Homework 2, due Wednesday February 25 COMS 4771 Spring 2015

Problem 1 (Bernoulli Naïve Bayes classifier)**.** Download the "20 Newsgroups data set" from Courseworks `news.mat`, and load it into MATLAB. The training feature vectors/labels and test feature vectors/labels are stored as `data`/`labels` and `testdata`/`testlabels`. Each data point corresponds to a message posted to one of 20 different newsgroups (i.e., message boards). The representation of a message is a (sparse) binary vector in $\{0,1\}^d$ for $d = 61188$) that indicates the words that are present in the message. If the i-th entry in the vector is 1, it means the message contains the word that is given on the i-th line of the text file `news.vocab`. The class labels are $\mathcal{Y} = \{1, 2, \ldots, 20\}$, where the mapping from classes to newsgroups is in the file `news.groups`.

In this problem, you'll develop another plug-in classifier based on a generative model. Here, we use class conditional distributions of the form

$$\Pr[\boldsymbol{X} = \boldsymbol{x} \,|\, Y = k] = \prod_{j=1}^{d} \Pr[X_i = x_i \,|\, Y = k] = \prod_{j=1}^{d} p_{j,k}^{x_j}(1 - p_{j,k})^{1-x_j}.$$

In other words, in this generative model (which we call the Bernoulli Naïve Bayes model, or the BNB model), the $X_1, X_2, \ldots, X_d$ are conditionally independent given Y. The parameter $p_{j,k} \in [0,1]$ for $j \in \{1, 2, \ldots, d\}$ and $k \in \mathcal{Y}$ is the probability that $X_i = 1$ given $Y = k$.

(a) Derive a formula for the MLE of $p_{j,k}$ based on training data S; show that the gradient of the likelihood function is zero at the MLE. (Assume the class priors don't depend on the $p_{j,k}$.)

(b) Suppose instead of 20 newsgroups, there were only 2 different newsgroups (say, 1 and 2). Explain why, in this case, a plug-in classifier based on the BNB model is a linear classifier.

(c) It turns out the MLE is not a good estimator for the class conditional parameters, in particular if the estimate turns out to be zero or one. An alternative to the MLE is the following estimator based on a technique called *Laplace smoothing*:

$$\hat{p}_{j,k} := \frac{1 + \sum_{(\boldsymbol{x},y)\in S} \mathbb{1}\{y = k\} x_j}{2 + \sum_{(\boldsymbol{x},y)\in S} \mathbb{1}\{y = k\}}$$

(where S is the training data). This ensures that $\hat{p}_{j,k} \in (0,1)$—never zero nor one.

Write a MATLAB function that takes as input a (sparse) matrix of training feature vectors `X` and a vector of labels `Y` (as `data` and `labels` above), and returns the parameters `params` of the plug-in classifier based on the BNB model. (Naturally, you should not use or look at the implementation already provided in MATLAB.)

```
function params = hw2_train_bnb(X,Y)
```

Use the MLE for estimating the class priors, and the Laplace smoothing estimator for the class conditional distribution parameters $p_{j,k}$. Also write a MATLAB function that takes as input the parameters of the above plug-in classifier `params`, and a matrix of test feature vectors `test`. The function should output a vector of predictions `preds` for all test feature vectors.

```
function preds = hw2_test_bnb(params,test)
```

Train and evaluate a BNB plug-in classifier using the data from `news.mat`. Report the training error and test error in your write-up.

Problem 2 (Perceptron)**.** The data sets `news2.mat` and `news3.mat` (on Courseworks) are for binary classification problems derived from the "20 Newsgroups data set". Specifically, each combines three newsgroups to comprise the "negative" class (with label -1), and three other newsgroups to comprise the "positive" class (with label $+1$). (The first data set compares "religious" topics to "political" topics, the second data set compares "computing" topics to "scientific" topics.) In this problem, you'll evaluate variants of the Perceptron algorithm and the BNB plug-in classifier on these data sets.

Implement the Online-Perceptron and Averaged-Perceptron algorithms as MATLAB functions: each takes as input a (sparse) matrix of training feature vectors `X`, a vector of labels `Y`, and a positive integer `num_passes` (the number of passes to go through the training data), and returns a linear classifier. You should also write a function the evaluates the linear classifier on test feature vectors; the same function should work for both the linear classifier returned by Online-Perceptron and that returned by Averaged-Perceptron. (Naturally, don't use or look at any existing implementations.)

```
function params = hw2_train_perc(X,Y,num_passes)
function params = hw2_train_avgperc(X,Y,num_passes)
function preds = hw2_test_perc(params,test)
```

(a) The feature vectors *have not* been extended to include an extra "offset" dimension. So your implementation of {Online,Averaged}-Perceptron should explicitly update both a weight vector $\boldsymbol{w} \in \mathbb{R}^d$ ($d = 61188$) and a threshold $\theta \in \mathbb{R}$ so that the linear classifier is $f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta)$. (It should be equivalent to having added an extra feature in $\boldsymbol{x}$ that always has value -1.) Write out the Online-Perceptron update on example $(\boldsymbol{x}, y)$:

if _____ **then**
$\boldsymbol{w}$:= _____; θ := _____.
end if

(b) Produce learning curves for each algorithm, using sample sizes $n \in \{500, 1000, 1500, N\}$, where $N = 3028$ for `news2.mat`, and $N = 3523$ for `news3.mat`. That is, first draw n random points from `data`, together with their corresponding labels. Then use these n points as the training data with *all* of the following learning algorithms: BNB, Online-Perceptron with 1 pass, Online-Perceptron with 5 passes, Averaged-Perceptron with 1 pass, Averaged-Perceptron with 5 passes; and compute the resulting test errors. Repeat this whole process independently five times, and plot the learning curves with error bars. (An optional MATLAB script that does much of this is provided as `hw2_runall.m` on Courseworks.)

You should produce two separate plots: one for `news2.mat`, and another for `news3.mat`. Make sure everything (e.g., axes, plot lines) is clearly labeled.

What can you conclude about BNB vs. Perceptron? About the (relative) importance of averaging vs. multiple passes?

Save the classifiers from the final run of each algorithm (with $n = N$) for part (c).

(c) For a linear classifier $f_{\boldsymbol{w},\theta}$, the *absolute values* of the weights in $\boldsymbol{w}$ (associated with the vocabulary words) can be interpreted as a measure of "importance" of the words. For the BNB plug-in classifier, you'll have to interpret your model parameters appropriately so that the classifier can be thought of as a linear classifier (Problem 1(b)). In each problem (`news2.mat` and `news3.mat`), what are the 20 most important words for the BNB classifier? And for the Averaged-Perceptron (five passes) classifier? Also give the actual weight values when reporting these words. A MATLAB function `hw2_topwords.m` that does this computation is provided on Courseworks. (Unfortunately, 20 is a rather arbitrary cut-off point.)

Problem 3 (Features)**.** It is common to pre-preprocess the feature vectors before passing them to a learning algorithm. Two simple ways to pre-process the data are:

- **Centering**: Subtract the mean $\boldsymbol{\mu} := \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S}\boldsymbol{x}$ (of the training data) from every feature vector:
$$\boldsymbol{x} \mapsto \boldsymbol{x} - \boldsymbol{\mu}.$$
- **Standardization**: Perform centering, and then divide every feature by the per-feature standard deviation $\sigma_i = \sqrt{\frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S}(x_i - \mu_i)^2}$:
$$(x_1, x_2, \ldots, x_d) \mapsto \left(\frac{x_1-\mu_1}{\sigma_1}, \frac{x_2-\mu_2}{\sigma_2}, \ldots, \frac{x_d-\mu_d}{\sigma_d}\right).$$

(The same transformations should be applied to *all* feature vectors you encounter, including any future test points.)

For each of the following learning algorithms, and each of the above pre-processing transformations, explain whether or not each of the transformation can affect the resulting learned classifier.

(a) The plug-in classifier where class conditional distributions are multivariate Gaussian distributions with a fixed covariance equal to the identity matrix $\boldsymbol{I}$.

(b) The 1-NN classifier using Euclidean distance.

(c) The greedy decision tree learning algorithm with axis-aligned splits. (For concreteness, assume Gini index is used as the uncertainty measure, and the algorithm stops after 20 leaf nodes.)

(d) Online Perceptron, where you should assume that an update is always made on the first training example $(\boldsymbol{x}, y)$, whether or not a mistake was made by the initial linear classifier. So, the initial linear classifier has $\boldsymbol{w} = 0$ and $\theta = 0$, and after this update, the linear classifier has $\boldsymbol{w} = y\boldsymbol{x}$ and $\theta = -y$. (For concreteness, assume the final linear classifier is returned.)

(e) Empirical Risk Minimization: the (intractable) algorithm that finds the linear classifier (both the weight vector and threshold) that has the smallest training error.

You may assume the per-feature standard deviations are never zero, and you may also ignore computational and numerical precision issues.

Problem 4 (Kernels)**.**

(a) Let $K\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be given by
$$K(\boldsymbol{x}, \boldsymbol{x}') := \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2.$$
Show that K is a positive definite kernel function by constructing an explicit feature map $\boldsymbol{\phi}\colon \mathbb{R}^d \to \mathbb{R}^D$ (for some $D \in \mathbb{N}$) such that $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle$.

(b) Let $K_1\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and $K_2\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be positive definite kernel functions. Explain why $K_3\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, given by
$$K_3(\boldsymbol{x}, \boldsymbol{x}') := K_1(\boldsymbol{x}, \boldsymbol{x}') \cdot K_2(\boldsymbol{x}, \boldsymbol{x}'),$$
is also a positive definite kernel function. You may assume that the RKHSs corresponding to K_1 and K_2 are both subspaces of $\mathbb{R}^D$ for some $D \in \mathbb{N}$.