COL 774: Assignment 3

Due Date: 11:50 pm, Sunday April 10, 2016. Total Points: 44 (+5 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 this assignment, you have a choice of using MATLAB/C/C++/Java/Python. If you would like to work with any other language, you need to check with us first.
- 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**).
- Many 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. **(20 points + 5 Extra Credit) Decision Trees for Classification**
   In this problem, we will work with one of the Kaggle competition datasets to predict Titanic survivors. Click here to read the problem description. You have to implement the decision tree algorithm for predicting which passengers survived the tragedy; some groups of people were more likely to survive than others, such as women, children, and the upper-class passengers etc. The dataset provided to you is split into 3 disjoint subsets: training, validation and test data. The dataset has been preprocessed to remove any missing values. It also specifies which attributes to treat as discrete and which ones as numerical. Look at the file feature_info.txt to get a basic understanding of the dataset. You can get more details about the original dataset (and the attributes) from here. In addition to the class notes/slides, Chapter 3 of Mitchell's book is a good reference for this problem. You may also want to read the original paper on the ID3 decision tree learning algorithm by Ross Quinlan (available on the course website).

(a) **(10 points)** Construct a decision tree using the given data to predict whether a passenger survives the incident. The dataset contains a mix of categorical as well as numerical attributes. Preprocess (before growing the tree) each numerical attribute into a Boolean attribute by a) computing the median value of the attribute in the training data (make sure not to ignore the duplicates) b) replacing each numerical value by a 1/0 value based on whether the value is greater than the median threshold or not. Note that this process should be repeated for each attribute independently. For non-Boolean (discrete valued) attributes, you should use a multi-way split as discussed in class. Use information gain as the criterion for choosing the attribute to split on. In case of a tie, choose the attribute which appears first in the ordering as given in the training data. 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) **(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 (a) above using the validation set. Again plot 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. We have posted the relevant sections from Mitchell's textbook for this part on the course website (aceessible only through an internal link inside IIT).

(c) **(4 points)** In Part(a) we used the median value of a numerical attribute to convert it in a 1/0 valued attribute as a pre-processing step. In a more sophisticated setting, no such pre-processing is done in the beginning. At any given internal node of the tree, a numerical attribute is considered for a two way split by calculating the median attribute value from the data instances coming to that node, and then computing the information gain if the data was split based on whether the numerical value of the attribute is greater than the median or not. As earlier, the node is split on the attribute which maximizes the information gain. Note that in this setting, the original value of a numerical attribute remains intact, and a numerical attribute can be considered for splitting in the tree mutliple times. Implement this new way of handing numerical attributes. Report the numerical attributes which are split multiple times in a branch (along with the maximum number of times they are split in any given branch and the corresponding thresholds). Replot the curves in part (a). Comment on which results (Part (a) or Part (c)) are better and why. You don't have to implement pruning for this part.

(d) **(Extra Credits: 5 points)** There are various libraries available for decision tree implmentation in Python, Matlab etc. Use the scikit-learn library of Python to grow a decision tree. Click here to read the documentation of the library and various parameter options to grow the decision tree. Try growing different trees by varying parameters of the learning algorithm. Some exampes of the parameters that you can play with include min_samples_split, min_samples_leaf and max_depth (feel free to vary any other paramters as well). Find the setting of parameters which gives you best accuracy on the validation set. Report the parameter setting giving you best validation set accuracy as well as the validation and test set accuracies for this setting. Explain briefly why the particular parameter setting you obtained works best.

To simplify the task, you are provided with a sample code (decision_tree.py) which shows how to read the data and implement the decision tree using scikit. It requires pandas, scikit-learn and numpy to be installed which you can obtain by searching on google (a command-line installation should also be possible on ubuntu).

Note: You should submit all your code including any code written for processing the data.

2. **(24 points) Newsgroup Classification**
In this problem, we will use the naïve Bayes algorithm to learn a model for classifying an article into a given set of newsgroups. The data and its description is available through the UCI data repository. The original data is description available here. We have processed the data further to remove punctuation symbols, stopwords etc. The processed dataset contains the subset of the articles in the newsgroups rec.* and talk.*. This corresponds to a total of 7230 articles in 8 different newsgroups. The processed data is made available to you in a single file with each row representing one article. Each row contains the information about the class of an article followed by the list of words appearing in the article.

(a) **(10 points)** Implement the Naïve Bayes algorithm to classify each of the articles into one of the newsgroup categories. Randomly divide your data into 5 equal sized splits of size 1446 each. Make sure NOT TO create the splits in any particular order (e.g. picking documents sequentially) otherwise it may lead to skewed distribution of classes across the splits. Now, perform 5-fold cross validation (train on each possible combination of 4 splits and the test on the remaining one). Report average test set accuracies.

Notes:

- Make sure to use the Laplace smoothing for Naïve Bayes (as discussed in class) to avoid any zero probabilities.
- You should implement your algorithm using logarithms to avoid underflow issues.
- You should implement naïve Bayes from the first principles and not use any existing Matlab modules.

(b) **(2 points)**What is the accuracy that you would obtain by randomly guessing one of the newsgroups as the target class for each of the articles. How much improvement does your algorithm give over a random prediction?

(c) **(2 points)** Some (4% in the original data) of the articles were cross-posted i.e. posted on more than one newsgroup. Does it create a problem for the naïve Bayes learner? Explain.

(d) **(6 points)** For each of the train/test splits in part (a) above, vary the number of articles used in training from 1000 to 5784 (training split size) at the interval of 1000 and evaluate on the test split (sized 1446). Plot on a graph the train as well as the test set accuracies (y-axis) averaged over the 5 random splits, as you vary the number of training examples (x-axis). This is called the learning curve. What do you observe? Explain.

(e) **(4 points)** Read about the confusion matrix. Draw the confusion matrix for your results in the part (a) above. Which newsgroup has the highest value of the diagonal entry in the confusion matrix? Which two newsgroups are confused the most with each other i.e. which is the highest entry amongst the non-diagonal entries in the confusion matrix? Explain your observations. Include the confusion matrix in your submission.

Note: You should submit all your code including any code written for processing the data.

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

1. **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.