Machine Learning

Lab 3: PCA

1 Overview

The method of principal component analysis is a useful tool for both data dimension reduction and to understand large data collections. In this lab, you will write code to find the principal components (PCs) of an arbitrary collection of data, then apply this to two well-known data sets: the MNIST data set which contains a large collection of handwritten digits, and the MIT-CBCL data set containing images of faces under various lighting situations (as well as images of non-faces, not used in this lab).

2 Provided Resources

- `princomp.m` - A placeholder function that calculates the principal components for a collection of vectors.
- `mnist.mat` - The MNIST handwritten digits data set. Contains `X` with observations in each column for a subset of the zeros and ones from the full collection. Also contains `dims` with the dimensions of each digit image (28-by-28).
- `cbcl.mat` - The CBCL faces data set. Contains `X` with observations in each column for each face in the collection. Also contains `dims` with the dimensions of each face image (19-by-19).
- `imcloud.m` - The first argument is a matrix of (x, y)-pairs to plot in the plane using a scatter plot. The function then overlays example images taken from the columns of the second argument, using the dimensions given in the third argument, onto this scatter plot. For example, to show images randomly in a square you can run `imcloud(rand(size(X,2),2), X, dims)`.
- `imgrid.m` - This helper function takes a matrix of observations (first argument) and the dimensions of each observation (second argument) then displays a grid (with size given by the third argument) of observations. For example, to visualize the data in `cbcl.mat` you can load the data file and run `imgrid(X, dims, [5,5])` to view the first 25 faces of the CBCL face collection, ordered from left-to-right then top-to-bottom.

3 Guide

1. Complete and turn in the function `princomp.m`. Do not use the built-in function for PCA (but do use, e.g., `eigs` or `svd`). Notes: This function removes the mean from the data—this makes computing the covariance matrix easier. `princomp.m` should then simply return the leading eigenvectors in W (and associated eigenvalues λ) of said covariance matrix. You can also easily project the zero-mean data onto these to get lower dimensional representations (the latent variables z_i for each x_i). To later reconstruct data (approximations) from latent data, use z_i and principal components; do not forget to add the mean back, at this time!
2. Load and look at the CBCL faces data set to get a feel for it, for example by using the provided function `imgrid.m` with the raw data (e.g., visualize an array of 25 randomly selected faces).
3. Run PCA on the CBCL faces data set and turn in a figure with the mean face and the first five principal components (leading eigenfaces). Explain why you think they look the way they do, and what sort of information in the data they are modeling. Note: for higher components this becomes more difficult. Explain as much as you can.
4. Project the faces onto the first two principal components and plot/turn in the result as a point cloud in 2D (the best representation of the data in two dimensions), using either `scatter` or `imcloud.m`.

5. Use the `imcloud.m` function to create this plot for projections onto other significant PCs, and explain the results you see as they compare to your response from looking at the PC vectors. How might you change or improve your description of the information captured by the principal components?
6. Plot the spectrum of your PCA: trace the eigenvalues of the covariance matrix in descending order. Does this maybe tell you how many principal components should be selected for "optimal" dimensionality reduction?
7. Reconstruct faces from varying number of principal components. For example, try $M = 1, 2, 5, 10, 25$, and compare the reconstructions against original faces (you can chose the same 25 faces as under item 1, for example, and use `imgrid.m` with the reconstructed data). What do you observe?
8. Follow the same steps as above for the MNIST dataset.

4 Extra

The subset of MNIST we worked with here does not contain the same number of zeros and ones.

1. How many zeros and ones are in this collection? How did you conclude this?
2. Based on your answer above, can you think of a way to automatically deduce the quantity of each digit in a data set if only these two digits are present?