Homework 6, Machine Learning, Fall 2023

***IMPORTANT* Homework Submission Instructions**

1. All homeworks must be submitted in one PDF file to Gradescope.
2. Please make sure to select the corresponding HW pages on Gradescope for each question
3. For all coding components, complete the solutions with a Jupyter notebook/Google Colab, and export the notebook (including both code and outputs) into a PDF file. Concatenate the theory solutions PDF file with the coding solutions PDF file into one PDF file which you will submit.
4. You must show work or give an explanation for every question. "Yes" and "No" are not sufficient answers. Always show the steps of your calculations. Only partial credit can be given if you do not explain your answer.
5. Failure to adhere to the above submission format may result in penalties.

All homework assignments must be your independent work product, no collaboration is allowed. You may check your assignment with others or the internet after you have done it, but you must actually do it by yourself. **Please copy the following statement at the top of your assignment file**:

Agreement: This assignment represents my own work. I did not work on this assignment with others. All coding was done by myself.

1 Constructing Kernels (Theory) 5 pts (Ishaan)

In class, we saw that by choosing a kernel

$$K(x_i, x_k) =< \Phi(x), \Phi(x) >_{\mathcal{H}_k},$$

we can implicitly map data to a high dimensional space, and have the SVM algorithm work in that space. One way to generate kernels is to explicitly define the mapping Φ to a higher dimensional space, and then work out the corresponding K. However in this question we are interested in direct construction of kernels. Let K be a kernel over $\mathbb{R}^n \times \mathbb{R}^n$, let $a \in \mathbb{R}$. $< c, d >$ denotes the dot product, $c^T d$. $\Phi(x)$ represents a mapping function that transforms the original data point x into a higher-dimensional space, and $\mathcal{H}_k$ refers to a Hilbert Space associated with the kernel k (the space where the kernel function operates).

For each of the function K below, state whether it is necessarily a kernel. If you think it is, prove it; Otherwise, please specify a counterexample.

(a) $K_1(x, z) = aK(x, z)$ for $a > 0$
(b) $K_2(x, z) = aK(x, z)$ for $a < 0$
(c) $K_3(x, z) =< x, z >^3 +(< x, z > -1)^2$
(d) $K_4(x, z) =< x, z >^2 + \exp(-\|x\|^2) \exp(-\|z\|^2)$

2 VC Dimension, Pattern Counting, and Kernels (Theory) 15 pts (Harry)

VC dimension is an important tool in theoretically guaranteeing the efficacy of various ML models. In this problem, we examine the VC dimension of the hypothesis space of linear models. We then show a generalization that allows us to calculate how well a linear model can perform even past the point of shattering.

(a) Find the VC dimension of the hypothesis space $\mathcal{F}_d = \{f \colon \mathbb{R}^d \to \{-1, 1\}, f(x) = \text{sgn}(w^T x), w \in \mathbb{R}^d\}$ of linear models without a constant term. Prove your answer.

Hints: To show that the VC dimension is k, you must show that there exists a set of k points that can be shattered, and you must also show that there exists no set of $k + 1$ points that can be shattered; Use the fact that any set of $k + 1$ vectors in a k-dimensional vector space must be linearly dependent.

(b) We wish to show that, under some simple assumptions, a linear model in $\mathcal{F}_d$ can classify n points in exactly $2\sum_{k=0}^{d-1} \binom{n-1}{k}$ different ways *no matter how the points are arranged*. We will do this by showing that this problem can be reduced to two other problems, one of which is much easier to solve. We start with some definitions. Suppose that we have some d-dimensional vector space V.

- We define a *hyperplane* to be a $(d-1)$-dimensional subspace in V. For example, the set of hyperplanes in $V = \mathbb{R}^3$ is the set of 2-dimensional planes intersecting the origin.
- Given a set of n linear functions $f_j : V \to \mathbb{R}$, suppose that each f_j is not always zero. Then, it can be shown that the set of solutions to $f_j(x) = 0$ forms a hyperplane S_j of V. We then say that the *cells* of the arrangement of hyperplanes $\{S_1, \ldots, S_n\}$ are the regions of V divided by the hyperplanes. More rigorously, a nonempty subset $C \subset V$ is a *cell* if and only if there exists some $p \in \{-1, 1\}^n$ such that $C = \{x \in V \colon \text{sgn}(f_j(x)) = p_j \ \forall j\}$ where sgn denotes the sign function.

- The *orthants* of $\mathbb{R}^d$ are a generalization of the concept of quadrants in $\mathbb{R}^2$ and octants in $\mathbb{R}^3$. For each $p \in \{-1, 1\}^d$, we define the *orthant* corresponding to p as the set of vectors $\{x \in \mathbb{R}^d \colon \operatorname{sgn}(x_j) = p_j \ \forall j\}$. Note that there are 2^d orthants in $\mathbb{R}^d$.
- Given a collection of n vectors $(x_i)_{i=1}^n$ where $x_i \in V$, an element $p \in \{-1, 1\}^n$ is a *pattern* of the collection if and only if there exists a linear function $f : V \to \mathbb{R}$ such that $\operatorname{sgn}(f(x_i)) = p_i$ for each i.
- We say that a set of n vectors in V is in *general position* if, for all $0 \leq k \leq d$, any subset of k of the vectors is linearly independent. Note that this is not the same as all n vectors being linearly independent. For example, $\{(1, 0), (0, 1), (1, 1)\} \subset \mathbb{R}^2$ is in general position but it is not linearly independent.

We now present the three problems we will be working with. Suppose that n and d are fixed and $n \geq d \geq 1$.

- The cell counting problem supposes that we have a collection of n hyperplanes in a d-dimensional real vector space V (and corresponding linear functions $f_j : V \to \mathbb{R}$). Then, the problem asks for the number of cells in the resulting arrangement. It can be proven using recursion that there at most

$$C(n, d) = 2 \sum_{k=0}^{d-1} \binom{n-1}{k}$$

  cells, but you may use this result directly.
- The orthant intersection problem supposes that we have a d-dimensional subspace S in $\mathbb{R}^n$. Then, the problem asks for the number of orthants of $\mathbb{R}^n$ that S intersects.
- The pattern counting problem supposes that we have n vectors in $\mathbb{R}^d$ in general position, and the problem asks for the number of distinct patterns of the set of vectors.

We now show that we can reduce the pattern counting problem to the orthant intersection problem, and that we can reduce the orthant intersection problem to the cell counting problem. Note that when you reduce problem A to problem B, you must show a correspondence between the answer to A and the answer to B, and you must show that the assumptions and conditions of B are satisfied. This way, you can cite the answer to B to solve A.

1. Given that S intersects at least one orthant, prove that the answer to the orthant intersection problem is at most $C(n, d)$ by reducing it to the cell counting problem.

   Hints: Let $V = S$, and consider the linear mappings $f_j : S \to \mathbb{R}$ where $f_j(x) = e_j^T x$ for each $j = 1, \ldots, n$; Make sure to show that each f_j is not always zero.

2. Prove that the answer to the pattern counting problem is at most $C(n, d)$ by reducing it into the orthant intersection problem. Make sure to show that the subspace you construct has dimension d and that it intersects at least one orthant.

   Hints: Since the patterns correspond to the outputs of the dot product of each point with a given weight vector, try consolidating the dot products into one matrix multiplication; Recall the relationship between the column space of a matrix and its rank.

3. (Optional, no points) Let $C(n, d)$ be the least upper bound to the solution of the cell counting problem over all choices of n hyperplanes in d dimensions. Prove that the formula $C(n, d) = 2 \sum_{k=0}^{d-1} \binom{n-1}{k}$ is indeed correct by showing that the recurrence

$$C(n, d) = C(n-1, d) + C(n-1, d-1)$$

holds when $n, d \geq 2$ with base cases $C(1, d) = C(n, 1) = 2$. Note that while we stated earlier that $n \geq d$, the formula holds when $n < d$ also. This problem is not for points (bonus or otherwise) and will not be graded, it is just for fun.

If we are more careful with our math, then we can actually show that the answer to the pattern counting problem is *exactly* $C(n, d)$ (try to prove this!). Furthermore, if we relax the assumption of the pattern counting problem so that the vectors need not be in general position, then the number of patterns is still bounded above by $C(n, d)$. We can now analyze some consequences of this result.

4. Use the fact above to give a **short** alternative proof for the VC dimension of $\mathcal{F}_d$.

   Hints: Use the fact that $\sum_{k=0}^{m} \binom{m}{k} = 2^m$; Recall that a set of m points is shattered if the number of patterns of the set of points is 2^m.

5. Suppose that d is held constant. What is the Big-O growth rate of $C(n, d)$ with respect to n when $n \geq d$? What does this say about the efficacy of linear models on (possibly very unrealistic) datasets with a large number of points?

(c) Suppose that we have a dataset $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$, and $n > d$, and suppose that we have some nonlinear kernel k with corresponding RKHS H_k. Furthermore, let $\ell : \mathbb{R}^2 \to \mathbb{R}$ be an arbitary loss function, and let $\Omega : \mathbb{R} \to \mathbb{R}$ be nondecreasing. By the Representer Theorem, we know that minimizing the empirical cost $\sum_{i=1}^n \ell(f(x_i), y_i) + \Omega(||f||)$ over H_k is equivalent to minimizing the empirical cost over the set of linear functions $S_k = \{f \in H_k \colon f = \sum_{i=1}^n \alpha_i k(x_i, \cdot), \alpha_i \in \mathbb{R}\}$. For a given $x \in \mathbb{R}^d$, what is the dimension of the vector $(k(x_1, x), k(x_2, x), \ldots, k(x_n, x))$? Referencing **(b)**, why does this suggest that the dataset is more likely to be separable using the kernel trick with an SVM as opposed to simply using an SVM directly?

3 Logistic Regression, Kernels, and Optimization (Theory) 10 pts (Allan)

Let $\{(x_i, y_i)\}_{i=1}^n$ be a set of training data, where $x_i \in \mathbb{R}^d$ for all i, and $y_i \in \{-1, 1\}$. Consider the l_2 regularized logistic regression model with parameter θ, where we want to find an f_θ that minimizes the loss function:

$$\sum_{i=1}^{n} \ln(1 + \exp(-y_i(f_\theta(x_i))) + \lambda \|f_\theta\|_{\mathcal{H}}^2$$

where $f_\theta(x)$ is of the form $\theta^T x$, and $\|f_\theta\|_{\mathcal{H}}^2 = \theta^T \theta$.

1. Let $\mathcal{H}$ be the reproducing kernel Hilbert space that corresponds to the above l_2 regularized logistic regression. Using the representer theorem, what is the form of the optimal predictive function?

2. Define the function g as $g(\zeta) = \ln(1 + \exp(-\zeta))$. It turns out that l_2 regularized logistic regression is closely related to the following primal optimization problem:

   $$\min_{\omega, \zeta} \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{n} g(\zeta_i)$$

   subject to:

   $$y_i(\omega^T x_i) \geq \zeta_i, \forall i$$

where $C > 0$ is a constant.

Derive the dual formulation of this problem (you are not required to solve the problem, just state it). You should simplify the dual formulation so that it only depends on α, the y_i's and x_i's and C.

4 Attention and Convolution (Coding) 10 pts (Jon)

In this question, we will use the PyTorch framework to explore some properties of self-attention and some connections between self-attention and convolution[1]. First, we define the notion of a "similarity metric", which can be thought of as the opposite of a distance metric. Formally, we call a function $s : \mathbb{R}^{d \times d} \to \mathbb{R}$ a similarity metric if, for any $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^d$:

1. s is bounded above by some value b: $s(\mathbf{x}, \mathbf{y}) \leq b$.
2. The similarity between an object and itself is the maximum similarity possible: $s(\mathbf{x}, \mathbf{x}) = b$
3. The similarity between two distinct objects is not the maximum similarity: $\mathbf{x} \neq \mathbf{y} \implies s(\mathbf{x}, \mathbf{y}) < b$
4. Similarity is symmetric: $s(\mathbf{x}, \mathbf{y}) = s(\mathbf{y}, \mathbf{x})$

(a) When learning about convolution, you may have learned that it is similar to template matching, where we find the similarity between a template and an image at each location. In the first part of this problem, we will use the PyTorch package to visually explore whether convolution is computing a similarity function. First, recall the definition of convolution. For an input matrix $\mathbf{Z} \in \mathbb{R}^{d_{\text{in}} \times h_{\text{in}} \times w_{\text{in}}}$ and a learned tensor of d_{out} convolutional filters $\mathbf{K} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}} \times h_{\text{kernel}} \times w_{\text{kernel}}}$, the output of a simple 2 dimensional convolution at position h, w for filter j is:

$$(\mathbf{Z} * \mathbf{K})_{j,h,w} = \sum_{a=1}^{d_{\text{in}}} \sum_{b=1}^{h_{\text{kernel}}} \sum_{c=1}^{w_{\text{kernel}}} k_{j,a,b,c} z_{a,h+b-\lfloor h_{\text{kernel}/2} \rfloor, w+c-\lfloor h_{\text{kernel}/2} \rfloor}, \tag{1}$$

where the $*$ operator denotes convolving the first tensor with the second. For simplicity, in this problem we will consider convolution with $h_{\text{kernel}} = w_{\text{kernel}} = 1$, simplifying Equation 1 to

$$(\mathbf{Z} * \mathbf{K})_{j,h,w} = \sum_{a=1}^{d_{\text{in}}} k_{j,a,1,1} z_{a,h,w} \tag{2}$$

We will also consider $d_{\text{in}} = 3$ to be the RGB representation of a pixel at each location, allowing easy visualization. The starting notebook available at this link provides an input tensor and a kernel tensor. Use this starting notebook to:

1. Visualize the provided input tensor using Matplotlib's imshow function. Additionally, visualize each filter in the given "kernels" tensor as a separate subplot in one figure. Note that PyTorch and Matplotlib follow different conventions around the semantics of each axis in a tensor. In PyTorch, an image is represented with the channel dimension (e.g., RGB) first, followed by height and width. In Matplotlib, the convention is height and width followed by channel. To display torch tensors using matplotlib, you will need to reorder the axes using torch.permute such that the channel dimension comes last rather than first. Note that the color of each filter appears in the input tensor.
2. Use the PyTorch function torch.nn.functional.conv2d to convolve the given input tensor with the given kernels. This should produce an output tensor in $\mathbb{R}^{d_{\text{out}} \times h_{\text{in}} \times w_{\text{in}}}$. Display the activation map for each kernel as a subplot of one plot, i.e., produce d_{out} subplots of shape $h_{\text{in}} \times w_{\text{in}}$. Based on these visualizations, you should notice that the entries in these matrices are not produced by a similarity function. State which of the rules for similarity functions is violated and how you know.

[1] In this problem "convolution" refers to the operation used in convolutional neural networks. Formally, we are actually going to discuss cross-correlation, but the machine learning community refers to this operation as convolution by convention.

3. Let's take a moment to consider something that *is* a similarity function. For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the cosine similarity between $\mathbf{x}$ and $\mathbf{y}$ is defined to be:

$$\text{CosSim}(\mathbf{x}, \mathbf{y}) := \cos(\theta(\mathbf{x}, \mathbf{y})), \tag{3}$$

where $\theta : \mathbb{R}^{d \times d} \to [0, 2\pi)$ is a function that gets the angle between two vectors. Prove that CosSim is a similarity metric. Hint: θ is a distance metric.

4. Implement CosSim using a processing step on $\mathbf{Z}$ and $\mathbf{K}$ and torch.nn.functional.conv2d; that is, perform a transformation T on $\mathbf{Z}$ and $\mathbf{K}$ such that $(\bar{\mathbf{Z}} * \bar{\mathbf{K}})_{j,h,w}$ computes a cosine similarity, where $\bar{\mathbf{Z}} = T(\mathbf{Z})$ and $\bar{\mathbf{K}} = T(\mathbf{K})$. Display the activation map for each kernel. Do the activation maps reflect a similarity function? What is the difference between what you just implemented and a normal convolution? Hint: Recall that $\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|_2\|\mathbf{y}\|_2 \cos(\theta(\mathbf{x}, \mathbf{y}))$.

(b) We now attend to attention. Recall the definition of dot product attention as computed in "Attention is All You Need". For matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{w_{\text{in}} \times d_{\text{in}}}$, we have

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) := \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{\text{in}}}}\right)\mathbf{V}. \tag{4}$$

In the simplest form of self-attention, an input matrix $\mathbf{Z} \in \mathbb{R}^{w_{\text{in}} \times d_{\text{in}}}$ is passed as all three arguments, i.e.

$$\text{SimpleSelfAttention}(\mathbf{Z}) := \text{Attention}(\mathbf{Z}, \mathbf{Z}, \mathbf{Z}) = \text{softmax}\left(\frac{\mathbf{Z}\mathbf{Z}^T}{\sqrt{d_{\text{in}}}}\right)\mathbf{Z}.$$

1. Note that you were given an input image in $\mathbb{R}^{d_{in} \times h_{in} \times w_{in}}$, which does not quite line up with the dimensions expected for attention. Flatten the given input into a matrix of shape $(d_{in},\ h_{in}w_{in})$, transpose the resulting matrix to shape $(h_{in}w_{in},\ d_{in})$, and use it to compute $\mathbf{Z}\mathbf{Z}^T$. Visualize the resulting matrix with imshow. Based on this visualization, could the entries in the resulting matrix be produced by a similarity function? State whether one of the rules for similarity functions is violated and how you know.

2. Compute softmax $\left(\frac{\mathbf{Z}\mathbf{Z}^T}{\sqrt{d_{\text{in}}}}\right)$, and visualize the resulting matrix. Based on this visualization, could the entries in this matrix be produced by a similarity function? State whether a rule for similarity metrics is violated and how you know.

3. Let $\bar{\mathbf{Z}} := \begin{bmatrix} 1/\|\mathbf{z}_{:,1}\|_2 & 0 & \dots & 0 \\ 0 & 1/\|\mathbf{z}_{:,2}\|_2 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & 1/\|\mathbf{z}_{:,w_{\text{in}}}\|_2 \end{bmatrix}$. Finally, compute softmax $\left(\frac{\bar{\mathbf{Z}}(\mathbf{Z}\mathbf{Z}^T)\bar{\mathbf{Z}}}{\sqrt{d_{\text{in}}}}\right)$, and visualize the resulting matrix. Based on this visualization, could the entries in this matrix be produced by a similarity function? State whether one of the rules for similarity functions is violated and how you know.

4. Based on the prior questions, you can see a connection between attention and convolution. Implement the function $f(\mathbf{Z}) := \frac{\bar{\mathbf{Z}}(\mathbf{Z}\mathbf{Z}^T)\bar{\mathbf{Z}}}{\sqrt{d_{\text{in}}}}$ using a convolution; reshape the output to a matrix of the appropriate size, and visualize the result. Hint: Recall that $\frac{\bar{\mathbf{Z}}(\mathbf{Z}\mathbf{Z}^T)\bar{\mathbf{Z}}}{\sqrt{d_{\text{in}}}} \in \mathbb{R}^{w_{\text{in}}h_{\text{in}} \times w_{\text{in}}h_{\text{in}}}$ and $(\mathbf{Z} * \mathbf{K}) \in \mathbb{R}^{d_{\text{out}} \times h_{\text{in}} \times w_{\text{in}}}$. How many convolutional kernels will you need to make these dimensions match?

5 K-Means and GMM (Coding) 15 pts (Yiyang)

In this problem, we will explore the Gaussian Mixture Model and K-Means clustering algorithm using the mall customers dataset. This dataset consists of 200 samples with two features: `annual income` and

`spending score`. Note: Before doing clustering for the dataset, please use the `MinMaxScalar` to normalize the entire dataset. Since there is no train and test split in this problem, there is no need to worry about data leakage. Please set up the random seed as 42 to keep the reproducibility of your result.
For the GMM model, suppose we assume that the dataset follows a Gaussian Mixture Model density in two-dimensional feature space, which we wrote as

$$g(x) = \sum_{k=1}^{K} \omega_k g_k(x)$$

where g_k is the normal distribution distribution with mean μ_k and 2×2 covariate matrix Σ_k for each class $k = 1, 2, \cdots K$, K is total number of classes, and $\omega_k \geq 0$ is the weight of each class where $\sum_{k=1}^{K} \omega_k = 1$. Here $\{\mu_k, \omega_k \Sigma_k\}$ are unknown parameters, and we would like to fit this mixture model.
As we have learned in class, the log-likelihood of the data is

$$l(\mu_k, \omega_k, \sigma^2) = \log\left(\prod_{i=1}^{N} g(x_i)\right) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \omega_k g_k(x_i)\right)$$

For the Expectation step, it computes the responsibility of x_i for class k for the t^{th} step as:

$$r_{i,t+1}^{(k)} = \frac{\omega_{k,t} g_{k,t}(x_i)}{\sum_{m=1}^{K} \omega_{m,t} g_{m,t}(x_i)} \tag{5}$$

where $g_{m,t} = N(\mu_m, \Sigma_m), m = 1, \cdots K$
For the maximization step, it computes the weighted means and covariances as:

$$\mu_{k,t} = \frac{\sum_{i=1}^{N} r_{i,t}^{(k)} x_i}{\sum_{i=1}^{N} r_{i,t}^{(k)}}, \Sigma_{k,t} = \frac{\sum_{i=1}^{N} r_i^{(k)} (x_i - \mu_{k,t})(x_i - \mu_{k,t})^T}{\sum_{i=1}^{N} r_{i,t}^{(k)}}$$

and the updated weights for $g_{k,t}(x)$ as

$$\omega_{k,t} = \frac{\sum_{i=1}^{N} r_{i,t}^{(k)}}{N}$$

Hint: When updating the covariate matrix, remember to add a regularizer to avoid making it a singular matrix. For example, you can add a matrix $10^{-6} \cdot I_{2\times 2}$ each time when updating $\Sigma_{k,t}$.

(a)[Gaussian Mixture Model] Based on the formulas for the EM algorithm above, please implement this GMM algorithm in Python from scratch, The arguments of your function are the number of clusters K, the maximum number of iterations `max_iter`, the tolerance of log-likelihood differences between two iterations `tol`. You are allowed to use any functions in `numpy` and `scipy` packages to implement the GMM Algorithm (you might need `scipy.stats.multivariate_normal.pdf()` to help you generate the Probability density function of $g_{k,t}(x_i)$). You are allowed to use `pandas` to load the dataset and `sklearn` package for min-max scaling the dataset. Set the initial weight of each class $\omega_{k,0} = \frac{1}{K}$, the initial mean of each class as a randomly sampled point in the dataset, and the initial covariates matrix as $0.01 I_{2\times 2}$ to ensure reproducibility, where $I_{2\times 2}$ is an identity matrix. Set the default maximum iteration number as 100, and `tol` as 10^{-4}. After implementing the GMM Algorithm, please answer the following questions:

1. Set up $K = 1$ to 10, and plot the relationship between K and the negative log-likelihood of the dataset. Based on the plotted graph, Can you select a proper K for this dataset?

2. Using the proper K you selected, train your GMM model based on the selected K, and plot the scatter plot of the dataset as well as the density estimation for this Gaussian mixture model. You can find more information on how to draw the density estimation for a Gaussian Mixture in the scikit-learn document. The `score_samples` function in Gaussian Mixture computes the log-likelihood of each sample. Hint: when you update the covariate matrix, make sure you add a small regularization matrix

$10^{-6} \cdot I_{2\times 2}$ into your updated covariates matrix in your GMM implementation. Otherwise, your graph would look a little bit messy.

3. Use the `GaussianMixture` function with the same K, the covariance type as `full`, and the initial params as `kmeans` in sklearn, and plot the density estimation for the Gaussian Mixture. Does it show similar results?

(b)[K-Means Algorithm] Please implement a K-Means algorithm using Euclidean distance from scratch. Your function's arguments are the number of clusters K, the maximum number of iterations `max_iter`, and the tolerance of the mean distance from the sample to their closest cluster differences between two iterations `tol`. You can initialize each center by randomly selecting a point from the input data. You are allowed to use any functions in `numpy` and `scipy` packages to implement K-Means. After implementing the K-Means Algorithm, please answer the following questions:

1. Set up $K = 1$ to 10, and plot the relationship between K and the mean distance between the cluster centroids and the assigned samples to the dataset. Based on the plotted graph, can you select a proper K for this dataset for K-Means?

2. Use the proper K you selected, train your K-means model based on the selected K, and then draw a scatter plot of the dataset, using different colors for different clusters. Mark the center of each cluster. Intuitively, what kind of customers does each cluster represent?

(c)[Connection between Gaussian Mixture Model and K-Means Algorithm] The cluster that each sample is assigned to in the Gaussian Mixture Model can be the largest responsibility of x_i, which we write as

$$c_i = \arg\max_k \{r_i^{(k)}\}$$

where c_i is the assigned cluster for this sample. Let the covariate matrix be fixed as a relatively small value $\Sigma_k = 10^{-3} I_{2\times 2}$, where $I_{2\times 2}$ is an identity matrix, you don't need to update the covariates the maximum step, and retrain your Gaussian Mixture Model with the same best K as your K-Means Algorithm, and plot the scatter plot of the dataset, using different colors for different clusters. Is it performing similarly to the K-Means Algorithm with the same K? By looking back to the responsibility updated function Equation 5 and considering the cases when the covariates matrix is close to 0, can you give some explanations for this result?