朴素贝叶斯在 mnist 数据集上的表现

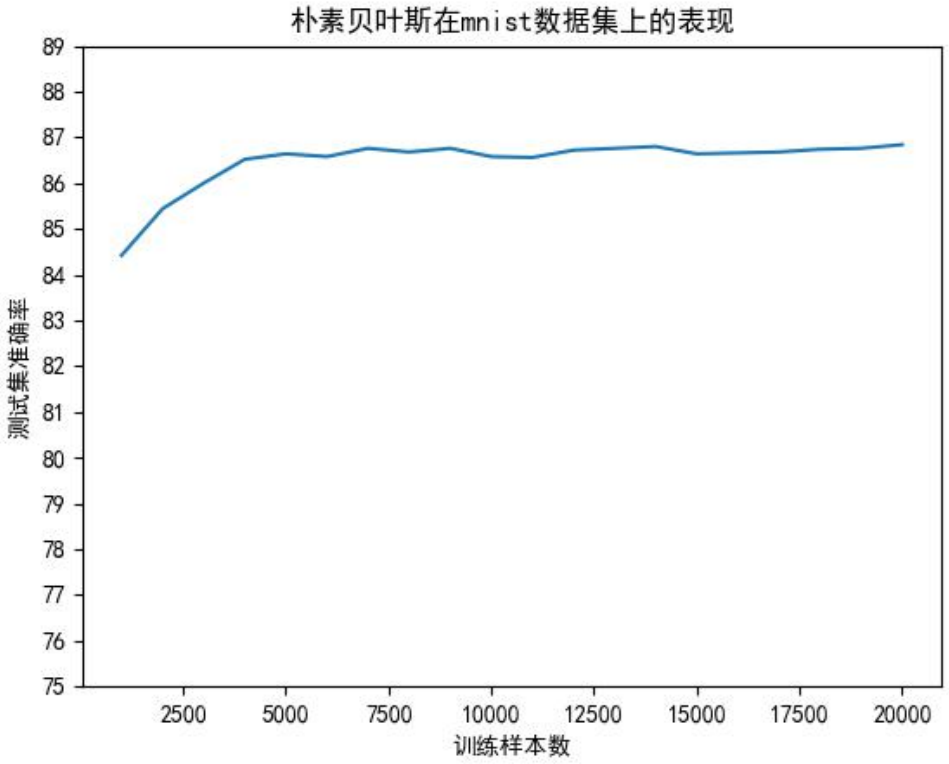
20090121 陈立标

1. 数据集处理

采用官网数据集，共有 60000 个样本。测试集数量依次选取 1000、2000、3000...21000 共 20 组，为保证准确率少受偶然因素影响，测试集选取固定 5000 个样本。

2. 训练结果

使用 matplotlib 进行绘图，如下所示。



Test_acc=[84.42, 85.44, 86.0, 86.52, 86.64, 86.58, 86.76, 86.68, 86.76, 86.58, 86.56, 86.72, 86.76, 86.8, 86.64, 86.66, 86.68, 86.74, 86.76, 86.84]

Acc_average=86.477%

3. 训练结果分析

由上图可看出，在样本数量较少时样本量的增大对准确率有提升作用，但当样本量逐渐增大，准确率保持平缓，不再和样本量有明显关系，增大样本量不能无限提高准确率。但由于朴素贝叶斯并没有通过类似梯度下降的方式进行优化模型参数，自然不会出现过拟合现象，图像也没有出现下降趋势。

4. 准确率分析

对于 mnist 数据集来说，86.477%的准确率并不出色，其原因在于朴素贝叶斯忽略了各特征(本题表现为 28*28 的像素点)之间的联系。但这种舍弃对贝叶斯来说是必须要做的，若考虑彼此间的关系，即使对像素点采取了二值化处理，不同的特征取值也达到了 2^784 的数量，六万的样本量远远不够，显然不可能进行真正的贝叶斯估计。

更进一步说，像素点之间的联系代表着其在图像上的位置关系。例如一个白色的点周围大概率也是白色的点，这些点连接起来组成了笔画，进而组成了整个数字，显然这些位置关系对判断数字是有贡献的，而朴素贝叶斯忽略了这些信息，自然导致准确率无法达到较高水平。

若使用全连接神经网络，先将 784 个特征展开成一列，使用线性函数进行不断降维至 10 类，经测试可以达到 95%的准确率，这种操作保留了一行之间的位置关系，但仍忽略了行与行之间的关系。若使用卷积神经网络，对图像进行多次卷积池化(个人认为可以理解为在保留整体位置信息的情况下通过不断牺牲清晰度来进行降维)，再展开为 10 类，可以达到 98%的准确率。

5. 代码

4.1 代码运行环境

Python 3.6.13
Numpy 1.19.5
Matplotlib 3.3.4
Scipy 1.5.4

4.2 大体思路

利用贝叶斯估计需求出每个类的先验概率及类条件概率，再利用极大似然估计的思想对全部 784 个特征的取值的类条件概率进行连乘再乘上先验概率，得到 10 个概率值取极大即为判断结果。

首先，需要对数据集进行处理。数据集中以 28*28 的二维矩阵展现图片，元素代表各个点的灰度值，其范围在 0~255 之间，其中 0 为黑色，255 为白色，数值越大表示像素越白（可以理解为笔迹更重），但对于数字的识别主要是看笔迹所停留的位置，其轻重并不重要，同时观察数据可以得知大多数的灰度取值都是 0（黑底白字），因此为了简化特征取值，若灰度值等于 0，令之为 0（笔迹没有通过此像素点），若灰度值大于 0，令之为 1（笔迹通过此像素点）。

对于先验概率，只需找出全部样本中分属于每一类的样本数，再除以总样本数；对于类条件概率，分别取出每一个类别的所有样本，对于其 784 个特征分别相加除以样本数即得到每个点的白色概率，因为已经对数据进行二值化处理，黑色的概率即为（1-白色概率），自此便可得到所有 2*10*784 个类条件概率取值。

对于测试集样本，利用极大似然估计的思想，其 784 个特征若取值为 0 则乘上白色概率，若取值为 1 则乘上黑色概率，最后再乘上对应先验概率，将得到的 10 个概率值进行取极大，对应的类别即为判断结果。值得注意的是，类条件概率是有可能取值为 0 的（例如有可能所有样本左上角的第一个

像素均为黑色），若进行连乘会导致最终结果为 0，没有比较意义。因此，在进行类条件概率的计算时进行了拉普拉斯平滑保证结果非 0。

4.2 代码

```
import scipy.io as scio    # 读取.mat 文件
import numpy as np
from functools import reduce    # 计算连乘用
import operator
import matplotlib.pyplot as plt

plt.rcParams["font.sans-serif"] = ["SimHei"]    # 解决图像乱码


def readData(xfile, yfile, index):
    # 加载文件
    x = scio.loadmat(xfile)
    y = scio.loadmat(yfile)

    x = x.get(xfile[8:-4])
    y = y.get(yfile[8:-4])

    # 二值化
    # 对于原始数据，x 范围是[0,255],0 是纯黑 255 是纯白
    # 此处处理为 0 就是黑，1 就是白
    x = np.where(x > 0, 1, 0)

    num = x.shape[0]    # 获取样本量

    x_train = x[0:index]
    y_train = y[0:index]
    x_test = x[num-5000:num]
    y_test = y[num-5000:num]

    return x_train, y_train, x_test, y_test


test_acc = []
train_acc = []
for j in range(20):
    # 读取训练集、测试集
    x_train, y_train, x_test, y_test = readData("dataset/mnist_train.mat",
"dataset/mnist_train_labels.mat",
                                                1000 + j * 1000)
```

```
        # 计算先验概率
        # 按 y 把数据分成 10 组
        px_group = []
        # 构建类似二维数组的结构
        for i in range(10):
            px_group.append(i)
            px_group[i] = []

        # px_group 的第一维表示类别[0-9], 第二维表示相应的所有 x 样例
        for i, y in enumerate(y_train):
            px_group[int(y)].append(x_train[i])

        # py[]即为先验概率

        py = []
        for i in range(10):
            py.append((len(px_group[i]) + 1) / (x_train.shape[0] + 10))  # 拉普拉斯平滑，避免概率 0 的出现

        py = np.array(py)
        py = py.reshape(-1, 1)

        # 计算类条件概率
        # 一共是 10*784 个数值，代表每个像素点为白色的概率
        # 对每个像素点来说，黑色的概率为(1-白色概率)，即得到了所有特征的所有可能取值的类条件概率值(2*10*784 个)
        pxy = []
        for i in range(10):
            group = np.array(px_group[i])  # i=0,group:(5466,784),即数字为 0 的所有 5466 个样本
            group = (np.sum(group, axis=0) + 1) / (len(px_group[i]) + 10)  # group:(784,)
            pxy.append(group)

        pxy = np.array(pxy)


        # 计算 先验概率*类条件概率并比较
        def predict_ber(x):
            result = []
            for i in range(10):
                # 提取出取值为 1 的特征(取值为 0 的会置 0)
                # 此处 temp 可认为是一个有 784 个数的列表
```

```
                    temp = x * pxy[i]
                    resulti = []
                    for j in np.where(temp == 0, 1 - pxy[i], temp):  # 若取值为 0，即为黑色，此时置概率为(1-pxy[i])对位的概率值；若取值为 1，不必改变
                        py_pxy = py[i][0] * reduce(operator.mul, j)  # 对 784 个特征取值(0 or 1)的概率进行连乘再乘上先验概率
                        resulti.append(py_pxy)
                    result.append(resulti)

                result = np.argmax(result, axis=0)  # 取 10 类中概率最大的类
                return result


        test_result = predict_ber(x_test)

        acc_test = np.where(test_result == np.squeeze(y_test), 1, 0)
        print("第{0}次测试准确率:{1}%".format(j + 1, np.sum(acc_test) * 100 / len(test_result)))
        test_acc.append(np.sum(acc_test) * 100 / len(test_result))

    print(test_acc)
    test_num = []
    for i in range(20):
        test_num.append(1000 + 1000 * i)

    plt.figure()
    plt.plot(test_num, test_acc)
    plt.xlabel("训练样本数")
    plt.ylabel("测试集准确率")
    plt.title("朴素贝叶斯在 mnist 数据集上的表现")
    plt.yticks(range(75, 90, 1))
    plt.show()
```