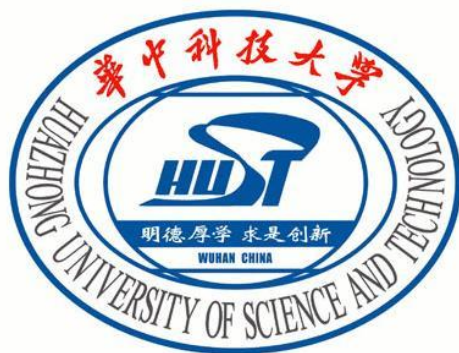


华中科技大学

计算机科学与技术学院

《机器学习》结课报告



专 业： 计算机科学与技术

班 级： 计算机科学与技术 2104 班

学 号： U202115424

姓 名： 张森磊

成 绩：

指导教师： 张腾

完成日期： 2023 年 5 月 21 日

目录

1. 实验要求.....	2
2. 算法设计与实现.....	2
2.1 数据预处理.....	2
2.2 对数几率回归.....	2
2.3 决策树桩.....	4
2.4 Adaboost 算法	4
3. 实验环境与平台.....	6
4. 结果与分析.....	6
5. 个人体会.....	7

Adaboost 算法实现

1. 实验要求

本实验要求分别实现以对数几率回归和决策树桩为基分类器的 AdaBoost 算法，输出在不同数目基分类器条件下的 10 折交叉验证的预测结果。

2. 算法设计与实现

2.1 数据预处理

(1) 归一化处理

在决策树桩分类器中，会设置一个特征的阈值，只要根据大小比较来判断类别即可，所以不同特征的数量级差别不会影响分类效果，因此不需要归一化。

在对数几率回归分类器中，特征的数量级会对分类效果产生较大的影响，过大的数量级会导致过大的权重。所以，要对各个特征做缩放，使得其数量级大致相同。

(2) k 折交叉验证

我们从.csv 文件中读取数据并进行归一化处理之后，将原始数据随机分成 K 份，每次选择 K-1 份作为训练集，剩余的 1 份作为测试集。交叉验证重复 K 次，取 K 次准确率的平均值作为最终模型的评价指标。如图所示：



图 2.1 k 折交叉验证

2.2 对数几率回归

对数几率回归模型是一种基于概率分布的非线性二分类模型，广泛应用于多种场景。

逻辑回归模型由条件概率分布 $P(y|x)$ 表示，其中 y 表示预测值， x 表示多维特征向量。当预测任务为二分类问题时，随机变量 y 的取值为 0 或 1，用 $y=0$ 来表示事件未发生，用 $y=1$ 来表示事件发生。通常使用 Sigmoid 函数来实现逻辑回归分类器，对数几率函数具备单调递增、连续以及值域是 $[0, 1]$ 等良好属性，对数几率函数如图 2.1 所示。

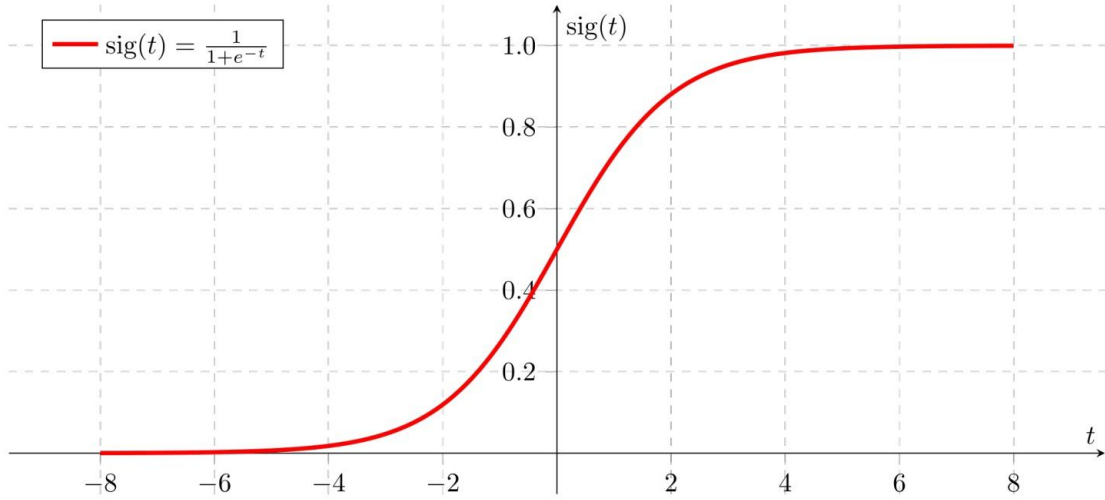


图 2.2 对数几率函数图像

对数几率回归通过式 (1) 对数几率函数将线性回归模型产生的预测值 $z = w^T x + b$ 转化为接近 0 或 1 的 y 值。得到式 (2)。

$$y = \frac{1}{1 + e^{-z}} \quad (1)$$

$$y = \frac{1}{1 + e^{-(w^T x + b)}} \quad (2)$$

对式 (2) 进行变形，最终可以得到预测函数式 (3)，式 (4)。

$$P(y = 1|x) = \frac{e^{-(w^T x + b)}}{1 + e^{-(w^T x + b)}} \quad (3)$$

$$P(y = 0|x) = \frac{1}{1 + e^{-(w^T x + b)}} \quad (4)$$

逻辑回归中一般使用交叉熵损失函数，损失函数如式 (5)。

$$L = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})] \quad (5)$$

根据式 (5)，使用梯度下降法最小化损失函数，可以得到权值向量 w 的

最优参数。由式（6）可得，偏置项 b 可以作为一个新的特征，加入训练集中。
训练公式如式（7）

$$t = \sum_{i=1}^n \omega_i x_i + b = \sum_{i=0}^n \omega_i x_i \quad (6)$$

$$\omega_i = \omega_i - \eta w(\hat{y} - y)x_i \quad (7)$$

2.3 决策树桩

决策树桩为单层的决策树。只根据数据集中的一个特征来进行分类，不考虑其他特征。

- 从树（数据结构）的观点来看，它由根节点（root）与叶子节点（leaves）直接相连。用作分类器的决策树桩的叶子节点也就意味着最终的分类结果。
- 从实际意义来看，决策树桩根据一个属性的一个判断就决定了最终的分类结果，这体现的是单一简单的规则（或叫特征）在起作用。

决策树桩具有如下形式：

$$f(\mathbf{x}) = s(x_k < threshold)$$

其中 $f(\mathbf{x})$ 是分类结果， x_k 是选定的特征， $threshold$ 是该特征对应的阈值， s 可以是 1 或 -1，以便处理小于阈值为 1 的情况。通常采用暴力的方法选取特征并确定阈值，即先遍历所有特征，再遍历该特征的各个不同取值求得最优的特征及其阈值。

由于决策树桩比随机分类稍微好一些，而且计算比较简单，所以它通常作集成学习的基分类器，而不单独使用。

2.4 Adaboost 算法

Adaboost 是一种集成学习的方法，它的核心思想是利用多个弱分类器来构造一个强分类器。所谓弱分类器，就是比随机猜测略好一点的分类器，例如决策树桩。所谓强分类器，就是能够在训练数据上达到很高准确率的分器。

Adaboost 的优点是它可以自动地调整每个弱分类器的权重，并且可以有效地降低过拟合的风险。它的具体流程如下：

- (1) 初始化每个样本的权重为 $w_i = 1/n$
- (2) 根据式 (8) (9) (10) (11) 分别计算每个分类器 $g^{(m)}$ 对应下的 $err^{(m)}$, 分类器权重 $\alpha^{(m)}$ 和样本权重及标准化 w_i

$$err^{(m)} = \frac{\sum_{i=1}^n w_i \mathbb{I}(y_i \neq g^{(m)}(x_i))}{\sum_{i=1}^n w_i} \quad (8)$$

$$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} \quad (9)$$

$$w_i \leftarrow w_i \exp\left(\alpha^{(m)} \cdot \mathbb{I}(y_i \neq g^{(m)}(x_i))\right), i = 1, 2, \dots, n \quad (10)$$

$$w_i \leftarrow \frac{w_i}{\sum_{j=1}^n w_j} \quad (11)$$

- (3) 将训练得到的 M 个分类器根据式 (12) 中的方式进行集成, 然后输出得到样本的预测结果。

$$C(x) = \operatorname{argmax}_k \sum_{m=1}^M \alpha^{(m)} \mathbb{I}(g^{(m)}(x_i) = k) \quad (12)$$

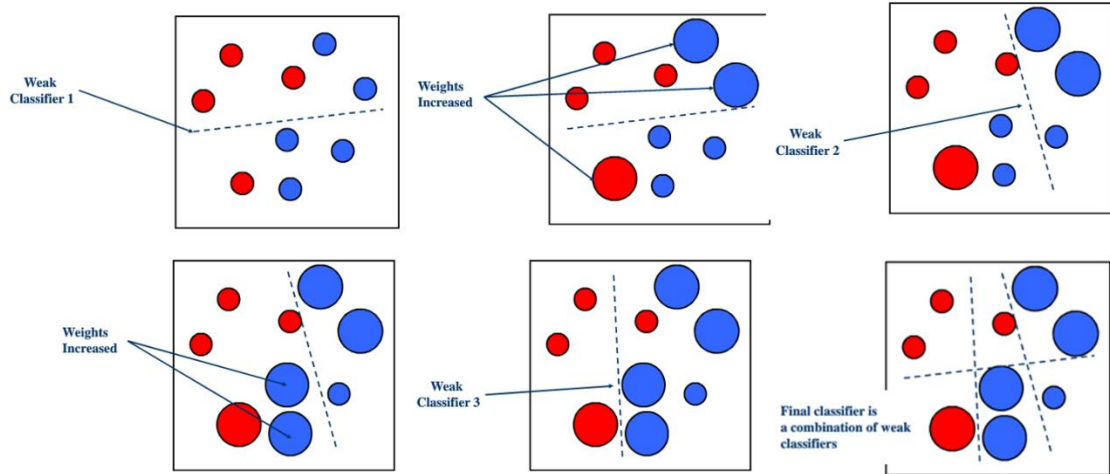


图 2.3 AdaBoost 算法训练过程

伪代码如下：

Algorithm 1: AdaBoost 算法

输入: 训练数据集 $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i \in [m]}$, 基学习算法 \mathcal{L} , 迭代轮数 T

```
1  $\mathcal{D}_1(i) \leftarrow 1/m$ ; // 初始化权重分布
2 for  $t \leftarrow 1$  to  $T$  do
3    $h_t \leftarrow \mathcal{L}(\mathcal{S}, \mathcal{D}_t)$ ; // 在  $\mathcal{S}$  上以权重  $\mathcal{D}_t$  训练  $h_t$ 
4    $\epsilon_t \leftarrow \mathbb{P}_{(\mathbf{x} \sim \mathcal{D}_t, y)} \mathbb{I}(h_t(\mathbf{x}) \neq y)$ ; // 计算加权错误率
5   if  $\epsilon_t > 0.5$  then break; // 若基分类器比随机猜测还差则中止算法
6    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$ ; // 计算  $h_t$  的权重系数
7   for  $i \leftarrow 1$  to  $m$  do
8     if  $h_t(\mathbf{x}_i) = y_i$  then
9        $\mathcal{D}_t(i) \leftarrow \mathcal{D}_t(i) \exp(-\alpha_t)$ 
10    else
11       $\mathcal{D}_t(i) \leftarrow \mathcal{D}_t(i) \exp(\alpha_t)$ 
12    end
13  end
14   $s \leftarrow \sum_{i \in [m]} \mathcal{D}_t(i)$ ;
15  for  $i \leftarrow 1$  to  $m$  do  $\mathcal{D}_t(i) \leftarrow \mathcal{D}_t(i)/s$ ; // 归一化权重
16 end
```

输出: $\text{sign}(\sum_{t \in [T]} \alpha_t h_t(\mathbf{x}))$

3. 实验环境与平台

处理器: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz

内存: 内存 16.0 GB

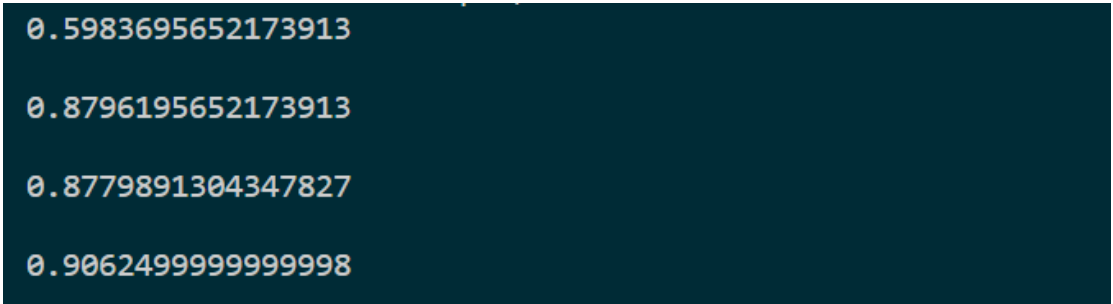
显卡: NVIDIA GeForce GTX 1650

系统: Windows 11

Python 版本: 3.6.5

4. 结果与分析

运行后能够成功将预测结果输出到对应的.csv 文件。evaluate.py 测试结果如下:



```
0.5983695652173913
0.8796195652173913
0.8779891304347827
0.9062499999999998
```

图 4.1 以对数几率回归作基分类器



图 4.2 以决策树桩进行基分类器

从准确率结果可以看到，基分类器数目对准确率结果的影响基本符合理论预测：分类器数目较少时准确率只比随机稍好，而后随着分类器数目增长，准确率大幅提升。另外，决策树桩在分类器较少时就能达到不错的准确率，但是上限低于对数几率回归。

5. 个人体会

1. 体会

《机器学习》是一门非常有挑战性和实用性的课程，通过学习这门课程，我深入了解了机器学习的基本概念、算法和应用。

- 机器学习的基本概念：在学习机器学习之前，我并不清楚什么是机器学习、机器学习的应用场景和基本概念。通过学习这门课程，我了解了机器学习的基本概念，如监督学习、无监督学习、半监督学习、强化学习等，以及它们在实际应用中的作用。
- 机器学习的算法：在学习这门课程中，我学习了很多机器学习的算法，如线性回归、逻辑回归、决策树、支持向量机、朴素贝叶斯、神经网络等。通过学习这些算法，我能够更好地理解机器学习的原理和应用。
- 机器学习的应用：机器学习在现实世界中有广泛的应用，如自然语言处理、图像识别、智能推荐等。通过学习这门课程，我了解了机器学习在不同领域的应用，并且能够使用机器学习算法解决实际问题。
- 编程实践：在学习这门课程中，我通过编写代码实现机器学习算法，如线性回归、逻辑回归、决策树等。通过实践，我更深入地理解了机器学习算法的原理和实现。

总的来说，学习《机器学习》这门课程是一次非常有价值的经历。通过学习这门课程，我深入了解了机器学习的基本概念、算法和应用，同时也提高了自己的编

程实践能力。

2. 回答实验问题

(1) 对 Adaboost 算法有什么新的认识？

Adaboost 算法是一种集成学习方法，具有以下优点和缺点：

优点：

- Adaboost 算法可以组合多个弱分类器，得到一个准确率较高的强分类器，可以有效提高分类准确率。
- Adaboost 算法可以针对错误分类的样本增加权值，对于正确分类的样本减小权值，从而更加精细地分类数据，具有较好的鲁棒性。
- Adaboost 算法对于噪声数据具有一定的鲁棒性，可以有效降低噪声对分类结果的影响。
- Adaboost 算法对于大规模数据集也具有较好的适应性，可以快速训练出一个准确率较高的分类器。

缺点：

- Adaboost 算法对于异常值敏感，如果数据集中存在异常值，可能会导致分类结果出现较大偏差。
- Adaboost 算法的计算复杂度较高，需要多次迭代训练弱分类器，因此在处理大规模数据集时可能会出现效率问题。
- Adaboost 算法对于数据集中的类别分布不均衡的情况，可能会导致分类结果出现偏差。

综上所述，Adaboost 算法是一种非常有效的分类算法，可以帮助我们解决很多实际问题。但是，在使用 Adaboost 算法时需要注意异常值和数据分布的问题，同时需要权衡计算复杂度和分类准确率等因素。

(2) 关于基分类器类型、超参数设置对最终模型性能的影响，你有何发现？

Adaboost 算法的超参数主要有两个：基分类器的数量和基分类器的类型。这两个超参数会影响 Adaboost 算法的性能和效率。

- 基分类器的数量越多，Adaboost 算法的分类准确率越高，但也可能导致过拟合和计算开销增大。因此，需要通过交叉验证或者其他方法来确定一个合适的基分类器数量，使得 Adaboost 算法能够在偏差和方差之间达到一个平衡。

- 基分类器的类型不同，Adaboost 算法的表现也会不同。一般来说，基分类器应该是简单而弱的，即准确率略高于随机猜测的分类器，例如决策树桩或者对数几率回归。如果基分类器过于复杂或者强大，例如深度神经网络，那么 Adaboost 算法可能会降低分类效果或者失去提升作用。

对于本实验使用的两种弱分类器：

对数几率回归作为基分类器时，需要更多的迭代次数才能达到较好的效果，而决策树桩作为基分类器时，通常只需要少量的迭代次数就能达到较好的效果。

对数几率回归作为基分类器时，对噪声数据和异常值更加健壮，而决策树桩作为基分类器时，对噪声数据和异常值更加敏感。