Visualization of Gradient Descent

Problem Statement:

visualizing and comparing the optimization trajectories of Gradient Descent (GD) in two distinct machine learning models logistic regression (convex) and a two-layer neural network (non-convex) during training on a binary classification task. Highlight the differences in convergence behavior between convex and non-convex optimization landscapes using dimensionality reduction technique.

Methodology:

1. Dataset Generation : A binary classification dataset is created with a total of 200 samples and each sample is characterised by 5 input features drawn from a gaussian distribution. A randomly initialised weight vector computes linear scores for each sample which are then converted into probabilities through the logistic sigmoid function. The samples that exceed the probability 0.5 are labeled as class 1 and others are labeled as class 0. This ensures linear separation of dataset but adds realistic probabilistic noise, making it suitable for evaluating both linear and nonlinear models.

2. Model Training using GD : A linear classifier is added with a bias term. The model predicts class probabilities using sigmoid function and then minimizes cross entropy loss. The cross entropy loss quantifies the contrast between the predicted probabilities and true labels. It penalizes confident incorrect predictions more heavily which ensures the model learns to align probabilities with ground truth labels. The gradient of the loss with respect to the weights is computed. This reflects how small changes in the weight will affect the loss. At each iteration the weights are updated by moving in the direction opposite to the gradient which is scaled by a learning rate (0.5). This process gradually minimizes the loss. The loss function for logistic regression is convex which guarantees a single global minimum. The GD follows a smooth and direct path towards this minimum and gradients consistently pointing downhill.

Another model, a two layer Neural Network is also used which is the non-linear counterpart to logistic regression. It accepts 5 - Dimensional feature vectors from the dataset. It consists of a hidden layer which consists of 5 units with tanh activation to introduce nonlinearity along with a sigmoid output layer. The input features are transformed by hidden-layer weights and biases into activations which then flow through the output layer to become probability scores. The model aims to minimize cross-entropy loss just like logistic regression but it relies on backpropagation to calculate gradients for both ‘input-to-hidden’ and ‘hidden-to-output’ weights and biases by moving errors backward from the output.

3. Dimensionality Reduction for Visualization: Two techniques are separately applied to interpret the high dimensional parameter trajectories. The Principal Component Analysis

reduces the parameter space to two principal components which captures the directions of maximum variance. This allows plotting the loss surface in a 2D subspace, where the global geometry of the optimization landscape becomes visible. The t-SNE (t-Distributed Stochastic Neighbor Embedding) is also used which projects the trajectories into a 2D space while preserving local geometric relationships. Unlike PCA, t-SNE emphasizes clusters and nonlinear patterns, providing insights into the parameter evolution.

4. Visualisation: Figure 1 and 2 depicts the optimization trajectories of logistic regression and the neural network after 50 iterations of gradient descent.

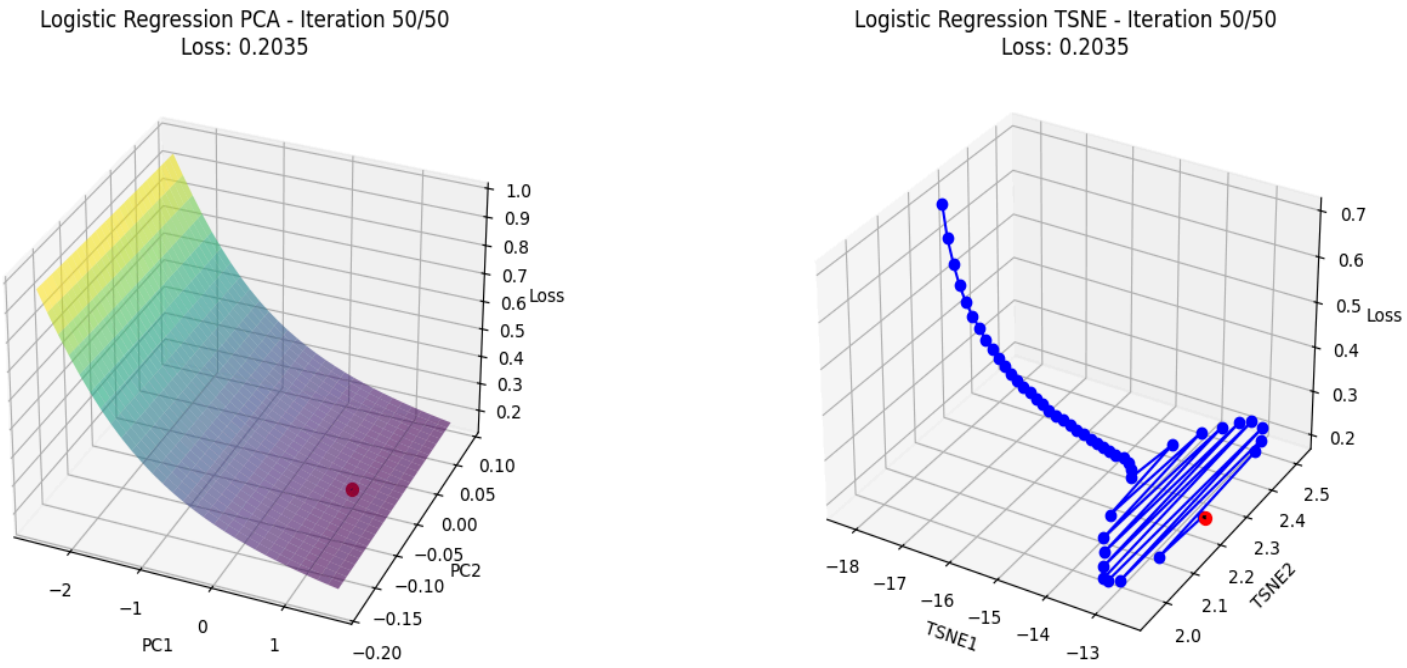

Figure 1

The PCA subplot shows a smooth descent along a gently sloping loss surface with coordinates clustered near stable values like -0.20 on the z-axis, reflecting its convex convergence. The t-SNE subplots add depth, logistic regression's trajectory follows a tight linear band

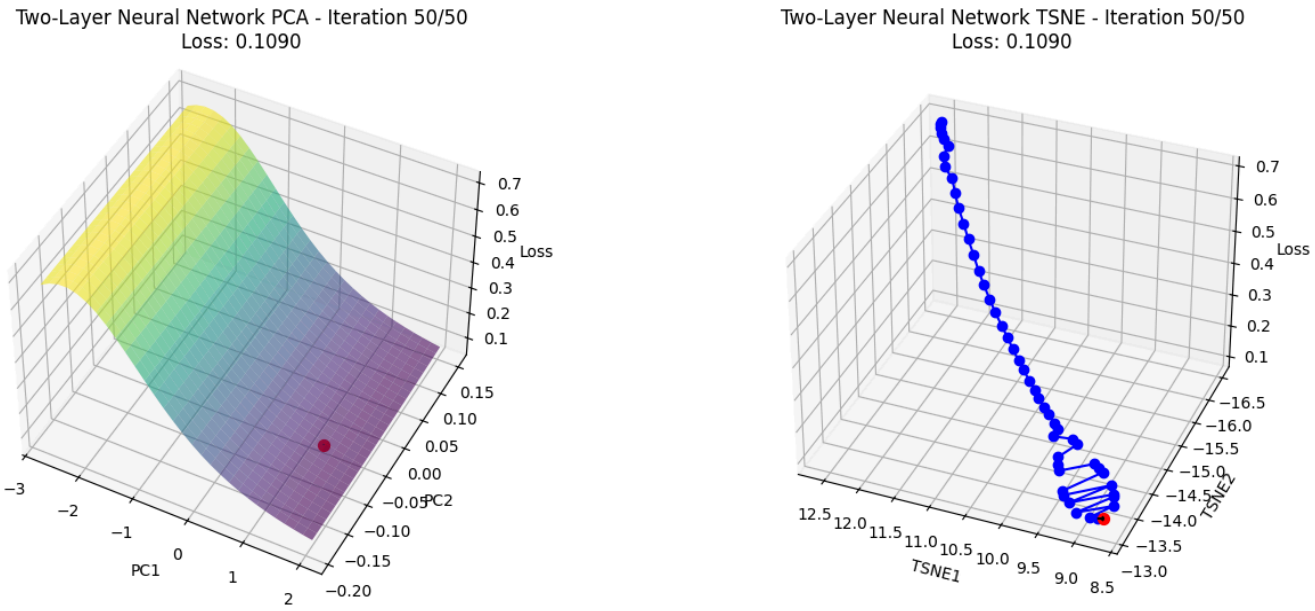

Figure 2

Figure 2 shows values with no coherent directional trend, indicating unpredictable parameter shifts across iterations. These erratic jumps stem from the NN navigating saddle points, plateaus, or local minima in its non-convex loss landscape.

Conclusion:

This project demonstrates a fundamental trade-off between model simplicity and optimization complexity in machine learning. Logistic regression, which has a convex loss landscape, achieves stable and predictable convergence which is guided by smooth gradients toward a global minimum. Its minor oscillations near convergence reflect fine-tuning rather than instability. The two-layer neural network attains a lower final loss but navigates a non-convex terrain fraught with erratic parameter shifts that illustrates the challenges of escaping saddle points and local minima.