

1 Decision Trees

Let \mathcal{X} be an input space (like the set of all people) and let \mathcal{Q} be a set of predicates on \mathcal{X} , i.e., for $P \in \mathcal{Q}$ and $x \in \mathcal{X}$ we have that $P(x)$ is either true or false. Typically there is a feature vector $\Phi(x)$ and \mathcal{Q} is the set of tests of the form $\Phi_i(x) \leq c$ for some threshold c . We can model discrete features, such as gender, with 0-1 valued features. For example $\Phi_i(x) = 1$ if x is female and 0 otherwise. A decision tree over inputs \mathcal{X} and tests \mathcal{Q} is a binary branching tree every branch of which is labeled with a test in \mathcal{Q} and every leaf of which is labeled with one of the two labels -1 or 1 . We let $T(\mathcal{X}, \mathcal{Q})$ be the set of all decision trees over \mathcal{X} and \mathcal{Q} . More formally, we can define the set $T(\mathcal{X}, \mathcal{Q})$ to be the set of “expressions” defined by the following grammar where P must be a predicate in \mathcal{Q} .

$$T ::= -1 \mid 1 \mid \text{if}(P, T_1, T_2)$$

For a tree $f \in T(\mathcal{X}, \mathcal{Q})$ and $x \in \mathcal{X}$, we define $f(x)$ to be the label at the leaf of the tree f that we reach by using the test at each branch to select which branch to take. More formally we can define $f(x)$ by the following equations.

$$\begin{aligned} -1(x) &= -1 \\ 1(x) &= 1 \\ \text{if}(P, T_1, T_2)(x) &= \begin{cases} T_1(x) & \text{if } P(x) \text{ is true} \\ T_2(x) & \text{if } P(x) \text{ is false} \end{cases} \end{aligned}$$

Now consider a sample S consisting of a sequence $\langle x_1, y_1 \rangle, \dots, \langle x_T, y_T \rangle$ with x_i in \mathcal{X} and $y_t \in \{-1, 1\}$. Typically, the test set \mathcal{Q} is rich enough that one can construct a decision tree f which is perfect on the training data, i.e., we have $y_t = f(x_t)$ for all t . However, such a tree typically over fits and is not a good predictor on fresh data. But Occum's theorem tells us that simple rules which do well on the training data are guaranteed to do well on fresh data. When the set of tests \mathcal{Q} is finite we can define the size $|f|$ of tree f (in bits) as follows where n is the number of leaves.

$$|f| = 3n + (n - 1) \log_2 |\mathcal{Q}|$$

Occum's theorem is now the following.

$$\forall^\delta S \forall f \in T(\mathcal{X}, \mathcal{Q}) P_{\langle x, y \rangle \sim D} (y \neq f(x)) \leq \left(\frac{1}{T} \sum_{t=1}^T I[y_t \neq f(x_t)] \right) + \sqrt{\frac{(\ln 2)|f| + \ln \frac{1}{\delta}}{2T}}$$

2 Regression Trees and Greedy Tree Growth

Most decision tree learning algorithms construct trees by repeatedly replacing a leaf with a branch. This is done in such a way as to greedily reduce some

measure of the quality of the tree. An obvious possibility is to try to greedily reduce the accuracy on the training data. But greedy reduction of accuracy does not work well. To appreciate the difficulty consider a sample which is mostly labeled with -1 and has only a few labels of 1. The algorithm starts with the a single node (a leaf) labeled with -1 , the most common label. Now suppose that there exists a test P such that $P(x_t)$ is true every pair of the form $\langle x_t, 1 \rangle$ and false on half of the pairs of the form $\langle x_t, -1 \rangle$. The true branch of this test has a subset of the sample that is enriched in 1 labels and the false branch can be labeled with -1 and not expanded further. So this is a good test assuming that further tests can further enrich the 1 labels in the subset of the sample where P is true. Unfortunately, the subset of the sample where P is true may still be mostly labeled with -1 . In this case the best label of the true branch is still -1 and introduction of the test P does not improve accuracy. So when greedily selecting tests to put in at leaves of the tree some other measure of tree quality is needed.

To get a measure of tree quality more amenable to greedy predicate selection we generalize the notion of a decision tree to a regression tree. a regression tree is just like a decision tree except that every leaf is labeled with a real number rather than one of the two labels $-1, 1$. If f is a regression tree then $f(x)$ is defined in the same way as for a decision tree — it is the value leaf that we reach when branching according to the tests in at the branches in T . However, for a regression tree we can have that $f(x)$ is an arbitrary real number. Here we are interested in labeling each leaf with a number in $[0, 1]$ which is the fraction of training data reaching that leaf labeled with 1. Then $f(x)$ has a natural interpretation as $P(y = 1|x)$. For $y \in \{-1, 1\}$ we then have the following.

$$P(y|x, f) = \begin{cases} f(x) & \text{if } y = 1 \\ 1 - f(x) & \text{if } y = -1 \end{cases}$$

We can then try to do a minimization of regularized log loss.

$$f^* = \operatorname{argmin}_f \left(\sum_{t=1}^T \ln \frac{1}{P(y_t|x_t, f)} \right) + \lambda|f| \quad (1)$$

Although computing the optimal f^* is difficult, a greed algorithm can now be used effectively. The algorithm greedily replaces a single leaf by a branch in such a way as to achieve the largest possible improvement in the above objective function until no improvement is possible. If λ is large the algorithm will terminate with a small tree. For smaller values of λ the resulting tree will be larger. The parameter λ can be tuned with holdout data.

There is a large literature on decision tree algorithms. Equation (1) is fairly simplistic but captures the essential ideas.