Machine Learning SPRING 2023-2024

SUBMITTED BY:

NAME: SUMIYA HUR TASNIM

ID: 21-44851-2

SECTION: B

SUPERVISED BY Taiman Arham Siddique Faculty

Submission Date: May 03, 2024

Bonus:

Simple Linear Regression:

```
def train_SGD(X, t, eta, epochs):
    costs = []
    ep = []
    w = np.zeros(X.shape[1])
    for epoch in range(epochs):
        for i in range(len(t)):
            idx = np.random.randint(len(t))
            X_i = X[idx:idx+1]
            t_i = t[idx:idx+1]
            grad = compute_gradient(X_i, t_i, w)
            w -= eta * grad
        if epoch % 10 == 0:
            cost = compute_cost(X, t, w)
            costs.append(cost)
            ep.append(epoch)
    return w, ep, costs
```

Multiple Linear Regression :

```
def train_SGD(X, t, eta, epochs):
    w = np.zeros(X.shape[1])
    costs = []
    ep = []
    for epoch in range(epochs):
        for i in range(len(t)):
            idx = np.random.randint(len(t))
            X_i = X[idx:idx+1]
            t_i = t[idx:idx+1]
```

```
            grad = compute_gradient(X_i, t_i, w)
            w -= eta * grad
        if epoch % 10 == 0:
            cost = compute_cost(X, t, w)
            costs.append(cost)
            ep.append(epoch)
    return w, ep, costs
```

Output:

Simple Linear Regression :

Params GD: [254449.99982048 93308.92004027]

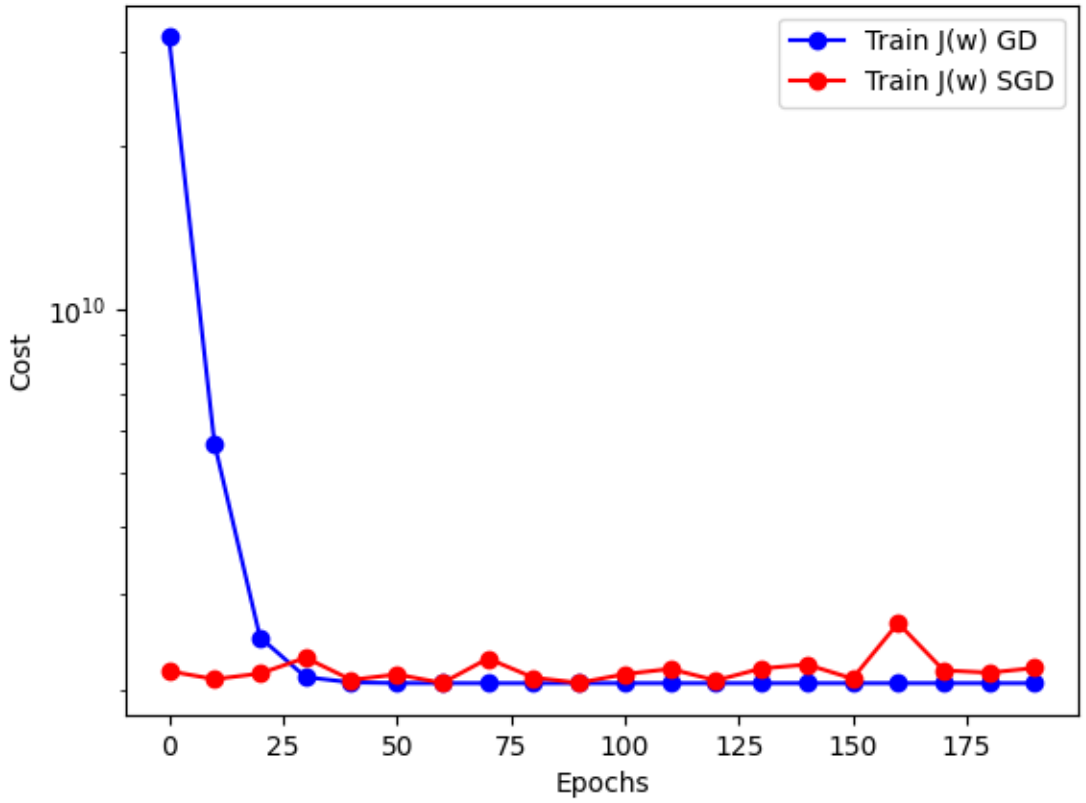
Params SGD: [265321.88322747 76430.73920718]

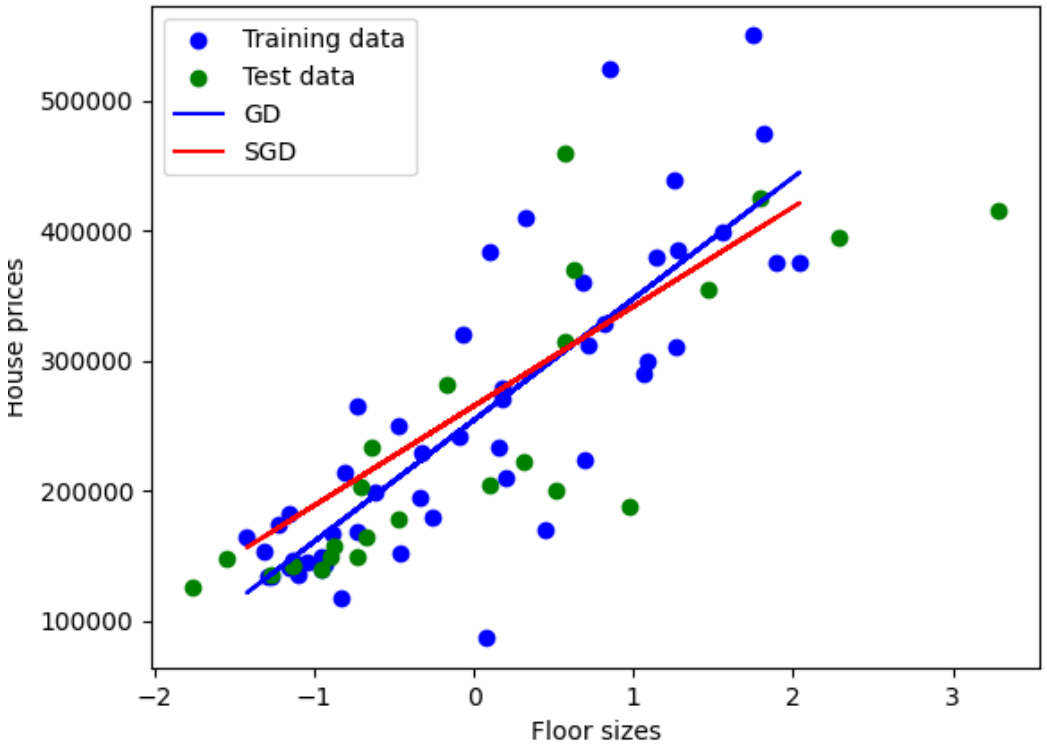
Training RMSE: 64083.51.

Training cost: 2053348364.32.

Test RMSE: 65773.19.

Test cost: 2163056350.22.





Multiple Linear Regression :

Params GD: [254449.99982048 78079.18106675 24442.5758378 2075.95636731]

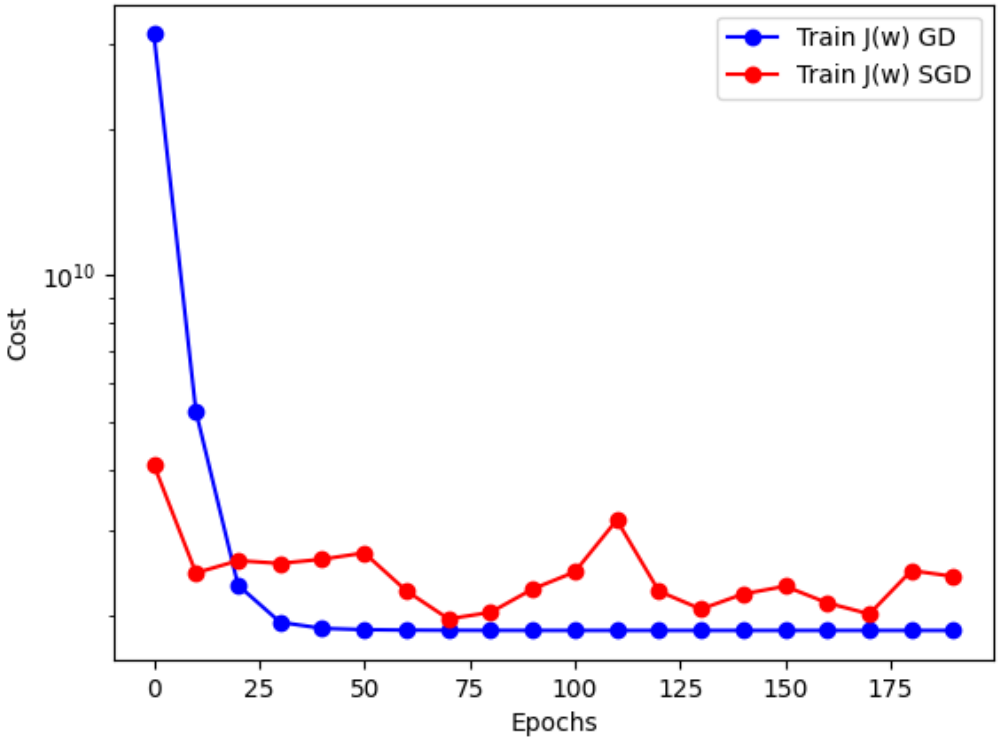
Params SGD: [261649.83652769 108448.35192255 46277.77510326 14796.21911656]

Training RMSE: 61070.62.

Training cost: 1864810304.94.

Test RMSE: 58473.59.

Test cost: 1709580288.69.



Results and Analysis:

Simple Linear Regression :

Results

Model Parameters (w):

The script outputs the final weights for both gradient descent methods. These weights, including the bias term, quantify the relationship between the features (after standardization) and the target variable.

Performance Metrics:

Training RMSE: Measures the model's accuracy on the training data, indicating how well the model fits this data.

Test RMSE: Evaluates the model's accuracy on the test data, providing insight into its generalization ability.

Costs: Displayed for both training methods over epochs, showing the mean squared error's decline, which helps assess model convergence.

Analysis

1. **Training Dynamics:**
   - ❑ The script visualizes the reduction in cost across training epochs for both GD and SGD. Batch gradient descent should show a steady decrease, while stochastic gradient descent may exhibit a more fluctuating descent due to its iterative updates from individual samples.
2. **Comparison between GD and SGD:**
   - ❑ Observing the difference in performance and cost trajectories between the two methods can highlight their respective efficiencies and suitability for the dataset. GD is expected to be more stable, while SGD might converge faster but with more variance in the learning path.

Graphical Visualization:

The script generates two plots:

- ❑ **Epochs vs. Cost:** Helps visualize and compare the learning curves of GD and SGD, using a logarithmic scale to detail changes across a broad range of values.
- ❑ **Feature vs. Target with Fitted Lines:** Displays how the model approximates the relationship between the feature (e.g., floor size) and the target (e.g., house prices) for both training and testing datasets. Lines representing predictions from both GD and SGD are plotted to show how closely they fit the actual data points.

Multiple Linear Regression :

Results

1) Model Parameters:
   - ❑ The weights (w) obtained reveal the impact of each feature on the target, indicating how attributes like house size or age influence the predicted prices.
2) Performance Metrics:
   - ❑ Training RMSE and Test RMSE gauge the model's accuracy on both training and unseen data, highlighting its effectiveness and potential overfitting.

Costs calculated during training show how well the model minimizes errors over epochs.

Analysis

- ❑ **Convergence:** The cost reduction plots for both batch and stochastic gradient descent indicate the training efficiency and whether the model is learning as expected. Batch

gradient descent typically converges smoothly, while stochastic can be quicker but more erratic.

- **Method Comparison:** Comparing the smoothness and rate of convergence between batch and stochastic gradient descent helps in selecting the method that best suits the dataset and training needs, balancing accuracy, and computation time.

Decision:

1. Feature Scaling:

I. Feature scaling is important for gradient descent because it helps to ensure that all features have similar ranges, which aids in faster convergence.

II. To implement standard scaling, subtract the mean of each feature from its values and then divide by the standard deviation of the feature.

2. Simple Regression: Training a simple linear regression model involves:

(I) Reading the data features and labels.

(II) Implementing gradient descent to update the parameters iteratively.

(III) Computing and storing the cost function periodically.

(IV) Printing the parameters, RMSE, and objective function values.

(V) Plotting the training and test examples along with the linear approximation.

(VI) Plotting J(w) vs. the number of epochs at intervals of 10 epochs.

3. Multiple Regression:

I) Additional steps for multiple regression include:

II) Handling multiple features instead of just one.

III) Adjusting the gradient computation and parameter updates accordingly.

IV) After training, the same parameters, RMSE, and objective function values are printed as in simple regression.

V) To compare test RMSE with the simple regression case, simply calculate the RMSE for both models and compare them.

4. BONUS: Stochastic Gradient Descent (SGD):

I) SGD is a variation of gradient descent where the parameters are updated using a single training example at a time instead of the entire dataset.

II) Implementing SGD involves modifying the gradient descent algorithm to update parameters using randomly selected training examples.

III) To compare SGD with batch GD, run both algorithms with the same learning rate and number of epochs, then compare the resulting parameters, RMSE, and convergence behavior.