CSL 302 – Artificial Intelligence

Project 3

You are allowed to work in pairs for this project. The teams have been predetermined. Please check the teams.pdf for the team members. In the pdf file that you submit, include the breakup of contributions made by each team member towards this project. The overall score for every student will depend on their individual contributions. This project is due on April 14th 2014, 11.59pm. Late submission is not allowed without the prior approval of instructor. You are expected to follow the honor code for the course while working on this homework.

Submission Instructions:

Email one zip file containing the following to ckn@iitrpr.ac.in.

1. 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.
2. For homework/projects that contain problems that require programming, include a separate folder named as “code” containing the scripts for the homework along with the necessary data files. Include a README file explaining how to execute the scripts.
3. Name the ZIP file using the following convention:
   rollnumber1_rollnumber2_p(number).zip.
4. Please **do not** submit a hardcopy of your solutions.

In this project you will be experimenting with ID3 decision trees and multi-layer perceptron that we discussed in the class.

Part I

In this part of the project you will be implementing the ID3 decision tree learning algorithm. The pseudo code for the algorithm is provided in Table 1.

1. Download the mushroom[1] dataset from the UCI repository. Import the dataset into the working memory of your code. [2 points]
2. The important step of the ID3 algorithm involves choosing attribute for creating the decision function at every node in the decision tree. The *best* attribute is the one with the highest information gain. Implement a function that takes as input an attribute index and a set of examples and computes the information gain measure for the attribute. [5 points]
3. Implement the ID3 algorithm to build the decision tree. [13 points]
4. Implement the ParseID3 function. Given a data point and a decision tree, this function parses through the decision tree and predicts the label of the data point. [7 points]

[1] http://archive.ics.uci.edu/ml/datasets/Mushroom

5. You will be running a 5 fold cross validation procedure for experimenting with the decision tree. The train and test splits for each run will contain 80% and 20% of the original dataset respectively. Ensure that the splits are mutually exclusive and stratified. Compute the accuracy for each run. [3 points]
6. Lastly, add some noise to the dataset and observe its effect on the decision tree. Add 1, 5, 10 and 15% noise to the dataset. You can add noise by randomly switching the label for a portion of data points. Now compute the accuracy for each of the noisy datasets. Comment on changes to the performance and structure of the decision trees. [10 points]

```
ID3 ($Example, Target\_attribute, Attributes$)
$Examples$ is the training set, $Target\_attribute$ is the attribute whose values us to be predicted by
the tree. $Attributes$ is a list of other attributes that may be tested by the learned decision tree. It
returns a tree that correctly classifies the given Examples.
    1. create a Root node for the tree
    2. If all $Examples$ are positive, Return single-node tree Root, with label = +
    3. If all $Examples$ are negative, Return single-node tree Root, with label = -
    4. Otherwise Begin
        a. $A \leftarrow$ the attribute from $Attributes$ that best classifies $Examples$
        b. The decision attribute for $Root \leftarrow A$
        c. For each possible value , $v \in values(A)$
            i. Add a new tree branch below Root, corresponding to the test $A = v$
            ii. Let $Examples_v$ be the subset of $Examples$ that have the value $v$ for $A$
            iii. If $Examples_v$ is empty
                A. Then below this new branch add a leaf node with label = most
                   common value of $Target\_attribute$ in $Examples$
            iv. Else
                A. below this new branch add the subtree
                   ID3($Examples_v, Target\_attribute, Attributes - \{A\}$)
```
Algorithm 1: ID3 Decision Tree Learning Algorithm

Part 2

In this part of the project you will implement a multilayer perceptron (MLP). Use the back propagation algorithm to update the connection weights of the output and hidden layer nodes.

1. Use the *p3mlpdata* file in the zip folder as the training data for this part of the project. Each row in this file corresponds to a data point. The last column corresponds to the class label associated with the data point. Import this dataset into the working memory of your program. [1 point]
2. You will notice that this is a multi-class dataset. The number of nodes in the output layer of the MLP will correspond to the number of classes. You will have to transform the symbolic labels into a binary representation to be able to use it while training the MLP. [1 point]
3. Implement the MLP training algorithm. The number of input nodes corresponds to the dimension of the data points. [15 points].
4. This will be the only set of data that will be made available to you for building the MLP model. Hence use stratified multi-fold cross validation to measure the generalizability of the learned model. Use classification accuracy as the performance measure. [3 points]
5. Plot the training accuracy and test accuracy as a function of number of learning epochs. [5 points]
6. Experiment with different number of hidden layer nodes and different values of the learning parameter. Select a wide range for both the parameters. Study their effect on the overall performance of the network. [15 points]

7. **Bonus Question** Based on the above experiments, design an MLP that will be learned from all the data that has been made available to you. You will be provided with a true test data (no labels) two days before the project is due. You will have to use the trained MLP to predict the labels for the test data. Include a text file with the predicted labels for the test data points as part of the submission. You will be awarded points relative to the best performance achieved by the class. [Maximum of 10 points].

You can implement the game in C, C++, Java, R, Python, MATLAB or any other language in which you are proficient. However, the program needs to be able to be compiled and run on Linux machines. Do not download and plagiarize code from the Internet.