Decision Trees

Peng Lingwei

August 15, 2019

Contents

18 Decision Trees

18.1 SAMPLE COMPLEXITY

First, we need a constrained decision trees model. Because we can easily build a decision tree with k leaves that shatters a set of k instances. Hence if we allow decision trees of arbitrary size with arbitrary feature used, we obtain a hypothesis class of infinite VC dimension, leading to overfitting.

Therefore, we assume that $\mathcal{X} = \{0,1\}^d$, which means that we only use finite binary features. Then any classifier from $\{0,1\}^d$ to $\{0,1\}$ can be represented by a decision tree with 2^d leaves and depth of $d+1$.Therefore, the VC dimension of the class is 2^d. If d is very large, the sample complexity is huge.

To overcome this obstacle, we rely on the MDL scheme described in Chapter 7 to punish the structure complexity of decision tree. The underlying prior knowledge is that we should prefer smaller trees over larger trees.

Here is one possible way to define a prefix free description language for decision trees ($\mathcal{H}$):

A tree with n nodes will be described in $n+1$ blocks, each of size $\log_2(d+3)$ bits. The first n blocks encode the nodes of the tree, in the preorder, and the last block marks the end of the code. Each block indicates whether the current node is:

- An internal node of the form $1_{[x_i=1]}$ for some $i \in [d]$;
- A leaf whose value is 1;
- A leaf whose value is 0;
- End of the code.

Overall, there are d+3 options, hence we need $\log_2(d+3)$ bits to describe each block. And the length of a tree with n nodes is $(n+1)\log_2(d+3)$.

It's easy to verify that this description for $\mathcal{H}_{decision\_tree}$ is prefix-free. By theorem in chapter7, we have

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left\{L_{\mathcal{D}}(h) \le L_S(h) + \sqrt{\frac{(n+1)\log_2(d+3)+\log(2/\delta)}{2m}}\right\} \ge 1-\delta$$

This kind or SRM is computationally hard.

18.2 DECISION TREE ALGORITHMS

Algorithm 1 Iterative Dichotomizer 3

Require: training set S, feature subset $A \subseteq [d]$
  If all examples in S are labeled by 1 **then return** a leaf 1.
  If all examples in S are labeled by 0 **then return** a leaf 0.
  If $A = \emptyset$ **then return** a leaf whose value = majority of labels in S.
  Let $j = \arg\max_{i\in A} Gain(S,i)$, and current node is internal node $x_j = 1$
  Let left child be the tree returned by $ID3(\{(\vec{x},y) \in S : x_j = 1\}, A\setminus\{j\})$.
  Let right child be the tree returned by $ID3(\{(\vec{x},y) \in S : x_j = 0\}, A\setminus\{j\})$.

Gain measure:

1. **Train Erros**: Let $C(a) = \{a, 1-a\}$. The error after splitting on feature i is $\mathbb{P}_S[x_i = 1]C(\mathbb{P}_S[y = 1|x_i = 1]) + \mathbb{P}_S[x_i = 0]C(\mathbb{P}_S[y = 1|x_i = 0])$, the corresponding gain is:

   $$Gain(S, i) = C(\mathbb{P}_S[y = 1]) - (\mathbb{P}_S[x_i = 1]C(\mathbb{P}_S[y = 1|x_i = 1]) + \mathbb{P}_S[x_i = 0]C(\mathbb{P}_S[y = 1|x_i = 0]))$$

2. **Information Gain**: $C(a) = -a\log(a) - (1-a)\log(1-a)$.

3. **Gini Index**: $C(a) = 2a(1-a)$.

The ID3 algorithm returned tree will usually be very large. One common solution is to prune the tree after it is built.

Algorithm 2 Generic Tree Pruning Procedure

Require: Tree T, Function $f(T, m)$ (estimate for the true error), based on a sample of size m.
for node j in a bottom-up walk on T **do**
Find T' which minimizes $f(T', m)$, where T' is any of the following:
the current tree after replacing node j with a leaf 1.
the current tree after replacing node j with a leaf 0.
the current tree after replacing node j with its left subtree.
the current tree after replacing node j with its right subtree.
the current tree.
Let $T = T'$.
end for.

If the features are real-valued features, such as: x_i. For training set $\vec{x}_1, \ldots, \vec{x}_m$, we can split ith feature by $\theta_{0,i}, \ldots, \theta_{m+1,j}$, where $\theta_{j,i} \in (x_{j,i}, x_{j+1,i})$, $x_{0,i} = -\infty, x_{m+1,i} = \infty$.

If the original number of real-valued features is d and the number of examples is m, then simple calculation of Gain needs $O(dm^2)$ operations, but clever implementation is $O(dm\log(m))$.

18.3 RANDOM FORESTS

Left...