KNN-机器学习作业1

实验数据预处理

拿到semeion.data数据后，先读取到pdframe当中，然后将标签和数据分开，在这个数据集当中，后十列是标签说明该行是什么数字，前面256列是需要分类的数据。

```
X=pd.read_csv("g://semeion.data", delimiter = r"\s+",
                header=None )
y=pd.DataFrame(X.iloc[:,[256,257,258,259,260,261,262,263,264,265]])
X=X.drop([256,257,258,259,260,261,262,263,264,265], axis=1)
```

由于后十行是独热的标签，所以将其转换为数字标签作为标记。

```
for i in range(1593):
    for j in range(10):
        if X[j+256][i]==1:
            X[256][i]=j
            break
y=pd.DataFrame(X.iloc[:,[256]])
X=X.drop([256,257,258,259,260,261,262,263,264,265], axis=1)
```

拆分测试集和训练集

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.05,random_state=5
)
```

KNN算法，即K个最近邻居，所谓K最近邻，就是K个最近的邻居的意思，说的是每个样本都可以用它最接近的K个邻近值来代表。近邻算法就是将数据集合中每一个记录进行分类的方法。

定义KNN距离计算和排序的函数,这里的参数K意味着在比较距离之后选取前K名的标签。

```
def similarity(tests,train,labels,k):
    data_hang=train.shape[0]
    dis=np.tile(tests,(data_hang,1))-train
    q=np.sqrt((dis**2).sum(axis=1)).argsort()
    print(q)
    my_dict = {}
    for i in range(k):          ## 根据k来统计出现频率，样本类别
        votelabel=labels[q[i]]# q[i]是索引值,通过labels来获取对应标签
        my_dict[votelabel[0]] = my_dict.get(votelabel[0],0)+1    ## 统计每个标签的次
数
    sortclasscount=
            sorted(dict_list(my_dict),key=operator.itemgetter(1),reverse=True)
    return sortclasscount[0][0]                    ## 返回出现频次最高的类别
```

预测识别

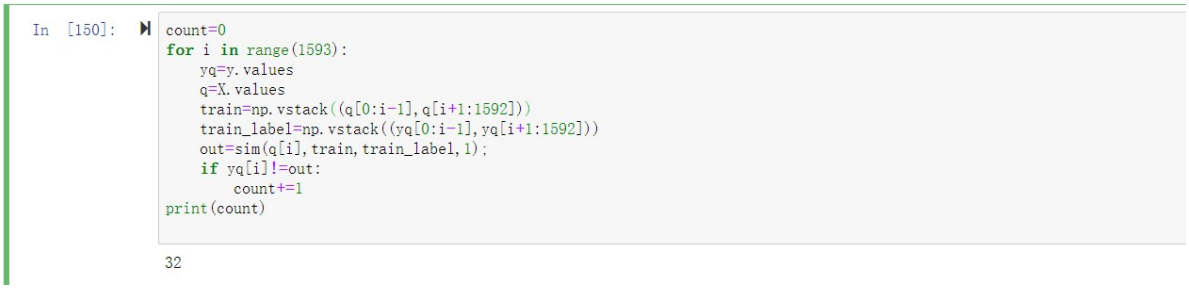
实验要求使用留一法进行识别精度的计算，那么我们就不再需要上面分割的测试集和训练集了。

```
count=0
for i in range(1593):
    yq=y.values
    q=X.values
    train=np.vstack((q[0:i-1],q[i+1:1592]))
    train_label=np.vstack((yq[0:i-1],yq[i+1:1592]))
    out=similarity(q[i],train,train_label,1);
    if yq[i]!=out:
        count+=1
print(count)
```

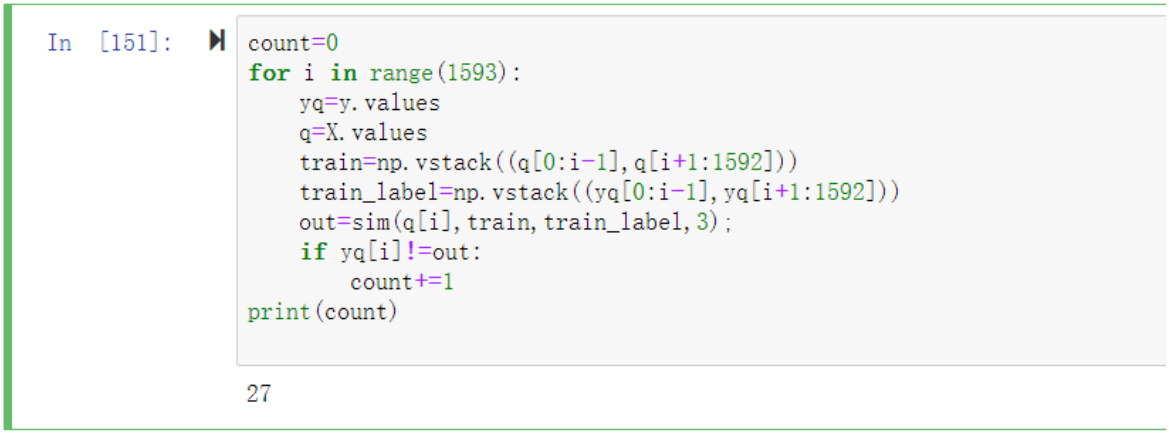
将原先的`pdframe`转换成`numpy`进行计算，对训练集的所有数据都去和想要识别的数据进行比对，得出最终标签。

应用留一法，将一个需要识别的数据和剩下的所有数据进行距离比较，然后按照相似度距离进行排序，得出最相似的标签获得预测结果。

我们在识别之后可以将所得到的预测结果和实际标签对比，查看预测是否正确。这就是留一法的交叉验证。



选择K=1进行识别，有32个识别错误。



选择K=3进行识别，有27个识别错误。

In [152]:

```
count=0
for i in range(1593):
    yq=y.values
    q=X.values
    train=np.vstack((q[0:i-1],q[i+1:1592]))
    train_label=np.vstack((yq[0:i-1],yq[i+1:1592]))
    out=sim(q[i],train,train_label,5);
    if yq[i]!=out:
        count+=1
print(count)
```

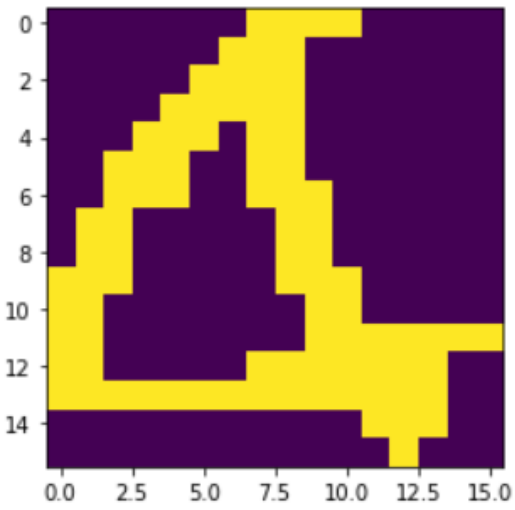
```
27
```

K=5识别，同样有27个错误。

识别可视化处理

```
import matplotlib.pyplot as plt
a=w.values
yq=y.values
plt.imshow(a[80].reshape(16,16))
out=sim(a[80],a,yq,5)
print(out)
print(yq[80])
```

```
4
[4]
```

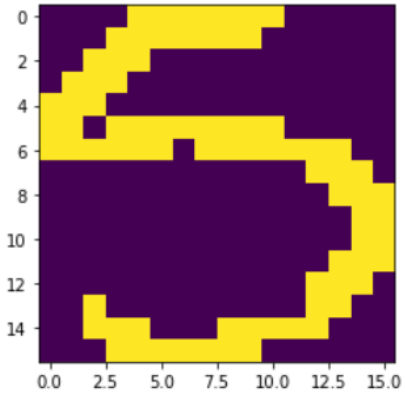


对手写数字识别进行可视化，print出预测结果、标签和数字图像，预测结果和标签都是4，而图像也可以依稀看出是4.

In [153]:

```
import matplotlib.pyplot as plt
a=w.values
yq=y.values
plt.imshow(a[1000].reshape(16,16))
out=sim(a[1000],a,yq,5)
print(out)
print(yq[1000])
```

```
5
[5]
```



换一个数据，可以看到识别依然是正确的。

weka包做分类识别

weka中选择加载数据，在加载之前我对数据进行了预处理，将最后的十列标签转换成为一列标签。在加载之后按照最后的标签做十分类，将其做好分类后选择KNN classfier，然后就可以直接点击识别了。

weka识别结果