项目3

基于 K-Means 算法的应用实践

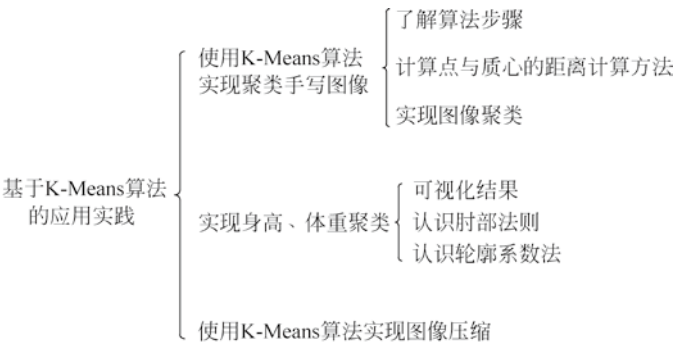
项目导读

在机器学习中，经常需要确定目标所属的类别。例如，要判定一个动物是狗、猫还是其他动物。解决这类问题的办法是先提供一些含有各种动物的数据集，让算法得到充分的训练学习，然后根据学习得到的先验信息对一个动物的类型做出判断。这种做法称为有监督学习，它分为训练和预测两个阶段。在训练阶段，使用大量的样本进行学习，得到一个判定动物类型的模型。在预测阶段，给出一张含有动物的图像，就可以使用这个模型预测出它的类别。

学习目标

- 掌握图像聚类算法 K-Means 的含义。
- 理解身高、体重聚类，确定 *K* 值。
- 掌握用 K-Means 算法压缩图像的方法。

知识导图



任务 3-1　使用 K-Means 算法实现聚类手写图像

任务描述

本任务学习聚类算法的基本原理、基本步骤，并使用 Sklearn 中的 K-Means 算法实现聚类手写数字图像。

任务目标

熟练使用 Sklearn 中的 K-Means 算法来完成一些图像聚类的操作。

知识准备

1. 聚类算法

与分类问题相似，聚类算法也需要确定一个物体的类别，不同的是，它没有事先定义类别，需要用户想办法把一批样本分开，形成多个类别。聚类算法需要保证每一个类中的样本之间是相似的，而不同类的样本之间是不同的。在这里，类型被称为“簇”（cluster）。例如有一群动物，事先没有说明有哪些动物，也没有一个训练好的判定各种动物的模型，聚类算法要自动将这群动物进行归类。这里没有统一的、确定的划分标准，可能有人将颜色相似的动物归在一起，有人将形状相似的动物归在一起。聚类算法是一种无监督学习，没有训练过程，这是和分类算法最本质的区别。算法要根据自己定义的规则，将相似的样本划分在一起，不相似的样本分成不同的类。

聚类就是按照某个特定标准把一个数据集分割成不同的类或簇，如图 3-1 所示，使同一个簇内的数据对象的相似性尽可能大，同时不同簇间的数据对象的差异性尽可能地大，本质上是集合划分问题。聚类算法的核心是如何定义簇，要求簇内的样本尽可能相似。簇的划分通常是根据簇内样本之间的距离或样本点在数据空间中的密度来确定。

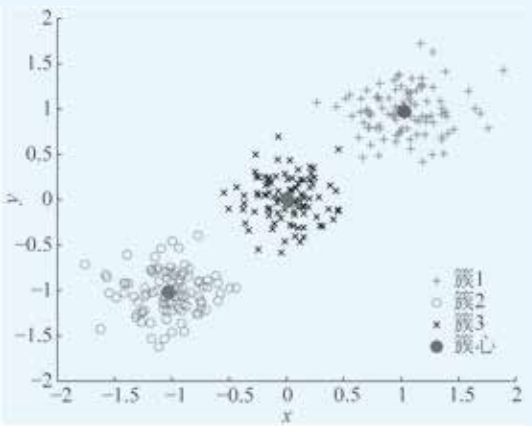


图 3-1　聚类划分结果

在生物学中，聚类可以用来发现具有类似表达模式的基因群组。在日常的网上购物中，聚类可以用来定位一个用户可能感兴趣的产品。在市场

营销中，聚类可以用来发现相似用户的分组。

2. K-Means 算法

K-Means 算法是机器学习中常用的无监督的聚类算法。对于给定的数据集，可以通过算法划分成 K 个不同的簇，且每个簇的中心（质心）可以通过属于该簇所有点坐标的平均或者加权平均计算而成。簇个数 K 是用户指定的，每一个簇通过其质心来描述。以距离作为数据对象间相似性度量的标准，即数据对象间距离越小，它们的相似性越高，越有可能在同一个簇中。K-Means 算法通常采用欧式距离来计算数据对象间的距离。

K-Means 算法在没有监督的情况会从经验中学习，性能可以通过一定的指标衡量。该算法的原理相对简单，可解释性好，收敛速度快，调参也只需要改变簇个数 K 一个参数。初始值的确定对于 K-Means 算法的结果影响较大，可能每次聚类的结果并不完全一致，也可能只是局部最优而不是全局最优。尤其是在两个簇距离过近时，影响较大。

任务实施

本任务将介绍 K-Means 的定义算法实现的步骤、质心的计算，并使用 K-Means 算法来实现二维图像聚类。

步骤 1　了解算法步骤

K-Means 算法的步骤如下。

（1）适当选择 K 个簇的初始质心。

（2）在第 k 次迭代中，对任意一个样本，求其到 K 个质心的距离，将该样本归到距离最短的质心所在的簇。

（3）利用均值等方法更新该类的质心值。

（4）对于所有的 K 个簇质心，如果利用（2）和（3）的迭代更新后值保持不变，则迭代结束，否则继续迭代。

步骤 2　计算点与质心的距离计算方法

K-Means 算法是基于质心或基于距离的算法，根据每个点到质心的距离分别计算出属于哪个簇。K-Means 算法的主要目标是计算出各个点到各自质心距离之和的最小值。计算两点之间的距离有很多公式，本项目以欧氏距离计算公式为例，如式（3-1）所示。

$$d=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \tag{3-1}$$

步骤 3　实现图像聚类

图像聚类可以用于划分图像数据集，并且发现同类图像的特征。接下来我们使用 Sklearn 中的 K-Means 算法来聚类手写数字图像，使用的数据集是 Sklearn 自带的手写数据库，其中包含了 0～9 的手写数字，一共 10 个类 1797 张图片，每张图片的分辨率是 8 像素 × 8 像素。图 3-2 表示数字 0 的手写数字图像，图 3-3 表示数字 9 的手写数字图像。

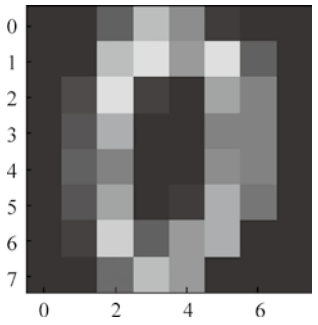


图 3-2　数字 0 的手写数字图像

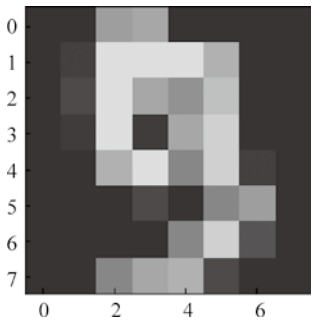


图 3-3　数字 9 的手写数字图像

【代码 3-1】

```
# 第一步：装载数据
from sklearn.datasets import load_digits
digits = load_digits()
X = digits.data;
y = digits.target
# 第二步：引入时间计时
import time
start = time.process_time()
# 第三步：进行 K-Means 聚类
from sklearn.cluster import KMeans
KM = KMeans(n_clusters=10)
c = KM.fit_predict(X)
end = time.process_time()
print('Time is %.3f' % (end - start))
# 第四步：计算聚类结果
import numpy as np
y_predict = np.zeros_like(c)
from scipy.stats import mode
for i in range(10):
      mask = (c == i)
      y_predict[mask] = mode(y[mask])[0]
KM.cluster_centers_.shape
# 第五步：显示聚类中心
```

```
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = [u'SimHei']
plt.rcParams['axes.unicode_minus'] = False
fig, ax = plt.subplots(2, 5, figsize = (8, 3))
centers = KM.cluster_centers_.reshape(10, 8, 8)
for axi, center in zip(ax.flat, centers):
    axi.set(xticks = [], yticks = [])
    axi.imshow(center, cmap = plt.cm.binary)
# 第六步：输出聚类准确率
from sklearn.metrics import accuracy_score
print('聚类准确率为：%.4f %%' % accuracy_score(y, y_predict))
```

输出结果如下，显示的聚类中心如图 3-4 所示。

```
Time is 0.406
聚类准确率为：0.7919 %
```

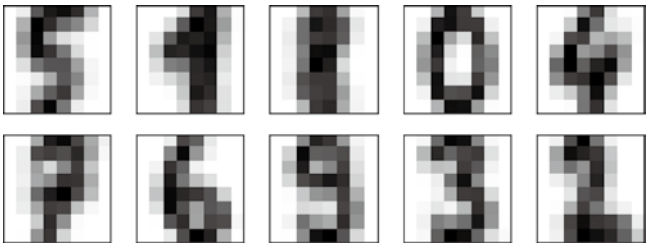


图 3-4　聚类中心

代码 3-1 采用 K-Means 算法对手写数字进行识别，其中，因为我们已经知道这些数字图片一共有 10 个类，所以 K-Means 函数中的 n_clusters 设置为 10。图 3-4 为聚类中心，一共 10 个，可以看见数字 0～9。

项目 2 中学习的 PCA 可以用于数据降维，提出数据中的主要信息，我们通过 PCA 对手写字符数据进行降维处理，观察降维后的聚类结果，使用的程序如下。

【代码 3-2】

```
# 第一步：装载数据
from sklearn.datasets import load_digits
digits = load_digits()
X = digits.data;
y = digits.target
# 第二步：引入时间计时
import time
from sklearn.decomposition import PCA
start = time.process_time()
# 第三步：PCA 降维
pca = PCA(n_components=10)
```

```
pca.fit(X)
X_reduction = pca.transform(X)
print(X_reduction.shape)
end = time.process_time()
print('PCA Time is %.3f' % (end - start))
# 第四步：K-Means 聚类
from sklearn.cluster import KMeans
start = time.process_time()
KM = KMeans(n_clusters = 10)
c = KM.fit_predict(X_reduction)
end = time.process_time()
print('K-Means Time is %.3f' % (end - start))
# 第五步：计算聚类结果
from scipy.stats import mode
import numpy as np
y_predict = np.zeros_like(c)
for i in range(10):
      mask = (c == i)
      y_predict[mask] = mode(y[mask])[0]
# 第六步：输出聚类准确率
from sklearn.metrics import accuracy_score
print(' 聚类准确率为：%.4f %%' % accuracy_score(y, y_predict))
```

输出结果如下。

```
(1797, 10)
PCA Time is 0.094
K-Means Time is 0.406
聚类准确率为：0.7869 %
```

通过 PCA 提取特征后再进行聚类，时间比没有 PCA 降维处理的时间还要多，这是因为 PCA 降维处理部分花了部分时间。

在上一个数据集中，数据量只有 1797。接下来，我们通过对 MNIST 数据集进行测试，来验证 PCA 对聚类的影响。MNIST 数据库是手写数字的数据集，它有 60000 个训练样本和 1000 个测试样本，每个图像大小为 28 像素 × 28 像素。

先观察不使用 PCA 降维的聚类结果，使用的程序如下。

【代码 3-3】

```
# 第一步：装载数据
from sklearn.datasets import load_digits
from sklearn.datasets import fetch_openml
digits = fetch_openml('mnist_784')
X = digits.data
y = digits.target
```

```
# 第二步：引入时间计时
import time
start = time.process_time()
# 第三步：K-Means 聚类
from sklearn.cluster import KMeans
KM = KMeans(n_clusters = 10)
c = KM.fit_predict(X)
end = time.process_time()
print('Time is %.3f' % (end - start))
# 第四步：计算聚类结果
from scipy.stats import mode
import numpy as np
y_predict = np.zeros_like(c)
for i in range(10):
      mask = (c == i)
      y_predict[mask] = mode(y[mask])[0]
# 第五步：显示聚类结果
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = [u'SimHei']
plt.rcParams['axes.unicode_minus'] = False
fig, ax = plt.subplots(2, 5, figsize = (8, 3))
centers = KM.cluster_centers_.reshape(10, 28, 28)
for axi, center in zip(ax.flat, centers):
    axi.set(xticks = [], yticks = [])
    axi.imshow(center, cmap = plt.cm.binary)
import pandas as pd
y = y.astype(np.int8)
y_predict = pd.DataFrame(y_predict)
y_predict = y_predict.astype(np.int8)
# 第六步：输出聚类准确率
from sklearn.metrics import accuracy_score
print('聚类准确率为：%.4f %%' % accuracy_score(y, y_predict))
```

聚类结果如图 3-5 所示。

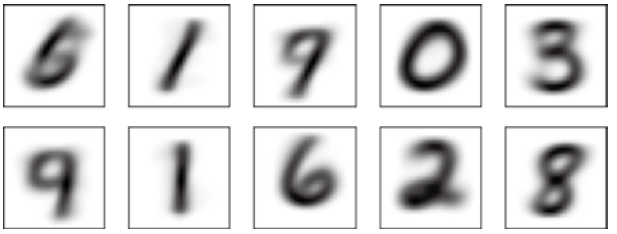


图 3-5　聚类结果

```
Time is 169.375
聚类准确率为：0.5851 %
```

代码 3-3 通过 K-Means 算法对 MNIST 数据集进行聚类，MNIST 数据集中包含了 60000 张图片，数据量相对较大，其中 K-Means 函数中的 n_clusters 设置为 10，图 3-5 为聚类中心，可以看见 0～9 十个数字。

使用 PCA 对数据先进行特征提取，然后再进行聚类，使用的程序如代码 3-4。

【代码 3-4】

```
# 第一步：装载数据
from sklearn.datasets import load_digits
from sklearn.datasets import fetch_openml
digits = fetch_openml('mnist_784')
X = digits.data
y = digits.target
# 第二步：PCA 降维
import time
start = time.process_time()
from sklearn.decomposition import PCA
pca = PCA(n_components = 64)
pca.fit(X)
X_reduction = pca.transform(X)
print(X_reduction.shape)

from sklearn.cluster import KMeans
KM = KMeans(n_clusters = 10)
c = KM.fit_predict(X_reduction)
end = time.process_time()
print('Time is %.3f' % (end - start))
# 第三步：计算聚类结果
from scipy.stats import mode
import numpy as np
y_predict = np.zeros_like(c)
for i in range(10):
      mask = (c == i)
      y_predict[mask] = mode(y[mask])[0]
# 第四步：显示聚类结果
from sklearn.metrics import accuracy_score
y = y.astype(np.int8)
import pandas as pd
y_predict = pd.DataFrame(y_predict)
y_predict = y_predict.astype(np.int8)
accuracy_score(y, y_predict)
```

输出结果如下。

```
Time is 28.719
聚类准确率为：0.5847 %
```

在没有使用 PCA 进行降维时，数据的维度是 784；使用 PCA 降维后，数据的维度是 64。在聚类的准确率基本相同的情况下，降维后聚类用时 28.719s；没有降维时，用时 169.375s。可见，降维后聚类用时只是之前的 1/6 左右，说明 PCA 可以有效地提高效率。

任务 3-2　实现身高、体重聚类

任务描述

使用身高、体重聚类的例子，来学习如何确定 K-Means 算法中的 *K* 值。

任务目标

熟练掌握确定 *K* 值的方法。

任务实施

在本任务中，将介绍根据学生的身高、体重，使用 K-Means 算法对学生的这些数据进行聚类，通过肘部法则与轮廓系数法观察、分析聚类结果，确定合适的 *K* 值。

步骤 1　可视化结果

对于 K-Means 算法而言，确定 *K* 值至关重要。下面介绍两种常用的确定 *K* 值的方法：肘部法则和轮廓系数法。表 3-1 是班级学生的身高、体重，通过 K-Means 算法来对这些数据进行聚类。接下来先通过代码 3-5 对数据进行二维可视化来观察数据，以确定数据划分最佳的类别数。

表 3-1　身高、体重数据集

序号	身高 /cm	体重 /kg	序号	身高 /cm	体重 /kg
1	159.3	60.5	11	185.3	81.0
2	160.3	60.2	12	161.3	62.2
3	165.2	60.4	13	164.2	64.4
4	162.5	62.1	14	163.5	65.1
5	175.4	75.1	15	176.4	75.1
6	178.6	75.3	16	185.6	85.3
7	177.1	78.2	17	175.1	78.0
8	176.4	75.4	18	168.4	60.4
9	189.4	85.8	19	187.4	80.8
10	176.2	73.7	20	185.2	79.7

【代码 3-5】

```
# 第一步：装载数据
import numpy as np
x = np.array([
              159.3,160.3,165.2,162.5,
              175.4,178.6,177.1,176.4,
              189.4,176.2,185.3,161.3,
              164.2,163.5,176.4,185.6,
              175.1,168.4,187.4,185.2
])
y = np.array([
              60.5,60.2,60.4,62.1,
              75.1,75.3,78.2,75.4,
              85.8, 73.7, 81,62.2,
              64.4, 65.1, 75.1,85.3,
              78,60.4,80.8,79.7
])
# 第二步：显示二维数据
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = [u'SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.scatter(x, y)
plt.xlim([150,190])
plt.ylim([45,90])
plt.xlabel('身高/cm')
plt.ylabel('体重/kg')
plt.show()
```

代码 3-5 将二维数据进行了可视化，便于对数据的观察。在图 3-6 中，数据分在三个簇中，因此可以看出 K 值取 3 最好，但是在实际的应用中，高维的数据不方便进行可视化。接下来将介绍如何通过肘部法则和轮廓系数法来确定 K 的取值。

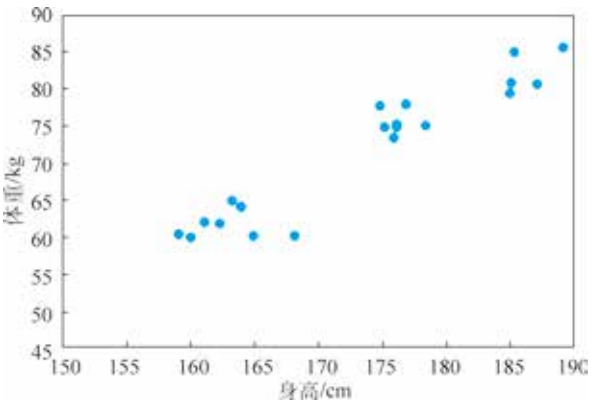


图 3-6　调整 K 值后的结果