# 模式识别实验报告

## 实验三 线性分类器

学院：

姓名：

学号：

1. **实验内容**
2. 使用Python或Matlab编程实现感知器算法和最小平方误差算法；
3. 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：



1. 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。
2. **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）

============================

感知器算法

=============================

|  |
| --- |
|  |
|  | Import numpy as np  import matplotlib.pyplot as plt |
|  |  |
|  | class perceptron(object): |
|  | def \_\_init\_\_(self, X, y): |
|  | self.X = np.concatenate((np.ones((X.shape[0], 1)), X), axis=1) |
|  | self.y = y |
|  | self.w = np.random.rand(self.X.shape[1]) |
|  |  |
|  | # Batch perceptron |
|  | def train(self): |
|  | while True: |
|  | # learning rate |
|  | alpha = np.random.rand(1) |
|  | # find points that are not be classified correctly |
|  | gradient = 0 |
|  | for i in range(self.X.shape[0]): |
|  | temp = np.dot(self.w, self.X[i]) \* self.y[i] |
|  | if temp < 0: |
|  | gradient += self.X[i] \* self.y[i] |
|  | # update self.w |
|  | self.w += alpha \* gradient |
|  | # if all the points are classified correctly, finish loop |
|  | e = np.sum((alpha \* gradient) \*\* 2) |
|  | print(e) |
|  | if e < 1e-30: |
|  | break |
|  |  |
|  | def plot(self): |
|  | idx\_1 = (self.y == 1) |
|  | x1\_1 = self.X[idx\_1, 1] |
|  | x2\_1 = self.X[idx\_1, 2] |
|  |  |
|  | idx\_2 = (self.y == -1) |
|  | x1\_2 = self.X[idx\_2, 1] |
|  | x2\_2 = self.X[idx\_2, 2] |
|  |  |
|  | plt.axis([-1, 3, -1, 3]) |
|  |  |
|  | plt.plot(x1\_1, x2\_1, 'o') |
|  | plt.plot(x1\_2, x2\_2, 'x') |
|  |  |
|  | plane\_x = np.array([np.min(self.X[:, 1]), np.max(self.X[:, 1])]) |
|  | plane\_y = (-self.w[0] - self.w[1] \* plane\_x) / self.w[2] |
|  | plt.plot(plane\_x, plane\_y) |
|  |  |
|  | plt.show() |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | X = [[1, 1], |
|  | [2, 2], |
|  | [2, 0], |
|  | [0, 0], |
|  | [1, 0], |
|  | [0, 1]] |
|  | y = [1, 1, 1, -1, -1, -1] |
|  | X = np.array(X).astype(float) |
|  | y = np.array(y) |
|  |  |
|  | model = perceptron(X, y) |
|  | model.train() |
|  | model.plot() |
|  | print(model.w) |

============================================

最小平方误差算法应用于仿真数据和Mnist数据集

============================================

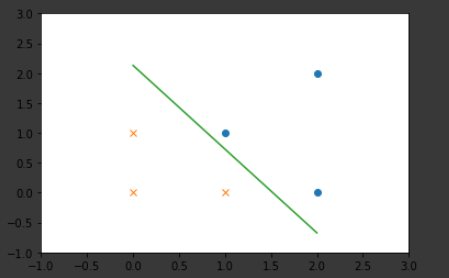
|  |
| --- |
|  |
|  | import csv  import numpy as np |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | class lmse(object): |
|  | def \_\_init\_\_(self, X, y): |
|  | self.X = np.concatenate((np.ones((X.shape[0], 1)), X), axis=1) |
|  | self.y = y |
|  |  |
|  | def train(self): |
|  | self.w = np.dot(np.linalg.pinv(self.X), self.y) |
|  |  |
|  |  |
|  | def predict(self, testData): |
|  | testData = np.concatenate((np.ones((testData.shape[0], 1)), testData), axis=1) |
|  | prediction = np.dot(testData, self.w) |
|  | prediction = np.argmax(prediction, axis=1) |
|  | return prediction |
|  |  |
|  | def plot(self): |
|  | idx\_1 = (self.y.flatten()==1) |
|  | x1\_1 = self.X[idx\_1, 1] |
|  | x2\_1 = self.X[idx\_1, 2] |
|  |  |
|  | idx\_2 = (self.y.flatten()==-1) |
|  | x1\_2 = self.X[idx\_2, 1] |
|  | x2\_2 = self.X[idx\_2, 2] |
|  |  |
|  | plt.axis([-1, 3, -1, 3]) |
|  |  |
|  | plt.plot(x1\_1, x2\_1, 'o') |
|  | plt.plot(x1\_2, x2\_2, 'x') |
|  |  |
|  | plane\_x = np.array([np.min(self.X[:, 1]), np.max(self.X[:, 1])]) |
|  | plane\_y = (-self.w[0]-self.w[1]\*plane\_x) / self.w[2] |
|  | plt.plot(plane\_x, plane\_y) |
|  |  |
|  | plt.show() |
|  |  |
|  | def eval(prediction, label): |
|  | total = label.shape[0] |
|  | temp = np.ones(label.shape[0]) |
|  | error = np.sum(temp[prediction!=label]) |
|  | return (total-error)/total |
|  |  |
|  | def readData(sampleFile, labelFile): |
|  | sampleReader = csv.reader(open(sampleFile, 'r')) |
|  | labelReader = csv.reader(open(labelFile, 'r')) |
|  |  |
|  | sample = [] |
|  | for line in sampleReader: |
|  | sample.append(line) |
|  |  |
|  | label = [] |
|  | for line in labelReader: |
|  | label.append(line) |
|  |  |
|  | return sample, label |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | """ |
|  | X = [[1, 1], |
|  | [2, 2], |
|  | [2, 0], |
|  | [0, 0], |
|  | [1, 0], |
|  | [0, 1]] |
|  | y = [1, 1, 1, -1, -1, -1] |
|  |  |
|  | X = np.array(X).astype(float) |
|  | y = np.array(y) |
|  | y = y.reshape((y.shape[0], 1)) |
|  |  |
|  | model = lmse(X, y) |
|  | model.train() |
|  | print(model.w) |
|  | model.plot() |
|  | """ |
|  | trainData, trainLabel\_temp = readData('TrainSamples.csv', 'TrainLabels.csv') |
|  | trainData = np.array(trainData).astype(float) |
|  | trainLabel\_temp = np.array(trainLabel\_temp).astype(int) |
|  | trainLabel = np.zeros((trainLabel\_temp.shape[0], 10)) |
|  | i = 0 |
|  | while i<trainLabel\_temp.shape[0]: |
|  | trainLabel[i][trainLabel\_temp[i]] = 1 |
|  | i += 1 |
|  |  |
|  | testData, testLabel = readData('TestSamples.csv', 'TestLabels.csv') |
|  | testData = np.array(testData).astype(float) |
|  | testLabel = np.array(testLabel).astype(int).flatten() |
|  |  |
|  | model = lmse(trainData, trainLabel) |
|  | model.train() |
|  | prediction = model.predict(testData) |
|  | print(eval(prediction, testLabel)) |

1. **实验结果**
2. 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

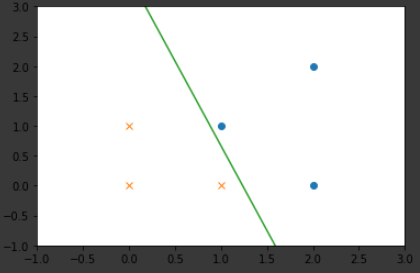
===============



==================

最小平方误差算法

==================



2.MNIST数据集实验结果：（多类别解决方案及分类正确率）

多类别解决方案: 每个类别对应一个线性函数，判别准则为线性函数输出最大的类别

10类别分类准确率: 77.51%