机器学习第二次作业项目报告

1601111294 谭 鑫

1601214435 姜 双

1601214452 陈庆英

一. 实验要求

参考给定的论文，将 adaboost 算法扩展，使其可以处理多分类问题，分析算法的分类效果和性能等。

二. 数据的选择及分析

1. 数据选择

我们选择了 UCI 上的 Letter Recognition 数据集 (http://archive.ics.uci.edu/ml/datasets/Letter+Recognition)。该数据集的分类目标是识别 26 个大写英文字母，共包含 20000 条数据，每条数据包含 16 个离散型特征，取值范围均为 0~15 的整数。各个特征定义如下：

x-box: horizontal position of box (integer)

y-box: vertical position of box (integer)

width: width of box (integer)

high: height of box (integer)

onpix total: # on pixels (integer)

x-bar: mean x of on pixels in box (integer)

y-bar: mean y of on pixels in box (integer)

x2bar: mean x variance (integer)

y2bar: mean y variance (integer)

xybar: mean x y correlation (integer)

x2ybr: mean of x * x * y (integer)

xy2br: mean of x * y * y (integer)

x-ege: mean edge count left to right (integer)

xegvy: correlation of x-ege with y (integer)

y-ege: mean edge count bottom to top (integer)

yegvx: correlation of y-ege with x (integer)

2. 数据分析

由于该数据集没有缺失项，我们不需要进行删除缺失值等预处理。

我们使用 sklearn. model_selection. train_test_split 函数将整个数据集按照 4:1 的比例分为训练集和测试集，样本数分别为 16000 和 4000.

三. 算法与实现

字母分类是一个多分类问题。为将 Adaboost 算法 用于多分类问题，我们对算法做了适当的修改。修改后的算法为：

1. 基本分类器的选取

我们选取了 sklearn 中的 DecisionTreeClassifier 作为基本分类器。

2. 计算每次每次迭代训练数据集的权值分布

A. 初始化样本权重为 1/N（其中 N 是训练数据集的大小），即为训练样本的初始概率分布；

B. for m=1 to M:

a) 使用当前分布 Dm 加权的训练数据集，学习基本分类器 Gm(x)。

b) 计算基本分类器 Gm(x)在加权训练数据集上的分类误差率：

```
eps = w.dot(pred != y)
```

其中：

eps：表示第 m 个基本分类器在加权训练数据集上的训练误差

w：当前训练数据集的权值分布

pred: 第 m 个基本分类器的预测结果

y: 真实值

c) 计算基本分类器 Gm(x)的系数 alpha：

```
alpha = (np.log((1 - eps)*(k-1)) - np.log(eps)) / 2
```

这是和面向二分类问题的 AdaBoost 算法的主要不同之处。在二分类问题

中，AdaBoost 对弱分类器的要求是准确率大于 50%，但是在多分类问题中，这一要求难以得到保证。能够保证的是弱分类器的准确率优于随机预测的准确率，即 1/k，k 为类别数量。此时，若修改 alpha 的计算式，即$\frac{1}{2}\log(\frac{1-eps}{eps})$，将可能小于 0，不合理。在修改之后，只要弱分类器的准确率大于 1/k，alpha 的值就会大于 0，意味着分类误差率越小的基本分类器在最终分类器中的作用越大。

d) 更新样本权重，降低上一个基本分类器分为正例的权重，增加反例的权重，实际上是将上一个基本分类器的训练结果降为随机猜：

正例样本权重之和 / 总的样本权重之和=1/k。

具体如下：

正例：
```
w[i] = w[i]*np.exp(-alpha)
```
反例：
```
w[i] = w[i]*np.exp(alpha)
```

由于权值之和为 1，所以再进行归一化：
```
w = w / w.sum()
```

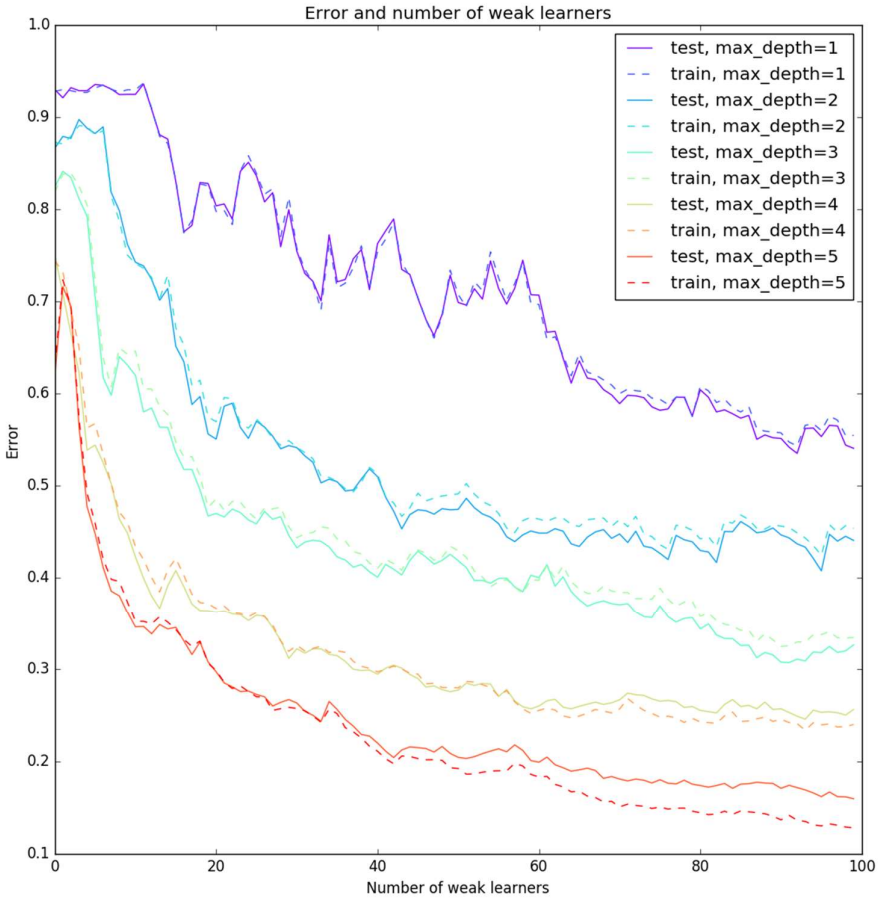
最终我们可以获得 m 个弱分类器以及其相应的权值。

3. 基本分类器合成

对所获的的 m 个基本分类器，在训练集上分别进行预测，预测结果用相应 alpha 值加权，再对这 m 组值按照预测结果相加，取最大值为最终分类器的预测结果。

四. 实验结果

我们选择的弱分类器是 sklearn 的 DecisionTreeClassifier，该分类器的一个关键参数是树的最大深度 max_depth。调整 max_depth 和 AdaBoost 迭代的次数（弱分类器的数量），我们得到了下面的折线图：



从上图可以看出，随着迭代次数（弱分类器个数）的增加，在训练集和测试集上的 error rate 整体上都呈下降趋势——在只使用 1 个或很少的分类器时，错误率超过了 60%；然后随着分类器个数在一定范围内增加而迅速下降，说明随着迭代次数的增加，集成方法能有效地提高模型的准确率；之后结果又逐渐趋于稳定，这可能是因为对于同一训练集而言，增加弱分类器已经将之前分类器预测错误的数据比重提升到了“饱和”的结果，各个特征和每个分类器的权重已经基本达到了收敛状态，不会再有快速显著的提升。当最大深度为 5、弱分类器的个数趋近 100 个时，对训练集和测试集的 error rate 都在 20%以下，已经得到了较好的分类效果。

五. 收获

在这次作业中，我们选取了 Letter Recognition 数据集，使用 sklearn 和 panda 包，以 DecisionTreeClassifier 作为基本分类器，通过逐次迭代提高误分类数据所占的权重的方法得到一组弱分类器；之后我们根据得到的权值将弱分类器的分类结果加权统计作为最终的分类结果，实现了多个弱分类器的集成，将 adaboost 算法扩展到多分类问题上，并比较了在弱分类器个数不同的情况下分类结果的表现情况。通过这次作业，进一步增长了数据处理、可视化和工具使用方面的经验，最重要的是，我们对分类算法和集成提升的算法有了更为深入的了解。