人工智能实验报告 实验六

姓名：丁晓琪　学号:22336057

一.实验题目

`data.csv`数据集包含三列共400条数据，其中第一列`Age`表示用户年龄，第二列`EstimatedSalary`表示用户估计的薪水，第三列`Purchased`表示用户是否购房。请根据用户的年龄以及估计的薪水，利用逻辑回归算法和感知机算法预测用户是否购房，并画出数据可视化图、loss曲线图，计算模型收敛后的分类准确率。

二.实验内容

2.1逻辑回归算法

1.算法原理

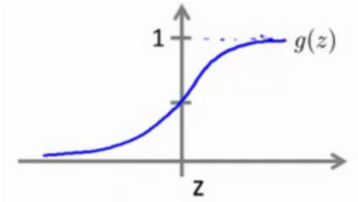
- **逻辑回归的假设函数公式**：θ^T为我们要训练的参数，x为训练数据数据输入的特征

h_θ可以近似看为得到的分类为1概率，当它大于等于0.5时判别分类标签为1，小于0.5时判别分类标签为0

$$h_\theta(x) = g(\theta^T x)$$

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

$g(z)$的图像如下：由此易得$\theta^T x >= 0$，$h_\theta(x) >= 0.5$,即分类为1



- $\theta^T(x)$**的选取**：有两个特征值分别为x_1(age)和x_2(薪水)还有一个隐藏的$x_0=1$作为偏置因子

  此次实验实现了两个模型，根据测试模型2的效果更好，以下皆为模型2的分析

$$
\begin{aligned}
1.\quad & \theta^T(x)=\theta_0+\theta_1x_1+\theta_2x_2 \\
2.\quad & \theta^T(x)=\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_2^2
\end{aligned}
$$

- **参数θ的训练**：使用梯度下降法，$J(\theta)$为代价函数，α为学习率，公式为：

$$\theta_i=\theta_i-\alpha\frac{\delta J(\theta_i)}{\delta\theta_i}$$

  注意：每次同时对所以特征参数迭代

- **代价函数$J(\theta)$和损失函数$Cost(h_\theta(x^i),y)$的定义**：(此处参考课外资料)

$$Cost(h_\theta(x^i),y)=-y*log(h_\theta(x))-(1-y)*log(1-h_\theta(x))$$

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

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

  m：数据的数量　$x^{(i)}$：第i个数据的特征　$x_j^{(i)}$第i个数据的第j个特征　$y^{(i)}$：第i个数据的标签值

- **加入正则化后的参数训练**：防止过拟合，为每个参数加上代价惩罚（一般不对偏置因子对应参数θ_0）做惩罚，

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}(Cost(h_\theta(x^{(i)})+\frac{\lambda}{m}\theta_j)$$

$$\theta_j=\theta_j-\alpha\frac{1}{m}[\sum_{i=1}^{m}[h_\theta(x^{(i)})-y^{(i)}]x_j^{(i)}+\frac{\lambda}{m}\theta_j]$$

- **数据的归一化处理**：由于该次实验中两个特征年龄和薪水的数据分布差异较大，所以要对数据采取归一化处理，让它们对模型的影响

  效果一致

$$x1_{new}=\frac{x_1-x1_{min}}{x1_{max}-x1_{min}}$$

$$x2_{new}=\frac{x_2-x2_{min}}{x2_{max}-x2_{min}}$$

2.算法实现:

1. 读取数据，整理数据，归一化数据
2. 确定使用模型，初始化模型参数和学习步长
3. 根据输入迭代次数进行多次迭代更新参数

```
#time是迭代次数
#loss_list:记录这次迭代中的代价，每隔10次迭代采一次样
#每次迭代，对参数做一次梯度下降更新
        for i in range(0,time):
            if(i%10==0):
                loss=self.Loss()
                self.loss_list.append(self.Loss())
                print(loss)
            self.Gradient_Descent()#更新所有参数
```

3.关键代码展示

- **梯度下降算法更新参数:**

  用code实现加入正则化后的参数训练中展示参数更新的公式

```
    #对intX求sigmod函数，对应逻辑回归假设函数中的g(z)计算，intX是列向量，用矩阵计算，实现同时计算多个数据的sigmod
    def sigmoid(self,inX):
        res = np.zeros(inX.shape)
        for i in range(inX.shape[0]):
            if inX[i] >= 0: #避免一些数值计算误差
                res[i] = 1 / (1 + np.exp(-inX[i]))
            else:
                res[i] = np.exp(inX[i]) / (1 + np.exp(inX[i]))
        return res

   # x1_x2中放置模型需要的特征（经过归一化处理）(m*4的矩阵)，x1_x2[i]是第i个数据的特征
    ## x1_x2[i][0]=1,[1]=age,[2]=salary [3]=salary*salary
   # lable[i]是第i个数据的标签  1*m的矩阵
   # arg是模型的所有参数 1*4的列表
    def Gradient_Descent(self):
        dataMatrix = np.mat(self.x1_x2)
        labelMat = np.mat(self.lable).transpose()  # 转置
        m, n = dataMatrix.shape         # m是数据总数，n是参数总数
        weights = np.ones((n,1))
        for i in range(0,n):
            weights[i][0]=self.arg[i]  #weight是arg的倒置 4*1的向量
        #梯度计算
        grad = -((labelMat - self.sigmoid(dataMatrix * weights)).transpose() * dataMatrix).transpose()
        #梯度更新参数，未正则化
        weights = weights - self.alpha * grad
        #参数正则化，并且写回arg
        for i in range(0,n):
            if i==0:
                self.arg[i]= (weights[i][0])[0,0]
```

```
                else:
                    self.arg[i]= (weights[i][0])[0,0]-self.alpha*self.reg/m*self.arg[i]
```

- **每次迭代的代价计算:**

  实现代价函数和损失函数定义中有关损失函数的公式和

```
    def Hypothesis_Function(self,index:int):#计算逻辑回归的的假设值，index是要计算的数据的标号
        #直接运算,第一个参数0次的不要乘特征
        z=0
        for h_i in range(0,self.arg_num):
            if(h_i>=len(self.x1_x2[index])):
                fag=1
            z+=(self.arg[h_i]*self.x1_x2[index][h_i])
        #z=self.arg[0]+self.x1_x2[index][1]*self.arg[1]+self.x1_x2[index][2]*self.arg[2]+self.x1_x2[index][3]*self.arg[3]
        if z>=0: #对sigmoid函数优化，避免出现极大的数据溢出
            return 1.0 / (1 + np.exp(-z))
        else:
            return np.exp(z)/(1+np.exp(z))

    def Cost(self,index):#单个data损失计算
       # theta=np.matrix(self.arg)
       h_arg=self.Hypothesis_Function(index)
       y=self.lable[index]
       ret=(-1*y*np.log(h_arg+ 1e-5))-((1-y)*np.log(1-h_arg+ 1e-5))#防止浮点数溢出
       return ret

    def Loss(self):#计算当前模型所有数据的预估和真实之间的误差之和
        num=len(self.x1_x2)
        sum=0
        for i in range(0,num):
            sum+=self.Cost(i)
        ##加入正则化
        sum1=0
        for i in range(1,self.arg_num):#不对第一个参数惩罚
            sum1+=(self.arg[i]*self.arg[i])
        sum1*=self.reg
        sum+=sum1
        return sum/num
```

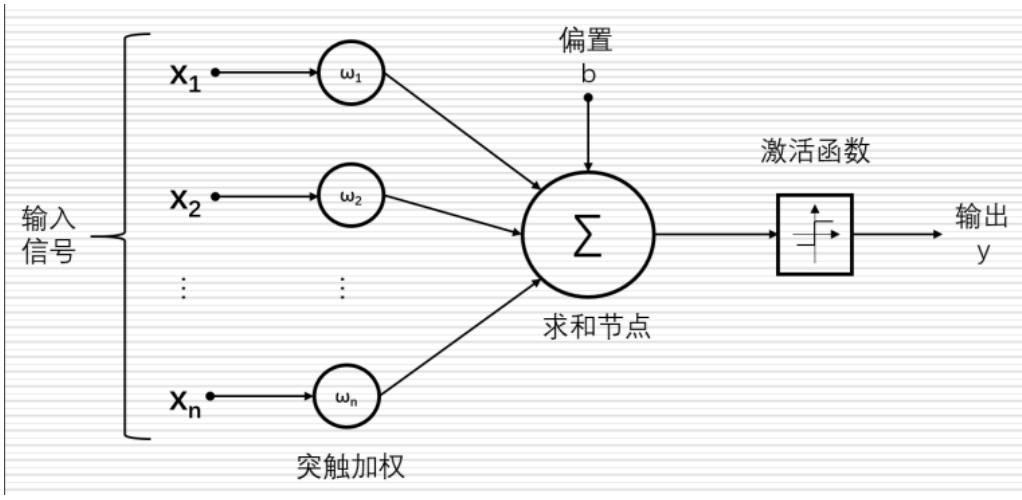
4.创新点&优化

- 为了更好的拟合将线性的模型改成了曲线模型:模型的选取
- 避免过拟合，加入正则化。

2.2感知机算法

1.算法原理

- **模型**



$$sign(x) = \begin{cases} -1, & \text{if x<0} \\ 1, & \text{if x>=0} \end{cases}$$

$$y = sign(\omega \cdot x + b)$$

- **代价函数：**

$$M\text{说预测中误分类点的集合}$$

$$L(\omega, b) = -\frac{1}{M}\sum_{x^{(i)} \in M} y^{(i)}(\omega \cdot x^{(i)} + b)$$

- **参数ω的随机梯度下降法**

$$\omega_i = \omega_i - \alpha\frac{\delta J(\omega_i)}{\delta\omega_i}$$

$$\omega = \omega + \alpha y^{(i)} x^{(i)}$$

$$\text{偏置因子更新：} b = b + \alpha y^{(i)} \quad (x^{(i)}, y^{(i)}) \in M$$

- **本次实验使用的模型**：

$$x_{age} = \frac{x_{age} - x_{age_{min}}}{x_{age_{max}} - x_{age_{min}}}$$

$$x_{salary} = \frac{x_{salary} - x_{salary_{min}}}{x_{salary_{max}} - x_{salary_{min}}}$$

$$y = sign(\omega_1 * x_{age} + \omega_2 * x_{salary} + b)$$

2.算法实现

1.获取数据，整理数据，提取数据的标签值和特征值，对特征值要进行归一化处理

　注意原数据的标签值是0，1，但是sign函数只能将标签区分为-1，1，所以要对原标签值做出改动，原标签值为0的改为-1

```
            #这里调整一下y的取值
            if int_list[2]==0:
                int_list[2]=-1
            self.lable.append(int_list[2])
```

2.初始化模型参数，学习步长

3.多次迭代中用梯度下降法更新参数，并在原数据集上检验

```
    def Iterate(self,time):#参数的迭代更新和每代的误差的获得
        for i in range(0,time):
            self.loss_list.append(self.Loss())
            if(i%10==0):
                print(self.Loss())
            self.Gradient_Descent()#更新所有参数
```

- **3.关键代码展示**
- **定义：**

```
self.lable #存储每个数据的标签
self.x1_x2 #存储每个数据特征值 其中[1]是归一化后的年龄，[2]是归一化后的薪水
self.arg  #存储偏置因子与参数 [0]是偏置因子 [1]是年龄的参数 [2]是薪水的参数
self.alpha #是训练步长
self.loss_list #存储迭代采样的代价
```

- **随机梯度下降的参数更新：**

实现ω的更新，编程实现算法原理中的[随机梯度下降法公式]

```
 def Gradient_Descent(self):
        for index in range(0,len(self.lable)):
            if self.Cost(index)>=0:
                #cost>=0是误分类点，更新参数    #随机梯度下降法，只有误分类点才能更新参数
                for i in range(0,self.arg_num):
                    if i==0: #偏置因子的更新
                        self.arg[i]=self.arg[i]+self.alpha*self.lable[index]
                    else:  #参数更新
                        self.arg[i]=self.arg[i]+self.alpha*self.lable[index]*self.x1_x2[index][i]
```

- **代价函数和损失函数：**

编程实现算法原理中的[代价函数]

```
        def Cost(self,index):#单个data损失计算
       # theta=np.matrix(self.arg)
       y=self.lable[index]
       ret=-y*(self.arg[0]+self.x1_x2[index][1]*self.arg[1]+self.x1_x2[index][2]*self.arg[2])
```

```
            return ret

        def Loss(self):#计算当前模型所有数据的预估和真实之间的误差之和
            num=len(self.data)
            sum=0
            count=0
            ##只有分类错误的才能算代价
            for i in range(0,num):
                if self.Cost(i)>=0:
                    sum+=self.Cost(i)
                    count+=1 #误分类的总数
            return sum/count
```

- **模型检验:**

  将整个数据集的数据输入到训练好的感知机模型中，检验感知机的判断的正确率

```
    def Hypothesis_Function(self,index:int):#计算多层感知机的的假设值，index是要计算的数据的标号
        #直接运算,第一个参数0次的不要乘特征   多层感知机是sign函数不是sigmod函数
        z=self.arg[0]+self.x1_x2[index][1]*self.arg[1]+self.x1_x2[index][2]*self.arg[2]
        if z>=0:
            return 1
        else:
            return -1
    def Iterate(self,time):#参数的迭代更新和每代的误差的获得
        for i in range(0,time):
            self.loss_list.append(self.Loss())
            if(i%10==0):
                print(self.Loss())
            self.Gradient_Descent()#更新所有参数
        #计算正确率arg[0]+arg[1]x1+ arg[2]x2>0 1小于0就是0
        sum=0
        for i in range(0,len(self.lable)):
            tm=self.Hypothesis_Function(i)
            if(tm>=0 and self.lable[i]==1):
                sum+=1
            elif(tm<0 and self.lable[i]==-1):
                sum+=1
        print("正确率：",sum/len(self.lable))
```

4.创新点&优化

三.实验结果及分析

3.1逻辑回归算法

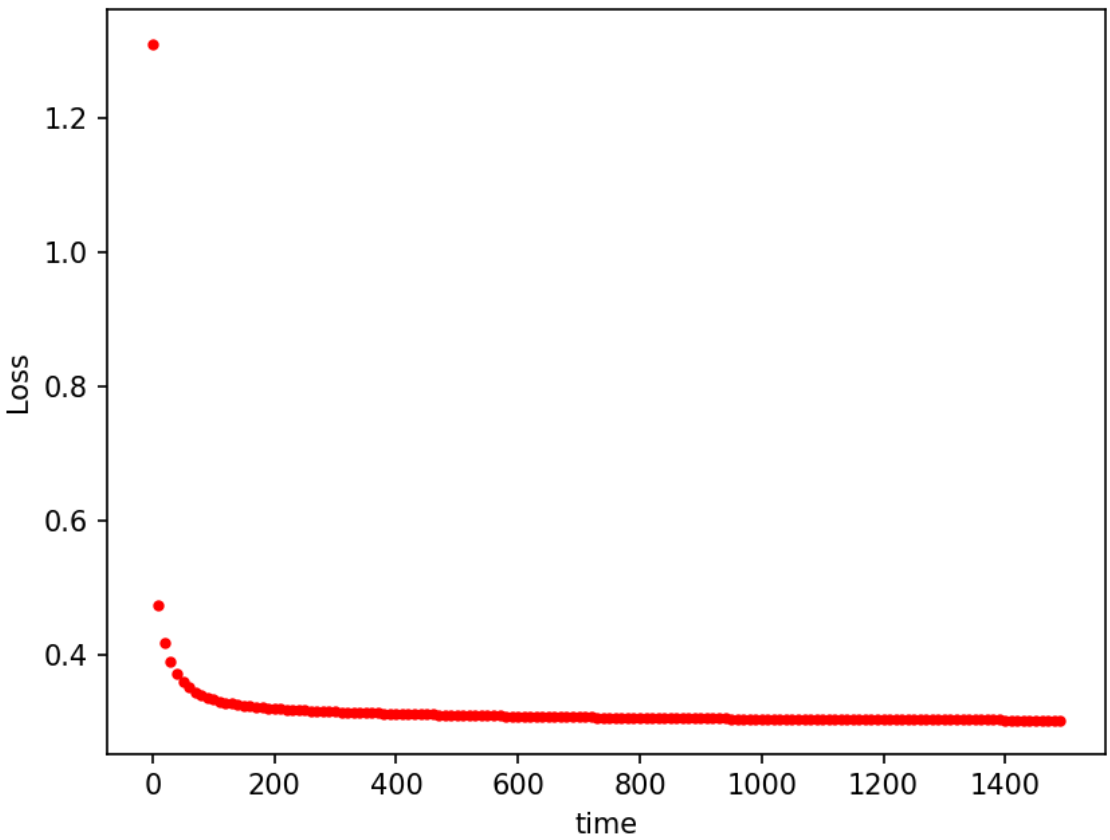
结果展示:

学习步长 α=0.01, 正则化参数 λ=0.01,迭代次数 1500

在整个数据集上训练和测试:

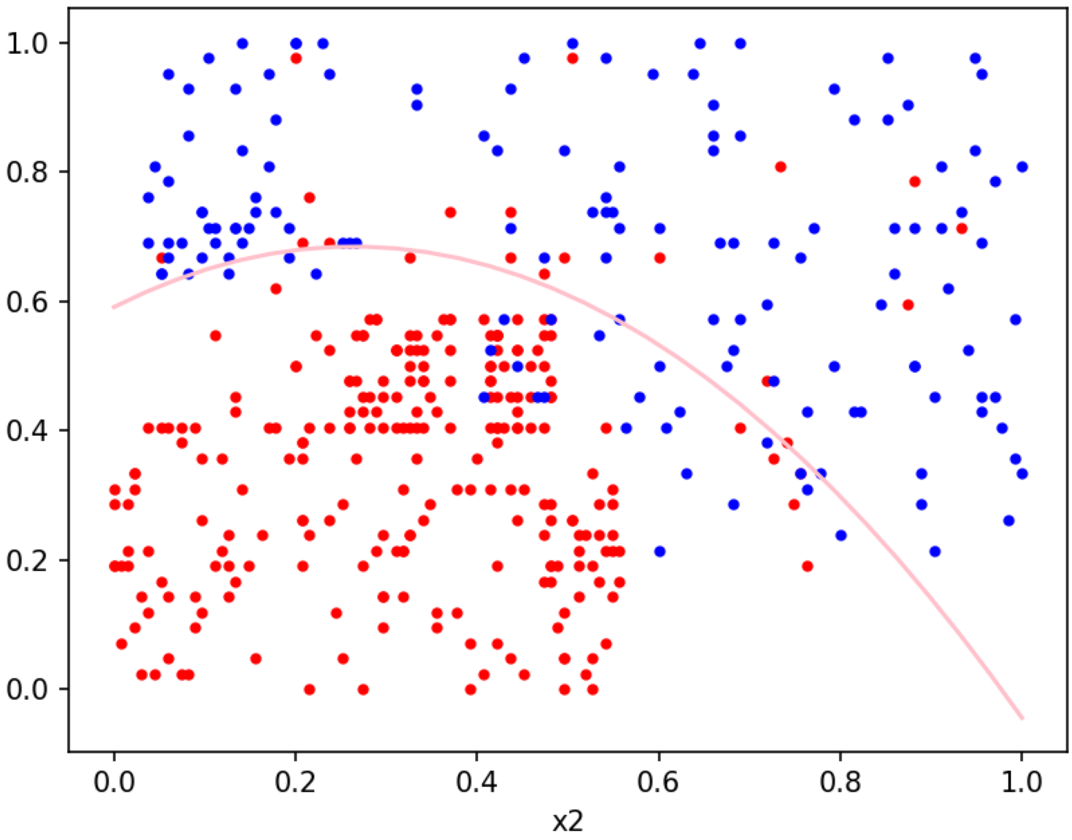
```
0.3033928379971466
正确率：  0.8975
[-5.637823532273525, 9.541262986245263, -6.76722742568166, 12.826908059157356]
```

逐次迭代和参数学习中数据的代价变化：



数据可视化：x2（横轴):归一化后的房价；x1（纵轴）：归一化后的年龄；blue point：标签为1的数据点；

red point：标签为0的数据点；曲线：拟合曲线，曲线上方是预测标签为1的数据，曲线下方说预测标签为0的数据



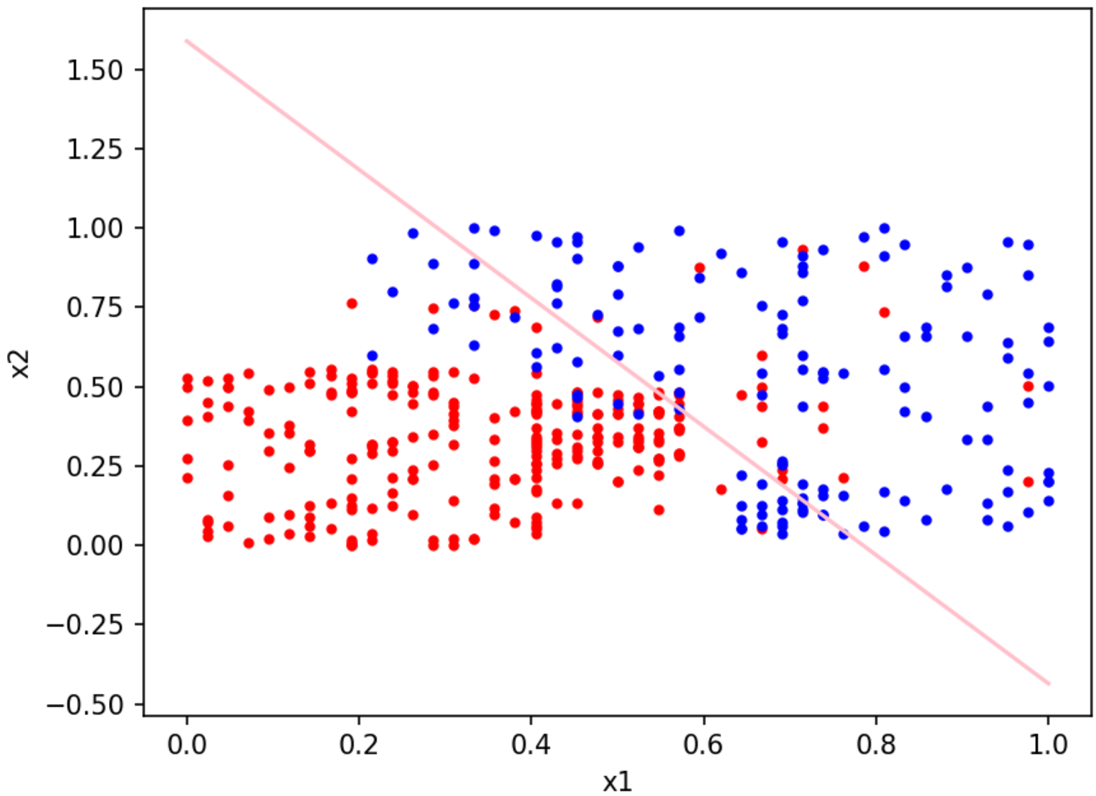
评测指标展示及分析：

1. 正确率：该模型预测正确率约90%，能将数据几乎正确的分类
2. 学习步长分析：从迭代过程中代价的变化Loss迭代曲线可以看出，学习步长为0.01时适合，没有产生因为α过大而产生Loss迭代曲线

   的震荡，也没有产生因为α过小而导致的Loss迭代曲线收敛慢，没有收敛
3. 模型分析：从拟合的分类曲线来看，采取第二个模型的拟合效果比第一个好

$$1.\quad \theta^T(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$
$$2.\quad \theta^T(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_2^2$$

第一个的拟合分类线如下：（且第一个的正确率比第二个低）




```
0.845
[-7.628743908214676, 9.722066801787374, 4.803561698879533]
```

3.2感知机算法:

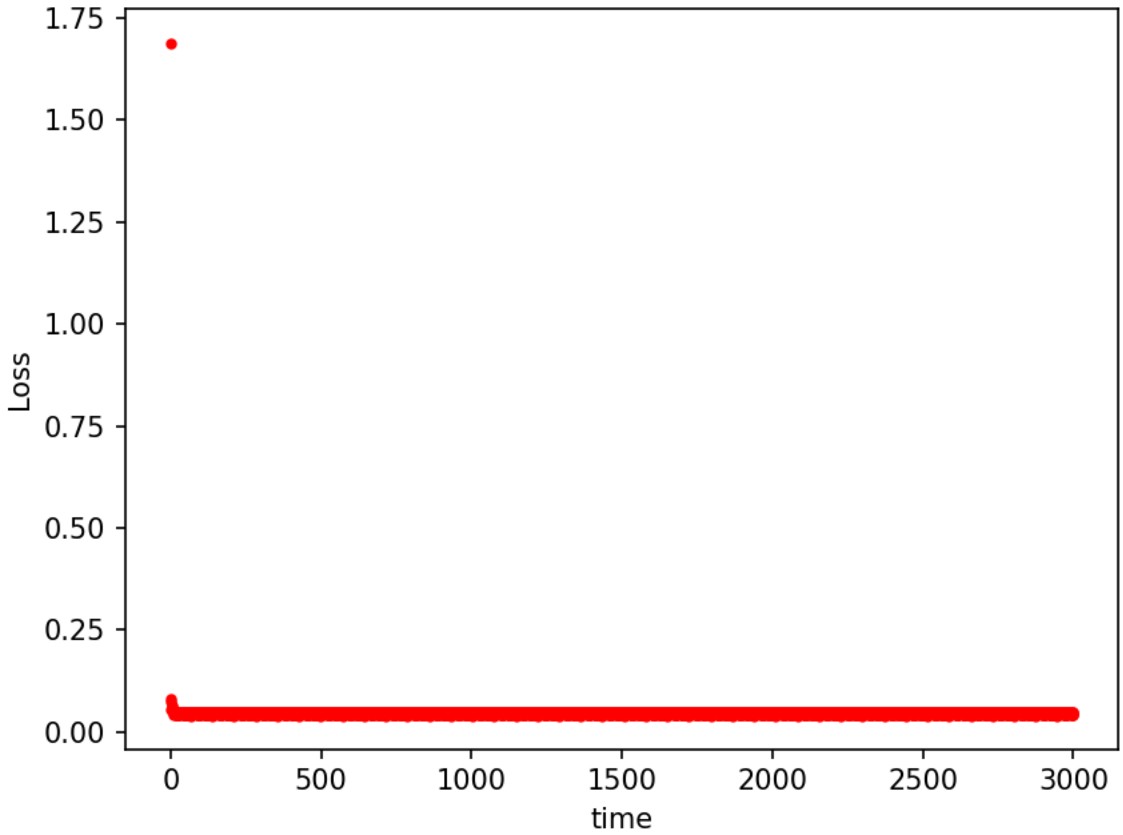
结果展示:

学习步长α:0.06,迭代次数3000次

在整个数据集上测试训练

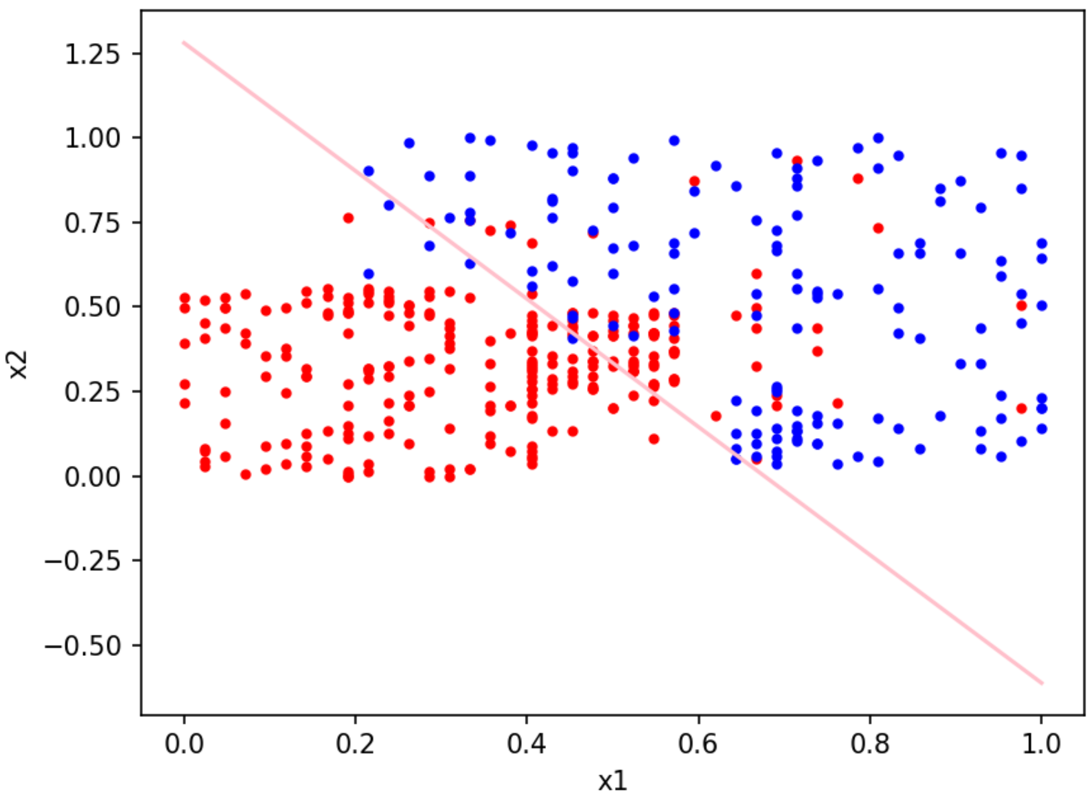
```
正确率:  0.8
[-0.26000000000000045, 0.38428571428601027, 0.2031111111115315]
```

Loss代价的迭代曲线:



分类拟合曲线：纵轴x2为归一化后的薪水，横轴x1为归一化后的年龄，red point: 标签为1的数据 blue point：标签为0的数据

分类拟合直线上方预测标签为0的数据，下方为预测标签为1的数据



评测指标展示及分析：

1. 正确率：该模型预测正确率约80%，能将数据大致正确的分类
2. 学习步长分析：从迭代过程中代价的变化Loss迭代曲线可以看出，学习步长为0.06时适合，没有产生因为α过大而产生Loss迭代曲线

   的震荡，也没有产生因为α过小而导致的Loss迭代曲线收敛慢，没有收敛。有额外测试0.01和0.05，但是相同的迭代

   次数下正确率约为78.5%，则0.06较优
3. 模型分析：从拟合的分类直线，能把数据分为两个区域，给予大致正确的预测

四.参考资料(可选)

逻辑回归算法参考：http://www.ai-start.com/ml2014/

感知机算法参考：https://blog.csdn.net/qs17809259715/article/details/100623719