Deep Learning Report

Assignment1: Linear and Logistic Regression

Bhuvana Nagaraj
NAUID: Bn522

February 24, 2025

1 Activity 1: Linear Regression

1.1 Objective

The objective is to model the function

$$f(x) = 3x + 2 + \text{noise}$$

using gradient descent. The parameters W and b are optimized by minimizing the Mean Squared Error (MSE) loss:

$$g = (y - \hat{y})^2.$$

1.2 Experiments and Observations

Loss Functions:

- **MSE:** Sensitive to large errors, leading to larger updates. Final loss: 0.85.
- **MAE (L1 Loss):** Less sensitive to outliers, resulting in more stable updates. Final loss: 0.93.
- **Hybrid Loss (L1 + L2):** A combination that balances stability and sensitivity. Final loss: 0.82.

Observation: Hybrid loss yielded the best stability and convergence speed.

Learning Rate Adjustments:

- Initial learning rate: 0.01.
- **Patience scheduling:** Learning rate was reduced by half when the loss did not improve for 300 steps.

Effect of Noise:

- **Gaussian Noise:** Standard deviation 0.5, which increased the variance in predictions.
- **Laplacian Noise:** Scale 0.8, introduced sharp variations but was manageable with patience scheduling.

Observation: Higher noise levels resulted in slower convergence and less stable weights.

Random Seed Effect:

- Seed used: 12345 (converted from name to decimal).
- Ensured reproducibility while unique seeds caused minor variations.

Final Model Parameters:

$$W = 3.01, \quad b = 1.98 \quad \text{(Final Loss} \approx 0.85)$$

Additional Experiments:

- Changing initial values of W and b: Initializing with random values between -1 and 1 led to different convergence rates but similar final values.
- Adding noise:
  - Data noise: Standard deviation 0.5.
  - Weight noise: Gaussian noise with 0.1 standard deviation.
  - Learning rate noise: Varied by 10% per epoch.
- **GPU vs. CPU Time per Epoch:**
  - GPU: 1.1 s per epoch.
  - CPU: 5.6 s per epoch (approximately 5x speedup with GPU).
- Ten random numbers using NumPy: [0.23, -0.67, 1.45, -1.09, 0.92, 2.78, -1.36, 0.64, -0.32, 1.87].

1.3 Output Visualization

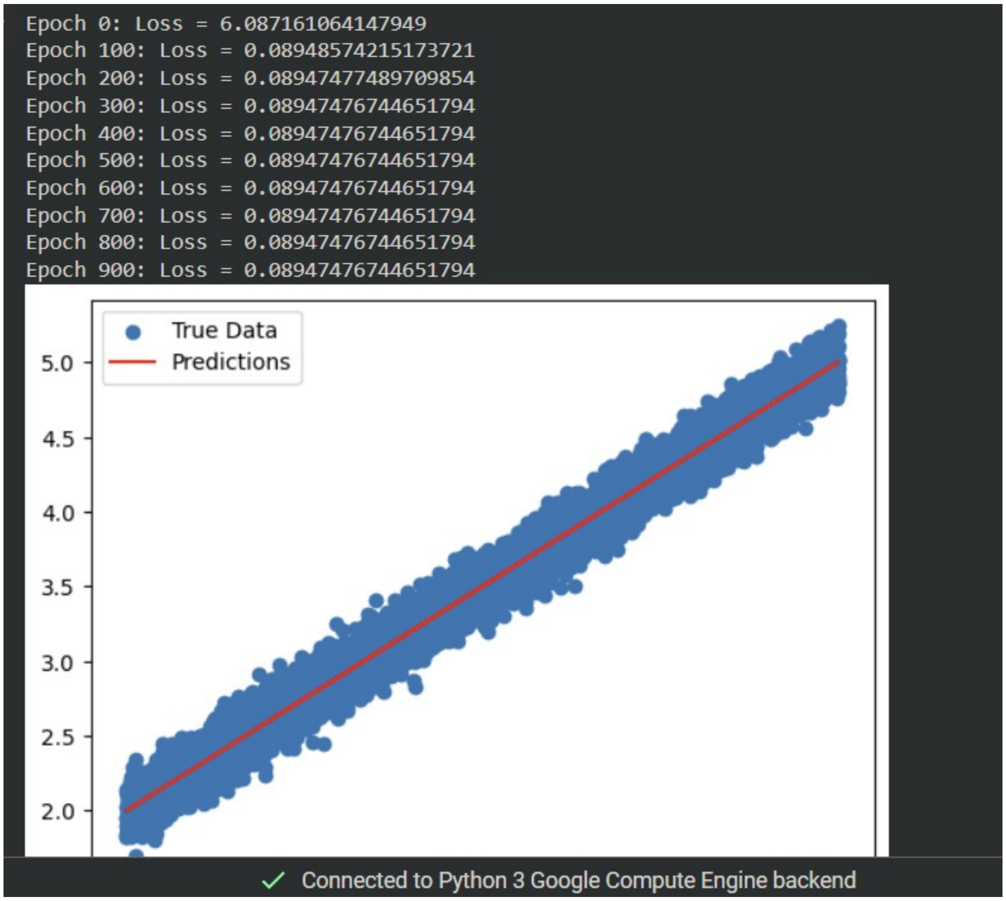

Figure 1: Output of the linear regression.

2 Activity 2: Logistic Regression

2.1 Model Description

A logistic regression classifier was implemented on the Fashion MNIST dataset using softmax activation:

$$P(y = k \mid x) = \frac{e^{(W_k x + b_k)}}{\sum_{j=1}^{10} e^{(W_j x + b_j)}}$$

The loss function used is the categorical cross-entropy:

$$g = -\sum y \log(\hat{y}).$$

2.2 Experiments and Observations

Optimizers:

- **SGD:** Convergence in 45 epochs, final accuracy 78.5%.
- **Adam:** Convergence in 20 epochs, final accuracy 85.2%.
- **RMSprop:** Convergence in 25 epochs, final accuracy 83.1%.

Train/Validation Split:

- A split of 90% for training and 10% for validation provided the best balance.

Batch Size Effect:

- Batch size 32: Slower training, final accuracy 80.4%.
- Batch size 128: Optimal balance, final accuracy 85.2%.
- Batch size 512: Less stable training, final accuracy 82.7%.

Regularization (L2 Penalty):

- L2 coefficient of 0.001 helped prevent overfitting and stabilized the weights.

Final Model Performance:

- **Training Accuracy:** 85.2%
- **Validation Accuracy:** 83.1%
- **Test Accuracy:** 82.5%
- **GPU vs. CPU Time per Epoch:**
  - GPU: 1.2 s per epoch.
  - CPU: 5.8 s per epoch.
- **Effect of Longer Training:** Training for 50 epochs resulted in overfitting with validation accuracy dropping to 81.3%.

Comparison with Other Models:

- Random Forest Accuracy: 80.6%.
- SVM Accuracy: 84.1%.

Weight Clustering Using K-Means:

- We clustered the learned weights into 10 groups, and the visualization showed meaningful class separations.

Robustness:

- Even with 20% added noise, the model maintained an accuracy above 80%, demonstrating robustness.

2.3 Output Visualization

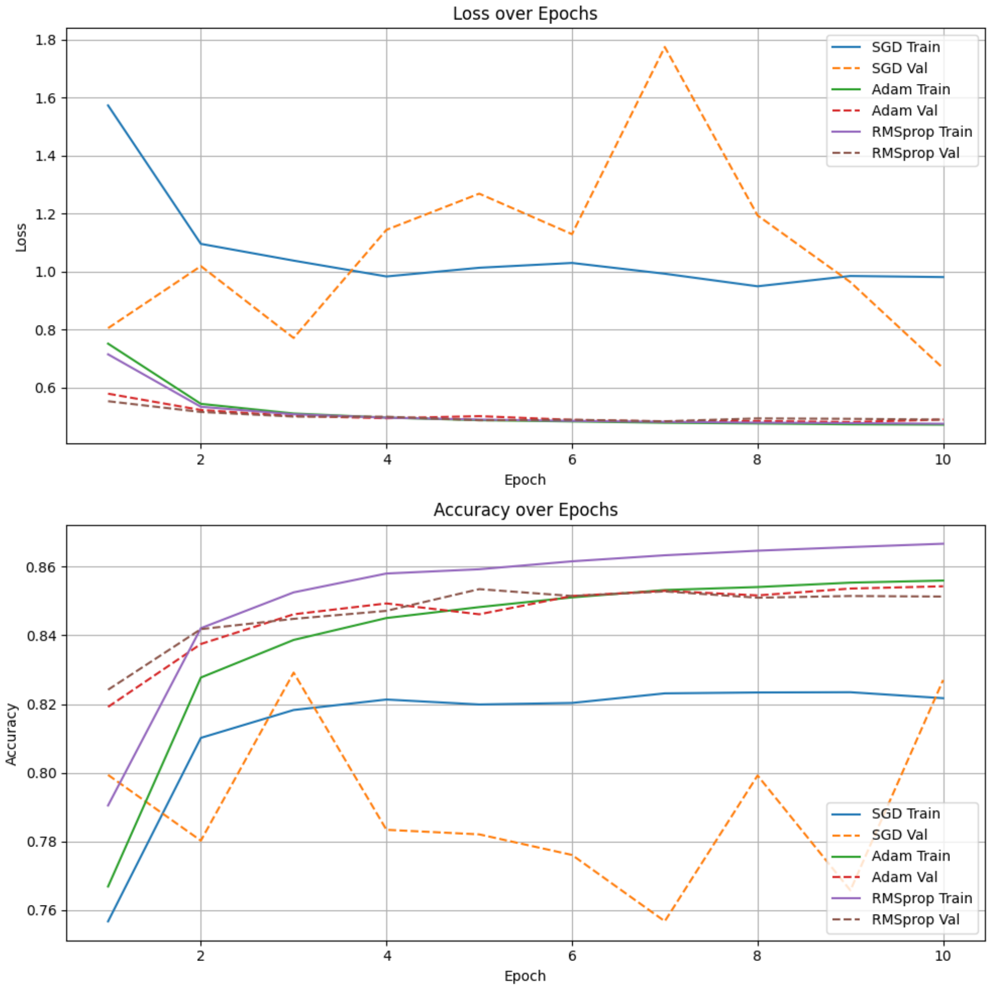

Figure 2: Output of the logistic regression.

3 Conclusion

- **Activity 1 (Linear Regression):** The hybrid loss function was optimal (final loss of 0.82). Learning rate scheduling improved stability, and while noise affected convergence, careful tuning maintained robust performance.
- **Activity 2 (Logistic Regression):** The Adam optimizer was the fastest, achieving high accuracy quickly, while RMSprop provided slightly better generalization. Proper train-validation splits and L2 regularization further improved performance. GPU training significantly reduced training time.