CS 536: Homework 4

Due: April 11, 11:59pm EST

Instructions: Same as homework 1.

1 Belief Propagation for MAP Assignment (20 pts)

In class we saw that given an MRF over $X_1, X_2, \ldots X_n$ such that the underlying graph is a tree, belief propagation runs in polynomial time and provides marginal probabilities $P(X_i)$ for each X_i. Show how you would modify belief propagation to also efficiently compute the *Maximum a Posteriori (MAP)* assignment, i..e, $argmax_{X_1,X_2,\ldots,X_n} P(X_1, X_2, \ldots, X_n)$, when the MRF is a tree.

2 Underfitting of Boosting Under Noise (20 pts)

In this problem you will show that AdaBoost is susceptible to uniform noise in the data. Consider a training set $(X_1, y_1), (X_2, y_2), \ldots (X_m, y_m)$ and a finite set of base classifiers or weak learners $H = \{h_1, h_2, \ldots h_{|H|}\}$ such that the target function can be written as $sgn(\sum_i w_i h_i(x))$ for some weights w_i's. Now consider a noise model where each $y_i \in \{+1, -1\}$ is flipped independently with probability 0.2. Design an appropriate training set and a set of base classifiers such that running AdaBoost for a long time on the noisy dataset will produce a final classifier that has error $\frac{1}{2}$ with respect to the target function, i.e., it disagrees with the target on exactly half of the training set.

[Note: Your construction must make sure that each step there is weak learner in the set H that can be output, i.e., AdaBoost should not get stuck.]

3 Learning Graph Structure of MRFs (20 pts)

Consider an MRF $P(X_1, X_2, \ldots, X_n)$ with an associated graph G of maximum degree d. Suppose each $X_i \in \{0, 1\}$ and the probability distribution has the following form:

$$P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n) \propto I_G(x_1, x_2, \ldots, x_n)\lambda^{\sum_i x_i}.$$

Here $I_G(x_1, x_2, \ldots, x_n) = 1$ if the variables that take on value 1 among $x_1, x_2, \ldots, x_n$, form an independent set in G[1]. Otherwise, $I_G(x_1, x_2, \ldots, x_n) = 0$. Given m samples generated from such an MRF design an efficient algorithm to learn the structure of the graph G. Your algorithm should run in time polynomial in n, m. How much data would you need to guarantee that, with high probability, the graph that your algorithm outputs is the correct underlying graph? You can assume that $\lambda > 0$ is known.

4 Algorithms That Know When They Don't Know (20 pts)

In many prediction scenarios the cost of making a mistake is huge. For instance, incorrectly predicting that a person does not have cancer. In such scenarios, it is more desirable if the algorithm just abstained and said "I don't know". Consider a binary prediction problem where examples are arriving in a stream, one after the other. The algorithm sees one example at a time and has to make

[1] An independent set in G is any subset of nodes that have no edges among themselves.

a prediction on the label instantly before seeing the next example. The algorithm can either make a prediction or say "I don't know". However, if the algorithm makes a prediction, it has to be absolutely sure that the prediction is correct.

- Suppose you are promised that the target function lies in a finite class of functions H. Design an algorithm in the above setting that says I don't know as few times as possible.
- Suppose the algorithm is allowed to make 1 mistake. Can you reduce the number of times the algorithm abstains from making a prediction?
- What if the algorithm is allowed to make up to k mistakes?

[Note: It's ok if your algorithms are not computationally efficient.]

5 Parameter Learning in Graphical Models (20 pts)

Consider an MRF $P(Z, Y)$ where Z is a multinomial taking values in $\{1, 2, \ldots, n\}$ and $P(Y|Z = z) \sim N(\mu_z, 1)$. This is known as the *Gaussian Mixture Model*. The data generation process is as follows: a value of Z is chosen from the multinomial distribution and conditioned on that, Y is drawn from the corresponding Gaussian. You can assume that $P(Z = z) = \frac{1}{n}, \forall z$. A fundamental problem here is the following: Given $Y_1, Y_2, \ldots, Y_m$ drawn from the MRF and $\epsilon > 0$, estimate the parameters $\mu_1, \mu_2, \ldots, \mu_n$, each up to error ϵ. Notice that Z_i values for the corresponding Y_i's are not observed. Such variables are known as *hidden* variables.

- Write down the log-likelihood of the data as a function of $\mu_1, \mu_2, \ldots, \mu_n$. Can the function be efficiently maximized to compute the parameter estimates of the μ_i's?
- Consider the following algorithm, known as *Expectation-Maximization*, for estimating the parameters
  1. Start with arbitrary guesses $\mu_{1,0}, \mu_{2,0}, \ldots, \mu_{n,0}$ for the μ_i's.
  2. For $t = 1, 2, ...$
     - Use the current guesses $\mu_{1,t-1}, \mu_{2,t-1}, \ldots, \mu_{n,t-1}$, to estimate the posterior distribution $P(Z_1, Z_2, \ldots, Z_m|Y_1, Y_2, \ldots Y_m)$ of the hidden variables.
     - Update the new guess to

       $$\mu_{1,t}, \mu_{2,t}, \ldots, \mu_{n,t} = argmax\, E[P(Y_1, Y_2, \ldots Y_m|Z_1, Z_2, \ldots Z_m)].$$

       Here the expectation is with respect to the posterior distribution of the hidden variables computed in the previous step.

  The algorithm stops when there is no change in the values of μ_i's.
- Write down explicit expressions for the update of the parameter values.
- Show that the above algorithm will stop eventually.
- Construct a scenario where the above algorithm will produce bad estimates of the parameters. Feel free to run experiments to help you construct bad scenarios.
- Describe intuitively the kind of scenarios where the above algorithm will produce accurate estimates of the parameters. Feel free to include experimental results.