CSCI 1950-F Homework 3: Handwritten Digit Classification

Brown University, Spring 2011

Homework due at 12:00pm on February 24, 2011

In this problem set, we consider the problem of handwritten digit recognition. We will use a subset of the MNIST database, which has become a benchmark for testing a wide range of classification algorithms. See `http://yann.lecun.com/exdb/mnist/` if you'd like to read more about it. The particular version of the data which we'll use is available here:

`/course/cs195f/asgn/knn_cv/mnist_all.mat`

In addition, some useful Matlab scripts are posted here:

`/course/cs195f/asgn/knn_cv/handout`

This code should be used to load and define matrices of training and test data for the various problems below.

In the MNIST database, each training or test example is a 28-by-28 grayscale image. To ease programming of learning algorithms, these images have been converted to vectors of length $28^2 = 784$ by sorting the pixels in raster scan (row-by-row) order. The Matlab `reshape` command can be used to convert these vectors back to images for visualization. For example, we can plot the third training example of class 1 as follows:

```
>> imagesc(reshape(train1(3,:), 28, 28)');
```

To reduce computational complexity and simulation time, in the following questions we focus on only three of the ten handwritten digits: "1", "2", and "7".

Question 1:

In this question, we explore the performance of K nearest neighbor (K-NN) classifiers at distinguishing handwritten digits. There are three classes, corresponding to the digits "1", "2", and "7". To determine neighborhoods, we use the Euclidean distance between pairs of vector-encoded digits x_i and x_j:

$$d(x_i, x_j) = ||x_i - x_j|| = \left(\sum_{\ell=1}^{784} (x_{i\ell} - x_{j\ell})^2 \right)^{0.5}$$

a) Implement and submit a function which finds the K nearest neighbors of any given test digit, and classifies it according to a majority vote of their class labels. Construct a training set with 200 examples of each class ($N = 600$ total examples). What is the empirical accuracy (fraction of data classified correctly) of 1-NN and 3-NN classifiers on the test examples from these classes?

b) Plot 5 test digits which are correctly classified by the 1-NN classifier, and 5 which are incorrectly classified. Do you see any patterns?

c) Implement and submit a function which uses 5-fold cross-validation, on the training dataset from part (a), to estimate the accuracy of a K-NN classifier. Determine a cross-validation accuracy estimate for five candidate classifiers, produced by the use of $K = \{1, 3, 5, 7, 9\}$ *nearest neighbors. Create a plot of these accuracy estimates versus* K*. Which classifier is estimated to be most accurate?*

d) For the classifier estimated to be most accurate in part (c), determine its performance on the held-out test data. Is this similar to the cross-validation accuracy estimate?

e) Suppose that, instead of using 200 examples per category, we had used 6,000 examples per category ($N = 18{,}000$ *total examples, almost the full MNIST database for these classes). By what factor would the computational cost of classifying test images increase? By what factor would the computational cost of the 5-fold cross-validation procedure in part (c) increase?*

Question 2:

In this question, we compare a naive Bayes classifier to the K-NN classifier from problem 1. Because the digits are represented by continuous intensities, we use a Gaussian distribution to model each feature (pixel).

a) Implement a function which determines maximum likelihood (ML) estimates of the Gaussian mean $\mu_{k\ell}$ *and variance* $\sigma^2_{k\ell}$ *parameters for each feature/pixel,* ℓ*, of each class,* k*. For some "background" pixels near image boundaries, the resulting variance estimates may be extremely small (or even 0). To avoid numerical problems, we suggest thresholding all estimated variances to a minimum value of* 10^{-6}*.*

b) Estimate naive Bayes models for each of the three digit classes, using the same 200 training examples per class ($N = 600$ *total examples) considered in problem 1. Plot the mean and variance of each class as images.*

c) Using the naive Bayes models from part (b), estimate categorization accuracy on the held-out test data from problem 1(d). How does this compare to the best K-NN classifier? How does the computational cost of the naive Bayes and K-NN classifiers compare?

d) Suppose that we trained both the K-NN and naive Bayes classifiers on a much larger database, with 6,000 examples per class. Which classifier do you think would see the larger increase in accuracy from using more training data? Why?

e) In part (a), we suggested a heuristic thresholding operation to avoid overly small ML variance estimates. What would be a more principled way to address this problem?

Question 3:

We now consider a modification of the Gaussian naive Bayes model from problem 2. As before, we provide a different mean parameter $\mu_{k\ell}$ for each feature/pixel of each class. However, we constrain all features of each class to have the same *variance σ_k^2, so that*

$$p(x_i \mid y_i = k, \mu_k, \sigma_k^2) = \prod_{\ell=1}^{784} \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left\{ -\frac{(x_{i\ell} - \mu_{k\ell})^2}{2\sigma_k^2} \right\}$$

For each class, there are 785 unknown parameters: 784 scalar mean parameters, and 1 scalar variance parameter.

a) *Derive formulas for the maximum likelihood (ML) estimates of the mean and variance parameters of the model above. Simplify your answers as much as possible.*

b) *Using the results from part (a), estimate constrained naive Bayes models for each of the three digit classes, using the same 200 training examples per class ($N = 600$ total examples) considered in problem 1. What is the categorization accuracy of the resulting models on the held-out test data? How does this compare to the unconstrained naive Bayes classifier from problem 2(c)?*

c) *Consider an extremely simple classifier, which measures the Euclidean distance of the test digit to each of the three mean vectors estimated in part (b), and assigns the class label of the closest. This is like applying a 1-NN classifier to a "synthetic" dataset with only one example per class. What is the performance of this classifier? Is it equivalent to the constrained naive Bayes classifier considered above? Why or why not?*