DDA3020 Machine Learning
Lecture 08 Decision Tree & Bagging & Random Forest

Baoyuan Wu
School of Data Science, CUHK-SZ

March 13/18, 2023

Outlines

Motivation

Parametric models:

- Until now, we have learned 3 supervised learning models, including linear regression (for regression and classification), logistic regression, and SVM.
- The commonality is that they are all parametric models:
  - In training, we define a **model** (*i.e.*, hypothesis function) over **the whole input space**, and learn its **parameters with fixed numbers** from all of the training data.
  - In testing, we use the same model and the same parameter set for any test input.
- The limitations of parametric models include:
  - The adopted model should be determined by the trainer, and it may be far from the ground-truth relationship between the input and the output.
  - The decision/prediction is difficult to explain/understand, there is no decision procedure

Motivation

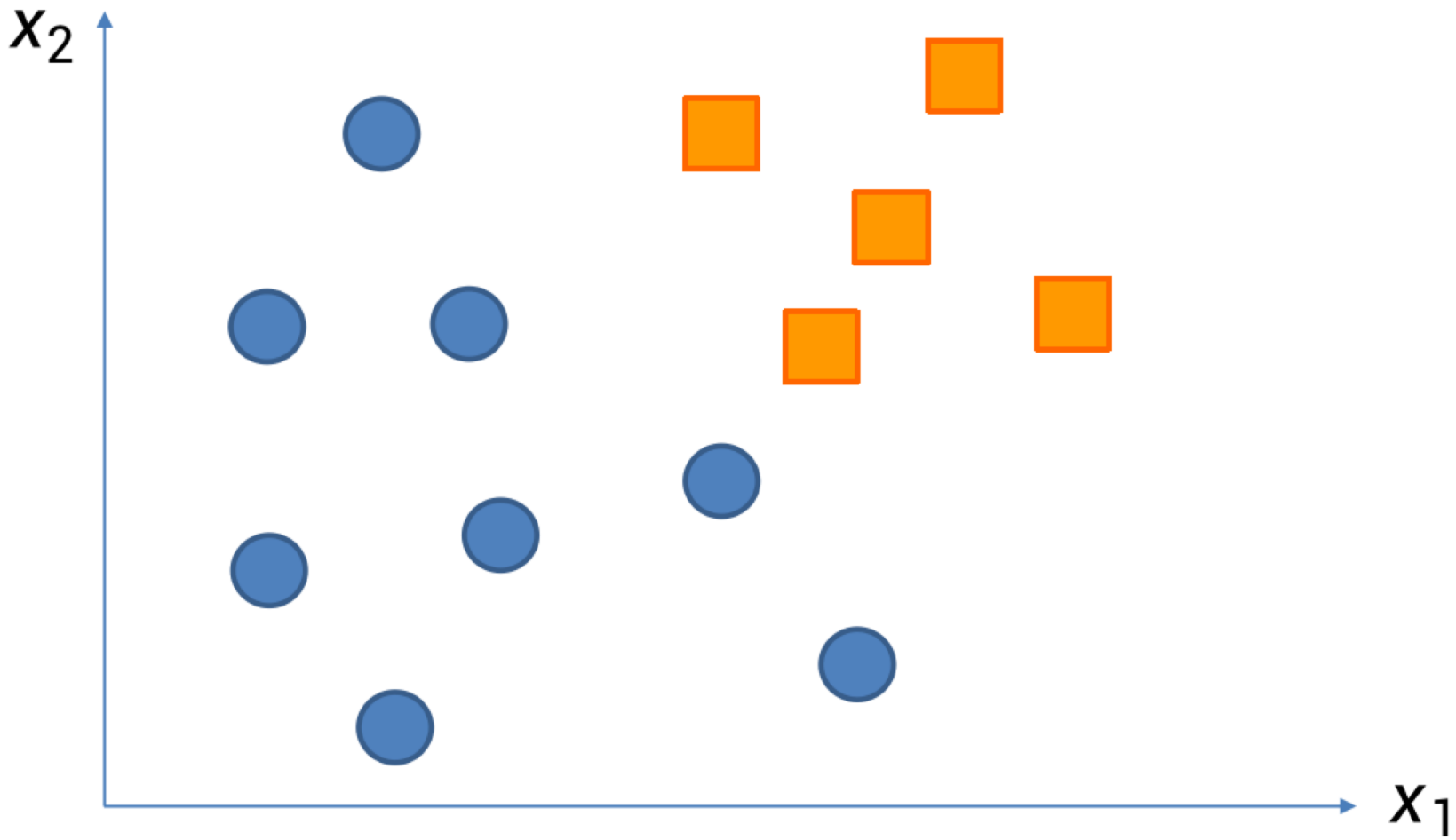
Thus, we introduce **nonparametric models**

- We divide the input space into local regions, defined by a distance measure like the Euclidean norm, and for each input, the corresponding local model computed from the training data in that region is used. The typical nonparametric models include k-nearest neighbors (KNN) classification model, and decision tree.
- It does not rely on strong assumptions regarding the shape of the relationship between the variables. Instead, the data are allowed to speak for themselves in determining the form of the fitted functions.
- Decision tree (DT) is a hierarchical nonparametric model. In addition to the above advantage, a special advantage of DT is the interpretability of its decision, which is a hierarchical decision process.

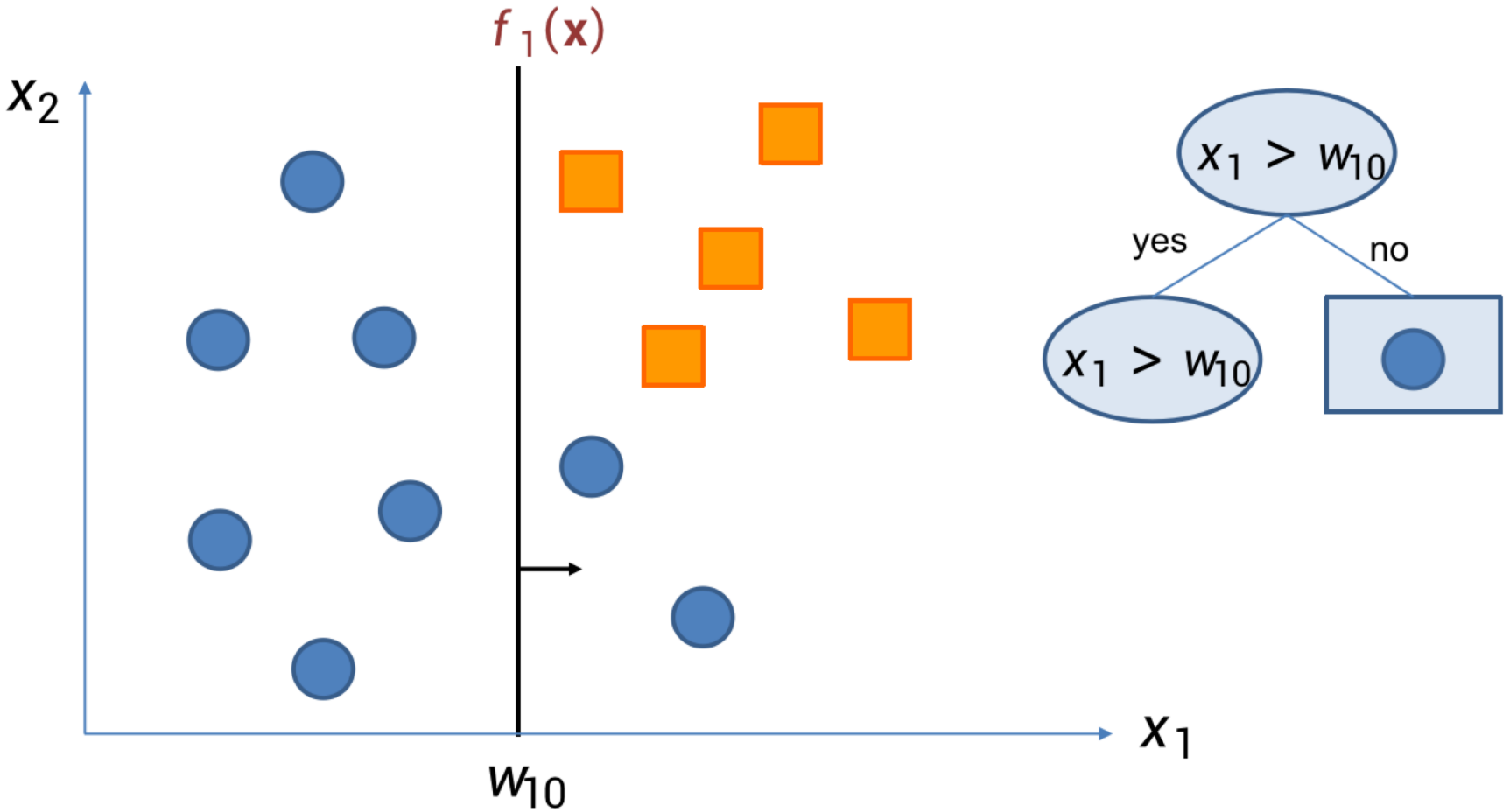
Definition

- A **decision tree** is a **hierarchical model** for supervised learning whereby the local region is identified in a sequence of **recursive splits**.
- In a **univariate tree**, in each internal node, the test uses only one of the input dimensions.

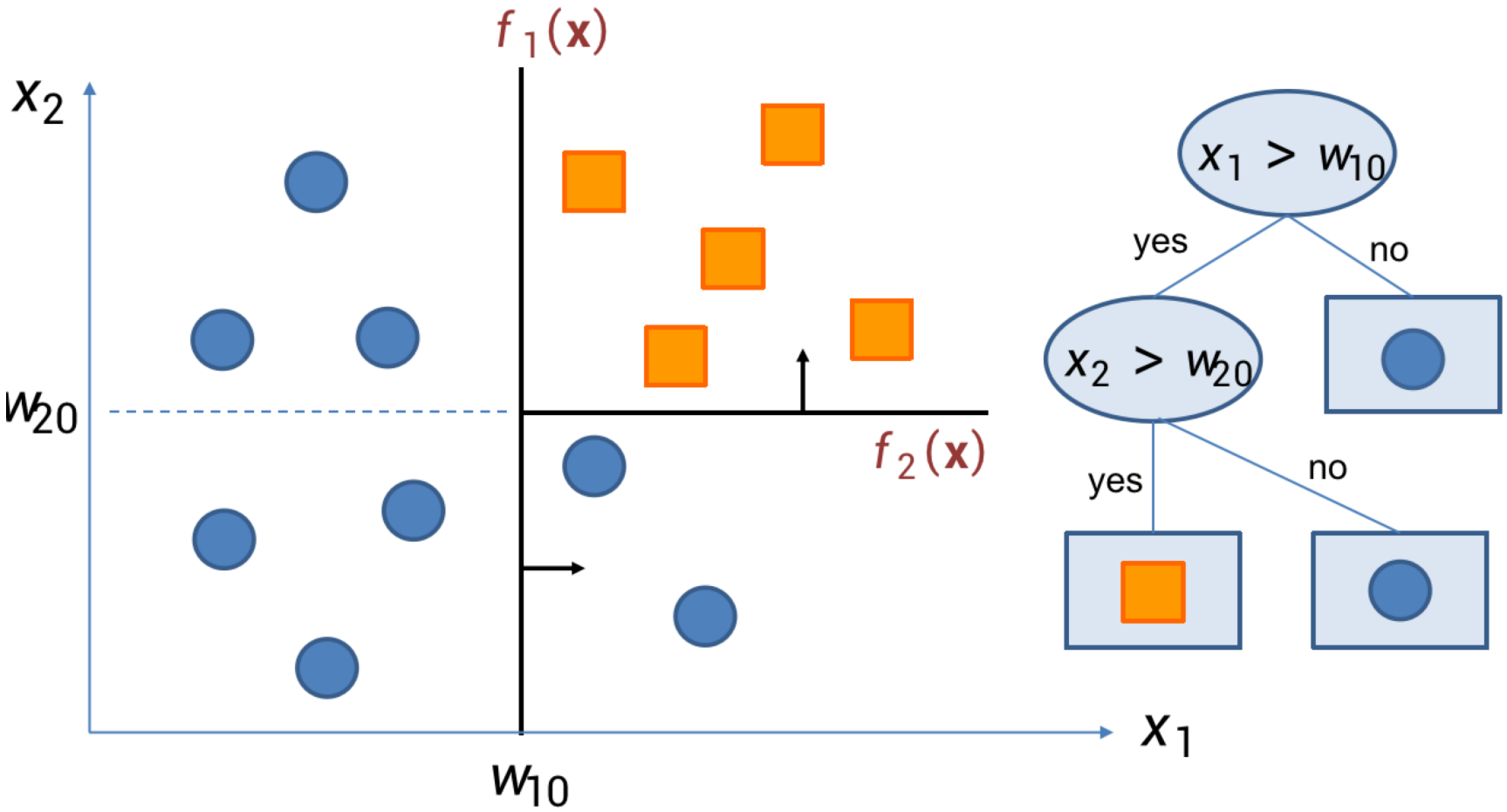
Example 1: A Classification Problem



Example 1: A Classification Problem

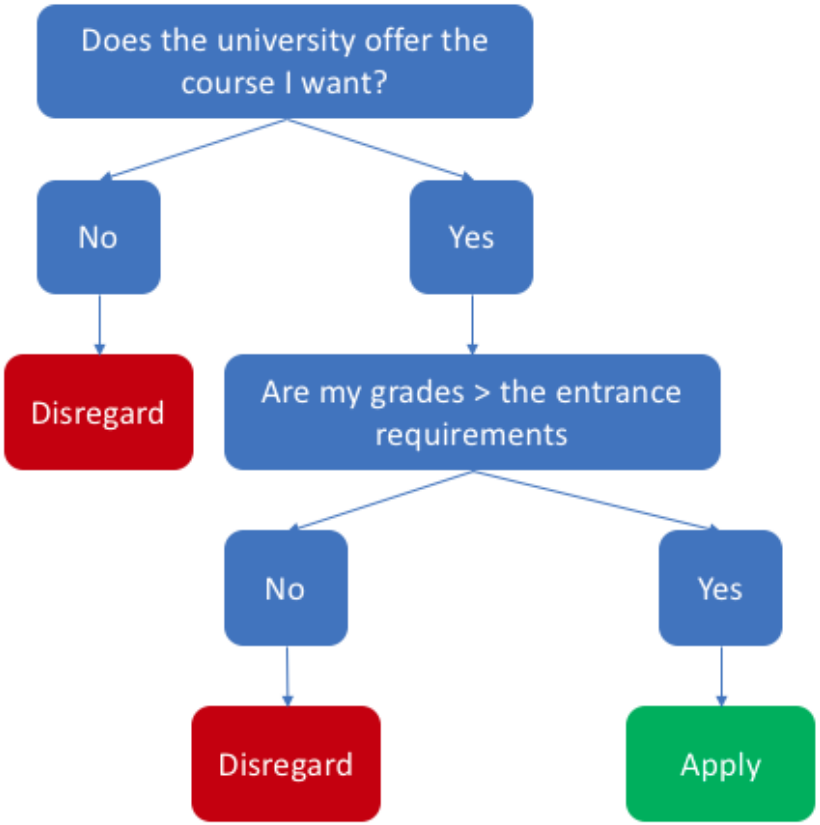


Example 1: A Classification Problem



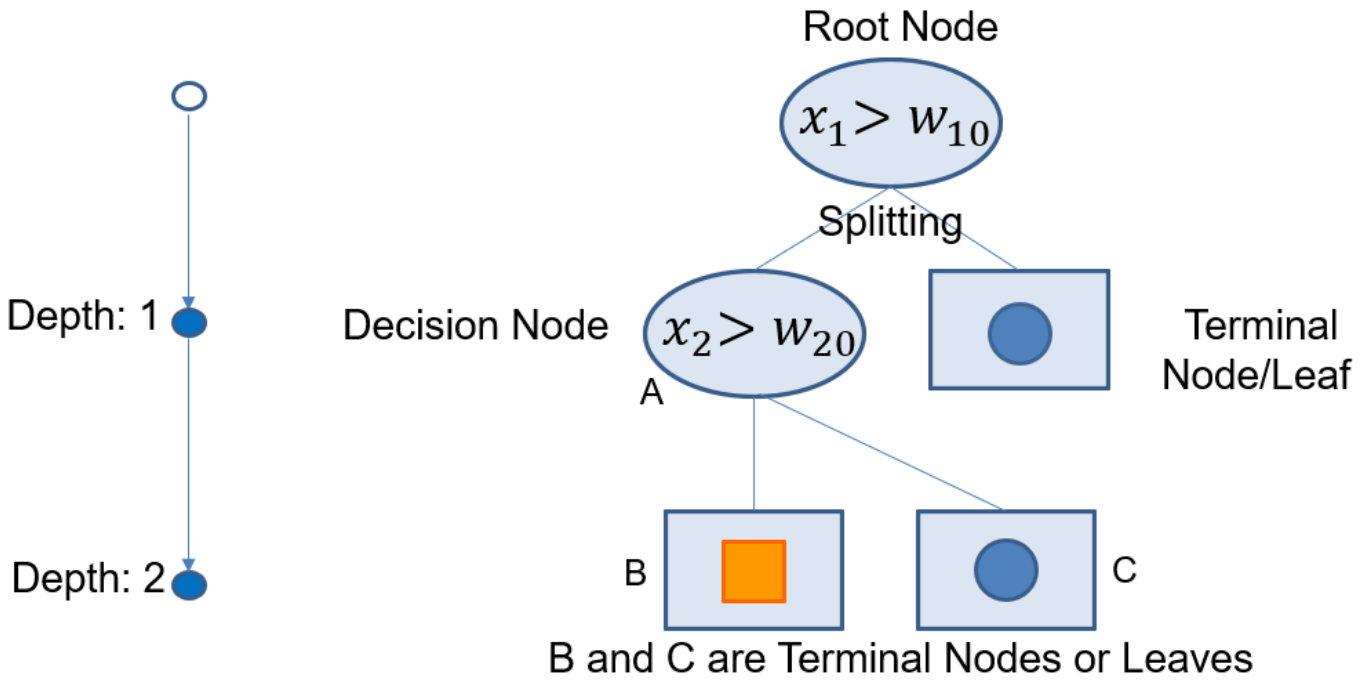
Example 2: A Practical Classification Problem

A simple example is the decision process to choose a university. Given the student has a course in mind, a decision making process could be:



Decision trees can be easily read and even mimic a human approach to decision making by breaking a big decision into many small ones.

Basic Terminologies



Note:
A-B-C forms a **sub-tree** or **branch.**
A is **parent node** of B and C; B and C are the **child nodes** of A.

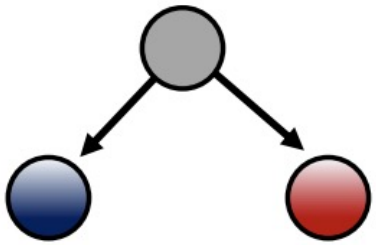
Tree Learning

Tree induction (also known as learning or growing) is the construction of the tree given a training set.

- Goal: For a given training set, there exist many trees that code it with no error, and, for simplicity, we are interested in finding the smallest among them, where tree size is measured as the number of nodes in the tree and the complexity of the decision nodes.
- Difficulty: Finding the smallest tree is NP-complete (Quinlan 1986). Thus, we are forced to use local search procedures based on heuristics that give reasonable trees in reasonable time.

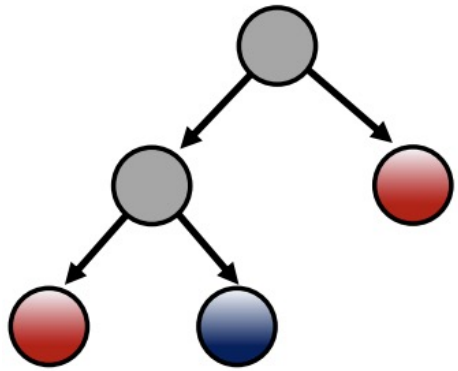
How to build a decision tree

- Select an attribute and split the data into its children in a tree



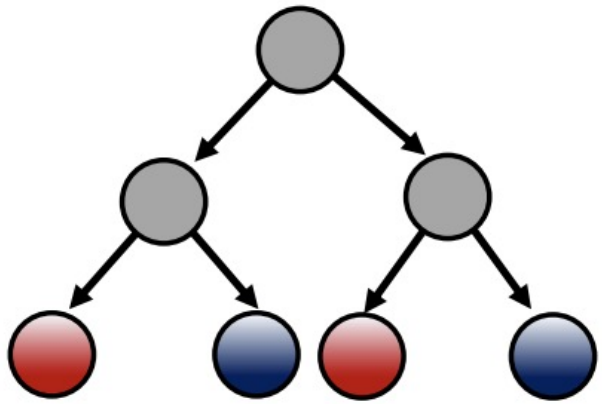
How to build a decision tree

- Select an attribute and split the data into its children in a tree
- Continue splitting with available attributes



How to build a decision tree

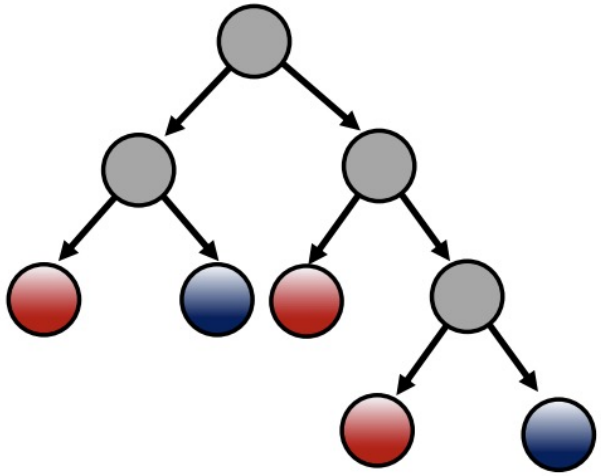
- Select an attribute and split the data into its children in a tree
- Continue splitting with available attributes



How to build a decision tree

Until

- Leaf node(s) are pure (only one class remains)
- A maximum depth is reached
- A performance metric is achieved



The tree construction is conducted in a recursive way.
The remaining issues are:

- Which is the "best attribute"?
- What defines the best split?

How to choose the best attribute for a classification tree

As only one input attribute (variable) is used at each step, which attribute and what split are the best for each step? There are some rules:

- Random: an attribute chosen at random
- **Least-Values**: the attribute with the smallest number of possible values
- **Most-Values**: the attribute with the largest number of possible values
- **Impurity Measure**: the attribute that has the largest reduction of impurity

Impurity

- Let us say for node m, N_m is the number of training instances reaching node m. For the root node, it is N.
- N_m^i of N_m belong to class $C_i, i = 1, \ldots, K$ with $\sum_i N_m^i = N_m$
- Given that an instance reaches node m, the estimate for the probability of class C_i is

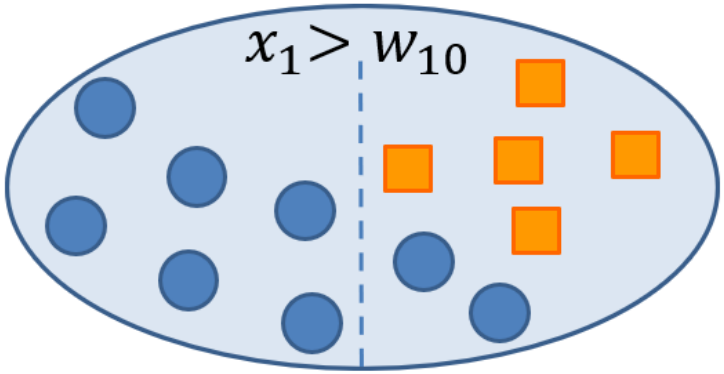
$$\widehat{P}\left(C_i \mid x, m\right) \equiv p_m^i = \frac{N_m^i}{N_m}$$

Impurity

- Node m is pure if p_m^i for all i are either 0 or 1.
- It is 0 when none of the instances reaching node m are of class C_i, and it is 1 if all such instances are of $C_i, i = 1, \ldots, K$.
- If the node is pure, we do not need to split any further and can add a leaf node labeled with the class for which p_m^i is 1 .
- How to measure the impurity of one node?
  - Classification error
  - Entropy

Impurity Measure 1: Classification Error

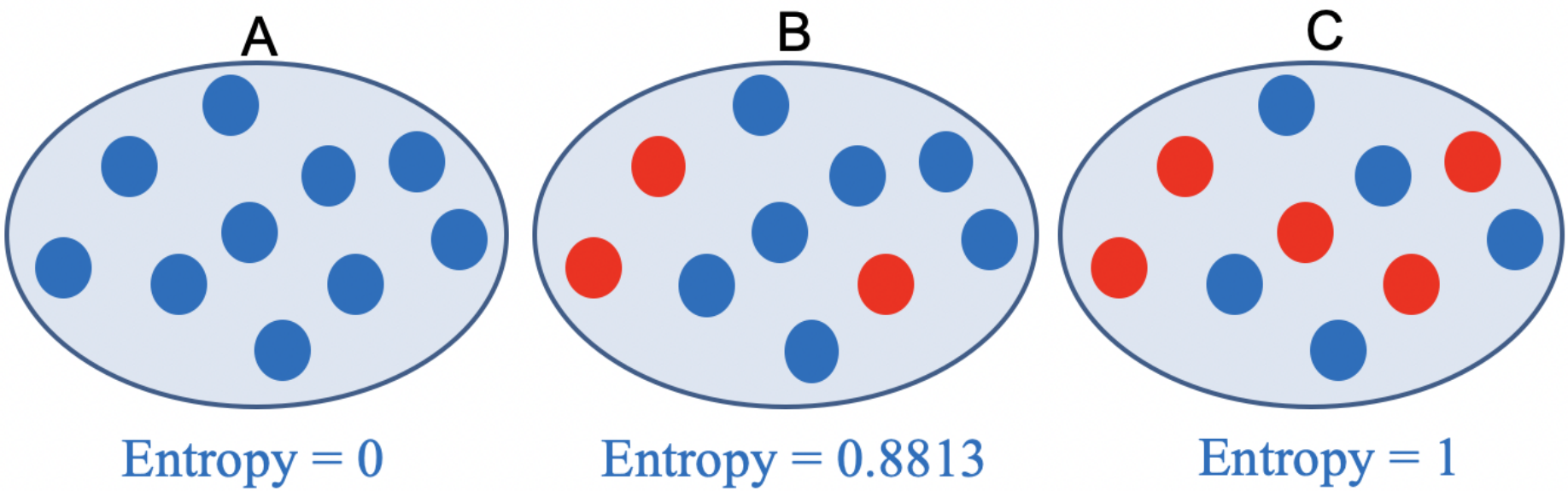
For classification tree, the goodness of split/impurity can be quantified by the classification error.



Count the errors, or using probability: $\phi(p, 1-p) = 1 - \max(p, 1-p)$

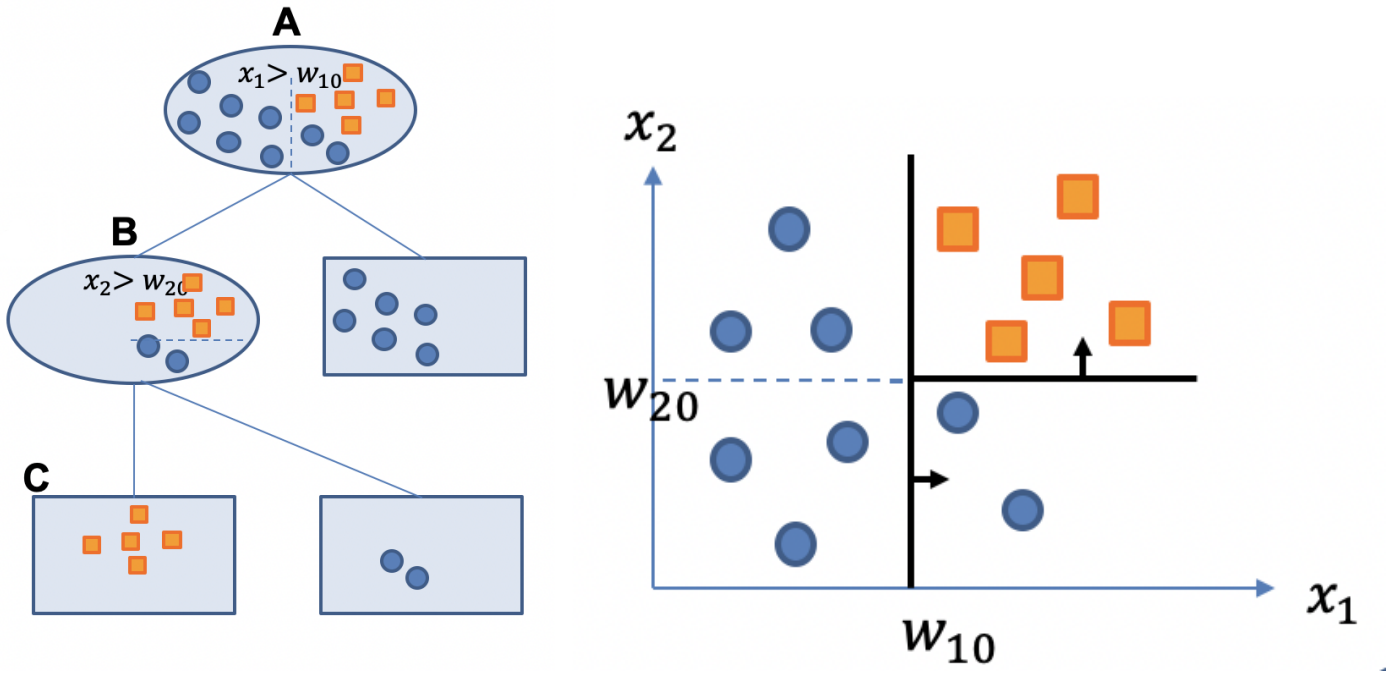
Impurity Measure 2: Entropy

- Entropy in information theory specifies the minimum number of bits needed to encode the class code of an instance.
- Entropy for multi-class node $I_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$.
- Entropy for binary (two-class) node $= -p \log_2 p - (1-p) \log_2 (1-p)$.



A is a pure node, B is more impure than A, and C is the most impure here.

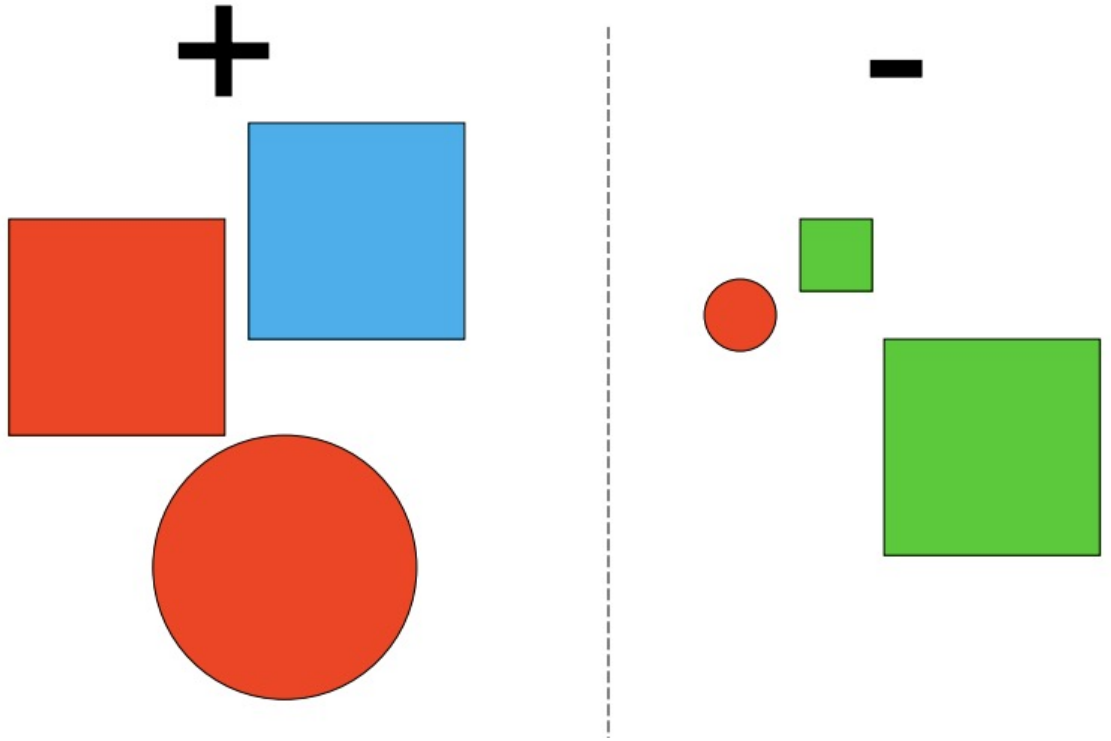
Impurity Measure 2: Entropy



Entropy for parent node A : $-\left(\frac{5}{13}\right)\log_2\left(\frac{5}{13}\right)-\left(\frac{8}{13}\right)\log_2\left(\frac{8}{13}\right)=0.96$
Entropy for node B := $-\left(\frac{5}{7}\right)\log_2\left(\frac{5}{7}\right)-\left(\frac{2}{7}\right)\log_2\left(\frac{2}{7}\right)=0.86$
Entropy for node C : $-\left(\frac{5}{5}\right)\log_2\left(\frac{5}{5}\right)-\left(\frac{0}{5}\right)\log_2\left(\frac{0}{5}\right)\triangleq 0$

Example 3: choosing attribute via information gain

Consider a binary classification with the following training data, left is positive and right is negative. Each data is described by 3 attributes: Color, Size, Shape
What's the best attribute for the root node?



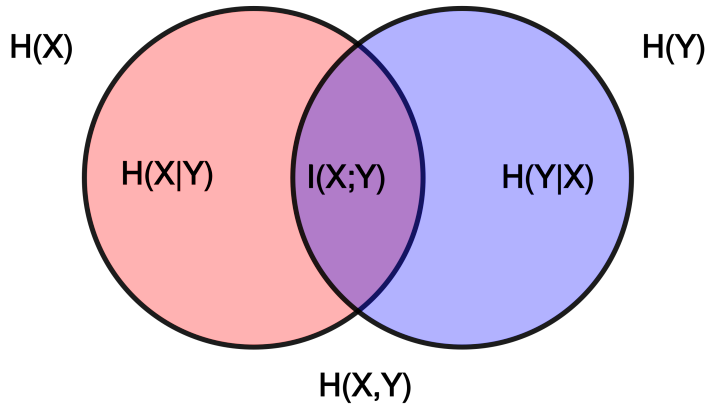
Reference: `https://pages.cs.wisc.edu/~dyer/cs540/notes/11_learning-decision-pdf`

Example 3: choosing attribute via information gain

Preliminaries:

- Entropy: $H(x) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$
- Conditional Entropy: $H(x|y) = -\sum_{(x,y) \in (\mathcal{X}, \mathcal{Y})} p(x,y) \log p(x|y)$
- Joint Entropy: $H(x,y) = -\sum_{(x,y) \in (\mathcal{X}, \mathcal{Y})} p(x,y) \log p(x,y)$
- Mutual Information: $I(x;y) = \sum_{(x,y) \in (\mathcal{X}, \mathcal{Y})} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$
- Relationship:

$$H(x,y) = H(x|y) + H(y) = H(y|x) + H(x) = H(x|y) + I(x;y) + H(y|x)$$



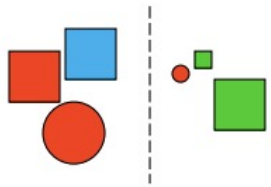
Reference: `https://en.wikipedia.org/wiki/Information_theory`

Example 3: choosing attribute via information gain

We record the training set as the following table.

The Training Set

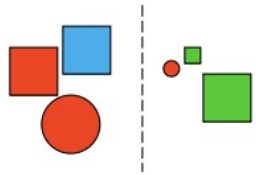
Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



Example 3: choosing attribute via information gain

We record the training set as the following table.

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



$H(Class) = H(3/6, 3/6) = 1$

$H(Class \mid Color) = 3/6\ H(2/3,1/3) + 2/6\ H(0/2, 2/2) + 1/6\ H(1/1, 0/1)$

3 out of 6 are red

2 of the red are +

2 out of 6 are green

both green are −

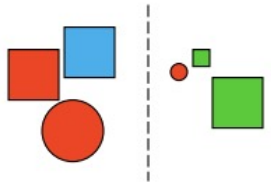
1 out of 6 is blue

blue is +

Example 3: choosing attribute via information gain

We firstly calculate the entropy of the whole training set.
Then, if we choose Color as the root node, we calculate the entropy of all child nodes and the reduction of entropy (*i.e.*, information gain)

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



$H(Class) = H(3/6, 3/6) = 1$

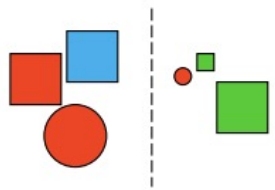
$H(Class \mid Color) = 3/6\ H(2/3,1/3) + 1/6\ H(1/1, 0/1) + 2/6\ H(0/2, 2/2)$

$I(Class; Color) = H(Class) - H(Class \mid Color) = 0.54$ bits

Example 3: choosing attribute via information gain

If we choose Shape as the root node, we calculate the entropy of all child nodes and the reduction of entropy (*i.e.*, information gain)

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



$H(Class) = H(3/6, 3/6) = 1$

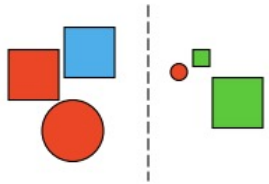
$H(Class \mid Shape) = 4/6 * H(2/4, 2/4) + 2/6 * H(1/2,1/2) = 1$

$I(Class; Shape) = H(Class) - H(Class \mid Shape) = 0$ bits

Example 3: choosing attribute via information gain

If we choose Size as the root node, we calculate the entropy of all child nodes and the reduction of entropy (*i.e.*, information gain)

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



$H(Class) = H(3/6, 3/6) = 1$

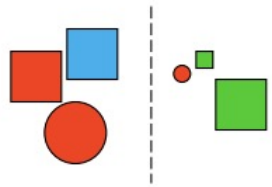
$H(Class \mid Size) = 4/6 * H(3/4, 1/4) + 2/6 * H(0/2, 2/2) = 0.54$

$I(Class; Size) = H(Class) - H(Class \mid Size) = 0.46$ bits

Example 3: choosing attribute via information gain

We pick the attribute with the largest information gain, *i.e.*, Color

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−

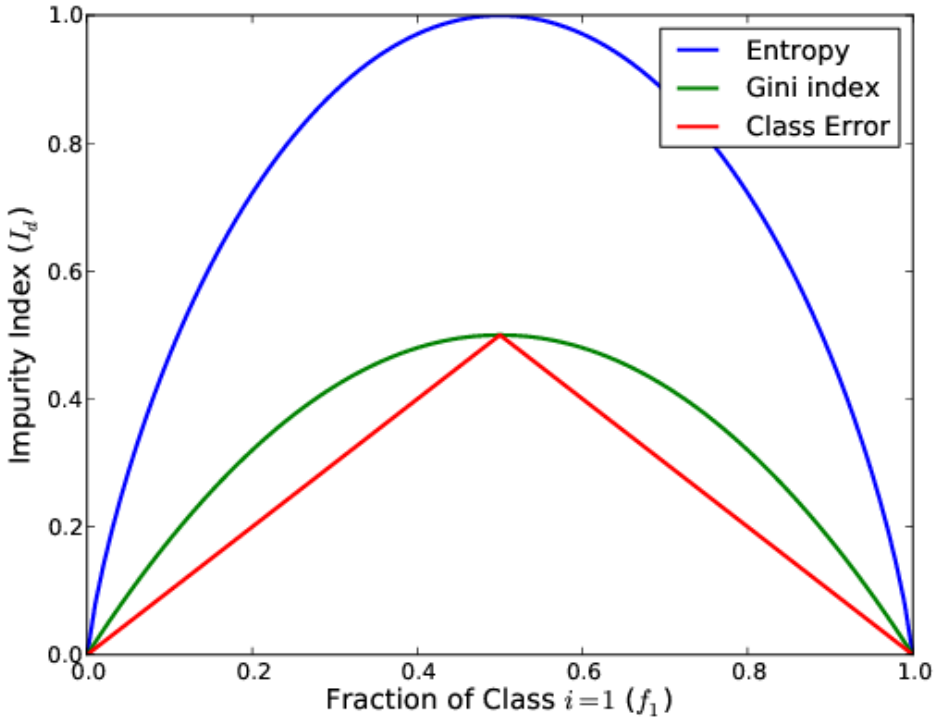


$I(Class; \textbf{Color}) = H(Class) - H(Class \mid Color) = 0.54$ bits
$I(Class; \textbf{Shape}) = H(Class) - H(Class \mid Shape) = 0$ bits
$I(Class; \textbf{Size}) = H(Class) - H(Class \mid Size) = 0.46$ bits

➔ Select ***Color*** as the best attribute at the root

Common Impurity Measures for Binary Problem

- Entropy: $\phi(p, 1-p) = -p \log_2 p - (1-p) \log_2 (1-p)$
- Gini Index: $\phi(p, 1-p) = 2p(1-p)$
- Misclassification Error: $\phi(p, 1-p) = 1 - \max(p, 1-p)$



Common Impurity Measures for Multicategory Problem

- Entropy:

$$\phi(p) = -\Sigma_{i=1}^{K} p_i \log_2 p_i$$

- Gini Index:

$$\phi(p) = \Sigma_{i=1}^{K} p_i \left(1 - p_i\right) = 1 - \Sigma_{i=1}^{K} p_i^2$$

Regression Trees

- A *regression tree* is constructed in almost the same manner as a classification tree, except that the impurity measure that is appropriate for classification is replaced by a measure appropriate for regression.
- In regression, the goodness of a split is measured by the mean square error (MSE) or the sum of squared errors (SSE) from the estimated value.
- The prediction for leaf c is $\bar{y}_c = \frac{\sum_{i \in \text{ leaf}_c} y_i}{N_c}$.
- Within each leave c, the MSE can be computed as $e_c = \frac{\sum_{i \in \text{leaf}_c} (y_i - \bar{y}_c)^2}{N_c}$
- The total MSE is $\mathcal{S} = \sum_{c \in \text{leaves(Tree)}} e_c$.
- The total SSE is $\mathcal{S} = \sum_{c \in \text{leaves(Tree)}} \sum_{i \in c} (y_i - \bar{y}_c)$. (preferred)

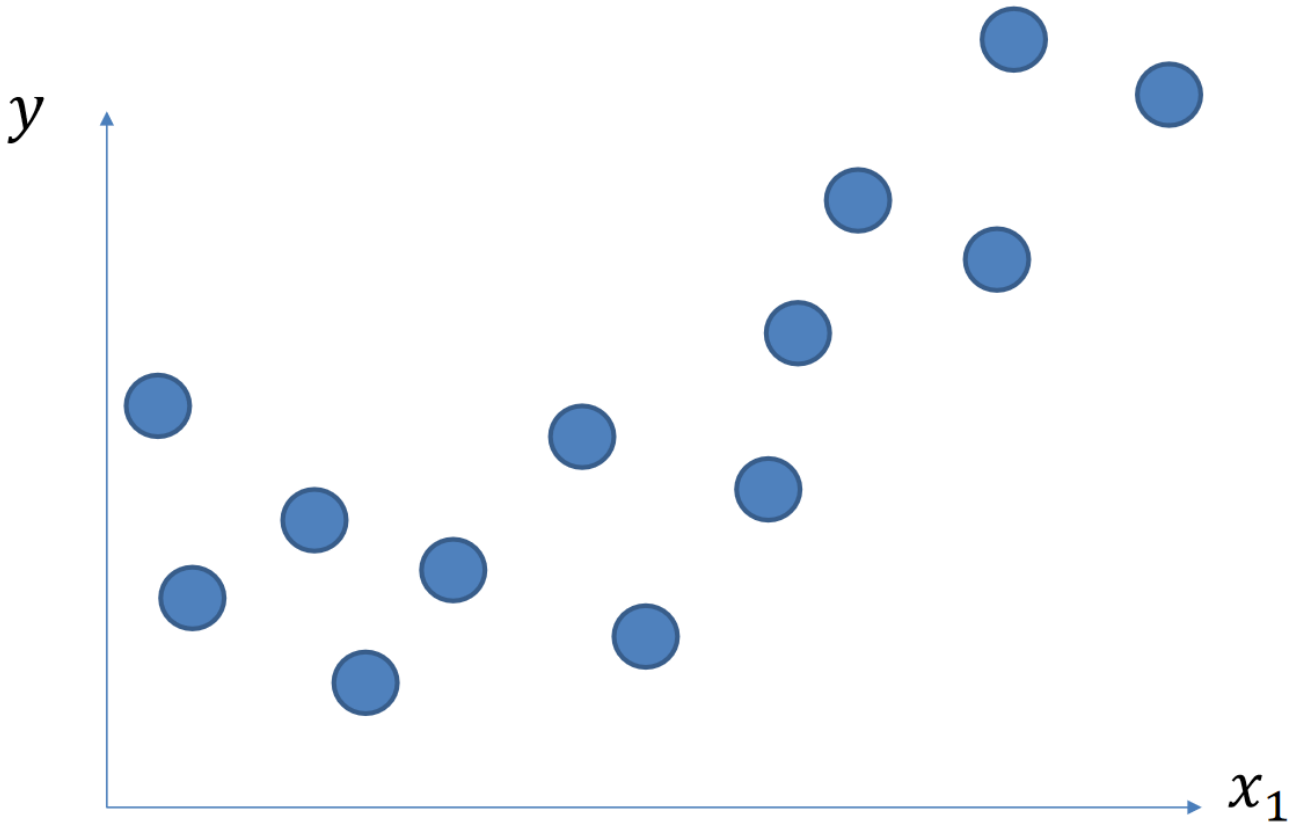
Regression Trees

Basic Regression-Tree-Growing Algorithm

1. Start with a single node containing all points. Calculate $\bar{y}$ and $\mathcal{S}$.
2. For each node,
   - If all the points in the node have the same value for all the independent variables, **stop**.
   - Otherwise, search over all binary splits of all variables for the one which will reduce $\mathcal{S}$ as much as possible.
     - If the largest decrease in $\mathcal{S}$ would be less than some threshold δ, or one of the resulting nodes would contain less than q points, **stop**.
     - Otherwise, **take that split**, creating two new nodes.
3. In each new node, go back to step 1.

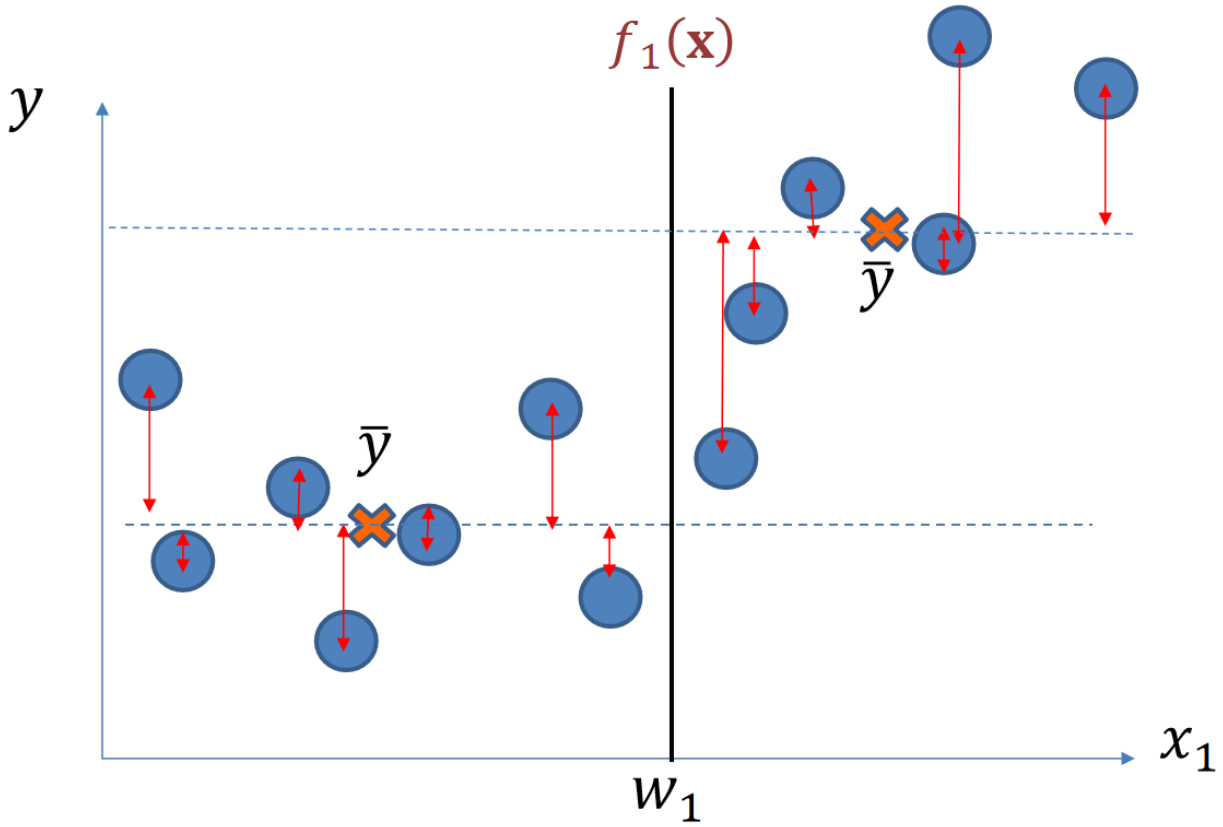
Example 4: build a regression tree

We consider to build regression tree based on the following 1-dimensional data.



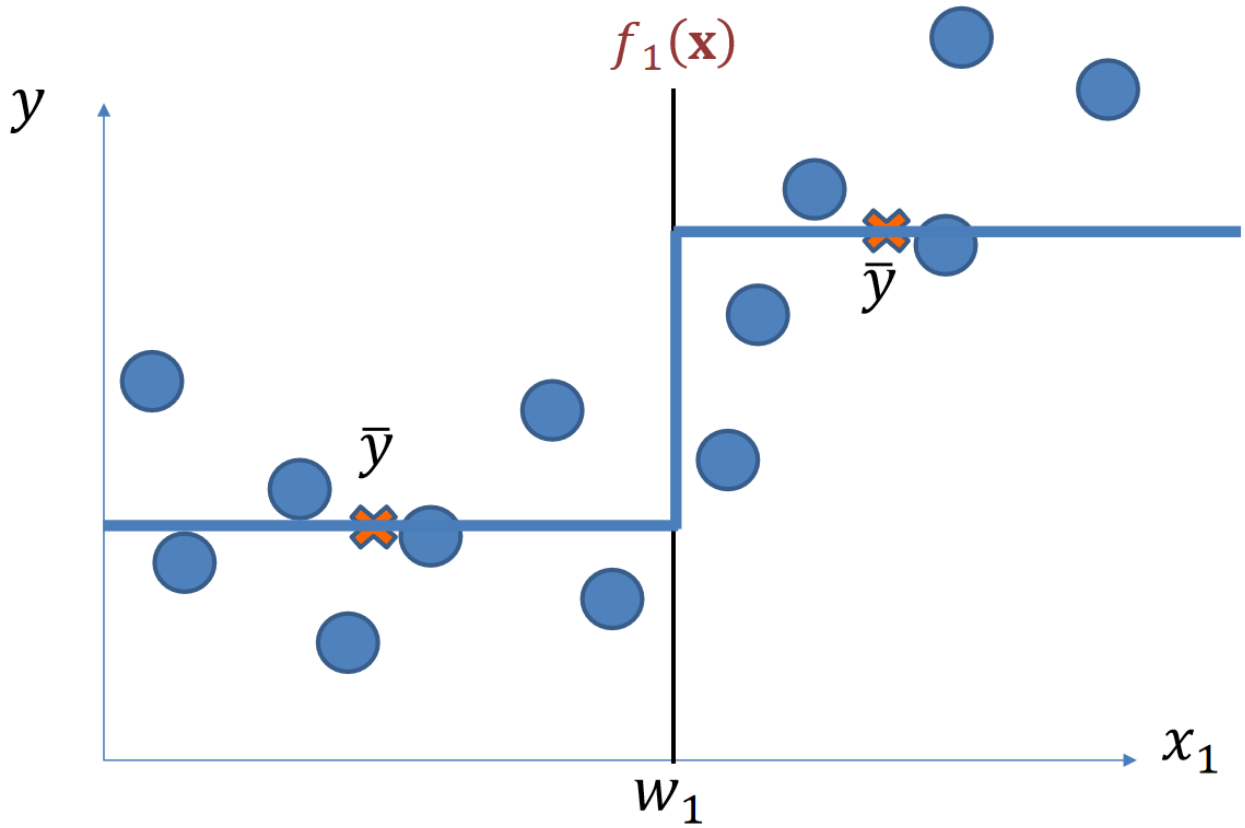
Example 4: build a regression tree

- Firstly, we calculate the average output $\bar{y}$ of all points, as well as the MSE $\mathcal{S}$.
- Then, we search for a threshold w_1 which leads to the largest decrease of $\mathcal{S}$.



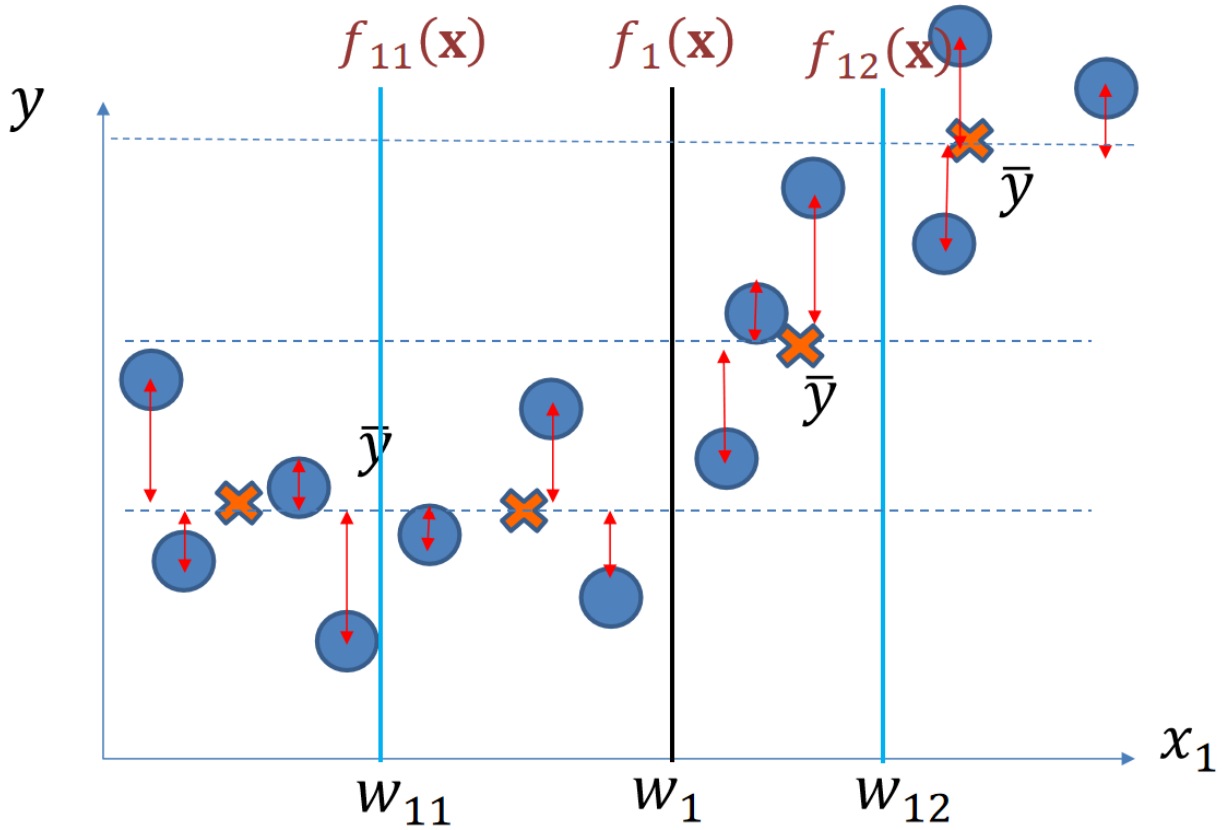
Example 4: build a regression tree

Repeat the above steps for each child node.



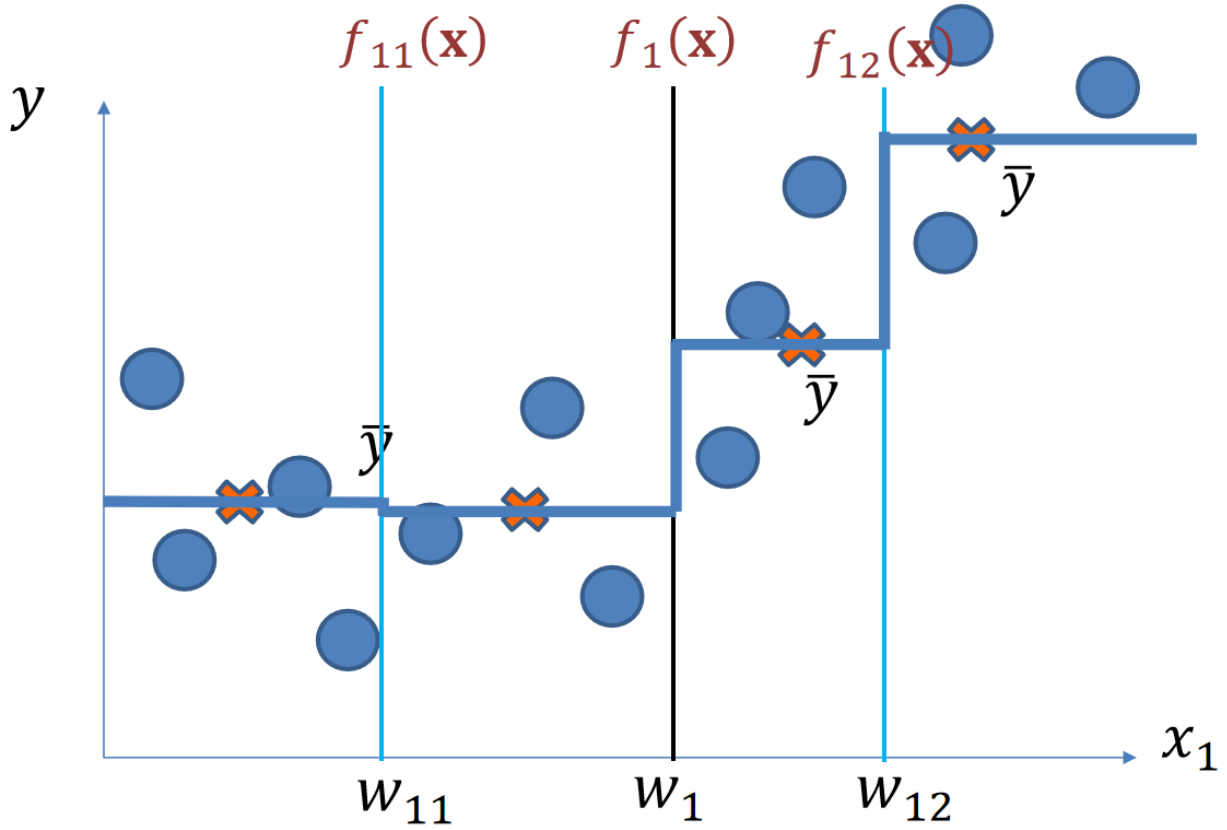
Example 4: build a regression tree

Repeat the above steps for each child node.



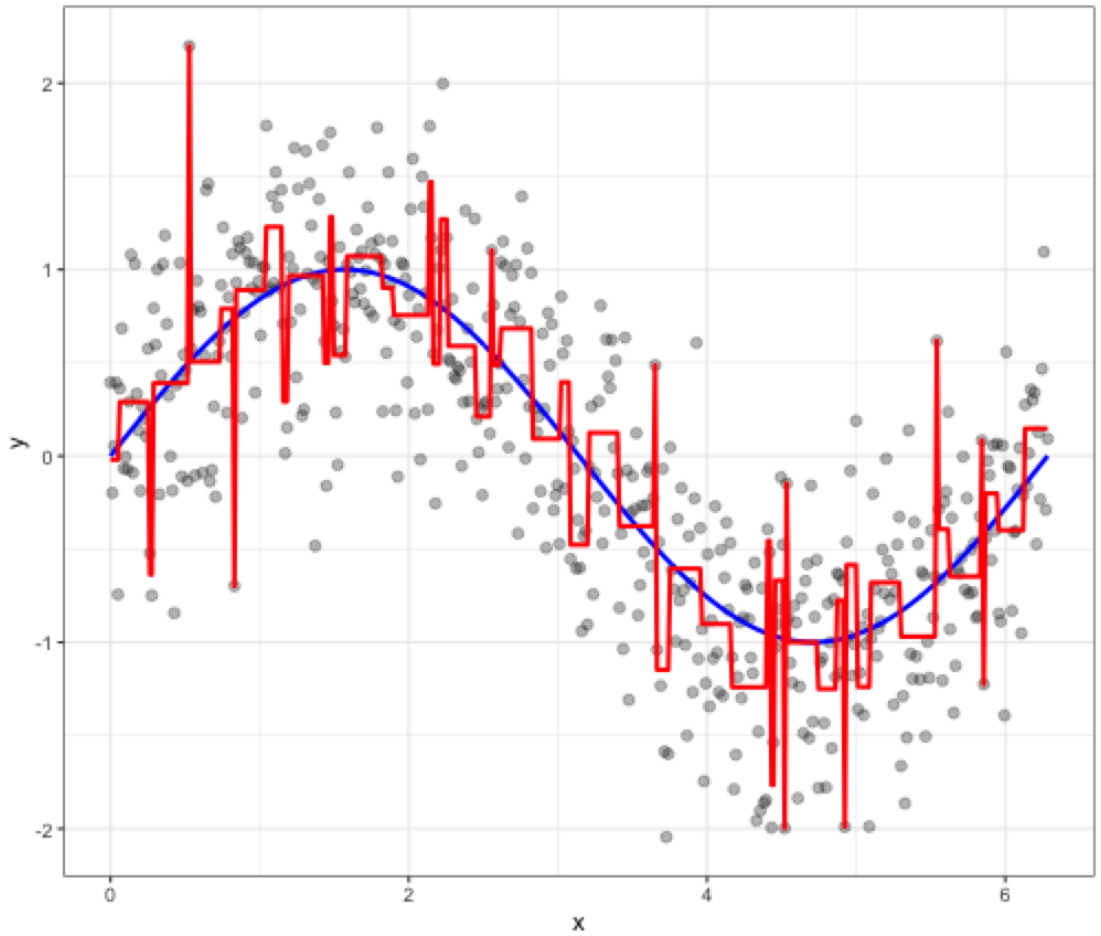
Example 4: build a regression tree

Repeat the above steps for each child node, until some stop conditions are satisfied.



Overfitting

Trees have a tendency to overfit to training data, such that the prediction error on testing data is likely to be high.
As shown in the following example, the prediction of the constructed tree is highly non-smooth.



Overfitting and Pruning

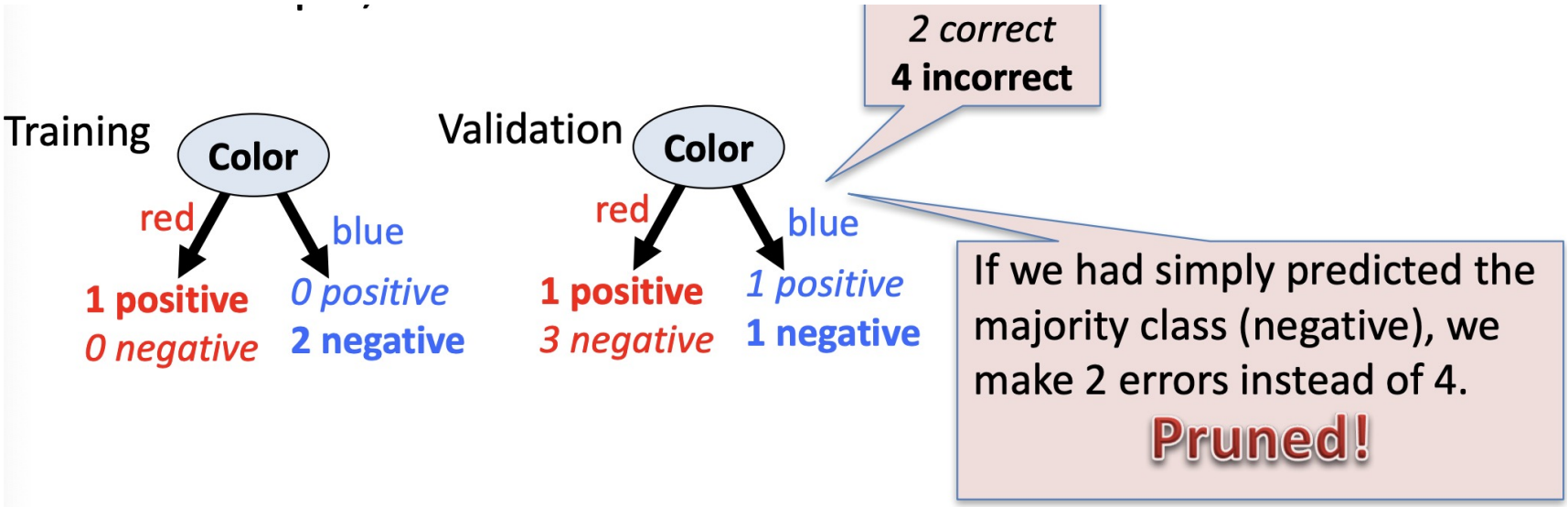
An effective approach to alleviate overfitting is pruning.
The general procedure of pruning is as follows:

- Split training data further into training and validation sets
- Grow a deep tree based on training set
- Do until further pruning is harmful:
  - Evaluate impact on validation set of pruning each possible node
  - Greedily remove the node that most improves validation set accuracy

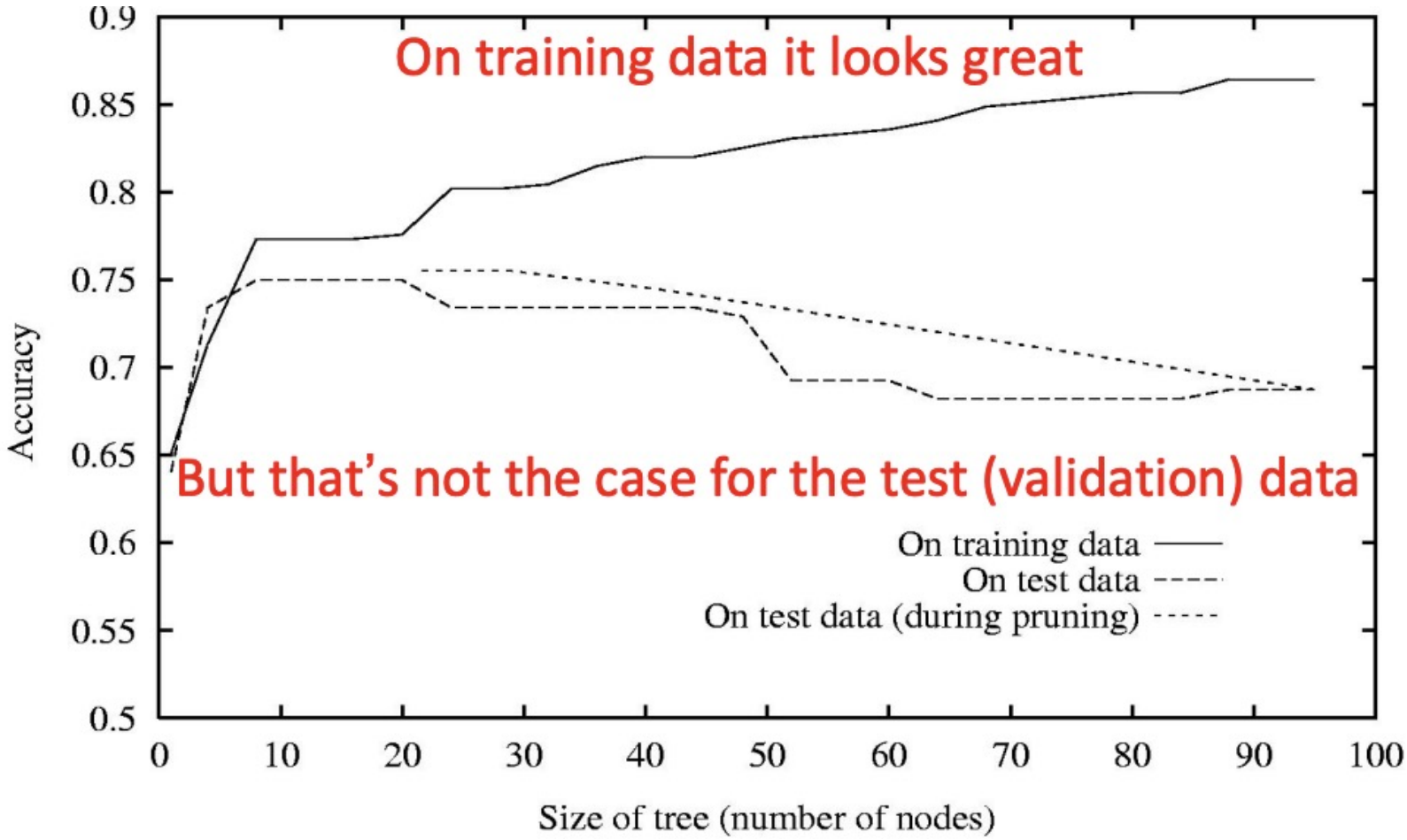
Reference: `https://faculty.cc.gatech.edu/~bboots3/CS4641-Fall2018/Lecture2/02_DecisionTrees.pdf`

Overfitting and Pruning

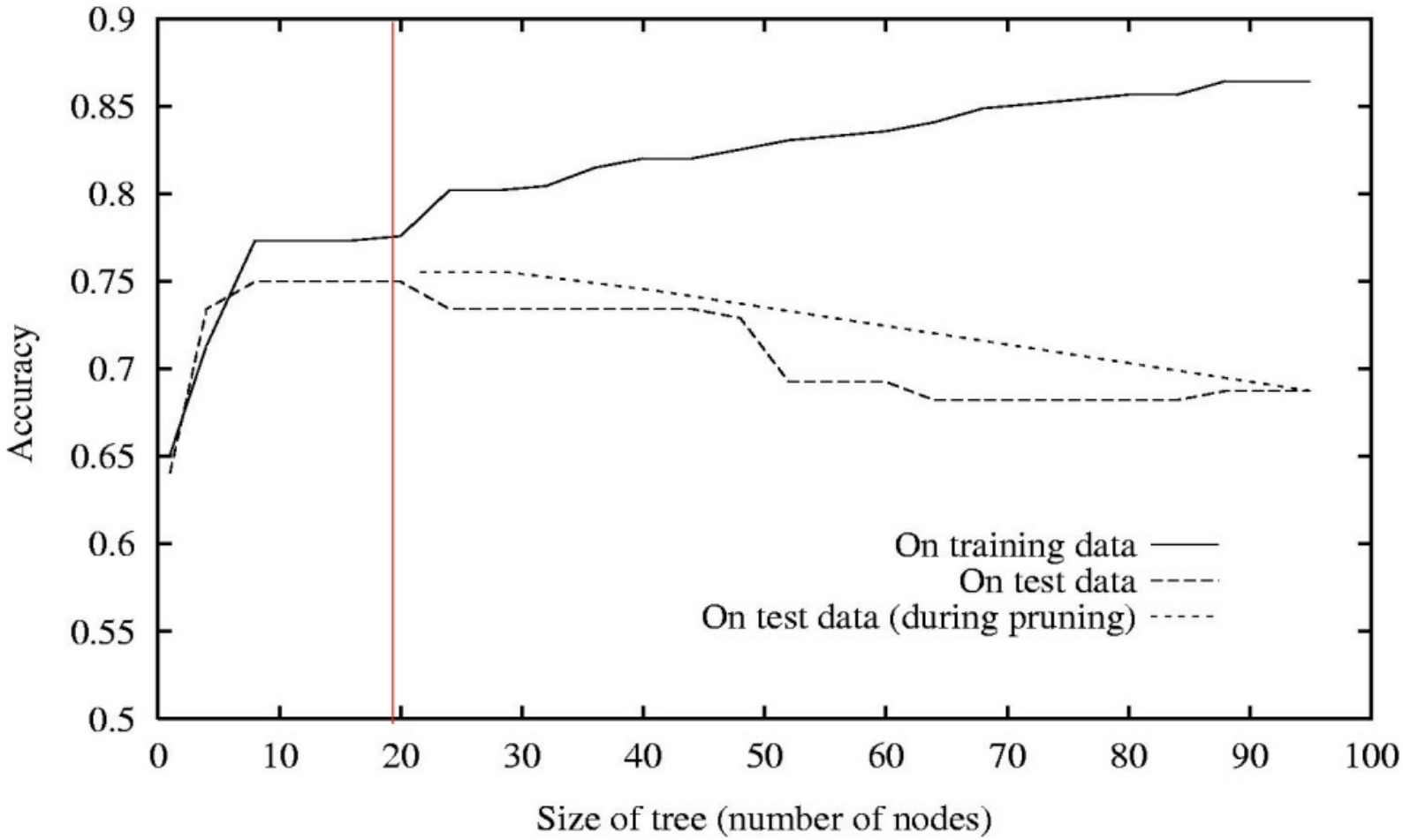
- Pruning of the decision tree is done by replacing a whole subtree by a leaf node
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf
- For example,



Effect of Reduced-Error Pruning



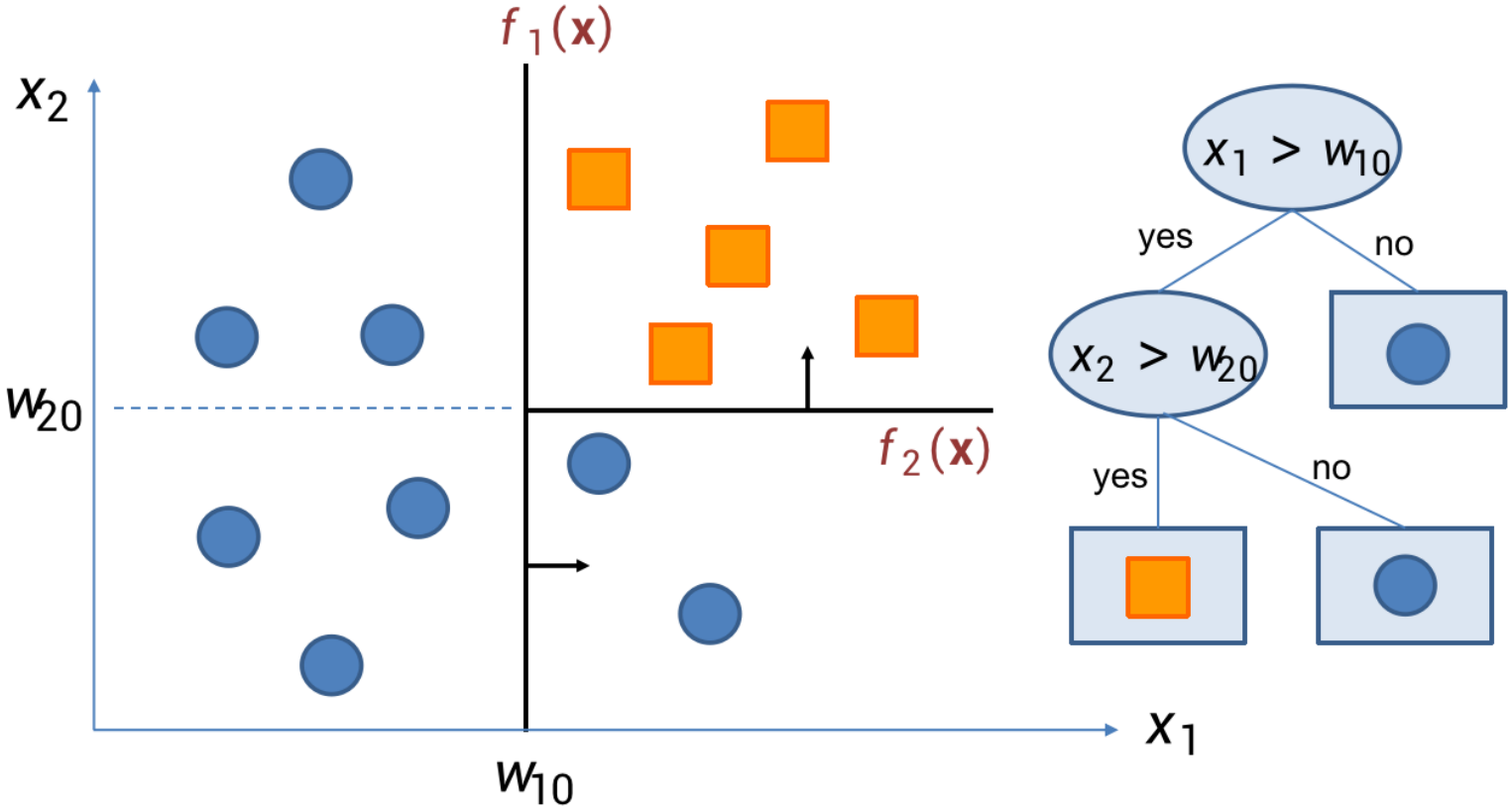
Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the test data

Rule Extraction

Rule Extraction From Trees, and the decision is interpretable

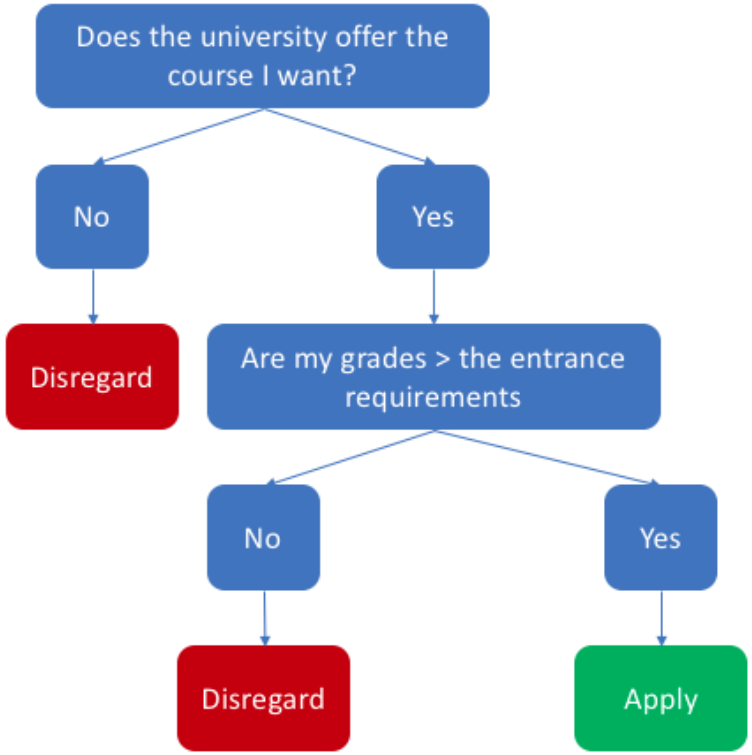


- R1: If $x_1 \leq w_{10}$ then classify as circle
- R2: If $x_1 > w_{10}$ and $x_2 > w_{20}$ then classify as square
- R3: If $x_1 > w_{10}$ and $x_2 \leq w_{20}$ then classify as circle

Rule Extraction

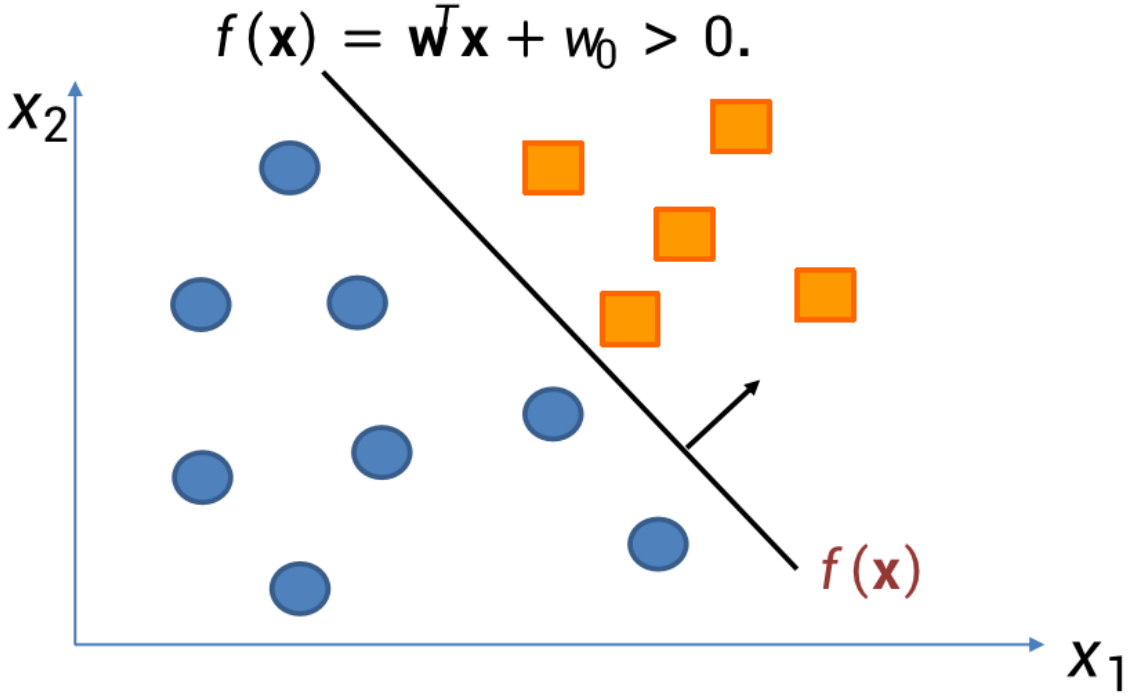
A simple example is how one may evaluate local universities when the leave high school.

- R1: If the university doesn't offer the course I want, then Disregard this university
- R2: If the university offers the course I want and my grades don't pass the entrance requirements, then Disregard this university
- R3: If the university offers the course I want and my grades pass the entrance requirements, then Apply this university



Multivariate Trees

- In a multivariate tree, at a decision node, all input dimensions can be used and thus it is more general. When all inputs are numeric, a binary linear multivariate node is defined as



Parameters of decision trees

Even though the algorithms are considered nonparametric, there are still some (hyper)-parameters used for defining a tree:

- Minimum samples for a node split
- Minimum samples for a terminal/leaf node
- Maximum attributes to consider for split
- Maximum depth of tree (vertical depth)

Parameters of decision trees

Minimum samples for a node split

- Defines the minimum number of samples (or observations) which are required in a node to be considered for splitting.
- Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
- Too high values can lead to under-fitting hence, it should be tuned using cross-validation.

Parameters of decision trees

Minimum samples for a terminal/leaf node

- Defines the minimum samples (or observations) required in a terminal node or leaf.
- Used to control over-fitting similar to min_samples_split.
- Generally lower values should be chosen for imbalanced class problems because the regions in which the minority class will be in majority will be very small.

Parameters of decision trees

Maximum attributes to consider for split

- The number of attributes to consider while searching for a best split. These will be randomly selected.
- As a thumb-rule, square root of the total number of attributes works great but we should check up to 30-40% of the total number of attributes.
- Higher values can lead to over-fitting but depends on case to case.

Parameters of decision trees

Maximum depth of tree (vertical depth)

- The maximum depth of a tree.
- Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
- Should be tuned using cross-validation.

Summary of Decision Trees

Advantages:

- Easy to understand (interpretability)
- Useful in data exploration (rule extraction)
- Less data cleaning/pre-processing required
- Data type is not a constraint
- Non-parametric method

Disadvantages:

- Overfitting: Overfitting is one of the most practical difficulty for decision tree models. This problem gets solved by setting constraints on model parameters and pruning.
- Continuous variables: While working with continuous numerical variables, decision tree loses information when it categorizes variables into categories (quantization error).

1. Decision Trees: Motivation

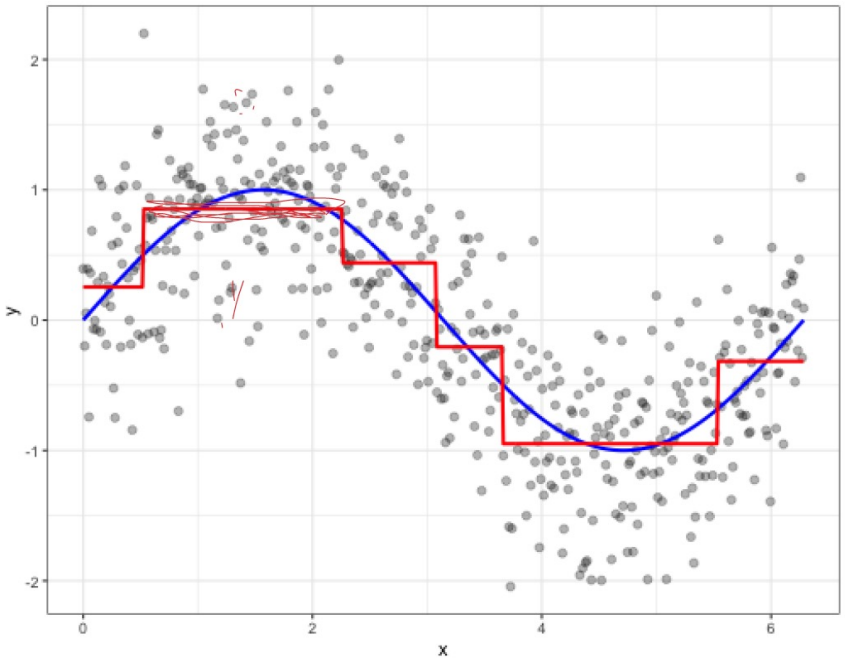
2. Univariate Trees
   - Definition and Example
   - How to build a decision tree
   - Classification Trees
   - Regression Trees
   - Others

3. Ensemble models
   - Bagging
   - Random Forest

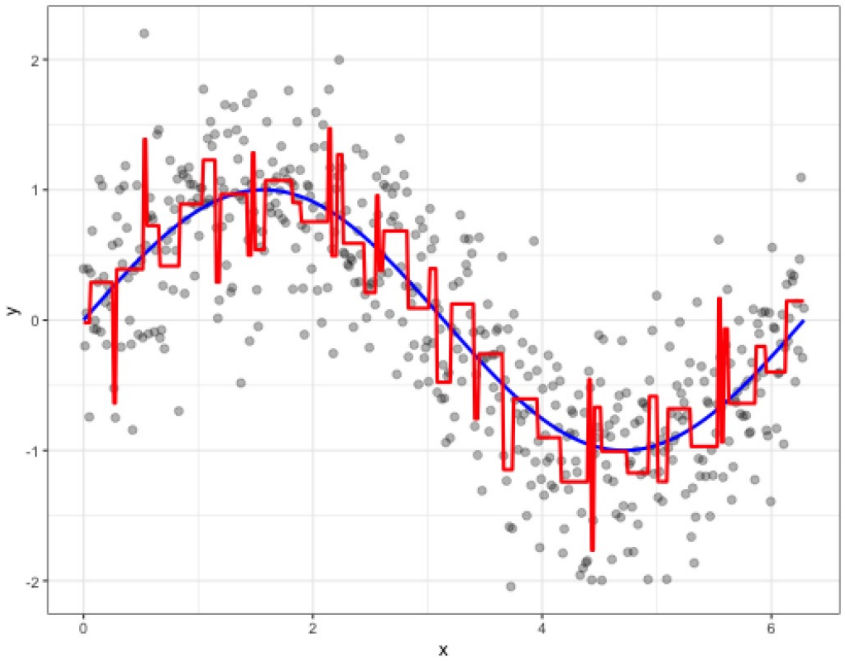
4. Further reading

Drawbacks of single decision tree

Single pruned trees are poor predictors



Single deep trees are overfitting

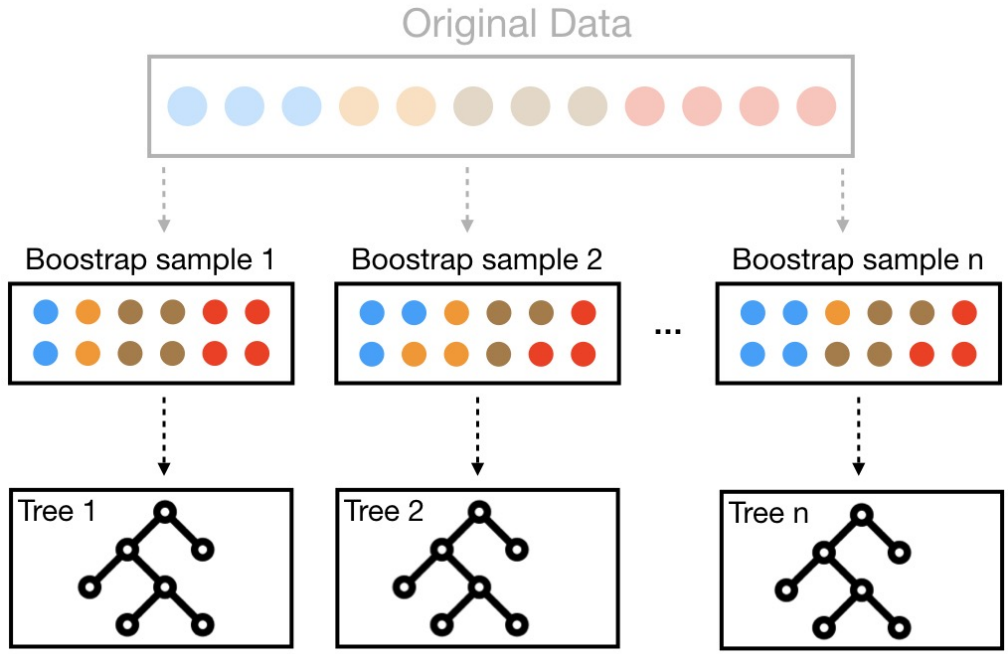


Ensemble models

- To address generalization issues of single decision trees, we introduce ensemble models.
- The basic idea is constructing many diverse decision trees, then combine their predictions as the final prediction (majority for classification, or average for regression).
- The philosophy is that the wisdom of the crowd is likely higher than singles. 三个臭皮匠，胜过诸葛亮
- We introduce two ensemble models of decision trees:
  - Bootstrap Aggregating (Bagging)
  - Random Forests

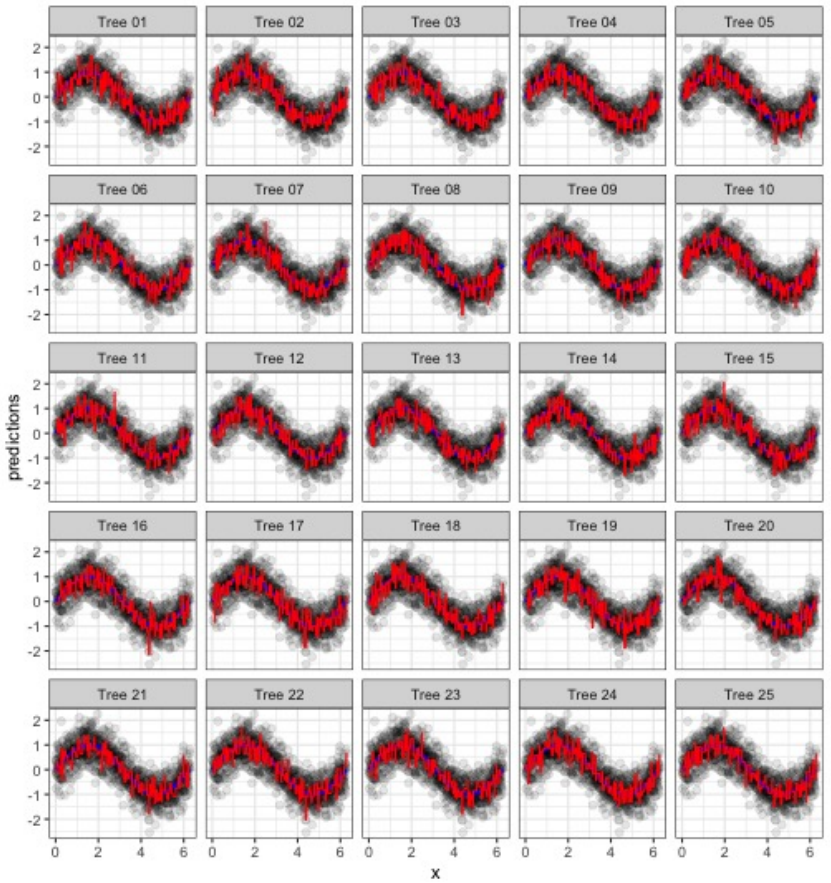
Bootstrap Aggregating: wisdom of the crowd (Bagging)

- **Step 1**: Sample records with replacement (aka "bootstrap" the training data), to obtain several diverse training data sets (data-level randomness)
- **Step 2**: Fit an overgrown tree to each resampled training data set.



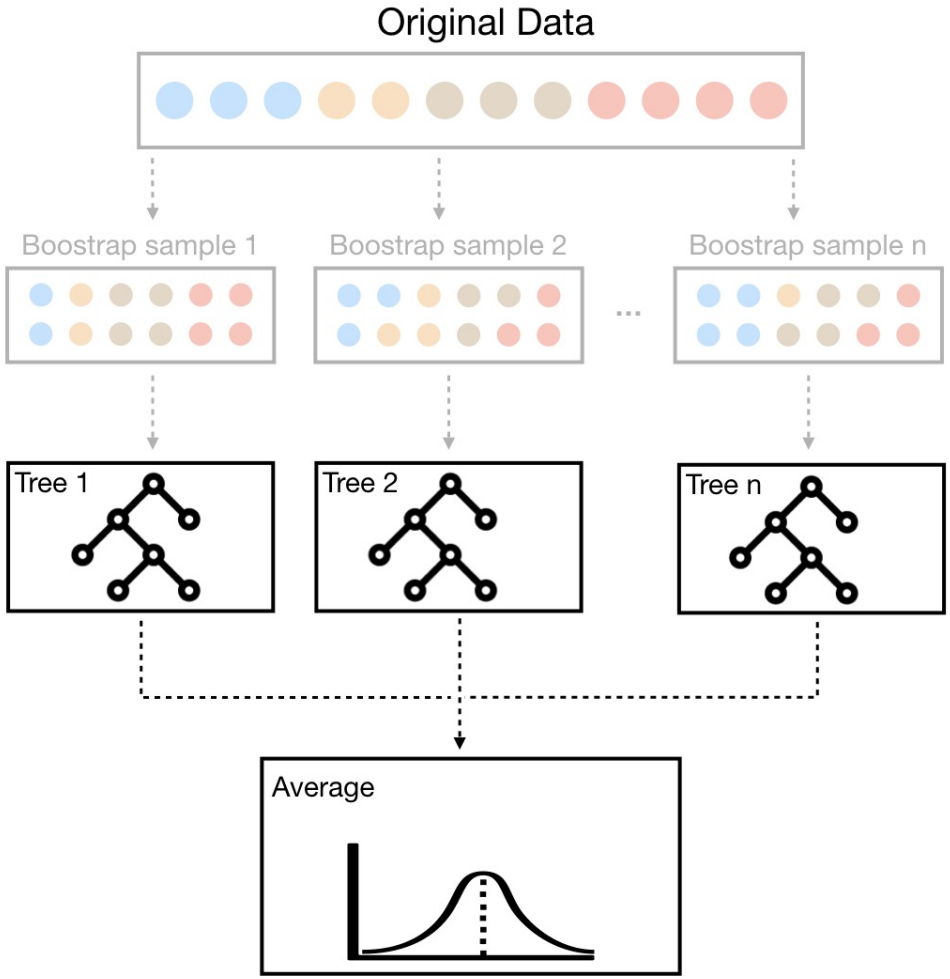
Bootstrap Aggregating: wisdom of the crowd (Bagging)

- **Step 1**: Sample records with replacement (aka "bootstrap" the training data), to obtain several diverse training data sets (data-level randomness)
- **Step 2**: Fit an overgrown tree to each resampled training data set, and we obtain several diverse decision trees



Bootstrap Aggregating (Bagging)

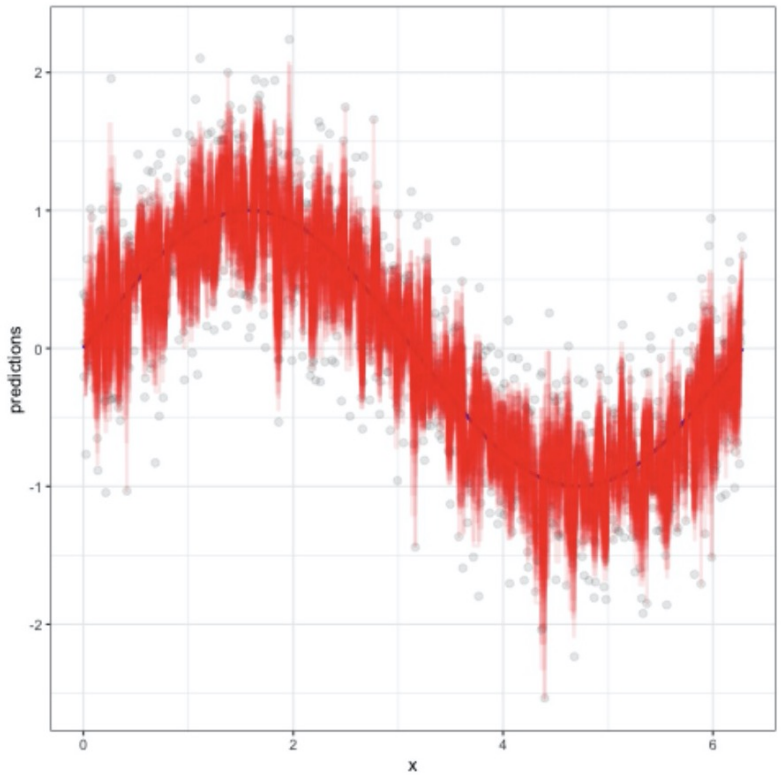
- **Step 1**: Sample records with replacement (aka "bootstrap" the training data), to obtain several diverse training data sets (data-level randomness)
- **Step 2**: Fit an overgrown tree to each resampled training data set
- Aggregate the predictions of all single trees (majority or average)



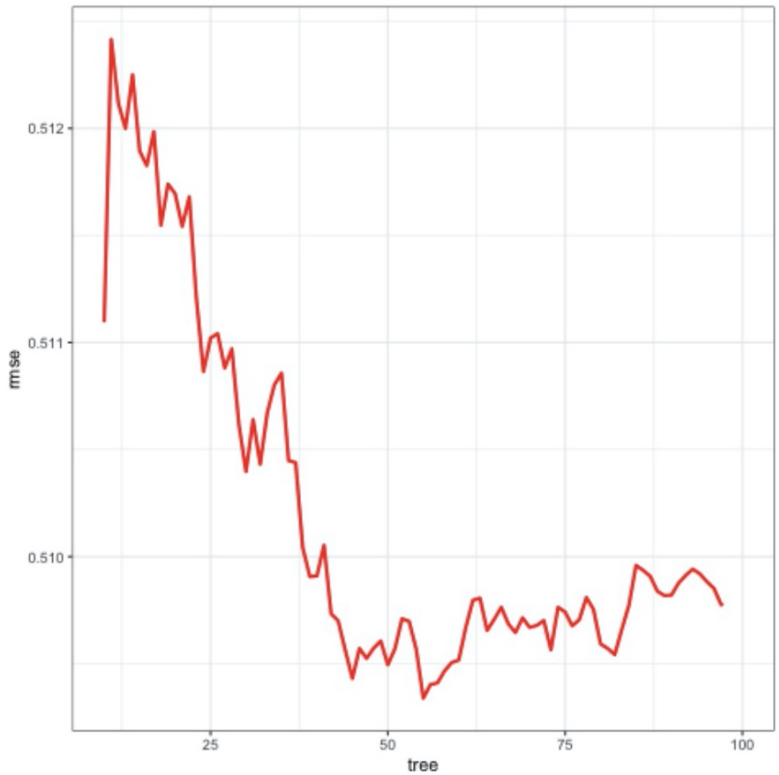
Bootstrap Aggregating (Bagging)

Bagging with more single trees will decrease the prediction error.

As we add more trees...

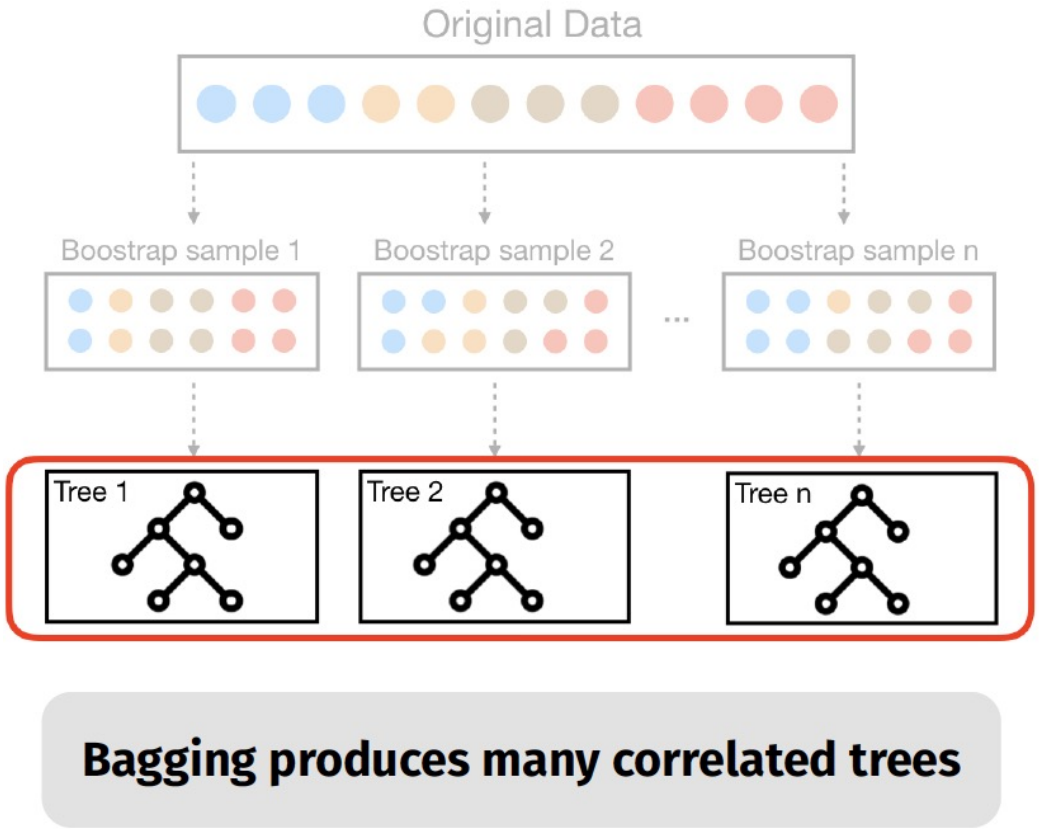


our average prediction error reduces



Bootstrap Aggregating (Bagging)

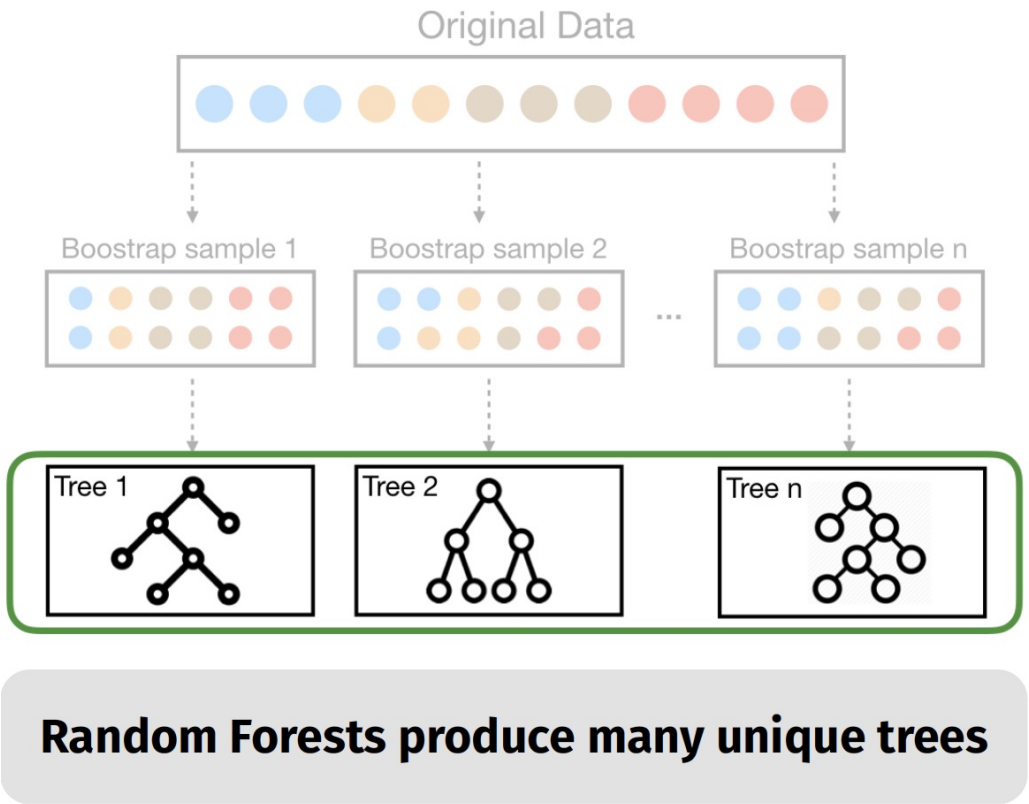
However, since there are many shared samples between two resampled training data sets, Bagging is likely to produce many correlated trees.
The diversity is not large enough.



Random Forest

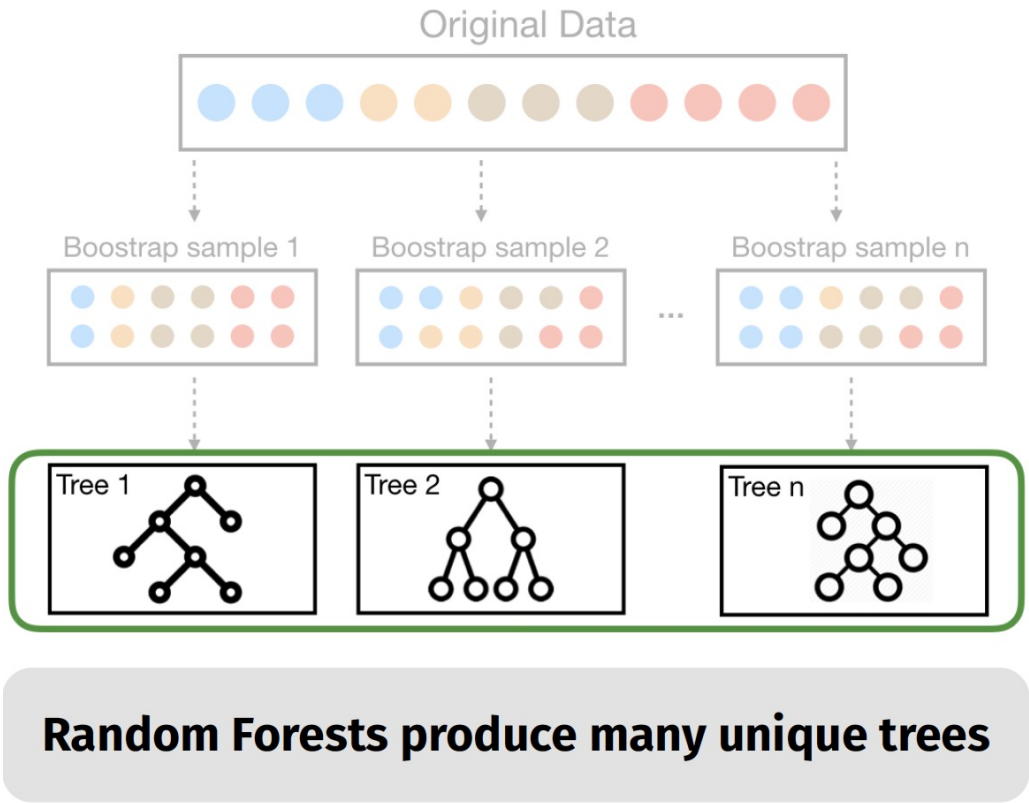
- To reduce the correlation among the trees produced by Bagging, we introduce split-attribute randomization into the model.
- It is called Random Forests. Many trees form a forest!

- Follow a similar bagging process but...
- Each time a split is to be performed, the search for the split attribute is limited to a random subset of m of the N attributes
  - For regression trees: $m = \frac{N}{3}$
  - For classification trees: $m = \sqrt{N}$



Random Forest

- Bagging introduces randomness into the data-level
- Random forests introduces randomness into both the data-level and attribute level
- Prediction error: Random forest $<$ Bagging $<$ single trees



The ensemble models can alleviate overfitting. A brief understanding is that each single decision tree in ensemble models overfits a different data set and attributes, to avoid the overfitting to the fixed original data set as did in single decision trees.

Further reading

- An Online dynamic slides about decision tree
- Function and demos of decision tree in sklearn
- Document of decision tree in sklearn