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**

教学班级	周四下午 7-8 节	专业（方向）	互联网
学号	15352010	姓名	蔡烨

一、 实验题目

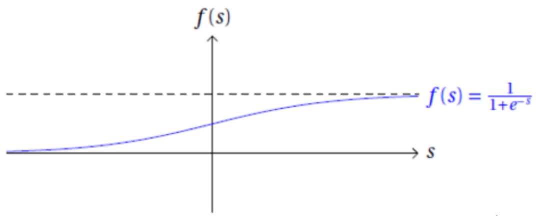
逻辑回归分类模型

二、 实验内容

1. 算法原理

（1）逻辑回归算法是一种软分类算法，通过计算数据权重，根据权重了解预测目标的可能性。目标函数定义为：$f(x) = P(label \mid x) \in [0,1]$，即在给定特征向量 x 的情况下，属于 label 类的可能性有多大。对每个分类求概率后，取概率最大的分类作为该数据的分类。

• 逻辑回归是基于线性回归的理论，但是如果对特征向量的每个属性加权相加，得到的可能是一个超出 0 到 1 的范围的数值，因此引入 logistic(sigmoid)函数：$f(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$，将数值 s 的范围由$(-\infty, +\infty)$映射到(0,1)。



• 权重向量 w 表示特征向量各个维度的权重，$w_i > 0$表示该维度的特征对正类别有正面影响，且值越大，正面影响越大，反之亦然。当权重向量无穷小，特征向量每个维度加权相加的结果显然也是无穷小，在 logistic(sigmoid)函数中，f(-∞)→0，表示该数据属于正类别的概率几乎为 0，所以应该将该数据划分为负类别。同理，当权重向量无穷大时，f(+∞)→1，该数据属于正类别的概率为接近 1，因此应该划分为正类别。

（2）基于上述理论，影响数据的类别的是权重向量，只要求出权重向量 w，则可以通过 logistic 函数判断该数据的类别。

由于$s = \sum_{i=0}^{d} w_i x_i = w^T x$，结合 logistic 函数，构造一个函数：$h(x) = \frac{1}{1+e^{-w^T x}}$。h(x)算出来的是属于正类别的概率，属于负类别的概率则为 1-h(x)。即 P(label=1 | x) = h(x)，P(label=0 | x) = 1-h(x)。当 h(x)>0.5 时，该数据属于正类别的可能性更大，因此划分为正类别。

•联合目标函数 f(x)和 h(x)，有伯努利分布：$f(x) = P(label \mid x) = h(x)^{y}(1-h(x))^{1-y}$，y 是 x 对应的分类标签（0 或 1）。显然，当 y=1 时，f(x) = P(label=1 | x) = h(x)表示的就是该数据划分为正类别的概率；当 y=0 时，f(x) = P(label=0 | x) = 1-h(x)表示的是划分为负类别的概率。可以看出，f(x)越大，意味着该数据样本属于某个类别的概率越大，因此该数据被划分为这个类别的最准确性越高。

（3）上面的理论是基于某个样本数据来考虑的，当考虑整个数据集时，得到整个数据集的似然函数：$\text{likelihood} = \prod_{i=1}^{M} P(label \mid x_i) = \prod_{i=1}^{M} h(x_i)^{y_i}(1-h(x_i))^{1-y_i}$。根据最大似然估计算法，我们要找到一组模型参数，使得这个似然函数最大。

•对 likelihood 取对数，再取负数之后，得到$-\log(\text{likelihood}) = -\log\prod_{i=1}^{M} P(label \mid x_i) = -\sum_{i=1}^{M} y_i\log(h(x_i)) + (1-y_i)\log(1-h(x_i))$，对该函数取最小，即达到最大似然的目的。

•令$\text{Err}(w_i) = -\sum_{n=1}^{M} y_n\log(h(x_n)) + (1-y_n)\log(1-h(x_n))$，联合$h(x) = \frac{1}{1+e^{-w^T x}}$，令$u = 1+e^{-w^T x}$，$v = -w^T x$，对 w 求导如下：

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

得到梯度$\nabla\text{Err}(w_i) = \sum_{i=1}^{n}(\frac{1}{1+e^{-w^T x_i}} - y_i)x_i^{(j)}$

（4）该梯度表达式是一个非线性函数，难以求解函数零点，因此采用迭代最优法的方式来求解。因为是一个凸函数，所以只要沿着梯度下降的方法去更新求解 w，就能找到最优解，因为梯度是函数变化最快的方向。

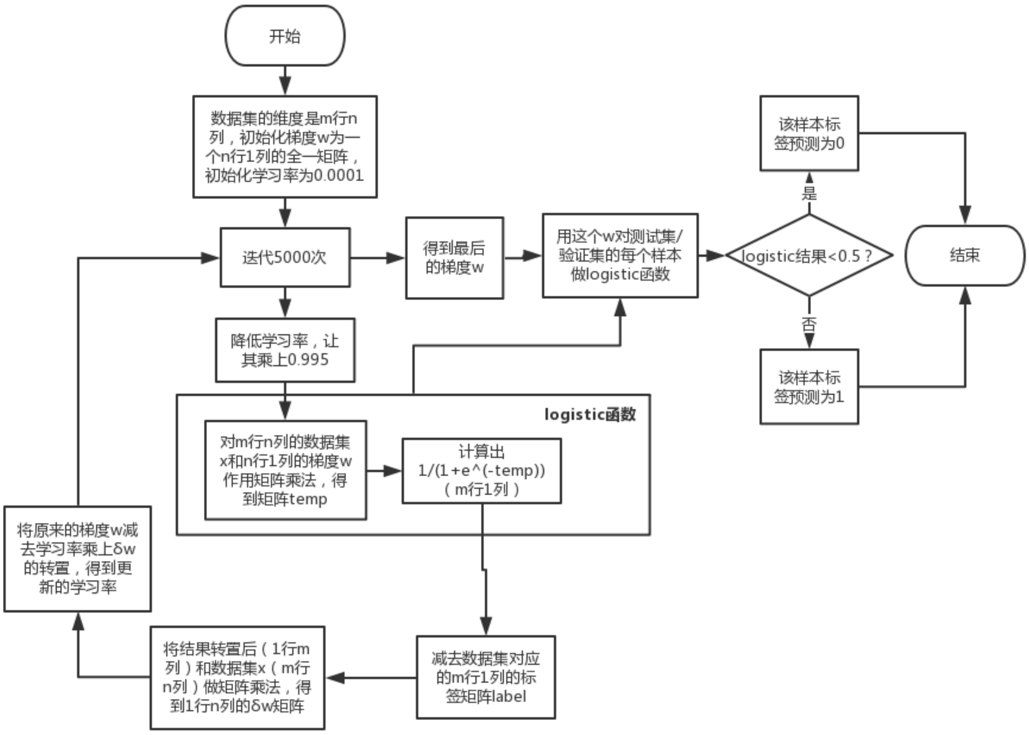
•利用梯度下降法，通过不断地迭代使 w 逼近最优解直至收敛

$$w_{new}^{(j)}=w^{(j)} - \mu\frac{\partial Err(w)}{\partial w^{(j)}} = w^{(j)} - \mu\sum_{i=1}^{n}[\left(\frac{1}{1+e^{-w^T x_i}} - y_i\right)x_i^{(j)}]$$

其中 μ 表示学习率，j 表示第几个维度，i 表示数据集中的第几个样本

（5）通过以上方法对训练集训练，得到最后的权重 w，用这个 w 和测试集的数据样本进行 logistic 函数作用：$h(x) = \frac{1}{1+e^{-w^T x}}$，得到的概率值 h(x)如果>0.5，则将该样本归为正类别，否则归为负类别。

2. 流程图



3. 关键代码截图（带注释）

验证集的代码由于不用跑测试集，而是多次打乱 train.csv 文件的数据，分割训练集和验证集，因此特征向量和标签 label 是一起放进同一个矩阵的，矩阵的最后一列就是每一行样本对应的 label。而测试集的代码由于测试集中有问号？存在，无法直接转化为 int 存进作为 label 的那一列，因此数据和 label 分开存在两个矩阵中。两种实现方式在理解上其实没有差别，只是存储的时候有一点点区别而已。在对测试集进行预测的时候，用的训练集是整个 train.csv 的数据。

预测测试集的代码：LR-test.py

```
def logistic(w, x):
    temp = np.dot(x, w)
    temp = 1 / (1 + np.exp(-temp))
    return np.mat(temp)


def process(file):
    data, label = readData(file)
    m, n = np.shape(data)
    weight = np.mat(np.ones((n, 1)))  # 初始化梯度w
    study_ratio = 0.0001  # 初始化学习率
    for iterate in range(5000):  # 更新多次w后再停止
        study_ratio *= 0.995  # 调整学习率
        logi = logistic(weight, data) - label
        detaW = np.dot(logi.T, data)
        weight = weight - study_ratio * detaW.T
    return weight


def predict(train_file, test_file):
    f = open('15352010_caiye.txt', 'w')
    w = process(train_file)
    test_data, test_label = readData(test_file)
    label = logistic(w, test_data)
    for num in label:  # 小于0.5判为0, 否则判为1
        num = 0 if num < 0.5 else 1
        f.write(str(num) + '\n')
    f.close()
```

随机划分训练集和验证集并计算准确率的代码：LR-validate.py

```python
def logistic(w, x):
    temp = np.dot(x, w)
    temp = 1 / (1 + np.exp(-temp))
    return np.mat(temp)


def process(file):
    dataAll = readData(file)
    m, n = np.shape(dataAll[:, :-1])
    for i in range(20):  # 随机打乱次数
        np.random.shuffle(dataAll)  # 随机打乱数据集
        data = dataAll[:int(m * 0.75), :]  # 训练集
        validate = dataAll[int(m * 0.75):m, :]  # 验证集
        weight = np.mat(np.ones((n, 1)))  # 初始化梯度w
        study_ratio = 0.0001  # 学习率
        for iterate in range(5000):
            study_ratio *= 0.995  # 调整学习率
            logi = logistic(weight, data[:, :-1]) - data[:, -1]
            detaW = np.dot(logi.T, data[:, :-1])
            weight = weight - study_ratio * detaW.T
        label = logistic(weight, validate[:, :-1])

        yes = 0  # 预测正确的个数
        for j in range(len(label)):  # 和真正的标签差值小于0.5则预测正确
            if abs(label[j] - validate[j, -1]) < 0.5:
                yes += 1
        print("随机" + str(i) + "准确率: ", yes / len(label))
```

4. 创新点&优化（如果有）

（1）使用 np.random.shuffle()随机打乱数据集，形成多个训练集和验证集，多次验证。此处有大坑，Python 内置的 random.shuffle 函数作用到矩阵上面时会出现问题，大概就是会使得数据集不断增加，因为不断增加重复的行，所以随着打乱次数增加，准确率不断上升，甚至达到百分百。换用 np.random.shuffle 后就能解决问题。感谢 TA 大大指点迷津，笔芯❤

（2）使用矩阵运算，减少程序运行时间

（3）使用动态学习率，每次更新梯度 w 后，降低学习率，步长越来越小，不会引起发散，或者跳过最优点。

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）

小数据测试：

训练集：

	ATTR1	ATTR2	LABEL
A	0	1	0
B	1	1	0
C	3	3	1
D	4	3	1

初始 w：-5，2，1（三行一列）

程序运行结果：

```
data:
 [[1.0, 0.0, 1.0], [1.0, 1.0, 1.0], [1.0, 3.0, 3.0], [1.0, 4.0, 3.0]]
initial w:
 [[-5], [2], [1]]
final w:
 [[-5.01167303]
 [ 1.99446462]
 [ 0.99241874]]
[Finished in 0.4s]
```

手算验证：

$$\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}_A=\begin{pmatrix}-5\\2\\1\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}1\\0\\1\end{pmatrix}=w_0\times1+w_1\times x_A^1+w_2\times x_A^2=-5\times1+2\times0+1\times1=-4$$

$$\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}_B=-5\times1+2\times1+1\times1=-2$$

$$\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}_C=-5\times1+2\times3+1\times3=4$$

$$\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}_D=-5\times1+2\times4+1\times3=6$$

$$\begin{aligned}w_0^{new}&=w_0^{old}-\eta\sum\left[\left(\frac{e^{\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}{1+e^{\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}-y\right)x_0\right]\\&=-5-\eta\left[\left(\frac{e^{-4}}{1+e^{-4}}-0\right)\times1+\left(\frac{e^{-2}}{1+e^{-2}}-0\right)\times1+\left(\frac{e^{4}}{1+e^{4}}-1\right)\times1+\left(\frac{e^{6}}{1+e^{6}}-1\right)\times1\right]\\&=-5-\eta\times0.1167\\&=-5.0117\end{aligned}$$

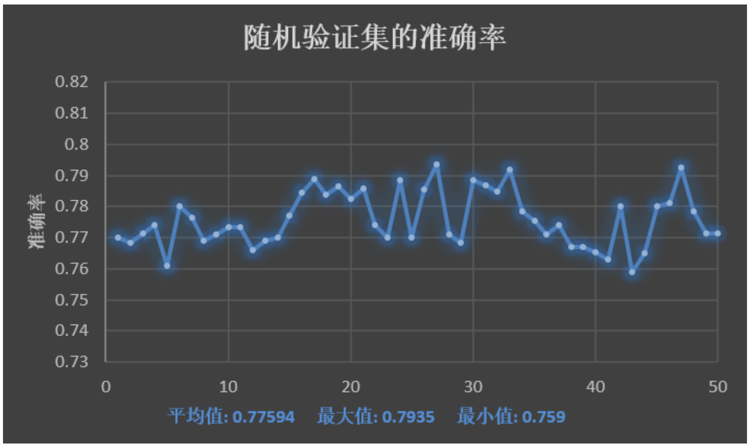
$$\begin{aligned}w_1^{new}&=w_1^{old}-\eta\sum\left[\left(\frac{e^{\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}{1+e^{\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}-y\right)x_1\right]\\&=2-\eta\left[\left(\frac{e^{-4}}{1+e^{-4}}-0\right)\times0+\left(\frac{e^{-2}}{1+e^{-2}}-0\right)\times1+\left(\frac{e^{4}}{1+e^{4}}-1\right)\times3+\left(\frac{e^{6}}{1+e^{6}}-1\right)\times4\right]\\&=2-\eta\times0.0554\\&=1.9945\end{aligned}$$

$$\begin{aligned}w_2^{new}&=w_2^{old}-\eta\sum\left[\left(\frac{e^{\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}{1+e^{\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}-y\right)x_2\right]\\&=1-\eta\left[\left(\frac{e^{-4}}{1+e^{-4}}-0\right)\times1+\left(\frac{e^{-2}}{1+e^{-2}}-0\right)\times1+\left(\frac{e^{4}}{1+e^{4}}-1\right)\times3+\left(\frac{e^{6}}{1+e^{6}}-1\right)\times3\right]\\&=1-\eta\times0.0758\\&=0.9924\end{aligned}$$

所以，$\tilde{\mathbf{w}}^{new}=\begin{pmatrix}-5.0117\\1.9945\\0.9924\end{pmatrix}$. 与程序运行结果一致。

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）

将数据集随机打乱，抽取前 3/4 作为训练集，后 1/4 作为验证集。打乱 50 次，得到 50 个准确率，结果如图。平均准确率为 0.77594。准确率波动不大。

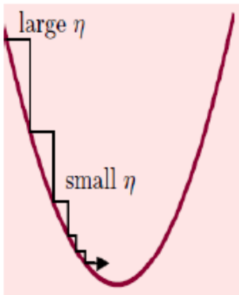


3. 思考题

（1）如果把梯度为 0 作为算法停止的条件，可能存在怎样的弊端？

答：把梯度为 0 作为算法停止的条件，意味着 w 的所有维度都要到 0.显然当维度多，或者学习率大的时候，需要很长的时间才可能到达这样的结果，甚至是永远也到达不了梯度为 0，导致程序陷入死循环。

（2）学习率的大小会怎么影响梯度下降的结果？给出具体的解释，可视化的解释最好，比如图形展示等



答：如图，梯度 w 在抛物线上“跳来跳去”，学习率就是每次的跨步有多大。显然，如果学习率很大，w 每次会跳得很远，这就可能发生 w 才刚靠近极点（我们希望 w 到达的地方），又马上跳得离极点更远的情况，甚至是跳到抛物线之外，发散了。而如果学习率比较小，意味着梯度 w 每次的变化只有一点点，那么就需要比较久的时间才能从远处走向极点。

（3）思考批梯度下降和随机梯度下降两张优化方法的优缺点

	批梯度下降	随机梯度下降
优点	每次更新都会朝着正确的方向进行	每次更新只用到一个样本数据，显著减少计算量； 能及时更新模型
缺点	数据集很大时，训练过程计算量很大； 需要得到所有的数据才能开始训练； 对于非凸目标函数，容易陷入次优的局部极值	每次更新可能不会按照正确的方向进行，可能带来优化波动； 对于非凸目标函数，容易陷入次优的局部极值

|---------------- 如有优化，重复 1，2 步的展示，分析优化后结果 -------------|

PS：可以自己设计报告模板，但是内容必须包括上述的几个部分，不需要写实验感想