Project 1, k-Nearest Neighbor Classification

CSC736 Machine Learning
Spring 2022
Max Score: 100

Objectives

In this assignment you will implement k-nearest neighbor classification. Your submission should have a folder called nearest neighbors, which contains your code and the README.txt file.

Command-line Arguments

Your program will be invoked as follows:

```
knn_classify pendigits_training.txt pendigits_test.txt <k>
```

The arguments provide to the program the following information: The first argument, pendigits_training, is the name of the training file with training data stored. The second argument, pendigits_test, is the test file with the test data is stored. The third argument specifies the value of k for the k-nearest neighbor classifier. The training and test files will follow the same format as the text files in the UCI datasets directory. A description of the datasets and the file format can be found in the folder. For each dataset, a training file and a test file are provided. The name of each file indicates what dataset the file belongs to, and whether the file contains training or test data.

Implementation Guidelines

1. Each dimension should be normalized, separately from all other dimensions. Specifically, for both training and test objects, each dimension should be transformed using function $F(v) = \frac{(v-mean)}{std}$, using the mean and std of the values of that dimension on the TRAINING data. To compute the std, using function $std = \sqrt{\frac{\sum |v-mean|^2}{N}}$.

2. Use the L2 distance (the Euclidean distance) for computing the nearest neighbors.

Classification Stage

For each test object you should print a line containing the following info:

- object ID. This is the line number where that object occurs in the test file. Start with 0 in numbering the objects, not with 1.
- predicted class (the result of the classification). If your classification result is a tie among two or more classes, choose one of them randomly.
- true class (from the last column of the test file).
- accuracy. This is defined as follows:
  - If there were no ties in your classification result, and the predicted class is correct, the accuracy is 1.
  - If there were no ties in your classification result, and the predicted class is incorrect, the accuracy is 0.
  - If there were ties in your classification result, and the correct class was one of the classes that tied for best, the accuracy is 1 divided by the number of classes that tied for best.
  - If there were ties in your classification result, and the correct class was NOT one of the classes that tied for best, the accuracy is 0.

To produce this output in a uniform manner, use these printing statements:

For C/C++, use:

```
printf("ID=%5d, predicted=%3d, true=%3d\n", obj_id, pred_class, true_class);
```

For Java, Python or any other language, just make sure that you use formatting specifies that produce aligned output that matches the specs given for C/C++.

After you have printed the results for all test objects, you should print the overall classification accuracy, which is defined as the average of the classification accuracies you printed out for each test object. To print the classification accuracy in a uniform manner, use these printing statements:

For C/C++, use:

```
printf("classification accuracy=%6.4lf\n", classification_accuracy);
```

For Java, Python or any other language, just make sure that you use formatting specifies that produce aligned output that matches the specs given for C/C++.

Output for answers.pdf

In your answers.pdf document, you need to provide parts of the output for some invocations of your program listed below. For each invocation, provide:

1. ONLY THE LAST LINE (the line printing the classification accuracy) of the output by the test stage.

Include this output for the following invocations of your program:

```
knn_classify pendigits_training.txt pendigits_test.txt 1
knn_classify pendigits_training.txt pendigits_test.txt 3
knn_classify pendigits_training.txt pendigits_test.txt 5
```