CSL 341: Assignment 3

Due Date: 11:55 pm, Sunday Nov 16, 2014. Total Points: 86 (+15 extra credit)

Notes:

- This assignment has a mix of theoretical as well as implementation questions.
- Only the 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.
- For problems 2,3 and 5 you should use Matlab. For problems 1 and 4, you are free to use any of the languages in the set {Java, C++, Python and Matlab}.
- 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. Any cheating will result in a penalty of **-7** points on your total course score (in addition to a 0 on the assignment). Stricter penalties (**including a fail grade**) may follow.
- Many of the problems (and the datasets) below have been adapted from the Machine Learning courses offered by Andrew Ng, Carlos Guestrin, Pedro Domingos and possibly other researchers at their respective universities.

1. **(22 points) Decision Trees for Classification** In this problem, we will work with one of the UCI datasets on US congressional voting. Click here to read the data description. The goal is to build a decision tree which would learn a model to predict whether a US congressman (equivalent of a Member of Parliament in India) is democrat or republican based their voting pattern on various issues [1]. The dataset provided to you has been split into 3 disjoint subsets: training data, validation data and test data [2]. For this problem, you may find it helpful to read Chapter 3 of Mitchell's book and/or the original paper on ID3 algorithm by Ross Quinlan (available on the website) in addition to the class notes/slides.

   (a) **(10 points)** Construct a decision tree using the given data to classify a congressman as democrat or republican. Use net error as the criterion for choosing the attribute to split on. In case of a tie, choose the attribute with the lowest index as the splitting attribute. For now, you can treat the missing values ("?") simply as another attribute value. Consider splitting each attribute using a 3-way split

[1] Read more about the distinction between democrats and republicans here.

[2] The splits were made available courtesy one of the machine learning courses offered at the University of Washington

i.e. using the values y/n/"?" [3] Plot the train, validation and test set accuracies against the number of nodes in the tree as you grow the tree. On X-axis you should plot the number of nodes in the tree and Y-axis should represent the accuracy. Comment on your observations.

(b) **(4 points)** Repeat the part above using the information gain as the criterion. Again plot the train, validation and test set accuracies as you grow the tree. Comment on your observations. Is the tree obtained in this part very different from the one obtained in part(a) above? Comment.

(c) **(6 points)** One of the ways to reduce overfitting in decision trees is to grow the tree fully and then use post-pruning based on a validation set. In post-pruning, we greedily prune the nodes of the tree (and sub-tree below them) by iteratively picking a node to prune so that resultant tree gives maximum increase in accuracy on the validation set. In other words, among all the nodes in the tree, we prune the node such that pruning it (and sub-tree below it) results in maximum increase in accuracy over the validation set. This is repeated until any further pruning leads to decrease in accuracy over the validation set. Post prune the tree obtained in step (b) above using the validation set. Again plot the the training, validation and test set accuracies against the number of nodes in the tree as you successively prune the tree. Comment on your findings.

(d) **(2 points)** In the above problem, we simply treated the missing values ("?") as another attribute value. What might be a more principled way of handling the missing data? (You do not have to write any code for this part.)

Note: For this question, you are free to write your code in any of the following languages: Java, C/C++, Python, Matlab. Include appropriate documentation for readability

2. **(20 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 [4] 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)!

   (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) **(10 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, greater 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

[3] Once you implement this basic version, you are free to try more fancy multiple two-way splits e.g. y/others followed by a possible further split on n/"?" at a descendant node.

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

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.

3. **(24 points) Discrminative Vs. Generative Classification** As we had discussed in the class, Naïve Bayes is a generative classifier and logistic regression can be considered its discriminative analog. Both these algorithms make different assumptions about the underlying data. In this problem, we will compare the learning curves for the two algorithms. The learning curve for an algorithm depicts the accuracy of the learned model when plotted against the amount of data used for training. The dataset directory (link on the website) includes a readme file describing the details of the data. For this question, you might find it insightful to read the (Ng & Jordan (2001)) paper posted on the website.

   (a) **(12 points)** Load the data in the file q3-sub.mat. Randomly divide the data in two splits sized 2/3rd and 1/3rd of the total data. Use the larger split (sized 2/3rd) for training and the other one for testing. Implement both Naïve Bayes and Logistic Regression models. Learn the models on the training set and report tha accuracies over the test set. To reduce the variance, you should repeat the above process for 5 different randomly generated splits of the data and report the average accuracies. Note: For Naïve Bayes, use the 1-smoothed estimate. For Logistic Regression, use the learning rate of .00001.

   (b) **(6 points)** Now, learn the two models as you vary the fraction of data used for training in the set [0.01,0.02,0.03,0.05,0.1,0.3,0.5,0.7,0.9,1]. Calculate the accuracy of the two models on the test data for each training fraction of the data. As before, repeat this process for 5 different randomly generated train/test splits of the data. Plot the average accuracy against the amount of data used for training. Comment on your observations.

   (c) **(6 points)** Repeat the above part for the data in the file q3-all.mat. This file has the full set of attributes (for details, look at the readme file). What do you observe? How are the results in this part different from the part above? Comment on your observations.

4. **(20 points) Expectation Maximization** Consider the Bayesian network given in Figure 1 defined over the variables Difficulty(D), Intelligence(I), Grade(G), SAT(S) and Letter (L). 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 D, I, G, S, L. Each row represents one example. Some of the values in the data could be missing (denoted by '?'). We have generated 5 different versions of the data corresponding to different amounts of missing values. File train-x.data corresponds to the case where each example has a missing value with $x/100$ probability e.g., if x is 10, each example has a missing value with 0.1 probability. The data is generated in a manner that each example is allowed to have at most one missing value. You are also given a test data with 2000 examples with no missing values.

   (a) **(4 points)** Calculate the ML (Maximum-Likelihood) estimate of the parameters using the fully observed 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) **(10 points)** Implement the EM algorithm over the above network. Carefully define what the E and M steps should be. 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.

   (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, calculate the log-likelihood of the test data using the parameters obtained at convergence. Plot on a graph with Y-axis depicting log-likelihood of the test data and X-axis depicting amount of missing information. Comment on your observations.

   Note: For this question, you are free to write your code in any of the following languages: Java, C/C++, Python, Matlab. Include appropriate documentation for readability

5. **(Extra Credit: 15 points) Principal Component Analysis** In this problem, you are given a database of grayscale face images for the task of face recognition (each pixel value belongs to the set $\{0, 1, 2, \ldots 255\}$.) [5].

[5] 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.

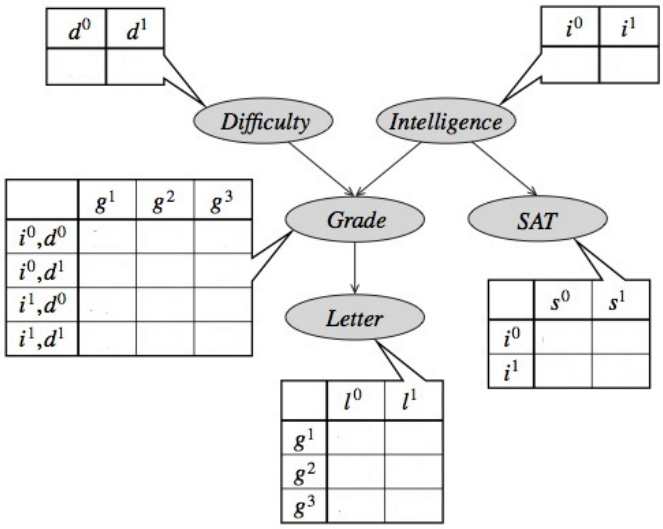

Figure 1: Source: Probabilistic Graphical Models. Daphne Koller and Nir Friedman (2009)

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) **(3 points)** Calculate the average face for the dataset and display it.

(b) **(6 points)** 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) **(3 points)** Display the eigenfaces corresponding the top five principal components. You might have to scale the pixel values so that maximum value touches 255.

(d) **(3 points)** 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) **(Extra fun: no credits)** 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.

Following problems are for your practice and will not be graded.

1. **Decision Trees**

   (a) Give decision trees to represent the following Boolean functions:

      i. $A \wedge \neg B$
      ii. $A \vee [B \wedge C]$
      iii. $A\ XOR\ B$
      iv. $[A \wedge B] \vee [C \wedge D]$

   (b) Consider the set of examples given in Table 1b:

      i. What is the entropy of this collection of training examples with respect to the target classification?
      ii. What is the information gain of a_2 relative to these training examples?

   (c) Given two random variables X and Y, the mutual information between X and Y is defined as $I(X,Y) = H(Y) - \sum_{x_k} H(Y|X = x_k) * P(X = x_k)$. Show that mutual information is symmetric i.e. $I(X,Y) = I(Y,X)$.

Instance	Classification	a_1	a_2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

Table 1: Examples

2. **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.

[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.]

3. **LOOCV and SVM**

(a) **Linear Case.** Consider training an SVM using a linear Kernel $K(x, z) = x^T z$ on a training set $\{(x^{(i)}, y^{(i)}) : i = 1, \ldots, m\}$ that is linearly separable, and suppose that it is noise free (i.e. we use the simplest SVM model without any slack variables). Let $|SV|$ be the number of support vectors obtained when training on the entire training set. (Recall $x(i)$ is a support vector if and only if $\alpha_i > 0$.) Let $\widehat{\epsilon}_{LOOCV}$ denote the leave one out cross validation error of our SVM.
Prove that

$$\widehat{\epsilon}_{LOOCV} \leq \frac{|SV|}{m}$$

(b) **General Case.** Consider a setting similar to in part (a), except that we now run an SVM using a general (Mercer) kernel. Assume that the data is linearly separable in the high dimensional feature space corresponding to the kernel. Does the bound in part (a) on $\widehat{\epsilon}_{LOOCV}$ still hold? Justify your answer.

4. **MAP estimates and weight decay**

Consider using a logistic regression model $h_\theta(x) = g(\theta^T x)$ where g is the sigmoid function, and let a training set $\{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$ be given as usual. The maximum likelihood estimate of the parameters θ is given by

$$\theta_{ML} = arg\max_\theta \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta).$$

Consider using a Bayesian prior $\theta \sim (0, \tau^2 I)$ (here, $\tau > 0$, and I is the $(n + 1) \times (n + 1)$ identity matrix). The MAP estimate is given by

$$\theta_{MAP} = arg\max_\theta p(\theta) \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta).$$

Prove that

$$||\theta_{MAP}||_2 \leq ||\theta_{ML}||_2$$

[Hint: Consider using a proof by contradiction.]
Remark. Above form of prior is sometimes called **weight decay**, since it encourages the weights (meaning parameters) to take on generally smaller values.

5. **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.

6. **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 $k = 1$.