Optimal Decision Tree

Xinlong Li

Outline

- Widely used algorithm for generating decision tree
    - ID3
    - C4.5
    - CART

- Local optimal decision tree
    - Post Prune
    - Cross-validation

- The efforts of Finding the Global Optimal decision tree
    - T2, T3, T3C
    - OCT

ID3 Algorithm

- Entropy Ent(D) is a measure of the amount of uncertainty in the data set D

$$Ent(D) = -\sum_{k=1}^{K} p_k \cdot log_2 p_k$$

where

- k is the set of classes in D
- p_k is the proportion of the number of elements in class k

- Information gain Gain(D,a) is the measure of the difference in entropy from before to after the D is split on an feature a. In other words, how much uncertainty in D was reduced after splitting set D on feature a.

$$Gain(D, a) = Ent(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} Ent(D^v)$$

where

- V is the number of values in feature a
- D^v is the number of samples which have value v in feature a

ID3 Algorithm

- ID3 uses a greedy strategy by selecting the locally best feature, which has the largest information gain, to split the dataset on each iteration

- ID3 is harder to use on numerical data than on categorical data

- ID3 does not guarantee an global optimal solution. It usually converge to a local optima

- ID3 is possible to overfit the training data

- ID3 tends to choose the feature, which has more values, to be the best one

C4.5 Algorithm

- C4.5 is an extension of ID3 algorithm

- C4.5 uses normalized information gain ratio instead of information gain

$$GainRatio(D, T) = \frac{Gain\ (D,T)}{IV(T)}$$

where

$$IV(T) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} log_2 \frac{|D^v|}{|D|}$$

Is the intrinsic value of feature T

- As an improvement, to handle continuous attributes, C4.5 creates a threshold and then splits the list into those whose attribute value is above the threshold and those that are less than or equal to it

- C4.5 prunes trees after creation

CART Algorithm

- Gini impurity is a measure of how often a randomly chosen element from the set T would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

where

- p_j is the fraction of items labeled with class j in the data set T

- Gini gain $gini_{split}(T)$ after splitting on feature a

$$Gini_{split}(T) = \sum_{i=1}^{2} \frac{N_i}{N} gini(T_i)$$

Where

- N_i is the number of samples belonging to the i-th value in feature a
- T_j is the subset belonging to the i-th value in feature a

CART Algorithm

- CART selects the locally best feature, which has the smallest Gini gain, to split the dataset on each iteration

- CART generates binary tree

- CART is able to handle both numerical and categorical data

- CART can can overfit the training data, needs pruning trees after generation

Optimal Decision Tree

- Two or more decision trees may represent the same distribution (giving the same classification result). The one with smallest size (with the fewest leaves) is the optimal decision tree

- ID3, C4.5, CART are all top-down induction methods, usually generating locally optimal trees.

- Constructing global optimal binary decision trees is NP-hard

- In financial field where computations are sometimes required to be done virtually online or at least to be conducted very quickly, the speed matter becomes crucial, that is why it is reasonable to apply locally optimal procedures

- Nowadays commercial versions of software capable of producing globally optimal structures are not present on the market

Methods to Stop Splitting

- In CART, Set up a threshold, stop splitting when Gini gain is smaller than this threshold

- Set up a restriction on the minimum number of samples at each terminal leaf

The above two simple methods usually don't result in an (local) optimal tree

- Cross-validation, usually 10-fold validation is used to compare the quality of different trees

- Get the optimal decision tree through cost-complexity pruning

Cost-complexity Function and Cross-validation

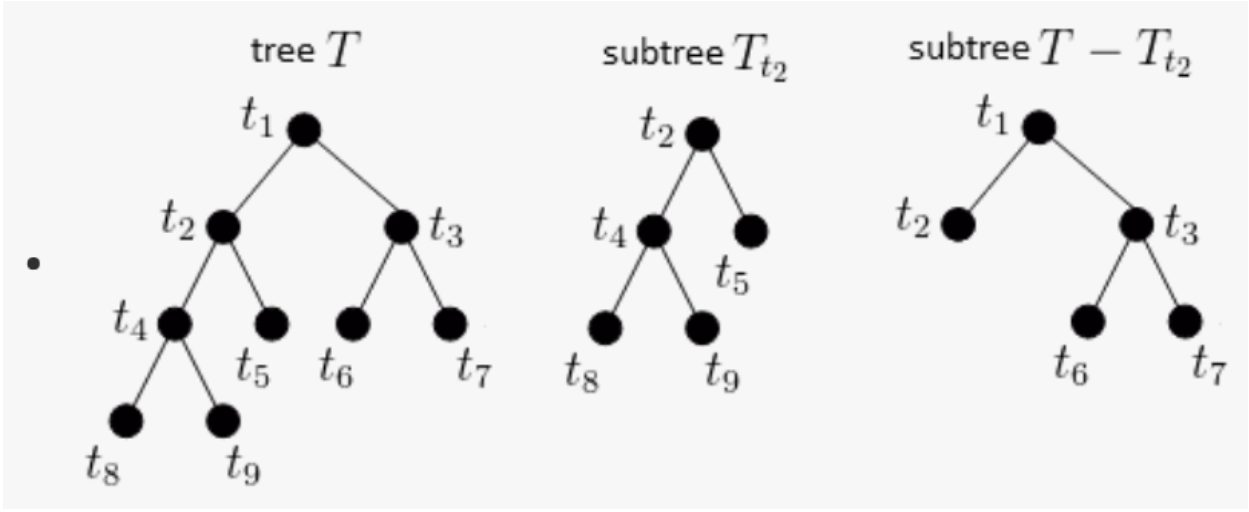
- Goal: to minimize the cost-complexity function $R_\alpha(T)$:

$$R_\alpha(T) = R(T) + \alpha \cdot |f(T)|$$

where:

- R(T) is the training/learning error related to the misclassification rate
- |f(T)| is the size of the tree T or the number of terminal leaves

- The value of α determines which tree is optimal



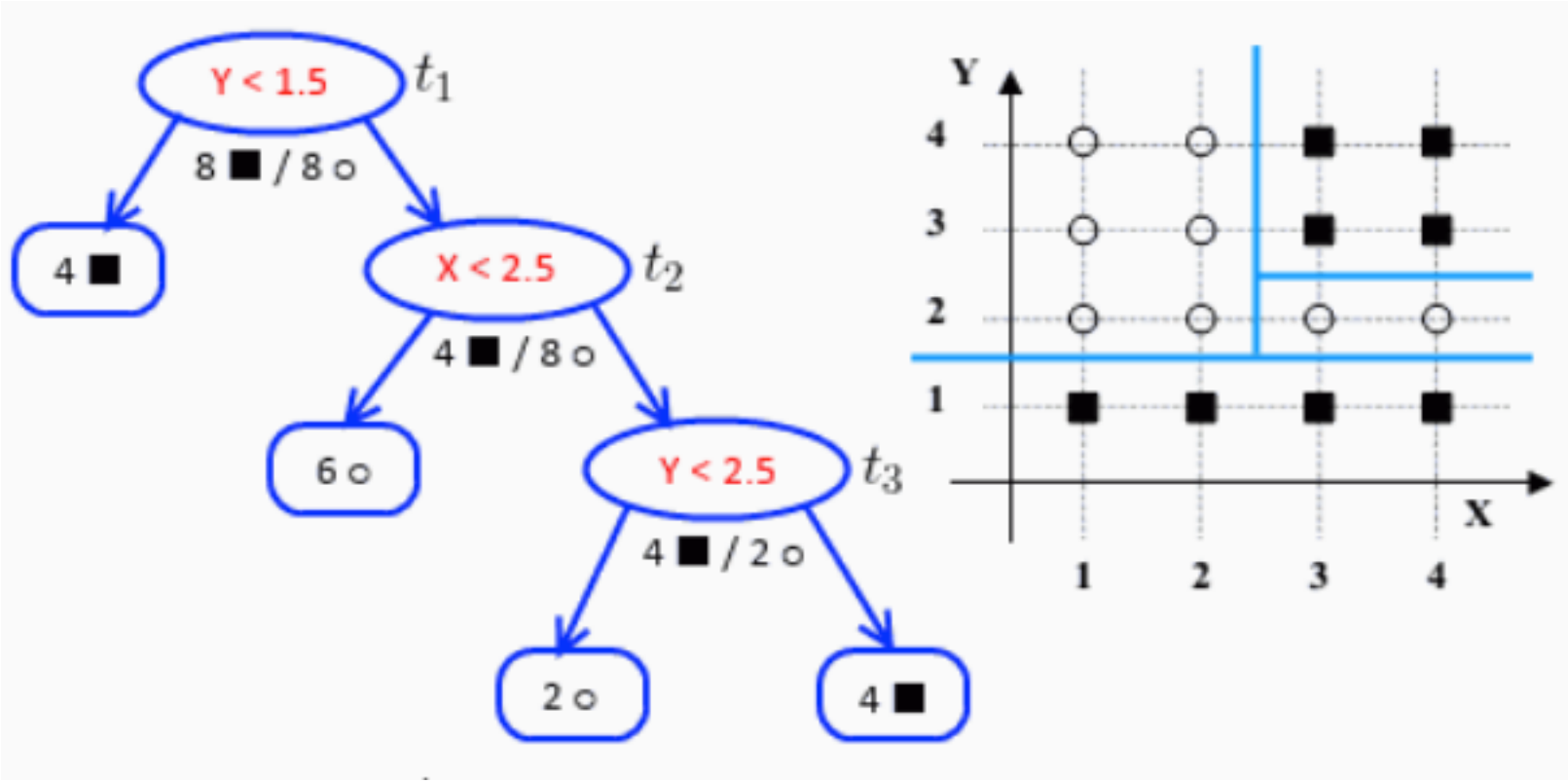
- We need to find the critical values of α, and determine which value is the best

Pruning Algorithm

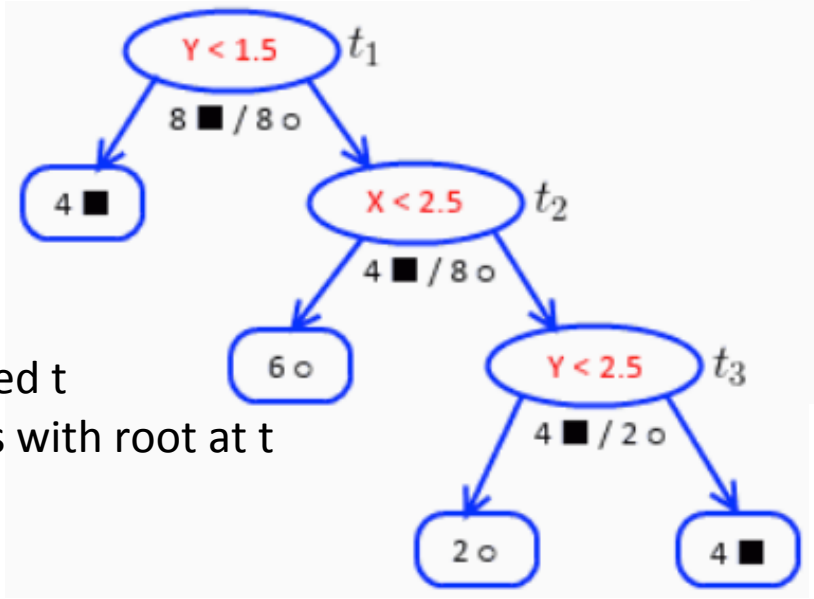
- After some mathematical derivation, we have following algorithm

- 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
    - $g_1(t) = \dfrac{R(t) - R(T_t^1)}{|f(T_t^1)| - 1}$
  - 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
    - $g_i(t) = \dfrac{R(t) - R(T_t^i)}{|f(T_t^i)| - 1}$
  - let t_i be this node
  - let $\alpha^{i+1} = g_i(t_i)$ and $T^{i+1} = T^i - T_{t_i}^i$

Pruning Algorithm – Example Part 1



Pruning Algorithm – Example Part 1



- R(t) is training error of node t
  - R(t) = r(t) p(t)
  - r(t) is the misclassification rate at node t
  - p(t) is the proportion of data items reached t
- $R(T_t)$ is sum of all training errors over all leaves with root at t

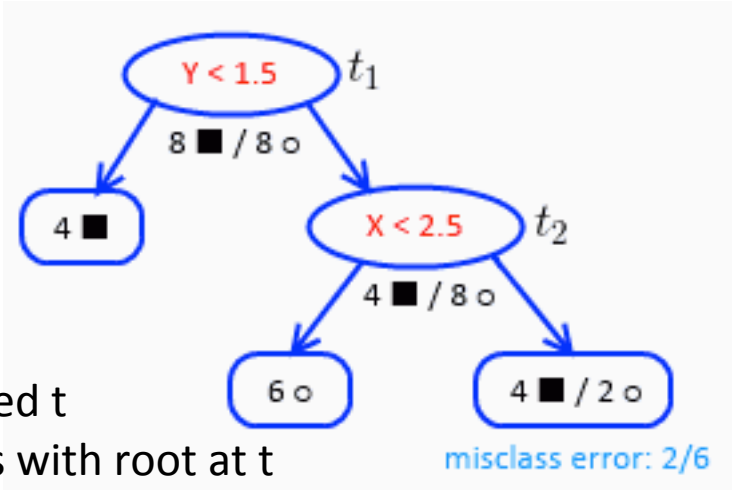
$$g(t) = \frac{R(t) - R(T_t)}{|f(T_t)| - 1}$$

- $|f(T_t)|$ is the number of leaves to prune

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	$\frac{2}{6} \cdot \frac{6}{16} = \frac{2}{16}$	$R(T_{t_3}) = 0$	$\frac{2/16 - 0}{2 - 1} = \frac{1}{8}$

$\alpha_1 = 1/8$

Pruning Algorithm – Example Part 2



- R(t) is training error of node t
  - R(t) = r(t) p(t)
  - r(t) is the misclassification rate at node t
  - p(t) is the proportion of data items reached t
- $R(T_t)$ is sum of all training errors over all leaves with root at t

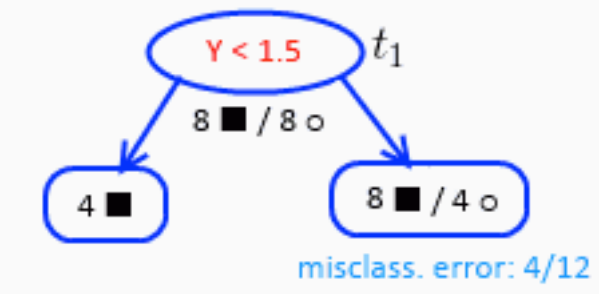
$$g(t) = \frac{R(t) - R(T_t)}{|f(T_t)| - 1}$$

- $|f(T_t)|$ is the number of leaves to prune

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	$\frac{4}{12} \cdot \frac{12}{16}$	$\frac{2}{16}$	$\frac{4/16 - 2/16}{2 - 1} = \frac{1}{8}$

$\alpha_2 = 1/8$

Pruning Algorithm – Example Part 3



- R(t) is training error of node t
  - R(t) = r(t) p(t)
  - r(t) is the misclassification rate at node t
  - p(t) is the proportion of data items reached t
- $R(T_t)$ is sum of all training errors over all leaves with root at t

$$g(t) = \frac{R(t) - R(T_{T_t})}{|f(T_t)| - 1}$$

- $|f(T_t)|$ is the number of leaves to prune

$$g(t_1) = \frac{8/16 - 4/16}{2 - 1} = \frac{1}{4}$$

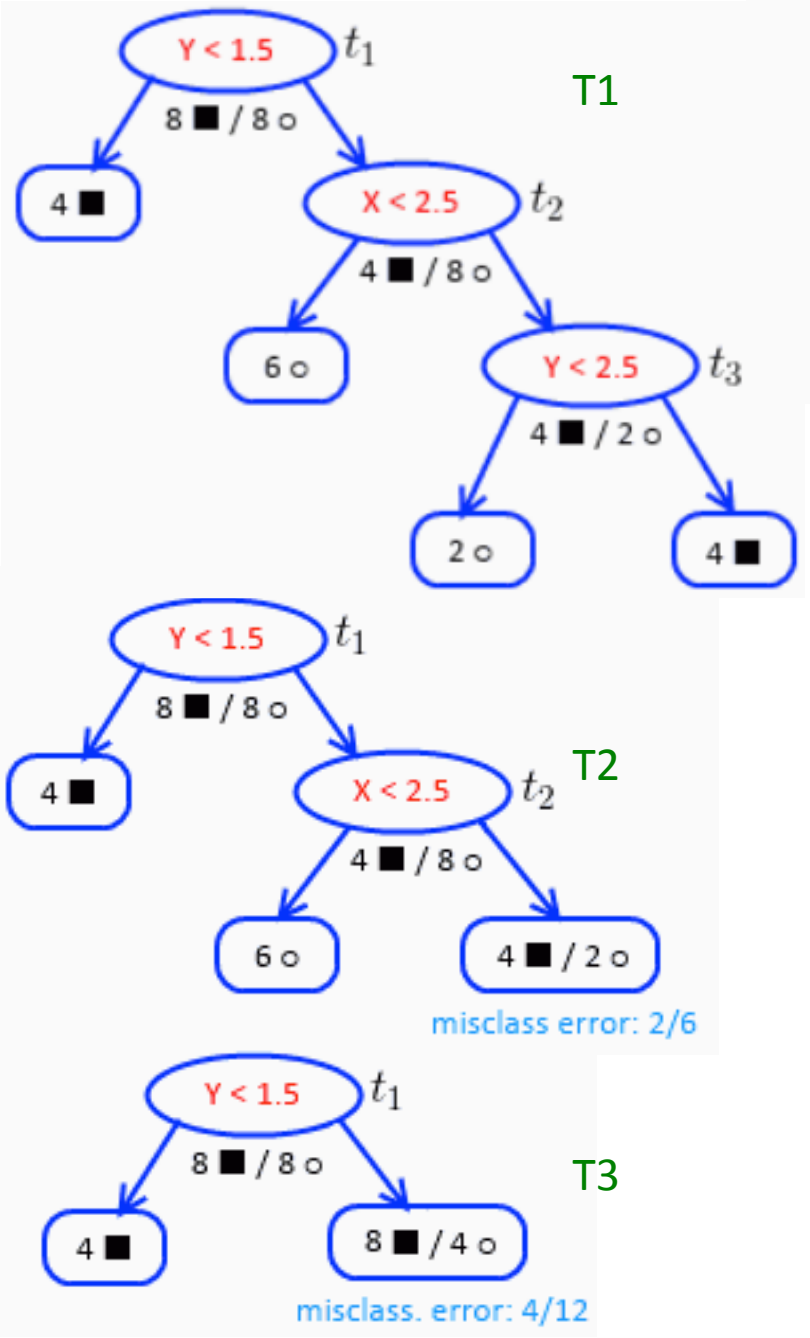
$\alpha_3 = 1/4$

Pruning Algorithm – Example Part 4

- We now have a series of 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 $0 \leq \alpha < 1/8$, then T1 is the best
  - if $\alpha = 1/8$, then T2 is the best
  - if $1/8 < \alpha < 1/4$, then T3 is the best
  - if $\alpha \geq 1/8$, then T3 is the best

- Next we use cross validation to determine which α is the best and thus which tree is the optimal tree



Finding the Global Optima - T3C

- ID3, C4.5, CART are all top-down induction methods. The choice of split cannot be guided by the possible influence of future splits.

- To get global optimal tree, we need to form the entire decision tree in a single step, allowing each split to be determined with full knowledge of all other splits.

- A family of efficient enumeration approaches, called T2, T3 and T3C, which create optimal non-binary decision trees of depths up to 3

- T3C can deal with both categorical and numerical data

- T3C splits a node into more than 2 branches

- T3C generates smaller trees with the same performance compared to C4.5 and C5.0

Finding the Global Optima - T3C

- Tree BuildTree (ItemNo Fp, ItemNo Lp, int Dep, ClassNo PreviousClass)

Algorithm 1 Tree BuildTree

1: **if** Fp..Lp is empty **then**
2: terminate and return a leaf node, which has PreviousClass as the winning class, and relative frequency and error rate as 0.
3: **else**(i.e. if there remain items for processing)
4: calculate the class distribution of this list of items, select the winning class and initialise the relative frequency of this class, by dividing the number of occurrences of this class by the total number of class occurrences in this set.
5: Create a new leaf called Node for this set, using this winning class, and the relative frequency and error rate, which have been calculated before.
6: **if** error rate $\leq$ MAE or Dep=0 **then**
7: **return** this Node and terminate
8: **else**
9: build BestNode with Node, using the function *Build* described below.
10: Release the Node created
11: **return** BestNode, and terminate

Finding the Global Optima - T3C

- Tree Build (ItemNo Fp, ItemNo Lp, int Dep, Tree Root)

Algorithm 2 Tree Build

```
1: Create a copy of the Root called BestNode
2: for all the attributes do
3:     if Special Status of the attribute is IGNORE then
4:         continue
5:     if the attribute under examination is continuous then
6:         build a Node using Build2Contin (builds a sub-tree based on a continuous attribute split) for that
               attribute
7:     elseit is discrete
8:         build Node using Build2Discr (builds a sub-tree based on a discrete attribute split)
9:     if the error of Node is less than the error of BestNode (i.e. the root) then
10:        release BestNode
11:        set BestNode := Node
12: return BestNode
```

Finding the Global Optima - T3C

- Tree Build2Discr(ItemNo Fp, ItemNo Lp, Attribute Att, int Dep, Tree Root)

Algorithm 3 Tree Build2Discr

1: BestNode is a copy of Root
2: **for** each possible value V of attribute Att **do**
3: Group data $(Fp...Lp)$ according to V and find Kp for the last of them
4: Create a branch for the value V using $BuildTree(Fp, Kp, Dep - 1, Root- > BestClass)$
5: Compute error for each branch and add it to the total error of $BestNode$
6: $Fp = Kp + 1$
7: **return** BestNode

Finding the Global Optima - T3C

Algorithm 4 Tree Build2Cont

```
1: if Dep = 1 then
2:     Call SecondSplitContin(Fp, Lp, None, Att)
3:     return IntervalTest
4: else
5:     Sort instances according to Att
6:     Find Kp, the first instance which has a known value for the attribute Att
7:     if Kp > Lp then
8:         return Root
9:     else
10:        for each attribute Att1 do
11:            if Special Status of the Att1 is IGNORE then
12:                Continue
13:            if Att1 is continuous then
14:                Call SecondSplitContin(Kp, Lp, Att, Att1)
15:            if Att1 is discrete then
16:                Call SecondSplitDiscr(Kp, Lp, Att, Att1)
17:        Find the best split BestSplit between Kp and Lp and
18:        Split node BestNode according to BestSplit
19:        Create three nodes-children for the BestNode
20:        return BestNode
```

Finding the Global Optima - T3C

- Gen is the number of incorrect instances divided by the total number of instances in the test set

Data set	T3C				C4.5				Improv
	Size	Class. (%)	Gen. (%)	F1 (%)	Size	Class. (%)	Gen. (%)	F1 (%)	Gen.
Lymphography	12.9 ± 4.63	18.98 ± 11.95	25.53 ± 7.29	48.03 ± 7.09	18.8 ± 5.29	9.23 ± 2.5	**22.6** ± 5.2	64.55 ± 15.83	−12.96
Hypotheroid	10.8 ± 0.63	0.61 ± 0.09	1.03 ± 0.26	89.01 ± 2.79	8.2 ± 1.03	0.62 ± 0.08	**0.9** ± 0.17	90.16 ± 2.12	−14.44
Hepatitis	27.6 ± 1.72	3.6 ± 0.67	**17.92** ± 5.43	47.33 ± 5.47	11.4 ± 6.17	8.53 ± 3.10	19.4 ± 5.27	39.81 ± 12.83	7.53
Crx	9.1 ± 5.70	14.18 ± 0.59	**14.24** ± 1.90	85.38 ± 2.21	31.8 ± 12.75	9.42 ± 1.79	15.1 ± 3.65	83.61 ± 3.77	5.7
Australian	10.5 ± 6.19	14.11 ± 0.81	**12.92** ± 1.99	86.85 ± 1.83	35.9 ± 17.3	9.36 ± 2.11	14.6 ± 2.47	87.12 ± 2.25	11.51
Cleve	36.1 ± 7.41	11 ± 1.04	**22.04** ± 6.29	80.42 ± 5.47	28.2 ± 6.14	8.83 ± 1.37	23.5 ± 3.72	79.16 ± 3.17	6.21
Med_123	17.4 ± 2.54	0.04 ± 0	**0.75** ± 0.58	99.44 ± 0.01	1 ± 0	33.28 ± 1.11	33.4 ± 2.15	79.9 ± 1.59	97.75
Iris	11.7 ± 0.91	1.5 ± 0.87	**4.75** ± 2.31	95.49 ± 2.31	6.2 ± 1.03	2.01 ± 1.49	6.1 ± 3.03	94.03 ± 3.05	22.13
Heart	4 ± 0	22.4 ± 1.35	29.32 ± 4.07	73.44 ± 2.95	27.4 ± 7.22	7.47 ± 2.80	**19.3** ± 3.52	80.59 ± 3.32	−51.92
Breast	10.2 ± 0.42	3.36 ± 0.49	**5.34** ± 1.98	95.8 ± 1.45	19 ± 6.18	1.8 ± 0.58	5.9 ± 0.86	95.41 ± 0.73	9.49
Diabetes	11.6 ± 0.31	21.72 ± 1.2	**26.24** ± 2.18	80.96 ± 1.29	36.2 ± 15.64	15.58 ± 2.97	27.4 ± 3.06	79.2 ± 2.18	4.23
Pima	11.1 ± 0.56	21.55 ± 1.04	**24.8** ± 2.42	81.87 ± 2.05	34.2 ± 13.03	16.2 ± 3.33	25.5 ± 1.71	80.77 ± 1.74	2.75
Waveform-21	14 ± 0	28.3 ± 0.26	31.17 ± 0.61	68.93 ± 0.61	413 ± 24.27	4.11 ± 0.88	**23.7** ± 1.27	76.38 ± 1.24	−31.52
Waveform-40	14 ± 0	27.92 ± 0.32	30.38 ± 0.91	69.7 ± 0.84	443.6 ± 18.83	2.56 ± 0.70	**24.4** ± 1.13	75.62 ± 1.14	−24.51
Avg improvement									2.29

Finding the Global Optima - T3C

- Future work

  - T3C does not perform better when the depth is larger than 3. Increasing depth improves classification accuracy

  - T3C can be improved by reducing the splitted branches for each node. Reducing splits could improve generalization accuracy

Finding the Global Optima - OCT

- To get global optimal tree, we need to form the entire decision tree in a single step, allowing each split to be determined with full knowledge of all other splits.

- Goal: to minimize the cost-complexity function $R_\alpha(T)$:

$$R_\alpha(T) = R(T) + \alpha \cdot |f(T)|$$

where:

- R(T) is the training/learning error related to the misclassification rate
- |f(T)| is related to the size of the tree T or the number of terminal leaves

- The OCT method is to change the above optimization problem to a mixed-integer optimization (MIO) problem. By solving the MIO problem, we can get the global optimal tree directly (without post-prune)

- Modern MIO solvers such as Gurobi (Gurobi Optimization Inc 2015b) and CPLEX (IBM ILOG CPLEX 2014) are able to solve linear MIO problems of considerable size

Finding the Global Optima - OCT

- To get global optimal tree, we need to form the entire decision tree in a single step, allowing each split to be determined with full knowledge of all other splits.

- Goal: to minimize the cost-complexity function $R_\alpha(T)$:

$$R_\alpha(T) = R(T) + \alpha \cdot |f(T)|$$

where:

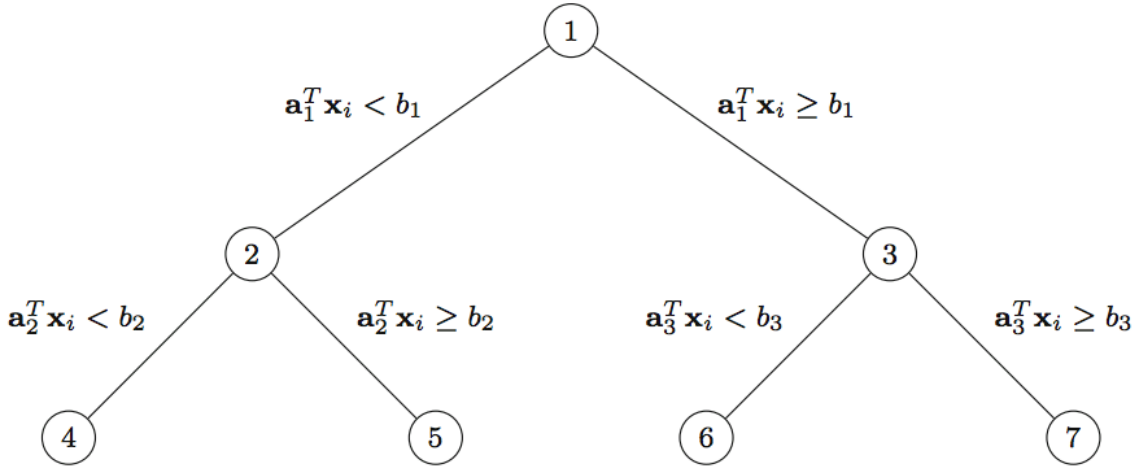
- $|f(T)|$ is the size of the tree T or the number of terminal leaves
- R(T) is the training/learning error related to the misclassification rate

- The OCT method need to be specify 3 parameters before solving the optimization problem:
  - the maximum depth D
  - the minimum leaf size N_{min}
  - the complexity parameter α

Finding the Global Optima - OCT

$$
\begin{aligned}
\min \quad & \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \sum_{t \in \mathcal{T}_B} d_t \\
\text{s.t.} \quad & L_t \geq N_t - N_{kt} - n(1 - c_{kt}), \quad k = 1, \ldots, K, \quad \forall t \in \mathcal{T}_L, \\
& L_t \leq N_t - N_{kt} + n c_{kt}, \quad k = 1, \ldots, K, \quad \forall t \in \mathcal{T}_L, \\
& L_t \geq 0, \quad \forall t \in \mathcal{T}_L, \\
& N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{it}, \quad k = 1, \ldots, K, \quad \forall t \in \mathcal{T}_L, \\
& N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in \mathcal{T}_L, \\
& \sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in \mathcal{T}_L, \\
& \boldsymbol{a}_m^\top \boldsymbol{x}_i \geq b_t - (1 - z_{it}), \quad i = 1, \ldots, n, \quad \forall t \in \mathcal{T}_B, \quad \forall m \in A_R(t), \\
& \boldsymbol{a}_m^\top (\boldsymbol{x}_i + \boldsymbol{\epsilon}) \leq b_t + (1 + \epsilon_{\max})(1 - z_{it}), \quad i = 1, \ldots, n, \ \forall t \in \mathcal{T}_B, \quad \forall m \in A_L(t), \\
& \sum_{t \in \mathcal{T}_L} z_{it} = 1, \quad i = 1, \ldots, n, \\
& z_{it} \leq l_t, \quad \forall t \in \mathcal{T}_L, \\
& \sum_{i=1}^{n} z_{it} \geq N_{\min} l_t, \quad \forall t \in \mathcal{T}_L,
\end{aligned}
$$

Finding the Global Optima - OCT



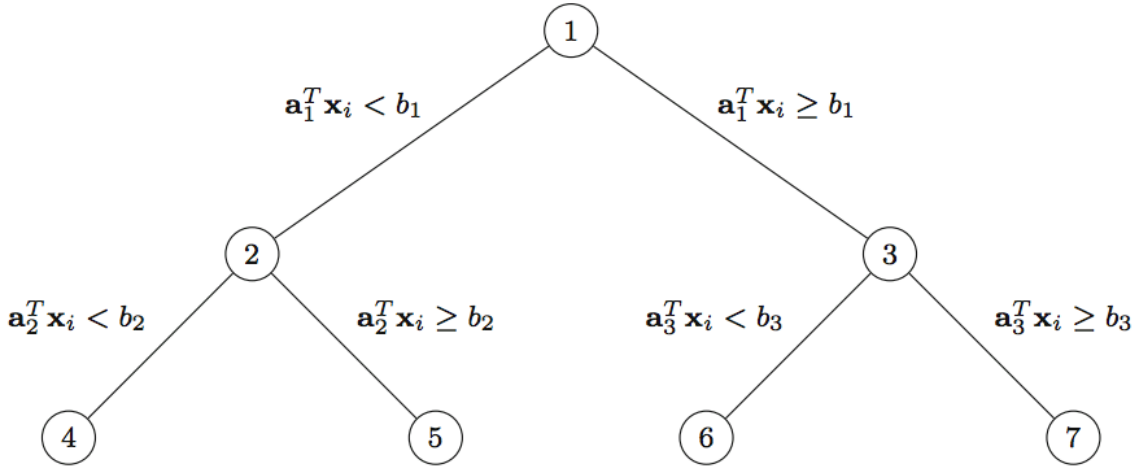
- OCT method generates the binary decision tree

- Given a depth D, we can construct the maximal tree of this depth, which has $T = 2^{(D+1)} - 1$ nodes.

- x_i is the value of i-th feature (categorical or numerical)

- OCT assume without loss of generality that the values for each dimension across the training data are normalized to the 0–1 interval, meaning $0 <= x_i <= 1$

Finding the Global Optima - OCT



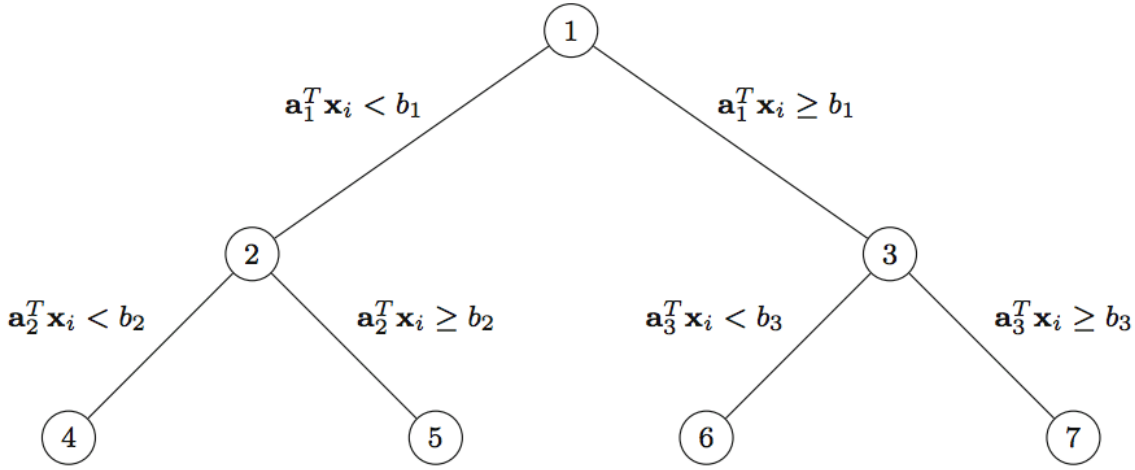
- If node 2 does not apply a split (by some judgment), set a=b=0. There will be no split from node 2 to node 4, 5 because 0 cannot smaller than 0.

- A indicator d_t will be set to 0, indicating no split at node 2 ($d_2 = 0$).

- The number of data points N_t in node 4 and 5 will be zero ($N_4 = 0$, $N_5 = 0$).

- If node 2 does not apply a split, then node 4 does not either. So d_{t1} is less than d_{t2} if t2 is the parent of t1.

Finding the Global Optima - OCT



- OCT introduces the indicator variables $z_{it} = 1$ if data point x_i is in leaf node t.

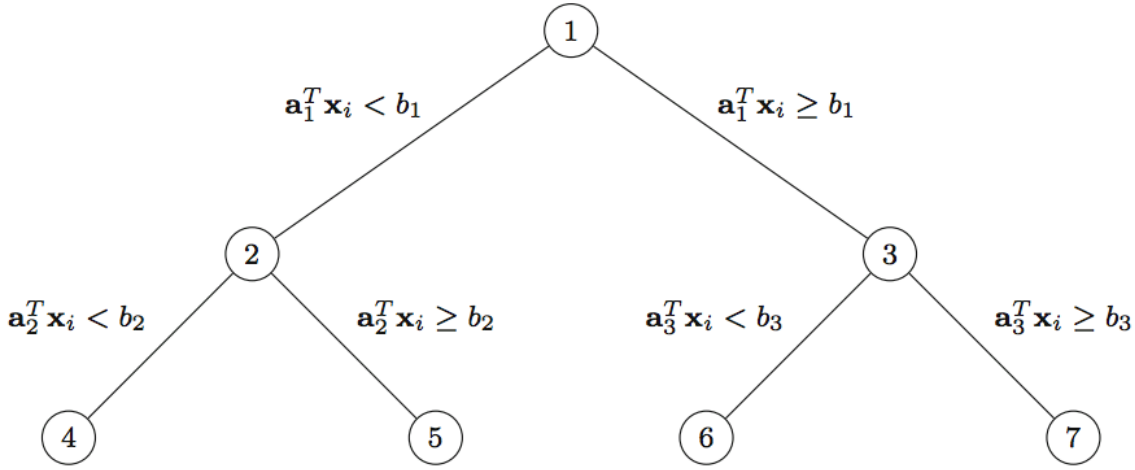
- OCT introduces the indicator variables $l_t = 1$ if leaf node t is not empty

- OCT defines a minimum number N_{min} of points at each leaf node

$$z_{it} \leq l_t,$$

$$\sum_{i=1}^{n} z_{it} \geq N_{\min} l_t,$$

Finding the Global Optima - OCT



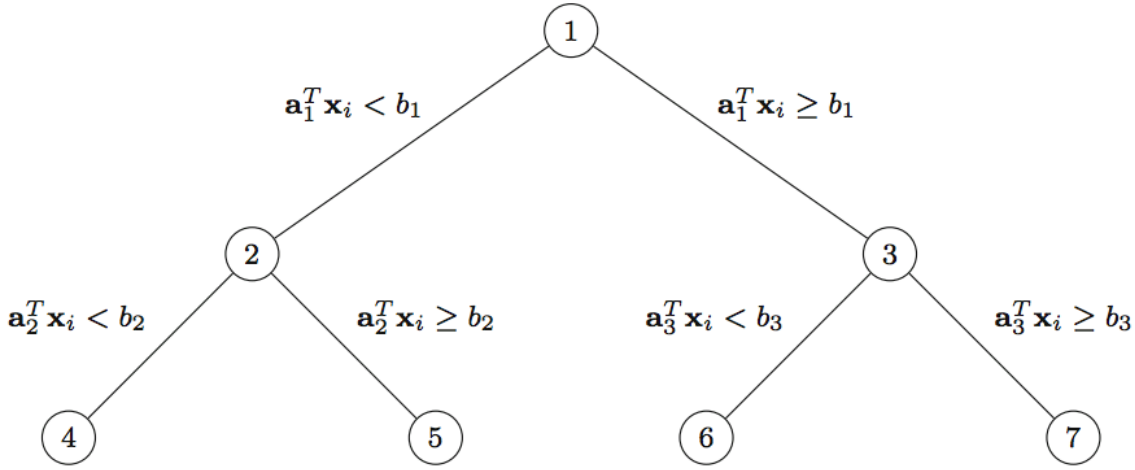
- The objective is to minimize the misclassification error. Matrix Y_{ik} is defined

$$Y_{ik} = \begin{cases} +1, & \text{if } y_i = k \\ -1, & \text{otherwise} \end{cases}, \quad k = 1, \ldots, K, \quad i = 1, \ldots, n.$$

- K is the labels of target classes

- Set N_{kt} to be the number of points of label k in node t, and Nt to be the total number of points in node t.

Finding the Global Optima - OCT



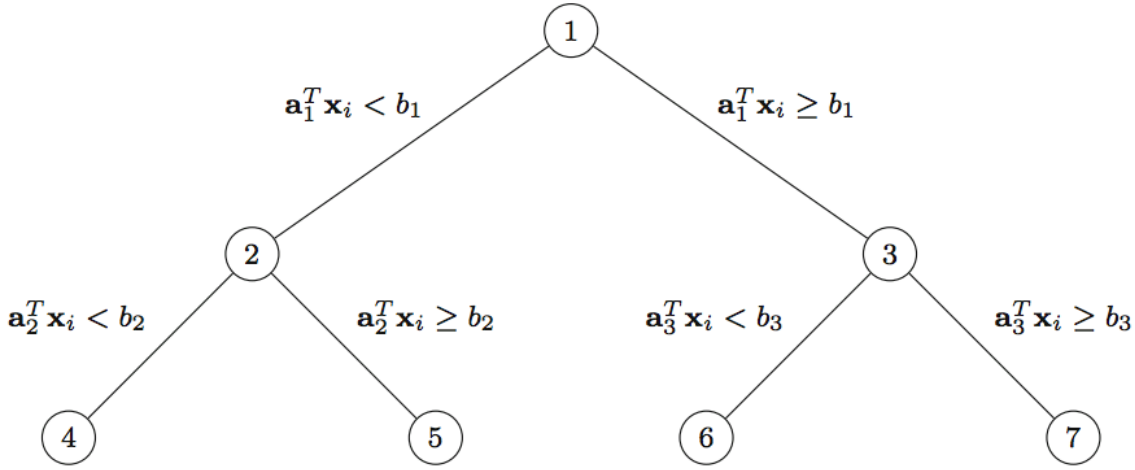
- Set N_{kt} to be the number of points of label k in node t, and Nt to be the total number of points in node t.

$$N_{kt} = \frac{1}{2}\sum_{i=1}^{n}(1 + Y_{ik})z_{it}, \quad k = 1, \ldots, K,$$

$$N_t = \sum_{i=1}^{n} z_{it},$$

- The k with largest N_{kt} will be the label of this leaf node t

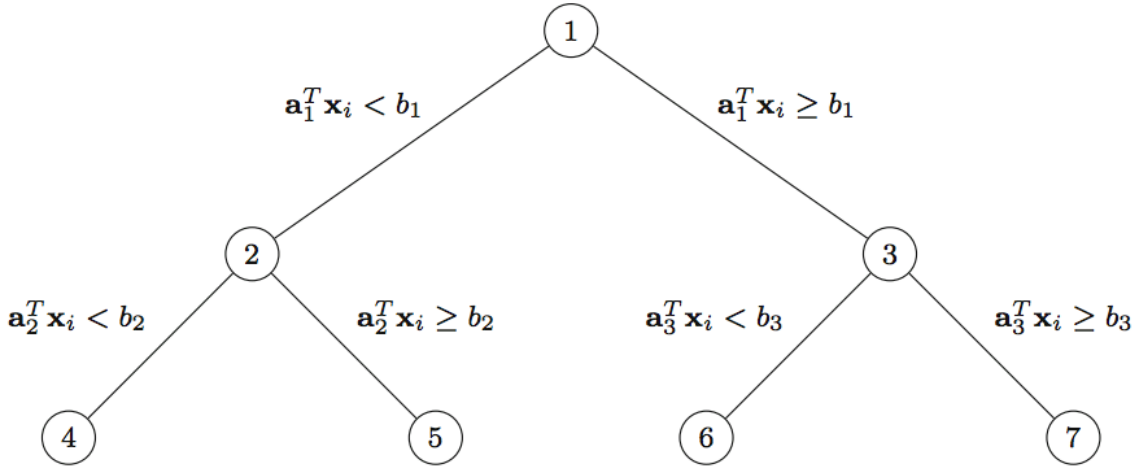
Finding the Global Optima - OCT



- The misclassification loss in each node, denoted L_t, is going to be equal to the number of points in the node minus the number of points of the most common label k

$$L_t = N_t - \max_{k=1,\ldots,K}\{N_{kt}\} = \min_{k=1,\ldots,K}\{N_t - N_{kt}\},$$

Finding the Global Optima - OCT



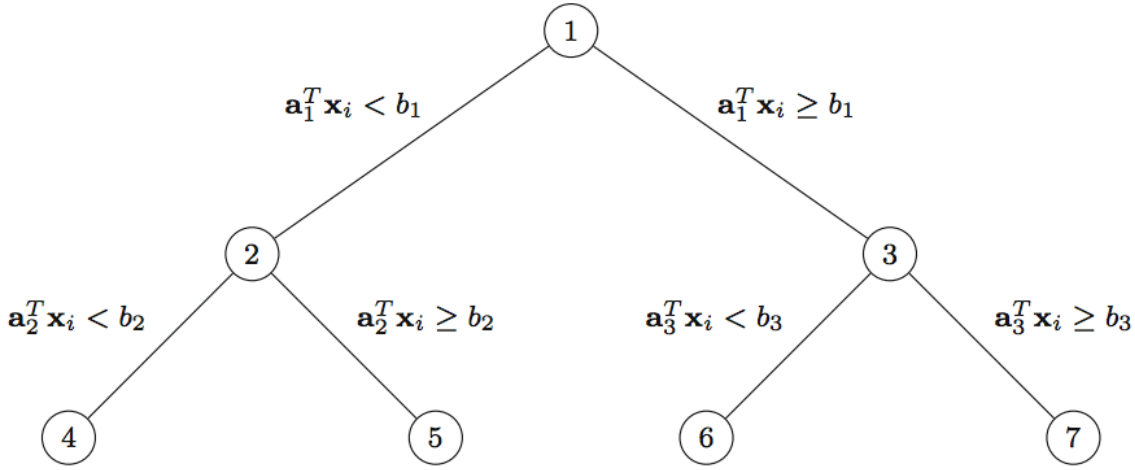
- The final objective function will be

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \sum_{t \in \mathcal{T}_B} d_t.$$

- The first term is the (normalized) total misclassification cost

- The second is the total splits when building the tree, which is related to the size of the tree

Finding the Global Optima - OCT



- It can find high-quality solutions in minutes for depths up to 4 for datasets with thousands of points. Beyond this depth or dataset size, the rate of finding solutions is slower, and more time is required

- It need to specify 3 parameters before optimization

Finding the Global Optima - OCT

- Comparison with post-prune CART

Max. depth	In high-perf. region?	CART wins	OCT wins	Ties	Accuracy improvement (%)
1	✓	0	1	13	0.01
	✗	4	18	17	0.49
2	✓	1	9	12	1.31
	✗	8	16	7	2.25
3	✓	2	9	12	1.15
	✗	12	12	6	0.41
4	✓	1	10	13	1.19
	✗	12	11	6	0.15

Finding the Global Optima – Other Approach

- linear optimization

- Dynamic programming

- Genetic algorithms

- Stochastic gradient descent

- However, none of these methods have been able to produce certifiably optimal trees in practical times

Thanks