DATA SCIENCE II: Machine Learning MTH 9899 Baruch College

Lecture 3: Decision Trees

Adrian Sisser   Dmytro Karabash

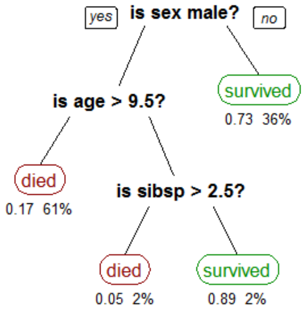
April 13, 2016

Outline

Outline

Decision Trees

Decision Trees are a form of supervised ML that seek to build a simple set of decision rules to make predictions. The consist of a series of decisions based on the data attributes, and end in a leaf.

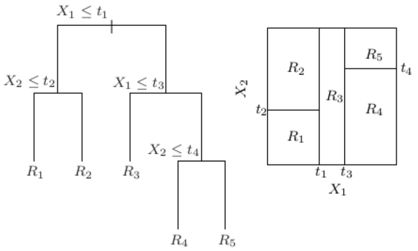


Types of Trees

- Classification Trees are used to predict what class a piece of data belongs to. The output can be a predicted class or a probability mass function across the different classes.
- Regression Trees estimate a continous variable for the output. Traditionally, Regression Trees give a point estimate, which is the average of all of the points in a given leaf. This effectively give us a piecewise constant fit.
- Model Trees are typically a form of regression tree, but instead of making having a constant prediction, each leaf consists of a fitted model. This could be any of the other models we have talked about, including: LSQ, LASSO, Ridge, Logistic, etc.

What can Decision Trees Do?

Decision trees partition the data up into complex regions, allowing us to find features in the data that we might not be able to otherwise.



Outline

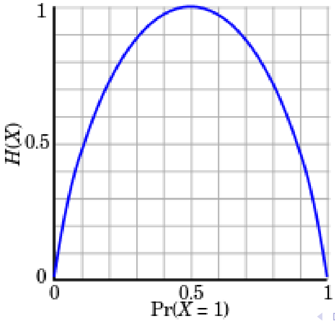
ID3 Algorithm

```
function BuildTree($S, A$)
    $a \leftarrow$ FindBestAttribute($S, A$)
    if $a$ is Null then
        return Leaf(S)                      ▷ The leaf summarizes the class(es) in $S$
    end if
    $\{S_i\} \leftarrow$ Split($S, c$)
    $A \leftarrow A - a$                    ▷ We won't split on $a$ again
    $N \leftarrow$ SplitNode($a$)
    for $a_i \in a$ do                      ▷ $a_i$ are the distinct values in $a$
        $N[a_i] \leftarrow$ BuildTree($S_i, A$)
    end for
    return $N$
end function
```

ID3 - Finding the Best Split

ID3 chooses it's splits by looking at Entropy and Information Gain. Entropy is a measure of the 'disorder' of a variable and is defined as:

$$H(X) = \sum_i -p(x_i) \log_2 p(x_i)$$



ID3 - Finding the Best Split

Our goal is to find a split that best reduces the Entropy in the subsets vs the original set. With ID3 we can end up with more than 2 splits. We define Information Gain as the decrease in Entropy when we split the input S into subsets, which we will try to maximize:

$$IG(S,a) = H(S) - \sum_i \frac{|S_i|}{|S|} H(S_i)$$

ID3 - Limitations

- Limitied to Classification
- We can only handle attributes with a relatively small set of discrete values. What happens when the number of possible values gets too high?
- Any Ordering information Categorical variables is lost, ie if you bin up a continuous variable, you lose most if it's value.
- We are using a **greedy** search to find split points, which doesn't lead to an optimal tree. **This is common to almost all tree algorithms**

C4.5

C4.5 is a successor to ID3. Major changes in C4.5 are:

- Continous attributes can be handled by looking at all possible values and we compute the information gain by splitting at this value. What are the drawbacks of this?
- We deal with the issues mentioned on the last slide, namely that too many possible values in a discrete variable create a bias. We do this by using Gain Ratio instead of Information Gain:

$$GainRatio(S,a) = \frac{IG(S,a)}{H(a)}$$

C4.5

Another major change from ID3 to C4.5 is *pruning*, or removing nodes from the tree that add complexity, but lead to overfitting. For C4.5, we look at the observed error rate, $\hat{r}$ in a given branch of the tree. We then calculate a confidence interval around what the true rate, r should be, at a given signficance, z:

$$r_{upper} = \hat{r} + z\sqrt{\frac{\hat{r}(1-\hat{r})}{N}}$$

We calculate r_{upper} for replacing a given branch with a leaf, and compare it to average weighted r_{upper} of the current branch leafs. If the r_{upper} for the leaf itself is less than the average, we prune.

Outline

CART

CART (Classification And Regression Trees) is alternative to ID3/C4.5 which extends to include regression. CART uses a new metric for Classification, GINI:

$$G(S) = \sum_{i=1}^{K} \frac{|S_i|}{|S|}(1 - \frac{|S_i|}{|S|})$$

We can compute $G(S)$ for each attribute a, and choose the attribute that minimizes $G(S)$.
For regression, we also need a metric. CART chooses Variance Reduction:

$$VR(S) = \operatorname{var} S - \sum_{i=0}^{K} \operatorname{var} S_i$$

CART - Pruning

CART also uses a different method for pruning. First, let's define a cost of complexity measure (for regression):

T	The set of leaf nodes in a given tree

$$C_\alpha = \sum_{i=1}^{|T|} \sum_{j=1}^{|T_i|} (y_j - \mu_i)^2 + \alpha |T|$$

We can now go through and build a series of trees, starting from the full tree down to one node, using a process called *weakest-link pruning*. When we do that, we will get a series of optimal trees for different values of α. Now, we know a set of optimal trees for each α, how do we pick what α to use?

MARS$^{\text{TM}}$ - Multivariate Adaptive Regression Splines

MARS$^{\text{TM}}$ is a regression methodology that an yield results similar to trees. It's a trademarked term so most open source packages are called Earth instead. The basic idea is to define sets of Hinge functions at every point:

$$\begin{aligned} h_+ &= \max(x-t, 0) \\ h_- &= \max(t-x, 0) \end{aligned}$$

If we have N examples of F features, we define $2NF$ hinge functions. We now build a set of basis functions, $\mathbb{B}$, starting with a constant, $\mathbb{B} = \{1\}$.

MARS$^{\text{TM}}$

Now that we have removed the mean, we can add a new basis function to $\mathbb{B}$, by taking the product of any of the existing functions in $\mathbb{B}$ and any pair of the hinge functions mentioned before, where the 'hinge' can be at any point:

$$\begin{aligned}\mathbb{B}_{i+1} &= \mathbb{B}_i \bigcup \{f(X)h_{+,t}(X), f(X)h_{-,t}(X)\} \\ \hat{y} &= \sum_{f\in\mathbb{B}} \beta_f \; f(X)\end{aligned}$$

One note is that to make this tracatable, we usually cap the order of any of the allowed basis functions.

MARS$^{\text{TM}}$

MARS$^{\text{TM}}$ let's us do some very intersting basis functions, including iteraction terms between variables. For example, imagine we have X_1 and X_2. We can end up with functions in $\mathbb{B}$ like:

$$f(X) = \max(X_1 - 10)\max(5 - X_2)$$

Ultimately, we have to use GCV to make sure we're not overfitting.

Model Trees

MARS$^{\text{TM}}$ can be seen as very similar to CART if we were to use Step functions instead of Hinge functions to form $\mathbb{B}$, but as it is it has a powerful advantage. We can fit a linear model instead of a simple point average in each node.
This is also the idea behind model trees. A *Model Tree* is a regression tree, where each leaf contains some form of model, rather than a simple point estimate.

Model Trees

By now, we know that the 2 main things we'll have to do are:

- Splitting - Come up with an efficient way to figure out which attribute (and where within that attribute) to split.
- Pruning (or Stopping) - How can we keep trees sizes reasonable sizes to avoid overfitting.

Model Trees - Splitting

Before, we looked at Variance Reduction as a useful metric for regression cases. We'll do the same, but using the variance of the residuals from the regression:

$$VR(S) = \operatorname{var} \epsilon_S - \sum_{i=0}^{K} \operatorname{var} \epsilon_{S_i}$$

For a continuous attribute, we have to worry about ways to quickly calculate this, we can choose to do a simple line search. If we do this along with some smart online regression techniques, we can make it fast enough to work. There are other techniques, such as the fluctation test, but they're quite involved and we won't go into more detail.
For pruning, we can look at analagous cost-complexity measures.

Model Trees

Some caveats and thoughts on Model Trees:

- Can we stop early? Look at the adjusted R^2 and see if we have a real noticable improvement - but this is complicated.
- Bayesian Shrinkage of 'small leaves'
- How to pick out 'fringe' values

Outline

Ensembles

Various styles of ensembles:

- In general ensemble is about joining different predictors into one. Can be viewed as a bottom layer of the machinelearning algorithm, however if the algorithm is complex it is not necesserily in the bottom.
- One-model ensemble utilizes training the same algorithm multiple times and then combining results to obtain better prediction
- Multi-model ensemble utilizes training different algorithms one or multiple times obtaining predictors that possibly capture different features of the reality.

Ensembles

Types of ensembles:

- Ensembles can be parallelized where each predictor is generated independently from the others, which allows for possibility of doing it in parallel. For example (sklearn.ensemble.RandomForestClassifier) is an example.
- Not all ensembles can be parallelized, for example (sklearn.ensemble.GradientBoostingClassifier).
- There can be mixtures, for example in multi-model ensembles.

Outline

- Historically an average (sometimes truncated) or median predictions have been used to combine predictions.
- In general any algorithm can be used, however one needs to remember what kind of ensemble it is.
- Same methods of combination would not apply to one-model ensembles as multi-model ensembles.
- In general one-model ensembles makes sense to combine in some some of truncated linear fassion.
- For multi-model ensemble one is might benefit for use more complex techniques from this course, build a little neural-network for example. The main reason for this is that in this case predictors are more independent.
- Instead of using predictors it is better to use differences of predictors with median or mean or something simple like that of predictors. The point is that you usually don't want your set on which you train neural-network.

Outline

There are several techniques that associate with ensemble methods but in the core they really come from statistics

- Boosting: focusing on hard examples Boosting does not always applies. One has to have to define some way to identify a hard set, in case of trees this is easy, some branches are harder than others, but in general one might want to use clustering techniques to split data into two sets (good and bad sets; where we would focus on bad).
- Bootstrapping and subsampling: using smaller subset for testing or training (bagging)
- Subsampling is sort of related to batch learning but there are differences: mainly size of “batch” and randomization of samples.