CIS 419/519: Applied Machine Learning | Fall 2024

**Homework 1**

*Handed Out: September 11* | *Due: 7:59 pm October 2*

**Name:** Alan Wu

**PennKey:** alanlwu

**PennID:** 41855518

# 1 Declaration

- **Person(s) discussed with:** *Peter Zhang, Arihant Tripathi*
- **Affiliation to the course: student, TA, prof etc.** *Students*
- **Which question(s) in coding / written HW did you discuss?** ***Question 1***
- **Briefly explain what was discussed.** *Effect of scaling c on the capacity and bias and variance of model*

# 2 Multiple Choice & Written Questions

1. (a) Increase variance; bias stays the same
   (b) Decrease variance; Increase bias
   (c) Decrease variance; Increase bias
   (d) Variance increases; Bias stays the same
   (e) Variance stays the same; Bias stays the same
   (f) for the following values to decrease test loss
   $n$: Increase n
   $\lambda$: Increase $\lambda$
   $d$: Decrease $d$
   $c$: Keep $c$ the same
   $\alpha$: Keep $\alpha$ the same

2. (a) To derive the gradient of the L1 regularization term for the:
   We need to simply take the gradient of the equation with respect to $B_j$.

$$L_{\ell_1} = \lambda \sum_{j=1}^{p} |B_j|$$

$$\frac{\partial L_{\ell_1}}{\partial B_j} = \lambda \, \mathrm{sign}(B_j)$$

We ignore the case where $B_j = 0$ and derive that equation from the loss function. Notice that the gradient of the L1 regularized loss function is only dependent on the sign of $B_j$ and the magnitude of $\lambda$.

(b) We can analyze the effect of the L1 regularization term on the parameters of the model by looking at the gradient of both the L1 regularization term and the MSE loss function with respect to $B_j$.

We can derive the full gradient of the loss function as such:

$$
\begin{aligned}
L_{\ell_1} &= \frac{1}{N}\sum_{i=1}^{N}(y_i - B^T x_i)^2 + \lambda \sum_{j=1}^{p} |B_j| \\
\frac{\partial L_{\ell_1}}{\partial B_j} &= \frac{-2}{N}\sum_{i=1}^{N}(y_i - B^T x_i)x_{ij} + \lambda \, \mathrm{sign}(B_j)
\end{aligned}
$$

Part 1: The MSE Loss Term

We will discuss the two following cases: There is no dependency on the value of $\lambda$ in the MSE loss term.

i. When $B_j$ is predictive of $y_i$

In this case, $B_j$ will be larger in magnitude and he gradient of the MSE loss term with respect to $B_j$ will be greater in magnitude. The gradient of the MSE loss term will also be larger, as it will influence changes in the gradient heavily.

ii. When $B_j$ is weakly or not predictive of $y_i$

When $B_j$ is weakly or not predictive of $y_i$, $B_j$ will be smaller in magnitude (based on least squares regression). The gradient of the MSE loss term with respect to $B_j$ will be smaller in magnitude. The gradient of the MSE loss term will also be smaller, as it will influence changes in the gradient less.

Thus, the magnitude of $B_j$ is dependent on how strongly correlated/predictive the feature $x_{ij}$ is in relation to $y_i$.

Part 2: The L1 Regularization term

As we derived earlier, the gradient of the L1 regularization term is only dependent on the sign of $B_j$ and the magnitude of $\lambda$.

The equation is: $\frac{\partial L_{\ell_1}}{\partial B_j} = \lambda \, \mathrm{sign}(B_j)$

This observation gives us two cases where the feature $B_j$ of the L1 loss regularization term will be scaled by the value of $\lambda$ and the sign of $B_j$.

And when we scale $\lambda$, we will observe the following:

i. Small $\lambda$

   For small $\lambda$, the L1 regularization term will have a small impact on the overall gradient. With respect to $B_j$, this means that many $B_j$ may be non-zero because the MSE loss term dominates the gradient. For both predictive and non-predictive features we may have non-zero values.

ii. Large $\lambda$

   For large $\lambda$, the L1 regularization term may overtake the MSE loss term in the gradient. For coeffcients $B_j$ where $B_j$ has small magnitude, the L1 regularization term will dominate and thus this parameter will be pushed down/up based on the sign of the parameter. Once the L1 regularization term dominates we will see that many $B_j$ will shrink to 0, as once the $B_j$ arrives at 0, it will stay there.

Regardless of the size of $\lambda$, it will always be the case that the regularization term will push the value of $B_j$ to 0. This is because the L1 regularization term has a constant dampening effect on the gradient, regardless of the magnitude of $B_j$.

(c) The L2 regularization loss equation:

We know that the L2 regularization loss function is defined as:

$L_{\ell_2} = \frac{1}{N}\sum_{i=1}^{N}(y_i - B^T x_i)^2 + \lambda \sum_{j=1}^{p} B_j^2$

Therefore, the gradient of the L2 regularization term with respect to $B_j$ is:

$\frac{\partial L_{\ell_2}}{\partial B_j} = 2\lambda B_j$

Just purely based on the gradient of the L2 regularization term, we can see that the L2 regularization term is dependent on the magnitude of both $\lambda$ and $B_j$, as well as the sign of $B_j$.

Compared to the gradient of the L1 regularization term, which solely depends on the sign of $B_j$ and the magnitude of $\lambda$. This is a key difference; because the L1 gradient only depends on the sign of the $B_j$, this means that regardless of the magnitude of $B_j$, L1 regualrization will apply a constant dampening term to the gradient. This is not the case for L2 regularization.

Because of this dependency, we can see that the L2 regularization term will not create a sparse matrix. In the scenario that we have values $B_j$ that are very small (feature $x_{ij}$ is weakly predictive of $y_i$), the gradient of the L2 regularization term will simply increase less as the value of $B_j$ approaches 0. However, the L2 regularization term will not push the value of $B_j$ to 0. As $B_j$ gets smaller, the L2 term will not overtake the MSE loss term like in L1 regularization. Therefore, it does not create a sparse parameter matrix.

3. (a) We will do two derivations in this question. The first derivation will be the gradient of the loss function with respect to $w_1^*$. The second will be the gradient of the loss function with respect to $w_0^*$

.

We are given: $J(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2$

First let us derive $\frac{\partial J(w)}{\partial w_1^*}$:

$$\begin{aligned}\frac{\partial J(w)}{\partial w_1^*} &= \frac{\partial}{\partial w_1^*}[\frac{1}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2] \\ &= \frac{-2}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)x_i\end{aligned}$$

Next, let us derive $\frac{\partial J(w)}{\partial w_0^*}$:

$$\begin{aligned}\frac{\partial J(w)}{\partial w_0^*} &= \frac{\partial}{\partial w_0^*}[\frac{1}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2] \\ &= \frac{-2}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)\end{aligned}$$

(b) In order to show that $\frac{1}{n}\sum_{i=1}^{n}(y_i - w_0^* - w_1^* x_i)(x_i - \bar{x}) = 0$, we need to manipulate the gradients found from the previous question to show that the equation is true.

We will find the optimal values of $w_0$ and $w_1$ by setting their gradient equations to zero, and algebraically determining a form for each term, then substituting it in to show that the above expression is true. Setting the gradient to zero and holding the other term constant, doing this twice, will yield the minimum of the gradient or the optimal solution we are looking for.

First, we will set the gradient of loss with respect to $w_0$ to 0:

$$\frac{-2}{n}\sum_{i=1}^{n}(y_i - w_0^* - w_1^* x_i) = 0$$
$$= \frac{-2}{n}[\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} w_0^* - w_1^* \sum_{i=1}^{n} x_i]$$
$$= -2 * (\bar{y} - w_0^* - w_1^* \bar{x}) = 0$$
$$w_0^* = \bar{y} - w_1^* \bar{x}$$

Next, we will set the gradient of loss with respect to $w_1$ to 0:

$$\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1^* x_i)x_i = 0$$
$$= \frac{-2}{n}[\sum_{i=1}^{n}(y_i x_i - w0 x_i - w_1^* x_i^2)] = 0$$
$$= \frac{-2}{n}[\sum_{i=1}^{n}(y_i x_i - (\bar{y} - w_1^* \bar{x})x_i - w_1^* x_i^2)] = 0$$
$$= \frac{-2}{n}[\sum_{i=1}^{n}(y_i - \bar{y})x_i + w_1^* \sum_{i=1}^{n}(\bar{x} - x_i)x_i] = 0$$
$$= \frac{-2}{n}[\sum_{i=1}^{n}(y_i - \bar{y})x_i - w_1^* \sum_{i=1}^{n}(x_i - \bar{x})x_i] = 0$$
$$w_1^* = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

A couple of notes in this derivation:

We substituted the equation for $w_0^*$ into the equation to simplify our derivation. We also utilized a statistical algebraic manipulation twice, saying that $\sum_{i=1}^{n}(y_i - \bar{y})x_i = \sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})$. Same algebraic manipulation to say that $\sum_{i=1}^{n}(x_i - \bar{x})x_i = \sum_{i=1}^{n}(x_i - \bar{x})^2$

Now that we have those two derivations, we can substitute the optimal values of $w_0^*$ and $w_1^*$ into the original equation to show that it is true:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - w_0^* - w_1^* x_i)(x_i - \bar{x}) = 0$$

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y} + w_1^*\bar{x} - w_1^* x_i)(x_i - \bar{x}) = 0$$

$$= \frac{1}{n}[\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}) - w_1^* \sum_{i=1}^{n}(x_i - \bar{x})^2]$$

$$= \frac{1}{n}[\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}) - \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \sum_{i=1}^{n}(x_i - \bar{x})^2]$$

$$= \frac{1}{n}[\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}) - \sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})]$$

$$= \frac{1}{n}[0] = 0$$

(c) The overarching solution to this problem is that linear regression will not yield a unique solution for every dataset. For some datasets, there will indeed be a unique solution. However, this is not the case for all datasets

We know the closed form solution for the linear regression problem is: $B_{\text{closed form}} = (X^T X)^{-1} X^T Y$

There are two cases for any given dataset:

i. The dataset does indeed have a unique solution

If the dataset does indeed have a unique solution, then this means that we have a solution to the analytical solution for $B$. This would simply be the expression $B = (X^T X)^{-1} X^T Y$. In this scenario, we know that the data matrix $X$ is full rank, and thus the matrix $X^T X$ is invertible. This means that all the columns or features in the dataset are linearly independent.

ii. The dataset has multiple solutions

If the dataset does not have a unique solution, then the analytical solution for $B$ is not applicable. In this case, the matrix $X^T X$ is not invertible. This could be because the columns of the data matrix $X$ are linearly dependent. This means that there are an infinite number of combinations to assign to the weights $B$ that will yield the same outcome y.

Therefore, linear regression is not guaranteed to have a unique solution for every dataset.

4. (a) The value of the loss function at the beginning is going to be the loss function evaluated with the initial weights of $w = [0, 0, 0]^T$

$$L(w) = \frac{1}{N}\sum_{i=1} N(y_i - wx_i)^2 + \lambda||w||^2$$

$$L(w_{\text{initial}}) = \frac{1}{2}[(0 - [0,0]\cdot[1,-1])^2 + (1 - [0,0]\cdot[-1,-1])^2] + 1\cdot||[0,0]||^2$$
$$= \frac{1}{2}(1) = 0.5$$

Therefore, the value of the loss function initially is **0.5**

(b) To find the final state of the trained weight vector after 2 steps and the corresponding value of the loss function, we need to compute the gradient at each step, and update the gradient and loss based on that value of the gradient. We will iteratively compute this:

Computing the gradient of the loss function:

$$\frac{\partial L(w)}{\partial w} = \frac{\partial}{\partial w}\frac{1}{N}\sum_{i=1} N(y_i - wx_i)^2 + \lambda||w||^2$$
$$= \frac{\partial}{\partial w}\frac{1}{N}\sum_{i=1} N(y_i - wx_i)^2 + \lambda\sum_{j=1}^{p} w_j^2$$
$$= \frac{-2}{N}\sum_{i=1}^{N}(y_i - wx_i)x_i + 2\lambda w$$

We also know that the gradient updates as the following: $w_{t+1} = w_t - \alpha\nabla L(w_t)$ where the weight vector of the next step is going to be the weight of the current step subtracted by the gradient of the loss function at the current step scaled by the learning rate

With this information, we can compute the weight vector after 2 steps by doing the following:

First we need to compute the value of the weight vector after step 1, of which we first need to compute the gradient at the first step:

$$\nabla L(w_{\text{initial}}) = \frac{-2}{2}\sum_{i=1}^{2}(y_i - [0,0]\cdot x_i)x_i + 0$$
$$= -1\cdot[(0-0)[1,-1] + (1-0)[-1,-1]]$$
$$= -1\cdot[-1,-1] = [1,1]$$

Now we update the weight vector:

$$\begin{aligned} w_{\text{step 1}} &= w_{\text{initial}} - \alpha \nabla L(w_{\text{initial}}) \\ &= [0, 0] - 1 \cdot [1, 1] = [-1, -1] \\ w_{\text{step 1}} &= [-1, -1] \end{aligned}$$

We need to compute the gradient at the second step with this new weights vector:

$$\begin{aligned} \nabla L(w_{\text{step 1}}) &= \frac{-2}{2} \sum_{i=1}^{2} (y_i - [-1, -1] \cdot x_i) x_i + 2 \cdot 1 \cdot [-1, -1] \\ = -1 \cdot [(0 - [-1, -1] \cdot [1, -1])[1, -1] &+ (1 - [-1, -1] \cdot [-1, -1])[-1, -1]] + [-2, -2] \\ &= -1 \cdot [[1, 1]] + [-2, -2] \\ &= [-1, -1] + [-2, -2] = [-3, -3] \end{aligned}$$

Now we update the weight vector again using the gradient:

$$\begin{aligned} w_{\text{step 2}} &= w_{\text{step 1}} - \alpha \nabla L(w_{\text{step 1}}) \\ &= [-1, -1] - 1 \cdot [-3, -3] = [2, 2] \\ w_{\text{step 2}} &= [2, 2] \end{aligned}$$

We have to remember that all the operations are vectors, so we are subtracting/-multiplying vector.

Finally, we will compute the loss function after 2 epochs:

$$\begin{aligned} L(w_{\text{step 2}}) &= \frac{1}{2} \sum_{i=1}^{2} (y_i - [2, 2] \cdot x_i)^2 + 1 \cdot ||[2, 2]||^2 \\ = \frac{1}{2}[(0 - [2, 2] \cdot [1, -1])^2 &+ (1 - [2, 2] \cdot [-1, -1])^2] + 1 \cdot (2^2 + 2^2) \\ &= \frac{1}{2}[(0 - 0)^2 + (1 - (-4))^2] + 8 \\ &= \frac{1}{2}(25) + 8 = 12.5 + 8 = 20.5 \end{aligned}$$

The final loss after 2 epochs of training is **20.5**

(c) In order to solve the closed form formula for the ridge regression loss function, we need to rewrite the loss function in matrix form and then derive the expression by taking the gradient and setting it equal to 0.

First we need to rewrite the loss function:

$$
\begin{aligned}
L(w) = \frac{1}{N} = \sum_{i=1} N(y_i - wx_i)^2 + \lambda||w||^2 \\
= \frac{1}{N}||y - xw||_2^2 + \lambda||w||^2 \\
= \frac{1}{N}(y - xw)^T(y - xw) + \lambda w^T w \\
= \frac{1}{N}(y^T y - 2y^T xw + w^T x^T xw) + \lambda w^T w
\end{aligned}
$$

Next we need to compute the gradient using matrix calculus:

$$
\begin{aligned}
L(w) &= \frac{1}{N}(y^T y - 2y^T xw + w^T x^T xw) + \lambda w^T w \\
\frac{\partial L(w)}{\partial w} &= \frac{\partial}{\partial w}\frac{1}{N}(y^T y - 2y^T xw + w^T x^T xw) + \lambda w^T w \\
&= \frac{1}{N}(0 + -2x^T y + 2x^T xw) + 2\lambda w \\
&= \frac{1}{N}(-2x^T y + 2x^T xw) + 2\lambda w
\end{aligned}
$$

Finally, we need to compute the expression for $w$ by minimizing the gradient:

$$
\begin{aligned}
\frac{\partial L(w)}{\partial w} &= \frac{1}{N}(-2x^T y + 2x^T xw) + 2\lambda w \\
0 &= \frac{1}{N}(-2x^T y + 2x^T xw) + 2\lambda w \\
0 &= \frac{-2}{N}x^T y + \frac{2}{N}x^T xw + 2\lambda w \\
\frac{2}{N}x^T y &= w(\frac{2}{N}x^T x + 2\lambda) \\
\frac{1}{N}x^T y &= w(\frac{1}{N}x^T x + \lambda) \\
x^T y &= w(x^T x + N\lambda) \\
w &= (x^T x + N\lambda)^{-1}x^T y
\end{aligned}
$$

Therefore, the solution to the ridge regression solution is $w = (x^T x + N\lambda)^{-1}x^T y$

# 3 Python Programming Questions

**Question 1.3**

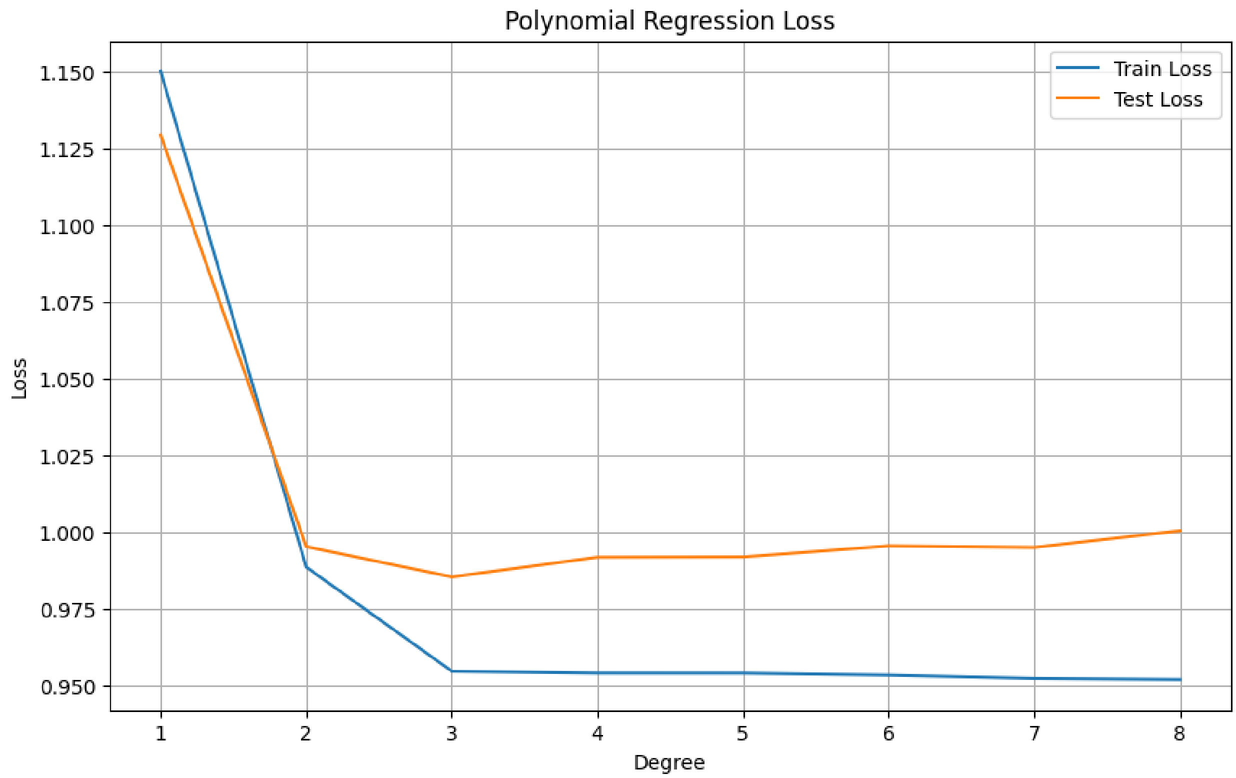


Figure 1: Plot of the Training and Test loss for a polynomial regression model

It is evident that when the degree is low, both the train and test loss are decreasing. This is indicative of the model fitting the data well. However, as the degree increases, specifically the trend from degree = 4 to degree = 8, we start seeing the test loss start to increase. This is a sign of overfiting of the data as the model is fitting the training data too well and cannot generalize to new unseen samples. This behavior makes sense, as when our model capacity gets too great (high degree polynomial), the model will start to fit the noise in the data and not the underlying pattern.

**Question 1.4**

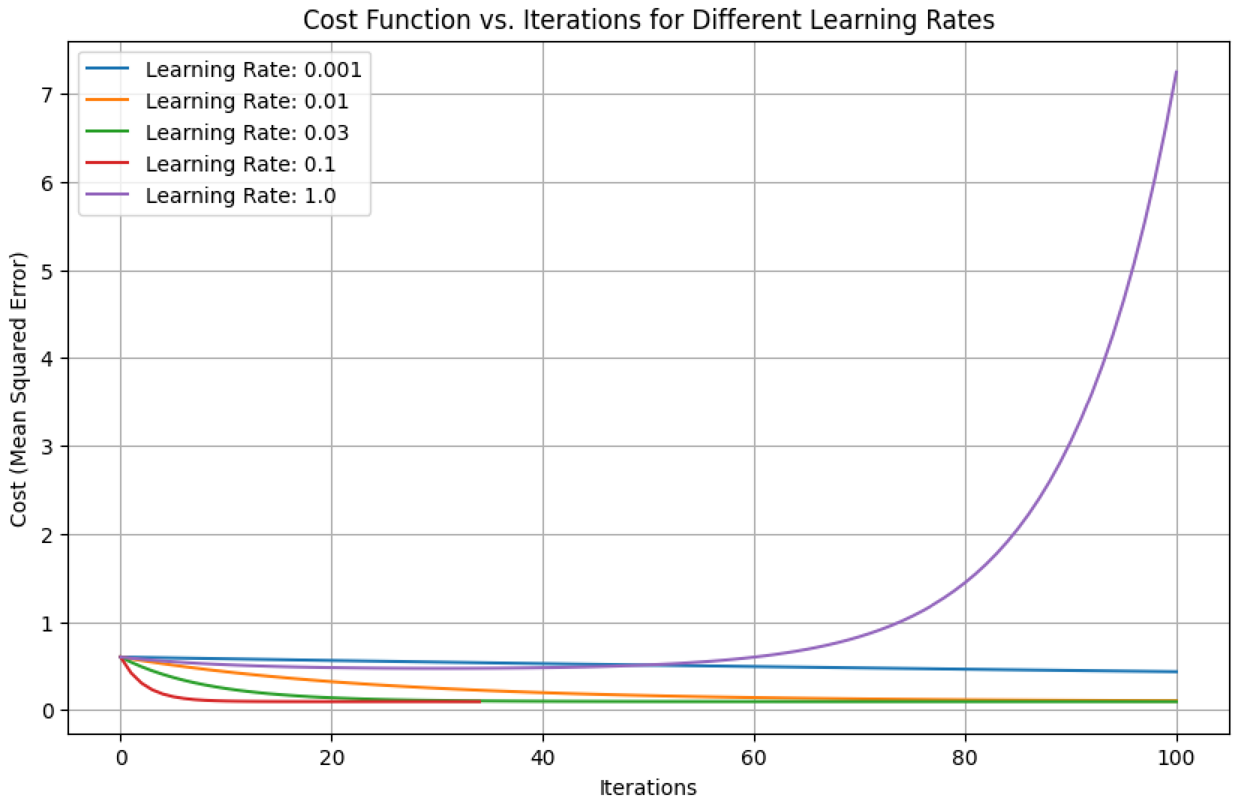


Figure 2: Training loss for different step sizes

We can see that for the lower learning rates, the model converges to a lower loss than the initial loss. We notice too that the learning rates of 0.001 - 0.1. It is evident, however, that once we increase the learning rate too much, the loss no longer decreases and the model diverges. The massive increase for the learning rate of 1.0 is due to the fact that the model is overshooting the minimum and is unable to converge (gradient explodes). In our tests and based on the graph, it seems that learning rates 0.1, 0.03, and 0.01 are all approaching the same loss after 100 iterations and thus perform the best lowest loss.