Anirudh Rao
BE21B004

DA5400 – Foundations of Machine Learning

Assignment 1

Contents

Part 1

The given dataset has features $X \in \mathbb{R}^{2\times1000}$ and labels $y \in \mathbb{R}^{1000}$.

The least squares linear regression problem in this case is to solve:

$$\underset{w \in \mathbb{R}^2}{\operatorname{argmin}} \|X^T w - y\|^2$$

The analytical solution to this problem is given by:

$$w_{\text{ML}} = (XX^T)^{-1}Xy$$

Using this for the given dataset, we obtain:

$$w_{\text{ML}} = \begin{bmatrix} 1.446 \\ 3.884 \end{bmatrix}$$

Part 2

Since calculating $(XX^T)^{-1}$ may be computationally expensive, we can solve the least squares problem using gradient descent.

We define:

$$f(w) = \|X^T w - y\|^2$$

Thus,

$$\nabla f(w) = 2XX^T w - 2Xy$$

The pseudocode for gradient descent is shown below:

initialize w^0

for $t = 0,1,2,\ldots,T$

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

end

For our dataset, we will initialize $w^0 = \begin{bmatrix}0\\0\end{bmatrix}$ and take the step size $\eta^t = 1 \times 10^{-4}$ to be a constant across all iterations. We set the maximum number of iterations T to be 1×10^5.

However, if for some iteration $t < T$ we find that $\nabla f(w^t) < \varepsilon$ for some pre-defined tolerance $\varepsilon \approx 0$, we stop our iterations and report w^t as our solution. In our algorithm, we have set the tolerance ε to be 1×10^{-5}. This helps us avoid unnecessary iterations.

Thus, our pseudocode is modified to:

initialize $w^0 = \begin{bmatrix}0\\0\end{bmatrix}$

for $t = 0,1,2,\ldots,10^5$

$\quad \nabla f(w^t) = 2XX^T w^t - 2Xy$

$\quad$ if $\nabla f(w^t) < 10^{-5}$

$\quad\quad$ end

$\quad$ else

$\quad\quad w^{t+1} = w^t - 10^{-4} \nabla f(w^t)$

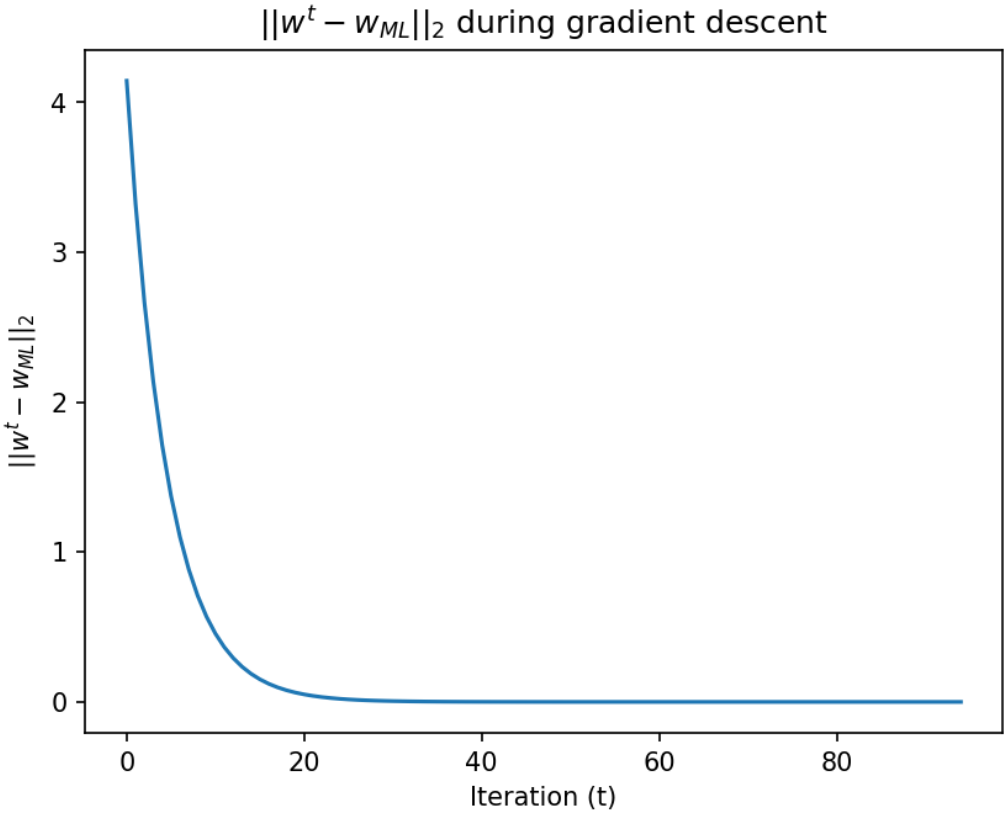
end

By running this on our dataset, we find that the algorithm converges after just 95 iterations. The w returned by the algorithm is given by:

$$w_{\text{GD}} = \begin{bmatrix}1.446\\3.884\end{bmatrix} = w_{\text{ML}}$$

Thus, the algorithm is able to correctly obtain the analytical solution w_{ML} using an iterative procedure.

To study the convergence of the gradient descent algorithm we plot $\|w^t - w_{\text{ML}}\|_2$ as a function of the iteration number t.



We can see that the value of $\|w^t - w_{\text{ML}}\|_2$ starts from $\|w_{\text{ML}}\|_2$ since $w^0 = [0 \quad 0]^T$ and smoothly and gradually converges to 0 when $w^t \to w_{\text{ML}}$.

The decrease in $\|w^t - w_{\text{ML}}\|_2$ is quite rapid in the initial iterations and slows down as the number of iterations increases.

Part 3

Computing XX^T for the gradient $\nabla f(w) = 2XX^T w - 2Xy$ may be computationally expensive for large n. Thus, we can use stochastic gradient descent. In this algorithm, we sample a subset of the data in order to compute the gradient at each iteration. It should also be noted that unlike in standard gradient descent, w^t in stochastic gradient descent is taken to be the average of $\{w^0, w^1, w^2, \ldots, w^t\}$.

We will take the size of our sampled subset to be 100, as mentioned in the question.

With suitable modification to the gradient descent algorithm, the pseudocode for stochastic gradient descent is shown below:

initialize $w^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

for $t = 0,1,2,\dots,10^4$

sample $\tilde{X} \in \mathbb{R}^{2\times 100}, \tilde{y} \in \mathbb{R}^{100}$ from X, y

$\nabla f(w^t) = 2\tilde{X}\tilde{X}^T w^t - 2\tilde{X}\tilde{y}$

if $\nabla f(w^t) < 10^{-5}$

end

else

$w^{t+1} = w^t - 10^{-2}\nabla f(w^t)$

$w^{t+1} = \frac{1}{t+1}\sum_{i=0}^{t+1} w^i$

end

It is to be noted that the maximum number of iterations and the step size have been changed to 1×10^4 and 1×10^{-2} respectively. These were found to have better convergence. The tolerance ε is still the same.

By running this on our dataset, the w returned by the algorithm is given by:

$$w_{\text{SGD}} = \begin{bmatrix} 1.427 \\ 3.877 \end{bmatrix}$$

The algorithm returns a solution quite close to $w_{\text{ML}} = w_{\text{GD}} = \begin{bmatrix} 1.446 \\ 3.884 \end{bmatrix}$.

This was achieved at the end of all 1×10^4 iterations, which is higher than the number of iterations required by standard gradient descent.

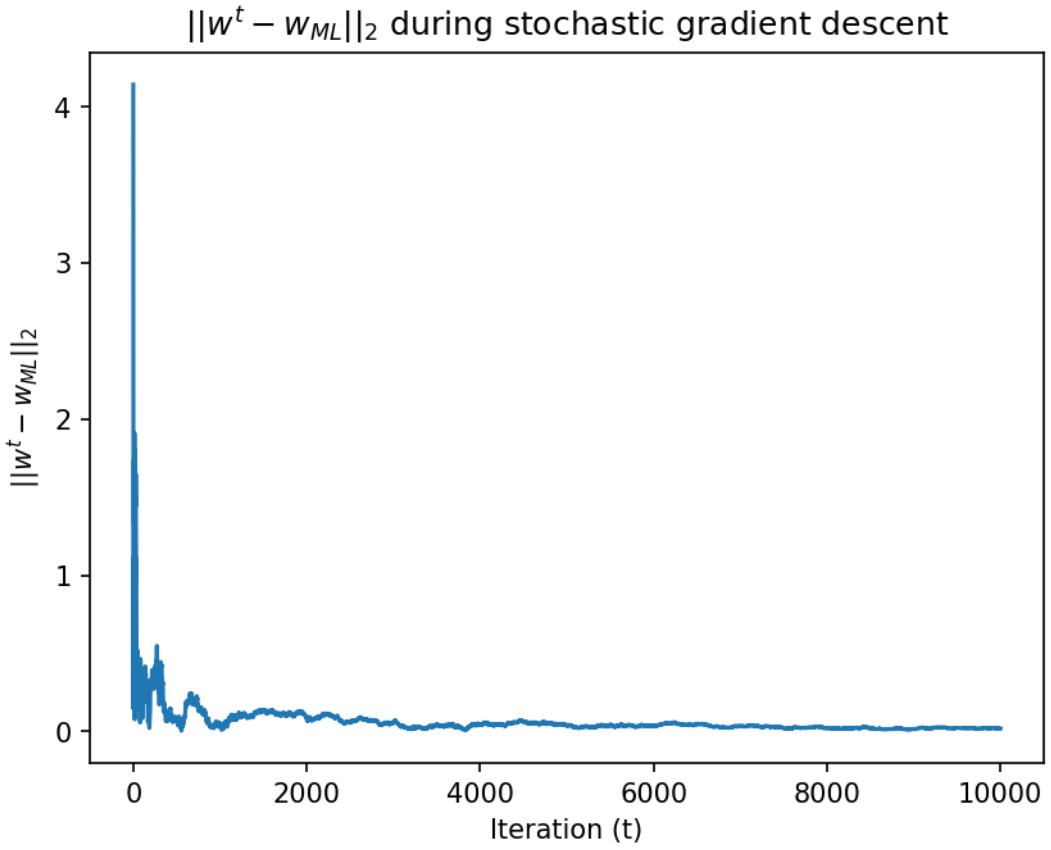
We also find that:

$$\|w_{\text{ML}}\|_2 = \|w_{\text{GD}}\|_2 = 4.145$$

$$\|w_{\text{SGD}}\|_2 = 4.131 \approx \|w_{\text{ML}}\|_2$$

$$\|w_{\text{ML}} - w_{\text{SGD}}\|_2 = 0.02 \approx 0$$

To study the convergence of the stochastic gradient descent algorithm we plot $\|w^t - w_{\text{ML}}\|_2$ as a function of the iteration number t.



We can see that the value of $\|w^t - w_{\text{ML}}\|_2$ starts from $\|w_{\text{ML}}\|_2$ since $w^0 = [0 \quad 0]^T$.

It then drops rapidly in the initial iterations and fluctuates in subsequent iterations.

These fluctuations settle down at around the 4000^{th} iteration and $\|w^t - w_{\text{ML}}\|_2$ remains relatively constant as it approaches 0.

The fluctuations in $\|w^t - w_{\text{ML}}\|_2$ can be attributed to the stochastic nature of the algorithm.

Part 4

For our dataset and for some $\lambda \in \mathbb{R}^+$, the ridge regression problem is to solve:

$$w_{\text{R}} = \underset{w \in \mathbb{R}^2}{\text{argmin}}(\|X^T w - y\|^2 + \lambda \|w\|^2)$$

We define:

$$f(w) = \|X^T w - y\|^2 + \lambda \|w\|^2$$

Thus,

$$\nabla f(w) = 2XX^T w - 2Xy + 2\lambda w$$

The pseudocode for gradient descent in this case is shown below:

initialize $w^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

for $t = 0,1,2,\ldots,10^5$

$\nabla f(w^t) = 2XX^T w^t - 2Xy + 2\lambda w^t$

if $\nabla f(w^t) < 10^{-10}$

end

else

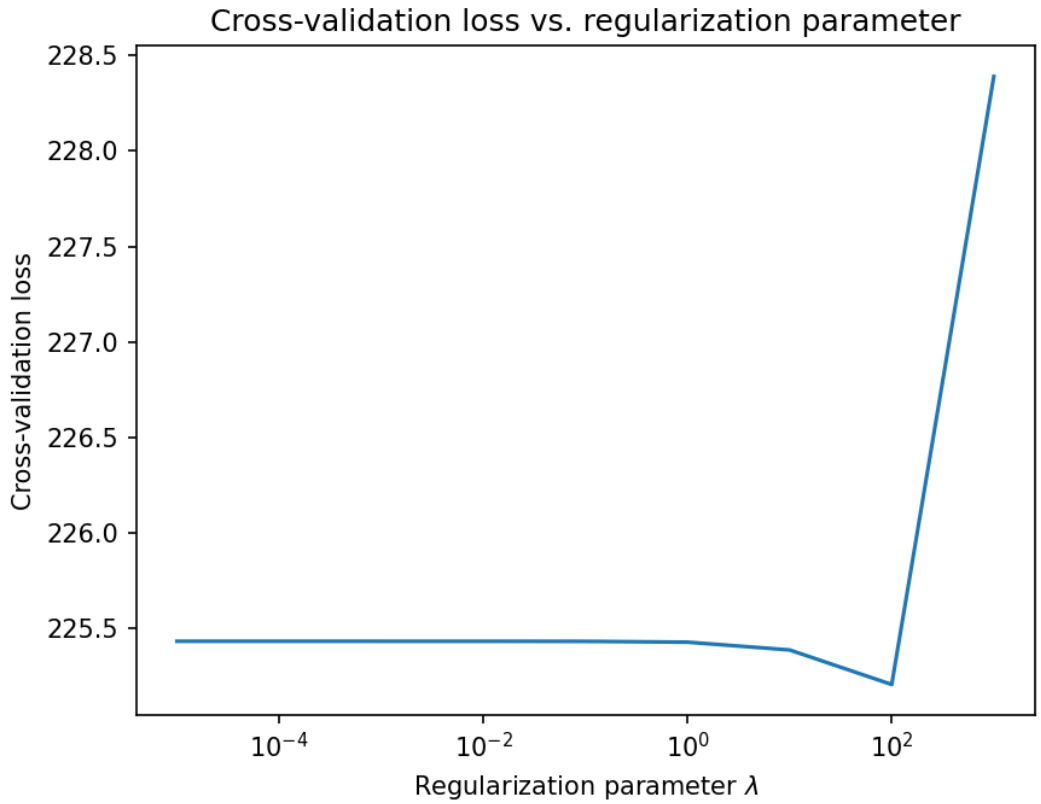
$w^{t+1} = w^t - 10^{-4} \nabla f(w^t)$

end

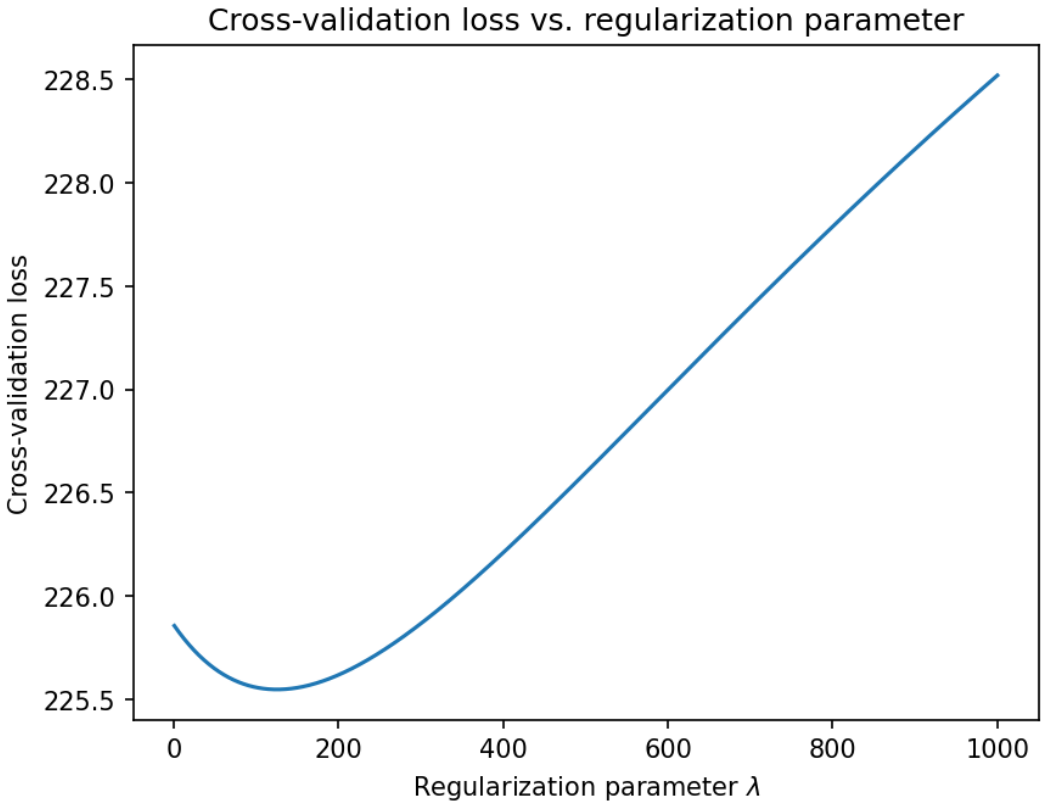
It is to be noted that the maximum number of iterations and the step size are the same as that of the standard gradient descent implemented earlier. The tolerance ε has been decreased to 1×10^{-10}.

To decide the best value of λ, we perform 5-fold cross-validation. The training data is divided into 5 folds (subsets). One of these folds is used as a validation set and w_R is computed by performing gradient descent using the remaining four folds. The mean squared error (MSE) is then computed using the validation set. This is repeated by using each of the 5 folds as the validation set. The average MSE across all 5 folds is reported as the error for a given λ. By repeating this procedure for different values of λ, we can find the best value to use for regularization.

To find the order of magnitude of the best λ, we try different values from $10^{-5}, 10^{-4}, \ldots, 10^{3}$ and report their cross-validation losses.



We find that λ has an order of magnitude of 10^2. To further narrow down the best value of λ, we scan different values between 10^0 and 10^3.



We find that the best λ is 126.

Using this λ, we compute w_{R} using gradient descent. It is found to be:

$$w_{\mathrm{R}} = \begin{bmatrix} 1.280 \\ 3.447 \end{bmatrix}$$

We can observe that:

$$\|w_{\mathrm{ML}}\|_2 = 4.145$$

$$\|w_{\mathrm{R}}\|_2 = 3.677$$

$$\Rightarrow \|w_{\mathrm{R}}\|_2 < \|w_{\mathrm{ML}}\|_2$$

We then compare the losses of w_{ML} of w_{R} on the test data provided. We find that:

$$\mathrm{MSE}(w_{\mathrm{ML}}) = 142.766$$

$$\mathrm{MSE}(w_{\mathrm{R}}) = 143.369$$

$$\Rightarrow \mathrm{MSE}(w_{\mathrm{R}}) \approx \mathrm{MSE}(w_{\mathrm{ML}})$$

Although the norm of w_{R} is lower than w_{ML}, it is able to achieve a similar loss to that of w_{ML}.

For our dataset, suppose the eigenvalues of $XX^T \in \mathbb{R}^{2\times 2}$ are λ_1, λ_2. These can be computed to be 1008.233 and 987.323 respectively.

We know that:

$$\mathrm{MSE}(w_{\mathrm{ML}}) \propto \mathrm{trace}((XX^T)^{-1}) = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} = 0.020$$

$$\mathrm{MSE}(w_{\mathrm{R}}) \propto \mathrm{trace}((XX^T + \lambda I)^{-1}) = \frac{1}{\lambda_1 + \lambda} + \frac{1}{\lambda_2 + \lambda} = 0.018$$

Since the values of λ_1, λ_2, and λ are quite high, $\frac{1}{\lambda_1+\lambda} + \frac{1}{\lambda_2+\lambda} \approx \frac{1}{\lambda_1} + \frac{1}{\lambda_2}$, and thus, we observe that $\mathrm{MSE}(w_{\mathrm{R}}) \approx \mathrm{MSE}(w_{\mathrm{ML}})$.

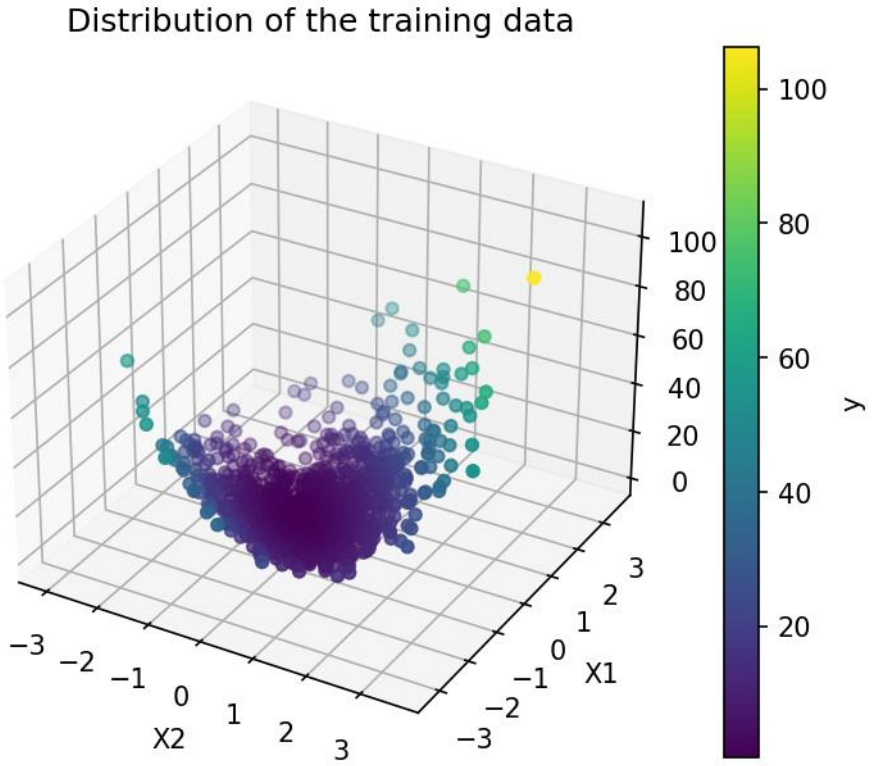
Part 5

For our dataset, we have to define an appropriate kernel function $k: \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$. We then compute the Gram matrix $K \in \mathbb{R}^{1000\times 1000}$ with $K_{ij} = k(x_i, x_j)$ for all pairs $x_i, x_j \in X$. Using this, we find $\alpha = K^{-1}y$. Here, $\alpha \in \mathbb{R}^{1000}$.

The kernel function is used as a "trick" to map the features into a higher-dimensional space.

When given a test data point $t \in \mathbb{R}^2$, we compute the predicted label as the weighted sum $\sum_{i=1}^{1000} \alpha_i \, k(x_i, t)$ using α and the value of the kernel function between the training points and the test point.

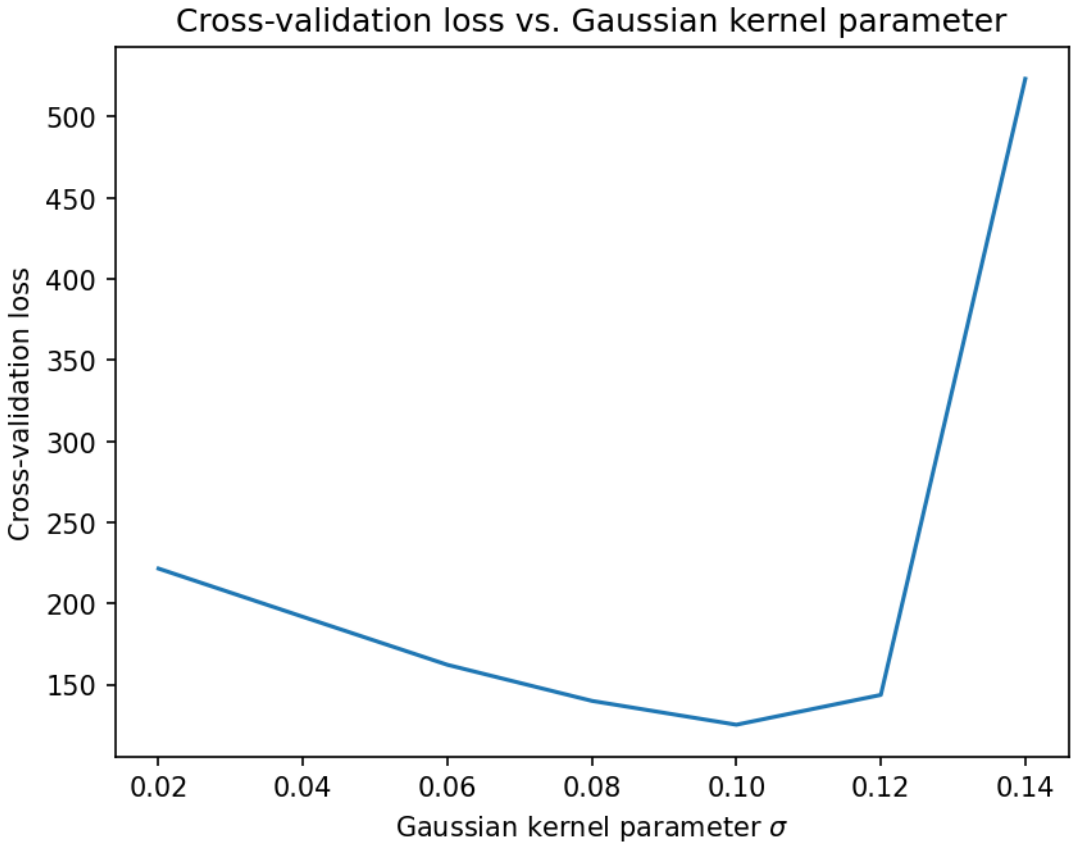
To decide the best kernel function to use, we first observe the distribution of the training data. Due to the dimensionality of the data, it can easily be visualized as a 3D plot.



We can observe that there is no clear "polynomial-like" dependence of the label y on the features. The dependence is complex and non-linear. Thus, an appropriate kernel to use is the Gaussian kernel:

$$k(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right)$$

To decide the best value of σ to use, we perform 5-fold cross-validation on the training data. We scan different values of σ between 0 and 0.15.



We observe that the best σ to use is 0.1.

Using this value of σ, we compute the loss of the kernel regression algorithm on the test data. We can observe that:

$$\text{MSE}(w_{\text{ML}}) = 142.766$$

$$\text{MSE}(\text{kernel}) = 78.088$$

$$\Rightarrow \text{MSE}(\text{kernel}) \ll \text{MSE}(w_{\text{ML}})$$

Kernel regression achieves a nearly 2-fold reduction in loss. This is because the Gaussian kernel effectively maps the training data to an "infinite"-dimensional space due to the nature of the exponential function. As our dataset is highly non-linear, the kernel function is able to capture these complex relationships. Standard linear regression cannot do so as it assumes a linear structure of the data. Thus, our kernel regression algorithm has significantly better performance than linear regression.