CS475 - Assignment 3 (PCA and Dimensionality Reduction)

REWARD: 180 points
DUE: Tue Mar 8 at 11PM PT
TEST DATA: testDataA3.tar or testDataA3.zip
LIBRARY: mat.tar, mat.zip

---

1 The Task

Consider data coming in as a matrix of R samples of C dimensional data that is represented as a R x C matrix. Let's compress this using PCA so that it is fewer than C dimensions. We want the retained dimensions to be the dimensions that carry the most variance (information). The axes of this new space will be determined by the eigenvectors we will compute.

The task is to write a C/C++ program called **pca** that uses principal component analysis (PCA) to pick the largest k eigenvalues in magnitude and encodes the data in K dimensional space instead of C dimensions. This turns out to be easy if you have something that will give you eigenvalues/eigenvectors of a matrix. Essentially you will be compressing the data by translating the data to the K most significant eigenvectors. Eigenvalues and eigenvectors can be computed in the matrix library. Here is a help sheet on PCA. Select from that to help your implementation.

To make this more entertaining our data will be pictures. A grayscale picture that is R x C in size can be thought of a R samples of C dimensional vectors of data. The vectors tend to have pattern and cluster because, in fact, neighboring columns of pixels tend to be highly correlated. A color picture is that is R x C can be thought of as a R x (3*C) matrix of points where each pixel has an R, G, and B value. Again, neighboring columns of pixels tend to be highly correlated.

This program will take an integer argument on the command line. This is the number K, that is, the number of eigenvectors you want to keep. Then read in a picture in ppm (color) or pgm (gray scale) format from stdin. No worries, the Matrix Library has a readImagePixmap procedure which auto-detects the ppm or pgm format and creates an appropriate matrix (see the Matrix Library documentation). It also has methods to write **simple** pgm and ppm (part of the netpbm open source graphics standard) files in and out of matrices!

Next center the columns of data about the mean of each column.

Your program will then compress the information using PCA this will give you an R x K matrix. This is the objective in a machine learning algorithm. This is the compressed or abstracted version of the input data. You can then train your ML algorithms on this data.

For us we will simply now turn around and decode the data (make a picture) and look at the fidelity of the encoding. So the next step is to decode the matrix using the reduced eigenvector matrix to recover the picture. Because there will be some loss though minute, let's compute the root mean square error (RMSE) distance between the original picture and the recovered picture and print that (see example output for full list of outputs). You can see as you increase K the RMSE should drop dramatically. Why?

RMSE is defined as

$$\sqrt{\frac{\sum_{i=1}^{n}(y_i - t_i)^2}{n}}$$

where y_i are the generated or predicted points and t_i are the target values. For our problem they are the reconstituted pixel values and the original pixel values. n is the number of samples. In our case this is the number of pixels in the picture.

Next your program should write either a ppm or pgm file called "z.ppm" or "z.pgm" depending on whether the original file read in was color. Use the matrix library write routines to save the recovered picture in the file. The test scripts may look at your picture file to see if you got the right picture back.

Example output using `pca 40 < girlWithPearlEarring.ppm`. Please use these names of matricies and labels for easy of grading.

```
(size of Image: 468 X 1200)
(size of Mean: 1 X 1200)
(size of EigenVectors: 1200 X 1200)
(size of EigenValues: 1 X 1200)
(size of Encoded: 468 X 40)
(size of Decoded: 468 X 1200)
Min/Max/Mean Value per Pixel of Original: 0 249 47.4877
Min/Max/Mean Value per Pixel of Recovered: -19.4195 271.701 47.4877
Root Mean Squared Error: 5.6352
```

One last thing. If command line argument is a negative integer then transpose the matrix before the doing the PCA. That is the dimension of the data is now the rows and not the columns. Don't forget to untranspose when done. K is, of course, in this case -K.

2 Submission

Tar up all the code necessary along with a ¡a href=make.html¿ makefile to build the program named **pca** that reads the sample data from stdin as described above. Homework should be submitted as an **uncompressed** tar file to the homework submission page linked from the class web page. 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.