计算机科学与技术学院实验报告

课程名称：机器学习

课程类型：选修

实验题目：Logistic 回归

学号：1171800323

姓名： 杨富祥

一、实验目的

理解逻辑回归模型，掌握逻辑回归模型的参数估计算法。

二、实验要求及实验环境

实验要求：

实现两种损失函数的参数估计：（1 无惩罚项；（2 加入对参数的惩罚，可以采用梯度下降、共轭梯度或者牛顿法等。

实验环境： Windows10 + matlabR2018a

三、设计思想

1．算法原理

朴素贝叶斯假设： 设特征 x 含有 n 个特征属性（x_1，x_2，…, x_n），那么在给定 y 的情况下，x_1，x_2，…，x_n 是条件独立的。

若要生成满足朴素贝叶斯假设的数据时，依靠协方差矩阵是对角阵，使用 matlab 的函数 mvnrnd 来产生数据。这样生成的数据各维度之间是不相关的，一般来说不相关不一定独立，但是对于多元正态分布来说，不相关性等价于独立性。相关推导见参考文献 3。更一般来说，可以采用多个独立一维高斯分布分别生成数据，然后组合起来。

而要生成不满足朴素贝叶斯假设的数据，可以让协方差矩阵不是对角阵，这样数据某些维度之间是相关的（一定不独立）。

贝叶斯公式： $P(A|B)=\frac{P(B|A)P(A)}{P(B)}=\frac{P(B|A)P(A)}{P(B|A)P(A)+P(B|\sim A)P(\sim A)}$（$\sim A$表示“非A”）。

似然函数： $P(x|\theta)$，如果x是已知确定的，θ是变量。它描述对于不同的模型参数，出现x这个样本点的概率是多少。

最大似然估计（MLE）： 利用已知的样本结果，反推最有可能导致这样结果的参数值。它提供了一种给定观察数据来评估模型参数的方法，即：“模型已定，参数未知”。通过若干次实验，观察其结果，利用实验结果得到参数值使样本出现的概率最大。

假设样本集 D 是独立同分布的，则参数θ对于数据D的似然是：

$$P(D|\theta)=p(x_1,x_2,\dots,x_n|\theta)=\prod_{i=1}^{n}p(x_i|\theta)$$

由于上式连乘容易造成下溢，通常使用对数似然（log-likelihood）：

$$LL(\theta) = logP(D|\theta)$$
$$= \sum_{x \epsilon D} \log P(x|\theta)$$

此时参数θ的极大似然估计为：

$$\hat{\theta} = \underset{\theta}{argmax} LL(\theta)$$

最大后验概率估计（MAP）：最大似然估计是求参数θ，使似然函数$P(x|\theta)$最大。最大后验估计则是想求θ使得$P(x|\theta)P(\theta)$最大，求得的θ不单单让似然函数大，θ自己出现的先验概率也得大（类似于正则化加惩罚项）。

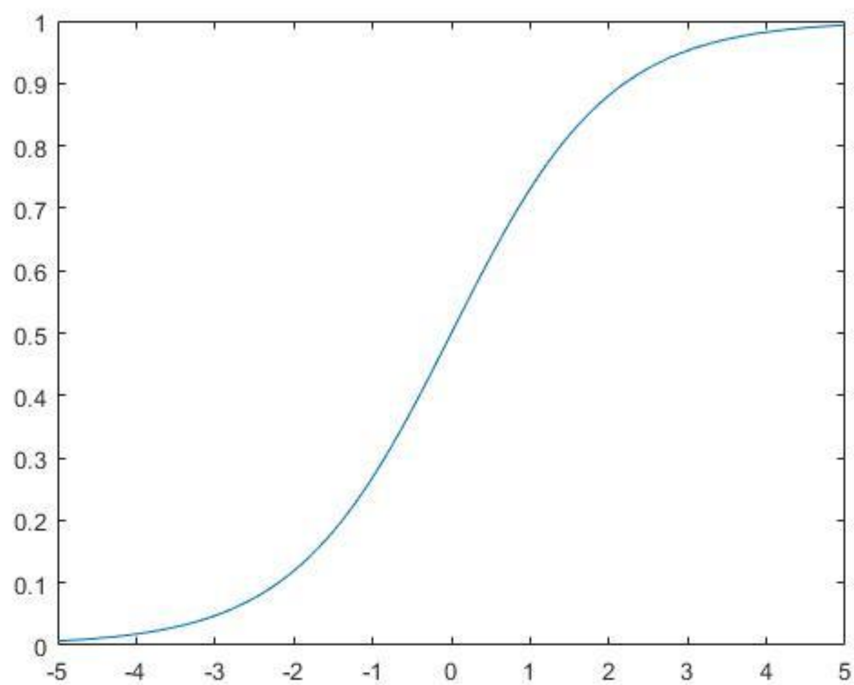
MAP 在最大化后验概率$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$，而 x 是已知确定的，$P(x)$是一个已知值，所以可以去掉分母$P(x)$。最大化$P(\theta|x)$即为 x 已经出现了，要求$\theta$取什么值使得$P(\theta|x)$最大。

logistic 回归：这是分类问题，只需要找一个单调可微函数将分类任务的真实标记 y 与线性回归模型的预测值联系起来。

考察二分类任务，其输出标记$y\epsilon\{0,1\}$，而线性回归模型产生的预测值$z = w^T x + b$是实值，于是，我们需要将实值 z 转换为 0/1 值。对数几率函数正是一个常用的替代函数：

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

logistic 函数是 sigmoid 函数，其图像如下：



这个函数关于点$(0,0.5)$中心对称：$y(z)+y(-z)=1$。当$z>0$时，$y(z)>0.5$；当$z<0$时，$y(z)<0.5$。因此，可以用这个函数以$(0,0.5)$为边界将数据分为两类，从而处理二分类问题。

将 z 代入 y 得到：

$$y=\frac{1}{1+e^{-(w^Tx+b)}}$$

可变形为：

$$\ln\frac{y}{1-y}=w^T+b$$

将上式中的 y 视为类后验概率$p(y=1|x)$可重写为：

$$\ln\frac{p(y=1|x)}{p(y=0|x)}=w^T+b$$

显然有：

$$p(y=1|x)=\frac{1}{1+e^{-(w^Tx+b)}}$$

$$p(y=0|x)=\frac{e^{-(w^Tx+b)}}{1+e^{-(w^Tx+b)}}$$

（老师关于上式由来的一种解释：

假设$\pi=p(Y=1)$，$p(x|y_k)=\frac{1}{\sigma_{ik}\sqrt{2\pi}}\exp\left(-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}\right).$

$$\begin{aligned}p(Y=1|X)&=\frac{p(Y=1)p(X|Y=1)}{p(Y=1)p(X|Y=1)+p(Y=0)p(X|Y=0)}\\&=\frac{1}{1+\frac{p(Y=0)p(X|Y=0)}{p(Y=1)p(X|Y=1)}}\\&=\frac{1}{1+\exp\left(ln\frac{p(Y=0)p(X|Y=0)}{p(Y=1)p(X|Y=1)}\right)}\\&=\frac{1}{1+\exp\left(\ln\frac{1-\pi}{\pi}+\sum_i ln\frac{p(X_i|Y=0)}{p(X_i|Y=1)}\right)}\quad\text{（朴素贝叶斯假设）}\end{aligned}$$

$$\sum_i ln\frac{p(X_i|Y=0)}{p(X_i|Y=1)}=\sum_i\left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i+\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right)$$

其中，w^Tx对应于$\sum_i\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i$，$b$对应于$\ln\frac{1-\pi}{\pi}+\sum_i\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}$

）

于是，我们可以通过 MLE 来估计w和b。给定数据集$\{(x_i,y_i)\}_{i=1}^m$，对数似然函数为：

$$L(w,b)=\sum_{i=1}^{m}\log p(y_i|x_i;w,b)$$

其中的似然项可重写为：

$$p(y_i|x_i;w,b)=\left(p(y_i=1|x_i;w,b)\right)^{y_i}(p(y_i=0|x_i,w,b))^{1-y_i}$$

从而得到：

$$\begin{aligned}L(w)&=\sum_{i=1}^{m}(\mathrm{y_i}\log p(y_i=1|x_i;w,b)+(1-\mathrm{y_i})\log p(y_i=0|x_i;w,b))\\&=\sum_{i=1}^{m}(y_i\log\frac{1}{1+e^{-(w^Tx+b)}}+(1-y_i)\log\frac{e^{-(w^Tx+b)}}{1+e^{-(w^Tx+b)}})\\&=\sum_{i=1}^{m}(y_i(w^Tx_i+b)-\log\left(1+e^{w^Tx_i+b}\right))\end{aligned}$$

而$maxlogL(w)$即为$min-L(w)$，函数$-L(w)$称为损失函数$\phi(w)$。

向量化表示为：

$$\begin{aligned}g(z)&=\frac{1}{1+e^{-z}}\\h&=g(Xw)\\\phi(w)&=\frac{1}{m}(-y^T\log h-(1-y)^T\log(1-h))\end{aligned}$$

接下来需要求解$\boldsymbol{min\phi(w)}$这个无约束优化问题。首先应该讨论这个问题是不是凸优化问题，通过目标函数$\boldsymbol{\phi(w)}$的海森矩阵$\boldsymbol{H=\frac{\partial^2\phi(w)}{\partial w\partial w^T}}$是否正定来判断。

先计算梯度：

$$\begin{aligned}\nabla\phi(w)&=\frac{\partial\phi(w)}{\partial w}\\&=-\frac{1}{m}\sum_{i=1}^{m}\left(y_i-g(w^Tx_i)\right)x_i\\&=\frac{1}{m}X^T(g(Xw)-\vec{y})\end{aligned}$$

其中第 k 个元素为：

$$\left(\nabla\phi(w)\right)_k=-\frac{1}{m}\sum_{i=1}^{m}\left(y_i-g(w^Tx_i)\right)x_{ik}$$

则计算海森矩阵第 k 行 l 列的元素为：

$$\frac{\partial\left(\nabla\phi(w)\right)_k}{\partial w_l}=\frac{1}{m}\sum_{i=1}^{m}\frac{x_{ik}\partial g(w^Tx_i)}{\partial w_l}$$
$$=\frac{1}{m}\sum_{i=1}^{m}g(w^Tx_i)\left(1-g(w^Tx_i)\right)x_{il}x_{ik}$$

则海森矩阵可表示为：

$$\begin{aligned}H&=\nabla\nabla\phi(w)\\&=\frac{1}{m}\sum_{i=1}^{m}g(w^Tx_i)\left(1-g(w^Tx_i)\right)x_ix_i^T\\&=\frac{1}{m}\sum_{i=1}^{m}A_{ii}x_ix_i^T\\&=XAX^T\end{aligned}$$

其中，A 是对角阵，$A_{ii}=g(w^Tx_i)\left(1-g(w^Tx_i)\right)>0.$

因为$u^THu=(X^Tu)^TA(X^Tu)>0,\forall u\neq 0$,所以 H 正定，函数$\phi(w)$为凸函数，$min\phi(w)$是无约束凸优化问题。

梯度下降法：梯度下降法的思路是，选择起点和让目标函数值$\phi(w)$下降最快的方向，朝这个方向移动一个步长，反复移动直到达到最小值。这个方向就是负梯度$-\nabla\phi(w)$。可以通过下式更新 w，直到收敛（可以通过损失函数图像判断）。

$$w\coloneqq w-\frac{\alpha}{m}X^T(g(Xw)-\vec{y})$$

牛顿法：牛顿法和梯度下降法的区别在于下降方向的选择，$\phi(w)$在 w 附近的二阶泰勒近似为：

$$\hat{\phi}(w+v)=\phi(w)+\nabla\phi(w)v+\frac{1}{2}v^T\nabla^2\phi(w)v$$

将 x 视为常数，则上式是 v 的二次函数，对 v 求最小，令梯度等于零得：

$$\nabla\phi(w)+\nabla^2\phi(w)v=0\Rightarrow v=-\left(\nabla^2\phi(w)\right)^{-1}\nabla\phi(w)$$

即为$v=-H^{-1}\nabla\phi(w)$. 这个就是牛顿法选择的方向，从而有下式来更新 w：

$$w\coloneqq w-H^{-1}\nabla_{\mathrm{w}}\phi(w)$$

梯度下降法是线性收敛的，而牛顿法二次收敛，速度要快很多。

为了解决过拟合问题，可以采用正则化的方法，对参数加惩罚项，而这里的惩罚项又与 MAP 有关。

正则化：如上所述，最大似然估计是求参数θ，使似然函数$P(x|\theta)$最大。最大后验估计则是想求θ使得$P(x|\theta)P(\theta)$最大，求得的θ不单单让似然函数大，θ自己出现的先验概率也得大。此处$P(\theta)$就是惩罚项，只不过这里用的是乘法。

给定数据集$\{(x_i,y_i)\}_{i=1}^m$，对数似然函数为：

$$L(w,b)=\sum_{i=1}^{m}\log p(y_i|x_i;w,b)$$

梯度下降法更新 w：

$$w\coloneqq w-\frac{\alpha}{m}X^T(g(Xw)-\vec{y})$$

假设 w 先验分布为$w\sim N(0,\sigma)$，则后验估计函数为：

$$L\_MAP(w,b)=\log p(w)+\sum_{i=1}^{m}\log p(y_i|x_i;w,b)$$

梯度下降法更新w，对$\log p(w)$求梯度可以得到ηw. 实际应用中需要自己设置对 w 的惩罚系数，记作λ：

$$w\coloneqq w-\frac{\lambda}{m}w-\frac{\alpha}{m}X^T(g(Xw)-\vec{y})$$

2. 算法的实现

上文已经较为详细展示了公式推导，重要的步骤已经向量化，便于编程。

梯度下降法：

$$\nabla\phi(w)=\frac{1}{m}X^T(g(Xw)-\vec{y})$$

$$w\coloneqq w-\frac{\alpha}{m}X^T(g(Xw)-\vec{y})$$

```
function theta = gradientDescent(X, y, theta, alpha)

m = size(X,1);% 样本的数量
number = 0;   % 记录迭代次数
J = zeros(1); % 记录损失函数值
while true
    theta = theta - alpha / m * X' * (g(X * theta) - y);
    number = number + 1;
    J(number) = costFunctionJ(X, y, theta);
    if number > 1 && abs(J(number) - J(number - 1)) < 1e-7
        disp(number);
        figure(2)
        plot(1:number,J);
        break;
    end
end
```

```
function theta = gradientDescentRegularization(X,y,theta,alpha,lambda)

m = size(X,1);
cost = 0;  % 计算代价函数值
number = 0;  %记录迭代次数
while true
    theta = theta * (1 - alpha * lambda / m) - alpha / m * X' * (g(X * theta) - y);
    J = costFunctionJ(X, y, theta);
    number = number + 1;
    if abs(J - cost) < 1e-5
        disp(number);
        break;
    end
    cost = J;
end
```

牛顿法：

$$w := w - H^{-1}\nabla_w \phi(w)$$

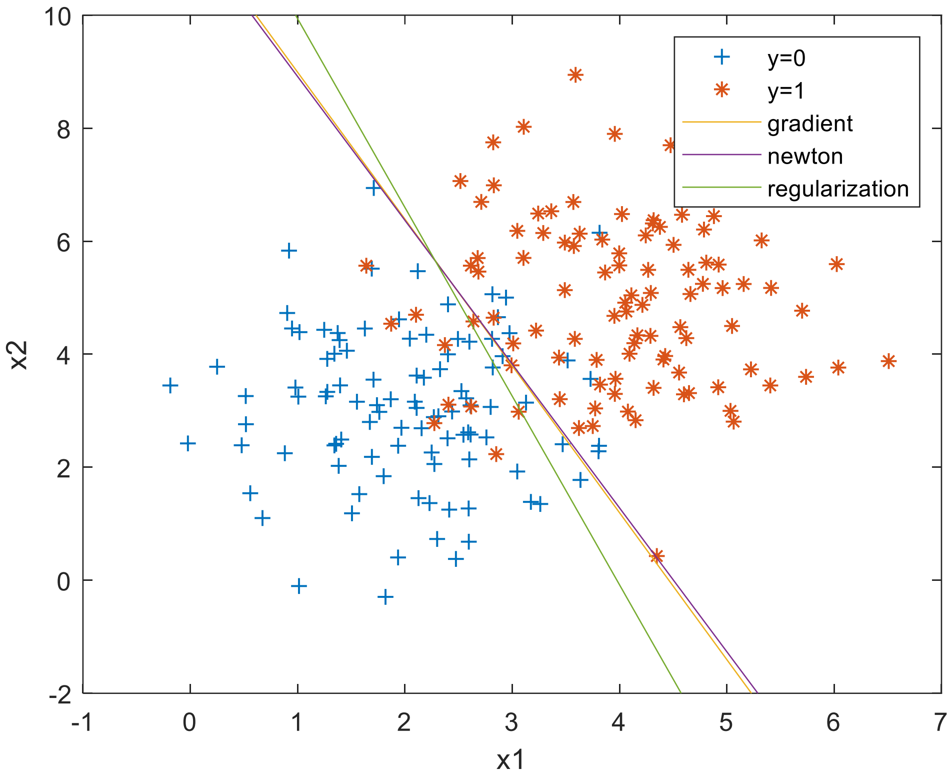
$$\begin{aligned} H &= \nabla\nabla\phi(w) \\ &= \frac{1}{m}\sum_{i=1}^{m} g(w^T x_i)\big(1 - g(w^T x_i)\big)x_i x_i^T \\ &= \frac{1}{m}\sum_{i=1}^{m} A_{ii} x_i x_i^T \\ &= XAX^T \end{aligned}$$

其中，A 是对角阵，$A_{ii} = g(w^T x_i)\big(1 - g(w^T x_i)\big)$.

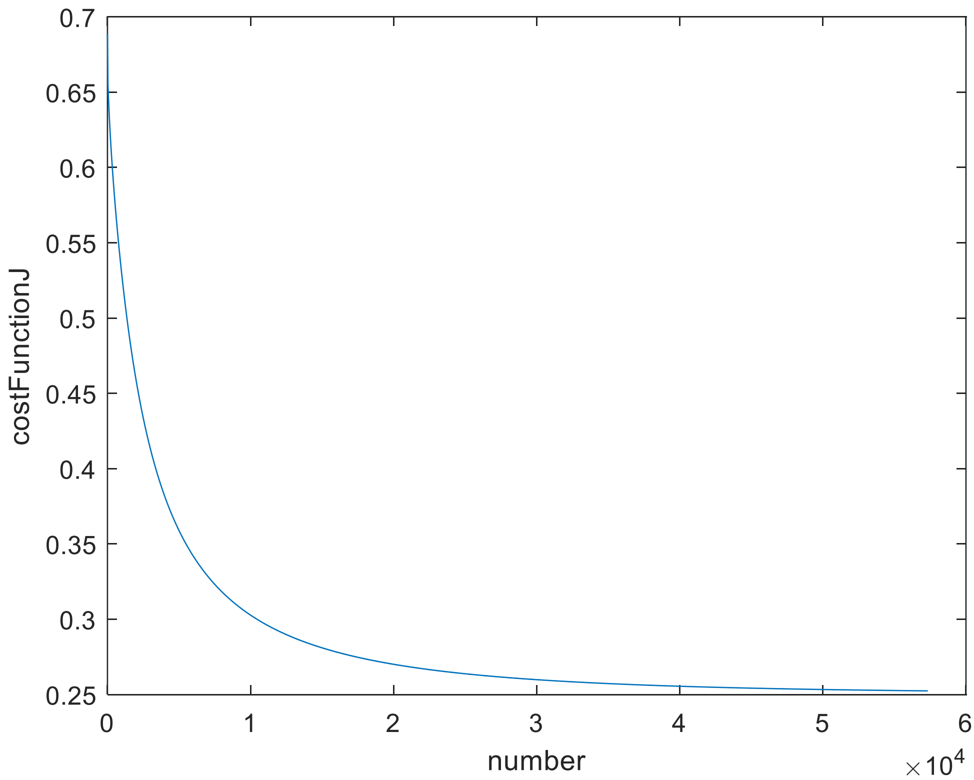
```
function theta = newton(X, y, theta)

m = size(X,1);
J = zeros(1);
number = 10;
for i = 1 : number
    h = g(X * theta);
    deltaJ = 1 / m * X' * (h - y);
    H = 1 / m * X' * diag(h) * diag(1 - h) * X;
    J(i) = costFunctionJ(X, y, theta);
    theta = theta - pinv(H) * deltaJ;
end
figure(3);
plot(1:number,J);
```

四、实验结果与分析

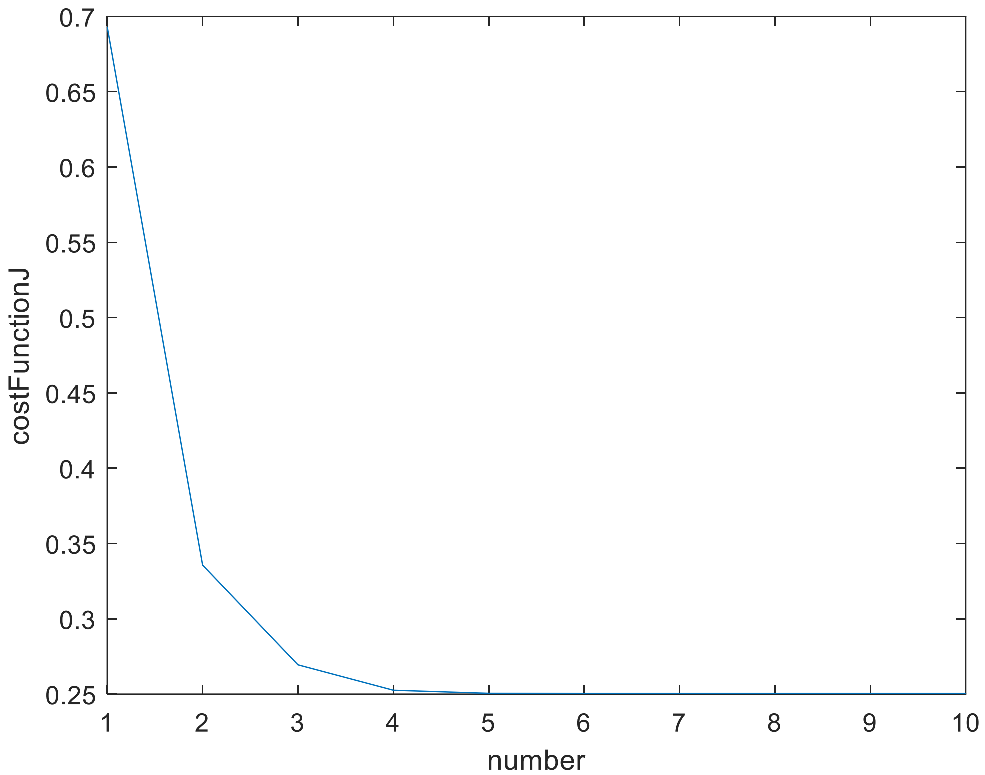
满足朴素贝叶斯假设（$\alpha = 0.01, \lambda = 0.1, \Sigma = [1\ 0; 0\ 2]$）：



两类数据点，梯度下降和牛顿法决策面几乎重合，绿色为加入正则项



梯度下降迭代次数与损失函数之间关系



牛顿法迭代次数与损失函数之间关系

牛顿法迭代几次就能收敛，而梯度下降则需要10^4次。

在训练集上的正确率：

梯度下降，未加惩罚项，共 200 个数据，分类正确的有 180 个，正确率为 90%。

梯度下降，加入正则项，共 200 个数据，分类正确的有 177 个，正确率为 88.5%。

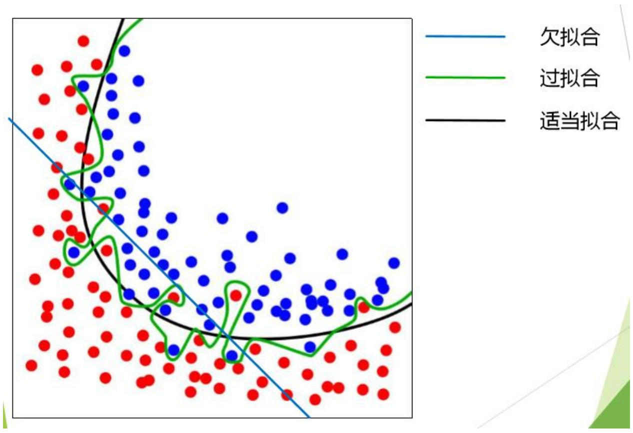
新生成的测试集上正确率：

梯度下降，未加惩罚项，91.5%，89%，90.5%

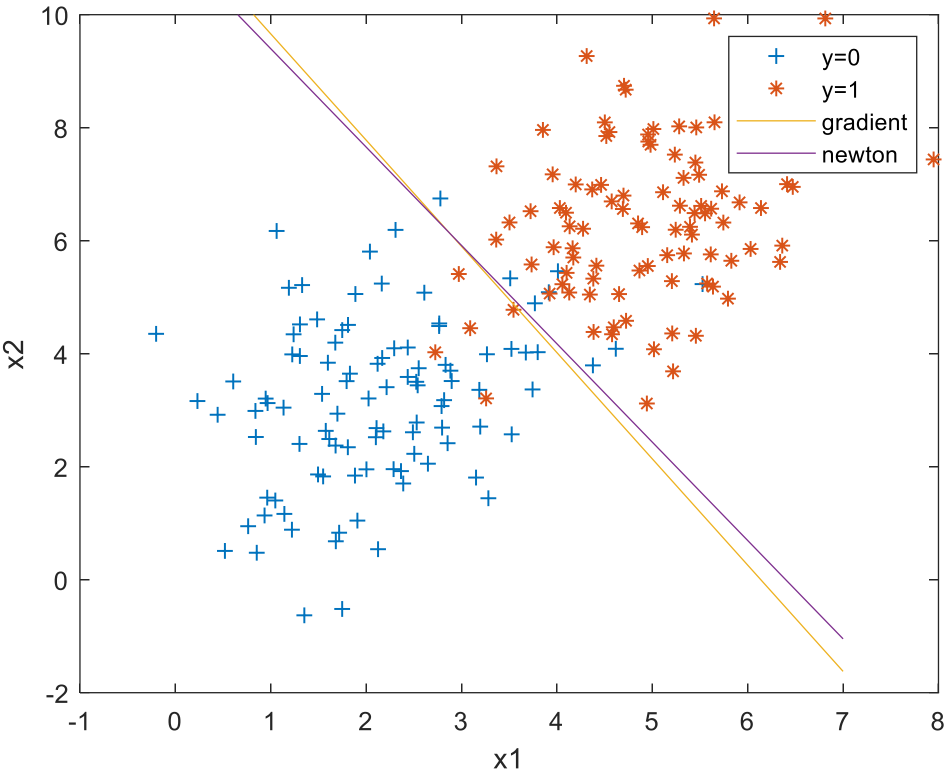
梯度下降，加入正则项，91%，88.5%，90%

我们希望得到的是在测试集上误差最小的模型，理论上加入正则项，会减少在训练集过拟合的程度，可能会造成训练集上正确率降低，但是在测试集上的“泛化误差”会减小。但遗憾的是，在本次实验中，线性分类，几乎没有过拟合，在几组测试集上没有看到加入正则项的好处。

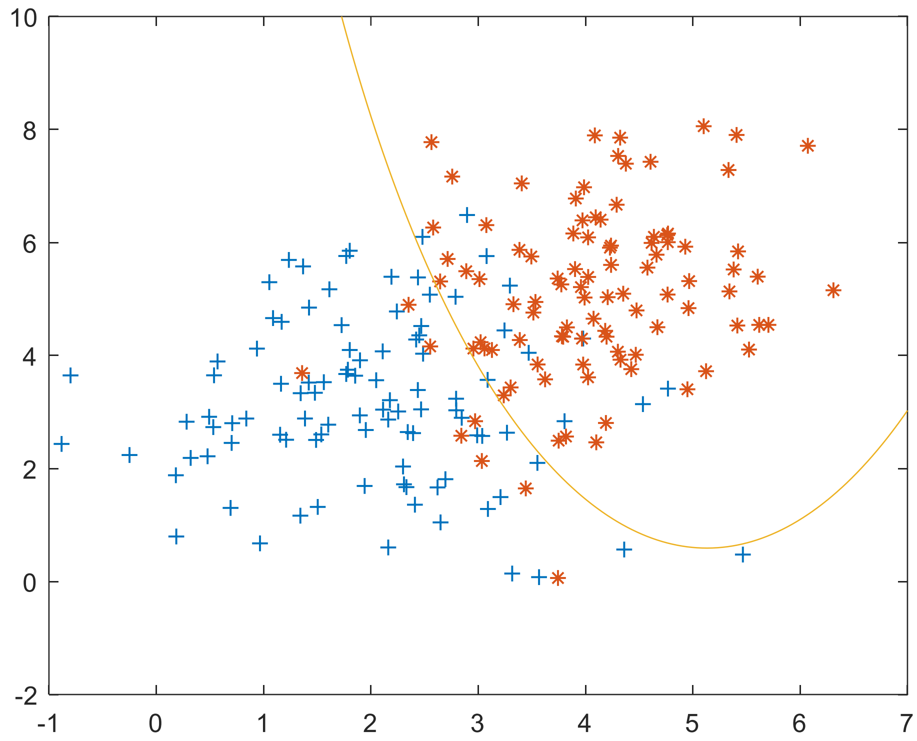
下图是欠拟合、适当拟合和过拟合的例子。如果通过非线性变化，让决策面变成非线性，增加更多高阶项，可以出现下图效果，造成过拟合，从而使得在新的测试集上“泛化误差”会很大。



不满足朴素贝叶斯假设（协方差矩阵$\Sigma = [1\ 0.5; 0.5\ 2]$）：

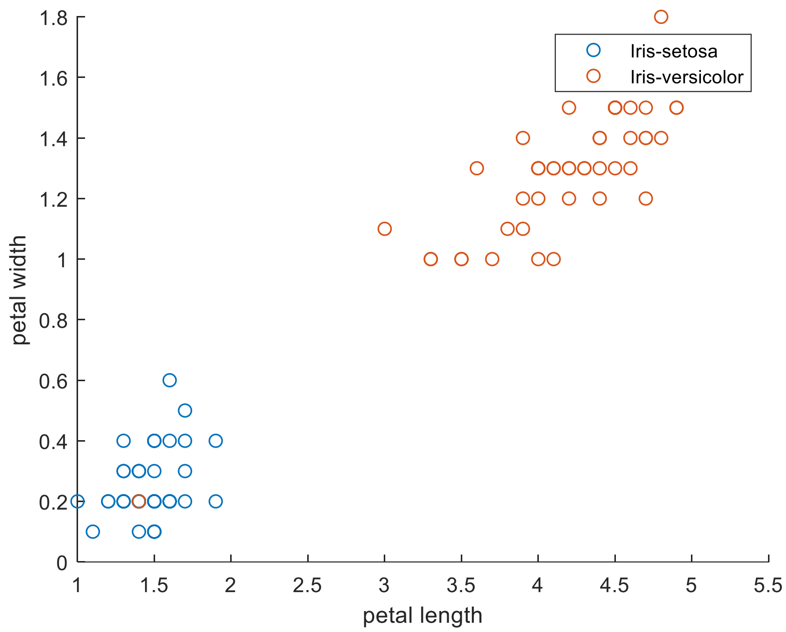


即使数据不满足朴素贝叶斯假设，直观上看 logistic 也能够很好的区分这两类数据。



如果数据线性不可分，作非线性变换，产生曲线决策面

UCI 数据集：使用 iris 数据集，共 150 个样本，三类，每个样本包括四个特征和一个类标签。使用前两组数据，它们线性可分。



使用后两个特征作图

每个类别有 50 个数据，可以留出一部分作为测试集。可以用每个类别的 45

个样本作为训练集，另外 5 个作为测试集。

1	4.8,3.0,1.4,0.3,Iris-setosa	0.0049
2	5.1,3.8,1.6,0.2,Iris-setosa	0.0010
3	4.6,3.2,1.4,0.2,Iris-setosa	0.0030
4	5.3,3.7,1.5,0.2,Iris-setosa	0.0008
5	5.0,3.3,1.4,0.2,Iris-setosa	0.0018
6	5.7,3.0,4.2,1.2,Iris-versicolor	0.9973
7	5.7,2.9,4.2,1.3,Iris-versicolor	0.9982
8	6.2,2.9,4.3,1.3,Iris-versicolor	0.9982
9	5.1,2.5,3.0,1.1,Iris-versicolor	0.9485
10	5.7,2.8,4.1,1.3,Iris-versicolor	0.9979

测试集与预测值，在测试集上准确率 100%

五、结论

1. 根据实验结果，如果数据满足条件独立假设，Logistic Regression 能够取得非常好的效果；当数据不满度条件独立假设时，Logistic Regression 仍然能够通过调整参数让模型最大化的符合数据的分布，从而训练得到在现有数据集下的一个最优模型。

2. 牛顿法二次收敛，速度极快，迭代次数可以在 10 以内，梯度下降是线性收敛的，在本次实验中而梯度下降法迭代收敛次数的数量级为10^4。

六、参考文献

1. https://blog.csdn.net/u011508640/article/details/72815981
2. https://zhuanlan.zhihu.com/p/67842740
3. https://www.cnblogs.com/bingjianing/p/9117330.html
4. 周志华.(2016).机器学习.清华大学出版社.北京
5. 刘杨.(2019).机器学习课件(5).哈尔滨工业大学.哈尔滨
6. 吴恩达.网易云课堂机器学习课程课时 50 参考资料

七、附录：源代码（带注释）

1. 产生数据

满足朴素贝叶斯假设：

```
function [R1, R2] = productData()

number = 100;                % 产生数据个数
mu = [2 3];                  % 生成数据的均值
sigma = [1 0;0 2];           % 数据的协方差矩阵
R1 = mvnrnd(mu, sigma, number); % y = 0
plot(R1(:,1),R1(:,2),'+');
hold on;

mu = [4 5];
sigma = [1 0; 0 2];
R2 = mvnrnd(mu, sigma, number); % y = 1
plot(R2(:,1),R2(:,2),'*');
```

不满足朴素贝叶斯假设：

```
function [R1, R2] = productDataNot()

number = 100;                % 产生数据个数
mu = [2 3];                  % 生成数据的均值
sigma = [1 0.5;0.5 2];           % 数据的协方差矩阵
R1 = mvnrnd(mu, sigma, number); % y = 0
plot(R1(:,1),R1(:,2),'+');
hold on;

mu = [5 6];
R2 = mvnrnd(mu, sigma, number); % y = 1
plot(R2(:,1),R2(:,2),'*');
```

2. sigmoid 函数

```
function h = g(z)
h = 1 ./ (1 + exp(-z));
```

3. 损失函数

```
function J = costFunctionJ(X, y, theta)

m = size(X,1);
J = 1 / m * (-y' * log(g(X * theta)) - (1 - y)' * log(1 - g(X * theta)));
```

4. 梯度下降

```
function theta = gradientDescent(X, y, theta, alpha)

m = size(X,1);% 样本的数量
number = 0;   % 记录迭代次数
J = zeros(1); % 记录损失函数值
while true
    theta = theta - alpha / m * X' * (g(X * theta) - y);
    number = number + 1;
    J(number) = costFunctionJ(X, y, theta);
    if number > 1 && abs(J(number) - J(number - 1)) < 1e-7
        disp(number);
        figure(2)
        plot(1:number,J);
        break;
    end
end
```

5. 带正则项的梯度下降

```
function theta = gradientDescentRegularization(X,y,theta,alpha,lambda)

m = size(X,1);
cost = 0;  % 计算代价函数值
number = 0;  %记录迭代次数
while true
    theta = theta * (1 - alpha * lambda / m) - alpha / m * X' * (g(X * theta) - y);
    J = costFunctionJ(X, y, theta);
    number = number + 1;
    if abs(J - cost) < 1e-5
        disp(number);
        break;
    end
    cost = J;
end
```

6. 牛顿法

```
function theta = newton(X, y, theta)

m = size(X,1);
J = zeros(1);
number = 10;
for i = 1 : number
    h = g(X * theta);
    deltaJ = 1 / m * X' * (h - y);
    H = 1 / m * X' * diag(h) * diag(1 - h) * X;
    J(i) = costFunctionJ(X, y, theta);
    theta = theta - pinv(H) * deltaJ;
end
figure(3);
plot(1:number,J);
```

7. 命令行输入

```
------------------------------------------------------------满足朴素贝叶斯
number = 100;
[R1,R2] = productData();
R1 = [ones(number,1) R1];
R2 = [ones(number,1) R2];
theta = zeros(size(R1,2),1);
alpha = 0.01;
y = [zeros(number,1);ones(number,1)];
X = [R1;R2];
---------------------不加惩罚项，梯度下降
ntheta = gradientDescent(X, y, theta, alpha);
h = g(X * ntheta);
figure(1);
f = @(x1,x2)ntheta(1) + ntheta(2) * x1 + ntheta(3) * x2;
fimplicit(f,[0 7 -2 10]);
---------------------牛顿法
newtontheta = newton(X,y,theta);
figure(1);
f = @(x1,x2)newtontheta(1) +newtontheta(2) * x1 + newtontheta(3) * x2;
fimplicit(f,[0 7 -2 10]);
----------------------正则化
lambda = 0.1;
rgtheta = gradientDescentRegularization(X,y,theta,alpha,lambda);
figure(1);
f = @(x1,x2)rgtheta(1) + rgtheta(2) * x1 + rgtheta(3) * x2;
fimplicit(f,[0 7 -2 10]);
```

```
------------------------------------------------------------不满足朴素贝叶斯
number = 100;
[R1,R2] = productDataNot();
R1 = [ones(number,1) R1];
R2 = [ones(number,1) R2];
theta = zeros(size(R1,2),1);
alpha = 0.01;
y = [zeros(number,1);ones(number,1)];
X = [R1;R2];
---------------------不加惩罚项，梯度下降
ntheta = gradientDescent(X, y, theta, alpha);
h = g(X * ntheta);
figure(1);
f = @(x1,x2)ntheta(1) + ntheta(2) * x1 + ntheta(3) * x2;
fimplicit(f,[0 7 -2 10]);
---------------------牛顿法
newtontheta = newton(X,y,theta);
figure(1);
f = @(x1,x2)newtontheta(1) +newtontheta(2) * x1 + newtontheta(3) * x2;
fimplicit(f,[0 7 -2 10]);
-----------------------------------------------------------------
------------------------------------------------------------------------------非线性变换
number = 100;
[R1,R2] = productData();
R1 = [ones(number,1) R1 R1.^2];
R2 = [ones(number,1) R2 R2.^2];
theta = zeros(size(R1,2),1);
alpha = 0.01;
y = [zeros(number,1);ones(number,1)];
X = [R1;R2];

ntheta = gradientDescent(X, y, theta, alpha);
h = g(X * ntheta);
figure(1);
f = @(x1,x2)ntheta(1) + ntheta(2) * x1 + ntheta(3) * x2 + ntheta(4) * x1.^ 2 +
ntheta(5) * x2.^2;
fimplicit(f,[0 7 -2 10]);
-------------------------------------newton method

newtontheta = newton(X,y,theta);
figure(1);
f = @(x1,x2)newtontheta(1) +newtontheta(2) * x1 + newtontheta(3) * x2 + newtontheta(4)
* x1.^ 2 + newtontheta(5) * x2.^2;
```

```
fimplicit(f,[0 7 -2 10]);
```
---UCI数据集

```
[attrib1, attrib2, attrib3, attrib4, class] = textread('iris.data', '%f%f%f%f%s', 'delimiter', ',');
scatter(attrib3(1:50,1),attrib4(1:50,1));
hold on;
scatter(attrib3(50:100,1),attrib4(50:100,1));
scatter(attrib3(100:150,1),attrib4(100:150,1));
X = [ones(100,1) attrib1(1:100,:) attrib2(1:100,:) attrib3(1:100,:) attrib4(1:100,:)];
y = zeros(100,1);
for i = 1 : 100
    if strcmp(class(i),'Iris-setosa')
        y(i) = 0;
    else
        y(i) = 1;
    end
end

theta = zeros(size(X,2),1);
alpha = 0.1;
ntheta = gradientDescent(X, y, theta, alpha);
h = g(X * ntheta);
```
--数据集与测试集

```
[attrib1, attrib2, attrib3, attrib4, class] = textread('trainingset.txt', '%f%f%f%f%s', 'delimiter', ',');
>> [tattrib1, tattrib2, tattrib3, tattrib4, tclass] = textread('testset.txt', '%f%f%f%f%s', 'delimiter', ',');
X = [ones(90,1) attrib1(:) attrib2(:) attrib3(:) attrib4(:)];
y = zeros(90,1);
for i = 1 : 90
    if strcmp(class(i),'Iris-setosa')
        y(i) = 0;
    else
        y(i) = 1;
    end
end

theta = zeros(size(X,2),1);
alpha = 0.01;
ntheta = gradientDescent(X, y, theta, alpha);
Xtest=[ones(10,1) tattrib1(:) tattrib2(:) tattrib3(:) tattrib4(:)];
h=g(Xtest * ntheta);
```