人工智能实验报告 LAB3

（2021学年秋季学期）

课程名称：Artificial Intelligence

教学班级	计科2班	专业（方向）	计算机科学与技术
学号	19335174	姓名	施天予

PLA感知机算法

一、 实验题目

PLA感知机算法

二、 实验内容

1、算法原理

感知机模型

感知机是一种线性分类模型，属于判别模型。假设输入空间是$x \in R^n$，输出空间是$y \in \{+1, -1\}$，由输入空间到输出空间的函数$f(x) = sign(w \cdot x + b)$称为感知机，w和b是模型的参数，w称为权值向量，b称为偏置，sign是符号函数，即

$$sign(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

线性方程$w \cdot x + b$对应着特征空间的一个分离超平面，将特征空间分成两个部分，位于两部分的点分别对应正样本和负样本。感知机利用损失函数对模型参量w和b进行更新学习。将误分类点(x_i, y_i)到分离超平面的距离定义为：

$$\frac{1}{||w||}||w \cdot x + b|| = -\frac{1}{||w||}(w \cdot x_i + b)y_i$$

假设误分类点的集合为M，那么损失函数的梯度为：

$$L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

可以采用随机梯度下降的方式来优化损失函数。w和b的损失函数梯度分别为：

$$\nabla_w L(w,b) = -\sum_{x_i \in M} y_i x_i$$
$$\nabla_b L(w,b) = -\sum_{x_i \in M} y_i$$

随机选取一个误分类点，对参数进行更新，其中 η 表示学习率：

$$w = w + \eta y_i x_i$$
$$b = b + \eta y_i$$

感知机算法

1. 随机初始化参数 w 和 b ，设置学习率η
2. 在训练集中选取数据(x_i, y_i)
3. 如果$y_i(w \cdot x + b) \le 0$

$$w \leftarrow w + \eta y_i x_i$$
$$b \leftarrow b + \eta y_i$$

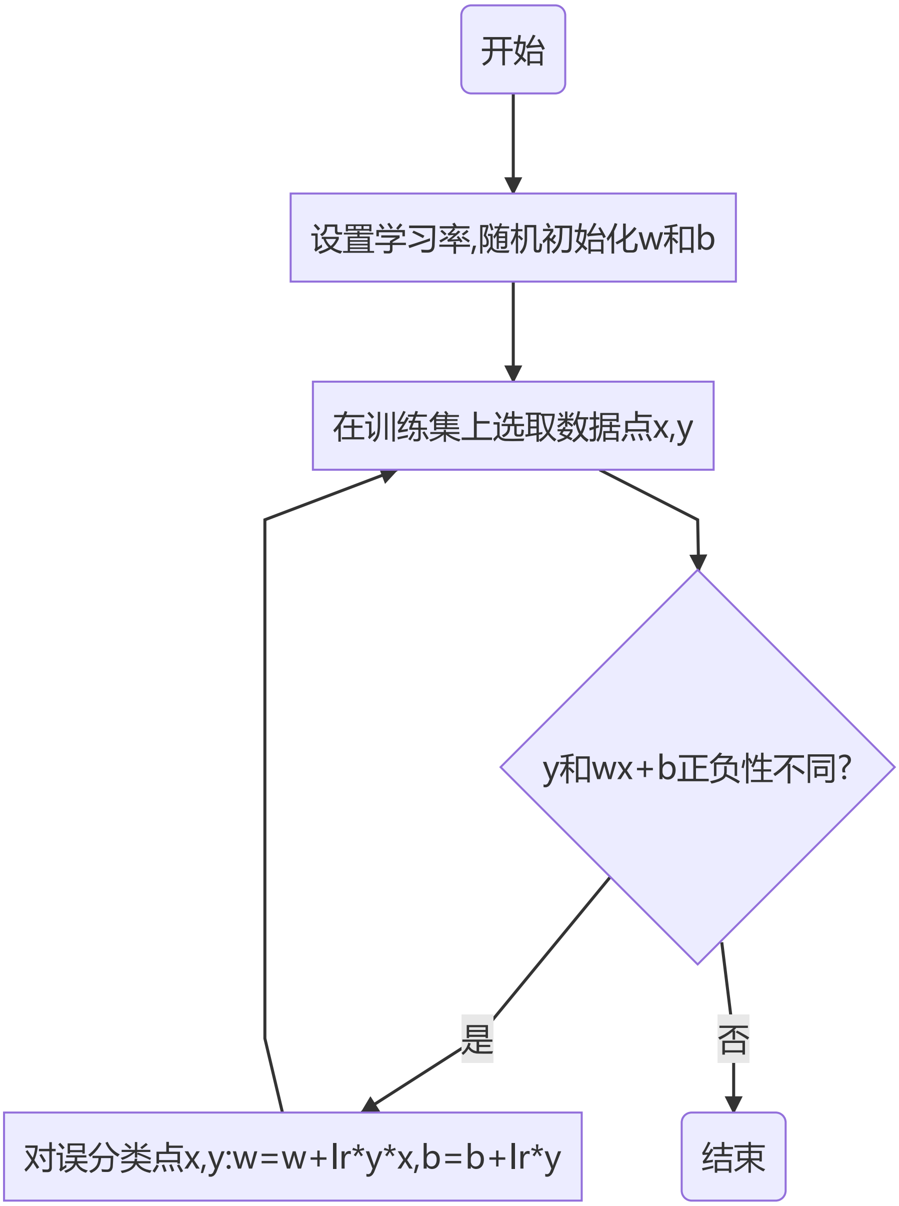
4. 转至步骤2，直到训练集中没有误分类点或者到达固定迭代次数

数据集的可分性

对于线性可分的数据集，存在一个分离超平面，总能将正、负样本完全正确划分到平面的两侧

对于线性不可分的数据集，可以设置最大迭代次数或者引入核函数

2、流程图和伪代码




```
Function PLA
Input: data_set, epoches, learning_rate/*数据集，迭代次数，学习率*/
Output: w,b
    n := 训练集的总特征数(去掉label)
    w := 大小为n的全零数组
    b := 0
    dataset1 := dataset的numpy形式
    for i in range(epoches):
        flag := true
        for data in data_set:
            x := data对应的特征向量
            y := x*w +b
            if y和训练集的label正负性不同 then
                flag := False
                w := w + learning_rate * label * x
                b := b + learning_rate * label
                break    /*检测完一个误分类点*/
            end if
```

```
        end for
        if flag=True then
            break   /*没有误分类点，停止迭代*/
        end if
    end for
    return w, b
```

3、关键代码展示

PLA算法

对于训练集的每个样例，如果分错就更新参数w，如果都没分错就停止迭代

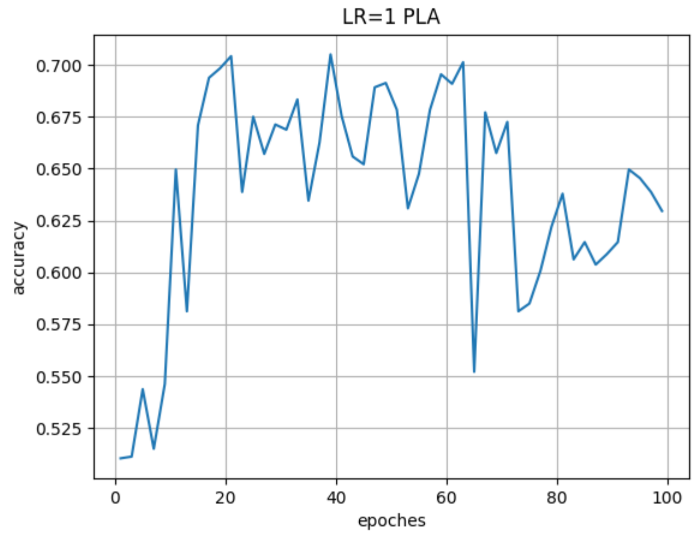
```python
def PLA(train_set, epoches, LR):
    w = np.zeros(len(train_set[0])-1)
    b = 0
    for _ in range(epoches):
        flag = True
        for data in train_set:
            x = data[:-1]
            y = np.dot(x, w) + b
            if np.sign(y) != data[-1]:
                flag = False
                w += LR * data[-1] * x
                b += LR * data[-1]
                break
        if flag == True:    # 没有分错，直接跳出循环
            break
    return w,b
```

三、 实验结果及分析

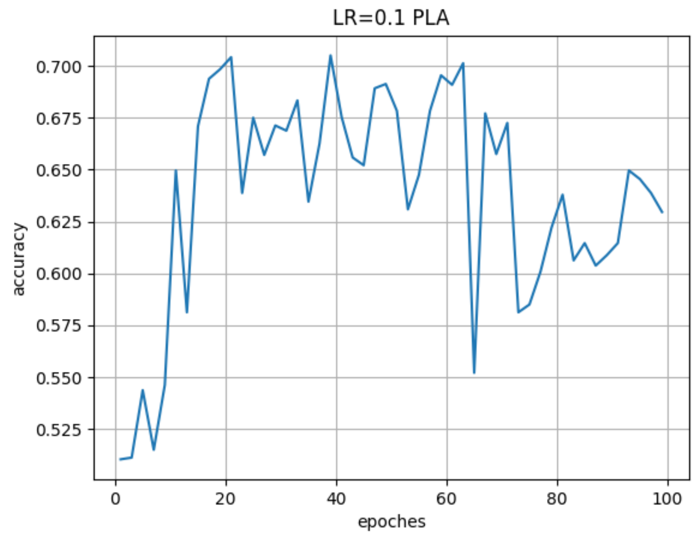
1、实验结果展示示例

将训练集和验证集分成73开，每次固定学习率，观察不同迭代次数的准确率

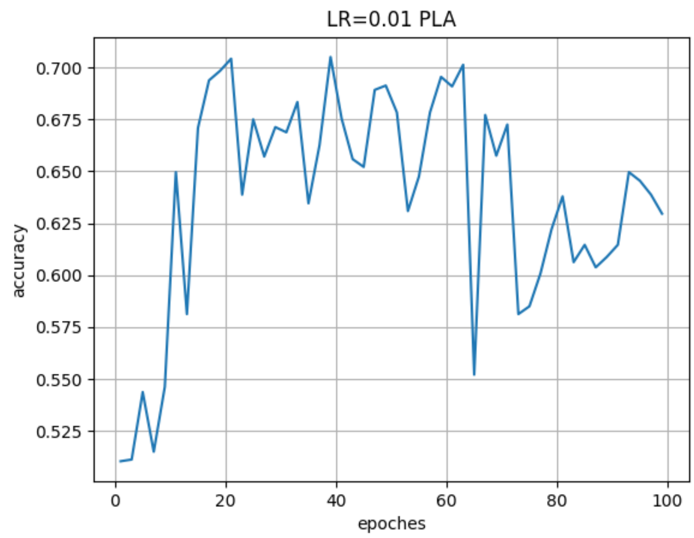
learning rate = 1



learning rate = 0.1



learning_rate = 0.01



可以看到对于不同的学习率，PLA算法得到的结果几乎相同，每幅图都十分相似。迭代次数决定了模型是否精确地贴合了训练集的分布。在每次w和b更新后输出二者的值，如果二者几乎没有变化则说明参数已经收敛，更多的迭代次数也没用。而迭代次数过少则模型会欠拟合，显然不利于模型的精度。因为数据集线性不可分，所以PLA算法无法找到最优解，每次迭代每次迭代更新后，准确率都会波动。

2、评测指标展示及分析

	学习率	迭代次数	准确率
初始	1	21	70.417%
优化1	0.1	39	70.5%
优化2	0.01	39	70.5%
最优效果	0.01	39	70.5%

可以看出，在PLA算法下学习率对准确率几乎是没有影响。我认为，是因为在每次遇到误分类点之后更新权值，我写的是重新迭代时又从头开始找误分类点，之后的多数样本都训练不到。少量的样本的学习导致了欠拟合，不管学习率如何，模型本身就不能很好地反映样本的分布。总而言之，PLA算法对于线形不可分的数据集表现并不是很出色。

LOGISTIC逻辑回归

一、 实验题目

LOGISTIC逻辑回归

二、 实验内容

1、算法原理

二项逻辑回归

逻辑回归也是一种分类问题，假设输入空间是$x \in R^n$，输出空间是样本属于某个类别$y \in \{0，1\}$的概率。

概率预测使用的是logistic函数：

$$F(x) = \frac{1}{1 + e^{-x}}$$

函数单调递增，取值为$(0,+1)$，当x值为$-\infty$则函数值为0，x值为$+\infty$则函数值为1，0到1之间的取值正好能作为概率。

二项逻辑回归模型是如下的条件概率分布：

$$P(y=1|x)=\frac{\exp(w\cdot x+b)}{1+\exp(w\cdot x+b)}$$
$$P(y=0|x)=\frac{1}{1+\exp(w\cdot x+b)}$$

如果$w=(w^T,b)^T$, $x=(x^T,1)^T$

$$P(y=1|x)=\frac{\exp(w\cdot x)}{1+\exp(w\cdot x)}=\frac{1}{1+\exp(-w\cdot x)}$$
$$P(y=0|x)=1-P(y=1|x)=\frac{1}{1+\exp(w\cdot x)}$$

模型参数估计

令y=1的概率函数为$\pi(x)$，则样品x属于某个类别y的概率表示为：

$$f(x)=P(y|x)=\pi(x)^y(1-\pi(x))^{1-y}$$

$f(x)$称为似然函数。整个训练集上，似然函数为：

$$\Pi_{i=1}^N\pi(x_i)^{y_i}(1-\pi(x_i))^{1-y_i}$$

为了减小不确定性，则需要对似然函数值取得最大值。方便计算可以对似然函数取对数：

$$L(w)=\sum_{i=1}^N[y_i\log\pi(x_i)+(1-y_i)\log(1-\pi(x_i))]$$
$$=\sum_{i=1}^N[y_i(w\cdot x_i)-\log(1+e^{w\cdot x_i})]$$

对$L(w)$取负值作为逻辑回归的损失函数，并使用梯度下降法进行优化。对$-L(w)$对w求导得损失函数的梯度：

$$-\nabla_w L(w)=-\sum_{i=1}^N[y_i-\pi(x_i)]x_i$$

更新参数 w 则使用以下公式：

$$w=w+\eta*\sum_{i=1}^N[y_i-\pi(x_i)]x_i$$

多项逻辑回归

逻辑回归也可用于多类分类，假设离散型随机变量Y的取值集合是{1,2,⋯,K}，那么多项逻辑回归模型是

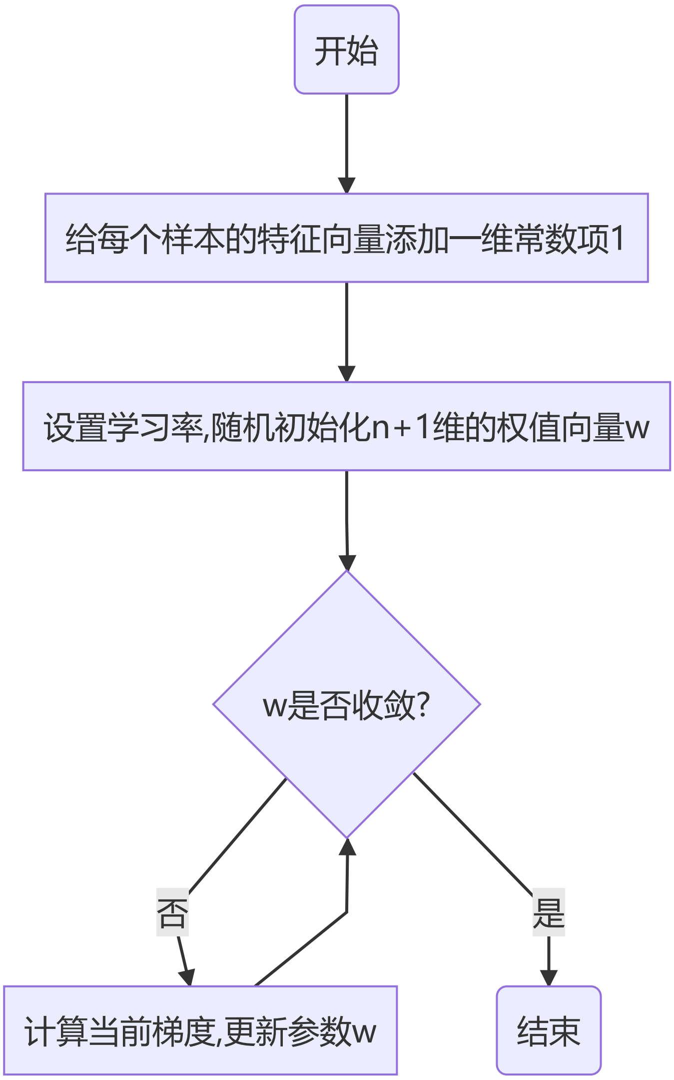
$$P(Y=k|X)=\frac{exp(w_k\cdot x)}{1+\sum_{k=1}^{K-1}exp(w_k+x)},k=1,2,...,K-1$$
$$P(Y=K|X)=\frac{1}{1+\sum_{k=1}^{K-1}exp(w_k+x)}$$

其中，$x\in R^{n+1}$，$w_k\in R^{n+1}$

逻辑回归算法

1. 先给每一个样本特征向量加一个`1`，即$x=(x^T,1)^T$。
2. 设置学习率，将权值向量w和偏置b合并为新的权值向量$w=(w^T,b)^T$，并随机初始化
3. 计算当前梯度，并对权值向量 w 进行更新
4. 重复步骤3，直到所有的训练样本都梯度收敛（小于阈值）或到达固定迭代次数
5. 用训练得到的权值向量 w 预测新数据集上的标签。

2、流程图和伪代码




```
Function logistic
input: data_set, epoches, learning_rate/*数据集，迭代次数，学习率*/
output: w
    n := 训练集的总特征数(包括label)
    w := 大小为n的全零数组
    for i in range(epoches):
        x := data_set的前n-1列 + “1”最后一列
        y := data_set的最后一列label值
        temp := y - pi(x)   /*pi()是logistic函数*/
        gradient = x和temp的点积
        w1 = w + learning_rate * gradient
        d := w和w1之间的梯度模长
        if d <= 0.0001 then /*收敛则停止迭代*/
            break
        end if
        w := w1
    end for
    return w
```

3、关键代码展示

logistic函数

```python
def pi(w, x):
    w = w.reshape((-1, 1))
    res = 1 / (1 + np.exp(-np.dot(x, w)))
    return res
```

逻辑回归

利用公式计算更新w，如果梯度变化小于阈值，则停止迭代

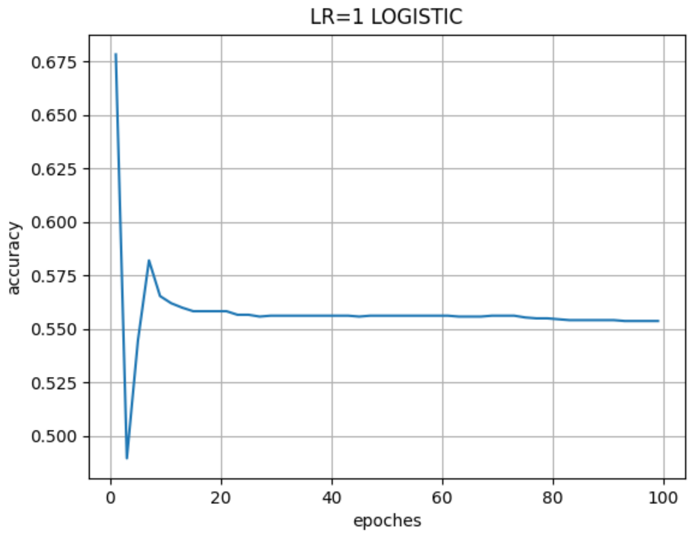
```python
def logistic(train_set, epoches, LR):
    w = np.zeros(len(train_set[0]))
    h = np.ones(train_set.shape[0])
    # 给train_set末尾加一列1
    train_set = np.insert(train_set, train_set.shape[1]-1, values=h, axis=1)
    for _ in range(epoches):
        gradient = np.empty([]) # 损失函数的梯度
        x = train_set[:, :-1]   # 40个属性和'1'
        y = train_set[:, -1]    # 标签
        y = y.reshape((-1, 1))
        temp = y - pi(w, x)     # 用logistic函数计算
        for i in range(x.shape[1]-1):
            t = np.dot(x[:, i], temp)
            gradient = np.append(gradient, -t)
        w1 = w - LR * gradient
        d = np.linalg.norm(w1 - w)  # 求梯度的模长
        if d <= 0.0001:     # 收敛则停止迭代
            break
        w = w1  # 更新参数
    return w
```

三、 实验结果及分析

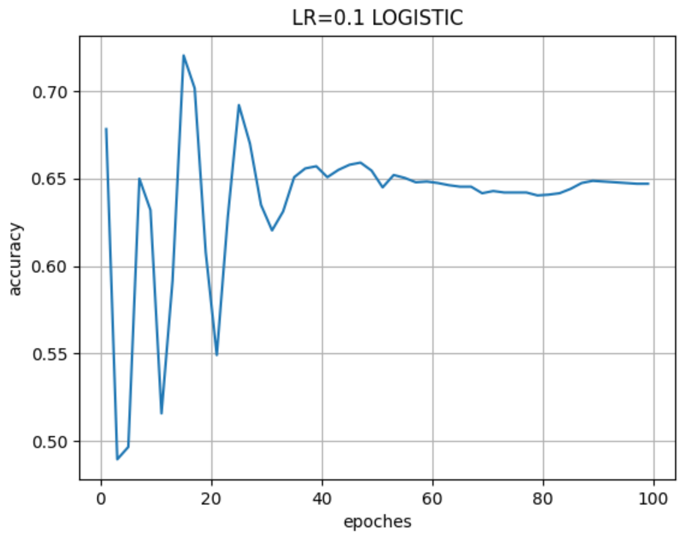
1、实验结果展示示例

将训练集和验证集分成73开，每次固定学习率，观察不同迭代次数的准确率

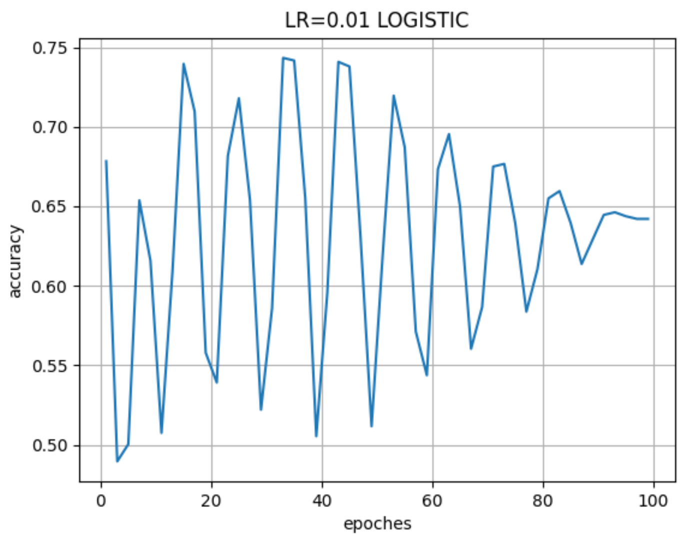
learning rate = 1



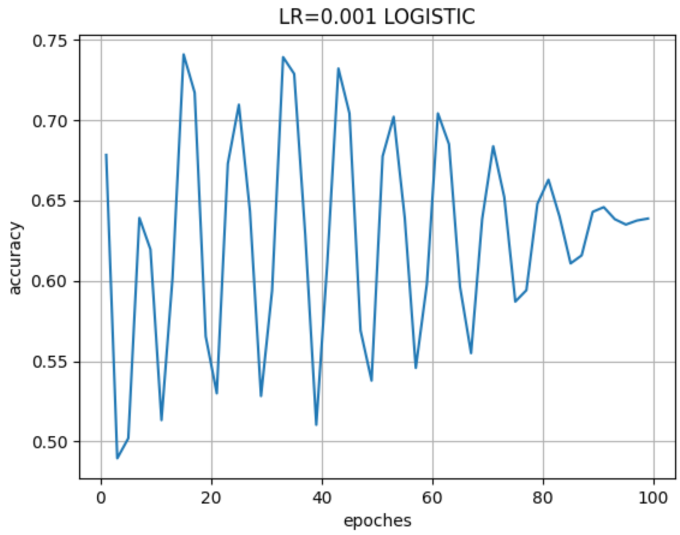
learning rate = 0.1



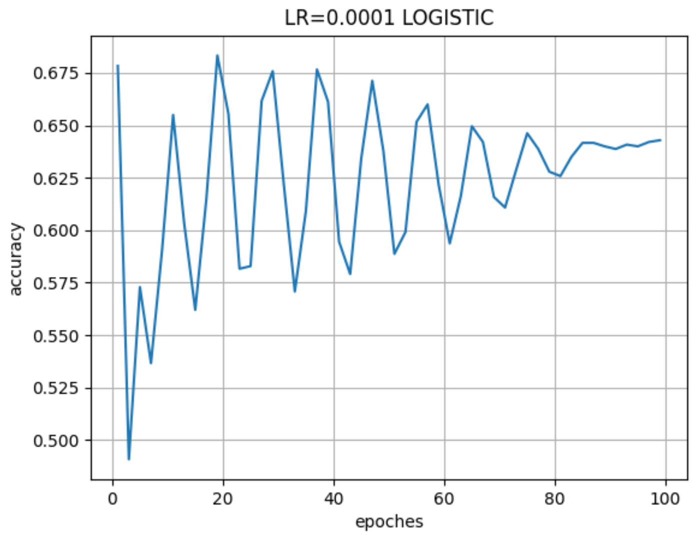
learning rate = 0.01



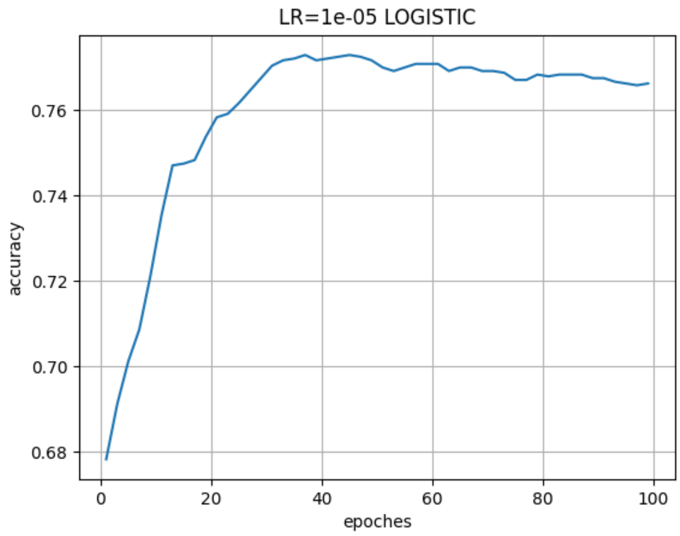
learning rate = 0.001



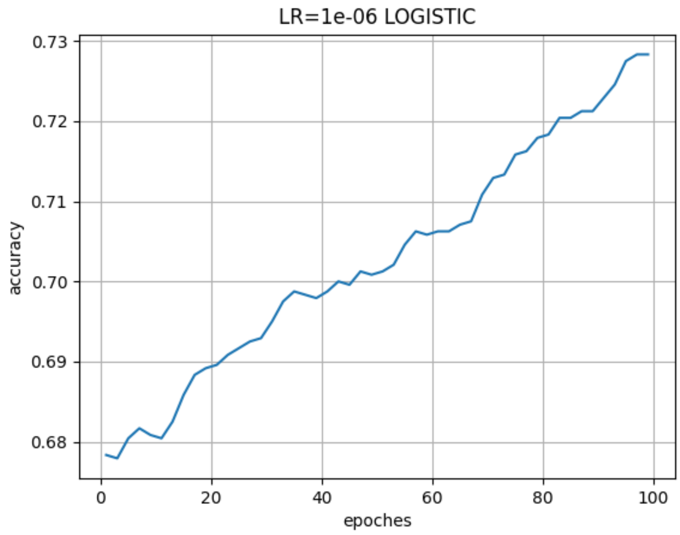
learning rate = 0.0001



learning rate = 0.00001



learning rate = 0.000001



可以发现，当学习率比较大时，准确率随着迭代次数的增加一直波动上升，但很难达到最优值。当学习率调整到10^{-5}或10^{-6}时，准确率稳步上升，并在一定迭代次数时保持平稳。学习率为10^{-5}时到达平稳比较快，而学习率为10^{-6}时要趋于平稳需要较大的迭代次数。由此可以看出，学习率对于一个模型的训练十分关键。

2、评测指标展示及分析

	学习率	迭代次数	准确率
初始	1	1	67.83%
优化1	0.1	15	72.04%
优化2	0.01	33	74.33%
优化3	0.001	15	74.08%
优化4	0.0001	37	67.67%
优化5	0.00001	45	77.29%
优化6	0.000001	97	72.83%
最优效果	0.00001	45	77.29%

由上表可知，当学习率为10^{-5}时，逻辑回归算法能快速到达收敛，且准确率最高。当学习率较大时，因为每次更新参数变化量都比较大，所以参数w在一定范围内波动，模型无法收敛，准确率也很难提高；而当学习率较小时，因为每次更新参数变化量都比较小，所以参数w变化很缓慢，需要不断增加迭代次数才能使模型收敛，准确率虽一直提升但需要的时间太长。总而言之，对于这次的逻辑回归模型，我认为学习率为10^{-5}左右是一个非常不错的选择，既有较高的准确率，训练时间也不会太长。

四、思考题

1. 随机梯度下降与批量梯度下降各自的优缺点?

	随机梯度下降	批量梯度下降
优点	每一轮参数的更新速度大大加快	一次迭代对所有样本进行计算，实现了并行;目标函数为凸函数时，一定能达到最优解
缺点	无法保证线性收敛；可能收敛到局部最优；不易于并行	只有当目标函数为凸函数时,梯度下降算法才能保证达到全局最优解；因为要在全部训练数据上最小化损失，计算时间太长

2. 不同的学习率η对模型收敛有何影响？从收敛速度和是否收敛两方面来回答。

   在较小的学习率下，每次参数更新值较小，从而减缓了收敛速度。但是如果迭代次数足够多，就能够在允许范围内收敛。

   在较大的学习率下，每次参数更新值较大，收敛速度也就更快。但是学习率过大，参数值更新过大时，在损失函数的极小值点附近，参数决定的点会左右横跳，难以取得极小值点，从而难以收敛。

3. 使用梯度的模长是否为零作为梯度下降的收敛终止条件是否合适，为什么？一般如何判断模型收敛?

   不合适。模型不可能完美拟合训练数据，梯度不可能完全为0，这样的收敛条件难以满足。收敛条件可以设置为梯度的模长小于某个设定好的阈值。当梯度模长较小时，参数更新变化不大，基本可以判断收敛。