CS 3750 Advanced Machine Learning

Lecture 19

Structural EM

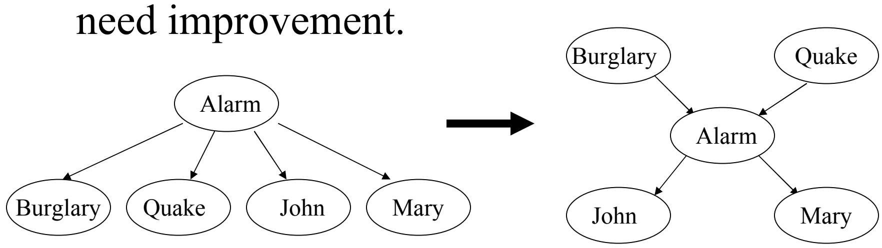
Richard Pelikan
pelikan@cs.pitt.edu
October 29, 2003

Problem Overview

- We have some data $\boldsymbol{D}=(D_1,D_2,\ldots,D_m)$
- with attributes $\boldsymbol{X}=(x_1,x_2,\ldots,x_n)$
  - We will try to model the data using Bayesian networks.
  - The result is a model of the *n* distributions of attributes in the data.
    - Model has parameters $\boldsymbol{\Theta}=(\Theta_1, \Theta_2,\ldots,\Theta_n)$.

Problem Overview

- First model may not be the best model.
- We want to find the parameters of the model that describe the data the best.
- In addition, the structure of the model may need improvement.



Learning Objectives

- Task 1) Learn the parameters $\hat{\Theta}$ which best describes the data.
- Task 2) Learn the structure of the model which best represents the data.
  - Task 1 is pretty easy when the data is complete.
  - Task 2 is usually not as easy, the space of possible models is too large to do a systematic search.

First Task – Learning Data Parameters

- Consider the Bayesian network we attempt to optimize
  - Has a structure $S_i \in \mathcal{S}$, the set of all possible structures
  - Given data *D, a complete dataset*
  - The probability of event E occurring given a model with structure S_i operating on data D is given as follows:

$$P(E \mid D) = \sum_{i=1}^{S} P(E \mid S_i, D) P(S_i \mid D)$$

You have to be kidding me!

First Task – Learning Data Parameters

$$P(E \mid D) = \sum_{i=1}^{S} P(E \mid S_i, D) P(S_i \mid D)$$

$$P(E \mid S_i, D) = \int_{\Theta} P(E \mid \Theta_{S_i}, S_i) P(\Theta_{S_i} \mid D, S_i) d\Theta_{S_i}$$

- Instead, approximate P(E|D) by choosing only the most important models in ***S***
  - Achievable through MAP estimate of $P(S_i|D)$
  - Some people use Gibbs Sampling and other Monte Carlo methods to reduce **S** to a subset of **S**

- Apply Bayes' Rule

$$P(E \mid D) = \sum_{i=1}^{S} P(E \mid S_i, D) P(S_i \mid D)$$

$$P(S_i \mid D) = \frac{P(D \mid S_i) P(S_i)}{P(D)}$$

  - It's the posterior distribution for each structure given data
    - If we maximize this, we can get away without having to exhaustively sum over all possible structures a model can take.
  - Computing the evidence and prior is easy. But how is the likelihood computed?

Decomposing the marginal likelihood

- Likelihood of data given structure is given by

$$P(D \mid S_i) = \int_{\Theta} P(D \mid S_i, \Theta) P(\Theta \mid S_i) d\Theta$$

- We can make several assumptions that simplify the decomposition of the likelihood
  - First assumption: the probability of the data given the structure and parameters of the model is a product of independent factors.

$$P(\boldsymbol{D} \mid S, \Theta) = \prod_{h=1}^{m} P(D_h \mid S, \Theta) = \prod_{h=1}^{m} \prod_{j=1}^{n} P(x_j^h \mid parents_j^h, \Theta)$$

When you make assumptions….

- Second Assumption (Parameter Independence):
  - Parameters associated with each attribute are probabilistically independent of the parameters for other attributes.
  - Parameters associated with an attribute given an instance of its parents are independent of parameters for that attribute given a different instance of its parents.
    - Use this notion to expand from our first assumption…

- From first assumption:

$$P(\boldsymbol{D} \mid S, \Theta) = \prod_{h=1}^{m} \prod_{i=1}^{n} P(x_i^h \mid parents_i^h, \Theta)$$

- Let r_i = number of values that attribute x_i can take
  q_i= number of possible parent combinations
  N_{ijk}= number of cases in D where x_i has value k and parents with values j.

$$= \prod_{i}^{n} \prod_{j}^{q_i} \prod_{k}^{r_i} P(x_i = k \mid parents_i = j, \Theta)^{N_{ijk}}$$

$$= \prod_{i}^{n} \prod_{j}^{q_i} \prod_{k}^{r_i} \Theta_{ijk}^{N_{ijk}}$$

- Third assumption (Parameter modularity):
  - If an attribute has the same parents in two distinct networks, then the parameters for that attribute are identical in both networks.
  - For simplicity's sake, lets assume that the prior distribution of parameters comes from the Dirichlet distribution.
    - The set of parameters for the model **Θ** $= (\Theta_{ij1},\ldots,\Theta_{ijr_i})$ has a set of Dirichlet distributions associated with it, with parameters **α** $= (\alpha_{ijk})_{i=1,\ldots,n;\, j=1,\ldots,q_i;\, k=1,\ldots,r_i}$ as long as the following holds:

$$P(\Theta_{ij1},\ldots,\Theta_{ijr_i} \mid S) = Dirichlet(\Theta_{ij1},\ldots,\Theta_{ijr_i} \mid \alpha)$$

$$= \frac{\Gamma(\sum_{k=1}^{r_i} \alpha_{ijk})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \prod_{k=1}^{r_i} \Theta_{ijk}^{\alpha_{ijk}-1}$$

- Finally, the assembly of assumptions

$$P(D \mid S_i) = \int_{\Theta} P(D \mid S_i, \Theta) P(\Theta \mid S_i) d\Theta$$

$$= \int \prod_i^n \prod_j^{q_i} \prod_k^{r_i} \Theta_{ijk}^{N_{ijk}} \cdot \frac{\Gamma(\sum_{k=1}^{r_i} \alpha_{ijk})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \prod_{k=1}^{r_i} \Theta_{ijk}^{\alpha_{ijk}-1} d\Theta$$

$$= \prod_i^n \prod_j^{q_i} \frac{\Gamma(\sum_{k=1}^{r_i} \alpha_{ijk})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \int \prod_{k=1}^{r_i} \Theta_{ijk}^{N_{ijk}+\alpha_{ijk}-1} d\Theta$$

$$= \prod_i^n \prod_j^{q_i} \frac{\Gamma(\alpha_{ij})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \cdot \frac{\prod_{k=1}^{r_i} \Gamma(a_{ijk} + N_{ijk})}{\Gamma(\alpha_{ij} + N_{ij})}$$

Finally, the marginal likelihood

$$P(D \mid S_i) = \prod_i^n \prod_j^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(a_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

- We can use the marginal likelihood to score the model.
  - Score consists of just factors multiplied together.
  - The score decomposes amongst variables.
- Learning models then amounts to searching the space to maximize the score.
  - Changes in the models alter few terms
  - Scores are rapidly computable if factors are cached

Life Isn't Always Perfect

- Tons of real-world applications have problems with missing values in the data.
  - Problems:
    - We lose that nice decomposition of the probability of data
    - The probability of parameters is also no longer a product of independent terms.

$$P(E \mid S_i, D) = \int_\Theta P(E \mid \Theta_{S_i}, S_i) P(\Theta_{S_i} \mid D, S_i) d\Theta_{S_i}$$

$$P(D \mid S_i) = \int_\Theta P(D \mid S_i, \Theta) P(\Theta \mid S_i) d\Theta$$

    - These integrals can no longer be solved in closed form!

But don't worry…

- We can try to approximate the prediction of an event given a model and data.

  $P(E \mid S_i, D) \approx P(E \mid S_i, \hat{\Theta})$
- $\hat{\Theta}$ Maximizes $P(\Theta \mid S_i, D) \propto P(D \mid \Theta, S_i) P(\Theta \mid S_i)$
  - Use EM or gradient ascent to estimate the parameters.
- In the case of the probability of data given a structure
  - We can estimate the marginal likelihood
    - Stochastic simulation
    - Laplace Approximations
    - Monte Carlo Methods

Structural EM Algorithm

- So we no longer have a complete dataset.
  - ***D*** now consists of two components
    - ***H***, the hidden attributes
    - ***O*,** the observable attributes
  - Our new goal is to find a MAP model
    - Maximizing $P(D \mid S_i) P(S_i)$
    - We just have to assume we can either compute or estimate the marginal likelihood at all times for this to work.

Structural EM Algorithm

- `Procedure Bayesian-SEM(`S_0`)`
  - `For (n=0 until convergence)`
    - `Compute posterior over parameters` $P(\Theta^{S_i} \mid S_n^h, o)$
    - **`E-Step`**`:`
      - `For each S compute`

$$Q(S \mid S_i) = E[\log P(H, o, S_i) \mid S_i^n, o]$$
$$= \sum_h P(h \mid o, S_i^n) \log P(h, o, S_i)$$

    - **`M-Step`**`:`
      - `Choose` S_{i+1} `that maximizes` $Q(S|S_i)$
    - `If` $Q(S|S_i) = Q(S_{i+1}|S_i)$ `then`
      - `Return` S_i

Advantages of Structural EM

- Structural EM is always making progress.
  - Let S_0, S_1,… be the sequence of model structures examined by the Structural EM algorithm
    - The difference in the expected score is always positive

$$\log P(o, S_i^{n+1}) - \log P(o, S_i^n) \geq Q(S_i^{n+1} \mid S^n) - Q(S^n \mid S^n)$$

    - Proof (Friedman)

$$\log \frac{P(o, S_i^{n+1})}{P(o, S_i^n)} = \log \sum_h \frac{P(h, o, S_i^{n+1})}{P(o, S_i^n)} \cdot \frac{P(h \mid o, S_i^n)}{P(h \mid o, S_i^n)}$$
$$= \log \sum_h P(h \mid o, S_i^n) \cdot \frac{P(h, o, S_i^{n+1})}{P(h, o, S_i^n)}$$
$$\geq \sum_h P(h \mid o, S_i^n) \log \frac{P(h, o, S_i^{n+1})}{P(h, o, S_i^n)}$$

Via Jensen's Inequality

Progress of Structural EM

$$\log \frac{P(o, S_i^{n+1})}{P(o, S_i^n)} \geq \sum_h P(h \mid o, S_i^n) \log \frac{P(h, o, S_i^{n+1})}{P(h, o, S_i^n)}$$

$$= E[\log \frac{P(H, o, S_i^{n+1})}{P(H, o, S_i^n)} \mid S_i^n, o]$$

$$= Q(S_i^{n+1} \mid S^n) - Q(S^n \mid S^n)$$

- As long as we choose models on successive iterations which maximizes the expected score at each iteration, then we are guaranteed to be making an improvement.

Applying the Structural EM Algorithm

- In every E-step, we evaluate the expected score $Q(S{:}S_i)$ for each model we examine.
  - The expected score assigns values to **H**
    - Data is complete when we have P(**h,o**|S_i)
    - This means the same decomposition is possible after all

$$E[\log P(H, o \mid S_i)] = \sum_i E[\log F_i(s_i)]$$

    - Ways to compute E[log $F_i(s_i)$]?
      - Many ways to approximate it. For instance, log $F_i(E[s_i])$

Computing Probability Over Hidden Variables

- In the E-Step, we compute the probability of assignments to hidden variables, P(***H***|***o***,S_i).
  - If we want to compute expectation of this term, we can learn the MAP parameters for S_i

$$P(H \mid S_i, o) \approx P(H \mid S_i, \hat{\Theta})$$

  - Use EM, gradient ascent, etc. to find an approximation

Structural EM Algorithm-Revised

- `Procedure Bayesian-SEM(`S_0`)`
  - `For (n=0 until convergence)`
    - `Compute MAP parameters` $\hat{\Theta}$ `for` $P(\hat{\Theta}^{S_i} \mid S_n^h, o)$
    - **`E-Step`**`:`
      - `For each S compute`

$$Q(S \mid S_i) = \sum_i E[\log F_i^M(s_i^M) \mid S_i^n, o, \hat{\Theta}]$$

    - **`M-Step`**`:`
      - `Choose` S_{i+1} `that maximizes Q(S|`S_i`)`
    - `If Q(S|`S_i`) = Q(`S_{i+1}`|`S_i`) then`
      - `Return` S_i

So how does it change structure?

- You have to choose a search method for the algorithm to search for new models.
  - New graph structures can be generated by adding, removing, or reversing an arc.
    - This typically doesn't change a lot of factors, allowing for efficient recomputation of scores for new but differently structured models.

Computing E[log F(**s**)]

- The last remaining question: how to properly approximate E[log F(**s**)]?
  - Linear approximation isn't suited well to exponential functions
  - We can do better by fitting a Gaussian approximation over the values of **s**.

$$E[\log F(s)] \approx \int \log F(s) \varphi(s \mid E[s], \Sigma[s]) ds$$

  - May be easy or hard to evaluate, depending on the dimension of **s**

Structural EM without the inference

- Singh(1997) proposed a way to learn Bayesian network structure by sampling from observed data.
  - Create M datasets by sampling M values for each missing variable from prior distributions of each attribute.
  - For every dataset in M,
    - compute the structure of the model S_i that maximizes $P(S_i|D)$, i.e. the MAP structure model given data.
    - Use EM to learn the conditional probabilities given the observed data and structure S_i
  - Fuse the resulting structures to form a single Bayesian network, and set $\mathbf{\Theta}$ to be the weighted average of parameters over the M datasets.
  - If no convergence occurs, re-sample from the new parameters $\mathbf{\Theta}$**,** M new datasets.

Structural EM Through Sampling

- The main differences between Singh's and Friedman's EM algorithms is that the search space in Singh's version is restricted to a very small set of model structures, whereas Friedman's algorithm is exposed to a wide number of possibilities
- It is also a bit easier to implement Singh's version; much less approximation happening
- But both approaches generally make the same assumptions about how the data and model structure interact