CSCI 2950-P Homework 2: Inference & Learning for Undirected Graphical Models

Brown University, Spring 2013

Homework due at 11:59pm on March 22, 2013

We begin by designing algorithms for reliable communication in the presence of noise. We focus on error correcting codes based on highly sparse, *low density parity check* (LDPC) matrices, and use the sum-product variant of the loopy belief propagation (BP) algorithm to estimate partially corrupted message bits. For background information on LDPC codes, see Chap. 47 of MacKay's *Information Theory, Inference, and Learning Algorithms*, which is freely available online: `http://www.inference.phy.cam.ac.uk/mackay/itila/`.

We consider rate 1/2 error correcting codes, which encode N message bits using a $2N$-bit codeword. LDPC codes are specified by a $N \times 2N$ binary parity check matrix H, whose columns correspond to codeword bits, and rows to parity check constraints. We define $H_{ij} = 1$ if parity check i depends on codeword bit j, and $H_{ij} = 0$ otherwise. Valid codewords are those for which the sum of the bits connected to each parity check, as indicated by H, equals zero in modulo-2 addition (i.e., the number of "active" bits must be even). Equivalently, the modulo-2 product of the parity check matrix with the $2N$-bit codeword vector must equal a N-bit vector of zeros. As illustrated in Fig. 1, we can visualize these parity check constraints via a corresponding factor graph. The parity check matrix H can then be thought of as an adjacency matrix, where rows correspond to factor (parity) nodes, columns to variable (codeword bit) nodes, and ones to edges linking factors to variables.

For the LDPC codes we consider, we also define a corresponding $2N \times N$ generator matrix G. To encode an N-bit message vector we would like to transmit, we take the modulo-2 matrix product of the generator matrix with the message. The generator matrix has been constructed (via linear algebra over the finite field GF(2)) such that this product always produces a valid $2N$-bit codeword. Geometrically, its columns are chosen to span the null space of H. We use a systematic encoding, in which the first N codeword bits are simply copies of the message bits. The problems below use precomputed (G, H) pairs produced by Neal's LDPC software: `http://www.cs.utoronto.ca/~radford/ldpc.software.html`.

Question 1:

a) Implement code that, given an arbitrary parity check matrix H, constructs a corresponding factor graph. The parity check factors should evaluate to 1 if an even number of adjacent bits are active (equal 1), and 0 otherwise. Your factor graph representation should interface with some implementation of the sum-product algorithm, either your own code from

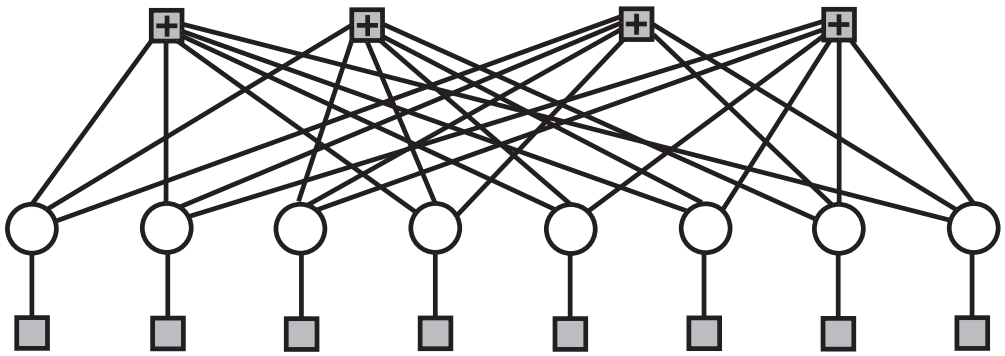

Figure 1: A factor graph representation of a LDPC code linking four factor (parity constraint) nodes to eight variable (message bit) nodes. The unary factors encode noisy observations of the message bits from the output of some communications channel.

Homework 1, or the provided solution code. Define a small test case, and verify that your graphical model assigns zero probability to invalid codewords.

b) *Load the* $N = 128$*-bit LDPC code provided in* `ldpc36-128.mat`*. To evaluate decoding performance, we assume that the all-zeros codeword is sent, which always satisfies any set of parity checks. Using the* `rand` *method, simulate the output of a binary symmetric channel: each transmitted bit is flipped to its complement with error probability* $\epsilon = 0.05$*, and equal to the transmitted bit otherwise. Define unary factors for each variable node which equal* $1 - \epsilon$ *if that bit equals the "received" bit at the channel output, and* ϵ *otherwise. Run the sum-product algorithm for 50 iterations of a parallel message update schedule, initializing by setting all variable-to-factor messages to be constant. After the final iteration, plot the estimated posterior probability that each codeword bit equals one. If we decode by setting each bit to the maximum of its corresponding marginal, would we find the right codeword?*

c) *Repeat the experiment from part (b) for 10 random channel noise realizations with error probability* $\epsilon = 0.05$*. For each trial, run sum-product for 50 iterations. After each iteration, determine an estimated codeword by taking the maximum of each bit's marginal distribution, and evaluate the Hamming distance (number of differing bits) between the estimated and true (all-zeros) codeword. On a single plot, display 10 curves showing Hamming distance versus iteration for each Monte Carlo trial. Is BP a reliable decoding algorithm?*

d) *Repeat part (c), but with a higher error probability* $\epsilon = 0.09$*. Discuss any qualitative differences in the behavior of the loopy BP decoder.*

e) *Load the* $N = 1600$*-bit LDPC code provided in* `ldpc36-1600.mat`*. Using this, we will replicate the visual decoding demonstration from MacKay's Fig. 47.5. Start by converting a* 40×40 *binary image to a* 1600*-bit message vector; you may use the* `logo` *image we provide, or create your own. Encode the message using the provided generator matrix* G*, and add noise with error probability* $\epsilon = 0.08$*. For this input, plot images showing the output of the sum-product decoder after 0, 1, 2, 3, 5, 10, 20, and 30 iterations. The* `rem`

method may be useful for computing modulo-2 sums. You can use the `reshape` *method to easily convert between images and rasterized message vectors.*

f) Repeat part (e) with a higher error probability of your choice, and discuss differences.

We now develop algorithms for learning, from complete observations, undirected graphical models of N binary variables $x_s \in \{0, 1\}$. We focus on models with *pairwise* dependencies, and use a minimal parameterization which allocates one parameter θ_s for each node $s \in \mathcal{V}$, and one parameter θ_{st} for each edge $(s, t) \in \mathcal{E}$. The overall joint distribution is then:

$$p(x \mid \theta) = \exp\left\{ \sum_{s \in \mathcal{V}} \theta_s x_s + \sum_{(s,t) \in \mathcal{E}} \theta_{st} x_s x_t - \Phi(\theta) \right\} \tag{1}$$

$$\Phi(\theta) = \log\left(\sum_{x} \exp\left\{ \sum_{s \in \mathcal{V}} \theta_s x_s + \sum_{(s,t) \in \mathcal{E}} \theta_{st} x_s x_t \right\} \right) \tag{2}$$

The questions below use voting records from senators during the 112th United States Congress (January 3, 2011 until January 3, 2013), as collected by `voteview.org`. We provide data for $N = 13$ senators, and all $L = 486$ bills, in the binary matrix `senatorVotes`. If $x_{s\ell}$ is the vote of senator s on measure ℓ, we let $x_{s\ell} = 1$ if their vote was "Yea", and $x_{s\ell} = 0$ otherwise (a vote of "Nay", or more rarely a failure to vote). We also provide the last names, home states, and party affiliations (1 for Democrat, 2 for Republican) of each Senator. For learning, we interpret the bills as L independent samples from some joint distribution on Senate votes.

For some of the questions below, you need to solve L_1-regularized optimization problems. We recommend using Schmidt's `L1General` Matlab package, which is available at: `http://www.di.ens.fr/~mschmidt/Software/L1General.html`.

Question 2:

a) Consider a pairwise binary MRF as in eq. (1)*. Suppose that the graph is fully disconnected* ($\mathcal{E} = \emptyset$)*. Derive a closed form expression for the maximum likelihood (ML) estimates of the node parameters θ_s. Compute these ML estimates using the full vote dataset, and plot the estimated θ_s values for each senator.*

b) Now allow the pairwise binary MRF of eq. (1) *to have some arbitrary, fixed set of edges. Consider the joint log-likelihood of a dataset with L independent observations. Derive an expression for the gradient of this log-likelihood with respect to some vectorized ordering of the parameters θ. Simplify your answer to be as explicit as possible. Write a function that, given some training dataset, computes the log-likelihood objective value and gradient corresponding to any candidate model parameters θ. Hint: It may be helpful to use your previously developed code for exhaustive, enumeration-based inference.*

c) Using the gradient objective from part (b), and an optimization package such as `L1General`*, write code which computes the ML estimate of the model parameters θ. Assume a fully connected pairwise graphical model, for which $\mathcal{E}$ contains an edge linking every pair of nodes. Apply this code to the full Senate voting record, and plot the log-likelihood of the estimated model after each optimization iteration. Initialize the node parameters θ_s to the ML estimates from part (a), and the edge parameters $\theta_{st} = 0$.*

d) Can the fully connected, pairwise graphical model estimated in part (c) represent an arbitrary joint distribution on N binary variables? If so, explain why. If not, discuss which statistics of the data it does capture.

e) Consider two different models learned from the voting data: the factorized model from part (a), and the fully connected model from part (c). For each model, compute the binary entropy of the corresponding joint distribution:

$$H(\theta) = -\sum_{x} p(x \mid \theta) \log_2 p(x \mid \theta)$$

What do these numbers suggest about the voting patterns of US senators?

f) Suppose that we place a factorized Laplacian prior on our model parameters:

$$p(\theta \mid \lambda) = \prod_{s \in \mathcal{V}} Lap(\theta_s \mid \lambda_s) \prod_{(s,t) \in \mathcal{E}} Lap(\theta_{st} \mid \lambda_{st})$$

$$Lap(\theta \mid \lambda) = \frac{\lambda}{2} \exp\{-\lambda|\theta|\}$$

Derive an objective whose minimum, given a dataset with L training examples and fixed hyperparameters λ, gives the maximum a posteriori *(MAP) estimate of θ. Adapt the code from part (c) to numerically compute MAP estimates.*

g) Let $\lambda_{st} = \bar{\lambda}$ for all pairs of nodes, and $\lambda_s = 0$ (the limit as the variance of the node parameter priors approaches infinity). Use the votes for the first 400 bills as a training set, and the remaining 86 bills as a validation set. For a range of possible $\bar{\lambda}$, find MAP parameter estimates. For each learned model, evaluate the log-probability of the validation data, and plot these probabilities as a function of $\bar{\lambda}$. Use a logarithmic scale when choosing candidate $\bar{\lambda}$, and when making this plot.

h) Suggest a way of associating graphical structures to the models learned in part (g). Plot the graph corresponding to the $\bar{\lambda}$ which produced the highest validation log-likelihood, as well as the graph with the smallest number of edges that was nevertheless connected. In both cases, label the nodes of the graphs with the names of the corresponding senators.

Consider the undirected graphical model in Figure 2, in which we assume the model potentials are associated with the cliques of the graph. In the questions below, assume that each variable in the input graphical model takes one of M discrete states.

Question 3:

a) By adding appropriate edges, create a triangulated version of the graph in Fig. 2. Hint: Recall that certain simple inference algorithms implicitly produce graph triangulations.

b) Construct a junction tree corresponding to the triangulated graph from part (a). Describe how you verify that your junction tree is valid.

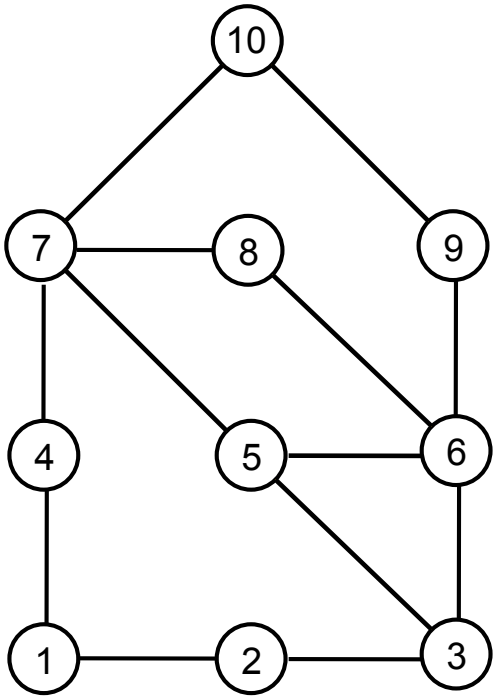

Figure 2: An undirected graphical model of ten random variables.

c) Suppose that you use this junction tree to implement the Shafer-Shenoy inference algorithm, with an optimal schedule that computes each message once. What are the computation requirements? What are the storage requirements? You may use order-of-magnitude estimates to quantify individual message updates, but your analysis should account for the topology and node cardinalities of this particular junction tree.