Description:

The program uses a list of vectors from a textfile as an input (based on parameters taken in the MATLAB console window). Based on this input, the program creates a random reordering of the input vectors, a reordering of the vectors based on euclidean distance from a target vector, a reordering of the vectors based on angular distance from the target vector, and a list of the original vectors projected onto its principal components.

The program was implemented in MATLAB. The main program to be run is 'PCA_second_practice.m', and this makes function calls to 'principal_components.m', 'computeEuclidDist.m', 'computeAngularDist.m', and 'quickSort.m'. To compile and run the program, you type 'PCA_second_practice' on the MATLAB console window (making sure all the listed programs are in the same directory).

The program makes use of the 'norm' function, which calculates the 'length' of a vector (in terms of the l_2 norm). It also makes use of the 'mean' and 'size' functions, which calculate the mean of a 1-D or 2-D array and the length and width of a matrix. Most importantly, the program uses the 'eig' function, which calculates the eigendecomposition of a given matrix (it gives the set of eigenvalues and eigenvectors for the given input matrix).

It should be noted MATLAB already demonstrates allocation for the pattern set struct in the main program, and can be deallocated using the 'clear' command (as an example: 'clear your_pattern_set_name'). The user can also write a collection of patterns to a text file using a series of commands:

diary('your_file_name.txt');diary on;PCA_second_practice;...;diary off;

The euclidean distance between two n-dimensional vectors $\vec{a}$ and $\vec{b}$ is given by:

$$d = (\sum_{i=1}^{n}(b_i - a_i)^2)^{\frac{1}{2}} \tag{1}$$

The angular distance between two n-dimensional vectors $\vec{a}$ and $\vec{b}$ is given by:

$$\theta = |arccos(\frac{\vec{a} * \vec{b}}{|\vec{a}||\vec{b}|})| \tag{2}$$

Given an m number of n-dimensional input vectors $\vec{a_i}$, we first find the covariance matrix. This is done by first finding the mean vector $\vec{\mu}$ of all the input vectors with the following equation:

$$\mu_s = \frac{1}{m}\sum_{i=1}^{m} a_{i,s} \tag{3}$$

where s indicates the index along the vector and i indicates the vector number.We define a matrix R to be

$$R = [\vec{r_1}, \vec{r_2}, \ldots, \vec{r_m}] \tag{4}$$

where $\vec{r_i} = \vec{a_i} - \vec{\mu}$. The covariance matrix becomes RR^T (Serrano).

To find the projection of the mean-shifted input vectors onto its principal components, we first note that we want to express a given vector $\vec{r}$ in terms of a linear combination of the n eigenvectors $\vec{v_i}$ of the covariance matrix, where the vectors are n-dimensional:

$$\vec{r} = \sum_{i=1}^{n} c_i \vec{v_i} \tag{5}$$

where c_i are the constants that make up $\vec{c}$, the new representation of $\vec{r}$ in terms of its projection onto the principal components (the eigenvectors). We also know the eigenvectors of the covariance matrix are orthogonal (meaning $\vec{v_j} * \vec{v_k} = 0$ when $j \neq k$) because the covariance matrix is symmetric. When we compute the dot product between each v_i and the left-hand and right-hand sides in (3), we get the set of m equations (Strang 51, 175, 245):

$$\vec{r} * \vec{v_i} = c_i \vec{v_i} * \vec{v_i} \tag{6}$$

Therefore to compute each element c_i for a given vector $\vec{r}$, we implement the following equation:

$$c_i = \frac{\vec{r} * \vec{v_i}}{\vec{v_i} * \vec{v_i}} \tag{7}$$

What was learned:

Through this assignment, I learned the importance of principal component analysis in the context of dimensionality reduction. The assignment demonstrated that a given set of pattern data can alternatively be represented using a basis of vectors that describe the maximum variance in a set of directions. As a result, we can negate the basis vectors that have little effect on the variance of the data (thus extracting the most relevant features for a given set of data). We can thus approach a problem such as a classification problem in terms of this reduced representation of the input data.

Acknowledgments:

Serrano, Santiago. "Eigenface Tutorial." *Eigenface_Tutorial*. Drexel University, n.d. Web. 13 Feb. 2013.

Strang, Gilbert. *Linear Algebra and Its Applications*. 4th ed. Belmont: Cengage Learning, 2006. Print.

"Sorting Algorithms/Quicksort." *Rosetta Code*. N.p., n.d. Web. 13 Feb. 2013.