

# 《 机 器 视 觉 》

## 实 验 报 告

学号： 2018213106

姓名： 刘嘉伟

班级： 计算机创新实验 18-1 班

# 实验三

## 一、实验目的

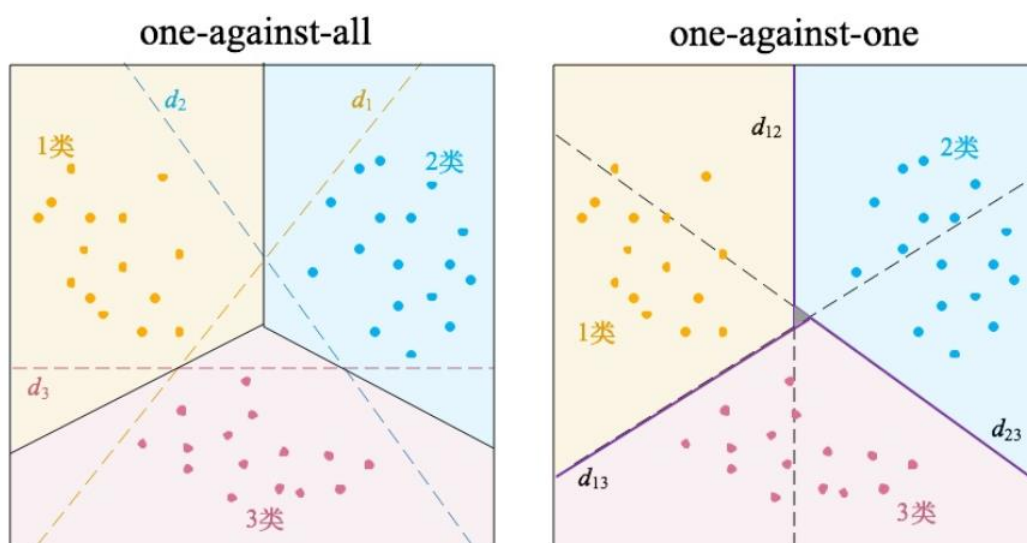
实现一种图像识别算法，并对算法所能达到的效果进行分析。

## 二、实验内容

基于 SVM 的手写体数字识别的实现

## 三、算法原理

支持向量机分类算法最初只用于解决二分类问题，缺乏处理多分类问题的能力。后来随着需求的变化，需要 svm 处理多分类分为。目前构造多分类支持向量机分类器的方法主要有两类：一类是“同时考虑所有分类”方法，另一类是组合二分类器解决多分类问题。



第一类方法主要思想是在优化公式的同时考虑所有的类别数据，J.Weston 和 C.Watkins 提出的“K-Class 多分类算法”就属于这一类方法。该算法在经典的 SVM 理论的基础上，重新构造多类分类型，同时考虑多个类别,然后将问题也转化为一个解决二次规划(Quadratic Programming,简称 QP)问题，从而实现多分类。该算法由于涉及到的变量繁多，选取的目标函数复杂，实现起来比较困难，计算复杂度高。

第二类方法的基本思想是通过组合多个二分类器实现对多分类器的构造，常见的构造方法有“一对一”(one-against-one)和“一对其余”(one-against-the rest 两种。其中“一对一”方法需要对  $n$  类训练数据两两组合，构建  $n(n-1)/2$  个支持向量机，每个支持向量机训练两种不同类别的数据，最后分类的时候采取“投票”的方式决定分类结果。“一对其余”方法对  $n$  分类问题构建  $n$  个支持向量机，每个支持向量机负

责区分本类数据和非本类数据。该分类器为每个类构造一个支持向量机，第  $k$  个支持向量机在第  $k$  类和其余  $n-1$  个类之间构造一个超平面，最后结果由输出离分界面距离  $wx+b$  最大的那个支持向量机决定。Sklearn 里的 SVM 包使用的就是一对其余分类，训练十个分类器，每个分类器的一边是该数字本身，另一边是其他的九个数字，从而实现了十种手写数字的分类与识别。

## 四、算法的 python 实现

### 1、读取 mnist 数据集中的训练集和测试集

```
def load_mnist_data(path, kind):
    labels_path = os.path.join(path, '%s-labels.idx1-ubyte' % kind)
    images_path = os.path.join(path, '%s-images.idx3-ubyte' % kind)
    with open(labels_path, 'rb') as lbpath:
        magic, n = struct.unpack('>II', lbpath.read(8))
        labels = np.fromfile(lbpath, dtype=np.uint8)
    with open(images_path, 'rb') as imgpath:
        magic, num, rows, cols = struct.unpack('>IIII', imgpath.read(16))
        images = np.fromfile(imgpath, dtype=np.uint8).reshape(len(labels),
784)
    return images, labels

train_images, train_labels = load_mnist_data(path, 'train')
test_images, test_labels = load_mnist_data(path, 't10k')
```

### 2、将测试集和训练集的输入特征向量进行归一化，以便拟合分类器

```
X = preprocessing.StandardScaler().fit_transform(train_images)
XT = preprocessing.StandardScaler().fit_transform(test_images)
```

### 3、取测试集的前 10000 个数据来进行测试

```
XT = XT[0:10000]
test_labels = test_labels[0:10000]
```

### 4、拟合分类器并保存分类器，以便测试使用

```
print(time.strftime('%Y-%m-%d %H:%M:%S'))
print("-----start training-----")
clf = SVC(C=100)
# model_svc = svm.SVC(kernel='poly', C=0.1, gamma=0.01)
clf.fit(X, train_labels)

with open('./model.pkl', 'wb') as file:
    pickle.dump(clf, file)
```

```
print(time.strftime('%Y-%m-%d %H:%M:%S'))
print("-----done-----")
```

5、用测试集对分类器进行评估，并保存分类错误的图片，以便调试，打印精度

```
with open('./model.pkl', 'rb') as file:
    clf = pickle.load(file)
predict = clf.predict(XT)
precision = np.mean(predict == test_labels)

for i in range(predict.shape[0]):
    if predict[i] != test_labels[i]:
        cv2.imwrite("errorimg/" + str(i) + '_label' + str(test_labels[i]) +
            '_pred' + str(predict[i]) + '.jpg',
            test_images[i].reshape(28, 28))




print(precision)
```





## 五、结果展示与评价

```
L:\software\python3.6\python.exe L:/Python/Pytorch/CV/exp3/exp3.py
2020-11-04 10:55:26
-----start training-----
2020-11-04 11:05:11
-----done-----
0.9716

Process finished with exit code 0
```

六万多张图片训练了十分钟左右，最后的精度达到了 97.16%，可以说是相当不错的，我们再来查看一下分类错的图片：

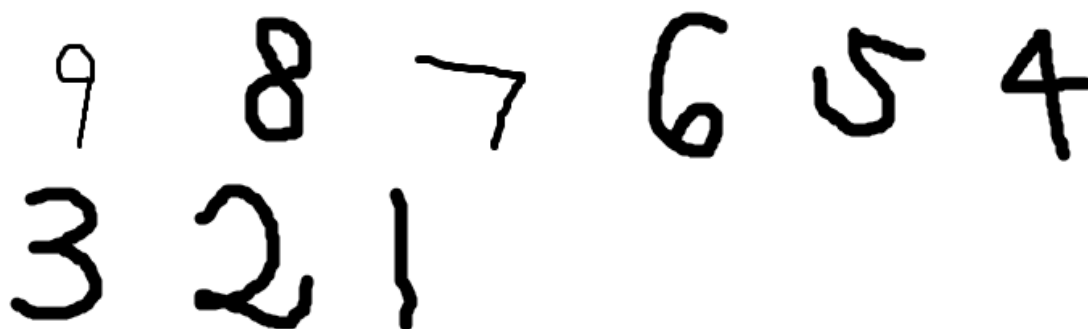
 label:9,pred:4,  label:7,pred:4,  label:2,pred:7,  label:9,pred:8

 label:3,pred:2,  label:4,pred:2,  label:2,pred:7,  label:5,pred:3

这些是预测错误的一部分结果，大体上来看，可以发现有些数字写的十分歪曲，即便是让人眼来区分都不一定能分得出来，所以可以说明，分类器的分类效果已经很好了，对于一些很歪的图片识别不出来也情有可原。

## 六、自己手写的字体的识别

原始手写图像：



转换为 28\*28 的二值图像

```
for imgs in os.listdir(test_path):
    img_path = os.path.join(test_path, imgs)
    img = cv2.imread(img_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    resize_img = cv2.resize(gray, (28, 28))
    ret, binary_img = cv2.threshold(resize_img, 127, 255,
cv2.THRESH_BINARY_INV)
    cv2.imwrite('./binarydata/' + imgs, binary_img)
    img_list.append(binary_img)
    label_list.append(eval(imgs.split('.')[0]))
```




对图像进行预测

```
f = img_list[0].reshape(1, -1)
for i in range(1, len(img_list)):
    f = np.vstack((f, img_list[i].reshape(1, -1)))

f = preprocessing.StandardScaler().fit_transform(f)
pred = model.predict(f)
pred = list(pred)
for i in range(len(label_list)):
    if label_list[i] != pred[i]:
        cv2.imwrite("testerrorimg/" + str(i) + ".jpg",img_list[i])
```

预测结果：

(0, 0)  
(1, 1)  
(2, 2)  
(3, 3)  
(4, 4)  
(5, 5)  
(6, 6)  
(7, 7)  
(8, 8)  
(4, 9)

误将 9 预测成了 4, , 可能是由于将图片缩小到 28\*28 之后图像变得不连续导致的, 我再画一张粗笔写的 9, 再次进行预测



, 二值图像为 , 最后看到预测结果

(0, 0)  
(1, 1)  
(2, 2)  
(3, 3)  
(4, 4)  
(5, 5)  
(6, 6)  
(7, 7)  
(8, 8)  
(9, 9)

, 全部都预测正确。