**哈尔滨工业大学计算机科学与技术学院**

**《机器学习》**

**Lab 2：逻辑回归**

|  |  |  |
| --- | --- | --- |
| **姓名** | **班级** | **学号** |
| 楼雨京 | 1436101 | 1140310415 |

**1 实验目标**

* 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。
* 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。
* 验证：

1.可以手工生成两个分别类别数据（可以用高斯分布），验证你的算法。考察类条件分布不满足朴素贝叶斯假设，会得到什么样的结果。

2. 逻辑回归有广泛的用处，例如广告预测。可以到UCI网站上，找一实际数据加以测试。

**2 实验环境**

python编译环境

**3 实验过程**

本次实验实现了逻辑回归，即一个完成二分类的模型。这里我利用梯度下降法进行优化，并且一共实现了三种场景，每一种场景都设计了有无惩罚项两种情况（两者的区别仅在计算损失函数与计算梯度的两个函数里面）。

场景一：给定样本属性数量，根据高斯分布随机生成的一组绝对可分正负的样本，即线性可分的，最后得到一个线性超平面进行分类。

场景二：给定样本属性数量，根据高斯分布随机生成一组不一定可分正负的样本，即线性不可分的。由于这种情况不一定能够找到一个线性超平面，所以我在这才用核函数的办法，将原始样本映射到一个更高维的空间，使得在这个高维空间线性可分，找到高维空间的超平面，从而将原空间的数据进行分类。

场景三：在UCI网站上找了一组，关于预测“做完乳腺癌手术后存活时间是否能够超过5年”的数据，验证逻辑回归模型的正确性。

**3.1 场景1**

为了方便演示，我这边设计样本具有两种属性值，x1和x2，在随机生成数据后，进行逻辑回归得到一条分类的曲线。这边我正例，负例各随机生成15个，并加入高斯分布。下图中黑线为用于生成数据的直线x+y=3，绿线为逻辑回归生成的直线。

1. 未加惩罚项时

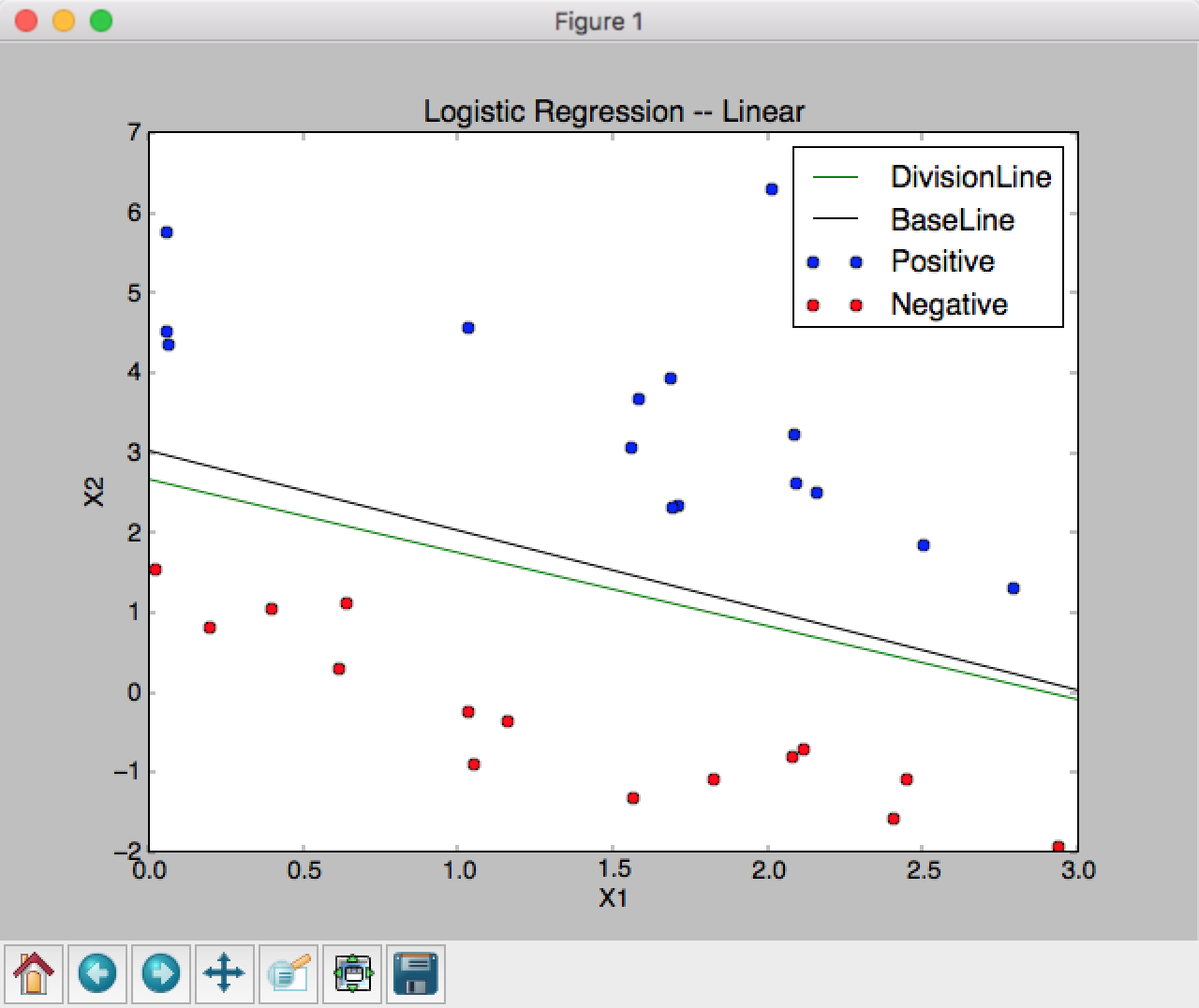


图1

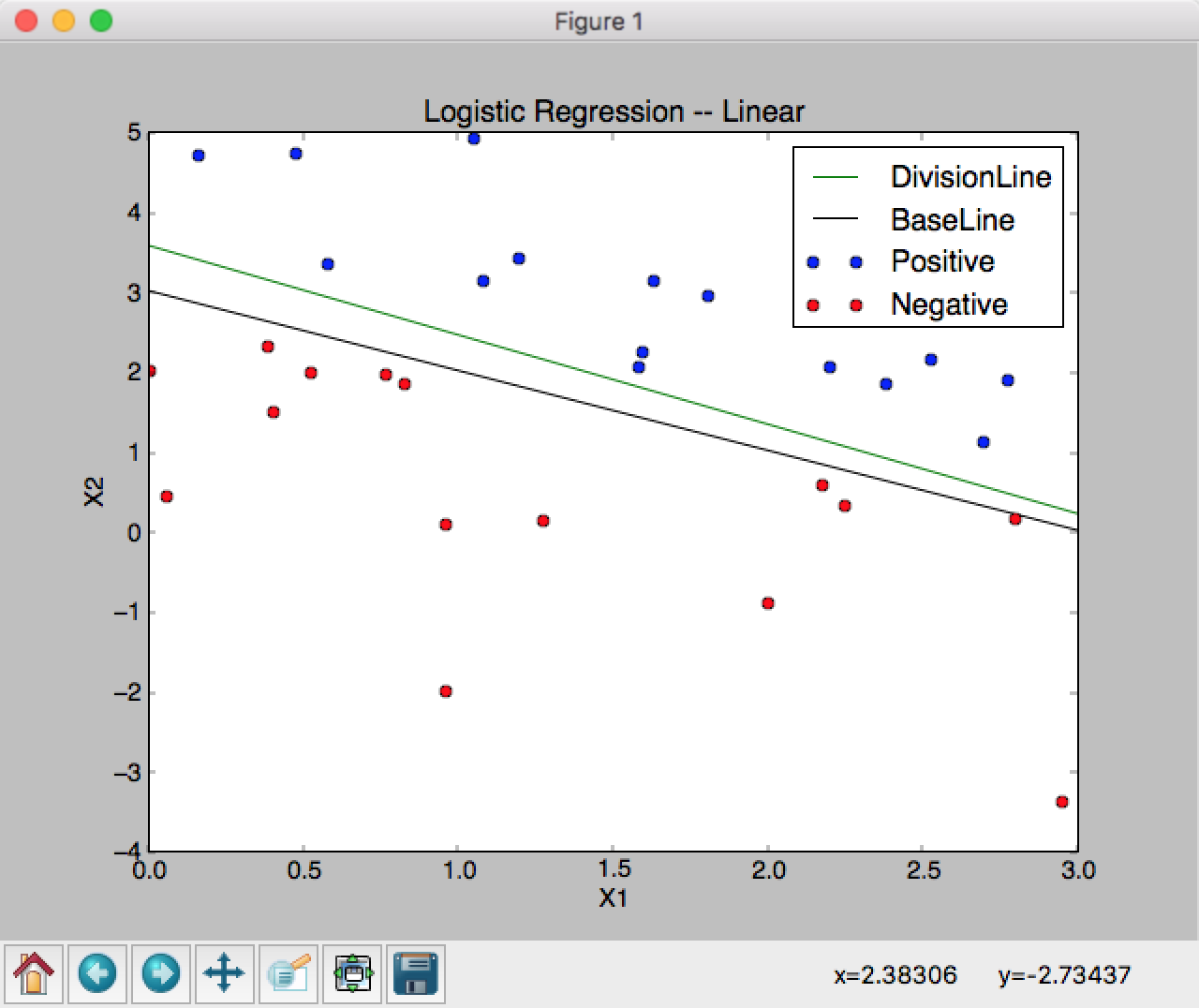


图2

1. 加入惩罚项（lambda=0.1）

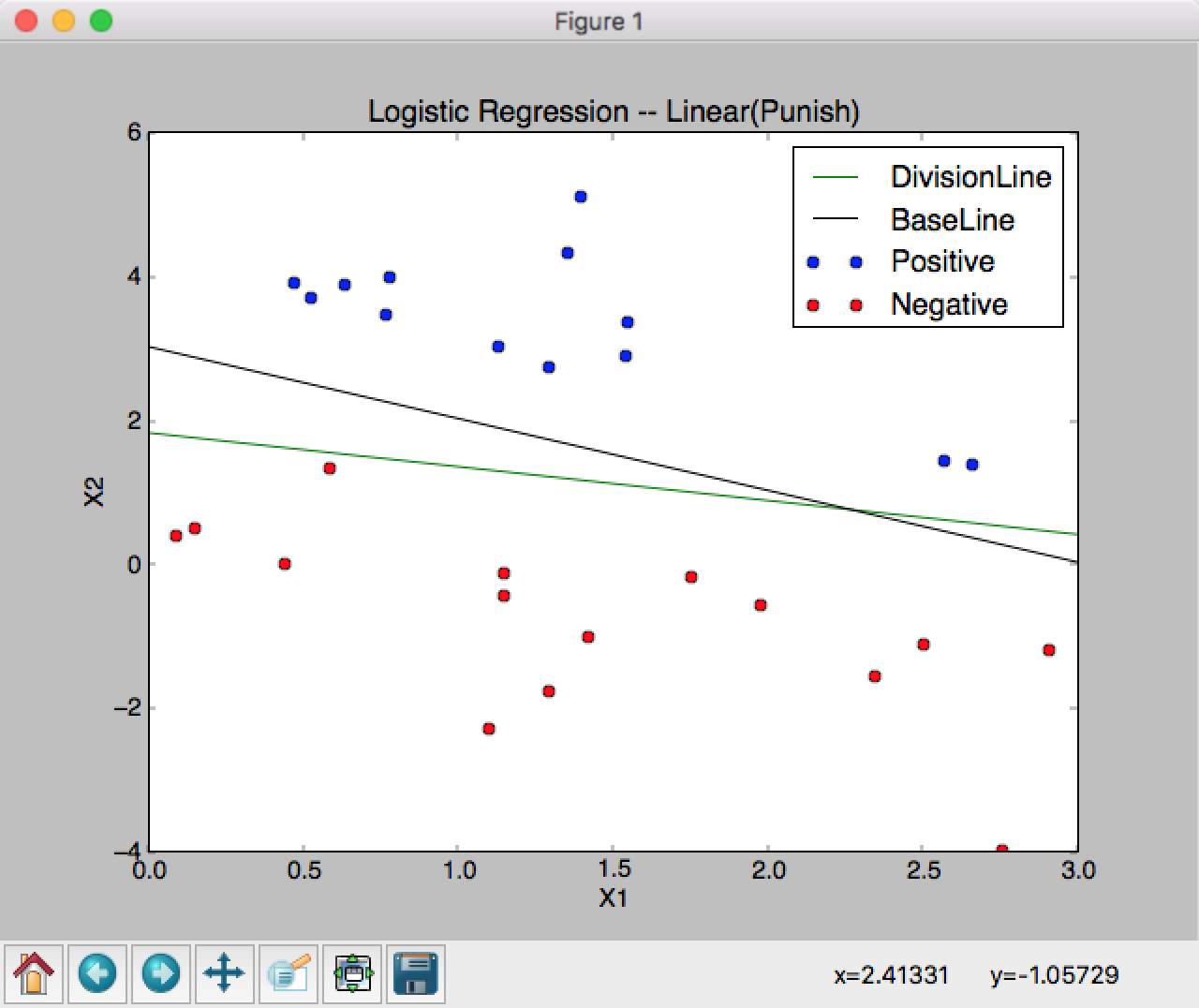


图3

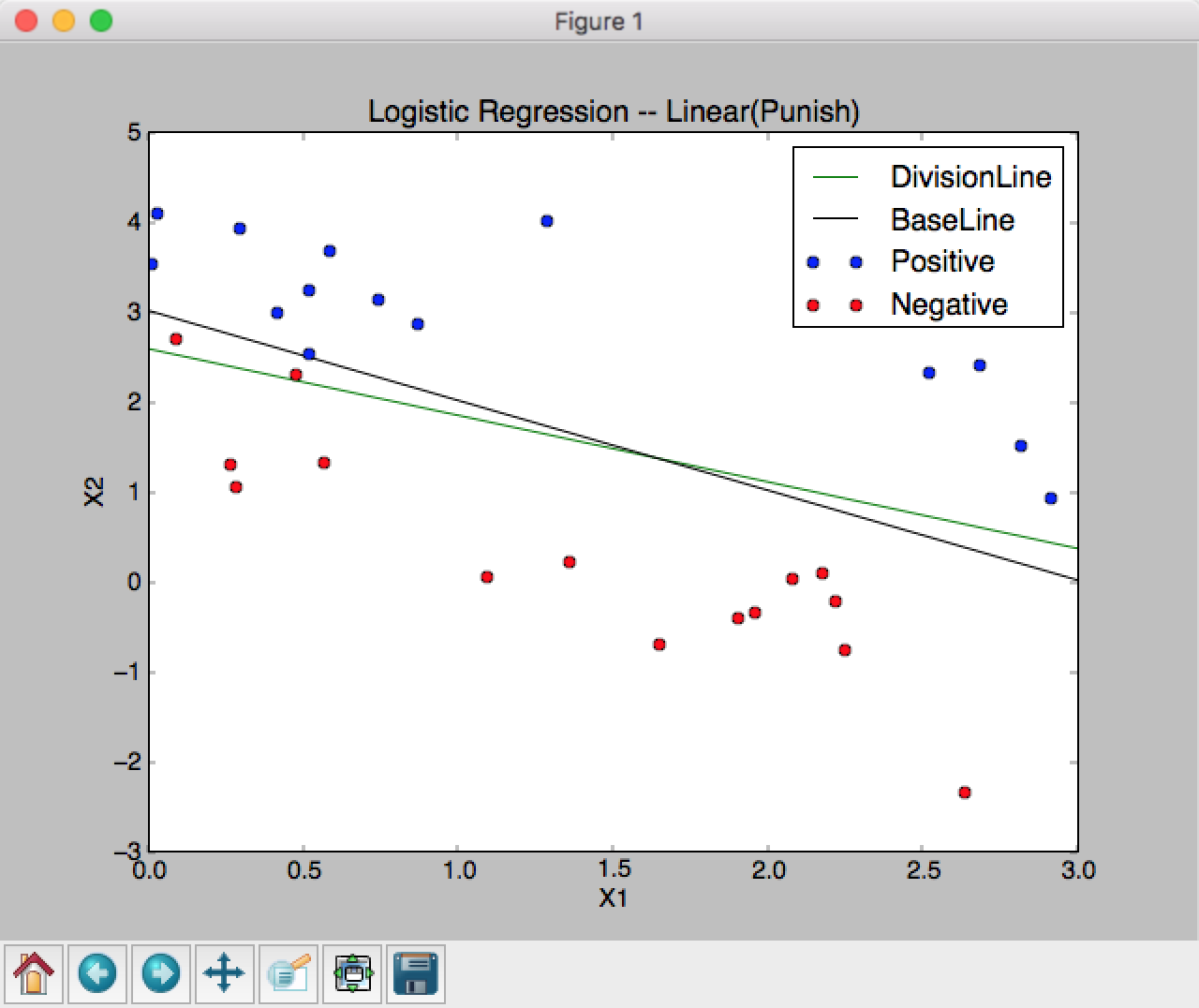


图4

在此，我们可以分析下实验结果。很显然最终我们较好的得到了逻辑回归的结果，实现了正反样例的分类。其中最明显的变化是加入惩罚项后迭代的速度大大增加（通过运行程序可以发现），能够在更少的迭代次数下得到相似的效果。其次就是加入惩罚项后能够很好地规避过拟合的问题，从图4可以看出，尽管没有将正反例全部划分，但是所得到的分类线与产生样本的直线更接近些。

当样本属性数量更多时，我们只需相应地增加预测函数的权值数量即可。这里我们完成了对线性可分的样本的分类。

**3.2 场景2**

场景2实现了对线性不可分的样本进行分类，为了方便演示，我这边展示的依旧是样本带有两个属性值的情况。为了解决线性不可分的情况，我这才用了增加样本维数的方法，即利用核函数的方法。我所用的核函数如图5。

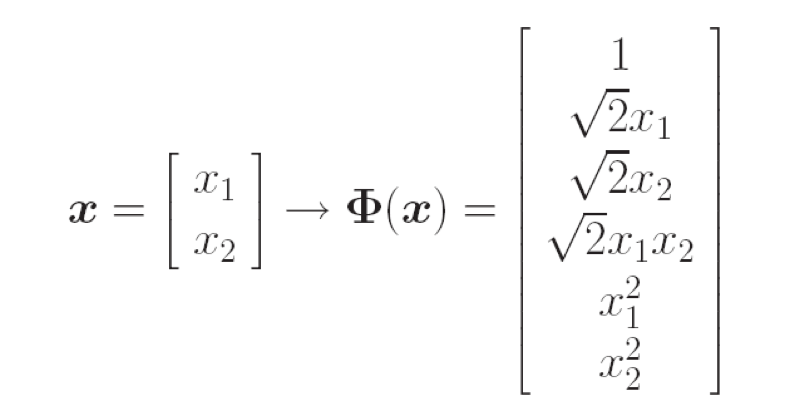


图5

1. 未加惩罚项

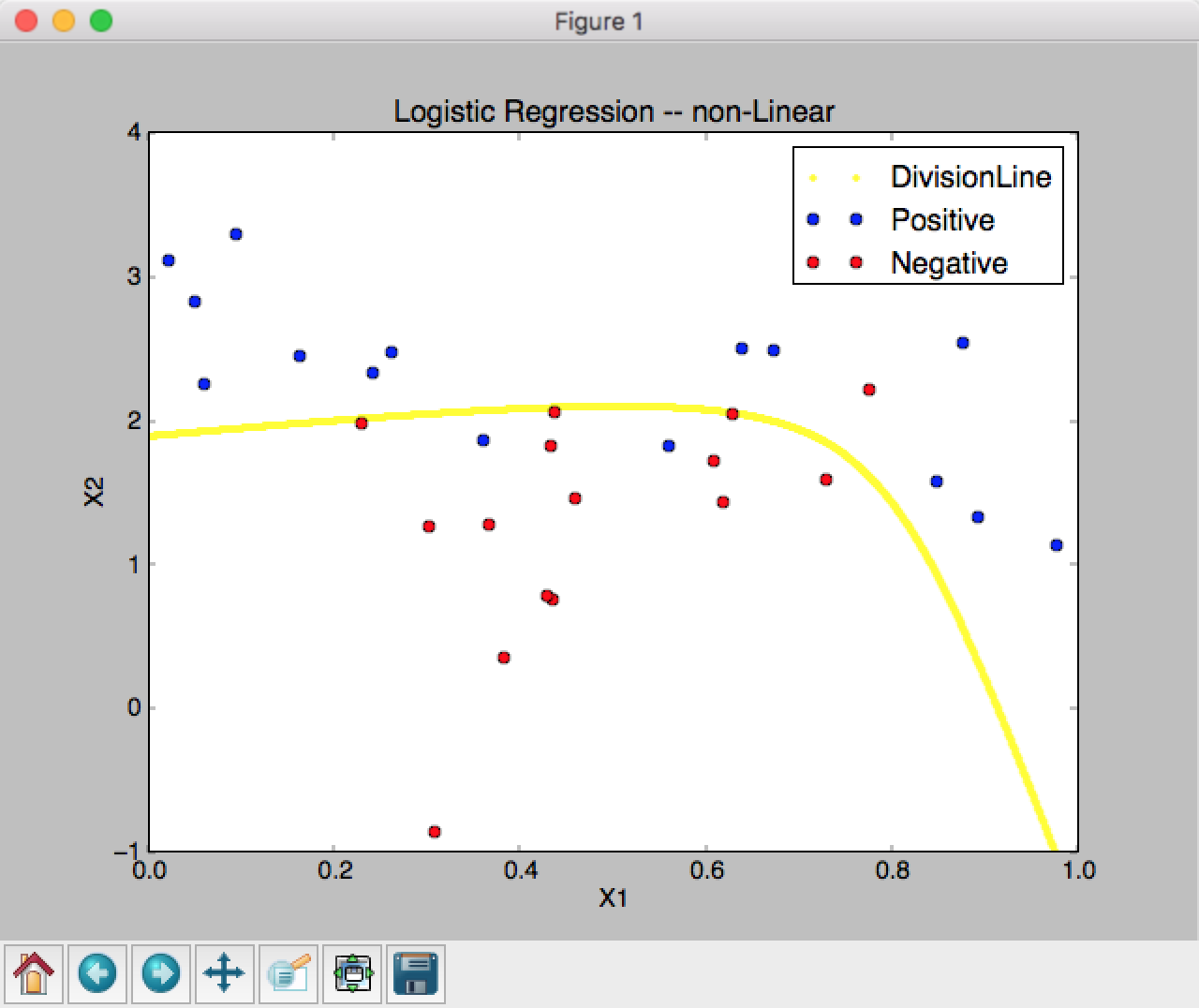


图6

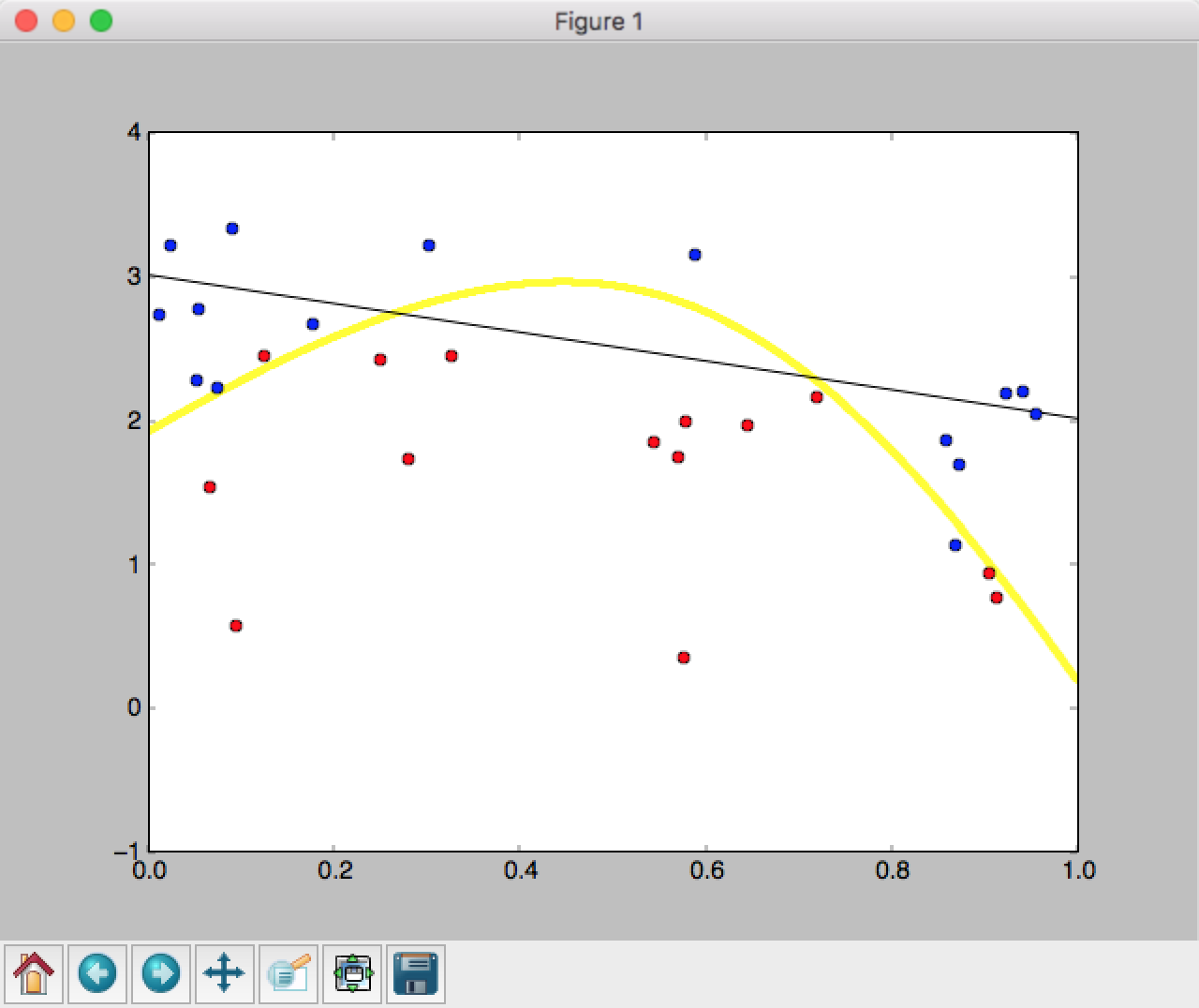


图7

1. 加惩罚项（lambda=0.01）

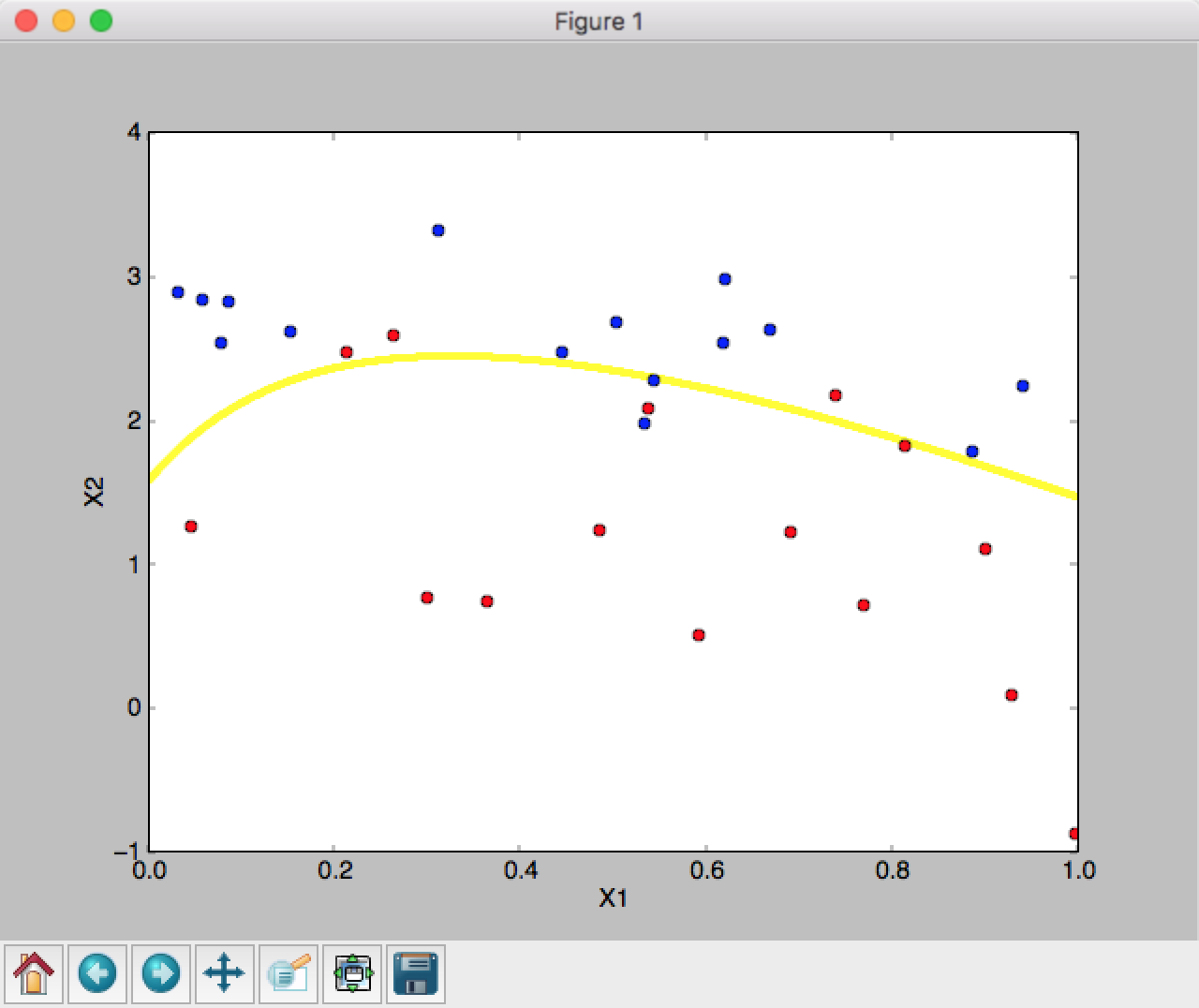


图8

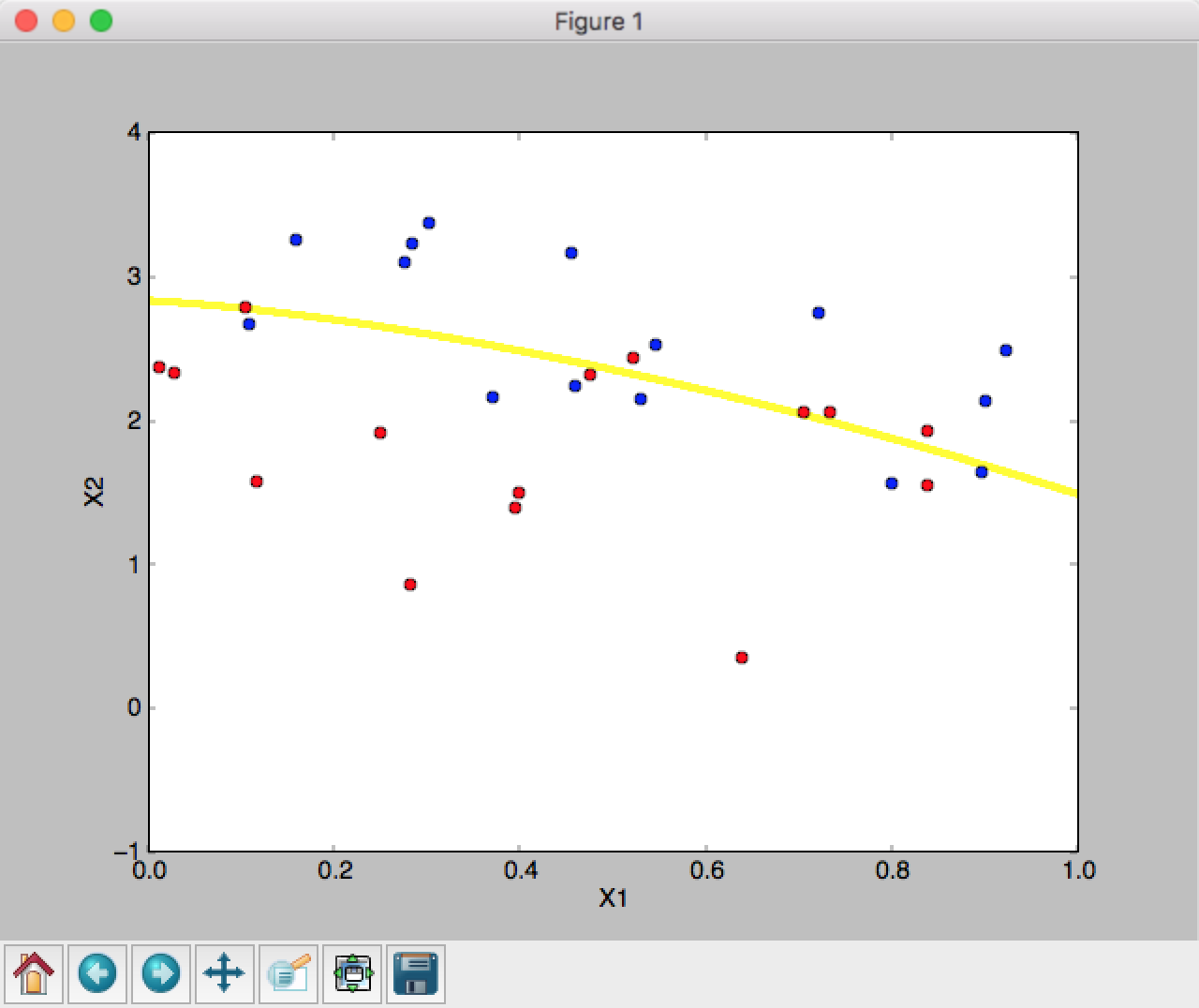


图9

在此我们可以分析一下。由于我这边展示的所用 函数最高为二次的，所以不能够很好地展示过拟合的情形，即下图所示的曲线。

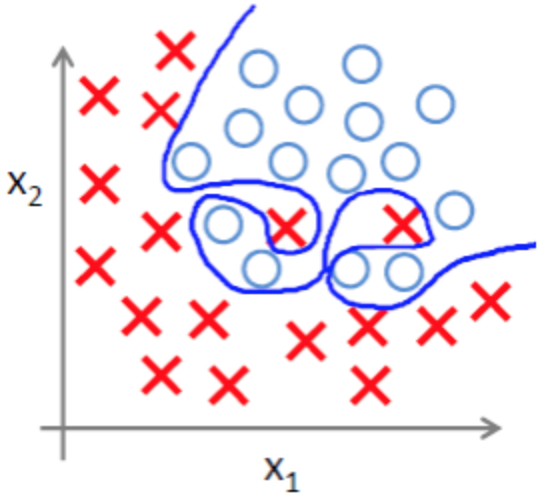


图10

解决过拟合的情况就是通过加惩罚项就能够很好地解决。虽然没有实现上图的情况，但是我们分析图6图7和图8图9可以发现，在加入惩罚项后，曲线会变得更平滑一些，不会出现在某处有较大的改变（有时甚至出现折线），这从一个侧面很好地体现出加入惩罚项的优点。

以上即实现了对线性不可分的数据样本进行分类。若增加样本属性数量，同样只需增加预测函数的权值数量即可。

**3.3 场景3**

场景3我实现的是对从UCI网站上下载下来的一组数据进行逻辑回归分类。

（<https://archive.ics.uci.edu/ml/datasets/Haberman's+Survival>）

具体数据以及数据说明请查看提交文件中的haberman.data和haberman.names。简单介绍下这组数据，共有四个属性，如下图

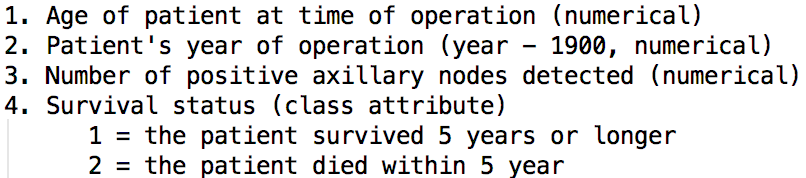


图11

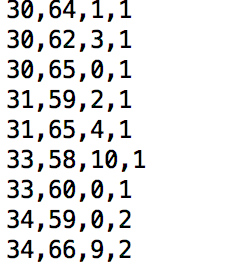


图12

我所实现的就是依据前三个属性进行预测某个病人的Survival status。数据集当中共有306个数据，我通过前200个数据进行模型的训练，最后再利用306个数据进行模型准确率的评估（由于对于高维度的线性不可分的数据集的分类我还没有掌握，所以我只用了线性模型进行分类。并且对于样本集是否线性可分不确定，所以不免会存在误差，因此需要对所有样本都进行验证）。

我通过随机生成初始权值，进行了几次训练。其中准确率最高的为76.47 %（具体测试结果请查看所提交的文件中，未加惩罚项：test开头的文件，加惩罚项：ptest开头的文件）。在此对测试结果进行一个说明，如下图。

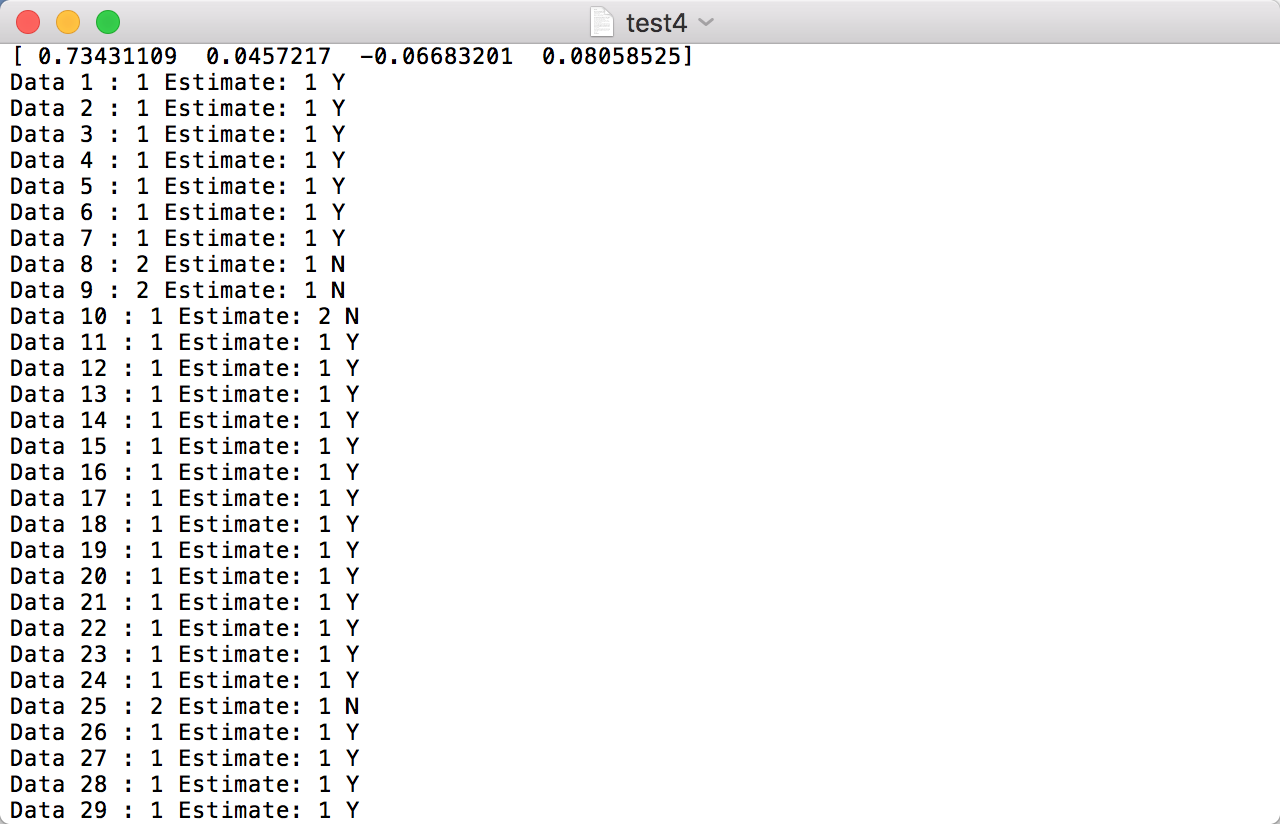


图13

文件中第一行显示了最后所得结果的权重向量。接下来每一行所代表的Data值为数据集当中的Survival status，Estimate表示预测结果，Y/N表示预测是否正确。在文件最后给出预测的准确率。



图14

对于是否加惩罚项的问题，其好处在前面已经介绍过，一是可以克服过拟合，二是能够一定地加快跌打的速度。

以上就是实验场景3的实验结果与分析。

**5 附录**

文件清单：

Linear.py：场景1代码

Linear\_p.py：场景1加惩罚项代码

nonLinear.py：场景2代码

nonLinear\_p.py：场景2加惩罚项代码

DataTest.py：场景3代码

DataTest\_p.py：场景3加惩罚项代码

haberman.data：测试数据集

haberman.names：数据集说名

场景3测试结果：

test1.txt

test2.txt

test3.txt

test4.txt

ptest1.txt

ptest2.txt