CSC 461 Programming Languages – Fall 2015
Programming Assignment #1: Pattern Recognition in Python

Introduction

Humans, unlike computers, appear to be "hard-wired" for a wide variety of pattern recognition tasks. To classify patterns on the computer, we define a feature vector [x_1,x_2,…,x_n] comprised of *n* measurements. These *n* measurements define an *n*-dimensional feature space. If the features are chosen properly, class samples will map to non-overlapping clusters in feature space.

A *minimum distance classifier* is one of the simplest statistical pattern recognition systems. In the training phase, we partition the feature space by computing the centroid of the training samples from each class. To classify an unknown sample, we use a distance formula to find the closest class centroid. The Euclidean distance between two points $\boldsymbol{p}$ and $\boldsymbol{q}$ in *n*-dimensional space is given by

$$dist(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

The class of an unknown sample is determined by the centroid of minimum distance from the sample.

Problem

Implement a minimum distance classifier in Python. For example, the popular iris flower data set contains four measurements from three different species of iris. Measurements include length and width of iris sepals and petals. Based on these four features, it is possible to classify a flower into one of three different iris species with high accuracy.

Implementation

Input data are stored in CSV (comma-separated value) files using a specific format. The first line of each input file contains a descriptive header, consisting of the database name followed by class names. The second line contains labels (sample, class, petal length, petal width, etc.). The remainder of the file contains sample data, corresponding to the labels on line 2. Each sample record is comprised of a sample ID, class label, and feature measurements for that data sample. You will not know in advance how many features are on each line, or how many lines are in the file. For simplicity, you may assume that sample ID and class label are integers, and measurements are floating point values. Class labels will typically be sequential integer values starting at 0 (e.g., *0=Iris setosa, 1=Iris versicolor, 2=Iris virginica*). Samples are not necessarily listed in any meaningful order. Data entry fields will be separated by commas in the CSV files.

Classification accuracy will generally improve when the sample measurements are normalized. Use the following formula to normalize measurements to the range [0,1]:

$$(value - min) / (max - min)$$

where *min* and *max* are the minimum and maximum values in the data. You should do this individually for each set of measurements (petal length, petal width, etc.). Be sure to normalize both training data (centroids) and testing data.

After the classifier is trained, a different data set (the testing data set) is used to determine classifier accuracy. When the data set is relatively small, a rigorous method for testing classification accuracy is leave-one-out cross-validation: select one sample, train on the remaining *n-1* samples, and test only the selected sample for classification accuracy. Repeat until you have tested each sample. The percentage of correctly classified samples is a simple measure of classifier accuracy. Percent accuracy is defined as

100 × (# of correct classifications) / (# of samples)

Program usage

```
python mdc.py datafile.csv
```

Program output

1) Print classification accuracy for each class, and overall classification accuracy, to the standard output.
2) Print classification accuracy and individual sample cross-validation results to a file named *datafile.cv* (for an input file named *datafile.csv*), marking incorrectly classified samples with an asterisk.

Sample run

```
% python mdc.py iris.csv
```

Output

```
Iris database
class 0 (Iris-setosa): 50 samples, 100.0% accuracy
class 1 (Iris-versicolor): 50 samples, 88.0% accuracy
class 2 (Iris-virginica): 50 samples, 90.0% accuracy
overall: 150 samples, 92.7% accuracy
```

Contents of iris.cv

```
Iris database
class 0 (Iris-setosa): 50 samples, 100.0% accuracy
class 1 (Iris-versicolor): 50 samples, 88.0% accuracy
class 2 (Iris-virginica): 50 samples, 90.0% accuracy
overall: 150 samples, 92.7% accuracy

Sample,Class,Predicted
1,0,0
2,0,0
3,0,0
. . .
50,0,0
51,1,2,*
52,1,1
. . .
148,2,2
149,2,2
150,2,2
```

Notes

- When you are finished writing, testing, and debugging your program, submit your source code in a *zip* or *tar* archive using the *Submit It!* link on the MCS Department Website. The submit program is accessed via the MCS Web page (http://www.mcs.sdsmt.edu), by selecting the list item on the left entitled "Submit it!". Usage is self-explanatory: enter your name, choose the instructor and click "Select Instructor", choose the appropriate course, browse to the filename you wish to submit, and click "Upload".
- Submit your program by the due date (Thursday September 24) in order to receive credit for this assignment. Late programs will not be accepted for partial credit unless prior arrangements have been made with the instructor. If you have any problems with the submit program, report them to your instructor and submit your program by email instead.
- To receive full credit, your code must be readable, modular, nicely formatted, and adequately documented, as well as complete and correct. It must build and run successfully using the current Python 3.4 interpreter under Windows and Linux. If your program does not run correctly, indicate why. This will make it easier to give you partial credit.
- Work in teams of two students on this assignment. Teams should make one joint submission, not individual submissions for each team member.

Partners for PA#1

Anderson, John and Hill, Grant
Bodvig, Allison and Veer, Carrie
Bonn, Charles and Owen, Zachary
De Young, Matthew and Singh, Akshay
Deziel, Kendra and Roloff, Benjamin
Doell, Taylor and Rarden, Elliott
Glass, Bryon and Herman, Alex
Johnson, Colby and Lane, Derek
Kaiser, Benjamin and Sieh, Christian
Li, Yanlin and Miller, Forrest
Mowry, Joseph and Schweigert, Joshua
Navarro, Christopher and Nienhueser, Alex

When you finish the assignment, email me a brief description of team member contributions (both you and your partner). Your comments will be kept private, but I may choose to grade team members individually.