# 模式识别实验报告

## 实验三 线性分类器

学院：计算学部

姓名：张景润

学号：1172510217

## 一、实验内容

1、 使用 Python 或 Matlab 编程实现感知器算法和最小平方误差算法；

2、 分别使用感知器算法学习区分下列两类样本的线性分类器：

$$\omega_1:(1,1)^t,(2,2)^t,(2,0)^t$$

$$\omega_2:(0,0)^t,(1,0)^t,(0,1)^t$$

3、 MNIST 数据集测试：使用 TrainSamples 中的 30000 个 17 维特征手写数字样本训练线性分类器区分 10 个类别，TrainLabels 中包含训练样本的标签；测试线性分类器对 TestSamples 中 10000 个样本的识别正确率。

## 二、程序代码

（感知器算法和最小平方误差算法，矩阵乘法和求逆可以调用其他函数库中的程序）

```
"""
作者：张景润
学号：1172510217
"""
import numpy as np


def get_data():
    return np.array([[1, 1], [2, 2], [2, 0], [0, 0], [1, 0], [0, 1]]), np.array([0, 0, 0, 1, 1, 1])


def read_data(path, dtype=float):
    return np.genfromtxt(path, delimiter=',', dtype=dtype)


def data_augment(data_np, label_np):  # 增广数据，将数值 1 添加到第 0 列，并将标签为 1 的数据取反
    data_np = np.insert(data_np, 0, 1, axis=1)
    for idx, label in enumerate(label_np):
        if label:
            data_np[idx] = - data_np[idx]
    return data_np


def predict(w_np, data_np, label_np):
    data_np, data_num = np.insert(data_np, 0, 1, axis=1), len(data_np)
    predict_np = np.array([0 if np.dot(w_np, data) > 0 else 1 for data in data_np])
    right_num = int(np.sum(predict_np == label_np))
    print('[%d/%d]=%.2f%%' % (right_num, data_num, (right_num / data_num) * 100))
```

```
29.
30. def predict_multi(w_np_lst, c, data_np, label_np, fit=False):  # 多分类中的预测函数
31.     if not fit:
32.         data_np = np.insert(data_np, 0, 1, axis=1)
33.     data_num = len(data_np)
34.     predict_np = np.array([np.argmax([np.dot(w_np_lst[i], data) for i in range(c)]) for data in data_np])
35.     right_num = (predict_np == label_np).sum()
36.     print('[%d/%d]=%.2f%%' % (right_num, data_num, (right_num / data_num) * 100))
37.     return right_num == data_num
38.
39. def perceptron_fit(data_np, label_np):  # 感知器二分类
40.     data_np = data_augment(data_np, label_np)
41.     data_num, data_width = data_np.shape
42.     k, w_np = 0, np.random.randn(data_width)
43.     while True:
44.         if np.dot(w_np, data_np[k]) <= 0:
45.             w_np = w_np + data_np[k]
46.         k = (k + 1) % data_num
47.         flag = np.sum([np.dot(w_np, data) <= 0 for data in data_np])  # 查看错误分类的数目
48.         if not flag:
49.             break
50.     return w_np
51.
52. def lmse_fit(data_np, label_np):  # 最小平方误差准则进行二分类
    Least Minimum Squared Error
53.     data_np = data_augment(data_np, label_np)
54.     data_num, data_width = data_np.shape
55.     label_np = np.ones((data_num, 1))
56.     w_np = np.linalg.inv(data_np.T.dot(data_np)).dot(data_np.T).dot(label_np).reshape((data_width,))
57.     return w_np
58.
59. class KeslerPerceptron:
60.     def __init__(self, c=10):
61.         self.c = c
62.         self.w_np_lst = []
63.
64.     def fit(self, data_np, label_np, iter_num=100, lr=1e-7):
65.         data_np = np.insert(data_np, 0, 1, axis=1)
66.         data_num, data_width = data_np.shape
```

```
67.
68.         k, self.w_np_lst = 0, [np.random.randn(data_width) for _ in range(se
    lf.c)]
69.         iter_count = 0
70.         while True:
71.             data, label = data_np[k], label_np[k]
72.             g_label = np.dot(self.w_np_lst[label], data)
73.             for idx in range(self.c):
74.                 g_idx = np.dot(self.w_np_lst[idx], data)
75.                 if idx != label and g_idx >= g_label:
76.                     self.w_np_lst[label] += lr * data
77.                     self.w_np_lst[idx] -= lr * data
78.             k = (k + 1) % data_num
79.             if k == 0:
80.                 iter_count += 1
81.                 print('当前迭代次数为：%d' % iter_count)
82.                 if iter_count == iter_num or predict_multi(self.w_np_lst, se
    lf.c, data_np, label_np, fit=True):
83.                     break
84.         print('训练集上得到的模型最终在训练集上的分类效果如下：')
85.         predict_multi(self.w_np_lst, self.c, data_np, label_np, fit=True)
86.
87. class LmseOva:
88.     def __init__(self, c=10):
89.         self.c = c
90.         self.w_np_lst = []
91.
92.     def fit(self, data_np, label_np):
93.         def get_ova_label(data, label, cls):
94.             label1 = label.copy()
95.             label1[label == cls], label1[label != cls] = 0, 1
96.             return data, label1
97.
98.         for idx in range(self.c):
99.             data_np, label_new = get_ova_label(data_np, label_np, idx)
100.             self.w_np_lst.append(lmse_fit(data_np, label_new))
101.         print('训练集上得到的模型最终在训练集上的分类效果如下：')
102.         predict_multi(self.w_np_lst, self.c, data_np, label_np, fit=False)
103.
104. def perceptron_main():
105.     data_np, label_np = get_data()
106.     w_np = perceptron_fit(data_np, label_np)
107.     predict(w_np, data_np, label_np)
```

```
    print(w_np)

def lmse_main():
    data_np, label_np = get_data()
    w_np = lmse_fit(data_np, label_np)
    predict(w_np, data_np, label_np)
    print(w_np)

def kesler_perceptron_main():
    kesler = KeslerPerceptron(c=10)
    kesler.fit(read_data('./TrainSamples.csv'), read_data('./TrainLabels.csv', dtype=int))
    print('模型最终在测试集上的分类效果如下：')
    predict_multi(kesler.w_np_lst, kesler.c, read_data('./TestSamples.csv'), read_data('./TestLabels.csv', dtype=int))

def lmse_ova_main():
    ova = LmseOva(10)
    ova.fit(read_data('./TrainSamples.csv'), read_data('./TrainLabels.csv', dtype=int))
    print('模型最终在测试集上的分类效果如下：')
    predict_multi(ova.w_np_lst, ova.c, read_data('./TestSamples.csv'), read_data('./TestLabels.csv', dtype=int))
if __name__ == '__main__':
    perceptron_main()
    # lmse_main()
    # kesler_perceptron_main()
    # lmse_ova_main()
```

## 三、实验结果

1、仿真数据实验结果：分别给出使用感知器算法和最小平方误差算法得到的线性判别函数。

感知器算法：G(x)= [-2.04838303, 1.82138524, 1.50047833]*[1, $x_1$, $x_2$]$^T$

最小平方误差：G(x)= [-1.13513514, 0.91891892, 0.32432432]*[1, $x_1$, $x_2$]$^T$

2、MNIST 数据集实验结果：（多类别解决方案及分类正确率）

当使用 Kesler 构造法时，设置学习率 lr=1e-7，迭代次数为 iter_num=100

| 数据集 | 正确分类数 | 准确率 |
|---|---|---|
| mnist_train (30000) | 24949 | 83.16% |
| mnist_test (10000) | 8262 | 82.62% |

当使用最小平方误差时

| 数据集 | 正确分类数 | 准确率 |
|---|---|---|
| mnist_train (30000) | 23341 | 77.80% |
| mnist_test (10000) | 7751 | 77.51% |