Lecture 15: Learning Bayesian networks

Today we still assume directed models, complete data

- Priors; Dirichlet priors
- Bias-variance trade-off
- Structure learning
  - Constraint-based approaches
  - Score-based approaches

Recall: MLE and Bayesian parameter estimation

- In MLE we make parameter guesses using *data only*
- This means that we compute sufficient statistics of the data (e.g., counts)
- In Bayes nets, the probability distribution *factorizes* so we can compute the CPDs locally
- The Bayesian approach phrases parameter learning as the problem of inferring the next data item
- This will allow us to work in assumptions we may have about the distributions

Example: Binomial data

- Suppose we observe 1 toss, $x_1 = H$. What would the MLE be?
- In the Bayesian approach,

$$p(\theta|x_1, \ldots x_n) \propto p(x_1, \ldots x_n|\theta)p(\theta)$$

- Assume we have a uniform prior for $\theta \in [0, 1]$, so $p(\theta) = 1$ (remember that θ is a continuous variable!)
- Then we have:

$$\begin{aligned} p(x_2 = H|x_1 = H) &\propto \int_0^1 p(x_1 = H|\theta)p(\theta)p(x_2 = H|\theta)d\theta \\ &= \int_0^1 \theta \cdot 1 \cdot \theta = \frac{1}{3} \end{aligned}$$

Example (continued)

- Likewise, we have:

$$\begin{aligned} p(x_2 = T|x_1 = H) &\propto \int_0^1 p(x_1 = H|\theta)p(\theta)p(x_2 = T|\theta)d\theta \\ &= \int_0^1 \theta \cdot 1 \cdot (1 - \theta) = \frac{1}{6} \end{aligned}$$

- By normalizing we get:

$$\begin{aligned} p(x_2 = H|x_1 = H) &= \frac{\frac{1}{3}}{\frac{1}{3} + \frac{1}{6}} = \frac{2}{3} \\ p(x_2 = T|x_1 = H) &= \frac{1}{3} \end{aligned}$$

- It is as if we had our original data, plus two more tosses! (one heads, one tails)
- Suppose now that we get another toss, $x_2 = T$. What is $p(X_3|x_1 = H, x_2 = T)$?

Prior knowledge

- The prior incorporates prior knowledge or beliefs about the parameters
- As data is gathered, these beliefs do not play a significant role anymore
- More specifically, if the prior is well-behaved (does not assign 0 probability to feasible parameter values), MLE and Bayesian approach both give consistent estimators, so they converge in the limit to the same answer
- But the MLE and Bayesian predictions typically differ after fixed amounts of data. But in the short run, the prior can impact the **speed** of learning!

Multinomial disttribution

- Suppose that instead of a coin toss, we have a discrete random variable with $k > 2$ possible values. We want to learn parameters $\theta_1, \ldots \theta_k$.
- The number of times each outcome is observed, $N_1, \ldots N_k$ represent sufficient statistics, and the likelihood function is:

$$L(\theta|D) = \prod_{i=1}^{k} \theta_i^{N_i}$$

- The MLE is, as expected,

$$\theta_i = \frac{N_i}{N_1 + \cdots + N_k}, \forall i = 1, \ldots k$$

Dirichlet priors

- A **Dirichlet prior** with parameters $\beta_1, \dots \beta_k$ is defined as:

$$P(\theta) = \alpha \prod \theta_i^{\beta_i - 1}$$

- Then the posterior will have the same form, with parameter $\beta_i + N_i$:

$$P(\theta|D) = P(\theta)P(D|\theta) = \alpha \prod \theta_i^{\beta_i - 1 + N_i}$$

- We can compute the prediction of a new event in closed form:

$$P(x_{n+1} = k|D) = \frac{\beta_k + N_k}{\sum(\beta_i + N_i)}$$

Conjugate families

- The property that the posterior distribution follows the same parametric form as the prior is called **conjugacy**
  E.g. the Dirichlet prior is a conjugate family for the multinomial likelihood
- **Conjugate families** are useful because:
  - They can be represented in closed form
  - Often we can do on-line, incremental updates to the parameters as data is gathered
  - Often there is a closed-form solution for the prediction problem

Prior knowledge and Dirichlet priors

- The parameters β_i can be thought of a "imaginary counts" from prior experience
- The **equivalent sample size** is $\beta_1 + \cdots + \beta_k$
- The magnitude of the equivalent sample size indicates how confident we are in your priors
- The larger the equivalent sample size, the more real data items it will take to wash out the effect of the prior knowledge

Bayesian prediction

- Given Dirichlet priors that are independent for each CPD, and given *complete data*, the posterior for each value of a variable will be Dirichlet with parameters $\beta(X_i = k|Parents(X_i))$.
- To choose the prior, $\beta(X_i = k, Parents(X_i))$, we can use an initial parameter vector θ_0, plus an equivalent sample size
- This allows the CPDs to be *updated incrementally* as data is collected

Bias-variance trade-off

- Bias
- Variance

Structure learning problem

- Suppose we have data D sampled from some network G^*, with associated joint distribution p^*. Can we recover G^*?
  - If we have enough data, we can compute p^* accurately
  - But as we have seen before, minimal I-maps are not unique. So in general we cannot recover G^* exactly
- Why should we still strive for perfection?
  - If we have too few edges, then we cannot recover p^*, no matter what we do!
    We introduce spurious dependencies
  - If we have too many edges, the network is too big
    Additionally, we cannot estimate the parameters accurately

Coin tossing example

Suppose we have two coins X_1 and X_2 that are tossed independently. We have a dataset with 100 instances of the experiment:

- 30 head-head instances
- 20 head-tail instances
- 25 tail-head instances
- 25 tail-tail instances

Based on the data, the tosses seem weakly correlated, so the learned network could have an edge between X_1 and X_2

Coin tossing example (continued)

If we wanted to estimate the parameters for the network $X_1 X_2$, we get:

$$P(X_1 = H) = \frac{50}{100} \; P(X_2 = H) = \frac{50}{100}$$

Assume that we build the net $X_1 \rightarrow X_2$. Our estimates are:

$$P(X_2 = H | X_1 = H) = \frac{30}{50} = 0.6 \; P(X_2 = H | X_1 = T) = \frac{25}{50} \approx 0.5$$

The first conditional probability is *not accurate*! This is due to the fact that we split the data into more partitions, so less data is available in each partition

Approaches: Constraint-based search

Perform statistical tests to determine conditional independence relations in the data; then search for a network respecting these independencies

- Very intuitive, in the spirit of Bayes nets
- Allows for efficient search, decoupled from the tests
- But *very sensitive to the tests!* If some test fails, we get a wrong network

Score-based search

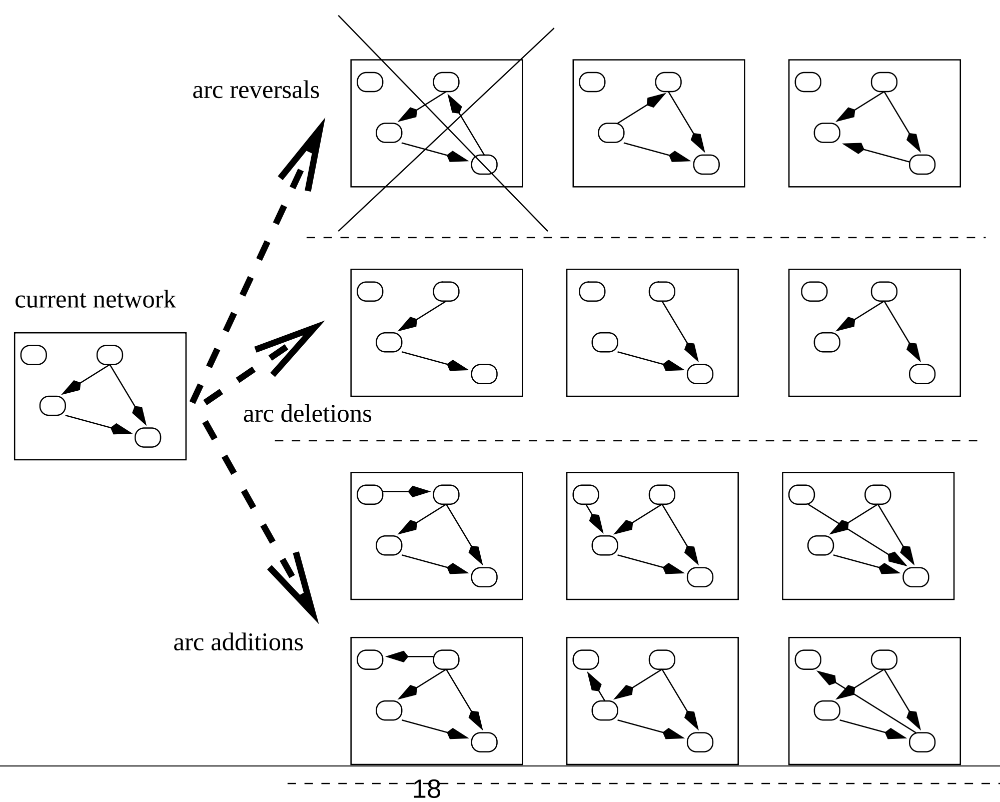
Define a score metric to measure how well the independencies in the structure match the data; search for a network maximizing the score

- Not sensitive to individual failures
- Can make compromises between the extent of dependencies and the cost of adding an edge
- But make a harder search problem

We focus on score-based methods

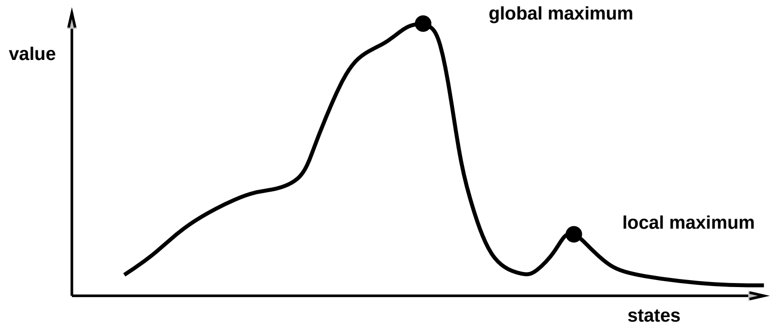
Score-based methods

- Suppose we can define a metric for how well a Bayes net represents a given set of data
- Then we can approach structure learning as a search problem:
  - Start with an initial network (e.g. random, tree, based on prior knowledge etc.)
  - Apply **operators** to change the network: add, delete or reverse arcs



Greedy (hill-climbing) search

Very simple idea: always choose the operator that produces the network with the best score.



- **Local maxima**
- **Plateaux**
- **Ridges:** a set of points might appear like a local maximum

The search space for Bayes net learning has many local maxima

Improvements on greedy search

- Greedily add arcs, then greedily remove arcs
  Works pretty well in some domains
- Tabu-search: keep a list of the k most visited structures, and always look for the best move not generating a structure on the list
  Avoids flat surfaces
- Random restarts: do greedy search repeatedly, starting with different initial networks. Keep the best network seen.
  Very effective in the Bayes net domain!
- Simulated annealing: do not always move to the best network; allow even moves that can lead to a lower scoring net

Simulated Annealing

Main idea: escape local maxima by allowing some apparently **"bad" moves**. But gradually *decrease their size and frequency*

1. Pick a start state *s*
2. Pick a *temperature* parameter *T*, which will control the probability of a random move
3. Repeat:
   (a) Select a random successor *s'* of the current state
   (b) Compute $\Delta E = Value(s') - Value(s)$
   (c) If $\Delta E > threshold$, move to *s'*
   (d) Else move to *s'* with probability $e^{\frac{\Delta E}{T}}$
   (e) Change the temperature according to a *schedule*

Properties of Simulated Annealing

- If T is decreased slowly enough, it is guaranteed to reach the best solution
- But it will take an infinite number of moves!
- When T is high, the algorithm is in an *exploratory phase* (all moves have about the same value)
- When T is low, the algorithm is in an *exploitation phase* (the greedy moves are most likely)

Scoring networks

- The search process requires scoring many networks!
- But the application of an operator only changes the local structure of the network
- We need scoring metrics that can be decomposed into scores for each family

  Then we can compute a **change in score** easily