Homework 1, due Monday February 9

COMS 4771 Spring 2015

Problem 1 (Classifiers via generative models)**.** Download the OCR image data set `mnist.mat` from Courseworks, and load it into MATLAB. The unlabeled training data (i.e., feature vectors) are contained in a matrix called `data` (one point per row), and the corresponding labels are in a vector called `labels`. The test feature vectors and labels are in, respectively, `testdata` and `testlabels`. To view the i-th image in the training data, use the following command:

```
imagesc(reshape(data(i,:),28,28)');
```

If the colors are too jarring for you, try the following:

```
colormap(1-gray);
```

1. Write a MATLAB function that takes as input a 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 multivariate Gaussian class conditional densities.

   ```
   function params = hw1_train1a(X,Y)
   ```

   You should use the MLE for estimating the class priors, as well as the means and covariances for each class conditional density. You can collect all of these parameters in a MATLAB struct (or anything else that works). Don't bother optimizing this; just get something that works correctly.

   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 = hw1_test1a(params,test)
   ```

   Before applying `hw_train` and `hw_test` to real data, it's often a good idea to test out your code on an easy problem where you know what to expect. Download the data set `hw1_data.mat`, also available on Courseworks. This data are drawn from a distribution over $\mathbb{R}^2 \times \{0, 1, 2\}$ where the class prior is uniform, and the class conditional densities are bivariate Gaussians $\mathcal{N}((0, 2), \boldsymbol{I})$, $\mathcal{N}((0, 0), \boldsymbol{I})$, $\mathcal{N}((2, 0), 0.25\boldsymbol{I})$; the MATLAB function used to produce these data is also on Courseworks. Apply `hw1_train1a` to the training data (`data` and `labels`) to get a classifer, and evaluate it on the test data (`testdata` and `testlabels`) using `hw1_test1a`.

   In your write-up, just report the training error and the test error from above.

2. Now apply `hw1_train1a` to the OCR training data, and evaluate the resulting classifer on the test data using `hw1_test1a`. Something bad should happen, even if your code is correct. Try to explain what goes wrong.

3. Create new versions of `hw1_train1a` and `hw1_test1a` so that now the class conditional distributions are multivariate Gaussians with a *fixed covariance matrix that is always equal to the identity matrix*. (Call the new functions `hw1_train1b` and `hw1_test1b`.) Use these new functions to train a classifer and evaluate it on the test data. In your write-up, report the training error and the test error, and explain why you don't encounter the same problem encountered earlier.

Problem 2 (Nearest neighbors)**.** Write a MATLAB function that implements the 1-nearest neighbor classifier. Your function should take as input a matrix of training feature vectors `X` and a vector of labels `Y` (just as in Problem 1), as well as a matrix of test feature vectors `test`. The output should be a vector of predicted labels `preds` for all the test points.

```
function preds = hw1_nn(X,Y,test)
```

Load the OCR image data set from Problem 1. Instead of using `hw1_nn` directly with `data` and `labels` as the training data, do the following. For each value $n \in \{1000, 2000, 4000, 8000\}$,

- Draw n random points from `data`, together with their corresponding labels.
- Use these n points as the training data with `hw1_nn`, with `testdata` as the test points, and compute the resulting test error.

A plot of the test error (on the y-axis) as a function of n (on the x-axis) is called a *learning curve*.

Since the above process involves some randomness, you should make repeat it independently several times (say, at least ten times). Produce a learning curve plot using the average of these test errors (that is, averaging over the repeated trials). Add error bars to your plot that extend to one standard deviation above and below the mean.

Problem 3 (Classification with different costs)**.** Suppose you face a binary classification problem with input space $\mathcal{X} = \mathbb{R}$ and output space $\mathcal{Y} = \{0, 1\}$, where it is c times as bad to commit a "false positive" as it is to commit a "false negative" (for some positive number $c > 0$). To make this concrete, let's say that if your classifier predicts 1 but the correct label is 0, you incur a penalty of $\$c$; whereas if your classifier predicts 0 but the correct label is 1, you incur a penalty of \$1. (And you incur no penalty if your classifier predicts the correct label.)

Assume the distribution you care about has a class prior with $\Pr(Y = 0) = 2/3$ and $\Pr(Y = 1) = 1/3$, and the class conditional densities are $\mathcal{N}(0, 1)$ for class 0, and $\mathcal{N}(1, 1/4)$ for class 1. What is the classifier $f^\star \colon \mathbb{R} \to \{0, 1\}$ that has smallest expected penalty? Your answer should be given in terms of c.