Ensemble Methods: Rationale

This section presents techniques for improving classification accuracy by **aggregating** the predictions of **multiple classifiers**. These techniques are known as **ensemble** methods. An ensemble method constructs a set of base classifiers from training data and performs a classification. There are a few kinds of ensemble methods, many of them **outperform** a single classifier.

We will present the rationale of the ensemble methods using majority vote and later move on to the topics of **bagging**, **random forests**, and **boosting** – more practical and powerful ensemble learning techniques to construct classification models. Most of the techniques presented herein are also applicable to regression problems.

1. Motivation: Majority Vote

Let us walk through an example by taking a **majority vote** on the predictions made by each base classifier.

Example: Consider an ensemble of 3 binary classifiers, each of which has an error rate[1] of $e = 0.35$. All the possible combinations of the ensemble are:

A	B	C
0.65	0.65	0.65
0.35	0.65	0.65
0.65	0.35	0.65
0.65	0.65	0.35
0.35	0.35	0.65
0.65	0.35	0.35
0.35	0.65	0.35
0.35	0.35	0.35

$i=1$

$\varepsilon_1 = C_1^3 e^1 (1-e)^2$

$= \frac{3!}{1!(2!)} \cdot 0.35 \cdot 0.65^2$

$= \frac{3\times 2}{2\times 1} \cdot 0.35 \cdot 0.65^2$

Formally, the error rate for each combination can be computed by:

$$\varepsilon_i = \binom{3}{i} e^i (1-e)^{3-i}$$

$\binom{n}{k}$ is read ***n* choose *k***. $\binom{n}{k}$ is called binomial coefficient and can be computed as $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

ε 0.35 $\quad 1-\varepsilon$ accuracy 0.65

[1] Error rate means how often it is wrong. For a binary classifier, error rate is defined as $e = \frac{\text{Number of wrong prediction}}{\text{Total Number of prediction}}$.

Table below lists all the error rates from each combination:

A	B	C	Error Rate
0.65	0.65	0.65	0.274625
0.35	0.65	0.65	0.147875
0.65	0.35	0.65	0.147875
0.65	0.65	0.35	0.147875
0.35	0.35	0.65	0.079625
0.65	0.35	0.35	0.079625
0.35	0.65	0.35	0.079625
0.35	0.35	0.35	0.042875

1.0

The ensemble classifier predicts the class label of a test example by taking a majority vote on the prediction made by the base classifiers. The ensemble error rate through majority vote is:

$$\varepsilon_{\text{ensemble}} = \sum_{i=2}^{3} \binom{3}{i} e^i (1-e)^{3-i} = 0.28175$$

which is slightly better than the error rate of a base learner $e = 0.35$.

Example: Consider an ensemble of 25 binary classifiers, each of which has an error rate of $\varepsilon = 0.35$. The ensemble classifier predicts the class label of a test example by taking a majority vote on the prediction made by the base classifiers. Assume that the base classifiers are independent of each other and the ensemble makes a wrong prediction only if more than half of the base classifiers predict incorrectly.

(1) What is the error rate of the ensemble for the case with 13 base classifiers predicting incorrect answer?

(Ans)

$$\varepsilon_{13} = \binom{25}{13} e^{13} (1-e)^{12}$$

$\binom{n}{k}$ is read ***n* choose *k***. $\binom{n}{k}$ is called binomial coefficient and can be computed as $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

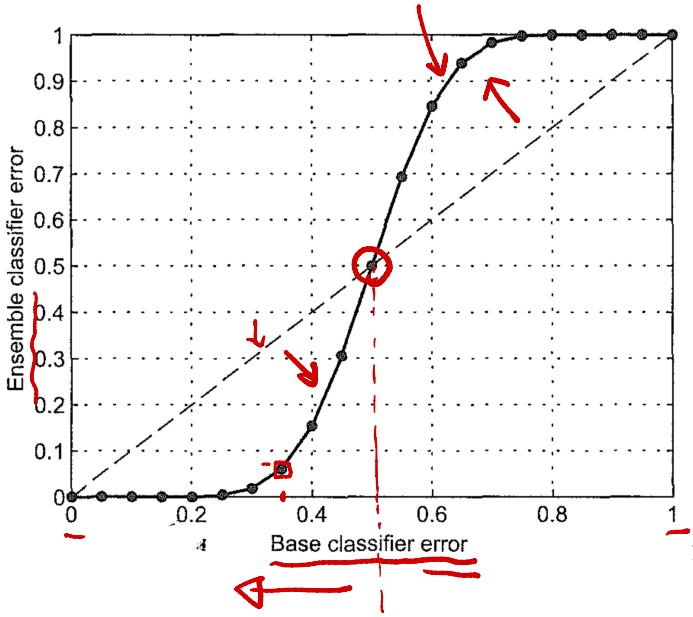
(2) What is the error rate of the ensemble method? through majority vote

(Ans)

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

which is considerably lower than the error rate of the base classifiers.

Figure below shows the error rate of an ensemble of 25 binary classifiers through the majority vote ($\varepsilon_{\text{ensemble}}$) for different base classifier error rate (ε).



for 25 binary classifiers

slido 073374

Fun Time: what have you observed from the above figure? (1) The ensemble classifier outperforms the base classifier of any error rate (2) The ensemble classifier outperforms the base classifier when $\varepsilon > 0.5$ (3) The ensemble classifier outperforms the base classifier when $\varepsilon < 0.5$.

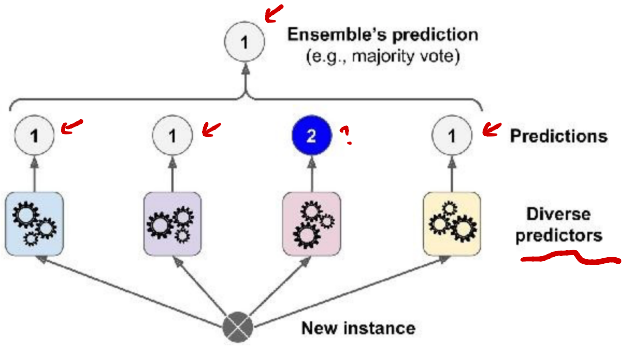
Remark: The error rate for random guessing is $\varepsilon = 0.5$.

Remark: The majority vote explains the statistical rationale behind election. 民主制度投票的合理性。However, there are two necessary conditions for an ensemble classifier to perform better than a single classifier:

1. The base classifier should be independent of each other.
2. The base classifier should do better than a classifier that performs random guessing.

In practice, it is difficult to ensure total independence among the base classifiers. Nevertheless, even that the base classifiers are correlated, improvements on classification accuracy has been observed in the ensemble methods. In fact, the winning solutions in machine learning competitions often involve several ensemble methods.

2. Python example: majority vote



Let us use `makes_moon` to make two interleaving half circles and creates and trains a voting classifier in `Scikit-Learn`, composed of three diverse classifiers. Let us first create and plot the data:

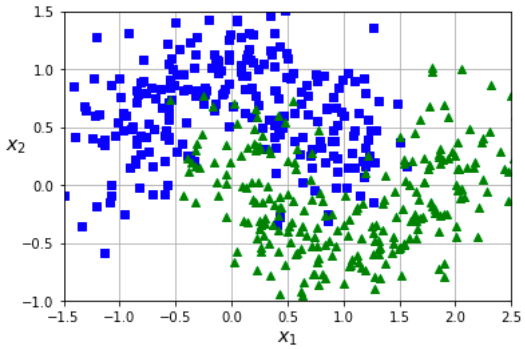
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons

X, y = make_moons(n_samples=500, noise=0.30, random_state=42)

def plot_dataset(X, y, axes):
    plt.plot(X[:, 0][y==0], X[:, 1][y==0], "bs")
    plt.plot(X[:, 0][y==1], X[:, 1][y==1], "g^")
    plt.axis(axes)
    plt.grid(True, which='both')
    plt.xlabel(r"$x_1$", fontsize=14)
    plt.ylabel(r"$x_2$", fontsize=14, rotation=0)

plot_dataset(X, y, [-1.5, 2.5, -1, 1.5])
plt.show()
```



Now the training and testing:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42)
```

```
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

log_clf = LogisticRegression(random_state=42)
rnd_clf = RandomForestClassifier(random_state=42)
svm_clf = SVC(random_state=42)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')

voting_clf.fit(X_train, y_train)
for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
LogisticRegression 0.864
RandomForestClassifier 0.872
SVC 0.888
VotingClassifier 0.896
```

There you have it! The voting classifier slightly outperforms all the individual classifiers. You can download the above source code `MajorityVote.ipynb` from the course website.