CS228 WINTER 2009 FINAL SOLUTION

ADAM VOGEL

(1) Using Bayes' rule,

$$\begin{aligned} P(\theta|D) &\propto P(D|\theta)P(\theta) \\ &\propto \prod_k \theta_k^{\alpha_k + M_k - 1}. \end{aligned}$$

To find the argmax, we are solving a constrained optimization problem. Furthermore, we can maximize the logarithm of $P(\theta|D)$, since log is strictly monotonically increasing. We then define the Lagrangian as

$$L(\theta, \lambda) = \sum_k (\alpha_k + M_k - 1) \log \theta_k + \lambda \left(\sum_k \theta_k - 1 \right)$$

Taking the partial derivative of the Lagrangian with respect to θ_i and setting it to zero:

$$\begin{aligned} \frac{\partial L(\theta, \lambda)}{\partial \theta_i} &= \frac{\alpha_i + M_i - 1}{\theta_i} + \lambda \\ &= 0 \end{aligned}$$

Solving for θ_i, and noting that $\sum_i \theta_i = 1$,

$$\begin{aligned} \theta_i^* &= \frac{\alpha_i + M_i - 1}{-\lambda} \\ &= \frac{\alpha_i + M_i - 1}{\sum_k \alpha_k + M_k - 1} \end{aligned}$$

(2) The key observation here is Proposition 14.3.2, which says for an ordering $\prec$ and decomposable score (such as BIC), G is a maximizing structure if

$$Pa_i^G = \arg \max_{U \subseteq \{X_j : X_j \prec X_i\}} \text{FamScore}(X_i | U_i : D).$$

When we swap two neighboring elements X_i, X_{i+1} in an ordering $\prec$, we need only re-evaluate the maximizing family scores of X_i and X_{i+1}, the rest stay the same. We thus cache all precomputed family scores, and then note that in $\prec'$, which is identical to $\prec$ except X_i and X_{i+1} are swapped, we need to compute

$$Pa_i^G = \arg \max_{U \subseteq \{X_j : X_j \prec X_i\} \cup X_{i+1}} \text{FamScore}(X_i | U_i : D)$$

and similarly for X_{i_1},

$$Pa_i^G = \arg \max_{U \subseteq \{X_j : X_j \prec X_i\} \setminus X_{i+1}} \text{FamScore}(X_i | U_i : D)$$

The rest of the family scores are the same, and the BIC score of $\prec'$ is simply the sum of the family scores for each X_i.

(3) (a) Our parameters are $\alpha : \mathcal{X} \to \{1, \ldots, k\}$ and $\theta^k_{x=1|u_k}$, the probabilities in the CPD's of G. We define our sufficient statistic $\tau : \mathcal{X} \to \mathbb{R}^l$ as follows. The first component is simply 1, which counts the number of data instances. Then, for every cluster k and assignment u_k to parents U_k, we have a component which is 1 if $x = 1, U_k = u_k$ and a component which is 1 if $x = 0, U_k = u_k$. Then, we define the likelihood of (A, G) by simply multiplying each CPD parameter by the number of times it occured in the dataset:

$$L(\theta : D) = \prod_k \prod_{u_k} \theta^k_{x=1|u_k}{}^{M[x=1|u_k]} \theta^k_{x=0|u_k}{}^{M[x=0|u_k]},$$

where $M[x = 1|u_k]$ is the obvious component of the sufficient statistic tuple.

(b) (i) Edge reversal is complicated in this model: if we have a cluster of size d connected to a single parent and reverse the edge, we would then really have d edges reversed. That just ain't right. Also, reverse = delete + bunch of adds.

(ii) The change in BIC score after applying $o_{k \to k'}(X)$ is the difference in mutual information from the clusters:

$$\Delta(o_{k \to k'}(X)) = I_{\hat{P}}(X; U'_k) - I_{\hat{P}}(X; U_k)$$

We can estimate this efficiently from the emperical distribution $\hat{P}$ from our data.

(iii)
- Add: If we add X as a parent to cluster k, we need to recompute the add operator for every other variable to cluster k, and also the delete operator for all other elements of U_k. Furthermore, we must recompute the node move for any variable from any cluster that moves to k.
- Delete: If we delete X as a parent from cluster k, we need to recompute the add operator for every other variable to cluster k (possible change in mutual information) and also the delete operator for all other elements of U_k. Furthermore, we must recompute the node move for any variable from any cluster that moves to k.
- Node move: If we move X from cluster k to cluster k', we need to evaluate the add and delete operators for every variable with respect to k and k', and also all move operators involving either cluster k or k'. The rest are the same.

(4) (a)
- Compute $P(x|e)$ exactly? No, requires full inference.
- Sample directly from $P(X|e)$? Not directly, but you could do importance sampling in the mutliated network.
- Compute $P(x, e)$? Yes, this is just forward inference (multiply the CPDs evaluated at the assignment).

(b) We construct the obvious clique tree: $C_1 = \{X_1\}$ and $C_i = \{X_i, X_{p(i)}\}$ for $i > 1$, with edges between C_i and $C_{p(i)}$. After calibration, we conduct sampling as follows.

Now draw samples from $Q_{E=e}$, where we force evidence nodes to take the value e, sampling non-evidence nodes as usual. Now, we have to weight this sample by $Q(X|E = e)$, which we can compute exactly

from our calibrated clique tree. Let $x[m]$ be our sample, and note that

$$Q(x[m], e) = \frac{\prod_i \beta_i(X_i, X_{p(i)})}{\prod_i \mu_i, p(i)(X_{p(i)})}.$$

and furthermore [1] we can write $Q(e)$ as

$$Q(e) = \prod_{i:X_i \in E} \sum_{x_{p(i)}} \beta_i(x_i, x_{p(i)}).$$

Therefore, we can compute the posterior of a sample using the calibrated clique tree as

$$\begin{aligned} Q(x[m]|E = e) &= \frac{Q(x[m], e)}{Q(e)} \\ &= \frac{\prod_i \beta_i(X_i, X_{p(i)})}{\left(\prod_{i:X_i \in E} \sum_{x_{p(i)}} \beta_i(x_i, x_{p(i)})\right) \prod_i \mu_i, p(i)(X_{p(i)})} \end{aligned}$$

To weight this particle for estimating $P(X|E = e)$, we define

$$\begin{aligned} w(x[m]) &= \frac{\tilde{P}(x[m]|e)}{Q(x[m]|e)} \\ &= \frac{P(x[m], e)}{Q(x[m]|e)} \end{aligned}$$

We can compute $P(x[m], e)$ by simply multiplying all the CPD's of P evaluated at $(x[m], e)$ and where $Q(x[m]|e)$ is computed as above. To estimate $P(X = x|E = e)$, we let $f(x[m]) = 1\{x = x[m]\}$ and then

$$\hat{P}(X = x|E = e) = \frac{\sum_{m=1}^{M} 1\{x = x[m]\} w(x[m])}{\sum_{m=1}^{M} w(x[m])}$$

(c) To maximize the likelihood in a tree-structured network, we want to weight edge $j \to i$ with $\text{FamScore}(X_i|X_j : D) - \text{FamScore}(X_i : D)$. In the MLE case, this is equivalent to $I_{\hat{P}}(X_j; X_i)$, where $\hat{P}$ is the emperical distribution defined by D.
In our case, we estimate the mutual information using our M samples from $P(X|E = e)$. First,

$$I(X_i; X_j) = E_P[\log \frac{P(X_i, X_j)}{P(X_i)P(X_j)}$$

, thus we estimate $\log P(X_i, X_j|E = e)$, $\log P(X_i|E = e)$ and $\log P(X_j|E = e)$ for all values of X_i and X_j, using our estimator. In the first case, $f(x[m]) = 1\{x_i[m] = x_i \wedge x_j[m] = x_j\}$ and in the second case, we use $f(x[m]) = 1\{X_i[m] = x_i\}$, and similarly for X_j. Thus, by linearity of expectation

$$w_{j \to i} = \log \left(\frac{\sum_{m=1}^{M} (1\{x_i = x_i[m] \wedge x_j = x_j[m]\} - 1\{x_i = x_i[m]\} - 1\{x_j = x_j[m]\}) w(x[m])}{\sum_{m=1}^{M} w(x[m])} \right)$$

[1] This problem abuses notation by defininining our tree-structure Q in terms of X_i and $X_{p(i)}$, as Q also is defined over evidence nodes E, which are not part of X in P. We continue the abuse in our solution.

We then use these weights in a maximum-weighted spanning tree algorithm in the usual way to generate a new structure for Q.

(5) Expectation Maximization with Uniform Initialization

We are given a Naive Bayes model, with data instances $D = \{X[1], \ldots, X[m]\}$ (the label is unobserved), and initial uniform probabilities:

$$\theta^0_c = \frac{1}{|\operatorname{Val}(C)|} \quad \theta^0_{x_i|c} = \frac{1}{|\operatorname{Val}(X_i)|}$$

We first compute the E-step at time 1:

$$\begin{aligned} &Pc\tilde{P}(x[m]|e)Q(x[m]|e) \\ (c|X_i = X_i[m], \theta^0) &= \frac{P(X_i[m]|c,\theta^0)P(c,\theta^0)}{\sum'_c P(X_i[m]|c',\theta^0)P(c',\theta^0)} \\ &= \frac{\theta^0_{x_i[m]|c}\theta^0_c}{\sum'_c \theta^0_{x_i[m]|c'}\theta^0_{c'}} \\ &= \frac{\frac{1}{|\operatorname{Val}(C)|}\frac{1}{|\operatorname{Val}(X_i)|}}{|\operatorname{Val}(C)|\frac{1}{|\operatorname{Val}(C)|}\frac{1}{|\operatorname{Val}(X_i)|}} \\ &= \frac{1}{|\operatorname{Val}(C)|} \\ &= \theta^0_c \end{aligned}$$

Furthermore, since C is not observed in D, $P(c|x[m], \theta^0) = \theta^0_c$. We can then compute the *expected* sufficient statistics:

$$\begin{aligned} \bar{M}_{\theta^0}[x_i, c] &= \sum_m P(x_i, c|x_i = x_i[m], \theta^0) \\ &= \sum_{m:x_i = x_i[m]} P(c|x_i, \theta^0) \\ &= |\{m : x_i = x_i[m]\}|\theta^0_c \end{aligned}$$

For C,

$$\begin{aligned} \bar{M}_{\theta^0}[c] &= \sum_m P(c, \theta^0) \\ &= \frac{M}{|\operatorname{Val}(C)|} \end{aligned}$$

The M-step is then easy:

$$\begin{aligned} \theta^1_{x_i|c} &= \frac{\frac{|\{m:x_i = x_i[m]\}|}{|\operatorname{Val}(C)|}}{\frac{M}{|\operatorname{Val}(C)|}} \\ &= \frac{|\{m : x_i = x_i[m]\}|}{M} \end{aligned}$$

and similarly for C:

$$\begin{aligned} \theta^1_c &= \frac{\frac{M}{|\operatorname{Val}(C)|}}{M} \\ &= \frac{1}{|\operatorname{Val}(C)|} \end{aligned}$$

Now, note that for the next E-step, $\theta^1_{x_i|c}$ is the same for $c = 1$ as $c = 0$, so our estimate of the sufficient statistics for x_i will remain the same:

$$\begin{aligned} P(c|X_i = X_i[m], \theta^1) &= \frac{P(X_i[m]|c, \theta^1)P(c, \theta^1)}{\sum'_c P(X_i[m]|c', \theta^1)P(c', \theta^1)} \\ &= \frac{\theta^1_{x_i[m]|c}\theta^1_c}{\sum'_c \theta^1_{x_i[m]|c'}\theta^1_{c'}} \\ &= \frac{1}{|\operatorname{Val}(C)|} \end{aligned}$$

Thus, the EM algorithm converges after 1 step.

(6) Message Passing with EM

(a) When we change the value of $P(X_i|U_i)$, we note that only messages involving a subset of $\{X_i\} \cup U_i$, will change - the changes in $P(X_i|U_i)$ will drop out of all other messages when we marginalize the clique potential.

Thus, since the clique potential stores a joint distribution over $\{X_i\} \cup U_i$, if we update the incoming messages to reflect the observed data, we can perform the EM update locally.

Our precomputation step is to perform a modified clique-tree calibration. Instead of each clique sending one message $\delta_{i\to j}$ along each edge in the clique tree, we modify this step to send M messages, $\delta^m_{i\to j}$ one for each observed point $d[m]$, which is the distribution over the variables in the sep-set, conditioned on $d[m]$, i.e.

$$\delta^m_{i\to j}(S_{i,j}) = P(S_{i,j}|d[m]).$$

This gives us a modified clique potential, one for each m:

$$\beta^m_{i\to j} = \psi^m_i \prod k \in Nb_i \delta^m_{k\to i},$$

where ψ^m_i is the potential ψ_i reduced by $d[m]$.

After performing this calibration process once, we can then run EM locally at C_j using the messages and the original clique potential. To perform the E-step, we need to compute the expected sufficient statistics for each observed data instance $d[m]$:

$$P(X_i, U_i|d_[m], \theta^t) = \frac{P(X_i|U_i, d[m], \theta^t)P(U_i|d[m], \theta^t)}{\sum_{x',u')P(X_i|U_i,d[m],\theta^t)P(U_i|d[m],\theta^t)}}$$

We can compute these probabilities from β_j in the usual way, except for data instance m we use incoming messages $\delta^m_{i\to j}$ to ensure that we include the relevant information from other cliques about $d[m]$:

$$P(X_i|U_i, d[m], \theta^t) = \frac{\sum_{C_j\setminus\{X_i\}\cup U_i} \beta^m_j(C_j)}{\sum_{C_j\setminus\cup U_i} \beta^m_j(C_j)}$$

and similarly for $P(U_i|d[m], \theta^t)$. We sum these values to compute the expected sufficient statistics, and the M-step is the same.

(b) This conclusion might change if we update the parameters of several families that are all assigned to the same cluster in the cluster tree. Changing the value of the CPD for one family might change an outgoing message, and hence the message we would receive back after recalibration, which possibly changes the update for the other CPD (for instance if they have the same parents). However, if we recalibrate the clique tree in the modified way described above, we could then update another family.

(7) Markov Network Learning

(a) We begin by writing $P(X_j|X_{-j})$:

$$\begin{aligned} P(X_j|X_{-j}) &= \frac{P(X_j, X_{-j})}{P(X_{-j})} \\ &= \frac{\tilde{P}(X_j, X_{-j})}{\sum_{x'_j} \tilde{P}(X'_j, X_{-j})} \end{aligned}$$

Where $\sum_{x'_j}$ is shorthand for $\sum_{x'_{j,1}} \cdots \sum_{x'_{j,N}}$.

(b) We first replace the P terms with their log-linear parametrization, and also take the log:

$$\ln P(X_j|X_{-j}) = \left(\sum_{i:\text{Scope}[f_i] \cap X_j \neq \emptyset} \theta_i f_i(x_j, u_j) \right) - \ln \left(\sum_{x'_j} \exp \left\{ \sum_{i:\text{Scope}[f_i] \cap X_j \neq \emptyset} \theta_i f_i(x'_j, u_j) \right\} \right)$$

where $f_i(x_j, u_j)$ is the value of the pairwise factor f_i evaluated at the subset of x_j, u_j over which it is defined. Taking the partial derivative with respect to a parameter:

$$\begin{aligned} \frac{\partial \ln P(X_j|X_{-j})}{\partial \theta_i} &= f_i[x_j, u_j] - \frac{\frac{\partial}{\partial \theta_i} \sum_{x'_j} \exp \left\{ \sum_{i:\text{Scope}[f_i] \cap X_j \neq \emptyset} \theta_i f_i(x'_j, u_j) \right\}}{\sum_{x'_j} \exp \left\{ \sum_{i:\text{Scope}[f_i] \cap X_j \neq \emptyset} \theta_i f_i(x'_j, u_j) \right\}} \\ &= f_i[x_j, u_j] - \frac{\sum_{x'_j} f_i(x'_j, u_j) \exp \left\{ \sum_{i:\text{Scope}[f_i] \cap X_j \neq \emptyset} \theta_i f_i(x'_j, u_j) \right\}}{\sum_{x'_j} \exp \left\{ \sum_{i:\text{Scope}[f_i] \cap X_j \neq \emptyset} \theta_i f_i(x'_j, u_j) \right\}} \\ &= f_i(x_j, u_j) - E_{x'_j \sim P(x_j|x_{-j}, \theta)}[f_i(x'_j, u_j)] \end{aligned}$$

(c) Applying this to take the gradient of the generalized pseudo-likelihood with respect to the parameters θ is straightforward:

$$\begin{aligned} \frac{\partial}{\partial \theta_i} l_{\text{gen-pseudo}}(\theta : D) &= \frac{\partial}{\partial \theta_i} \frac{1}{M} \sum_m \sum_j \ln P(x_j[m]|x_{-j}[m], \theta) \\ &= \frac{1}{M} \sum_m \sum_j \frac{\partial}{\partial \theta_i} \ln P(x_j[m]|x_{-j}[m], \theta) \\ &= \frac{1}{M} \sum_m \sum_j f_i(x_j[m], u_j[m]) - E_{x'_j \sim P(X_j|x_{-j}[m], \theta)}[f_i(x'_j, u_j[m])] \end{aligned}$$

(8) Decomposable Utility Functions

(a) We use the max-product clique tree algorithm. Form a Markov Random Field G over variables $X_1, \ldots, X_m$, with k factors ϕ_i, where $\text{Scope}(\phi_i) = Y_i$ and $\phi_i(y_i) = U(y_i)$. From this, form a clique tree T over G in the usual way. By family-preservation, after max-calibration of T, each set Y_i will be contained in some clique C_j. In this case, the maximum a posteriori assignment to Y_i is simply that which maximizes β_j, the local clique potential. Furthermore, since we are given that the optimum is unambiguous, we can form the global maximizing assignment by simply setting Y_i to the maximum at its *local* β_j (Proposition 10.3.5). We can calculate the local maximizing assignment by brute force (try all assignments), since we assume that the Y_i's are small.

(b) In the case of a chain decomposition of the utility function, we have that

$$U(X_1, \ldots, X_M) = \sum_{i=1}^{M-1} U(X_i, X_{i+1}).$$

Thus, we can form a clique tree of $M-1$ cliques with $C_i = \{X_i, X_{i+1}\}$ and only the potential $U(X_i, X_{i+1})$. Since this clique tree has treewidth 1, calibration takes time $O(M)$, and calculating the maximizing assignment for each max-calibrated potential takes time $O(d^2)$, giving us $O(Md^2)$ performance overall.

(9) Value of Imperfect Information

Let I be an influence diagram, X and chance variable, and D a decision variable, with no directed path from D to X. Form a new influence diagram I', which is the same as I except for a new binary chance variable $\hat{X}$, whose only parent is X, and which obeys the following conditional distribution:

$$P(\hat{X}|X) = \begin{cases} 1-p & \hat{X} = 0, X = 0 \\ p & \hat{X} = 1, X = 0 \\ q & \hat{X} = 0, X = 1 \\ 1-q & \hat{X} = 1, X = 1 \end{cases}$$

Then the value of imperfect information between X and D is identical to the value of perfect information between $\hat{X}$ and D in I'.