[INF552 - Machine Learning for Data Informatics]

Technical Specification and Study Report

[K-means & Expectation Maximization using Gaussian Mixture Model]

Author: [Cheng-Lin Li (9799716705), Jianfa Lin (6950769029)]
Date: [Oct., 2, 2016]
Version: [1.1]

Document Control

Author

Position	Name	USC ID
Student	Cheng-Lin Li	9799716705
Student	Jianfa Lin	6950769029

Revision history

Version	Issue date	Author/editor	Description/Summary of changes
0.8	2016/9/24	Cheng-Lin Li	First draft release
0.9	2016/9/28	Cheng-Lin Li	Include K-Means implementation results, and E-M GMM implementation. Reference package results for K-Means and EM-GMM
1.0	2016/10/1	Cheng-Lin Li	First release
1.1	2016/10/2	Cheng-Lin Li	1.Minor revise on the description of K-Means data structure and procedure in this document. 2. Revise the typo in gmm implementation from 'likelyhood' to 'likelihood'.

Related documents

Document	Description
kmeans.py	K-Means implementation by Python 3.5
gmm.py	EM-GMM implementation by Python 3.5
clusters.txt	Data Set
k-means_EM-GMM.py	Clustering testing code by kmeans & gmm library from scikit-learn python package for clustering

Contribution

Name	USC ID	Labour
Jianfa Lin	6950769029	Implement the code
Cheng-Lin Li	9799716705	Write the document

Table of Contents

1 INTRODUCTION

1.1 Objectives

According to a given record sets, implement K-means algorithm AND the Expectation Maximization algorithm (EM) for clustering using a Gaussian Mixture Model (GMM).

The results should print the number of clusters, k, and:

K-means: Report the centroid of each cluster.

EM algorithm by GMM: The mean, amplitude and covariance matrix of each Gaussian in GMM.

The record sets are two dimension points; total number of points is n=150 which can be extend to a larger number sets too. The number of clusters are set to 3 in this assignment, and it should be extended to k which k <= n.

2 Functionality

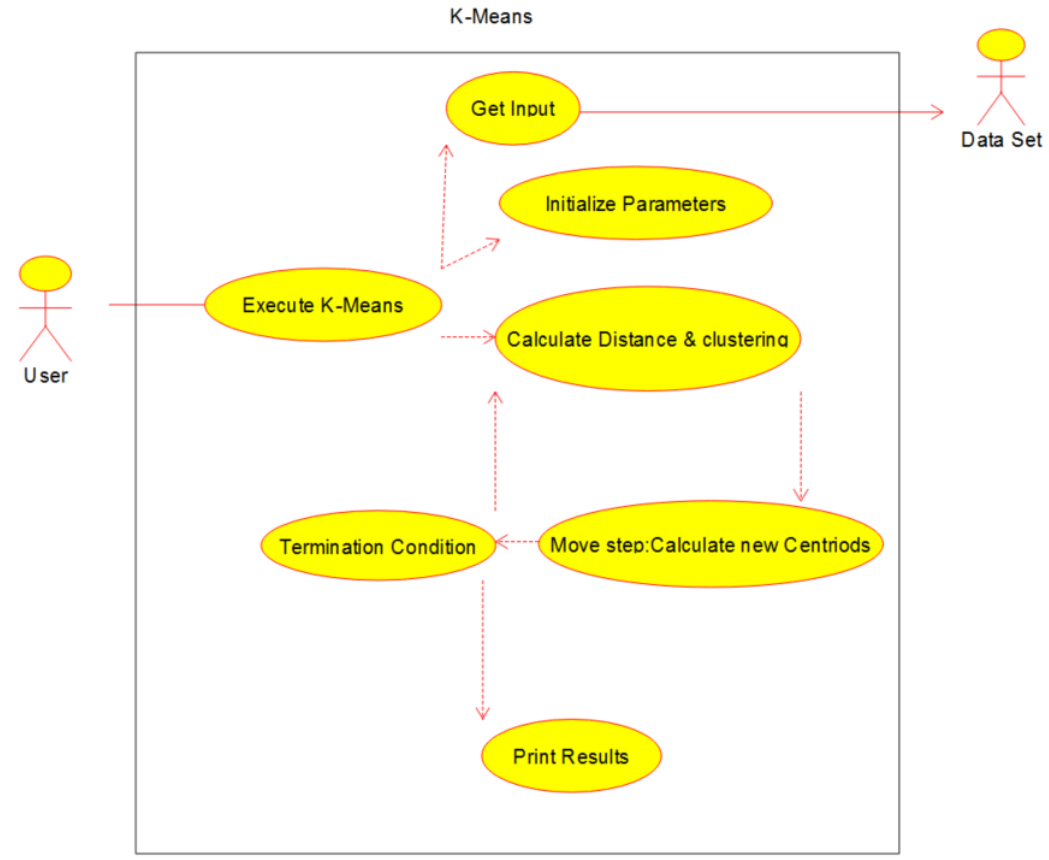
2.1 Description: K-Means

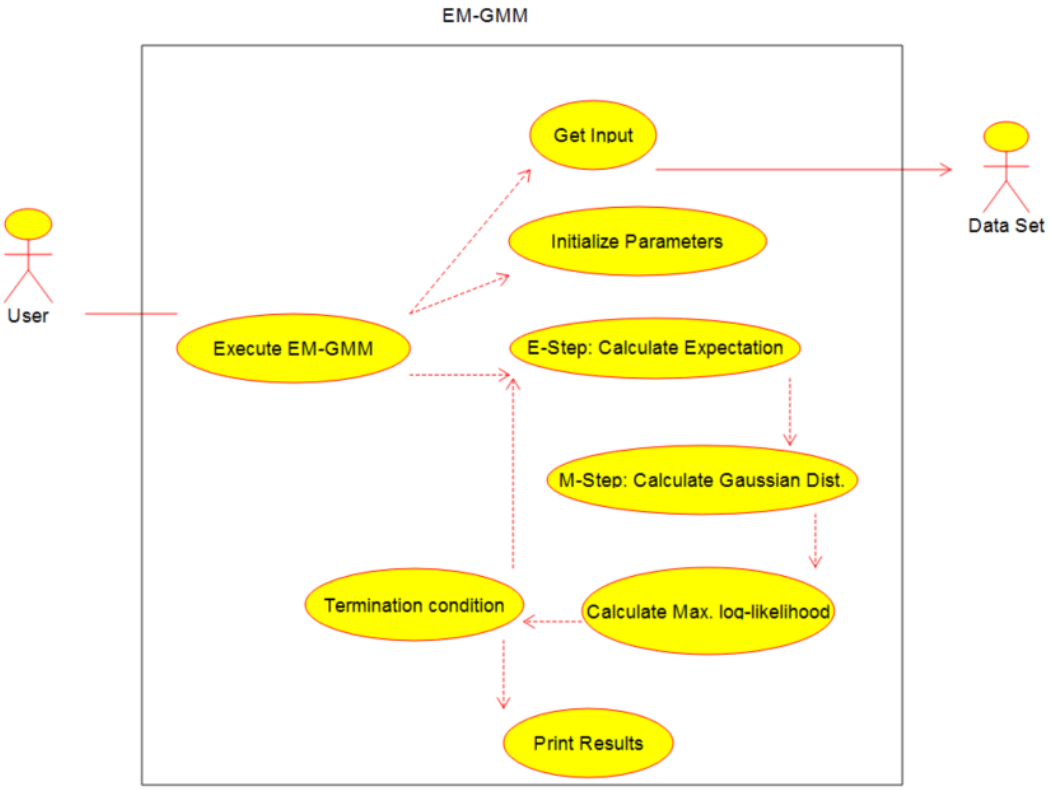
1. According to the require number of clusters, K, randomly get K initial centroids.
2. Get all data points from input
3. Calculate the distance for each data point and label the cluster.
4. Calculate the new centroids.
    a. Base on the new groups of data points, calculate new centroids.
    b. Calculate the total distance for each new centroid with its own group of data points.
5. Compare the new distance with old distance.
    a. Termination condition:
        i. If no significant improvement of total distance, we stop the process.
    b. Else repeat step 3 ~ 5.

2.2 Description: Expectation Maximization using Gaussian Mixture Model

1. Randomly pick up K points from data points as initial value of mean for each cluster.
2. Assume the initial covariance matrix is [[1,0], [0,1]] for 2 dimensions' data set for each cluster.
3. Assume the initial weight for each Gaussian distribution is 1/K; where K is number of clusters.
4. Get all data points from input file
5. Calculate the Probability under k Gaussian distribution for each data point.
6. Label the data point to maximum probability of Gaussian distribution.
7. Calculate the new Gaussian distributions base on new classification.
8. Summary the total probability of each Gaussian distribution.
    a. If no significant improvement on total probability, we stop the process.
    b. Else repeat step 5 ~ 7.

2.3 Use Case

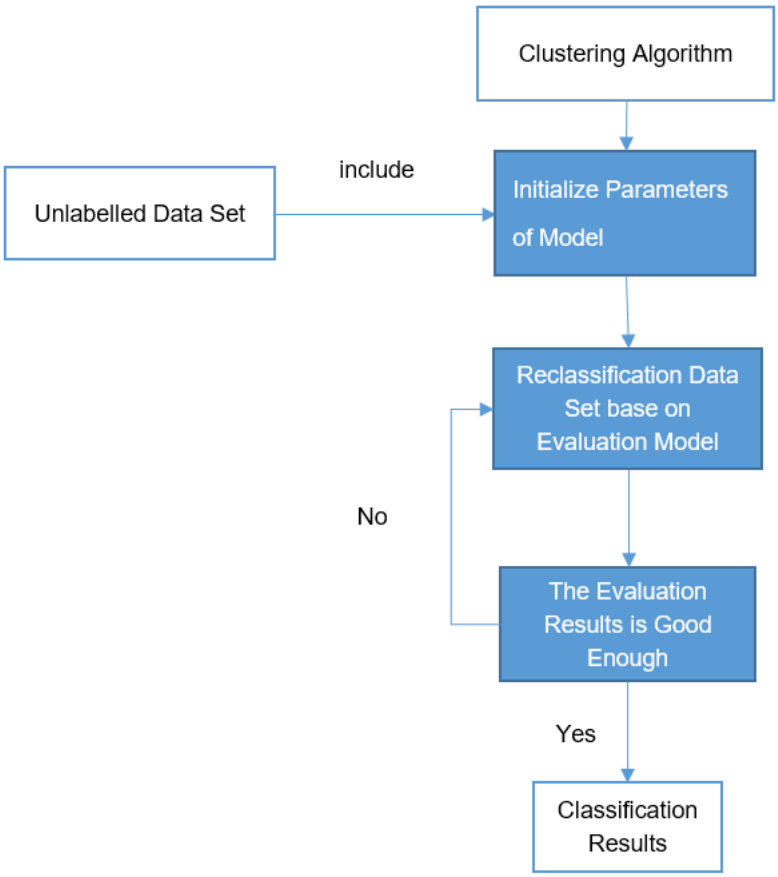

K-Means
Get Input
Data Set
Initialize Parameters
User
Execute K-Means
Calculate Distance & clustering
Termination Condition
Move step:Calculate new Centriods
Print Results


EM-GMM
Get Input
Data Set
Initialize Parameters
User
Execute EM-GMM
E-Step: Calculate Expectation
M-Step: Calculate Gaussian Dist.
Termination condition
Calculate Max. log-likelihood
Print Results

3 Architecture

3.1 Architecture Overview

Here is the generalized architecture diagram for clustering algorithms.



4 Design

4.1 Data Structure

4.1.1 Basic data structure

1. All centroids/means store in a list/array.
    a. There are two sets of centroids: centroids and new_centroids.
    b. centroid = [centroid point1, …, centroid point K]
    c. The data structure of each centroid is [x, y]

4.1.2 K-Means

1. All data points store in a list/array.
    a. datapoints = [point 1, point 2, …, point N]
    b. The data structure of each point is [x, y]
        i. x, y is float number.
2. Distance store in a list/array.
    a. Current distance: Store distance between each point with associated current centroid.
        i. current_distance = [point 1 distance with associated centroid k, …, point N distance with associate centroid k]
    b. New distance: Store total distance for each new centroid.
        i. new_distance = [Total distance of new centroid 1, …, Total distance of new centroid k]
3. Data Classification in a list/array
    a. An array structure to record cluster for each point.
    b. data_classification = [cluster of point 1, cluster of point 2, …, cluster of point N]
        i. The default value of identification number of clusters is -1.
4. new_centroid has same data structure as data point but add a third dimensions to record down total number of points belong to this new centroid.
    a. new_centroid = [[x0, y0, n0], [x1, y1, n1], [x2, y2, n2]]
    b. n0, n1, n2 = total number of points which identify belonging to these new centroids.

4.1.3 Expectation Maximization using Gaussian Mixture Model

1. All data points store in a list/array.
    a. datapoints = [point 0, point 1, …, point N-1]
    b. The data structure of each point is [x, y]
        i. x, y is float number.
2. A cluster identification list/array, “c”, to label the cluster of each data point.
    a. [label of point 1, label of point 2, …, label of point N]
    b. The default value is -1.
3. Covariance matrix store in a list/array with K, 2D points.
    a. current_cov matrix = [[x1,y1],…,[xk, yk]]
    b. new_cov_matrix has same data structure.

4. A weight storage list/array, “w”, to record the weight of each cluster.
    a. [weight of cluster 1, weight of cluster 2, …, weight of point K].
    b. current_weight = [float 1, …, float k]
    c. new_weight has same data structure.
5. The posteriori probability of Xi for each cluster (r_{ic}) will store in list/array for each cluster.
    a. Structure are

    [
    [[r_1 in cluster 1], [r_2 in cluster 1],…, [r_n in cluster 1]],
    [[r_1 in cluster 2], [r_2 in cluster 2],…, [r_n in cluster 2]],
    …
    [[r_1 in cluster K], [r_2 in cluster K],…, [r_n in cluster K]]
    ]

4.2 Procedures [1]

4.2.1 K-Means

1. According to the required number of clusters, K, program randomly pick up K initial centroids from data set.
2. Get all data points from input file, clusters.txt, into system. The data file contains 150 points of 2 dimensions' data set.
3. Calculate the distance for each data point and label the cluster:
    a. Calculate the distance between K centroids and each data point.
        i. Label the data point to shortest distance of cluster by identification variable "data_classification" in data structure.
        ii. If the distance is the same, the data point will belong to first shortest centroid.
        iii. Record down the shortest distance for each point base on exist/old centroid points.
4. Calculate the new centroids.
    a. Base on the new groups of data points, calculate new centroids.
    b. Calculate the total distance for each new centroid with its own group of data points.
5. Termination:
    a. Condition:
        i. If the improve of total distance is less than 1e-9, we stop the process.
    b. Else repeat step 3 ~ 5.

4.2.2 Expectation Maximization using Gaussian Mixture Model

1. Get all data points from input file, clusters.txt, into system. The data file contains 150 points of 2 dimensions' data points.
2. Call GMM_Machine(X, cluster_no) to initialize the class.
3. Assume the initial covariance matrix store in a list/array, sigma, is [[[1,0],[0,1]], [[1,0],[0,1]], [[1,0],[0,1]]] for 3 clusters.
4. Base on K-Means results to group data points into K initial clusters.
5. Create a weight storage list/array, "w", to record the weight of each cluster.
6. Create a cluster identification list/array, "c", to label the cluster of each data point.
    a. The default cluster value is the result of K-Means.
7. M-step: Calculate the log-likelihood:
8. E-step: Calculate the new Gaussian distributions.
    a. Calculate new Gaussian distributions to get mean, and covariance matrix of each Gaussian distribution.
9. Calculate the maximum log likelihood base on my Gaussian distributions.
10. Termination:
    a. Condition:
        i. If the improve of maximum log likelihood is less than threshold (1e-3), we stop the process.
    b. Else repeat step 7 ~ 9.

4.3 Optimizations and Challenges

4.3.1 Optimization: K-Means

- Calculate and store shortest distance and store in current_distance to reduce redundant calculation in termination condition.
- Base on the improvement of summary on distance of each centroid to make a best termination adjustment.

4.3.2 Optimization: EM-GMM

- Base on the results of K-Means as initial parameters for better prediction.

4.3.3 Challenge: K-Means

- How to pick up a good initialization point for each cluster is critical for this algorism. We randomly pick up 3 points may not have good final results. If resource available, we will pick points in data set and have a good distance for each random point.
- How to evaluate the best results. We try several runs to get the minimum distances in this data set. It may be improved as automatically run by a given number of trials.

4.3.4 Challenge: EM-GMM

- How to define data structure for multiple dimensions and perform matrix operation is the biggest challenge for us.
- We study and leverage numpy library to help us on this part to make program more easy to understand and implement.
- The matrix operation in Python is new for us. It takes a lot of time to confirm the definition on those multiple dimension variables and study how to use them.

5 Usage Instruction

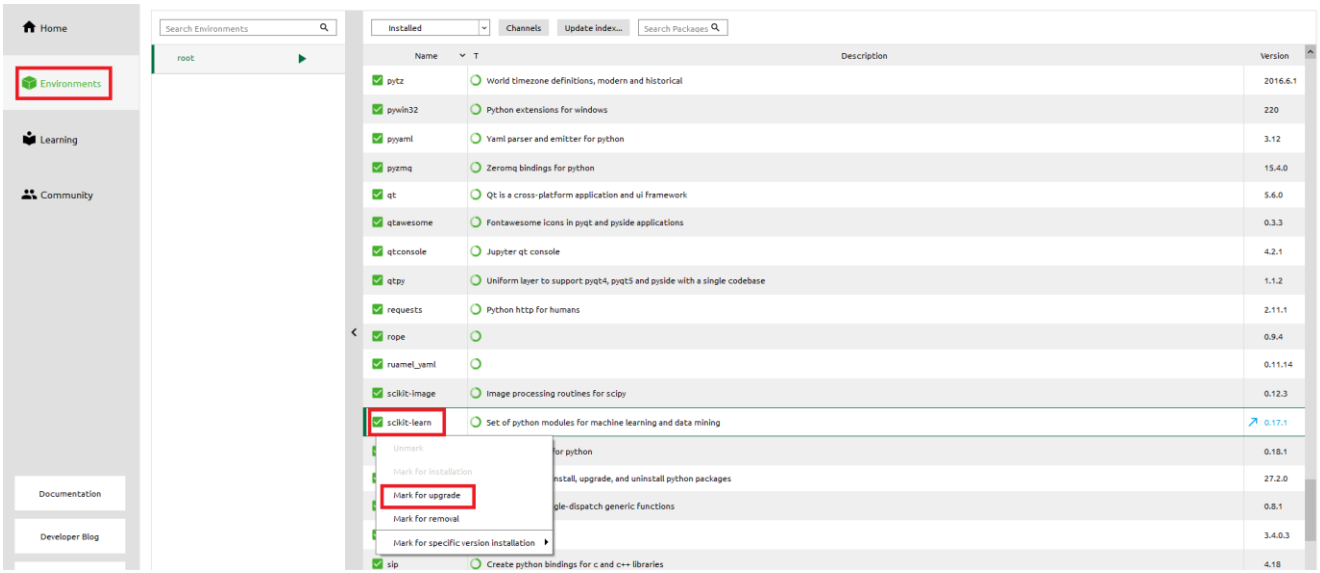
The implementation and clustering test code will leverage Python 3.5 with a lot of libraries. We recommend the latest “Anaconda version 4.2.0” data science platform with Python version 3.5 as execution environment. It is a library management environment and collects 175 libraries for developer to build up their own code. Spyder 3.0.0 includes in the platform as IDE.

The URL of this package is https://www.continuum.io/downloads

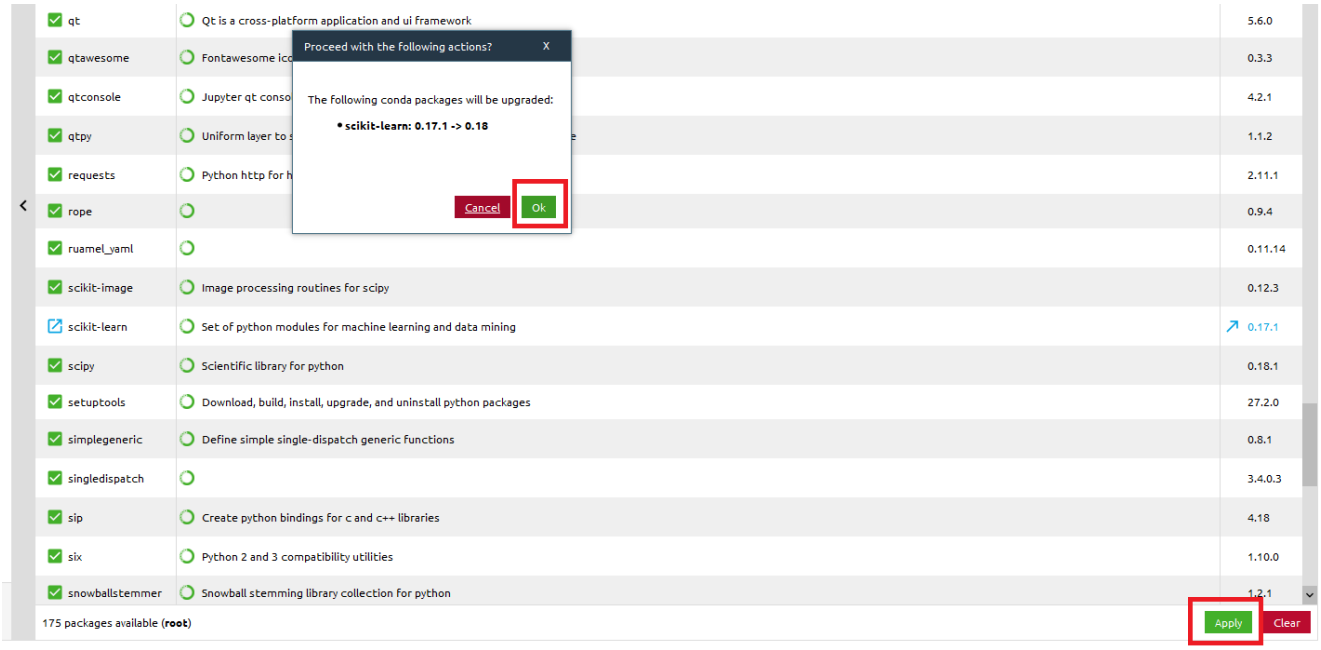
Alternatively, these program can be executed on Python 3.5 with scikit-learn version 0.18 environment. Below instructions are based on Anaconda v4.2.0 with Spyder v3.0.

5.1 Upgrade scikit-learn to version 0.18

Due to a new implementation of Gaussian Mixture module was included in the latest official release of scikit-learn version 0.18 which does not include in the latest Anaconda version 4.2.0. Please upgrade scikit-learn in Anaconda environment.



Apply the upgrade.

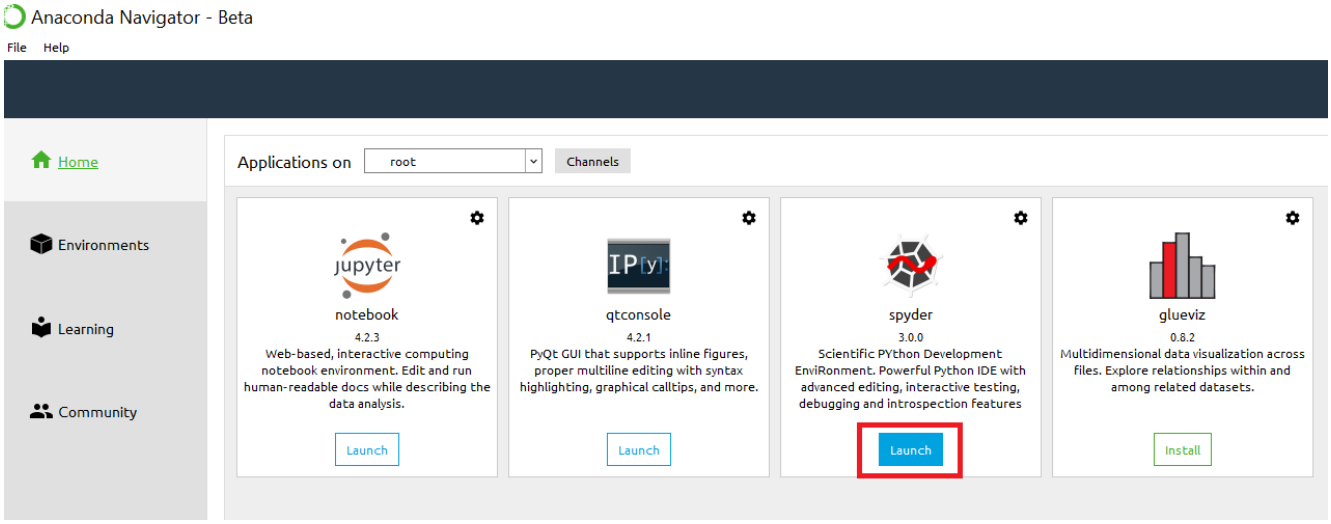


5.2 Script Files and Dataset File in same folder

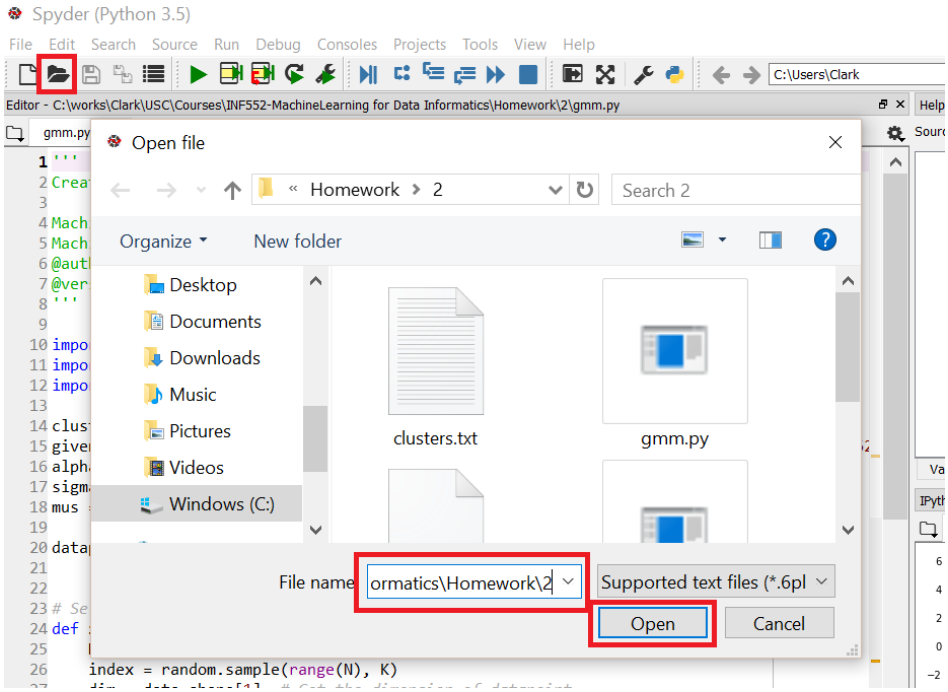
- kmeans.py (Implementation code)
- gmm.py (Implementation code)
- k-means_EM-GMM.py (Clustering testing code)
- clusters.txt

5.3 Execution from Spyder version 3.0

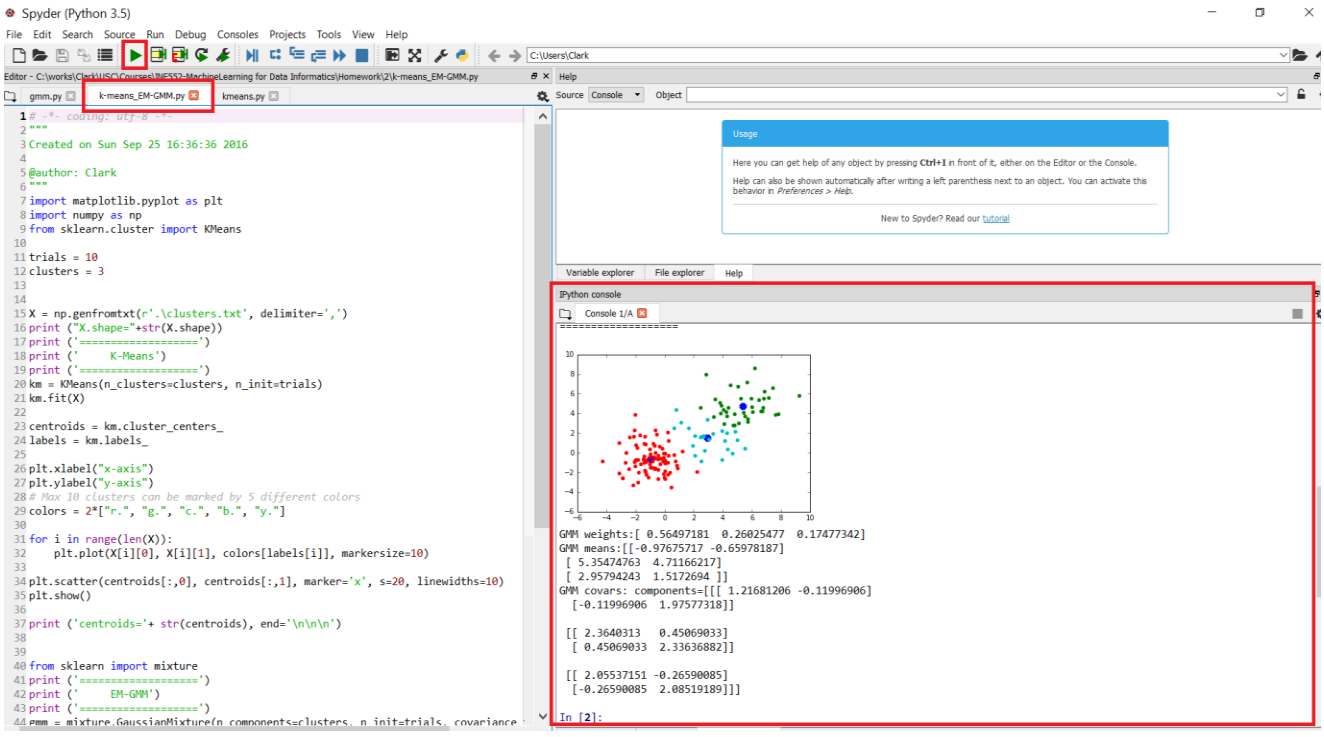
1. Under the “Home” of Anaconda Navigator, Click “Launch” to enable spyder 3.0.0 environment.



2. In Spyder IDE, please select open file icon on tool bar and open implementation and testing files.



3. Please select the implementation code you want to execute, then click the “Run file” icon. The result will show in IPython console window.



5.4 Execution Results

5.4.1 K-Means

After several trials, this is the best results we can get.

```
This program execute
```

```
Initial centroids:[[4.141706131, 1.222944183], [-2.17009237, -3.292317782],
[5.058353716, 6.7328904]]
new_distance:564.291858181973, current_distance:1277.6485043547532
new_distance:564.291858181973, current_distance:1277.6485043547532
new_distance:564.291858181973, current_distance:1277.6485043547532
new_distance:510.66373626226783, current_distance:516.8430146574296
new_distance:510.66373626226783, current_distance:516.8430146574296
new_distance:510.66373626226783, current_distance:516.8430146574296
new_distance:510.14922003575344, current_distance:510.4023009301893
new_distance:510.14922003575344, current_distance:510.4023009301893
new_distance:510.14922003575344, current_distance:510.4023009301893
new_distance:510.14922003575344, current_distance:510.1492200357535
Recursion time: 4
Final centroids: [[3.0831825570000002, 1.776213738], [-0.974765718, -0.684193041],
[5.6201657349999996, 5.0262263440000003]]
Weight: [0.20666666666666667, 0.5666666666666667, 0.22666666666666666]
```

5.4.2 EM-GMM

We leverage the best result from K-Means as initial parameters for EM-GMM model. The execution results as below:

```
This program execute

New_likelihood: -121.787491555
Enter E step.
Enter M step.
New_likelihood: -88.6022656054
Enter E step.
Enter M step.
New_likelihood: -88.0089974157
Enter E step.
Enter M step.
New_likelihood: -87.845752524
Enter E step.
Enter M step.
New_likelihood: -87.7758070463
Enter E step.
Enter M step.
New_likelihood: -87.7400201773
Enter E step.
Enter M step.
New_likelihood: -87.7200915891
Enter E step.
Enter M step.
New_likelihood: -87.7083945092
Enter E step.
Enter M step.
New_likelihood: -87.7012308752
Enter E step.
Enter M step.
New_likelihood: -87.6966574151
Enter E step.
Enter M step.
New_likelihood: -87.6936023015
Enter E step.
Enter M step.
New_likelihood: -87.6914542647
Enter E step.
Enter M step.
New_likelihood: -87.6898557768
```

```
Enter E step.
Enter M step.
New_likelihood: -87.6885935234
Enter E step.
Enter M step.
New_likelihood: -87.6875382016
Enter E step.
Enter M step.
New_likelihood: -87.6866104385
Recursion time: 15
The likelihood is: -87.6875382016
The amplitudes are: [ 0.19089455  0.56680006  0.24230539]
The means are: [[ 3.10891774  1.67404492]
 [-0.97569452 -0.64458101]
 [ 5.44055714  4.80565151]]
The covariances are: [[[  1.97947717e+00   1.07700353e-03]
  [  1.07700353e-03   2.38748205e+00]]

 [[  1.21202573e+00  -1.02968692e-01]
  [ -1.02968692e-01   2.00921936e+00]]

 [[  2.35584460e+00   3.74290084e-01]
        [  3.74290084e-01   2.28515063e+00]]]
```

6 Software Familiarization

6.1 Overview

As previous describe, we find a good data science platform, Anaconda, which includes major open source of Python libraries for data science. The latest version as of Oct., 1, 2016 is 4.2.0. The platform includes IDE (Integrated Development Environment), Python open source library management for developer to build up their own code.

Under this environment, a clustering testing code implements base on Numpy library, scikit-learn library, and matplotlib library as a comparison to our implementations.

6.2 KMeans in scikit-learn library

6.2.1 API introduction[4]

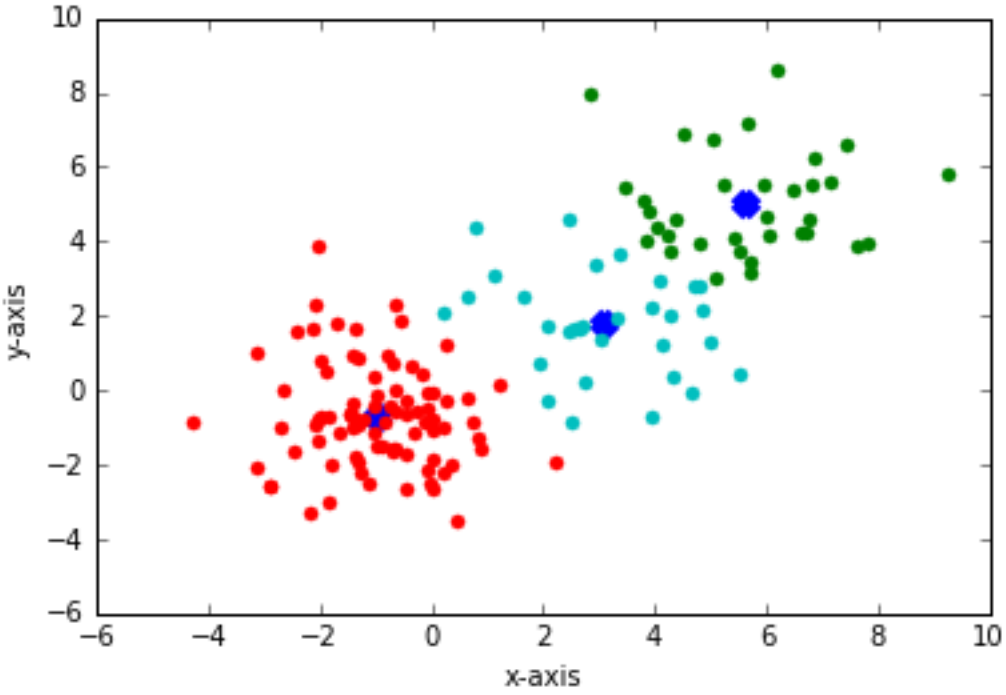
There are several interesting parameters can be assigned into this implementation.

1. Number of clusters, default is 8
2. “max_iter”: maximum iteration number can be restricted. Default is 300.
3. “Init”: {‘k-means++’, ‘random’ or an ndarray}, default is ‘k-means++’. The k-means++[2] will make sure all random points are well separated to make a better result. You can also provide your own initial grouping of data set as an array to perform calculation.
4. “Algorithm”: the parameter provides ‘auto’, ‘full’, or ‘elkan’, the classical EM style algorithm is ‘full’ which we used. The “elkan” variation[3] is more efficient approach by using the triangle inequality, but currently doesn’t support sparse data. “auto” chooses “elkan” for dense data and “full” for sparse data. default is ‘auto’.
5. “n_init”: the number of trials will perform. Number of time the k-means algorithm will be run with different centroid seeds. The best results are kept. The default value is 10.

6.2.2 Execution results

The best result after 10 trials as below, the result is the same as the best results of our implementation:

```
X.shape=(150, 2)
==================
     K-Means
==================
```




```
centroids=[[-0.97476572 -0.68419304]
 [ 5.62016573  5.02622634]
 [ 3.08318256  1.77621374]]
```

6.3 Gaussian Mixture in scikit-learn library

6.3.1 API Introduction[5]

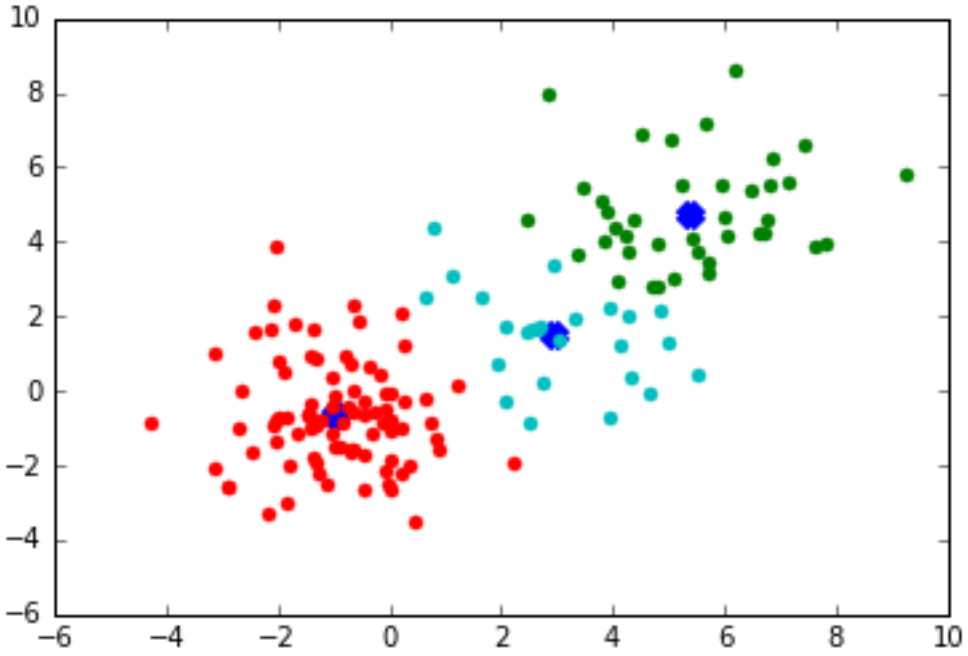
There are several interesting parameters can be assigned into this implementation.

1. The implementation provides 4 different “covariance type”[6] for developer to adopt.
    a. ‘full’, each cluster has its own general covariance matrix. Our case will fit into this covariance type.
    b. ‘tied’, all clusters share the same general covariance matrix.
    c. ‘diag’, each cluster has its own diagonal covariance matrix.
    d. ‘spherical’, each cluster has its own single variance.
2. “max_iter”: the number of EM iterations to perform. Default is 100.
3. “init_params”: The method used to initialize the weights. It provides two types of approach. {‘kmeans’, ’random’}. Default is kmeans.
4. “warm_start”: If warm_start is true; the solution of the last fitting is used as initialization for the next call. It will speed up convergence when developer repeat called several time on similar problems. The default value is False.
5. “n_init”: the number of trials will perform. The best results are kept. Default value is 1.

6.3.2 Execution results

The best result after 10 trials as below:

```
====================
      EM-GMM
====================
```




```
GMM weights:[ 0.56497181  0.26025477  0.17477342]
GMM means:[[-0.97675717 -0.65978187]
```

```
 [ 5.35474763  4.71166217]
 [ 2.95794243  1.5172694 ]]
GMM covars: components=[[[ 1.21681206 -0.11996906]
  [-0.11996906  1.97577318]]

 [[ 2.3640313   0.45069033]
  [ 0.45069033  2.33636882]]

 [[ 2.05537151 -0.26590085]
  [-0.26590085  2.08519189]]]
```

6.4 How to improve our implementation

6.4.1 Improvement opportunities on K-Means

1. Improve the random point selection. We can improve our code to split our data space into K areas, randomly select initial points in the area to have a better initialization.
2. Add retrial number parameters to automatically keep best results as output.
3. Maximum iteration can be included as one of termination criteria.
4. Different approach to speed up the process, like Elkan's approach, is advanced improvement item.

6.4.2 Improvement opportunities on EM-GMM

1. Improve the initialization process to automatically combine K-Means best results as initial input for EM-GMM algorithm. Or user can select random parameters as initial values for the algorithm.
2. Add retrial number parameters to automatically keep best results as output.
3. Maximum iteration can be included as one of termination criteria.
4. Multiple covariance types support is one of advanced improvement item.

7 Application

7.1 The applications on K-Means

Except on traditional data classification, K-Means usually adopt as initial parameter of EM-GMM. One of application for K-Means is colour image segmentation.[7] According to the research of siddheswar Ray and Rose H. Turi on “Determination of Number of Clusters in K-Means Clustering and Application in Colour Image Segmentation” [7], They present a simple validity measure based on the intra-cluster and inter-cluster distance measures which allows the number of clusters to be determined automatically.

Another interesting area to apply K-means is automatic network intrusion detection.[8] Jianliang, Meng, Shang Haikun, and Bian Ling. use the K-means algorithm to cluster and analyse the real network connection data to detect unknown intrusions efficiently.

7.2 The applications on EM-GMM

EM-GMM algorithm is widely applied into signal analysis area. Combine with different model like UBMs (universal background models), the algorithm can classify age and gender base on the voice in telephone[9], The other common application of GMM is background subtraction in computer vision task[10]. Based on a statistical model of the scene, an intruding object can be detected by spotting the parts of the image that don't fit the model.

8 Reference

[1] Alpaydin E. Introduction to machine learning[M]. MIT press, 2014.

[2] Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007.

[3] Elkan, Charles. "Using the triangle inequality to accelerate k-means." *ICML*. Vol. 3. 2003.

[4] scikit-learn, version 0.18, KMeans, URL http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans

[5] scikit-learn, version 0.18, Gaussian Mixture, URL http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture

[6] Erar, Bahar, "Mixture model cluster analysis under different covariance structures using information complexity. " Master's Thesis, University of Tennessee, 2011. http://trace.tennessee.edu/utk_gradthes/968

[7] Ray, Siddheswar, and Rose H. Turi. "Determination of number of clusters in k-means clustering and application in colour image segmentation." *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*. 1999.

[8] Jianliang, Meng, Shang Haikun, and Bian Ling. "The application on intrusion detection based on k-means cluster algorithm." *Information Technology and Applications, 2009. IFITA'09. International Forum on*. Vol. 1. IEEE, 2009.

[9] Bocklet, Tobias, et al. "Age and gender recognition for telephone applications based on gmm supervectors and support vector machines." *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008.

[10] Zivkovic, Zoran. "Improved adaptive Gaussian mixture model for background subtraction." *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 2. IEEE, 2004.