Assignment 2

Gaurav Baney gbaney@usc.edu
02/22/2020

K - Means Clustering and Gaussian Mixture Modelling

1. Implementation

1.1 K - Means Clustering

My implementation of K - Means Clustering is in the file 'kmeans.py'
The input file has been hardcoded to run upon executing the file.

I also have a display function commented out on line 31 that can be uncommented to generate a visualization of the cluster allotments of each point. I got the code for that function online since it was supplemental to the project.

To make some operations easier for me I defined a custom Point object that would handle the point functionality and encapsulate required data. This also helped me debug and test my implementation. I decided to approach it this way because I struggled with containers for the data in the previous assignment.

```python
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def equal(self, other):
        return (self.x == other.x and self.y == other.y)

    def distance(self, other):
        return np.sqrt((((self.x - other.x) * (self.x - other.x)) + ((self.y -
    other.y) * (self.y - other.y))))

    def __repr__(self):
        return ("(X: " + str(self.x) + ", Y: " + str(self.y) + ")")
```

The results of my implementation were as follows:

```
C1:: X = 5.433123874157894, Y = 4.862675026605265
C2:: X = 2.8834971090689647, Y = 1.358261948
C3:: X = -1.0394086163975909, Y = -0.6791968002650602
```

In my implementation I use numpy arrays to improve the speed of the algorithm and also for certain math features that can be directly applied to the numpy arrays

1.1.1 Data Structures

I used a global list to keep track of all the points from the provided file
I use a list to contains the centroid values
I use a dictionary to map the centroid to a list of points that belong in its cluster
I use a list of tuples to compare the new centroid values against the old centroid values to check for convergence of the algorithm

1.1.2 Experience

Overall K-Means was the simpler part of the assignment so I didn't face any challenges during my implementation besides actually having to plot the points and see if the results made sense

1.2 Gaussian Mixture Modeling

My Implementation of Gaussian Mixture Modeling and the EM algorithm is in the file 'gmm.py'
The input file has been hardcoded to run upon executing the file.

I followed the math and the process from here (link)

I also have a custom Gaussian Object that handle gaussian related functionality and for the same reasons mentioned above with the Point object.

```
class Gaussian:
    def __init__(self, mean, covariance, pi):
        self.mean = mean
        self.covariance = covariance
        self.pi = pi

    def equal(self, other):
        return (np.array_equal(self.mean, other.mean) and np.array_equal(self.
    covariance, other.covariance) and self.pi == other.pi)

    def __repr__(self):
        return ("mean: " + str(self.mean) + "\n covariance " + str(self.
    covariance) + "\n pi " + str(self.pi))
```

I was going to add the gaussian specific log likelihood calculation mentioned in the link but chose to implement it as a gmm function instead
My E step takes place in the function recomputeRics and my M step takes place in recomputeGaussians

The results of my implementation were as follows:

```
G1::
    Mean = (5.36678814, 4.7210651)
    Covariance = [[ 2.38839832, 0.45796861]
                  [0.45796861,  2.36555434]]
    Pi = 0.25650330779566216
```

```



G2::
    Mean = (3.02196159, 1.56224676)
    Covariance = [[1.9798507, -0.16780424]
                  [-0.16780424, 2.19718547]]
    Pi = 0.17672523937888188



G3::
    Mean = (-0.97381006, -0.65000538)
    Covariance = [[1.21862022, -0.10979342]
                  [-0.10979342, 1.99675829]]
    Pi = 0.566771452825456
```

1.2.1 Data Structures

I used a global list to keep track of all the points from the provided file
I used a list to keep track of the working gaussians for my data model
I also changed my Point class to keep track of the R value to emulate "Ric"

1.2.2 Experience

I actually struggled to get started with this assignment because the math really confused me.
I think I spent more time figuring out how the math worked and reading about the gaussians online than actually coding.
Thanks to the high amount of resources online I was able to get it working.

1.3 Software Familiarization

1.3.1 Sklearn.Cluster

I used the sklearn.cluster package for K Means clustering.
The results were as follows:

```
C1:: X = 5.62016573, Y = 5.02622634
C2:: X = 3.08318256, Y = 1.77621374
C3:: X = -0.97476572, Y = -0.68419304
```

The sklearn package was a lot more consistent with its results than my implementation and allowed you to adjust a lot more parameters to get a tighter fit of the data. The package also has a lot of cool data visualization tools to help the user understand how the data is being handled. The packages answers were also pretty close to mine. My code relies on a lot of assumptions based on the input and arrangement of data and can definitely be improved.

1.3.2 SkLearn.Mixture

I used the sklearn.mixuture package for Gaussian mixture modeling.
The results were as follows:

```
Means:
[[-0.96251642 -0.64669019]
 [ 5.47804494  4.85130933]
 [ 3.13133022  1.71921331]]

Covariances:
[[[ 1.24276739 -0.10763273]
  [-0.10763273  2.00006547]]

 [[ 2.28078442  0.29776281]
  [ 0.29776281  2.18625177]]

 [[ 1.94166193 -0.10787169]
  [-0.10787169  2.25348199]]]

Weights:
[0.56980611 0.23867267 0.19152122]
```

The sklearn package was a lot more consistent with its results than my implementation and allowed you to adjust a lot more parameters to get a tighter fit of the data. The package also has a lot of cool data visualization tools to help the user understand how the data is being handled. The package also did not need an or did not expose a need for initial seed values. The packages answers were also pretty close to mine. I'm not entirely certain my code performs its calculations accurately and it could definitely affect its consistency.

2. Applications

2.1 K MEANS

K Means can be used for a variety of data analysis applications like looking at sales activity over the year or product popularity. It can also be used to analyze data related to people and group event involvement and be used to optimize in terms of event popularity.

2.2 GMM

Since Mixture Models only soft label the data points cluster membership it proved to be really useful for feature extraction in speech recognition models as substrings and phrases can hold different meaning.