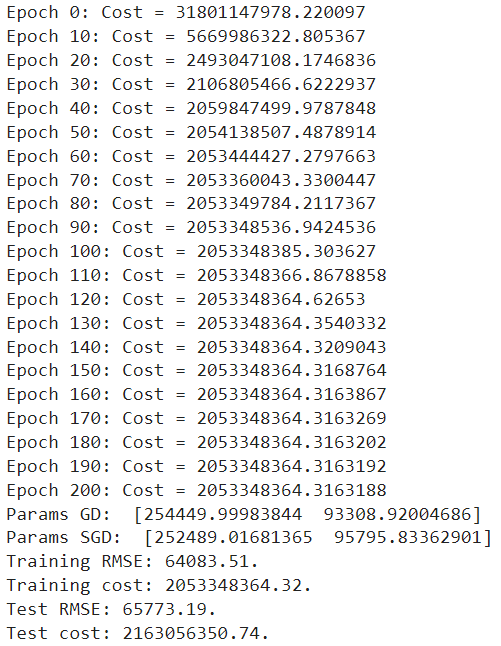
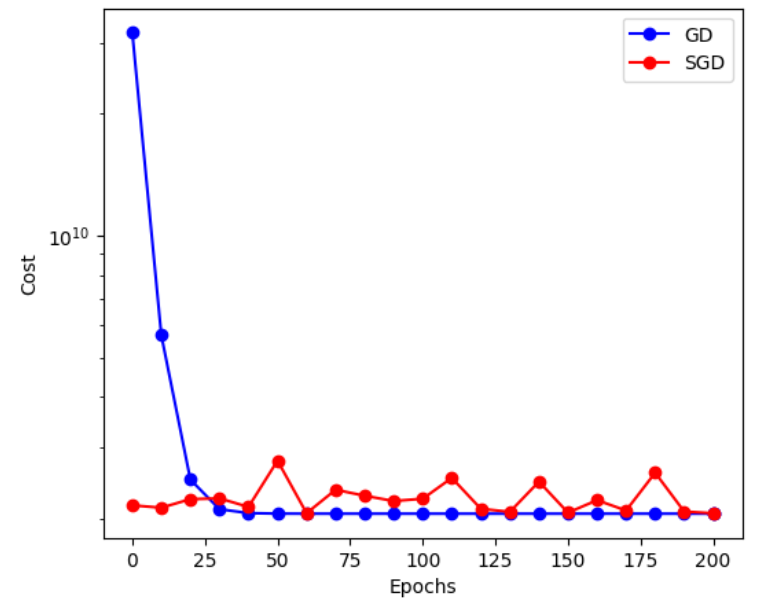
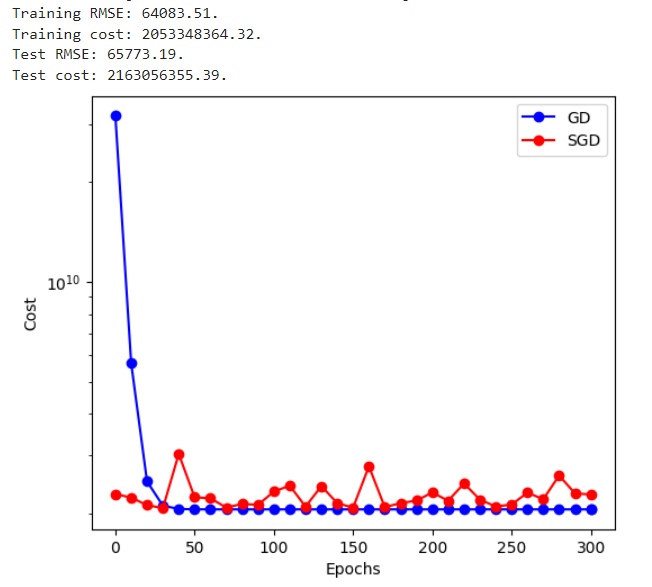
**Results:**





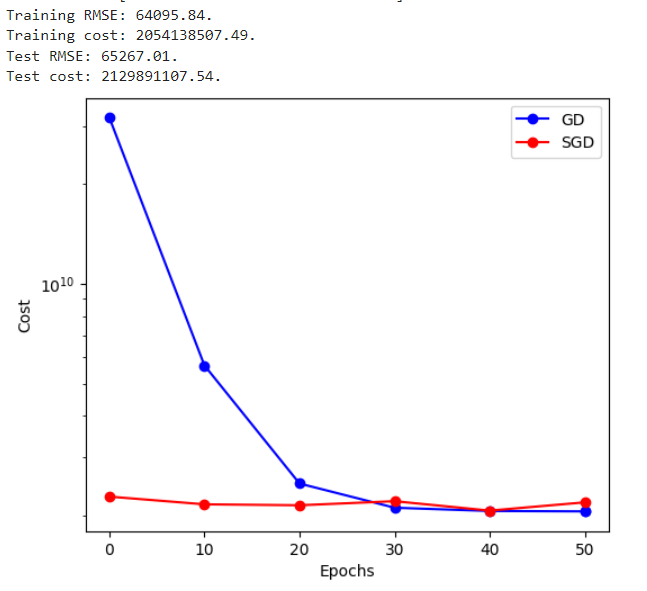
A graph of a training data

Description automatically generated with medium confidence



A graph of a line graph

Description automatically generated with medium confidence



A diagram of a training data

Description automatically generated

With 200 epochs, both SGD and batch GD typically have sufficient iterations to converge to a stable solution for many datasets.

The model parameters usually have enough updates to minimize the cost function and achieve convergence.

For simple linear regression, where the optimization problem is relatively straightforward, 200 epochs often result in good convergence.

Increasing the number of epochs to 300 allows for more iterations, which can improve the convergence and accuracy of the trained model.

With more epochs, the model has additional opportunities to refine its parameters and reduce the cost further.

This can be beneficial for datasets with noisy or complex relationships between features and targets.

With only 50 epochs, the model may not have sufficient iterations to converge effectively, especially for datasets with large variations or noise.

Both SGD and batch GD may struggle to reach a stable solution within a limited number of epochs.

The trained model may underfit the data, leading to suboptimal performance and higher errors, particularly if the dataset is complex or requires more iterations to converge.

In summary, choosing the appropriate number of epochs for training a simple linear regression model depends on factors such as the dataset's complexity, noise level, and desired level of convergence. While fewer epochs may lead to faster training, they may also result in underfitting and poorer performance. Conversely, more epochs can improve convergence but may also increase computational time and the risk of overfitting.

**Discussion:**

The code provided is designed to train a linear regression model on data stored in text files using Google Colab.

Data Reading and Preprocessing: The read\_data function reads the data from text files. It assumes that the data is stored in a specific format where each row represents an example and each column represents a feature, with the last column containing the target variable (house price). The standardize function scales the features by subtracting the mean and dividing by the standard deviation. This preprocessing step is crucial for gradient descent algorithms to converge efficiently.

Training: The train function implements gradient descent to optimize the weights of the linear regression model. The implementation iterates over a specified number of epochs, updating the weights in each iteration based on the computed gradient. It also computes and stores the cost function (mean squared error) every 10 epochs.

Evaluation: After training, the code prints the learned parameters (weights), as well as the RMSE and the objective function values on both the training and test data. This allows for an evaluation of the model's performance on both the training and unseen test data.

Visualization: The code includes plotting functionality to visualize the cost function versus the number of epochs. This helps in understanding the convergence of the gradient descent algorithm. It also plots the linear approximation along with the training and test data, providing a visual representation of the model's performance.

Implementation Problems: One potential implementation problem is related to data formatting. If the data is not in the expected format (e.g., incorrect number of columns, missing values), it may lead to errors during data reading and preprocessing. Another issue could arise if the data contains outliers or extreme values, which may adversely affect the performance of the model and the convergence of the gradient descent algorithm. Outlier detection and handling may be necessary. Additionally, the code assumes that the data fits into memory, which may not be the case for large datasets. For such scenarios, batch processing or distributed computing techniques may be required.

Overall, while the code provides a solid foundation for training a linear regression model on text data using Google Colab, it's essential to ensure that the data is properly formatted and preprocessed to avoid potential issues during training and evaluation. Additionally, addressing scalability concerns for large datasets is important for real-world applications.