Programming Project 1 –Classification Using Maximum-likelihood, Parzen Window, and K-Nearest Neighbor
CAP 5638, Pattern Recognition, Fall, 2015
Department of Computer Science, Florida State University

Points: 100
Due: Monday, October 26, 2015
Maximum Team Size: 2
Submission: Hardcopy (including programs) is required and is due at the beginning of the class on the due date.

Background: Bayesian decision theory provides the optimal decision rule for classification when the probabilities are known. In practice, however, these required probability models are rarely available and are typically estimated based on the labeled training samples. When the parametric forms of the underlying probability distributions are known, one can use maximum likelihood or Bayesian method to estimate the probability distributions; however, the commonly used parametric forms are often not sufficient for real world applications. Due to the increasingly available computational power, nonparametric methods are widely used for pattern recognition even though they often require substantially more computation compared to the parametric techniques and require much larger datasets in order for them to be accurate, especially when the dimensionality of the feature space is large. In this programming project, you will implement and compare three methods for classifying patterns: maximum-likelihood estimation and Parzen window estimation for estimating probability models to be used with the minimum error rate classifier, and k-nearest neighbor method.

Purpose: To know how to implement commonly used classifiers for pattern classification and how to choose parameters using cross validation.

Assignment: Implement programs to do the following.

1) **Bayesian classifier based on maximum-likelihood estimation**
   You need to choose proper parametric forms for the underlying probability distributions. In the report, you need to specify the parametric forms, the maximum-likelihood estimate for the parametric forms, and the resulting discriminant functions for classification for each dataset.
2) **Bayesian classifier based on Parzen window estimation**
   You need to choose proper window functions, which you need to specify in the report along with the resulting discriminant functions for classification for each dataset.
3) **Basic k-nearest neighbor rule**
   In this case, you need to implement the k-nearest neighbor classifier by first finding the first k nearest neighbors of an input and then classifying it as the class that appeared most among the nearest neighbors. Here you need to specify the distance measure between two inputs.

For Parzen window estimation and the k-nearest neighbor rule, proper choice of parameters and data normalization is necessary in order to achieve satisfactory performance. Here we choose the parameter values using the leave-one-out performance on the training set: For a set of candidate values, we compute the leave-one-out performance on the training set for each candidate and the optimal one is the one that gives the best leave-one-out performance (in case there are ties, specify how the ties will be broken).

After programming is done, apply your programs on the following three datasets.

- Iris dataset
  This is a classical dataset, consisting of 150 labeled samples with 50 samples in each of the three classes. For this assignment, it is divided into a training set (http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/iris_training.txt), and a test set (http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/iris_test.txt). The training set consists of 99 samples with 33 samples for each class and the test set includes 17 samples for each of the classes. A description of the dataset is available at http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/readme_iris.txt.

- UCI wine dataset
  The dataset is divided into a training set (http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/wine_uci_train.txt) and a test set (http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/wine_uci_test.txt). The training set consists of 89 examples in three classes (30 for class 1, 35 for class 2 and 24 for class 3) and the test set consists also of 89 examples (29 for class 1, 36 for class 2, and 24 for class 3). A description of the dataset is available at http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/wine_uci_desc.txt.

- A small handwritten digit dataset
  Similar to the other two datasets, it is also divided into a training set (http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/zip_train_small.txt) and a test set (http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/zip_test.txt). The training set consists of 200 training samples (20 samples for each digit) and the test set consists of 2007 samples (359 for digit 0, 264 for digit 1, 198 for digit 2, 166 for digit 3, 200 for digit 4, 160 for digit 5, 170 for digit 6, 147 for digit 7, 166 for digit 8, and 177 for digit 9). A description of the entire dataset is available at http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/zip_info.txt. For non-CS students who do not have sufficient computational resources, you can use a subset of the test set (http://www.cs.fsu.edu/~liux/courses/cap5638-2015/assignments/zip_test_small.txt) with 20 samples for each digit. Hint: for maximum-likelihood estimation using a normal distribution, as the estimated co-variance matrices are not invertible, you need to do feature selection so that the co-variance matrices become invertible or you can use Equation (77) from Chapter 3 of the textbook to make the co-variances matrixes invertible; for details, see Section 3.7.3 of the textbook.

For each dataset, you need to first estimate the necessary data normalization and parameters using the training set only (you need to document the results for different choices of parameters on the training set in the leave-one-out sense for Parzen windows and k-nearest neighbor rule) and then document the classification results on the test set using each of the methods. Then you need to compare different methods in terms of classification performance and required time for classification, and give justifications for your observed empirical results.

Submission:

- **Report** – You need to turn in a report, including description of your algorithms and implementations, experimental results, and comparison and evaluation and analysis of the three different methods.
- **Source code** – You should attach all the source programs you developed for this programming assignment.
- **Test results** – Running results of your implementation.

Hardcopy is required for submission. Each group only needs to turn in one copy of the report.

Extra Credit Options: Please state clearly in your report if you have implemented any of the following extra credit options.

a. **(10 points) Speed up using k-d tree**
   With k from 1 to 10 with an increment of 1, first build a k-d tree from the training set and then classify the test samples using the k-nearest neighbor classifier by finding the nearest neighbors using the k-d tree. Compare the classification accuracy and the number of distance calculations with the basic k nearest neighbor implementation on the three datasets. Summarize your observations and justify your results.
b. **(10 points) Recognition of your written digits**
   Apply the best classifier you have for hand written digit recognition on a test set consisting of your own written digits (you need to create the dataset). Document the classification performance, what you have done to improve the performance, and any additional issues you have handled.

Grading

- **Report** – 10 points
- **Correct implementation** – 50 points
  - **Maximum likelihood estimation** – 20 points
  - **Parzen window estimation** – 15 points
  - **Basic k-nearest neighbor** – 15 points
- **Results** - 25 points
- **Analysis** – 15 points
- **Speed up using k-d tree** – 10 points
- **Recognition of your written digits** – 10 points

Additional information:

The following shows the hand written digits in the training set so that you know what the digits look like. If you like to show a digit, you need to scale the values from [-1 1] to [0 255] and reorganize the pixels from a 256-element vector to a 16×16 image; you may need to rotate the image.