Assignment 10: Digit Recognizer
CS 730/830, Fall 2015

Electronic submission due at **11:30pm on Mon, Nov 23**
Hardcopy submission due at **start of class on Tue, Nov 24**

Overview

You will write a trainable classifier for data with integer-valued attributes. Your program will be given on standard input a sequence of labeled examples to use for training, followed by a sequence of unlabeled examples for which it must provide a classification and a confidence.

Your program should accept a command line argument indicating the algorithm it should use: `knn` (k-nearest neighbors with some preset k that you can choose) or `linear` (on-line linear regression using LMS). Graduate students need to implement a third option, `nb` (naive Bayes).

Although your classifier should be reasonably general-purpose, we will mainly test it on the task of classifying images of handwritten digits. An image will be presented as a vector of about 200 integer values between 0 and 4 inclusive, representing pixel intensity values.

Input/Output

The input will consist of a short header giving the number of input attributes per instance, the number of values each attribute can take, and the number of classes. After a special indicator line, the training instances will follow, one per line, with attributes separated by spaces and the class label at the start of the line. The test instances will follow after a special indicator line:

```
4 attributes, 4 values, 10 classes
-- training --
0 0 0 1 0
9 3 3 2 2
1 0 2 0 2
-- test --
0 1 1 0
3 1 2 1
```

Your program must emit a class label and a confidence (between 0 and 1) for each of the test instances. Each label-confidence pair should be on its own line and in the same order as the corresponding instances appeared in the input. Eg:

```
0 0.8
9 0.56
```

Supplied Utilities

We supply:

`digits.data` a file containing 60,000 randomly ordered training examples. This file is also provided in `~cs730/public/digits.data` on `agate.cs.unh.edu`.

`class-reference` a sample solution.

`class_tool` a program to help you test your classifier. It reads digits data on standard input and uses this data to train and test your classifier (your program's name can be specified on the command-line, the default is `class`). `class_tool` has two forms of output:

1. A confusion matrix is printed to standard output, along with the accuracy of the classifier. The confusion matrix shows the percentage of times that an item of a given label (the row) is classified as a certain label (the column) by the classifier.

2. A (optional) HTML output shows information about the most confident and least confident correct matches along with the most confident false positives and false negatives for each digit. If used on agate, the output will go to a website which you can browse to. The URL for this site is based on your username on agate and it has the form:

   `http://cs.unh.edu/~cs730/asn5/<username>`

`class_tool` takes the following arguments:

-seed Specify the seed for the random number generator (the default is to use a seed value of 1).

-maxsize Specify the max size of training data for the trials. This option is useful for debugging when you do not want to wait for all of the trials to complete.

-quiet Do not print output for each trial to standard output.

-html Output HTML.

-htmldir Specify the root directory for the HTML output. This option cannot be used with the version in cs730/public on agate. The default is to output to the cs730 website on agate in a directory with your username. If you use `class_tool`on your own machine, you should specify a different "htmldir".

To use `class_tool` on agate.cs.unh.edu, use the following command:

```
~cs730/public/class_tool [-<class_tool_arg> ...] [<agent_arg> ...] < ~cs730/public/digits.data
```

i.e.

```
~cs730/public/class_tool -seed ‘date +%M%S‘ -maxsize 500 \
                -agent ./myagent nb < ~cs730/public/digits.data
```

will run the program `myagent` with a seed based on the current minute and second for all trials with less than or equal to 500 instances of training data using naive Bayes.

Write-up

Submit a brief write-up with your hardcopy solution answering the following questions:

1. Describe any implementation choices you made that you felt were important. Mention anything else that we should know when evaluating your program. If you implement anything beyond the assignment as written, please be sure to discuss it
2. What can you say about the the time and space complexity of your program?
3. For each algorithm you implemented, describe a learning problem that you think it will perform poorly on and explain why.
4. What suggestions do you have for improving this assignment in the future?

Evaluation

We will look at your code mainly to try to give you partial credit.

k-NN: learns at all 1

k-NN: learns well 3

k-NN: not super-slow 1

linear: learns at all 1

linear: learns well 3

linear: not super-slow 1

NB: learns at all 1

NB: learns well 3

NB: not super-slow 1

Write-up 2

Design Suggestions

No significant suggestions. It might be helpful to keep in mind that the number of instances is much larger than the number of attributes which is larger than the number of classes.