Adaboost算法实现

实验目标

通过本案例的学习和课后作业的练习：

1. 了解Adaboost算法的基本思想；
2. 能够使用SKlearn实现Adaboost算法。

你也可以将本案例相关的 ipynb 学习笔记分享到 AI Gallery Notebook (https://marketplace.huaweicloud.com/markets/aihub/notebook/list/) 版块获得成长值 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=9b8d7e7a-a150-449e-ac17-2dcf76d8b492)，分享方法请查看此文档 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=8afec58a-b797-4bf9-acca-76ed512a3acb)。

案例内容介绍

Adaboost算法基本原理就是将多个弱分类器（弱分类器一般选用单层决策树）进行合理的结合，使其成为一个强分类器。

Adaboost采用迭代的思想，每次迭代只训练一个弱分类器，训练好的弱分类器将参与下一次迭代的使用。也就是说，在第N次迭代中，一共就有N个弱分类器，其中N-1个是以前训练好的，其各种参数都不再改变，本次训练第N个分类器。其中弱分类器的关系是第N个弱分类器更可能分对前N-1个弱分类器没分对的数据，最终分类输出要看这N个分类器的综合效果。

AdaBoost算法的优缺点

优点：

1.很好的利用了弱分类器进行级联

2.可以将不同的分类算法作为弱分类器

3.AdaBoost具有很高的精度

4.相对于bagging算法和Random Forest算法，AdaBoost充分考虑的每个分类器的权重

缺点:

1.AdaBoost迭代次数也就是弱分类器数目不太好设定，可以使用交叉验证来进行确定

2.数据不平衡导致分类精度下降

3.训练比较耗时，每次重新选择当前分类器最好切分点

本案例推荐的理论学习视频：

- 《AI技术领域课程--机器学习》 Adaboost (https://education.huaweicloud.com/courses/course-v1:HuaweiX+CBUCNXE086+Self-paced/courseware/26b0023a6d244591af642fcbaba14ed5/0e8ca99d70b34b30a9da26f6d1103331/)

注意事项

1. 如果您是第一次使用 JupyterLab，请查看《ModelArts JupyterLab使用指导》(https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=03676d0a-0630-4a3f-b62c-07fba43d2857)了解使用方法；
2. 如果您在使用 JupyterLab 过程中碰到报错，请参考《ModelArts JupyterLab常见问题解决办法》(https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=9ad8ce7d-06f7-4394-80ef-4dbf6cfb4be1)尝试解决问题。

实验步骤

1、首先我们载入需要的类库。

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_gaussian_quantiles
%matplotlib inline
```

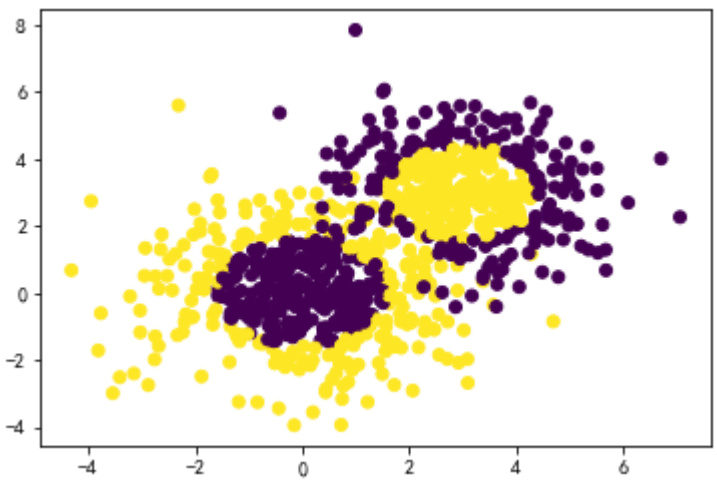
2、接着我们生成一些随机数据来做二元分类。

In [2]:
```
# 生成2维正态分布，生成的数据按分位数分为两类，500个样本,2个样本特征，协方差系数为2
X1, y1 = make_gaussian_quantiles(cov=2.0, n_samples=500, n_features=2,
n_classes=2, random_state=1)
# 生成2维正态分布，生成的数据按分位数分为两类，400个样本,2个样本特征均值都为3，协方差系数为2
X2, y2 = make_gaussian_quantiles(mean=(3, 3), cov=1.5, n_samples=400, n
_features=2, n_classes=2, random_state=1)
# 讲两组数据合成一组数据
X = np.concatenate((X1, X2))
y = np.concatenate((y1, - y2 + 1))
```

3、我们通过可视化看看我们的分类数据，它有两个特征，两个输出类别，用颜色区别。

In [3]:
```
plt.scatter(X[:, 0], X[:, 1], marker='o', c=y)
```

Out[3]: <matplotlib.collections.PathCollection at 0x7f40a4084710>



4、可以看到数据有些混杂，我们现在用基于决策树的Adaboost来做分类拟合。

这里我们选择了SAMME算法，最多200个弱分类器，步长0.8，在实际运用中你可能需要通过交叉验证调参而选择最好的参数。

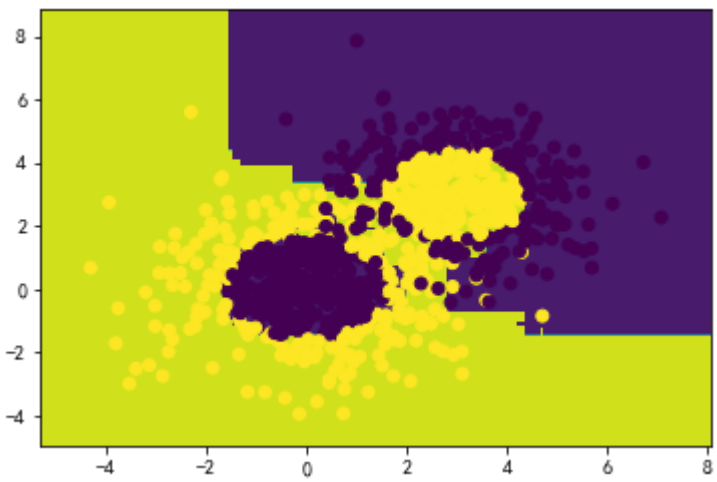
In [4]:
```
bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2, min_samples_split=20, min_samples_leaf=5),
                         algorithm="SAMME",
                         n_estimators=200, learning_rate=0.8)
bdt.fit(X, y)
```

Out[4]:
```
AdaBoostClassifier(algorithm='SAMME',
          base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=2,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=20,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
          learning_rate=0.8, n_estimators=200, random_state=None)
```

5、拟合完了后，我们用网格图来看看它拟合的区域。

In [5]:
```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))

Z = bdt.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
cs = plt.contourf(xx, yy, Z)
plt.scatter(X[:, 0], X[:, 1], marker='o', c=y)
plt.show()
```



6、从图中可以看出，Adaboost的拟合效果还是不错的，现在我们看看拟合分数。

In [6]:
```
print("Score:", bdt.score(X, y))
```

```
Score: 0.9133333333333333
```

7、现在我们将最大弱分离器个数从200增加到300。再来看看拟合分数。

In [7]:
```
bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2, min_samples_split=20, min_samples_leaf=5),
                         algorithm="SAMME",
                         n_estimators=300, learning_rate=0.8)
bdt.fit(X, y)
print("Score:", bdt.score(X, y))
```

```
Score: 0.9622222222222222
```

8、降低步长，将步长从上面的0.8减少到0.5，再来看看拟合分数。

In [8]:
```
bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2, min_samples_split=20, min_samples_leaf=5),
                         algorithm="SAMME",
                         n_estimators=300, learning_rate=0.5)
bdt.fit(X, y)
print("Score:", bdt.score(X, y))
```

```
Score: 0.8944444444444445
```

9、同样的弱分类器的个数情况下，如果减少步长，拟合效果会下降。最后我们看看当弱分类器个数为700，步长为0.7时候的情况。

In [9]:
```
bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2, min_samples_split=20, min_samples_leaf=5),
                         algorithm="SAMME",
                         n_estimators=600, learning_rate=0.7)
bdt.fit(X, y)
print("Score:", bdt.score(X, y))
```

```
Score: 0.9611111111111111
```

此时的拟合分数和我们最初的300弱分类器，0.8步长的拟合程度相当。也就是说，在我们这个例子中，如果步长从0.8降到0.7，则弱分类器个数要从300增加到700才能达到类似的拟合效果。

以上是 Adaboost 的实现方法，受限于篇幅原因，本案例未完全覆盖 Adaboost 的全部操作，欢迎你将更全面的 Adaboost 学习笔记分享到 AI Gallery Notebook (https://marketplace.huaweicloud.com/markets/aihub/notebook/list/) 版块获得成长值 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=9b8d7e7a-a150-449e-ac17-2dcf76d8b492)，分享方法请查看此文档 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=8afec58a-b797-4bf9-acca-76ed512a3acb)。

作业

请你利用本实验中学到的知识点，完成以下编程题：

1. 请你尝试修改 AdaBoostClassifier() 函数的 n_estimators（弱学习器数量）参数的不同取值，看看该参数的修改对模型会有怎样的影响。(https://marketplace.huaweicloud.com/markets/aihub/notebook/detail/?id=3d524d0b-b8df-48c0-93f0-6dfd47093ad4)
2. 请你尝试修改 AdaBoostClassifier() 函数的 learning_rate（弱学习器的权重缩减系数）参数的不同取值，看看该参数的修改对模型会有怎样的影响。(https://marketplace.huaweicloud.com/markets/aihub/notebook/detail/?id=cc3abbda-a181-4db1-9369-8012060bf687)
3. 请你尝试修改 AdaBoostClassifier() 函数的所有可调参数的不同取值，看看不同参数的不同取值组合，对模型会有怎样的影响。(https://marketplace.huaweicloud.com/markets/aihub/notebook/detail/?id=44f87517-9fb8-4c81-8743-f1658ea116dc)