《数据挖掘与机器学习》实验报告

题　　目：分类算法 - 手写数字识别

学　　号：19024075

姓　　名：陶盛皿

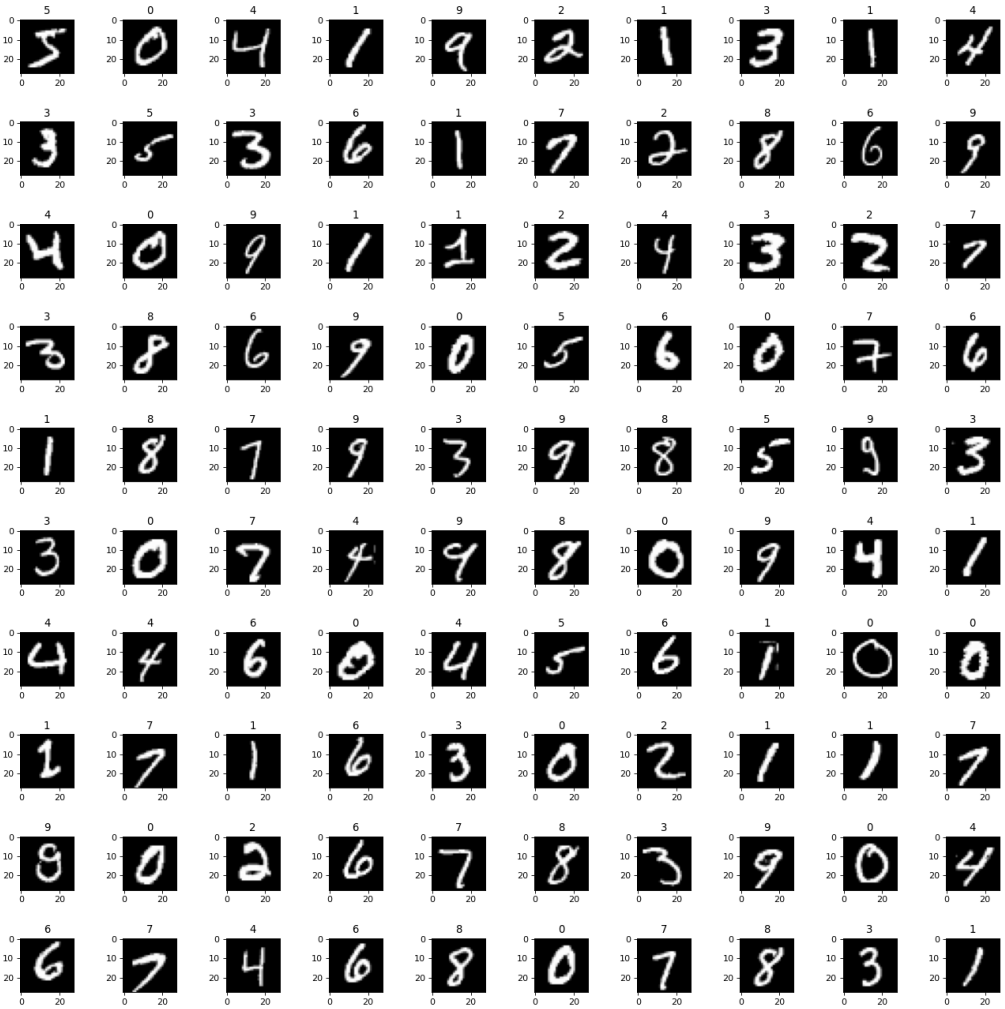
分类算法 - 手写数字识别

1. 实验要求

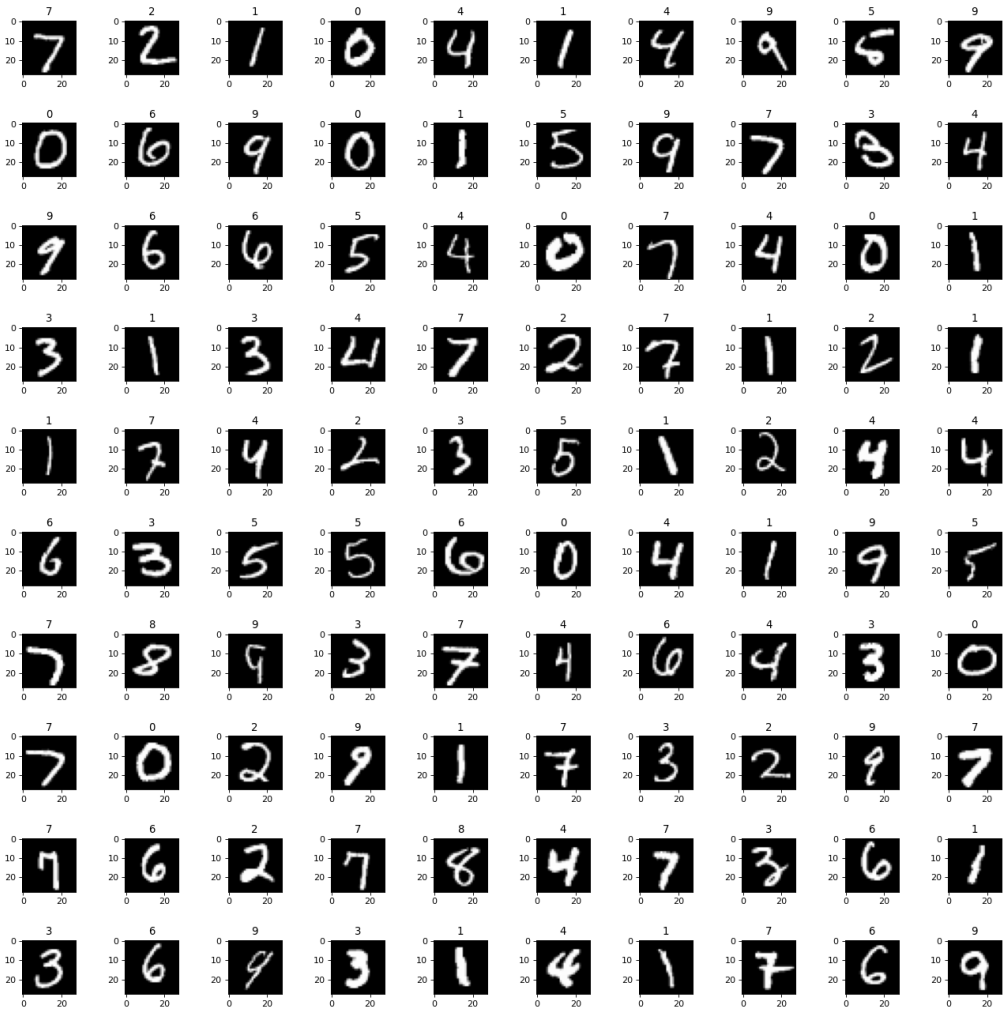
1. 对原始数据集二进制转化，并可视化。
2. 使用数据集进行分类算法性能比较。

2. 实验内容

1. 对原始数据集't10k-images.idx3-ubyte'，'t10k-labels.idx1-ubyte'，'train-images.idx3-ubyte'，'train-labels.idx1-ubyte'利用定义函数进行二进制转化，并进行可视化。
    - 训练集Top100可视化



    - 测试集Top100可视化



2. 使用sklearn中fetch openml，下载现有mnist数据集，进行K-近邻，随机森林， KMeans聚类，SGD梯度下降算法，对手写数字分类的实现， 并比较准确率和调用时间。
    - 结果展示

	Random Forest	KNN	KMeans	SGDClassifier
accuracy	0.968714	0.967286	0.095429	0.878357
adjusted_rand_score	0.932508	0.928644	0.360800	0.753056
time_diff	51.282473	48.700803	58.966298	5.087178

结论：随机森林分类法准确率最高约96.98%， K-近邻其次月96.73%，梯度下降分类法月87.84%，聚类算法效果不好，明显聚类算法不适用。在保证准确率大于90%的前提下，由于K近邻算法花费时间较少，并且最准确率最高的算法准确率相差仅0.25%，故选用K近邻算法。

---

3. 算法思想

- 随机森林算法思想
    - 将多个不同的决策树进行组合，利用这种组合降低单一决策树有可能带来的片面性和判断不准确性。
        - 具体来讲，随机森林是用随机的方式建立一个森林。随机森林是由很多的决策树组成，但每一棵决策树之间是没有关联的。在得到森林之后，当对一个新的样本进行判断或预测的时候，让森林

中的每一棵决策树分别进行判断，看看这个样本应该属于哪一类（对于分类算法），然后看看哪一类被选择最多，就预测这个样本为那一类。

- K近邻算法思想
  - 对未知类别属性的数据集中的每个点依次执行以下操作：
    1. 计算已知类别数据集中的点与当前点之间的距离；
    2. 按照距离递增次序排序；
    3. 选取与当前点距离最小的 k 个点；
    4. 确定前 k 个点所在类别的出现频率；
    5. 返回前k个点出现频率最高的类别作为当前点的预测分类。

- KMeans聚类算法思想
  - 在数据集中根据一定策略选择K个点作为每个簇的初始中心，然后观察剩余的数据，将数据划分到距离这K个点最近的簇中，也就是说将数据划分成K个簇完成一次划分，但形成的新簇并不一定是最好的划分，因此生成的新簇中，重新计算每个簇的中心点，然后在重新进行划分，直到每次划分的结果保持不变。在实际应用中往往经过很多次迭代仍然达不到每次划分结果保持不变，甚至因为数据的关系，根本就达不到这个终止条件，实际应用中往往采用变通的方法设置一个最大迭代次数，当达到最大迭代次数时，终止计算。
  - 具体的算法步骤如下：
    1. 随机选择K个中心点
    2. 把每个数据点分配到离它最近的中心点；
    3. 重新计算每类中的点到该类中心点距离的平均值
    4. 分配每个数据到它最近的中心点；
    5. 重复步骤3和4，直到所有的观测值不再被分配或是达到最大的迭代次数（R把10次作为默认迭代次数）。

- 随机梯度下降分类算法思想
  1. 正向传递：数据，经过模型，得到预测值，再计算出损失值
  2. 损失值：帮助我们判断现有模型的好坏，需要改进多少
  3. 优化算法： 帮助我们从损失值出发，一步一步更新参数，完善模型
  4. brute force：比如随机选择1000个值，依次作为某个参数的值，得到1000个损失值，选择其中那个让损失值最小的值，作为最优的参数值

     因此，产生了随机梯度下降算法，基于损失值，去更新参数，且要大幅降低计算次数。

---

4. 代码展示

```
# 可视化数据集
#coding = utf-8
import numpy as np
import struct
import matplotlib.pyplot as plt
%matplotlib inline
```

```
def readfile(file1, file2):
    '''

    :param file1:
    :param file2:
    :return:
    '''
    binFile = open(file1,'rb')
    binFile_buf = binFile.read()

    Label = open(file2,'rb')
    lbl = Label.read()
    return binFile_buf, lbl

def get_image(binFile_buf):
    image_idx = 0
    image_idx += struct.calcsize('>IIII')
    magic, nImage, nImgRows, nImgCols = struct.unpack_from('>IIII', binFile_buf, 0)
    im = []
    for i in range(100):
        tmp = struct.unpack_from('>784B', binFile_buf, image_idx)
        im.append(np.reshape(tmp, (28,28)))
        image_idx += struct.calcsize('>784B')
    return im

def get_label(lbl):
    label_idx = 0
    label_idx += struct.calcsize('>II')
    return struct.unpack_from('>100B', lbl, label_idx)

if __name__ == "__main__":
    test_image_data, test_label_data = readfile('t10k-images.idx3-ubyte','t10k-labels.idx1-ubyte')
    test_im = get_image(test_image_data)
    test_label = get_label(test_label_data)
    plt.figure(figsize=(20, 20), dpi=80)
    for i in range(100):
        plt.subplot(10, 10, i+1)
        plt.subplots_adjust(wspace=0.9, hspace=0.9)
        title = str(test_label[i])
        plt.title(title)
        plt.imshow(test_im[i], cmap='gray')
    plt.show()

    train_image_data, train_label_data = readfile('train-images.idx3-ubyte', 'train-labels.idx1-ubyte')
    train_im = get_image(train_image_data)
    train_label = get_label(train_label_data)
    plt.figure(figsize=(20, 20), dpi=80)
    for i in range(100):
        plt.subplot(10, 10, i + 1)
        plt.subplots_adjust(wspace=0.9, hspace=0.9)
        title = str(train_label[i])
        plt.title(title)
```

```
        plt.imshow(train_im[i], cmap='gray')

    plt.show()
```

```
# 建模训练模型、预测
from sklearn.model_selection import train_test_split #数据集训练集划分
from sklearn.ensemble import RandomForestClassifier #随机森林
from sklearn.neighbors import KNeighborsClassifier #k邻近算法
from sklearn.cluster import KMeans #聚类算法
from sklearn.linear_model import SGDClassifier
#模型评估
from sklearn.metrics import adjusted_rand_score
from sklearn.metrics import accuracy_score #分类报告
import pandas as pd
import time

x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=.2,random_state=22)
#建模分类算法，模型初始化
Classifiers=[["Random Forest",RandomForestClassifier()],
           ['KNN',KNeighborsClassifier(n_neighbors=10)],
           ['KMeans', KMeans(n_clusters =10)],
           ['SGDClassifier', SGDClassifier(max_iter = 5, tol=- np.infty,
random_state = 42)]]

Classify_result=[]
names=[]
prediction=[]
for name,classifier in Classifiers:
    classifier=classifier
    t1 = time.time()
    classifier.fit(x_train, y_train)
    y_pred=classifier.predict(x_test)
    t2 = time.time()
    time_diff = t2 - t1
    accuracy=accuracy_score(y_test,y_pred)
    score = adjusted_rand_score(y_test, y_pred)
    class_eva=pd.DataFrame([accuracy, score, time_diff])
    Classify_result.append(class_eva)
    name=pd.Series(name)
    names.append(name)
    y_pred=pd.Series(y_pred)
    prediction.append(y_pred)

names = pd.DataFrame(names)
names = names[0].tolist()
result = pd.concat(Classify_result, axis =1)
result.columns = names
result.index = ['accuracy', 'adjusted_rand_score', 'time_diff']
result

```