華東理工大學

模式识别大作业

题　　目　基于多种模型的 MNIST 手写数字分类

学　　院　信息科学与工程

专　　业　控制科学与工程

组　　员　徐庆　李倩玉　李雪梅　何烈芝

指导教师　赵海涛

完成日期：　2019 年 12 月 11 日

一、课题背景

本次课题选取了 Kaggle InClass 上的 MNIST 手写体数字识别项目。MNIST 全称是 MixedNationl Istute of Standards and Technology databasco，是机器学习领域中非常经典的一个数据集。该数据集有两个功能，一是提供了大量的数据作为训练集和验证集，为一些学习人员提供了丰富的样本信息。另一个功能就是可以形成一个在业内相对有普适性的基准来比对项目，因为大家用的数据集都是一样的，那么每个人设计出来的网络就可以在这些数据集上不断互相比较，从而验证谁的网络设计得识别率更高。

目前识别手写数字的方法主要有决策树、神经网络和支持向量机方法。本课题通过采用 k 近邻、SVM、Logistic 以及决策树模型分别实现手写体数字识别任务，对比这四个模型用于手写体数字识别的效果，以实践课堂上所学的知识。

二、数据集预处理

2.1 数据集分析

MNIST 数据集由 60000 个训练样本和 10000 个测试样本组成，每个样本都是一张 28 * 28 像素的灰度手写数字图片。

在网上可以下载这 4 个文件，训练集、训练集标签、测试集、测试集标签。

文件名称	大小	内容
train-images-idx3-ubyte.gz	9,681 kb	55000张训练集，5000张验证集
train-labels-idx1-ubyte.gz	29 kb	训练集图片对应的标签
t10k-images-idx3-ubyte.gz	1,611 kb	10000张测试集
t10k-labels-idx1-ubyte.gz	5 kb	测试集图片对应的标签

读入 MNIST 数据集：直接下载下来的数据是无法通过　解压或者应用程序打开的，因为这些文件不是任何标准的图像格式而是以字节的形式进行存储的，所以必须编写程序来打开它。

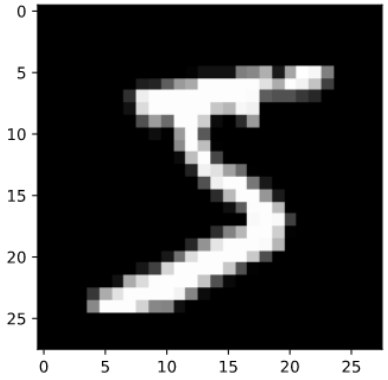
通常对于分类问题，我们会将数据集分成三部分，训练集、测试集、交叉验证集。用训练集训练生成模型，用测试集和交叉验证集进行验证模型的准确性。

2.2 数据集导入

关键代码如下

```
def load_mnist(path):
    #path:mnist data path
    mnist=fetch_openml('mnist_784',data_home=path)
    image=mnist['data']#shape:(70000,784)
    image=image/255
    label=mnist['target']#shape:(70000,)
    return image,label
```

先利用 fetch_mldata 包直接导入 mnist 二进制格式数据，并进行归一化，返回处理后图像矩阵与对应文本格式的标签。Mnist 数据图如下例：



导出代码为：

```
image,label=load_mnist(mnist_path)

image_=(image[0,:]*255).reshape(28,28)

plt.imshow(image_,cmap='gray')
```

三、基于 *k* 近邻法的手写体数字识别

3.1 实验原理

k 近邻（K-Nearest Neighbor，简称 KNN）分类算法是最简单的机器学习算法之一。其核心思想是将样本用它最接近的 *k* 个邻居来代表。

KNN 算法首先将待分类样本表达成和训练样本一致的特征向量，然后根据距离计算待测试样本和每个训练样本的距离，选择距离最小的 *k* 个样本作为近邻样本，最后根据 *k* 个近邻样本判断待分类样本的类别。最后选择 k 个近邻样本中

出现次数最多的分类，作为新数据的分类。

在 k 近邻算法中，对于给定训练数据集

$$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$$

其中，$x_i \in x \subseteq R^n$ 为实例的特征向量，$y_i \in y=\{c_1,c_2,\cdots,c_K\}$ 为实例的类别，$i=1,2,\cdots,N$，实例特征向量 x；

来判断实例 x 所属的类 y。

首先根据给定的距离度量，在训练 T 中找到与 x 最近邻的 k 个点，涵盖着 k 个点的 x 的邻域记作 $N_k(x)$。

然后在 $N_k(x)$ 中根据分类决策规则（如多数表决）决定 x 的类别 y：

$$y=\arg\max_{c_j}\sum_{x_i\in N_k(x)} I(y_i=c_j), i=1,2,\cdots N; j=1,2,\cdots,K$$

其中，I 为指示函数，即当 $y_i=c_j$ 时 I 为 1，否则 I 为 0。

k 近邻法的特殊情况是 k= 1 的情形，称为最近邻算法。对于输入的实例点（特征向量）x，最近邻法将训练数据集中与 x 最邻近点的类作为 x 的类。

收集和准备数据，这里使用的是 MNIST 数据集，输入样本数据和结构化的输出结果，可以调整 k 的值，然后运行 k-近邻算法判断输入数据分别属于哪个分类，最后计算错误率和准确率。详细实现过程是将 MNIS 数据集包括训练集和验证集导入到工程文件中，接着计算验证集和训练集的距离，从小到大排序得到距离最近的 k 个邻居，并通过投票得到所属类别中最高的类别，判断该验证集的图片是否属于该类别，接着将该类别的标签和验证集的标签进行比对，相符合则是正确的，不相符合则属于出错，最后输出计算的错误率和准确率。

3.2 实验内容

选择距离度量为“minkowski”，近邻数 k=10，权重设为“uniform”，关键代码段如下：

```
image,label=load_mnist(mnist_path)
train_image=image[:60000,:]
test_image=image[60000:,:]
train_label=label[:60000]
test_label=label[60000:]
```

```
neigh=KNeighborsClassifier(n_neighbors=10)
neigh.fit(train_image,train_label)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                     weights='uniform')
```

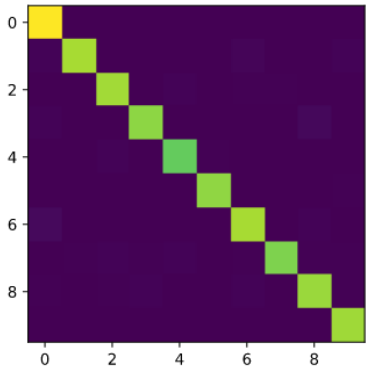
```
predict_label=neigh.predict(test_image)
```

测试结果为：

准确率：0.94333333 0.982　0.96825397 0.97409326 0.97075366 0.97720207 0.95715677 0.98597627 0.95143707 0.96237624

召回率：0.99735683 0.95155039 0.96633663 0.95723014 0.96748879 0.98434238 0.95622568 0.93839836 0.95143707 0.99183673

混淆矩阵可视化为：



四、基于 Logistic 的手写体数字识别

4.1 实验原理

Logistic 回归又称逻辑回归分析，是一种广义的线性回归分析模型，常用于数据挖掘，疾病自动诊断，经济预测等领域。

Logistic 回归的算法原理和线性回归的算法步骤大致相同，只是预测函数 H 和权值更新规则不同。逻辑回归算法在这里应用于多分类，由于 MNIST 的数据集是共有十类的手写数字图片，所以应该使用十个分类器模型，分别求出每

类最好的权值向量，并将其应用到预测函数中，预测函数值相当于概率，使得预测函数值最大对应的类就是所预测的类。

Logistic 回归模型学习时，对于给定的数据集

$$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\},$$

其中，$x_i\in R^n$，$y_i\in\{0,1\}$，可以应用极大似然估计法估计模型参数，从而得到 Logistic 回归模型。

设 $$P(Y=1|x)=\pi(x)，\quad P(Y=0|x)=1-\pi(x)$$

似然函数为

$$\prod_{i=1}^{N}[\pi(x_i)]^{y_i}[1-\pi(x_i)]^{1-y_i}$$

对数似然函数为

$$\begin{aligned}L(w)&=\sum_{i=1}^{N}[y_i\log\pi(x_i)+(1-y_i)\log(1-\pi(x_i))\\&=\sum_{i=1}^{N}\left[y_i\log\frac{\pi(x_i)}{1-\pi(x_i)}+\log(1-\pi(x_i)\right]\\&=\sum_{i=1}^{N}\left[y_i(w\cdot x_i)-\log(1+\exp(w\cdot x_i)\right]\end{aligned}$$

对 $L(w)$ 求极大值，得到 w 的估计值。

这样，问题就变成了以对数似然函数为目标函数的最优化问题，Logistic 回归学习中通常采用的方法是梯度下降法及拟牛顿法。

假设 w 的极大似然估计值是 $\hat{w}$，那么学到的 Logistic 回归模型为

$$P(Y=1|x)=\frac{\exp(\hat{w}\cdot x)}{1+\exp(\hat{w}\cdot x)}$$

$$P(Y=0|x)=\frac{1}{1+\exp(\hat{w}\cdot x)}$$

4.2 实验内容

模型训练迭代算法选择拟牛顿法，算法迭代次数 1000 次，关键代码段：

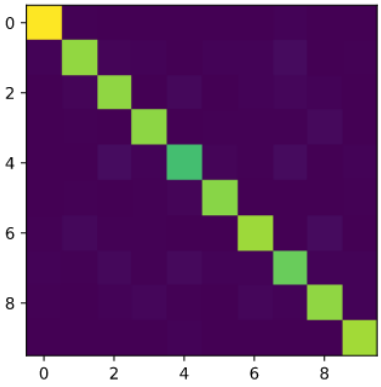
```
[24]
image,label=load_mnist(mnist_path)
train_image=image[:60000,:]
test_image=image[60000:,:]
train_label=label[:60000]
test_label=label[60000:]

[25]
model = LogisticRegression(max_iter=1000)
print(model)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)

[27]
model.fit(train_image, train_label)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
```

得到如下结果：

准确率：0.96187175 0.92907093 0.90420332 0.93692777 0.89516129 0.94312306 0.93208661 0.88053553 0.91034483 0.95309381

召回率：0.97797357 0.90116279 0.91584158 0.93788187 0.87107623 0.9519833 0.92120623 0.87782341 0.91575818 0.9744898

可视化混淆矩阵如下：



五、基于 SVM 的手写体数字识别

5.1 实验原理

支持向量机（简称 SVM）是一类按监督学习方式对数据进行二元分类的广义线性分类器，可以支持线性和非线性的分类。自从上世纪 90 年代被提出以来，SVM 就一直是手写数字识别领域的热门方法。

SVM 分类算法的基本思想是利用最大间隔进行分类。训练 SVM 等价于解

一个线性约束的二次规划问题，使得分隔特征空间中两类模式点的两个超平面之间距离最大，并保证得到的解为全局最优点。

SVM 在用于实际问题解决中面对的往往是线性不可分的情况，即在样本空间不存在一个超平面能够正确划分样本类别。面对这样的情况，可以将样本投影到高维特征空间，样本在这个高维特征空间是线性可分的。则划分超平面可以表示为：

$$f(x)=\omega^T\phi(x)+b$$

关于 w, b 的原始问题为：

$$\min_{\omega,b}\frac{1}{2}\|\omega\|^2$$

$$s.t \quad y_i(\omega^T\phi(x_i)+b)\geq 1 \quad i=1,2,...,m$$

对偶问题变为：

$$\min_{\lambda}\frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{m}\lambda_i\lambda_j y_i y_j\phi(x_i)^T\phi(x_j)-\sum_{i=1}^{m}\lambda_i$$

$$s.t \quad \sum_{i=1}^{m}\lambda_i y_i \quad i=1,2,...,m$$

其中，$\phi(x_i)^T\phi(x_j)$表示映射在特征里的内积，直接计算中$\phi(x_i)^T\phi(x_j)$是比较困难。所以引入了核函数：

$$\kappa(x_i,x_j)=\phi(x_i)^T\phi(x_j)$$

即 x_i，x_j 在特征空间的内积等于在样本的原空间中通过 $\kappa(.,.)$ 计算得到，称 $\kappa(x,z)$ 为核函数。

5.2 实验内容

选择核函数为 rbf 函数，模型格式为“OvR”（即针对 10 类样本创建 10 个模型，第 i 个分类器对第 i 个样本与其余样本做二分类），关键代码段如下：

```
svc=svm.SVC()
image,label=load_mnist(mnist_path)
train_image=image[:60000,:]
test_image=image[60000:,:]
train_label=label[:60000]
test_label=label[60000:]
svc.fit(train_image,train_label)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
predict_label=svc.predict(test_image)
```

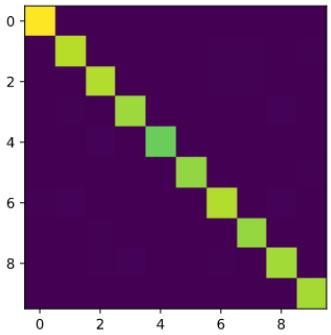
```
result_matrix=np.zeros((10,10))
for i in range(len(predict_label)):
    x=int(test_label[i])-1
    y=int(predict_label[i])-1
    result_matrix[x,y]=result_matrix[x,y]+1
```

得到如下结果：

准确率：0.98858648 0.9757517　0.97453477 0.98261759 0.98640997 0.98538622 0.9755142　0.97137014 0.97194389 0.97985901

召回率：0.99207048 0.9748062　0.98514851 0.97861507 0.9764574　0.98538622 0.9688716　0.97535934 0.96134787 0.99285714

混淆矩阵可视化为：



六、基于决策树的手写体数字识别

6.1 实验原理

决策树是一种树形结构，其中每个内部节点表示一个属性上的测试，每个分支代表一个测试输出，每个叶节点代表一种类别。

决策树是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是

直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。在机器学习中，决策树是一个预测模型，它代表的是对象属性与对象值之间的一种映射关系。Entropy=系统的凌乱程度，使用算法 ID3，C4.5 和 C5.0 生成树算法使用熵。这一度量是基于信息学理论中熵的概念。决策树学习的方法通常是一个递归地选择最优特征，并根据该特征对训练数据进行分割，从而使得对子数据集有一个最好的分类的过程。决策树学习的本质是从训练数据集中归纳出一组分类规则，使其与训练数据矛盾较小并且具有较好的泛化能力。决策树学习的损失函数常采用正则化的极大似然函数，学习策略为损失函数最小化，但从所有决策树模型中选择最优决策树是一个 NP 完全问题，常采用启发式方法近似求解这个问题，这样求解到的决策树是次最优的。

6.2 实验内容

以基尼系数作为模型训练中的评价指标，基于 sklearn 决策树模型进行训练并测试各类样本的准确率与召回率，关键代码段如下：

```
image,label=load_mnist(mnist_path)
train_image=image[:60000,:]
test_image=image[60000:,:]
train_label=label[:60000]
test_label=label[60000:]
```

```
tree=DecisionTreeClassifier()
```

```
tree.fit(train_image,train_label)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

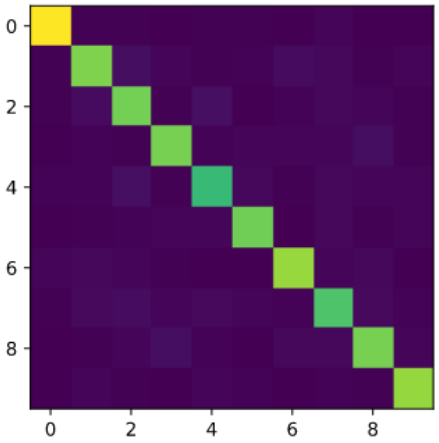
```
predict_label=tree.predict(test_image)
```

得到如下结果：

准 确 率 ： 0.95426561 0.86234022 0.82279693 0.87589013 0.83409091 0.90435707 0.89746094 0.81808732 0.85247525 0.91117764

召 回 率 ： 0.95594714 0.8498062 0.85049505 0.87678208 0.82286996 0.88830898 0.89396887 0.80800821 0.85332012 0.93163265

混淆矩阵可视化：



综合上述实验结果汇总如下：

model	KNN		Logistic		SVM		Decision_tree	
class	P	r	p	r	p	r	p	r
0	0.943	0.997	0.962	0.978	0.989	0.992	0.954	0.956
1	0.982	0.952	0.929	0.901	0.976	0.975	0.862	0.850
2	0.968	0.966	0.904	0.916	0.975	0.985	0.822	0.850
3	0.974	0.957	0.937	0.938	0.983	0.979	0.876	0.877
4	0.971	0.967	0.895	0.871	0.986	0.976	0.834	0.823
5	0.977	0.984	0.943	0.952	0.985	0.985	0.904	0.888
6	0.957	0.956	0.932	0.921	0.976	0.969	0.897	0.894
7	0.986	0.938	0.881	0.878	0.971	0.975	0.818	0.808
8	0.951	0.951	0.910	0.916	0.972	0.961	0.852	0.853
9	0.962	0.992	0.953	0.974	0.980	0.993	0.911	0.931

(p:precision/r:recall)

七、心得体会

MNIST 数据集是机器学习的入门级数据库，很多学者都利用不同的方法在该数据集上取得了非常出色的分类效果，这一点是本课题所无法实现的。我们的目的在于，利用此次课题实践的机会，熟悉基本理论，并实践课上所学的理论知识。

此次大作业给我们小组最深刻的体会是：理论是抽象的，问题是具体的。站

在岸上学不会游泳，光看着梨子不可能知道梨子的滋味，特别是码代码这种事。真正动手做起来，才发现自己学会了理论推导，学会了考试做题，但是一到实践还是脑袋空空。感谢老师能给我们这次机会，能将课堂上所学的理论知识真正用于实践。此次实验的最大收获就是了解了 sklearn 的强大，以及利用它来调用各种分类器解决分类问题。在实验程序的调试过程也出现了许多问题，于是查阅了相关的资料并反复的检查核对程序，最后才运行成功。

最后要感谢赵老师的辛勤教导。老师的上课方式幽默有趣，不时穿插些其他内容，给课堂带来欢乐。虽然课程有些难懂，课下需要花更多的时间去消化理解，但老师的指导让我们对模式识别这个方向有了基本的了解。我们也深知半个学期的学习是远远不够的，以后还会花更多的时间去探知。