DS3

方法来源于Dissimilarity-based Sparse Subset Selection

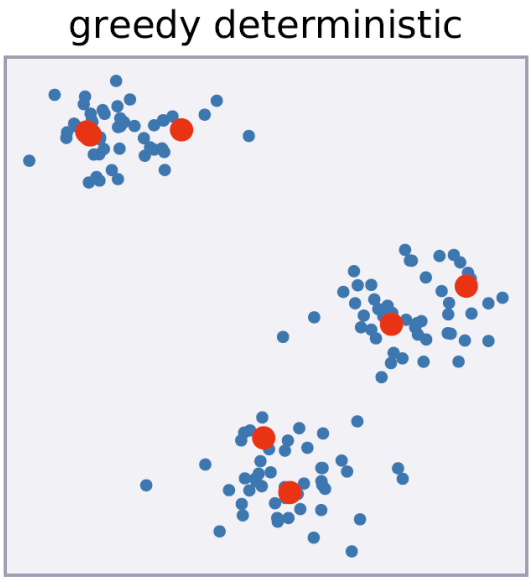
给定集合A，B，以及集合A中每个元素代表集合B中元素的代价，该方法会选择A中的一个子集，选取尽量少的元素达到尽量好的代表效果。其基本思想是每选择一个元素就会有一个额外的选择代价，然后评估代表的代价+选择的代价，寻找一个总代价最小的组合。

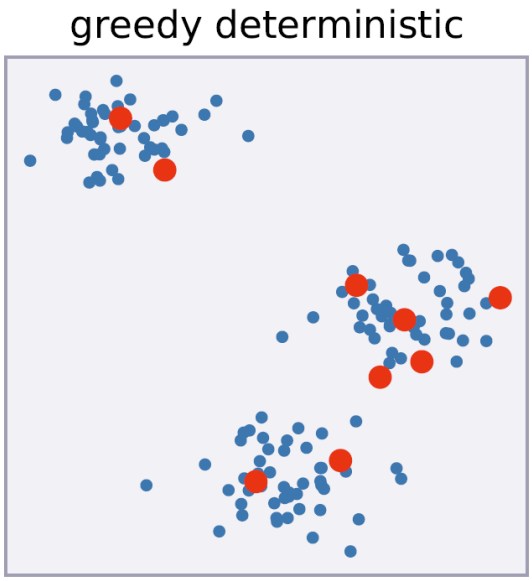
更详细的使用可以参考 `ds3.py` 里的 `main`，这里简单介绍基本的用法和一些结果。

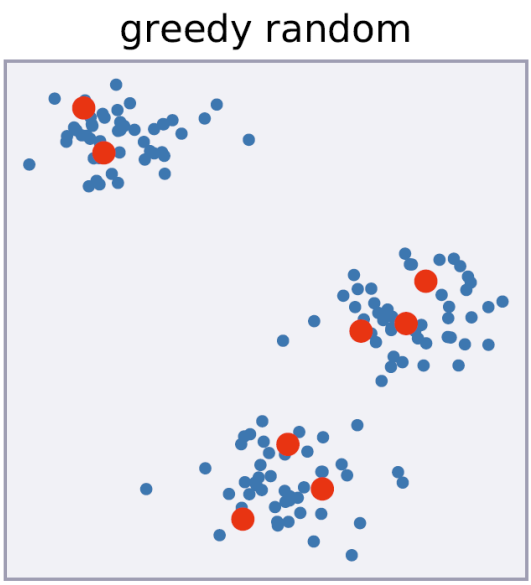
直接选择有代表性的元素

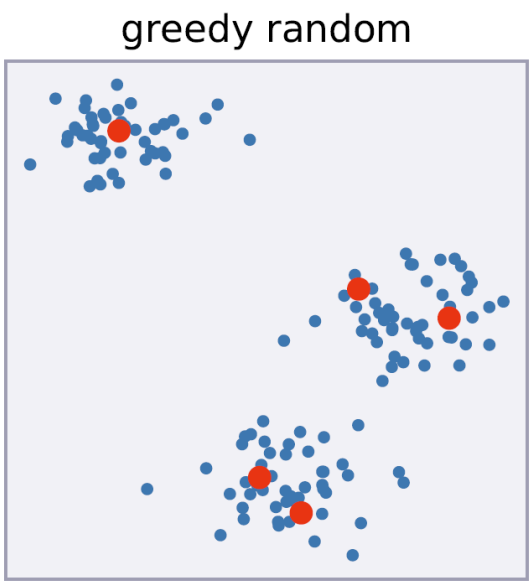
主要需要调节的参数就是`reg`

```
# dist 为集合A到集合B
# reg 为正则项的强度，值越大选出的样本越多
# 用三种方法选择有代表性的元素
SS = SubsetSelection(dist, reg)
selected1 = SS.ADMM(1e-1, 1e-5, 1e4, np.inf)['selected']
selected2 = SS.greedyDeterministic()['selected']
selected3 = SS.greedyRandomized()['selected']
```







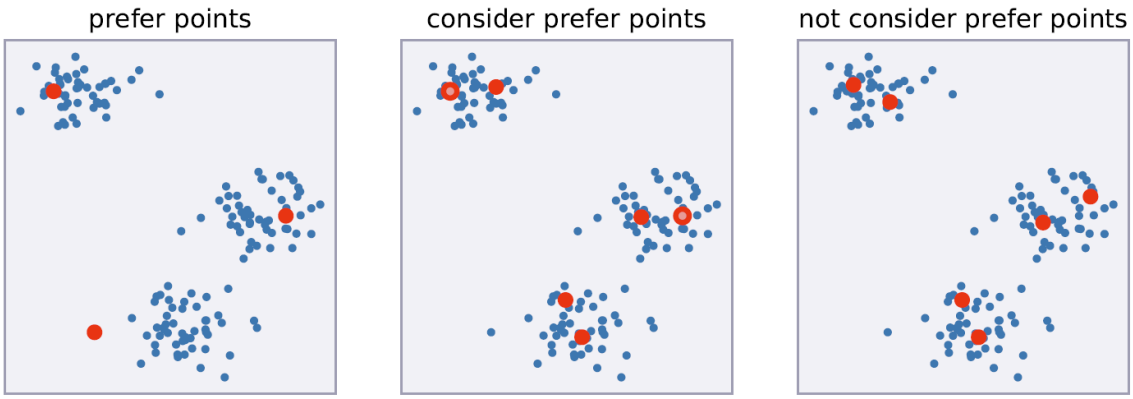


希望保留某些选出的元素

可以在ADMM的方法里设置（其他两个方法暂不支持）

`prefer_row`是你打算保留的A的元素的下标，`prefer_coef_shrink`是保留的强度（1表示没有偏好，小于1的值会让选出相关元素的代价更小，因此更容易被选出，设为0则意味着一定会被选出，设置为大于1则表示尽量不选出）

```
selected1 = SubsetSelection(dist, reg).ADMM(1e-1, 1e-5, 1e4, np.inf,
prefer_row=prefer_row, prefer_coef_shrink=0.3)['selected']
```



上图给了一个例子，坐标是给出的三个倾向保留的点，右边时完全不考虑这三个点给出的选择结果。可以看到每个聚类簇选出两个，非常合理，但这六个和倾向给出的三个完全不重叠。中间的结果则考虑了倾向保留的点：首先可以看到选择的效果和最右边基本一样，但是左图中两个质量比较高的点被保留下来，第三个比较偏离聚类簇的就被丢弃了。

如果有更精细的先验，也可以直接设置 `rows_coef`， 每个元素选出的代价

```
selected1 = SubsetSelection(dist, reg).ADMM(1e-1, 1e-5, 1e4, np.inf,
rows_coef=rows_coef)['selected']
```

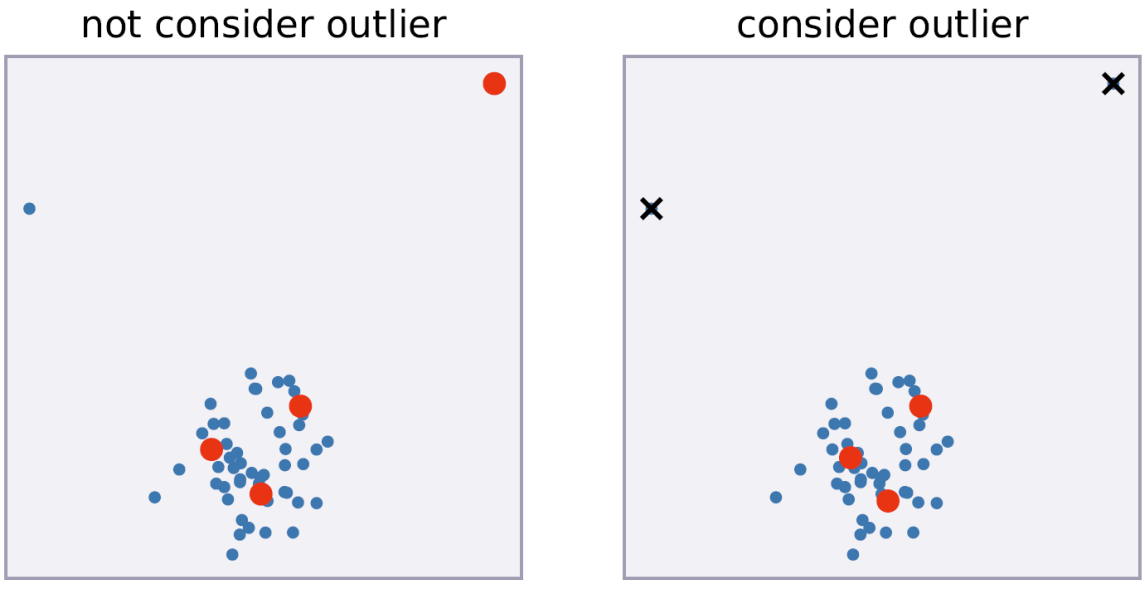
有些时候我们希望两个元素同时被选出，此时可以设置`cooperation`，它在同时选择`i`和`j`的时候，会加上额外的选择代价`cooperation[i,j]`。因此`cooperation[i,j]`越小会越容易同时选出`i`和`j`。

希望处理outlier

有时候outlier会导致我们花费一个额外的选择去代表他，我们可以将它们收入一个额外的类即outlier类中

```
result = SubsetSelection(dist, reg).ADMM(1e-1, 1e-5, 1e4, np.inf, outlier=True,
beta=2, tau=1)
```

其中`beta`和`tau`的值需要设置，具体的含义可以参见论文。



效果如上，不考虑outlier时，额外选择了一个元素来使得整体的代表代价降低。考虑outlier时，两个outlier被标记出来。