哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2020 年春季学期

计算学部《机器学习》课程

Lab 2 实验报告

姓名	刘璟烁
学号	
班号	1803104
电子邮件	
手机号码	

目录

一、实验目的

1.理解逻辑回归模型

2.掌握逻辑回归模型的参数估计算法

二、实验要求及实验环境

实验要求

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：可以手工生成两个分别类别数据（可以用高斯分布），验证算法；考察类条件分布不满足朴素贝叶斯。

实验环境

操作系统：MacOS 10.15.7
python 版本：Python 3.8.3

三、设计思想及算法实现

算法设计原理

1.样本生成算法

为了生成适用于二分类的数据集，可以分别生成两个对应不同类别的集合。在本次实验中，我们令每个类别的样本集中的所有点均有两个特征属性（x_1 及 x_2），

两个属性均满足高斯分布。根据题目要求的类条件分布满足朴素贝叶斯与否，两个属性之间的协方差应该分别置为零或非零值。

2.logistic 回归模型

logistic 回归模型用于分类任务，它的主要思想是利用给定样本特征向量 X 对样本所属的类别 Y 进行预测，即计算概率 P(Y|X)。基于朴素贝叶斯的思想，该模型假设样本的特征量中各个属性对于给定标签 T 满足类条件独立，并且各个属性遵从 Gaussian 分布，而标签 T 则满足 Bernoulli 分布。

模型推导过程如下：

首先，我们考虑样本的标签 Y 取 0 的概率，根据贝叶斯定理有：

$$P(Y=0|X)=\frac{P(Y=0)P(X|Y=0)}{P(Y=0)P(X|Y=0)+P(Y=1)P(X|Y=1)}$$

通过上下同除，上述公式可以化简为如下形式：

$$P(Y=0|X)=\frac{1}{1+exp(ln(\frac{P(Y=1)P(X|Y=1)}{P(Y=0)P(X|Y=0)}))}$$

根据我们前面提到的假设，不妨设 P(Y = 1)为π_1，并且 P(X|Y = i)可以写各属性单独取值的累乘形式，则原式可以写为：

$$P(Y=0|X)=\frac{1}{1+exp(ln(\frac{\pi_1}{1-\pi_1})+\sum_{i=1}^{m}(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i+\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}))}$$

这里我们用到了 X 的每一个特种属性均满足高斯分布 N(μ_{ik},σ_{ik}),即与特征属性对应的类别及维数有关。根据上式，我们可以将其中与特征属性相关的部分写为向量内积的形式，即：

$$P(Y=0|X)=\frac{1}{1+exp(\mathbf{w}^TX)}$$

则预测为另一类的概率为(一减上式并上下约分可得)：

$$P(Y=1|X)=\frac{1}{1+exp(-\mathbf{w}^T X)}$$

通过上述两式，我们可以仅仅根据关于特征向量 X 的线性方程判断样本所属类别：

$$ln(\frac{P(Y=1|X)}{P(Y=0|X)})=\mathbf{w}^T X$$

该式取值>0 时预测为 Y = 1，否则 Y = 0。

为了能够通过数据集训练出上述模型，我们需要给出其损失函数，由于对联合分布 P(X,Y)的计算条件不充足，模型用极大条件似然（MCLE）估计代替极大似然估计，似然函数可写作下式：

$$l(W)=\prod_i y_i^{t_i}(1-y_i)^{1-t_i}$$

其中 t_i 为第 i 个样本的标签，y_i 为预测该样本类别为 1 的概率，则对上式取对数并分解,取相反数可得损失函数：

$$E(W)=-\sum_{i=1}^{n}[t_i ln(P(y_i=1|X_i))+(1-t_i)ln(P(y_i=0|X_i))]$$

则求其关于 w 的偏导,这里我们直接为了方便用 numpy 实现，将偏导写为矩阵乘法的形式，其中 X 为(m,n)的矩阵，Y、T 均为(1,n)的行向量：

$$\frac{\partial E}{\partial \mathbf{w}}=X(Y-T)^T$$

为防止模式过拟合，可以对其进行正则化，这里采用老师所说的 MAP 代替 MCLE，即增加一个关于参数向量 **w** 的先验概率 P(**w**)，以将其正则化，则模型似然函数改写为：

$$l(w)=P(\mathbf{w})\prod_i y_i^{t_i}(1-y_i)^{1-t_i}$$

其中先验概率中 w 服从 Gaussian 分布。

算法实现

1.梯度下降法(无惩罚项)：

梯度下降法是一种求取能使得损失函数近似最小的参数的最优化算法，使用梯度下降法可以找到一个函数的局部极小值。我们知道某一点梯度的反方向是函数值下降最快的方向，通过设定一个步长，沿向函数上当前点对应梯度（或者是近似梯度）的反方向的规定步长距离点进行迭代搜索，也就是每次迭代为向量参数 w 减去一个步长和梯度的乘积。

利用前一部分给出的损失函数 E，我们可以得到 E 关于 **w** 的偏导：

$$\frac{\partial E}{\partial \mathbf{w}} = X(Y-T)^T$$

梯度下降法利用上面求出的偏导，通过下述方程对 **w** 进行迭代，更新 w 使损失函数收敛(其中α为学习率)：

$$\mathbf{w} = \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$$

简化版本的代码如下：

```
def logis_without_pun(X,W,T,num_item,alpha):
    for i in range(num_item):
        P_1 = 1/(1+np.exp(-1*np.dot(W.T,X))) #y=1 的概率
        A = P_1 - T
        dW = np.dot(X,A.T)
        W = W - alpha * dW
    return W
```

为解决过拟合的问题，我们可以为模式的损失函数添加正则项，与之对应的，梯度下降的迭代方程改写为：

$$\mathbf{w} = \mathbf{w} - \alpha\frac{\partial E}{\partial \mathbf{w}} + \alpha\lambda\mathbf{w}$$

上述等式右边第三项为我们添加的正则项，代码如下：

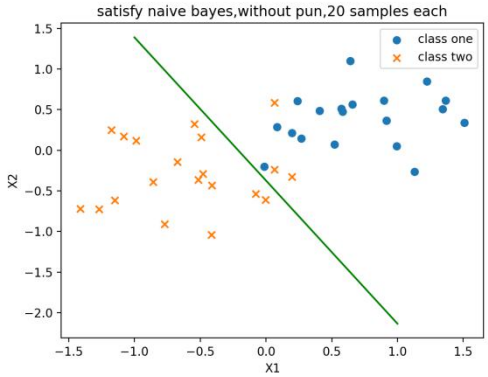
```
def logis_without_pun(X,W,T,num_item,alpha):
        for i in range(num_item):
         P_1 = 1/(1+np.exp(-1*np.dot(W.T,X))) #y=1 的概率
         A = P_1 - T
         dW = np.dot(X,A.T)
         W = W - alpha * dW + alpha * lambd * W
    return W
```

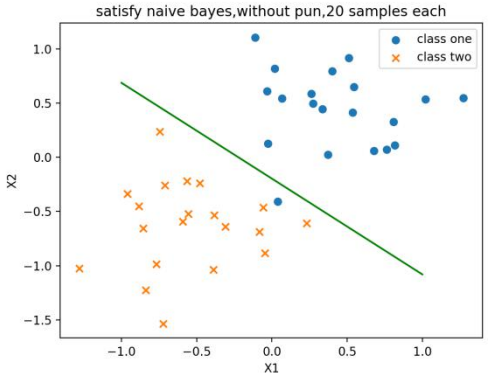
四、实验结果分析

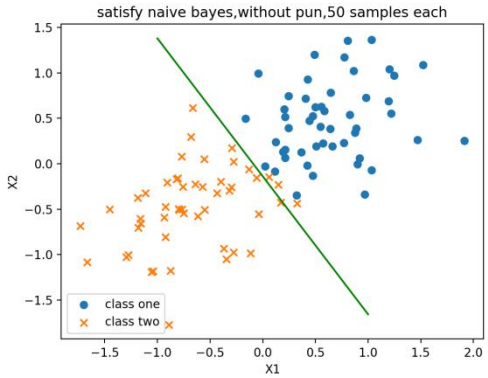
梯度下降法

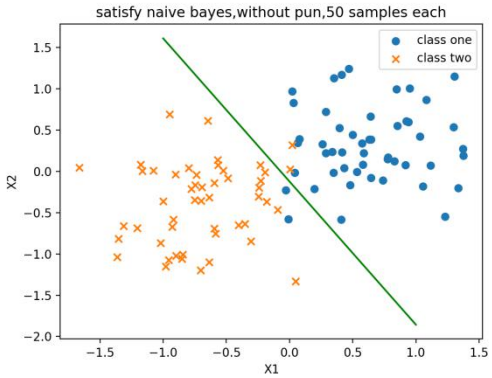
(1).无正则项、满足朴素贝叶斯假设

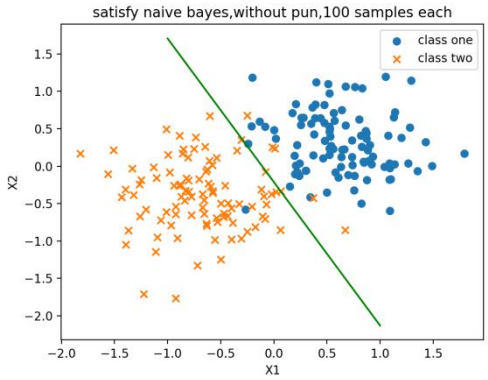
令两个类别的样本数量均分别取 20、50、100，并按照 30%的比例划分出测试集，训练出的模型分类效果如下：

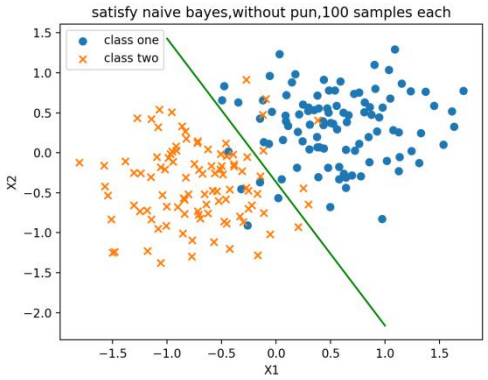












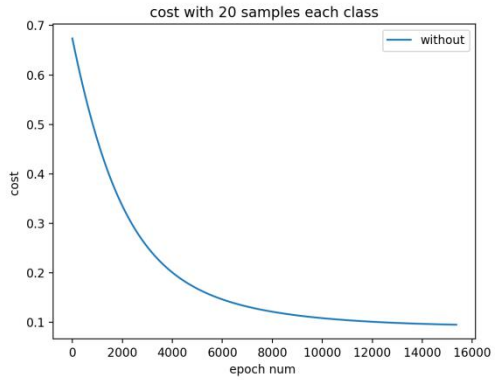
下面是其正确率：

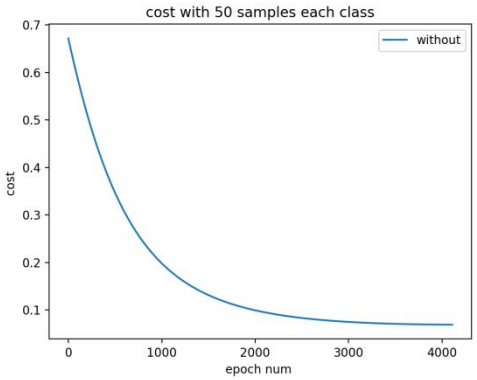
样本	训练 acc	测试 acc
20	0.961	0.917
20	0.892	0.913

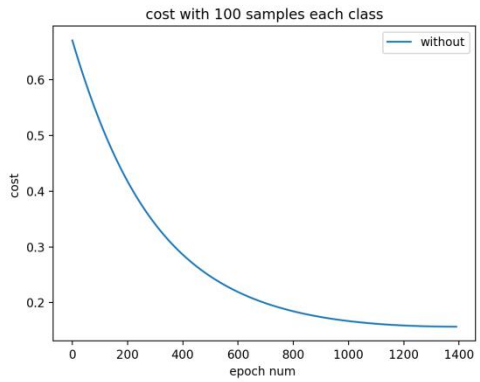
50	0.948	0.933
50	0.935	0.980
100	0.932	0.936
100	0.961	0.955

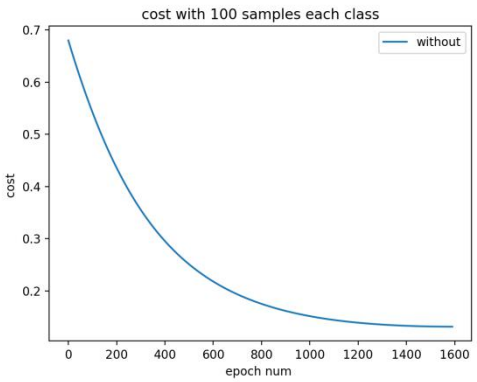
观察上述数据可知，不同样本数据规模下 logistic 回归模型均可较好的分类。但随着样本规模的增大，我们可以看出不同次训练模式的正确率的波动程度变小，并且测试集与训练集上准确率的差距在不断减小。

同时，给定迭代变化量阈值的情况下，训练停止的迭代次数随着样本数量的增加而减少：



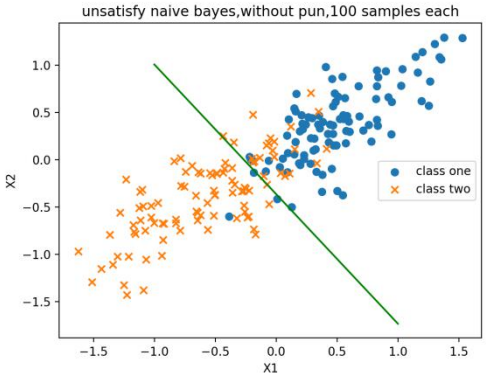


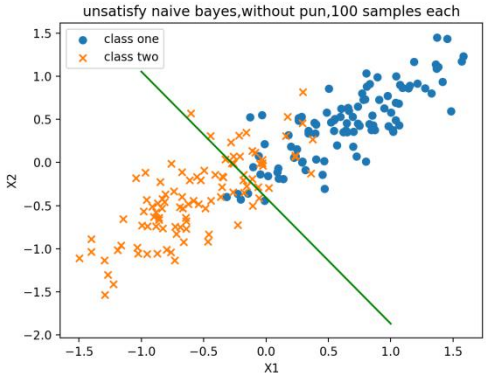




(2).无正则项、不满足朴素贝叶斯假设

令两个特征属性间的血

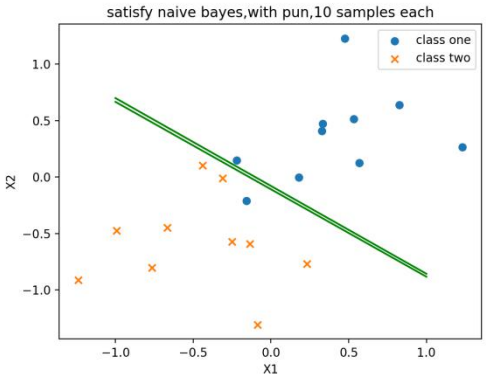


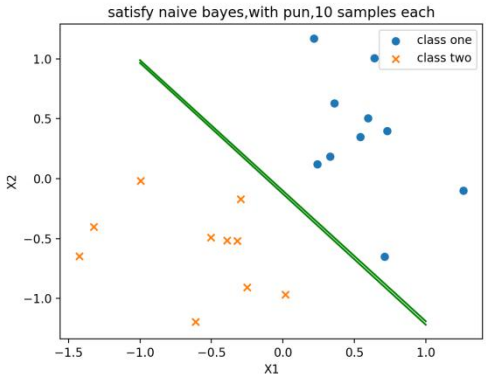


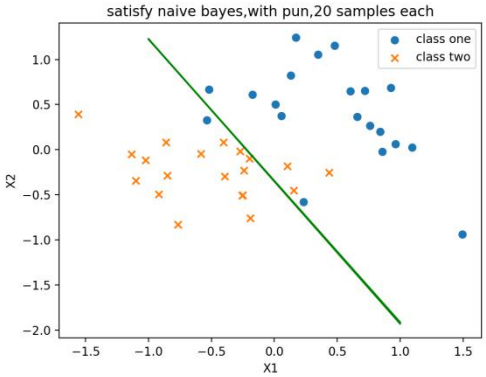
两次训练，training set 的正确率分别为 0.9、0.89，测试集正确率分别为 0.88,0.89，相比满足贝叶斯假设的同规模的训练情况误差有所增大。

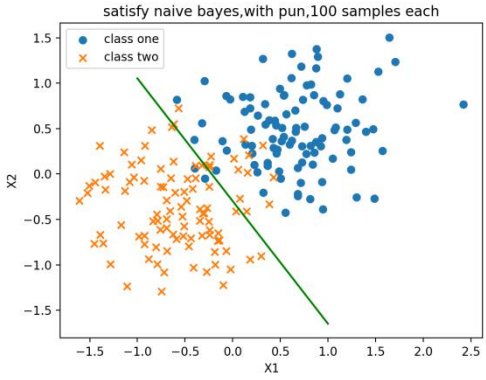
(3).有正则项、不满足朴素贝叶斯假设

对于类规模分别为 10、20、100 的情况，对比添加正则项与未添加正则项所产生的决策边界：









具体的 accuracy 如下，

样本量为 10 时：

```
the accuracy of the trainning set without punishment is :0.8571428571428572
the accuracy of the test set without punishment is:1.0
the accuracy of the trainning set with punishment is :0.8571428571428572
the accuracy of the test set with punishment is :0.8571428571428572
```

样本量为 20 时：

```
the accuracy of the trainning set without punishment is :0.8928571428571429
the accuracy of the test set without punishment is:0.8333333333333334
the accuracy of the trainning set with punishment is :0.8928571428571429
the accuracy of the test set with punishment is :0.8928571428571429
```

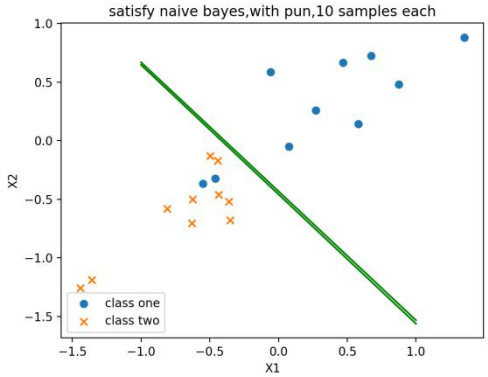
样本量为 100 时：

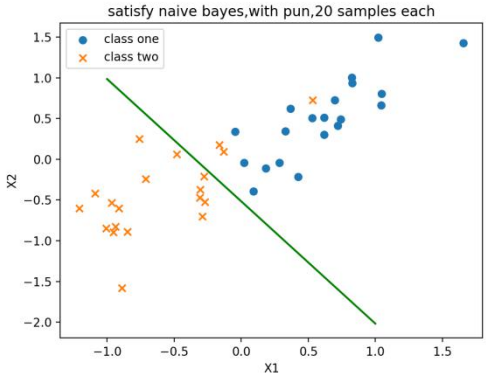
```
the accuracy of the trainning set without punishment is :0.9285714285714286
the accuracy of the test set without punishment is:0.9333333333333333
the accuracy of the trainning set with punishment is :0.9285714285714286
the accuracy of the test set with punishment is :0.9285714285714286
```

可以观察得到，在数据量较小时添加正则项会对训练出的模式产生较小影响，数据量略微增大后两者差别几乎可以忽略。

(4).有正则项、不满足朴素贝叶斯假设

类似上述内容，在不满足朴素贝叶斯的条件下，准确率相比同规模满足该条件时有所下降，但正则项添加后对原模式的影响不大。





样本量为 100 时，两次训练的准确率：

```
the accuracy of the trainning set without punishment is :0.9142857142857143
the accuracy of the test set without punishment is:0.8333333333333334
the accuracy of the trainning set with punishment is :0.9142857142857143
the accuracy of the test set with punishment is :0.9142857142857143
```

```
the accuracy of the trainning set without punishment is :0.8928571428571429
the accuracy of the test set without punishment is:0.8166666666666667
the accuracy of the trainning set with punishment is :0.8928571428571429
the accuracy of the test set with punishment is :0.8928571428571429
```

对比 11 页满足朴素贝叶斯的情况下准确率有所下降。

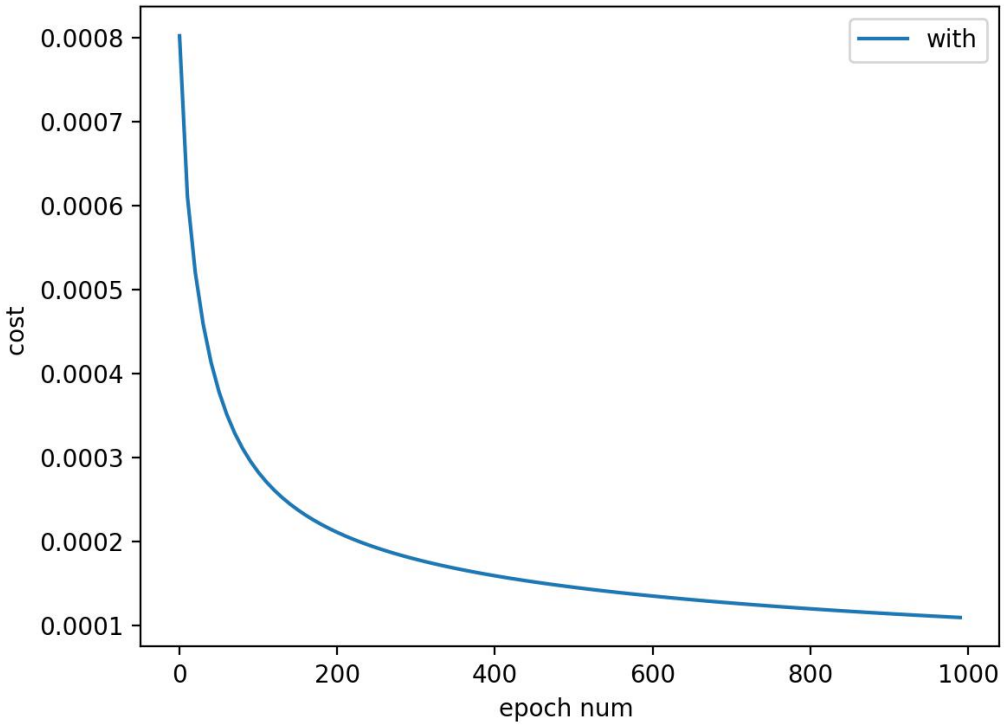
(5).引入 UCI 数据进行测试

1.在本次试验中首先将选用钞票数据集 Banknote Dataset，集合样本数量为 1372 个，每个样本含有四个特征属性，分别为图像经小波变换后的方差(variance)(连续值)、图像经小波变换后的偏态(skewness)(连续值)、图像经小波变换后的峰度(curtosis)(连续值)、图像的熵(entropy)(连续值)。

在 lambda 取 0.1 的情况下，训练出模型的准确率为：

```
the accuracy of the trainning set with punishment is :0.982
the accuracy of the test set with punishment is :0.971666666666
```

损失函数值的变化为：

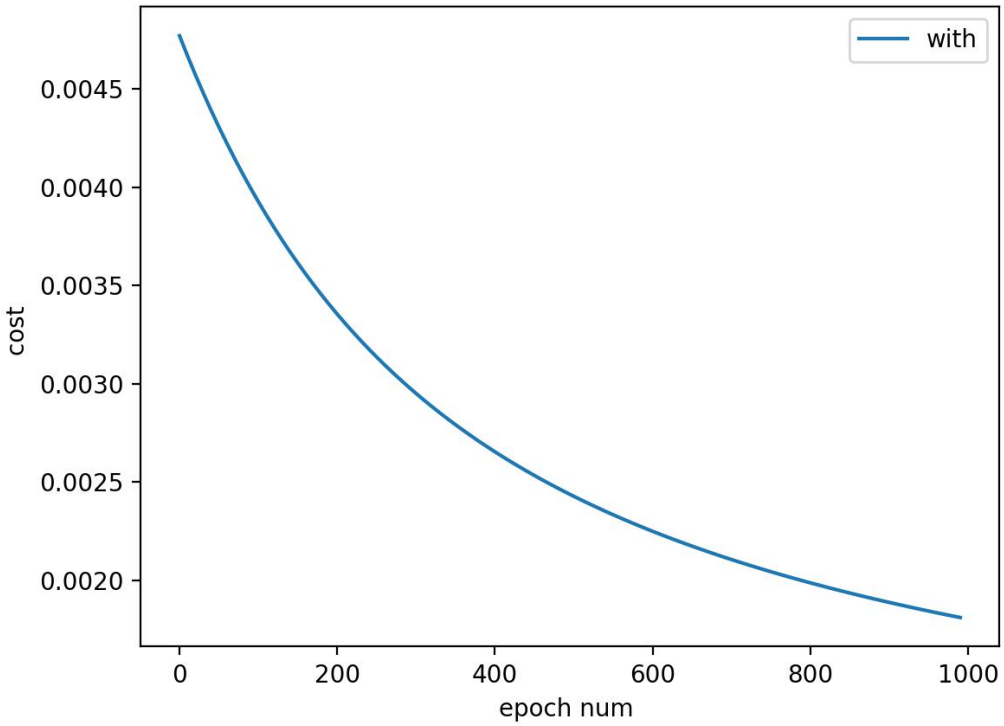


2.同时还使用了 seeds 数据集，该数据集测量属于三种不同小麦品种的籽粒的几何特性。使用软 X 射线技术和 GRAINS 软件包构造了所有七个实值属性。检验的组包括属于三种不同小麦品种的籽粒：卡马，罗莎和加拿大小麦，这里我们将选择前两种进行分类。每种 70 种元素，均随机选择用于实验。使用软 X 射线技术检测到内部内核结构的高质量可视化。

模型训练结果如下：

```
the accuracy of the trainning set with punishment is :0.9714285714285714
the accuracy of the test set with punishment is :0.95
```

迭代过程损失如下：



五、结论

根据上述实验结果以及算法原理的分析，我们可以得出以下结论：

1. 根据决策函数的图像可以看出，正则项的有无对模型的影响不大，特别是训练样本集较大时，差别可忽略不计。

2. 当类条件分布满足朴素贝叶斯时，其他条件相同的情况下训练结果略好一些。

六、参考资料

吴恩达：机器学习

Christopher Bishop. Pattern Recognition and Machine Learning.

周志华：机器学习

七、源代码

main.py:

该部分为 client 端，可以通过调整“控制参数区”中的阶数、样本量来更改模型训练的参量。

```
import sklearn.model_selection
import numpy as np
import matplotlib.pyplot as plt
import gene_Samp
import logisticFunc


#对一个样本集计算预测准确率


def gene_W(dim):
    W = 0.1*np.random.rand(dim).reshape(-1,1)
    return W
def calc_Accu_with_logistic(X,W,T):
    Y = 1 / (1 + np.exp(-1*np.dot(W.T,X)))
    Y = (Y>=0.5).astype(int)
    res = Y^T #做异或运算  找出预测与原标记相同的样本
```

```
    return 1.0 - np.sum(res)/T.shape[1] #计算出比例


#画出训练得到的决策边界
def draw_decision_boundary(W):
    X1 = np.linspace(-1,1,100)
    X2 = (-1-W[1][0]*X1)/W[2][0]
    plt.plot(X1,X2,c = 'g')
    # plt.show()
    return X1,X2


def banknote_data_load():
    data = open('data_banknote_authentication.txt').readlines()
    data_set = []
    for data_line in data :
        this_line = data_line.strip().split(',')
        this_line_float = []
        for str in this_line:
            fl = float(str)
            this_line_float.append(fl)
        data_set.append(this_line_float)
    data_set = np.array(data_set)
    X = data_set[:, 0:4]
```

```
    Y_raw = data_set[:, 4]
    Y = []
    for i in Y_raw:
        Y.append(int(i))
    Y = np.array([Y]).T
    return X,Y


#客户端




# X,Y = banknote_data_load()
X1, Y1,X2,Y2 = gene_Samp.gene_gauss_Samples(100, 100 ,True) #生成
两类样本
# X1, Y1,X2,Y2 = gene_Samp.gene_gauss_Samples(100, 100 ,False) #
生成两类样本



X = np.c_[X1,X2].T #200,2 #合成总集合
Y = np.c_[Y1,Y2].T #200,1


#将总集合随机划分为训练集和测试集
```

```
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(X,
Y, test_size = 0.3)
#生成样本对应的特征矩阵  第一行为 1


x_train = np.array(x_train).T #(m,n)
train_ones = np.array([np.ones(x_train.shape[1])])
x_train = np.r_[train_ones,x_train]
y_train = y_train.T #(1,n)
x_test = np.array(x_test).T
test_ones = np.array([np.ones(x_test.shape[1])])
x_test = np.r_[test_ones,x_test]
y_test = y_test.T



#随机生成函数
W = gene_W(X.shape[1]+1)
W1,cost1,list1 =
logisticFunc.logis_without_pun(x_train,W,y_train,50000,0.0001)
W2,cost2,list2 =
logisticFunc.logis_with_pun(x_train,W,y_train,400,0.001,0.1)
print("the accuracy of the trainning set without punishment is :"+
str(calc_Accu_with_logistic(x_train,W1,y_train)))
```

```
print("the accuracy of the test set without punishment
is:"+str(calc_Accu_with_logistic(x_test,W1,y_test) ))
print("the accuracy of the trainning set with punishment is :"+
str(calc_Accu_with_logistic(x_train,W2,y_train)))
print("the accuracy of the test set with punishment is :"+
str(calc_Accu_with_logistic(x_test,W2,y_test)))


#画出决策边界
plt.figure(1)
draw_decision_boundary(W1)
draw_decision_boundary(W2)
plt.scatter(X1[0,:],X1[1,:],marker = 'o',label = 'class one') #第一类点
plt.scatter(X2[0,:],X2[1,:],marker = 'x',label = 'class two') #第二类点
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('satisfy naive bayes,with pun,20 samples each')
plt.legend()
plt.figure(2)
plt.plot(list1,cost1,label = 'without')
plt.plot(list2,cost2 ,label = 'with')
plt.xlabel('epoch num ')
plt.ylabel('cost ')
```

```
plt.title('cost with 10 samples each class')
plt.legend()
plt.show()
```

logisticFunc.py

```
import numpy as np
import CostFunc


def logis_without_pun(X,W,T,num_item,alpha):
    #    W 为 n+1 维包含 w0 的向量，  X 为 n+1 维、第一维为 1 的列向量组成的矩阵
    cost = []
    list = []
    for i in range(num_item):
        P_1 = 1/(1+np.exp(-1*np.dot(W.T,X))) #y=1 的概率
        last_cost = -1 / T.shape[1] * CostFunc.cost_func(X, W, T)
        A = P_1 - T
        dW = 1/T.shape[1]*np.dot(X,A.T)
        W = W - alpha * dW
        this_cost = -1/T.shape[1] * CostFunc.cost_func(X,W,T)
        # if last_cost - this_cost < 0.000001:
        #     break
```

```
        if last_cost < this_cost:
            alpha = alpha/2
        if i%10 == 0:
            cost.append( this_cost )
            list.append(i)
    return W,cost,list


def logis_with_pun(X,W,T,num_item,alpha,lambd):
    cost = []
    list = []
    for i in range(num_item):
        P_1 = 1 / (1 + np.exp(-1 * np.dot(W.T, X)))   # y=1 的概率
        last_cost = -1 / T.shape[1] * CostFunc.cost_func(X, W, T)
        A = P_1 - T
        dW = 1/T.shape[1]*np.dot(X, A.T)
        W = W - alpha * dW + alpha * lambd * W #加入惩罚项
        this_cost = -1 / T.shape[1] * CostFunc.cost_func(X, W, T)
        # if last_cost - this_cost < 0.00001:
        #     break
        if last_cost < this_cost:
            alpha = alpha/2
        if i % 10 == 0:
```

```
            cost.append(this_cost)
            list.append(i)
    return W, cost, list
```

gene_Samp.py

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.datasets import make_gaussian_quantiles
import matplotlib.pyplot as plt


#确定两个类的样本数量 num1, num2,并告知属性类条件分布是否满足朴素贝叶斯
def gene_gauss_Samples(num_A,num_B,satisfy_naive):

    #指定分布中两个类的均值，并控制其协方差矩阵
    mean_A = [0.6,0.4]
    mean_B = [-0.6,-0.4]
    cov_AB = 0.15
    if satisfy_naive: #  直接调用 sklearn 中的方法生成高斯分布
        X1 = np.random.multivariate_normal(mean=mean_A, cov=[[0.2, cov_AB], [cov_AB, 0.2]], size=num_A).T
        X2 = np.random.multivariate_normal(mean=mean_B, cov=[[0.2,
```

```
cov_AB], [cov_AB, 0.2]], size=num_B).T


            # X1,Y1 = make_gaussian_quantiles(mean = [1,3],n_samples =
num1,n_classes=1,n_features= 2,cov = 1.1)
            # X2,Y2 = make_gaussian_quantiles(mean =
[5,6],n_samples=num2,n_classes=1,n_features=2,cov = 1.2)
            # Y2[:] = 1
        else: #由于 sklearn 中上述方法默认协方差矩阵为对角阵，我们该用
numpy 中的方法
            X1 = np.random.multivariate_normal(mean=mean_A, cov=[[0.2,
0], [0, 0.2]], size=num_A).T
            X2 = np.random.multivariate_normal(mean=mean_B, cov=[[0.2,
0], [0, 0.2]], size=num_B).T
        Y1 = np.array([[1 for i in range(num_A)]])
        Y2 = np.array([[0 for i in range(num_B)]])


        #X1、X2 均为(2,n)矩阵，Y1、Y2 为(1,n)矩阵
        return X1,Y1,X2,Y2
```

CostFunc.py

```
import numpy as np
```

```
def cost_func(X,W,T):
    # res1 = np.dot(np.log(1/(1+np.exp(-1*np.dot(W.T,X)))),T.T)
    # res2 = np.dot(np.log(1-1/(1+np.exp(-1*np.dot(W.T,X)))),1-T.T)
    # res = -1*res1 - res2
    res_f = np.dot(np.dot(W.T,X),T.T)
    res_l = np.sum(np.log(1+np.exp(np.dot(W.T,X))))
    res = res_f - res_l
    return   np.asscalar(res) #代价函数
```