天津大学

机器学习实验报告

题目：机器学习实验---logistic regression

学　　院　智能与计算学部

专　　业　人工智能

年　　级　**2021**

姓　　名　**666**

学　　号　直到今天还是最有用的算法

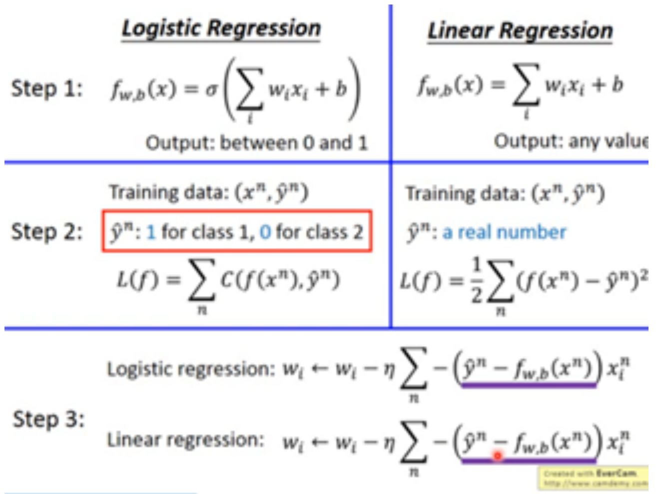
2023 年 3 月 6 日

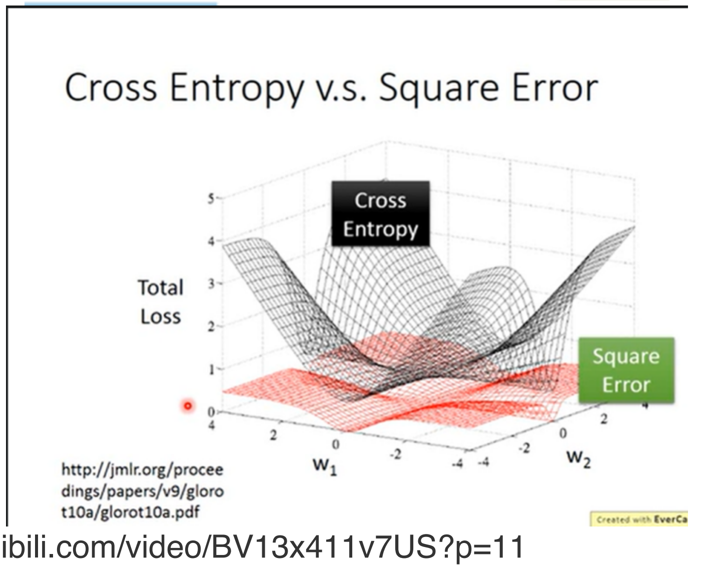
机器学习实验---logistic regression

一、实验目的

理解 Logistic 回归算法原理，能实现 Logistic 回归算法；

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

掌握梯度下降法更新权重；

能够熟练使用归一化方法，并深刻理解归一化的意义；

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

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

按实验内容撰写实验过程；

报告中涉及到的代码，每一个模块需要有详细的注释；
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、实验记录

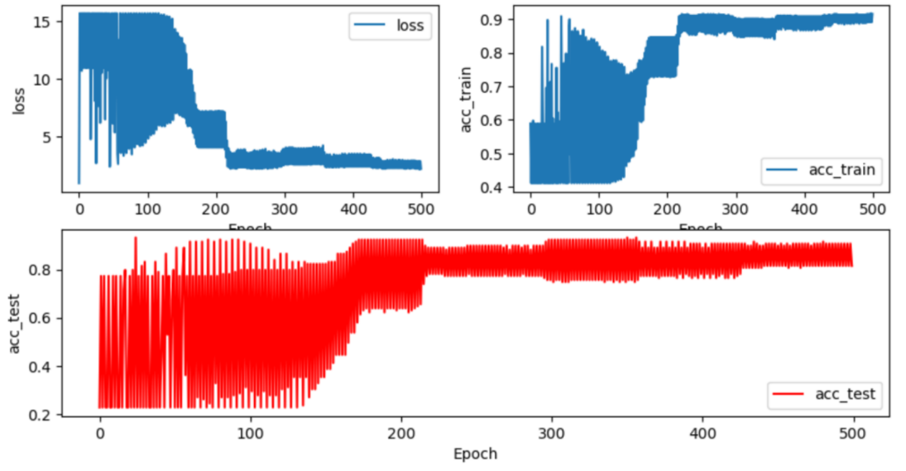
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer

cancers = load_breast_cancer()
train_X = cancers['data'][:450]
train_Y = cancers['target'][:450]
# 以 80%的训练集，20%的测试集
test_X = cancers['data'][450:]
test_Y = cancers['target'][450:]


class LR:
    def __init__(self, data, data_test):
        # data = train_x
        # data_test = test_x
        self.m = data.shape[0]   # 读取行数 450
        self.cols = data.shape[1] + 1   # 读取列数 31
        self.w = np.zeros(self.cols)   # 给每个数据的每一列设置一个权值，1 * 31

        self.b = np.ones(self.m).reshape(self.m, 1)   # 偏置值，450 * 1
        self.lr = 0.5   # 学习率
        self.train_X = np.hstack([self.b, data])   # 水平堆叠，450 * 31

        # 测试数据，我看都不看一眼的
        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_strategy(self, loss, loss_update, threshold):
        return np.abs(loss - loss_update) < threshold

    def Logistic_Regression(self, X, Y, X_test, Y_test, epochs):   # Logistic
        # X = self.train.X | 450 * 31
        # Y = train.Y | 450 * 1
        # self.w | 1 * 31
        # epoch 迭代次数
        i = 0
        loss_record = []   # 损失率
        acc_record = []   # 准确率
        acc_test_record = []   # 测试准确率
        for i in range(epochs):

            ## predict
            # 预测函数
            Sx = np.dot(X, self.w.T)   # 450 * 1
            Hx = self.sigmoid(Sx)   # 激活函数计算得到预测函数

            ## compute loss
            Cost = -(Y * np.log2(Hx)) - (1 - Y) * np.log2(1 - Hx)   # 450 * 1
            loss = sum(Cost) / self.m   # 损失函数 J
            ### -sum((Y * ln(f(x))+(1-Y) * ln(1-f(x))))
            #           if Y[i] == 1 :
            #               Cost = -np.log(Hx)
            #           else:
            #               Cost = -np.log(1-Hx)

            # print("the loss in " + str(i) + " is : ", loss)
            loss_record.append(loss)

            ## compute gradiant
            Hx_y_xi = np.dot((Hx - Y).T, X)   # 1 * 450 X 450 * 31
            grad = Hx_y_xi / self.m   # 1 * 31

            ## update weight
```

```
            self.w = self.w - self.lr * grad    # 1 * 31

            ## compute acc in training
            predict_Y = [1 if x >= 0.5 else 0 for x in Hx]    # 450 * 1
            TPandTN = [1 if ((a == 1 and b == 1) or (a == 0 and b == 0)) else 0 for (a, b) in
                       zip(predict_Y, Y)]   # 这语法糖，woc
            acc = (sum(TPandTN)) / len(TPandTN)    # (TP + TN) / (TP + FP + TN + FN)
            acc_record.append(acc)

#             if self.stop_stratege(loss_record[i-1],loss_record[i],1e-8):#超出就停止
#                 break
            ## print every 20 iteration
            if i % 500 == 1 and i > 500:
                self.visulization(i, X, Y)   # 此时开始画图

            ## 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):   # 我就很不理解，你封装一这个有什么病吗？？？？？
        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)

        predict_test_Y = [1 if x >= 0.5 else 0 for x in y]    # 450 * 1  预测值
        TPandTN = [1 if ((a == 1 and b == 1) or (a == 0 and b == 0)) else 0 for (a, b) in
                   zip(predict_test_Y, Y)]   # 我的代码不言自明

        acc_test = (sum(TPandTN)) / len(TPandTN)    # (TP + TN) / (TP + FP + TN + FN)
        return acc_test

    def visulization(self, which_step, X_, Y_):
        x = np.linspace(-10, 10)    # 生成等间距的浮点数
        y = -(self.w[0] + self.w[1] * x) / self.w[2]    # 这他妈的啥啊？？？？
        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):   # 按全体值归一化

    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): #按通道归一化

    # 这传进来的 X 是 450 * 30 的
    m, cols = X.shape
    Y = np.zeros((m, cols))
    for i in range(cols):
        curCol = X[:, [i]]
        minVals = np.min(curCol)
        maxVals = np.max(curCol)
        # tile 函数在指定方向上重复次数
```

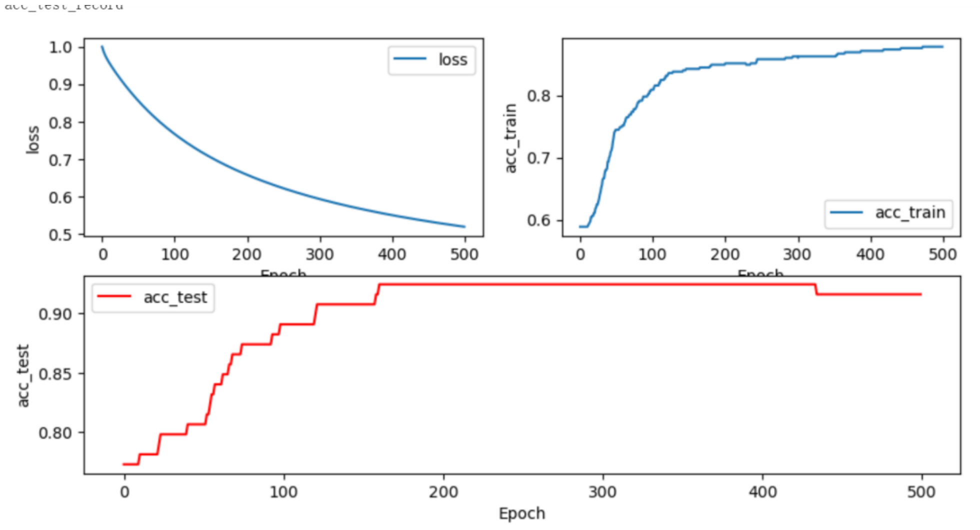
```
            ranges = maxVals - minVals
            normDataCol = curCol - np.tile(minVals, (m, 1))   # 在行方向重复 minVals m 次 和 列方向上重复 minVals 1 次
            normDataCol = normDataCol / np.tile(ranges, (m, 1))
            # print(normDataCol)
            Y[:, [i]] = normDataCol
        return Y


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()
        print("loss_record")

        plt.subplot(2, 2, 2)
        plt.xlabel('Epoch')
        plt.ylabel('acc_train')
        plt.plot(range(epochs), acc_record, label='acc_train')
        print("acc_record")
        plt.legend()

        plt.subplot(2, 1, 2)
        plt.xlabel('Epoch')
        plt.ylabel('acc_test')
        plt.plot(range(epochs), acc_test_record, label='acc_test', color='red')
        print("acc_test_record")
        # plt.plot(plt.hist['epoch'], plt.hist['acc'], label='acc', color='red')
        plt.legend()


if __name__ == "__main__":
        # data = [[-1, -1], [2, -1], [5, 3], [-1, 6], [-4, -1], [1, 4]]   # 6 * 2
        # label = [1, 1, 1, 0, 0, 0]   # 6 * 1
        # train_X = np.array(data)
        # train_Y = np.array(label)


#         train_X = AllNorm(train_X)
#         test_X = AllNorm(test_X)

        train_X = ChannalNorm(train_X)
        test_X = ChannalNorm(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)
```

No norm:

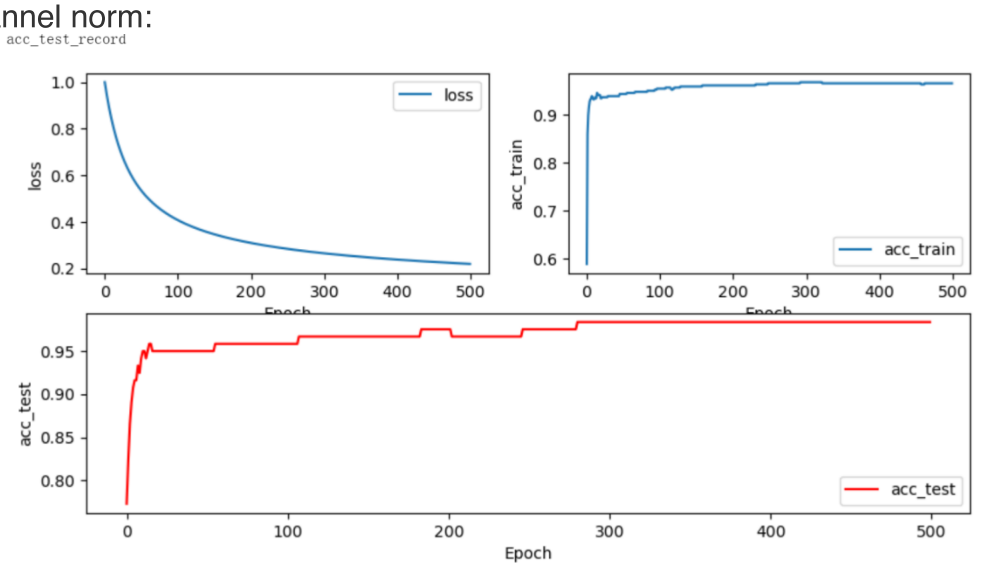
```
C:\Users\zq\AppData\Local\Temp\ipykernel_4380\1991474896.py:32: RuntimeWarning: overflow encountered in exp
  res = 1 / (1 + np.exp(-x))
```



All norm:



Channel norm:



五、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。