Exploring Gradient Descent for Linear Regression

Raghda Al Taei

2024

1 Introduction

In this document, we will explore the application of the Gradient Descent method to optimize a cost function in the context of linear regression. We will investigate various aspects that influence the performance of the algorithm and provide a structured approach to understanding and implementing these techniques.

2 Cost Function for Linear Regression

The cost function for a linear regression problem with two parameters is defined as:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

where: - $h_\theta(x_i)$ is the hypothesis function, - y_i is the actual output, - m is the number of data points.

3 Gradient Descent Algorithm

The algorithm starts with random initial values for the parameters. At each iteration, each parameter is updated according to the formula:

$$\theta_i = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

where: - α is the learning rate.

This process continues until the algorithm converges or a maximum number of iterations is reached.

4 Aspects to Explore

4.1 Estimating the Function with Different Numbers of Parameters

To explore how the number of parameters affects the model, various linear models (e.g., simple linear regression with one parameter versus multiple linear regression with more parameters) can be estimated. The performance of these models can be compared based on the mean squared error (MSE) or other criteria to identify the best fit.

4.2 Impact of Learning Rate and Initial Parameters

The learning rate is a crucial hyperparameter that influences convergence speed and accuracy. A small learning rate might result in slow convergence, while a large learning rate might cause overshooting and divergence. It is essential to experiment with different learning rates and initial parameter values to observe their effects on the optimization process.

4.3 Improving Algorithm Performance Using Momentum

Momentum can be implemented to accelerate convergence by incorporating a fraction of the previous update into the current update. This technique helps the algorithm to navigate through ravines in the error surface more effectively, potentially leading to faster convergence.

$$v_t = \beta v_{t-1} + (1 - \beta)\nabla J(\theta)$$
$$\theta = \theta - \alpha v_t$$

where v_t is the velocity vector, β is the momentum term, and $\nabla J(\theta)$ is the gradient of the cost function.

4.4 Effect of Shuffling Data at Each Iteration

Shuffling the training data at each iteration can reduce correlations between consecutive updates and potentially improve convergence. It can also help mitigate the effects of local minima, as the updates may explore the error surface more diversely.

4.5 Impact of Parameter Updates Using Batch Processing

Testing different batch sizes for updating parameters provides insights into the overall error of the algorithm. Smaller batch sizes can lead to noisier updates, while larger batch sizes can produce smoother gradients, influencing convergence speed and stability.

4.6 Proposing a Method to Prevent Overfitting

Overfitting can be mitigated using several techniques, such as: - **Regularization**: Implementing L1 (Lasso) or L2 (Ridge) regularization to penalize large coefficients. - **Early Stopping**: Monitoring validation loss and stopping training when performance begins to degrade. - **Cross-Validation**: Using techniques like k-fold cross-validation to ensure the model generalizes well to unseen data.

5 Implementation Notes

The implemented code should be flexible enough to work with any dataset, accommodating varying numbers of features and data points. Alongside the code, comprehensive documentation is required, including: - Analysis of results, - Answers to the questions outlined in this document, - A detailed explanation of the approach taken.

6 Conclusion

This structured exploration provides a deeper understanding of the Gradient Descent algorithm in optimizing linear regression models. By experimenting with various parameters and techniques, we can enhance model performance and gain practical insights into effective machine learning practices.