基于逻辑回归的 0/1 二分类问题求解模型

姓名：王杰永

学号：03190886

班级：计算机科学与技术 2019-03 班

中国矿业大学

二〇二一年五月

摘　要

机器学习问题中，把不同种类的机器学习方法分为两大类——监督学习和无监督学习。两种最常见的监督学习任务是回归与分类。在回归问题中，程序必须从一个或多个特征预测一个或多个连续值；而在分类问题中，程序则是需要学习去从一个或者多个特征去预测一个或多个响应特征的离散值。

本文主要介绍了一个简单的 0/1 二分类问题以及其求解程序。教师根据学生的两部分成绩决定学生是否取得该课程学分。利用逻辑回归（Logistic Regression）/对数几率回归为二分类问题建立数学模型；使用梯度下降算法求解模型中的参数。同时，利用向量化的思想简化了程序的编写难度。

关键词：分类；逻辑回归；梯度下降；向量

ABSTRACT

In the machine learning problem, we divide the different kinds of machine learning methods into two categories: supervised and unsupervised learning. The two most common supervised learning tasks are regression and classification. In a regression problem, the program must predict one or more continuous values from one or more features; And in the classification problem, the program needs to learn to predict the discrete values of one or more response features from one or more features.

This paper mainly introduces the principle and solution program of the 0/1 dichotomy problem. Teachers decide whether students will receive credit for the course based on their grades in two exams. Logistic Regression was used to establish a mathematical model for dichotomy problem. Gradient descent algorithm is used to solve the parameters in the model. At the same time, vectorization is used to simplify the difficulty of program writing.

Keywords： Classification; Logistic Regression; Gradient descent; Vectorization

目　　录

1 问题概述

以下是本次课题作业所选取的问题：

假设你是一名大学教师，想要探究两次模拟考试的分数与最终期末测试通过之间的关系，即已知某学生两次模拟考试的分数，预测该学生在期末测试中能否通过并成功取得该课程学分。

现已有历年 100 位学生的两次模拟考试的成绩以及所有学生是否取得该课程学分的情况，我们想要分析这 100 组数据，建立数学模型，最终根据模型预测一些新的学生能否通过期末测试并取得学分。

历年 100 位学生的相关数据见附录 1。

简要分析一下该问题：

我们要做的是找到两次成绩与能否通过考试之间的数学关系，并用新的数据去检测该关系的正确性与准确性。由于最终要预测的是学生能否通过期末测试，而非期末测试的具体分数，故该问题是要把学生分为两类——通过期末考试的一类学生与不能通过期末考试的一类学生。

该问题属于**分类**问题。后文将使用逻辑回归模型（尽管模型叫做“逻辑回归”，但该模型是用来解决分类问题的）来预测学生是否通过考试取得学分。

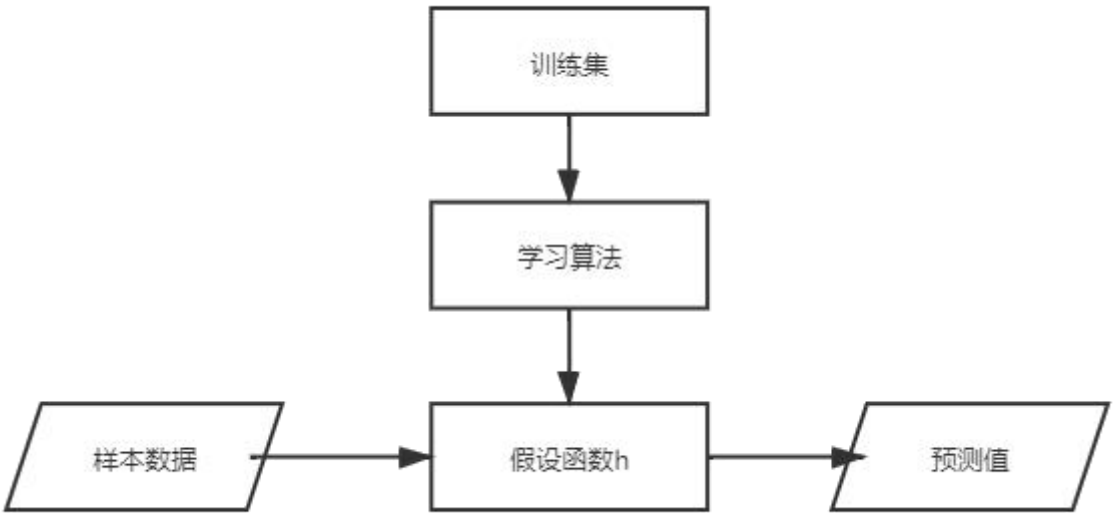
2 基本术语介绍

我们可以发现本题是预测学生是否取得学分的，我们将使用大量数据，数据包含历年学生两次考试分数以及最终是否取得学分。这些历年的大量已知数据我们称为**数据集**，或者叫做**训练集**。

训练集中的每一条记录是关于一个对象（这里指一名学生）的描述，称为一个**样本**或是**观察实例**。训练集包含的样本数量用 m 表示。

每个样本都有若干数量的**特征**，一般用 x 来表示。同时，每个样本还有一个独特的**标签**，一般用 y 来表示。例如，在本题的训练集中，每个样本的第一次模拟考试的成绩和第二次模拟考试的成绩是样本的两个特征，使用x_1,x_2分别表示。样本是否通过考试则作为标签，用 y 表示。$y=1$ 代表通过考试，$y=0$ 代表没有通过考试。由于训练集一般包含大量的样本信息，我们使用 $\left(x^{(i)},y^{(i)}\right)$ 表示第 i 个样本。

我们的逻辑回归模型的工作方式如下：



这就是一个监督学习算法（逻辑回归属于监督学习）的工作方式。我们将训练集作为输入“喂”给算法，通过算法得到一个**假设函数** h，在这之后，就可以将要预测的数据的特征作为假设函数 h 的输入，其输出就是我们想要的预测结果。

通常，假设函数中存在若干可变的参数。所以很容易的想到，假设函数是不唯一的。为了解决选取什么参数的假设函数的预测准确性高的问题，我们需要一种评价假设函数好坏的方式。在机器学习问题中，使用**代价函数（损失函数）**作为标准，判断一个假设函数的优劣。代价函数值越小，则该参数对应的假设函数越好。

所以，预测准确性最高的假设函数的参数应该是使代价函数取得最小值的参数值。所以逻辑回归模型的学习过程就是找到使代价函数取得最小值时的参数的过程。为了求出代价函数的最小值对应的参数，我们使用**梯度下降**算法。

3 回归问题的基础理论

3.1 假设函数

谈到预测问题，最容易想到的就是线性回归。单变量线性回归的假设函数形如 $y = kx + b$。如果将该函数作为逻辑回归的假设函数就会出现很多问题。

首先，我们想要的逻辑回归模型的输出应该是 0 1 这样的离散量，而 $y = kx + b$ 的输出是实数域内的连续值。尽管我们可以规定，当 $y > 0.5$ 时 $h = 1$，当 $y \leq 0.5$ 时 $h = 0$。这样的做法看起来是可行的，但如果训练样本的标签 y 远大于 1 或远小于 0 的话，就会感觉很奇怪。

逻辑回归模型的输出变量范围始终在 0 和 1 之间。记线性回归的假设函数为：

$$p(x) = \theta_0 + \theta_1 x$$

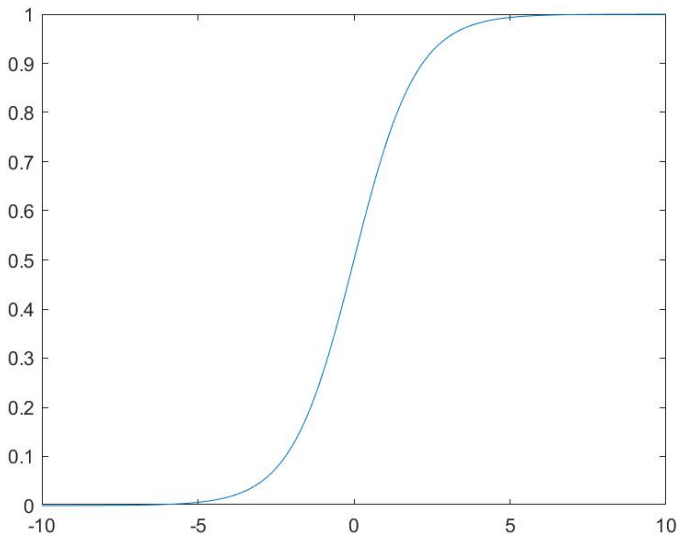
那么逻辑回归的假设函数为：

$$h_\theta(x) = g(p(x))$$

其中，外层函数 g 代表的函数是一个 S 形函数（Sigmoid function），表达式如下：

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

该函数的图像为：



可以看出，对于给定的 x，函数 g 的输出值均介于 0~1 之间。最终假设函数$h_\theta(x)$的作用是：对于给定的输入变量，根据选择的参数计算出变量=1 的概率。

可以看出，当我们选择一组合适的参数θ后，逻辑回归模型也就建立起来了。

3.2 代价函数

如何来选择合适的参数θ呢？需要引入代价函数。

还是从最熟悉的线性回归问题入手考虑。在线性回归问题中，对于一个假设函数，通常使用“距离”来表示模型的好坏。其代价函数如下：

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

所有预测值与标签值之差的距离平方和的平均值作为代价函数，自然，代价函数值越小，误差就会越小，模型的准确度就越高。

理论上来说，我们可以对逻辑回归模型沿用与线性回归相同的定义。但是问题在于，当我们将逻辑回归的假设函数带入到这样定义的代价函数中，将得到一个非凸的代价函数。这就意味着我们的代价函数有着许多局部最小值，这将影响梯度下降算法（在第二部分中提到的，寻找代价函数最小值对应参数的算法）寻找全局最小值。

我们重新定义逻辑回归的代价函数为：

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

其中

$$Cost\left(h_\theta\left(x^{(i)}\right),y^{(i)}\right)=\begin{cases}-\ln\left(h_\theta(x)\right), & y=1\\ -\ln\left(1-h_\theta(x)\right), & y=0\end{cases}$$

这样构建的 $Cost\left(h_\theta\left(x^{(i)}\right),y^{(i)}\right)$函数的特点是：当实际的 $y=1$ 且$h_\theta(x)$也为 1 时误差为 0，而$h_\theta(x)$不为 1 时，误差随着$h_\theta(x)$的变小而变大；当实际的 $y=0$ 且$h_\theta(x)$也为 0 时误差为 0，而$h_\theta(x)$不为 0 时，误差随着$h_\theta(x)$的变大而变大。符合我们想要的代价函数的特点。

同时，构建的 $Cost\left(h_\theta\left(x^{(i)}\right),y^{(i)}\right)$可以简化如下：

$$Cost\left(h_\theta\left(x^{(i)}\right),y^{(i)}\right)=-y\times\ln\left(h_\theta(x)\right)-(1-y)\times\ln\left(1-h_\theta(x)\right)$$

带入代价函数得到最终代价函数的表达式

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

接下来就可以通过梯度下降算法，求出使代价函数取得最小值时的参数θ了

3.3 梯度下降算法

梯度下降是一个用来求函数最小值的算法，我们将使用梯度下降算法来求代价函数$J(\theta_0,\theta_1)$的最小值。

梯度下降的思想是：算法开始时，我们随机选择一个参数的组合$(\theta_0,\theta_1,\cdots,\theta_n)$，计算当前参数组合下的代价函数值，之后，我们寻找下一个**让代价函数值下降的最多**的参数组合。重复上述步骤直到找到一个局部最小值。但是因为我们并没有尝试完所有的参数组合，所以不能确定我们得到的局部最小值是不是全局最小值。选择不同的初始参数组合$(\theta_0,\theta_1,\cdots,\theta_n)$，可能会找到不同的局部最小值。

可以想到，梯度下降的最关键部分是如何找到让代价函数值下降的最多的参数组合，函数值下降最多即（偏）导数值最大，于是可以得到梯度下降算法的伪代码：

$while(not\ \ convergence)$

{

$\theta_j=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta)$

}

其中，α是学习率，它决定了沿着代价函数下降程度最大的方向改变的量的大小。学习率的值需要设置的合适。太大会导致代价函数在局部最小值（全局最小值）点的两侧不断跳跃，无法收敛；太小则会导致代价函数收敛到局部最小值（全局最小值）的速度过慢，经过有限次训练很难收敛到极值点。因此学习率的大小要设置合适。

在梯度下降中，每一次下降需要**同时**让所有的参数减去学习率乘以代价函数对应的偏导数。也就是，在计算出下次循环的参数值之后，再去更新当前参数值。例如：

$$
\begin{aligned}
&while(not\ \ convergence)\\
&\{\\
&\quad temp_0 = \theta_0 - \alpha\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1);\\
&\quad temp_1 = \theta_1 - \alpha\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1);\\
&\quad \theta_0 = temp_0;\\
&\quad \theta_1 = temp_1;\\
&\}
\end{aligned}
$$

事实上，对于梯度下降来说，从技术上讲，并不需要写出代码来计算代价函数$J(\theta)$，只需要写代码来计算这些导数项并更新相应的参数。但如果希望代码还可以监控$J(\theta)$的敛散性，或者说，如果希望可视化梯度下降的过程，我们是需要计算代价函数$J(\theta)$的。

在只有一个特征的简单单变量逻辑回归问题中，假设函数的形式如下：

$$h_\theta(x) = g(p(x)),\ p(x) = \theta_0 + \theta_1 x$$

可以看出，假设函数中θ_0和θ_1形式上是不对称的，因此代价函数的偏导数的形式不统一，如果特征超过两个，会导致代码很长，可读性差。但如果将假设函数写成下面的形式：

$$h_\theta(x) = g(p(x)),\ p(x) = \theta_0 x_0 + \theta_1 x_1 \ \ 且 \ \ x_0 = 1$$

则在后续的操作中，不同的偏导数形式也会完全统一，可以很方便的运用各种编程语言的线性代数相关库进行处理，代码可读性加强。

3.4 运算的向量化

当我们对特征列加入特征值恒为 1 的一列后，假设函数等价为：

$$h_\theta(x) = g(p(x)),\ p(x) = \theta_0 x_0 + \theta_1 x_1 \ \ 且 \ \ x_0 = 1$$

我们记$\theta = \begin{bmatrix}\theta_0\\ \theta_1\end{bmatrix}$，$x = \begin{bmatrix}x_0\\ x_1\end{bmatrix}$，$x_0 = 1$，则可以改写为：

$$p(x) = \theta^T X$$

最终，逻辑回归的假设函数可以简化为：

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

由于将特征使用特征矩阵表示，引入了恒为 1 的特征列，假设函数形式上具有了一定的对称性，最终梯度下降算法中的偏导数项形式上也更加简洁（此处省略求导过程）。

$while(not \ \ convergence)$

{

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

}

3.5 决策边界

决策边界告诉我们逻辑回归模型通过梯度下降算法最终拟合出的参数的几何意义是什么对于逻辑回归的代价函数

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

我们知道，函数的值域在区间[0,1]中，其意义是对于给定的输入变量，根据选择的参数计算出样本标签=1 的概率。我们预测

当$h_\theta(x) \geq 0.5$ 时，预测 $y = 1$。

当$h_\theta(x) < 0.5$ 时，预测 $y = 0$。

我们已经通过 Matlab 绘制出了 Sigmoid 函数的图像，根据图像可以很容易的看出以下结论：

当$h_\theta(x) \geq 0.5$ 时，$\theta^T x \geq 0$，即$\theta_0 x_0 + \theta,x_1 + \ldots + \theta_n x_n \geq 0$，预测 $y = 1$。

当$h_\theta(x) < 0.5$ 时，$\theta^T x < 0$，即$\theta_0 x_0 + \theta,x_1 + \ldots + \theta_n x_n < 0$，预测 $y = 0$。

我们知道$\theta_0 x_0 + \theta, x_1 + ... + \theta_n x_n = 0$ 这个方程可以将 n 个特征值构成的 n 维空间划分成两部分，使得方程大于等于 0 的部分中分布着所有的负样本，使得方程小于 0 的部分中分布着所有的负样本。

以 n=2 为例，上述方程是一条二维平面中的直线，该直线将二维平面划分成两部分。所有的正负样本被分布在了直线两侧。

于是我们可以得出结论，逻辑回归得到的参数 theta 的几何意义是一条分界线，将预测为 1 的区域和预测为 0 的区域分隔开。

4 代码分析

4.1 主函数文件 main.m

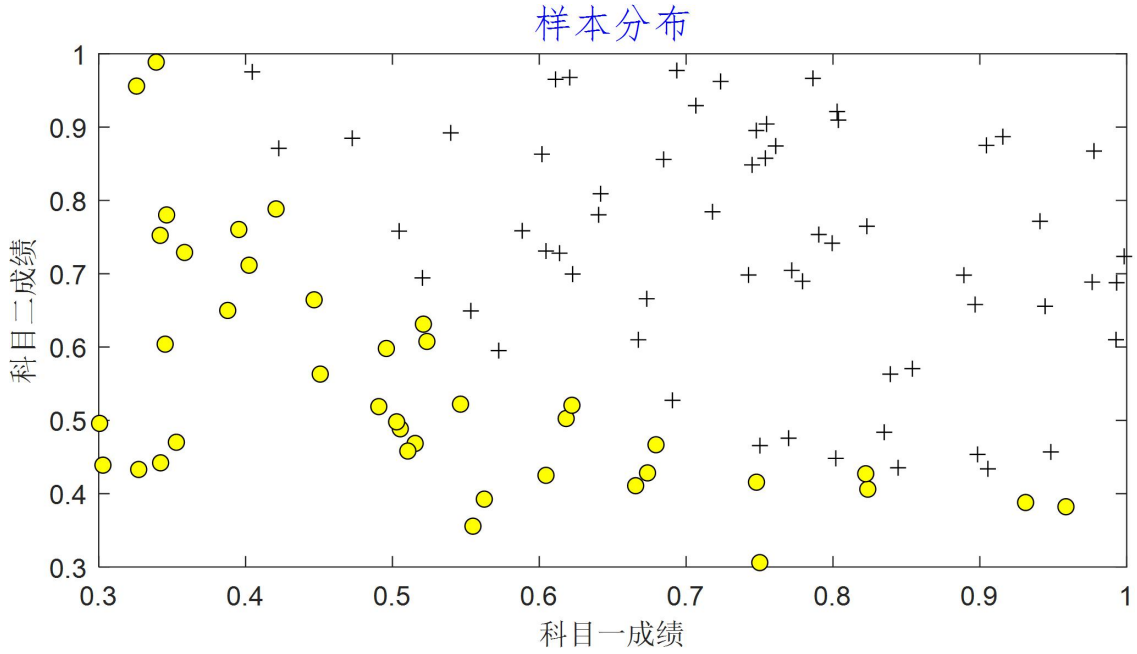
4.1.1 Part 1 加载数据，将训练集导入

首先调用 load 函数将训练集文件导入，此时变量 data 是一个 100 行 3 列的矩阵。

对 data 矩阵切片，其前两列数据是训练集样本的两个特征，赋值给变量 X；其最后一列是每个样本的标签，赋值给变量 y，0/1 的两种值代表该样本所属不同地类别。

同时，对特征矩阵 X 中的所有数据缩放到原值的百分之一，这样所有的特征值均是处于[0,1]区间内的值，这样在运用梯度下降算法中比较容易收敛（通过实践得出的结论，没有分析过原因）。

使用 plot 函数查看我们得到的训练集的样本分布情况，得到的图像如下：



训练集样本分布图

其中，横轴是科目一成绩（第一个特征值）的百分之一，纵轴是科目二成绩（第二个特征值）的百分之一。对应样本点如果是正样本（通过考试），则标记为黑色叉号；对应样本点如果是负样本（未通过考试），则标记为黄色圆圈。从而得到了上图所示的训练集样本分布图。

从分布图中可以很容易看出，训练集样本确实大致分为两类。通过逻辑回归确定的假设函数的参数，其实质上是确定了一条决策边界。

4.1.2 Part 2 拟合模型前的准备工作

通过内置函数 size()，得到训练集样本的特征矩阵 X 的行数 m 与列数 n，其中 m 代表训练集样本数，n 代表每一个样本的特征数量。

在特征矩阵 X 的最前面插入一列常数 1，方便后续的求导计算。

最终我们需要求出假设函数 $h_\theta(x)=\frac{1}{1+e^{-\theta^T x}}$ 的参数即θ列向量，所以需要一个临时变量记录每一次梯度下降的过程中θ的值更新为了多少。使用 n+1 列的零向量（维度对应于$x_0,x_1,x_2,...,x_n$总共 n+1 个参数，其中$x_0=1$）来初始化变量 theta。

4.1.3 Part 3 梯度下降

参数矩阵 theta 初始化成了全零的向量，在开始梯度下降前，首先计算一下当前的代价函数值。

$$cost1 = costFunction(theta,X,y)$$

$$fprintf('训练前的代价函数值：%f\n', cost1)$$

将计算的结果存储在临时变量 $cost1$ 中，$costFunction()$是计算代价函数值的自定义函数，其原型保存在文件 $costFunction.m$ 中。

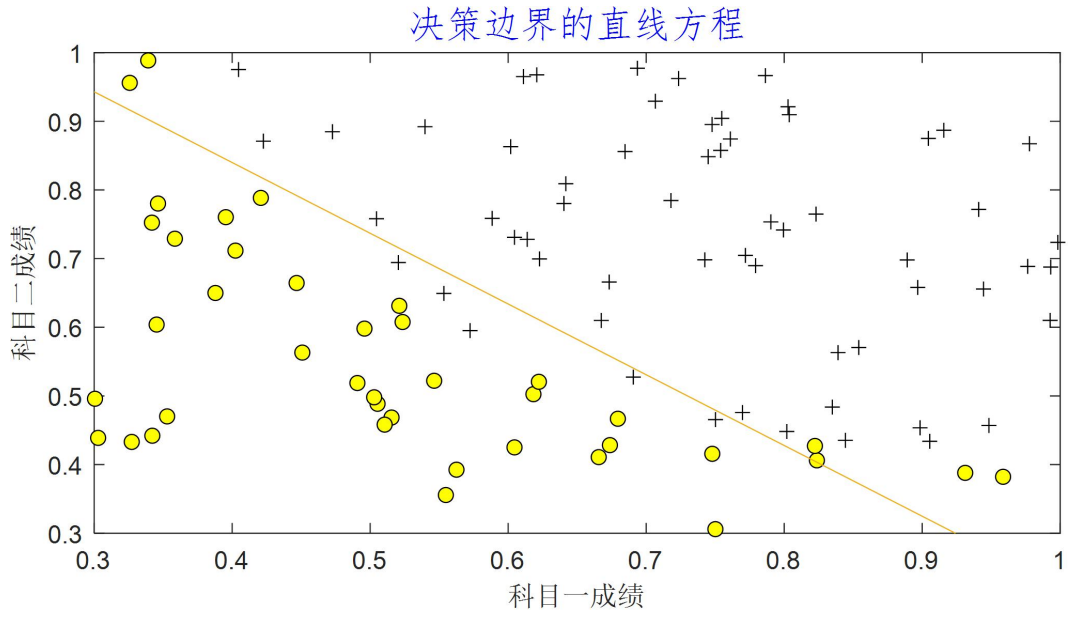
提前设置梯度下降的学习率 $learn\_rate = 0.05$，迭代次数 $iter = 150000$。

在每一次梯度下降过程中，理论上以当前参数 $theta$ 作为参数的代价函数的值会逐渐变小，最终收敛于某一常数。为了可以将代价函数收敛的过程体现出来，使用一个$(iter,1)$维度的列向量 $vector\_cost$ 记录每一次迭代后代价函数的值，其中，$iter$ 是设置的迭代次数。

（具体梯度下降的构成见附件代码）

4.1.4 Part 4 打印输出

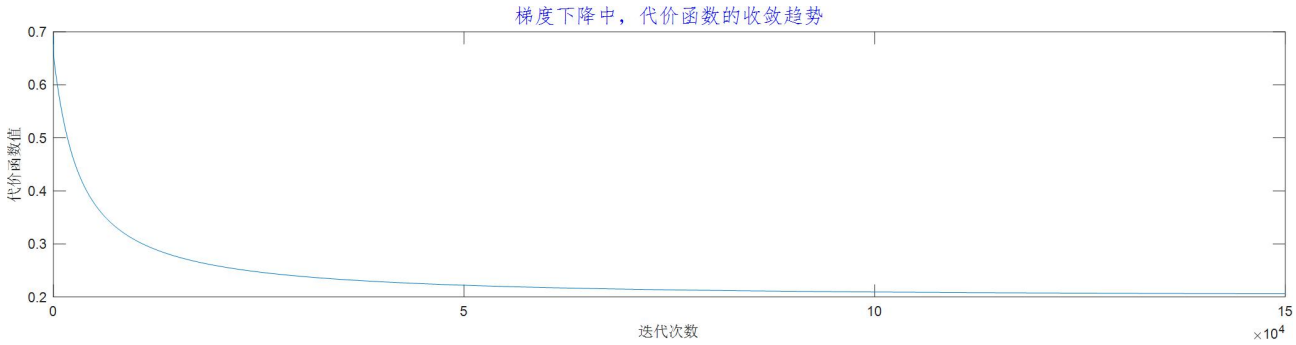
首先将最终得到的决策边界方程作为隐函数，使用 fimplicit 函数绘制其图像。其函数图像如下图所示：



拟合出的决策边界直线方程图

由图像可以看出，最终拟合出的决策边界直线很好的将训练集样本的分布空间划分为两部分，区分出了正负样本。拟合成功。

在梯度下降的执行过程中，我们使用变量 $vector\_cost$ 记录了每一次迭代时的代价函数值，将该向量使用 plot 函数绘制，可以很明显的感受到代价函数收敛的过程。如下图：



梯度下降算法中代价函数的收敛过程

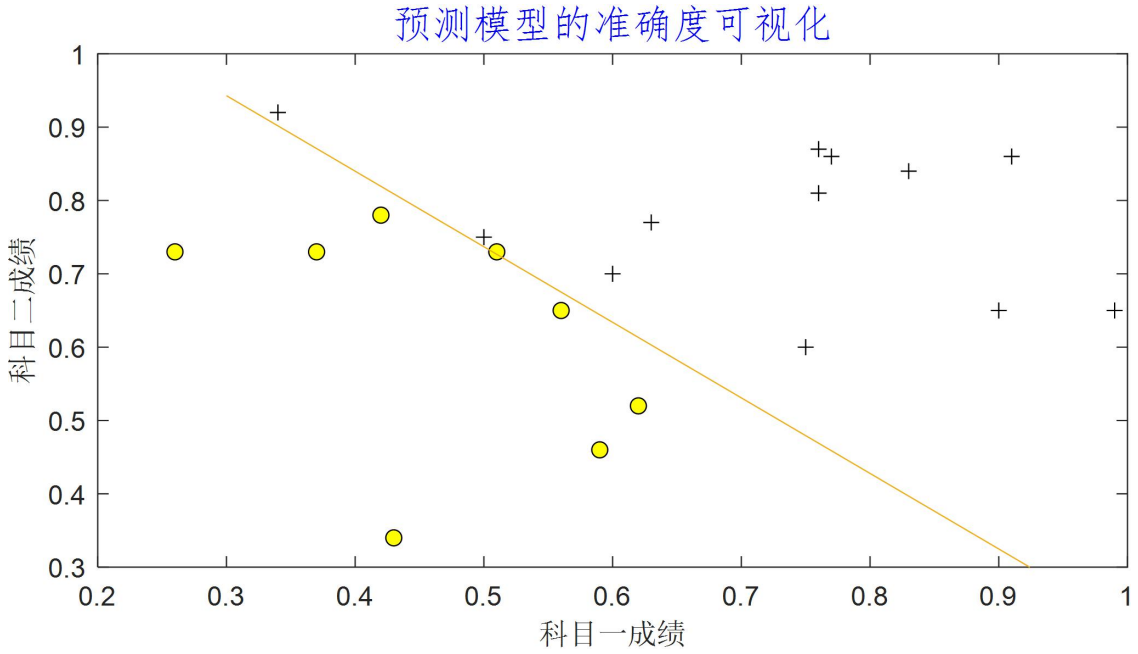
从图像看出，在大约前 10000 次训练中，代价函数收敛的速度很快，在后面的十四万此训练中，代价函数收敛速度明显放缓。在最后，代价函数值几乎不变，说明我们的梯度下降算法很好的找到了代价函数的局部最小值（也是全局最小值）。

4.1.5 Part 5 验证模型的准确性

首先调用 load 函数将测试集（testdata.txt，见附件 2）中的数据导入。利用矩阵的切片方法，分别将两列特征值向量与标签值向量存入三个不同变量中，同时对特征值向量进行 100 倍的缩放。

利用已经建立好的模型，建立模型的预测函数，分别对测试集中的 20 个样本数据进行预测。

将预测结果与真实结果比较，打印比较结果以及预测模型的准确率。最终结果的可视化图表如下：



测试集样本分布以及模型准确性

从图中可以看出，预测模型在测试集上表现良好，只有一个样本预测错误，其余样本均正确，正确率 95%。

4.2 自定义函数文件 sigmoid.m

函数实现了标准的 S 型 sigmoid 函数的功能。

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

4.3 自定义函数文件 costFunction.m

函数实现了逻辑回归模型中代价函数的计算。

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

5 代码运行结果展示

主函数的第三部分梯度下降中，执行梯度下降前，计算了以全零初始化的参数 theta 矩阵的代价函数值 0.693147。运行结果如图：

```
Part 3 梯度下降
训练前的代价函数值：0.693147
```

主函数的第四部分打印输出中，代价函数最终收敛于值 0.205816，三个参数值分别是 -21.4718，17.6678，17.1499。

最终拟合得到的决策边界方程：$-21.4718 + 17.6678 \times x_1 + 17.1499 \times x_2 = 0$

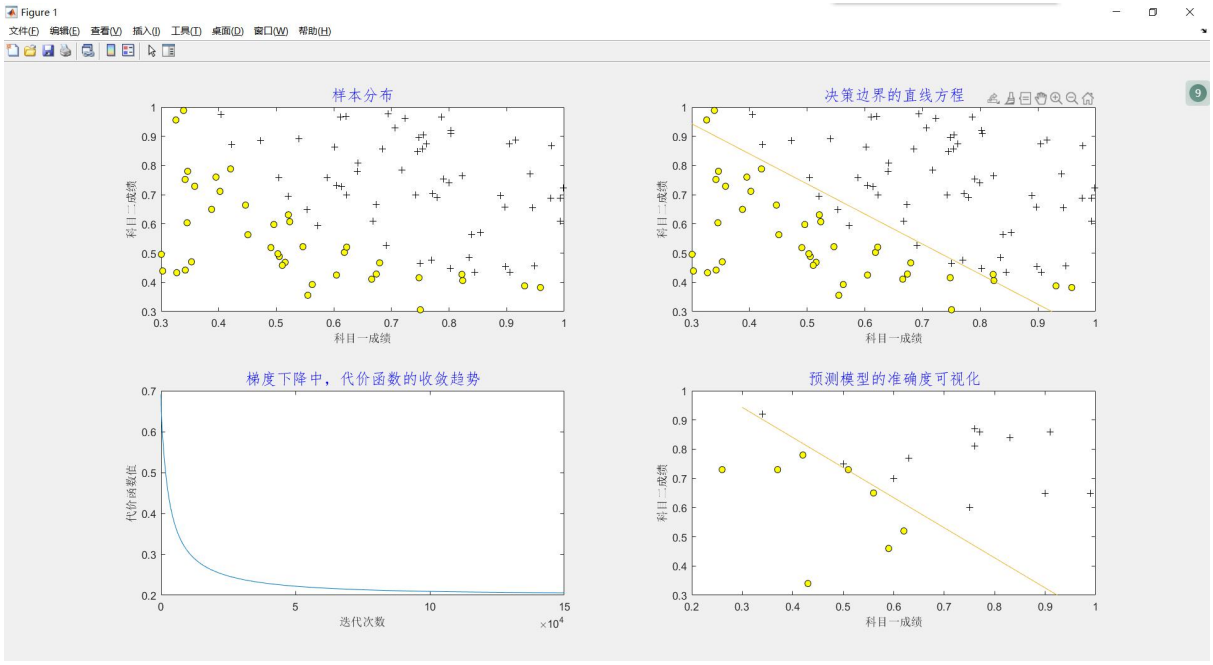
运行结果如图：

```
Part 4 打印输出

最终代价函数收敛于值：0.205816
得到的参数theta：
  -21.4718   17.6678   17.1499

拟合得到的决策边界方程：-21.4718+17.6678*x1+17.1499*x2
```

主函数的第五部分验证模型准确性中，输出了 20 个测试样本的预测结果，并计算了预测模型在测试集上预测成功的准确率为 0.95。

```
Part 5 验证模型准确性
第1个样本的预测结果是：预测成功
第2个样本的预测结果是：预测成功
第3个样本的预测结果是：预测成功
第4个样本的预测结果是：预测成功
第5个样本的预测结果是：预测失败
第6个样本的预测结果是：预测成功
第7个样本的预测结果是：预测成功
第8个样本的预测结果是：预测成功
第9个样本的预测结果是：预测成功
第10个样本的预测结果是：预测成功
第11个样本的预测结果是：预测成功
第12个样本的预测结果是：预测成功
第13个样本的预测结果是：预测成功
第14个样本的预测结果是：预测成功
第15个样本的预测结果是：预测成功
第16个样本的预测结果是：预测成功
第17个样本的预测结果是：预测成功
第18个样本的预测结果是：预测成功
第19个样本的预测结果是：预测成功
第20个样本的预测结果是：预测成功
该预测模型在测试集上预测成功的准确率为0.950000
```

最终，代码中绘图部分的绘图结果如下图所示：



6 总结

从最终的测试集预测结果来看，通过梯度下降算法建立的逻辑回归预测模型的预测准确率还是很高的，该监督学习模型是成功的。相比于 C/C++，使用 MATLAB 强大的计算能力让代码复杂程度大大简化。

通过本门课程的学习，我又掌握了 MATLAB 这样一门简洁高效实用的计算机语言。今后在学习中可能遇到的科学计算相关的问题又多了一种解决问题的方式。

收获颇丰。

7 附录

7.1 附录 1

附录 1：历年 100 位学生的两次模拟考试成绩以及最终是否通过期末考试数据 文件：trainingdata.txt

（逗号作为间隔，100 行 3 列的数据，每一行代表一个学生的相关数据，第一列第二列分别表示两次模拟考试成绩，第三列表示该学生是否通过了期末考试，0 代表不通过，1 代表通过。）

34.62365962451697,78.0246928153624,0

30.28671076822607,43.89499752400101,0

35.84740876993872,72.90219802708364,0

60.18259938620976,86.30855209546826,1

79.0327360507101,75.3443764369103,1

45.08327747668339,56.3163717815305,0

61.10666453684766,96.51142588489624,1

75.02474556738889,46.55401354116538,1

76.09878670226257,87.42056971926803,1

84.43281996120035,43.53339331072109,1

95.86155507093572,38.22527805795094,0

75.01365838958247,30.60326323428011,0

82.30705337399482,76.48196330235604,1

69.36458875970939,97.71869196188608,1

39.53833914367223,76.03681085115882,0

53.9710521485623,89.20735013750205,1

69.07014406283025,52.74046973016765,1

67.94685547711617,46.67857410673128,0

70.66150955499435,92.92713789364831,1

76.97878372747498,47.57596364975532,1

67.37202754570876,42.83843832029179,0

89.67677575072079,65.79936592745237,1

50.534788289883,48.85581152764205,0

34.21206097786789,44.20952859866288,0

77.9240914545704,68.9723599933059,1

62.27101367004632,69.95445795447587,1

80.1901807509566,44.82162893218353,1

93.114388797442,38.80067033713209,0

61.83020602312595,50.25610789244621,0

38.78580379679423,64.99568095539578,0

61.379289447425,72.80788731317097,1

85.40451939411645,57.05198397627122,1

52.10797973193984,63.12762376881715,0

52.04540476831827,69.43286012045222,1

40.23689373545111,71.16774802184875,0

54.63510555424817,52.21388588061123,0

33.91550010906887,98.86943574220611,0

64.17698887494485,80.90806058670817,1

74.78925295941542,41.57341522824434,0

34.1836400264419,75.2377203360134,0

83.90239366249155,56.30804621605327,1

51.54772026906181,46.85629026349976,0

94.44336776917852,65.56892160559052,1

82.36875375713919,40.61825515970618,0

51.04775177128865,45.82270145776001,0

62.22267576120188,52.06099194836679,0

77.19303492601364,70.45820000180959,1

97.77159928000232,86.7278223300282,1

62.07306379667647,96.76882412413983,1

91.56497449807442,88.69629254546599,1

79.94481794066932,74.16311935043758,1

99.2725269292572,60.99903099844988,1

90.54671411399852,43.39060180650027,1

34.52451385320009,60.39634245837173,0

50.2864961189907,49.80453881323059,0

49.58667721632031,59.80895099453265,0

97.64563396007767,68.86157272420604,1

32.57720016809309,95.59854761387875,0

74.24869136721598,69.82457122657193,1

71.79646205863379,78.45356224515052,1

75.3956114656803,85.75993667331619,1

35.28611281526193,47.02051394723416,0

56.25381749711624,39.26147251058019,0

30.05882244669796,49.59297386723685,0

44.66826172480893,66.45008614558913,0

66.56089447242954,41.09209807936973,0

40.45755098375164,97.53518548909936,1

49.07256321908844,51.88321182073966,0

80.27957401466998,92.11606081344084,1

66.74671856944039,60.99139402740988,1

32.72283304060323,43.30717306430063,0

64.0393204150601,78.03168802018232,1

72.34649422579923,96.22759296761404,1

60.45788573918959,73.09499809758037,1

58.84095621726802,75.85844831279042,1

99.82785779692128,72.36925193383885,1

47.26426910848174,88.47586499559782,1

50.45815980285988,75.80985952982456,1

60.45555629271532,42.50840943572217,0

82.22666157785568,42.71987853716458,0

88.9138964166533,69.80378889835472,1

94.83450672430196,45.69430680250754,1

67.31925746917527,66.58935317747915,1

57.23870631569862,59.51428198012956,1

80.36675600171273,90.96014789746954,1

68.46852178591112,85.59430710452014,1

42.0754545384731,78.84478600148043,0

75.47770200533905,90.42453899753964,1

78.63542434898018,96.64742716885644,1

52.34800398794107,60.76950525602592,0

94.09433112516793,77.15910509073893,1

90.44855097096364,87.50879176484702,1

55.48216114069585,35.57070347228866,0

74.49269241843041,84.84513684930135,1

89.84580670720979,45.35828361091658,1

83.48916274498238,48.38028579728175,1

42.2617008099817,87.10385094025457,1

99.31500880510394,68.77540947206617,1

55.34001756003703,64.9319380069486,1

74.77589300092767,89.52981289513276,1

7.2 附录 2

附录 2：测试集中的 20 个样本数据 文件：testdata.txt

（逗号作为间隔，20 行 3 列的数据，每一行代表一个学生的相关数据，第一列第二列分别表示两次模拟考试成绩，第三列表示该学生是否通过了期末考试，0 代表不通过，1 代表通过。）

60,70,1

34,92,1

26,73,0

50,75,1

51,73,0

91,86,1

59,46,0

43,34,0

62,52,0

77,86,1

63,77,1

99,65,1

42,78,0

37,73,0

75,60,1

76,87,1

83,84,1

90,65,1

76,81,1

56,65,0

7.3 附录 3 相关代码

7.3.1 代码文件 sigmoid.m

```
function g = sigmoid(z)
% Sigmoid 函数
    g = 1./(1+exp(-z));
end
```

7.3.2 代码文件 costFunction.m

```
function J = costFunction(theta, X, y)
% 返回代价函数值
   m = length(y);
   % 逻辑回归的代价函数有两部分相加组成
   first = -y.*log(sigmoid(X*theta'));          % 第一部分
   second = -(1-y).*log(1-sigmoid(X*theta'));   % 第二部分
   J = sum(first+second)/m;
end
```

7.3.3 代码文件 main.m

```
%
%  main.m 主函数文件
%
clear; close all; clc;


% Part 1 加载数据，将训练集导入 %

fprintf('Part 1 加载数据，将训练集导入\n');
data = load('trainingdata.txt');
X = data(:, 1:2);   % 特征矩阵
y = data(:, 3);     % 标签向量
X = X/100;

% 先看一下样本数据 %
pos = find(y==1); neg = find(y == 0); % 找到正样本与负样本

% 在子图1中绘制样本分布图
subplot(2, 2, 1);
plot(X(pos,1), X(pos,2), 'k+', X(neg,1), X(neg,2), 'ko', 'MarkerFaceColor', 'y');
xlabel('科目一成绩');
ylabel('科目二成绩');
title('样本分布', 'fontname', '仿宋', 'Color', 'b', 'FontSize', 15);

% 在子图2中绘制样本分布，在完成拟合后，再添加决策边界方程
subplot(2, 2, 2);
plot(X(pos,1), X(pos,2), 'k+', X(neg,1), X(neg,2), 'ko', 'MarkerFaceColor', 'y');
hold on;


% Part 2 拟合模型前的准备工作 %

fprintf('\nPart 2 拟合模型前的准备工作\n');
[m, n] = size(X);       % m是训练集样本数，n是特征数
X = [ones(m, 1) X];     % 特征矩阵加入第一列常数1
theta = zeros(1,n+1);   % 初始化拟合参数，全部为0
```

```
% Part 3 梯度下降 %

fprintf('\nPart 3 梯度下降\n');
cost1 = costFunction(theta,X,y); % 先看一下训练前的代价函数值
fprintf('训练前的代价函数值：%f\n', cost1);

learn_rate = 0.05;  % 学习率
iter = 150000;        % 迭代次数
vector_cost = zeros(iter, 1);   % 存储每一次迭代后的代价函数值

for i=1:iter    % 进行iter次迭代
    % theta参数矩阵中的每一个参数值都应该在当前次的梯度下降之后更新，故设置一临时变量保存theta的值。
    % 在对n+1个参数更新的时候，只需要更新temp矩阵，最后将temp重新赋给theta
    temp = theta;
    for j=1:n+1   % 更新每一个theta的值
        temp(j) = temp(j)-learn_rate/m*sum( (sigmoid(X*theta')-y).*X(:,j) );
    end
    theta = temp;   % 将临时变量temp的值重新赋给theta
    vector_cost(i) = costFunction(theta, X, y); % 记录当前参数对应的代价函数值
end



% Part 4 打印输出 %

fprintf('\nPart 4 打印输出\n');
fprintf('\n最终代价函数收敛于值：%f\n', vector_cost(iter));
fprintf('得到的参数theta：\n');
disp(theta);

% 得到决策边界方程的字符串表示，并打印
fin_equation = string(theta(1))+'+'+string(theta(2))+'*x1+'+string(theta(3))+'*x2';
fprintf('拟合得到的决策边界方程：%s\n', fin_equation);

subplot(2, 2, 2);
equation = @(x1,x2)theta(1)+theta(2)*x1+theta(3)*x2;     % 决策边界的隐函数方程
fimplicit(equation, [0.3 1 0.3 1]); % 训练集中，特征一最小值为0.30588，特征值二的最小值为30.6033，故绘图的边界设置为0.3
xlabel('科目一成绩');
ylabel('科目二成绩');
title('决策边界的直线方程', 'fontname', '仿宋', 'Color', 'b', 'FontSize', 15);

subplot(2, 2, 3);
```

```
plot(vector_cost);
title('梯度下降中，代价函数的收敛趋势', 'fontname', '仿宋', 'Color', 'b', 'FontSize',
15);
xlabel('迭代次数');
ylabel('代价函数值');

% Part 5 验证模型准确性 %
fprintf('\nPart 5 验证模型准确性\n');

test_data = load('testdata.txt'); % 加载测试集数据
[test_m, test_n] = size(test_data); % m代表测试集数据的样本数量
% 取出测试集样本的两个特征列向量及标签列向量
test_X1 = test_data(:, 1);
test_X2 = test_data(:, 2);
test_Y = test_data(:, 3);
% 两个特征值均缩小100倍
test_X1 = test_X1./100;
test_X2 = test_X2./100;

% 写出建立的预测模型方程
syms x1 x2
predict_function(x1, x2) = sigmoid(theta(1)+theta(2)*x1+theta(3)*x2);
% 储存预测结果的列向量
predict_tags = zeros(test_m, 1);
% 开始预测
for i=1:test_m
    temp = predict_function(test_X1(i), test_X2(i));
    % 预测值大于0.5，则认为该样本属于正样本
    if temp >= 0.5
        predict_tags(i) = 1;
    % 否则，认为该样本属于负样本
    else
        predict_tags(i) = 0;
    end
end
flags = predict_tags == test_Y; % 相同则为1，预测成功；不同则为0，预测失败

% 打印预测结果
for i=1:test_m
    temp = '预测成功';
    if flags(i)==0
        temp = '预测失败';
    end
```

```
    fprintf('第%d个样本的预测结果是：%s\n', i, temp);
end
fprintf('该预测模型在测试集上预测成功的准确率为%f\n', sum(flags)/test_m);

subplot(2, 2, 4);
pos = find(test_Y==1); neg = find(test_Y == 0); % 找到正样本与负样本
plot(X(pos,1), X(pos,2), 'k+', X(neg,1), X(neg,2), 'ko', 'MarkerFaceColor', 'y');
plot(test_X1(pos), test_X2(pos), 'k+', test_X1(neg), test_X2(neg), 'ko',
'MarkerFaceColor', 'y');
hold on;
fimplicit(equation, [0.3 1 0.3 1]);
xlabel('科目一成绩');
ylabel('科目二成绩');
title('预测模型的准确度可视化', 'fontname', '仿宋', 'Color', 'b', 'FontSize', 15);
```

参考文献

[1] 周志华. 机器学习.北京：清华大学出版社，2016.

[2] Gavin Hackeling 著，张浩然 译.scikit-learn.北京：人民邮电出版社，2019.