1 Principal Component Analysis (30 points)

1.1 Deriving PCA in terms of minimum reconstruction error (15 points)

In class, we have derived PCA formulation in terms of maximum variance of the projection. Here we would like to derive PCA from another perspective. Given a set of data $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$ where $\boldsymbol{x}_i \in \mathbb{R}^D$ and $\sum_i \boldsymbol{x}_i = \boldsymbol{0}$, we project the data into a d-dimensional subspace. Let $\boldsymbol{U} \in \mathbb{R}^{D\times d}$ be the basis of the subspace, i.e. $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_d$, and $\boldsymbol{z}_i \in \mathbb{R}^d$ represents the corresponding coordinates of $\boldsymbol{x}_i$ in the subspace. Now we are able to reconstruct $\boldsymbol{x}_i$ by $\hat{\boldsymbol{x}}_i = \boldsymbol{U}\boldsymbol{z}_i$. Our goal is to find optimal $\boldsymbol{U}$ and $\{\boldsymbol{z}_i\}$ such that the *reconstruction error* $\sum_{i=1}^N \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2$ is minimized.

(a) (5 points) Show that $\boldsymbol{z}_i = \boldsymbol{U}^\top \boldsymbol{x}_i$ minimizes the reconstruction error when $\boldsymbol{U}$ is fixed.

(b) (10 points) Show that the optimal $\boldsymbol{U}$ leads to the PCA solution, i.e. $\boldsymbol{U}$ is the eigenvectors of the covariance matrix.

1.2 Projecting a Gaussian distribution (15 points)

Assume random variables $\boldsymbol{x} \in \mathbb{R}^D$ follow a Gaussian distribution, $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$. Let $\boldsymbol{p} \in \mathbb{R}^D$ be a direction which projects $\boldsymbol{x}$ into one-dimensional space $z = \boldsymbol{p}^\top \boldsymbol{x}$. We would like to find $\boldsymbol{p}$ such that the entropy of z is maximized.

(a) (10 points) Derive the optimal solution $\boldsymbol{p}^*$.

(b) (5 points) Show that $\boldsymbol{p}^*$ also maximizes the variance of z.

2 Hidden Markov Models (30 points)

In this problem, you will use an HMM to decode a simple DNA sequence with components A, C, G, T. Assume there is one hidden variable X that controls the generation of DNA sequence. X takes two possible states S_1, S_2. Assume the following probabilities for HMM $\theta = \{\pi_i, a_{ij}, b_{ij}\}$:
Transition probabilities a_{ij} (i.e. $P(X_{t+1} = S_j | X_t = S_i, \theta)$):

$$P(S_1|S_1) = 0.7, P(S_2|S_1) = 0.3,$$
$$P(S_1|S_2) = 0.3, P(S_2|S_2) = 0.7.$$

Emission probabilities e_{ij}:

$$P(A|S_1) = 0.4, P(C|S_1) = 0.1, P(G|S_1) = 0.4, P(T|S_1) = 0.1,$$
$$P(A|S_2) = 0.1, P(C|S_2) = 0.4, P(G|S_2) = 0.1, P(T|S_2) = 0.4$$

and initial state distribution π_i :

$$\pi_1 = P(X_1 = S_1) = 0.5,\ \pi_2 = 0.5$$

Assume the observed sequence is $e = CGTCAG$:

(a) (5 points) Calculate the probability of the sequence e for the HMM θ, i.e. $P(e|\theta)$.

(b) (10 points) Calculate the probability for the hidden state X_t to be S_i for each time step individually, i.e. $P(X_t = S_i|e, \theta)$ for t = 1,...,6

(c) (5 points) Calculate the most likely path of hidden states X_t for emissions e, i.e. $\arg\max_x P(x|e, \theta)$, where x is a sequence of hidden states X_t, i.e. $x = \{X_1, ...X_T\}$. The sequence of most likely states estimated independently does not necessarily form the most likely path. Does it happen in this example?

(d) (10 points) Predict the most likely emitted symbol at the end of sequence e, i.e. $p(e_{T+1}|e, \theta)$, with $T = 6$.

Show your work for full credit.

3 Quiz #2 - Sample questions (0 points)

Note: the sample questions are just for practices and will not be graded.

3.1 Expection Maximization

Consider the following Gaussian mixture model (GMM)

$$p(\boldsymbol{x}) = \sum_{k=1}^{N} w_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \sigma \boldsymbol{I})$$

where $\boldsymbol{I}$ represents the identity matrix. Note that we are assuming all covariance matrices are the same as $\sigma \boldsymbol{I}$.
We will use the EM algorithm to estimate the paramters w_k, $\boldsymbol{\mu}_k$ and σ.

(a) Write down the E-step; simplify (as much as possible) the quantity that is being computed.

(b) Write down the M-step, i.e. give the expression of the parameter update. (You do not need to produce the proof or even have to derive - you can use your intuition, such as when we first described EM for GMMs. Your parameter update, however, needs to be correct).

3.2 Dimensionality Reduction

We have learned the technique of PCA, a method of reducing $\boldsymbol{x} \in \mathbb{R}^D$ to $\boldsymbol{y} \in \mathbb{R}^d$ where $d < D$. Note that in PCA the directions are computed as to maximize variances of the projected data points onto those directions. I am proposing a new algorithm called Coordinate Component Analysis (CCA) [Not to be confused with Canonical Correlation Analysis]. The algorithm goes as follows:

- For each dimension of the data, I compute the variance in that direction.
- I then sort descendingly the variances and pick the top d dimensions corresponding the largest d variances.
- I then throw away the remaining $D - d$ dimensions. I have thus reduced the data to d-dimensions.

Answer the following questions:

(a) What is the difference between CCA and PCA?

(b) In what cases, CCA and PCA yield same results and why so? (i.e. the projection directions are precisely those d dimensions)

(c) Is there any advantage or disadvantage of using CCA? If so, in what cases?

4 Programming (PCA) (40 points)

Face recognition is an important task in computer vision and machine learning. In this part you will implement a classical method called Eigenface. You will use face images from the `Yale Face Database B` (http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html) which contains face images from 10 people under 64 lighting conditions.

(a) **Dataset**. Download the data file `face_data.mat` from Blackboard. It contains three sets of variables:

- `image`: each element is a face image (50×50 matrix). You can use matlab function `imshow` to visualize the image. The data is stored in a cell array.
- `personID`: each element is the ID of the person, which takes values from 1 to 10.
- `subsetID`: each element is the ID of the subset which takes values from 1 to 5. Here the face images are divided into 5 subsets. Each subset contains face images from all people with certain lighting conditions. Note that the lighting conditions in different subsets are different.

(b) (10 points) **Implement PCA**. Please fill in the function `pca_fun` in `pca_fun.m` file (in Blackboard). The function takes the data matrix (each row being a sample) and target dimensionality d (lower than or equal to the original dimensionality) as the input, and outputs the eigenvectors.

(c) (15 points) **Obtain Eigenfaces**. Take each 50×50 training image and vectorize it into a 2500-dimensional vector. Perform PCA on all vectorized face images, and retrain the first d eigenvectors. These eigenvectors are called *eigenfaces* (when displayed as images). Please display the top 5 eigenfaces (use `imshow`) in your report.

(d) (15 points) **Classification**. First project each image into the eigenspace to obtain a d-dimensional vector. For each $d \in \{20, 50, 100, 200\}$, train a classifier and report its classification performance. The classification performance is evaluated using the *leave-one-subset-out* strategy: treat each subset as the test set and the remaining four subsets as the training set, and then report the average accuracy. Please experiment with both linear SVM and RBF kernel SVM (use LIBSVM toolbox http://www.csie.ntu.edu.tw/~cjlin/libsvm/). For linear SVM, the tuning parameter is C; for RBF kernel SVM, the tuning parameters include both C and g. Note that when tuning these parameters, you should also apply the *leave-one-subset-out* strategy on the training set instead of traditional cross validation strategy (which randomly divides the data into subsets). To summarize, what you should report include (a) optimal parameters for linear and kernel SVMs (b) average test accuracy, for each $d \in \{20, 50, 100, 200\}$.

5 Programming (HMM) [Bonus Question, 30 points]

In this exercise you will implement an expectation-maximization algorithm, the Baum-Welch algorithm, for HMMs in Matlab. Please refer to the textbook ([MLaPP] 17.5.2) for the description of the algorithm. You should **not** use any of the built-in functions related to HMMs.

(a) **Dataset.** Download the data file `hmm_data.mat` from Blackboard. It contains a 1000×6 matrix with 1000 sample trajectories of length 6 each. Each trajectory contains a sequence of state 1, 2, 3 and 4.

(b) **Implement the Baum-Welch algorithm.** Please fill in the function `baumwelch` in `baumwelch.m`. The function takes the data matrix (each row being a sample) and initial guesses for the transition matrix A and emission probability matrix E, as well as the number of iterations of expectation-maximization as inputs and returns estimates for A and E.

(c) **Obtain parameter estimates.** Using the the function you designed above and the data provided, obtain estimates for A and E. We assume there exist two hidden states, and initial guesses are $A = [0.7, 0.3; 0.3, 0.7]$ and $E = [0.25, 0.25, 0.25, 0.25; 0.25, 0.25, 0.25, 0.25]$ where $E_{ij} = p(\text{state} = j|\text{hidden state} = i)$. Let the algorithm run for 500 iterations. Compare your solution to the results obtained from the built-in function `hmmtrain` (read the Matlab docs for hints on the input format).

6 Submission instructions

You need to provide the followings:

- Provide your answers for all of the problems **in hard copy**. The papers need to be stapled and submitted to the CS front desk. We suggest printing it as double-sided to save papers.
- Submit ALL the code and report via Blackboard. The only acceptable language is MATLAB.
  - For your program, you MUST include the main function called `CSCI567_hw5.m` in the root of your folder. After running this main file, your program should be able to generate all of the results needed for this programming assignment, either as plots or console outputs. You can have multiple files (i.e your sub-functions), however, the only requirement is that once we unzip your folder and execute your main file, your program should execute correctly. Please double-check your program before submitting. You should only submit one `.zip` file. No other formats are allowed except `.zip` file. Also, please name it as `[lastname]_[firstname]_hw5.zip`

Collaboration You may collaborate. However, you need to write your own solution and submit separately. You also need to list with whom you have discussed.