Learning in HMMs and Bayes Nets

Ron Parr

CPS 271

With some content courtesy of Lise Getoor

The Usual Caveats

- This is only a very brief overview
- A proper treatment takes:
  - More time than we have
  - More patience than you have this late in the semester ☺

First: HMMs

- The easy case:
  - Suppose you have a complete record of
    - Underlying states
    - Observations
- What would you do???

Estimate Probabilities

- Estimate probabilities from relative frequencies
- $P(s_i|s_j)$ = #(s_j to s_i transitions)/#(s_j occurrences)
- $P(o_i|s_j)$ = #(o_i observations in s_j) /#(s_j occurrences)
- Why is this valid???
  - Markov assumption
  - Relative frequency is max likelihood solution

What if Underlying States not Known?

- All we see is trajectories of observations
- True states are hidden variables
- EM to the rescue!
- E step:
  - Use forward-backward/variable elimination to estimate $P(s_is_j)$ for all pairs, for all trajectories
- M step:
  - Compute, $P(s_i|s_j)$, $P(o_i|s_j)$ given results of E-step
  - Note: For soft EM, this means adding fractional state transitions

Getting the pairwise probabilities

- Forward and backward steps are already doing most of the work for this:

$$
\begin{aligned}
P(S_t | e_t \ldots e_0) &= \frac{P(e_t | S_t, e_{t-1} \ldots e_0) P(S_t | e_{t-1} \ldots e_0)}{P(e_t | e_{t-1} \ldots e_0)} \\
&= \alpha P(e_t | S_t e_{t-1} \ldots e_0) P(S_t | e_{t-1} \ldots e_0) \\
&= \alpha P(e_t | S_t) P(S_t | e_{t-1} \ldots e_0) \\
&= \alpha P(e_t | S_t) \sum_{S_{t-1}} P(S_t | S_{t-1}) P(S_{t-1} | e_{t-1} \ldots e_0)
\end{aligned}
$$

$$
\begin{aligned}
P(S_k | e_t \ldots e_0) &= \alpha P(e_t \ldots e_{k+1} | S_k, e_k \ldots e_0) P(S_k | e_k \ldots e_0) \\
&= \alpha P(e_t \ldots e_{k+1} | S_k) P(S_k | e_k \ldots e_0) \\
P(e_t \ldots e_{k+1} | S_k) &= \sum_{S_{k+1}} P(e_t \ldots e_{k+1} | S_k S_{k+1}) P(S_{k+1} | S_k) \\
&= \sum_{S_{k+1}} P(e_t \ldots e_{k+1} | S_{k+1}) P(S_{k+1} | S_k) \\
&= \sum_{S_{k+1}} P(e_{k+1} | S_{k+1}) P(e_t \ldots e_{k+2} | S_{k+1}) P(S_{k+1} | S_k)
\end{aligned}
$$

- Also expressed as $p(x_i,x_j)_t \propto f(x_i)_t P(x_j|x_i) b(x_j)_t$
- Smoothed estimate is the normalized product of the forward and backward estimates

EM for HMMs

- Converges to local optimum
- Most commonly used method for HMM learning
- Used for speech recognition, target detection, tracking, etc.

- How to pick number of states?
  - Just guess?
  - Just pick a big number?
  - When does it matter?
  - Can cross validation help?

Learning Bayesian Networks: Known Structure Case

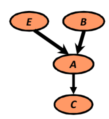
- Given a network structure G

- Learn parameters for network

Goal

- Construct a network that is “closest” to probability that generated the data

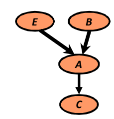
Learning Parameters for a Bayesian Network



- Training data have the form:

$$D = \begin{bmatrix} E[1] & B[1] & A[1] & C[1] \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ E[M] & B[M] & A[M] & C[M] \end{bmatrix}$$

Learning Parameters for a Bayesian Network

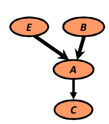


- Since we assume i.i.d. samples, likelihood function is

$$L(\Theta : D) = \prod_m P(E[m], B[m], A[m], C[m] : \Theta)$$

Network parameters (CPTs)

Learning Parameters for a Bayesian Network



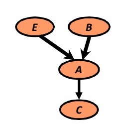
- By definition of Bayes net, we get

$$L(\Theta : D) = \prod_m P(E[m], B[m], A[m], C[m] : \Theta)$$

$$= \prod_m P(E[m] : \Theta) P(B[m] : \Theta) P(A[m] \mid B[m], E[m] : \Theta) P(C[m] \mid A[m] : \Theta)$$

$$\begin{bmatrix} E[1] & B[1] & A[1] & C[1] \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ E[M] & B[M] & A[M] & C[M] \end{bmatrix}$$

Learning Parameters for a Bayesian Network



- Rewriting terms, we get

$$L(\Theta : D) = \prod_m P(E[m], B[m], A[m], C[m] : \Theta)$$

$$= \prod_m P(E[m] : \Theta) \prod_m P(B[m] : \Theta) \prod_m P(A[m] \mid B[m], E[m] : \Theta) \prod_m P(C[m] \mid A[m] : \Theta)$$

$$\begin{bmatrix} E[1] & B[1] & A[1] & C[1] \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ E[M] & B[M] & A[M] & C[M] \end{bmatrix}$$

General Bayesian Networks

Generalizing for any Bayesian network**:**

$$
\begin{aligned}
L(\Theta : D) &= \prod_m P(x_1[m], \ldots, x_n[m] : \Theta) \quad \text{i.i.d. samples} \\
&= \prod_m \prod_i P(x_i[m] \mid Pa_i[m] : \Theta_i) \quad \text{Network factorization} \\
&= \prod_i \prod_m P(x_i[m] \mid Pa_i[m] : \Theta_i) \\
&= \prod_i L_i(\Theta_i : D)
\end{aligned}
$$

- The likelihood **decomposes** according to the structure of the network.
- Bottom line: Optimize each CPT individually

Learning BNs with Missing Data

- Some variable values may be missing
- Assume no systematic pattern to omissions
- And the trick is…
- You guessed it, EM!
- E step computes distribution over missing vars
- M step does "fully obervable" Bayes net learning (using results from E step)

Learning w/unknown Structure

- Great application: Learning structure of biological regulatory networks
- Recall that we typically want to maximize:

$$\frac{P(D \mid \theta) P(\theta)}{P(D)}$$

- Our M step previously maximized P(D|Θ)
- Θ now includes the space of models
- Whis is this more complicated now?
  - Closed form solution? (kind of)
  - Need for regularization

Need for Regularization

- Fully connected model will almost always have higher likelihood if P(Θ) is uniform!
- Typically introduce a structural prior to penalize structures that are complex (e.g. high parent count)
- This complicates search:
  - No closed form solution for best BN
  - Usually do some kind of local search

$$\frac{P(D \mid \theta) P(\theta)}{P(D)}$$

Conclusions

- Learning w/known structure, full observability is easy
- Learning with partial observability is trickier
  - EM is our friend
  - We like EM
- Learning with unknown structure is quite tricky