COL 774: Assignment 4

Due Date: 11:50 pm, Sunday May 1, 2016. Total Points:

Notes:

- This assignment has a mix of theoretical as well as implementation questions.
- Only the **first three** implementation questions will be graded.
- You are strongly encouraged to try out theoretical questions though they are not graded.
- You should submit all your code as well as any graphs that you might plot. Do not submit answers to theoretical questions.
- Do not submit the datasets.
- Include a **single write-up (pdf) file** which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You should use MATLAB/Python for the first two problems.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you have submitted as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. We will also include submissions from previous years since some of the questions may be repeated. Any cheating will result in a zero on the assignment, a penalty of -10 points and possibly much stricter penalties (including a **fail grade** and/or a **DISCO**).
- Some of the problems below have been adapted from the Machine Learning course(s) offered by various researchers/faculty members (e.g., Andrew Ng at Stanford) at their respective universities.

1. **(18 points) K-means for Digit Recognition**
   In this problem, you will be working with a subset of the OCR (optical character recognition dataset) from the Kaggle website. This data was originally taken from the MNIST digit recognition database, but was converted into an easier to use file format. Each of the images in the dataset is represented by a set of 784 (28×28) grayscale pixel values varying in the range $[0, 255]$. The data was further processed [1] so that it could be easily used to experiment with K-means clustering. The processed dataset contains 1000 images for four different $(1, 3, 5, 7)$ handwritten digits. Each image is represented by a sequence of 157 grayscale pixel values (a subset of the original 784 pixel values). A separate file is provided which contains the actual digit value for each of the images.

   Note that dataset above is similar to the one used in Neural Network learning problem in Assignment 2 but presented differently. Here, we will model the problem of digit recognition as a clustering problem. We will implement the k-means clustering algorithm to categorize each of the images into one of the $k = 4$ digit clusters given the pixel values. Follow the instructions below. Be aware that some of these operations can take a while (several minutes even on a fast computer)!

[1] Pre-processing was made available courtesy one of the machine learning courses offered at the University of Washington.

(a) **(2 points)** Write a program to visualize the digits in the data. Your script should take an example index and display the image corresponding to the gray scale pixel values. You can assume a grayscale value of 0 for the pixel values not present in the file.

(b) **(8 points)** Implement the k-means ($k = 4$) clustering algorithm (as discussed in class) until convergence to discover the clusters in the data. Start from an initial random assignment of the cluster means. You can stop at 30 iterations if you find that the algorithm has still not converged. Report your findings.

(c) **(4 points)** One of the ways to characterize the "goodness" of the clusters obtained is to calculate the sum of squares of distance of each of the data points $x^{(i)}$ from the mean of the cluster it is assigned to i.e. the quantity $S = \Sigma_i(x^{(i)} - \mu_{k_i})^2$ where $x^{(i)}$ denotes the ith data point, k_i is the index of the cluster that $x^{(i)}$ belongs to ($1 \leq k_i \leq k$), k is the total number of clusters and μ_{k_i} denotes the mean of the cluster with index k_i. S essentially captures how close the points within each cluster are (or equivalently, how far points across different clusters are). Presumably, smaller the value of S, better the clustering is. Plot the quantity S as we vary the number of iterations from 1 to 20. Comment on your findings.

(d) **(4 points)** Since we have the true labels of the data in this example, we can plot the error of clustering as we run the K-means algorithm. For each cluster obtained at a given step of k-means, assign to the cluster the label which occurs most frequently in the examples in the cluster. The remaining examples are treated as misclassified. Plot the error (ratio of the number of mis-classified examples to the total number of examples) as we vary the number of iterations from 1 to 20. Comment on your observations.

2. **(22 points) Expectation Maximization**
Consider the Bayesian network given in Figure 1. Assume each of the variables in the network is Boolean valued. Recall that the table associated with each variable node in the network represents its conditional distribution given the values of its parent nodes. In this problem, we will implement the EM algorithm over the Bayesian network given above. You are provided with a training set with $10,000$ examples giving values for each of five variables in the network in the order H, B, L, X, F. Each row represents one example. Some of the values in the data could be missing (denoted by '?'). We have generated 3 different versions of the data corresponding to different amounts of missing values. File train.data corresponds to the case when there are no missing values. File train-m1.data has exactly one missing value in each example. File train-m2.data has one or two missing values in each example. You are also given a test file test.data with 2000 examples and no missing values.

(a) **(4 points)** Calculate the ML (Maximum-Likelihood) estimate of the parameters using the fully observed data (train.data). Output the learned parameters in a tabular format for each of the variables in the network. Calculate the log-likelihood of the test data using the ML estimate of the parameters obtained in the previous step.

(b) **(12 points)** Implement the EM algorithm over the above network. Initialize the parameters by ignoring missing values (as discussed in class). Carefully define what the E and M steps should be. Note that your program should be able to handle the cases when there is a varying number (one or two) of missing values in each example. Note that different examples might have different variables with missing values. E-step should fill-in the missing values accordingly. Carefully define your convergence criteria.
Hint: You may want to start with an implementation which handles only one missing value in each example and then extend it to the case to handle more than one missing value.

(c) (**6 points)** Run the EM algorithm implemented above for each of the training data versions with different amounts of missing values. For each case, report the initial set of parameters as well as the parameters obtained after running EM until convergence. In each case, also report the log-likelihood of the test data using the parameters obtained at convergence. Compare these likelihoods with the likelihood obtained in part(a) of the problem (i.e. with no missing data). Did you expect the likelihoods to be very different from each other? Why or why not? Also, comment on your observations.

3. **Coming Soon!**

4. **(For fun only - No points) Principal Component Analysis**
In this problem, you are given a database of grayscale face images for the task of face recognition (each

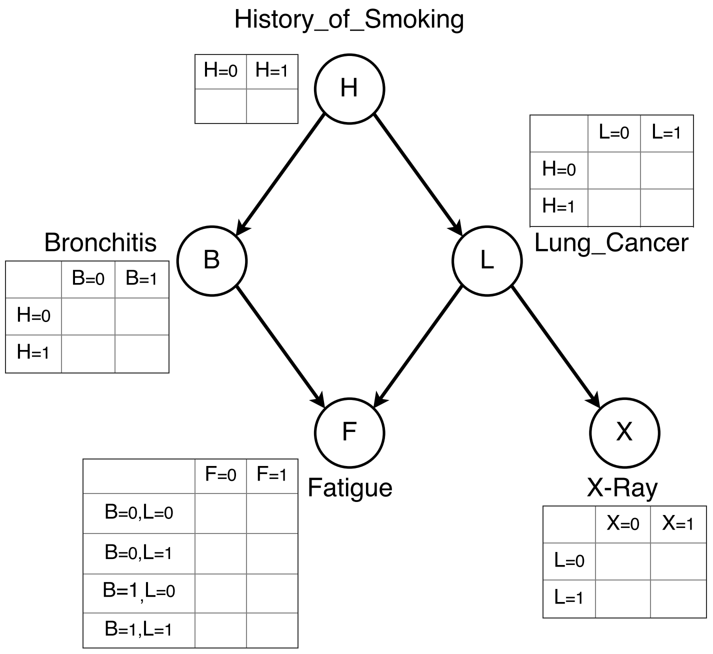

Figure 1: Health Network. Adapted from online Sources

pixel value belongs to the set $\{0, 1, 2, \ldots 255\}$.) [2]. You will apply PCA on the given data which can be seen as a pre-processing step for the recognition task. You should not normalize the data since each pixel value comes from the same scale (0-255).

(a) Calculate the average face for the dataset and display it.

(b) Implement the PCA algorithm using the SVD formulation discussed in class. You should be able to perform SVD in Matlab using a function call (you should not call any function to perform PCA directly but rather do it explicitly using SVD). Note that since data is not normalized to zero mean, you might have to change the co-variance matrix calculation accordingly. Calculate the principal components corresponding to top 50 eigenvalues. You can store the principal components in a single file as a sequence of $361(19 \times 19)$ numbers.

(c) Display the eigenfaces corresponding the top five principal components. You might have to scale the pixel values so that maximum value touches 255.

(d) Write a small program which takes as input a face id in the database, projects it on to the top 50 dimensions and displays the projected image. Try it for a few different faces. How different is the projected image from the original image? Comment on your observations.

(e) Download a few non-faces images from the web. Convert them to grayscale and the size same as those in the face image database. Project these images on the 50 dimensions obtained from PCA. How do the projected images look like? Comment on your observations.

Theoretical Questions - Not Graded.

1. **VC dimension**

   Let the domain of the inputs for a learning problem be $X = R$. Consider using hypotheses of the following form:

$$h_\theta(x) = 1\{\theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_d x^d \geq 0\},$$

   and let $H = \{h_\theta : \theta \in R^{d+1}\}$ be the corresponding hypothesis class. What is the VC dimension of H? Justify your answer.

[2]This is a subset of the dataset available at http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html. The face-image set provided for this problem is exactly the training set of face images given at the above site. You can read the documentation for more details.

[Hint: You may use the fact that a polynomial of degree d has at most d real roots. When doing this problem, you should not assume any other non-trivial result (such as that the VC dimension of linear classifiers in d-dimensions is $d+1$) that was not formally proved in class.]

2. **EM for MAP estimation**

   The EM algorithm that we covered in class was for solving a maximum likelihood estimation problem in which we wished to maximize

$$\prod_{i=1}^{m} p(x^{(i)};\theta) = \prod_{i=1}^{m} \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta)$$

   where the $z^{(i)}$'s were latent random variables. Suppose we are working in a Bayesian framework, and would like to find the MAP estimate of the parameters θ by maximizing

$$\Big(\prod_{i=1}^{m} p(x^{(i)}|\theta)\Big) p(\theta) = \Big(\prod_{i=1}^{m} \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\theta)\Big) p(\theta).$$

   Here, $p(\theta)$ is our prior on the parameters. Generalize the EM algorithm to work for MAP estimation. You may assume that $\log p(x,z|\theta)$ and $\log p(\theta)$ are both concave in θ, and hence, maximizing any linear combination of these quantities is tractable. Show that your M-step is tractable, and also prove that $\prod_{i=1}^{m} p(x^{(i)}|\theta)p(\theta)$ (viewed as a function of θ) monotonically increases with each iteration of your algorithm.

3. **PCA**

   In class, we showed that PCA finds the "variance maximizing"directions onto which to project the data. In this problem, we will look at another interpretation of PCA. Suppose we are given a set of points $\{x^{(1)}, ..., x^{(m)}\}$. Let us assume that we have as usual pre-processed the data to have zero-mean and unit variance in each coordinate. For a given unit-length vector u, let $f_u(x)$ denote the projection of point x onto the direction given by u. I.e., if $\mathcal{V} = \{\alpha u : \alpha \in R\}$, then

$$f_u(x) = arg \min_{v \in \mathcal{V}} ||x - v||^2.$$

   Show that the unit-length vector u that minimizes the mean squared error between projected points and original points corresponds to the first principal component for the data. I.e., show that

$$arg \min_{u:u^T u=1} \sum_{i=1}^{m} ||x^{(i)} - f_u(x^{(i)})||_2^2.$$

   gives the first principal component.

   Remark. If we are asked to find a k-dimensional subspace onto which to project the data so as to minimize the sum of squared distances between the original data and their projections, then we should choose the k-dimensional subspace spanned by the first k principal components of the data. This problem shows that this result holds for the case of $k = 1$.