# CS5691 Pattern Recognition and Machine Learning

Assignment 2


**Ruthwik Chivukula**

ME21B166

# Contents

# 1 Probabilistic Mixtures

In this question, we are given a dataset of 400 data points, each with a dimension of 50x1. Each data point is a string of 0s and 1s. We need to identify which probabilistic mixture is more likely to have generated the dataset. It has been given that the clusters are fixed and equal to 4.

## 1.1 Bernoulli Mixture Model

As each data point is a 50 x 1 vector with 0s and 1s, the potential distribution that comes to mind is the **Bernoulli distribution**. We fit individual Bernoulli distributions on each of the 50 elements across all four clusters.

So, the total number of independent parameters:

$50 \cdot 4 + 3 = 203$

$$p_k^m \quad \forall k \in \{1, ...4\}, \forall m \in \{1, 2, ..., 50\}$$

$$\pi_k \quad \forall k \in \{1, ..., 4\}$$

such that

$$\sum_{k=1}^{4} \pi_k = 1$$

where $p_k^m$ denotes the probability of occurrence of 1 in the $m^{th}$ component of the data point from $k^{th}$ cluster and $\pi_k$ denotes the probability of picking $k^{th}$ cluster.

We know that the likelihood function is as follows:

$$L(\theta_1, \theta_2, \ldots, \theta_k; x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} f_{mixture}(x_i; \theta_1, \theta_2, \ldots, \theta_k)$$

where $\theta_1, \theta_2, \ldots, \theta_k$ are the underlying parameter and $x_1, \ldots, x_n$ are data points. $\theta_k$ consists of $\pi_k$ and $p_k$ Because of the mutual exclusiveness of the events, we can rewrite the likelihood function as:

$$L(\theta_1, \theta_2, \ldots, \theta_k; x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f(x_i; p_k)$$

As the data points are vectors, we will have individual Bernoulli distributions for each of the 50 components to generate 0s and 1s.

**Note:** $p_k$ isn't a single scalar value. Rather, it is an array of $p_k^1, p_k^2, \ldots, p_k^m$ where $m$ denotes the component index.

Given that we are independently fitting Bernoulli on each component, the $f(x_i; p_k)$ can be expressed as:

$$f(x_i; p_k) = \prod_{m=1}^{50} (p_k^m)^{x_i^m} (1 - p_k^m)^{1 - x_i^m}$$

$x_i^m = 1$ if the $m^{th}$ component of the $i^{th}$ data point has entry 1, else $x_i^m = 0$

Plugging the above value into the likelihood function, setting n = 400 and K = 4, and applying logarithm (monotonic function) on both sides boils down the RHS as follows:

$$\log(L(\theta, \pi)) = \sum_{i=1}^{400} \log \left( \sum_{k=1}^{4} \pi_k \prod_{m=1}^{50} (p_k^m)^{x_i} (1 - p_k^m)^{1 - x_i} \right)$$

Solving the above log-likelihood function analytically is not possible. Hence we find a workaround by invoking the Jensen's inequality. Using it, we can convert this into a modified log-likelihood function as shown below:

$$Modified\log(L(\theta,\lambda,p,\pi)) = \sum_{i=1}^{400}\sum_{k=1}^{4}\lambda_k^i \log\left(\frac{\pi_k \prod_{m=1}^{50}(p_k^m)^{x_i}(1-p_k^m)^{1-x_i}}{\lambda_k^i}\right)$$

such that

$$\sum_{k=1}^{4}\lambda_k^i = 1$$

and

$$\sum_{k=1}^{4}\pi_k = 1$$

Now we apply the EM algorithm to this. In the Expectation step, the function is maximized wrt $\lambda$, keeping $\pi$ and $p$ fixed. This is done by setting the derivative wrt to the $\lambda$ to zero. We get $\lambda$ as:

$$\lambda_k^i = \left(\prod_{m=1}^{50}(p_k^m)^{x_i}(1-p_k^m)^{1-x_i}\right)\pi_k \Bigg/ \sum_{k=1}^{4}\left(\prod_{m=1}^{50}(p_k^m)^{x_i}(1-p_k^m)^{1-x_i}\right)$$

In the Maximization step, we have to maximize the modified log-likelihood function wrt to $p_m^k$ and $\pi_k$ values while keeping $\lambda$ fixed. As it is a closed-form equation, we can take the derivative and set it to zero. By doing that, we obtain $p_m^k$ and $\pi_k$ values as:

$$\hat{p}_k^m = \frac{\sum_{i=1}^{n}\lambda_k^i x_i^m}{\sum_{i=1}^{n}\lambda_k^i}$$

$$\hat{\pi}_k = \frac{\sum_{i=1}^{n}\lambda_k^i x_i^m}{n}$$

The expectation and maximization steps are performed alternatively until convergence.

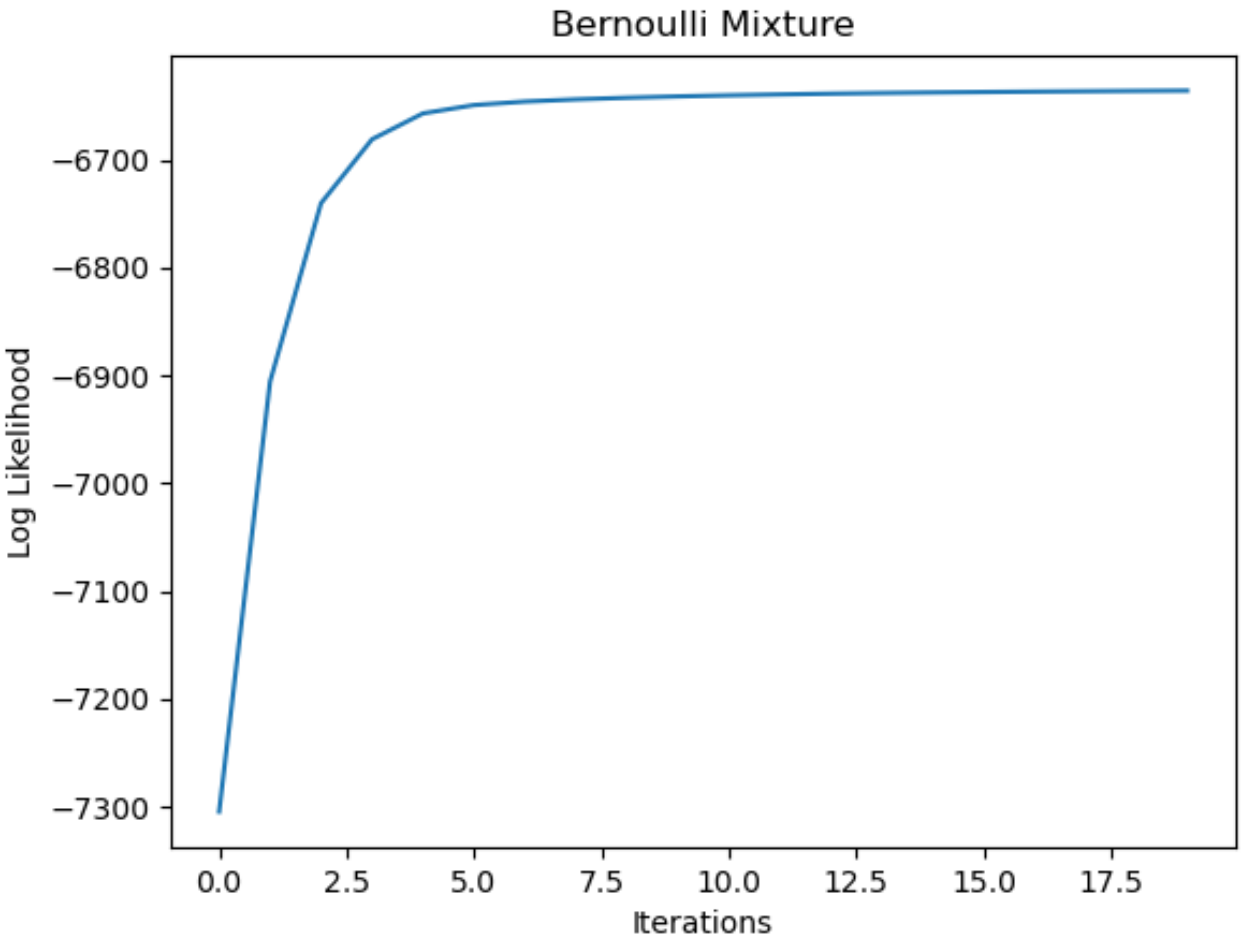


Figure 1: Log Likelihood vs Iterations for Bernoulli EM

## 1.2 Gaussian Mixture Model

We will fit a multivariate Gaussian to the dataset and observe the Log Likelihood value over iterations. We are using 4 different multivariate Gaussian to fit the data points, one for each cluster. So the parameters are mean (50 x 1), covariance matrix (50 x 50) and *pi*. We could have even tried fitting Gaussian for individual components of the data points instead of a single multivariate Gaussian for all components but that would assume independence

amongst component values, which may or may not be the case. Hence, to generalize, I have adopted the former approach.

**Note:** To prevent the covariance matrix from becoming invertible, an identity matrix times a small factor has been added.

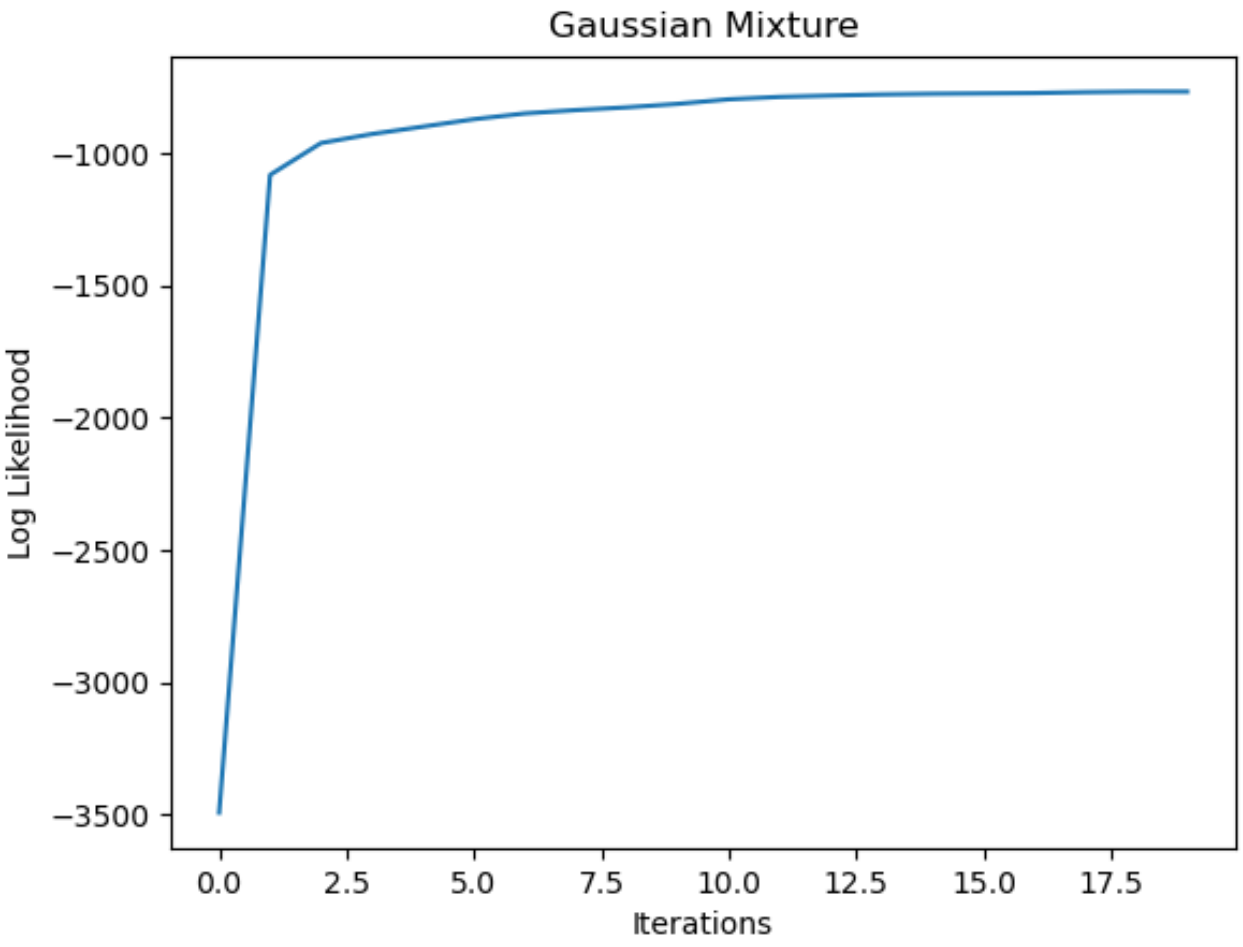


Figure 2: Log Likelihood vs Iterations for Gaussian EM

By observing the plots, Gaussian Mixture Models (GMM) have a higher log-likelihood value in comparison to Bernoulli Mixture Models and they are better. However, the time taken for computation is more, given the inverse calculations it involves.

## 1.3 K-means

The earlier methods that we have implemented have a probabilistic approach. Let us see how a deterministic approach like KMeans performs on the dataset.

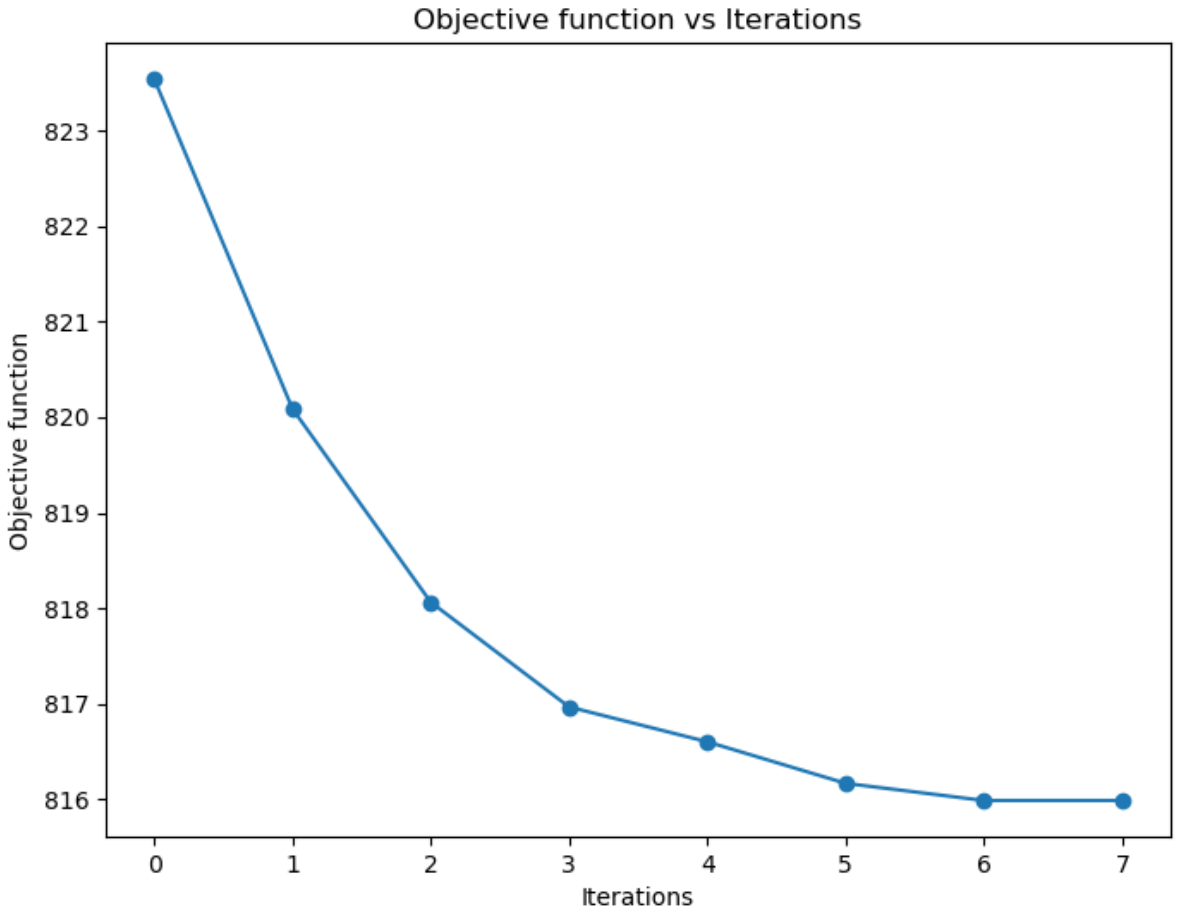


Figure 3: KMeans

| **Model** | **Score** |
|---|---|
| Gaussian Mixture Model | 823.06 |
| Bernoulli Mixture Model | 829.415 |
| K-means | 823.29 |

If we use the objective function (the one which we used for K-means) to compare all three models, we can observe that GMMs outperform the rest, followed by K-means and Bernoulli at the end. We can try visualizing the same by taking the top 2 components out of 50 using PCA and plotting it to study the clusters and patterns.

# 2 Regression

In this section, we will be dealing with regression problems. We will first compute the $w_{ML}$ analytically for the given dataset and then attempt to obtain the optimal $w$ using iterative approaches.

## 2.1 Analytical Solution

We set up a least squares function as the objective function and find the $w_{ML}$ which minimizes it. The result is given as follows:

$$\mathbf{w}_{ML} = \left(\mathbf{X} \otimes \mathbf{X}^{\top}\right)^{-1} \mathbf{X} \otimes \mathbf{Y}$$

The only drawback of this method is it involves inverse calculation, which is a computational bottleneck, something we desire to avoid. Hence, we look into iterative approaches which save computing time and effort.

## 2.2 Gradient Descent Algorithm

We will be implementing the gradient descent algorithm and see how close the $w^t$ value approaches to $w_{ML}$. The algorithm update step is mentioned below:

$$w^{t+1} = w^t - \eta^t \nabla f(w^t)$$

After experimenting with different values finally used learning rate = 1e-6 and the number of iterations = 5000.

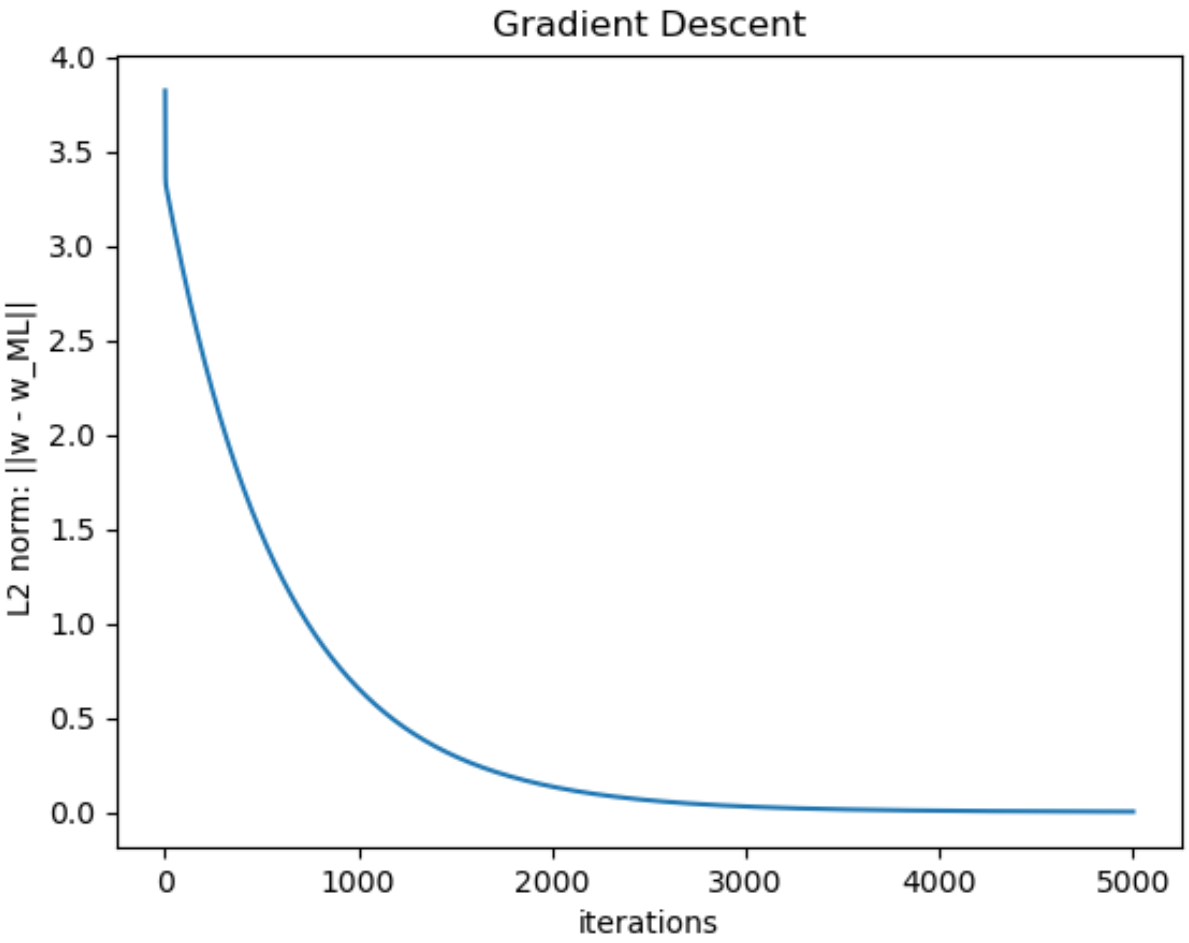


Figure 4: Gradient Descent Algorithm

It can be observed that the norm has a sudden drop in the first iteration, after which the norm decreases gradually. After 5000 iterations, the norm is almost zero which implies we have converged to our analytical solution.

## 2.3 Stochastic Gradient Descent

Here instead of using the entire dataset to compute $\nabla f(w^t)$ we sample batches from the dataset and use only the data point from the sampled batch for one update. We repeat by sampling several times until convergence.

$$\nabla f(w^t) = 2(\widetilde{\mathbf{X}} \otimes \widetilde{\mathbf{X}}^\top)\mathbf{w}^t - 2\widetilde{\mathbf{X}} \otimes \widetilde{\mathbf{Y}}$$

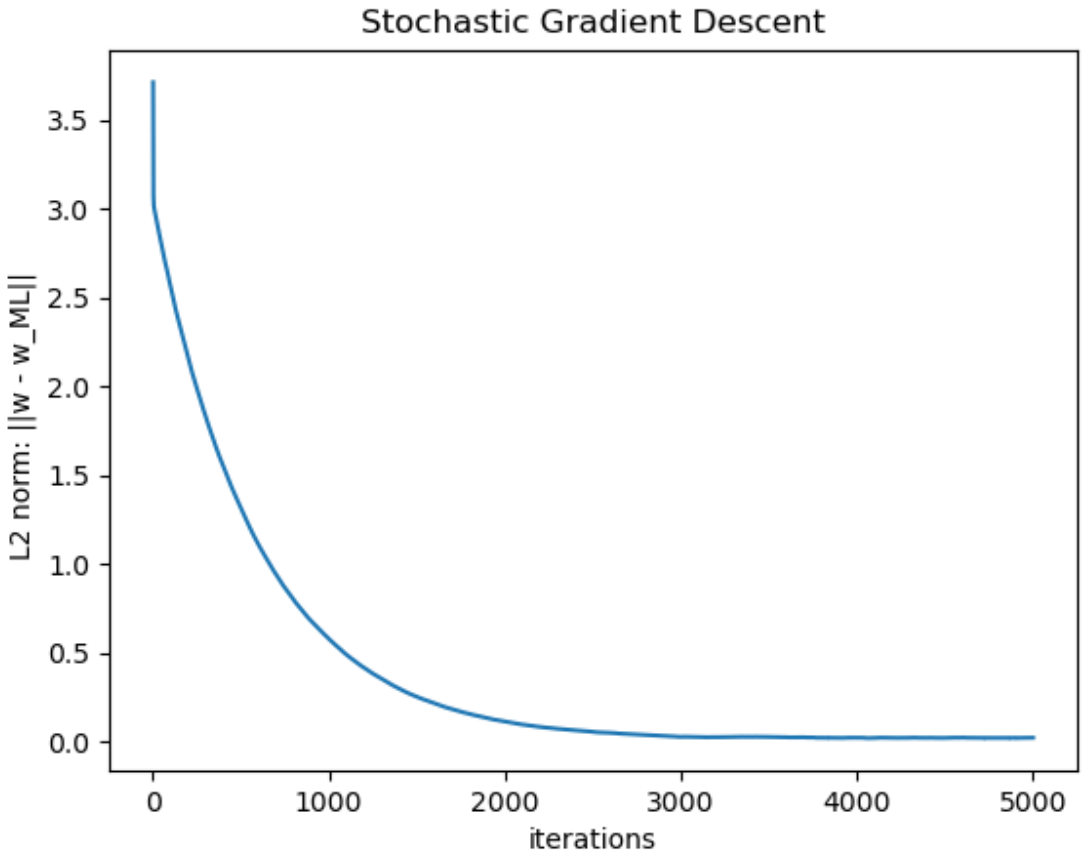


Figure 5: Stochastic Gradient Descent

The time taken for running the same number of iterations was less in SGD. After 5000 iterations, even SGD converges to zero norm, so computationally, SGD is preferable over GD. Learning rate of SGD was found to be greater than learning rate of GD.

## 2.4 Ridge Regression

We add an additional regularization term $\lambda||\mathbf{w}||_2^2$ to the earlier loss function, which prevents the model from overfitting and enhances its performance on unseen data. To begin with, for choosing the right $\lambda$, we do cross-validation. I split the dataset into a train set and a validation set in the ratio of 70:30. Tried out different $\lambda$ values and chose the $\lambda$, which has the least validation loss.

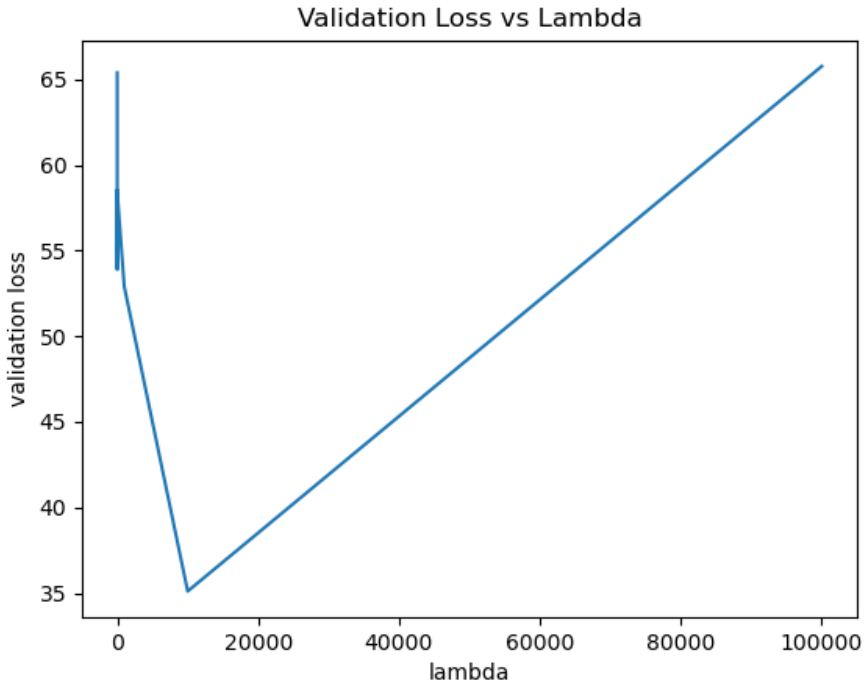


Figure 6: Validation loss vs Lambdas

From the graph, we can learn that $\lambda = 10000$ produces the least validation loss, and hence we will set our $\lambda$ to be that.

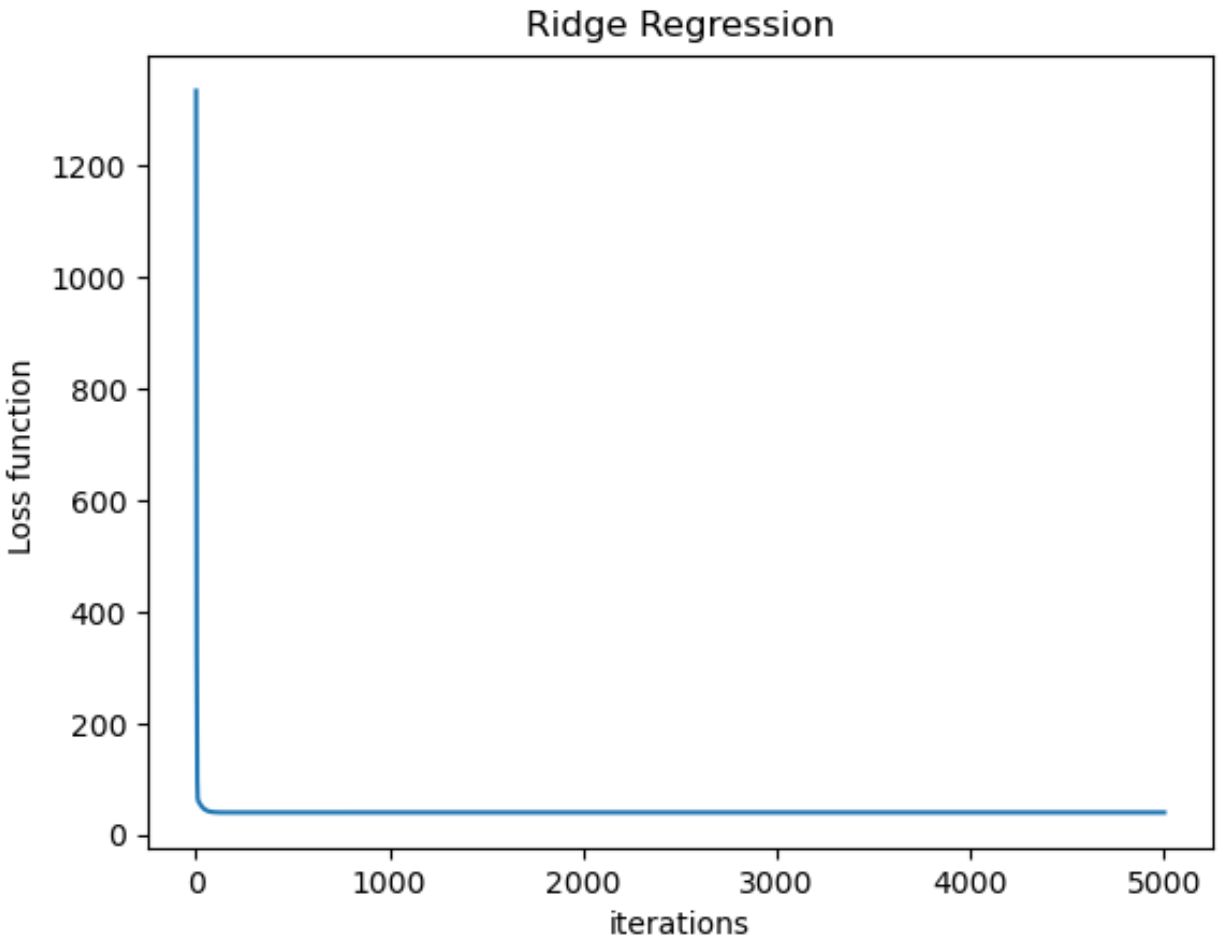


Figure 7: Loss function vs iterations for Ridge Regression $\lambda = 10000$

The test error for $\lambda_R$ is lesser in comparison to $\lambda_{ML}$. For analytical, the test error comes out to be 13.63, whereas for ridge, it comes out to be 10.63. The reason why ridge performs better on the test dataset is because of the additional regularization term, which is added to the loss function, which prevents it from overfitting.