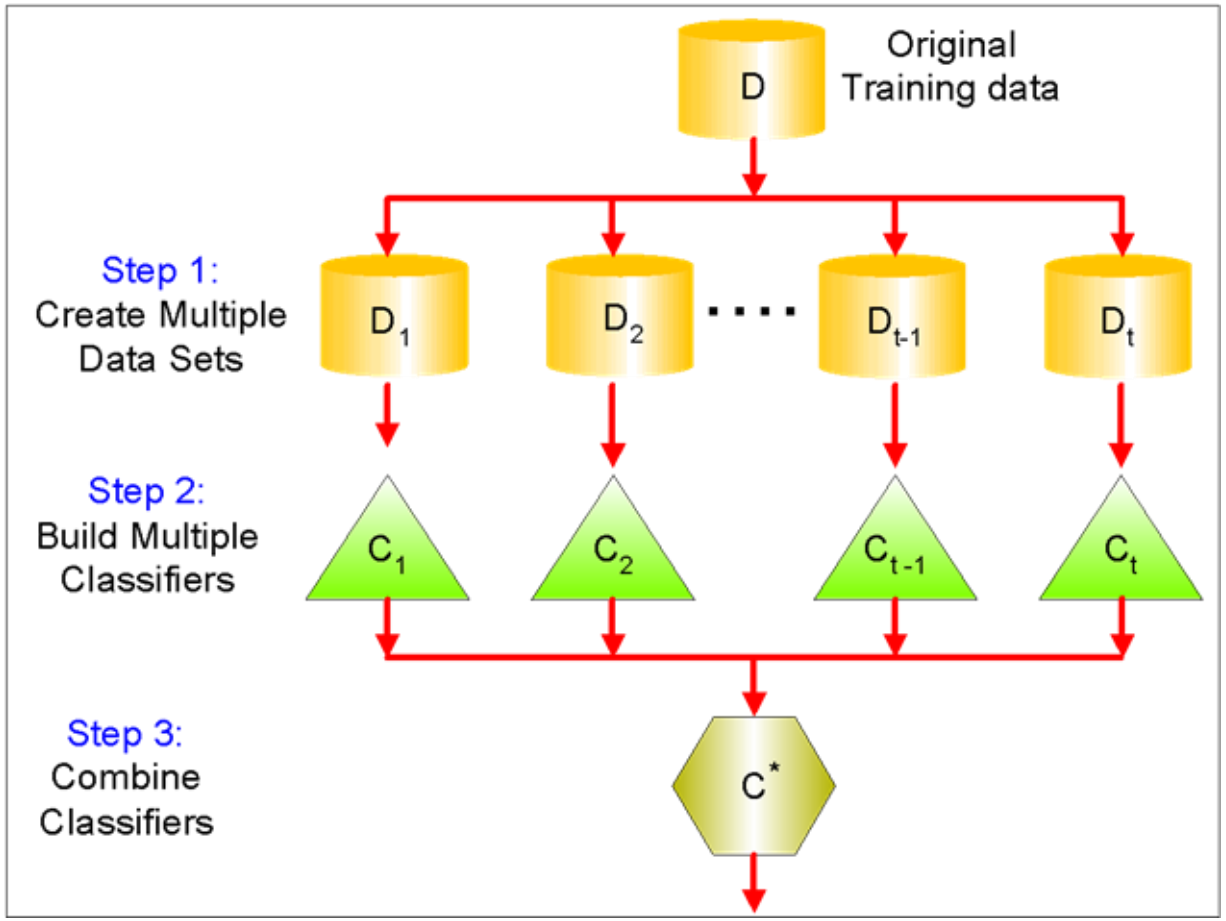
Ensemble Methods



UNIVERSITÀ DI PISA

Ensemble Methods

- Improves the accuracy by **aggregating the predictions** of multiple classifiers.
- Construct a set of **base classifiers** from the training data.
- Predict class label of test records by combining the predictions made by multiple classifiers.



Back to Machine Learning

It will exploit *Wisdom of crowd* ideas for specific tasks

- By combining classifier predictions and
- aims to combine independent and diverse classifiers.

But it will use labelled training data

- to identify the **expert** classifiers in the pool;
- to identify **complementary** classifiers;
- to indicate how to the best **combine** them.

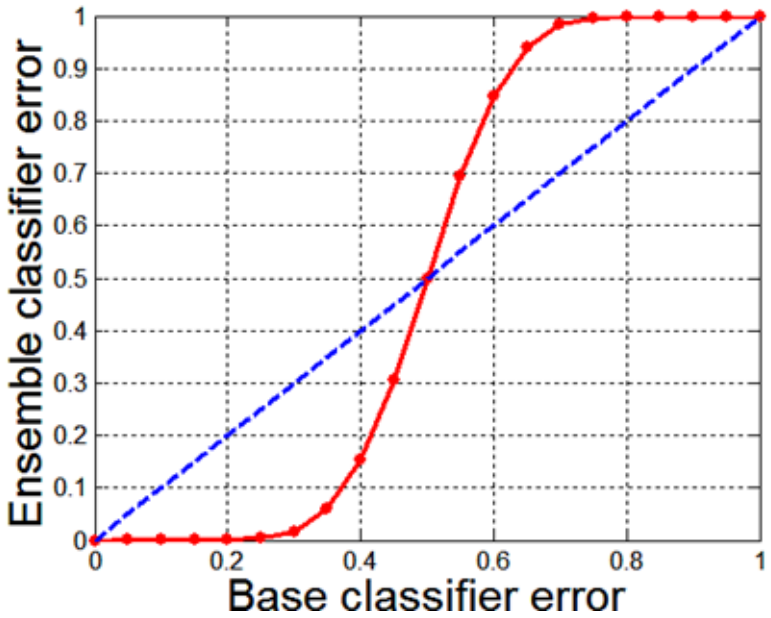
Why Ensemble Methods work?

The example illustrates that majority voting helps when classifiers are sufficiently accurate and make different mistakes.

Consider the two cases for the 25 classifiers.

1. **Identical classifiers**: voting provides no improvement. If all classifiers always make the same predictions, they all fail on the same examples. Whenever one is wrong, all 25 are wrong. Therefore: $\text{Ensemble error} = e$. This is the diagonal dashed line: for example, a base error of 0.35 gives an ensemble error of 0.35.
2. Independent errors**: voting can correct individual mistakes.** Now suppose each classifier has error probability e=0.35, independently of the others. One classifier being wrong does not imply the others are wrong. The ensemble is wrong **only when at least 13 of the 25 classifiers are wrong**:

$$P(\text{ensemble error}) = \sum_{k=13}^{25} \binom{25}{k} e^k (1-e)^{25-k}.$$

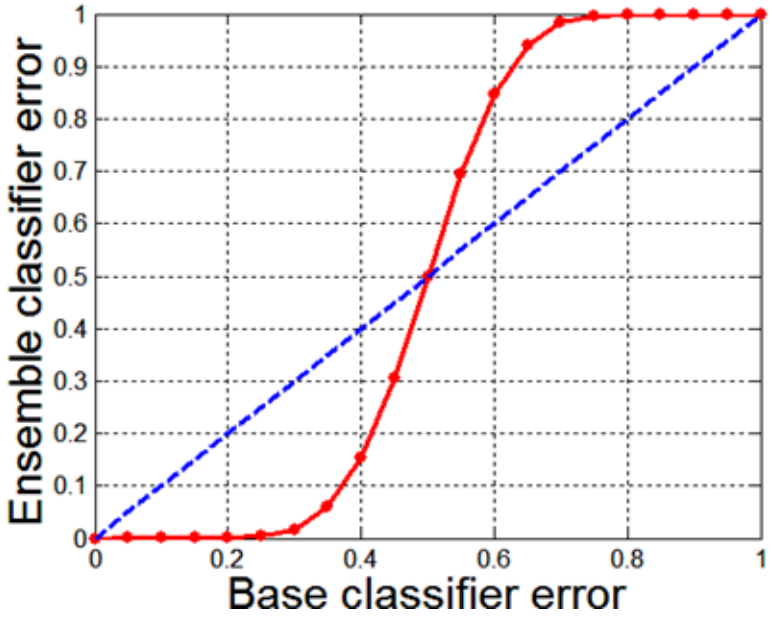


Why Ensemble Methods work?

Why does the solid curve rise above the diagonal when e>0.5?

Voting reinforces whatever each classifier is more likely to do:

Base error	Each classifier is…	Effect of independent majority voting
e<0.5	More likely correct than wrong	Reduces error
e=0.5	Equally likely correct or wrong	Error remains 0.5
e>0.5	More likely wrong than correct	Increases error



Having many classifiers is useful when their individual performance is better than chance and their errors do not all occur together.

Types of Ensemble Methods

- Manipulate data distribution
  - Example: bagging, boosting
- Manipulate input features
  - Example: random forests
- Manipulate class labels
  - Example: error-correcting output coding

Bagging

Bagging (a.k.a. Bootstrap AGGregatING)

- Sampling with replacement
  - Each bootstrap sample has the same size as the original data.
  - Some instances may appear several times in the same training set, while others may be omitted from the training set.

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

- Build classifier on each bootstrap sample

Bagging (a.k.a. Bootstrap AGGregatING)

- Sampling with replacement

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

- Build classifier on each bootstrap sample

Algorithm 5.6 Bagging 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: $C^*(x) = \arg\max_y \sum_i \delta\big(C_i(x) = y\big)$, $\{\delta(\cdot) = 1$ if its argument is true, and 0 otherwise.$\}$

Bagging (a.k.a. Bootstrap AGGregatING)

Suppose the dataset contains n samples, and each bootstrap dataset is built by drawing n times with replacement.
For a particular sample:

- **Probability of being selected in one draw:** $\frac{1}{n}$.
- **Probability of never being selected in the n draws:** $\left(1-\frac{1}{n}\right)^n$
- **Probability of being selected at least once:**

$$\boxed{1-\left(1-\frac{1}{n}\right)^n \underset{n\to\infty}{\to} 1-e^{-1}\approx 0.632.}$$

Thus, a bootstrap dataset contains approximately 63.2% of the original samples as distinct observations. The remaining 36.8% are out-of-bag (OOB) samples. Some selected samples appear multiple times.

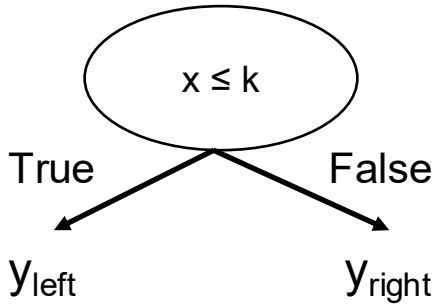
Bagging Example

- Consider 1-dimensional data set:

Original Data:

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

- Classifier is a decision stump
  - Decision rule: x <= k versus x > k
  - Split point k is chosen based on entropy



Bagging Example

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 <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Prediction with decision stump

Bagging Example

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 <= 0.35 ➔ y = 1
x > 0.35 ➔ 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 <= 0.7 ➔ y = 1
x > 0.7 ➔ 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 <= 0.35 ➔ y = 1
x > 0.35 ➔ 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 <= 0.3 ➔ y = 1
x > 0.3 ➔ 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 <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

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 <= 0.75 ➔ y = -1
x > 0.75 ➔ 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 <= 0.75 ➔ y = -1
x > 0.75 ➔ 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 <= 0.75 ➔ y = -1
x > 0.75 ➔ 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 <= 0.75 ➔ y = -1
x > 0.75 ➔ 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 <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Bagging Example

- Summary of Training sets:

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

Bagging Example

- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

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

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

Predicted Class

Bagging Advantages and Limitations

- **Advantages**
- Reduces variance, especially for unstable models such as deep decision trees.
- Improves robustness to changes in the training data.
- Base models can be trained independently and in parallel.
- Out-of-bag samples (samples that are nevere selected) provide an estimate of generalization performance.

Limitations

- Usually provides limited reduction in bias: weak or underfitting base models may remain inadequate.
- Benefits decrease when base models make highly correlated errors.
- Requires more computation and memory than a single model.
- The ensemble is less interpretable than an individual decision tree.

Boosting

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records.
- Initially, all the records are assigned equal weights.
- Unlike bagging, weights may change at the end of each boosting round.

Boosting

- Records that are wrongly classified will have their weights increased.
- Records that are classified correctly will have their weights decreased.

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

- Example 4 is hard to classify
- Its weight is increased; therefore it is more likely to be chosen again in subsequent rounds

AdaBoost

- AdaBoost uses a weighted majority vote: more accurate classifiers receive more voting power.

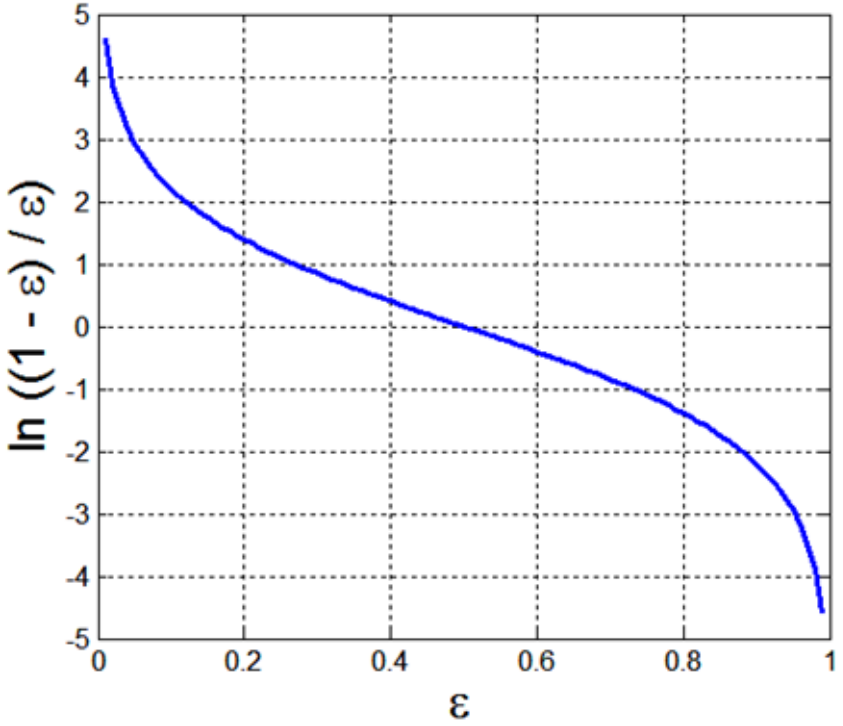
- Base classifiers: C_1, C_2, …, C_T
- Error rate:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance classifier depends on its error rate:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- α_i High positive importance when error is close to 0,
- α_i High negative importance when error is close to 1



Plot of α as a function of training error .

AdaBoost Algorithm

- Weight update associated to x_i during the j boosting round:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} e^{-\alpha_j}, & \text{if } C_j(x_i) = y_i, \\ e^{+\alpha_j}, & \text{if } C_j(x_i) \neq y_i. \end{cases}$$

The normalization factor Z_j ensures that $\sum_i w_i^{(j+1)} = 1$

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to the original value $1/n$ and the resampling procedure is repeated
- Weighted voting classification:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

AdaBoost Algorithm

Algorithm 5.7 AdaBoost Algorithm

1: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$. {Initialize the weights for all n instances.}
2: Let k be the number of boosting rounds.
3: **for** $i = 1$ to k **do**
4: Create training set D_i by sampling (with replacement) from D according to w.
5: Train a base classifier C_i on D_i.
6: Apply C_i to all instances in the original training set, D.
7: $\epsilon_i = \frac{1}{n}\left[\sum_j w_j \, \delta(C_i(x_j) \neq y_j)\right]$ {Calculate the weighted error}
8: **if** $\epsilon_i > 0.5$ **then**
9: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$. {Reset the weights for all n instances.}
10: Go back to Step 4.
11: **end if**
12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13: Update the weight of each instance according to equation (5.88).
14: **end for**
15: $C^*(\mathbf{x}) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y))$.

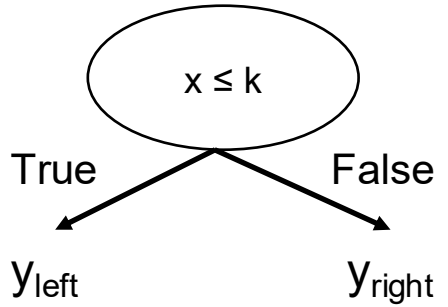
AdaBoost Example

- Consider 1-dimensional data set:

Original Data:

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

- Classifier is a decision stump
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

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

- Weights:

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	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

AdaBoost Example

- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

- Classification

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

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
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

Boosting Advantages and Limitations

Advantages

- Combines weak learners into a strong predictive model.
- Often reduces bias; it can also reduce variance.
- Builds models sequentially to address weaknesses of the current ensemble.
- Can achieve high predictive accuracy with simple base learners.

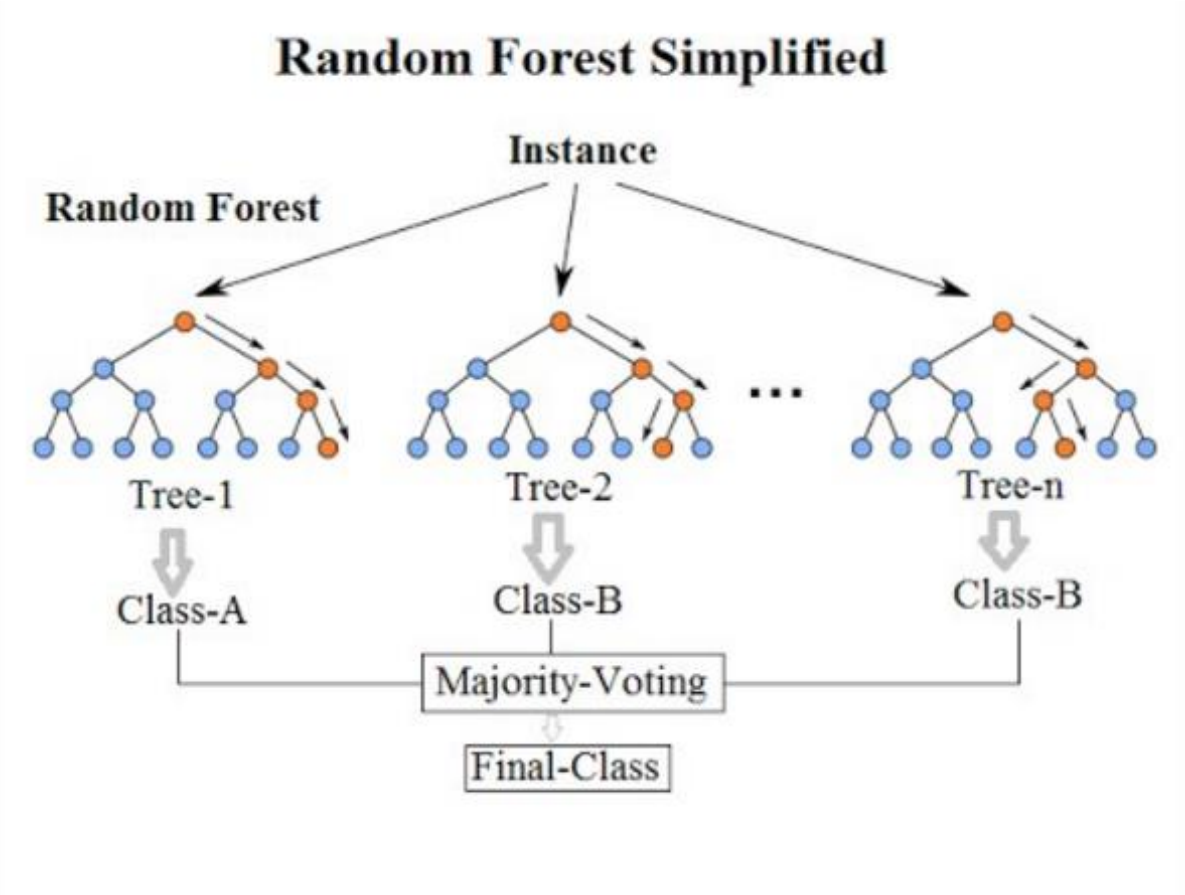
Limitations

- Can be sensitive to noise, outliers, and incorrect labels, especially AdaBoost.
- Sequential training limits parallelization across boosting rounds.
- Requires careful tuning of the learning rate, number of learners, and base-model complexity.
- Can overfit with excessive model complexity or insufficient regularization.
- The final ensemble is less interpretable than an individual simple learner.

Random Forests

Random Forests

- Is a class of ensemble methods specifically designed for **decision trees**.
- It combines the predictions made by multiple decision trees and outputs the class that **is the mode of the class's output** by individual trees.



Only Bagging?

- With bagging we have an ensemble of structurally similar trees. This causes **highly correlated trees**.

- Random Forest aims at creating trees that have **no correlation** or **weak correlation**.

RF solution

- To de-correlate the trees:
  1. Take a random sample of size N with replacement from the data.
  2. Learn a decision tree where for each split a random sample without replacement of *m* attributes is considered

- m ~ $\sqrt{d}$ or m ~ $\log d$+1 turns out to be a good choice

- *Example*: if we have 100 features, each split will be allowed to choose from among 10 randomly selected features

Random Forest - Advantages

- It is one of the most accurate learning algorithms available. For many data sets, it produces a high accurate classifier.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.

Final Thoughts on Random Forests

- When the number of attributes is large, but the number of relevant predictors is small, random forests can perform poorly.
- In each split, the chances of selected a relevant attribute will be low and hence most trees in the ensemble will be **weak models**.
- Increasing the number of trees in the ensemble generally does not increase the risk of overfitting.
- Decomposing the generalization error in terms of bias and variance, we see that increasing the number of trees produces a model that is at least as robust as a single tree.
- However, if the number of trees is too large, then the trees in the ensemble may become more correlated, increase the variance.

References

- Ensemble Methods. Chapter 5.6. Introduction to Data Mining.

