CSE 569 Fundamentals of Statistical Learning HW4 Solutions

Akshay Shah, 1215153316

Fall 2019

1 Solution

1.1 Code

```python

# coding: utf-8
import math
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans

def get_x(fname):
    f = open(fname)
    return np.array([int(x) for x in f.readlines()])

def initialize_params():
    theta = np.array([np.random.uniform(0,1) for i in range(3)])
    pi = np.array([np.random.uniform(0,1) for i in range(3)])
    return theta, pi

def E_step(data,pi,theta,no_of_clusters,no_of_tossses,no_of_experiments):
    si_mat = np.zeros((no_of_experiments,no_of_clusters))
    for i in range(no_of_experiments):
        factorial_cmp = math.factorial(no_of_tossses) // (math.factorial(no_of_tossses - no_of_clusters) * math.factorial(no_of_clusters))
        X = pi * factorial_cmp * (theta**data[i][0] * (1-theta)**(no_of_tossses - data[i][0]))
        si_mat[i] = X / np.sum(X)
    return si_mat

def M_step(si_mat,data,no_of_experiments):
    n_k = np.sum(si_mat,axis=0)
    theta = (1/(20 * n_k)) * np.sum(si_mat * data,axis=0)
    pi = n_k/no_of_experiments
    return pi,theta

def log_likelihood(data,pi,theta,no_of_clusters,no_of_tossses,no_of_experiments):
    ll = 0
    for i in range(no_of_experiments):
        factorial_cmp = math.factorial(no_of_tossses) // (math.factorial(no_of_tossses - data[i][0]) * math.factorial(data[i][0]))
        theta_cmp = theta**data[i][0] * ((1-theta)**((no_of_tossses - data[i][0])))
        ll += (np.log(np.sum(factorial_cmp * theta_cmp * pi,axis=0)))
    return ll

#Plotting neg log likelihood
```

```python
def plot_neg_likelihood_vs_iterations(iterations,likelihood_list):
    plt.plot(list(range(iterations-1)),likelihood_list)
    plt.title("No. of Iterations vs Negative Log Likelihood ")
    plt.xlabel("Iterations")
    plt.ylabel("Negative Log Likelihood")
    plt.show()

def EM(x,pi,theta,no_of_clusters,no_of_experiments,no_of_tosses):
    iterations = 0
    ll = log_likelihood(x,pi,theta,no_of_clusters,no_of_tosses,no_of_experiments)
    lls = [ll]
    while 1:
        si = E_step(x,pi,theta,no_of_clusters,no_of_tosses,no_of_experiments)
        pi,theta = M_step(si,x,no_of_experiments)
        ll = log_likelihood(x,pi,theta,no_of_clusters,no_of_tosses,no_of_experiments)
        iterations+=1
        if abs(lls[-1] - ll) < 0.00001:
            break
        lls.append(ll)

    lls = [-x for x in lls[1:]]
    return iterations,pi,theta,lls

def print_params(pi,theta):
    for i in range(3):
        print("Prior probability for coin%s:%s"%(i+1,pi[i]))
        print("Probability of head for coin%s:%s"%(i+1,theta[i]))

no_of_clusters = 3
no_of_tosses = 20
no_of_experiments = 1000

x = get_x("Binomial_20_flips.txt").reshape(-1,1)

#Kmeans experiment
kmeans = KMeans(n_clusters=3, random_state=0).fit(x)
cluster_assignments = kmeans.labels_

coin1_points = cluster_assignments[cluster_assignments == 0]
coin2_points = cluster_assignments[cluster_assignments == 1]
coin3_points = cluster_assignments[cluster_assignments == 2]

coin1_pi = coin1_points.shape[0] / no_of_experiments
coin2_pi = coin2_points.shape[0] / no_of_experiments
coin3_pi = coin3_points.shape[0] / no_of_experiments

theta1 =  kmeans.cluster_centers_[0][0] / no_of_tosses
theta2 =  kmeans.cluster_centers_[1][0] / no_of_tosses
theta3 =  kmeans.cluster_centers_[2][0] / no_of_tosses

pi_kmeans = [coin1_pi,coin2_pi,coin3_pi]
theta_kmeans = [theta1,theta2,theta3]

print("Estimated Parameters by Kmeans clustering algorithm")
print_params(pi_kmeans,theta_kmeans)

# pi = np.array([0.333, 0.333, 0.333])
# theta = np.array([0.5, 0.6, 0.4])

#Non Kmeans initialization
```

```python
pi,theta = initialize_params()
iterations,pi,theta,likelihood_list = EM(x,pi,theta,no_of_clusters,no_of_experiments,no_of_tosses
    )
print("#################################################")
print("Estimated Parameters by EM with random initialization")
print_params(pi,theta)
plot_neg_likelihood_vs_iterations(iterations,likelihood_list)

#Kmeans initialization
pi,theta = np.array(pi_kmeans),np.array(theta_kmeans)
iterations,pi,theta,likelihood_list = EM(x,pi,theta,no_of_clusters,no_of_experiments,no_of_tosses
    )
print("#################################################")
print("Estimated Parameters by EM with Kmeans initialization")
print_params(pi,theta)

plot_neg_likelihood_vs_iterations(iterations,likelihood_list)
```

Listing 1: Kmeans and Binomial Mixture Model

1.2 Outputs

Estimated Parameters by Kmeans clustering algorithm
Prior probability for coin1:**0.341**
Probability of head for coin1:**0.6653958944281524**
Prior probability for coin2:**0.212**
Probability of head for coin2:**0.2646226415094358**
Prior probability for coin3:**0.447**
Probability of head for coin3:**0.8731543624161094**
Estimated Parameters by EM with random initialization
Prior probability for coin1:**0.3146525869543541**
Probability of head for coin1:**0.6656711767117973**
Prior probability for coin2:**0.4827779465602027**
Probability of head for coin2:**0.852239260664435**
Prior probability for coin3:**0.2025694664854404**
Probability of head for coin3:**0.2586891442935458**
Estimated Parameters by EM with Kmeans initialization
Prior probability for coin1:**0.31530369152313953**
Probability of head for coin1:**0.6659352656434103**
Prior probability for coin2:**0.20261831177194986**
Probability of head for coin2:**0.2587294546111306**
Prior probability for coin3:**0.482077996704911**
Probability of head for coin3:**0.8523617131186518**

1.3 Plots

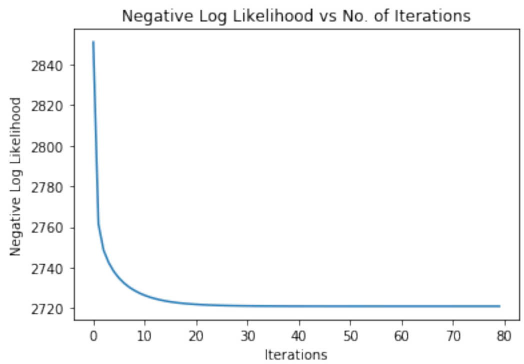

Figure 1: EM without KMeans

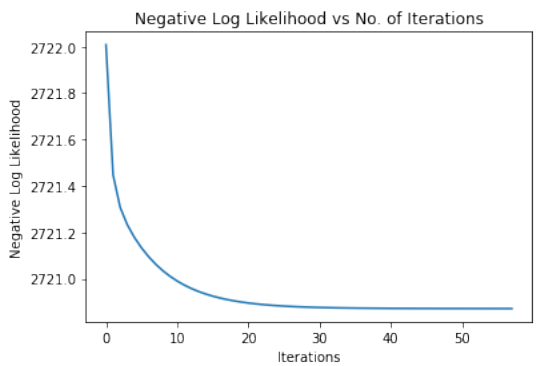

Figure 2: EM with Kmeans Initialization

1.4 Discussion

The EM algorithm is deterministic, meaning that upon repeated initializations, a given set of starting values will necessarily converge to the same solution.Hence,the start values become very important to obtain good final solution.The results from the experiment of EM with Kmeans parameter initialization got us very similar results to that of pure Kmeans implementation as well as closer than random values intialization. Additionally, we can also infer that although we get well formed clusters with both approaches, it might not be theoretically possible to get closer is actual ground truth with respect to the main coin experiment.Moreover, it is seen that EM with Kmeans converges much faster than EM with random initialization of parameters which can be seen in the plots which also show that curves depicted show similar trend of negative log likelihood barring actual values.We can see that negative log likelihood flattens out after 10-15 iterations in EM experiment. Overall key reflections of this exercise is we see how probability densities can be used to cluster set of points be it single cluster distribution or mixture density(in this case a mixture of binomial distribution) as depicted in the mixture model via EM algorithm.But it can be easily extended to mixture of gaussians with respective parameters update in the main algorithm. Mixture model allows to overcome the limitations of Kmeans such as hard cluster assignment which helps in real time use cases where you can have different datapoint aligned with different hidden topics/latent semantics, a prime method such as Latent Dirichlet Allocation helps to cluster hidden topics from set of documents. Thus we can see that the EM algorithm can be generalized to different domains based on the use case.