CS475 - Assignment 4

(K Nearest Neighbor Practice)

REWARD: 150 points

DUE: Tue Mar 22 at 11pm PT

TEST DATA: testDataA4.tar or testDataA4.zip
LIBRARY: mat.tar, mat.zip

---

1 The Task

It is usually the case that you can consider the training data to be exemplars of correct answers and so points that are similar to a training data point might reasonably be assumed to have the same output value as that training data point. We'll call the following the Rule of Similarity:

$$\text{if } \mathbf{f(x)} \rightarrow \mathbf{y} \text{ then } \mathbf{f(z)} \rightarrow \mathbf{y} \text{ for } \mathbf{z} \text{ near } \mathbf{x}.$$

As practitioners we have several choices to make. 1) what points to we pick? and 2) what do we mean by distance? In the first case, we have some tools we have discussed already for picking points. SVMs, for instance, try to pick points along the boundaries know to delineate different values of **y**. If the dimensions are somewhat large and we don't have a lot of training points or the boundary is very non-linear then perhaps we just use all the training points. If we have too many dimensions then we can try to reduce the dimensions with PCA first and then use K Nearest Neighbor (KNN).

For the Rule of Similarity to holds we need to know what is meant by "near". That is, what distance measure do we use?

For the KNN algorithm we will want a true distance measure in which the Triangle Inequality holds. The "go to" distance measure is Euclidean distance which is in the family of Lebesgue norms or Lp norms or just p-norms. These norms range from $L1$-norm or taxi cab distance and $L2$-norm which is Euclidean distance, to $L\infty$-norm which max of the magnitude of distances along each dimension. For this assignment just use the Euclidean distance (L2-norm).

Input is first a labeled matrix of rows where each row begins with a word (no whitespace in the word) followed by d columns of data. The matrix can be read by the readLabeledRow() function which returns a symbol table of labels (see example in the test data). The result of the read is the first column in the matrix is the index of the corresponding label in the symbol table. Not ideal, but workable. The second half of the input is a second matrix of unlabeled data of d columns each. For each row in the matrix use a brute force KNN algorithm to look up the K closest answers and print out that and their labels. Sample output will be provided. The program will take K on the command line. An example call might be:

```
knn 5 < colorsUniq.txt
```

An example input file containing the "training" data and the "test" data:

```
20 4
AcidGreen                       176 191 26
Aero                            124 185 232
AeroBlue                        201 255 229
AfricanViolet                   178 132 190
AirForceBlue(RAF)               93  138 168
AirForceBlue(USAF)              0   48  143
AirSuperiorityBlue              114 160 193
AlabamaCrimson                  175 0   42
AliceBlue                       240 248 255
AlizarinCrimson                 227 38  54
AlloyOrange                     196 98  16
Almond                          239 222 205
Amaranth                        229 43  80
AmaranthPink                    241 156 187
AmaranthPurple                  171 39  79
AmaranthRed                     211 33  45
Amazon                          59  122 87
Amber                           255 191 0
AmericanRose                    255 3   62
Zomp                            57  167 142
2 3
197 97  17
210 30  45
```

Implement a C++ program called **knn** that does a brute force comparison of a row in

the test data to all the elements of the training data and selects the K nearest elements using L2-norm (Euclidean distance) distance. Here is example output for:

```
knn 5 < colorsUniq.txt

SOLVE: 197.0000  97.0000  17.0000
Sorted (size of dist: 6 X 5)
  1.73205 AlloyOrange 196 98 16
  71.2461 AmaranthRed 211 33 45
  75.8288 AlizarinCrimson 227 38 54
  88.7919 AmaranthPurple 171 39 79
  88.9326 Amaranth 229 43 80
  96.7368 AcidGreen 176 191 26

  SOLVE: 210.0000  30.0000  45.0000
  Sorted (size of dist: 6 X 5)
  3.16228 AmaranthRed 211 33 45
  20.8327 AlizarinCrimson 227 38 54
  41.8927 Amaranth 229 43 80
  46.1952 AlabamaCrimson 175 0 42
  52.5167 AmaranthPurple 171 39 79
  55.1634 AmericanRose 255 3 62
```

Hint: in my version of the program the only loop I had was on the rows in the test matrix. In fact, if I had used something like: `testdata.map(knn)` I probably could have had no loops in my code at all.

2 Submission

Homework will be submitted as an **uncompressed** tar file to the homework submission page linked from the class web page. Be sure to include a make file to build your code and do NOT include the picture files. I will supply some. You can submit as many times as you like. **The LAST file you submit BEFORE the deadline will be the one graded.** For all submissions you will receive email at your uidaho.edu mail address giving you some automated feedback on the unpacking and compiling and running of code and possibly some other things that can be autotested.

Have fun.