Programming1：Adaboost

2.1 具体实现见 decision_stump 函数。

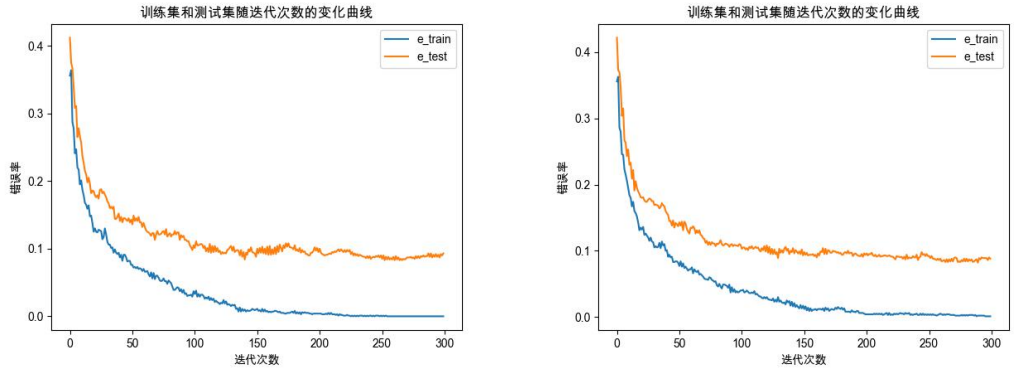
算法以一定的步长遍历所有阈值，并在所有组合的误差中选取最小的误差所对应的维度 k，阈值 a 和标签 d。

```
    for i in range(len(X[0]): #p
        for j in range(2): # d= {1,-1} #  常数复杂度
            for k in range(step_num): #  常数复杂度
                a = X_min + k*step
                e[j,k] = decision_stump_error(X, y, i, a, d, w)
        e = np.sort(e, axis=0) # O(k logk)
  K,d,a = np.argmin(error)
```

k 取值与步长有关，算法复杂度为 O(pk logk)，满足要求。300 次运行时间大概为 10 分钟

2.2 具体实现见 update weights 和 adaboost error 函数.

2.3 得到结果如图 1 所示。随着迭代次数的增加，训练集错误率快速下降接近零，测试集错误率快速下降，稳定在 0.1 左右。不同步长区别不大。



step_size = 0.05　　step_size = 0.1

图 1：不同步长下训练集和测试集错误率随迭代次数变化曲线

Programming2：Random Forest

3.1 基于上周实现的决策树生成随机森林，决策树在 thresh=0.1 的情况下，有最好的正确率 75.4%。单个决策树运行时间 12 分钟左右。随机森林需要的时间更长，因此尝试的 n_tree 参数有限。应该是越多越好。

尝试不同 n_tree 参数 3，5，7，9，在 thresh=0.1 的情况下从原始数据集中有放回的采样，生成随机森林，在 val 验证集上得到最好的 n_tree 参数为 9。在 test 测试集上正确率为 79.3%，与决策树相比，决策森林有一定的性能提升。用 sklearn 包验证 n_tree=30，100 等较大时的情况，可以得出随着随机森林 n_tree 的增大，正确率不断提升，但是 n_tree 越大提升得越少，考虑到经济性，n_tree=10 左右可以满足基本要求。

n_tree	3	5	7	9	30	100
acc	74.1%	77.7%	76.6%	79.3%	84.4%	86.3%

3.2 通过在所有特征中随机选择 max_features 个待选参数，来增加随机森林的随机性。在 n_tree=10 的情况下，尝试不同的 max_features 参数，sqrt(n_features)=34.64(3%),25%,50%，75%，在验证集上得到最好的 max_features 参数为 120，在 test 测试集上正确率为

82.3%，与没有随机选择特征的随机森林相比，即与 max_features=1200 相比，性能有一些提升，说明在这个分类问题中，所给的特征有较多的冗余。

max_features	20	35	120	300	600	900	1200
acc	78.5%	79.7%	82.3%	81.5%	80.9%	80.3%	80.1%

3.3 偏差-方差均衡是机器学习的重要概念。单棵决策树有更高的风险过拟合，因此在测试集上表现不如引入了重采样和随机选特征的随机森林优秀。随机森林作为限制树深度的替代方案，可以减小方差，代价是增大偏差。

随机森林 5 次重复训练后测试集正确率(n_tree=10,max_features=120)

iteration	1	2	3	4	5
acc	81.1%	81.9%	81.1%	81.9%	82.2%

计算可得均值为 81.64%，方差为 2.58E-05。

决策树 5 次重复训练后测试集正确率(max_features = 120)

iteration	1	2	3	4	5
acc	71.5%	68.8%	69.4%	69.1%	68.1%

计算可得均值为 69.38%，方差为 1.64E-04。

可以看到随机森林提升了决策树的正确率，减小了方差，差了一个数量级。