Machine Learning

Lecture 6-Decision Tree & MDL

Lecturer: Haim Permuter *Scribes: Asaf Lavi and Ben Marinberg*

This lecture discusses decision trees and the minimum description length principle. the goal of desicion trees is to have a series of questions(tree), that ends in decisions(leaves). Discussed as a regulating mechanism, the MDL shows that the best hypothesis for a given set of data is that which leads to the best compression of the data. The last part of the lecture discuses two methods for avoiding overfitting: Random Forest and Pruning.

I. Decision Tree

Here we have a method in which we create a k'-nary tree. Each node represents a question about the features, Θ_n. Each branch represents an answer to its origin node, one of k options. Each leaf represents a decision. A decision tree is easily implemented in a variety of applications and it is useful for both classification and regression. This lecture introduces the notation of the binary decision tree, but extending it to k'-nary is straightforward. Let $\{x_i\}$ be the i'th sample, $i = 1, 2, .., m$, each of which has n features, and Θ is the question asked in each node. Note that a large part of this section is taken from the book and lecture by Prof. Shai Shalev-Shwartz[1], [2].

Example 1 (classification-Papaya) Assume X is a papaya. The goal is to decide whether or not its tasty, by using a decision tree. Each node represents a question about the papaya. As we can see from the example (Fig.1), the top node asks about the color of the papaya, which, if is not the right color, leads to the decision that the papaya is not tasty. If it is the correct color we proceed to the next node and ask the next question. If the answers to the question about both color and softness were "right" (i.e according to our definitions of tasty papaya), we decide that the papaya is tasty.

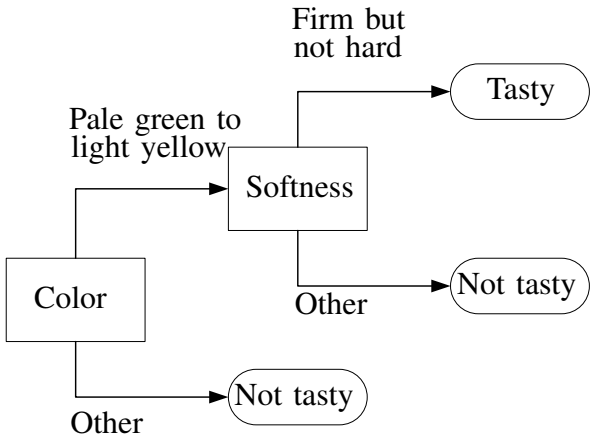

Fig. 1. Classic Decision tree for papaya classification,using two features: color and softness.

A. *Cost functions*

Let $\{x_{i,j}\}$ be the j'th feature of the i'th sample, and $\{y_i\}_{i=1..m}$ be the classifications (labels). Define gain, where C(x) is the cost function, as

$$Gain(Y;\Theta) = C(Y) - C(Y|\Theta) = C(Y) - \sum_{\theta\in\Theta} P(\Theta=\theta)C(Y|\Theta=\theta) \tag{1}$$

Examples of cost functions

1) Entropy $H(a) = \mathbb{E}[-\log_2 P_\Theta(a)]$
2) Gini index $G(a) = 4a(1-a)$
3) Error $E(a) = 2min(a, 1-a)$

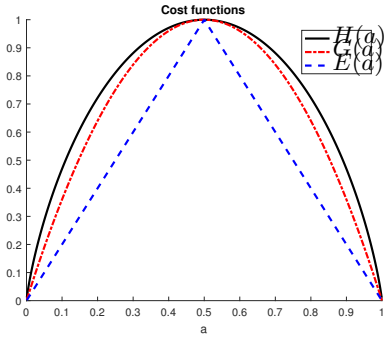

Fig. 2. Common cost function

Gain with entropy as the cost is called information gain, and it is widely used in applications. Note that information gain is the mutual information we learned about in the first lecture of the course.

$$Information\ Gain(Y;\Theta) = H(Y) - H(Y|\Theta) \triangleq I(X;\Theta) \tag{2}$$

Example 2 (One layer decision tree)

Lets begin with one-layer depth. Assume that we have 100 strawberries - 50 are tasty and 50 are not. To aid in their classification, we ask about their color. Determine $\theta = 0$ as not red, and $\theta = 1$ as red.

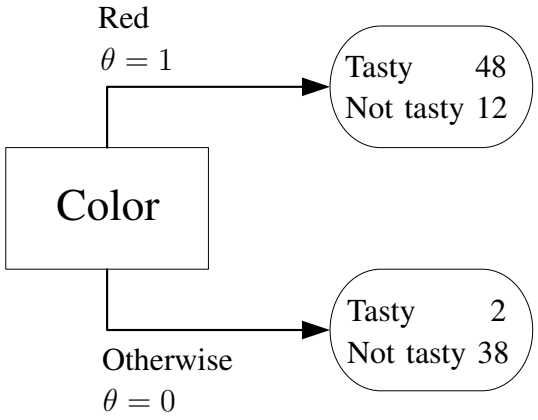

Fig. 3. One layer decision tree

$$Information\ Gain = H(Y) - H(Y|\Theta) = 1 - \frac{2}{5}H_{binary}(\frac{2}{40}) - \frac{3}{5}H_{binary}(\frac{12}{60}) \tag{3}$$

Set cost to be the Gini index and we have

$$Gain(Y;\Theta) = 1 - \frac{2}{5}G(\frac{2}{40}) - \frac{3}{5}G(\frac{12}{60}) \tag{4}$$

B. Algorithm

ID3-Iterative Dichotomiser 3 is an algorithm invented by Quinlan[3] for creating a decision tree. It is a recursive and greedy algorithm whose inputs are the S-data set and A-features. Its stopping conditions are simple: either all the remaining labels are the same or it ran out of unused features.

ID3(S,A)
create a new node;
if *all samples are ones* **then**
    label new node as leaf labeled 1;
else
    if *all samples are zeros* **then**
        label new node as leaf labeled 0 ;
    else
        if $A \in \varnothing$ **then**
            label new node as leaf labeled as the majority of labels in S;
        else
            $p = \underset{a \in A}{\operatorname{argmax}} \quad G(S, 1_{x_a=1})$;
            ID3($S_{1_{x_a}=0}, A \backslash p$);
            ID3($S_{1_{x_a}=1}, A \backslash p$);
        end
    end
end

C. Regression

Like with classification problems, decision trees can also be simply applied in regression problems. Generally speaking, we substitute the label decision with the average or mean. So instead of asking questions with k answers, we ask about k ranges. And instead of taking the majority of the labels we have the average or mean of the leaf's data. For example when putting a price on a new house, its estimated value will be the mean of the values of all houses with the same features (i.e. identical answers to questions on the tree).

II. MDL

The minimum description length principle describes a way to minimize models. It is similar to combining the length (or in our case, the tree depth) and the cost into a new and improved cost function. The goal of MDL can be described as "to find regularity in the data". where 'regularity' means compressibility. MDL combines the insights it gains by viewing learning as data compression: it tells us that, for a given set of hypotheses h and data set S, we should try to find the hypothesis or combination of hypotheses in $\mathcal{H}$ that compresses S the most. MDL procedures automatically and inherently protect against overfitting and can be used to estimate both the parameters and the structure of a model. Also, in contrast to other statistical methods, MDL procedures have a clear statistical interpretation.

Let P be the distribution, $S = \{x_i, y_i\}_{i=1}^m \overset{i.i.d}{\sim} Q$ to be the samples set, and h to be a hypothesis (function) $\mathcal{X} \xrightarrow{h} Y$. Define $d(h)$ to be the description of h with the prefix property(such that-there is no whole description in the system that is a prefix of any other description in the system). Define $L_D(h) = \Pr(Y \neq h(X))$ as theoretical risk, and define empirical risk as $L_S(h) = \frac{1}{m}\{i : h(x_i) \neq y_i\} = \frac{1}{m}\sum_{i=1}^{m} 1_{\{h(x_i)\neq y_i\}}$. Define the length of the description of h as $w(h) = 2^{-|d(h)|}$. Denote the Kraft inequality proved in this course's first lecture,

$$\sum_h w(h) = \sum_h 2^{-|d(h)|} \leq 1 \tag{5}$$

Theorem 1 (MDL bound) :
Let w be $\mathcal{H} \xrightarrow{w} \mathbb{R}$ such that $\sum_{h\in\mathcal{H}} w(h) \leq 1$. Then, with a probability of at least $1-\delta$ over $S \sim Q^m$ we have:

$$\forall h \in \mathcal{H}, \quad L_D(h) \leq L_S(h) + \sqrt{\frac{-\log(w(h)) + \log(\frac{2}{\delta})}{2m}}$$

Or alternately

$$\forall h \in \mathcal{H}, \quad \Pr\left(L_d(h) - L_S(h) \leq \sqrt{\frac{-\log(w(h)) + \log(\frac{2}{\delta})}{2m}}\right) \geq 1-\delta$$

The proof of the MDL bound is based mainly on Hoeffding's inequality, wherein setting some free parameters we bring the inequality to seen like the bound we need. From here on the proof is straightforward.

Lemma 1 (Hoeffding's Inequality) :

Let $X_1, X_2, ..., X_n$ be independent random variables , bounded by $[a_i, b_i]$. Following Hoeffding:

$$\Pr\left(\overline{X} - \mathbb{E}[\overline{X}] \geq t\right) \leq \exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) \tag{6}$$

Or similarly:

$$\Pr\left(|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu| > \epsilon\right) \leq 2\exp\left(-\frac{2m\epsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) \tag{7}$$

See the appendix for a proof of this lemma.

Proof 1 : Assume $\{Z_i\}$, where the independent and bounded r.v. $a \leq Z_i \leq b$. So by Lemma 1

$$\Pr\left(|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu| > \epsilon\right) \leq 2\exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right) \tag{8}$$

Set $\epsilon = \sqrt{\frac{\log(\frac{2}{\delta_h})}{2m}}$. Define $\delta_h = w(h) \cdot \delta$ for every h

$$\Pr\left(|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu| > \sqrt{\frac{\log(\frac{2}{\delta_n})}{2m}}\right) \leq 2\exp\left(-\frac{2m\frac{\log(\frac{2}{\delta_h})}{2m}}{(b-a)^2}\right) = \delta_n \tag{9}$$

Mark $Z_i = 1_{\{h(x_i) \neq y_i\}}$ and $\mu = \mathbb{E}[Z_i]$

$$\Pr\left(|\frac{1}{m}\sum_{i=1}^{m} 1_{\{h(x_i) \neq y_i\}} - \mu| > \sqrt{\frac{\log(\frac{2}{\delta_h})}{2m}}\right) \leq \delta_h \tag{10}$$

Denote that

$$\mathbb{E}[L_S] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} 1_{\{h(x_i) \neq y_i\}}\right] = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[1_{\{h(x_i) \neq y_i\}}] = \Pr(Y \neq h(X)) = L_D. \tag{11}$$

So for a fixed h

$$\forall h \in \mathcal{H} \quad \Pr\left(|L_S(h) - L_D(h)| > \sqrt{\frac{-\log(w(h)) + \log(\frac{2}{\delta})}{2m}}\right) \leq \delta_h \tag{12}$$

Applying the union bound

$$\Pr\left(|L_D - L_S| > \sqrt{\frac{-\log(w(h)) + \log(\frac{2}{\delta})}{2m}}\right) \leq \underbrace{\sum_{h\in\mathcal{H}} w(n)\delta}_{\leq 1} = \delta \tag{13}$$

Hence, we have

$$\Pr\left(|L_D - L_S| \leq \sqrt{\frac{-\log(w(h)) + \log(\frac{2}{\delta})}{2m}}\right) \geq 1 - \delta \qquad \blacksquare \tag{14}$$

When exploiting the MDL bound to optimize a decision tree, the most common method employed is pruning. The simple algorithm suggests testing each node of the tree with the following statement, which we define as a cost for the algorithm:

$$Cost(h) = \sqrt{\frac{-\log(w(h)) + \log(\frac{2}{\delta})}{2m}}. \tag{15}$$

Now all that's left is to calculate the cost with and without each node. If the cost difference is significant, prune (i.e. delete the node). *Pruning will be farther discussed in the next section*. An alternative approach is to use the MDL bound to set the tree's optimal depth, and then to use a modified algorithm to construct the tree by using combinations of features.

III. Avoiding over-fitting

A. Random forest

The general method of random decision forests was first proposed in 1995 by Ho[7] who established that if forests of trees split with oblique hyperplanes, are randomly restricted to be sensitive to only selected feature dimensions, they can gain accuracy as they grow without suffering from over-training. Random forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time. The output of the trees comprises the class, i.e. the mode (the value that appears most often), of the classes for classification problems, or the mean prediction of the individual trees for regression. Random decision forests correct the habit of decision trees to overfit to their training set.[5]

1) Algorithm: The idea behind random decision forests is to train many small trees with subsets of the samples. For each tree, we randomly select n samples of all the samples, for those n samples we choose (again randomly) k features that will be used to train the specific tree. When we have the new subsets, we should use the ID3 algorithm and Feedforward for each tree. For classification, we take the majority of all trees to make a decision, whereas for regression problems, we calculate the average of all trees to reach a decision. The reason for the randomness is to decrease the correlation between trees, and in so doing, to contribute to gains in accuracy. For example, if one or a few features are very strong predictors for the response variable (the label), these features will be selected in many of the B trees, causing them to become correlated.

Let B be the numbers of trees. For b = 1,...,B

- Randomly choose x' = $\{X_{i,j}\}$, a matrix with n rows (samples) and k columns (features).
- Train a classification or regression tree $f_b(x)$ on X'.

After training all trees, we want to make a decision. For classification, we will take the majority of all trees, and for regression, we will take the mean value:

$\hat{\text{f}}= \frac{1}{B}\sum_{b=1}^{B} f_b(x'_b)$

Typically, for a classification problem with p features,$\sqrt{p}$ (rounded down) features are used in each split. For regression problems the inventors recommend $\frac{p}{3}$ (rounded down) with a minimum node size of 5 as the default.

B. Pruning

The optimal final tree size in a decision tree algorithm is not a trivial matter. A tree that is too large risks overfitting the training data, thus rendering it poorly generalizable to new samples. Conversely, a tree that is too small might not capture important structural information about the sample space. Determining when a tree algorithm should stop, however, is difficult because one cannot know whether the addition of a single extra node will dramatically decrease error. This problem, which is known as the horizon

effect,, is commonly addressed by growing the tree until each node contains a small number of instances and then by using pruning to remove nodes that do not provide additional information. Pruning should reduce the size of a learning tree without reducing its predictive accuracy as measured by a cross-validation set. There are many techniques for tree pruning that differ in terms of the measurement that is used to optimize performance.[6]

1) Techniques: Generally speaking, tree pruning techniques can be classified as either top down - traverse nodes and trim subtrees starting at the root, or up - start at the leaf nodes. Each pruning technique has advantages and weaknesses that, if known and understood, can aid us in selecting the most appropriate method in every instance.Here we will learn two popular algorithms, one each from the top-down and bottom-up approaches.

1) Cost complexity pruning (top-down) -

   Let's define:

   - $\{T_i\}_{i=0,\ldots,m}$ - series of trees where T_0 is the original tree and T_m is the root alone.
   - Error Rate of a tree - $err(T, S)$ where T is the current tree and S is the data set.
   - Function - $prune(T, t)$ - defines the resultant tree after removal of subtrees t from T.
   - Function - $leaves(T)$ - defines the quantity of leaves at tree T.

   At each step,i, the tree is created by removing a subtree t from tree $i-1$ and replacing it with a leaf node whose value is chosen as in the tree building algorithm. First we find the error rate $err(T_i, S)$ for the current tree T_i and then we search for the subtree that minimizes the expression: $\frac{err(prune(T_i,t),S)-err(T_i,S)}{|leaves(T_i)|-|leaves(prune(T_i,t))|}$
   Once the series of trees has been created, the best tree is chosen base on generalized accuracy as measured by a training set or by cross-validation.

2) Reduced error pruning (bottom-up) -

   Starting at the leaves, each node is replaced with its most popular class. If the prediction accuracy is not affected then the change is kept. Very simple to apply,

this technique has the added advantage of being rapid.

IV. APPENDIX

Recall Lemma 1 - Let $X_1, X_2, ..., X_n$ be independent random variables bounded by $[a_i, b_i]$. Define the empirical mean of these by

$$\overline{X} = \frac{1}{n}(X_1 + X_2 + ... + X_n).$$

Lemma 1 states that:

$$\Pr(|\overline{X} - \mathbb{E}[\overline{X}]| \geq t) \leq 2\exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right). \tag{16}$$

Here $\mathbb{E}[\overline{X}]$ is the expected value of $\overline{X}$. Setting the following

$$S_n = X_1 + X_2 + ... + X_n. \tag{17}$$

We have:

$$P(|S_n - \mathbb{E}[S_n]| \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) \tag{18}$$

Lemma 2 (Hoeffding Lemma) Suppose X is a real random variable with mean equal to zero such that $P(X \in [a, b]) = 1$. Then

$$\mathbb{E}[e^{sX} \leq \exp(\frac{1}{8}s^2(b-a)^2).$$

Another inequality we need is

Lemma 3 (Markov's inequality) Let X be a non negative r.v. and $a > 0$, then:

$$P(X \geq a) \leq \frac{\mathbb{E}[X]}{a}.$$

Proof 2 Suppose $X_1, X_2, ..., X_n$ are n independent r.v. such that

$$P(X_i \in [a_i, b_i]) = 1, \quad 1 \leq i \leq n.$$

Let

$$S_n = X_1 + X_2 + ... + X_n.$$

Then for $s, t \geq 0$, the independence of X_i and Markov's inequality implies:

$$P(S_n - \mathbb{E}[S_n] \geq t) = P(e^{s(S_n - \mathbb{E}[S_n])} \geq e^{st}) \tag{19}$$

$$\leq e^{-st}\mathbb{E}\left[e^{s(S_n-\mathbb{E}[S_n])}\right] \tag{20}$$

$$= e^{-st}\prod_{i=1}^{n}\mathbb{E}\left[e^{s(X_i-\mathbb{E}[X_i])}\right] \tag{21}$$

$$\leq e^{-st}\prod_{i=1}^{n}e^{\frac{s^2(b_i-a_i)^2}{8}} \tag{22}$$

$$= \exp\left(-st+\frac{1}{8}s^2\sum_{i=1}^{n}(b_i-a_i)^2\right). \tag{23}$$

Now, for us to obtain the best possible upper bound, we need to find the minima of right-hand side of the last inequality as a function of s. Define $g:\mathbb{R}_+\xrightarrow{g}\mathbb{R}$ as

$$g(s) = -st+\frac{s^2}{8}\sum_{i=1}^{n}(b_i-a_i)^2$$

Note that $g(s)$ is a quadric function, and hence, its minimum is at

$$s = \frac{4t}{\sum_{i=1}^{n}(b_i-a_i)^2}.$$

Thus we get

$$P(S_n-\mathbb{E}[S_n]\geq t)\leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right) \tag{24}$$

$$P(|S_n-\mathbb{E}[S_n]|\geq t)\leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right) \quad \blacksquare \tag{25}$$

Refer to[4] for further reading and better understanding about this subject.

REFERENCES

[1] Shai Shalev-Shwartz. *Understanding Machine Learning course, Lecture No.4,* Hebrew University of Jerusalem, 2014.

[2] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms* May,2015.

[3] Quinlan, J. R. 1986. *Induction of Decision Trees* http://hunch.net/~coms-4771/quinlan.pdf Netherlands, 1986

[4] Hoeffding, Wassily. "Probability inequalities for sums of bounded random variables". *Journal of the American Statistical Association* ,Vol. 58, No. 301 (Mar., 1963), pp. 13-30

[5] Wikipedia https://en.wikipedia.org/wiki/Random_forest

[6] Wikipedia https://en.wikipedia.org/wiki/Pruning_(decision_trees)

[7] Ho, Tin Kam (1995). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 1416 August 1995. pp. 278282. http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf