深 圳 大 学 实 验 报 告

课 程 名 称：机器学习

实验项目名称：面向城市房价的线性回归/逻辑回归预测

学　　院：计算机与软件学院

专　　业：计算机科学与技术

指 导 教 师：贾森 徐萌

报告人：刘睿辰 学号：2018152051 班级：数计班

实 验 时 间：2021.3.8-2021.3.31

实验报告提交时间：2021.3.31

教务处制

一、 实验目标：

1. 理解监督学习的基本过程；
2. 掌握线性回归/逻辑回归方法中的模型选择和参数估计方法；
3. 利用真实的城市房价数据，掌握线性回归/逻辑回归方法，实现房价的有效预测。

二、实验环境：

1. 操作系统：*windows* 10。
2. 实验平台：*Matlab R*2019*a*。

三、实验内容、步骤与结果

1、根据 ex1.rar 和 ex2.rar 文件中的英文 pdf 文档，完成文档中的实验内容。

1.1 线性回归 (linear regression)

1.1.1 线性回归的数学基础

线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，运用十分广泛。回归分析中，只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为**一元线性回归分析**(linear regression with one variable)。如果回归分析中包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，则称为**多元线性回归分析**(linear regression with multiple variables)。

线性回归的基本格式如下所示：

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_{n-1} x_{n-1}$$

以二维空间为例，我们利用线性回归可以得到二维空间的一条直线，其表达式为

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

其中x_1为自变量，$h_\theta(x)$为因变量。

在得到了线性回归的基本表达式之后，为了评价拟合效果的优劣，我们引入一个代价函数(cost function)，表达式为：

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

这里m代表散点的个数（训练集的规模），$y^{(i)}$代表第i个点的真正的函数值，$h_\theta\left(x^{(i)}\right)$则是我们回归出来的线性方程在$x^{(i)}$处的取值。该表达式不难理解，衡量的关系就是拟合方程和真实点之间在y方向的几何距离，这也就反映了点与曲线的偏离程度，如图 1 所示。

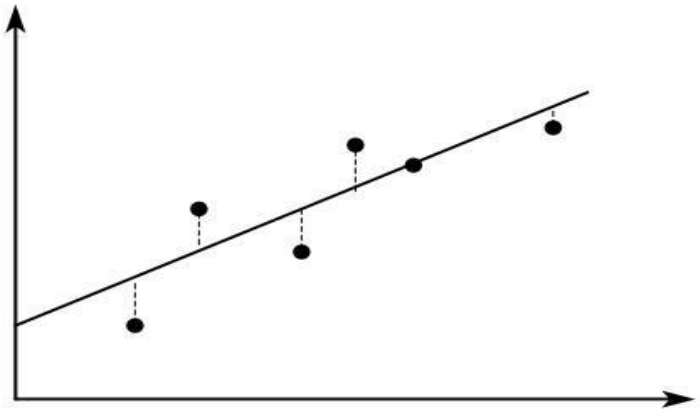


图 1. 各点与拟合直线的几何距离，由虚线表示

综上所述，我们要考虑的问题如下：
Task: 利用线性回归拟合出散点的线性关系；

Performance Measure: $\min\limits_{\theta_0,\theta_1,...,\theta_{n-1}} J(\theta_0,\theta_1,...,\theta_{n-1})$；

Experience: 训练集，也就是点的集合$\{(x^{(i)},y^{(i)})|\ i=1,...,m\}$。

为了尽可能地使我们的回归效果达到最佳，我们可以采用迭代的方法逐步逼近最优解。这里面我们采用梯度下降法 (gradient descent)来进行迭代。

梯度下降法：从几何上来说，在代价函数$J(\theta_0,\theta_1,...,\theta_{n-1})$的曲面上，任意给定一个起始点$\theta_0,\theta_1,...,\theta_{n-1}$，沿着梯度下降的方向，即对代价函数$J(\theta_0,\theta_1,...,\theta_{n-1})$中的$\theta_0,\theta_1,...,\theta_{n-1}$分别求偏导数的方向，按照一定的步长$\alpha$进行不断迭代，直到$\theta_0,\theta_1,...,\theta_{n-1}$收敛，也就是达到局部或全局最低点。
为了更好地理解这个方法，我们从直观的角度进行解释说明。比如某人准备下山，他每次向下迈出一步，就测量此时山坡的坡度，然后按照一定的步长逐渐下山，那么他最后可能到达山脚，也有可能到达了一个非山脚的平地，也就是到达了某一个最低点。
了解了梯度下降法的原理，我们不难发现我们需要对代价函数求偏导数，这样就确定了各个维度下降的方向。对各分量求偏导可得：

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j}=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j^{(i)}$$

然后按照步长α，我们可以求得迭代公式：

$$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j^{(i)}$$

最初的$\boldsymbol{\theta}$可以设置为各分量都是 0 的列向量，然后通过迭代公式来求得每一轮新的$\boldsymbol{\theta}$向量，实现逐步逼近的目的。

此外，对于多元线性回归，每一个$x^{(i)}$都是由多个分量来组成的，也就是有

$$x^{(i)}=\begin{pmatrix}x_1^{(i)}\\x_2^{(i)}\\\vdots\\x_n^{(i)}\end{pmatrix}$$

所以根据之前的代价函数，将其写成矩阵形式我们有：

$$J(\boldsymbol{\theta})=\frac{1}{2m}\left(X\boldsymbol{\theta}-\vec{y}\right)^T\left(X\boldsymbol{\theta}-\vec{y}\right)$$

其中有$X=\begin{pmatrix}\left(x^{(1)}\right)^T\\\left(x^{(2)}\right)^T\\\vdots\\\left(x^{(m)}\right)^T\end{pmatrix}$，$\boldsymbol{\theta}=\begin{pmatrix}\theta_0\\\theta_1\\\vdots\\\theta_{n-1}\end{pmatrix}$，同时$\vec{y}=\begin{pmatrix}y^{(1)}\\y^{(2)}\\\vdots\\y^{(m)}\end{pmatrix}$。

之前的方法都是通过迭代来解决的。事实上，通过非迭代方法也可以求解$\boldsymbol{\theta}$。

事实上，由于

$$X\boldsymbol{\theta}=\vec{y}$$

那么由于X不一定是方阵，所以将其方阵化，得到

$$X^TX\boldsymbol{\theta}=X^T\vec{y}$$

对于$\forall x\in R^{n\times1}$，有

$$x^TX^TXx=(Xx)^TXx\geqslant0$$

所以X^TX就是半正定矩阵，在X^TX正定的情况下X^TX可逆，则有

$$\boldsymbol{\theta}=(X^TX)^{-1}X^T\vec{y}$$

我们把这个式子称之为正规方程 (normal equations)，通过这个方法免除了迭代的复杂，
现在我们总结一下梯度下降法和正规方程法的优劣：
毫无疑问，正规方程免除了迭代的复杂性，同时也不需要对参数的设置，比如梯度下降法中的α值，但是如果X^TX是一个规模很大的矩阵，也就是样本点很多而且样本点维度都很高的时候，该矩阵的计算就非常复杂，况且还要求它的逆矩阵，这是非常大的计算量。所以该方法只针对规模较小且维度不太高的情况。在样本点很多而且样本点维度都很高的情况下，梯度下降法的效率不会受影响，但是迭代次数多也造成计算量比较大，而且它需要参数的估计，比如α的取值就是需要进行估计的。

1.1.2 线性回归的实践 1：快餐车利润预测

问题描述：我们如今有这样一组数据，即快餐车(food truck)与城市人口数的样本数据，现在我们需要对这二者之间的关系进行刻画。很明显这是一个一元线性回归问题，我们只需要使用梯度下降法得出一个函数表达式即可。

首先将样本数据以散点的形式进行可视化，观察其是否符合线性关系，如图 2 所示。

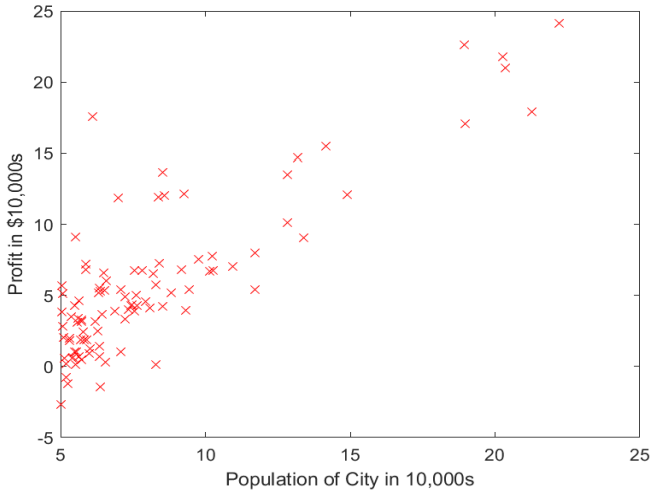


图 2. 散点图

从图中可以大致看出散点符合一次函数的分布规律。

所以应用梯度下降法，由于这是二维空间，则有

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

这里设置$\boldsymbol{\theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix}$，$X = (x^{(1)} \quad x^{(2)})^T = (1 \quad x)^T$

这里代价函数的公式为

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

迭代公式为

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x$$

然后取$\alpha = 0.01$，迭代次数为 1500 次，得到最终的$\boldsymbol{\theta}$的值。

接下来我们根据给出的代码框架，得知$ex1.m$文件相当于主函数，他需要调用$gradientdescent.m$文件来计算最终的结果。而$gradientdescent.m$文件也需要调用$computecost.m$文件来逐步使用更新的值来计算代价。所以函数调用关系如图 3 所示。

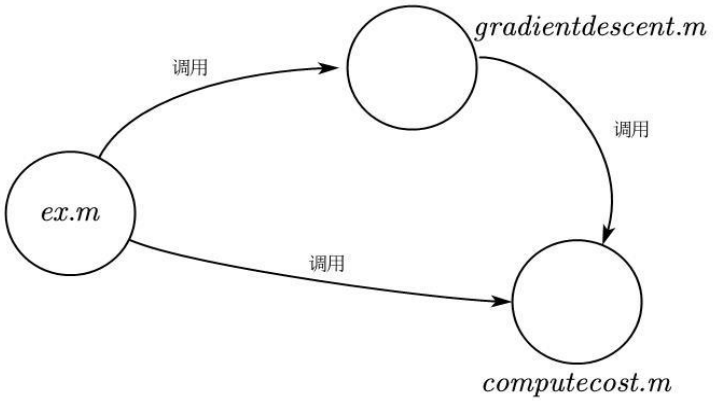


图 3. 函数调用关系

了解了函数调用关系之后，我们主要完成$gradientdescent.m$以及$computecost.m$文件的内容。

首先，将前面$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$公式代入，即可完成$computecost.m$文件。主要部分的 MATLAB 版本代码如图 4 所示。

```
tmp1 = theta(1,1);
tmp2 = theta(2,1);

sum = 0;
for j=1:m
    sum = sum+(tmp1+tmp2*X(j,2)-y(j))*(tmp1+tmp2*X(j,2)-y(j));
end

J = sum/(2*m);
```

图 4. 一元线性回归代价函数求解

然后，我们根据之前推导过的迭代公式来实现梯度下降。源代码如图 5 所示，通过循环来实现迭代的过程。

```
for iter = 1:num_iters

    sum1 = 0;
    for j=1:m
        sum1 = sum1+(theta(1,1)+theta(2,1)*X(j,2)-y(j))*1;
    end

    sum2 = 0;
    for j=1:m
        sum2 = sum2+(theta(1,1)+theta(2,1)*X(j,2)-y(j))*X(j,2);
    end

    theta(1,1) = theta(1,1)-alpha*(1/m)*sum1;
    theta(2,1) = theta(2,1)-alpha*(1/m)*sum2;
    % ============================================================
    % Save the cost J in every iteration
    J_history(iter) = computeCost(X, y, theta);

end
```

图 5. 一元线性回归梯度下降实现

上述代码编写完毕之后，我们可以得出函数解析式为$(1 \quad x)\begin{pmatrix}\theta_0\\ \theta_1\end{pmatrix}$，我们绘制出这条直线，得到图 6 的拟合结果。

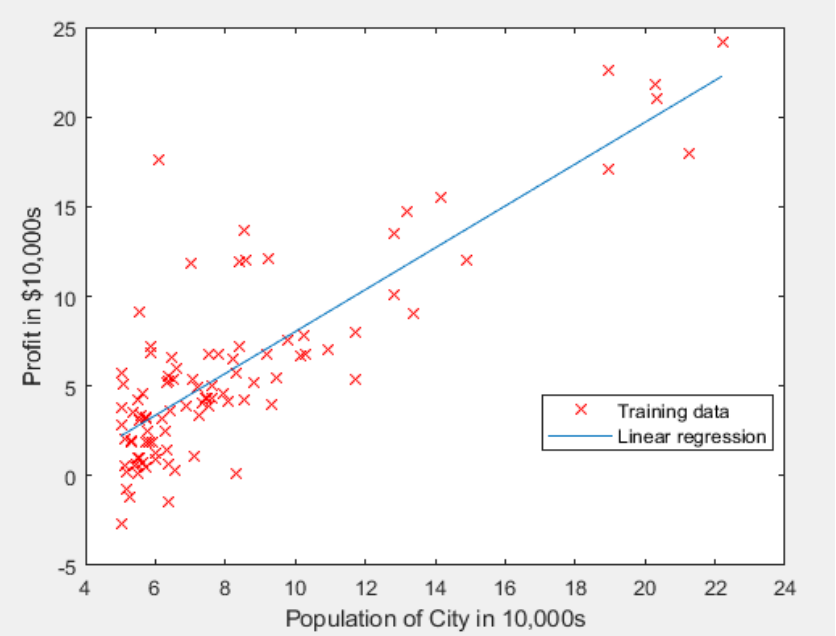


图 6. 一元线性回归结果

根据返回的$\boldsymbol{\theta}$向量的值，我们可以求得函数解析式为

$$h_\theta(\mathbf{x}) = (1 \quad x)\begin{pmatrix}-3.630291\\ 1.166362\end{pmatrix}$$

根据解析式，我们可以进行预测。如，当人口数为 35000 的时候，利润值为

$$(1 \quad 3.5)\begin{pmatrix}-3.630291\\ 1.166362\end{pmatrix} = 0.451976 \times 10^4 = \$4519.76$$

同理，当人数为 700000 的时候，同样的方法预测利润值为$\$45342.450129$

MATLAB 的运行结果与我们的计算完全吻合，如图 7 所示。

```
For population = 35,000, we predict a profit of 4519.767868
For population = 70,000, we predict a profit of 45342.450129
```

图 7. 用线性表达式进行预测的结果

此外，为了观察代价函数的分布规律，我们可以绘制代价函数的图象。我们知道$J(\theta_0,\theta_1)$是一个三维函数，所以我们可以绘制该函数表面或者绘制函数的等高线。

这里我们取区间范围$\theta_0 \in [-10,10]$，$\theta_1 \in [-1,4]$，然后对应不同位置求得$J(\theta_0,\theta_1)$的值，绘制函数表面，如图 8 所示。

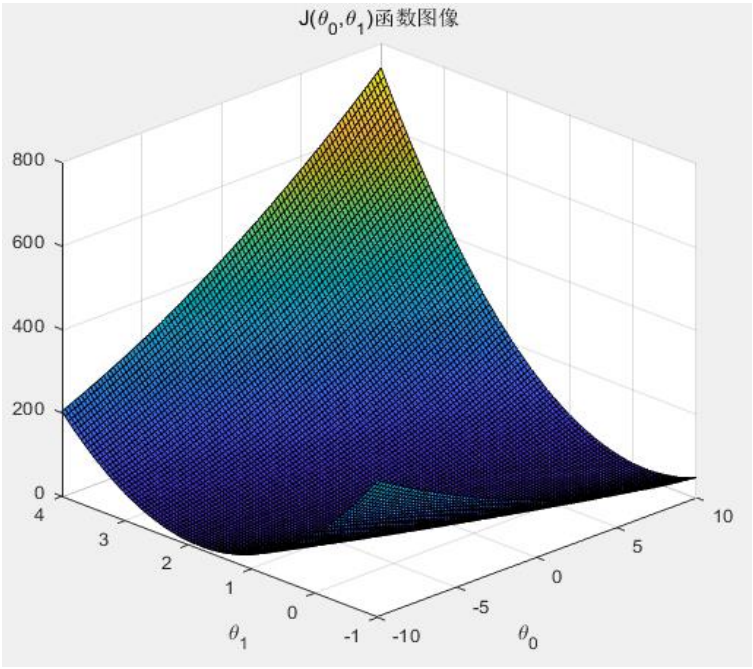


图 8. 函数$J(\theta_0, \theta_1)$表面图

我们从图 8 中可以看出函数存在一个全局最低点，这个点必然就是代价最小点。为了观察最优解范围以及点的迭代过程，我们绘制等高线图来观察点的位置，如图 9 所示。

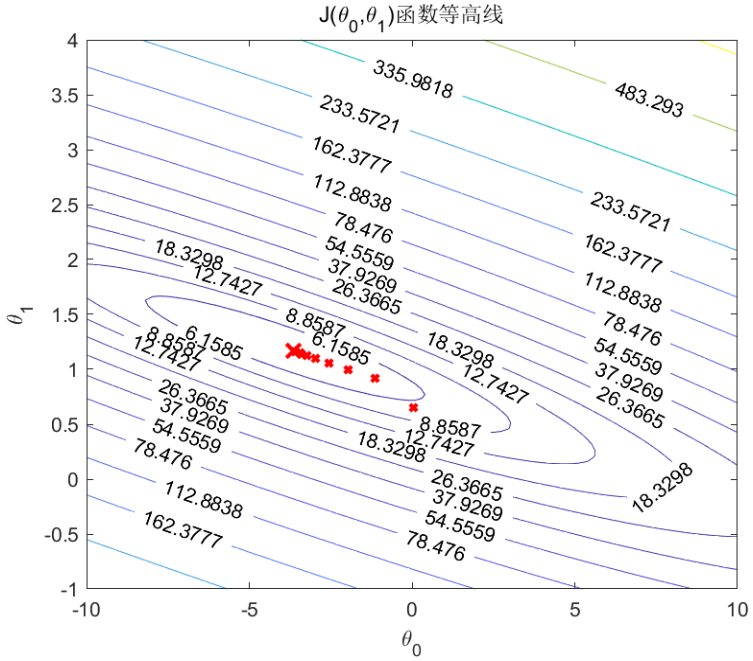


图 9. 函数$J(\theta_0, \theta_1)$等高线图

根据 MATLAB 的运行结果，我们得知最优解存在于$(-3.630291, 1.166362)$，这与图 9 中最优解的位置是吻合的。

1.1.3 线性回归的实践 2：房价预测

问题描述：我们如今有这样一组数据，刻画了房价和两个变量之间的关系，这两个变量分别

是房子大小以及卧室数目。我们需要做的就是通过数据集中这两个因素对房价进行预测。
对于多元回归分析，我们一般来说会先进行标准化。为什么要进行标准化呢？
事实上，由于多元回归涉及到不同的变量，而不同的变量之间单位是不同的，为了更加直观地比较，也就是我们更想直接用数值进行互相比较，所以进行数据标准化是十分有效的。一个最简单的例子就是，100 这个数值，在身高(cm)中可以说比较矮，但是放在体重(kg)中就非常重了。
此外，数据标准化之后可以加快收敛速度。由于不同的量之间存在区别，所以损失函数的等高线图可能是椭圆形。由于梯度方向垂直于等高线，可能不会指向局部最优。对特征进行特征化之后，其损失函数的等高线图更接近圆形，梯度下降的方向震荡更小，所以收敛更快。
标准化我们可以采用很多形式，这里我们取 Z-score Normalization，公式为：

$$x^* = \frac{x - \bar{x}}{\sigma}$$

应用到这里，我们需要对房子大小以及卧室数目进行标准化。这一部分比较简单，分别求两个量的平均值以及标准差进行转化即可。
在数据进行了处理之后，我们的操作和之前是一样的，需要计算代价函数并使用梯度下降法对每一个维度进行迭代求解。这里使用梯度下降进行迭代的代码如图 10 所示。

```
for iter = 2:num_iters
    % n-1 is the number of variables
    % m is the size of samples
    n = size(X,2);
    for i=1:n
        Sum = 0;
        for j=1:m
            sum = X(j,:) * theta;
            Sum = Sum + (sum - y(j)) * X(j,i);
        end
        theta1(i,1) = theta(i,1) - alpha * (1/m) * Sum;
    end
    theta = theta1;
    % Save the cost J in every iteration
    J_history(iter) = computeCostMulti(X, y, theta);
end
```

图 10. 多元线性回归进行迭代

这里面我们需要用$J_history$数组保存每次迭代的代价值，以便后期我们分析代价函数的收敛规律。

通过多次检验，我们设置$\alpha = 0.1$并迭代 400 次，如图 11 所示，我们看到了代价函数的值随着迭代次数的变化。从图中我们可以看出，迭代 50 次之后J值的变化不明显，所以 50 次之后趋于稳定。通过分析代价函数随迭代次数的变化可以确定迭代的次数，从而减少不必要的计算。

最终的$\boldsymbol{\theta}$确定之后就可以进行预测。例如，给定面积为 1650 square feet、卧室数量为 3 的房子，我们可以进行估价。这里要注意，$\boldsymbol{\theta}$的值是根据标准化的样本求得的，所以不能直接代入计算，应该也将 1650 和 3 分别进行标准化处理之后才能代入计算，经过 MATLAB 验证，

该房子的价格为293081.464622。

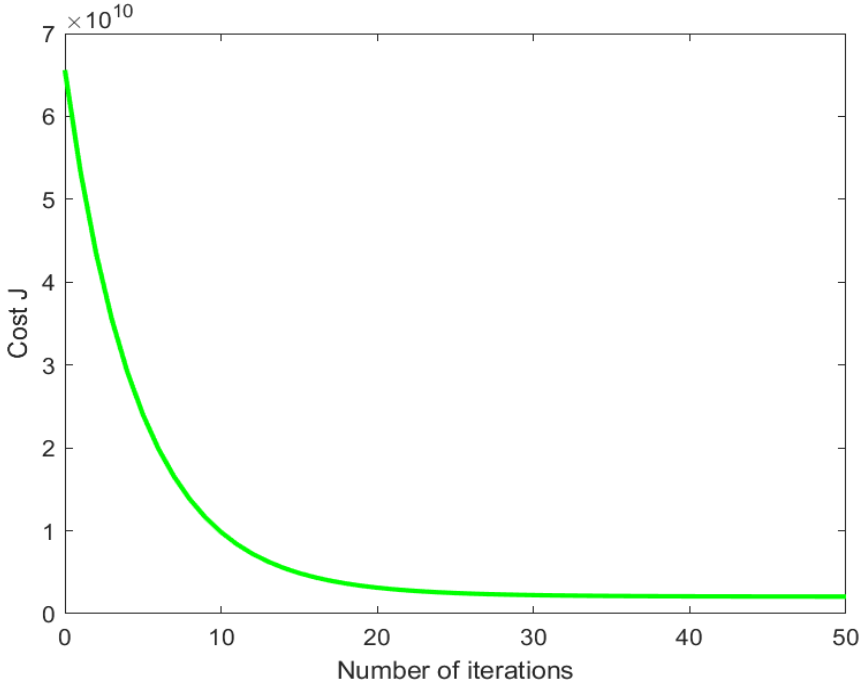


图 11. 多元线性回归代价函数随迭代次数的变化

我们也可以用正规方程来进行预测。正规方程法免去了迭代的复杂性，代码也比较简单。这里值得一提的是，类似$Ax=b$这种线性方程，如果A是可逆矩阵，那么可用$x=inv(A).b$来求解。但实际上，A^{-1}是一个理论值，用 MATLAB 求解的话速度比较慢而且不准确。MATLAB 比较建议使用除法计算。也就是使用$A\backslash b$来计算。

应用到这里，那就是

$$\boldsymbol{\theta}=(X^T X)\backslash(X^T y)$$

应用这个方法计算出房价为$\$293081.464335$，和多元回归计算出来的$\$293081.464622$相差无几。

1.2 逻辑回归 (Logistic Regression)

1.2.1 逻辑回归的数学基础

逻辑回归虽称之为回归，但实际上是一种二分类模型。我们首先要讨论一下关于 Logistic 分布的问题。

Logistic 分布实际上是一种连续型分布，它的分布函数为：

$$F(x)=P(X\leqslant x)=\frac{1}{1+e^{-\frac{x-\mu}{\gamma}}}$$

函数在$x=\mu$的时候概率为$\frac{1}{2}$。

那么如何将此用于分类问题呢？

对于二分类问题，我们考虑两个类，也就是$y=0$和$y=1$两个类。那么拿到了一个样本，它属于哪个类呢？我们的思想是估计它属于某个类的概率。最简单的情况，我们设想一下，如果存在一个分界值，当样本值大于这个分界值的时候并入$y=1$这一组，小于分界值的时候并入$y=0$这一组，那么我们考虑阶跃函数

$$P(x)=\begin{cases}0 & x<0\\ 0.5 & x=0\\ 1 & x>0\end{cases}$$

事实上，阶跃函数存在间断点，这就使得我们需要一个函数具有好的性质来替换这个阶跃函数。我们发现 Logistic 分布函数在$\mu=0$，γ趋近于 0 的时候化为阶跃函数，所以我们可以用 Logistic 分布函数来替换阶跃函数，我们把这个函数称之为 sigmoid 函数。其基本形式为：

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

我们用 MATLAB 去绘制函数图象，如图 12 所示。

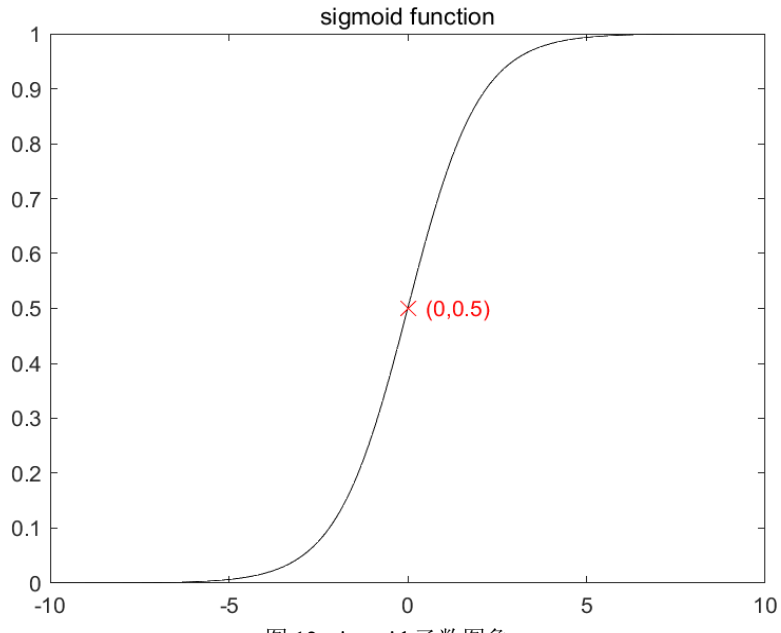


图 12. sigmoid 函数图象

根据函数图像，我们得到 sigmoid 函数的基本性质如下：

（1） sigmoid 函数连续可微分，不具有断点；

（2） sigmoid 函数在原点取值为 0.5，当自变量大于 0 则函数值大于 0.5，否则小于 0.5；

（3） sigmoid 函数存在阈值，值域为$[0,1]$，且单调分布。

如果我们拟合出一个函数曲线$\theta^T x$作为分类标准，样本点在曲线之上就是 1 类，样本点在曲线下就是 0 类，那么样本点在 1 类的概率就是$g(\theta^T x)$。我们可以这样理解，样本点与分类器的距离如果大于 0，也就是$\theta^T x > 0$，在这种情况下，如果距离分类器越远，属于该类的概率就越大。反之，如果$\theta^T x < 0$，在这种情况下，如果距离分类器越远，属于该类的概率就越小。这一点在 sigmoid 函数中也可以体现，如图 13 所示。

所以我们有下式：

$$h_\theta(\boldsymbol{x}) = P(y=1 \mid \boldsymbol{x}) = \frac{1}{1+e^{-\theta^T \boldsymbol{x}}}$$

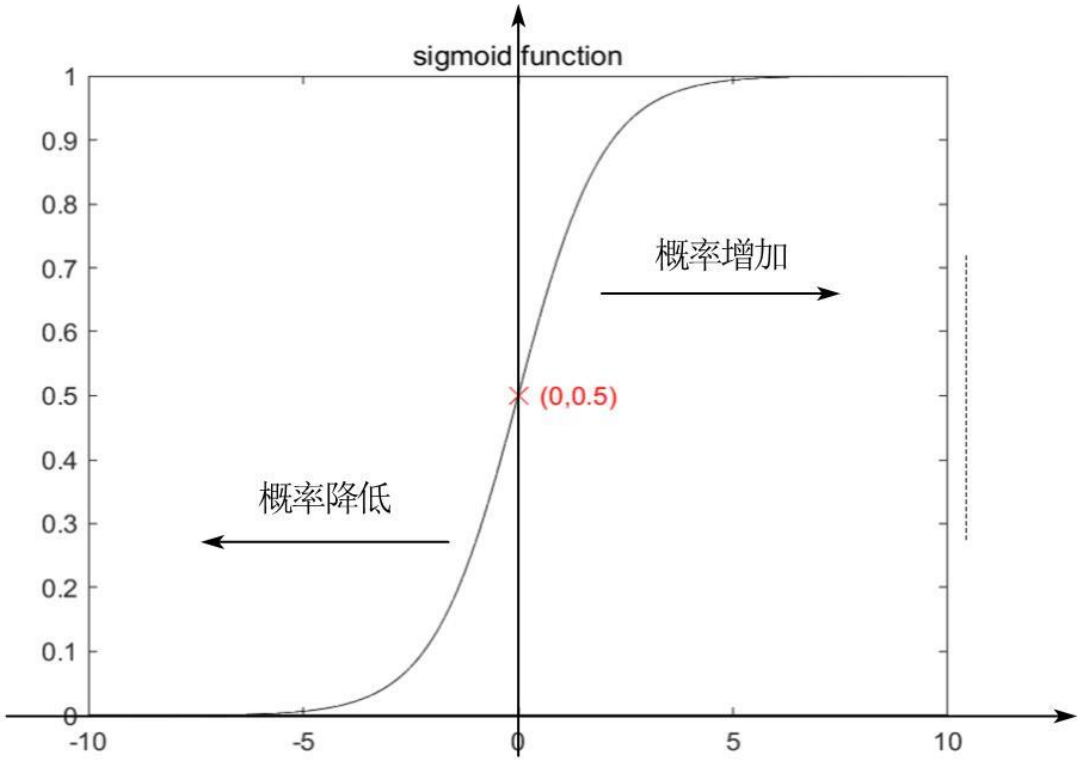


图 13. sigmoid 函数性质

若想让预测出的结果全部正确的概率最大,根据最大似然估计,就是所有样本预测正确的概率相乘得到的 P(总体正确)最大，似然函数为

$$L(\theta) = \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}}$$

由于连乘积形式难于处理，我们两端取对数得

$$\ln L(\theta) = \sum_{i=1}^{m} \left(y^{(i)} \ln\left(h_\theta(x^{(i)})\right) + (1-y^{(i)}) \ln\left(1-h_\theta(x^{(i)})\right)\right)$$

我们是要想使此值达到最大。但是在优化过程中比较习惯的是让某个值越小越好，所以我们

取其相反数并取一个平均值，得到

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}\ln\left(h_\theta(x^{(i)})\right)-(1-y^{(i)})\ln\left(1-h_\theta(x^{(i)})\right)\right)$$

这便是我们逻辑回归得到的代价函数。依照我们之前的方法，继续进行梯度下降。首先求得偏导数

$$\frac{\partial J(\theta)}{\partial\theta_j}=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j^{(i)}$$

然后在各个分量的方向进行迭代

$$\theta_j:=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta)=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j^{(i)}$$

正如我们之前的操作，通过迭代找到全局最优解或局部最优解。
基本思想确定之后，我们需要考虑另外一个比较重要的问题：拟合效果。

我们给出样例，如图 14 所示的三种拟合结果。

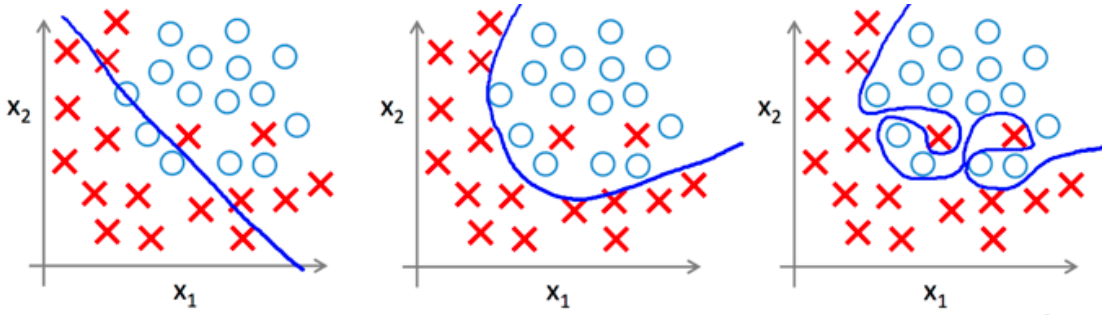


图 14. 三种拟合效果对比

第一个决策边界是$\theta^T x=\theta_0+\theta_1x_1+\theta_2x_2$；

第二个决策边界为$\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_1^2+\theta_4x_2^2+\theta_5x_1x_2$；

而第三个决策边界为$\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_1^2x_2+\theta_4x_1^2x_2^2+\theta_5x_1^2x_2^3+\ldots$

哪种拟合的效果最好呢？很明显是第三个，曲线几乎完美地经过了所有的样本点，但是它真的是最好的吗？
我们看到，第三种拟合的决策边界十分复杂，包括了大量的高阶多项式，这就需要我们用大量的样本数据来约束这个模型，但有时候我们并没有这么多的数据来进行模型的训练。此外，由于曲线的拟合程度太高，导致模型对于已知样本的拟合非常好，但是对于未知的点可能预测效果就比较差。我们把这种情况叫做**过拟合(overfitting)**。规避过拟合现象的方法有如下几种：

1） 数据集扩增(data augmentation)，由于过拟合是由于样本量不足引起的，所以数据集的扩增有助于防止过拟合现象发生；
2） 正则化(regularization)
我们这里面重点叙述一下正则化的含义所在。正则化的基本数学思想是，为了减少多项式中某些项的比重，那么就提高这些项的系数。

首先我们来讲一下特征映射(feature mapping)的概念。如果样本量很多无法用线性边界进行分类，我们就需要借助高阶函数。但是自变量我们只有x_1和x_2，所以我们可以使阶数升高，让分割线形成高阶函数的形状。例如，在二维空间上，我们要想使得最高阶达到6，就应该让x_1和x_2乘积的阶数不断升高直到阶数变为 6。我们将这些乘积表示成列向量如下所示

$$\text{mapFeature}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \\ x_1^3 \\ x_1^2x_2 \\ \vdots \\ x_1x_2^5 \\ x_2^6 \end{bmatrix}$$

这个向量的行数为$\dfrac{(1+7)\times 7}{2}=28$，所以我们将 2 维的特征映射成为 28 维向量。然后我们思考一下，假设我们想要的决策边界$\theta_0+\theta_1x_1+\theta_2x_2^2+\theta_3x_3^3+\theta_4x_4^4$具有二次型，那么应使得$\theta_3=0$且$\theta_4=0$，但是我们可以不完全消除这两项，设置一个代价函数

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}\ln\left(h_\theta\left(x^{(i)}\right)\right)-\left(1-y^{(i)}\right)\ln\left(1-h_\theta\left(x^{(i)}\right)\right)\right)+\lambda\sum_{i=3}^{4}\theta_i^2$$

我们在后面添加了一个**正则项**，通过代价函数不断减小使得θ_3、θ_4两项得到“惩罚”，通过这种方式减小这两个可能造成过拟合的项，达到规避过拟合的目的。
因此，我们得到了正则化逻辑回归的代价函数

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}\ln\left(h_\theta\left(x^{(i)}\right)\right)-\left(1-y^{(i)}\right)\ln\left(1-h_\theta\left(x^{(i)}\right)\right)\right)+\frac{\lambda}{2m}\sum_{i=1}^{n}\theta_i^2$$

这里我们从$i=1$开始正则化，因为我们认为分类最低的维度就是一次性，所以对θ_0进行正则化是没有必要的。

综上所述，如果发现过拟合的情况，就应该适当提高λ的大小；如果发现欠拟合，也就是分类准确率过低的时候可以减小λ的值。

我们继续分析正则化的代价函数，为了进行梯度下降，我们需要对代价函数进行求偏导：

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} \qquad j=0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m}\theta_j \qquad j\geq 1$$

这里面对θ_0的偏导数比较特殊，因为只有这一项我们没有进行正则化。按照这个方法进行梯度下降，我们可以得到局部最低点或全局最低点，而最终的边界函数就是$\theta^T.\mathrm{mapFeature}(\boldsymbol{x})$。

1.2.2 逻辑回归的实践 1：学生录取问题

我们假设学生能否得到录取取决于学生两场考试的成绩。我们现有训练集，每一行都是某位学生的考试成绩 1 和考试成绩 2 以及是否通过（0/1）。我们首先将散点图进行可视化，如图 15 所示。

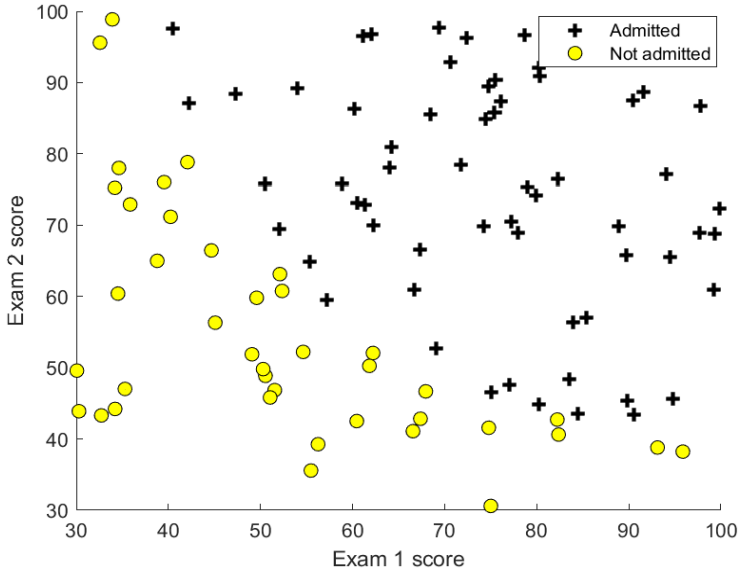


图 15. 学生成绩散点图

这是一个典型的二分类问题，从图中我们也可以看出两类点被一条直线一分为二。接下来我们按照之前的公式，编制计算代价函数的代码，如图 16 所示。

我们这里面要介绍MATLAB的一个库函数$fminunc$。这是MATLAB的一个优化函数，利用的就是梯度下降的方法，利用代价函数的值以及每次更新的梯度去逼近最优点。具体的调用格式如图17所示，迭代次数由option来控制，这里设置为400次。

为了调用该函数，我们需要计算梯度向量。对于每个分量θ_j，梯度就是代价函数在这个维度的偏导数，即

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

```
% m is the size of sample
Sum = 0;
for i=1:m
    sum = 0;
    %get the value of h_theta
    for j=1:size(theta,1)
        sum = sum + X(i,j) * theta(j,1);
    end
    sum = sigmoid(sum);
    Sum = Sum + ((-1)*y(i)*log(sum)-(1-y(i))*log(1-sum));
end
J = Sum/m;
```

图 16. 计算代价函数

```
%  Set options for fminunc
option = optimset('GradObj', 'on', 'MaxIter', 400);

%  Run fminunc to obtain the optimal theta
%  This function will return theta and the cost
[theta, cost] = ...
        fminunc(@(t)(costFunction(t, X, y)), initial_theta, option)
```

图 17. 调用优化函数$fminunc$

根据以上偏导数的公式，我们编制计算梯度的 MATLAB 代码如图 18 所示。

```
for i=1:size(theta,1)
    Sum = 0;
    for j=1:m
        sum = 0;
        %calculate the value of h_theta
        for k=1:size(theta,1)
            sum = sum + X(j,k) * theta(k,1);
        end
        sum = sigmoid(sum);
        Sum = Sum + (sum-y(j))*X(j,i);
    end
    grad(i,1) = Sum/m;
end
```

图 18. 计算每一个维度的梯度

在计算了代价函数以及梯度的数值之后，我们可以调用优化函数$fminunc$求解最终的$\boldsymbol{\theta}$以及代价。然后根据$\boldsymbol{\theta}$向量的值，我们可以绘制出边界函数。

如图 19 所示，边界函数基本将两边的点进行了分类。但是我们需要知道准确率的大小，此外还需要根据边界函数来进行预测，这才是我们的最终目的。

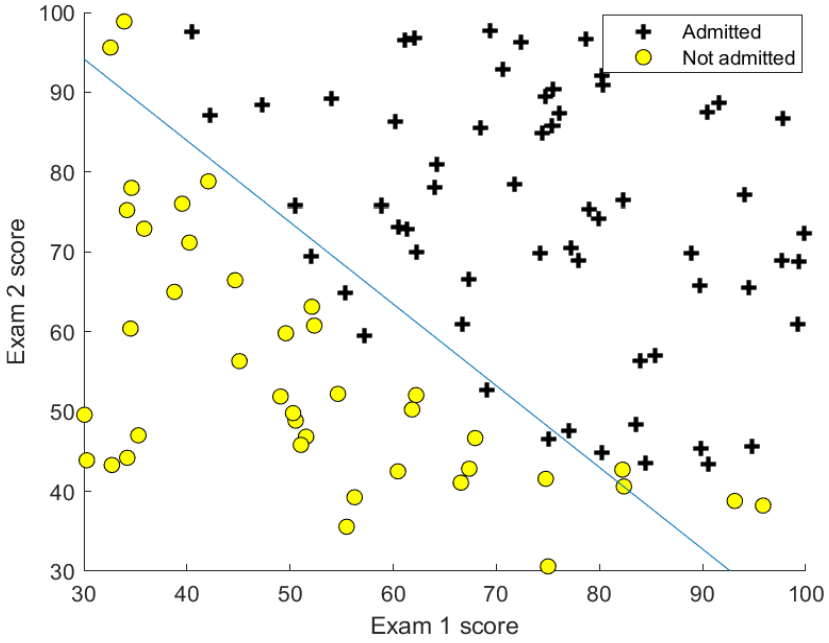


图 19. 分类结果进行可视化

我们已知了边界函数$\theta^T x$，而我们之前在讲 sigmoid 函数的时候提到过，当 sigmoid($\theta^T x$)的值大于 0.5 的时候认为是 1 类，反之属于 0 类。所以我们可以根据我们的边界函数对所有的样本点进行验证，然后和数据集中的y向量进行比较，这样就可以计算准确率了。

为此我们编制 MATLAB 代码来得到我们的验证结果，如图 20 所示。

```
m = size(X, 1); % Number of training examples

% You need to return the following variables correctly
p = zeros(m, 1);

% ====================== YOUR CODE HERE ======================
% Instructions: Complete the following code to make predictions
   using
%               your learned logistic regression parameters.
%               You should set p to a vector of 0's and 1's
%
for i=1:m
    tmp = sigmoid(X(i,:) * theta);
    if tmp>=0.5
        p(i) = 1;
    else
        p(i) = 0;
    end
end
```

图 20. 根据边界函数得到结果

然后和原数据集中的y向量进行比较，这样我们就得到了准确率(accuracy)。MATLAB运行结果显示准确率为89%。

接下来我们来进行预测。对于考试1成绩为45、考试2成绩为85的学生，我们将其代入边界函数，并代入sigmoid函数进行计算，也就是`sigmoid([1 45 85] * theta)`，计算结

果显示其录取概率为77.6291%.

1.2.3 逻辑回归的实践 2：芯片质量检测

我们如今有一组数据，显示了两个不同的芯片测试对于芯片是否能被接受的影响，可视化之后如图 21 所示。

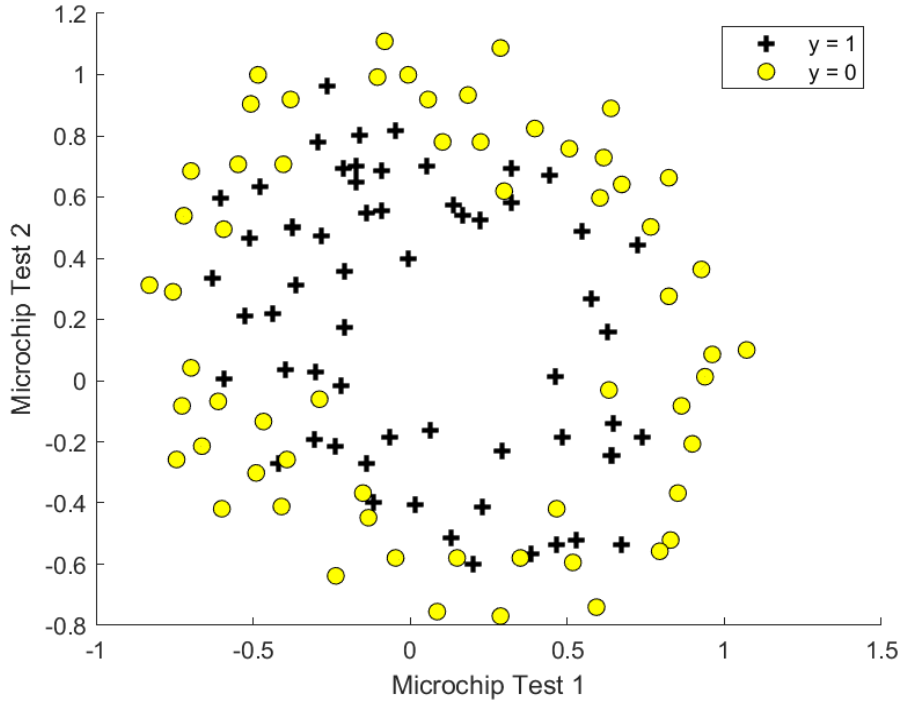


图 21. 芯片测试散点图

这个例子中我们可以看出不再是简单的线性分类，需要用到高阶的边界函数。所以我们必须要进行特征映射。我们取最高阶 6，生成特征映射矩阵，代码如图 22 所示。

```
degree = 6;
out = ones(size(X1(:,1)));
for i = 1:degree
    for j = 0:i
        out(:, end+1) = (X1.^(i-j)).*(X2.^j);
    end
end
```

图 22. 生成特征映射矩阵

值得一提的是，这里面的特征映射矩阵应该是一个样本数$m \times 28$ 的矩阵。

接下来我们应该计算的是代价函数以及梯度。这里面由于矩阵相对复杂，我们用下式表示以进行明确：

$$sigmoid\left(X_{m\times 28}\theta_{28\times 1}\right)=h_\theta\left(x^{(i)}\right)_{m\times 1}$$

我们计算代价函数的代码如图 23 所示。这部分反映的就是之前提到过的计算代价函数的公式。

```
Sum = 0;
for i=1:m
    sum = 0;
    %get the value of h_theta
    sum = X(i,:) * theta;
    sum = sigmoid(sum);
    Sum = Sum + ((-1)*y(i)*log(sum)-(1-y(i))*log(1-sum));
end
J = Sum/m;
Q = 0;
for i=1:size(theta,1)
  Q = Q + theta(i,1) * theta(i,1);
end
J = J + (lambda/(2*m))*Q;
```

图 23. 计算代价函数

为了调用库函数$fminunc$，我们还需要计算梯度。这里需要注意的是对θ_0求偏导的结果比较特殊，因为我们没有对θ_0进行正则化，代码计算也很简单，如图 24 所示。

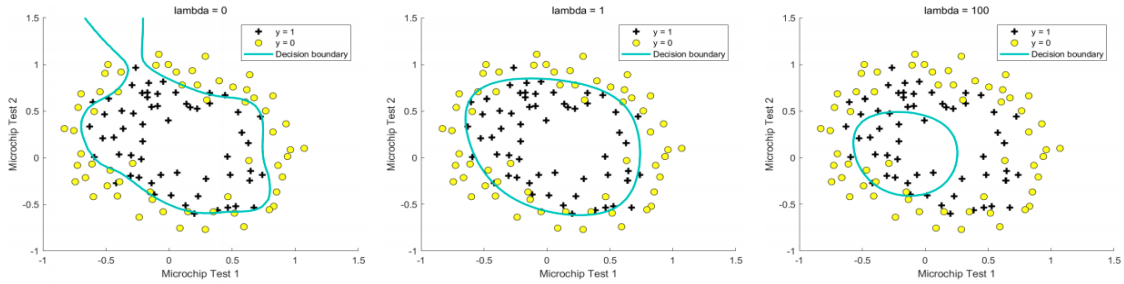
```
for i=1:size(theta,1)
    Sum = 0;
    for j=1:m
        sum = 0;
        %get the value of h_theta
        for k=1:size(theta,1)
            sum = sum + X(j,k) * theta(k,1);
        end
        sum = sigmoid(sum);
        Sum = Sum + (sum-y(j))*X(j,i);
    end
    % special case
    if i == 1
        grad(i,1) = Sum/m;
    else
        grad(i,1) = Sum/m+(lambda/m)*theta(i,1);
    end
end
```

图 24. 计算各个维度的梯度

我们首先测试一下没有正则化的情况。设置迭代次数为 225 次，结果如图 25(a)所示。从图中我们可以看出，这个结果可能已经发生了过拟合，此时的准确率为 87.288136%.

为了改善过拟合的情况，根据我们之前的观点，应该适当提高λ的值。我们设置$\lambda=1$，迭代次数不变的情况下，结果如图 25(b)所示。这次的决策边界比较合适，此事的准确率为 83.050847%.

如果我们提升λ的值过高，比如$\lambda=100$，这样的话就会造成拟合不足，结果图如图 25(c)所示。此时的准确率为 61.016949%.



(a) overfitting($\lambda = 0$) (b) good fitting($\lambda = 1$) (c) underfitting($\lambda = 100$)

图 25. 三种拟合效果对比

我们可以基于边界函数进行预测。我们就以$\lambda = 1$时的情况为例，根据 MATLAB 的运行结果，我们计算出$\boldsymbol{\theta}$向量的值，然后选择样本点$(x_1, x_2)^T = (0.3, 0.6)^T$计算特征映射矩阵，根据$\boldsymbol{\theta}$向量求出边界函数，并将其值代入 sigmoid 函数计算概率即可。

经过 MATLAB 计算，其属于正例，即在$y = 1$这一类的概率为 70.6828%，大于 0.5，我们认为该样本属于 1 类。

此外，为了验证正则化的作用，我们查看一下在正则化和未正则化的情况下$\boldsymbol{\theta}$向量的值分别是多少。经过 MATLAB 计算，以高阶项系数θ_{27}为例，未正则化的时候$\theta_{27} = -12.6603517605311$，而正则化之后$\theta_{27} = -0.98372836$，高阶项系数减小了 10 倍之多，可以说正则化减小了高阶项的比重，降低了过拟合的风险。

四、实验总结与体会

本次实验实现了用梯度下降法实现一元线性回归分析与多元线性回归分析，以及用正规方程实现多元线性回归分析。此外，还通过逻辑分析的方法实现了二分类问题。

本次实验是机器学习的第一次实验，也是真正接触机器学习的第一个过程。在这次实验中，我不仅仅加深了对回归以及分类的数学思想的理解，还从本质上理解了机器学习的一个过程。实际上，机器学习的过程就是确定一个任务(task)，然后指定一个指标(performance measure)来训练数据集(experience)。这就是机器学习的基本过程，以后不管是多么复杂的机器学习的算法，基本过程都是如上所述的过程。

指导教师批阅意见： **成绩评定：** 指导教师签字：　贾森　徐萌 2020 年 10　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
　　2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。