实验五　利用 Adaboost 实现分类器优化实验

实验目标：理解元算法、boosting 等的分类原理；

能使用弱分类器来构建一个强分类器；

熟悉分类器的性能度量指标：正确率、召回率、ROC 曲线等。

实验步骤：

一、集成分类器方法原理

1. boosting

分类中通常使用将多个弱分类器组合成强分类器进行分类的方法，统称为集成分类方法（Ensemble Method），比如 Bagging、Boosting 方法，首先从从整体样本集合中抽样采取不同的训练集训练弱分类器，然后使用多个弱分类器进行 voting，最终的结果是分类器投票的优胜结果。这种简单的 voting 策略通常难以有很好的效果。

Boosting 意为加强、提升，即将弱分类器提升为强分类器。boosting 通过集中关注被已有分类器错分的那些数据来获得新的分类器，其分类的结果基于所有分类器的加权求和结果，分类器权重不相等，每个权重代表其对应分类器在上一轮迭代中的成功度。

2. Adaboost

AdaBoost（Adaptive Boosting）是 Boosting 发展到后来最为代表性的一类。弱分类器根据学习的结果反馈 Adaptively 调整假设的错误率。

AdaBoost 的一般流程：

(1) 收集数据：可以使用任意方法。

(2) 准备数据：依赖于所使用的弱分类器类型，本章使用的是单层决策树，这种分类器可以处理任何数据类型。当然也可以使用任意分类器作为弱分类器，第 2 章到第 6 章中的任一分类器都可以充当弱分类器。作为弱分类器，简单分类器的效果更好。

(3) 分析数据：可以使用任意方法。

(4) 训练算法：AdaBoost 的大部分时间都用在训练上，分类器将多次在同一数据集上训练弱分类器。

(5) 测试算法：计算分类的错误率。

(6) 使用算法：同 SVM 一样，AdaBoost 预测两个类别中的一个。如果想把它应用到多个类别的场

合，那么就要像多类 SVM 中的做法一样对 AdaBoost 进行修改。

3. 基于错误提升分类器的性能

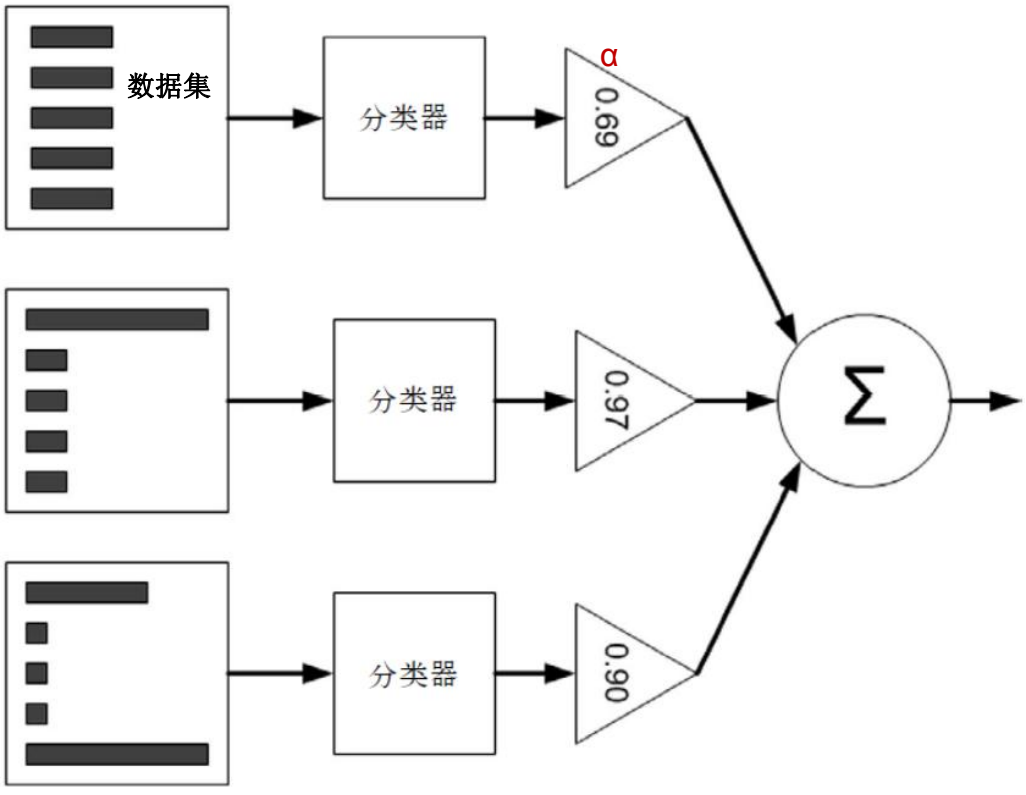
训练数据中的每个样本初始化相等的权重，构成了向量 D。首先在训练数据上训练出一个弱分类器并计算该分类器的错误率，然后在同一数据集上再次训练弱分类器。在分类器的第二次训练当中，将会重新调整每个样本的权重，其中第一次分对的样本的权重将会降低，而第一次分错的样本的权重将会提高。

为了从所有弱分类器中得到最终的分类结果，AdaBoost 为每个分类器都分配了一个权重值 alpha，这些 alpha 值是基于每个弱分类器的错误率：

$$\varepsilon（错误率）=\frac{未正确分类的样本数目}{所有样本数目}$$

$$\alpha=\frac{1}{2}\ln(\frac{1-\varepsilon}{\varepsilon})$$

算法流程如下图：



在经过一个分类器之后，加权的预测结果会通过三角形中的 alpha 值进行加权。每个三角形中输出的加权结果在圆形中求和，从而得到最终的输出结果。

计算出 alpha 值之后，可以对权重向量 D 进行更新，以使得那些正确分类的样本的权重降低而错分样本的权重升高。D 的计算方法如下。

如果某个样本被正确分类，那么该样本的权重更改为：$D_i^{(t+1)} = \frac{D_i^{(t)} e^{-\alpha}}{Sum(D)}$

如果某个样本被错误分类，那么该样本的权重更改为：$D_i^{(t+1)} = \frac{D_i^{(t)} e^{\alpha}}{Sum(D)}$

在计算出 D 之后，AdaBoost 又开始进入下一轮迭代。AdaBoost 算法会不断地重复训练和调整权重的过程，直到训练错误率为 0 或者弱分类器的数目达到用户的指定值为止。

4. 分类器的性能度量

（1）**混淆矩阵** 为了不掩盖样例如何被分错的事实，在机器学习中，除了错误率之外，混淆矩阵（confusion matrix）可以帮助人们更好地了解分类中的错误。

以二类问题为例，如果将一个正例判为正例，即真正例（True Positive，TP，也称真阳）；如果对一个反例正确地判为反例，即真反例（True Negative，TN，也称真阴）。相应地，另外两种情况则分别称为伪反例（False Negative，FN，也称假阴）和伪正例（FalsePositive，FP，也称假阳)。

- **正确率（Precision）**，它等于 TP/(TP+FP)，给出的是预测为正例的样本中的真正正例的比例。
- **召回率（Recall）**，它等于 TP/(TP+FN)，给出的是预测为正例的真实正例占所有真实正例的比例。

（2）**ROC 曲线** 即当阈值变化时假阳率和真阳率的变化情况。左下角的点所对应的是将所有样例判为反例的情况，而右上角的点对应的则是将所有样例判为正例的情况。虚线给出的是随机猜测的结果曲线。

在理想的情况下，最佳的分类器应该尽可能地处于左上角，这就意味着分类器在假阳率很低的同时获得了很高的真阳率。

曲线下的面积（Area Unser the Curve，AUC）给出的是分类器的平均性能值，一个完美分类器的 AUC 为 1.0，而随机猜测的 AUC 则为 0.5。

二、使用 Adaboost 进行分类

1. 基于单层决策树构建弱分类器（详细参考 stumpClassify.py）

单层决策树（decision stump，也称决策树桩）是一种简单的决策树,它仅基于单个特征来做决策，

只有一次分裂过程。

整个实现的伪代码如下：

```
将最小错误率 minError 设为+∞
对数据集中的每一个特征（第一层循环）：
    对每个步长（第二层循环）：
        对每个不等号（第三层循环）：
            建立一棵单层决策树并利用加权数据集对它进行测试
            如果错误率低于 minError，则将当前单层决策树设为最佳单层决策树
返回最佳决单层策树
```

2. 基于单层决策树的 AdaBoost 训练（详细参考 adaboost.py 中的 adaBoostTrainDS 函数）

算法过程大体如下：

```
对每次迭代：
    利用 buildStump()函数找到最佳的单层决策树
    将最佳单层决策树加入到单层决策树数组
    计算 alpha
    计算新的权重向量 D
    更新累计类别估计值
    如果错误率等于 0.0，则退出循环
```

3. 在一个难数据集上应用 Adaboost 进行分类（详细参考 adaboost.py）

(1) 收集数据：提供的文本文件。

(2) 准备数据：确保类别标签是+1 和-1 而非 1 和 0。

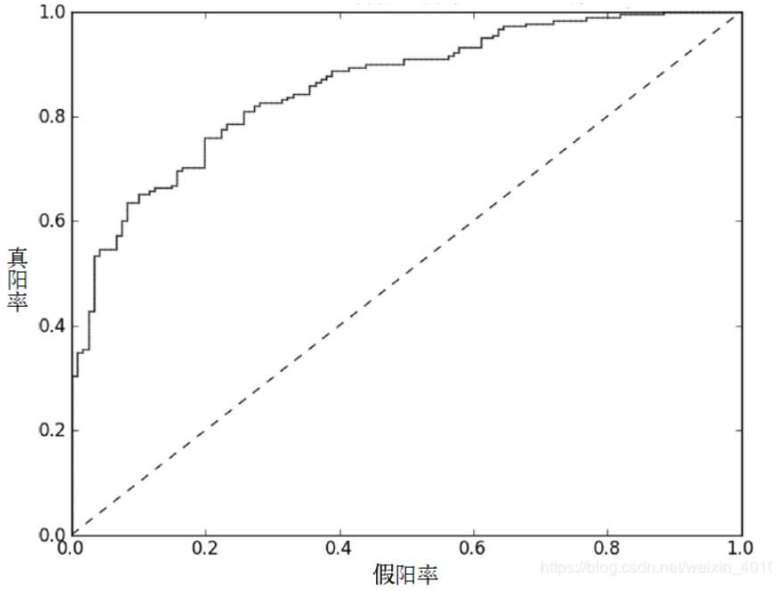
(3) 分析数据：手工检查数据。

(4) 训练算法：在数据上，利用 adaBoostTrainDS()函数训练出一系列的分类器。

(5) 测试算法：我们拥有两个数据集。在不采用随机抽样的方法下，我们就会对 AdaBoost 和 Logistic 回归的结果进行完全对等的比较。

(6) 使用算法：观察该例子上的错误率。不过，也可以构建一个 Web 网站，让驯马师输入马的症状然后预测马是否会死去。

4. 画出分类器的 **ROC 曲线**（ROC curve），ROC 代表接收者操作特征（receiver operating characteristic）。（详细参考 ROC_plot.py）



三、实验要求

1. 使用马疝气病数据集（来源：https://archive.ics.uci.edu/ml/datasets/Horse+Colic（2010 年 1 月 11 日的 UCI 机器学习数据库））， 应用 adaboost 进行分类。代码请参考对应的 Python 代码，调试并查看结果，写出 adaboost 的训练流程图。

2. 画出 ROC 曲线，并分析有哪些优化方法？

注意：请将实验报告以文件形式提交到 QQ 群里，统一命名为：学号+姓名+实验几.doc/rar/zip/pdf