




# ENSEMBLE LEARNING

CS576 MACHINE LEARNING



Dr. Jin Soung Yoo, Professor  
Department of Computer Science  
Purdue University Fort Wayne

# Reference

- J. D. Kelleher et al., Machine Learning for Predictive Data Analytics, Ch 4.4.5
- G. James et al., An Introduction to Statistical Learning, Ch 8.2
- P. Tan et al., Introduction to Data Mining, Ch 4.10
- A. Géron, Hands-On Machine Learning, 3<sup>nd</sup> ed., Ch7

# Outline

- Introduction
- Types of Constructing Ensembles
- Ensemble Learning Methods
  - Bagging
  - Random Forests
  - Boosting
  - Gradient Boosting

# Motivation

- Ideally, we would like our predictions to be as close to the true target as possible. However different predictions are possible based on differences in the training data or the approach used for model selection.
- The **generalization (test) error of a model  $m$**  can be decomposed into terms involving the *bias*, *variance*, and *noise* components of the model in the following way:

$$\text{gen.error}(m) = c_1 \times \text{noise} + \text{bias}(m) + c_2 \times \text{variance}(m)$$

, where  $c_1$  and  $c_2$  are constants

- The bias and variance terms depend on the choice of prediction model
- The **bias** of a model represents how close the average prediction of the model is to the average target.
- The **variance** of a model captures the *stability* of its predictions in response to minor perturbations in the training set or the model selection approach.
- A model shows better generalization (test) performance **if it has a lower bias and lower variance.**

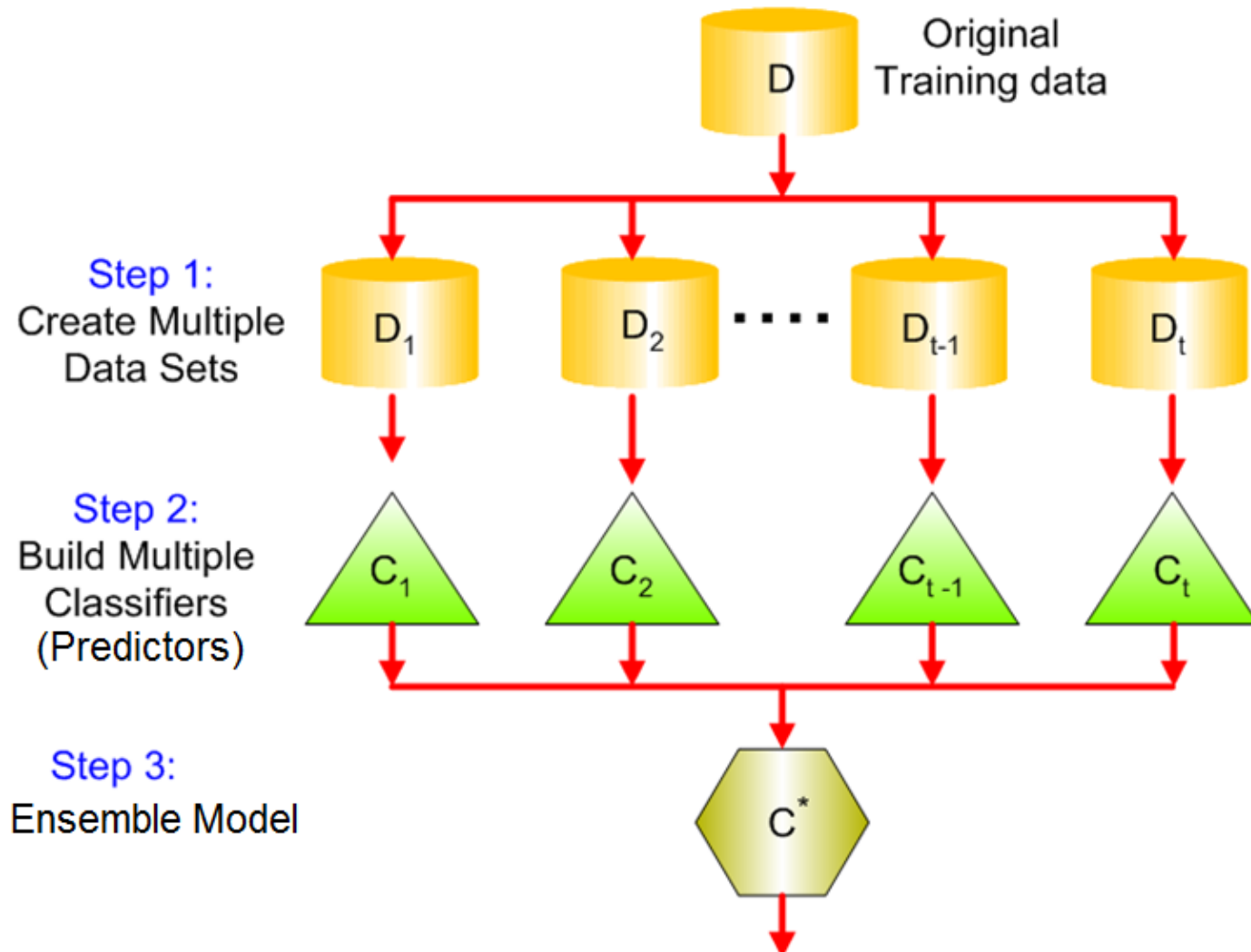
# Introduction: *Wisdom of the Crowd*

- Rather than creating a single model, **by combining the responses of multiple base models, we can expect to reduce the overall variance of the prediction.**
- A group of predictors (such as classifiers or regressors), is called an *ensemble* (*model ensemble*).
- The technique is called *ensemble learning*.
- An ensemble learning algorithm is called an *ensemble method* (*classifier combination method*).
- Ensemble methods show better performance primarily by lowering the variance in the predictions, although they can even help in reducing the bias.
- Ensemble methods can be applied to many (statistical) learning methods for classification or prediction.

# Introduction (cont.)

- The ensemble approach aims to **combine many “weak learners” into a single strong learner.**
- Any model can be used at the base model training steps
  - It is common to use **very shallow decision trees**, or *decision stumps* – often just one level deep.
- **Two necessary conditions for an ensemble model** to perform better than a single model
  - 1) The base models should be **independent** of each other
  - 2) The base models should do **better than a model which performs random guessing.**
- It is hard to ensure total independence (1) among the base classifiers. Nevertheless, improvements in classification accuracies have been observed in ensemble methods in which the base classifiers are somewhat correlated.

# General Framework of Ensemble Learning



**Figure:** Training diverse base models for an ensemble model

# Classification using Ensemble

- Once an ensemble of classification models has been learned, a test example  $x$  is classified by combining the predictions made by the base classifier  $C_i(x)$ :

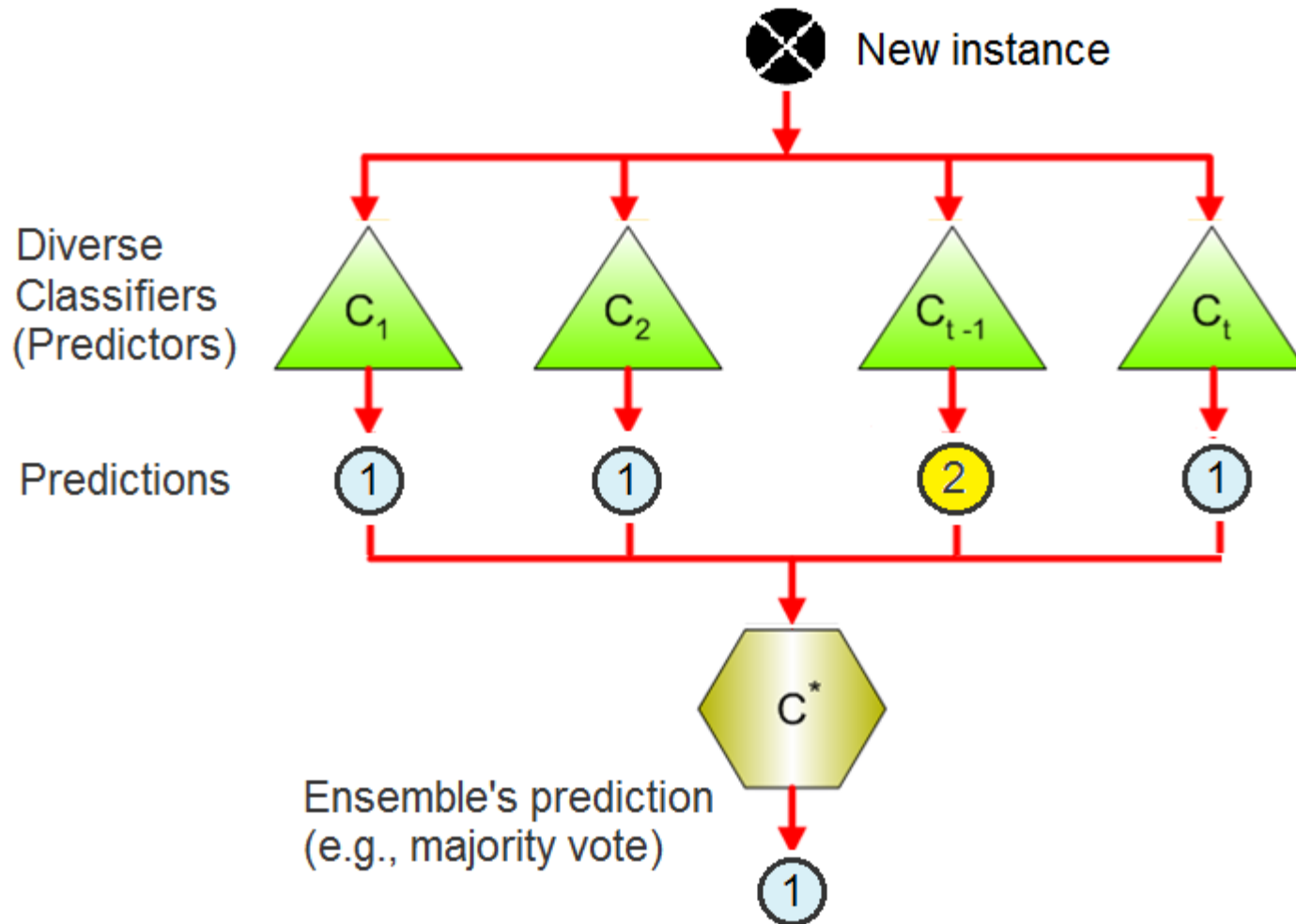
$$C^*(\mathbf{x}) = f(C_1(\mathbf{x}), \dots, C_k(\mathbf{x}))$$

, where  $f$  is the function that combines the ensemble responses.

- Output: the class label that gets
  - a *majority vote* of the individual predictions (simplest approach), or
  - a *weighted majority vote*, where the weight of a base classifier denotes its accuracy or relevance.
- The majority-vote classifier is called a *hard voting* classifier.



# Classification using Ensemble



# Outline

- Introduction
- ☞ **Types of Constructing Ensembles**
- Ensemble Learning Methods
  - Bagging
  - Random Forests
  - Boosting
  - Gradient Boosting

# Types of Constructing Ensembles

## 1. By manipulating the training dataset

- Multiple training sets are created by resampling the original data according to the some sampling distribution
  - The sampling distribution determines how likely it is that an example will be selected for training.
- Construct a classifier from each training set.
- E.g., **Bagging** and **Boosting**

## 2. By manipulating the input features

- A subset of input features is chosen to form each training set. The subset can be either chosen randomly or based on the domain knowledge.
- Works well with data sets that contain highly redundant features
- E.g., **Random Forests**

# Methods for Constructing an Ensemble (Cont.)

## 3. By manipulating the class labels

- When the number of classes is sufficiently large
- Conduct the classification by generating a variety of binary classifiers that predict output.
- The training data is transformed into a binary class problem
  - by randomly partitioning the data into two disjoint subsets,  $A_0$  (with Class = 0) and  $A_1$  (with Class = 1).
  - The relabeled examples are then used to train a base classifier.
  - This process is repeated multiple times to get an ensemble of binary classifiers
  - If the test example is predicted as class 0, then all the classes that belong to  $A_0$  will receive a vote. The class that receives the highest vote is assigned to the test example.
- E.g., **Error-Correcting Output Codes (ECOC)**

# Methods for Constructing an Ensemble (Cont.)

## 4. By manipulating the learning algorithm

- Apply a learning algorithm (with different hyper-parameter values) on the same training data multiple times to get different base models
    - E.g., an artificial neural network can change its network topology or the initial weights of the links between neurons.
  - The trained base models are combined to make the ensemble model.
- 
- 1, 2 and 3 approaches are generic, whereas 4 approach depends on the type of a learning algorithm used
  - The base models from most of these approaches can be generated **sequentially** (one after another) **or in parallel** (all at once).

# Outline

- Introduction
- Types of Constructing Ensembles
- ☞ **Ensemble Learning Methods**
  - Bagging
  - Random Forests
  - Boosting
  - Gradient Boosting

# Ensemble Methods

- Two standard approaches to creating ensembles are:
  - **Bagging.**
  - **Boosting**
- Their variations are:
  - **Random Forests**
  - **Gradient Boosting**

# General Ensemble Methods

- Two standard ensemble methods of model average are:
  - **Bagging** (Breiman, 1996): Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote. Bagging is also known as **bootstrap aggregation**
  - **Boosting** (Freund & Shapire, 1996): Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.
- Bagging and boosting are a general approach that can be applied to many (statistical) learning methods for regression or classification.



# Idea of Bagging

- A natural way to reduce the variance of a statistical learning method and hence increase the prediction accuracy is to
  - take many training sets from the population,
  - build a separate prediction model  $\hat{f}^1(x), \hat{f}^2(x), \dots, \hat{f}^B(x)$  using each training set, and
  - average the resulting predictions  $\hat{f}_{avg}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x)$  (for regression problem)
- But this is not practical because we generally do not have access to multiple training sets.
- Instead, we can **bootstrap**, by taking repeated samples from the (single) training data set.
- **Bagging** (ensemble) trains each base model on a random sample of the training data known as **bootstrap samples**.

# Bootstrap Samples

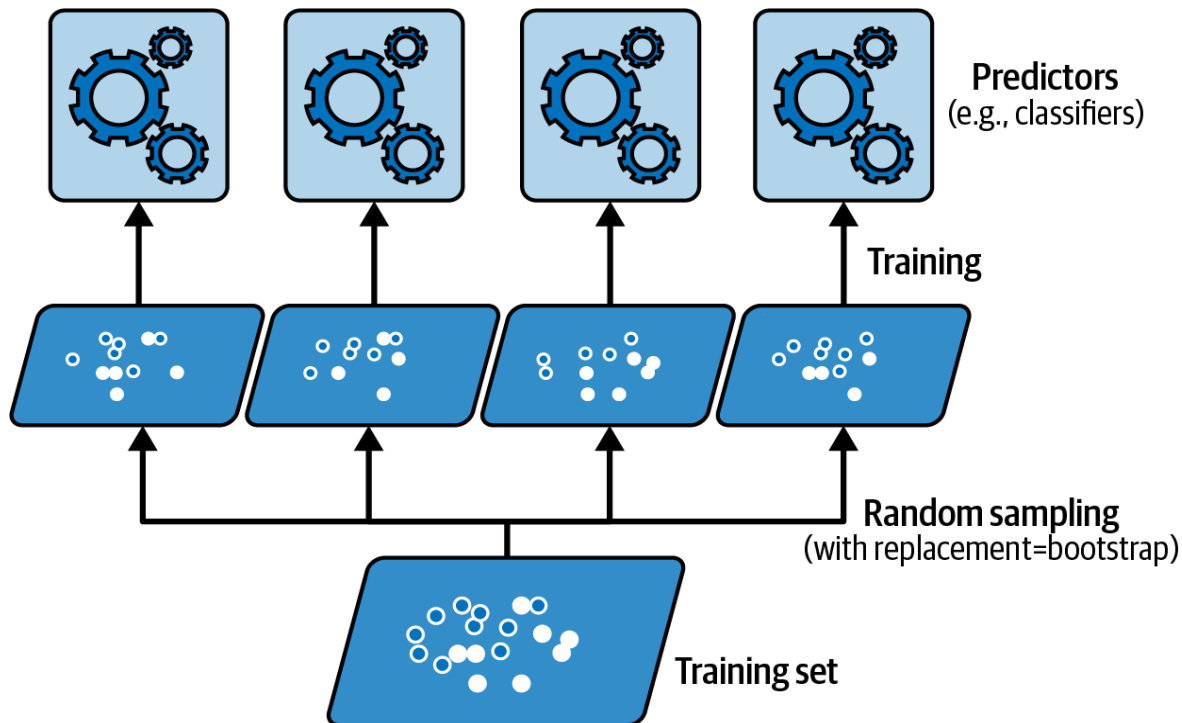
- Each of **bootstrap** data sets is created by **sampling with replacement from the original training data set**
- Each bootstrap size is **the same size as the original training set**
- Example

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	(8)	10	(8)	2	(5)	(10)	(10)	(5)	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- On average, a bootstrap sample contains approximately 63% of the original training data.

# General Approach of Bagging Ensemble

1. Generate  $m$  bootstrapped training data sets,  $D^{*1}, \dots, D^{*m}$  from the original training data set
2. Then train a base model on each bootstrapped training set.



## General Approach of Bagging Ensemble (cont.)

3. Once all predictors (classifiers) are trained, the ensemble can make a prediction for a new instance by simple **aggregating** the predictions of all predictors.
- For the **classification** for a test example  $x$ , take **a majority vote** (*statistical mode*) where the predicted class label is the most commonly occurring class among the  $m$  predictions, i.e.,  $\hat{C}_{bag}(x) = \text{Majority Vote } \{C(X^{*i}, x)\}_{i=1}^m$
  - For the **prediction (regression)** at a test example  $x$ , then **average** all the predictions to obtain:

$$\hat{f}_{avg}(x) = \frac{1}{B} \sum_{i=1}^m \hat{f}^{*i}(x)$$

, where  $\hat{f}^{*i}()$  is the base predictor (model) generated from  $i$ th bootstrapped set.

# Pseudo Code of Bagging Classification Algorithm

- 1: Let  $k$  be the number of bootstrap samples.
- 2: for  $i = 1$  to  $k$  do
- 3:   Create a bootstrap sample of size  $n$ ,  $D_i$ .
- 4:   Train a base classifier  $C_i$  on the bootstrap sample  $D_i$ .
- 5: end for
- 6:  $C^*(x) = \arg \max_y \sum_i \delta(C_i(x) = y)$ ,  $\{\delta(\cdot) = 1$  if its argument is true, and 0 otherwise. $\}$

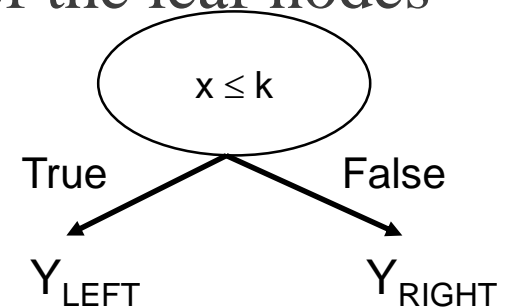
# Example

- Dataset -  $x$ : a descriptive feature,  $y$ : a target feature (class)

$x$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$y$	1	1	1	-1	-1	-1	-1	1	1	1

- For the classifier, consider a **decision stump** which has only one-level binary decision tree with a simple decision rule:  
 $x \leq k$  vs.  $x > k$  to minimize the entropy of the leaf nodes

- Example:** Classifier:  $x \leq 0.35 \rightarrow y = 1$   
 $x > 0.35 \rightarrow y = -1$



$x$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$y$	1	1	1	-1	-1	-1	-1	1	1	1

- Without bagging,  $k=0.35$  or  $k=0.75$  for the best. Accuracy of the single decision tree is 70% (3 training errors)

# Example: Building Base Classifiers Using Bootstraps

- When applying the **bagging procedure** using 10 bootstrap samples

Decision stump classifier

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$   
 $x > 0.35 \rightarrow y = -1$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

$x \leq 0.7 \rightarrow y = 1$   
 $x > 0.7 \rightarrow y = 1$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$   
 $x > 0.35 \rightarrow y = -1$

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.3 \rightarrow y = 1$   
 $x > 0.3 \rightarrow y = -1$

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \rightarrow y = 1$   
 $x > 0.35 \rightarrow y = -1$

# Example: Building Base Classifiers (cont.)

Decision stump classifier

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$   
 $x > 0.75 \rightarrow y = 1$

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$x \leq 0.75 \rightarrow y = -1$   
 $x > 0.75 \rightarrow y = 1$

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$   
 $x > 0.75 \rightarrow y = 1$

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$   
 $x > 0.75 \rightarrow y = 1$

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$x \leq 0.05 \rightarrow y = 1$   
 $x > 0.05 \rightarrow y = 1$



# Example: Bagging Ensemble

- With bagging, **accuracy: 100%**  
(0 training error)

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Classification  
by individual  
classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Classification  
by ensemble

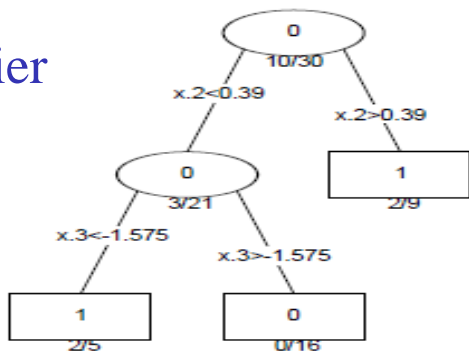
True class

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

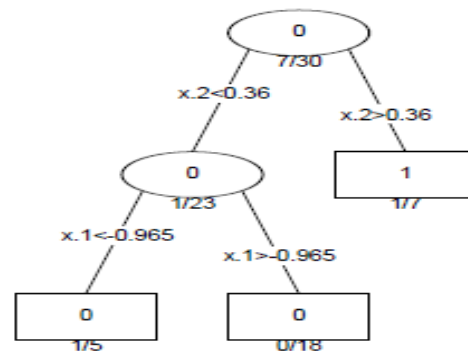
# Example:

## Bagging Classifier with Decision Trees

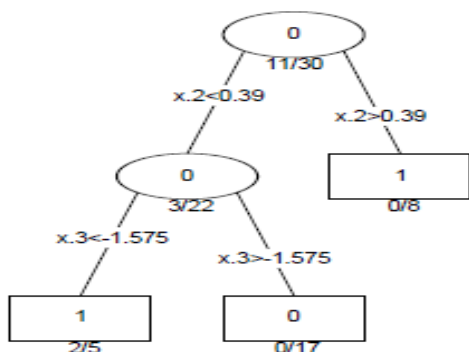
Original Tree



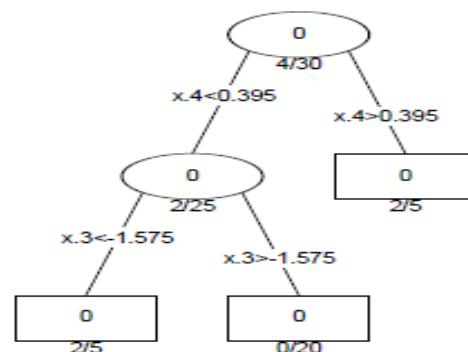
Bootstrap Tree 1



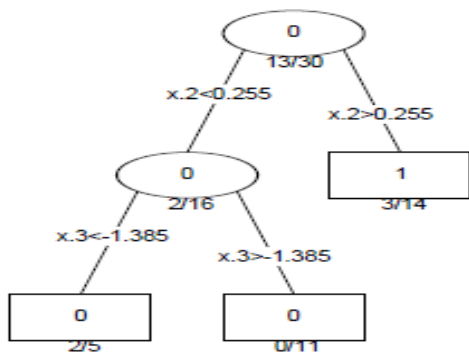
Bootstrap Tree 2



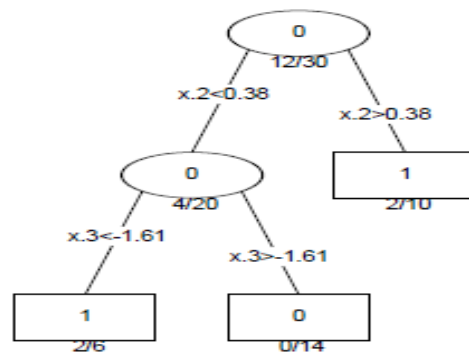
Bootstrap Tree 3



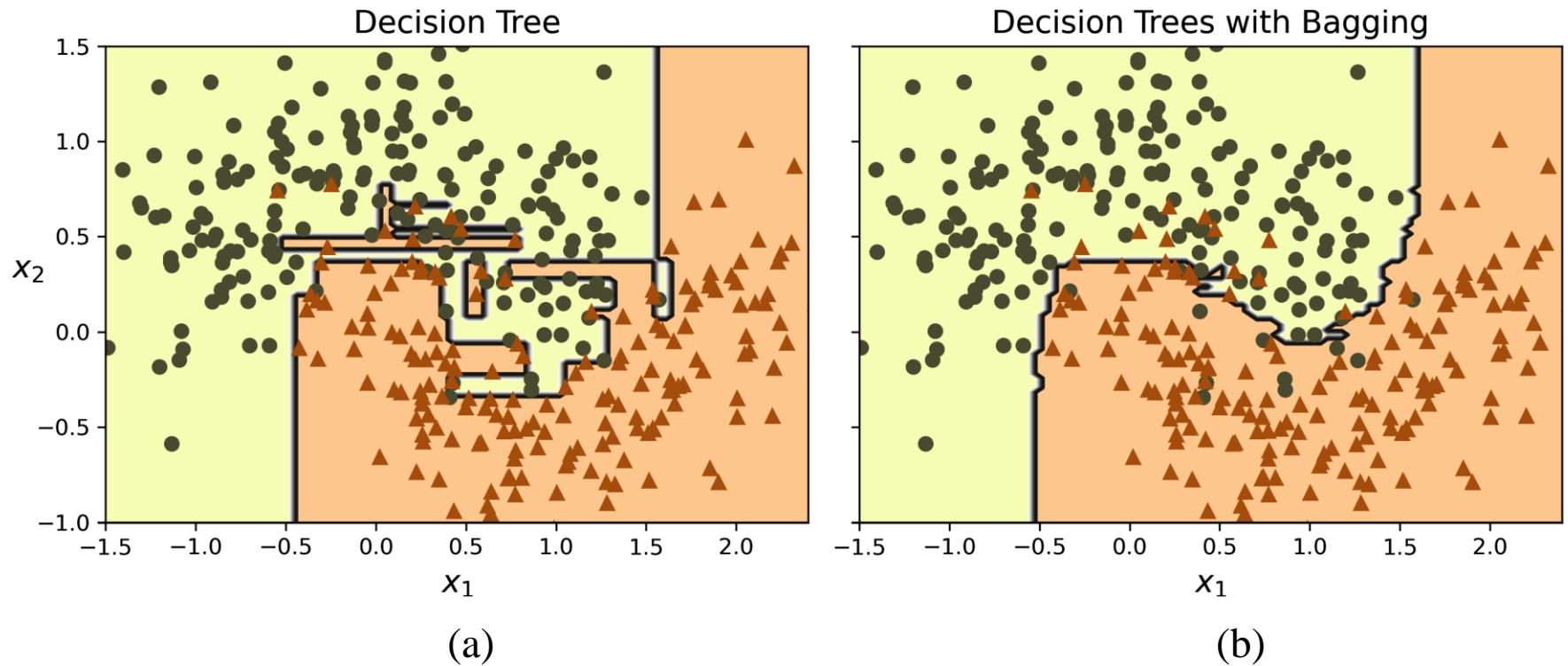
Bootstrap Tree 4



Bootstrap Tree 5



# Performance of a Single Model vs Ensemble Model

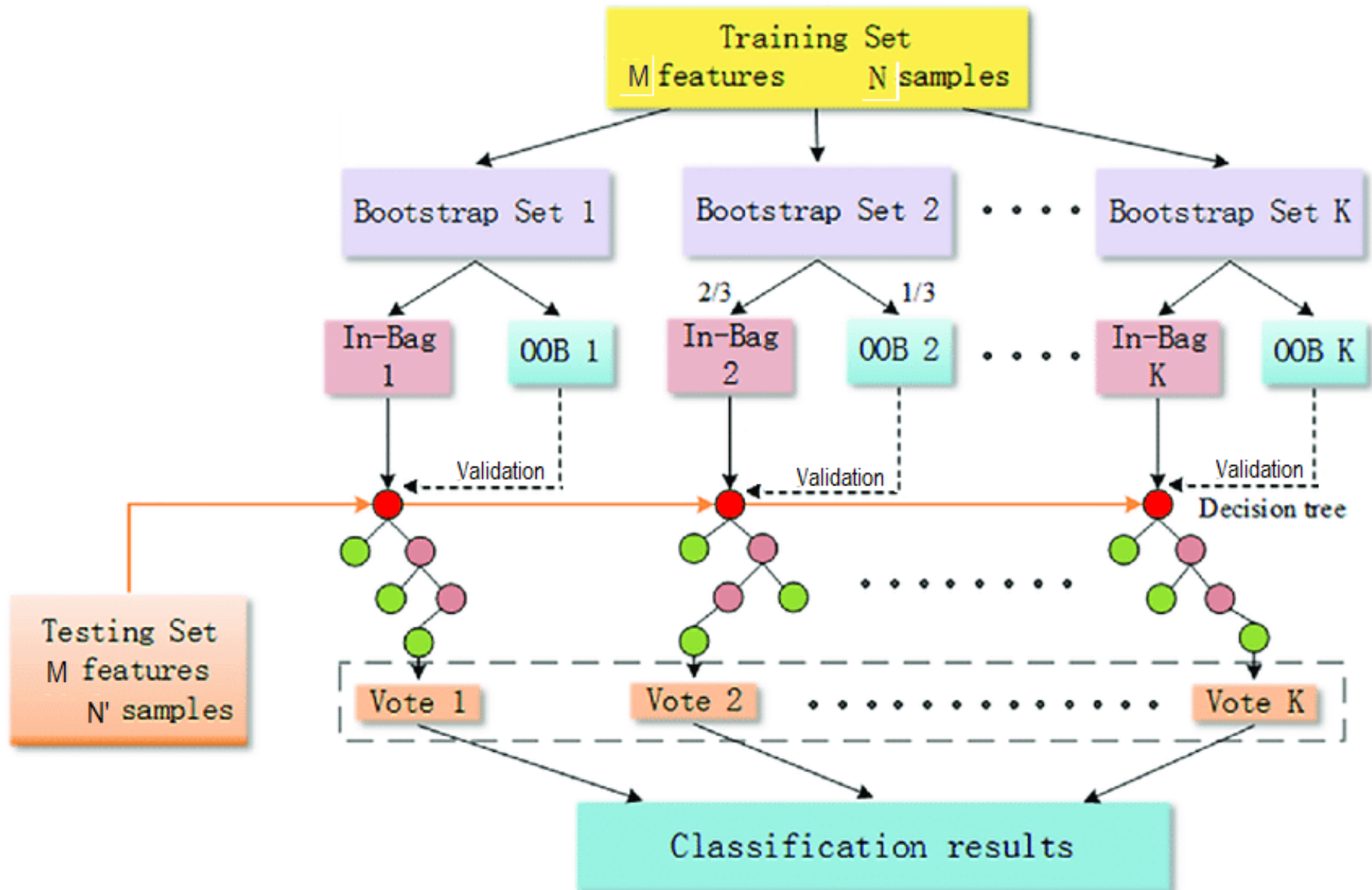


**Figure.** (a) A single decision tree versus (b) a bagging ensemble of 500 trees

# Out-of-Bag (OOB) Evaluation

- **Bagging** uses *sampling with replacement* from original training examples to create a training set for each base classifier or regressor.
- It is shown mathematically that **only about 63% of the training instances are sampled on average** for each predictor.
- The **remaining 37%** of the training instances that are not sampled are called *out-of-bag (OOB) instances*
- **A bagging ensemble can be evaluated using OOB instances**, without the need for a separate validation set.
- ***Out-of-bag (OOB) error*** (also called *out-of-bag estimate*) is a method of measuring the prediction error of machine learning models utilizing bootstrap aggregating such as random forests and boosted decision trees.

# Bagging Ensemble and Validation with OOB



# Characteristics of Bagging

- Bagging typically results in **improved accuracy** over prediction using a single classifier.
- However, it can be **difficult to interpret the resulting model**, e.g., a single tree vs. an ensemble classifier of multiple trees
- Although difficult to interpret, one can obtain an overall summary of the importance of each predictor using the Gini index for bagging classification trees, the RSS (Residual Sum of Squares) for bagging regression trees, or RSS measures the level of variance in the error term (residuals) of a regression model

# Outline

- Introduction
- Types of Constructing Ensembles
- Ensemble Learning Methods
  - Bagging
  - 👉 **Random Forests**
  - Boosting
  - Gradient Boosting

# Random Forests

- **Random forests** (Breiman, 2001), or **random decision forests** construct ensembles of decision trees by not only manipulating training instances (**by using bootstrap samples** similar to **bagging**), but also the input attributes (**by using different subsets of attributes** at every internal node)
- A key difference of random forests from bagging is that at every internal node of a tree, the best splitting criterion is chosen among a small set of randomly selected attributes.
  - Keeping all training instances but sampling features is called **subspace sampling** (or **random subspaces**) methods.
- Random forests attempt to improve the generalization performance by constructing **an ensemble of *de-correlated* decision trees**.



# Random Forests Algorithm

## ■ Training a random forests classifier (model)

1. **Construct a bootstrap sample  $D_i$**  of the training set by randomly sampling examples with replacement from the original training set  $D$ .
2. Use  $D_i$  to learn a decision tree  $T_i$  as follows.
  - **At every internal node of  $T_i$ , randomly sample a set of  $p$  features and**
  - **choose a test feature from this subset** that shows the maximum reduction in an impurity measure for splitting.
3. Repeat the step 2 until every leaf is pure or satisfies the terminal criteria.

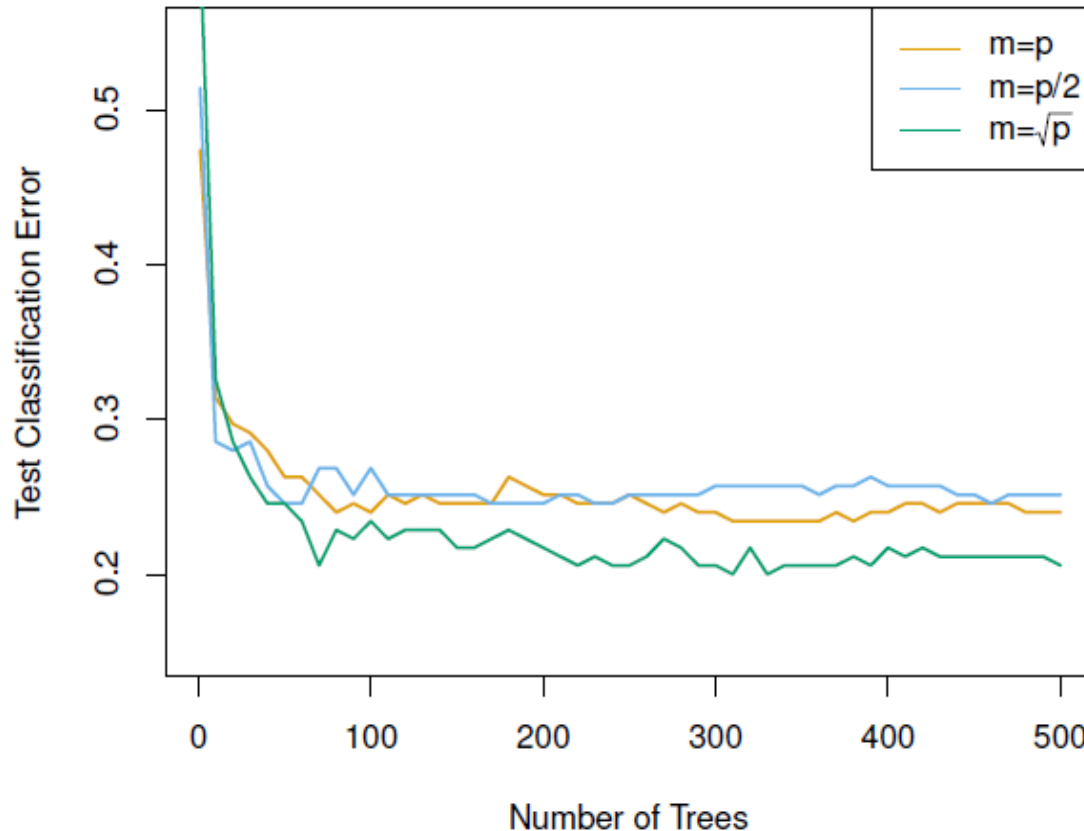
## ■ Classification by the random forests model

- Once an ensemble of base trees has been constructed, their output using a **majority vote** on a test example is used as the final prediction of the random forests.

# Hyper-parameters for Random Forests

- A random forest classifier uses all the hyper-parameters of a decision tree classifier (to control how trees are grown), plus all the hyper-parameters of a bagging classifier (to control the ensemble itself).
- In addition, **the number of attributes to be selected at every node,  $m$** , is another hyper-parameter for random forests
  - A small value of  $m$  can reduce the correlation among the classifiers but may also reduce their strength.
  - A large value of  $m$  can improve their strength but may result in correlated trees similar to bagging.
  - $\sqrt{p}$  or  $\log_2 p + 1$  are usually recommended for  $m$ , where  $p$  is the number of descriptive attributes.
    - When  $m=p$ , bagging

# Performance Comparison with Different $m$

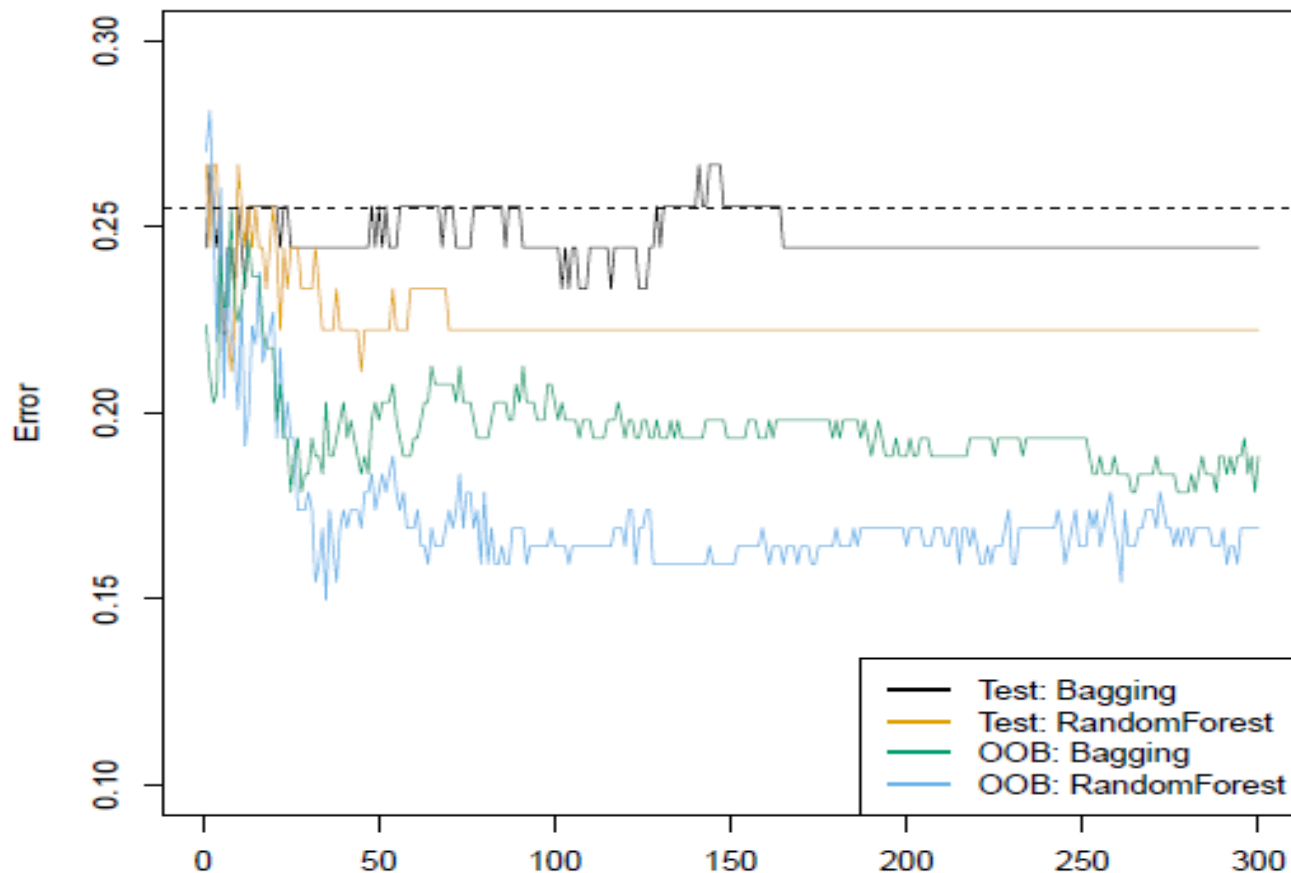


- While a single classification tree has an error rate of around 45.7%, random forests led to much performance improvement
- Random forests ( $m = \sqrt{p}$  or  $p/2$ ) give a small improvement in test error over bagging ( $m = p$ ).

**Figure.** Performance of random forest classifiers with different  $m$ s, where  $p$ : the number of descriptive features  $m$ : the number of features available for splitting at each interior tree node

# Performance Comparison

- Performance comparison of the test error and OOB error (validation error) of Bagging and Random Forests



← The test error resulting from a single classification tree

Random forest reduces both test error and OOB error over bagging

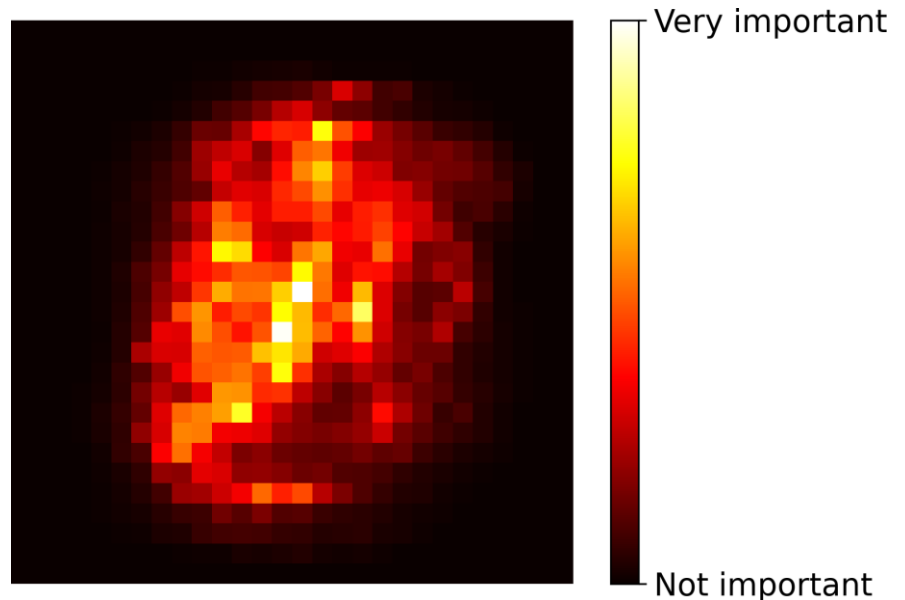
Number of Trees = Number of bootstrapped training sets

# Characteristics of Random Forests

- **The base classifiers are the *lack of correlation*** among their model parameters and test predictions because of the two principles following:
  - 1) They use independently sampled data set
  - 2) They use a randomly selected subset of attributes for choosing a splitting criterion at every internal node
- The predictions of an ensemble of strong and decorrelated decision trees make random forests quite **robust to overfitting**.
- Because the base classifier considers only small subset of attributes at every inter node, random forests are **computationally fast and robust even in high-dimensional setting**

# Characteristics of Random Forests (cont.)

- Random forests are not only valuable for making accurate predictions, but they can also be **a very useful tool for feature evaluation through feature importance.**
  - For each feature, the forest aggregates the decrease in impurity over all trees and takes an average. This average decrease in impurity due to a particular feature can be then considered as the importance of that feature.
- **Example:** MNIST pixel importance (according to a random forest classifier)



# Disadvantage of Random Forests

- Random Forest creates a lot of trees and merges their outputs. So, they're **computationally intensive and slower**.
- Random Forests are often viewed as a black-box method because, while it's easy to understand an individual tree, **it's challenging to grasp the logic of hundreds or thousands of trees combined**.
- **It's important to adjust parameters** like the number of trees, the number of features selected, depth of the tree, etc., **for optimal performance**.

# Outline

- Introduction
- Types of Constructing Ensembles
- Ensemble Learning Methods
  - Bagging
  - Random Forest
  - ☞ **Boosting**
  - Guardian Boosting



# Boosting

- **Boosting** (originally called **hypothesis boosting**) is another standard ensemble technique that combines the outputs of many “weak” models to produce a strong model.
- The **general idea** of most boosting methods is to **train models sequentially**, each trying to correct its predecessor.
- **Boosting** is **an iterative procedure** used to adaptively change the distribution of training examples for learning base classifiers so that they **increasingly focus on examples that are hard to classify**.

# Illustrative Example of Boosting

- Initially, all  $n$  training examples are assigned equal weights,  $1/n$ , so that they are equally likely to be chosen for training

Original Data	1	2	3	4	5	6	7	8	9	10
---------------	---	---	---	---	---	---	---	---	---	----

- Examples to make a bootstrap training set are drawn according to the sampling distribution of the training examples

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3

- A classifier is built from the bootstrapping sample and then used to classify all the examples in the original data set
- The weights of all the training examples are updated at the end of the boosting round.
  - Examples that are classified incorrectly will have their weights increased
  - Examples that are classified correctly will have their weights decreased

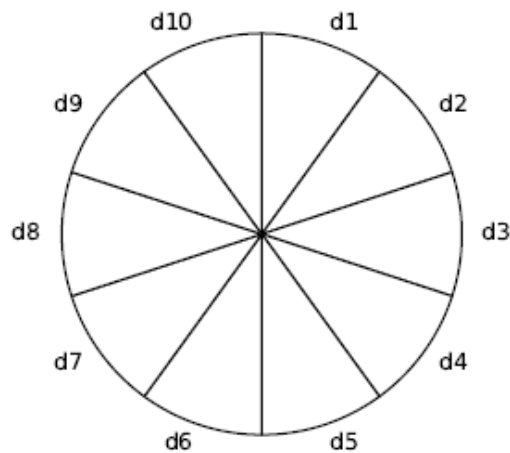
# Illustrative Example of Boosting(cont.)

- So, boosting focuses on examples that are difficult to classify in subsequent iterations
- **Example:** Assume that Example 4 is hard to classify
  - Its weight is increased; therefore it is more likely to be chosen again in subsequent rounds
  - As the boosting rounds proceed, examples that are the hardest to classify tend to become even more prevalent

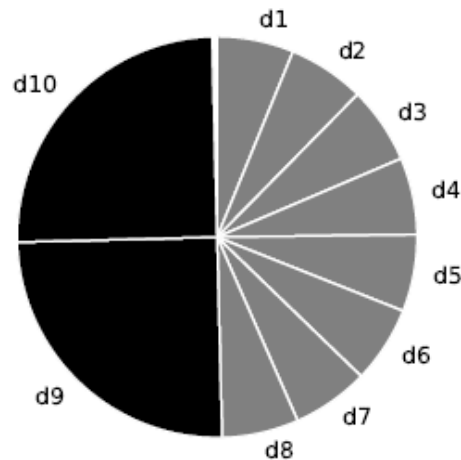
Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

# Weighted Dataset for Boosting

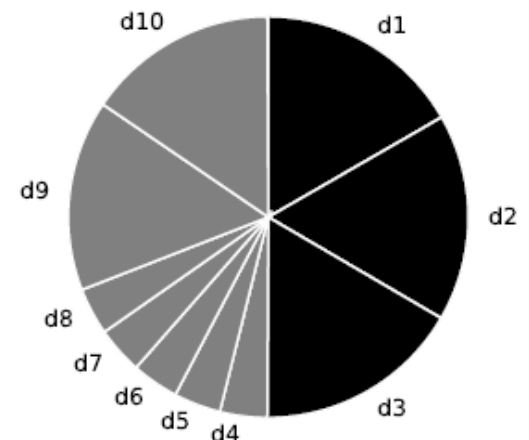
- The updated weight of each training example is used as a distribution over which the dataset is sampled to create a **replicated training set**, where the replication of an example is proportional to its weight



(c) Distribution 0



(d) Distribution 1



(e) Distribution 2

**Figure:** A representation of the changing weights used to generate sample datasets for the first iterations of the boosting process.

# Boosting Algorithms

- Several implementations of boosting
- **Boosting algorithms differ in terms of**
  - (1) how the weights of the training examples are updated at the end of each boosting round, and
  - (2) how the predictions made by each classifier are combined
- The most popular boosting methods are:
  - **AdaBoost**
  - **Gradient Boosting**

# Simpler Boosting Algorithm - AdaBoost

- **AdaBoost** works in a similar way with bagging. However, unlike bagging, boosting assigns a weight to each training example (weighted dataset) and may adaptively change the weight at the end of each boosting round.
- **Initialization**
  - Given a labeled dataset  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$  where  $y_i \in \{-1, 1\}$  (assuming a binary classification problem)
  - Each example has an associated **weight**  $w_i \geq 0$ . Initially set to  $w_i = \frac{1}{n}$  where  $n$  is the number of examples in the dataset

# AdaBoost (Cont.)

- During each **training iteration**  $t = 1..T$ , the algorithm:
  1. induces a model  $t$
  2. calculates the **total error**,  $\epsilon_t$ , by summing the weights of the training examples  $i$  for which the predictions made by the model are incorrect

$$\epsilon_t \leftarrow \sum_{i: h_t(x_i) \neq y_i} w_i$$

3. increases the weights for examples misclassified using:

$$w_i \leftarrow w_i \times \left( \frac{1}{2 \times \epsilon_t} \right)$$

decreases the weights for the examples correctly classified:

$$w_i \leftarrow w_i \times \left( \frac{1}{2 \times (1 - \epsilon_t)} \right)$$

4. calculates a **confidence factor**  $\alpha_t$  (the model weight) for the **model**  $t$  such that  $\alpha_t$  increases as  $\epsilon_t$  decreases:

$$\alpha_t \leftarrow \frac{1}{2} \times \log_e \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$$

## AdaBoost (cont.)

- Once a set of the weak models have been created, the ensemble makes predictions using a **weighted aggregate of the predictions** made by the individual models.

$$H(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x))$$

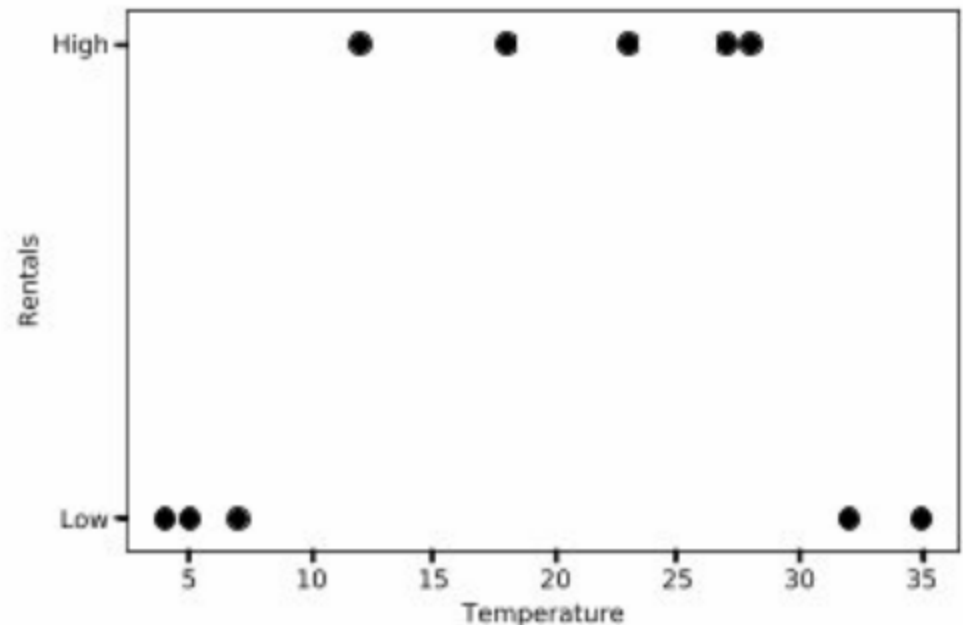
- The weights used in this aggregation are simply the confidence factors  $\alpha_t$  associated with each model  $t$ .
- In this binary classification problem, if the sum is positive, the ensemble predicts +1; otherwise, it predicts -1.



# Example: Simpler Boosting (AdaBoost)

- A problem to predict RENTALS given a single descriptive TEMP, the forecasted temperature for a day.

ID	TEMP	RENTALS
1	4	<i>Low</i>
2	5	<i>Low</i>
3	7	<i>Low</i>
4	12	<i>High</i>
5	18	<i>High</i>
6	23	<i>High</i>
7	27	<i>High</i>
8	28	<i>High</i>
9	32	<i>Low</i>
10	35	<i>Low</i>



**Table :** A simple bicycle demand predictions dataset

**Figure:** The data visualization

## Example (cont.) - Training a Boosting Model

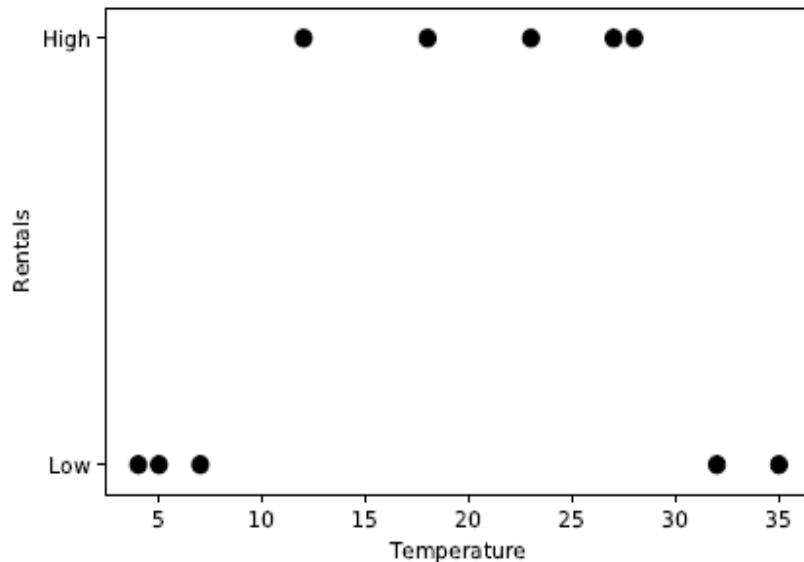
ID	TEMP	RENTALS	Iteration 0			Iteration 1			Iteration 2		
			Dist.	Freq.	$M_0(\mathbf{d})$	Dist.	Freq.	$M_1(\mathbf{d})$	Dist.	Freq.	$M_2(\mathbf{d})$
1	4	Low	0.100	2	Low	0.062	0	High	0.167	2	Low
2	5	Low	0.100	1	Low	0.062	1	High	0.167	1	Low
3	7	Low	0.100	0	Low	0.062	1	High	0.167	3	Low
4	12	High	0.100	1	High	0.062	2	High	0.038	0	Low
5	18	High	0.100	1	High	0.062	0	High	0.038	0	Low
6	23	High	0.100	1	High	0.062	0	High	0.038	0	Low
7	27	High	0.100	1	High	0.062	1	High	0.038	0	Low
8	28	High	0.100	1	High	0.062	1	High	0.038	1	Low
9	32	Low	0.100	2	High	0.250	3	Low	0.154	1	Low
10	35	Low	0.100	0	High	0.250	1	Low	0.154	2	Low

- For each iteration of the boosting process, the columns labeled **Dist.** gives the **weight** in the sampling distribution for each training instance, and the columns labeled **Freq.** give **the number of times a training instance was included** in the training set sampled using the sample distribution.
- The columns labeled  $M_i(\mathbf{d})$  give the **predictions** made by the model trained at iteration  $i$  for each instance in the training dataset.

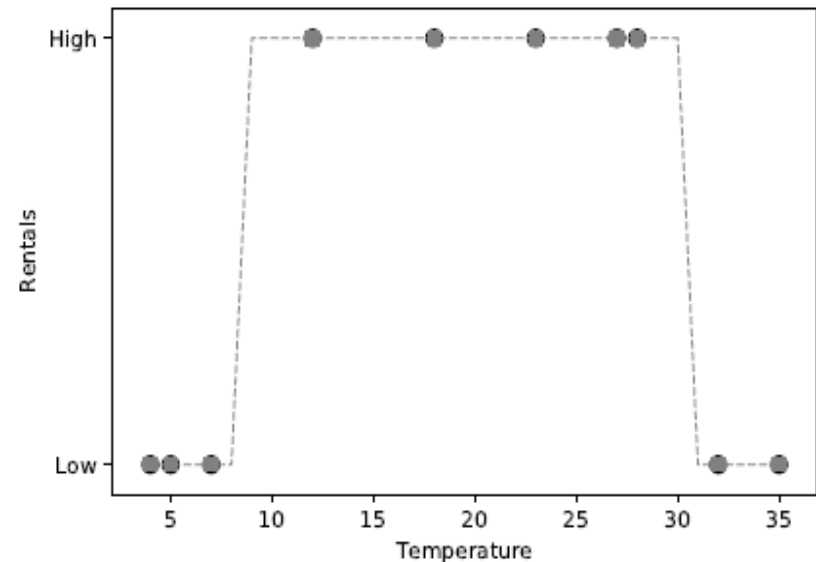
## Example (cont.) - Training a Boosting model

- At the beginning of the boosting process the sampling distribution is initialized so that each instance has **an equal weight of  $\frac{1}{10}$** . A dataset is sampled using this distribution
- **A decision tree is then trained** using the sample
- **The decision tree is used to make predictions for each instance in the complete training set.**
- On the basis of the prediction result,  $\epsilon$  is calculated.  **$\epsilon$  is the sum of the weights of the instances misclassified by the model.** (in the example, d9 and d10.  $\epsilon$  is 0.2)
- The weights of all correctly classified instances are then updated using the equation, e.g., for d1 instance,  $w[1] \leftarrow 0.1 \times \left(\frac{1}{2 \times (1-0.2)}\right) = 0.0625$
- The weights of all incorrectly classified instances are then updated using the equation, e.g., for d9 instance,  $w[9] \leftarrow 0.1 \times \left(\frac{1}{2 \times 0.2}\right) = 0.25$

# Example (cont.) : Final Boosting Model



(a) Training data



(b) The final ensemble model,  $M$

**Figure :** (a) A plot of the bike rental dataset. (b) An illustration of the final ensemble model trained using the boosting algorithm.

# Characteristics of AdaBoost

- The algorithm gives more importance to the examples that are harder to classify, forcing the weak learner to focus more on them in the next round.
- If a weak classifier does better than random guessing, its output is weighted more in the final decision. If it does worse than random, its output is weighted in the opposite direction and may not be considered.
- **The number of boosting rounds  $T$  is a hyper-parameter** that can be adjusted based on the problem and the performance on validation data.
- AdaBoost can be combined with any learning algorithm, but **decision trees (often just decision stumps, which are one-level trees)** are a common choice.

# Drawback of Boosting

- Because boosting is a sequential learning technique – each predictor can only be trained after the previous predictor has been trained and evaluated, **training cannot be parallelized.**
- As a result, **boosting does not scale as well as bagging.**

# Outline

- Introduction
- Types of Constructing Ensembles
- Ensemble Learning Methods
  - Bagging
  - Random Forest
  - Boosting
  - 👉 **Gradient Boosting**

# Gradient Boosting

- **Gradient boosting** is a more recently developed, and very effective, algorithm for training ensemble models using boosting.
- Like other boosting methods, gradient boosting combines weak “learners” into a single strong learner in an iterative fashion. – sequentially adding predictors to an ensemble
- However, instead of tweaking the instance weights at every iteration like AdaBoost does, Gradient boosting tries to make later models specialize in areas that earlier models struggled with. - **Later models are trained to directly correct errors made by earlier models,**



# General Idea of Gradient Boosting

- It is easiest to explain the gradient boosting algorithm in the context of a regression problem (**Gradient Tree Boosting** or **Gradient Boosted Regression Trees**) with a contiguous target
- Given a training dataset  $D$  with  $(\mathbf{d}, t)$ , where  $\mathbf{d}$  is a vector of descriptive features,  $t$  is the target (output) feature, **the goal is** to “teach” a model  $M$  to predict target values of the form  $\hat{t} = M(\mathbf{d})$  by minimizing the *mean squared error*,  $\frac{1}{n} \sum_i (t - \hat{t})^2$  (or simply the difference,  $t - \hat{t}$ ), where  $n$  is the number of samples in  $t$

# General Idea of Gradient Boosting

- At each stage  $i$  of gradient boosting, suppose some imperfect previous model  $\mathbb{M}_{i-1}$ .
- In order to improve  $\mathbb{M}_{i-1}$ , the algorithm **add some new estimator,  $\mathbb{M}_{\Delta i}$** . Thus

$$\mathbb{M}_i(\mathbf{d}) = \mathbb{M}_{i-1}(\mathbf{d}) + \mathbb{M}_{\Delta i}(\mathbf{d}) = t$$

or, equivalently,

$$\mathbb{M}_{\Delta i}(\mathbf{d}) = t - \mathbb{M}_{i-1}(\mathbf{d})$$

- Therefore, gradient boosting will **fit  $\mathbb{M}_{\Delta i}$  to the *residual*,  $t - \mathbb{M}_{i-1}(\mathbf{d})$** .
- As in other boosting variants, each  $\mathbb{M}_i$  attempts to correct the errors of its predecessor  $\mathbb{M}_{i-1}$

# Gradient Boosting Algorithm

- Given a training dataset  $D$  with  $(\mathbf{d}, t)$ , where  $\mathbf{d}$  is a vector of descriptive features,  $t$  is the target (output) feature.
- 1. Build a very simple initial model (i.e., weak learner),  $\mathbb{M}_0$ 
  - e.g., The initial model simply returns the overall average target value from the training set,  $\mathbb{M}_0(\mathbf{d}) = \frac{1}{n} \sum_i t_i$  where  $n$  is the number of samples in  $t$
- 2. Train the first base model for an ensemble at the first iteration

$$\mathbb{M}_1(\mathbf{d}) = \mathbb{M}_0(\mathbf{d}) + \mathbb{M}_{\Delta 1}(\mathbf{d})$$

where  $\mathbb{M}_{\Delta 1}(\mathbf{d})$  is the real model trained to predict the errors made by the initial model  $\mathbb{M}_0$ .  $\mathbb{M}_{\Delta 1}(\mathbf{d}) = t - \mathbb{M}_0(\mathbf{d})$

# Gradient Boosting Algorithm (cont.)

3. At each stage  $i$  ( $1 \leq i \leq m$ ) of the ensemble learning with  $m$  stages, **iteratively adds more and more models to the ensemble** where each model is trained to **predict the errors** made by the previous ones

$$\mathbb{M}_i(\mathbf{d}) = \mathbb{M}_{i-1}(\mathbf{d}) + \mathbb{M}_{\Delta i}(\mathbf{d})$$

4. The **final model** is then a model that makes a initial prediction and adds a number of improvements to this prediction.

$$\hat{\mathbb{M}}_m(\mathbf{d}) = \mathbb{M}_0(\mathbf{d}) + \sum_{i=1}^m \mathbb{M}_{\Delta i}(\mathbf{d})$$

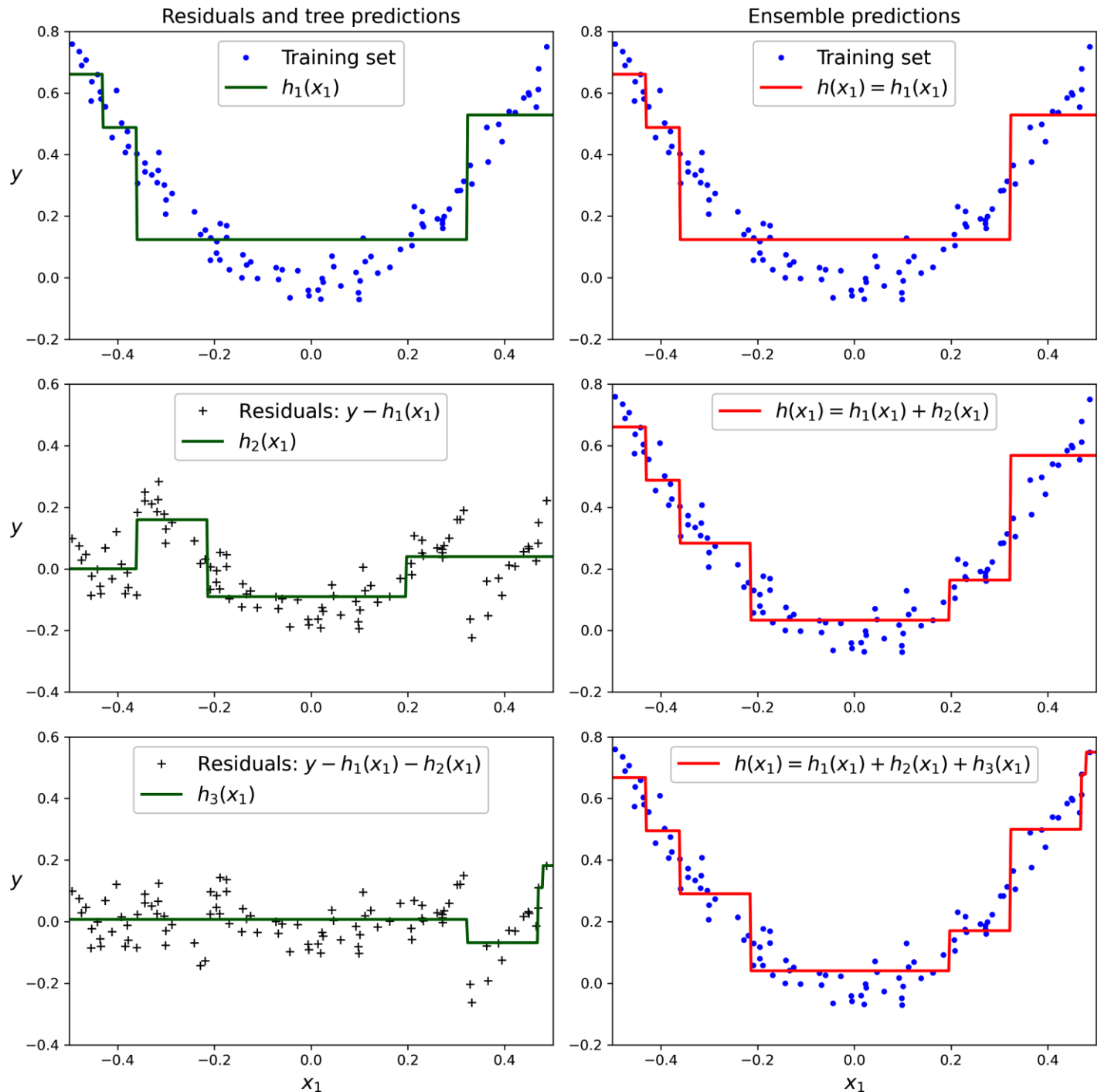
E.g., The model trained by gradient boosting with 4 stages

$$\begin{aligned}\mathbb{M}_4(\mathbf{d}) &= \mathbb{M}_3(\mathbf{d}) + \mathbb{M}_{\Delta 4}(\mathbf{d}) \\ &= (\mathbb{M}_2(\mathbf{d}) + \mathbb{M}_{\Delta 3}(\mathbf{d})) + \mathbb{M}_{\Delta 4}(\mathbf{d}) \\ &= ((\mathbb{M}_1(\mathbf{d}) + \mathbb{M}_{\Delta 2}(\mathbf{d})) + \mathbb{M}_{\Delta 3}(\mathbf{d})) + \mathbb{M}_{\Delta 4}(\mathbf{d}) \\ &= \mathbb{M}_0(\mathbf{d}) + \mathbb{M}_{\Delta 1}(\mathbf{d}) + \mathbb{M}_{\Delta 2}(\mathbf{d}) + \mathbb{M}_{\Delta 3}(\mathbf{d}) + \mathbb{M}_{\Delta 4}(\mathbf{d})\end{aligned}$$

# Characteristics of Gradient Boosting

- The idea of the gradient boosting which fits  $\mathbb{M}_{\Delta i}(\mathbf{d})$  to *residual*,  $t - \mathbb{M}_{i-1}(\mathbf{d})$ , **can be generalized to other loss function, and to classification and ranking problem.**
- The method tries to find an approximation  $\widehat{\mathbb{M}}_m(\mathbf{d})$  that minimizes the average value of the loss function on the training set.
- It does by starting with a model, consisting of a constant function  $\mathbb{M}_0(\mathbf{d})$ , and incrementally expands it in a greedy fashion.

# Gradient Boosting: Illustrative Example

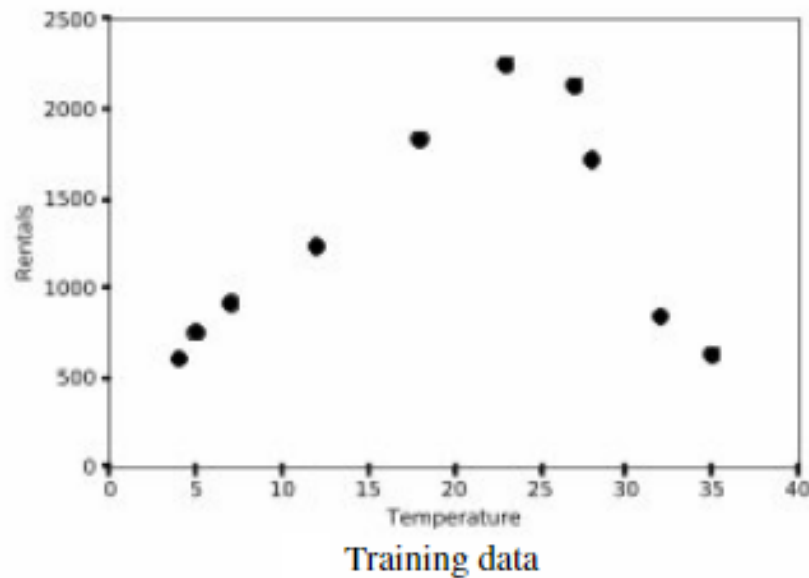


# Example

- A problem to predict the expected rental demand on the bicycle of the forecasted temperature for a day.

ID	TEMP	RENTALS
1	4	602
2	5	750
3	7	913
4	12	1229
5	18	1827
6	23	2246
7	27	2127
8	28	1714
9	32	838
10	35	625

**Table:** A simple bicycle demand predictions dataset

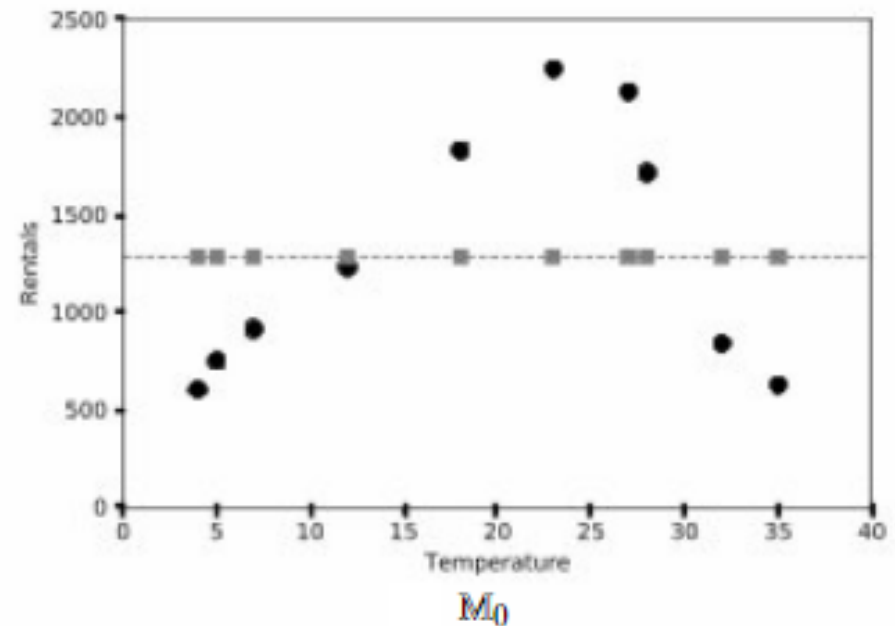


**Figure:** The data visualization

## Example (cont.)

- The initial model  $M_0$  predicts a rental demand of 1287.1 bicycles with the average value for the target feature

ID	TEMP	RENTALS	$M_0(d)$
1	4	602	1287.1
2	5	750	1287.1
3	7	913	1287.1
4	12	1229	1287.1
5	18	1827	1287.1
6	23	2246	1287.1
7	27	2127	1287.1
8	28	1714	1287.1
9	32	838	1287.1
10	35	625	1287.1





## Example (cont.) : First Round of Gradient Boosting

1. Calculate the first set of errors,  $t - \mathbb{M}_0$
2. The first base model for this ensemble,  $\mathbb{M}_{\Delta 1}$  is trained to predict these errors on the basis of the descriptive features in the training set.
  - **Example:** using a simple one-level decision tree, for input temperatures less than or equal to 12 degrees, a correction value of -460.9 value is predicted. For temperature above this threshold, a correction value of 691.4 is predicted.
3. These corrections ( $\mathbb{M}_{\Delta 1}$ ) are combined with the  $\mathbb{M}_0$  predictions to give the ensemble predictions after the first iteration,  $\mathbb{M}_1$

ID	TEMP	RENTALS	$\mathbb{M}_0(\mathbf{d})$	$t - \mathbb{M}_0(\mathbf{d})$	$\mathbb{M}_{\Delta 1}(\mathbf{d})$	$\mathbb{M}_1(\mathbf{d})$
1	4	602	1 287.1	-685.1	-460.9	826.2
2	5	750	1 287.1	-537.1	-460.9	826.2
3	7	913	1 287.1	-374.1	-460.9	826.2
4	12	1229	1 287.1	-58.1	-460.9	826.2
5	18	1827	1 287.1	539.9	691.4	1 978.5
6	23	2246	1 287.1	958.9	691.4	1 978.5
7	27	2127	1 287.1	839.9	691.4	1 978.5
8	28	1714	1 287.1	426.9	691.4	1 978.5
9	32	838	1 287.1	-449.1	-460.9	826.2
10	35	625	1 287.1	-662.1	-460.9	826.2

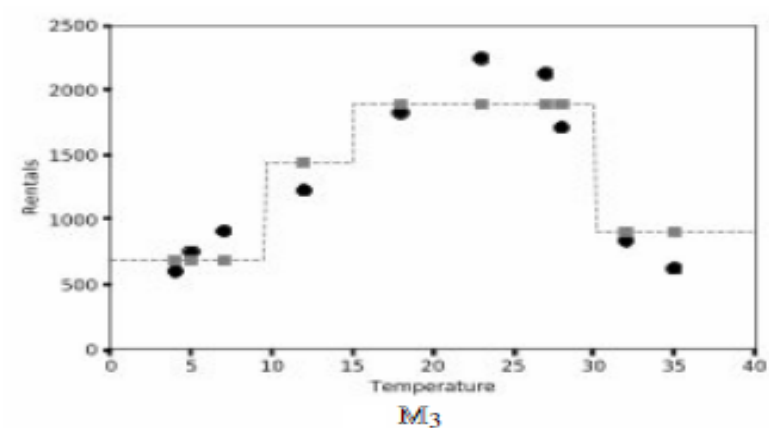
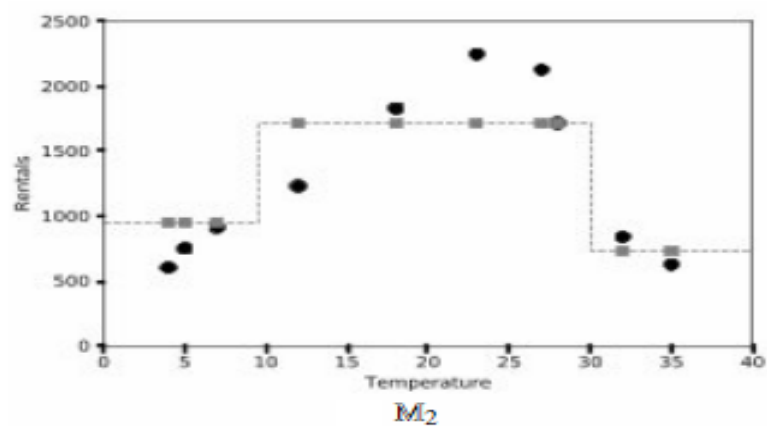
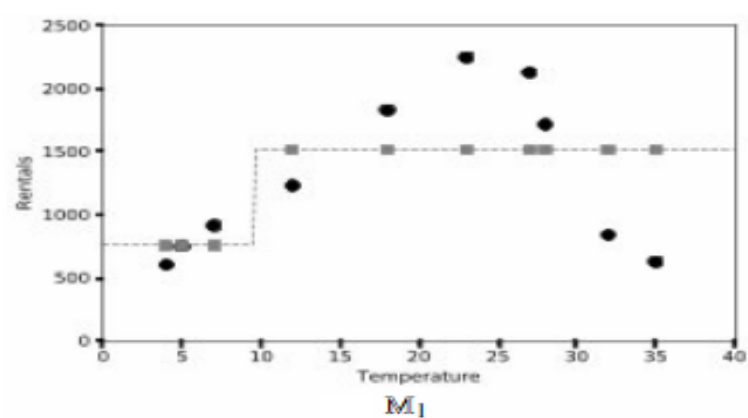
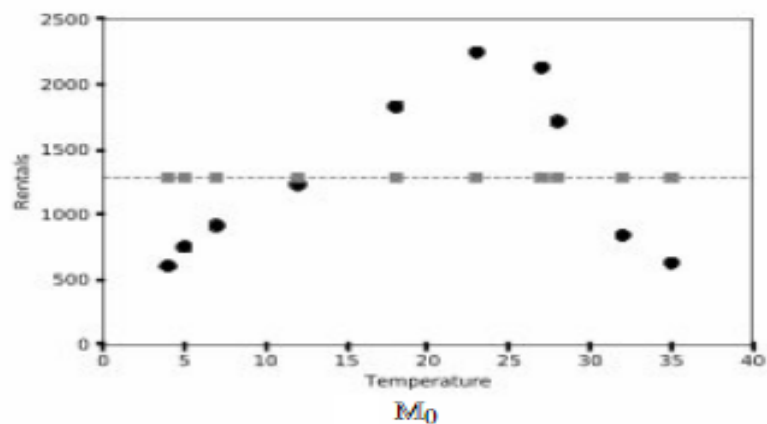
## Example (cont.) : Second Round

1. A new set of errors  $t - \mathbb{M}_1$  are then calculated by comparing the  $\mathbb{M}_1$  predictions to the target feature values.
2. A new model,  $\mathbb{M}_{\Delta 2}$ , is trained to predict these errors on the basis of the original descriptive feature values.
3. The outputs of this model are combined with the  $\mathbb{M}_1$  predictions to give the full model for this step in the ensemble building process,  $\mathbb{M}_2$

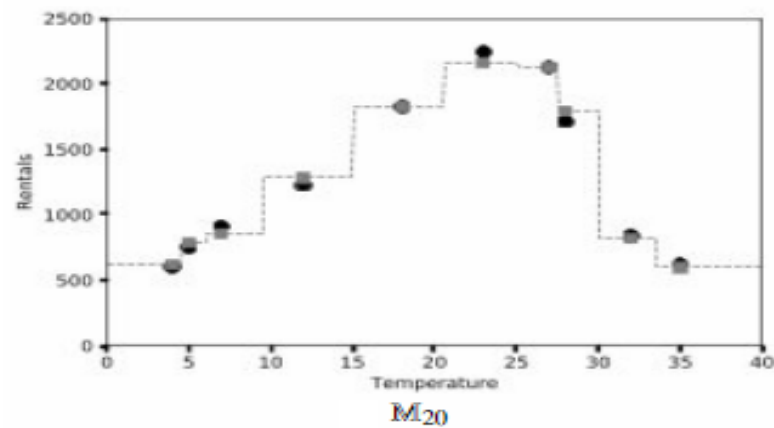
ID	TEMP	RENTALS	$\mathbb{M}_0(\mathbf{d})$	$t - \mathbb{M}_0(\mathbf{d})$	$\mathbb{M}_{\Delta 1}(\mathbf{d})$	$\mathbb{M}_1(\mathbf{d})$	$t - \mathbb{M}_1(\mathbf{d})$	$\mathbb{M}_{\Delta 2}(\mathbf{d})$	$\mathbb{M}_2(\mathbf{d})$
1	4	602	1 287.1	-685.1	-460.9	826.2	-224.2	-167.2	659.0
2	5	750	1 287.1	-537.1	-460.9	826.2	-76.2	-167.2	659.0
3	7	913	1 287.1	-374.1	-460.9	826.2	86.8	71.6	897.8
4	12	1229	1 287.1	-58.1	-460.9	826.2	402.8	71.6	897.8
5	18	1827	1 287.1	539.9	691.4	1 978.5	-151.5	71.6	2 050.1
6	23	2246	1 287.1	958.9	691.4	1 978.5	267.5	71.6	2 050.1
7	27	2127	1 287.1	839.9	691.4	1 978.5	148.5	71.6	2 050.1
8	28	1714	1 287.1	426.9	691.4	1 978.5	-264.5	71.6	2 050.1
9	32	838	1 287.1	-449.1	-460.9	826.2	11.8	71.6	897.8
10	35	625	1 287.1	-662.1	-460.9	826.2	-201.2	-167.2	659.0

## Example (cont.) : Remaining Rounds

- To complete the process of building the ensemble model, the algorithm continues to iteratively calculate errors and train new models to predict these errors, which are added as correction terms to the previous model output.



...



# Characteristics of Gradient Boosting

- **Gradient boosting method does not have an explicit aggregation step**, as do the bagging and boosting algorithm described previously (in which individual model predictions were combined through averaging or voting)
- Instead, **the final model trained provides the overall output of the ensemble** as it implicitly combines the outputs of all models trained.

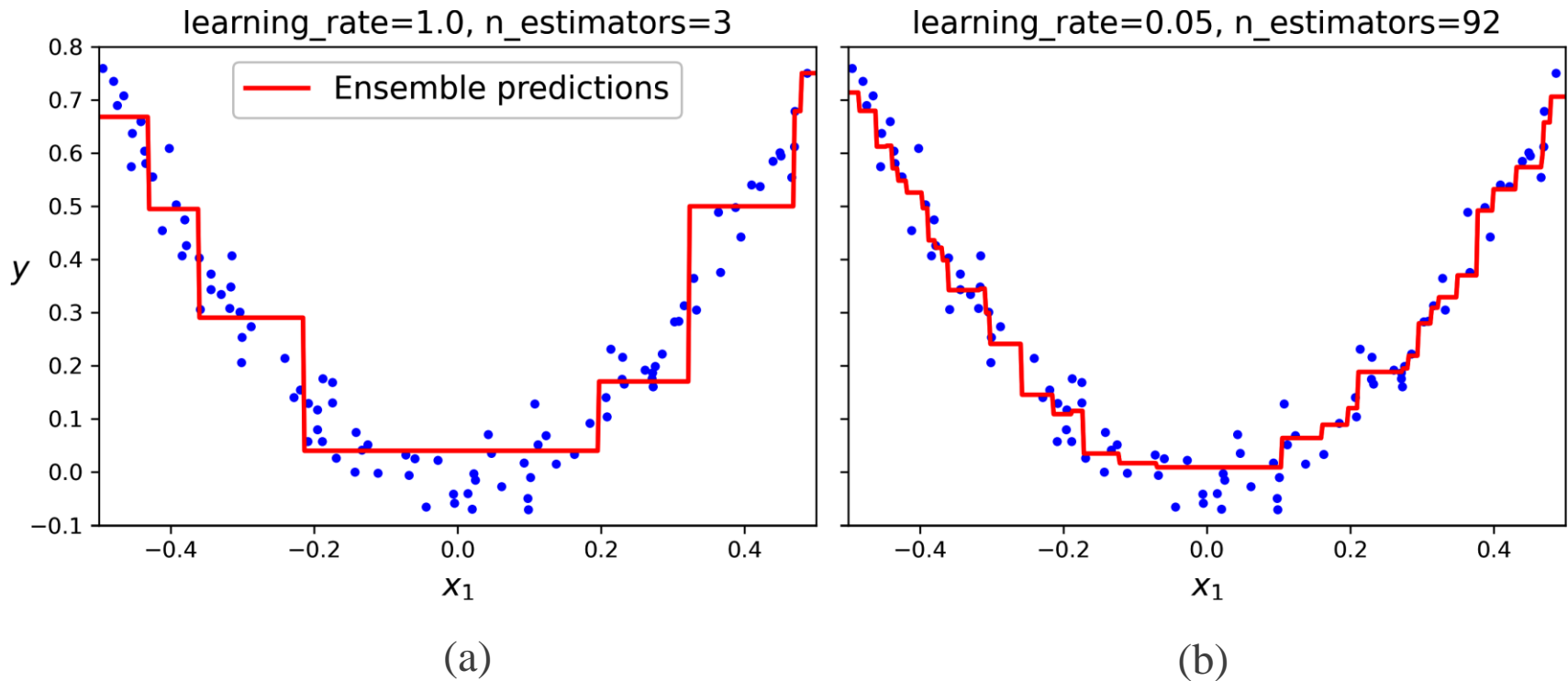
# Variation of Simple Gradient Boosting

- The simple version of gradient boosting can be extended in many way.
- **Variation 1:** a learning rate parameter,  $\alpha$  can be added to the Equation to scale the contribution of each tree.

$$\mathbb{M}_i(\mathbf{d}) = \mathbb{M}_{i-1}(\mathbf{d}) + \alpha \times \mathbb{M}_{\Delta i}(\mathbf{d})$$

- If the learning rate  $\alpha$  is a low value (e.g., 0.05), more trees are needed in the ensemble to fit the train set.
- **Variation 2:** Another way to make models robust to outliers is to train the model to predict **only the sign of the errors** of the previous predictions rather than the magnitudes of the errors.

# Gradient Boosting: Underfit, Just Right, and Overfit



**Figure.** Gradient Boosting ensemble with (a) not enough predictors and (b) just enough

- If we add more trees, the ensemble would start to overfit the training set.

# Summary

- Introduction
- Types of Constructing Ensembles
- Ensemble Learning Methods
  - Bagging
  - Random Forests
  - Boosting
  - Gradient Boosting



# Summary

- **Which approach should we use?**
  - Bagging is simpler to implement and parallelize than boosting and, so, may be better with respect to ease of use and training time.
- **Empirical results indicate:**
  - boosted decision tree ensembles were the best performing model of those tested for datasets containing up to 4,000 descriptive features.
  - random forest ensembles (based on bagging) performed better for datasets containing more than 4,000 features.