This post will explore the mathematics behind information gain. We'll start with the base intuition behind information gain, but then explain why it has the calculation that it does.

What is information gain?

Information gain is a measure frequently used in decision trees to determine which variable to split the input dataset on at each step in the tree. Before we formally define this measure we need to first understand the concept of *entropy*. Entropy measures the amount of information or uncertainty in a variable's possible values.

How to calculate entropy

Entropy of a random variable **X** is given by the following formula:

$$-\Sigma_i[p(X_i) * \log_2(X_i)]$$

Here, each X_i represents each possible (i^{th}) value of **X**. $p(x_i)$ is the probability of a particular (the i^{th}) possible value of **X**.

Why is it calculated this way?

First, let's build some intuition behind the entropy formula. The formula has several useful properties. For example, it's always non-negative. Also, entropy as a function is monotonically decreasing in the probability, *p*. In other words, the amount of information about an event (or value) of **X** decreases as the probability of that event *increases*.

That may sound a little abstract at first, so let's consider a specific example. Suppose you're predicting how much a movie will make in revenue. One of your predictors is a binary indicator – 1 if the record refers to a movie, and 0 otherwise. Well – that predictor is useless! Mathematically speaking, it's useless because every record in the dataset is a movie – so there's a 100% probability of that event (i.e. the record being a movie) occurring. This means that the variable provides no real information about the data. The closer you get to a variable having a single possible value, the less information that single value gives you.

Why the $\log_2$? Technically, entropy can be calculated using a logarithm of a different base (e.g. natural log). However, it's common to use base 2 because this returns a result in terms of *bits*. In this way, entropy can be thought of as the average number of *bits* needed to encode a value for a specific variable.

Case Example

Information gain in the context of decision trees is the reduction in entropy when splitting on variable **X**. Let's do an example to make this clear. In the below mini-dataset, the label we're trying to predict is the type of fruit. This is based off the size, color, and shape variables.

Fruit	Size	Color	Shape
Watermelon	Big	Green	Round
Apple	Medium	Red	Round
Banana	Medium	Yellow	Thin
Grape	Small	Green	Round
Grapefruit	Medium	Yellow	Round
Lemon	Small	Yellow	Round

Suppose we want to calculate the information gained if we select the *color* variable. 3 out of the 6 records are yellow, 2 are green, and 1 is red. Proportionally, the probability of a yellow fruit is 3 / 6 = 0.5; 2 / 6 = 0.333.. for green, and 1 / 6 = 0.1666… for red. Using the formula from above, we can calculate it like this:

$-([3/6 * \log_2(3/6)] + [2/6 * \log_2(2/6)] + [1/6 * \log_2(1/6)])$

= 1.459148

Likewise, we want to get the information gain for the *size* variable.

$-([3/6 * \log_2(3/6)] + [2/6 * \log_2(2/6)] + [1/6 * \log_2(1/6)])$

= 1.459148

In this case, 3 / 6 of the fruits are medium-sized, 2 / 6 are small, 1 / 6 is big.

Lastly, we have the *shape* variable. Here, 5 / 6 of the fruits are round and 1 / 6 is thin.

$-([5/6 * \log_2(5/6)] + [1/6 * \log_2(1/6)])$

= 0.650022

How to calculate information gain in R

So, how do we calculate information gain in R? Thankfully, this is fairly simple to do using the **FSelector** package:

```
library(FSelector)

info <- data.frame(fruits = c("watermelon", "apple", "banana", "grape",
"grapefruit", "lemon"))
info$sizes <- c("big", "medium", "medium","small" ,"medium", "small")
info$colors <- c("green", "red", "yellow", "green", "yellow", "yellow")
info$shapes <- c("round", "round", "thin", "round", "round", "round")

# get information gain results
information.gain(formula(info), info)
```

Conclusion

That's all for now! Information gain is just one of many possible feature importance methods, and I'll have more articles in the future to explore other possibilities.…