

# 《基于 MindSpore 的 K-Means 算法实践》



华为技术有限公司

**版权所有 © 华为技术有限公司 2024 保留一切权利。**

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

#### **商标声明**



HUAWEI 和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

#### **注意**

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

### **华为技术有限公司**

地址：	深圳市龙岗区坂田华为总部办公楼	邮编：518129
网址：	<a href="http://e.huawei.com">http://e.huawei.com</a>	



# 目录

---

<b>1 K-Means 算法实践 .....</b>	<b>2</b>
1.1 实验介绍 .....	2
1.1.1 简介 .....	2
1.1.2 实验目的 .....	3
1.1.3 预备知识 .....	3
1.2 实验环境搭建 .....	3
1.3 实验过程 .....	4
1.3.1 数据准备 .....	4
1.3.2 导入 MindSpore 模块和辅助模块 .....	4
1.3.3 加载并处理数据 .....	5
1.3.4 聚类运算 .....	5
1.3.5 K-Means 聚类结果可视化 .....	7
1.4 实验小结 .....	7

# 1 K-Means 算法实践

## 1.1 实验介绍

### 1.1.1 简介

K-Means 算法是最为经典的基于划分的聚类方法，是十大经典数据挖掘算法之一。K-Means 算法的基本思想是：以空间中  $k$  个点为中心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。K-Means 算法接受参数  $k$ ，然后将事先输入的  $n$  个数据对象划分为  $k$  个聚类以便使得所获得的聚类满足同一聚类中的对象相似度较高，而不同聚类中的对象相似度较小。聚类相似度是利用各聚类中对象的均值所获得一个“中心对象”来进行计算的。

聚类算法有很多种（几十种），K-Means 是聚类算法中的最常用的一种，算法最大的特点是简单，好理解，运算速度快，但是只能应用于连续型的数据，并且一定要在聚类前需要手工指定要分成几类。

下面，我们描述一下 K-Means 算法的过程，为了尽量不用数学符号，所以描述的不是很严谨，大概就是这个意思，“物以类聚、人以群分”：

首先输入  $K$  的值，即我们希望将数据集经过聚类得到  $K$  个分组。

从数据集中随机选择  $K$  个数据点作为初始数据（质心，Centroid）也可选择非样本点。

对集合中每一个样本，分别计算其与  $K$  个质心的距离（这里的距离一般取欧氏距离或余弦距离），找到离该点最近的质心，将它归属到对应的簇。

所有点都归属到簇之后， $M$  个样本就分为了  $K$  个簇，之后重新计算每个簇的质心（平均距离中心），将其定为新的质心。

如果新质心和老质心之间的距离小于某一个设置的阈值（表示重新计算的质心的位置变化不大，趋于稳定，或者说收敛），可以认为我们进行的聚类已经达到期望的结果，算法终止。

如果新质心和老质心距离变化很大，需要迭代 3~5 步骤。

K-Means 是个简单实用的聚类算法，这里对 K-Means 的优缺点做一个总结。

K-Means 的主要优点有：

- 原理比较简单，实现也是很容易，收敛速度快。

- 聚类效果较优。
- 算法的可解释度比较强。
- 主要需要调参的参数仅仅是簇数 k。

K-Means 的主要缺点有：

- K 值的选取不好把握
- 对于不是凸的数据集比较难收敛
- 如果各隐含类别的数据不平衡，比如各隐含类别的数据量严重失衡，或者各隐含类别的方差不同，则聚类效果不佳。
- 采用迭代方法，得到的结果只是局部最优。
- 对噪音和异常点比较的敏感。

### 1.1.2 实验目的

- 了解聚类算法和 K-Means 的基本概念；
- 了解如何使用 MindSpore 进行 K-Means 聚类实验。
- 了解全流程开发工具链 MindStudio 的使用；

### 1.1.3 预备知识

- 熟练使用 Python。
- 具备一定的机器学习理论知识，如 K-Means、监督学习、无监督学习，训练策略等。
- 了解并熟悉MindSpore AI计算框架，MindSpore官网：<https://www.mindspore.cn/>

## 1.2 实验环境搭建

本实验在华为云 ModelArts 平台进行，实验环境搭建操作请参考如下环境搭建手册。





## 1.3 实验过程

### 1.3.1 数据准备

Iris 数据集是模式识别最著名的数据集之一。数据集包含 3 类，每类 50 个实例，其中每个类都涉及一种鸢尾植物。本实验使用 iris 数据集，做一个降维数据集。

下载 Iris.data 数据文件，下载链接：<https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/MachineLearning/iris.data>

每个样本含有 4 个数值属性和一个类别属性：

4 个数值属性：sepal length；sepal width；petal length；petal width。

一个类别属性，属性值如下：

- Iris Setosa
- Iris Versicolour
- Iris Virginica

概括统计：

Attribute	Min	Max	Mean	SD	Class Correlation
sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

### 1.3.2 导入 MindSpore 模块和辅助模块

```
import os
import csv
import random
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import mindspore as ms
from mindspore import Tensor, nn
import mindspore.ops as ops

ms.set_context(mode=ms.PYNATIVE_MODE, device_target="Ascend")
```



### 1.3.3 加载并处理数据

读取 Iris 数据集 iris.data；数据集有 3 类样本共 150 条，将样本的 4 个属性作为自变量 X，将样本的 3 个类别映射为 {0, 1, 2}，作为因变量 Y。本实验为了让聚类结果更明显，取样本的后两个属性演示。

```
def create_dataset(data_path):
    with open(data_path) as csv_file:
        data = list(csv.reader(csv_file, delimiter=','))
    label_map = {
        'Iris-setosa': 0,
        'Iris-versicolor': 1,
        'Iris-virginica': 2
    }
    X = np.array([[float(x) for x in s[2:-1]] for s in data[:150]], np.float32)
    Y = np.array([label_map[s[-1]] for s in data[:150]], np.int32)
    return X, Y
```

设置 K 值为 3。iris 数据集有三类花，实际上是分类任务的数据集，所以堆的数量是确定的；迭代次数为 100。随机选取数据点准备进行聚类运算。

```
k=3
generations = 100
x,y = create_dataset('./iris.data')
num_pts = len(x)
num_feats = len(x[0])
data_points = Tensor(x)
cluster_labels = np.zeros(num_pts)
# 先随机选择 iris 数据集中的三个数据点作为每个堆的中心点
rand_starts = np.array([x[np.random.choice(len(x))] for _ in range(k)])
centroids = Tensor(rand_starts)
```

### 1.3.4 聚类运算

聚类运算，利用 MindSpore 提供的 Reshape，Tile，ReduceSum，Square，Argmin 等算子，通过矩阵运算的方式对输入样本数据进行计算，计算每个数据点到每个中心点的欧氏距离，这里是将数据点都放入矩阵，直接按矩阵进行运算。返回聚类后的数据。后续再进行迭代。

```
def calculate():
    reshape = ops.Reshape()
    tile = ops.Tile()
    reduce_sum = ops.ReduceSum(keep_dims=False)
    square = ops.Square()
    argmin = ops.Argmin()

    centroid_matrix = reshape(tile(centroids, (num_pts,1)), (num_pts, k, num_feats))
    point_matrix = reshape(tile(data_points, (1,k)), (num_pts, k, num_feats))
```



```
distances = reduce_sum(square(point_matrix-centroid_matrix),2)
# 分配时，以每个数据点最小距离为最接近的中心点
centroid_group = argmin(distances)

return centroid_group
```

通过分组求和、计算堆大小和求距离均值，计算三个堆的平均距离更新堆中新的中心点。

```
unsorted_segment_sum = ops.UnsortedSegmentSum()
ones_like = ops.OnesLike()
assign = ops.Assign()

def data_group_avg(group_ids, data):
    # 分组求和
    sum_total = unsorted_segment_sum(data, group_ids, 3)
    #计算堆大小
    num_total = unsorted_segment_sum(ones_like(data), group_ids, 3)
    #求距离均值
    avg_by_group = sum_total/num_total
    return avg_by_group
```

遍历循环训练，更新每组分类的中心点。

```
for i in range(generations):
    print('Calculating gen {}'.format(i))
    centroid_group = calculate()
    means = data_group_avg(centroid_group, data_points)
    centroids = assign(ms.Parameter(centroids, name='w'), means)
    cluster_labels = assign(ms.Parameter(Tensor(cluster_labels,ms.int64),name='w'), centroid_group)
    centroid_group_count = assign(ms.Parameter(Tensor(cluster_labels,ms.int64),name='w'),
    centroid_group).asnumpy()
    group_count = []
    for ix in range(k):
        group_count.append(np.sum(centroid_group_count==ix))
    print('Group counts: {}'.format(group_count))
```

输出准确率。将聚类结果和 iris 数据集中的标签进行对比。

```
centers, assignments = centroids, cluster_labels.asnumpy()

def most_common(my_list):
    return(max(set(my_list), key=my_list.count))

label0 = most_common(list(assignments[0:50]))
label1 = most_common(list(assignments[50:100]))
label2 = most_common(list(assignments[100:150]))
group0_count = np.sum(assignments[0:50]==label0)
group1_count = np.sum(assignments[50:100]==label1)
group2_count = np.sum(assignments[100:150]==label2)
accuracy = (group0_count + group1_count + group2_count)/150.
```

```
print('=====Accuracy: {:.2}====='.format(accuracy))
```

### 1.3.5 K-Means 聚类结果可视化

通过 matplotlib 模块将数据进行 K-Means 聚类后的结果进行可视化展示出来。红叉为多次迭代计算出的聚类中心。

```
reduced_data = x
reduced_centers = means
# 设置图例
symbols = ['o']
label_name = ['Setosa', 'Versicolour', 'Virginica']
for i in range(3):
    temp_group = reduced_data[(i*50):(50)*(i+1)]
    plt.plot(temp_group[:, 0], temp_group[:, 1], symbols[0], markersize=8, label=label_name[i])
# 绘图
plt.scatter(reduced_centers[:, 0].asnumpy(), reduced_centers[:, 1].asnumpy(),
            marker='x', s=169, linewidths=3,color='red', zorder=10)
plt.title('K-Means clustering on Iris Dataset\n'
          'Centroids are marked with white cross')
plt.legend(loc='lower right')
plt.show()
```

结果：



## 1.4 实验小结

本实验使用 MindSpore 实现了对鸢尾花数据集做聚类分析，k-Means 算法使用简单的迭代将数据集聚成 k 个类，迭代的核心步骤有：（1）更新聚簇中心；（2）更新聚簇标识。尽管它有不



少缺陷，但是其实现简单、可解释性好、伸缩性良好等优点使得它成为聚类中最常用的算法之一。