CSL465/603– Machine Learning
Lab 1

Due on 19/8/2016 11.55pm

Instructions: Upload to your moodle account one zip file containing the following. Please do not submit hardcopy of your solutions. In case moodle is not accessible email the zip file to the instructor at ckn@iitrpr.ac.in. Late submission is not allowed without prior approval of the instructor. You are expected to follow the honor code of the course while doing this homework.

1. **You are allowed to work in pairs for this lab. However, this does not mean only one team member works on the lab, while the other only gets to add his/her name. Both the team members should know the inner workings of the code. I and the TAs reserve the right to question you if required. If you are found unable to explain the code, you will receive no points for the lab.**

2. This lab must be implemented in C++ or Java.

3. A neatly formatted PDF document with your answers for each of the questions in the homework. You can use latex, MS word or any other software to create the PDF.

4. Include a separate folder named as 'code' containing the scripts for the homework along with the necessary data files. Ensure the code is documented properly.

5. Include a README file explaining how to execute the scripts.

6. Name the ZIP file using the following convention rollnumberhwnumber.zip

---

Decisions Trees and Forests

This homework will help you gain a better understanding of decision tree based inductive classifier. You will be implementing the ID3 decision tree algorithm that we discussed in the class. Accompanying in the zip folder is the dataset that you can use for conducting the experiments of this lab. This is the only set of instances that will be provided to you for training the algorithms. This is the Insurance Company Benchmark (COIL 2000) data set [1, 2].

The important step of the ID3 algorithm involves choosing attribute for creating the decision function at every node in the decision tree. Use information gain as the measure to select the *best* attribute for splitting a node. Also implement a functionality of given a text file containing test instances, the code should classify the test data and output the prediction accuracy.

1. Split the data set into a training and a test set. Create a training set containing a random sample of 1000 observations, and a test set containing the remaining observations. Use the training set to learn a decision tree. Discuss the statistics of the learned tree, for example, effect of early stopping on the number of terminal nodes, effect of early stopping on the prediction accuracy on the training dataset and the left out test dataset, attributes that are most frequently used as split functions in the internal nodes of the tree, etc.
2. Let us add some noise to the dataset and observe its effect on the decision tree. Add 0.5, 1, 5 and 10% noise to the dataset. You can add this noise by randomly switching the label of a proportion of data points. Construct the decision tree for each noise setting and quantify the complexity of the learned decision tree using the number of nodes in the tree. Document and discuss your observations about the quality of the model learned under these noise conditions.
3. Produce a pruned tree corresponding to the optimal tree size by computing the prediction accuracy on the test set. Use a post-pruning strategy to prune the tree that has been learned without applying any early stopping criteria. Document the change in the prediction accuracy as a function of pruning (reduction in the number of nodes or rules/clauses obtained from the tree).
4. Learn a decision forest that used feature bagging. Use majority voting for predicting the label for a test data point. Study the effect of number of trees in the forest on the prediction accuracy of the test data set. Document and discuss your results.

Input Format:

The input to the code is the name of the file containing the dataset and the experiment number. There are four experiments to be conducted as listed above. The format of the dataset is as follows

N

D

$x_{11}\ x_{12}\ \ldots x_{1D}\ \ y_1$

$x_{21}\ x_{22}\ \ \ldots\ \ x_{2D}\ y_2$

$\vdots$

$x_{N1}\ x_{N2}\ \ldots x_{ND}\ \ y_D$

N- total number of data points, D – number of attributes describing the dataset. This is followed by the dataset itself, one instance in one line with the attribute values delimited by space. The last value in each line is the value of the target attribute (y).

We should be able to execute your code on institute linux machines in this format.

Executable *filename exptno*

Where **Executable** is the name of the executable file, *filename* is the name of the file containing the data and *exptno* is the experiment number (1-4).

Output Format:

The output of your code should closely match the results presented in your report. The results for the different runs must be grouped together. For example, if for experiment 1, you are determining attributes that are most frequently used as split function in the internal nodes of the tree, the output should closely resemble

Number of times an attribute is used as the splitting function

$x_1\ \ count_1$

$x_2 \quad count_2$
⋮
$x_D \quad count_D$
where *Number of times an attribute is used as the splitting function* is the title of the study, x_1 is the attribute and $count_1$ is the number of times it was used as a splitting function.

Another example is for experiment 4 where you would like to compute the prediction accuracy on the training and test set as a function of number of trees in the decision forest. Here the output should closely resemble
Effect of number of trees in the forest on train and test accuracies
$ntrees_1 \; tr - acc_1 \; ts - acc_1$
$ntrees_2 \; tr - acc_2 \; ts - acc_2$
⋮
$ntrees_n \; tr - acc_n \; ts - acc_n$
where *Effect of number of trees in the forest on train and test accuracies* is the title of the study $ntrees_1$ is the initial number of trees in the forest, $tr - acc_1$ and $ts - acc_1$ are the training and test accuracies obtained using this learned decision forest model respectively.

An important aspect of machine learning is reproducibility of the results presented in a paper/report. Therefore, we will run your code to see if the results are closely matching with what you have presented in the report. Any deviation beyond a reasonable threshold will be considered as fudging of results and will invite severe penalty.

Reference

[1] P. van der Putten and M. van Someren (eds). CoIL Challenge 2000: The Insurance Company Case. Published by Sentient Machine Research, Amsterdam. Also a Leiden Institute of Advanced Computer Science Technical Report 2000-09. June 22, 2000.

[2] https://archive.ics.uci.edu/ml/datasets/Insurance+Company+Benchmark+(COIL+2000)