ORF350 (Spring 2017) Analysis of Big Data **Assignment: 5**

Clustering Analysis and EM Algorithm

Lecturer: Han Liu *Email: hanliu@princeton.edu*

Due Date: This assignment is due on Thursday, April 27th by 5:00PM.

Submission: Please submit your hard copy at the ORF 350 dropbox in the Sherrerd Hall student lounge. Electronic submissions will not be accepted unless there is an extreme and compelling case. Late submission should refer to the the Late Day policy section in the Class Rules.

Q1. Clustering Algorithms (30 points)

Please download the MNIST handwritten digit dataset from `http://yann.lecun.com/exdb/mnist/`. It contains 28×28-pixel images for the hand-written digits {0,1,...,9} by storing each pixel value ranging between 0 and 255, and their corresponding true labels. Load the data into R and preprocess the data by compressing each image to $1/4$ of the original size in the following way: Divide each 28×28 image into 2×2 non-overlapping blocks. Calculate the mean pixel value of each 2×2 block, and create a new 14×14 image. This preprocessing step will drastically help your computation. We will be clustering the digits $\{0, 1, \ldots, 4\}$ in this homework.

You will not be graded on the performance of your code but rather on the correctness of your algorithm.
For Questions 1.1 and 1.2, you should attach the code and also generate the following plots/table for each algorithm. (When doing these for the EM algorithm, you may assign each image to its most likely cluster.)

- Plot 1: (Visualization) For each cluster, plot 12 random data points in that cluster (as illustrated in Figure 1(a) for clustering digits {0,1,2,3,4}).
- Plot 2: (Cluster Center) For each cluster j, visualize the cluster center $\boldsymbol{\mu}_j$. (An example is shown in Figure 1(b) for clustering digits {0,1,2,3,4}.)
- Plot 3: (Variance Visualization) For each cluster, visualize the variance among all the data points in that cluster as a heat map. That is, for each cluster, for each pixel position in the 14×14 image, calculate the variance of the values for that pixel position among all images in the cluster. After you have done this, scale all 196 values and load them into a new 14×14 image in such a way that black pixels in this image denote very low variance in that pixel position among all the digits in the cluster, white pixels denote very high variance. (You should rescale all the 196 pixel values in the heat map into the range [0, 255]. An example is shown in Figure 1(c) for clustering digits {0,1,2,3,4}.)
- Table 1: (Distribution of Labels and Error Rates) For each cluster, using the assigned true labels (provided in the dataset), denote which cluster represents which digit. Count how many images in that cluster had the actual label (as illustrated in Figure 1(d) for clustering digits {0,1,2,3,4}) and report the error rate (the percentage of misclassified digits) within each cluster.

Question 1.1: K-Means

Program (in R) the K-means algorithm and use it to cluster the digit dataset . We set $K = 5$. If a tie happens, we assign the data to the lowest-indexed cluster. Try 3 random initializations and pick the one that minimizes the objective

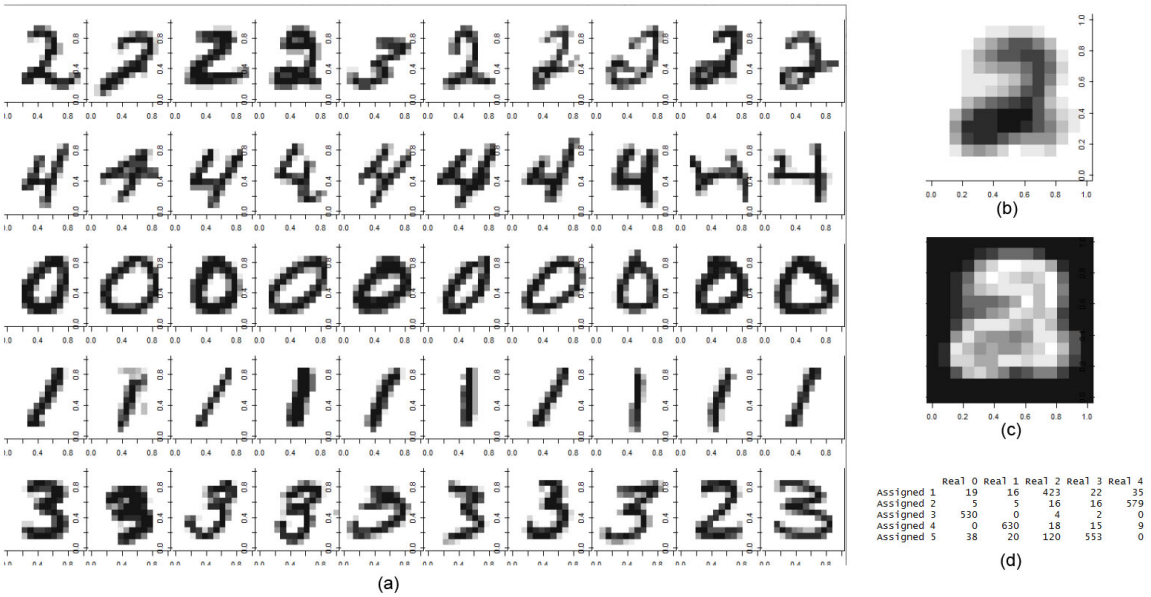

Figure 1: (a): Example of Plot 1. Notice that we do not have perfect clusters. (b): Example of Plot 2 for cluster "2". The mean digit looks like a 2, so this is good. (c): Example of Plot 3 for cluster 2. The area of high variance is in the shape of a 2, so this means our cluster did indeed cluster mostly 2's. If our plot was amorphous, this would be bad. (d): Example of Table 1 on only a subset of the data. Notice that each column has one unique row-element that dominates all the others. We can see here we did not prefectly cluster.

function

$$\sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}_i}\|_2^2 \tag{0.1}$$

where $\boldsymbol{\mu}_{\mathbf{x}_i}$ is the cluster center that the data point $\mathbf{x}_i$ is closest to.

Question 1.2: Expectation-Maximization Algorithm

Use the Gaussian mixture model as described in lecture to cluster the digit dataset.

Closely following the derivation in class, we now derive the EM algorithm for Gaussian mixture models. More specifically, we will use 2 models, the "mixture of spherical Gaussians" and the "mixture of diagonal Gaussians". By the end of this question, you should have derived two EM algorithms, one for each models.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be n i.i.d. data points, where $\mathbf{x}_i \in \mathbb{R}^d$. The likelihood of the data for k clusters is:

$$\begin{aligned} L(\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \eta_j\}_{j=1}^k) &= \prod_{i=1}^{n} p(\mathbf{x}_i; \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \eta_j\}_{j=1}^k) \\ &= \prod_{i=1}^{n} \sum_{j=1}^{k} p(\mathbf{x}_i | Z_i = j; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) p(Z_i = j; \eta_j) \\ &= \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{\eta_j}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_j|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)\right) \end{aligned} \tag{0.2}$$

The following denote the meaning of each symbol:

- each $\boldsymbol{\mu}_j$ is a d-dimensional vector representing the cluster center for cluster j
- each $\boldsymbol{\Sigma}_j$ is a $d \times d$ matrix representing the covariance matrix for cluster j
- $\sum_{j=1}^{k} \eta_j = 1$ and $\forall j, \eta_j \geq 0$ where all the η_j are the mixing coefficients.
- Z_i represents the hidden variable associated with x_i for $i \in \{1, \ldots, n\}$. It takes integer values $1, \ldots, k$.

In a **mixture of diagonal Gaussians** model, $\boldsymbol{\Sigma}_j = \mathrm{diag}(\sigma_{j_1}^2, \ldots, \sigma_{j_d}^2)$ is a diagonal matrix for all $j = 1, \ldots, k$.

In a **mixture of spherical Gaussians** model, $\boldsymbol{\Sigma}_j = \sigma_j^2 \mathbf{I}_d$ for all $j = 1, \ldots, k$. Here $\sigma_j > 0$ is a scalar and $\mathbf{I}_d$ is the $d \times d$ identity matrix.

(i) First, we do some preliminary work that will be useful later on.

- **Part a:** Write down the marginal distribution of the Z_i (Hint: What is the probability $p(Z_i = j)$).
- **Part b:** Calculate $p(Z_i = j \mid \mathbf{x}_i)$. (Hint: Bayes Rule)

(ii). We now derive the E- and M- steps. We denote $\boldsymbol{\theta} := \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \eta_j\}_{j=1}^{k}$. From Equation (0.2), we can write down the log-likelihood and derive a lower bound:

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) && (0.3) \\ &= \sum_{i=1}^{n} \log \left[\sum_{j=1}^{k} p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j) \right] && (0.4) \\ &= \sum_{i=1}^{n} \log \left[\sum_{j=1}^{k} \gamma_{ij} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{\gamma_{ij}} \right], && (0.5) \end{aligned}$$

where $\gamma_{ij} > 0$ for all i, j and satisfies

$$\sum_{j=1}^{k} \gamma_{ij} = 1 \text{ for } i = 1, \ldots, n. \tag{0.6}$$

- **Part c:** Prove the following lower bound of the log-likelihood function:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left[\sum_{j=1}^{k} \gamma_{ij} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{\gamma_{ij}} \right] \geq \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \log \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{\gamma_{ij}} \right]. \tag{0.7}$$

  Hint: you can use the Jensen's inequality $\log \mathbb{E}X \geq \mathbb{E} \log X$ (You are not required to prove the Jensen's inequality).

- **Part d:** (E-Step) We define

$$F(\boldsymbol{\gamma}, \boldsymbol{\theta}) := \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \log \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{\gamma_{ij}} \right] \tag{0.8}$$

to be the lower bound function of $\ell(\boldsymbol{\theta})$. Let $\boldsymbol{\theta}^{\text{old}}$ be a fixed value of $\boldsymbol{\theta}$ (e.g., $\boldsymbol{\theta}^{\text{old}}$ could be the parameter value of the current iteration). Recall from Part c that we designed $F(\boldsymbol{\gamma}, \boldsymbol{\theta}^{\text{old}})$ to be a lower bound for $\ell(\boldsymbol{\theta}^{\text{old}})$. We want to make this lower bound as tight as possible.

Prove that $\ell(\boldsymbol{\theta}^{\text{old}}) = F(\boldsymbol{\gamma}, \boldsymbol{\theta}^{\text{old}})$ when

$$\gamma_{ij} = p_{\boldsymbol{\theta}^{\text{old}}}(Z_i = j \mid \mathbf{x}_i). \tag{0.9}$$

This result implies that, at the current parameter value $\boldsymbol{\theta}^{\text{old}}$, the function $F(\gamma, \boldsymbol{\theta}^{\text{old}})$ is maximized when γ_{ij} takes the form as in Equation (0.9). This justifies the E-step of the EM algorithm.

(iii). Once the E-step is derived, we now derive the M-step. First, we plug $\gamma_{ij} = p_{\boldsymbol{\theta}^{\text{old}}}(Z_i = j \mid \mathbf{x}_i)$ into Equation (0.8) and define the lower bound function at $\boldsymbol{\theta}^{\text{old}}$ to be

$$F_{\boldsymbol{\theta}^{\text{old}}}(\boldsymbol{\theta}) := \sum_{i=1}^{n} \sum_{j=1}^{k} p_{\boldsymbol{\theta}^{\text{old}}}(Z_i = j \mid \mathbf{x}_i) \log \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{p_{\boldsymbol{\theta}^{\text{old}}}(Z_i = j \mid \mathbf{x}_i)} \right]. \tag{0.10}$$

- **Part e** (M-step for mixture of spherical Gaussians) In the M-step, we aim to find $\boldsymbol{\theta}$ that maximize the lower bound function $F_{\boldsymbol{\theta}^{\text{old}}}(\boldsymbol{\theta})$. Under the **mixture of spherical Gaussians** model, derive the M-step updating equations for $\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ and η_j.
- **Part f** (M-step for mixture of diagonal Gaussians) Under the **mixture of diagonal Gaussians** model, derive the M-step updating equations for $\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ and η_j.

(iv) Program (in R) the EM algorithm you derived for mixture of spherical Gaussians. Assume 5 clusters. Terminate the algorithm when the fractional change of the log-likelihood goes under 0.0001. (Try 3 random initializations and present the best one in terms of maximizing the likelihood function).

(v) Program (in R) the EM algorithm you derived for mixture of diagonal Gaussians. Assume 5 clusters. Terminate the algorithm when the fractional change in the log-likelihood goes under 0.0001. (Try 3 random initializations and present the best one in terms of maximizing the likelihood function).

Hint: For these implementations, you will run into three different problems. Apply these following hints to solve each problem.

- 1) Use the log-sum-exp trick to avoid underflow on the computer. You will run into this problem when computing the log-likelihood. That is, when you calculate $\log \sum_j \exp^{a_j}$ for some sequence of variables a_j, calculate instead $A + \log \sum_j \exp^{a_j - A}$ where $A = \max_j a_j$.
- 2) Some pixels in the images do not change throughout the entire dataset. (For example, the top-left pixel of each image is always 0, pure white.) To solve this, after updating the covariance matrix $\boldsymbol{\Sigma}_j$ for the mixture of diagonal Gaussians, add $0.05\mathbf{I}_d$ to $\boldsymbol{\Sigma}_j$ (ie: add 0.05 to all the diagonal elements).
- 3) Be mindful of how you initialize $\boldsymbol{\Sigma}_j$. Note that for a diagonal matrix $\boldsymbol{\Sigma}_j$, the determinant $|\boldsymbol{\Sigma}_j|$ is the product of all the diagonal elements. Setting each diagonal element to a number too big at initialization will result in overflow on the computer.

Question 1.3: Relationship between K-Means and EM

Explain why in the lecture we say the K-Means algorithm can be derived from as a limiting case of the EM algorithm for mixture of spherical Gaussians? That is, start from the EM algorithm and find the appropriate settings for the E- and M- steps to reconstruct the K-Means algorithm. Justify. (Hint: Calculate the limit $\lim_{\sigma_j^2 \to 0} \gamma_{ij}$)

Question 1.4: Comparison of K-Means and EM Instructions: Answer each of the following questions with a few sentences.

- Compare the outputs of all your three algorithms. In your opinion, which one worked the best? Were any of the outputs (in your opinion) satisfactory? Which ones, and why or why not? What aspect of the K-Means or EM algorithm made these good traits discoverable, and what traits might have the algorithms overlooked?
- Note the time it took to run each algorithm. Were the quality of the outputs worth the amount of time needed to run the algorithm?
- In your opinion, were mixture models for Gaussian distributions good for modelling the data? Could you think of some other ways to model the data?
- What was your initialization strategy for K-Means and EM? Informally describe if you think your strategy was "successful."

Question 2. EM Algorithms for Arbitrary Distributions (20 points)

In this problem, you will derive an EM algorithm for estimating the mixing parameter of a mixture of arbitrary probability densities $p_1(x)$ and $p_2(x)$. You can think about such mixtures in the following way: First, you flip a coin. With probability η (i.e., the coin comes up heads), you will sample X from density $p_1(x)$, with probability $(1-\eta)$ you sample from density $p_2(x)$.

More formally, let

$$p_\eta(x) = \eta p_1(x) + (1-\eta)p_2(x)$$

where $p_1(x)$ and $p_2(x)$ are two arbitrary probability density functions on $\mathbb{R}$, and $\eta \in (0,1)$ is an unknown mixture parameter.

Question 2.1 Given a data point X, and a value for the mixture parameter η, compute the probability that X was generated from density $p_1(x)$. In another word, let $Z \in \{1,0\}$ be the variable representing the outcome of the coin toss ($Z=1$ when the coin comes up with head), calculate $\mathbb{P}(Z=1 \mid X)$.

Question 2.2 Now you are given a dataset $\{X_1,\ldots,X_n\}$ drawn independently and identically distributed from the mixture density, and set of coin flips $\{Z_1,Z_2,\ldots,Z_n\}$, such that $Z_i=1$ means that X_i is a sample from $p_1(x)$, and $Z_i=0$ means that X_i was generated from density $p_2(x)$. For a fixed parameter η, compute the complete log-likelihood of the data, i.e., $\log p(X_1,Z_1,X_2,Z_2,\ldots,X_n,Z_n)$. (Hint: you may want to use the fact that $p(x \mid z) = p_1(x)^z p_2(x)^{1-z}$ and $p(z)=\eta^z(1-\eta)^{1-z}$).

Question 2.3 Now you are given a dataset $\{X_1,\ldots,X_n\}$ drawn i.i.d. from the mixture density. Without the knowledge about which component the samples were drawn from (i.e., the Z_i are unknown). Using your derivations from part 1 and 2, derive the E-Step and M-step for the EM-algorithm to compute the maximum likelihood estimate of the mixture parameter η. More specifically, assuming we are now at the t-th iteration with the current $\eta=\eta^{(t)}$. Please describe your derivation of the E- and M-step clearly in your answer.

[Hint: In E-step, we calculate the conditional expectation of the indicator function of the latent class assignment for each data point, while in M-step, we solve an "expected" maximum likelihood problem with complete data.]

Question 2.4 Use one or two sentences to explain why the EM algorithm is guaranteed to converge?

Question 3. K-Means Wikipedia Documents Clustering (30 points)

In this problem set, we will be using K-means clustering on Wikipedia introductions of famous individuals who are associated with Princeton University. Through our analysis, we will determine broad categories that capture the different academic and cultural backgrounds of these individuals. This analysis is part of the field called "topic modeling".

Question 3.1 Load in the dataset `Wikipedia.RData`. Print out the head of the dataset. Who are the first 3 individuals in the dataset and their corresponding professions? (It's okay to give a broad category). Then use the script `script1_HW5.R` to convert the matrix into a document-term matrix. How many individuals are there? How many words are there? What are the top 10 most frequent words in the entire dataset? List the 0%, 25%, 50%, 75% and 100% quantiles of the word counts across the entire dataset. The phenomenon you will see can be attributed to "Zipf's Law", saying the distribution of word counts in any text data follows an exponential-like curve. In the later questions, we will like to handle these "over-represented" words.

Question 3.2 We notice that the top ten words are not very interesting with "the" and "and" being among the most frequent words. One way to combat this is to re-parameterize the document-term matrix using the TF-IDF (Term Frequency, Inverse Document Frequency) parameterization. The essential idea of TF-IDF is that if a word appears frequently in a document, it's important. However, if a word appears in many documents, it's not a unique identifier. In this way, common words like "the" and "for", which appear in many documents, will be scaled down. Run the script `script2_HW5.R` to implement TF-IDF parameterization. Based on this new parameterization, what are the 10 words with highest total weight across the entire dataset?

Question 3.3 Great! Hopefully we are starting to see meaningful topics appear, but the TF-IDF parameterization might not be highly interpretable as our `dtm.mat` matrix contain weights instead of word counts. We would still like to know which words are common in the dataset (for our downstream analysis) but we'll like to remove "common" words. Run the `script3_HW5.R`, which does the following two things: First, it removes words that appear too frequently across all the documents. For example, the word "Princeton" is in every document. Second, it removes the top 300 most frequent words according to Google.

After running the following script, how many words are left in our analysis? What are the top ten words in our remaining matrix `dtm.mat.raw`?

Question 3.4 We see that the most common words in `dtm.mat.raw` are now similar to the top-weighted words in `dtm.mat`. This is great for our future interpretability. Now, let's zoom in one particular person and make sure our data was processed correctly. Search for "Ben Bernanke" in our dataset. Print out the raw text for him according to `dat` (our original dataset). Find in this text the 10 top-weighted words according to `dtm.mat` and the 10 words with the highest word counts according to `dtm.mat.raw`.

Question 3.5 We will now implement K-Means analysis. One of the most important ingredients for K-Means is a good distance function. Typically, the Euclidean distance is used between two vectors, but in this question, we will assume that the cosine-distance is more appropriate (`https://en.wikipedia.org/wiki/Cosine_similarity`). We will work with the `dtm.mat` (the TF-IDF parameterized) matrix when we use K-Means (as it includes all the words), but we will use `dtm.mat.raw` to interpret our results (as it includes only the important words we want to see).

Renormalize all the rows of `dtm.mat` to have norm 1. Then using `norm.sim.ksc` in the `akmeans` package, set the seed to 10 (for reproducibility) and find 8 clusters according to `dtm.mat`. How big are each of resulting clusters? Based on the cluster assignments, print out the top 25 words in each cluster according to `dtm.mat.raw`. Also print out the quantiles (0%, 25%, 50%, 75%, 100%) of the word counts with non-zero word counts within each cluster to

get a sense of how significant the top words are.

Q4. SVM Theory (20 points)

Suppose we have i.i.d data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i$ is a d-by-1 numeric vector and $y_i = 1$ or -1. Support Vector Machine (SVM) solves the following optimization problem.

$$\min_{\boldsymbol{\beta},b} \frac{1}{n}\sum_{i=1}^{n}[1 - y_i(\boldsymbol{\beta}^T\mathbf{X}_i - b)]_+ + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{0.11}$$

where $[x]_+ := \max\{x, 0\}$ and $\lambda > 0$. Here, we explore the geometric meaning behind this optimization problem.

Question 4.1: Geometric Interpretation The geometric motivation of SVM is to select two parallel hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. Any two parallel hyperplanes in $\mathbb{R}^d$ can be expressed as two equations below.

$$\boldsymbol{\beta}^T\mathbf{x} - b = 1$$

and

$$\boldsymbol{\beta}^T\mathbf{x} - b = -1,$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$ and b are constants. Show the distance between these two hyperplanes is $2/\|\boldsymbol{\beta}\|_2$.

(Hint 1: Two vectors $\mathbf{y}, \mathbf{z} \in \mathbb{R}^d$ are perpendicular if $\mathbf{y}^T\mathbf{z} = 0$.)
(Hint 2: Note that for a hyperplane $\{\mathbf{x} \in \mathbb{R}^d : \boldsymbol{\beta}^T\mathbf{x} = C\}$, the vector $\boldsymbol{\beta}$ is perpendicular to the hyperplane. We often call $\boldsymbol{\beta}$ the normal vector of the hyperplane.)
(Hint 3: Consider two parallel hyperplanes. Given a point $\mathbf{x}_0$ on one hyperplane, find another point $\mathbf{x}_1$ on the other hyperplane such that $(\mathbf{x}_1 - \mathbf{x}_0)$ is parallel to $\boldsymbol{\beta}$. Then $\|\mathbf{x}_1 - \mathbf{x}_0\|_2$ is the distance between the two hyperplanes.)

Question 4.2: Simple Reformulation of SVM

(a) We can therefore implement the idea of SVM by solving the following optimization problem.

$$\begin{aligned} \min_{\boldsymbol{\beta},b} \quad & \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & y_i(\boldsymbol{\beta}^T\mathbf{x}_i - b) \geq 1, 1 \leq i \leq n \end{aligned} \tag{0.12}$$

In a few sentences, answer the following questions:

- Why do we wish to minimize the objective function $\frac{1}{2}\|\boldsymbol{\beta}\|_2^2$?
- For each datapoint $(\mathbf{x}_i, y_i)$, why do we use the constraint $y_i(\boldsymbol{\beta}^T\mathbf{x}_i - b) \geq 1$? Analyze the case where $y_i = 1$ and $y_i = -1$ separately.

(b) However, sometimes data points cannot be linearly separated, which means there are no such two hyperplanes that can separate data points without error. Construct such a case where $d = 2$ with at least three data points where (0.12) has no feasible solutions, i.e. there is no $\boldsymbol{\beta}$ and b to satisfy the constraint of (0.12).

Question 4.3: General Reformulation of SVM For data that cannot be linearly separated, we introduce non-negative slack variables ζ_i for each data point, which serves as a measure of misclassification degree, and we instead consider

the following optimization problem.

$$\begin{aligned} \min_{\boldsymbol{\beta}, b, \{\zeta_i\}_{i=1}^n} \quad & C\sum_{i=1}^{n}\zeta_i + \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & \zeta_i \geq 1 - y_i(\boldsymbol{\beta}^T\mathbf{x}_i - b) \\ & \zeta_i \geq 0 \\ & \text{for all } 1 \leq i \leq n \end{aligned} \tag{0.13}$$

This method is called the soft margin method, which essentially means to optimize a trade-off between a large margin and a small misclassification error.

Assume that, for some given λ parameter, $\widehat{\boldsymbol{\beta}}$ and $\widehat{b}$ optimizes (0.11). Show that there exists some constant C and $\left\{\widehat{\zeta}_i\right\}_{i=1}^n$ such that $\widehat{\boldsymbol{\beta}}$, $\widehat{b}$, and $\left\{\widehat{\zeta}_i\right\}_{i=1}^n$ is a feasible set of solutions that optimizes (0.13).