Assignment 5 Project

B351 / Q351

Due: March 27th, 2018 @ 11:59PM

1 Summary

- Gain a basic understanding of machine learning
- Implement entropy and information gain methods for a decision tree algorithm

To run your program, you may find that you need to install the `numpy` module. You will do this utilizing Pip and using your terminal to execute the command `pip install numpy`. If your code still does not run, please post on piazza or come by office hours.

Please submit your completed files to your private GitHub repository for this class, and upload them to Autolab.

This assignment can be completed with **one** partner (or none if you prefer.) If you choose to work with a partner, you **must** include your partner's name in your files. Both you and your partner must submit your own files to your own repositories. Your files will be run against Moss to check for plagiarism within the class outside of your partnership.

2 Background

Decision Tree Learning is a machine learning algorithm often used for classification and regression problems. Decision trees offer several advantages over other algorithms, such as ease of implementation and reasonable interpretability, while facing several drawbacks such as a tendency towards overfitting. Our reasoning behind choosing decision trees as the subject for this assignment is that they require minimal mathematical background as compared to other machine learning methods (Linear Regression, Neural Networks, etc). In this assignment, we hope to provide you with some general background knowledge regarding machine learning, as well as give you a chance to work with a real, commonly used machine learning algorithm.

In order to discuss decision trees, it will help to have a common background on machine learning terminology. The following videos provide some background on machine learning, and I highly recommend you watch them. (Having this baseline of machine learning knowledge is a prerequisite for a good machine learning related final project!)

1. What is Machine Learning? (7:15)

2. Introduction to Supervised Learning (12:29)

Decision trees can be used for both classification and regression problems, but for this assignment we will be limiting our scope to classification. Even further, we limit the type of data which our algorithm can receive to categorical data. If you would like to try some of your own datasets with this decision tree implementation, you must ensure that your data is categorical, and the attributes (not the labels) are one-hot encoded. If you aren't sure what this means, try searching it or don't worry about it.

How does a decision tree work? There are two main phases: 1. learn the tree and 2. classify data.

2.1 Learning the Tree

Learning the tree is a recursive process. We start with a root node, which contains all data points. We will use some criterion to decide which attribute to use to split the data into 2 subnodes. Repeat this process at each of the subnodes, using the data they receive from their parent node. The process continues until a node is reached, wherein all the data is of the same class, or has identical attributes. Your job is to implement that criterion. For implementation details of the algorithm itself, refer to the `DecisionTreeFactory` class in `decision_tree_factory.py`.

What is the job of the criterion within this algorithm? It seems logical that it is to best separate the data coming into the node into subcategories. This is a fairly loose goal, and there are several popular criteria used for this purpose. In this assignment you will implement the criterion known as **information gain** (described here: `https://en.wikipedia.org/wiki/Decision_tree_learning#Metrics`).

Information gain works by finding the difference between the entropy in a parent node, and the entropy in all of its children. Further information can be found here: `https://en.wikipedia.org/wiki/Information_gain_in_decision_trees`.

2.2 Classify Data

Once the decision tree is built, classifying new datapoints is trivial (and already implemented for you). Simply propagate that datapoint through the tree, and when you arrive at a leaf node, return the class of the most common data in that node.

3 Programming Component

3.1 Objectives

Your goal is to complete the following tasks. (Probably in the order they are presented).

3.1.1 DTreeNode.get_entropy() (50%)

Entropy is a concept from information theory, which in vague terms, describes the amount of information missing from a system. This method should use the information contained in the given labels to determine the entropy value for a node in a decision tree.

3.1.2 DecisionTreeFactory._calc_information_gain() (50%)

Information Gain is a metric for choosing decision tree splits, returns a value based on the difference between the entropy of the parent, and the normalized entropy of all the children.

3.2 Bonus (up to 30%)

This assignment covers only a tiny portion of decision trees. If you have an interest in exploring the topic further, then we will offer bonus points for students who demonstrate significant additional work in exploring the topic. To receive bonus points, your bonus attempt code must be added to your GitHub repository, and you must submit a neat, well formatted pdf report to the 'Assignment 5 Project' assignment on Canvas. The report should explain, with examples, what you did and why, and how it affected performance compared to other methods on at least 2 datasets (the provided datasets are acceptable). In this report, you must clearly specify how many bonus points you expect to receive for your work.

Some ideas for a bonus implementation include using a different metric for the decision tree (e.g. GINI, misclassification error) (expect 10% bonus), making the algorithm run on continuous attributes (expect 20% bonus), making the algorithm perform both classification and regression, or making a random forest algorithm (expect 30% bonus).