Data Science

Introduction to Machine Learning:
Decision Trees. Overfitting.

April 19, 2021

Recap on the general problem

Many Machine Learning problems take the following form:

$$\text{minimize}_\theta \sum_{i=1}^{m} l(h_\theta(x^{(i)}), y^{(i)})$$

We've now looked at some ls and an h.

Previously, on...

Hypothesis function

1. We looked at a linear regression
2. We 'fit' this linear regression to our dataset
3. If our data is actually linear, we also get *predictive* power

A wild h appears

Linear Regressions aren't the only possible hypothesis function! We've also got:

1. *Decision Trees* : 20-questions, the ML technique
2. *Polynomials* : For when a straight line isn't cutting it
3. *Neural networks* : What if we misunderstood neurons and made it a program?
4. *Arbitrary Programs*: What is computers wrote the programs?

Do you realize?

A learning problem is said to be *realizable* if the true function exists within the learning problem's *hypothesis space*

1. This means that the more expressive the hypothesis space (polynomials vs straight lines) the more likely that the problem is realizable.
2. What's the downside?
3. Occam's[1] Razor is a data-scientist's best friend

[1] Also written as 'Ockham' or 'Ocham'

Decision Trees

We can view our tagged dataset (values of (x, tag)), as standing in for values of $(x, f(x))$.

1. As with the linear regression the goal is to find an h that approximates f.
2. But instead of a regression, we want a tree of *decisions*.
3. What's a decision?

Decisions! Decisions!

Each decision has two parts:

1. *Input* : An object[2] event/situation, that is described by a set of attributes (or *features*)
2. *Output*: A prediction of the 'value' based on the input
3. The boolean case (yes/no) is easy to visualize, but the values do not have to be discrete.

[2] not in the OO sense

Consider

You are asked to identify an animal based on a set of features (number of legs, weight, number of eyes, etc.)

1. The challenge is that the *order* of questions can matter!
2. You'll want the 'most significant' question first.
3. Unfortunately, it can be very expensive(!!) to find the most significant question.

A tiny bit more formally:

A decision tree has two types of nodes:

1. Decision nodes: Specifies a test on some attribute
2. Leaf node: A final classification/prediction

Small example:

We want to determine whether someone has ever seen an episode of Sponge Bob:

1. Are they older than 70: no.
2. Are they older then 40: if yes...
    2.1 Do they have kids: if yes, yes.
    2.2 no.
3. Are they older than 4: yes.
4. Do they have older siblings: yes.
5. no.

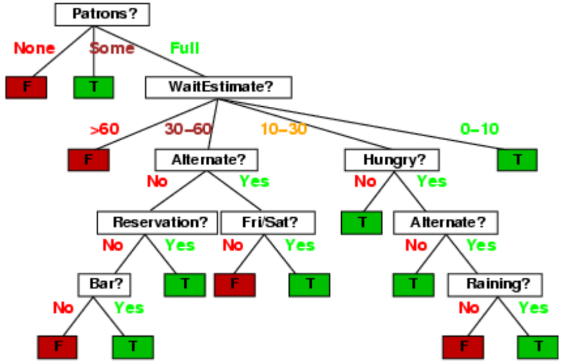
Oof

Even for such a small example, it starts getting unwieldy.

1. Luckily, libraries will be able to display trees nicely
2. For many trees it's not necessarily true that each 'decision', will have a meaningful-in-English question associated with it.

A prettier example

Should we wait for a table?



How many are there?

Decision Trees can encode arbitrary boolean functions.

1. Each attribute can be 0/1
   1.1 So our *input* space is 2^N
2. Each decision value can be 0/1, *for each possible combination of features!*
   2.1 So our *hypothesis space* is 2^{2^N}

Basic Algorithm

The goal is to find a *small* tree that correctly predicts the training samples

1. Choose the "most significant" attribute
2. Once you make a choice for "most significant", you don't backtrack (greedy)
3. Now you've split your dataset, repeat the process for each subset.

Significant?

How do we pick the “most significant”?

1. We can’t always :(
2. We want to try and maximize *information gain*
3. For this class: let the libraries do the work for you.

Fit and Finish

For all ML techniques, there's a danger of *overfitting*.

1. What does overfitting imply?
2. How would we know if we've done it?

Holdout Cross Validation

Idea: Don't use all your training data!

1. Instead of training your model on every you have, train it on some subset (training set).
2. Once you have the trained model, you *test* it on the rest, the *test set* (since you know the classifications).
3. You *must* ensure that your subsets are independent!

This slide is a trick

Consider:

1. We train four different hypothesis functions
2. We use our test set to see which hypothesis function performs best
3. We publish our awesome model!
4. Is the celebration warranted?

Thanks for your time!

:)