Computer Science 315
Assignment 1

In this assignment, you will implement a basic data pre-processing technique for reducing the dimensionality of data, namely principal component analysis (PCA). You will apply this technique to data sets, and investigate and interpret the results you obtain.

1 Project Description

You are given two data sets that consist of n observations of dimension d. Ensure that each data set is read into an (n, d) ndarray[1].

The first part of the assignment involves doing the investigations described below, using the PCA class in `scikit-learn`:
http://scikit-learn.org/stable/modules/decomposition.html#decompositions.

For the second part of the assignment you are given a data set of handwritten signatures. In this this case you need to write a small amount of code yourself in order to do the subsequent investigation.

Investigation — Part I

1. Load the iris dataset from your resources file using `load_iris.py`. This data is in the form of an $(n, 4)$ array, where n is the number of observations and 4 is the number of features of each observation. More information about the dataset may be found at http://archive.ics.uci.edu/ml/datasets/Iris. Note the historical significance of the Iris data set!

Using the `PCA` class of `scikit-learn`, project the iris data onto the first two principal components that explain most of the the variance in the data, and state how much of the variance is explained by these two components. Plot the data on a 2D plot using the first two principal components as the x and y axes, using different colors for each of the three classes. If you reproduce the example in the `scikit-learn` description, you can be confident that you doing the right thing.

For future reference, note the separation between the three classes. In your next assignment you will compare these results to those obtained with linear discriminant analysis (LDA), which is designed to produce maximum class separation.

[1] If not, you need to transform your data appropriately.

2. A number of years ago the use of PCA for facial recognition, widely known as the eigenface approach, was very much in fashion. Briefly describe the main ideas behind this. You have to summarize the essential ideas and your findings in no more than two pages, excluding illustrations and the results from your code. **Hint:** One of the key investigators was Alex Pentland; do an internet search.

You can get a feeling of how the approach works by using the Olivetti facial data set, provided in your resources folder. This folder contains facial images, each of shape $(112, 92)$, i.e. each image has dimensionality $112 \times 92 = 10304$. The original files are in `PGM` format, but have been converted to `PNG` format for your convenience.

1. Read these files using `skimage`'s `ImageCollection` routine — see the `read_images` function in the `utils.py` module in your resources file for guidance on how to do this. You can view the faces using `iPython` widgets — see `image_viewer.py` in the resources file for guidance.
2. The next step is to flatten the faces, using either the `reshape` or `flatten` method associated with `numpy` arrays. Collect these faces in an (n, d) array, where n is the number of faces (you should find that $n = 399$), and d is the number of features of each face, i.e. $d = 10304$. This means that the number of observations is significantly lower than the dimension of each observation. What impact does this situation have on your investigation?
3. Find the PCA components of this face data set using the `PCA` class of `scikit-learn`. Specify 300 components and plot the `explained_variance_ratio_`. Roughly how many components are needed to explain most of the variance? Motivate your answer with reference to the plot.
4. The eigenfaces mentioned above are a visualization of the principal components. Display several of the principal components as images, starting at the ones associated with the most variance (i.e. those corresponding to the largest eigenvalues). What, if anything, do you learn from looking at them? **Hint:** How much detail do you see in the eigenfaces?
5. Project all the faces onto 200 components. (How much of the variance is explained by 200 components?). This means that the original dimension of 10304 of each face has been reduced to 200. Transform these 200-dimensional vectors back to the original feature space to (partially) reconstruct the faces, and display the results again using `iPython` widgets. Comment on how much visual information is lost by projecting onto 200 components and then returning to the original feature space. Experiment with smaller numbers of components. At what number of components do you lose all visual recognition of the individuals? There is no exact answer, but you should get an idea of how much you can reduce the dimension and still expect reasonable recognition results. **Note:** You will not be able to make the widget demonstration part of your report. You therefore will need to select a face to display and compare between the various situations. Make sure you choose meaningful images to display.

Investigation — Part II

For any signature verification problem one has to *normalize* the signatures. You are given a number of example signatures for each of a number of individuals, and your task is to normalize these signatures with respect to their position, scale, and orientation. Each signature is in the form of an $(n, 5)$ dimensional array, where n is the number of time steps used to sample the signature. Each sample consists of 5 features (measurements) namely, the x- and y-coordinates, the pen pressure, the pen direction, and the pen tilt. For this assignment we are chiefly interested in the x- and y-coordinates.

In order to view the signature, you can read the signature in using `numpy`'s `loadtxt` function, and plot the first column against the second column. (Try something like `plt.plot(s[:,0,],s[:,1])`).

Now that you know what the signature looks like, you should normalize each signature by removing its mean, rotating it so that it is aligned with its principal directions, and scaled so that the largest singular value equals 1.

1. Develop the code necessary to perform this task, and call your module `sign.py`. You may use the singular value decomposition (SVD) solvers in your software library, but the rest of the implementation has to be your own code. You will find it find it convenient to set the argument `full_matrices=False` in `numpy.linalg.svd`.

2. Using the SVD decomposition of the first two columns of the data (i.e. just the x and y coordinate data), rotate the signatures so that that the principal directions coincide with the coordinate axes. (Remember to remove the mean first!)

3. Draw the ellipse that corresponds to the one standard deviation curve of the covariance matrix. There is a nifty trick to do this. First construct a unit circle, using `x = np.cos(t)` and `np.sin(t)` where `t=np.linspace(0.,2*np.pi,101)`. Use the SVD you calculated for the signature to map this circle to the ellipse.

4. Scale the signature so that the largest variance is 1, keeping the same aspect ratio. Again draw the one standard deviation ellipse.

5. Whiten the signature so that the variances along both axes are 1. Draw the one standard deviation circle.

6. Plot all the normalized sample signatures of a single individual on the same axes. How well does this normalization work? Have you spotted any anomalies? Can you think of any situation where this procedure will not work very well?

7. Since the signs of the eigenvectors are not specified, a specific sign can cause the signature to be reflected around one or both of the axes. Modify your software to attempt to identify the problematic cases and correct this issue automatically. **Hint:** Assume that the signatures are produced by writing from left to right. This means that the x-component of the first principal direction should be positive, and the y component of the second principal direction should also be positive.

8. Now use all five features of the signatures to construct the SVD. Investigate whether the data is really 5 dimensional by investigating the magnitudes of the singular values.

9. Project the 5-dimensional signature onto the first two principal components. Then reconstruct the 5 dimensional signature from the projected data and display the signature again by plotting the reconstructed (x, y) points in the signature. How much information do you lose by projecting down to two dimensions?