Mohan Srinivas Kanarapu

811434915

mk84800@uga.edu

02/10/2025

Machine Learning Homework - 1

1. Comparison of Closed-Form Solution and Gradient Descent

Closed-Form Solution:

- The closed-form solution directly calculates the optimal parameters 'w' using the formula: $w = (X^T X)^{-1} X^T y$
- It's efficient for small datasets because it provides an exact solution in one step. As seen in the plot, the predicted prices closely align with the actual prices, indicating a good fit.

Gradient Descent Solution:

- Gradient descent iteratively updates Θ using the gradient of the cost function: $\Theta = \Theta - \alpha \nabla J(\Theta)$
- where α is the learning rate. It requires tuning hyperparameters (learning rate, number of iterations) and may take multiple iterations to converge. The plot shows similar alignment between actual and predicted prices, though the method's effectiveness depends on appropriate tuning.

2. Differences in Parameters

Parameter Values:

- The closed-form solution produces a unique set of parameters because it solves the equation exactly.
- Gradient descent approximates the parameters over iterations and may yield slightly different values depending on the number of iterations and the learning rate.

Interpretation:

- If the gradient descent method converges properly, the parameter values should be close to those obtained via the closed-form solution.
- Minor discrepancies could arise due to numerical precision or insufficient convergence.

3. Computational Efficiency and Preference Scenarios

Time Complexity:

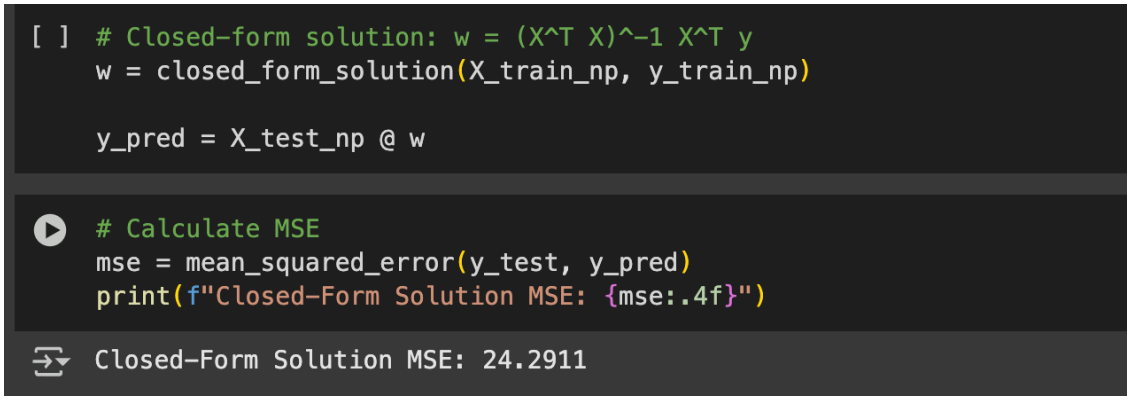
- **Closed-Form Solution**:
  The time complexity is $O(n^3)$ due to the matrix inversion step $(X^T X)^{-1}$. It is efficient for small to moderately sized datasets but becomes impractical for large datasets.
- **Gradient Descent**:
  The time complexity is $O(n \cdot m \cdot k)$, where n is the number of features, m is the number of data points, and k is the number of iterations. It is scalable for large datasets, but the convergence rate depends on hyperparameters.

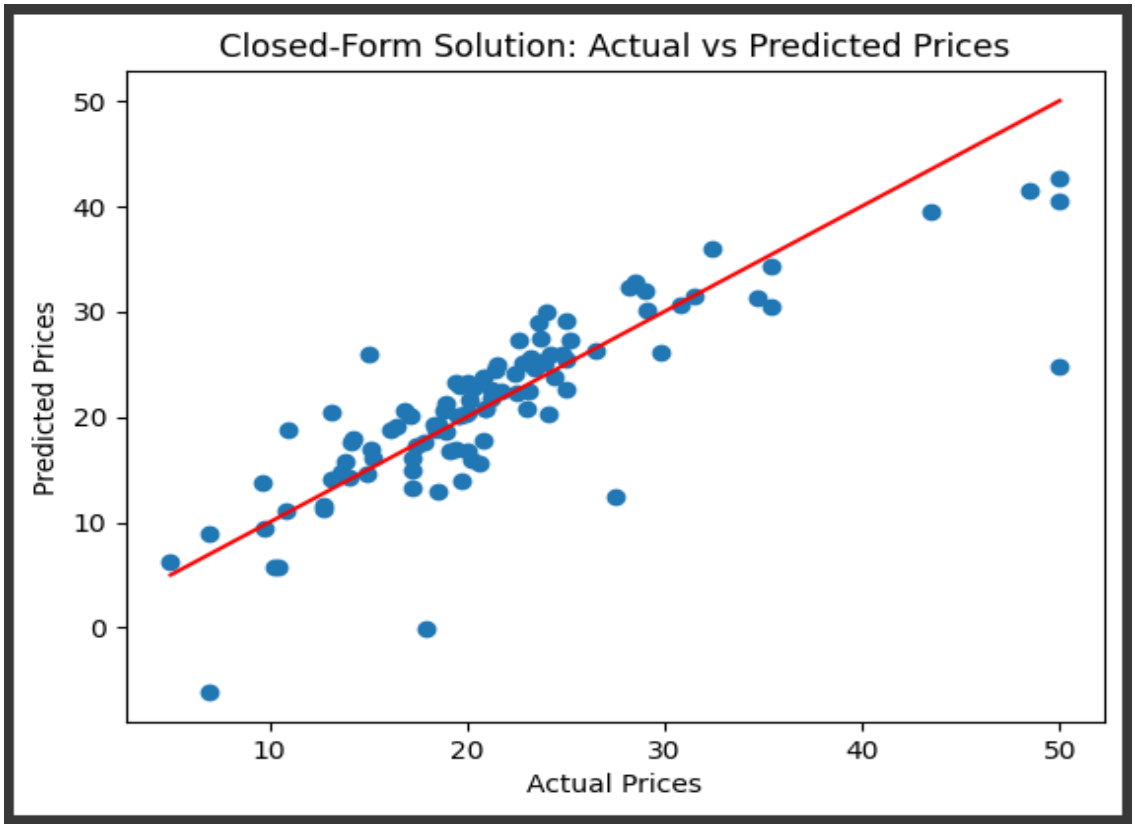
Situational Preference:

- Use the **closed-form solution** for small datasets where computational resources allow for matrix inversion, opt for **gradient descent** for large datasets, datasets with many features, or when distributed computation is required.

Screenshots of Output plots:

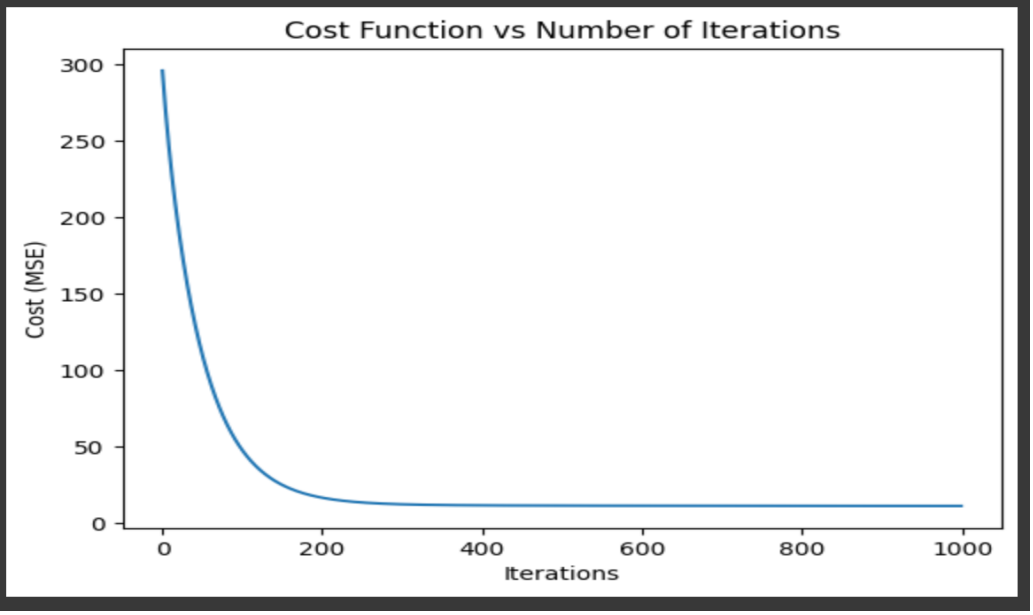
Closed Form Solution:





Gradient Descent Solution:

```
# Plot the cost function vs number of iterations
plt.plot(range(num_iters), cost_history)
plt.xlabel("Iterations")
plt.ylabel("Cost (MSE)")
plt.title("Cost Function vs Number of Iterations")
plt.show()
```



```
[25] # Evaluate the model
     y_pred = X_test_scaled.dot(theta)
     mse = mean_squared_error(y_test, y_pred)
     r2 = r2_score(y_test, y_pred)
```

```
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"R-squared Score (R²): {r2:.4f}")
```

```
Mean Squared Error (MSE): 25.3497
R-squared Score (R²): 0.6543
```