Homework4

August 9, 2022

1 K-means and K-medoids

Assume we have a 2D dataset consisting of $(0, -6), (4, 4), (0, 0), (-5, 2)$. We wish to do k-means and k-medoids clustering with $k = 2$. We initialize the cluster centers with $(-5, 2), (0, -6)$.

For this small dataset, in choosing between two equally valid exemplars for a cluster in k-medoids, choose them with priority in the order given above (i.e. all other things being equal, you would choose $(0, -6)$ as a center over $(-5, 2)$).

For the following scenarios, give the clusters and cluster centers after the algorithm converges. Enter the coordinate of each cluster center as a square-bracketed list (e.g. [0, 0]); enter each cluster's members in a similar format, separated by semicolons (e.g. [1, 2]; [3, 4]).

[98]:
```
ds = ((0, -6), (4, 4), (0, 0), (-5, 2))
c = ((-5, 2), (0, -6))

norm = lambda lst, dim: sum(el**dim for el in lst)**(1/dim)
l1 = lambda lst: norm(lst, 1)
l2 = lambda lst: norm(lst, 2)
abs_diff = lambda a, b: map(lambda x,y: abs(x-y), a, b)
first_min_arg = lambda lst: lst.index(min(lst))


def closest_center(dp, norm, centers):
    distc = [norm( abs_diff(dp, cn) ) for cn in centers]
    return first_min_arg(distc)

def assign_to_closest_center(ds, norm, centers):
    return [closest_center(dp, norm, centers) for dp in ds]

def shortest_distance_to_other_points_in_group(ds_group, l_norm):
    return first_min_arg([sum(l_norm(abs_diff(point, other)) for other in␣
 ↪ds_group) for point in ds_group])

def step1_k_medoids(ds, l_norm, center):
    ds_groups = list(list(filter(lambda dp: closest_center(dp, l_norm, center)␣
 ↪== center_index, ds)) for center_index in (0,1))
    return ds_groups
```

```
def step2_k_medoids(ds_groups, l_norm):
    """Returns center indices"""
    return [shortest_distance_to_other_points_in_group(ds_group, l_norm) for␣
 ↪ds_group in ds_groups]


def step_k_medoids(ds, l_norm):
    centers = c
    ds_groups = step1_k_medoids(ds, l_norm, centers)
    new_center_indices = step2_k_medoids(ds_groups, l_norm)
    centers =list(group[new_center_indices[idx]] for idx, group in␣
 ↪enumerate(ds_groups))
    return ds_groups, centers

def print_clusters(ds_groups):
    [print(
        f"Group {grp_no}: ",
        '; '.join([
            str([el for el in point]) for point in group
            ]))
    for grp_no, group in enumerate(ds_groups)]

# print_clusters(step1_k_medoids(ds, l1, c))
# print_clusters(step1_k_medoids(ds, l2, c))
# print(*[shortest_distance_to_other_points_in_group(ds_group, l2) for ds_group␣
 ↪in step1_k_medoids(ds, l2)])
ds_groups, centers = step_k_medoids(ds, l2)
print_clusters(ds_groups)
print(centers)
```

```
Group 0:  [4, 4]; [0, 0]; [-5, 2]
Group 1:  [0, -6]
[(0, 0), (0, -6)]
```

2 Maximum Likelihood Estimation

[116]:
```
seq = 'A B A B B C A B A A B C A C'.split(' ')
print(*[el+ ": " + str(seq.count(el)) + '/' + str(len(seq)) + ' or ' +␣
 ↪str(round(seq.count(el)/len(seq),3)) for el in set(seq)], sep='\n')
```

```
C: 3/14 or 0.214
B: 5/14 or 0.357
A: 6/14 or 0.429
```

[127]:
```
from functools import reduce
ThA, ThB, ThC = [round(seq.count(el)/len(seq),3) for el in ['A', 'B', 'C']]
```

```
seqs = ['ABC', 'BBB', 'ABB', 'AAC']
probability = lambda ch: ThA if ch=='A' else ThB if ch=='B' else ThC
product = lambda x,y: x*y

for s in seqs:
   print(s, reduce(product, map(probability, s)))
```

```
ABC 0.032774741999999996
BBB 0.04549929299999999
ABB 0.054675620999999994
AAC 0.039384774
```

[ ]: