CSCC11 Machine Learning and Data Mining, Fall 2016

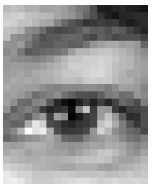
Assignment 4: Principal Components Analysis (13%, with additional 2% bonus)

Due Thursday, Dec 1st, 11:59pm

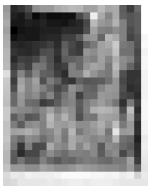
This assignment involves some programming and some thinking. Mainly, your job is to write the code we ask for in a script called EyeDetector.m. You will also write a report. Your report should address any discussion or comments requested of you in the assignment requirements below. It should also include any plots or images requested below, as well as any plots or images that help to explain or justify your comments in the report. You DO NOT need to submit a hardcopy of your report, instead create a PDF document for your report and submit a tar file named `A4.tar` *that includes your report and all Matlab files required to run your script (please don't include the training and test data).*

Introduction. Suppose you are asked to build an algorithm to find eyes in images. A digital image is a two-dimensional array of pixel elements (called pixels) that represents brightness or color. Even small images are high-dimensional objects since the number of rows and columns is typically 1000 or more. For this reason, dimensionality reduction is a common precursor to learning models for detecting specific objects in images.

In this assignment you are given two sets of small image patches. One set contains images of eyes. The other is a random set of image patches which are not eyes (called non-eyes). Here is a typical eye and non-eye images.



Eye Image



Noneye Image

The first goal is to perform PCA on these two image ensembles. Your next main goal is to build a PCA-based classifier capable of distinguishing between eye and non-eye images. You will also compare the performance of this classifier against kNN on the original data, and kNN applied to the low-dimensional PCA representations of the images.

To get started, download the starter file from the course web site. In the file you will find `EyeDetector.m`. This Matlab script provides detailed instructions on what you need to implement (also outlined below). It also specifies a list of printouts and questions that you should include in your report. This assignment also makes use of several helper functions to perform kNN (so you do not need to implement kNN). The first section of `EyeDetector.m` shows a kNN classifier applied to the original training and testing images.

Note: Your printouts and analysis are worth a significant portion of the marks, so be sure to include all required results. And comment on all the questions below and contained in `EyeDetector.m`.

Your Task

1. **PCA.** As described in the comments of `EyeDetector.m`, your first task is to apply PCA to the eye image ensemble and to the non-eye ensemble.

(a) Compute the averages of the eye and non-eye images. Print these out (as images) for your report.

(b) Subtract the means from the training images, calculate the two covariance matrices, and then calculate their eigenvectors and eigenvalues. (See Matlab function `eig`).

(c) Sort the eigenvectors based on their associated eigenvalues. Print out the leading 5 eigenvectors (as images) for your report.

(d) The number of PCA basis images to use is given by `PCAcomp` in `EyeDetector.m`). Place these PCA basis images (the eigenvectors) into two arrays called `eyeVec` and `noneyeVec`. These matrices have N rows, and `PCAcomp` columns.

(e) Project each of the test images onto both of the eye and non-eye model matrices. That is, as described in `EyeDetector.m`, project each test image onto each of the two PCA subspaces. This yields low-dimensional PCA subspace coefficients for the model of eye images, and for the model of non-eye images.

(f) Compute the PCA subspace coefficients for each training image projected onto each of the two PCA subspace models.

2. **Compute Classification Rates for a kNN Classifier.** Compute the correct classification rate, and the false-positive rate for the kNN classifier *based on the low-dimensional PCA representations* and compare with these results to those based on the original high-dimensional data (already computed for you earlier in `EyeDetector.m`). Are the results better or worse? Why?

3. **PCA Classifier based on Reconstruction Errors.** Given a point in the low-dimensional PCA subspace, one can use the generative model to reconstruct the corresponding image (i.e., by multiplying the basis matrix with the low-dimensional subspace coefficients and then adding the mean image). If an image reconstructed from the eye subspace model is closer to the original image than that from the noneye model, we'll conclude that the image is an eye image.

   From the eye and non-eye coefficients of each test image, reconstruct two images, one from the eye model and one from the non-eye model. Then, compute the mean squared-errors for each of the two reconstructions. The classification decision is based on the ratio of mean squared errors.

   Print the correct classification rate and the false positive rate for this classifier, and compare them to the low-dimensional kNN classifier. Do this for models with different subspace dimensions; use `PCAcomp` equal to 5, 10, 15, 25 and 50. Plot graphs of the correct classification rates vs the number of PCA components for the two types of classifier. Do the same for the false-positive rate vs the number of PCA components.

Questions

- Which classifier gives the overall best performance?
- What conclusions can you draw about the usefulness of dimensionality reduction?
- Which classifier would you use on a large training set consisting of high-dimensional data?
- Which classifier would you use on a large training set of low-dimensional (e.g. 4D) data? Why?

(Bonus 2% course mark) Clustering by ignoring labels for training. We can ignore the labels for eye vs noneye, and simply perform K-means clustering on the entire collection of data by concatenating eye and noneye images data into one big data matrix (training and testing set together).

Implement K-means with K being 2 and apply it in this way. Initialize the two cluster centers randomly to two images in your dataset, and initialize cluster membership randomly.

After K-means converges, show the cluster centers as images, and also show a few members from each cluster. Does one cluster correspond to the eye class and the other correspond to the noneye class? If so, use the cluster membership as classification output, what is the correct classification rate on training and testing set? If we want to use probabilistic PCA to model data in each cluster, how can you combine PPCA into a clustering model? How would the learning work on high level?