**Project 1. Deep Clustering**

**Project description**

The goal of this project is to get you familiar with one deep learning platform and use it to solve one of the most fundamental problems, data clustering, with the help of Autoencoders to map the data into a latent space so that clustering purity can be improved..

**Data set.**

1. Toy data set: mixture of three Gaussians data set. You can use codes like the following

**X = torch.cat((torch.randn(2000, 2),torch.randn(1500, 2)+13,torch.mm(torch.randn(1000, 2)+6,3\*(torch.rand(2,2)-0.2))),0)**

1. MNIST data sets. We will not use the image structure although this is image data set. Instead we will simply treat it as vectors. The data sets can be found <http://yann.lecun.com/exdb/mnist/>; or if you use pytorch you can follow the following.

import torchvision

import torchvision.transforms as transforms

import torchvision.datasets as datasets

train\_loader = torch.utils.data.DataLoader(

  torchvision.datasets.MNIST('/files/', train=True, download=True,

                             transform=torchvision.transforms.Compose([

                               torchvision.transforms.ToTensor(),

                             ])),

  batch\_size=**10000**, shuffle=False)

examples = enumerate(train\_loader)

batch\_idx, (example\_data, example\_targets) = next(examples)

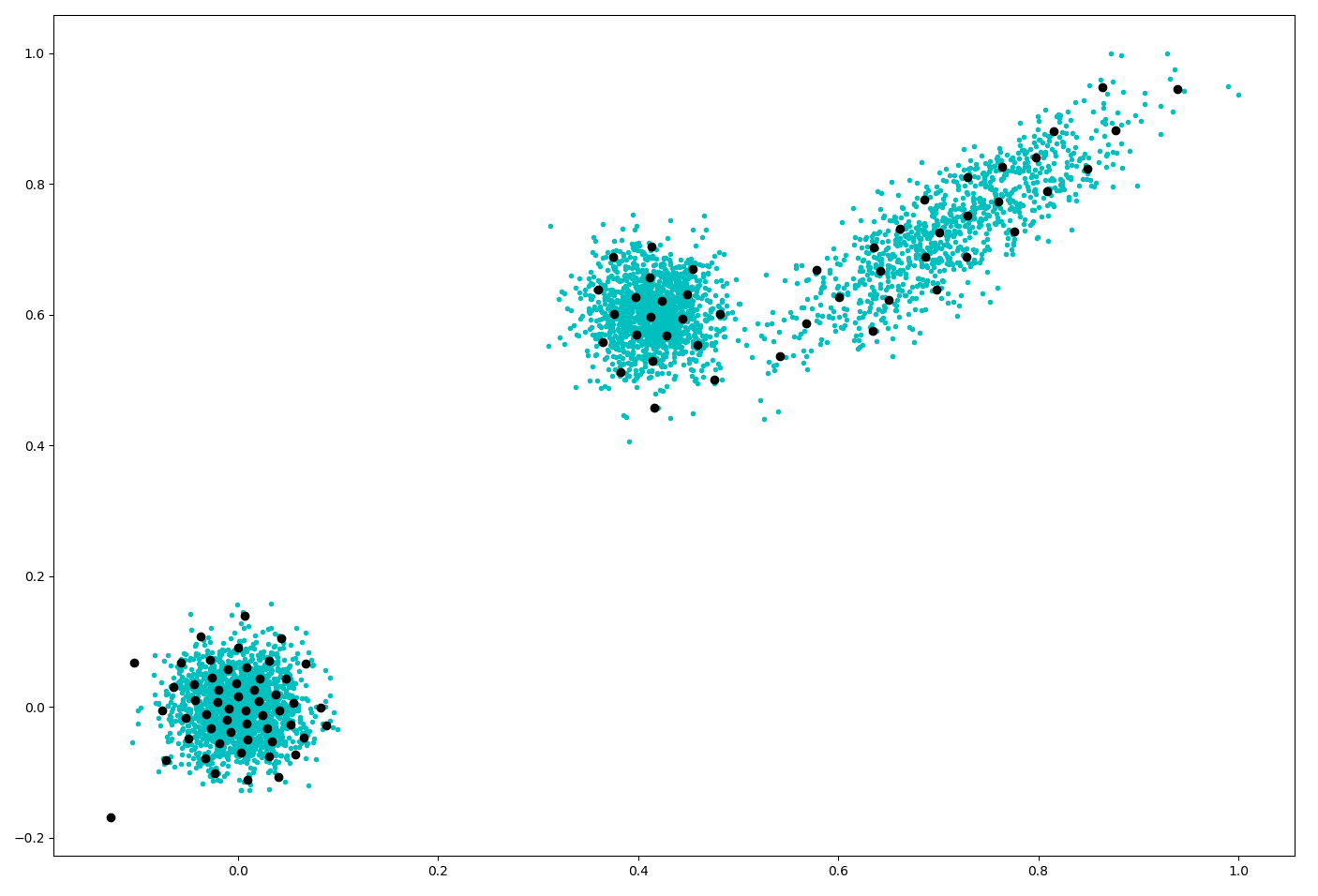
X = example\_data.view(-1,784)

Y = example\_targets

1. If you are interested, you can download other data sets used in the reading list papers (bonus).

**Tasks**

1. Implement the k-means algorithm on the toy data set. Plot the loss function curve over epochs, and also the cluster centers (like following)



1. Implement differentiable k-means clustering with k = 10 directly on the 784-dimensional images (training set of MNIST 60K images). Plot the evolution of the clustering accuracy over epochs, and report the average clustering accuracy over 10 random initialized runs.
   1. There are a number of variants of k-means clustering as has been discussed in the class, such as how to choose the variables to optimize, relations between variables if there are multiple of them, how to choose bandwidth, loss function, etc. please compare at least two variants and report the better one.
   2. You need to learn how to perform gradient updates and choose different optimization schemes using deep learning platforms (basically, just how to call related functions)
2. Deep clustering: Perform k-means clustering on the bottleneck layer of an auto-encoder.
   1. You need to test with different choices of: number of layers in AE; nonlinear activation functions, the dimension of each latent layers, and report the best configuration and clustering accuracies.
   2. You need to consider the scale of the bottleneck layer, and see whether there is a need to prevent it from diminishing.
   3. Use t-SNE to visualize the clusters you have obtained and see how they look like. (tSNE package <https://lvdmaaten.github.io/tsne/> and useful likes <https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1>)

You can google relevant posts on t-SNE freely.

1. Bonus task (can be very challenging, but optional):
   1. Bi-clustering. How to perform bi-clustering of the input data matrix. For MNIST, that means both samples and pixels will be clustered. (Hint: you may want to reformulate AE).
   2. Irregular shaped clusters. How to handle clusters of various shapes. (Hint. Revise the loss function)