Computational Learning Theory	**Spring Semester, 1997**
Lecture 6: April 2, 1997	
Lecturer: Yishay Mansour	*Scribe: Hed Bar-Nissan*

6.1 Decision Tree Learning Algorithms - Heuristic Methods

6.1.1 Introduction

We are intersted in learning using decision trees as the classification method. We are given sapmples $< x_i, b_i >$ for some problem. We want to use this collection to build a decision tree. We would like our decision tree to have two qualities :
1.)We would like the tree classify most/all the sample points.
2.)We would like the tree to be small.
Having those two qualities the tree is guaranteed to supply learning by Occam razor.

A Tree Example

There is a general predicates set $H = \{x_i \geq \alpha | \alpha \in R\}$ Each internal node in the decision tree is labeled with some predicat and each leaf is labeled with 0 or 1, thus giving classification to the input.
To classify the input one starts with the root, going down the tree according to the values of predicats in the internal nodes untill the leaf is reached and its label detirmines the input's classification (see Figure 9.1).

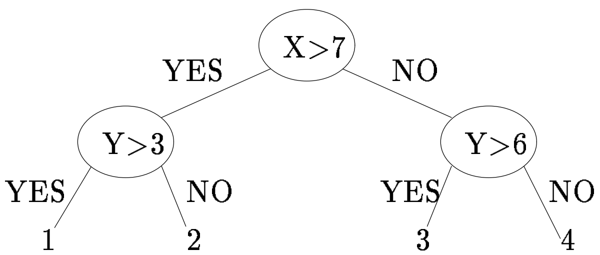


Figure 6.1: Example of a Decision tree

Some algorithms for building decision trees were suggested in the Machine Learning literature (C4.5, CART software packages). They become very popular in practice , because decision trees have an advantege to be easy interpretable by humans, relatively simple and fast to compute.

6.1.2 Top-Down Approach and Cost Function

The general idea how to build a decision tree is to grow it from root to the leaves by repeatedly replacing an existing leaf with a "good" internal node.
What do we consider as a "good" node ?
Knowing to answer that question will supply us a recursive algorithm for building a decision tree.
First decide what predicat is "good" for the root then build the left and right subtrees recursively. Let's try to define "good". Our goal for the tree is to reduce the error.// One simple option is to try to minimize the error at each node. Let the error at leaf v be $E(v)$,then the tree error is $E(T) = \sum_{v \in T} Prob[v]E(v)$
Still that doesn't give us any idea of what should be considered as a good node. So we'll use a greedy algorithem that will minimize a local cost function $G(q)$ in each step.
The most intuitive local cost function $G(q)$ is the leaf error function : $G(q) = min\{q, 1-q\}$. where q is the error in classifing the positive samples.
In the example below, if a q fraction of the root input is positive and each leaf classify the reaching data as 1 then $q = r * p + (1-r)t$

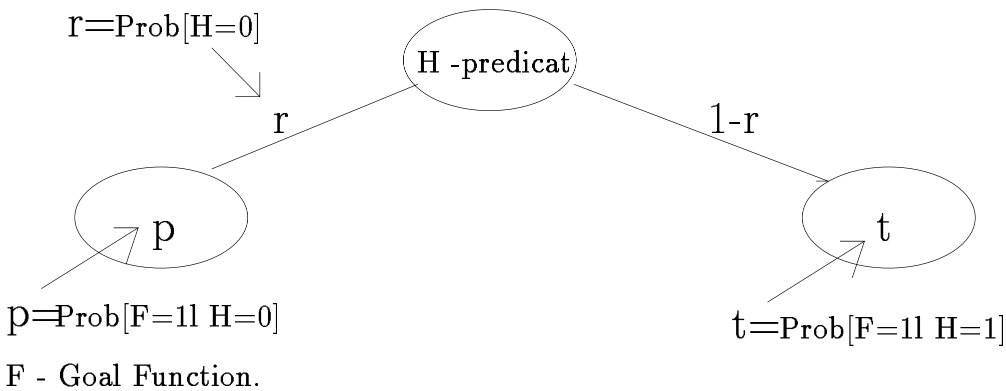
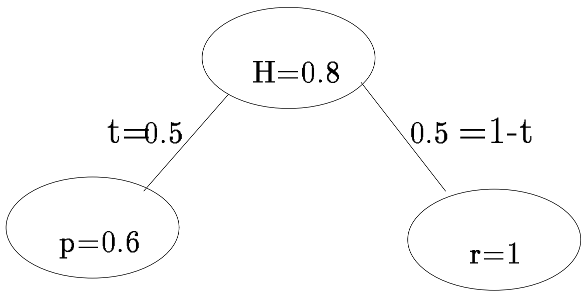


Figure 6.2: Split of Probability

As we shall see in the next example, though intuitive the error function is not the right function for us to minimize localy since it does not give us sufficiant progress.

Lets consider the following example. In the above example we have a situation where the inputs that reach the root, a q fraction of them is 1 positive. With probability τ the predicat H is 0 (and $1-\tau$ it is 1). At the leaves we have a p and r fraction of the inputs are positive. Clearly $q = \tau p + (1-\tau)r$.

As we can see in the tree above the error before the root is 0.2.



H,p,r - Prob [F=~~1~~The node has been Reached]

Figure 6.3: An example in which the error does not decrease although intuitively we made progress since half of input are classified perfectly.

Now again judging by intuition it seems that the node (it's predicat) does a very good job, since for half of the inputs there is no need to build any more trees.
Still the created subtree has error $E(T) = \sum_{v \in T} Prob[v] E(v) = 0.5*0+0.5*0.4 = 0.2$ which stays as before. So our greedy algorithem won't take this step if it will use the error function as the local minimizing function.
We need to search for a different local function. In general, we want from the cost function $G(q)$ to meet the following criteria:
1. $G(q)$ should be simmetric around 1/2, that is $G(q) = G(1-q)$.
2. $G(T)$ should be convex.
This also guarantees $G(x) \geq x$ for $1/2 \geq x \geq 0$. Under these conditions $G(T) \geq error(T)$ and it will be sufficient to show that $G(T)$ is small.

6.1.3 The Idea of Top-Down Decision Tree building

We want to build a tree that minimizes the error. In general,it is NPC problem. However it is possible to construct a greedy algorithm that uses $G(T)$ to grow the tree stage by stage. If an algorithm is succesfull in building a tree with a small $G(T)$ then $error(T)$ is also small.

For the sake of simplicity, instead of talking about sampling to approximate probabilities we assume that it is possible for given input to compute exact probabilities of arriving to an internal node and a leaf.

In growing the tree the idea is to choose one of the leaves and turn it to the internal node by labeling the choosen leaf with hypothesis h. We suppose, that there is the set of hypothesis functions (predicats) H that provides the possible hypothesises (predicats) for all nodes.

For some $h \in H$, $T(l,h)$ designates the tree that created by labeling leaf l with h and attaching to l two new leaves. The classification of each of new leaf is determined by the "most popular" classification value of the inputs that arrives at it. We choose hypothesis $h \in H$ that maximize the drop in the value of the cost function $G(T)$, that is $\max_{h_i \in H} G(T) - G(T(l,h))$.

input

For target function f let $q = Prob[f = 1 | \textit{ arrived to } l]$. After l became an internal node, then q fraction is splitted into two probabilities ,p and r, one for each of the new leaves (see Figure 9.2). The probabilities have to maintain :

$$q = r\tau + p(1-\tau),$$

where τ is the probability that $h = 1$, given that we reached node l.

Because all the leaves except l remained unchanged, they canceal in the difference $G(T) - G(T(l,H))$ and we get :

$$G(T) - G(T(l,H)) = G(q) - [\tau G(r) + (1-\tau)G(p)].$$

Our goal is to maximize the above expression. To achieve this goal we check the maximal value of the drop for each leaf and then choose the leaf with the maximal drop i.e $\max_{h,l} G(T) - G(T(l,h))$..

6.1.4 Top-Down Algorithm

The algorithm is given the number of desired nodes t, the set of predicats H and splitting function (cost function) $G(T)$.

During t steps, in each step i algorithm does the following:

1. For T_i chooses the pair (l,h), that maximizes $G(T_i) - G(T_i(l,h))$ [1]

2. $T_{i+1} \leftarrow T_i(l,h)$

The existing algorithms (such as C4.5 and CART) use simple class $H = \{x_i \geq \alpha | \alpha \in \Re\}$. It is clear, that the more complex class H the more possibilities exist, but this is compencated by the computation complexity for functions that belong to H and thus, the search in the tree may take a longer time.

[1] T_0 is the tree with only one node

For splitting there are several possibilities (see Figure 9.3) :

1. Ginn Index <u>CART</u>

$$G(q) = 2q(1-q)$$

2. Entropy <u>C4.5</u>

$$G(q) = \frac{1}{2}[q \cdot \log_2 \frac{1}{q} + (1-q) \cdot \log_2 \frac{1}{1-q}]$$

3. New function

$$G(q) = \sqrt{q(1-q)}$$

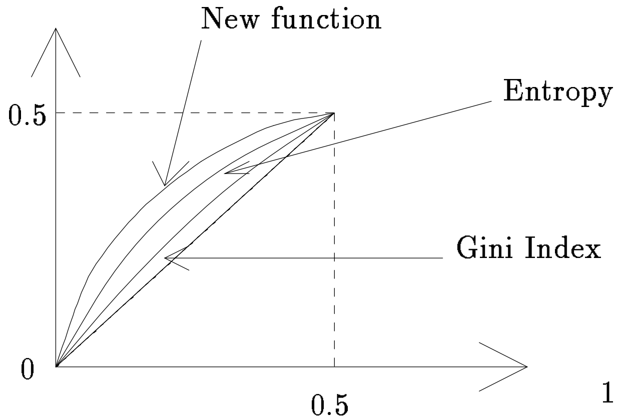


Figure 6.4: Relationships between different cost functions.

Why $G(q) = min(q, 1-q)$ is not a good choice? The reason is that this function behaves lineary. If function behaves lineary then split does not reduce the error of the tree.
If we recall the example we saw in the beginning, let $Prob[f = 1|\overset{input}{arrived\ to\ l}] = 0.8$, $\tau = 0.5$, $Prob[f = 1|h = 0] = 0.6$, $Prob[f = 1|h = 1] = 1$ (see Figure 9.3). Then, for $G(q) = min(q, 1-q)$ we have befor splitting

$$G(0.8) = min(0.8, 0.2) = 0.2$$

and after the splitting

$$0.5G(0.6) + 0.5G(1) = 0.5 \cdot min(0.6, 0.4) + 0.5 \cdot min(0, 1) = 0.2.$$

That is, there is no drop in the cost of the tree after the splitting of the leaf. For $G = 2q(1-q)$ we have befor splitting

$$G(0.8) = 2 \cdot 0.8 \cdot 0.2 = 0.32$$

and after the splitting

$$0.5G(0.6) + 0.5G(1) = 0.5 \cdot 2 \cdot 0.4 \cdot 0.6 = 0.24.$$

In this case, there is a drop in the cost of the tree after the splitting of the leaf. The drop is due to the fact that $G()$ is convex and not linear. similar drops would be observed for the other two optional cost functions. (see figure 9.4)

6.1.5 Theoretic analysis

The Role of Weak Learning Model

We conduct theoretic analysis in the model of Weak Learning. We assume that for every distribution D over the inputs there exists $h \in H$ such that

$$Prob[c_t \neq h] \leq \frac{1}{2} - \gamma.$$

It will be shown that heuristic decision tree algorithms ,under this assumptions decrease the error (though they were not designed to work in this model). The performance of heuristic algorithms is less impressive than performance of algorithms specially designed for Weak Learning (such as AdaBoost). Part of the problem is the decision tree representation that limits the algorithms.

The Results of Weak Learning

At the first stage, to use the Weak Learning assumption we must create a balanced distribution in the leaf. That is, half of the inputs that arrive at the leaf will be positive and half will be negative. On such distribution we find hypothesis $h \in H$ that is better than simple random guessing. It may be shown that such hypothesis reduces $G(T)$ considerably and does this locally in the splitted leaf. More specifically, it may be shown that split in the leaf reduces the total cost of the tree at least by

$$16\gamma^2[q(1-q)]^2.$$

This result may be used to find the number of steps until the algorithm reduces the error of the tree to at most ϵ.

We will show that at any stage t there is a leaf l such that:
1. The probability for input to arrive at l is $Prob[l] \geq \frac{\epsilon_t}{2t}$.
2. $error(l) = min\{q_l, 1 - q_l\} \geq \frac{\epsilon_t}{2}$.
Here ϵ_t denotes the prediction error of the tree at the stage t.

Proof: Leaves, that for them $Prob[l] \leq \epsilon_t/2t$,meaning they don't have property 1, contribute no more than $\epsilon_t/2$ to the prediction error of T. (there are exectly $t+1$ leaves at step t)
Leaves, that the error in them is less than $\epsilon_t/2$,meaning they don't have property 2, also contribute no more than $\epsilon_t/2$ to the prediction error of T.
Therefore, if there is no leaf with properties **1** and **2** the total error of the tree T would be less then ϵ_t.) □

From this follows,

$$Prob[l]c_1\gamma^2[q_l(1-q_l)]^2 \geq c_2\gamma^2 Prob[l](min\{q_l, 1-q_l\})^2$$

$$\geq c_2\gamma^2 \tfrac{\epsilon_t}{2t}\left(\tfrac{\epsilon_t}{2}\right)^2 \geq c_3\gamma^2 \frac{G_t^3}{t} \ ,$$

where G_t is the cost of the tree at the stage t and c_1, c_2, c_3 are constants. (We used the fact, that $\epsilon_t \geq G_t/2 = \epsilon_t(1-\epsilon_t)$ for Gini Index.) Since the above expression is the drop in the cost of the tree we may conclude that

$$G_{t+1} \leq G_t - c_3\frac{\gamma^2}{t}G_t^3$$

We want the total error of the tree to be less then ϵ. We can solve the above expression for G_t and get that if $t \geq 2^{c/\gamma^2\epsilon^2}$ then $G_t \leq \epsilon$. That is, if we run the algorithm at least $2^{c/\gamma^2\epsilon^2}$ steps than the error of the tree will be less than ϵ.

To evaluate the value of the drop in the error after the split we need to prove the following lemma.

Lemma 6.1 *Let D be a probability distribution and D' a balanced distribution of D. If $Pr_{D'}[h \neq C_t] \leq \frac{1}{2} - \gamma$ then $\tau(1-\tau)\delta \geq 2\gamma q(1-q)$,*
where $\delta = \mid Pr_D[C_t = 1|h = 0] - Pr_D[C_t = 1|h = 1] \mid$.

Proof: The error with respect to the balanced distribution is $Pr_{D'}[h \neq C_t] = \frac{1}{2(1-q)}Pr_D[C_t = 0|h = 1] + \frac{1}{2q}Pr_D[C_t = 1|h = 0]$. We multiply $Pr[C_t = 0|h = 1]$ by $\frac{1}{2(1-q)}$ and $Pr[C_t = 1|h = 0]$ by $\frac{1}{2q}$ to create the balanced distribution D', that gives equal weight to the positive and negative examples of C_t that reach the children of node l. Using our notation, $Pr[C_t = 0|h = 1] = \tau(1-r)$ and $Pr + D[C_t = 1|h = 0] = q - \tau r$. (This is because $q = Pr_D[C_t = 1] = Pr_D[C_t = 1|h = 1] + Pr_D[C_t = 1|h = 0] = \tau r + Pr_D[C_t = 1|h = 0]$). Thus we may write

$$Pr_{D'}[C_t \neq h] = \frac{1}{2(1-q)}\tau(1-r) + \frac{1}{2q}(q - \tau r)$$

$$= \frac{1}{2} + \frac{\tau}{2}\left(\frac{1-r}{1-q} - \frac{r}{q}\right).$$

On the contrary, according to the Weak Learning assumption on h $Pr_{D'}[C_t \neq h] \leq \frac{1}{2} - \gamma$. From this follows,

$$\frac{\tau}{2}\left(\frac{1-r}{1-q} - \frac{r}{q}\right) \geq \gamma.$$

Substituting $r = q + (1-\tau)\delta$ yields

$$\frac{\tau(1-\tau)\delta}{2}\left(\frac{1}{q}+\frac{1}{1-q}\right) \geq \gamma.$$

Since $1/q + 1/(1-q) = 1/q(1-q) = 1/q(1-q)$, the lemma follows. □
Now we can calculate the drop in the cost function

$$\begin{aligned}
\triangle_G(q,\tau,\delta) &= G(q) - [(1-\tau)G(q-\tau\delta) + \tau G\left(q+(1-\tau)\delta\right)] \\
&= 2q(1-q) - 2[(1-\tau)(q-\tau\delta)(1-q+\tau\delta) + \tau\left(q+(1-\tau)\delta\right)\left(1-q-(1-\tau)\delta\right)] \\
&= 2\tau(1-\tau)\delta^2.
\end{aligned}$$

From the previous lemma we know that $\tau(1-\tau)\delta \geq 2\gamma q(1-q)$ and thus, it is possible to conclude that

$$\delta \geq \gamma q(1-q)$$

and

$$(1-\tau)\tau \geq 2\gamma q(1-q).$$

From this follows that

$$\begin{aligned}
\triangle_G(q,\tau,\gamma) &= 2\tau(1-\tau)\delta^2 \\
&= 2\tau(1-\tau)\delta \cdot \delta \\
&\geq 2\gamma q(1-q) \cdot \gamma q(1-q) \\
&= 2\gamma^2 q^2 (1-q)^2.
\end{aligned}$$

This result we proved for Ginn Index. Using the results for other cost functions (without proof) we may summarize :

For Ginn Index

$$t \geq 2^{c/\gamma^2\epsilon^2}$$

For Entropy

$$t \geq 2^{\frac{c}{\gamma^2}\frac{1}{\log^2\frac{1}{\epsilon}}}$$

For $G(q) = \sqrt{q(1-q)}$

$$t \geq (\frac{1}{\epsilon})^{c/\gamma}$$