中山大学数据科学与计算机学院

移动信息工程专业-人工智能

本科生实验报告

（2017-2018 学年秋季学期）

课程名称：**Artificial Intelligence**

[redacted]			
[redacted]			

一、 实验题目

逻辑回归（Logistic Regression）

二、 实验内容

1. 算法原理

- **logistic 函数**

logistic（sigmoid）函数：$\theta(s)=\frac{e^s}{1+e^s}=\frac{1}{1+e^{-s}}$

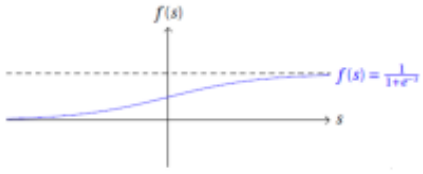
logistic（sigmoid）函数的特征：

① θ（-∞）= 0，当加权分数无穷小即 s->-∞,该数据属于正类别的概率为 0；

② θ（0）= 0.5，当加权分数为 0 即 s=0，该数据属于正/负类别的概率为 0.5，即该数据属于任一类别的概率相同；

③ θ（+∞）= 1，当加权分数无穷大即 s->+∞,该数据属于正类别的概率为 1；

利用该函数能将数据 s 从（-∞，+∞）映射到（0,1）



- **逻辑回归模型**

一个机器学习的模型，实际上是把决策函数限定在某一组条件下，这组限定条件就决定了模型的假设空间。在逻辑回归模型中，我们利用 logistics 函数所做的假设是：

$$h(x)=P(y=1|x)=\frac{1}{1+e^{-W^Tx}}$$

相应的决策函数为 如果 h(x)>0.5，则 y=1,即该数据属于正类别。

当模型的数学形式确定后，接下来要做的是如何求解模型中的参数，即求 W。在逻辑回归模型中，我们用最大似然估计算法来求 W，即找到一个 W，使得在这

个 W 下，我们的数据的似然度最大。在某种模型下利用给定数据 x 得到给定标签 y 的概率是这个问题的似然，所以在逻辑回归模型中，似然度

$likelihood = \prod_{i=1}^{M} P(label|x_i) = \prod_{i=1}^{M} h(x_i)^{y_i}\left(1-h(x_i)\right)^{1-y_i}$ 。

取对数得到对数似然度：$log(likelihood) = \sum_{i=1}^{M} y_i log(h(x_i)) + (1-y_i)log(1-h(x_i))$

再取负数：Err(W)= $-log(likelihood) = -\sum_{i=1}^{M} y_i log(h(x_i)) + (1-y_i)log(1-h(x_i))$

所以要取得最大似然度，即取得 Err(W)的最小值。

由于 Err(W)是一个连续可导，并且二阶可微的凸函数，所以根据凸优化理论，存在全局最优解（即存在最小值），即 ▽Err(W)=0；首先我们要推导得到▽Err(W_i)，令$u = 1 + e^{-\boldsymbol{w}^T \boldsymbol{x}_n}$和$v = -\boldsymbol{w}^T \boldsymbol{x}_n$，则

$$\begin{aligned}
\frac{\partial Err(w_i)}{\partial w_i} &= -\sum_{n=1}^{N}\left[(y_n)\left(\frac{\partial \log(h(\boldsymbol{x}_n))}{\partial h(\boldsymbol{x}_n)}\right)\left(\frac{\partial h(\boldsymbol{x}_n)}{\partial u}\right)\left(\frac{\partial u}{\partial v}\right)\left(\frac{\partial v}{\partial w_i}\right) + (1-y_n)\left(\frac{\partial \log(1-h(\boldsymbol{x}_n))}{\partial h(\boldsymbol{x}_n)}\right)\left(\frac{\partial h(\boldsymbol{x}_n)}{\partial u}\right)\left(\frac{\partial u}{\partial v}\right)\left(\frac{\partial v}{\partial w_i}\right)\right] \\
&= -\sum_{n=1}^{N}\left[(y_n)\left(\frac{1}{h(\boldsymbol{x}_n)}\right) + (1-y_n)\left(\frac{-1}{1-h(\boldsymbol{x}_n)}\right)\right]\left[\left(\frac{-1}{u^2}\right)(e^v)(-x_{n,i})\right] \\
&= -\sum_{n=1}^{N}\left[(y_n)\left(\frac{1}{h(\boldsymbol{x}_n)}\right) - (1-y_n)\left(\frac{1}{1-h(\boldsymbol{x}_n)}\right)\right][h(\boldsymbol{x}_n)(1-h(\boldsymbol{x}_n))](x_{n,i}) \\
&= -\sum_{n=1}^{N}[(y_n)(1-h(\boldsymbol{x}_n)) - (1-y_n)h(\boldsymbol{x}_n))](x_{n,i}) \\
&= -\sum_{n=1}^{N}(y_n - h(\boldsymbol{x}_n))(x_{n,i})
\end{aligned} \tag{2.11}$$

则▽Err(W_i)的梯度：$\nabla Err(w_i) = \sum_{n=1}^{N}\left(\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} - y_n\right)(x_{n,i})$

因为这个梯度是非线性函数，所以我们很难求得它的零点，所以我们采用迭代最优化的方式去求解。因为 Err(W)是一个凸函数，并且梯度是函数变化最快的方向，所以我们只要沿着梯度下降的方向去更新 W，就一定能较迅速地找到最优解。

所以 W 的更新公式为：$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\nabla Err(\boldsymbol{w}_t)$ ，η 代表梯度下降到的步长。

◆ 批梯度下降法

批梯度下降法是在更新参数（即更新 W）时使用所有的样本来进行更新。即

$\nabla Err(w_{t,i}) = \sum_{n=1}^{N}\left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$

➢ 优点：① 最小化所有训练样本的损失函数，使得最终求解的是全局的最优解，即求解的参数是使得风险函数（Err）最小。

➢ 缺点：① 当样本量很大时，训练速度慢；② 需要得到所有的样本数据才能开始训练；

◆ 随机梯度下降法

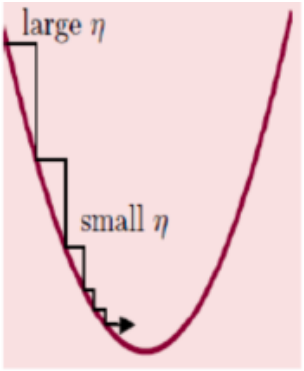
随机梯度法是在更新参数时只使用选择的一个样本 i 来进行更新。即

$\nabla Err(w_{t,i}) = \left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$

- 优点：因为每次只采用一个样本来迭代，训练速度很快。
- 缺点：① 随机梯度下降仅仅用一个样本决定梯度方向，可能导致解不是最优。② 由于随机梯度下降法一次迭代一个样本，导致迭代方向变化很大，不能很快地收敛到局部最优解。

◆ 动态调整学习率(步长)

初始步长加大，当梯度下降到接近最优值时，将步长减小。



2. 伪代码

逻辑回归训练算法：（批梯度下降）

设定步长 η；

while(迭代次数<设定的迭代次数){

for（遍历 W 的每一个维度）{

通过公式 $\nabla Err(w_{t,i}) = \sum_{n=1}^{N}\left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$ 计算该维度的梯度；

通过公式 $w_{t+1,i} = w_{t,i} - \eta \nabla Err(w_{t,i})$ 更新 W 的该维度的值；

}

迭代次数++；

}

逻辑回归训练算法：（随机梯度下降）

设定步长 η；

while(迭代次数<设定的迭代次数){

随机选择一行 i；

for（遍历 W 的每一个维度）{

通过公式 $\nabla Err(w_{t,i}) = \left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$ 计算该维度的梯度；

通过公式 $w_{t+1,i} = w_{t,i} - \eta \nabla Err(w_{t,i})$ 更新 W 的该维度的值；

}

迭代次数++；

}

逻辑回归训练算法：（动态调整步长）

设定初始步长 alpha；

while(迭代次数<设定的迭代次数){

根据迭代次数设置步长：

```
alpha=1.0*alpha/(1.0+num_of_iterator)+0.01;
```

for（遍历 W 的每一个维度）{

通过公式 $\nabla Err(w_{t,i}) = \sum_{n=1}^{N}\left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$ 计算该维度的梯度；

通过公式 $w_{t+1,i} = w_{t,i} - \eta \nabla Err(w_{t,i})$ 更新 W 的该维度的值；

}

迭代次数++；

}

3. 关键代码截图（带注释）

（1）批梯度下降

① 设定步长（学习率）：

```
double alpha=10;//设定步长（学习率）
```

② 遍历 W 的每个维度，计算每个维度的梯度：$\nabla Err(w_{t,i}) = \sum_{n=1}^{N}\left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$

```
/***************计算该维度的梯度***********/
double Err=0;
for(int i=0;i<train_row;i++){
    double r=0;
    for(int j=0;j<train_column;j++){
    //计算Err的前部分
        r=r+w[j]*train[i][j];
    }
    Err+=1.0*(1.0/(1+exp(-r))-train_label[i])*train[i][k];
}
/***************计算该维度的梯度***********/
```

③ 通过公式 $w_{t+1,i} = w_{t,i} - \eta \nabla Err(w_{t,i})$ 更新 W 的该维度的值；

```
new_w.push_back(w[k]-1.0*alpha*Err);      //更新该维度的值
```

（2）随机梯度下降

① 设定步长（学习率）：

```
double alpha=10;//设定步长（学习率）
```

② 随机选择一行来更新 W：

```
srand(num_of_iterator);//设定随机数种子
int i=rand()%train_row;  //随机选择一行来更新w
```

③ 计算该维度的梯度：$\nabla Err(w_{t,i}) = \left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$

```
/***************计算该维度的梯度***********/
double Err=0;
double r=0;
for(int j=0;j<train_column;j++){
//计算Err的前部分
    r=r+w[j]*train[i][j];
}
Err=1.0*(1.0/(1+exp(-r))-train_label[i])*train[i][k];
/***************计算该维度的梯度***********/
```

④ 通过公式 $w_{t+1,i} = w_{t,i} - \eta \nabla Err(w_{t,i})$ 更新 W 的该维度的值；

```
new_w.push_back(w[k]-1.0*alpha*Err);      //更新该维度的值
```

（3）动态调整步长（学习率）

① 设定初始步长（学习率）： `double alpha=10;`

② 每次迭代都根据迭代的次数来更新步长（学习率）：

```
alpha=1.0*alpha/(1.0+num_of_iterator)+0.01;
```

③ 遍历 W 的每个维度，根据公式 $\nabla Err(w_{t,i}) = \sum_{n=1}^{N}\left(\frac{1}{1+e^{-w_i^T x_n}} - y_n\right)(x_{n,i})$ 计算该维度的梯度；然后根据公式 $w_{t+1,i} = w_{t,i} - \eta \nabla Err(w_{t,i})$ 更新 W 的每个维度。（这部分和（1）批梯度下降相同）

三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

计算小数据集的对数似然的值来验证模型中梯度计算是否正确，W 的更新是否正确：

3	3	1
4	3	1
1	1	0

初始的增广权向量：		增广特征向量	增广权向量的转置乘以增广特征向量	对数似然的值
-3	A	1	3	-0.04859
1		3		
1		3		
	B	1	4	-0.01815
		4		
		3		
	C	1	-1	-0.31326
		1		
		1		

```
第0次迭代
W0=-3
W1=1
W2=1
r=3
对数似然likehood=-0.0485874
r=4
对数似然likehood=-0.0181499
r=-1
对数似然likehood=-0.313262
```

步长为0.1更新后的增广权向量：	增广特征向量	增广权向量的转置乘以增广特征向量	对数似然的值
-3.02	A	1 2.92	-0.05253
0.99		3	
0.99		3	
	B	1 3.91	-0.01984
		4	
		3	
	C	1 -1.04	-0.30266
		1	
		1	

```
W0=-3.02035
W1=0.994528
W2=0.992729
r=2.94142
对数似然likehood=-0.0514445
r=3.93595
对数似然likehood=-0.019339
r=-1.0331
对数似然likehood=-0.304468
```

步长为1更新后的增广权向量：	增广特征向量	增广权向量的转置乘以增广特征向量	对数似然的值
-3.2	A	1 2.44	-0.08357
0.95		3	
0.93		3	
	B	1 3.39	-0.03315
		4	
		3	
	C	1 -1.32	-0.23676
		1	
		1	

```
W0=-3.20353
W1=0.945281
W2=0.927295
r=2.4142
对数似然likehood=-0.0856629
r=3.35948
对数似然likehood=-0.0341631
r=-1.33095
对数似然likehood=-0.234459
```

步长为10更新后的增广权向量：	增广特征向量	增广权向量的转置乘以增广特征向量	对数似然的值
-5.04	A	1 -2.86	-2.91381
0.45		3	
0.27		3	
	B	1 -2.41	-2.4916
		4	
		3	
	C	1 -4.31	-0.01335
		1	
		1	

```
W0=-5.03529
W1=0.45281
W2=0.272948
r=-2.85802
对数似然likehood=-2.91381
r=-2.40521
对数似然likehood=-2.49161
r=-4.30953
对数似然likehood=-0.0133503
```

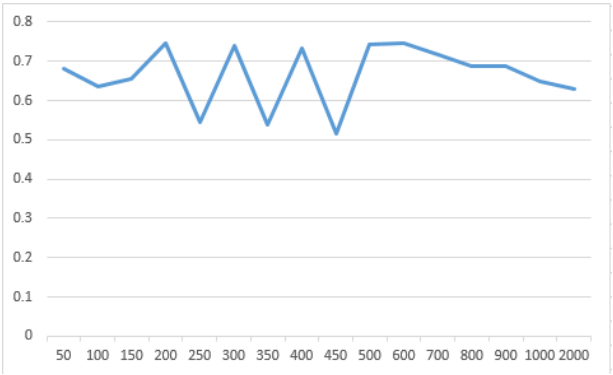
2. 评测指标展示即分析（如果实验题目有特殊要求，否则使用准确率）

在该实验中，我在读取文本的函数中设置了一个参数，该参数的含义是每隔 n 条抽出一条作为验证集。当 n=4 时，则读取完文本之后，每次抽取 4 行作为测试集，然后再抽取 1 行作为验证集。

```
get_train("F:\\学习资料\\大三上\\人工智能\\实验\\lab5\\train.csv",4);
```

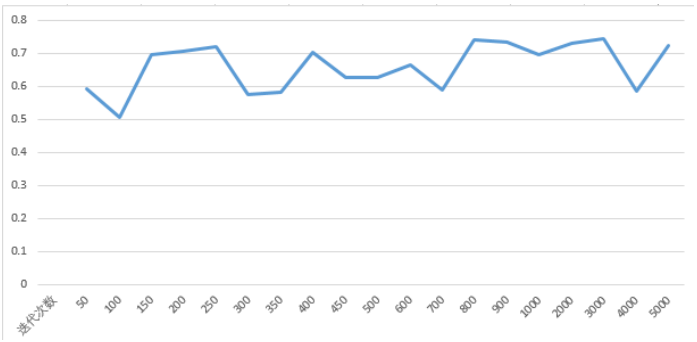
（1）批地图下降法：

迭代次数	50	100	150	200	250	300	350	400	450	500	600	700	800	900	1000	2000
准确率	0.682341	0.635318	0.655828	0.745373	0.543272	0.74037	0.536768	0.733367	0.514257	0.743372	0.744872	0.715858	0.687344	0.687844	0.649325	0.628814



（2）随机梯度下降法：

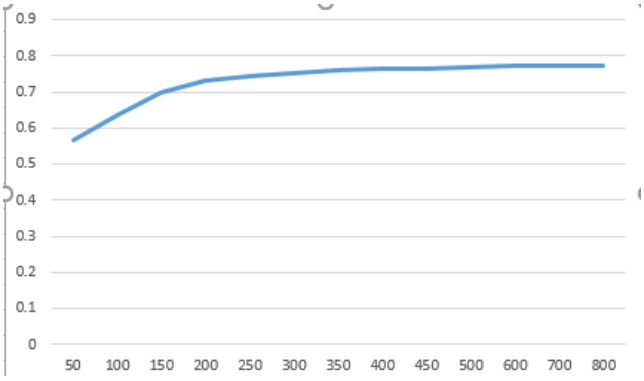
迭代次数	50	100	150	200	250	300	350	400	450	500	600	700	800	900	1000	2000	3000	4000	5000	6000
准确率	0.594297	0.507254	0.695348	0.706353	0.721861	0.577289	0.583292	0.702351	0.626813	0.628814	0.666333	0.588794	0.742371	0.735868	0.695348	0.730865	0.744372	0.587294	0.723362	0.688844



随机梯度下降最大的缺点在于每次更新可能并不会按照正确的方向进行，因此可以带来优化波动。

（3）动态调整步长：

迭代次数	50	100	150	200	250	300	350	400	450	500	600	700	800
准确率	0.566283	0.635818	0.69985	0.731366	0.744872	0.753377	0.76038	0.762381	0.764382	0.766383	0.773387	0.774387	0.771386



四、 思考题

1.如果把梯度为 0 作为算法停止的条件，可能存在怎么样的弊端？

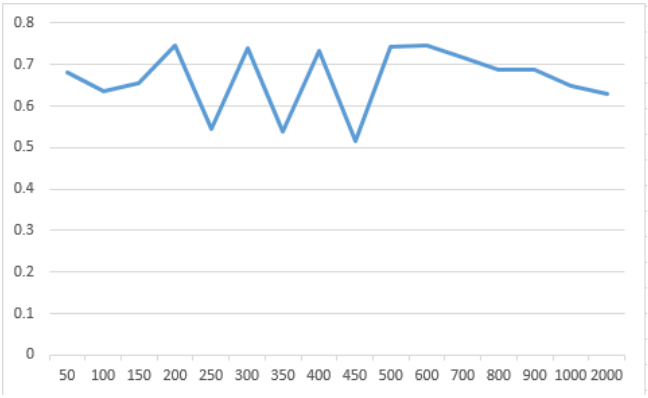
答：① 如果把梯度为 0 作为算法停止的条件，如果函数是非凸函数，即函数有多个局部最优解时，往往不能保证它收敛到全局最小值。

② 在实际情况中，很难满足梯度为 0 的条件。因为即使逼近极值点，梯度已近乎为 0 时，步长很难取到让更新的下一个梯度刚好为 0，一般都是会在极值点处左右徘徊。如果一直到不了让梯度为 0 的点的话，就会死循环。

2. 学习率 η 的大小会怎么影响梯度下降的结果，给出具体解释。

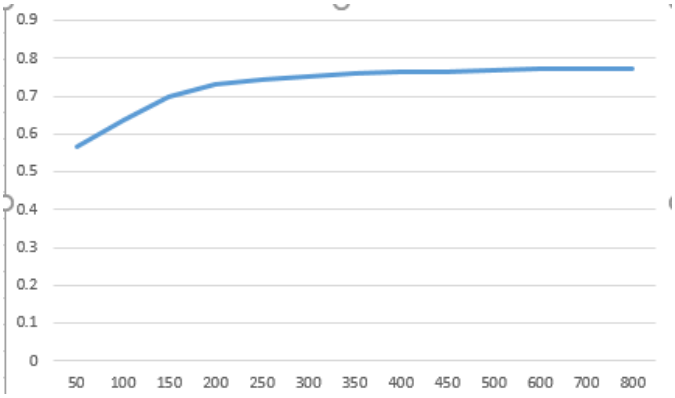
答：学习率决定了参数移动到最优解的速度的快慢。如果学习率过大，很可能会越过最优解；但是如果学习率过低，优化的效率可能会很低，长时间无法收敛。

例如在该实验中，如果设置学习率为定值，然后学习率不再改变，结果如下图所示：



由上图可知，在学习率为固定值时，结果有波动，在极值点处波动。

但是如果设置学习率动态变化，每次迭代都调整学习率，结果如下图：



由上图可知，当我们设置学习率动态变化时，比较不容易错过最优解。

3. 思考随机梯度下降和批梯度下降这两种优化方法的优缺点。

答：**批梯度下降法**

批梯度下降法是在更新参数（即更新 W）时使用所有的样本来进行更新。即

$$\nabla Err(w_{t,i}) = \sum_{n=1}^{N}\left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$$

- 优点：① 最小化所有训练样本的损失函数，使得最终求解的是全局的最优解，即求解的参数是使得风险函数最小。
- 缺点：① 当样本量很大时，训练速度慢；② 需要得到所有的样本数据才能开始训练；

随机梯度下降法

随机梯度法是在更新参数时只使用选择的一个样本 i 来进行更新。即

$$\nabla Err(w_{t,i}) = \left(\frac{1}{1+e^{-w_t^T x_n}} - y_n\right)(x_{n,i})$$

- 优点：因为每次只采用一个样本来迭代，训练速度很快。
- 缺点：① 随机梯度下降仅仅用一个样本决定梯度方向，可能导致解不是最优。② 由于随机梯度下降法一次迭代一个样本，导致迭代方向变化很大，不能很快地收敛到局部最优解。