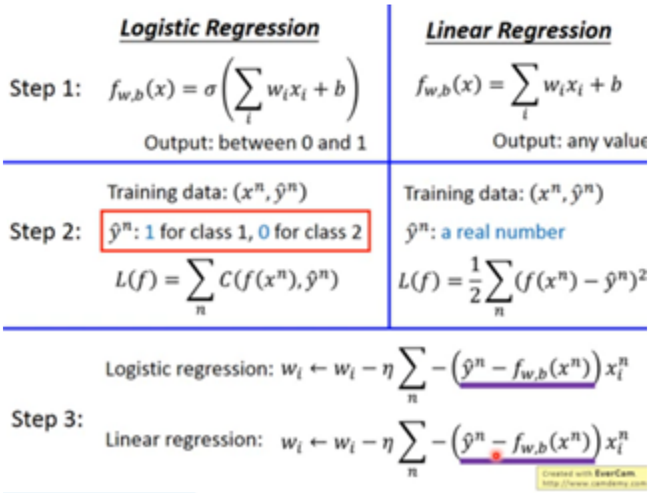
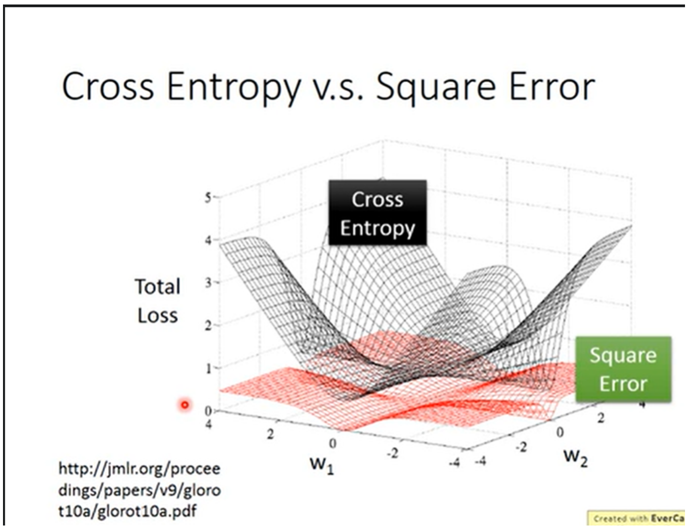
机器学习实验---logistic regression

一、实验目的

1. 理解 Logistic 回归算法原理，能实现 Logistic 回归算法；
2. 理解 Logistic 回归和线性回归的区别，损失函数的不同；





https://www.bilibili.com/video/BV13x411v7US?p=11

3. 掌握梯度下降法更新权重；
4. 能够熟练使用归一化方法，并深刻理解归一化的意义；
5. 针对特定应用场景及数据，能构建 Logistic 回归模型并进行预测。

二、实验内容

1. 利用 sklearn 的 Breast_cancer 数据集，设计一个基于 Logistic 回归的二分类模型。以 80%的训练集，20%的测试集。观察训练集的损失和精度变化，以及测试集上的精度变化，并绘制出来。（可以参考样例，需补充标 XXX 的部分）

2*. Kaggle 的 ADULT 数据，设计一个基于 Logistic 回归的二分类模型。以 80%的训练集，20%的测试集。

   https://archive.ics.uci.edu/ml/datasets/Adult

三、实验报告要求

1. 按实验内容撰写实验过程；
2. 报告中涉及到的代码，每一个模块需要有详细的注释；
3. 绘制出对数据用归一化和不用归一化的结果，以及用不同的归一化的结果。

四、参考样例

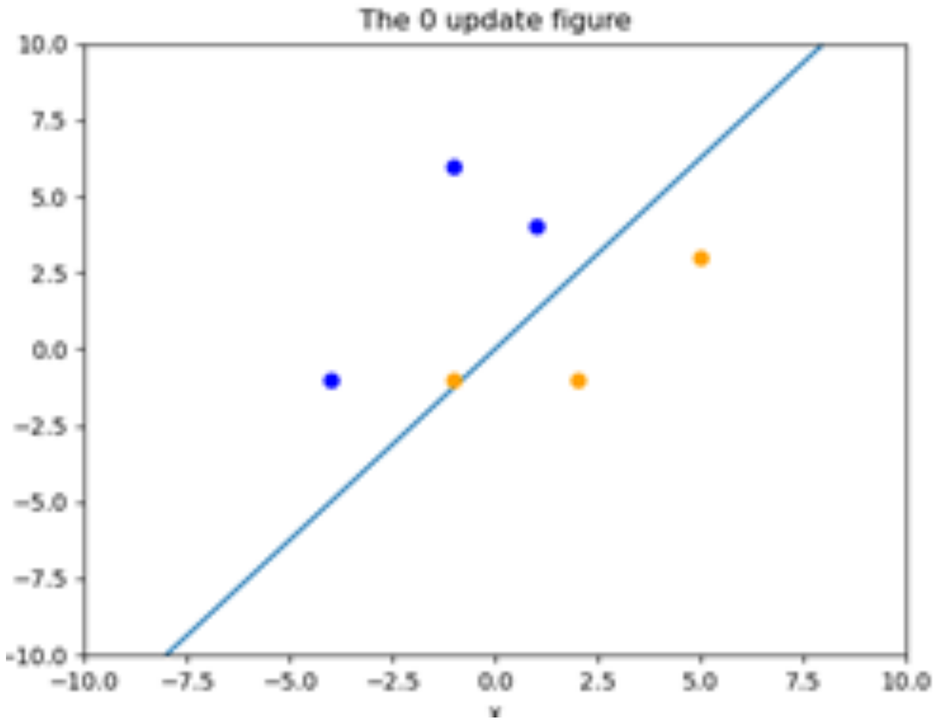
```
# -*- coding: utf-8 -*-
import csv
import numpy as np
import  pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets

cancers = datasets.load_breast_cancer()
train_X = cancers['data'][:450]
train_Y = cancers['target'][:450]
test_X = cancers['data'][450:]
test_Y = cancers['target'][450:]


class LR:
    def  __init__(self, data, data_test):
        self.m = data.shape[0]
        self.cols = data.shape[1] + 1
        self.w = np.zeros(self.cols)
        self.b = np.ones(self.m).reshape(self.m, 1)
        self.lr = 0.001
        self.train_X = np.hstack([self.b,data])

        self.m_test = data_test.shape[0]
        self.b_test = np.ones(self.m_test).reshape(self.m_test, 1)
        self.test_X = np.hstack([self.b_test,data_test])

    def sigmoid(self,x):
        res = 1/(1 + np.exp(-x))
        return np.clip(res, 1e-8, (1-(1e-8)))


    def stop_stratege(self,loss,loss_update,threshold):
        return  loss-loss_update <threshold

    def Logistic_Regression(self, X, Y, X_test, Y_test, epochs):  # 
Logistic
        i = 0
        loss_record = []
        acc_record = []
```

```
        acc_test_record = []
        for i in range(epochs):

            ## predict
            XXX

            ## compute loss
            ### -sum((Y * ln(f(x))+(1-Y) * ln(1-f(x))))
            loss = XXX
            print(loss)
            loss_record.append(loss)

            ## compute gradiant
            grad = XXX

            ## update weight
            self.w = XXX

            ## compute acc in training
            acc = XXX
            acc_record.append(acc)

            ## print every 20 iteration
            if i % 500 == 1 and i>500:
                self.visulization(i, X, Y)

            ## test each epoch
            acc_test = self.test(X_test, Y_test)
            acc_test_record.append(acc_test)

        return  loss_record, acc_record, acc_test_record



    def train(self, X, Y, X_test, Y_test, epochs):
        loss_record, acc_record, acc_test_record =
self.Logistic_Regression(X, Y, X_test, Y_test, epochs)

        return loss_record, acc_record, acc_test_record

    def test(self, X, Y):
        z = np.dot(X, self.w)
        y = self.sigmoid(z)
        acc_test = XXX
        return acc_test
```

```python
    def visulization(self,which_step,X_,Y_):
        x = np.linspace(-10,10)
        y = -(self.w[0] +self.w[1]*x)/self.w[2]
        plt.plot(x,y)

        index_0 = np.where(Y_==0)[0]
        index_1 = np.where(Y_==1)[0]

        plt.plot(X_[index_0, 1], X_[index_0, 2], 'bo', color='blue',
label='0')
        plt.plot(X_[index_1, 1], X_[index_1, 2], 'bo', color='orange',
label='0')
        plt.xlabel('x')
        plt.ylabel('y')
        plt.title('The '+str(which_step)+' update figure')
        plt.xlim(-10,10)
        plt.ylim(-10,10)
        plt.show()

def AllNorm(X):

    minVals=np.min(X)
    maxVals=np.max(X)

    ranges=maxVals-minVals
    normDataSet=np.zeros(np.shape(X))
    m=X.shape[0]
    normDataSet=X-np.tile(minVals,(m,1))#在行方向重复 minVals m 次和列方向
上重复 minVals 1 次
    normDataSet=normDataSet/np.tile(ranges,(m,1))
    return normDataSet

def ChannalNorm(X):
    XXX
    return normDataSet

def plot_history(loss_record, acc_record, acc_test_record, epochs):
        plt.figure(figsize=(10,5))
        plt.subplot(2,2,1)
        plt.xlabel('Epoch')
        plt.ylabel('loss')
        plt.plot(range(epochs), loss_record, label='loss')
        plt.legend()
```

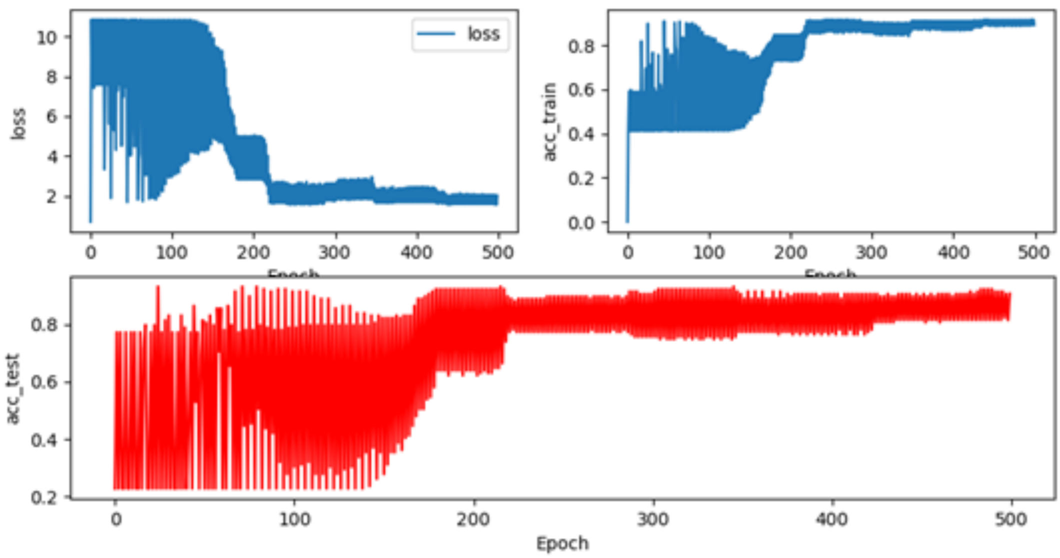
```python
        plt.subplot(2,2,2)
        plt.xlabel('Epoch')
        plt.ylabel('acc_train')
        plt.plot(range(epochs), acc_record, label='acc_train')
        plt.subplot(2,1,2)
        plt.xlabel('Epoch')
        plt.ylabel('acc_test')
        plt.plot(range(epochs), acc_test_record, label='acc_test', color
= 'red')

        #plt.plot(hist['epoch'], hist['acc'], label = 'acc',color =
'red')
        plt.legend()

if __name__ == "__main__":

    '''
    data = [[-1,-1],[2,-1],[5,3],[-1,6],[-4,-1],[1,4]]
    label = [1,1,1,0,0,0]
    train_X = np.array(data)
    train_Y = np.array(label)
    '''
    #train_X = AllNorm(train_X)
    #test_X = AllNorm(test_X)
    Logist = LR(train_X, test_X)   # 实例化

    epochs = 500

    loss_record, acc_record, acc_test_record =
Logist.train(Logist.train_X, train_Y, Logist.test_X, test_Y, epochs)

    plot_history(loss_record, acc_record, acc_test_record, epochs)
```

五、运行结果

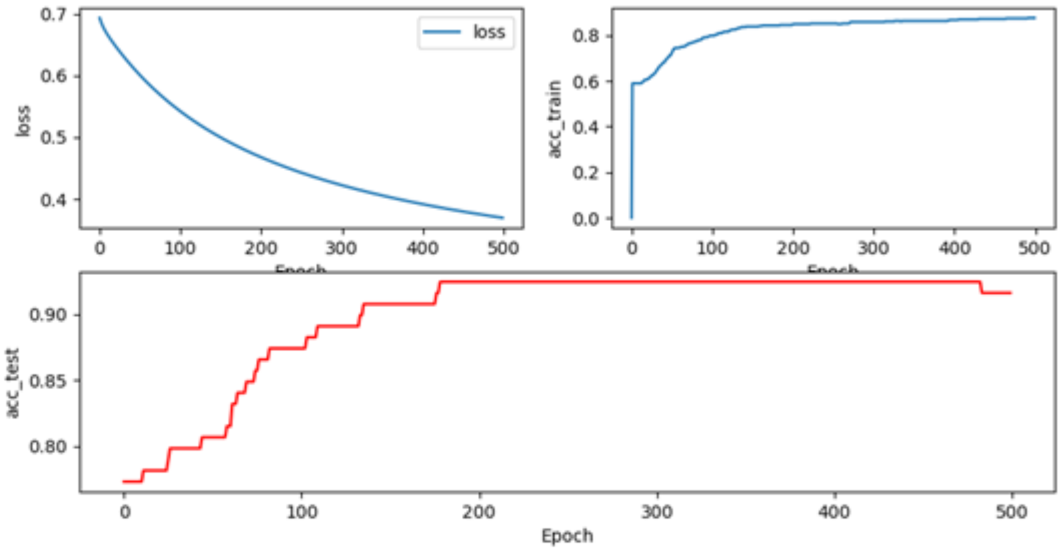


归一化对实验的影响

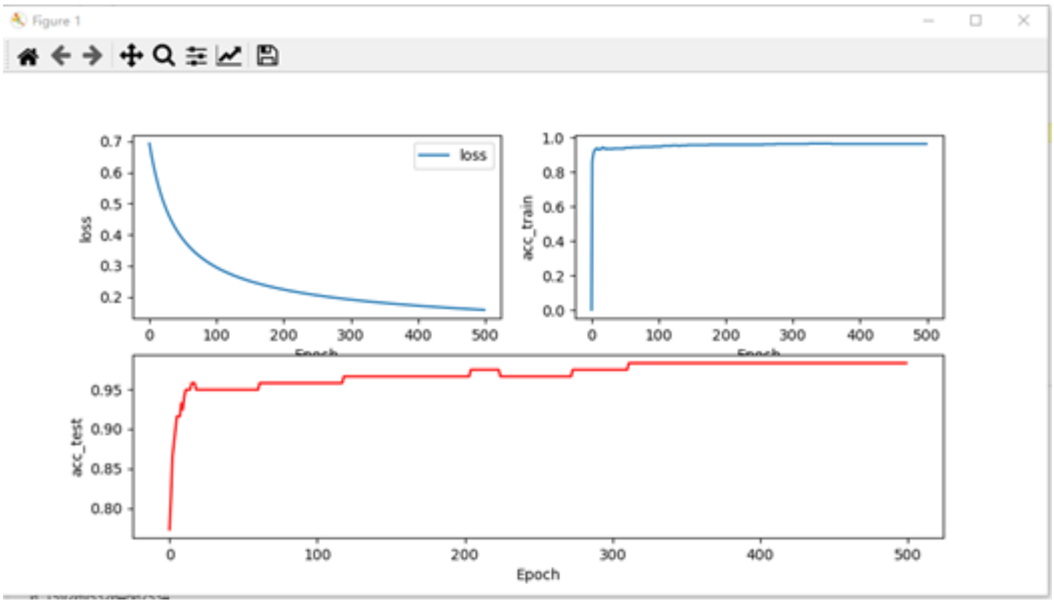
NoNorm:



AllNorm:



ChannalNorm：



六、实验小结

本次实验是理解 logistic 回归算法的原理并实现，logistic 回归属于分类模型，采用极大似然估计作为优化目标，形式上和交叉熵一致，并且可以使用梯度下降法或牛顿法作为优化算法。由于其模型中以非线性函数 sigmoid 作为激活函数，因此属于非线性分类器的一种。在进行数据处理的时候，往往需要对数据进行归一化，否则会出现梯度方向不稳定，导致损失函数波动剧烈，收敛性变差。