投票法

投票法的原理

古语有云“三个臭皮匠，顶个诸葛亮”，这也是集成学习的核心理念。训练一个性能很好的单一分类器可能是很困难的，但是，我们是否可以通过一系列性能一般的分类器组合成一个性能更好的分类器呢？

假设有一个二分类问题，真实标签为y。我们有n个相互独立的基分类器$h_1, \dots, h_n$，它们的预测错误率都是ϵ，也就是说：

$$P(h_i(x) \neq y) = \epsilon, i = 1, \dots, n$$

简单来说，我们可以认为$h_i(x)$服从$p = 1 - \epsilon$的伯努利分布。

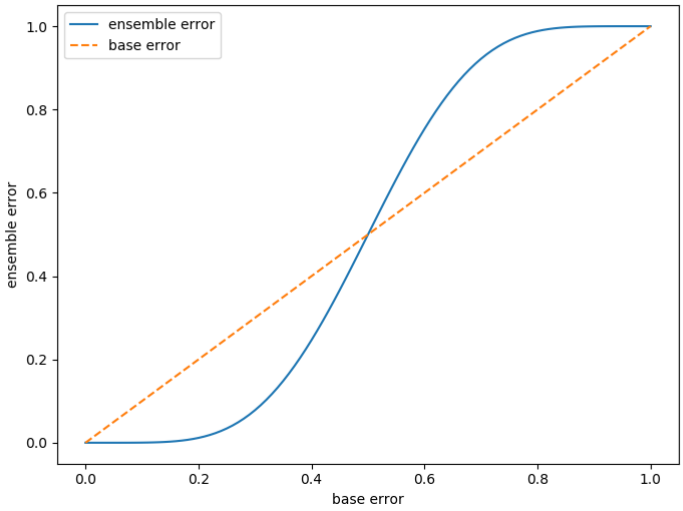
我们使用一个简单的投票法来进行集成，如果超过$n/2$个基分类器将其分为正样本，就集成为正样本，否则集成为负样本。也就是说，超过半数的基分类器分类正确，则集成分类器也分类正确。此时集成分类器$H(x)$服从二项分布：

$$P(H(x) \neq y) = P(\sum_{i=1}^{n} \mathbb{I}_{h_i(x)=y} \leq \frac{n}{2}) = \sum_{i=0}^{\lceil n/2 \rceil} \binom{n}{i} (1-\epsilon)^i \epsilon^{n-i}$$

```python
import numpy as np
from scipy.special import comb
from matplotlib import pyplot as plt

def ensemble_error(n, eps):
    k = int(np.ceil(n/2))
    errors = [comb(n, i) * (1-eps)**i * eps**(n-i) for i in range(0, k)]
    return sum(errors)

n = 11
base_errors = np.arange(0, 1.01, 0.01)
ensemble_errors = [ensemble_error(n, eps) for eps in base_errors]
plt.figure(figsize=(8, 6))
plt.plot(base_errors, ensemble_errors, label="ensemble error")
plt.plot(base_errors, base_errors, linestyle="--", label="base error")
plt.xlabel("base error")
plt.ylabel("ensemble error")
plt.legend()
plt.show()
```



只有当单个基分类器的错误率小于0.5时，绝对多数投票的错误率才会小于单个基分类器的错误率。

也就是说，想要得到一个好的集成学习模型，个体学习器应该“**好而不同**”，既要有一定的“准确性”（错误率小于0.5），又要有一定的“多样性”（各基分类器尽量独立）。

集成学习主要包含三种集合策略：

- 平均法（适用于数值型输出）
  - 简单平均法：$H(x)=\sum_{i=1}^{n}h_i(x)$
  - 加权平均法：$H(x)=\sum_{i=1}^{n}w_ih_i(x)$

    一般而言，**在个体学习器性能相差较大时宜使用加权平均法，而在个体学习器性能相近时宜使用简单平均法。**
- 投票法（适用于类别型输出）
  - 硬投票法：预测结果是所有投票结果最多出现的类
    - 绝对多数投票法：若某类别得票过半数，则预测为该类别，否则拒绝预测
    - 相对多数投票法：选择得票最多的类别作为预测类别（不要求得票过半数），若同时有多个类别获最高票，则从中随机选取一个
  - 软投票法：预测结果是所有投票结果中概率加和最大的类
- 学习法（通过另一个学习器来进行集合）

当投票合集中使用的模型能预测出清晰的类别标签时，适合使用硬投票。当投票集合中使用的模型能预测类别的概率时，适合使用软投票。

投票法的局限性在于，它对所有模型的处理是一样的，这意味着所有模型对预测的贡献是一样的。如果一些模型在某些情况下很好，而在其他情况下很差，这是使用投票法时需要考虑到的一个问题。

投票法的案例分析

以iris数据集为例。

导入相关包

```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier
```

读入数据

```
iris = datasets.load_iris()
X = iris.data
y = iris.target
feature = iris.feature_names
data = pd.DataFrame(X, columns=feature)
data['target'] = y
```

构造基分类器

```
pipe_lr = make_pipeline(StandardScaler(), LogisticRegression())
pipe_dt = make_pipeline(StandardScaler(), DecisionTreeClassifier())
pipe_knn = make_pipeline(StandardScaler(),
KNeighborsClassifier(n_neighbors=10))
base_models = [("lr", pipe_lr), ("dt", pipe_dt), ("knn", pipe_knn)]
```

基于投票法构造集成分类器

```
hard_ensemble = VotingClassifier(estimators=base_models, voting="hard")
models = base_models + [("hard_ensemble", hard_ensemble)]
```

五折交叉验证对比预测结果

```
def evaluate_model(model, X, y):
    score = cross_val_score(model, X, y, scoring='accuracy', cv=5,
error_score='raise')
    return score

for (name, model) in models:
    score = evaluate_model(model, X, y)
    print(f"Model:{name}; Mean: {score.mean():.3f}; Std: {score.std():.3f}")
```

> Model:lr; Mean: 0.960; Std: 0.039
> Model:dt; Mean: 0.960; Std: 0.033
> Model:knn; Mean: 0.960; Std: 0.013
> Model:hard_ensemble; Mean: 0.967; Std: 0.021

可以看到，使用硬投票效果略好于任何一个基模型。