NYCU Introduction to Machine Learning, Homework 1

109612019, Albert Lin

Part. 1, Coding (50%):

(10%) Linear Regression Model - Closed-form Solution

1. (10%) Show the weights and intercepts of your linear model.

```
Closed-form Solution
Weights: [2.85817945 1.01815987 0.48198413 0.1923993 ], Intercept: -33.78832665744907
```

(40%) Linear Regression Model - Gradient Descent Solution

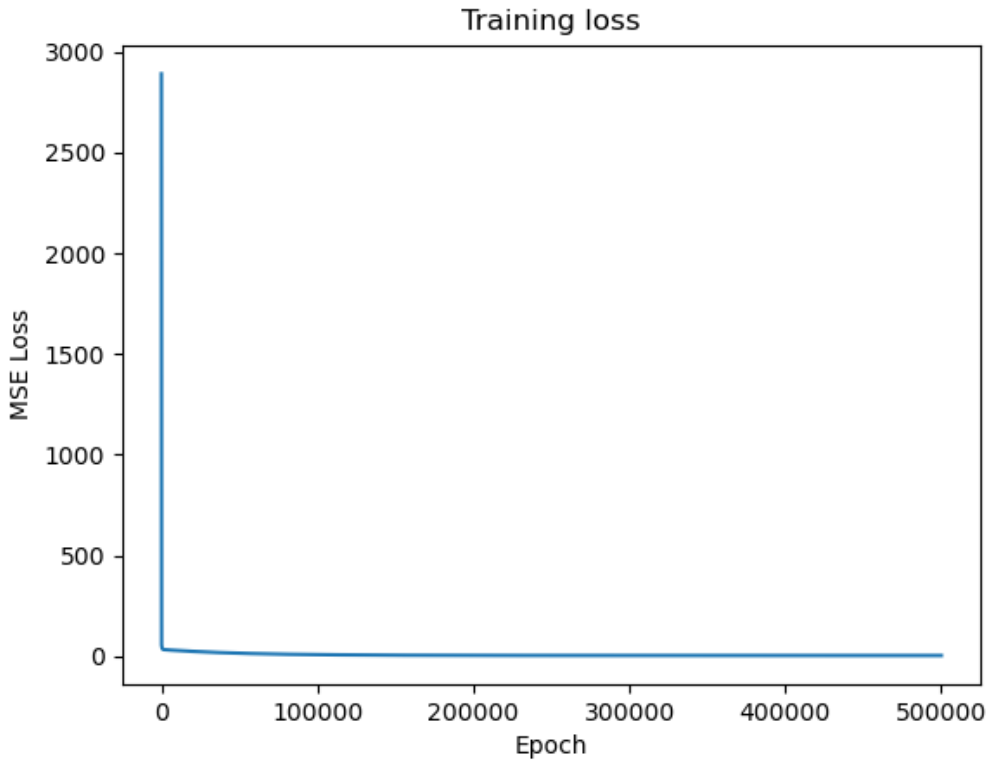
2. (0%) Show the learning rate and epoch (and batch size if you implement mini-batch gradient descent) you choose.

```
LR.gradient_descent_fit(train_x, train_y, lr=0.000185, epochs=500000)
```

3. (10%) Show the weights and intercepts of your linear model.

```
Gradient Descent Solution
Weights: [2.85325986 1.0166196  0.46679613 0.18881485], Intercept: -33.53461732982482
```

4. (10%) Plot the learning curve. (x-axis=epoch, y-axis=training loss)



5. (20%) Show your error rate between your closed-form solution and the gradient descent solution.

```
Error Rate: 0.0%
```

Part. 2, Questions (50%):

1. (10%) How does the value of learning rate impact the training process in gradient descent? Please explain in detail.

   - **Learning Rate Selection:**
     A very small learning rate will result in very tiny steps during each iteration. While this can lead to precise convergence to the minimum of the loss function, it can be extremely slow. If the learning rate is too small, the algorithm may converge too slowly or get stuck in a local minimum. Conversely, A large learning rate results in large steps during each iteration. While this can lead to faster convergence, it can also make the optimization process unstable. If the learning rate is too large, the algorithm may overshoot the minimum and diverge, result in failing to converge.

   - **Effects on Loss Trajectory**:
     A well-chosen learning rate will result in a smooth decrease in the loss function over time. The loss trajectory should gradually approach a minimum without oscillations or large fluctuations. Conversely, an improper learning rate can lead to unstable behavior in the loss trajectory. For example, a high learning rate can cause the loss to jump around or increase, making it difficult to determine whether the model is converging.

2. (10%) There are some cases where gradient descent may fail to converge. Please provide at least two scenarios and explain in detail.

   - **Exploding Gradients:**
     Exploding gradients occur when the gradients of the loss function with respect to the model's parameters become exceedingly large, typically due to deep networks, improper weight initialization, or a high learning rate. When gradients become too large, they can cause weight updates that overshoot the minimum of the loss function, making the optimization process unstable. If exploding gradients are not addressed, the optimization process may diverge, and the loss function can become NaN (not-a-number), rendering the training process unsuccessful.

   - **Vanishing Gradients:**
     When computing gradients during backpropagation, the gradient values for layers close to the input layer can become extremely small. As a result, the updates to the network's weights in those layers become negligibly small, effectively causing those layers to stop learning. This issue propagates backward through the network, making it difficult for the gradients to reach the deeper layers and adjust their weights properly. In such cases, the network may not learn meaningful representations in the deep layers, and training might stall or progress extremely slowly. This can lead to suboptimal or non-convergent models.

3. (15%) Is mean square error (MSE) the optimal selection when modeling a simple linear regression model? Describe why MSE is effective for resolving most linear regression problems and list scenarios where MSE may be inappropriate for data modeling, proposing alternative loss functions suitable for linear regression modeling in those cases.

➢ **Is mean square error (MSE) the optimal selection when modeling a simple linear regression model?**

Ans: When modeling a simple liner regression model the mean square error is one of the most optimal selections available.

➢ **Why MSE is effective for resolving most linear regression problems?**

- **Convexity:**
  MSE is a convex loss function, which means it has a single global minimum. This property ensures that gradient-based optimization algorithms can efficiently find the optimal parameters of the linear regression model
- **Least Squares Solution:**
  MSE corresponds to finding the parameters (coefficients) of the linear regression model that minimize the sum of the squared differences between the predicted and actual values. This aligns with the "least squares" principle, which is a common approach for fitting linear models.
- **Gaussian Assumption:**
  MSE is justified when the errors (residuals) in the regression model are normally distributed and have constant variance. This assumption is often made in classical linear regression.

➢ **Scenarios where MSE may be inappropriate for data modeling along with alternative loss functions:**

- **Outliers:**
  If your dataset contains outliers (data points significantly deviating from the overall pattern), MSE can be sensitive to them. Alternative loss functions like Huber Loss or Tukey's Biweight Loss are more robust to outliers. Huber loss combines the best properties of MSE and absolute error loss (L1 loss). It is less sensitive to outliers and provides a compromise between the mean absolute error (MAE) and MSE. Huber loss uses a parameter delta (a threshold) to determine when to use the L2 loss (MSE) and when to use the L1 loss (MAE).
- **Heteroscedasticity:**
  When the variance of the errors is not constant across all data points (heteroscedasticity), MSE may not be appropriate. In such cases, a Weighted -MSE or a log-likelihood-based loss like the Gaussian Negative Log-Likelihood (GNLL) can be used. Weighted MSE assign different weights to different data points based on their variance. This gives more importance to data points with lower variance.

4. (15%) In the lecture, we learned that there is a regularization method for linear regression models to boost the model's performance. (p18 in linear_regression.pdf)

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

- (5%) Will the use of the regularization term always enhance the model's performance? Choose one of the following options: "Yes, it will always improve," "No, it will always worsen," or "Not necessarily always better or worse."

  Ans: Not necessarily always better or worse. The type of the data, and the choice of regularization strength all influence whether regularization improves model performance.

- We know that λ is a parameter that should be carefully tuned. Discuss the following situations: (both in 100 words)
  1. (5%) Discuss how the model's performance may be affected when λ is set too small. For example, λ=10^(-100) or λ=0

  ➢ When λ is set too small, the regularization term becomes negligible in the loss function, effectively having little to even no impact. In such cases, the model's performance can be adversely affected by overfitting. Without sufficient regularization, the model is free to fit the training data very closely, potentially capturing noise and outliers. This results in a highly flexible and complex model that may perform well on the training data but poorly on unseen data, leading to poor generalization.

  2. (5%) Discuss how the model's performance may be affected when λ is set too large. For example, λ=1000000 or λ=10^100

  ➢ When λ is set too large, the regularization term dominates the loss function. This strong regularization enforces extreme simplification of the model, causing underfitting. The model becomes overly biased, failing to capture the underlying patterns in the data. Consequently, the model's performance deteriorates both on the training and test data as it lacks the complexity necessary to represent the relationships within the dataset. It results in high bias and low variance, which can lead to suboptimal predictive performance.