

25 | KNN（下）：如何对手写数字进行识别？

2019-02-08 陈旻



讲述：陈旻

时长 08:33 大小 7.85M



今天我来带你进行 KNN 的实战。上节课，我讲了 KNN 实际上是计算待分类物体与其他物体之间的距离，然后通过统计最近的 K 个邻居的分类情况，来决定这个物体的分类情况。

这节课，我们先看下如何在 sklearn 中使用 KNN 算法，然后通过 sklearn 中自带的手写数字数据集来进行实战。

之前我还讲过 SVM、朴素贝叶斯和决策树分类，我们还可以用这个数据集来做下训练，对比下这四个分类器的训练结果。

如何在 sklearn 中使用 KNN

在 Python 的 sklearn 工具包中有 KNN 算法。KNN 既可以做分类器，也可以做回归。如果是做分类，你需要引用：

```
1 from sklearn.neighbors import KNeighborsClassifier
```

如果是做回归，你需要引用：

```
1 from sklearn.neighbors import KNeighborsRegressor
2
```

从名字上你也能看出来 Classifier 对应的是分类，Regressor 对应的是回归。一般来说如果一个算法有 Classifier 类，都能找到相应的 Regressor 类。比如在决策树分类中，你可以使用 DecisionTreeClassifier，也可以使用决策树来做回归 DecisionTreeRegressor。

好了，我们看下如何在 sklearn 中创建 KNN 分类器。

这里，我们使用构造函数 KNeighborsClassifier(n_neighbors=5, weights= 'uniform' , algorithm= 'auto' , leaf_size=30)，这里有几个比较主要的参数，我分别来讲解下：

1.n_neighbors：即 KNN 中的 K 值，代表的是邻居的数量。K 值如果比较小，会造成过拟合。如果 K 值比较大，无法将未知物体分类出来。一般我们使用默认值 5。

2.weights：是用来确定邻居的权重，有三种方式：

weights=uniform，代表所有邻居的权重相同；

weights=distance，代表权重是距离的倒数，即与距离成反比；

自定义函数，你可以自定义不同距离所对应的权重。大部分情况下不需要自己定义函数。

3.algorithm：用来规定计算邻居的方法，它有四种方式：

algorithm=auto，根据数据的情况自动选择适合的算法，默认情况选择 auto；

`algorithm=kd_tree`，也叫作 KD 树，是多维空间的数据结构，方便对关键数据进行检索，不过 KD 树适用于维度少的情况，一般维数不超过 20，如果维数大于 20 之后，效率反而会下降；

`algorithm=ball_tree`，也叫作球树，它和 KD 树一样都是多维空间的数据结果，不同于 KD 树，球树更适用于维度大的情况；

`algorithm=brute`，也叫作暴力搜索，它和 KD 树不同的地方是在于采用的是线性扫描，而不是通过构造树结构进行快速检索。当训练集大的时候，效率很低。

4.`leaf_size`：代表构造 KD 树或球树时的叶子数，默认是 30，调整 `leaf_size` 会影响到树的构造和搜索速度。

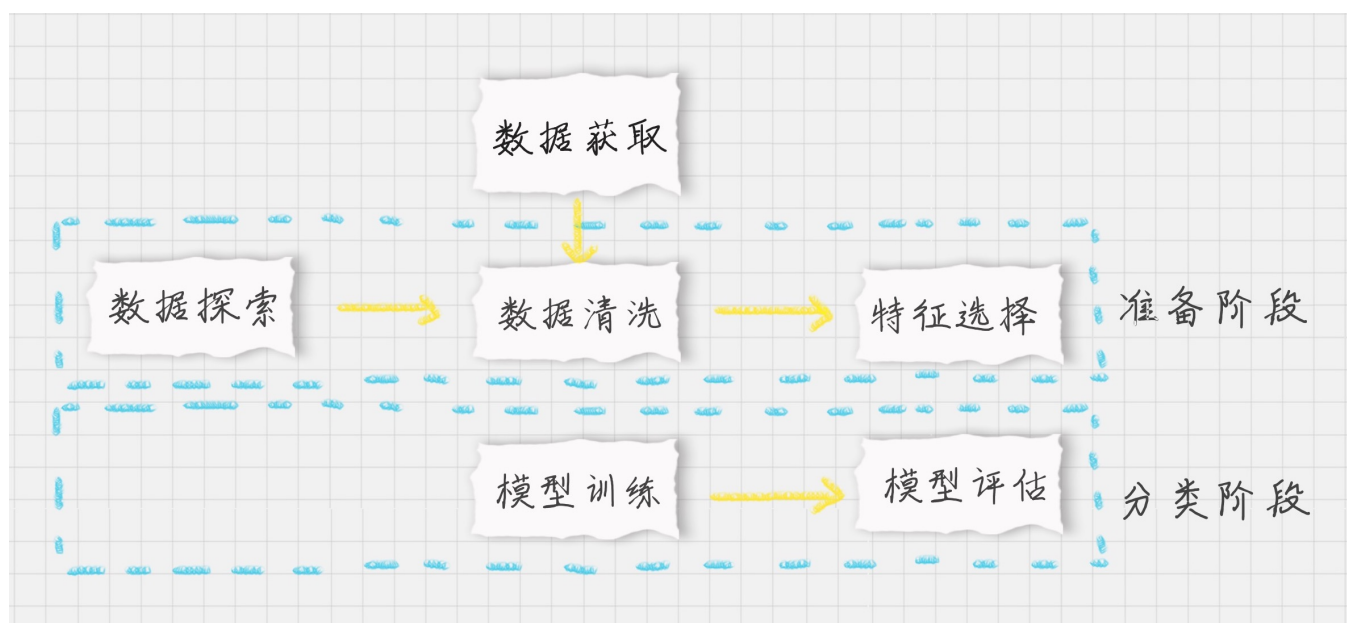
创建完 KNN 分类器之后，我们就可以输入训练集对它进行训练，这里我们使用 `fit()` 函数，传入训练集中的样本特征矩阵和分类标识，会自动得到训练好的 KNN 分类器。然后可以使用 `predict()` 函数来对结果进行预测，这里传入测试集的特征矩阵，可以得到测试集的预测分类结果。

如何用 KNN 对手写数字进行识别分类

手写数字数据集是个非常有名的用于图像识别的数据集。数字识别的过程就是将这些图片与分类结果 0-9 一一对应起来。完整的手写数字数据集 MNIST 里面包括了 60000 个训练样本，以及 10000 个测试样本。如果你学习深度学习的话，MNIST 基本上是你接触的第一个数据集。

今天我们用 `sklearn` 自带的手写数字数据集做 KNN 分类，你可以把这个数据集理解成一个简版的 MNIST 数据集，它只包括了 1797 幅数字图像，每幅图像大小是 8*8 像素。

好了，我们先来规划下整个 KNN 分类的流程：




整个训练过程基本上都会包括三个阶段：

1. 数据加载：我们可以直接从 sklearn 中加载自带的手写数字数据集；
2. 准备阶段：在这个阶段中，我们需要对数据集有个初步的了解，比如样本的个数、图像长什么样、识别结果是怎样的。你可以通过可视化的方式来查看图像的呈现。通过数据规范化可以让数据都在同一个数量级的维度。另外，因为训练集是图像，每幅图像是个 8×8 的矩阵，我们不需要对它进行特征选择，将全部的图像数据作为特征值矩阵即可；
3. 分类阶段：通过训练可以得到分类器，然后用测试集进行准确率的计算。


好了，按照上面的步骤，我们一起来实现下这个项目。

首先是加载数据和对数据的探索：

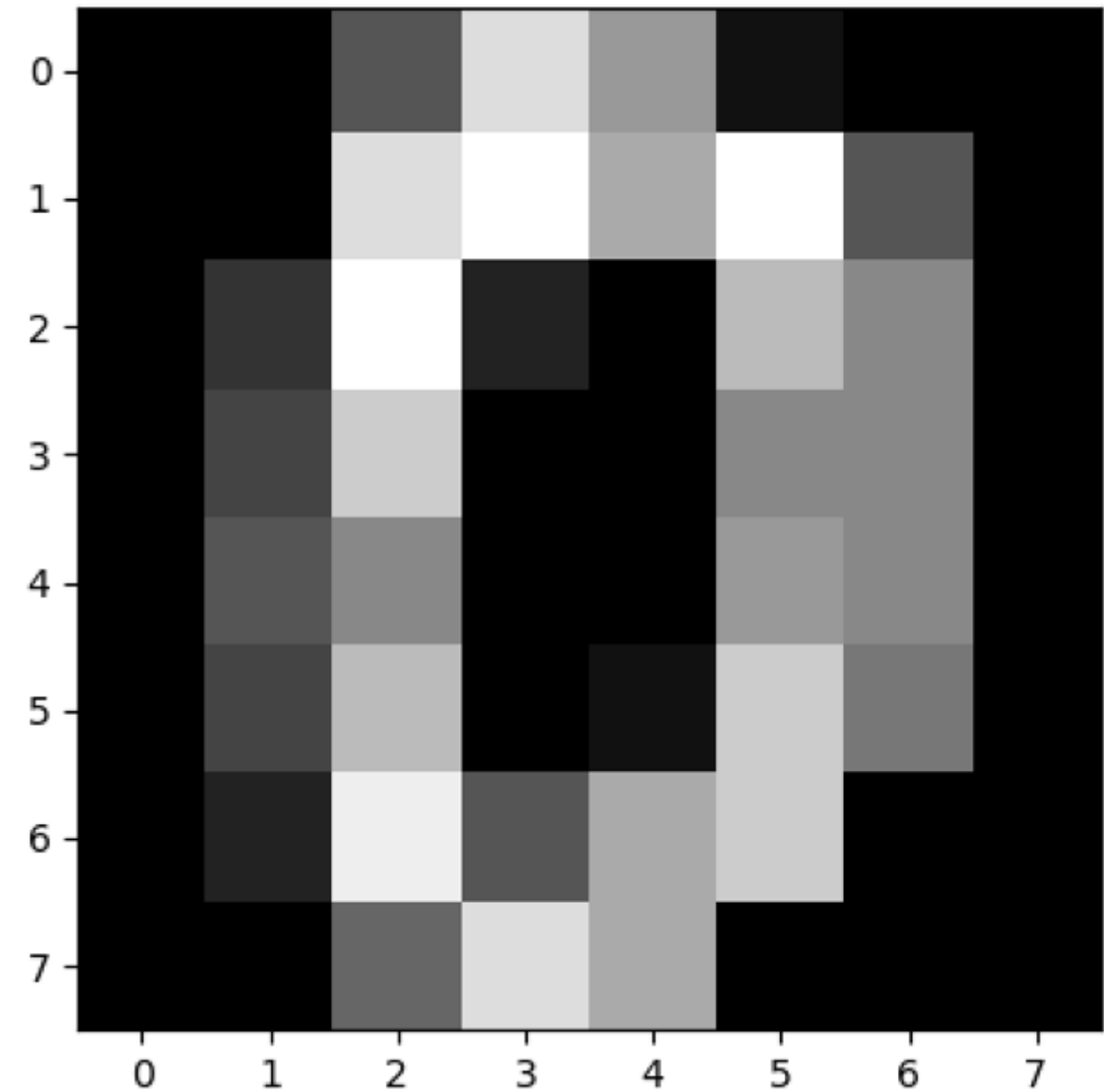
 复制代码

```
1 # 加载数据
2 digits = load_digits()
3 data = digits.data
4 # 数据探索
5 print(data.shape)
6 # 查看第一幅图像
7 print(digits.images[0])
8 # 第一幅图像代表的数字含义
9 print(digits.target[0])
10 # 将第一幅图像显示出来
11 plt.gray()
12 plt.imshow(digits.images[0])
13 plt.show()
```

运行结果：

 复制代码


```
1 (1797, 64)
2 [[ 0.  0.  5. 13.  9.  1.  0.  0.]
3  [ 0.  0. 13. 15. 10. 15.  5.  0.]
4  [ 0.  3. 15.  2.  0. 11.  8.  0.]
5  [ 0.  4. 12.  0.  0.  8.  8.  0.]
6  [ 0.  5.  8.  0.  0.  9.  8.  0.]
7  [ 0.  4. 11.  0.  1. 12.  7.  0.]
8  [ 0.  2. 14.  5. 10. 12.  0.  0.]
9  [ 0.  0.  6. 13. 10.  0.  0.  0.]]
10 0
```



我们对原始数据集中的第一幅进行数据可视化，可以看到图像是个 8*8 的像素矩阵，上面


这幅图像是一个“0”，从训练集的分类标注中我们也可以看到分类标注为“0”。

sklearn 自带的手写数字数据集一共包括了 1797 个样本，每幅图像都是 8*8 像素的矩阵。因为并没有专门的测试集，所以我们需要对数据集做划分，划分成训练集和测试集。因为 KNN 算法和距离定义相关，我们需要对数据进行规范化处理，采用 Z-Score 规范化，代码如下：

 复制代码

```
1 # 分割数据，将 25% 的数据作为测试集，其余作为训练集（你也可以指定其他比例的数据作为训练集）
2 train_x, test_x, train_y, test_y = train_test_split(data, digits.target, test_size=0.25,
3 # 采用 Z-Score 规范化
4 ss = preprocessing.StandardScaler()
5 train_ss_x = ss.fit_transform(train_x)
6 test_ss_x = ss.transform(test_x)
```

然后我们构造一个 KNN 分类器 knn，把训练集的数据传入构造好的 knn，并通过测试集进行结果预测，与测试集的结果进行对比，得到 KNN 分类器准确率，代码如下：

 复制代码

```
1 # 创建 KNN 分类器
2 knn = KNeighborsClassifier()
3 knn.fit(train_ss_x, train_y)
4 predict_y = knn.predict(test_ss_x)
5 print("KNN 准确率: %.4lf" % accuracy_score(predict_y, test_y))
```

运行结果：

 复制代码


```
1 KNN 准确率: 0.9756
```

好了，这样我们就构造好了一个 KNN 分类器。之前我们还讲过 SVM、朴素贝叶斯和决策树分类。我们用手写数字数据集一起来训练下这些分类器，然后对比下哪个分类器的效果更好。代码如下：

 复制代码

```
1 # 创建 SVM 分类器
2 svm = SVC()
3 svm.fit(train_ss_x, train_y)
4 predict_y=svm.predict(test_ss_x)
5 print('SVM 准确率: %0.4lf' % accuracy_score(predict_y, test_y))
6 # 采用 Min-Max 规范化
7 mm = preprocessing.MinMaxScaler()
8 train_mm_x = mm.fit_transform(train_x)
9 test_mm_x = mm.transform(test_x)
10 # 创建 Naive Bayes 分类器
11 mnb = MultinomialNB()
12 mnb.fit(train_mm_x, train_y)
13 predict_y = mnb.predict(test_mm_x)
14 print(" 多项式朴素贝叶斯准确率: %.4lf" % accuracy_score(predict_y, test_y))
15 # 创建 CART 决策树分类器
16 dtc = DecisionTreeClassifier()
17 dtc.fit(train_mm_x, train_y)
18 predict_y = dtc.predict(test_mm_x)
19 print("CART 决策树准确率: %.4lf" % accuracy_score(predict_y, test_y))
```

运行结果如下：

 复制代码

```
1 SVM 准确率: 0.9867
2 多项式朴素贝叶斯准确率: 0.8844
3 CART 决策树准确率: 0.8556
```


这里需要注意的是，我们在做多项式朴素贝叶斯分类的时候，传入的数据不能有负数。因为 Z-Score 会将数值规范化为一个标准的正态分布，即均值为 0，方差为 1，数值会包含负数。因此我们需要采用 Min-Max 规范化，将数据规范化到 [0,1] 范围内。

好了，我们整理下这 4 个分类器的结果。

分类器	准确率	排名
SVM	0.9867	1
KNN	0.9756	2
多项式朴素贝叶斯	0.8844	3
CART决策树	0.8556	4

你能看出来 KNN 的准确率还是不错的，和 SVM 不相上下。

你可以自己跑一遍整个代码，在运行前还需要 import 相关的工具包（下面的这些工具包你都会用到，所以都需要引用）：

 复制代码

```
1 from sklearn.model_selection import train_test_split
2 from sklearn import preprocessing
3 from sklearn.metrics import accuracy_score
4 from sklearn.datasets import load_digits
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.svm import SVC
7 from sklearn.naive_bayes import MultinomialNB
8 from sklearn.tree import DecisionTreeClassifier
9 import matplotlib.pyplot as plt
```

代码中，我使用了 train_test_split 做数据集的拆分，使用 matplotlib.pyplot 工具包显示图像，使用 accuracy_score 进行分类器准确率的计算，使用 preprocessing 中的 StandardScaler 和 MinMaxScaler 做数据的规范化。

完整的代码你可以从[GitHub](#)上下载。

总结

今天我带你一起做了手写数字分类识别的实战，分别用 KNN、SVM、朴素贝叶斯和决策树做分类器，并统计了四个分类器的准确率。在这个过程中你应该对数据探索、数据可视化、数据规范化、模型训练和结果评估的使用过程有了一定的体会。在数据量不大的情况下，使用 sklearn 还是方便的。

如果数据量很大，比如 MNIST 数据集中的 6 万个训练数据和 1 万个测试数据，那么采用深度学习 + GPU 运算的方式会更适合。因为深度学习的特点就是需要大量并行的重复计算，GPU 最擅长的就是做大量的并行计算。



最后留两道思考题吧，请你说说项目中 KNN 分类器的常用构造参数，功能函数都有哪些，以及你对 KNN 使用的理解？如果把 KNN 中的 K 值设置为 200，数据集还是 sklearn 中的手写数字数据集，再跑一遍程序，看看分类器的准确率是多少？

欢迎在评论区与我分享你的答案，也欢迎点击“请朋友读”，把这篇文章分享给你的朋友或者同事。

数据分析实战 45 讲

即学即用的数据分析入门课

陈旻

清华大学计算机博士



新版升级：点击「👤请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得转载

上一篇 24 | KNN (上) : 如何根据打斗和接吻次数来划分电影类型？

下一篇 26 | K-Means (上) : 如何给20支亚洲球队做聚类？

精选留言 (11)

写留言



third

2019-02-18

2

KNN常用的构造参数

`KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=n_neighbors)`是邻居的数目

`weights`是权重...

展开



FORWARD-M...

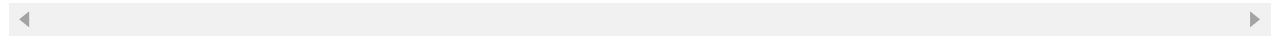
2019-02-16

1

`train_x`与`train_y`都是训练集？

展开 ▾

编辑回复: 对 训练集的特征矩阵和分类结果。对应test_x和test_y是测试集的特征矩阵和分类结果。



JingZ

2019-02-15

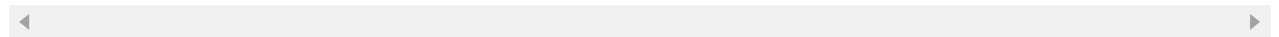
👍 1

#knn 将K值调为200，准确率变为0.8489了，相比较默认K=5的准确率 0.9756，下降13%

```
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt...
```

展开 ▾

编辑回复: 对的K值大未必好



Lee

2019-02-14

👍 1

KNN 中的 K 值设置为 200，KNN 准确率: 0.8489，k值过大，导致部分未知物体没有分类出来，所以准确率下降了



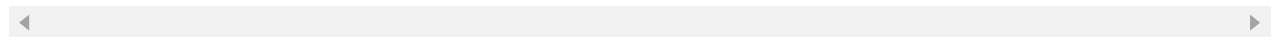
牛奶布丁

2019-02-13

👍 1

老师，为什么做多项式朴素贝叶斯分类的时候，传入的数据不能有负数呢，之前老师讲文本分类的时候好像没有提到这一点？

编辑回复: 多项式朴素贝叶斯实际上是符合多项式分布，不会存在负数。而高斯朴素贝叶斯呈现的是高斯分布，也就是正态分布，比如均值为0，方差为1的标准正态分布，可以存在负数。



不做键盘侠

2019-02-08

👍 1

为什么test只需要使用transform就可以了？`test_ss_x = ss.transform(test_x)`

展开 ▾

编辑回复: 一个很好的问题。我在train的时候用到了：`train_ss_x = ss.fit_transform(train_x)`

实际上：`fit_transform`是`fit`和`transform`两个函数都执行一次。所以`ss`是进行了`fit`拟合的。只有在`fit`拟合之后，才能进行`transform`

在进行test的时候，我们已经在train的时候`fit`过了，所以直接`transform`即可。

另外，如果我们没有`fit`，直接进行`transform`会报错，因为需要先`fit`拟合，才可以进行`transform`。



fancy

2019-03-03



1. KNN分类器的常用构造参数：

`n_neighbors := k = 5`(默认下)

`weights = 'uniform'/'distance'/'自定义函数`

`algorithm='auto'/'ball_tree'/'kd_tree'/'brute'`

`leaf_size...`

展开 ▾



mickey

2019-02-27



A1：

参数

1. `n_neighbors`

KNN中的K值，代表的是邻居的数量

K值如果比较小，会造成过拟合...

展开 ▾



Destroy_

2019-02-22



```
from sklearn.metrics import accuracy_score
```

```
# 创建KNN分类器
```

```
knn = KNeighborsClassifier(n_neighbors=200)
```

```
knn.fit(train_ss_x, train_y)
```

```
predict_y = knn.predict(test_ss_x)...
```

展开 ▾



王彬成

2019-02-22



1、项目中 KNN 分类器的常用构造参数，功能函数都有哪些

1) n_neighbors : 即 KNN 中的 K 值，代表的是邻居的数量。

2) weights : 是用来确定邻居的权重，有三种方式：

weights=uniform，代表所有邻居的权重相同；

weights=distance，代表权重是距离的倒数，即与距离成反比；...

展开 ∨



从未在此

2019-02-12



那个标准化函数已经在训练集上拟合并产生了平均值和标准差。所以测试集用同样的标准直接拿来用就行了