PROJECT REPORT

INTRODUCTION

Neural Networks have very good generalization capability. The most important weakness of Neural Networks is that they are like black boxes. Understanding the reasoning behind a Neural Network's output is not easy. Knowledge acquired by a Neural Network is represented by its topology, by the weights on the connections and by the activation functions of the hidden and output nodes. These representations are not easily understandable.

We present new splitting technique for extracting classical Decision Trees from trained Neural Networks. Decision Trees are more understandable and they can easily be converted to If-Then rules. We show that the new method developed is able to create decision trees which are close in accuracy to the neural networks they are extracted from and their fidelity to the networks are high. Fidelity is the ability of the extracted decision tree to imitate the neural network it is extracted from.

ALGORITHM

DecText is a Decision Tree extraction algorithm. The Decision Tree extracted from the network mimics the network's behavior. It is the network's representation in a more understandable form. DecText tries to make Decision Tree learn the concept represented by the Neural Network. We have developed a new splitting techniques which tries to optimize fidelity between the tree and the network.

The algorithm proceeds in the same way as a classical decision tree algorithm.At every node it uses a splitting criteria to decide the feature and the value to split the feature on to split the nodes into child nodes.

The most commonly used splitting criterias are information gain and gini criteria.In This algorithm we use a different splitting criteria which takes into account the neural network that we are trying to mimic.

At any node our first objective is to find the best feature and the best split of that feature.The following two algorithms helps us decide that.

1. Best-Feature:

The Best-Feature algorithm returns the index of the feature upon which the node should split on.In

this algorithm, the feature which effects the outputs of the Neural Network the most is chosen for splitting the dataset. Changes in the value of more relevant features affect classification more than the changes in the value of less relevant features. Therefore, the most relevant feature has the most predictive power on classification. For each node by using the dataset and the feature set the Best-Feature algorithm returns the index of the feature upon which the node should split on.

For each node by using the dataset and the feature set available at that node we can find the feature with the most predictive power for the dataset at that node. For finding the relevance of a feature, we set its value to zero in each instance(to simulate removing the corresponding input node from the Neural Network) and find the difference in Neural Network's response to the new instance and the original instance. For each instance, the output values for the original instance are found. Then, a feature from the instance is removed and the output values for this new instance are found. The difference between the original output values and the new output values tell us how much the feature removed effects the outputs.

SSE (Sum Squared Error) is calculated using the formula:

$$SSE = \sum_{i}^{n}\sum_{j}^{m}(t_{i,j} - o_{i,j})^2 \qquad (1)$$

t: Teaching output **o**: NN's output **n**: # of instances **m**: # of classes

The above formula tells us how much a feature is affecting the neural network output.Here Teaching output is the network output of the original dataset and NN's output is the network output after zeroing the column to the corresponding feature.The feature that gives the highest SSE value is the most relevant feature at that point.

2. Best-Splitting Point:

After deciding the best feature using the above algorithm the next step is to determine the cutpoint to divide the dataset.. For each candidate cut point, the dataset is divided into two subsets and ClassDiff value is calculated for each subset. To find the ClassDiff value for a cut point, the ClassDiff values for the left and right subsets created by the cutpoint are found. The final value is calculated by using the formula:

$$ClassDiff_p(S) = \frac{|S_{left}|}{|S|}ClassDiff(S_{left}) + \frac{|S_{right}|}{|S|}ClassDiff(S_{right}) \qquad (3)$$

ClassDiff value of a dataset is calculated by the formula given in (1). First the dominating class for

the subset at a node (by using Neural Network's outputs) is found. Then, it is assumed that all the instances in that subset are supposed to be classified as the dominating class.

ti,j=1 for the dominating class and 0 for others.oi,j is the actual neural network output.

The split that give the highest ClassDiff value is chosen to split the dataset.

The algorithm uses minimum number of instances in a leaf and maximum depth as an early stopping criteria.

- **CONCLUSION AND RESULTS**

The algorithm was run on a neural network with one hidden layer with 1 neuron ,input layer with 4 neurons and output layer with 3 neurons,trained on the iris dataset to 100 percent accuracy.The algorithm returned an 88.66 percent fidelity rate , where fidelity rate is the accuracy of the decision tree when the neural network output is taken as ground truth.

Description Of Neural Network(with respect to PyTorch convention) :

```
Net(
  (fc1): MaskedLinear(in_features=4, out_features=1, bias=True)
  (tanh): Tanh()
  (fc2): MaskedLinear(in_features=1, out_features=3, bias=True)
  (sigm): Sigmoid()
)
```

One of the limitations of conventional decision tree algorithm is that it runs out of enough instances to classify towards the leaf node.In the algorithm we use this can be solved by generating new instances and its output using the neural network model to meet a given minimum criteria.Moreover the algorithm can be further improved by using appropriate pruning techniques to simplify the tree further.