6.891 Machine learning andd neural networks

Problem set 4

Deadline: December 5, in class

Note: You will need to use MATLAB in this problem set. Information about Matlab can be found on the course web site[1]. Data files for the problems will be made available via the course Athena locker: `/mit/6.891`.

Reading: Lecture notes 1-20. A handout paper on hidden Markov models.

Some of the problems have been marked as optional. This means that we want you to submit solutions to only one of the optional problems. We encourage everyone to at least read through all the questions, You are expected to understand the solutions to these problems even if you do not have the background to solve them on your own at this point.

Please try to be concise and clear in your explanations.

Problem 1: option I (theoretical)

Here you are to show that the EM-algorithm increases the likelihood of the training examples monotonically after each iteration.

We have a mixture m Gaussians where we use y and j to denote specific mixture components and i indexes training examples. θ contains all the adjustable parameters in the mixture m Gaussians model.

For example, the log-likelihood of the observed data is given by

$$\begin{aligned} J(\theta) &= \sum_{i=1}^{n} \log P(\mathbf{x}_i|\theta) = \sum_{i=1}^{n} \log \left[\sum_{j=1}^{m} P(\mathbf{x}_i, y = j|\theta) \right] && (1) \\ &= \sum_{i=1}^{n} \log \left[\sum_{j=1}^{m} P(y = j|\theta) P(\mathbf{x}_i|y = j, \theta) \right] && (2) \\ &= \sum_{i=1}^{n} \log \left[\sum_{j=1}^{m} p_j \, P(\mathbf{x}_i|\mu_j, \Sigma_j) \right] && (3) \end{aligned}$$

[1] http://www.ai.mit.edu/courses/6.891

where $\theta = \{p_1, \ldots, p_m, \mu_1, \ldots, \mu_m, \Sigma_1, \ldots, \Sigma_m\}$ and p_j is the prior probability of selecting component j.

a) Let $\theta = \theta_0$ be the current setting of all the parameters in our mixture model. Show that performing the M-step of the EM-algorithm corresponds to maximizing

$$J(\theta|\theta_0) = \sum_{i=1}^{n} E_{\theta_0} \{ \log P(\mathbf{x}_i, y|\theta) \,|\, \mathbf{x}_i \} \tag{4}$$

$$= \sum_{i=1}^{n} \left[\sum_{j=1}^{m} P(y = j|\mathbf{x}_i, \theta_0) \, \log P(\mathbf{x}_i, y = j|\theta) \right] \tag{5}$$

with respect to θ. Here the expectation is taken with respect to $P(y|\mathbf{x}_i, \theta_0)$, the posterior probability over the unobserved mixture index y. We ask that you show the equivalence only for the mixture coefficients, $\{p_1, \ldots, p_m\}$. In other words, showing that maximizing $J(\theta|\theta_0)$ yields the EM mixture coefficient updates is enough for this part.

b) Before we move on we need an intermediate result. Show that for any distribution q_j, $j = 1, \ldots, m$, $\sum_j q_j = 1$ and positive coefficients c_j

$$\sum_{j=1}^{m} q_j \log \frac{c_j}{q_j} \leq \log \sum_{j=1}^{m} c_j \tag{6}$$

(*hint: you can show this by relying on the properties of KL-divergence or that log*(z) *is a concave function of* z). If this is hard, skip to the next problem and just use the result.

c) Use a) and b) to show that

$$J(\theta) - J(\theta_0) \geq J(\theta|\theta_0) - J(\theta_0|\theta_0) \tag{7}$$

d) Provide the summary argument for why part c) shows that EM increases the log-likelihood of the training examples monotonically.

Problem 1, option II (more practical)

Naive Bayes model is a very simple probability model that makes the tell-tale independence asumption about the components of the vector $\mathbf{x}$ given the mixture component. In other words,

$$P(\mathbf{x}, y) = P(y)P(\mathbf{x}|y) =^{*} P(y) \left[\prod_{j=1}^{d} P(x_j|y) \right] \tag{8}$$

where $y = 1, \ldots, m$ indexes the mixture components, d is the dimensionality of $\mathbf{x}$, and x_j is the j^{th} component of $\mathbf{x}$. We assume for simplicity that each conditional distribution $P(x_j|y)$ is a Gaussian with mean $\mu_{j|y}$ and variance $\sigma^2_{j|y}$.

a) We have given you a mixture of Gaussians code. Modify this code for the naive Bayes mixture model. You can simply modify the M-step of the estimation algorithm (this won't lead to the most efficient implementation of the naive Bayes model but suffices for our purposes). The relevant files are

   "inimix.m" initializes the mixture of Gaussians model.

   "runmix.m", the overall script that initializes the mixture model and runs the em iterations.

   "em.m" performs one EM-iteration, returning the new parameters.

   "evalgauss.m" evaluates the log-likelihood that a single Gaussian assigns to an observation vector

   "evalmix.m" computes the log-likelihood of the data for a mixture of Gaussians distribution

   "softmax.m" computes the softmax distribution of the input.

   Return only the modified part of your MATLAB code.

b) Even with this simplification or additional constraint, we can easily run into problems when trying to estimate these models on the basis of only a few data points. For this reason, we regularize the variance parameter $\sigma^2_{j|y}$ associated with each of the component Gaussian distributions $P(x_j|y)$. Our regularization simply constrains the variance $\sigma^2_{j|y}$ to be larger than a prespecified limit c (for example, $c = 0.1$). Modify your MATLAB code to incorporate this regularization.

c) Apply your code to the digit recognition problem using 1 and 2 components for each type of digit. You will run EM four times: twice on the 3's (once with one component, once with two components) and twice on the 5's.

d) Classify the test examples ("digit_x_test.dat") based on which mixture model assigns a higher likelihood to the test example. Evaluate the number of misclassified digits for the two mixture models. Briefly discuss the results and how the value of the regularization parameter c might affect the results.

Problem 2: Markov and hidden Markov Models

We have provided you with a brief dataset "X.dat" that contains equal length (25) sequences where the observations take two possible values (1 and 2). The sequences look like:

```
1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 2 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2
...
```

Part 1: Markov models

a) Estimate a (first order homogeneous) Markov model over these sequences. In other words, estimate the initial state distribution and the transition probability matrix for a two-state Markov chain model. You can use the "count.m" function that we have provided. Alternatively, if you like, you can modify and use the hmm code provided primarily for the second part of this problem.

b) Define a good criterion for an "outlier". Justify why this measure is appropriate in the context of models such as Markov chains and identify a few "outliers" in the dataset "X.dat".

c) On the basis your observations and by looking at the dataset, determine whether a Markov chain is an appropriate model. Justify your answer.

Part 2: hidden Markov models

We have already written the MATLAB functions you need for this part: "alpha.m", "beta.m", "esthmm.m", "evalhmm.m", "newhmm.m", "poststate.m", "viterbi.m".

a) The data in "X.dat" may be better modeled with a Hidden Markov model. Cite at least one example in the data for why this might be true. In using an HMM model, what would be a good choice for the number of hidden states? Justify your answer

b) Estimate a HMM using the MATLAB code provided. The function "esthmm.m" also returns the log-probability of data after each iteration. Plot these likelihoods as a function of the number of iterations and explain the plot. For example, the log-probability increases monotonically. Anything else worth explaining?

c) Study the resulting transition probability matrix. Can you hypothesize what the true generating distribution might be? (it is a HMM...)

d) Load the test sequence "Xtest.dat" and evaluate the posterior state probabilities for this sequence (see function "poststate.m"). Also generate the most likely hidden state sequence given this observation sequence using "viterbi.m". Compare the two results. Describe the differences and explain why they arise.

Problem 3: Graph models

In figure 1 we have two diseases d_1 and d_2 as well as two potential findings f_1 and f_2. The diseases and findings are assumed to be binary. We assume further that the conditional probabilities of $P(f_1|d_1, d_2)$ and $P(f_2|d_2)$ are noisy-OR models as described in the lectures.

a) Now, suppose we know that the outcome of finding f_1 was positive ($f_1 = 1$). What happens to the disease probabilities? Justify your answer.

b) Suppose later on we learn that finding f_2 was positive as well. How will the probability of disease d_1 being present change as a result? Justify your answer.

c) Construct a junction tree for the graph model in figure 1. Also initialiaze the potential functions for the junction tree. Are there many ways of initializing the potentials?

d) Roughly speaking, how many operations do we need to perform to complete both "collect" and "distribute" steps? You can assume here that we will blindly apply both propagation steps whatever the evidence might be.

e) Suppose we would like to determine which finding to query (which of the corresponding tests to carry out). For this we need to evaluate the mutual information between the disease configurations and the possible values of each of the findings. In other words, we have to compute $I(f_1; d_1, d_2)$ and $I(f_2; d_1, d_2)$, where, for example,

$$I(f_1; d_1, d_2) = \sum_{d_1, d_2, f_1 = 0,1} P(d_1, d_2, f_1) \log \frac{P(d_1, d_2, f_1)}{P(d_1, d_2)P(f_1)} \tag{9}$$

Now, show that the graph structure (original or the junction tree) implies that $I(f_2; d_1, d_2) = I(f_2; d_2)$.

f) Based on your results for c) and e) show that the marginal probabilities that we compute in the junction tree suffice for evaluating which test we should query next ("active learning").

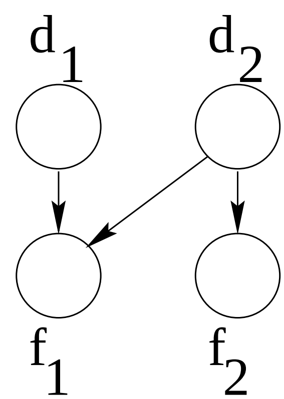

Figure 1: Medical diagnosis example.