Homework #1: Principal Component Analysis (PCA) and Canonical Correlation Analysis (CCA) [100 points total]
Due Date: Wednesday, 25 April 2018

Homework policies

CS233 is a technical course, so doing the homework is the only way to acquire a working knowledge of the material presented. We encourage you strongly to start working on the homework problems right away—the problems in this assignment, as well as those to follow, require both some mathematical understanding and some computational data analysis, so you are unlikely to be able to solve them if you wait until the evening before the due date.

Collaboration in solving the problems is encouraged in this class—you have a lot to learn from your fellow students. However, in order to make grading the homeworks a meaningful way to measure your effort and your understanding of the material, we allow collaboration groups of at most three students. A single write-up is sufficient for a collaboration group.

Please be sure to respect the honor code in all assignments: the work you present must be your own, or obtained jointly with your team partner. Other than that, it is not permitted to give or receive help from others in doing the assignments. Please be sure to reference all sources and resources used in obtaining your solution, as you would when publishing a scientific paper.

It is very important in this course that every homework be turned in on time. We recognize that occasionally there are circumstances beyond your control that prevent an assignment from being completed on time. You will be allowed two classes of grace during the quarter. This means that you can either hand in two assignments each late by one class, or one assignment late by two classes. Any further assignments handed in late will be penalized by 20% for each class that they are late and by 100% after two late periods, unless special arrangements have been made previously with the instructor or the CAs. However, no assignment may be handed in past the last class in CS233, on 6 June 2018.

Problem 1. PCA on Images of Faces [60 points]

In this problem we will consider two tasks. First, we will explore the efficiency of PCA as a tool for dimensionality reduction and compression. Then, we will utilize PCA for constructing a rudimentary face recognition algorithm. Although, current state-of-the-art systems use deep learning for the latter task, it is still appealing to see how very basic techniques, like PCA, perform in this setting. If interested, you can read the original paper[1]. We have prepared a dataset with images portraying human faces (YaleFace) and some MATLAB starter code for you. YaleFace is comprised by the images of the faces of 15 individuals. For each individual there are 11 images taken under a variety of conditions e.g., the person makes a happy expression, wears glasses etc. Finally, the total of 165 images has been split into a training and test dataset. Please follow the instructions below on how to proceed.

Figure 1: Sample face images from Yale face dataset.

(a) (5 points) Load the YaleFace dataset on MATLAB. Write a script to visualize the mean images of all the images labeled as 'happy' and 'sad' respectively. Also, generate and plot the mean image corresponding to all the training images of the collection.

(b) (15 points) Perform PCA on the training images viewed as points in a high-dimensional space (using their pixel values). Plot a curve displaying the amount of "energy" captured by the first k principal components, where energy is the cumulative sum of top-k components' variances, divided by the sum of all the variances. How many components do we need in order to capture 50% of the energy? How much of the energy is captured with $k = 25$?

(c) (5 points) Visualize the previously discovered top 25 eigenfaces (eigenvectors obtained from PCA). Order them according to the magnitudes of their corresponding eigenvalues and plot them in a single figure.

(d) (10 points) Use the eigenfaces to reconstruct the first image of the cell array containing the training images. Vary the number of principal components used for the reconstruction

[1]Turk, Matthew A., and Alex P. Pentland. "Face recognition using eigenfaces." Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on. IEEE, 1991.

and compare the quality of the resulting images. Visualize the reconstructed image using $1, 10, 20, 30, 40, 50$ components. How many components do we need to achieve a visually good result? If we "clean up" the dataset (remove all images with 'leftlight', 'rightlight' labels and all images with individuals wearing glasses from the training set), can we use less components for visually equal good results? Why? Plot the same set of reconstructed images using eigenfaces from "cleaned" training set.

(e) (15 points) Let us now try to recognize the identity of a person's face in a previously unseen image. Load an image from the test set, subtract from it the mean of the training images and project it to the previously computed top-25 principal components. Then, use a nearest neighbor search to find its closest image in the training set. If the nearest neighbor found depicts the face of the same person as the one of the unseen image, consider this as a successful discovery of the person's identity. Repeat this experiment for all the test images and report the mean accuracy on the entire test data set. What is the accuracy for test images labeled as 'leftlight' or 'rightlight'? Make comments on the test images that are mistakenly identified.

(f) (10 points) What happens when we try the previous approach with an input image that does not portray a face? Load the provided image of a car, project it to the PCA subspace, and then reconstruct it via the eigenfaces. Compare the reconstructed image with the original image. How can we design a simple face vs. non-face classifier based on this observation?

Problem 2. CCA for Word Appearance Compression [40 points]

CCA (Canonical Correlation Analysis) focuses on the discovery of linear associations between multi-dimensional variables. Two-View CCA, the method covered in class, is the application of CCA on a single pair of sets of such variables. Concretely, it aims to find two distinct subspaces, where the projection of the corresponding original variables are maximally correlated.

In this problem we explore how to use CCA for feature aggregation and compression in image data, even when nuisance variables make the images visually different. Intuitively, the CCA projections "undo" the spurious variability in the original variables, providing a new representation where their underlying similarity is easier to detect. This representation captures the shared properties of the two original signals and thus offers a way of describing them more compactly by, for instance, using a simple statistic, such as their mean. When it comes to multiple variables, we extend the two-view CCA to the multi-view CCA and, by similar arguments, compress the original signals using the average of their projections.

The specific application that follows is concerned with the compression of a database of printed word images used for text recognition of word images captured by a smartphone camera[2].

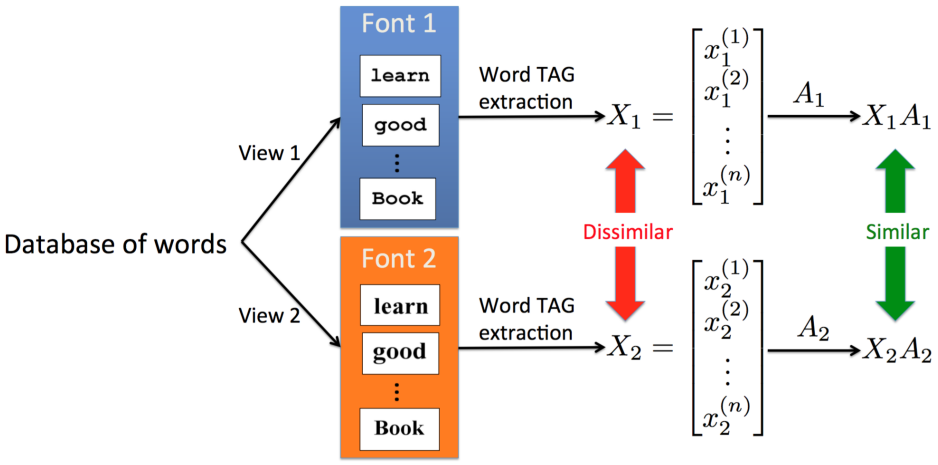

Figure 2: CCA is used here to remove some of the variation in word appearance caused by different fonts. In the figure, $x_j^{(i)}$ denotes the feature vector (called TAG in the original thesis) of the i-th word in the j-th font. X_1 and X_2 correspond to database word TAGs of font 1 and font 2 respectively. X_1 and X_2 are dissimilar due to the font variation. However, after applying the CCA transformation matrices A_1 and A_2, the database features of different fonts can be made more similar.

(a) (10 points) In two-view CCA as in Fig 2, we assume we have variables X_1 and X_2 where $X_i \in R^{n \times D}$, represent zero-mean features of word image patches in the i-th font ($i = 1, 2$),

[2]Huizhong Chen, Visual Word Recognition with Large-Scale Image Retrieval, Stanford University, August 2015. PhD Thesis.

with n being the number of distinct words and D the features' dimension. Assume you are given two projection vectors a_1 and a_2, where $a_i \in R^{D\times 1}$. What is the correlation coefficient: $\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$, of the projected variables $X_1 a_1$ and $X_2 a_2$?

(b) (10 points) Two-view CCA only works with two variables — however, here we want to aggregate features of multiple fonts in one cluster. We need to extend the two-view CCA its multi-view counterpart. Mathematically, in K-view CCA, we want to find K projections $a_1, ..., a_K$ such that thew pairwise correlations of $X_1 a_1, ..., X_K a_K$ are maximized:

$$\begin{aligned} \underset{a_1,...,a_K}{\text{maximize}} & \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} a_i^T \Sigma_{ij} a_j \\ \text{subject to} \quad & a_i^T \Sigma_{ii} a_i = 1 \end{aligned} \tag{1}$$

However, this optimization problem is not easy to solve. We can relax it to the following form:

$$\begin{aligned} \underset{a_1,...,a_K}{\text{maximize}} & \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} a_i^T \Sigma_{ij} a_j \\ \text{subject to} \quad & \sum_{i=1}^{K} a_i^T \Sigma_{ii} a_i = K \end{aligned} \tag{2}$$

The Lagrangian of the above optimization problem is formulated as:

$$\mathscr{L} = \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} a_i^T \Sigma_{ij} a_j - \frac{\lambda}{2} (\sum_{i=1}^{K} a_i^T \Sigma_{ii} a_i - K) \tag{3}$$

Show how by taking derivatives of the Lagrangian, we can derive closed form formulas to compute a_i for $i = 1, ..., K$. Hint: try to arrive at a generalized eigenvalue problem: $A\boldsymbol{v} = \lambda B\boldsymbol{v}$.

(c) (5 points). Show that when $K = 2$ (a special case for multi-view CCA) the solution from (a) is consistent with what has been derived for standard CCA in class.

(d) (15 points). Extend the provided code for implementing multi-view CCA in MATLAB. Given $X_1, X_2, ..., X_K$, your job is to construct and solve the generalized eigenvalue problem of previous question and return the projection vectors for each view. Use these vectors to aggregate the features like:

$$Y = \frac{\sum_{i=1}^{K} X_i A_i}{K} \in R^{n\times d}. \tag{4}$$

Another way to compress the database is to directly average the feature vectors, i.e. $\tilde{Y} = \frac{\sum_{i=1}^{K} X_i}{K}$. We will compare our aggregation using multi-view CCA and direct feature averaging using the metric of word recognition accuracy. We have prepared code

for nearest-neighbor word query on a small test set. Compute and report recognition accuracy using Y and $\tilde{Y}$. Which method has higher accuracy? What is the database compression rate for each of them? Reduce the CCA dimension d to 200 and 100 — what is the recognition accuracy and compression rate in these two cases?