CSCI 567 - Programming Assignment 3

Exercise 1: PCA and EM

The starter code for both parts of Exercise 1 can be accessed at **this link** with your school email address. It should also be fairly easy to code both ideas entirely from scratch—and please feel free to work on a dataset/distribution you are especially interested in! The ones provided are just obvious points :)

1. **PCA image compression** - In this exercise, you will be implementing Principal Component Analysis (PCA) from scratch (or, well… numpy) to compress and decompress images of handwritten digits from the MNIST dataset. You will start by normalizing the data, followed by computing the covariance matrix and its eigenvalues and eigenvectors. By sorting the eigenvectors and selecting the top principal components, you will project the data onto a lower-dimensional space, effectively compressing the images. Finally, you will reconstruct the images from the compressed data and compare them to the original images to observe the effects of PCA compression. Play around with different quantities of principal components to observe their effect on the decompressed data.
2. **Expectation maximization** - In this exercise, you will be implementing and comparing two clustering algorithms: the Expectation-Maximization (EM) algorithm for Gaussian Mixture Models (GMMs) and the K-Means algorithm. For the GMM part, you will generate synthetic data from multiple Gaussian distributions, initialize the parameters of the GMM, and then iteratively update them by alternating between the E-step, where you calculate the responsibilities of each Gaussian for each data point, and the M-step, where you update the parameters based on the calculated responsibilities. In the K-Means part, you will initialize cluster centers, assign each data point to the nearest center, and then update the centers to minimize the within-cluster variance. By comparing the results of the two algorithms, you will gain insights into their similarities and differences, and understand how they perform in clustering data with different shapes and sizes.

Exercise 2: Implementing and training VAEs

In this exercise, you'll be implementing a Variational Autoencoder (VAE) in PyTorch and training it on the MNIST dataset. VAEs were briefly covered in the 10/20 discussion section (see the course website for related slides), and the goal with this question is to quickly get some hands-on experience with the model architecture. You can find starter code at this link, which provides most of the needed boilerplate (e.g., setting up the dataset, training loop, etc).