worksheet_05

February 13, 2024

1 Worksheet 05

Name: Daniyal Ahmed UID: U11469883

1.0.1 Topics

- Cost Functions
- Kmeans

1.0.2 Cost Function

Solving Data Science problems often starts by defining a metric with which to evaluate solutions were you able to find some. This metric is called a cost function. Data Science then backtracks and tries to find a process / algorithm to find solutions that can optimize for that cost function.

For example suppose you are asked to cluster three points A, B, C into two non-empty clusters. If someone gave you the solution `{A, B}, {C}`, how would you evaluate that this is a good solution?

Notice that because the clusters need to be non-empty and all points must be assigned to a cluster, it must be that two of the three points will be together in one cluster and the third will be alone in the other cluster.

In the above solution, if A and B are closer than A and C, and B and C, then this is a good solution. The smaller the distance between the two points in the same cluster (here A and B), the better the solution. So we can define our cost function to be that distance (between A and B here)!

The algorithm / process would involve clustering together the two closest points and put the third in its own cluster. This process optimizes for that cost function because no other pair of points could have a lower distance (although it could equal it).

1.0.3 K means

a) (1-dimensional clustering) Walk through Lloyd's algorithm step by step on the following dataset:

`[0, .5, 1.5, 2, 6, 6.5, 7]` (note: each of these are 1-dimensional data points)

Given the initial centroids:

`[0, 2]`

Initial clusters based on the initial centriod [0, .5] and [1.5 , 2, 6 ,6.5 ,7]

- New Centriod[.25, 4.6] Cluster for .25 is [0, .5, 1.5, 2] and cluster for 4.6 is [6, 6.5, 7]

- Final Centriod [1, 6.5] Converges cluster for 1 is [0, .5, 1.5, 2] and cluster for 4.6 is [6, 6.5, 7]

b) Describe in plain english what the cost function for k means is.

The cost function in terms Kmeans can be defined as distance between clusters. In the example putting A and B together minimizes the cost function since no other pair would be more optimal. Therefore in the sense minimizing the cost function would mean recalculating the clusters in the K means until groups of points that are close are assigned a single cluster while groups of points are apart are assigned different clusters all while ensuring that there are only K clusters.

c) For the same number of clusters K, why could there be very different solutions to the K means algorithm on a given dataset?

Yes This is because the Kmeans algorithm is not guarrented to be optimal since the Optimal solution is NP hard. Furthermore because Lloyds algorithm initializes at random points whenever it converges it may produce a slighty different cluster since the intiial centriods aren't starting at the same points each time

d) Does Lloyd's Algorithm always converge? Why / why not?

Yes it always converges . This is because what Lloyds algorthim does it recalculates the centriod everytime it assigns clusters, eventually it will get to a point where no matter how many times it recalculates the centriods they are the same each time meaning it cannot find a more optimal solution thus forcing convergence

e) Follow along in class the implementation of Kmeans

[1]:
```
import numpy as np
from PIL import Image as im
import matplotlib.pyplot as plt
import sklearn.datasets as datasets

centers = [[0, 0], [2, 2], [-3, 2], [2, -4]]
X, _ = datasets.make_blobs(n_samples=300, centers=centers, cluster_std=1,␣
 ↪random_state=0)

class KMeans():

    def __init__(self, data, k):
        self.data = data
        self.k = k
        self.assignment = [-1 for _ in range(len(data))] ##Negative one means␣
 ↪unassigned
        self.snaps = []

    def snap(self, centers):
        TEMPFILE = "temp.png"

        fig, ax = plt.subplots()
        ax.scatter(X[:, 0], X[:, 1], c=self.assignment)
        ax.scatter(centers[:,0], centers[:, 1], c='r')
```

```
        fig.savefig(TEMPFILE)
        plt.close()
        self.snaps.append(im.fromarray(np.asarray(im.open(TEMPFILE))))




    def unassign_all(self):
        self.assignment = [-1 for _ in range(len(data))]


    def initialize(self):
       self.data [np.random.choice(range(len(self.data)), size=self.k,␣
 ↪replace=False)]


    def lloyds(self):
        centers = self.initialize()
        self.snaps.append(self.snap(centers))
        self.assign(centers)
        new_centers= self.calculate_new_centers()

        new_centers = self.calculate_new_centers()

        while  self.are_centers_diff(centers, new_centers):

            centers=new_centers
            self.unassign_all()
            self.assign(centers)
            self.calculate_new_centers()

            new_centers = self.calculate_new_centers()







        return


    def is_unassigned(self,i):
        return self.assignment[i]

kmeans = KMeans(X, 6)
kmeans.lloyds()
images = kmeans.snaps
```

```
images[0].save(
    'kmeans.gif',
    optimize=False,
    save_all=True,
    append_images=images[1:],
    loop=0,
    duration=500
)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[1], line 37
     34 kmeans.lloyds()
     35 images = kmeans.snaps
---> 37 images[0].save(
     38     'kmeans.gif',
     39     optimize=False,
     40     save_all=True,
     41     append_images=images[1:],
     42     loop=0,
     43     duration=500
     44 )

IndexError: list index out of range
```