Homework 3

Course: Introduction to applied data science (PHYS247) Winter 2020
TA: Amartya Mitra Due Date: March 1, 11:59 p.m.

Problem 1: Homework 2!

In this problem, we want to use MCMC method to re-answer parts ”f” and ”g” of the previous homework. Last homework, we have created a 3D mesh-grid and have found the posterior for all the points. You noticed that even with a very low resolution in λ_1, λ_2 and W$_s$ space, it takes a long time to get the posterior. You found that you have to deal with the normalization of the posterior, but MCMC avoids computation of the normalization. Although we will use MCMC to get the posteriors in Bayesian framework, it can be used in more common situations where we want to sample from a complicated distribution. There are three common methods of sampling: Metropolis-Hastings, Gibbs sampling and Slice sampling. In this problem, we will use the Metropolis-Hastings algorithm to construct the posterior of the three parameters we had in homework 2. The algorithm is as follows:

1. Initialize all the variables (θ) and number of iterations (n)

For n iterations:

2. Find the posterior at point θ
3. Define a reasonable function to move from point θ to θ'
4. Find $r = \frac{\text{Posterior}(\theta')}{\text{Posterior}(\theta)}$
4. Generate a uniform random number ($rand$) between 0 and 1.
5. if $r > rand$; then move to new point θ', otherwise, stay in the same point θ

After implementing the above algorithm you will get a vector for each variable with the length of the number of iterations (n). The histogram of those vectors will give you the posterior distributions.

a) Write a Python code to implement the Metropolis-Hastings algorithm for the twitter problem in homework 2 and find the posterior distributions for λ_1, λ_2 and W_s. How do the initial value of parameters (the starting point) and number of iteration affect your results?
b) Are the posteriors consistent with those you found in the previous homework?
c) As you did in part g of homework 2, find $P((\lambda_2 - \lambda_1) > 5)$, the probability that Bob's weekly tweet counts have increased by five at some point.

Problem 2: K-means Clustering

In this problem we want to implement K-means clustering based on Lloyd's algorithm. The algorithm is as follows:

1- Start with k centers.

2- Cluster each point with the center nearest to it.

3- Find the centroid of each cluster and replace the set of old centers with the centroids.

4- Repeat the above two steps until the centers converge.

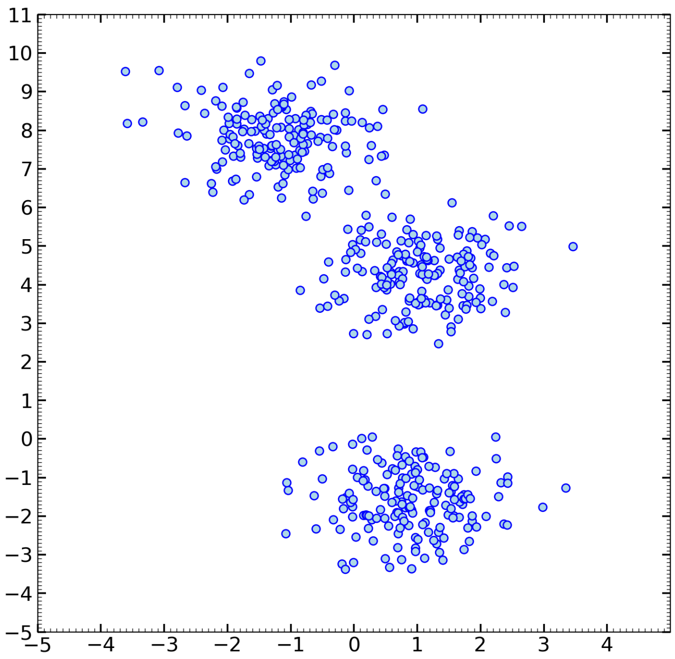
a) Write a python function that takes data points with two features (two dimensional data) and a number of clusters and returns partitioned data. Consider that the data points are located in Euclidean space.

We want to generate pseudo data and apply the code you have written in part a. Use the following code to generate the data:

```
from sklearn.datasets import make_blobs
X,Y = make_blobs(n_samples=500, cluster_std=0.8, centers=3, n_features=3, random_state=0)
```

You can plot the data using:

```
import matplotlib.pylab as plt
plt.scatter(X[:,0],X[:,1])
```



b) Considering 3 clusters, apply your function in part a to the dataset you have created.
c) Here we knew the number of clusters, but this is not the case in real problems. Explain a method where we can determine optimal number of clusters in K-means algorithm.
d) Apply your method in part c to the data you have generated and verify that the optimal number of the cluster for part b is three.