CSE 517a: Homework #1

Due on Wednesday, January 30th, 2014, at 1:00pm

Kilian Q. Weinberger

Contents

Theory

1 Matrices

MATLAB is an interpreted language with a lot of slow execution overhead. However, matrix operations are extremely optimized and very fast. In order to be a successful MATLAB programmer, you need to free yourself from the "for-loop" thinking and start thinking in terms of matrix operations.

Assume we have n data vectors $\underline{x}_1, \dots, \underline{x}_n \in R^d$. And let us define a matrix $X = [\underline{x}_1, \dots, \underline{x}_n] \in R^{d \times n}$, where the i^{th} column is a vector $\underline{x}_i$.
(a) Show that the Gram matrix (aka inner-product matrix)

$$G_{ij} = \underline{x}_i^T \underline{x}_j$$

can be expressed in terms of a pure matrix multiplication.

□
(b) Let us define a new matrix $S \in R^{n \times n}$

$$S_{ij} = \underline{x}_i^T \underline{x}_i$$

Show that the squared-euclidean matrix $D \in R^{n \times n}$,

$$D_{ij} = (\underline{x}_i - \underline{x}_j)^2, \tag{1}$$

can be expressed as a linear combination of the matrix S and G. (Hint: It might help to first express (1) in terms of inner-products.) What can you say about the diagonal of D? What do you need to do to obtain the true Euclidean distance matrix E?

□
(c) Generalize the previous results to inner-products and squared Euclidean distances of two data sets X and $Z = [\underline{z}_1, \dots, \underline{z}_m]$. (ie $D_{ij} = (\underline{x}_i - \underline{z}_j)^2$. What are the dimensions of S and D.
□

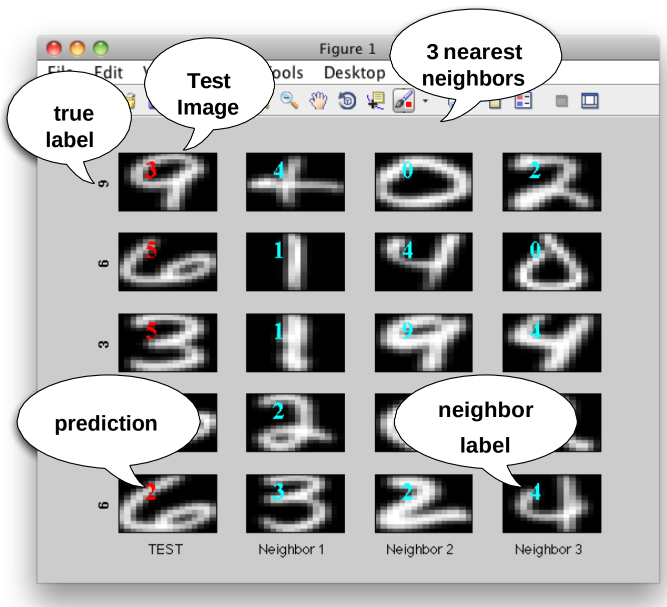

Figure 1: The nearest neighbor classifier visualization tool. **visualhw1.m**

Programming

2 kNN

In this programming assignment, you will built a classifier for handwritten digits classification and face recognition.

Install MATLAB, subversion and check out the files for hw1. Note: Subversion (SVN) is pre-installed on Macs and probably Linux. To install subversion for Windows, you can download it from CollabNet Subversion Downloads and install it following the instruction. A list of other options are available under Other Subversion package. Once SVN is installed call:

```
svn co https://shell.cec.wustl.edu:8443/cse517a_sp14/svn/YOURWUSTLKEY/
```

You should now have the following files:

```
analyze.m hw1tests.m
innerproduct.m l2distance.m
digits.mat  findknn.m
hw1tictoc.m  knnclassifier.m
visualhw1.m  faces.mat
```

These files are all partially implemented, the specifications of the functions are in the header of the files. To obtain some data to test your code run

```
>> load faces
>> whos
  Name          Size                Bytes  Class     Attributes

  xTe        1178x120             1130880  double
  xTr        1178x280             2638720  double
```

2 kNN continued on next page. . .

```
  yTe            1x120                  960   double
  yTr            1x280                 2240   double
```

Here, **xTr** are the training vectors with labels **yTr** and **xTe** are the testing vectors with labels **yTe**. You can visualize one of the faces by running

```
>>  figure(1);
>> clf;
>>      imagesc(reshape(xTr(:,1),38,31));
>>  colormap  gray;
```

There is also a little k-nearest neighbor classifier visualization tool, called **visualhw1.m**. Run the following commands to test it (it should give random predictions as in Figure 1):

```
>>  visualhw1  faces
>> visualhw1  digits
```

The following questions will ask you to finish these functions in a pre-defined order. There is also a function **hw1tests.m**, which contains tests that evaluate if your code is correct. You must include one test in **hw1tests.m** per implemented function, that tests for some possible bug (e.g. after you implement the **l2distance.m** function, include a test that checks if distances are always non-negative). These tests will later be run on other students' s implementations.

Important rule: You are not allowed to use any loops (*e.g.* for-loops, while-loops) in your code.[1]

(a) Implement the function **innerproduct.m**. Use your results from the previous part of the homework.

□

(b) Implement the function **l2distance.m**. Use your results from the previous part of the homework.

□

(c) Implement the function **findknn.m**, which should find the k nearest neighbors of a set of vectors within a given training data set.

□

(d) The function **analyze.m** should compute various metrics to evaluate a classifier. Implement the "accuracy" metric (ie the percentage of data points that were classified correctly).

□

(e) Take a look at **hw1tictoc.m**. This script runs the (not implemented) k-nearest neighbor classifier over the faces and digits data set. In its current implementation the results are picked at random. The faces data set has 40 classes, the digits data set 10. What classification accuracy would you expect from a random classifier?

□

(f) Implement the function **knnclassifier.m**, which should perform k nearest neighbor classification on a given test data set. Test your function with **hw1tictoc.m**. You can now also run **visualhw1.m**. Hopefully you will get better results than in figure 1.

□

Well done! That was it. Make sure you added at least one test for every single function that you implemented.

[1]For someone who is used to Java or C/C++ this might seem strange, but I really want you to get used to thinking in terms of matrices. Matrix operations are very efficient and often highly parallelized.