Machine Learning

Decision Trees

Karol Przystalski

April 1, 2020

Department of Information Technologies, Jagiellonian University

Agenda

Introduction

What is a tree?

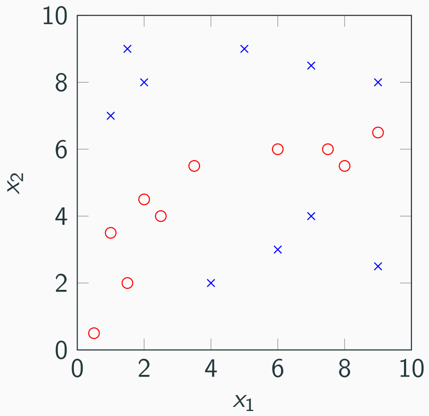
Decision trees are most likely binary trees. There are some exceptions like CHAID or C4.5 methods, but most methods stick to binary rules. Binary trees are trees where each leaf/node has maximum two childs. It is a structure where each child can has it's own childs and so on.

How does a classification works?

Two-dimensional feature space of data set given in table

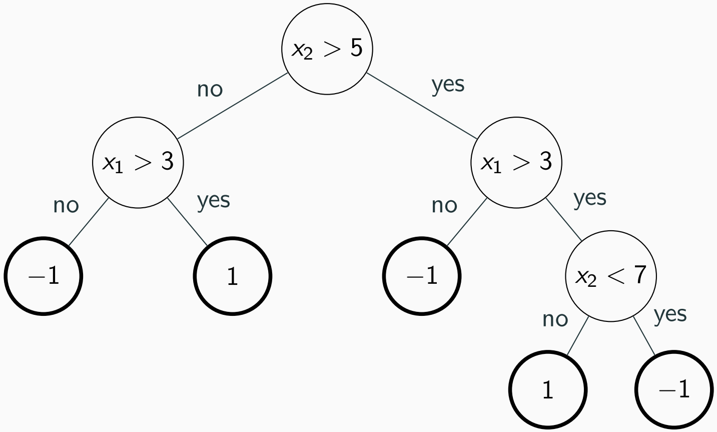
feature	x_{i1}	x_{i2}	label		x_{i1}	x_{i2}	label
x_1	0.5	0.5	-1	x_{11}	1	7	1
x_2	1.5	2	-1	x_{12}	2	8	1
x_3	2	4.5	-1	x_{13}	1.5	9	1
x_4	1	3.5	-1	x_{14}	4	2	1
x_5	2.5	4	-1	x_{15}	6	3	1
x_6	3.5	5.5	-1	x_{16}	7	4	1
x_7	6	6	-1	x_{17}	9	2.5	1
x_8	8	5.5	-1	x_{18}	5	9	1
x_9	9	6.5	-1	x_{19}	7	8.5	1
x_{10}	7.5	6	-1	x_{20}	9	8	1

Example on feature space



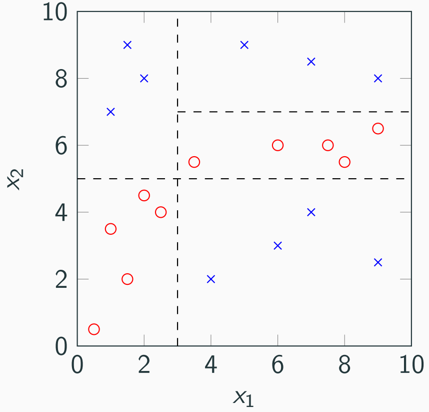
Example on feature space

Decision tree constructed for data shown in table



Decision tree

Decision tree in two-dimensional feature space



Advantages and disadvantages

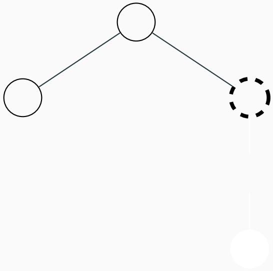
Advantages:

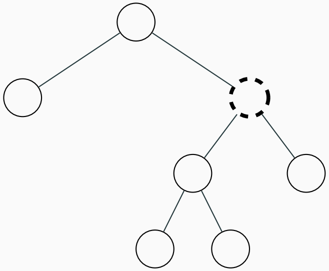
- easy to understand as it can be written as a set of rules or drawn as a tree that is readable by humans,
- not so many parameters to tune,
- perform feature selection.

Disadvantages:

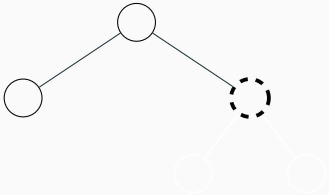
- easy to overfit,
- usually lower accuracy – does not count for random forest or any other ensemble methods based on decision trees.

Tree operations – Tree growth


(a) Before


(b) After

Tree operations – Tree division


(c) Before

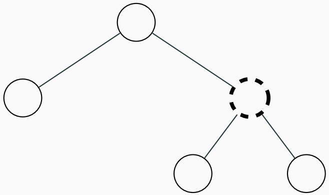
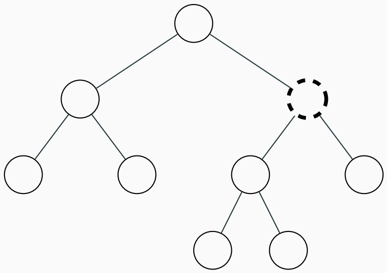
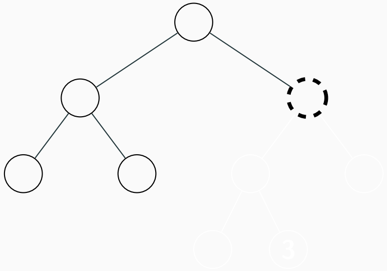

(d) After

Figure 1: Tree division

Tree operations – Tree pruning

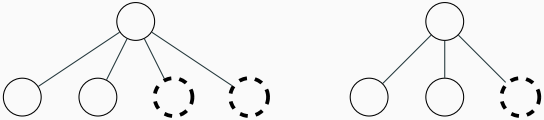


(a) Before



(b) Aft

Tree operations – Tree aggregation



(c) Before (d) After

Figure 2: Tree aggregation

Impurity measures

How to build a decision tree?

Impurity measures are used to check how homogeneous or heterogeneous a given data set is. If a data set has many classes it means that it is heterogeneous (impure). The opposite is homogeneous, what means that the data set is pure. The impurity measures are used to split a data set into two child nodes in a decision tree. There are many impurity measures:

- Gini index,
- Entropy (information gain),
- Classification error,
- Likelihood-ratio method,
- DKM impurity method,
- Orthogonal criterion.

Impurity measure – example

Let's take an example of customer segmentation:

ID	Location	Category	Gender	Product reviews	Customer decision
1	Berlin	Furniture	Male	Checked Reviews	Bought
2	London	Furniture	Male	Checked Reviews	Bought
3	Berlin	Furniture	Female	Checked Reviews	Did not bought
4	Berlin	Textile	Female	Checked Reviews	Bought
5	London	Electronics	Male	Checked Reviews	Did not bough
6	London	Textile	Female	Checked Reviews	Did not bough
7	Paris	Textile	Male	Did not checked	Bought
8	Berlin	Electronics	Male	Checked Reviews	Bought
9	Paris	Electronics	Male	Did not checked	Bought
10	London	Electronics	Female	Checked Reviews	Bought
11	Paris	Furniture	Female	Did not checked	Bought
12	Berlin	Textile	Female	Did not checked	Bought
13	London	Electronics	Female	Did not checked	Did not bough
14	London	Furniture	Female	Checked Reviews	Did not bough
15	London	Textile	Female	Did not checked	Did not bough

Impurity measure

Customer decision is the label/class. The customer can buy or not. From our Company point of view it is important to find a pattern, what are the characteristics of a frequent buyer.

We have four features: location, category, gender and product reviews status. Probabilities of each class are as following:

$$P(\text{Location}) = \frac{3}{15}$$

$$P(\text{Category}) = \frac{3}{15}$$

$$P(\text{Gender}) = \frac{2}{15}$$

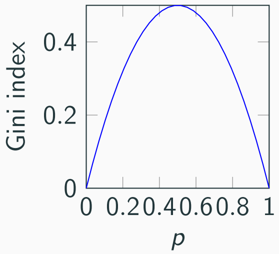
$$P(\text{Product review status}) = \frac{2}{15}$$

Gini index

Gini index is defined as:

$$I_G(X) = 1 - \sum_{i=1}^{m} p_i^2, \tag{1}$$

where m is the number of potential discrete values or ranges of continuous values, p is the probability of a specific feature value in data set. The values of Gini index are like shown in below figure.



Gini index

For the customer segmentation example the Gini index would be calculated as:

$$I_G(\text{Berlin}) = 1 - (\frac{4}{5})^2 + (\frac{1}{5})^2 = 1 - (0.64 + 0.04) = 0.32.$$

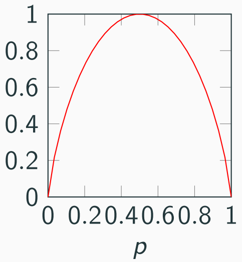
We have 5 cases when the customer is from Berlin. Four times the customer bought the product and one time did not.

Information gain

Entropy (information gain) can be calculated as:

$$E(X) = -\sum_{i=1}^{m} p_i \log_2 p_i. \tag{2}$$

The values of entropy can be as shown in figure below:



Information gain

For the customer segmentation example the entropy would be calculated as:

$$\begin{aligned} E(\text{London}) &= -1((\frac{2}{7})^2 * \log_2(\frac{2}{7}) + (\frac{5}{7})^2 * \log_2(\frac{5}{7})) \\ &= -1(0.2857 * (-1.807) + 0.7142 * (-0.4855)) \\ &= -1(-0.516 - 0.34671) = 0.8617 \end{aligned}$$

Univariate decision trees

Univariate decision tree methods

There are many univariate methods. Most popular are:

- ID3,
- CART,
- QUEST,
- CHAID,
- CRUISE,
- C4.5
- CAL5,
- FACT,
- and many more.

Classification and Regression Trees

CART method is one of the oldest decision tree method. It generates a binary tree with nodes:

1. t – tree node,
2. t_l – left child node of t,
3. t_r – right child node of t.

The child nodes can be defined as:

$$P_L = \frac{\#t_L}{\#X_{\text{train}}}, \tag{3}$$

and

$$P_R = \frac{\#t_R}{\#X_{\text{train}}}. \tag{4}$$

CART – child nodes

The child nodes can be one of a class:

$$P(i|t_L) = \frac{\#t_L \text{ of class } i}{\#t}, \tag{5}$$

and

$$P(i|t_R) = \frac{\#t_R \text{ of class } i}{\#t}. \tag{6}$$

In the perfect scenario we have elements of one class in each child node.

The decision on which node needs to be split is taken

$$\Phi(s|t) = 2P_L P_R \sum_{i=0}^{m} |P(i|t_L) - P(i|t_R)| \tag{7}$$

We get the same results using the Gini index to distinguish between features to split.

CART – example

Let's take again the example of customer segmentation:

ID	Location	Category	Gender	Product reviews	Customer decision
1	Berlin	Furniture	Male	Checked Reviews	Bought
2	London	Furniture	Male	Checked Reviews	Bought
3	Berlin	Furniture	Female	Checked Reviews	Did not bought
4	Berlin	Textile	Female	Checked Reviews	Bought
5	London	Electronics	Male	Checked Reviews	Did not bough
6	London	Textile	Female	Checked Reviews	Did not bough
7	Paris	Textile	Male	Did not checked	Bought
8	Berlin	Electronics	Male	Checked Reviews	Bought
9	Paris	Electronics	Male	Did not checked	Bought
10	London	Electronics	Female	Checked Reviews	Bought
11	Paris	Furniture	Female	Did not checked	Bought
12	Berlin	Textile	Female	Did not checked	Bought
13	London	Electronics	Female	Did not checked	Did not bough
14	London	Furniture	Female	Checked Reviews	Did not bough
15	London	Textile	Female	Did not checked	Did not bough

CART – algorithm

CART method consist of the following steps:

1. calculate the gini index for each feature,
2. take the lowest value of ω and split the node into two child nodes,
3. repeat the steps until we have all child nodes

CART – first step

CART generates binary trees, so we need to calculate the gini index and ω for each feature. Let's take the location as an example and calculate the Gini index for this feature:

$$I_G(\text{London}) = 1 - (\frac{2}{7})^2 + (\frac{5}{7})^2 = 1 - (0.081 + 0.51) = 0.4087$$

$$I_G(\text{Paris and Berlin}) = 1 - (\frac{7}{8})^2 + (\frac{1}{8})^2 = 1 - (0.76 + 0.015) = 0.22$$

Now, we can calculate the Gini index of location feature:

$$I_G(\text{feature}) = 1 - \sum_{i=1}^{n} p_i * I_G(X_i), \tag{8}$$

what for location feature makes it:

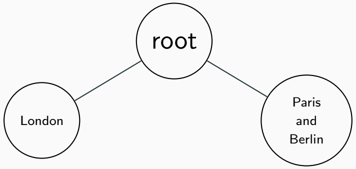
$$I_G(\text{Location}) = \frac{7}{15} * 0.4087 + \frac{8}{15} * 0.22 = 0.1907 + 0.1173 = 0.3080$$

CART – Gini index results

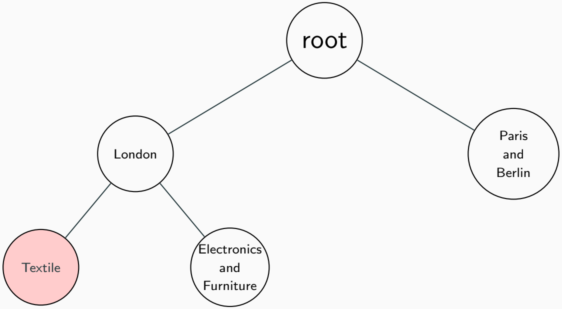
The results for all combinations are like following:

Nodes	**Gini index**
London	0.4087
Paris and Berlin	0.22
Gini index	0.3080
Berlin	0.32
London and Berlin	0.5
Gini index	0.4365
Paris	0
London and Berlin	0.5
Gini index	0.4

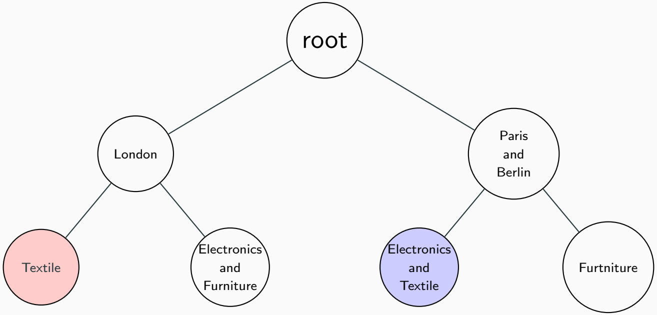
CART – tree after step one



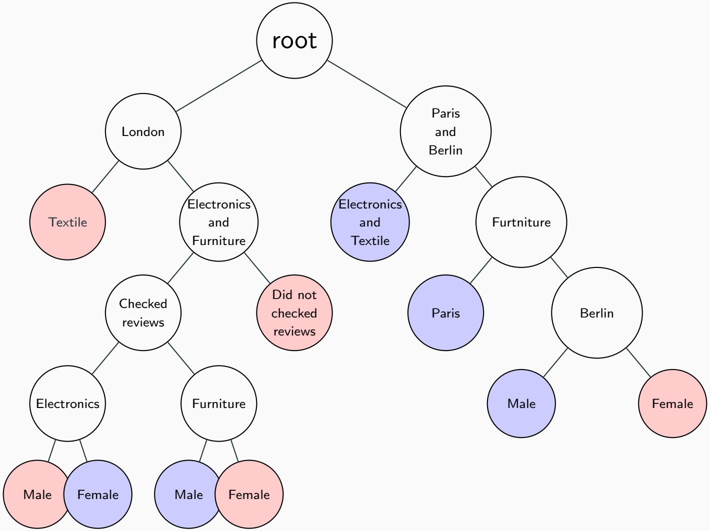
CART – tree after step two



CART – – tree after step three



CART – final tree



C4.5 – concept

This method use entropy (information gain) as a measure for node split. It does not create a binary tree, so each node level can have more than two childs.

The algorithm steps are similar as in case of CART.

C4.5 – Calculate entropies

We calculate the entropies of each feature value in case of discrete values. For the customer segmentation example we have:

$$\begin{aligned} E(\text{London}) &= -1((\frac{2}{7})\log_2(\frac{2}{7}) + (\frac{5}{7})2\log_2(\frac{5}{7})) \\ &= -1(0.2857 * (-1.807) + 0.7142 * (-0.4895)) = 0.8617 \end{aligned}$$

$$\begin{aligned} E(\text{Berlin}) &= -1((\frac{4}{5})\log_2(\frac{4}{5}) + (\frac{1}{5})\log_2(\frac{1}{5})) \\ &= -1(0.8 * (-0.3219) + 0.2 * (-2.319)) = 0.7218 \end{aligned}$$

$$E(\text{Paris}) = -1(\frac{3}{3}\log_2(1)) = 0$$

C4.5 – customer decision entropy

The next step is to calculate the customer decision entropy. It is called also as target or total entropy:

$$\begin{aligned} E(\text{Customer}) &= -1(\frac{9}{15}\log_2(\frac{9}{15}) + \frac{6}{15}\log_2(\frac{6}{15})) \\ &= -1(0.6 * (-0.7365) + 0.4 * (-1.3219)) = 0.9708. \end{aligned}$$

Next, we need to calculate the entropy for the location feature:

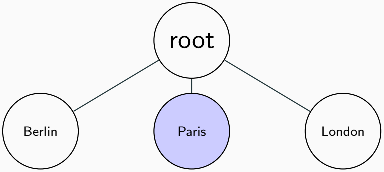
$$E(\text{Location}) = \frac{5}{15} * 0.7218 + \frac{7}{15} * 0.8617 + \frac{3}{15} * 0 = 0.2406 + 0.4021 = 0.6427$$

Finally we can calculate the information gain for location feature:

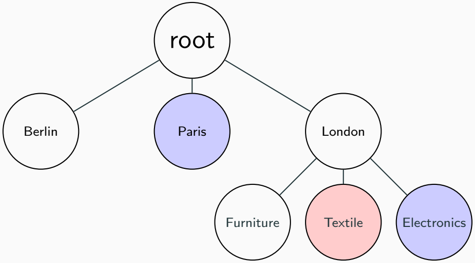
$$IG(\text{Location}) = E(\text{Customer}) - E(\text{Location}) = 0.9708 - 0.6427 = 0.3281.$$

We calculate the information gain for each feature. The customer decision entropy $E(\text{Customer})$ stays the same for each feature. The node is split by feature of the highest information gain.

C4.5 – first step

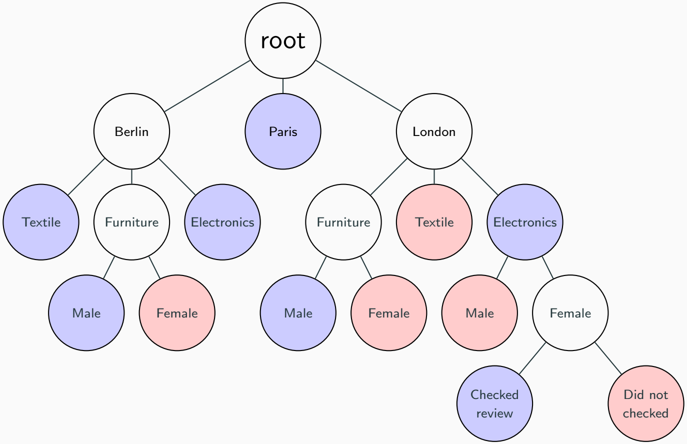


C4.5 – second step



C4.5 – same example, different results

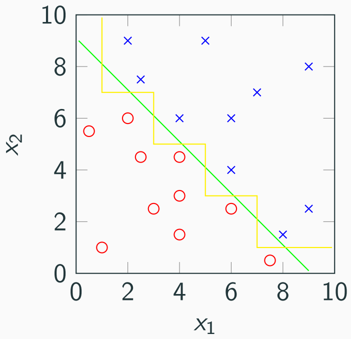
Final decision tree for customer segmentation using C4.5 method.



Multivariate

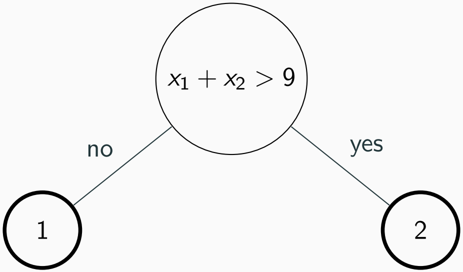
Multivariate

Linear separable example classified with univariate (marked yellow) and multivariate decision trees (marked green).



Multivariate

Multivariate decision trees take more than one feature into consideration. It means that we split using more than one feature. It can be written as a linear function as shown in the figure:



Multivariant decision trees – advantages

There are a few advantages of multivariate decision trees:

- we split on more than one feature what usually give a tree with a smaller high value what means a faster prediction,
- solve non-linear classification problems.

The disadvantage of a multivariant method are the parameters that needs to be set. In most cases there are more to set in multivariant decision trees.

Almost each univariat decision tree method can be easily changed to a multivariant methods.

Multivariant decision trees methods

There are plenty of multivariant decision trees methods. A few most popular are:

- OC1,
- LMDT,
- CART-LC,
- MARS,
- and many more.

Linear Machine Decision Trees – general algorithm

The LMDT method consist of the following steps:

1. If all the instances are from a single class, then set tree to be a leaf node with the class name of the single class, return.
2. Otherwise, set tree to be a decision node containing a test constructed by training a linear machine.
3. If the test partitions the instances into two or more subsets, then for each subset build a subtree recursively, return.
4. Otherwise, set tree to be a leaf node with the class name of the most frequently occurring class, return

LMDT – linear machine

A linear machine is a set of R linear discriminant functions that are used collectively to assign an instance to one of the R classes. Let Y be an instance description, also known as a pattern vector, consisting of a constant threshold value 1 and the numerically encoded features by which the instance is described. Then each discriminant function $g_i(Y)$ has the form $W_i^T Y$, where W is a vector of adjustable coefficients, also known as weights. A linear machine infers instance Y to belong to class i if and only if

$$(\forall_i, i \neq j) g_i(Y) > g_j(Y). \tag{9}$$

In general, to train a linear machine, one adjusts the weight vectors W of the discriminant functions g in response to any instance that the linear machine would misclassify.

Quality metrics

Tree quality

There is one known and obvious metric that we use to measure the quality of tree. It's the tree high. A smaller tree is a better tree if we have the same error rate (accuracy).

Quality metric types

We have two types of quality metrics:

- pre-pruning (direct) methods,
- post-pruning (validation) methods.

Direct methods are done during the tree construction. It is more cost efficient than validation methods as we can change/prune the tree immediately. Examples of direct methods are:

- minimum number of object pruning,
- Chi-square pruning.

Minimum number of objects pruning

In this method of pruning, the minimum number of object is specified as a threshold value.
Whenever the split is made which yields a child leaf that represents less than the minimum number from the data set, the parent node and children node are compressed to a single node.

Chi-square pruning

This approach to pruning is to apply a statistical test to the data to determine whether a split on some feature X_k is statistically significant, in terms of the effect of the split on the distribution of classes in the partition on data induced by the split. We can perform chi-squared test as:

$$\chi^2 = \sum \frac{(\text{observed value} - \text{expected value})^2}{\text{expected value}}. \tag{10}$$

We reject the split if feature X_k is unrelated to the classification of data given features

Quality metrics – validation methods

Validation methods usually go through the whole tree and calculate some metrics at each node or level to prune the tree. Examples of validation methods are:

- reduced error pruning,
- error complexity pruning,
- minimum error pruning,
- cost-based pruning,
- and many more.

Reduced error pruning

It is simplest and most understandable method in decision tree pruning. This method considers each of the decision nodes in the tree to be candidates for pruning, consist of removing the subtree rooted at that node, making it a leaf node. The available data is divided into three parts: the training examples, the validation examples used for pruning the tree, and a set of test examples used to provide an unbiased estimate of accuracy over future unseen examples. If the error rate of the new tree would be equal to or smaller than that of the original tree and that subtree contains no subtree with the same property, then subtree is replaced by leaf node, means pruning is done.

Error complexity pruning

In error complexity pruning is concern with calculating error cost of a node. It finds error complexity at each node. The error cost of the node is calculated using following equation:

$$R(t) = r(t) \times p(t), \tag{11}$$

where $r(t)$ is error rate of a node which is given as:

$$r(t) = \frac{\text{number of misclassified}}{\text{numer of all objects in node}}, \tag{12}$$

and $p(t)$ is probability of occurrence of a node which is given as:

$$p(t) = \frac{\text{number of objects in node}}{\text{number of all objects}}. \tag{13}$$

Error complexity pruning

Additionally, we need to calculate the error cost of subtree T of a given node:

$$R(T) = \sum R(i), \tag{14}$$

where i is the number of leaves of the node t. The error complexity is then calcualted as follows:

$$a(t) = \frac{R(t) - R(T)_t}{\text{number of leaves} - 1} \tag{15}$$

The method consists of following steps:

1. a is computed for each node,
2. the minimum a node is pruned,
3. the above is repeated and a forest of pruned tree is formed,
4. the tree with best accuracy is selected.

Questions?