Decision Tree Learning

Notes based on Russell & Norvig, Chapter 18 and Mitchell, Chapter 3

Decision trees can represent all Boolean functions

How many Boolean functions are there on n variables?

Well, there are 2^n rows in the truth table

If there are x rows in the truth table, there are 2^x possible functions. So 2^{2^n} possible functions!

Are decision trees a good representation for all these functions? Parity, majority require exponentially large decision trees...

Impose an Inductive Bias

Suppose we want to favor short trees over long ones

Algorithm 1: Search breadth-first through trees of increasing depth. First all trees of depth 1, then all trees of depth 2, and so on, until you find a tree with (minimum/zero) error. Impractical.

Instead we'll focus on a greedy search algorithm called ID3.

Basic question: Which attribute should be tested at the root of the tree?

Information Theoretic Measures

First *entropy*, a measure of homogeneity of examples. Equivalently, a measure of the uncertainty associated with a random variable.

In the general case, when X can take on n values

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Boolean case:

$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p)$$

(define $0 \log 0 = 0$)

Maximal with equally likely outcomes, minimal when there is no uncertainty in the outcome

Suppose we have 14 examples:

$0+, 14- : 0$

$7+, 7- : 1$

$9+, 5- : 0.940$

Interpretation: minimum number of bits necessary to encode the information in a message on average

Information Gain: Expected reduction in entropy made possible by one feature. Where S is the training set and X is the feature

$$\text{Gain}(S, X) = H(S) - \sum_i \frac{|S_i|}{|S|} H(S_i)$$

where

$S_i = \{s \in S | \text{Example } s \text{ has feature value } i \text{ for } X\}$

Intuitively it is the amount of information provided by feature X about the class Y

An example from Tom Mitchell's book:

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Computing Gain(S, Humidity):

Split into High (3+, 4-, Entropy = 0.985) and Low (6+, 1-, Entropy = 0.592)

Then information gain is $0.940 - \frac{7}{14}0.985 - \frac{7}{14}0.592 = 0.151$

Computing Gain(S, Outlook):

Split into Sunny (2+, 3-, Entropy = 0.971), Overcast (4+, 0-, Entropy = 0), and Rain (3+, 2-, Entropy = 0.971)

Information gain is 0.246

The ID3 Algorithm

Given inputs: Training Set S, Feature Set Z

1. If all examples are positive (or negative), return the leaf node with label "Positive" (or "Negative")

2. Find the feature $X \in Z$ with highest information gain on the Training Set

3. For each possible value of X, call it X_i, add a branch that tests for $X = X_i$. Let the set of examples with $X = X_i$ be S_{X_i}. If S_{X_i} is empty then add a leaf node to the branch with the label set to the most common label in S. Otherwise, create a subtree beneath the branch by calling ID3 with training set S_{X_i} and feature set $Z \setminus \{X\}$

Inductive bias of ID3? Prefer shorter trees with higher information gain attributes closer to the root

How does this relate to our notions of hypothesis complexity?

Caution: Information gain prefers features with lots of possible values!

Continuous Attributes

Splitting into a Boolean attribute

Threshold must lie in between two points that are classified differently

Find all such points and evaluate the information gain of the feature for each of these possible threshold values. Pick the threshold that gives highest information gain

Pruning

Basic question in the context of decision trees: when do additional tests stop being useful for generalization? The option is to just use the most common class at that node as the label

Three possibilities:

1. Use a validation set to decide whether adding another test improves accuracy (*reduced-error pruning*, a post-pruning method). Problem: you lose potentially valuable training data

2. Use an explicit measure of complexity (e.g. minimum description length principle) to decide when to stop growing the tree

   MDL principle: We prefer "short" hypotheses in some coding scheme. In Bayesian

terms, we give higher priors to less complex hypotheses in some encoding scheme. Essentially, we are trying to maximize over h

$$P(D|h)P(h)$$

Equivalently, minimize

$$-\log_2 P(D|h) - \log_2 P(h)$$

If we think about this information theoretically, it turns out that the hypothesis we want is the one that minimizes the sum of the space required to describe the hypothesis and the space required to describe the data *given the hypothesis*

What's the tradeoff? Well, if we develop a hypothesis that is correct on all examples, the space required to describe the examples given the hypothesis is 0! So this is an intuitive way to think about the problem

3. Use a statistical method to decide whether adding another test is useful. Standard method: χ^2-pruning.

   The χ^2_k distribution is the distribution of the sum of squares of k iid standard normal RVs. k is the number of degrees of freedom

   Null hypothesis: attribute is irrelevant (if we had infinite examples, the information gain would be 0)

   Expected numbers of positives and negatives for the ith possible value under the null hypothesis:

$$\hat{p}_i = p\frac{p_i + n_i}{p + n}$$

$$\hat{n}_i = n\frac{p_i + n_i}{p + n}$$

   Measure of total deviation:

$$D = \sum_i [\frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}]$$

Under the null hypothesis, D is distributed according to a χ^2_{k-1} distribution. We can check for significance of our deviation easily

Evaluating Accuracy: Random Training/Test Splits

Divide dataset into **training set** and **test set**

Apply learning algorithm to training set, generating hypothesis h

Classify examples in test set using h, and measure percentage of correct predictions made by h (accuracy)

Repeat a set number of times.

Repeat the whole thing for differently sized training and test sets, if you want to construct a learning curve...

How do you compute confidence intervals for test accuracy?

What is the central limit theorem? Sum of independent random variables with finite mean and variance tends to normality in the limit. Notice: no condition on the distribution from which they are drawn!

Therefore, so does the mean of the independent r.v.s

Sampling distribution of the mean then tells us: 95% confidence interval given by mean $\pm 1.96\hat{\sigma}/\sqrt{n}$

Cross-Validation

Another possible method if you have two different candidate models

Attempt to estimate accuracy of the models on simulated "test" data

Standard approach: n-fold cross validation (very typical: $n = 10$). Divide the data into n equally sized sets. Train on $n-1$ of them and test on the nth. Repeat for all n folds

Is the accuracy of the better one then a good estimate of expected accuracy on unseen test data?

If you tune your parameters in *any* way on training data (including for model selection), you must test on fresh test data to get a good estimate!

Leave-One-Out Cross-Validation

Just what it sounds like. Train on all examples except one and then test on that one example. Repeat for all examples in the training data

Most efficient use of available data in terms of getting an estimate of accuracy

Can be horribly computationally inefficient, unless you can figure out a smart way to retrain without throwing away everything when swapping in one example for another