

記事

[Minoru Horita](#) · 2020年6月26日 6m read

アイリスデータセットのK平均ラスタリング

アイリスデータセットのK平均ラスタリング

みなさん、こんにちは。今回はアイリスデータセットでK平均アルゴリズムを使用します。

注意: Ubuntu 18.04, Apache Zeppelin 0.8.0, python 3.6.5で実行しました。

概要

K平均法は、ラスタリングの問題を解決する最も単純な教師なし学習アルゴリズムの1つです。このアルゴリズムは、同じグループ内のオブジェクト(グループはクラスターです)他のグループ内のオブジェクトよりも意味的に互いに類似するようにすべてのオブジェクトをグループ化します。

例えば、緑の芝生に赤いポピーのある画像があります。

K平均法はすべてのピクセルを2つのクラスターに分割します。

1番目のクラスターにはポピーのピクセルが含まれ、2番目のクラスターには芝生のピクセルが含まれます。

アイリスデータセットは、3種のアイリスの特徴をいくつか含むデータセットです。種には'Iris-setosa'、'Iris-versicolor'、'Iris-virginica'があります。

それぞれの種には5つの特徴(**びらの長さ**、 **びらの幅**、 **くく片の長さ**、 **くく片の幅**、 **種**)があります。

要件の確認

まず、すべての要件を確認しましょう。次のように、ターミナルに `which python3` を入力してください。

```
guardian@guardian:~$ which python3
/usr/bin/python3
guardian@guardian:~$ which python42
guardian@guardian:~$
```

Python 3がインストール

されている場合は、最初のパスが表示されます。

空の場合(' `which python42`' の出力のように表示される場合)、ターミナルに `which python3` を入力してください。

```
sudo apt-get update
```

```
sudo apt-get -y upgrade
```

```
sudo apt install python3
```

```
sudo apt-get install -y python3-pip
```

次に、pysparkをインストールします。

```
pip3 install pyspark
```

Spark インタプリタの設定で、zeppelin.pyspark.python を「which python3」で出力されたパスに変更してください。

```
zeppelin.pyspark.python
```

```
/usr/bin/python3
```

最後にSparkインタプリタで新しいメモを作成、以て新しい段落に切り付けて実行してください。

```
%pyspark  
  
import sys  
  
print(sys.version)
```

無問題なれば、Pythonのバージョンが表示されます。

```
%pyspark  
import sys  
print(sys.version)
```

```
3.6.5 (default, Apr 1 2018, 05:46:30)  
[GCC 7.3.0]
```

クラスタリング

InterSyst

ems IRISをセットアップし、ZeppelinおよびSparkと連携できるようにします。例えば、以前の [IRIS, Apache Zeppelin, Apache Spark 接続に関する記事](#) を参照してください。

[InterSystems](#)

まず、次のようにアイリスデータセットをロードします。

```
%pyspark
dataFrame=spark.read.format("com.intersystems.spark").\
option("url", "IRIS://localhost:51773/NEWSAMPLE").option("user", "dev").\
option("password", "123").\
option("dbtable", "DataMining.IrisDataset").load()

dataFrame.show()
```

```
+---+-----+-----+-----+-----+-----+
| ID|PetalLength|PetalWidth|SepalLength|SepalWidth| Species|
+---+-----+-----+-----+-----+-----+
| 1| 1.4| 0.2| 5.1| 3.5|Iris-setosa|
| 2| 1.4| 0.2| 4.9| 3.0|Iris-setosa|
| 3| 1.3| 0.2| 4.7| 3.2|Iris-setosa|
| 4| 1.5| 0.2| 4.6| 3.1|Iris-setosa|
| 5| 1.4| 0.2| 5.0| 3.6|Iris-setosa|
| 6| 1.7| 0.4| 5.4| 3.9|Iris-setosa|
| 7| 1.4| 0.3| 4.6| 3.4|Iris-setosa|
| 8| 1.5| 0.2| 5.0| 3.4|Iris-setosa|
| 9| 1.4| 0.2| 4.4| 2.9|Iris-setosa|
|10| 1.5| 0.1| 4.9| 3.1|Iris-setosa|
|11| 1.5| 0.2| 5.4| 3.7|Iris-setosa|
|12| 1.6| 0.2| 4.8| 3.4|Iris-setosa|
|13| 1.4| 0.1| 4.8| 3.0|Iris-setosa|
|14| 1.1| 0.1| 4.3| 3.0|Iris-setosa|
|15| 1.2| 0.2| 5.0| 3.4|Iris-setosa|
```

Speciesの内容を確認するには、以を新しい段落にり付けて実行してください。

```
%pyspark

dataFrame.select("Species").show(150)
```

先に進む前に、データの内容を検討することをお勧めします。
新しい段落を追加し、以をり付けて実行してください。

```
%pyspark

z.show(dataFrame)
```

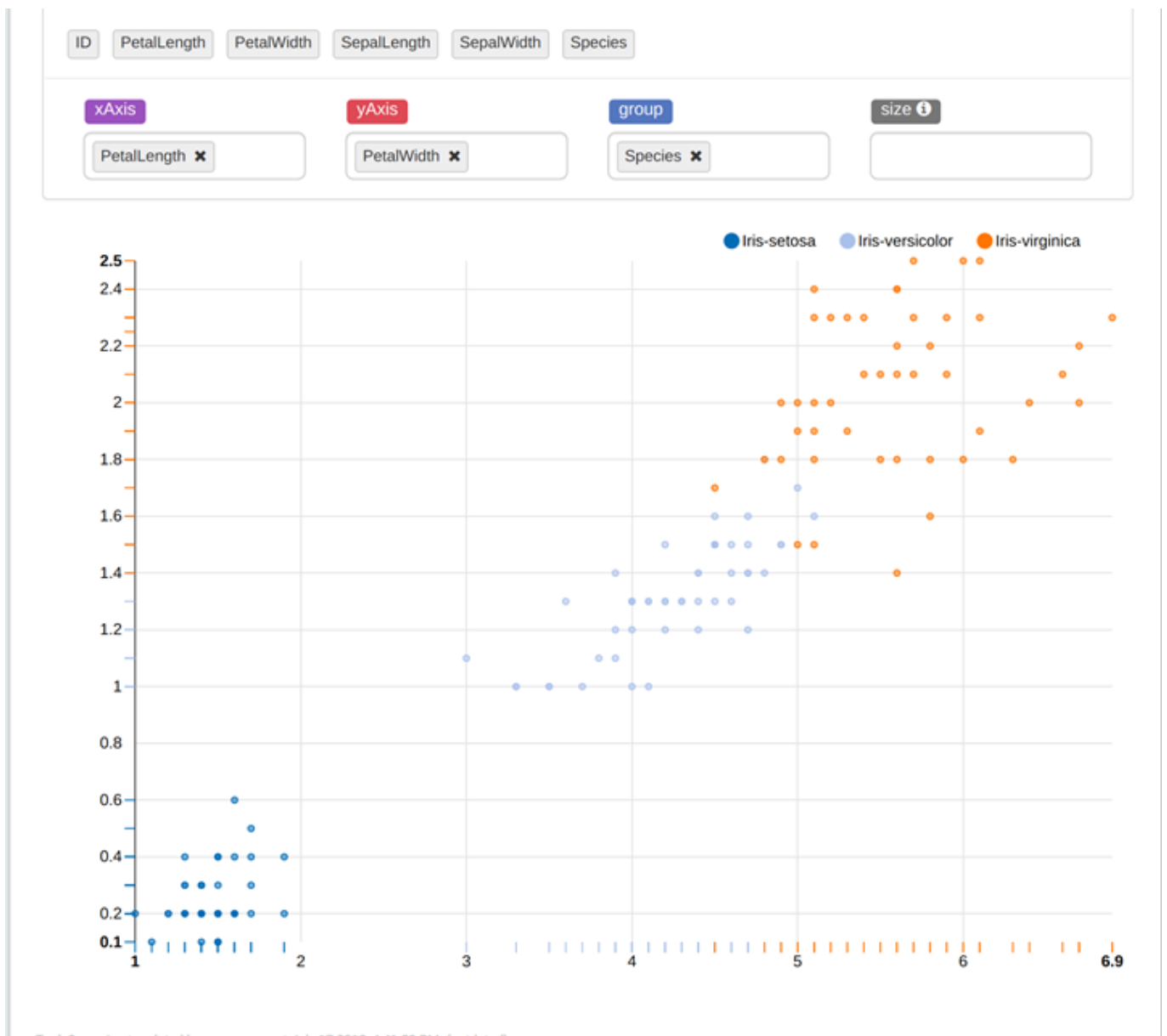
以のように、テーブルには5つの特徴である PetalLength(花びらの長さ)、PetalWidth(花びらの幅)、SepalLength(萼く片の長さ)、SepalWidth(萼く片の幅)、Species(種) があります。

toyspark
z.show(dataFrame)

settings

ID	Petal.Length	Petal.Width	Sepal.Length	Sepal.Width	Species
1	1.4	0.2	5.1	3.5	Iris-setosa
2	1.4	0.2	4.9	3	Iris-setosa
3	1.3	0.2	4.7	3.2	Iris-setosa
4	1.5	0.2	4.6	3.1	Iris-setosa
5	1.4	0.2	5	3.6	Iris-setosa
6	1.7	0.4	5.4	3.9	Iris-setosa
7	1.4	0.3	4.6	3.4	Iris-setosa
8	1.5	0.2	5	3.4	Iris-setosa

Scatter Chart(散布図) を選択してください。
 ここでは、別の画像でラスタを確認できます(4次元ベクトルのため、完全な画像は確認できません)。
 別の視点からラスタを確認する場合は、 xAxisおよび yAxisパラメータを変更してください。



そして、私たちの目的は特徴を使用して花の種類を予測することです。
 新しい段落を追加し、そこに以をり付けて実行してください。

```
from pyspark.ml.linalg import Vectors

from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols = ["PetalLength", "PetalWidth", "SepalLength",
"SepalWidth"], outputCol="features")
// inputColsで指定された4つのパラメータをベクトルを作成、           outputColで名前を指定しています。

irisFeatures = assembler.transform(dataFrame) // これはベクトルを含む           outputCol
列をテーブルに追加します。

irisFeatures.show(5)
```

次に、以下を新しい段落に切り付けて実行します。

```
%pyspark

from pyspark.ml.clustering import KMeans

(trainingData, testData) = irisFeatures.randomSplit([0.7, 0.3]) // データをランダムに2つの部分に分割します

kmeans = KMeans().setK(3).setSeed(101010) //
3つのクラスターを持つK平均モデルです。setSeedは再現可能な結果を作成します。

model = kmeans.fit(trainingData) // KMeansモデルをトレーニングします。

transformed = model.transform(trainingData) // 予測結果を含む新しい列をテーブルに追加します。

transformed.show(150)
```

```
%pyspark
from pyspark.ml.clustering import KMeans

(trainingData, testData) = irisFeatures.randomSplit([0.7, 0.3])
kmeans = KMeans().setK(3).setSeed(101010)
model = kmeans.fit(trainingData)

transformed = model.transform(trainingData)
transformed.show(150)
```

```
+-----+-----+-----+-----+-----+-----+-----+
| ID|PetalLength|PetalWidth|SepalLength|SepalWidth| Species| features|prediction|
+-----+-----+-----+-----+-----+-----+-----+
| 3| 1.3| 0.2| 4.7| 3.2| Iris-setosa|[1.3,0.2,4.7,3.2]| 0|
| 4| 1.5| 0.2| 4.6| 3.1| Iris-setosa|[1.5,0.2,4.6,3.1]| 0|
| 5| 1.4| 0.2| 5.0| 3.6| Iris-setosa|[1.4,0.2,5.0,3.6]| 0|
| 7| 1.4| 0.3| 4.6| 3.4| Iris-setosa|[1.4,0.3,4.6,3.4]| 0|
| 9| 1.4| 0.2| 4.4| 2.9| Iris-setosa|[1.4,0.2,4.4,2.9]| 0|
| 10| 1.5| 0.1| 4.9| 3.1| Iris-setosa|[1.5,0.1,4.9,3.1]| 0|
| 11| 1.5| 0.2| 5.4| 3.7| Iris-setosa|[1.5,0.2,5.4,3.7]| 0|
| 12| 1.6| 0.2| 4.8| 3.4| Iris-setosa|[1.6,0.2,4.8,3.4]| 0|
| 15| 1.2| 0.2| 5.8| 4.0| Iris-setosa|[1.2,0.2,5.8,4.0]| 0|
| 16| 1.5| 0.4| 5.7| 4.4| Iris-setosa|[1.5,0.4,5.7,4.4]| 0|
| 17| 1.3| 0.4| 5.4| 3.9| Iris-setosa|[1.3,0.4,5.4,3.9]| 0|
| 18| 1.4| 0.3| 5.1| 3.5| Iris-setosa|[1.4,0.3,5.1,3.5]| 0|
| 19| 1.7| 0.3| 5.7| 3.8| Iris-setosa|[1.7,0.3,5.7,3.8]| 0|
| 22| 1.5| 0.4| 5.1| 3.7| Iris-setosa|[1.5,0.4,5.1,3.7]| 0|
| 24| 1.7| 0.5| 5.1| 3.2| Iris-setosa|[1.7,0.5,5.1,3.2]| 0|
```

次のように、testDataでモデルを使用します。

```
%pyspark

predictions = model.transform(testData)

predictions.show(151)
```

そして、モデルの精度を計算します。

```
%pyspark

SpeciesAndPreds = predictions.select("Species", "prediction").collect()

def getCluster(specie):
    if specie == "Iris-setosa":
        return 0
    elif specie == "Iris-versicolor":
        return 1
    else:
```

```
return 2

def getAccuracy(flowers):

    counter = 0;

    for flower in flowers:

        if getCluster(flower[0]) == flower[1]:

            counter += 1

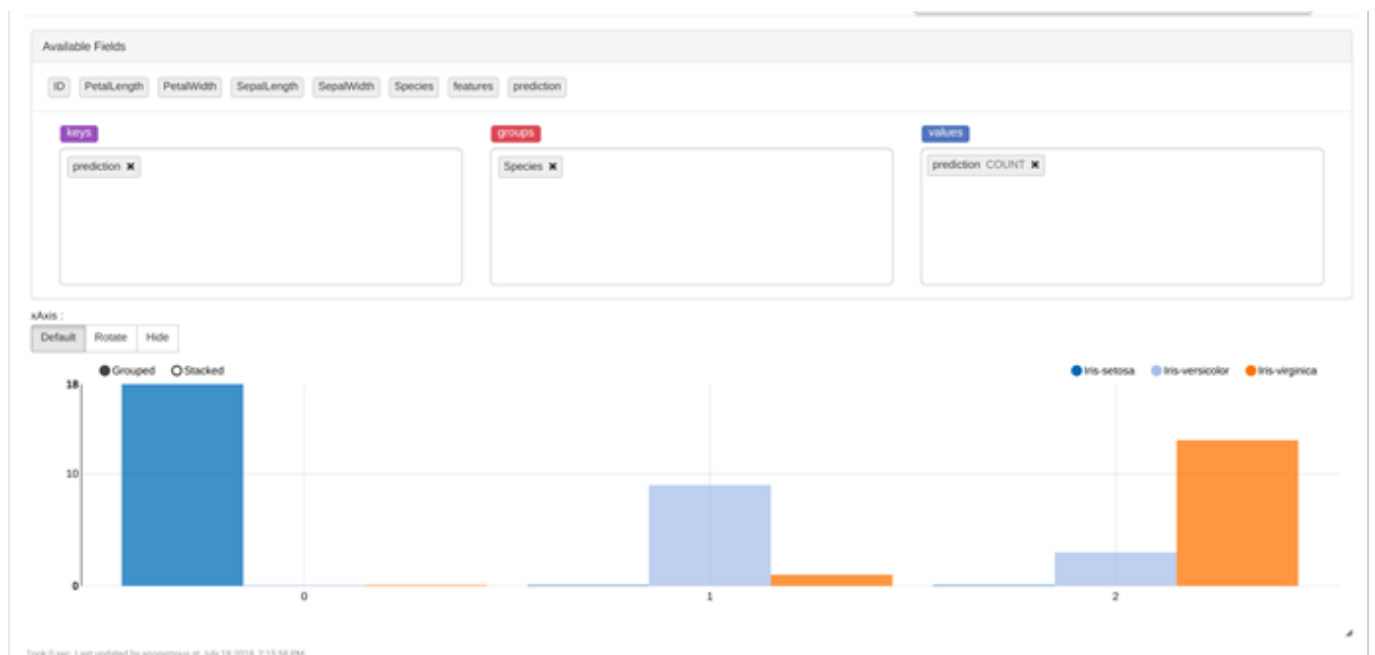
    return counter / len(flowers)

accuracy = getAccuracy(SpeciesAndPreds)

print("accuracy is " + str(accuracy))

// 私の結果は0.9090909090909091です
```

グラフにある数の数を確認するには、「z.show(predictions)」を使用してBar Chart(棒グラフ)を選択します。



まとめ

ここでは、InterSystems IRISでアイリスの種の種をかなり正確に予測するモデルを作成しました(0.9超の精度)。また、「Iris-setosa」は分離可能であり、「Iris-virginica」と「Iris-versicolor」はK平均アルゴリズムでは分離できないことがわかりました。そのため、精度を高めるためには別の方法を使用することをお勧めします。

リンク

[Apache Spark + Apache Zeppelin + InterSystems IRISを起動する方法](#)

[Spark SQL, データフレームおよびデータセットガイド](#)

[機械学習ライブラリガイド](#)

[K平均法](#)

[ラスタリング\(ただし、このAPIはRDD用であるため、アルゴリズムに関する関連情報を参照するだけにしてください\)](#)

[アイリスデータセット](#)

[#AI](#) [#API](#) [#Python](#) [#初心者](#) [#機械学習](#) [#InterSystems IRIS](#)

ソースURL: <https://jp.community.intersystems.com/post/%E3%82%A2%E3%82%A4%E3%83%AA%E3%82%B9%E3%83%87%E3%83%BC%E3%82%BF%E3%82%BB%E3%83%83%E3%83%88%E3%81%AEk%E5%B9%B3%E5%9D%87%E3%82%AF%E3%83%A9%E3%82%B9%E3%82%BF%E3%83%AA%E3%83%B3%E3%82%B0>