

## HW-1

Q.1 Polynomial Regression as linear least squares

①

$$\hat{b} = \arg \min_b \|x b - y\|_2^2$$

$x \rightarrow$  design matrix of size  $N \times (d+1)$

$y \rightarrow$  response vector

$b \rightarrow$  parameter vector.

Loss Function -

$$\|x b - y\|_2^2 = (x b - y)^T (x b - y).$$

$$= (x b)^T x b - 2y^T x b + y^T y.$$

To minimizer, taking derivative of the loss function wrt  $b$ .

$$\frac{\partial}{\partial b} ((x b)^T x b - 2y^T x b + y^T y).$$

$$\therefore \frac{\partial}{\partial b} = 2x^T [2x^T x b - 2x^T y].$$

To minimize the loss  $\uparrow$ ,

$$\text{Let } 2x^T x b - 2x^T y = 0$$

$$\therefore \boxed{2x^T x b = 2x^T y}$$

To solve for  $\hat{b}$ ,

Let us assume that  $X^T X$  is invertible.

$$\boxed{\hat{b} = (X^T X)^{-1} X^T y.}$$

Therefore, empirical risk minimizer  $\hat{b}$  is given by

$$\boxed{\hat{b} = (X^T X)^{-1} X^T y}$$

$$(y - d_x)^T (y - d_x) = \|y - d_x\|^2$$

$$y^T y + d_x^T d_x - d_x^T (y - d_x) =$$

(2)

Given Loss Function -

$$L(b) = \|x_b - y\|_2^2 = (x_b - y)^T (x_b - y).$$

$$\therefore L(b) = (x_b)^T x_b - 2y^T x_b + y^T y.$$

$$\frac{\partial L(b)}{\partial b} = 2x^T x_b - 2y^T x_b$$

Simplifying,

$$\boxed{x^T x_b = x^T y}$$

Now, if  $X^T X$  is invertible,

$$\hat{b} = (X^T X)^{-1} X^T y$$

Need for cond<sup>n</sup>  $N > d$  &  $X$  being full rank.

condition 1 : ( $N > d$ ) (more samples less features).

If  $N > d$ , it means that there are more samples  $a$  than features. This ensures that matrix  $X^T X$  is at least square  $[(d+1) \times (d+1)]$  & hence potentially invertible.

If  $N \leq d$ ,  $X^T X$  would not be invertible because it would be singular. and we wouldn't be able to compute the inverse.

condition 2 -  $X$  being full rank.

$X$  being full rank means that the columns of  $X$  are ~~weakly~~ linearly dependent, which causes  $X^T X$  to be singular.

In this case, normal equation

$X^T X b = X^T y$  would not have a unique solution.

## Ridge Regression -

Q. 11]

### Objective Function -

$$J_{\lambda}(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i)^2 + \lambda \theta^T \theta$$

$$h_{\theta}(x_i) = \theta^T x \quad (\text{hypothesis Function})$$

$m \rightarrow$  number of training Examples.

$\lambda \rightarrow$  Regularization Parameter

Converting it in a matrix form,

$$J_{\lambda}(\theta) = \frac{1}{2m} \|X\theta - y\|_2^2 + \frac{\lambda}{2m} \|\theta\|_2^2$$

where,

$X$  is the design matrix ( $m \times n$ )

$y$  is the target vector ( $m \times 1$ )

$\theta$  is the parameter vector ( $n \times 1$ ).

Gradient of the loss term,

$$\nabla_{\theta} \left[ \left( \frac{1}{2m} \|X\theta - y\|_2^2 \right) + \left( \frac{\lambda}{2m} \|\theta\|_2^2 \right) \right] = \left( \frac{1}{m} \right) X^T (X\theta - y)$$

Gradient of regularization term,

$$\nabla_{\theta} \left[ \left( \frac{\lambda}{m} \right) \|\theta\|_2^2 \right] = \left( \frac{\lambda}{m} \right) \theta$$

Full gradient,

$$\nabla_{\theta} J_{\lambda}(\theta) = \frac{1}{m} x^T (x\theta - y) + \frac{\lambda}{m} \theta.$$

Gradient descent update rule,

$$\theta = \theta - \alpha \nabla_{\theta} J_{\lambda}(\theta).$$

Sub. the value of gradient,

$$\theta = \theta - \alpha \left[ \left( \frac{1}{m} \right) x^T (x\theta - y) + \left( \frac{\lambda}{m} \right) \theta \right].$$

$$= \theta - \frac{\alpha}{m} \left[ x^T (x\theta - y) + \left( \frac{\lambda}{m} \right) \theta \right]$$

Thus,

$$\text{Gradient} \rightarrow \nabla_{\theta} J_{\lambda}(\theta) = \left( \frac{1}{m} \right) x^T (x\theta - y) + \left( \frac{\lambda}{m} \right) \theta.$$

$$\text{Update Rule} \rightarrow \theta - \left( \frac{\alpha}{m} \right) \left[ x^T (x\theta - y) + \lambda \theta \right].$$

### 3. Image classification with regularized logistic regression.

(16)

Given function for Logistic Loss for Binary classification:

$$L(\theta, b) = \frac{1}{m} \sum_{i=1}^m \log(1 + e^{-y_i h_{\theta,b}(x_i)}).$$

where,

$$y_i \in \{-1, 1\}, \hat{y}_i = h_{\theta,b}(x_i) = \theta^T x_i + b$$

The margin  $m_i$  for a given sample  $i$  is

$$m_i = y_i h_{\theta,b}(x_i) = y_i (\theta^T x_i + b).$$

Now

Logistic loss in terms of margin

$$L(\theta, b) = \frac{1}{m} \sum_{i=1}^m \log(1 + e^{-m_i}).$$

$$\text{where } m_i = y_i h_{\theta,b}(x_i).$$

We can write the objective  $f$  in terms of  $y_i \in \{-1, 1\}$  as

$$L(\theta, b) = \frac{1}{m} \sum_{i=1}^m [y_i \log(1 + e^{-h_{\theta,b}(x_i)}) + (1-y_i) \log(1 + e^{h_{\theta,b}(x_i)})]$$

Rewriting this using  $y_i \in \{-1, 1\}$ :

$$(1+y_i)/2 = 1 \text{ if } y_i = 1 \text{ and}$$

$$\text{if } y_i = 0$$

Also,

$$(1-y_i)/2 = 1 \text{ if } y_i = -1 \text{ and } y_i = 1.$$

Using this, we can write the final objective function as

$$L(\theta) = \frac{1}{2m} \sum_{i=1}^m [(1+y_i) \log(1 + e^{-h_{\theta,b}(x_i)}) + (1-y_i) \log(1 + e^{-h_{\theta,b}(x_i)})]$$

17

→ We know that,

The logistic function for binary classification

$$L(\theta) = \frac{1}{2m} \sum_{i=1}^m \left[ (y_i \log(1 + e^{-\theta^\top x_i})) + (1 - y_i) \log(1 + e^{-\theta^\top x_i}) \right]$$

for labels  $y_i \in \{-1, 1\}$ .

with  $L_1$  regularization, it becomes:

$$L_{\text{regularized}}(\theta) = L(\theta) + \alpha \|\theta\|_1$$

where,

$\alpha \rightarrow$  Regularization parameter. A higher  $\alpha$  increases the regularization effect.

$\|\theta\|_1 \rightarrow$  sum of the absolute values of co-efficients of  $\theta$ .

After regularizing the coefficients of  $\theta$  with  $L_1$  penalty using reg. param  $\alpha$ :

$$L(\theta)_{\text{reg.}} = \frac{1}{2m} \left[ (y_i \log(1 + e^{-\theta^T x_i}) + (1 - y_i) \log(1 + e^{-\theta^T x_i})) \right] + \alpha \|\theta\|_1$$

After L<sub>1</sub> regularization,

- ① Some coefficients in  $\theta$  may be driven exactly to zero.
- ② The loss function effectively performs feature selection by eliminating less important features.
- ③ Increasing  $\alpha$  can potentially reduce model accuracy.

(20)

→ Sources of Randomness -

(i) sampling of data -

Each time the experiment is repeated, a different random subset of training data is selected. Since we are subsampling the dataset to a smaller number of experiments, selected data points in each repetition vary.

This causes randomness because different subsets of the data might represent slightly different distributions, affecting the model's performance.

(ii)

Random Initialization -

The SGD classifier uses stochastic gradient descent to optimize model parameters.

During training process, there is inherent randomness in how model weights are initialized & how the algorithm proceeds with updates. Repeating the experiment multiple times & averaging over results helps mitigate the effect of this randomness.

(21)

→ The optimal value of the regularization parameter  $\alpha$  is the one that minimizes the mean test classification error.

In our experiment, [0.001025] would be the optimal value for  $\alpha$  after conducting for the values ranging from  $10^{-4}$  to  $10^{-1}$ .

This value produced the lowest mean test classification error across 10 repeated experiments.

(23)

Pattern in  $\theta$  -

- i Sparsity - Higher  $\alpha$  leads to sparser  $\theta$ , with many co-efficients near zero due to regularization.
- ii Feature selection - Lower  $\alpha$  uses more features, while higher  $\alpha$  keeps only the most important ones.
- iii Smaller  $\alpha$  allows larger coefficients, with larger  $\alpha$  shrinks them.
- iv With low  $\alpha$ , areas corresponding to digits shapes have higher coefficients.

## Regularization Effects -

(i) Increasing  $\alpha$  produces simpler, sparser models to prevent overfitting

(ii) Bias - Variance Relation -

smaller  $\alpha$  results in models with lower bias but higher variance, leading to overfitting

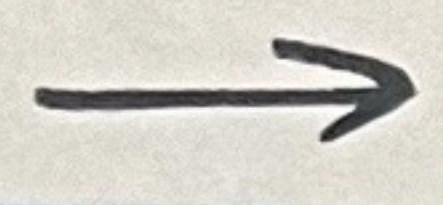
Larger  $\alpha$  results in higher bias but lower variance. which can cause in underfitting if the  $\alpha$  is too large

(iii) If the  $\alpha$  is smaller, the plot results in a smoother pattern.

But if the  $\alpha$  is larger, the plot results in a blocked pattern.

## Linear Regression

Q. 2.



Given - Hypothesis of linear function,

$$h_{\theta} : \mathbb{R}^d \rightarrow \mathbb{R}$$

where

$$h_{\theta}(x) = \theta^T x$$

for  $\theta, x \in \mathbb{R}^d$

Avg. square loss function,

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i)^2$$

Now,

$x \in \mathbb{R}^{m \times (d+1)}$  be design matrix

$y = (y_1, \dots, y_m)^T \in \mathbb{R}^{m \times 1}$  be response vector

Let  $\theta \in \mathbb{R}^{(d+1) \times 1}$  be parameter vector.

The objective function in matrix/vector:

$$J(\theta) = \frac{1}{m} \|x\theta - y\|_2^2$$

$x\theta \rightarrow$  vector of predictions for all training examples

$\| \cdot \|_2^2 \rightarrow$  sq. Euclidean form which sums the sq. of elements of vector

$y \rightarrow$  vector of actual outputs.

$\| x\theta - y \|_2^2 \rightarrow$  it represents the sum. of sq. differences, hence we don't need an explicit summation.

Thus, final matrix form is

$$J(\theta) = \frac{1}{m} (x\theta - y)^T (x\theta - y)$$

Q. 3

→ Matrix form of  $J(\theta)$ :

$$J(\theta) = \frac{1}{m} (x\theta - y)^T (x\theta - y).$$

Expanding,

$$J(\theta) = \frac{1}{m} (x^T \theta^T - y^T)(x\theta - y)$$

$$= \frac{1}{m} (\theta^T x^T x\theta - \theta^T x^T y - y^T x\theta + y^T y)$$

Since,  $\theta^T x^T y$  and  $y^T x \theta$  are scalars & equal to their own transpose, they are equal.

By the rules of matrix calculus,

$$\nabla_{\theta} (\theta^T A \theta) = (A + A^T) \theta$$

$$\nabla_{\theta} (\beta^T \theta) = \beta$$

$$\nabla_{\theta} (\theta^T \beta) = \beta$$

Applying these & chain rule,

$$\nabla_{\theta} J(\theta) = \frac{1}{m} [2x^T x \theta - 2x^T y]$$

$$\boxed{\nabla_{\theta} J(\theta) = \frac{2}{m} x^T [x \theta - y]}$$

Q. 4.

→ The general update rule for gradient descent :

$$\theta : \theta - \eta \nabla_{\theta} J(\theta)$$

where,

$\theta \rightarrow$  generally current parameter vector.

$\eta \rightarrow$  learning rate

$\nabla_{\theta} J(\theta) \rightarrow$  gradient w.r.t  $\theta$

We have calculated the expression  
of gradient,

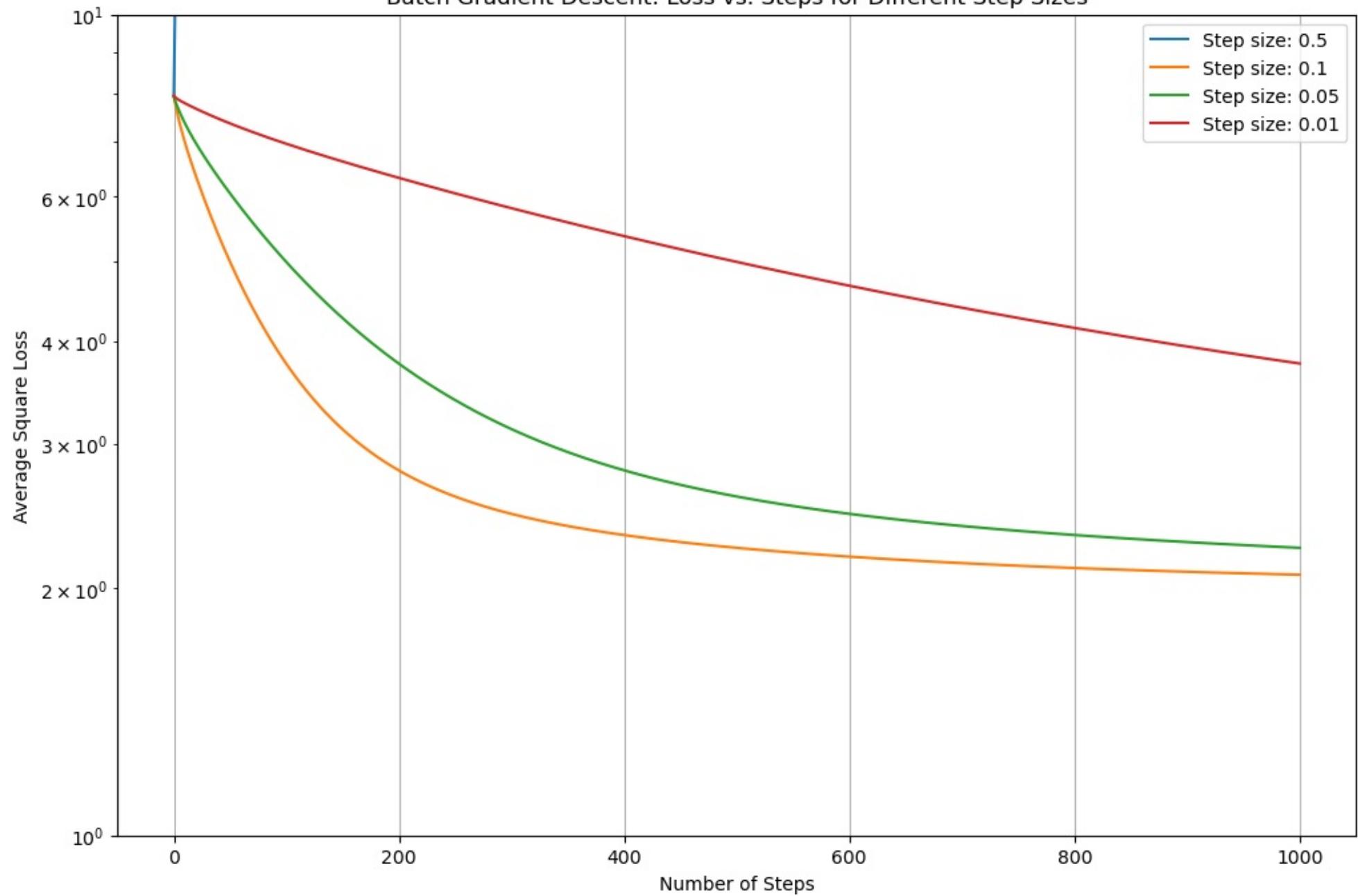
$$\nabla_{\theta} J(\theta) = \frac{2}{m} x^T (x\theta - y)$$

substituting,

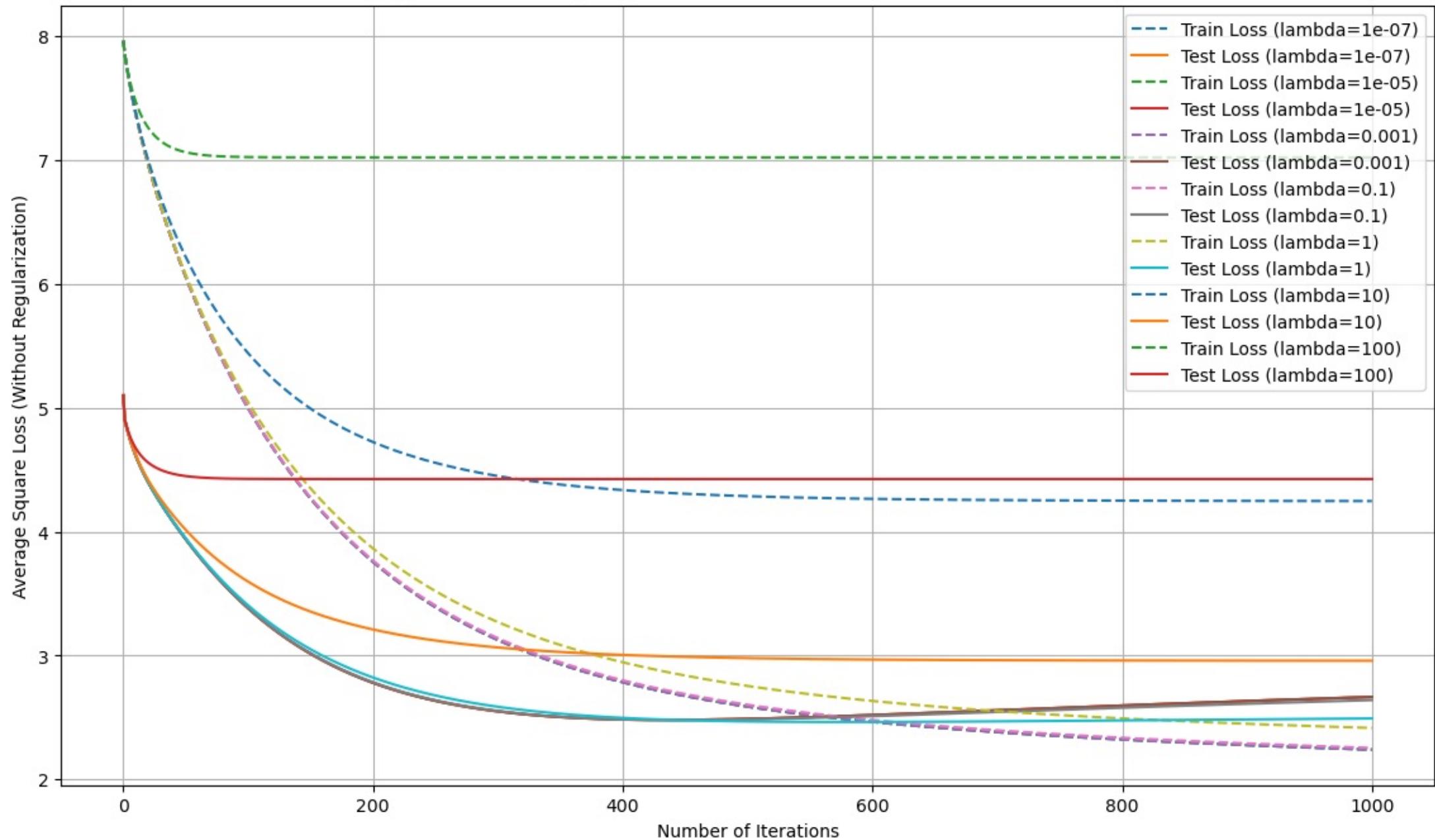
$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \left( \frac{2}{m} \right) x^T (x\theta_{\text{old}} - y).$$

$$\therefore \theta_{\text{new}} = \theta_{\text{old}} - \left( \frac{2\eta}{m} \right) x^T (x\theta_{\text{old}} - y).$$

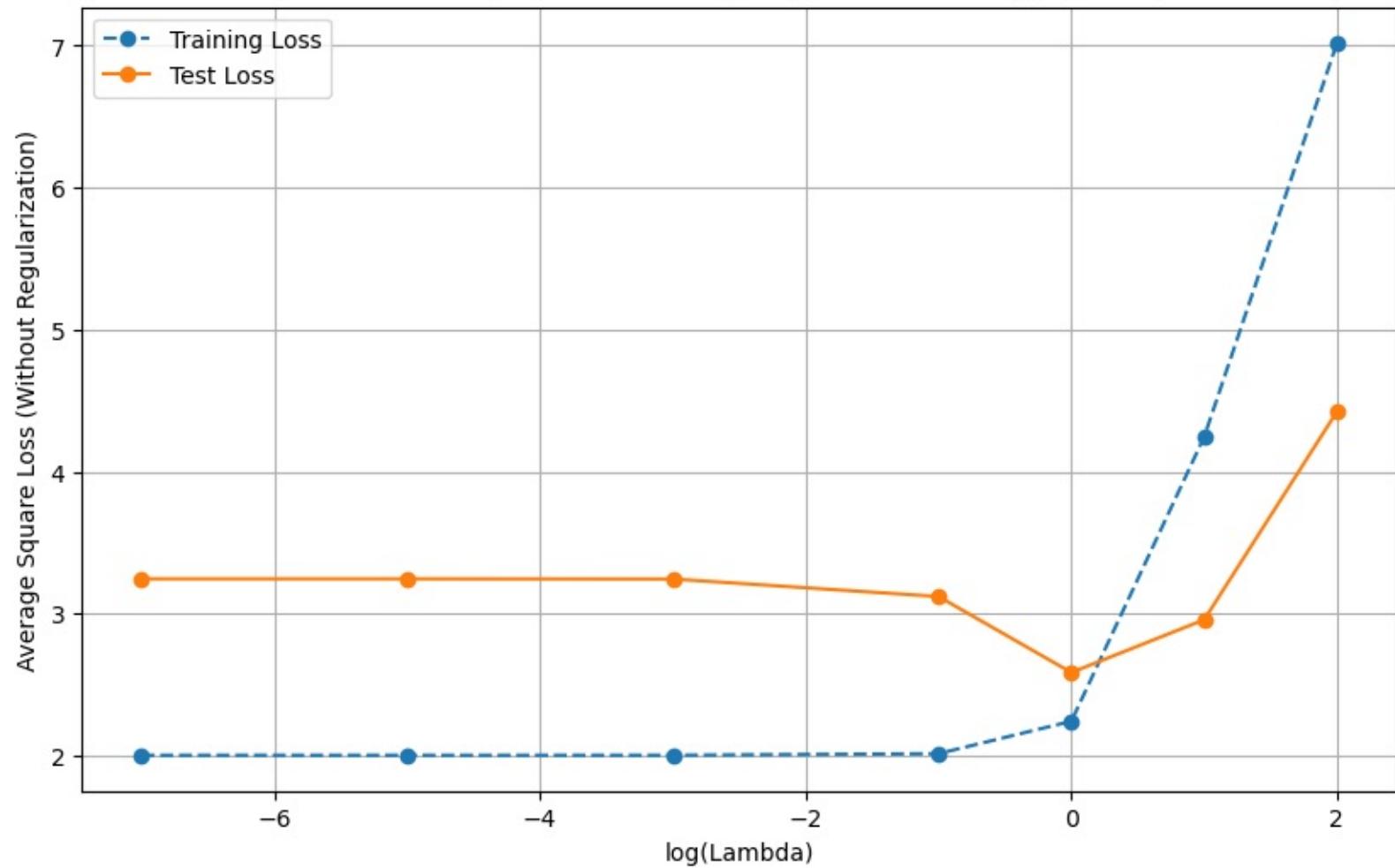
### Batch Gradient Descent: Loss vs. Steps for Different Step Sizes



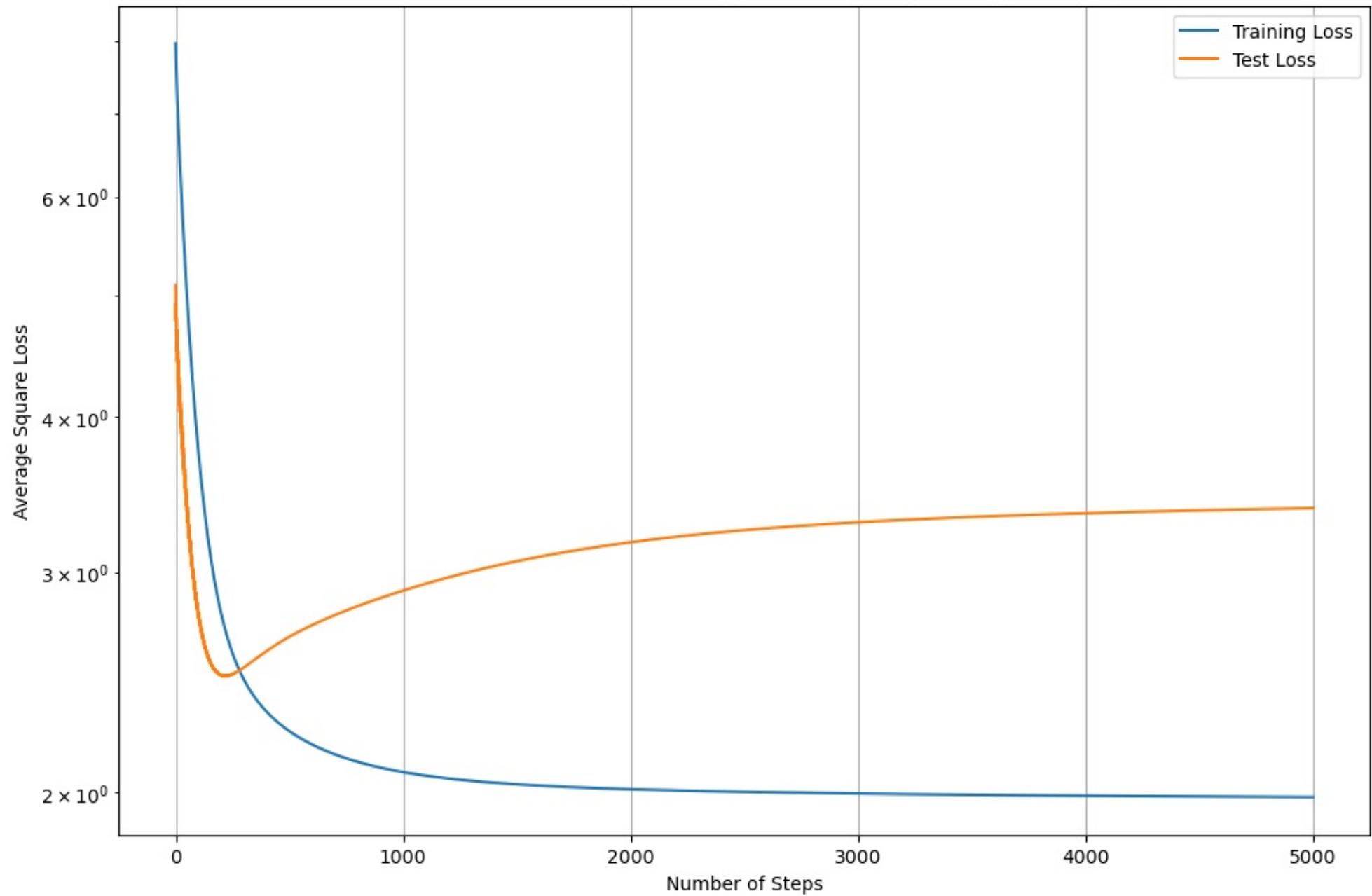
### Training and Test Average Square Loss vs. Iterations for Various Lambda Values



### Final Training and Test Average Square Loss vs log(Lambda)



Batch Gradient Descent: Training and Test Loss (Learning Rate: 0.1)



### Test Classification Error vs Regularization Parameter (alpha)

