**算法思路：**

**1.GD求解logistics回归:**

首先根据题目所给数据，给出目标函数形式为。且y只在1和0之间取值。同时将x数据进行标准化处理，公式为：(X-X\_均值)/X\_标准差。统计每次迭代的损失值和权重，利用损失值绘制折线图观察变化趋势；每次迭代后绘制决策边界，观察变化情况。

我们取初始𝜃向量为[0;0;0]（即全0列向量）。并设置学习率a=0.1，按照梯度下降法的公式进行迭代。当两次迭代结果之差小于0.001，认为结果收敛，停止迭代。我的代码总迭代次数为11次。观察决策边界，可以发现初次初次迭代后就已经有了很好的结果。

**2.SGD求解logistics回归:**

思路与梯度下降法类似，只是在每次迭代时，并不是选择全部样本进行计算，在这里我使用了每个样本来更新梯度，在我的代码中，即使设置迭代次数为1，由于内部分别于每个样本更新了一次权重，实际更新次数为（外部迭代次数\*样本数量）。

在这里我没有设置收敛条件，因为每次会迅速收敛，也可能是我的代码存在问题。在这里我将外部迭代次数设置为10，可以发现loss曲线不如GD求解的logistic回归。

**3.GD求解softmax回归:**

实际上softmax回归可以说是logistics回归的扩展，而logistics回归正是softmax回归当分类数为2时的特殊情况。同样地，我们先将x数据进行标准化处理，这里我使用的模型没有偏置项。我们取𝜃向量为[[0，0]，[0，0]]（即2\*2全0矩阵），将标签Y设置为one-hot编码，其中第一列为类别1，第二列为类别0。设置学习率a=0.1，按照梯度下降法的更新公式进行迭代。当两次迭代结果之差小于0.001，我们认为结果收敛，停止迭代。

运行后可以发现迭代了13次，较logistic慢，同时损失曲线收敛速度也不让logistic快

**4.SGD求解softmax回归:**

思路与梯度下降法类似，只是在每次迭代时，并不是选择全部样本进行计算，在这里我尝试了每次迭代只使用了一个样本进行更新，也使用了和SGD求解logistic的方法，当一次迭代只使用一个样本时收敛速度很慢。

通过绘制的决策边界变化情况和损失值收敛速度可以看出SGD求解的收敛速度。