of instantiated values for these nodes
Overall: $P(E_{Y_j \setminus X}|X = x) \propto \Sigma_{y_j} P(E_{Y_j}^-|Y_j = y_j)\Sigma_{\overrightarrow{z}_j} P(y_j|\overrightarrow{z}_j, x)\Pi_k P(z_{jk}|E_{Z_{jk} \setminus Y_j})$
$P(E_{Y_j}^-|Y_j = y_j)$ is recursive instance
$\Sigma_{\overrightarrow{z}_j} P(y_j|\overrightarrow{z}_j, x)$ is the CPT given for node Y_j
$\Pi_k P(z_{jk}|E_{Z_{jk} \setminus Y_j})$ is the recursive instance
Where does it end?
Root nodes (no parents), leaf nodes (no children), evidence nodes (to begin we assumed $X \notin E$, otherwise the inference $P(X = x|E)$ was trivial
Use dynamic programming to actually solve the recursion
Runtime is linear in the number of nodes of the BN and size of the CPTs
Exact inference is tractable in polytrees, scaling linearly with the size of BN and CPTs

6 10/15

BNs that are not polytrees
What are general strategies for exact inference in BNs?
Can we transform a loopy BN into a polytree? Then we can run the poly tree algorithm
Using the following example:
Disease D, affects symptoms S_1, S_2, S_3, which all affect visit to doctor V
Node clustering: group nodes into clusters, such that there are no more loops, so what remains is a polytree
In the example, clusty S_1, S_2, S_3 into a cluster S
Merge CPTs at these nodes into a mega-CPT $P(S|D)$
If S_1, S_2, S_3 were binary, then S would have 8 options
$P(S|D)$ would have 8 rows, each row corresponding to a possible value of S ($|S| = |S_1| \cdot |S_2| \cdot |S_3|$)
Easily simplifies graph to a polytree, but node becomes exponentially more complex
Old: $P(S_1|D), P(S_2|D), P(S_3|D)$, now it's $P(S|D) = P(S_1, S_2, S_3|D) = \Pi_{i=1}^3 P(S_i|D)$
$P(V|S_1, S_2, S_3)$, now $P(V|S) = P(V|S_1, S_2, S_3)$
To figure out which nodes to cluster, can just eyeball it
Can we optimize the trade off (nodes to remove to limit the CPT complexities)
No way of solving which clustering leads to maximally efficient inference
Alternative method: instead of clustering nodes, remove them
Remove one or more nodes by instantiating them as evidence, and call the polytree algorithm for each pos-

sible instatiation
For the example: "remove" D; in the first instance, set $D = 0$, in the second instance, set $D = 1$
Changes from $P(S_1|D)$ to $P(S_1|D = 0)$
To calculate $P(V = 1)$: Compute $P(V = 1|D = 0)$ and $P(V = 1|D = 1)$
Combining: $P(V = 1) = \Sigma_d P(D = d, V = 1)$ (marginalization)
$= \Sigma_d P(D = d)P(V = 1|D = d)$ (product rule)
$= P(D = 0)P(V = 1|D = 0) + P(D = 1)P(V = 1|D = 1)$, where $P(D = 0)$ and $P(D = 1)$ are given by the CPTs of the graph
Set of instantiated nodes is called a cutset (in this case, D)
Again, to choose which node to instantiate, eyeball it, then apply polytree algorithm to the resulting BNs
Tradeoff: have to run the polytree algorithm an exponential number of times based on the size of the cutset
No efficient algorithm for choosing the cutset either!
This is why we do approximate inference

Exact inference in belief networks is worse than NP-Hard: #P-Hard
Practical tools: many large loopy BNs remain useful models, so we must resort to approximate methods
Approximate inference in loopy BNs:
Focus on stochastic simulation methods
Rejection sampling
Likelihood weighting
Markov chain Monte Carlo (MCMC)
Sampling a discrete random variable $X \sim P(X)$:
Let X be a discrete random variable with probabilities $P(X = x_i)$. Suppose we can generate random numbers uniformly distributed in $[0, 1]$
Problem: how to sample values of X so that repeated samples are distributed according to $P(X)$?
Solution: Note that $P(X = x_i)$ defines a partition of unity, which maps a random number $r \in [0, 1]$ into a discrete value of X
Sampling from the joint distribution in a discrete BN:
Let $\{X_1, X_2, \ldots, X_m\}$ be discrete random variables in the BN
Problem: how to sample values of $\{X_1, X_2, \ldots, X_m\}$ so that repeated samples are distributed according to $P(X_1, X_2, \ldots, X_m)$?
Solution: the BN defines a generative model with joint distribution $P(X_1, X_2, \ldots, X_m) = \Pi_{i=1}^{m} P(X_i|\text{pa}(X_i))$
To draw samples, we can simply take $X_i \sim P(X_i|\text{pa}(X_i))$ Joint sample:
To draw a joint sample $\{b, e, a, j, m\}$ from $P(B, E, A, J, M)$, we can draw the individual samples:
$b =\sim P(B)$
$e =\sim P(E)$ (from the CPTs)
$a \sim P(A|B = b, E = e)$
$j \sim P(J|A = a)$
$m \sim P(M|A = a)$
Convergence in the limit: $P(b, e, a, j, m) = \lim_{N\to\infty} \frac{\text{count}(B=b,E=e,A=a,J=j,M=m)}{N}$
Let Q denote a set of query nodes, and E denote a set of evidence nodes. How to estimate $P|E)$?
Easy to sample from the BN's joint distribution, harder to sample directly from $P(Q|E)$
Solutions: rejection sampling (very inefficient), likelihood weighting (more efficient), Markov chain Monte Carlo (most efficient)

Rejection sampling:
Problem: how to estimate the relative of the bullseye to the board?
Solution: throw N_0 darts at the wall, ignore/reject those that miss the board, count the number N_1 that hit the board, count the number N_2 that hit the bullseye, take the ratio of the counts
Relative area $= \lim_{N_0\to\infty} \frac{N_2}{N_1}$ where $N_2 \leq N_1 \leq N_0$
Issue is that some of your samples are rejected/go to waste
Generalized: Let Q denote a set of query nodes, and E denote a set of evidence nodes. How to estimate $P(Q = q|E = e)$?
Generate N samples from the BN's joint distribution, reject samples where $E \neq e$, count the samples $N(q, e)$

where $Q = q$ and $E = e$, count the samples $N(e)$ where $E = e$, take the ratio of the counts
$P(Q = q|E = e) \approx \frac{N(q,e)}{N(e)}$
where $N(q, e) \leq N(e) \leq N$
Sample N times; $x_i \sim P(X), y_i \sim P(Y|X = x_i), e_i \sim P(E|X = x_i), q_i \sim P(Q|Y = y_i, E = e_i)$
Define indicator function: $I(z, z') = 1$ if $z = z'$, 0 otherwise
Estimate from ratio; $P(Q = q|E = e) \approx \frac{N(q,e)}{N(e)} = \frac{\Sigma_{i=1}^{N} I(q,q_i)I(e,e_i)}{\Sigma_{i=1}^{N} I(e,e_i)}$
Converges very slowly for rare evidence

Likelihood weighting: Instantiate evidence nodes instead of sampling them, then weight each sample using CPTs at evidence nodes
"Cheat" by fixing evidence nodes to their desired values, "Correct" for cheating by penalizing especially unlikely values
Doesn't discard uninformative samples, no wasted computation, faster convergence
Sample N times; $x_i \sim P(X), y_i \sim P(Y|X = x_i), q_i \sim P(Q|Y = y_i, E = e)$ - E is fixed to e
$P(Q = q|E = e) \approx \frac{\Sigma_{i=1}^{N} I(q,q_i)P(E=e|X=x_i)}{\Sigma_{i=1}^{N} P(E=e|X=x_i)}$ where $P(E = e|X = x_i)$ is the likelihood weight
Example with multiple evidence nodes: estimate $P(Q = q, E_1 = e, E_2 = e')$
Sample N times; $x_i \sim P(X), y_i \sim P(Y|X = x_i), q_i \sim P(Q|Y = y_i, E_1 = e, E_2 = e')$ - (E_1, E_2) is fixed to (e, e')
$P(Q = q, E_1 = e, E_2 = e') \approx \frac{\Sigma_{i=1}^{N} I(q,q_i)P(E_1=e|X=x_i)P(E_2=e'|E_1=e)}{\Sigma_{i=1}^{N} P(E_1=e|X=x_i)P(E_2=e'|E_1=e)}$
Converges in the limit $N \to \infty$
More efficient than rejection sampling; no need to discard, descendants of evidence nodes are conditioned on evidence
Still can be very slow; worse case is when rare evidence is descended from query nodes

7 10/17

If rare evidence affects how query nodes are sampled, likelihood weighting is a good fit
If rare evidence is dependent on query nodes, likelihood weighting is a bad fit (evidence is downstream of query nodes)
We want the evidence nodes to affect how other nodes are sampled, not the other way around
We need a way to sample nodes in any order - not only in a forward pass when they are conditioned on their parents
Markov blanket B_X of a node X is the set of nodes that renders X conditionally independent of all other nodes in the graph
Consists of parents, children, and spouses (parents of children)
Theorem: The node X is conditionally independent of the nodes outside of its Markov blanket given the nodes inside the Markov blanket
Query Nodes Q, Q', evidence nodes E, E'
Monte Carlo is just simulations which are stochastic/randomized - large scale
Markov chain: state is only dependent on the state right before it
MCMC is an idea
Gibbs sampling: initialize and fix evidence nodes to observed values e, e'
Initialize all non-evidence nodes to random variables
Repeat N times:
Pick a non-evidence node X at random
Use Bayes rule to compute $P(X|B_X)$
Resample $x \sim P(X|B_X)$
Take a snapshot of all the nodes in the BN
Estimate:
Count the snapshots $N(q, q') \leq N$ with $Q = q, Q' = q'$
$P(Q = q, Q' = q'|E = e, E' = e') \approx \frac{N(q,q')}{N}$

Properties of MCMC: under reasonable conditions (No CPTs can be deterministic, no deterministic probabilities (no 0 or 1 probabilities))
The sampling procedure defines an ergodic (irreducible and aperiodic) Markov chain over the non-evidence nodes of the BN
Irreducible: from any state, you can reach any other state
Aperiodic: no fixed period of time to reach a state; should not periodically reach the same state
The stationary distribution of this Markov chain is equal to the BN's posterior distribution over its non-evidence nodes
Theoretical guarantees for mixing time, in practice we use burn in time
Mixing time: time it takes to reach stationary distribution
Burn in time: time it takes to reach stationary distribution from a random initial state (initial results are not accurate)
The estimates from MCMC converge in the limit: $\lim_{N\to\infty} \frac{N(q,q')}{N} \to P(Q = q, Q' = q'|E = e, E' = e')$
Difference between LW and MCMC:
LW samples non-evidence nodes from $P(X|\text{pa}(X))$, MCMC from $P(X|B_X)$
LW can read off $P(X|\text{pa}(X))$ from each CPT, MCMC must compute $P(X|B_X)$ for each sample
LW converges slowly for rare evidence in leaf nodes, MCMC is much faster in this case
Learning in BNs:
Given a graph, given data, need to learn the parameters (CPTs)
Sometimes, an expert can provide the DAG and CPTs, but not always especially in complex domains
Alternative: with sufficient data, we can estimate useful models - central idea of machine learning
Applications: language modeling, visual object recognition, recommender systems
Maximum likelihood (ML) estimation:
Model data by the BN that assigns it the highest probability
In other words, choose the DAG and CPTs to maximize P(observed data|DAG and CPTs) (CPT is the unknown)
This probability is known as the likelihood
Data may be unrepresentative or too limited; one failure mode of ML estimation
Assumptions:
DAG is fixed and known over a finite set of discrete random variables $\{X_1, X_2, \ldots, X_n\}$
The data consists of T complete or fully observed instantiations of all the nodes in the BN
CPTs enumerate $P(X_i = x|\text{pa}(X_i) = \pi)$ as lookup tables; each must be estimated for all values of x and π
Data set can be denoted as $\{x_1^{(t)}, x_2^{(t)}, \ldots, x_n^{(t)}\}_t^T = 1$ for $t = 1, 2, \ldots, T$
How to choose CPTs so that the BN maximizes the probability of this dataset?
IID assumption: examples are assumed to be independently and identically distributed from the joint distribution of the BN
Probability of IID data: $P(\text{data}) = \Pi_{t=1}^T P(X_1 = x_1^{(t)}, X_2 = x_2^{(t)}, \ldots, X_n = x_n^{(t)})$
Probability of the tth example: (from the product rule)
$P(X_1 = x_1^{(t)}, X_2 = x_2^{(t)}, \ldots, X_n = x_n^{(t)}) = \Pi_{i=1}^n P(X_i = x_i^{(t)}|X_1 = x_1^{(t)}, \ldots, X_{i-1} = x_{i-1}^{(t)})$
$= \Pi_{i=1}^n P(X_i = x_i^{(t)}|\text{pa}(X_i) = \pi^{(t)})$ (conditional independence)
Calculate log-likelihood:
$\mathcal{L} = logP(\text{data})$
$= log\Pi_{t=1}^T P(x_1^{(t)}, x_2^{(t)}, \ldots, x_n^{(t)})$ (IID)
$= log\Pi_{t=1}^T \Pi_{i=1}^n P(x_i^{(t)}|\text{pa}_i^{(t)})$ (product rule and CI)
$= \Sigma_{t=1}^T \Sigma_{i=1}^n logP(x_i^{(t)}|\text{pa}_i^{(t)})$ ($\log pq = \log p + \log q$)
$= \Sigma_{i=1}^n \Sigma_{t=1}^T logP(x_i^{(t)}|\text{pa}_i^{(t)})$, where $\Sigma_{t=1}^T logP(x_i^{(t)}|\text{pa}_i^{(t)})$ is the sum over examples in t (can be reordered)
Counts: let $\text{count}(X_i = x, \text{pa}_i = \pi)$ denote the number of examples where $X_i = x$ and $\text{pa}_i = \pi$
Next, replace the unweighted sum over examples at each node by a weighted sum over its values and those of its parents
Unweighted: $\mathcal{L} = \Sigma_{i=1}^n \Sigma_{t=1}^T logP(x_i^{(t)}|\text{pa}_i^{(t)})$
$= \Sigma_{i=1}^n \Sigma_x \Sigma_\pi \text{count}(X_i = x, \text{pa}_i = \pi) logP(X_i = x|\text{pa}_i = \pi)$
In this case, $\text{count}(X_i = x, \text{pa}_i = \pi)$ are the constants of the data, and $logP(X_i = x|\text{pa}_i = \pi)$ are the CPTs

to optimize
Log-likelihood for complete data is a triple sum over i - the nodes in the BN; x - the values of each node X_i; and π - the values π of the parents of X_i
How to optimize? Intuitively, the larger the count$(X_i = x, \text{pa}_i = \pi)$, the larger we should choose $P(X_i = x|\text{pa}_i = \pi)$
Solution without proof:
$P_{ML}(X_i = x|\text{pa}_i = \pi) \propto \text{count}(X_i = x, \text{pa}_i = \pi)$
Normalized expressions: $P_{ML}(X_i = x|\text{pa}_i = \frac{\text{count}(X_i=x,\text{pa}_i=\pi)}{\text{count}(\text{pa}_i=\pi)}$ (node with parents)
$P_{ML}(X_i = x) = \frac{\text{count}(X_i=x)}{T}$ (root node)
Each sample is a different value of t
Example problems on how to do ML estimation

8 10/22

Assumptions for learning in BNs:
Discrete random variables $\{X_1, X_2, \ldots, X_n\}$
DAG is specified, assumed to be fixed and known
CPTs enumerate $P(X_i = x|\text{pa}_i = \pi)$
IID data $\{x_1^{(t)}, x_2^{(t)}, \ldots, x_n^{(t)}\}_{t=1}^T$
Each example gives a complete instantiation of the nodes in the belief network
Decomposing the log-likelihood for BN:
$\mathcal{L} = \Sigma_i \Sigma_p i \Sigma_x \text{count}(X_i = x, \text{pa}_i = \pi) \log P(X_i = x|\text{pa}_i = \pi)$
$\mathcal{L}_{i\pi} = \Sigma_x \text{count}(X_i = x, \text{pa}_i = \pi) \log P(X_i = x|\text{pa}_i = \pi)$ represents contribution from row π (parents) of ith node's CPT
Divide and conquer: the overall optimization over $\mathcal{L}$ reduces to many simpler and smaller optimizations over each $\mathcal{L}_{i\pi}$
Special property of ML for complete data

For each node X_i in the BN and for each row π of its CPT, goal is to maximize $\mathcal{L}_{i\pi}$
subject to two contraints: $\sum_x P(X_i = x|\text{pa}_i = \pi) = 1$ (normalized) and $P(X_i = x|\text{pa}_i = \pi) \geq 0$ (nonnegative)
Shorthand: $C_\alpha = \text{count}(X_i = x_\alpha, \text{pa}_i = \pi)$; $P_\alpha = P(X_i = x_\alpha|\text{pa}_i = \pi)$
Compute normalized counts: define $q_\alpha = \frac{C_\alpha}{\Sigma_\beta C_\beta}$ such that $\sum_\alpha q_{alpha} = 1$
All of the following problems are equivalent:
Maximize $\sum_\alpha C_\alpha \log p_\alpha$ such that $\sum_\alpha p_\alpha = 1, p_\alpha \geq 0$
Minimize $\sum_\alpha C_\alpha \log \frac{1}{p_\alpha}$ such that $\sum_\alpha p_\alpha = 1, p_\alpha \geq 0$ (log of 1/a is -log a)
Minimize $\sum_\alpha C_\alpha \log \frac{C_\alpha}{p_\alpha}$ such that $\sum_\alpha p_\alpha = 1, p_\alpha \geq 0$ (C_α is a constant)
Minimize $\sum_\alpha q_\alpha \log \frac{q_\alpha}{p_\alpha}$ such that $\sum_\alpha p_\alpha = 1, p_\alpha \geq 0$ (same thing)
Last one is $KL(q, p)$; proved that $p_\alpha = q_\alpha$ is the unique solution to this problem
Result is $P_{ML}(X_i = x_\alpha|\text{pa}_i = \pi) = \frac{\text{count}(X_i=x,\text{pa}_i=\pi)}{\Sigma_{x'}\text{count}(X_i=x',\text{pa}_i=\pi)}$
For nodes with parents: $P_{ML}(X_i = x_\alpha|\text{pa}_i = \pi) = \frac{\text{count}(X_i=x,\text{pa}_i=\pi)}{\text{count}(\text{pa}_i=\pi)}$
For root nodes: $P_{ML}(X_i = x_\alpha) = \frac{\text{count}(X_i=x)}{T}$
Properties of ML solution:
Asymptotically correct - more data leads to better estimates: $\lim_{T\to\infty} P_{ML}(x_1, x_2, \ldots, x_n) = P(x_1, x_2, \ldots, x_n)$
Problematic for sparse data: what if count$(\text{pa}_i = \pi) = 0$? Then the equation is undefined
If count$(X_i = x, \text{pa}_i = \pi) = 0$, then the equation is 0
Markov models:
Statistical language modeling:
Let w_l denote the lth word in a sentence. How to model $P(w_1, w_2, \ldots, w_n)$?
Simplifying assumptions:

Finite context - to predict the lth work, it is sufficient to consider a finite number of preceding words
$P(w_l|w_1, w_2, \ldots, w_{l-1}) \approx P(w_l|w_{l-(n-1)}, w_{l-(n-2)}, \ldots, w_{l-1})$: $n-1$ previous words
Position invariance: predictions should not depend on where the context occurs in the sentence or text
By product rule: $P(w_1, w_2, \ldots, w_n) = \prod_l P(w_l|w_1, w_2, \ldots, w_{l-1})$
By conditional independence: $= \prod_l P(w_l|w_{l-(n-1)}, w_{l-(n-2)}, \ldots, w_{l-1})$
Different orders of Markov models:
Unigram model: $n = 1$; each word is independent on everything else
Bigram model: $n = 2$; each word depends on the previous word; with d-separation, given the node w_{n-1}, w_n is independent of all other nodes
Next is trigram model; not used in class
Tradeoff is that increasing n also decreases the chance you see that series of words
Same CPT for $P(w_l = w'|w_{l-1} = w)$ for all $l > 1$
How to learn? Collect a large corpus of text with well-defined vocabulary
Count how often word w is followed by the work w'
Count how often word w is followed by any word
Estimate from empirical frequencies:
$P_{ML}(w_l = w'|w_{l-1} = w) = \frac{\text{count}(w \to w')}{\text{count}(w \to *)} = \frac{\text{count}(w \to w')}{\sum_{w''} \text{count}(w \to w'')}$
Problems: no generalization to unseen n-grams;
ML estimates assign zero probability to unseen n-grams
The larger n, the worse the problem
n-gram counts become increasingly sparse as n increases. Many possible but improbable n-grams are not observed

Naive Bayes' model/classifier:
Document classification
Each document can be one of m topics; each document consists of words from a finite vocabulary
Random variables: let $Y \in 1, 2, \ldots, m$ denote the topic of the document; let $X_i \in \{0, 1\}$ denote whether the ith word appears
Each document is mapped to a sparse binary vector of fixed length
Belief network is Y with children $X_1, X_2, \ldots, X_n$
Makes the fairly drastic assumption of conditional independence of the words given the topic
$P(X_1, X_2, \ldots, X_n|Y) = \Pi_{i=1}^n P(X_i|Y)$
Suppose DAG is given, CPTs not specified: how to learn CPTs?
Collect large corpus of documents, label each document by topic, estimate CPTs by maximizing likelihoods
$P_{ML}(Y = y)$ = fraction of documents labeled y in the corpus
$P_{ML}(X_i = x|Y = y)$ = fraction of documents with the label y that contain the ith word in the vocabulary
Once model is learned, now we can infer the label given a new document
Use Bayes' rule: $P(Y = y|X_1, X_2, \ldots, X_n) = \frac{P(X_1, X_2, \ldots, X_n|Y=y)P(Y=y)}{P(X_1, X_2, \ldots, X_n)}$
$= \frac{P(Y=y)\prod_{i=1}^n P(X_i|Y=y)}{P(X_1, X_2, \ldots, X_n)}$ (conditional independence)
$= \frac{P(Y=y)\prod_{i=1}^n P(X_i|Y=y)}{\sum_{y'}\prod_{i=1}^n P(X_i|Y=y')}$ (normalization)
Strengths: easy to learn from data, easy to classify unlabelled documents
Weaknesses: strong conditional independence assumption, no interactions between words, binarization of word counts

9 10/24

Not on midterm :)))))
Data isn't always clean or complete; how to impute missing data?