Machine Learning *CS 4364/6364, Fall 2016*

HW2

Instructions

Collaboration: You are allowed to discuss the problems with other students. However, you must write up your own solutions for the math questions, and implement your own solutions for the programming problems. Please list all collaborators and sources consulted at the top of your homework.

Submission: Please submit homework pdf's at this address: machine.learning.gwu 'at' gmail.com. Electronic submissions are preferred. Math can be formatted in Latex (some easy editors for Latex are Lyx, TexShop), Word, or other editors. Math solutions can be optionally submitted on paper at the Department of Computer Science (in the dropbox to the right of the entrance to SEH 4000), but all code should be submitted electronically. The assignment is due Friday, October 14, at 5 pm.

Questions

1. **Probability Review - 5 pts**

   Consider a single Boolean random variable Y (the "classification"). Let the prior probability $P(Y = true)$ be π. Let's try to find π, given a training set $D = (y_1, \ldots, y_N)$ with N independent samples of Y. Furthermore, suppose p of the N samples are positive(i.e., labeled as true) and n of the N samples are negative.

   1. Write down an expression for the likelihood of D (that is, the probability of seeing this particular sequence of examples, given a fixed value of π) in terms of π, p and n.
   2. By differentiating the log likelihood L, find the value of π that maximizes the likelihood.
   3. Now suppose we add k Boolean variables $X_1, X_2, \ldots, X_k$ (the "attributes") that describe each sample, and suppose we assume the attributes are conditionally independent of each other given Y. Draw the Bayes net corresponding to this assumption.
   4. Write down the likelihood for the data including the attributes, using the following additional notation:
      - α_i is $P(X_i = true|Y = true)$.
      - β_i is $P(X_i = true|Y = false)$
      - p_i^+ is the count of samples for which $X_i = true$ and $Y = true$
      - n_i^+ is the count of samples for which $X_i = false$ and $Y = true$
      - p_i^- is the count of samples for which $X_i = true$ and $Y = false$

- n_i^- is the count of samples for which $X_i = false$ and $Y = false$ [Hint: consider first the probability of seeing a single example with specified values for $X_1, \ldots, X_k$ and Y]

5. By differentiating the log likelihood L, find the values of α_i and β_i (in terms of the various counts) that maximize the likelihood.

2. **Boosting - 5 pts**

NOTE: *This problem is required for graduate students only. Undergraduate students can solve it for extra credit.*

In this problem, we slightly modify the AdaBoost algorithm to better explore some properties of the algorithm. Specifically, we no longer normalize the weights on the training examples after each iteration. The modified algorithm, which is set to run for T iterations, is shown in Algorithm 1.

Note that in the modified version, the weights associated with the training examples are no longer guaranteed to sum to one after each iteration (and therefore can not be viewed as a "distribution"), but the algorithm is still valid. Let us denote the sum of weights at the start of iteration t by $Z_t = \sum_{i=1}^{n} w_i^{(t)}$. At the start of the first iteration of boosting, $Z_1 = n$. Let us now investigate the behavior of Z_t, as a function of t.

(a) At the t^{th} iteration, we found a weak classifier that achieves a weighted training error ϵ_t. Show that the choice, $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$ is optimal in the sense that it minimizes Z_{t+1}.
Hint : Look at Z_{t+1} as a function of α and find the value of α for which the function achieves its minimum. You may also find the following notational shorthand useful:

$$W_l = \sum_{i=1}^{n} w_i^{(t)} (1 - \delta(y_i, h_t(x_i)))$$

$$W_c = \sum_{i=1}^{n} w_i^{(t)} (\delta(y_i, h_t(x_i)))$$

where, W_c is the total weight of the points classified correctly by h_t and W_l is the total weight of the misclassified points. $\delta(y, h_t(x)) = 1$ whenever the label predicted by h_t is correct and zero otherwise. The weights here are those available at the start of iteration t.

(b) Show that the sum of weights Z_t is monotonically decreasing as a function of t.
[*An interesting side note :* Why is Z_t so interesting? It is useful in bounding how successive boosting iterations reduce the training error. Specifically, it can be shown that the average number of misclassified training examples of the combined classifier after m iterations of boosting is bounded from above by Z_{m+1}/n. You are not required to prove this here.]

Algorithm 1 AdaBoost: slightly modified version (the weights are not normalized to sum to one)

Input: Set of n labeled examples $(x_1, y_1), ..., (x_n, y_n), y_i = \pm 1$
Set of associate weights $w_1^{(1)}, ..., w_n^{(1)}$, initially all $w_i^{(1)} = 1$
Required number of iterations T.
Procedure:
for $t = 1..T$ **do**
Find a weak classifier h_t, which outputs binary predictions $h_t(x) = \pm 1$, such that its weighted training error

$$\epsilon_t = \frac{1}{Z_t} \sum_{i=1}^{n} w_i^{(t)} (1 - \delta(h_t(x_i), y_i))$$

satisfies, $\epsilon_t < 1/2$
Set the vote $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
Update the weights :
for $i = 1...n$ **do**
$w_i^{(t+1)} = w_i^{(t)} \exp(-y_i \alpha_t h_t(x_i))$
end for
end for
Output: The combined classifier defined by

$$\hat{h}_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

3. **Classification with "Naive Bayes" generative model - MATLAB Programming - 10 pts**

   You are given the `SPECTtrain` and `SPECTtest` data and labels files. These were created from the medical data on cardiac Single Proton Emission Tomography (SPECT) images of patients and each patient is classified into two categories: normal or abnormal. The database of 267 SPECT image sets (patients) was processed to extract features that summarize the original SPECT images. As a result, you are given a training set of 187 patterns and a test set of 80 patterns, each with 22 binary feature. The goal is to build a generative model of each group (normal and abnormal), and to use these models to classify future patients.

   (a) For each of the groups, use the training data to build a simple probabilistic model, assuming that the different features are independent. The model for a group should have 22 parameters $p_i \in [0, 1]$; the probability of a particular data point $x \in \{0, 1\}^{22}$ is then

   $$\prod_{i=1}^{22} p_i^{x_i}(1 - p_i)^{1-x_i}$$

   A natural choice is to set p_i to the proportion of training documents (from that particular group) for which $x_i = 1$. In practice, this can be dangerous - when these are lots of features, and any given feature is 1 only a tiny fraction of the time, there often isn't enough data to reliably estimate all the p_i in this way. Therefore, it is common to smooth the estimates somewhat, by setting:

   $$p_i = \frac{(\text{number of points with } x_i = 1) + n\tilde{p}}{(\text{number of points}) + n} \tag{1}$$

   where n is a small integer and $\tilde{p}$ is a prior estimate of the value of p_i. To keep this simple, use $n = 2$ and $\tilde{p} = 0.5$.

   (b) Now implement the `Naive Bayes` model to classify the test documents. Recall that `Naive Bayes` assumes that the conditional probability of the features are independent given the label variable.

   (c) Find the error rate of your Naive Bayes algorithm on the test set.

4. **Learning mixtures of Gaussians with EM algorithm - MATLAB Programming - 10 pts**

   In this problem, you'll implement the Expectation Maximization(EM) algorithm you learned in class. Follow the guidelines below to build your program

   (a) Your function should be in the form $[params, loglikes] = EM(X, k)$, where `params` contains the estimated parameters: **mixing proportions, the means, and covariances of each of the component Gaussians**. `loglikes` stores the expected log-likelihoods of the mixture models you produce in the course of the EM iterations (so it'll be a vector that keeps track of the expected log-likelihood after each iteration).

   (b) The algorithm is provided below. Let ω_j , μ_j denote the mixing proportion and mean of each Gaussian component $y = j$. We assume that the Gaussians are spherical, i.e. their covariance matrices can be represented as $\sigma_j I$, where σ_j is a scalar and I is the identity matrix. Let x_i denote a data point, let k denote the number of Gaussian components, let

Algorithm 2 EM algorithm

initialize ω_j,μ_j,σ_j^2 // various ways to do this; see (e)(i).
while number of iterations $<$ max **do**
  For each data point x_i, for each component j:
    update $P(y = j|x_i)$, using Bayes rule `E Step`
  where:
    $P(x_i|y = j) = \mathcal{N}(x_i; \mu_j, \sigma_j^2 I)$
    $P(y = j) = \omega_j$
    $P(x_i) = \Sigma_{j=1}^{k} P(y = j)P(x_i|y = j)$

  Let $n_j = \Sigma_{i=1}^{N} P(y = j|x_i)$
  for $j = 1 : k$ **do**
    $\mu_j \leftarrow \Sigma_{i=1}^{N} \frac{P(y=j|x_i)x_i}{n_j}$ `M Step`
    $\sigma_j^2 \leftarrow \Sigma_{i=1}^{N} \frac{P(y=j|x_i)||x_i-\mu_j||^2}{dn_j}$
    $\omega_j \leftarrow \frac{n_j}{N}$
  end for
  `loglikes[iteration]` = log-likelihood of whole training data set, with respect to the current model
end while
return `params, loglikes`

N denote the number of data points, and let n_j denote the effective number of data points currently assigned to component j.

(c) To compute $P(\boldsymbol{x}|y = j)$, you should use the spherical Gaussian model

$$P(\boldsymbol{x}|y = j) = N(\boldsymbol{x}; \boldsymbol{\mu_j}, \sigma_j^2 I) = \frac{1}{(2\pi\sigma_j^2)^{\frac{d}{2}}} e^{-\frac{1}{2\sigma_j^2}||\boldsymbol{x}-\boldsymbol{\mu_j}||^2}$$

where d is the dimension of each data point.

(d) Let $D = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ denote the data set, and let θ denote all the current model parameters. The expected log-likelihood of the mixture model is:

$$\begin{aligned} L(D; \theta) &= \sum_{i=1}^{N} \sum_{j=1}^{k} P(y = j|x_i; \theta) \log P(x_i, j; \theta) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{k} P(y = j|x_i; \theta) \left[\log P(x_i|j; \theta) + \log P(j; \theta)\right] \end{aligned}$$

The first term inside the summations was computed at the beginning of the iteration. To compute the first term inside the square brackets, use the Gaussian formula above, with the current values of μ_j, σ_j^2. The second term inside the square brackets is simply $\log(\omega_j)$.

(e) Initialization of the EM algorithm

EM is sensitive to the initialization of the parameters. Now you are going to explore the performance of your EM algorithm with different initializations using the `IRIS` data.

(i) Initialize your parameters μ_j,σ_j^2, and ω_j: randomly select different k data points from your training data to be your k means; let all of your covariance matrices be identity matrices; initialize the mixing proportion to be $\frac{1}{k}$ for each of the Gaussians.

(ii) Use the `IRIS` data set, `trainData`, which have 150 randomized examples. Run your EM algorithm with k = 2 and 3 on the training data for number of iterations = 10,100, and 1000, respectively. Plot the expected log-likelihood graph for each of these settings. What's the trend of the expected log-likelihood in each of these settings? (Is it increasing, decreasing, or varying?) What are the maximum expected log-likelihood you get in each of those settings? Does the maximum expected log-likelihood improve significantly when you increase the number of iterations?

(iii) What would happen if you initialize your parameters in the following way:

- Using k=2 and 3, set all the means to vector $\mathbf{1}$. Initialize the remaining parameters as described in (i)
- Using $k = 3$, set two of the means to $\mathbf{1}$ and select a random datapoint as a mean for the remaining Gaussian.

(Compare the above, and briefly describe what happens in each case)

(iv) The `IRIS` data you used is already randomized for you, i.e., the data points come in a random order. What would happen if the data come in their group order? Sort the data according to their train labels (`trainLabels`) and feed the sorted `trainData` to your EM algorithm. Describe what happens compared to the previous results. (For example, do the means and covariances change? Do the loglikehoods change?)