CZ4041/SC4000: Machine Learning

Lesson 9: Ensemble Learning

Li Boyang, Albert

School of Computer Science and Engineering, NTU, Singapore

Acknowledgements: some figures are adapted from the lecture notes of the books "Introduction to Data Mining" (Chap. 5). Slides are modified from the version prepared by Dr. Sinno Pan.

Ensemble Methods

- Objective:
  - To improve model performance in terms of accuracy by aggregating the predictions of multiple models
- How to do it?
  - Construct a set of base models from the training data
  - Make predictions by combing the predicted results made by each base model

"Two heads are better than one"

Stories of Success

- Data mining competitions on Kaggle
  - Winning teams employ ensembles of classifiers

kaggle

Feature Engineering ⟹ Ensemble Learning

Why Ensemble Work?

- Suppose there are 3 base binary classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$ or accuracy $\text{acc} = 0.65$
  - Given a test instance, if we choose any one of these classifiers to make prediction, the probability that the classifier makes a wrong prediction is 35%

Base classifiers: f_1 f_2 f_3

A test instance: x

Why Ensemble Work (cont.)

- Consider to combine the 3 base classifiers to make a prediction on a test instance using a majority vote
- The ensemble makes a wrong prediction only if more than 1 (i.e. 2 or 3) of the base classifiers predict incorrectly

Why Ensemble Work (cont.)

x	f_1	f_2	f_3	f_M	
Truth label: -1	+1	+1	+1	+1	✗
	+1	+1	-1	+1	✗
	+1	-1	+1	+1	✗
	-1	+1	+1	+1	✗
	+1	-1	-1	-1	✓
	-1	+1	-1	-1	✓
	-1	-1	+1	-1	✓
	-1	-1	-1	-1	✓

The combined model makes a wrong prediction if at least two of the three base classifiers make a wrong prediction at the same time

Why Ensemble Work (cont.)

- Assuming the three base classifiers are independent
  - The decision of one classifier does not tell us anything about another classifier.
- Each classifier has an error rate of $\epsilon = 35\%$
- What is the overall error rate of the ensemble?

Why Ensemble Work (cont.)

x	f_1		f_2		f_3		f_M	
Truth label: -1	+1	35%	+1	35%	+1	35%	+1	✗
3 lines with the same probability	+1	35%	+1	35%	-1	65%	+1	✗
	+1		-1		+1		+1	✗
	-1		+1		+1		+1	✗
	+1		-1		-1		-1	✓
	-1		+1		-1		-1	✓
	-1		-1		+1		-1	✓
	-1		-1		-1		-1	✓

The combined model makes a wrong prediction if at least two of the three base classifiers make a wrong prediction at the same time

Why Ensemble Work (cont.)

- Therefore, probability that the ensemble classifier makes a wrong prediction is:

$$\sum_{i=2}^{3} \binom{3}{i} \varepsilon^{i}(1-\varepsilon)^{3-i} = 3 \times 0.35^2 \times 0.65 + 1 \times 0.35^3 \times 1 = 0.2817$$

  - Case 1: when there are two exact classifiers make wrong predictions, the probability is

Two classifiers make wrong predictions

All possible combination $\rightarrow \boxed{\binom{3}{2}}\boxed{\varepsilon^2}\boxed{(1-\varepsilon)^{3-2}} \leftarrow$ The rest one makes correct prediction

  - Case 2: when all the three classifiers make wrong predictions, the probability is

$$\binom{3}{3} \varepsilon^{3}(1-\varepsilon)^{3-3}$$

- That is the accuracy of the ensemble classifier is 71.83%

$$\varepsilon_{f_i} = 35\% \longrightarrow \varepsilon_M = 28.17\%$$

Why Ensemble Work (cont.)

- Suppose there are 25 independent base classifiers
  - Therefore, probability that the ensemble classifier makes a wrong prediction is:

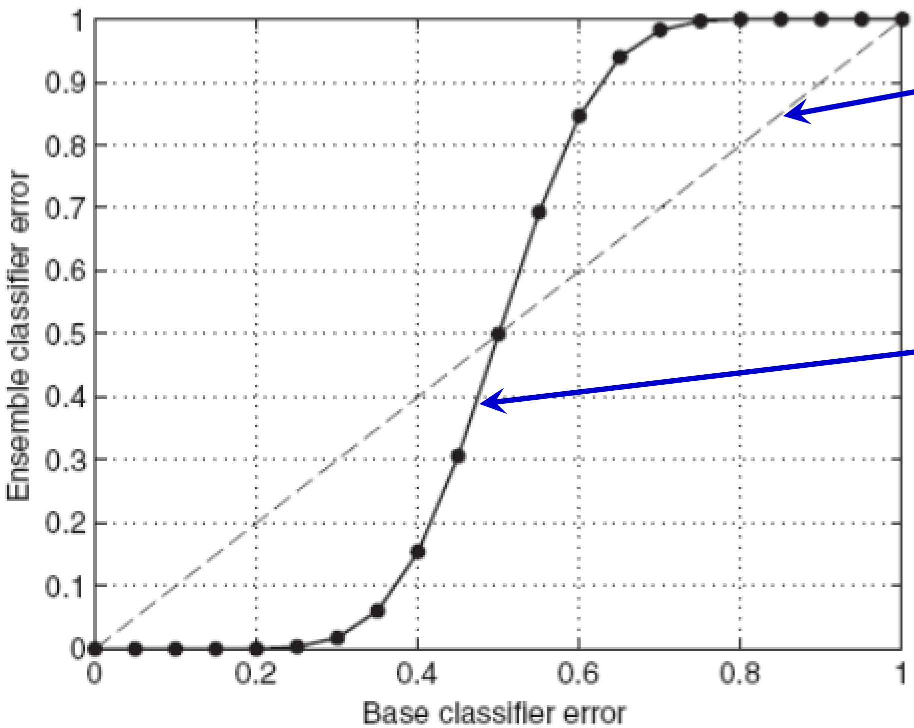
$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

  - That is the accuracy of the ensemble classifier is 94%

$$\varepsilon_{f_i} = 35\%$$

$$\varepsilon_M = 6\%$$

Error Rate of Base Classifiers



The base classifiers are identical (perfectly correlated)

The base classifiers are independent

Error rate of an ensemble of 25 binary classifiers for different base classifier error rates

Observation: the ensemble classifier performs worse than the base classifiers when the base classifier error rate is larger than 0.5

Correlation between Base Classifiers

- Sometimes the base classifiers are not completely independent.
- An extreme case is when they are perfectly correlated. That is, they always have the same predictions.

f_1	f_2
$+1$	$+1$
-1	-1
-1	-1
$+1$	$+1$
$+1$	$+1$

Perfectly positively correlated

f_1	f_2
$+1$	-1
-1	$+1$
-1	$+1$
$+1$	-1
$+1$	-1

Perfectly negatively correlated

Review: Random Variables

- Consider two random variables X and Y, which may take on values from $\{0, 1\}$.
  - Random variables are variables whose values are randomly assigned (from some unspecified random experiment).
- Imagine X and Y are the results of two coin tosses.
  - Head for the first toss $\Rightarrow X = 1$
  - Tail for the first toss $\Rightarrow X = 0$
  - Head for the second toss $\Rightarrow Y = 1$
  - Tail for the second toss $\Rightarrow Y = 0$
- The probabilities are denoted as $P(X = 0)$, $P(X = 1)$, $P(Y = 0)$, $P(Y = 0)$

Review: Joint Probability

- $P(X = 0, Y = 0)$ is the probability of both toss getting tail.
- What about $P(X = 1, Y = 0)$?
- What about $P(Y = 0, Y = 1)$?

Review: Conditional Probability

- $P(X = 0|Y = 0)$ is the probability of the first toss being a tail, knowing the second toss is a tail.
- What about $P(Y = 0|X = 1)$?
- What about $P(X = 0|X = 0)$?

Review: Conditional Probability

- Conditional and joint probabilities are related as
- $P(X = x, Y = y) = P(Y = y|X = x)P(X = x)$
- $P(X = x, Y = y) = P(X = x|Y = y)P(Y = y)$

Knowing that Y takes on value y, the probability of X takes on value x

The probability of Y taking on value y

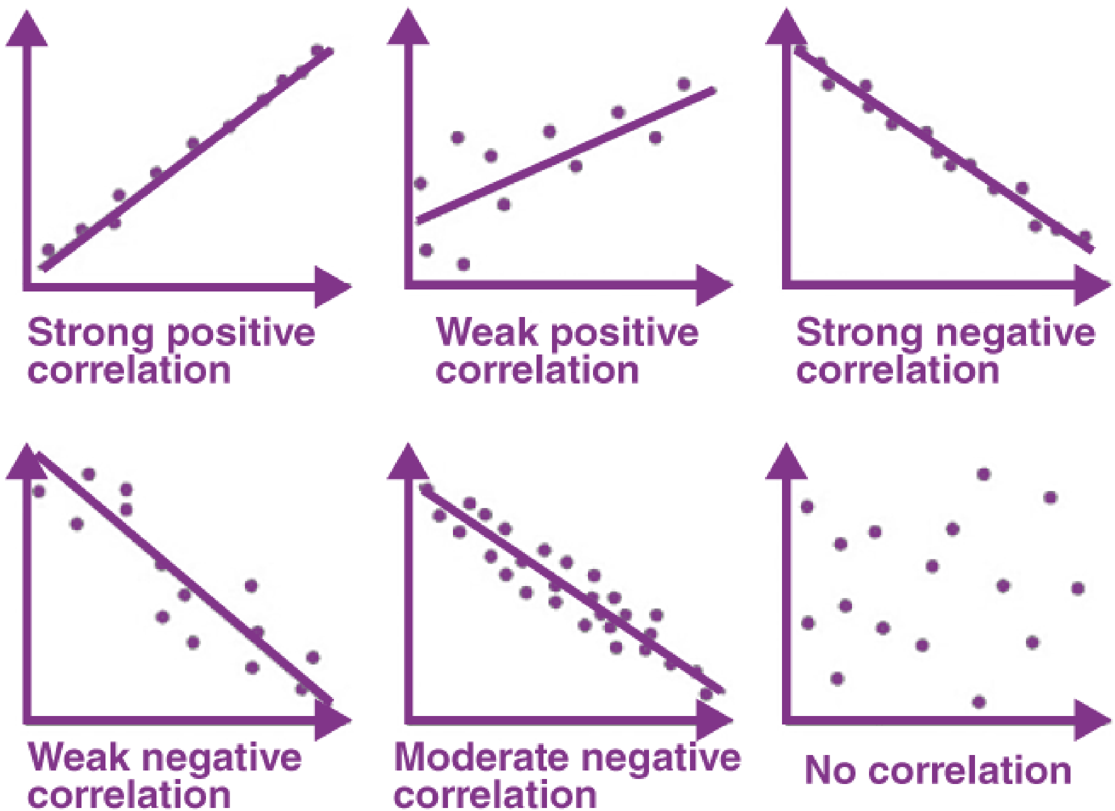
The probability that Y takes on value y **and** X takes on value x

Review: Independence

- If X and Y are independent, then we have
- $\mathrm{P}(X = x|Y = y) = \mathrm{P}(X = x)$
  - Y provides no knowledge about X
  - E.g., no matter the outcome of the first coin toss, the second toss is still 50/50
  - OR, no matter the second toss, the first toss is still 50/50
- Equivalently, $\mathrm{P}(X = x, Y = y) = \mathrm{P}(X = x)\mathrm{P}(Y = y)$
- Why is this equivalent?

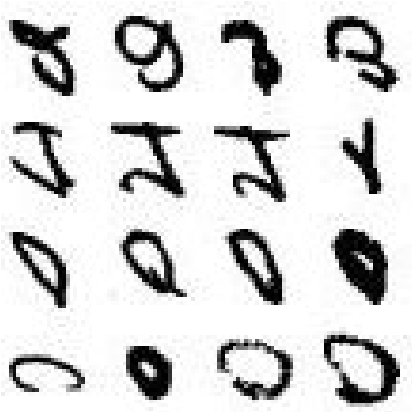
$$\begin{aligned}\mathrm{P}(X = x, Y = y) &= \mathrm{P}(X = x|Y = y)\mathrm{P}(Y = y) \\ &= \mathrm{P}(X = x)\mathrm{P}(Y = y)\end{aligned}$$

Review: Correlation between RVs



One variable tells us something about the other variable.

Difficult Input Causes Correlation



- These are supposedly digits from 0 to 9.
- Most models probably make incorrect predictions on these data points.
- Having multiple models would not help too much here.

Necessary Conditions

- Two necessary conditions for an ensemble classifier to perform better than a single classifier:
  1. The base classifiers are not perfectly correlated with each other.
     - In practice, the base classifiers are usually somewhat correlated.
  2. The base classifiers should do better than a classifier that performs random guessing (e.g., for binary classification, accuracy should be better than 0.5)

Tutorial

Ensemble Methods

- How to generate a set of base classifiers?
  - By manipulating the training set: multiple training sets are created by resampling the original data according to some sampling distribution. A classifier is then trained from each training set, such as Bagging, Boosting
  - By manipulating the input features: a subset of input features is chosen to form each training set. A classifier is then built from each training set, such as Random forest
  - By manipulating the learning algorithm(s): applying the algorithm several times on the same training data using different parameters or applying different algorithms
- How to combine the base classifiers for predictions?

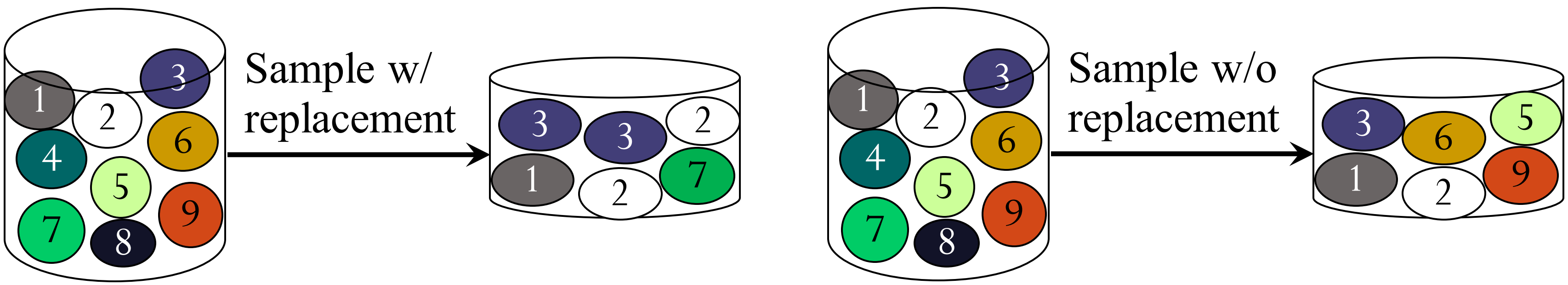
General Procedure

1. Let D denote the original training data, k denote the number of base classifiers, and T be the test dataset.
2. **for** $i = 1$ to k **do**
3. Train a base classifier f_i from D
4. **end for**
5. **for** each test instance $\boldsymbol{x} \in T$ **do**
6. Generate $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$, …, and $f_k(\boldsymbol{x})$
7. Calculate $f_M(\boldsymbol{x}) = \boxed{\text{Merge}}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$
8. **end for**

For example, majority voting
(can be other schemes)

Bagging

- Known as bootstrap aggregating (bootstrapping), to repeatedly sample with replacement according to a uniform probability distribution



- Build classifier on each bootstrap sample, which is of the same size of the original data
- Use majority voting to determine the class label of ensemble classifier

Bagging (cont.)

Index of an instance

Original Data	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	D
Round 1	**7**	**8**	**10**	**8**	**2**	**5**	**10**	**10**	**5**	**9**	D_1

f_1

Round 2	**1**	**4**	**9**	**1**	**2**	**3**	**2**	**7**	**3**	**2**	D_2

f_2

Round 3	**1**	**8**	**5**	**10**	**5**	**5**	**9**	**6**	**3**	**7**	D_3

f_3

Bagging (cont.)

Index of an instance

Test Data	1	2	3	4	5	6	7	8	9	10
Round 1 (f_1)	+	+	+	+	-	-	-	+	-	-
Round 2 (f_2)	-	+	-	+	-	-	+	-	+	+
Round 3 (f_3)	+	-	-	+	+	-	-	+	+	-
Ensemble (f_M)	+	+	-	+	-	-	-	+	+	-

Majority Voting

Bagging (cont.)

- Suppose a training set D contains N examples
- A training instance has a probability of $1 - \frac{1}{N}$ of *not* being selected
- Its probability of ending up *not* in a training set D_i is $\left(1 - \frac{1}{N}\right)^N \approx \frac{1}{e} = 0.368$
- A bootstrap sample D_i contains approximately 63.2% of the original training data

Implementation Example

```
>>> from sklearn.ensemble import BaggingClassifier
>>> from sklearn.svm import SVC
```

Classification algorithm used in bagging

…

Specify how many base classifiers

```
>>> bagC = BaggingClassifier(base_estimator=SVC(), n_estimators=10 )
>>> bagC.fit(X, y)
>>> pred= bagC.predict(X)
```

Specify a base classification algorithm (usually decision tree is used)

Boosting

- Principles:
  - Boost a set of weak learners to a strong learner
  - Make instances currently misclassified more important
- Generally,
  - To adaptively change the distribution of training data so that the base classifiers will focus more on previously misclassified records

Boosting (cont.)

- Specifically,
  - Initially, all N instances are assigned equal weights
  - Unlike bagging, weights may change at the end of each boosting round
  - In each boosting round, after the weights are assigned to the training instances,
    - Draw a bootstrap sample from the original data by using the weights as a sampling distribution to build a model

Boosting: Procedure

1. Initially, all instances are assigned equal weights $\frac{1}{N}$, so that they are equally likely to be chosen for training. A sample is drawn uniformly to obtain a new training set.
2. A classifier is induced from the training set, and used to classify all the examples in the original training set
3. The weights of the training instances are updated at the end of each boosting round
    - Instances that are wrongly classified will have their weights increased
    - Instances that are classified correctly will have their weights decreased
4. Repeat Steps 2 and 3 until the stopping condition is met
5. Finally, the ensemble is obtained by aggregating the base classifiers obtained from each boosting round

Boosting: Example

Initially, all the instances are assigned the same weights.

	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	
Original Data	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	D
	+1	+1	+1	+1	-1	-1	-1	+1	+1	-1	

Uniformly randomly sample using bootstrapping (sampling with replacement)

Round 1	**7**	**3**	**2**	**8**	**7**	**9**	**4**	**10**	**6**	**3**	D_1

A classifier built from the data

f_1

Perform the classifier on all original instances

Not selected in Round 1

Misclassified

Original Data	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
True Label:	+1	+1	+1	+1	-1	-1	-1	+1	+1	-1
Prediction:	+1	-1	+1	-1	-1	-1	-1	+1	+1	-1

Weights increased (relatively) for instances not selected in Round 1

Weights increased for misclassified instances

Weights decreased for correctly classified instances

	$\frac{1}{10}$	$\frac{1}{4}$	$\frac{1}{20}$	$\frac{1}{4}$	$\frac{1}{10}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$
Original Data	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	+1	+1	+1	+1	-1	-1	-1	+1	+1	-1

D

Randomly sample based on the weights of each instance

Round 2	**5**	**4**	**9**	**4**	**2**	**5**	**1**	**7**	**4**	**2**

D_2

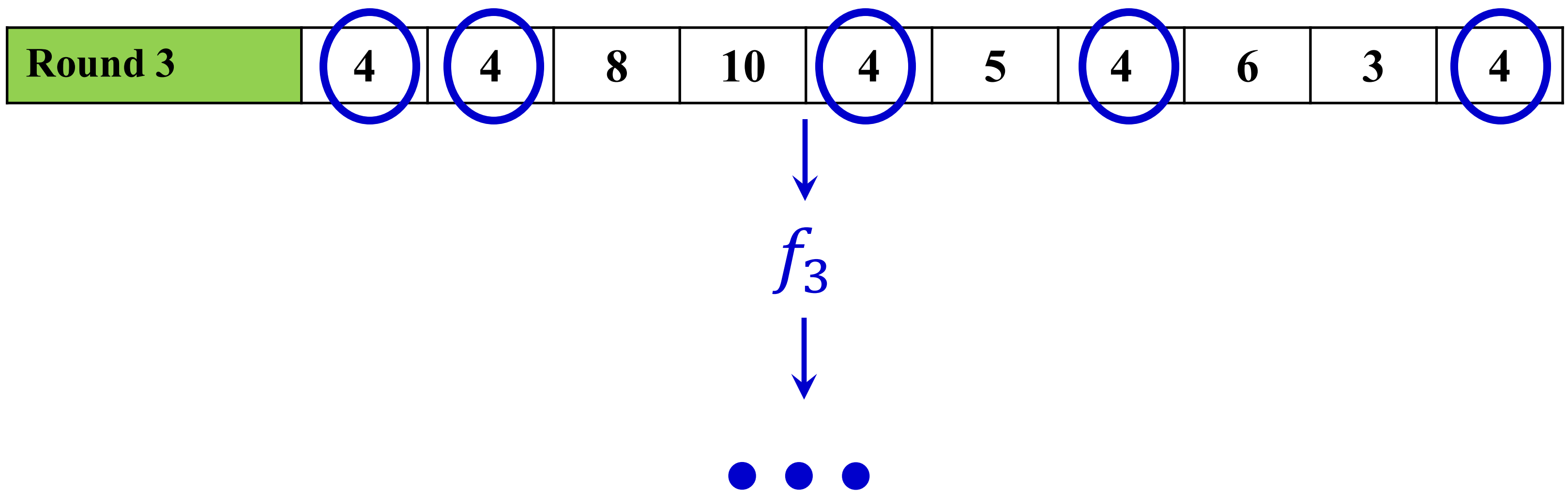
A classifier built from the data

f_2

Perform the classifier on all original instances

Original Data	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
True Label:	+1	+1	+1	+1	-1	-1	-1	+1	+1	-1
Prediction:	+1	+1	+1	-1	-1	-1	-1	+1	+1	-1

Randomly sample based on the updated weights of each instance



As the boosting rounds proceed, examples that are the hardest to classify tend to become even more prevalent, e.g., instance 4

Weights for each classifier

Index of an instance

Test Data	1	2	3	4	5	6	7	8	9	10	Weights
Round 1 (f_1)	+	+	+	+	-	-	-	+	-	-	**1**
Round 2 (f_2)	-	+	-	+	-	-	+	-	+	+	**1.8**
Round 3 (f_3)	+	-	-	+	+	-	-	+	+	-	**3**
Ensemble (f_M)	+4 v.s. -1.8	+2.8 v.s. -3	+1 v.s. -4.8	+5.8	+3 v.s. -2.8	-5.8	+1.8 v.s. -4	+4 v.s. -1.8	+4.8 v.s. -1	+1.8 v.s. -4	
Prediction	+	-	-	+	+	-	-	+	+	-	

Weighted Voting

Note: here the values of the classifiers weights as well as the updated instance weights in each boosting round are just examples. Different boosting algorithms have different ways to compute these weights.

Implementation Example

```
>>> from sklearn.ensemble import AdaBoostClassifier
>>> from sklearn.svm import SVC
...
>>> adaC = AdaBoostClassifier (base_estimator=SVC(), n_estimators=10 )
>>> adaC.fit(X, y)
>>> pred= adaC.predict(X)
```

AdaBoost, a classic boosting algorithm

Specify how many base classifiers

Specify a base classification algorithm

Random Forests

- A class of ensemble methods specifically designed for decision tree classifiers
- Random Forests grow many trees
- Each tree is generated based on a random subset of features
- Final result on classifying a new instance – voting
  - Forest chooses the classification result having the most votes (over all the trees in the forest)

Random Forests (cont.)

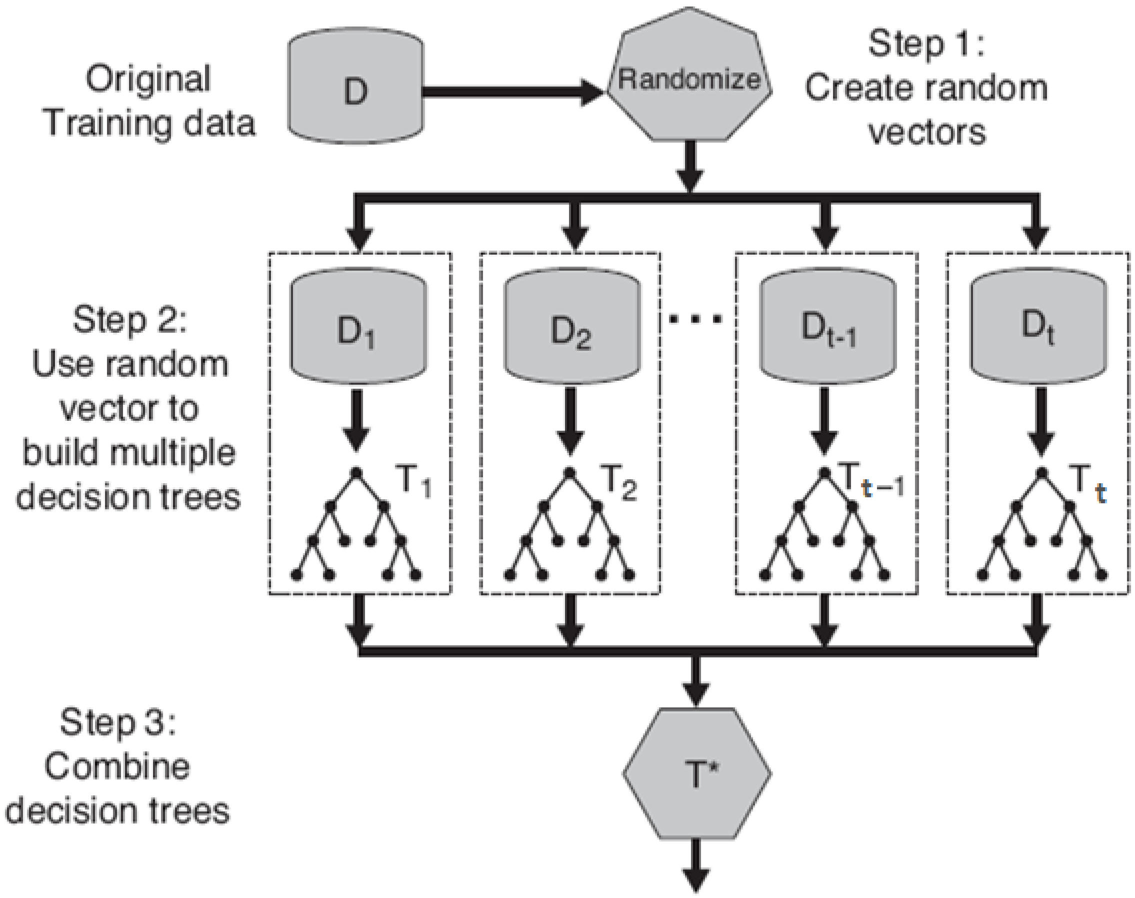
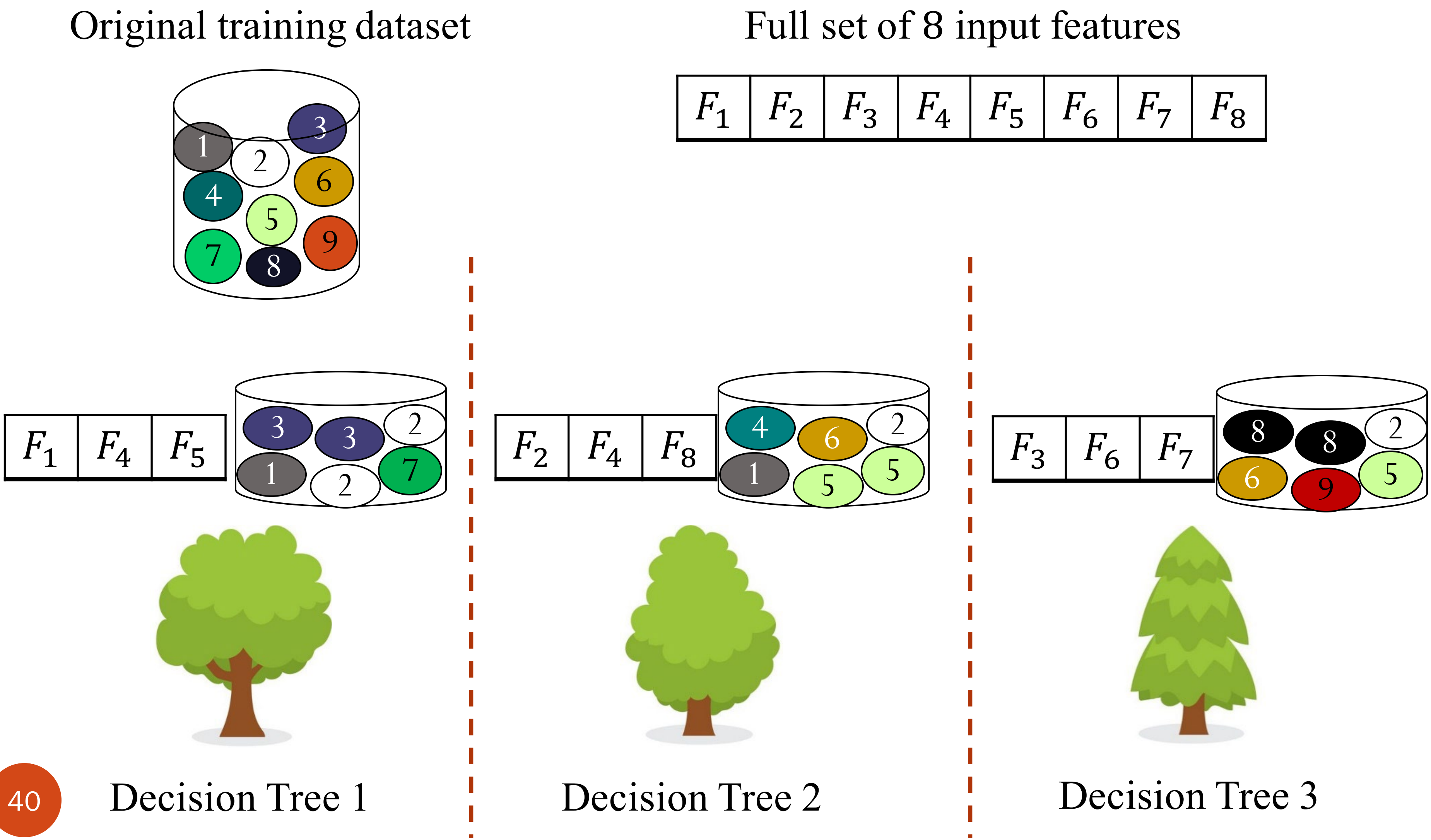


Illustration of random forests

Random Forests: Algorithm

- Choose T: number of trees to grow
- Choose $m' < m$ (m is the number of total features): number of features used to calculate the best split at each node (typically 20%)
- For each tree
  - Choose a training set via bootstrapping
  - For each node, randomly choose m' features and calculate the best split
  - Fully grown and not pruned
- Use majority vote among all the trees

Example



Original training dataset
Full set of 8 input features
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8
1 2 3 4 5 6 7 8 9
F_1 F_4 F_5
3 3 2 1 2 7
F_2 F_4 F_8
4 6 2 1 5 5
F_3 F_6 F_7
8 8 2 6 9 5
Decision Tree 1
Decision Tree 2
Decision Tree 3

Random Forests: Discussions

- Bagging + random features
- Improve Accuracy
  - Incorporate more diversity
- Improve Efficiency
  - Searching among subsets of features is much faster than searching among the complete set

Implementation Example

```
>>> from sklearn.ensemble import RandomForestClassifier
```

…

Set parameter
for the base tree

```
>>> rfC = RandomForestClassifier(n_estimators = 20, max_depth = 3)
>>> rfC.fit(X, y)
>>> pred= rfC.predict(X)
```

Specify how many
base classifiers

Combination Methods

- Voting
  - Majority voting
  - Weighted voting
- Average
  - Simple average
  - Weighted average
- Combining by learning

Voting

- Majority voting:
  - Takes the class label that receives the largest number of votes as the final winner
- Weighted voting:
  - A generalized version of majority voting by introducing weights for each classifier

Average

- Simple average:

$$f_M(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} f_i(\boldsymbol{x})$$

- Weighted average:

$$f_M(\boldsymbol{x}) = \sum_{i=1}^{T} w_i f_i(\boldsymbol{x})$$

$$\text{where } w_i \geq 0, \text{and } \sum_{i=1}^{T} w_i = 1$$

- Prediction:

$$\hat{y}(\boldsymbol{x}) = \begin{cases} 1, & f_M(\boldsymbol{x}) \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

Combining by Learning

- Stacking:
  - A general procedure where a learner is trained to combine the individual learners
  - Individual learners: first-level learners
  - Combiner: second-level learner, or meta-learner

Combining by Learning (cont.)

- Suppose given a binary classification problem,

m input features

N training instances

	X_1	X_2	…	X_m	Y
x_1	3	1	…	0.3	1
x_2	4	1	…	0.6	-1
…	…	…	…	…	…
x_N	-10	0	…	0.2	1

k base classifiers are trained

$f_1 \quad f_2 \quad \cdots \quad f_k$

Predicted value of classifier f_1 on the instance $\boldsymbol{x}_1$

k base classifiers

Labels

N instances

	f_1	f_2	…	f_k	Y
x_1	0.6	1	…	0.3	1
x_2	0.3	1	…	0.6	-1
…	…	…	…	…	…
x_N	0.1	-1	…	0.2	1

A meta classifier is learned: $f_M\colon \mathbb{R}^k \to \{-1,1\}$

Test instance

	X_1	X_2	…	X_m
x^*	0.8	-1	…	0.4

$\xrightarrow{\text{Apply } f_1,\ldots,f_k}$

	f_1	f_2	…	f_k
$x^{*\prime}$	0.3	0.6	…	1

Prediction $\boxed{f_M(\boldsymbol{x}^{*\prime})}$

Combining by Learning: Summary

- Represent each training instances using classifier-generated outputs

$$\boldsymbol{x}_i' = (f_1(\boldsymbol{x}_i), f_2(\boldsymbol{x}_i), \dots, f_k(\boldsymbol{x}_i))$$

- Learn a meta classifier f_M from $\{\boldsymbol{x}_i', y_i\}, i = 1, \dots, N$.
- For a test instance $\boldsymbol{x}^*$
  - Represent it by $\boldsymbol{x}^{*\prime} = (f_1(\boldsymbol{x}^*), f_2(\boldsymbol{x}^*), \dots, f_k(\boldsymbol{x}^*))$
  - Use f_M to make a prediction $f_M(\boldsymbol{x}^{*\prime})$

Thank you!