プログラミング言語

kmeans 法

令和 5 年 8 月 10 日 20-416 小柴 颯太

1 .kmeans 法について

1. 1 kmeans 法とは

互いに近いデータ同士は同じクラスタであるという考えに基づいたデータ群を k 個に分類するクラスタリングの手法。

クラスが 3 つの場合。
（１）初期値を 3 つ決める。
（２） 3 つの初期値から最も近いクラスタに割り振る。
（３）各クラスの重心を測る。
重心が変化しないまで（２）と（３）を繰り返す。

1. 2 欠点

初期値によってクラスタリング結果が異なる。
例）初期値が(0.1,0.1) (-0.25,1) (0.75,0.25)の場合
　　結果：(0.12630833,0.04005) (-0.03502727,1.00790909) (0.65487143,0.43904286)

　　初期値が(0,0) (-0.25,1) (0.75,0.25)の場合
　　結果(0,0) (-0.03502727,1.00790909) (0.61925,0.430475)

1. 3 kmeans++法

このような結果が初期値に依存してしまうことを改善した kmeans++法がある。
kmeans++法とはクラスタの中心の初期値を遠ざけるように選択することで、kmeans 法の初期値問題を改善したアルゴリズムである。
つまり、クラスタの中心同士は離れていた方が良いという考えである。
参考文献：https://hogetech.info/machine-learning/algorithm/kmeans

2.python による kmeans 法の実装

2. 1 〜2. 4 は無限ループの中とする。

2.1 クラス分け

```
while index < len(data):
    dist[0] = np.linalg.norm(val0-data[index])

    dist[1] = np.linalg.norm(val1-data[index])

    dist[2] = np.linalg.norm(val2-data[index])
    print(dist)
    minIndex  = dist.index(min(dist))
    print(minIndex)
    if minIndex == 0:
        class0.append(data[index])
    elif minIndex == 1:
        class1.append(data[index])
    elif minIndex == 2:
        class2.append(data[index])
    dist[0] = 0
    dist[1] = 0
    dist[2] = 0
    index += 1
```

各クラスタの重心との距離を測り最も小さい値をとったものにクラスタリングする。

2.2 比較するために重心を記録、重心を計測するための各クラスの合計を測る変数を生成

```
beforeVal0x = val0[0]
beforeVal0y = val0[1]
beforeVal1x = val1[0]
beforeVal1y = val1[1]
beforeVal2x = val2[0]
beforeVal2y = val2[1]

val0x = 0
val0y = 0
val1x = 0
val1y = 0
val2x = 0
val2y = 0
```

2.3 各クラスの重心を計算

```
size = 0
while size < len(class0):
    val0x += class0[size][0]
    val0y += class0[size][1]
    size += 1
size = 0
while size < len(class1):
    val1x += class1[size][0]
    val1y += class1[size][1]
    size += 1
size = 0
while size < len(class2):
    val2x += class2[size][0]
    val2y += class2[size][1]
    size += 1

val0[0] = round(val0x/len(class0),5)
val0[1] = round(val0y/len(class0),5)

val1[0] = round(val1x/len(class1),5)
val1[1] = round(val1y/len(class1),5)

val2[0] = round(val2x/len(class2),5)
val2[1] = round(val2y/len(class2),5)
```

有効桁数を指定しないと重心が更新し続けるので、round()を使い小数点第5位までを出す。

2.4 重心が移動したか比較をする。

```
if beforeVal0x == val0[0] and beforeVal0y == val0[0]:
    count += 1

if beforeVal1x == val1[0] and beforeVal1y == val1[0]:
    count += 1

if beforeVal2x == val2[0] and beforeVal2y == val2[0]:
    count += 1
if count == 3:
    break
```

重心が移動していなかったら、break でループから抜ける。

2.5 点をプロット、表示

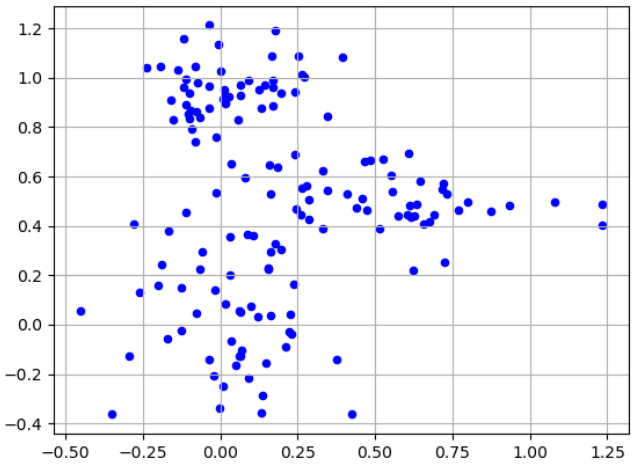
```
plt.scatter(val0[0],val0[1],marker = "x",color = "r",alpha = 0.5)
plt.scatter(val1[0],val1[1],marker = "x",color = "b",alpha = 0.5)
plt.scatter(val2[0],val2[1],marker = "x",color = "g",alpha = 0.5)
size = 0
while size < len(class0):
    plt.scatter(class0[size][0],class0[size][1],s = 10,color = "r")
    size += 1
size = 0
while size < len(class1):
    plt.scatter(class1[size][0],class1[size][1],s = 10,color = "b")
    size += 1
size = 0
while size < len(class2):
    plt.scatter(class2[size][0],class2[size][1],s = 10,color = "g")
    size += 1

plt.grid()
plt.show()
```

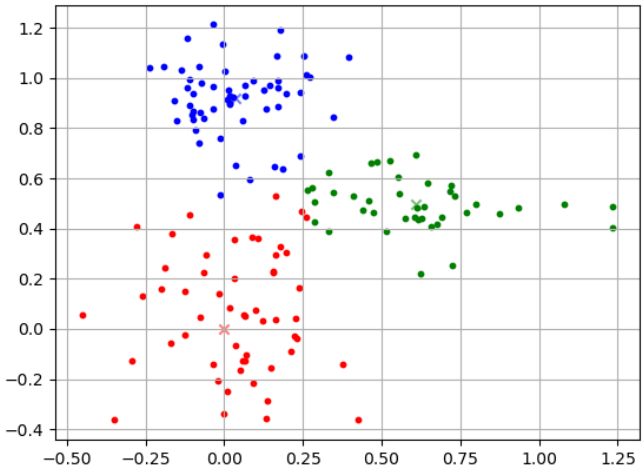
クラス 0 を赤、クラス 1 を青、クラス 2 を緑で表示し各重心を×とする。

３.実行と結果

makeData.py を実行



結果



重心

Class0	Class1	Class2
0,0	0.3517,0.91866	0.60982,0.4975

4.最後に

プログラミング言語の授業を通して、最小二乗法、kmeans 法を学習し今まで名前は聞いたことあるが深くは理解していなかったデータサイエンスというものを学習することができ興味を持つことができた。データサイエンスを触れることができ自分にとって有意義なものとなった。この授業で機械学習について興味をもったため自身でも学習をして見たいと思った。