Assignment 2 Report

Artificial Intelligence

Task 1: Optimizer Performance on Non-Convex Functions

1. Objective

The objective of Task 1 is to study and compare the performance of different gradient-based optimization algorithms on non-convex functions. Non-convex optimization problems are challenging due to the presence of local minima, saddle points, steep curvature, and oscillatory regions. The focus of this task is to analyze convergence behavior, stability, and sensitivity to learning rates.

2. Functions Considered

(a) Rosenbrock Function

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

The Rosenbrock function is a classic non-convex benchmark function with a narrow curved valley leading to a global minimum at $(1, 1)$. While the minimum is easy to identify, the curved valley makes convergence difficult for simple gradient-based methods.

(b) Oscillatory Function

$$f(x) = \sin\left(\frac{1}{x}\right), \quad f(0) = 0$$

This function exhibits infinite oscillations near $x = 0$, resulting in unstable gradients and making optimization extremely challenging.

3. Optimizers Implemented

All optimizers were implemented from scratch using Python:

- Gradient Descent (GD)
- Gradient Descent with Momentum
- Adagrad
- RMSProp

- Adam

Learning rates considered were:

$$\alpha \in \{0.01, 0.05, 0.1\}$$

4. Stopping Criterion

The optimization process was terminated when the norm of the gradient satisfied:

$$\|\nabla f\| < 10^{-6}$$

or when a maximum number of iterations was reached.

5. Observations

Rosenbrock Function:

- Gradient Descent and Momentum often diverged due to steep curvature and exploding gradients.
- Adagrad converged early but frequently stagnated because of aggressive learning rate decay.
- RMSProp showed stable convergence but was sensitive to learning rate choice.
- Adam consistently converged fastest and reached values closest to the global minimum.

sin(1/x) Function:

- GD and Momentum failed to converge due to persistent oscillations.
- Adagrad prematurely stopped learning.
- RMSProp exhibited instability for large learning rates.
- Adam demonstrated the most stable behavior and fastest numerical convergence.

6. Conclusion

Adaptive optimization methods such as Adam and RMSProp are significantly more robust on non-convex functions due to adaptive learning rates and gradient smoothing, whereas vanilla gradient descent is highly sensitive to learning rate and curvature.

Task 2: Linear Regression Using a Multi-Layer Neural Network

1. Objective

The objective of Task 2 is to implement linear regression using a multi-layer neural network from scratch and to study the effect of optimizers, learning rates, network depth, and regularization on model performance.

2. Dataset and Preprocessing

The Boston Housing dataset was used for this task. The goal is to predict the median value of homes (MEDV) using two features:

- RM: Average number of rooms per dwelling
- CRIM: Per-capita crime rate

Preprocessing steps included:

- Feature normalization to zero mean and unit variance
- Target normalization for numerical stability
- Train-test split of 80% training and 20% testing

3. Neural Network Architecture

Base Architecture:

$$2 \to 5 \to 3 \to 1$$

Bonus Architecture:

$$2 \to 5 \to 3 \to 2 \to 1$$

ReLU activation was used in all hidden layers, while the output layer employed a linear activation function suitable for regression.

4. Forward Propagation

For each layer l, the forward pass is given by:

$$Z^{(l)} = A^{(l-1)}W^{(l)} + b^{(l)}$$

$$A^{(l)} = \text{ReLU}(Z^{(l)})$$

The output layer computes:

$$\hat{y} = Z^{(L)}$$

5. Loss Function

Mean Squared Error (MSE):

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

L2 Regularization (Bonus):

$$\mathcal{L} = \mathcal{L}_{MSE} + \lambda \sum W^2$$

L2 regularization penalizes large weights and reduces overfitting.

6. Backpropagation

Backpropagation computes gradients using the chain rule. The error at the output layer is:

$$\delta^{(L)} = \frac{\hat{y} - y}{N}$$

For hidden layers:

$$\delta^{(l)} = (\delta^{(l+1)} W^{(l+1)^T}) \odot \text{ReLU}'(Z^{(l)})$$

Gradients of weights and biases are then used by the optimizer to update parameters.

7. Optimizers Compared

The following optimizers were implemented from scratch:

- Gradient Descent
- Momentum
- Adam

Adam consistently demonstrated faster and more stable convergence.

8. Experiments and Results

- Adam converged faster than GD and Momentum.
- Lower learning rates were more stable but converged slower.
- The deeper architecture did not consistently outperform the simpler model.
- L2 regularization improved generalization by reducing overfitting.

9. Evaluation

Model performance was evaluated using Mean Squared Error on the test set. Predicted vs actual plots showed good alignment along the diagonal, indicating effective regression performance.

10. Final Conclusion

Increasing model complexity does not guarantee improved performance, particularly for small datasets. Adaptive optimizers and regularization play a crucial role in achieving stable and generalizable learning. All neural networks and optimizers were implemented entirely from scratch.