Trees

Sharad Chitlangia

04-24-2021

Trees

Discriminatory models take the form of $P(Y|X)$ in the probabilistic format. Here X and Y are Random Variables. What this means essentially is that we are modelling the conditional probability distribution $P(Y|X)$. This distribution can be later utilised to predict y from x.

Naive Bayes Classifier

The Naive Bayes Classifier predicts the final value to be the maximum from the posterior distribution. The NB Classifier has the least expected error but the problem with NB classifier is that we dont know the posterior distribution. For instance, we approximate the posterior distribution in logistic regression to be representable with the logistic function.

Structure and Parameters

Every Machine Learning models usually has two components: * Structure * Parameters

For instance, for a Neural network the structure is multiple layers connected together and the parameters are its weights. For a linear regression model, the structure is the linear line and the parameter is the weight. Similarly for a logistic model, the structure is the sigmoid function and the parameter is the weight.

In the case of a tree based classifier, the structure is a tree and the parameters are the probabilities.

Building a Tree

The problem of learning a tree is unique in the sense that we also need to build the tree instead of just learning some weights. But lets assume we have a tree and binary valued data with a binary classification problem. The nodes in this tree are assigned variables where if a variable is 1, we move to the left or if is 0, then we move to the right. We can perform this procedure for the data instance and finally reach a leaf node where we keep track of the counts of the two classes

Finally for this tree T_0 at each leaf node, we can calculate the the class conditional probability as:

$$P(Y|X) = \sum_{i=0}^{n} \frac{\mathbf{1}}{n}$$

For a binary problem, if we choose a Beta Prior $Beta(a, b)$ then the posterior can be calculated simply as $Beta(a + c_0, b + c_1)$ where c_0 and c_1 just represent the counts that have been stored after running through the dataset at that leaf.

For a problem with more than 2 classes, a similar relationship is exhibited by the multinomial and dirichlet distributions.

Choosing a Tree

But why should we pick a particular tree? We can create so many trees out of a set of variables. Lets assume we have a set of trees $T = \{T_0, T_1, ..., T_k\}$. How do we calculate the class conditional probability now?

The answer is that we could use a simple Bayes rule over all these trees as so:

$$P(Y|X, D) = \sum_{j=o}^{k} (P(T_j|D)P(Y|X, D, T_j))$$

This will give us a final probabilistic estimate of the class conditional probability. We know how to compute $P(Y|X, D, T_j)$ but how do we find $P(T_j|D)$? This is the posterior of a tree given a dataset. We can again use the Bayes Rule here:

$$P(T_j|D) = P(\pi_j, \theta_j|D)$$
$$P(\pi_j, \theta_j|D) \propto P(D|\pi_j, \theta_j)p(\theta_j|\pi_j)P(\pi_j)$$

The likelihood term here can be decomposed into indvidual likelihood terms of the leaves which is:

$$P(D|c_0, c_1; a, b) = \frac{\Gamma(a + c_0, b + c_1)}{\Gamma(a, b)}$$

The total likelihood from the tree now would just be a product of the individual likelihood terms:

$$P(D|Tj) = \prod_{k=0}^{m} P(D|leaf)$$

So we have the likelihood term calculated in the following equation

$$P(\pi_j, \theta_j|D) \propto P(D|\pi_j, \theta_j)p(\theta_j|\pi_j)P(\pi_j)$$

Now what is $P(\theta_j, \pi_j)$? θ_j is just the set of all priors in the tree hence:

$$P(\theta_j, \pi_j) = P(\theta_{j0}, \theta_{j1}...|\pi_j)P(\pi_j)$$

And this can be calculated quite simply as this is just expected value of each individual distribution (before running the data i.e., prior distribution).

Now $P(\pi_j)$ is just a regularization term at this point because we can keep going more and more deep like this but we have to stop at some point.

So finally, now we have each term calculated in the equation

$$P(Y|X, D) = \sum_{j=o}^{k} (P(T_j|D)P(Y|X, D, T_j))$$

Enumerating Trees

But what is the problem with this kind of formulation? For each dataset, we can create so many trees that the computation described above would not be tractable in any sense.

But how can we enumerate all possible trees. Enumeration of all possible trees is straightforward as long it chooses its procedure in a stochastic manner over all possible.

Procedure

1. Pick a box.
2. Pick a feature.
3. Change the box to an internal node (with the selected feature) and add leaves.

This procedure will generate all possible trees if the picking is stochastic. Hence, for each tree we can now calculate $P(T|D)$.

Conceptually this is feasible but the number of trees is exceptionally large. Hence, practically considering all possible trees is not feasible, so we search the space from some **optimal** trees.

An example of this is to use *greedy search* which maximizes $P(T_i|D)$.

Overfitting

As we go from top to bottom while expanding a tree, we're imposing more and more conditions, so the number of instances keeps decreasing. At a point we will hit a tree with only pure leaves (instances of one class are not seen at this leaf). Here, the likelihood is maximum and it makes no sense to expand further. Theoretically, with enough features we would always end up with a tree with pure leaves (*overfitting*).

It is the prior in a tree, which ensures that we do not overfit. This is analogous to regularization in numeric prediction.

Going from one tree to another

When we're calculating the posterior for a tree, we will perform a lot of repeated calculations at each step. That is, only perhaps a few leaves are added but the posterior requires considering the older leaves as well. Hence, instead of going with the highest value of $P(T_i|D)$, we will rather go in the direction of highest value of the ratio $\frac{P(T_k|D)}{P(T_i|D)}$, assuming the new tree is T_k and the older tree is T_i. In this ratio, most of the terms get cancelled.

In practise, algorithms use other metrics such as **entropy** or **gini**. (Higher entropy correlated with higher likelihood). As we just saw this is clearly prone to overfitting. Libraries dont handle this explicitly but rather implicitly through conditions on maximum possible depth or pruning.

Entropy of a Tree

We've already seen how to calculate the entropy of a leaf. Now we'll see how to calculate the entropy of a tree. The entropy of a tree is just the weighted average of its leaves. Now we pick the tree with the maximum entropy gain.

Decision Trees

Classification trees can be made into a decision tree by simply setting a probability threshold. The choice of the threshold is usually dependent on the loss function.

n-ary trees

If the attributes are not binary, we can still use the entropy gain as we have conditioned it to be dependent on 2 classes.

Real Valued Attributes

Real Valued attributes present a novel challenge since they are not bucketed. The solution to this is simple - We can discretize the an attribute based on its histogram. That is, we could use a threshold type of relation e.g. if $X < 11.8$, etc. Each of these trees generated in this manner would have an entropy associated with them and the previously described procedure can be used here.

Regression Trees (Real Valued Classes)

It is not that straightforward to calculate the entropy gain if the classes are real valued (e.g. in regression). Recall that entropy gain measures how much uncertainity exists in a distribution. The measure of uncertainity for a real valued distribution can simply be its standard deviation. Hence, the search can be directed using standard deviation over subtrees.