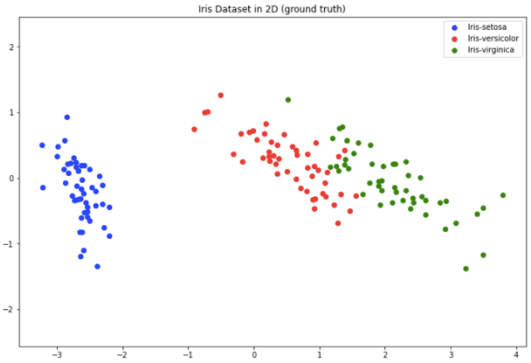
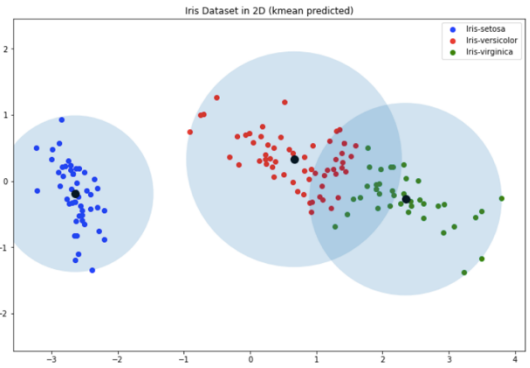
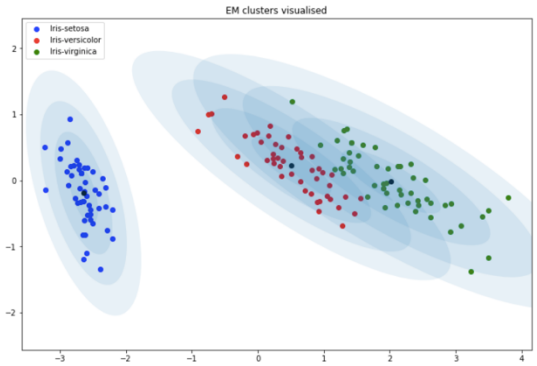
U1839931- Ajan Sittampalam

Report on the k-mean clustering and EM algorithm for clustering

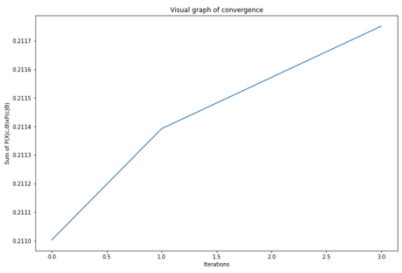


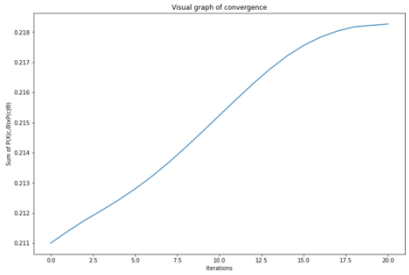


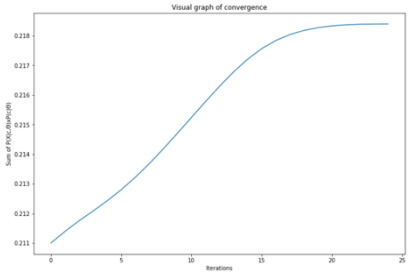
By comparing the ground truth data to the k-mean clustering's predictions, we can see that k-mean clustering is good at clustering data when they are far apart and distinct but fails to differentiate where the clusters may overlap. This is due the hard assignment of k-mean and the lack of variances of each cluster. Using this k-mean clustering I had an accuracy of 88.7% when compared with the ground truth. To try and improve this we could try and have a soft assignment of k-mean and introduce the variances, so that the region is an ellipse instead of a circle. To achieve this, I used a two-step algorithm, the Expectation Maximisation algorithm. When I implemented the EM algorithm, I got an accuracy of 98.0% when compared to the ground truth after having 20 iterations. This result shows us that using variances and using soft assignment for the clustering yielded a better accuracy for clustering.



It was also noted that after 20 iterations there was no more improvements in accuracy. To get this number of iterations, I used a tolerance to determine the number of iterations required before the results started to converge. The tolerance was a hyper parameter, which I had to look at some graphs to determine its value. The graphs below are the iterations against the difference in the sum of the probability of landing in each cluster in each iteration.







Tolerance = 0.001 Tolerance = 0.0001 Tolerance = 0.00001

As you can see from the graphs above, for the tolerance of 0.001 the line was straight indicating no convergence yet, whereas for the tolerance of 0.0001 you can just see the line plateau. For the lowest tolerance of 0.00001 you can see it has already plateaued. From this graph I decided to keep the tolerance at 0.0001 as this seems to be the optimal tolerance since 0.001 was too low to see any convergence and 0.00001 did not give any extra information that was needed.

The EM algorithm was implemented so that it would loop until it has converged. After this loop the final parameters was printed out. As expected, the pi0, pi1, and pi2 were all around 1/3 since there were equal amount of data for each label: 50 Iris-setosa, 50 Iris-versicolor and 50 Iris-virginica. Also looking at mu0, mu1, and mu2 we find out that the centers predicted by k-means is like the centers predicted by EM algorithm. Looking at the sigma0, sigma1 and sigma2 we can see that none of them have any large numbers indicating that the ellipses aren't enlarged to fit the whole data into one ellipse, and since the diagonals of the variance matrix are of similar it indicates there are no narrow ellipses. To run the EM algorithm, I had to get initial values for the parameters using k-means.

```
pi0: 0.30947121134728545
pi1: 0.35719545531939234
pi2: 0.33333333333332227
mu_0: [0.50957573 0.22506105]
mu_1: [ 2.02292953 -0.01719861]
mu_2: [-2.64084076 -0.19051995]
sigma_0:
 [[ 0.36455711 -0.21780862]
 [-0.21780862  0.18811298]]
sigma_1:
 [[ 0.5631686  -0.2932121 ]
 [-0.2932121   0.23224853]]
sigma_2:
 [[ 0.04777048 -0.05590782]
 [-0.05590782  0.21472356]]
total steps:  20
```

```
def initialize_random_params():
    params = {'pi0': x_y[relabel==0].shape[0]/x_y.shape[0],
              'pi1': x_y[relabel==1].shape[0]/x_y.shape[0],
              'pi2': x_y[relabel==2].shape[0]/x_y.shape[0],
              'mu0': recenters[0],
              'mu1': recenters[1],
              'mu2': recenters[2],
              'sigma0': np.cov(x_y[relabel == 0].T, bias= True),
              'sigma1': np.cov(x_y[relabel == 1].T, bias= True),
              'sigma2': np.cov(x_y[relabel == 2].T, bias= True)}
    return params
```

As you can see to get the initial values I used the predicted labels from k-mean (relabel) to estimate pi. The mu was estimated using the centers of the clusters predicted from k-mean (recenters). The sigmas were calculated by filtering the x_y data using the relabels and then getting the covariance of it.

By comparing the accuracy of k-mean, 88.7%, to the accuracy of EM algorithm,98%, we can clearly see that EM algorithm is more accurate. EM algorithm is also at a higher risk of overfitting since the variance could allow the ellipses to be very narrow. Since the EM algorithm and k-mean both come to its results by iterative methods, it can both give the result of a local minimum rather than the global minimum. Also, when looking at the accuracy we also must consider the initial values for these methods were good, since both of these methods are sensitive to their initial parameters.

The number of iterations it took for the EM algorithm to converge is 20. To get this we used the hyper parameter, tolerance, to determine when the clusters converged to its final positions. The tolerance was used so if the difference in the sum of the probability of landing in each cluster in each iteration, given by the denominator in the equation shown, was less than the tolerance then the while loop will break.

$$r_c^i \triangleq p\left(z_i = c|\mathbf{x}_i, \Theta\right) = \frac{p\left(\mathbf{x}_i|z_i = c, \Theta\right) p\left(z_i = c|\Theta\right)}{\sum_{c'=1}^{k} p\left(\mathbf{x}_i|z_i = c', \Theta\right) p\left(z_i = c'|\Theta\right)},$$

```
if len(iteration) > 2:
    if abs(iteration[-1] - iteration[-2]) < tolerance:
        i+=1
```