人工智能实验三

18340146 计算机科学与技术 宋渝杰

任务一：

实现感知机和基于批梯度下降的逻辑回归，分别提交一份代码

本次的实验数据是 train.csv，前 40 列表示特征，最后一列表示标签 (0 或 1)

请自行分好训练验证集（在报告里说明怎么分的）， 评测指标为验证集上的准确率。

因此，本次实验的任务为：对数据 car_train.csv 分为训练集和验证集，对训练集实现感知机和逻辑回归

算法原理：

数据集的分类：选取前 t 条数据为训练集，其余 8000-t 条数据为验证集（ t 为可变参数）

感知机（PLA）：感知机针对二分类问题，输入是样本的特征向量 $x \in R^n$ ，输出是样本的类别 $y \in \{+1, -1\}$

感知机可以表示为：

$$f(x) = \text{sign}(w * x + b)$$

其中 $\text{sign}(x)$ 为符号函数，w 和 b 为模型参数，$w \in R^n$ 为权值向量，$b \in R$ 为偏置值，通过训练学习得来

损失函数：所有误分类点到分类超平面的距离和：

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

梯度：对上述公式求偏导，可得损失函数的梯度：

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

根据**随机梯度下降**法，选择一个误分类点 (x_i, y_i) 对参数进行更新，其中 η 表示学习率：

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

值得注意的是，在这里 η 的取值只会影响 w 和 b 的具体数值，并不会影响分类的过程以及最终验证的结果。证明如下：

假设 w 和 b 均初始化为 0，则根据上述更新公式，每次更新之后的 w 和 b 均为 η 的倍数，则 $w * x + b$ 为 η 的倍数。又 $f(x) = \text{sign}(w * x + b)$，$\eta$ 的取值不影响 $f(x)$ 的值，因此最终验证的结果相同。

不断重复更新步骤，直至训练集中没有误分类点或达到最大迭代次数，则停止迭代，此时的 w 和 b 为最终的模型参数。

训练：总结上述训练过程，主要分为以下三个步骤：

1. 初始化 w 和 b 为 0，随意设置学习率 η，读取训练集，并将标签为 0 的数据更改其标签为 -1
2. 选取一个误分类点 (x_i, y_i)，即 $y_i(w * x_i + b) \leq 0$，对参数进行上述更新操作
3. 重复步骤 2，当训练集中没有误分类点或达到最大迭代次数时，停止迭代，得到最终参数 w 和 b

验证：对于验证集的每一条数据，通过函数 $f(x) = \text{sign}(w * x + b)$ 预测样本类别，与其真实值进行对比，最终计算其正确率。

逻辑回归（LR）：同样针对二分类问题，输入是样本的特征向量 $x \in R^n$ ，输出是样本属于某个类别 $y \in \{0, 1\}$ 的概率

逻辑回归可以表示为：

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

$$\pi(x) = \frac{1}{1 + \exp(-w * x)}$$

其中 $f(x, y)$ 为某个样本 x 属于某个类别 y 的概率，特征向量 x 添加一维常数项 1，$w \in R^{n+1}$ 为权值向量，通过训练学习得来

似然函数：$\prod_{i=1}^{n} (\pi(x_i))^{y_i} (1 - \pi(x_i))^{1-y_i}$，考虑到函数求积难度偏大，取对数似然函数：

$$L(w) = \sum_{i=1}^{n} (y_i (w * x_i) - \log(1 + \exp(w * x_i)))$$

梯度：对似然函数 $L(w)$ 取负，为**损失函数**，并对其求偏导：

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

根据**批梯度下降**法，对参数进行更新，其中 η 表示学习率：

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

不断重复更新步骤，直至满足一定的收敛条件（此处设置为损失函数更新前后差别不超过 10^{-2}）或达到最大迭代次数，则停止迭代，此时的 w 为最终的模型参数。

训练：总结上述训练过程，主要分为以下三个步骤：

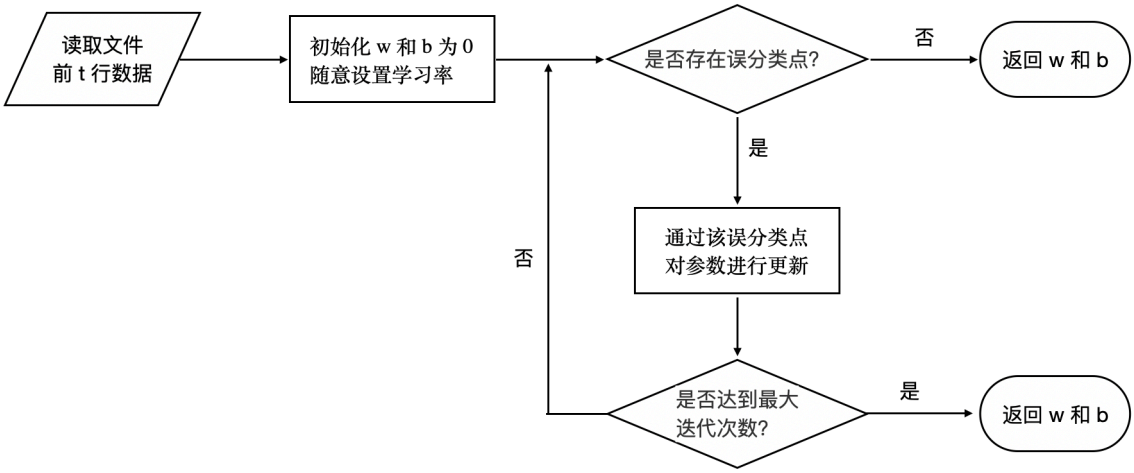
1. 初始化 w 和 b 为 0，设置学习率 η，读取训练集，并将每个样本的特征向量添加一维常数项 1，即 $x = (x^T, 1)^T$
2. 计算当前梯度，并对参数进行上述更新操作
3. 重复步骤 2，直至满足一定的收敛条件（此处设置为损失函数更新前后差别不超过 10^{-2}）或达到最大迭代次数，则停止迭代，得到最终参数 w

验证：对于验证集的每一条数据，通过函数 $f(x) = \pi(x)^y (1 - \pi(x))^{1-y}$ 预测样本属于 0 和 1 的概率，取其概率更高的为预测标签，与其真实值进行对比，最终计算其正确率。

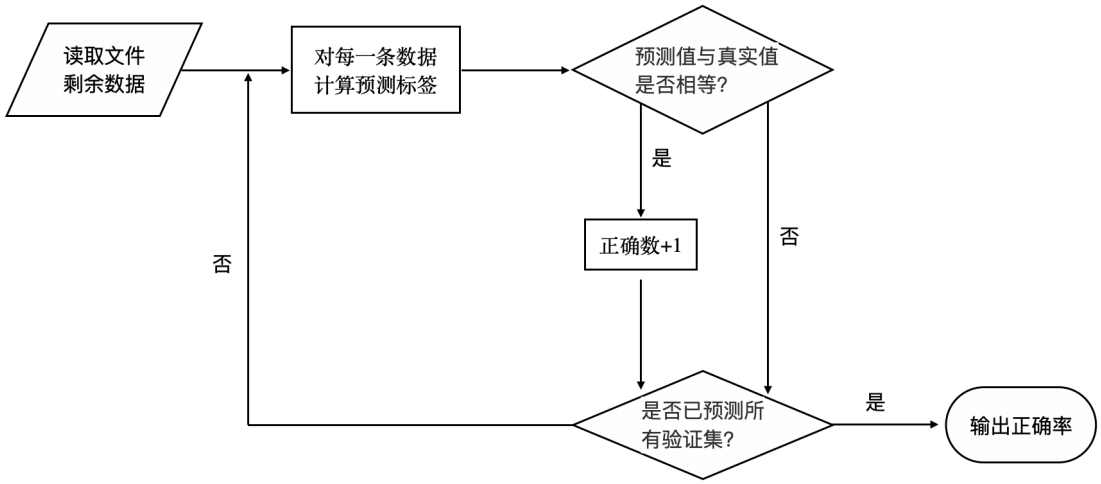
伪代码/流程图：

感知机（PLA）：

训练：

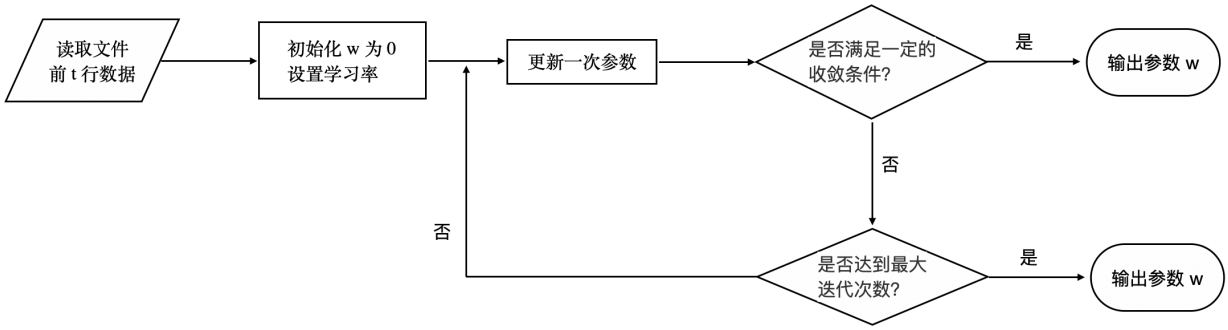


验证：

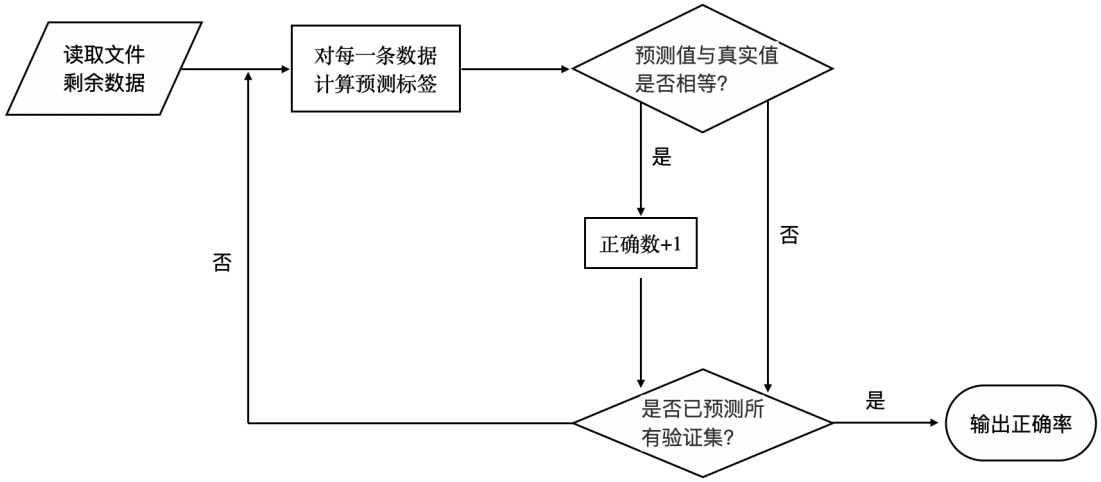


逻辑回归（LR）：

训练：



验证：



代码展示：

感知机（PLA）：下面仅展示关键代码，全部代码请移步 lab3_pla.py 文件

读取测试集和验证集：

```
def read(f,t,v): # 读取训练集、验证集
    data,vali = [],[]
    for l in range(t):
        line = [float(x) for x in f.readline().strip('\n').split(',')]
        if line[-1] == 0: # 修改标签为-1
            line[-1] = -1
        data.append(line)
    for l in range(v):
        line = [float(x) for x in f.readline().strip('\n').split(',')]
        if line[-1] == 0: # 修改标签为-1
            line[-1] = -1
        vali.append(line)
    return data,vali
```

PLA 训练：

```
def PLA(data,eta,count): # PLA
    w,b = [0 for i in range(40)],0
    for i in range(count): # 最大迭代count次
        judge = 0
        for x in data:
            ans = 0
            for j in range(40): ans += w[j]*x[j]
            if (ans+b)*x[-1] <= 0: # 误判断点
                for j in range(40): w[j] += eta*x[-1]*x[j] # 更新参数
                b += eta*x[-1]
                judge = 1
                break
        if judge == 0: break # 没有误分类点
```

```
    return w,b
```

PLA 验证：

```
def validation(vali,w,b):
    num = 0
    for x in vali:
        ans = 0
        for j in range(40): ans += w[j]*x[j]
        if (ans+b)*x[-1] > 0: num += 1 # 预测正确
    print("{:.2f}%".format(num/len(vali)*100)) # 输出正确率
```

逻辑回归（LR）：下面仅展示关键代码，全部代码请移步 lab3_lr.py 文件

读取测试集和验证集：

```
def read(f,t,v): # 读取训练集、验证集
    data,vali = [],[]
    for l in range(t):
        line = [float(x) for x in f.readline().strip('\n').split(',')]
        line.insert(-1,1) # 添加一维常数项1
        data.append(line)
    for l in range(v):
        line = [float(x) for x in f.readline().strip('\n').split(',')]
        line.insert(-1,1) # 添加一维常数项1
        vali.append(line)
    return data,vali
```

计算概率 $\pi(x)$：

```
def dot(w,x): # 向量点积
    ans = 0
    for i in range(41): ans += w[i]*x[i]
    return ans

def Exp(t): # 防止exp溢出报错
    if t >= 100: return 1e40
    elif t <= -100: return 1e-40
    else: return exp(t)

def pi(w,x): # pi(x)
    return 1/(1+Exp(-dot(w,x))) # 1/(1+exp(-w*x))
```

对数似然函数（损失函数）：

```
def L(data,w): # 对数似然函数
    ans = 0
    for x in data: # sum(y*w*x-log(1+exp(w*x))
        t = dot(w,x)
        ans += x[-1]*t-log(1+Exp(t))
    return ans
```

计算当前梯度：

```
def Gra(data,w): # 梯度
    ans = [0 for i in range(41)]
    for x in data:
        t = x[-1]-pi(w,x) # -sum((y-pi(x))*x)
        for i in range(41): ans[i] -= t*x[i]
    return ans
```

LR 训练：

```
def update(data,w,eta): # 更新
    g = Gra(data,w) # 计算梯度
    for i in range(41): w[i] -= eta*g[i] # 梯度下降

def LR(data,eta,count): # 逻辑回归count次
    w = [0 for i in range(41)]
    for i in range(count):
        Lost = L(data,w)
        update(data,w,eta)
        if abs(Lost-L(data,w)) < 1e-2: break
    return w
```

LR 验证：

```
def validation(vali,w):
    num = 0
    for x in vali: # 对f(x)进行逻辑简化如下
        if x[-1] == 1 and pi(w,x) > 0.5: num += 1
        elif x[-1] == 0 and pi(w,x) < 0.5: num += 1
    print('{:.2f}%'.format(num/len(vali)*100))
```

实验结果以及分析：

感知机（PLA）：由于参数 η 对验证结果无关，参数 t 根据 7:3 原则取 5600，故只需对参数 $count$ 进行调参：

$count$	验证准确率
500	57.33%
1000	56.58%
2000	57.29%
5000	58.33%
10000	58.96%
15000	59.67%
18000	**60.25%**
20000	59.75%
25000	58.13%

最终参数 $count$ 取值为 18000，验证准确率为 60.25%

```
====== RESTART: /Users/songyujie/Desktop/Homework/人工智能/lab3/lab3_pla.py =====
=
60.25%
>>>
```

逻辑回归（LR）：由于具有三个参数 $\eta, count, t$，调参过程较为繁琐，这里只放上最优结果：

	η	count	t	验证准确率
初始参数	1	1000	5600	68.96%
最优参数	0.00002	100	6000	78.05%

```
====== RESTART: /Users/songyujie/Desktop/Homework/人工智能/lab3/lab3_lr.py ======
78.05%
>>>
```

从两种方法的验证准确率来看，逻辑回归的准确率高于感知机不少，个人认为是不同的梯度下降方法（**随机梯度下降**和**批梯度下降**）带来的影响，原因如下：

修改 PLA 的迭代过程为：从前往后开始遍历，当找到一个误分类点时，更新模型参数，更新之后**从下一个分类点开始遍历**，每遍历完一遍数据集为一次迭代过程

对新的参数 $count$ 进行调参：

$count$	验证准确率
10	72.83%
50	71.17%
100	71.92%
150	73.04%
200	**75.29%**
250	74.75%
300	71.25%
350	72.33%

```
====== RESTART: /Users/songyujie/Desktop/Homework/人工智能/lab3/lab3_pla.py =====
=
75.29%
>>>
```

在新的迭代条件下，PLA 的最优验证准确率达到 75.29%，已和 LR 最优结果相差不大（有趣的是，当对 t 作微小的修改时，正确率浮动变化很大）

思考题：

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。

- 收敛速度：一般情况下 η 越大收敛速度越快，当 η 超过一定值时由于不一定收敛而导致收敛速度变慢
- 是否收敛：较大会导致收敛至不正确的位置，过大甚至会导致模型不收敛

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，因为极难产生梯度模长为零的情况
- 一般通过某次更新参数之后损失函数变化低于一个阈值则判断为收敛