Decision Trees

The Entropy of a Partition

Entropy can be defined as the measure of uncertainty in a sequence of random events. It is the rate of disorderliness in a sample space and is directly opposed to knowledge. When the entropy of a system is high, the knowledge that can be derived from the system is low and vice versa. An intuitive understanding of entropy is thinking of it as the amount of questions required to ask to arrive at some knowledge. For example, if I picked a random number and you were trying to guess what number it is. Asking a question like, “Is it an odd number”, reduces the possibilities space by half. This means that the entropy or the degree of uncertainty in trying to determine which number I choose is reduced. In the same vein, the amount of information gain is large because the question moved you closer to the answer by dividing the sample space. Entropy usually ranges from 0 to 1. A system with an entropy of 0 is highly stable and the knowledge that can be derived from such a system is high. In general terms, low entropy in a system indicates high knowledge while high entropy indicates low knowledge or instability.

Entropy can be represented mathematically as:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

The formula above is the negative sum of log probabilities of an event happening. Remember that probability indicates the confidence we have in an event occurring, therefore entropy is how surprising it would be, for a sequence of events to occur together.

In machine learning as we would see later with decision trees, the entropy of two or more attributes of a classifier is defined by:

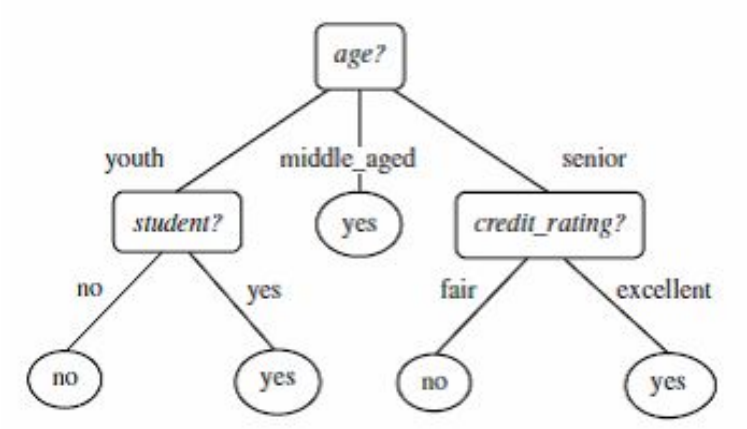
$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

Decision trees are a machine learning algorithm that rely heavily on the entropy of an attribute and the information gain to determine how to classify samples in a classification problem. Let us look at decision trees in depth in the next section.

Creating a Decision Tree

A decision tree is a machine learning algorithm which is mainly used for classification that constructs a tree of possibilities where the branches in the tree represents decisions and the leaves represents label classification. The purpose of a decision tree is to create a structure where samples in each branch are homogenous or of the same type. It does this by splitting samples in the training data according to specific attributes that increase homogeneity in branches. These attributes form the decision node along which samples are separated. The process continues until all sample are correctly predicted as represented by the leaves of the tree.

To explain the concept of a decision tree further, let us look at a toy example below that demonstrates its capability.



Let us assume that we are a laptop manufacturer and we want to predict which customers from an online store are likely to buy our new top of the range laptop, so that we can focus our marketing efforts accordingly. This problem can be modelled using a decision tree with two classes (yes or no), for whether a person is likely to purchase or not.

At the root of the tree, we want to choose an attribute about customers that reduces entropy the most. As we saw in the last section, by reducing the entropy, we increase the amount of knowledge that is contained in the system. We choose the appropriate attribute by calculating the entropy of each branch and the entropy of the targets (yes or no). The information gain is closely related to the entropy and is defined as the difference in entropy of the targets (final entropy) and the entropy given a particular attribute was chosen as the root node.

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

The formula above is used to calculate the decrease in entropy. The attribute with the largest information gain or decrease in entropy is chosen as the root node. This means that the attribute reduces the decision space the most when compared to other attributes. The process is repeated to find other decision nodes via attributes until all samples are correctly classified through the leaves of the decision tree.

In the example above, age is the attribute that offers the most information gain so samples are split on that decision node. If the customer is middle aged, then they are likely to purchase a new laptop as they are probably working and have higher spending power. If the customer is a youth this brings us to another decision node. The attribute used is whether the youth is a student or not. If the youth is a student, they are likely to buy else they are not. That brings us to the leaves (classes) of the node following the youth branch of the tree. For the senior branch, we again split samples on an informative attribute, in this case credit rating. If the senior has an excellent credit rating that means they are likely to buy, else the leaf or classification for that sample along this branch of the tree is no.

Let us now work on an example using Python, Scikit-Learn and decision trees. We would tackle a multi-class classification problem where the the challenge is to classify wine into three types using features such as alcohol, color intensity, hue etc. The data we would use comes from the wine recognition dataset by UC Irvine. It can be downloaded at https://gist.github.com/tijptjik/9408623/archive/b237fa5848349a14a14e5d4107dc7897c21951f5.zip

First, lets load the dataset and use Pandas **head** method to have a look at it.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# comment the magic command below if not running in Jupyter notebook
%matplotlib inline

dataset = pd.read_csv('wine.csv')
dataset.head(5)
```

	Wine	Alcohol	Malic.acid	Ash	Acl	Mg	Phenols	Flavanoids	Nonflavanoid.phenols	Proanth	Color.int	Hue	OD	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

There are 13 predictors and the first column “wine” contains the targets. The next thing we do is split the dataset into predictors and targets, sometimes referred to as features and labels respectively.

```
features = dataset.drop(['Wine'], axis=1)
labels = dataset['Wine']
```

As is the custom to ensure good evaluation of our model, we divide the dataset into a train and test split.

```
from sklearn.model_selection import train_test_split
features_train, features_test, labels_train, labels_test = train_test_split(features, labels,
test_size=0.25)
```

All that is left is for us to import the decision tree classifier and fit it to our data.

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()

classifier.fit(features_train, labels_train)
```

We can now evaluate the trained model on the test set and print out the accuracy.

```
pred = classifier.predict(features_test)
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(labels_test, pred)
print('Accuracy: {:.2f}'.format(accuracy))
```

```
Accuracy: 0.91
```

We achieve an accuracy of 0.91 which is very impressive. It means that 91% of samples in our test set were correctly classified.