1 Principal Component Analysis

For the following problems, we have N *zero-mean* data points $\mathbf{x}_i \in \mathbb{R}^{D\times 1}$ and $\boldsymbol{S} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}} \in \mathbb{R}^{D\times D}$ is the sample covariance matrix of the dataset.

1.1 Derivation of Second Principal Component

(a) (**5 points**) Let cost function

$$J = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}_i - p_{i1}\mathbf{e}_1 - p_{i2}\mathbf{e}_2\right)^{\mathrm{T}}\left(\mathbf{x}_i - p_{i1}\mathbf{e}_1 - p_{i2}\mathbf{e}_2\right)$$

with $\mathbf{e}_1$ and $\mathbf{e}_2$ are the orthonormal vector basis for the dimensionality reduction, i.e. $\|\mathbf{e}_1\|_2 = 1$, $\|\mathbf{e}_2\|_2 = 1$, and $\mathbf{e}_1^{\mathrm{T}}\mathbf{e}_2 = 0$, and some coefficients p_{i1} and p_{i2}.
Show that $\frac{\partial J}{\partial p_{i2}} = 0$ yields $p_{i2} = \mathbf{e}_2^{\mathrm{T}}\mathbf{x}_i$, i.e. the projection length of data point $\mathbf{x}_i$ along vector $\mathbf{e}_2$.

(b) (**5 points**) Show that the value of $\mathbf{e}_2$ that minimizes cost function

$$\tilde{J} = -\mathbf{e}_2^{\mathrm{T}}\boldsymbol{S}\mathbf{e}_2 + \lambda_2\left(\mathbf{e}_2^{\mathrm{T}}\mathbf{e}_2 - 1\right) + \lambda_{12}\left(\mathbf{e}_2^{\mathrm{T}}\mathbf{e}_1 - 0\right)$$

is given by the eigenvector associated with the second largest eigenvalue of $\boldsymbol{S}$.
λ_2 is the Lagrange Multiplier for equality constraint $\mathbf{e}_2^{\mathrm{T}}\mathbf{e}_2 = 1$ and λ_{12} is the Lagrange Multiplier for equality constraint $\mathbf{e}_2^{\mathrm{T}}\mathbf{e}_1 = 0$.
Hint: Recall that $\boldsymbol{S}\mathbf{e}_1 = \lambda_1\mathbf{e}_1$ ($\mathbf{e}_1$ is the normalized eigenvector associated with the largest eigenvalue λ_1 of $\boldsymbol{S}$) and $\frac{\partial \mathbf{y}^{\mathrm{T}}\boldsymbol{A}\mathbf{y}}{\partial \mathbf{y}} = \left(\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}}\right)\mathbf{y}$. Also notice that $\boldsymbol{S}$ is a symmetric matrix.

1.2 Derivation of PCA Residual Error

(a) (**5 points**) Prove that for a data point $\mathbf{x}_i$:

$$\|\mathbf{x}_i - \sum_{j=1}^{K} p_{ij}\mathbf{e}_j\|_2^2 = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i - \sum_{j=1}^{K}\mathbf{e}_j^{\mathrm{T}}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}\mathbf{e}_j$$

Hint: The most common method to proof a mathematical equation of this flavor is by using mathematical induction. To perform a proof by mathematical induction in this case, first show that the equation above holds for the base case $K = 1$, and then using the assumption that the equation holds for $K = k-1$, show that the equation also holds for $K = k$, for any $k > 1$.
Use the fact that $\mathbf{e}_j^{\mathrm{T}}\mathbf{e}_j = 1$ (length of eigenvector $\mathbf{e}_j$ is 1) and $\mathbf{e}_j^{\mathrm{T}}\mathbf{e}_m = 0$ for $j \neq m$ (eigenvectors are perpendicular each other for square symmetric matrix $\boldsymbol{S}$). Also, use definition $p_{ij} = \mathbf{e}_j^{\mathrm{T}}\mathbf{x}_i$.

(b) (**5 points**) Now show that

$$J_K \triangleq \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i - \sum_{j=1}^{K}\mathbf{e}_j^{\mathrm{T}}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}\mathbf{e}_j\right) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i - \sum_{j=1}^{K}\lambda_j$$

Hint: recall that $\mathbf{e}_j^{\mathrm{T}}\boldsymbol{S}\mathbf{e}_j = \lambda_j\mathbf{e}_j^{\mathrm{T}}\mathbf{e}_j = \lambda_j$

(c) (**5 points**) If $K = D$ principal components are used, there is no truncation, so $J_D = 0$. Use this to show that the error from only using $K < D$ principal components is given by

$$J_K = \sum_{j=K+1}^{D} \lambda_j$$

2 Hidden Markov Model

A simple DNA sequence is $\boldsymbol{O} = \overline{O_1 O_2 \cdots O_T}$, with each component O_i takes from $\{A, C, G, T\}$. Assume it is generated from a Hidden Markov Model controlled by a hidden variable X, which takes two possible states S_1, S_2.

This HMM has the following parameters $\boldsymbol{\Theta} = \{\pi_i, a_{ij}, b_{ik}\}$ for $i, j = 1, 2$ and $k \in \{A, C, G, T\}$:

- Initial state distribution π_i for $i = 1, 2$:

$$\pi_1 = P(X_1 = S_1) = 0.7; \pi_2 = P(X_1 = S_2) = 0.3.$$

- Transition probabilities $a_{ij} = P(X_{t+1} = S_j | X_t = S_i)$ for any $t \in \mathbb{N}^+$, $i = 1, 2$, and $j = 1, 2$:

$$a_{11} = 0.8, a_{12} = 0.2; a_{21} = 0.4, a_{22} = 0.6.$$

- Emission probabilities $b_{ik} = P(O_t = k | X_t = S_i)$ for any $t \in \mathbb{N}^+$, $i = 1, 2$, and $k \in \{A, C, G, T\}$:

$$b_{1A} = 0.4, b_{1C} = 0.1, b_{1G} = 0.4, b_{1T} = 0.1;$$

$$b_{2A} = 0.2, b_{2C} = 0.3, b_{2G} = 0.2, b_{2T} = 0.3;$$

Assume we have an observed sequence $\boldsymbol{O} = \overline{O_1 O_2 \cdots O_6} = AGCGTA$, please answer the following questions with step-by-step computations and explanations.

(a) (**5 points**) *Probability of an observed sequence.* Calculate $P(\boldsymbol{O}; \boldsymbol{\Theta})$.

(b) (**5 points**) *Filtering.* Calculate $P(X_6 = S_i | \boldsymbol{O}; \boldsymbol{\Theta})$ for $i = 1, 2$.

(c) (**5 points**) *Smoothing.* Calculate $P(X_4 = S_i | \boldsymbol{O}; \boldsymbol{\Theta})$ for $i = 1, 2$.

(d) (**5 points**) *Most likely explanation.* Compute $\boldsymbol{X} = \overline{X_1 X_2 \cdots X_6} = \arg\max_{\boldsymbol{X}} P(\boldsymbol{X} | \boldsymbol{O}; \boldsymbol{\Theta})$.

(e) (**5 points**) *Prediction.* Compute $O_7 = \arg\max_O P(O | \boldsymbol{O}; \boldsymbol{\Theta})$.

3 Programming Part

3.1 Principal Component Analysis (25 points)

In this programming assignment, you will be implementing the Principal Component Analysis (PCA) algorithm on MATLAB for data (image) representation compression and then use the compressed representation for classification.

For this purpose, the datasets could be loaded from file `hw6_pca.mat`, which consists of labeled training dataset (`X.train` and `y.train`) and labeled test dataset (`X.test` and `y.test`). The training dataset `X.train` contains 9000 samples (9000 rows), each row being 1 sample. Each sample (each row) is a 16-by-16 grayscale pixel intensity (with possible values between 0 and 255, inclusive), which mostly represent a single digit handwritten number, and can be visualized on MATLAB. For example if you want to visualize the 5438-th handwritten digit in the training dataset, you can use the following commands on MATLAB Command Window:

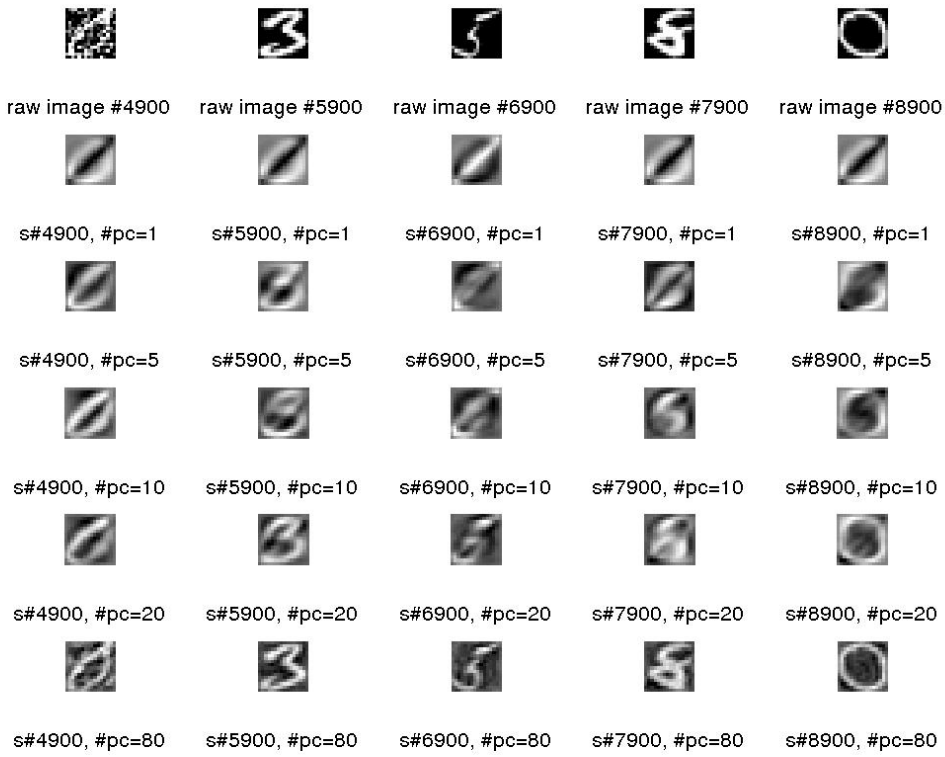
```
load('hw6_pca.mat')
imshow(double(reshape(X.train(5438,:), 16, 16)),[]);
```

which will display a handwritten number '2'. The training label `y.train` contains the ground-truth label of this handwritten digit. The test dataset `X.test` and its label `y.test` are similar correspondingly to `X.train` and `y.train`, but with much lesser samples, 2000 in total.

(a) (**5 points**) To begin with, you will be implementing eigenvectors computation and sorting based on eigenvalues' magnitude in the provided template file `get_sorted_eigenvecs.m`. Please see the description inside the file for more details.

(b) (**5 points**) Each of the computed eigenvectors is a vector $\in \mathbb{R}^{256\times 1}$, thus by itself it can be displayed as an 16×16 image, using the command:
`imshow(double(reshape(eigenvecs(:,i), 16, 16)),[]);`
to display the image of `i`-th eigenvector (let us call it an "eigendigit").
Please plot the top 8 eigendigits, corresponding to the top 8 biggest eigenvalues. You may use MATLAB commands `figure`, `hold on`, `subplot`, `imshow`, `hold off` for this purpose. Report this plot on your `*.pdf` file (as well as the hardcopy).

(c) (**10 points**) Perform data representation compression on the training data. Since `X.train` is $\mathbb{R}^{9000\times 256}$, if you pick top K eigendigits to project into, you will get the compressed representation `X_compressed`, which is $\mathbb{R}^{9000\times K}$. Now, from this compressed representation `X_compressed`, you can reconstruct the data, producing `X_reconstruction`, which is $\mathbb{R}^{9000\times 256}$ again, but somewhat distorted as compared to the original image. The degree of distortion depends on how many eigendigits K that you used to represent the data in compression. The less K you used, the more severe the distortion is. Here is an example image of the reconstruction.



In the picture above, the first row is the raw image drawn from `X.train`. The second to the sixth row are the reconstructed digits, when using $K = 1$, 5, 10, 20, 80 eigendigits (pc stands for principal components, or eigendigits), respectively. The columns from left-to-right corresponds to sample # 4900, 5900, 6900, 7900, and 8900, respectively, drawn from `X.train`. For this part, you need to report similar images like the above, but for sample # 5500, 6500, 7500, 8000, and 8500, respectively, drawn from `X.train`. Again, you may use MATLAB commands `figure`, `hold on`, `subplot`, `imshow`, `hold off` for this purpose.
Report this plot on your `*.pdf` file (as well as the hardcopy).

(d) (**5 points**) In this final part of programming assignment for PCA, we will do classification on the compressed data. Therefore you need to compress both `X.train` and `X.test` in the same manner (using the same number of eigendigits representer). **Do NOT forget to subtract each sample in `X.test` with the mean of `X.train`**, before projecting them to get the compressed representation. For classification, we will just use the simplest and readily-available-on-MATLAB Decision Tree algorithm. You may use either `ClassificationTree.fit` in MATLAB R2013b or `fitctree` in MATLAB R2015b. Use the following command to perform classification:

```
tree = ClassificationTree.fit(train_data,train_label,'SplitCriterion', 'deviance');
train_label_inferred = predict(tree,train_data);
test_label_inferred = predict(tree,test_data);
```

Please report the accuracy of the training prediction and test prediction (similar like in Homework 1), as well the amount of time (in seconds) required to finish the computation in your computer, **for 5 different choices of K (number of top eigendigits used in the representation): 1, 5, 10, 20, 80**. Also, provide the analysis on this results, why the accuracy and computation time differs between different choices of K.
Report both the results and the analysis on your `*.pdf` file (as well as the hardcopy).

3.2 Hidden Markov Model (25 points)

In this programming assignment, you are supposed to implement an HMM model to discover the anomaly patterns in the computer systems.

3.2.1 Data Descriptions

We take part of the *UMN login and ps* dataset from these pages: `http://www.cs.unm.edu/~immsec/systemcalls.htm` and `http://www.cs.unm.edu/~immsec/data/login-ps.html`. Please carefully study the data descriptions in the two web pages.

The training file `hw6_hmm_train.txt` is from one of the *ps normal data* file, and the test files `hw6_hmm_test_normal.txt` and `hw6_hmm_test_trojan.txt` are from the other *ps normal data* file and the *recovered Trojan ps data* file, respectively.

Basically, `hw6_hmm_train.txt`, `hw6_hmm_test_normal.txt`, and `hw6_hmm_test_trojan.txt` contain 15, 9, and 11 traces (time series), respectively, with varied lengths. Notice that the same PID in different files refers to different traces.

3.2.2 Algorithm

Implement the EM algorithm to train a single HMM for all traces using EM algorithm. Set the number of different hidden states S equal to number of different observations O (different system call numbers shown in the training file).

We denote the parameters of HMM as $\Theta = \{\pi_i, a_{ij}, b_{ik}\}$, with initial state distributions π, transition probabilities a_{ij}, and emission probabilities b_{ik}.

3.2.3 Questions

Please answer these questions in your `*.pdf` file report.

(a) (**2 points**) What are the lengths of the longest and shortest traces in the training data? How many different observations?

(b) (**5 points**) Since you are supposed to train a single HMM from multiple time series, the original *Baum-Welch algorithm* is no longer applicable, but a modified EM algorithm still works. Thus you need to compute the log likelihood of the training data $\mathcal{D}$ when you know the hidden states S for all the time series.

Please give the formula to compute the log likelihood $L_{\mathcal{D}}$ of the training data $\mathcal{D}$. It might contains terms related to the HMM parameters Θ, the hidden states S, and the observations O. Please describe the steps and notations clearly.

(c) (**9 points**) Please provide the updating formulas in M-step for π_i, a_{ij}, b_{ik}.

We define the soft counts: $\gamma_t^d(i)$ is the soft count of the hidden state of time series d at time t is i, i.e., $\#\{s_t^d = i\}$, and $\delta_t^d(i,j)$ is the soft count of the hidden states of time series d at time t and $t+1$ are i, j, respectively, i.e., $\#\{s_t^d s_{t+1}^d = ij\}$.

You can introduce other notations for intermediate computation results if necessary. Please describe the steps and notations clearly.

(d) (**9 points**) Based on the update rules you have derived, write a Matlab program to train HMM on the training dataset. Please do not use any existing packages.

Then apply the well-trained model on the two test datasets: For each time series O,

(a) compute its log likelihood $P(O|\Theta)$, and

(b) generate 100 time series samples with the same length of O, and compute the mean and standard deviation of the log likelihood of these samples.

Please report these values in tables. For each time series, you need to report its log likelihood, the sampled log likelihood mean and standard deviation, in total $(9+11)\times 3 = 60$ values.

You are supposed to see difference of the log likelihood values for the normal and trojan(anomaly) time series in this question.

Note: You need to avoid the numerical underflow caused by multiplying many small numbers (probabilities), so you may choose to store and manipulate $\log x$ instead of x in your programs. Then instead of computing $x \times y$, you just compute $\log x + \log y$. However, you need be careful to compute $x + y$ without converting the logarithm number back into original number explicitly. For example, assuming $x > y$, then $\log(x+y) = \log x + \log\left(1 + \exp(\log y - \log x)\right)$.

Submission Instruction: You need to provide the followings:

- Provide your answers to problems 1-2, 3.1(b-d), and 3.2 in `*.pdf` file, named as `CSCI567_hw6_fall15.pdf`. You need to submit the homework in both hard copy (at the collection locker #19 at the PHE building 1st floor by 5pm of the deadline date) and electronic version as `*.pdf` file on Blackboard. If you choose handwriting instead of typing all the answers, you will get 40% points deducted.
- 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_hw6_fall15.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]_hw6_fall15.zip`.

Collaboration: You may collaborate. However, collaboration has to be limited to discussion only and you need to write your own solution and submit separately. You also need to list with whom you have discussed.