ex3

April 6, 2017

```
In [4]: import numpy as np
        from __future__ import division
        from scipy.stats import multivariate_normal

        #data = n x d
        #k = # clusters
        #max_iter = max iterations
        #conv_tol = tolerance for convergence
        #pi = k x 1
        #mean = d x k
        #z = n x k probability matrix
        #assign = n x 1 matrix


        def myGMM(data, k, max_iter, conv_tol):
            mean = np.zeros((data.shape[1], k))
            mixing_coeffs = np.full(k, 1/k)
            cov_matrices = np.zeros((data.shape[1], data.shape[1]))
            assign = np.zeros(data.shape[0])
            for i in range(0, k):
                mean[i] = np.mean(data[i], axis=1)
                cov_matrices[i] = np.identity(k)

            init_likelihood = np.sum(np.log2(np.sum(np.dot(mixing_coeffs,
                multivariate_normal.pdf(data, mean=mean, cov=cov_matrices)))))

            gamma = np.zeros(k)

            likelihoods = np.zeros
            for n in range(0, max_iter):
                for i in range(0, k):
                    gamma[i] = (np.dot(mixing_coeffs[i] * multivariate_normal.pdf(
                        cov=cov_matrices[i])))/(np.sum(np.dot(mixing_coeffs,
                        multivariate_normal.pdf(data, mean=mean, cov=cov_matrices))

                nk = np.sum(gamma)

                for i in range(0, k):
```

```
            mean[i] = (1/nk) * np.sum(gamma)
            cov_matrices[i] = (1/nk) * np.sum(np.dot(np.dot(gamma, (x - mea
            mixing_coeffs[i] = nk / data.shape[0]

        log_likelihood = np.sum(np.log2(np.sum(np.dot(mixing_coeffs,
            multivariate_normal.pdf(data, mean=mean, cov=cov_matrices)))))

        change = log_likelihood - init_likelihood
        init_likelihood = log_likelihood
```

In [ ]: