Project. Due in class, November 28, 2007.

ECE 1508F

Probabilistic Inference Algorithms and Machine Learning

Prof. Brendan Frey, frey@psi.toronto.edu

1. Model construction and variable elimination. Think of an interesting toy problem with roughly 6 random variables, that can be represented in a "causal" way, using a Bayesian network. The problem should have some conditional independence structure, *i.e.*, some variables should be independent given other variables. Also, the relationships between variables should be uncertain, rather than deterministic. Use discrete variables only.

a) Briefly describe the problem and the meaning of each random variable.

b) Draw a Bayesian network and provide the conditional distributions (conditional probability tables with numerical entries, or conditional probability density functions). Describe a few conditional independencies represented by the network and why these independencies make sense in your toy problem.

c) Use the Bayes net to write down the joint distribution for all variables as a product of conditional distributions. Don't substitute numbers, just use terms like $P(x|y)$.

d) Convert the Bayes net to an MRF. Indicate the maximal cliques and for each maximal clique, give the corresponding potential in terms of the conditional distributions from the Bayes net (again, don't substitute numbers, use terms like $P(x|y)$). Check your solution by showing that the product of the clique potentials gives the same product of conditional distributions as found in (c).

e) Convert the Bayes net to a factor graph. For each function node, give the function in terms of the conditional distributions from the Bayes net. Again, check your solution.

f) Convert the *MRF* found in (d) to a factor graph. Is the structure of this factor graph different from the structure in (e)? Why or why not?

g) Choose an interesting inference, *i.e.*, a conditional probability distribution that answers an interesting question. The variables you condition on can be thought of as observed variables – pick values for these variables. Noting that an observed variable acts as a constant in a function, simplify the factor graph from (e). Draw the simplified factor graph and give the functions (numerically).

h) Select an ordering of the variables not involved in the conditional distribution selected in (g), and perform elimination until you have a single function node connected to the variable(s) you are computing the distribution over. In each step of elimination, draw the factor graph and give the function (numerically) for the newly-created function node. (This function will depend on the second neighbors of the eliminated variable.) Normalize the final distribution so that it is a conditional distribution. Comment on the result.

2. Affinity propagation. Affinity propagation (manuscript and software available at http://www.psi.toronto.edu/affinitypropagation) takes as input a set of real-valued similarities between pairs of data points and identifies a subset of "exemplars" that represent clusters of data.

a) Using a data set of your choice, with fewer than 2000 training cases, compute a set of similarities and, in 1 or 2 paragraphs, explain how you did this and why.

b) Apply affinity propagation and k-centers clustering (software also available at the above web site) to find clusters in your data. Vary the number of clusters from small (*e.g.* 5) to moderately large (*e.g.* 100) by varying the preference value in affinity propagation or setting k in k-centers clustering. Compare the solutions in terms of the net data similarity achieved, *i.e.*, the sum of the similarities of the training cases to their exemplars.

c) Explore one of the following modifications to affinity propagation and see how it influences the objective function for your data (the software will plot the objective function).

- The availabilities are normally initialized to 0. What happens to the achieved net similarity if the availabilities are initialized to small random numbers and the algorithm is run multiple times? How does the range of the initial values affect the net similarity?

- Damping is critical for avoiding oscillations. However, it may be that a large damping level is needed to avoid oscillations at some points during learning, but can be relaxed later on. Explore methods for adjusting the damping factor during learning. Does adaptive damping achieve the same net similarity? Does it achieve it more quickly?

- The availabilities and responsibilities are normally damped equally. What happens to the dynamics, rate of convergence and the achieved net similarity if they are damped unequally?

- Affinity propagation does not require that all n^2 pairs of similarities are provided. Similarities can be withheld at random, or according to their values (*i.e.*, withholding the lowest ones). How does the achieved net similarity degrade as a function of the number of similarities that are not provided? How does the method used to choose the withheld similarities influence that result?

- Affinity propagation has many advantages over other methods, but one disadvantage (for some applications) is that the number of clusters cannot be pre-specified. However, if you threshold the fused messages $(a + r)$ after each iteration, it is often the case that the resulting number of clusters starts high (all data points) and shrinks down to the final number of clusters. Investigate a method for adaptively changing the preference value while monitoring the solution after each iteration, so as to home in on a particular number of clusters. Is your method stable? Does your method work for a wide range of cluster numbers? How well does the net similarity achieved for a particular k compare to that achieved by manually adjusting the preference until k clusters are found? (Be careful to adjust the net similarity so that one method or the other is not penalized by using a different preference value – perhaps you should only compare the similarities of data points to their exemplars).

3. Gaussian posteriors. Suppose we take n independent, noisy measurements $x_1, \ldots, x_n$ of a real-valued variable z. The noise in measurement x_i is additive Gaussian noise with mean 0 and variance σ_i^2, and the *a priori* distribution of z is Gaussian with mean μ and variance ξ^2.

a) Draw a Bayesian network for the random variables $z, x_1, \ldots, x_n$, and justify the model structure you chose.

b) Using the Bayesian network, give an expression for the joint distribution $P(x_1, \ldots, x_n, z)$ as a product of conditional distributions.

c) Find the *a posteriori* distribution of z, $P(z|x_1, \ldots, x_n)$. To do this, substitute $P(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(z-\mu)^2/2\sigma^2}$ and $P(x_i|z) = \frac{1}{\sqrt{2\pi\xi_i^2}} e^{-(x_i-z)^2/2\xi^2}$ into the above expression for the joint distribution and normalize the joint distribution to obtain an expression for $P(z|x_1, \ldots, x_n)$. It turns out that $P(z|x_1, \ldots, x_n)$ is Gaussian. Find the mean and variance of this Gaussian, in terms of μ, σ^2, $x_1, \ldots, x_n$ and $\sigma_1^2, \ldots, \sigma_n^2$.

4. Variational factor analysis. Factor analysis (see textbook) is useful for learning linear relationships between elements of an input vector. A vector of observed variables $\mathbf{x} = (x_1, \ldots, x_K)$ is modeled using a vector of hidden variables $\mathbf{z} = (z_1, \ldots, z_J)$. The hidden variables are assumed to be independent and Gaussian: $P(\mathbf{z}) = \prod_{j=1}^{J} P(z_j)$, where $P(z_j) = \exp(-z_j^2/2)/\sqrt{2\pi}$. Given the hidden variables, the visible variables are assumed to be independent: $P(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} P(x_k|\mathbf{z})$. Each visible variable is a linear function of the hidden variables, plus Gaussian noise: $P(x_k|\mathbf{z}) = \exp(-(x_k - \sum_j \lambda_{kj} z_j)^2/2\psi_k)/\sqrt{2\pi\psi_k}$, where ψ_k is the noise variance and $\lambda_{k\cdot}$ are the weights used to map the hidden variables to x_k. In this case, the mean of the data is assumed to be 0 (it can be computed and subtracted off to make sure this is true).

Often, in real data sets, each data vector contains a small number of elements that have a large amount of noise. Examples of such 'outliers' include faulty sensor readings, specular reflections in images, and cross-hybridizations in gene expression array data.

We propose to extend factor analysis to include one binary variable, s_k for each input element. $s_k = 1$ indicates that x_k is sporadic noise, while $s_k = 0$ indicates that x_k should fit the factor analysis model. Here, we assume the s's are i.i.d. with $P(s_k = 1) = \rho$. Assume that if $s_k = 1$, the variance of x_k is ψ_{1k} and if $s_k = 0$, the variance of x_k is ψ_{0k}. In practice, we can **fix** ψ_{1k} to the maximum squared difference between pairs of values of x_k in the training set. The joint distribution is

$$P(\mathbf{x}, \mathbf{s}, \mathbf{z}) = \Big(\prod_{j=1}^{J} \frac{1}{\sqrt{2\pi}} e^{-z_j^2/2}\Big)\Big(\prod_{k=1}^{K} \rho^{s_k}(1-\rho)^{1-s_k} \frac{1}{\sqrt{2\pi\psi_{s_k k}}} e^{-(x_k - \sum_j \lambda_{kj} z_j)^2/2\psi_{s_k k}}\Big).$$

a) One might hope that the Gaussian sensor noise model in factor analysis (with variance ψ_k) will account for sporadic deviations in elements of the input vector, making the proposed model unnecessary. Explain why the Gaussian noise model is not appropriate.

b) A different approach is to include an outlier model of the **entire** data vector $\mathbf{x}$. That is, $P(\mathbf{x}) = \alpha P_{\text{FA}}(\mathbf{x}) + (1-\alpha) P_{\text{Outlier}}(\mathbf{x})$. In this case, either the entire data vector is accounted for by the factor analyzer or the outlier model. Explain why this approach won't always work.

c) Exact EM for the proposed model (the one with a different outlier variable for each sensor) would require computing $P(\mathbf{s}, \mathbf{z}|\mathbf{x})$ for each training case. This is a mixture of 2^K Gaussians and for large K computing this distribution is intractable. Derive a variational technique, where $P(\mathbf{s}, \mathbf{z}|\mathbf{x})$ is approximated by $\prod_{k=1}^{K} Q(s_k) \prod_{j=1}^{J} Q(z_j)$. These distributions should be parameterized as follows: $Q(s_k = 1) = \gamma_k$, $Q(z_j) = e^{-(z_j-\mu_j)^2/2\phi_j}/\sqrt{2\pi\phi_j}$. Substitute these expressions into the free energy and simplify it by analytically solving the integrals. Hand in a derivation of the simplified free energy and **also** a derivation of the updates for the variational parameters (E step) **and** model parameters (M step). Describe the order in which you will apply your updates and discuss what stopping criterion you chose for your implementation.

d) Implement the variational method for arbitrary J and K, so that in the E step your software performs I iterations of variational updates before moving on to the M step. Make sure to compute the free energy and ensure it is always decreasing. When debugging your software, it may be helpful to hold all but one variational parameter or model parameter fixed and only update the remaining parameter – if the free energy increases, there is an inconsistency with the implemented update for the parameter in question and the free energy that you are computing. Submit a printout of the code and explain encountered bugs.

e) Apply the algorithm to the toy data posted on the course web site, using $K = 3$, $J = 2$ and $I = 10$. After each E step and M step, compute the free energy and plot it as a function of the number of **steps**, not EM iterations (each iteration of EM will have many points on the plot, corresponding to variational updates and model parameter updates). Check that it is a monotonically decreasing function. Repeat this twice and hand in 3 plots in total.

f) Download the real data, which consists of raster-scanned images of handwritten digits. The goal here is to learn a low-dimensional representation of the data, in a way that is robust to sporadic outlying pixel intensities. For this problem, **fix** $\rho = 0.1$, so that the model will assume most pixels are inliers. Split the data randomly into a training set and a test set. For each value of $J = 1, \ldots, 20$, train a model using 10 random restarts and pick the model with the lowest free energy. Plot the free energy on both the training set and the test set for $J = 1, \ldots, 20$ and choose a value for J. Justify your choice.

g) Repeat (6), using factor analysis (software available on course web site). What value of J was chosen for factor analysis? Discuss the values of J and the achieved free energies for the proposed technique and factor analysis. Use the same training/test set split as was used above.

h) For a randomly selected set of 20 test images, use both the proposed method and factor analysis to reconstruct the test image, without the noise. Print out the 20 test images, the 20 reconstructions produced by the proposed method and the 20 reconstructions produce by factor analysis (these printouts should be of good quality). Discuss how the methods differ in their reconstructions.