Machine Learning
ICT-4261

By-
Dr. Jesmin Akhter
Professor
Institute of Information Technology
Jahangirnagar University

Contents

The course will mainly cover the following topics:

- ✔ A Gentle Introduction to Machine Learning
- ✔ Important Elements in Machine Learning
- ✔ Linear Regression
- ✔ Logistic Regression
- ✔ Naive Bayes
- ✔ Support Vector Machines
- ✔ Decision Trees and Ensemble Learning
- ✔ Neural Networks and Deep Learning
- ✔ Unsupervised Learning

Outline

- ✔ Decision Trees
  - ID3+CART

Solved Example

Consider the training examples shown in Table for a binary classification problem.

(a) What is the entropy of this collection of training examples with respect to the positive class?

(b) What are the information gains of a1 and a2 relative to these training examples?

(c) What is the best split (among a1 and a2) according to the information gain?

(d) What is the best split (between a1 and a2) according to the classification error rate?

(e) What is the best split (between a1 and a2) according to the Gini index?

Instance	Classification	a1	a2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T

Solved Example

Step 1: Calculate the entropy of the target class

The target class has 3 positive (+) and 2 negative (-) instances. The total number of instances is 5.

The probability of a positive instance is p(+) = 3/5 = 0.6 The probability of a negative instance is p(-) = 2/5 = 0.4

Entropy is calculated as: Entropy = $-p(+) * \log_2(p(+)) - p(-) * \log_2(p(-))$

Entropy = $-(3/5) * \log_2(3/5) - (2/5) * \log_2(2/5) \approx -0.6 * (-0.737) - 0.4 * (-1.322) \approx 0.442 + 0.529 \approx 0.971$

Step 2: Calculate the information gain for a1

First, let's analyze the distribution of the target class based on a1:

a1 = T: 2 positive, 1 negative a1 = F: 1 positive, 1 negative

Calculate the entropy for a1 = T: p(+) = 2/3, p(-) = 1/3 Entropy(a1=T) = $-(2/3)\log_2(2/3) - (1/3)\log_2(1/3) \approx 0.918$

Calculate the entropy for a1 = F: p(+) = 1/2, p(-) = 1/2 Entropy(a1=F) = $-(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$

Weighted average entropy for a1: (3/5) * Entropy(a1=T) + (2/5) * Entropy(a1=F) ≈ 0.551 + 0.4 = 0.951

Information Gain(a1) = Entropy(Target) - Weighted Entropy(a1) ≈ 0.971 - 0.951 ≈ 0.02

Step 3: Calculate the information gain for a2

Analyze the target class distribution based on a2:

a2 = T: 2 positive, 1 negative a2 = F: 1 positive, 1 negative

Notice this is the same distribution as a1. Therefore, the information gain for a2 will also be approximately 0.02.

Solved Example

Step 4: Determine the best split based on information gain

Both a1 and a2 have approximately the same information gain (0.02). Neither is significantly better than the other based on this metric.

Step 5: Determine the best split based on classification error rate

For a1: Error(a1=T) = 1/3 Error(a1=F) = 1/2 Weighted Error(a1) = (3/5)(1/3) + (2/5)(1/2) = 0.4

For a2: Error(a2=T) = 1/3 Error(a2=F) = 1/2 Weighted Error(a2) = (3/5)(1/3) + (2/5)(1/2) = 0.4

Both a1 and a2 have the same classification error rate.

Step 6: Determine the best split based on the Gini index

The Gini index measures impurity. A lower Gini index is better. The calculation is similar to entropy but uses a different formula. Since the distributions for a1 and a2 are identical, their Gini indices will also be the same.

Final Answer

(a) Entropy ≈ 0.971 (b) Information Gain(a1) ≈ 0.02, Information Gain(a2) ≈ 0.02 (c) Neither a1 nor a2 is significantly better based on information gain. (d) Neither a1 nor a2 is better based on classification error rate. (e) Neither a1 nor a2 is better based on the Gini index.

Solved Example

1. Given below is a summary table of a data set. Solve the following problems by hand (and a calculator).

A	B	C	Number of instances	
			Class = +	Class = -
T	T	T	5	0
F	T	T	0	20
T	F	T	20	0
F	F	T	0	5
T	T	F	0	0
F	T	F	25	0
T	F	F	0	0
F	F	F	0	25

Please note each row in the table represents a group of records with the same attribute values. There are 100 records in total: (5 + 20 + 25) positives and (20 + 5 + 25) negatives.

1) What is the entropy of this collection of training examples?

2) What are the information gains of A, B and C relative to these training examples?

3) What is the best split (among A, B and C) according to the information gain?

4) What is the best split (between A, B and C) according to the Gini index?

Question:

1) Develop a DT using entropy measures. Split the DT until level 2 (root at level 1) so that the (deepest) leaf nodes are located at level 3. After training the tree, calculate the total misclassification error rate of the DT on the same train set (i.e., resubstitution error).

2) Repeat part 1, but fix C as the split criterion at the root node and grow the rest of the tree.1

3) Compare misclassification rates of the two DTs. Which one is better? Based on the comparison, comment on the greedy strategy used in the DT algorithm.

Which is the best splitting attribute between attributes A, B, C

Solved Example

1) To develop a decision tree using entropy measures, we need to calculate the entropy for each attribute and choose the attribute with the highest information gain to split the tree.

First, let's calculate the entropy for the initial dataset: -

Number of positive instances (Class = +): 5 + 20 + 25 = 50

Number of negative instances (Class = -): 20 + 5 + 25 = 50

Entropy = - (p+ * log2(p+)) - (p- * log2(p-)) = - (50/100 * log2(50/100)) - (50/100 * log2(50/100))

= - (0.5 * log2(0.5)) - (0.5 * log2(0.5)) = - (0.5 * (-1)) - (0.5 * (-1))

Entropy = 1

Now, let's calculate the information gain for each attribute: -

A	B	C	Number of instances	
			Class = +	Class = -
T	T	T	5	0
F	T	T	0	20
T	F	T	20	0
F	F	T	0	5
T	T	F	0	0
F	T	F	25	0
T	F	F	0	0
F	F	F	0	25

Attribute A: -

Number of positive instances (A=T): 5 + 20 = 25

Number of negative instances (A = T) = 0

H(A=T)= 0,

Number of positive instances (A=F): 5 + 20 = 50

Number of negative instances (A = F) = 75

H(A=F)= .918296

information gain(A)=1-(25*0/1oo+75*.918296/100)=1-.6888=.3112

Solved Example

Attribute B: -
Number of positive instances (B=T) = 5+25=30
Number of negative instances (B = T) = 20
H(B=T)= 0.970946,
Number of positive instances (B=F)= 20
Number of negative instances (B= F) = 30
H(B=F)= 0.970946,
information gain(B)= 0.029054
Attribute C: -
Number of positive instances (C =T) = 25
Number of negative instances (C = T) = 25
H(C =T)= 1,
Number of positive instances (C =F)= 25
Number of negative instances (C = F) = 25
H(C =F)= 1,
information gain(C)= 0

A	B	C	Number of instances	
			Class = +	Class = -
T	T	T	5	0
F	T	T	0	20
T	F	T	20	0
F	F	T	0	5
T	T	F	0	0
F	T	F	25	0
T	F	F	0	0
F	F	F	0	25

Based on the information gain, the attribute with the highest information gain is A.
So A is the best splitting attribute.

Outline

- ✔ Decision Trees
  - Pruning

Pruning: Getting an Optimal Decision tree

- ✔ Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree. It is a method that can help us avoid overfitting. It helps in improving the performance of the tree by cutting the nodes or sub-nodes which are not significant. Additionally, it removes the branches which have very low importance.
- ✔ A too-large tree increases the risk of **overfitting**, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning.
- ✔ **How to avoid/counter Overfitting in Decision Trees?**
- ✔ The common problem with Decision trees, especially having a table full of columns, they fit a lot. Sometimes it looks like the tree memorized the training data set. If there is no limit set on a decision tree, it will give you 100% accuracy on the training data set, thus this affects the accuracy when predicting samples that are not part of the training set.
- ✔ In simpler terms, the aim of Decision Tree Pruning is to construct an algorithm that will perform worse on training data but will generalize better on test data.
- ✔ Here are two ways to remove overfitting:
  - Pruning Decision Trees.
  - Random Forest

Pruning: Getting an Optimal Decision tree

- ✔ In **pruning**, you trim off the branches of the tree, i.e., remove the decision nodes starting from the leaf node such that the overall accuracy is not disturbed. This is done by segregating the actual training set into two sets: training data set, D and validation data set, V. Prepare the decision tree using the segregated training data set, D. Then continue trimming the tree accordingly to optimize the accuracy of the validation data set, V.

There are two types of pruning:

- ✔ **Pre-pruning** – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance **while growing** the tree.
- ✔ **Post-pruning** – once our **tree is built to its depth**, we can start pruning the nodes based on their significance.

Pre-pruning

- ✔ The pre-pruning technique of Decision Trees is tuning the hyperparameters prior to the training pipeline. It involves the heuristic known as ‘early stopping’ which stops the growth of the decision tree - preventing it from reaching its full depth.
- ✔ It stops the tree-building process to avoid producing leaves with small samples. During each stage of the splitting of the tree, **the cross-validation error will be monitored**. If the value of the **error does not decrease anymore -** then we **stop the growth of the decision tree.**
- ✔ The hyperparameters that can be tuned for early stopping and preventing overfitting are:
  max_depth, min_samples_leaf, and min_samples_split
- ✔ These same parameters can also be used to tune to get a robust model. However, you should be cautious as early stopping can also lead to underfitting.
- ✔ These pre- pruning parameters: max_depth , minimum_sample_leaf often require careful tuning to find the right balance between underfitting and overfitting.

Pre- Pruning Techniques

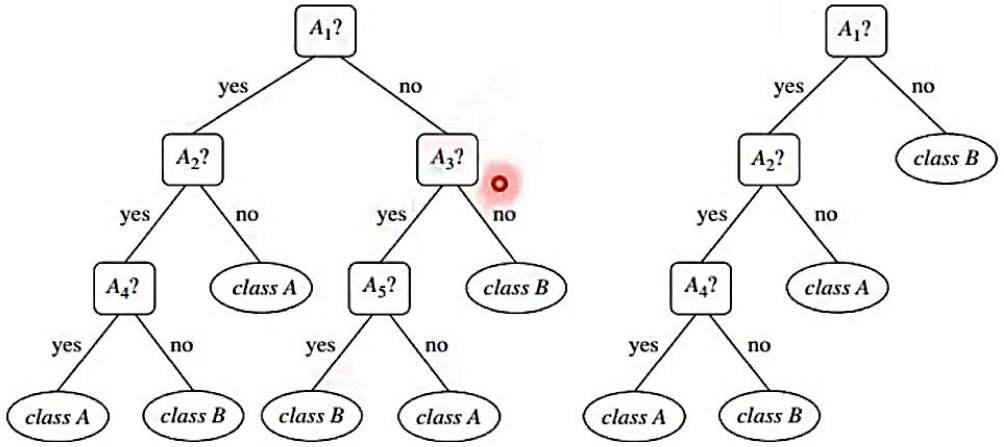
- ✔ There are several strategies for pruning:
- ✔ These all techniques are **“hyperparameters”** of the “Decision Tree”
- ✔ **1. Maximum Depth:-**
- ✔ It reduces the depth of the tree. It is one of the simplest forms of pre-pruning. So starting only will decide the number of max_depth so the tree won't grows after the given depth.
- ✔ **If Max_depth is “None” — Overfit**
- ✔ **If max_depth is “1” — Underfit**
- ✔ Clearly a “Bias variance trade-off” happening so we need to provide a value of max_depth which is not very less or max.

Pre- Pruning Techniques

- ✔ **2. Minimum Sample Split:-**
- ✔ This is a condition where a node only be split if the number sample of that node is above a certain threshold. If the **number of samples is too small**, then the node is not split and become a leaf node instead. This can prevent overfitting by not allowing the model to learn noise in the data.
  - Choosing the number wisely is important. This number of samples should not be very high **-** tree will be very small.
  - **Here we control on "decision node level/ "Parent level"**
- ✔ **3. Minimum Sample Leaf:-**
- ✔ This condition requires that a split at a node must leave at least a minimum number of training examples in each of the leaf nodes. Like the minimum sample split, this strategy can prevent overfitting by not allowing the model to learn from noise in the data based on the given sample leaf.
  - Ex- if sample leaf has given 50 then after this number won't split further.
  - **Here we control on "leaf level/ "child level"**

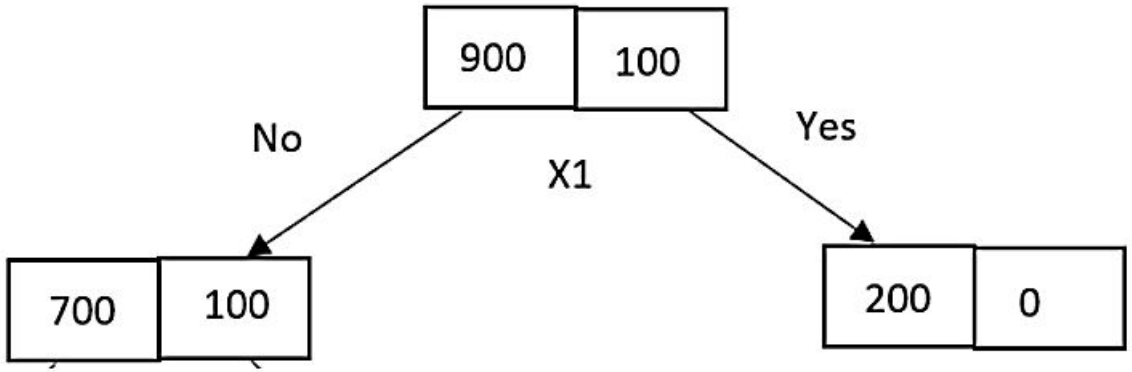
Post-pruning

- ✔ Post-pruning does the opposite of pre-pruning and allows the Decision Tree model to grow to its full depth. Once the model grows to its full depth, tree branches are removed to prevent the model from overfitting.
- ✔ The algorithm will continue to partition data into smaller subsets until the final subsets produced are similar in terms of the outcome variable. The final subset of the tree will consist of only a few data points allowing the tree to have learned the data to the T. However, when a new data point is introduced that differs from the learned data - it may not get predicted well.
- ✔ The hyperparameter that can be tuned for post-pruning and preventing overfitting is: ccp_alpha: ccp stands for Cost Complexity Pruning and can be used as another option to control the size of a tree. A higher value of ccp_alpha will lead to an increase in the number of nodes pruned.



Post-pruning

- Consider 900 “positive” samples and 100 “negative” samples. Let’s assume the X1 attribute is used for splitting at the parent node. Consider the following decision tree with unequal distribution of data samples after splitting.

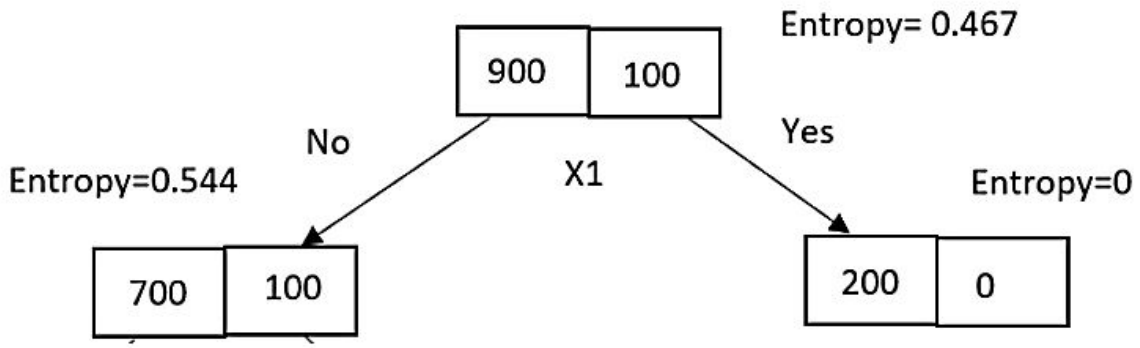


Decision Tree

- It has one pure node classified as 200 “positive” samples and an impure node with 700 “positive” and 100 “negative” samples.

Post-pruning

- ✔ With **entropy** as a loss function, parent loss is 0.467, and children loss is 0.544. As one node is pure, the entropy is zero, and the impure node has a non-zero entropy value.

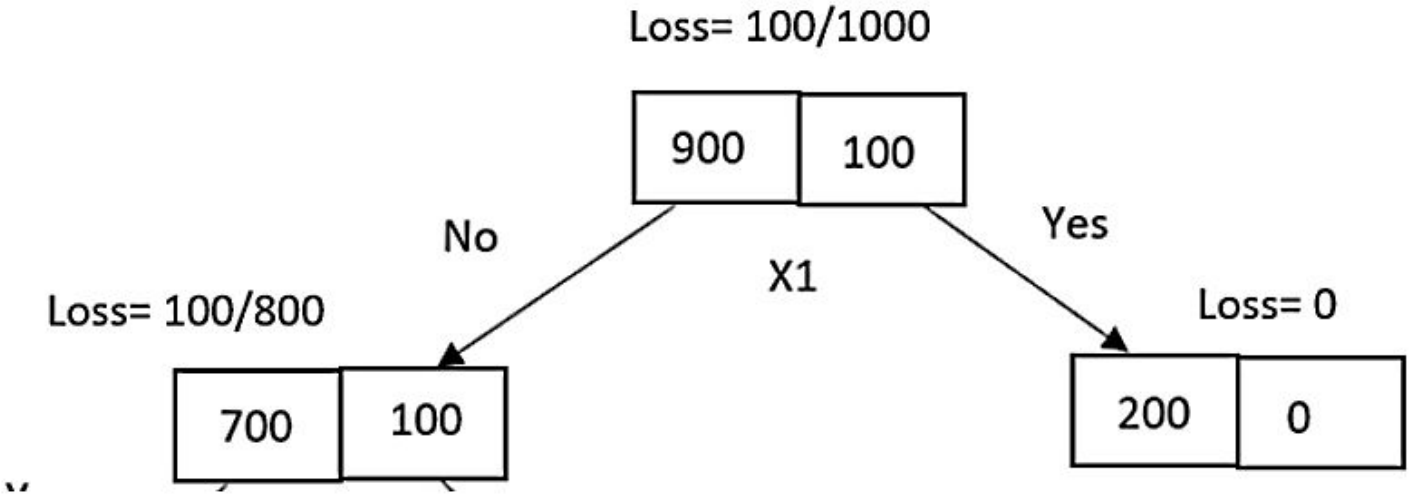


A decision tree with entropy values

- ✔ Using the information gain formula, the loss reduction from parent to children region is calculated as,
- ✔ Gain = Entropy(parent)—[Entropy(left child)*(No of samples in left child/No of samples in parent) + Entropy(right child)*(No of samples in right child/No of samples in parent)]
- ✔ Gain = 0.467 –[0.544*(800/1000) + 0 *(200/1000)]
- ✔ Gain = 0.0318

Post-pruning

- ✔ With a **misclassification error**, parent loss is 0.1, and children loss is 0.125.

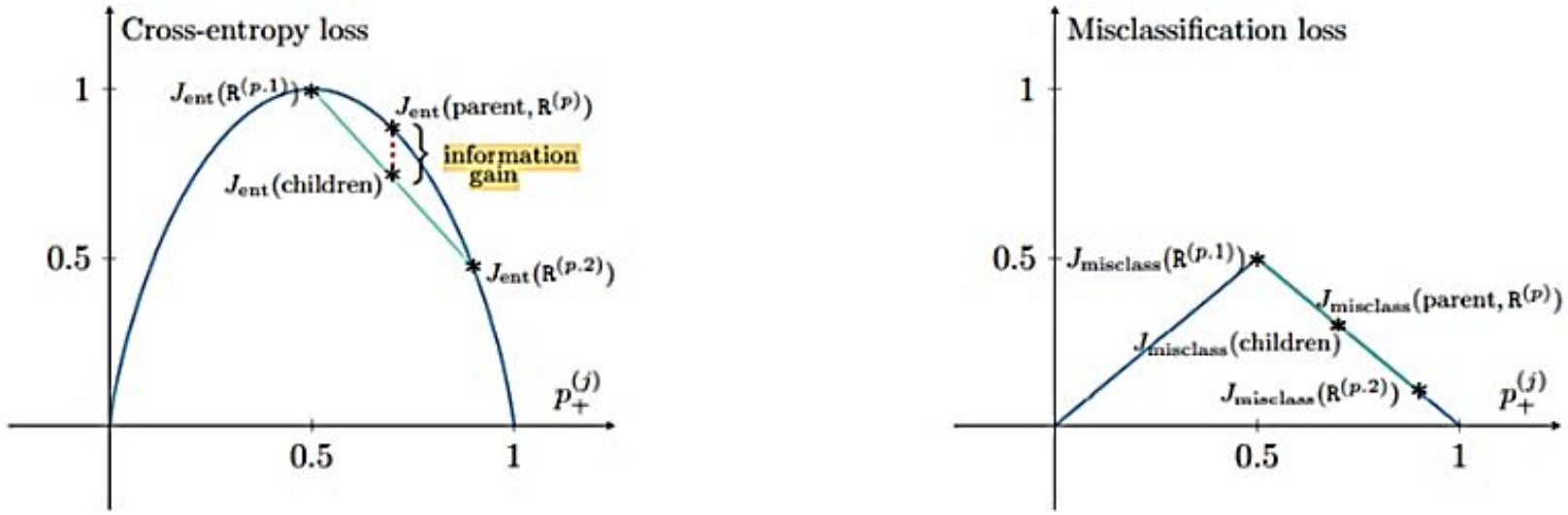


A decision tree with misclassification loss

- ✔ The information gain is calculated as,
- ✔ Gain = ME(parent)—[ME(left child)*(No of samples in left child/No of samples in the parent) + ME(right child)*(No of samples in right child/No of samples in the parent)]
- ✔ Gain = (100/1000)—[(100/800)*(800/1000) + 0*(200/1000)]
- ✔ Gain =0

Post-pruning

- ✔ From the obtained gain values, as the misclassification error has not gained any information hence, further splitting of the tree is not required, and the decision tree is stopped growing. But in the case of entropy, the decision tree can be partitioned further until the leaf node is reached and the entropy value becomes zero.
- ✔ The graphs are plotted with the assumption of an even split of data into two nodes. The cross-entropy function has **concave** nature that proves the loss of children is always less than that of the parent. But this is not the case with the misclassification error. Hence the children and parent loss are equal.
- ✔ The **Gini impurity** has the same nature as entropy which is also preferred for decision tree building over misclassification loss.



Entropy and misclassification error graphs.

Cost-Complexity Pruning

- ✔ Cost-complexity pruning is a common method for post-pruning decision trees. It involves assigning a cost to each subtree in the fully grown tree and then selecting the subtree with the smallest cost as the pruned tree. The cost of a subtree is determined by a complexity parameter (often denoted as alpha) and the number of leaf nodes in the subtree.
- ✔ **Calculate Cost for Subtrees:** For each subtree, calculate the total error (such as misclassification rate) on the validation data and add a complexity penalty based on the number of leaf nodes and the complexity parameter.

  Cost = Error + alpha * (Number of Leaf Nodes)
- ✔ **Select the Best Subtree:** Choose the subtree with the smallest cost as the pruned tree.

Algorithm

Pruning Algorithm:

- Initialization:
  - let T^1 be the tree obtained with $\alpha^1 = 0$
  - by minimizing $R(T)$
- Step 1
  - select node $t \in T^1$ that minimizes
    - $g_1(t) = \frac{R(t) - R(T_t^1)}{|f(T_t^1)| - 1}$
  - let t_1 be this node
  - let $\alpha^2 = g_1(t_1)$ and $T^2 = T^1 - T_{t_1}^1$
- step i
  - select node $t \in T^i$ that minimizes
    - $g_i(t) = \frac{R(t) - R(T_t^i)}{|f(T_t^i)| - 1}$
  - let t_i be this node
  - let $\alpha^{i+1} = g_i(t_i)$ and $T^{i+1} = T^i - T_{t_i}^i$

- we have 3 inner nodes where we can prune: $t_1 \equiv \text{root}, t_2, t_3$

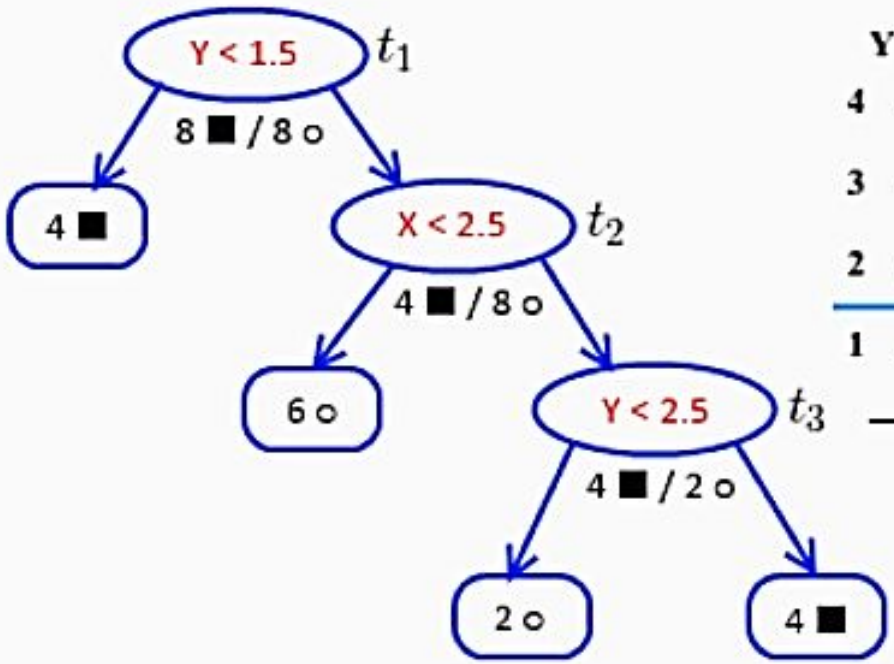
ome formulas:

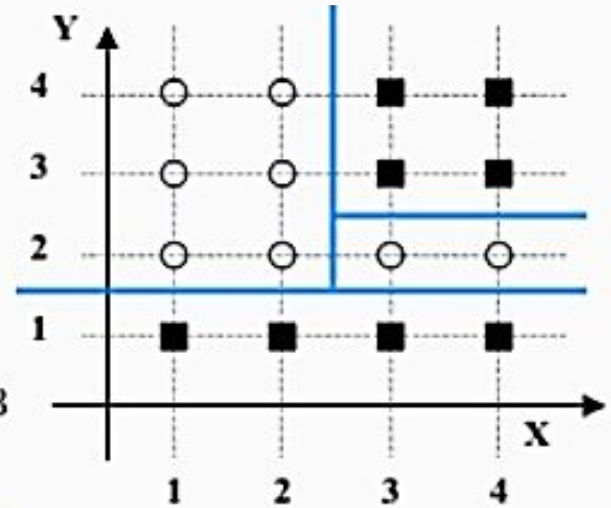
- $R(T_t)$ - training error of a subtree T_t - a tree with root at node t
  - $R(T_t) = \sum_{l \in f(T_t)} R(l)$ - sum of all training errors over all leaves
- $R(t)$ - training error of node t
  - $R(t) = r(t) \cdot p(t)$
  - $r(t)$ - misclassification error at this node (without considering the leaves)
  - $p(t)$ - proportion of data items reached t (i.e. # of items reached t divided by # of training items)
- $g(t) = \frac{R(t) - R(T_{T_t})}{|f(T_t)| - 1}$
  - $|f(T_t)| - 1$ is the number of leaves to prune

Example 1

Suppose we have the following tree:

- we want to prune it
- we have 3 inner nodes where we can prune: $t_1 \equiv \text{root}, t_2, t_3$

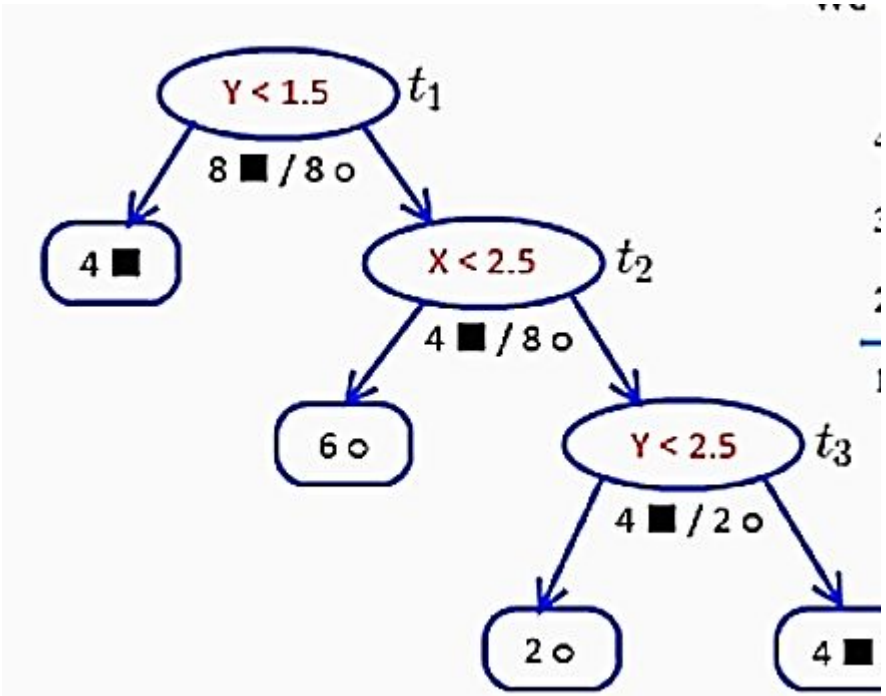




Iteration 1:

- let $\alpha^{(1)} = 0$

t	$R_(t)$	$R(T_t)$	$g(t)$
t_1	$\frac{8}{16} \cdot \frac{16}{16}$	T_{t_1} - the entire tree all leaves are pure $R(T_{t_1}) = 0$	$\frac{8/16 - 0}{4 - 1} = \frac{1}{6}$
t_2	$\frac{4}{12} \cdot \frac{12}{16} = \frac{4}{16}$ (there are 12 records, 4 ■ + 8 ◯)	$R(T_{t_2}) = 0$	$\frac{4/16 - 0}{3 - 1} = \frac{1}{8}$
t_3	$\frac{2}{6} \cdot \frac{6}{16} = \frac{2}{16}$	$R(T_{t_3}) = 0$	$\frac{2/16 - 0}{2 - 1} = \frac{1}{8}$



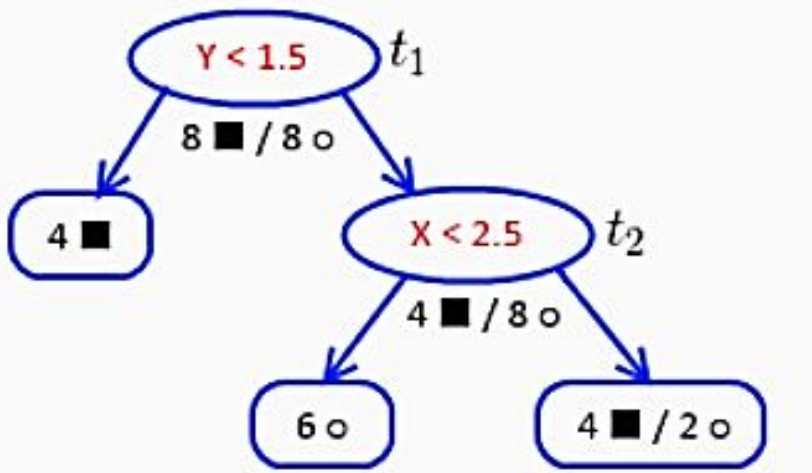
2-1

We want to find the minimal $g(t)$

- it's $g(t_2)$ and $g(t_3)$
- in case of a tie, we choose the one that prunes fewer nodes
- i.e. $g(t_3)$
- so prune at t_3
- let $\alpha^{(2)} = 1/8$ (the min $g(t)$)

We want to find the minimal $g(t)$

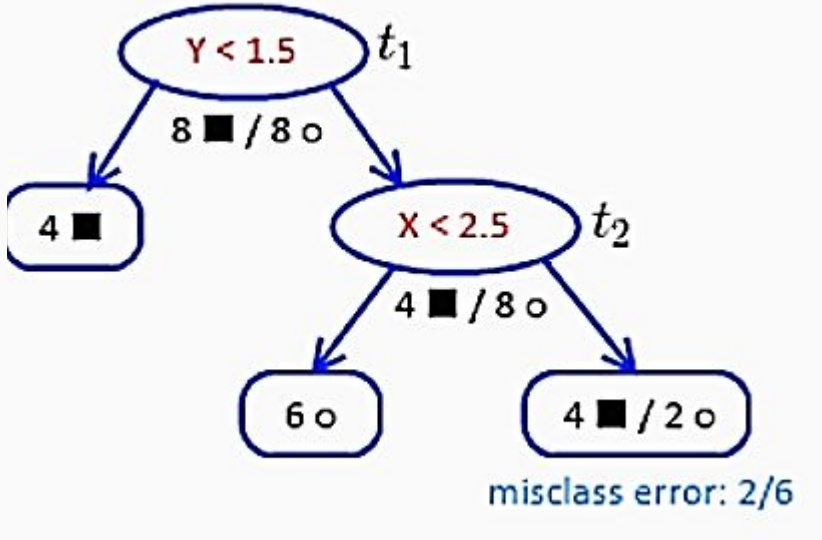
- it's $g(t_2)$ and $g(t_3)$
- in case of a tie, we choose the one that prunes fewer nodes
- i.e. $g(t_3)$
- so prune at t_3
- let $\alpha^{(2)} = 1/8$ (the min $g(t)$)



misclass error: 2/6

Iteration 2:

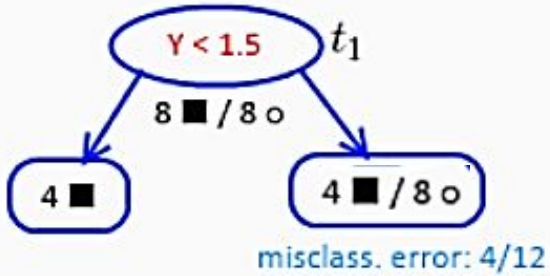
- in the tree now we have only candidates: t_1 and t_2



t	$R_(t)$	$R(T_t)$	$g(t)$
t_1	$\frac{8}{16} \cdot \frac{16}{16}$	$\frac{2}{16}$	$\frac{8/16 - 2/16}{3-1} = \frac{6}{32}$
t_2	$\frac{4}{12} \cdot \frac{12}{16}$	$\frac{2}{16}$	$\frac{4/16 - 2/16}{2-1} = \frac{1}{8}$

Find minimal $g(t)$:

- it's $g(t_2) = 1/8$
- let $\alpha^{(3)} = 1/8$
- prune at t_2
-



Iteration 3:

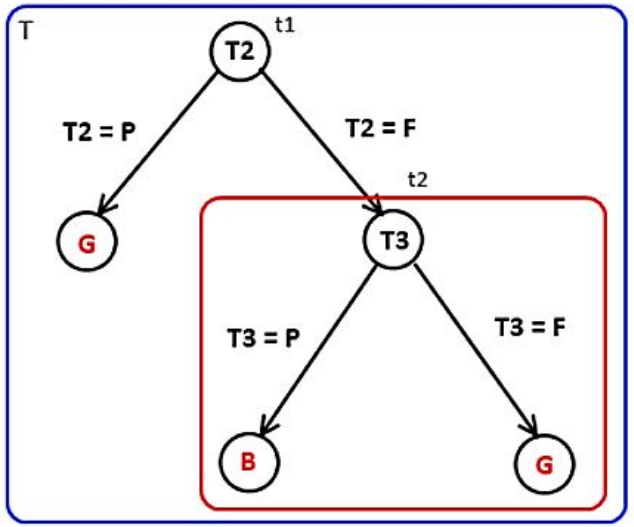
- only one candidate for pruning: t_1
- $\alpha^{(4)} = g(t_1) = \dfrac{8/16 - 4/16}{2 - 1} = \dfrac{1}{4}$

Selecting the best:

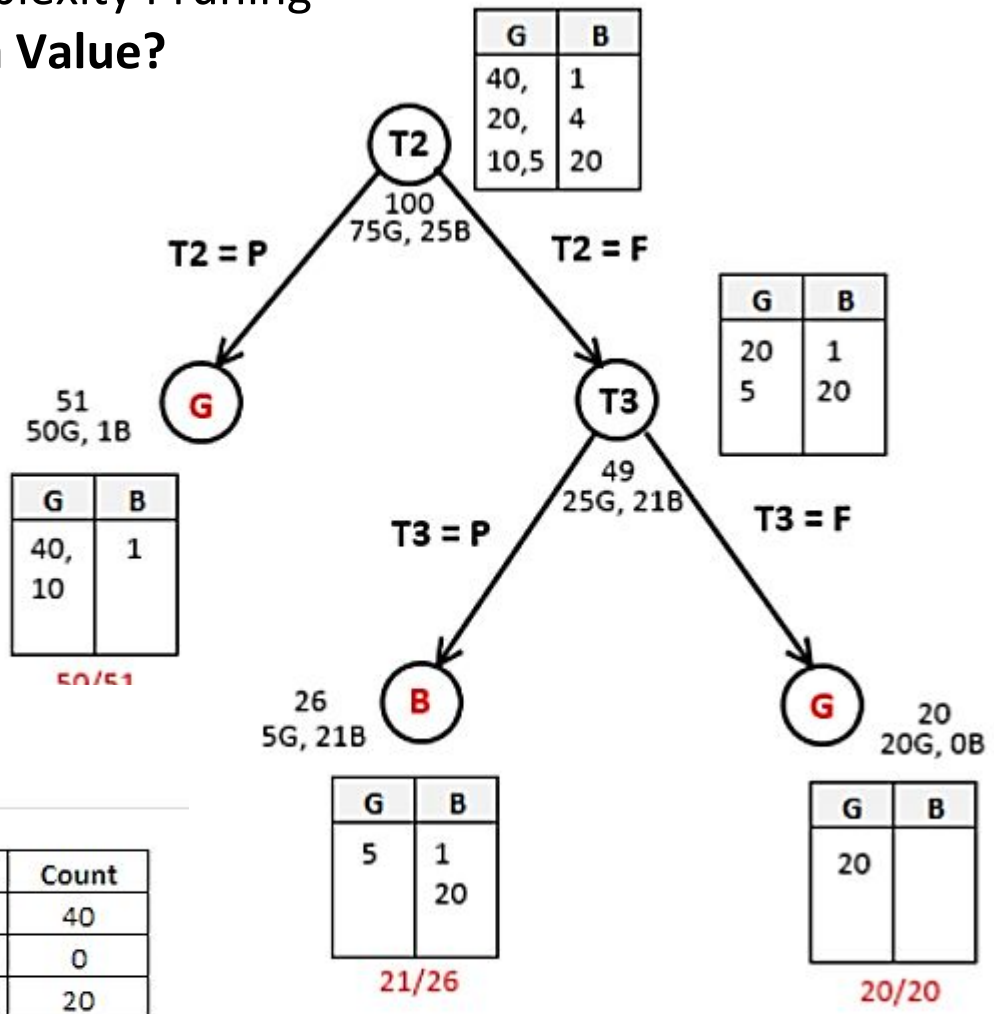
- we have these values: $\alpha^{(0)} = 0, \alpha^{(1)} = 1/8, \alpha^{(2)} = 1/8, \alpha^{(3)} = 1/4$
- by the theorem we want to find tree such T that minimizes the cost-complexity function
  - if $0 \geqslant \alpha < 1/8$, then T_1 is the best
  - if $\alpha = 1/8$, then T_2 is the best
  - if $1/8 < \alpha < 1/4$, then T_3 is the best
- to choose α use Cross-Validation

Example 02 Minimal Cost-Complexity Pruning

How to Choose the Best Alpha Value?



So we evaluate the entire tree on the validation set



G	B
40, 20, 10,5	1 4 20

G	B
40, 10	1

G	B
20 5	1 20

G	B
5	1 20

G	B
20	

T1	T2	T3	S	Count
P	P	P	Good	40
P	P	F	Bad	0
P	F	F	Good	20
P	F	P	Bad	1
F	P	P	Good	10
F	P	F	Bad	4
F	F	P	Good	5
F	F	P	Bad	20

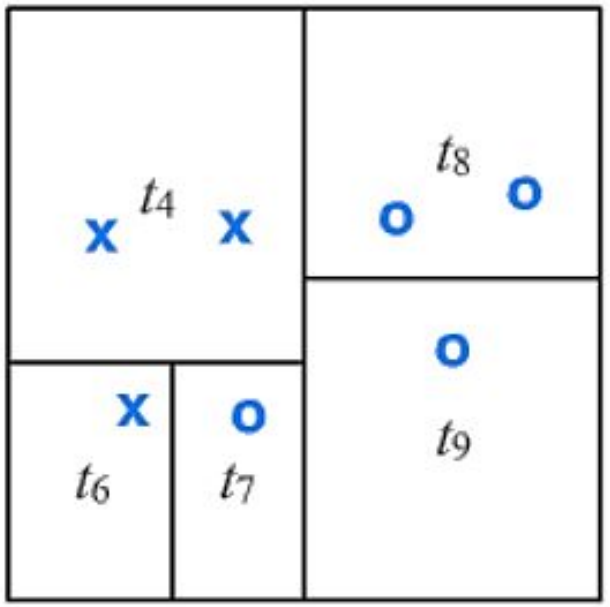
Candidate	R(t)	R(T_t)	g_i(t)
t1	25/100 * 100/100 (1 - 75/100) this node is root so it covers all the nodes (100/100)	T_t1 - the entire tree 6/100 are misclassified	(25/100 - 6/100)/(3 - 1) = 19/100 / 2 = **19/200**
t2	21/49 * 49/100 covers only 49 out of 100	T_t2 - tree with root at t2 5/100 are misclassified	(21/100 - 5/100)/(2 - 1) = 16/100 = 32/200

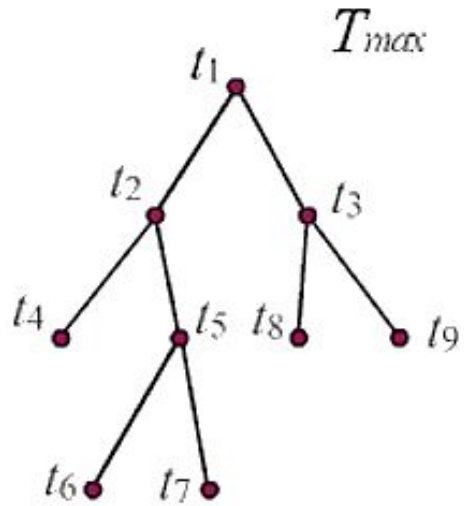
T_t1 minimizes g_i(t) value - so selecting it as the best tree, i.e. prune the tree to the root

\alpha^0 = 0, \alpha^1 = 19/200

Example 03 Minimal Cost-Complexity Pruning

Find Cost-Complexity parameter α





✔

α=0 (Full Tree)

α=1/14(without considering t8 and t9)

Sample Q.

Q. Does pruning reduce complexity?

Yes, pruning reduces complexity in decision trees by trimming unnecessary branches and nodes. By eliminating redundant splits and reducing the number of leaf nodes, pruning simplifies the tree structure, making it more interpretable and improving its generalization capability.

Q. How to choose α in cost-complexity pruning?

Choosing α in cost-complexity pruning involves finding the value that optimally balances model complexity and predictive accuracy. This can be achieved through techniques such as cross-validation, where different values of α are evaluated on validation data to select the one that minimizes a specified criterion, such as error rate or information gain.

Thank You