Introduction to Machine Learning
Lab Session 4

Group 39
Laura Lall (s6221858) & Rachel Yang (s6213154)

March 12, 2025

Introduction

Vector Quantization (VQ) is a classical unsupervised learning technique used for data compression and clustering. The goal of VQ is to represent a set of data points using a smaller number of prototypes, which are learned through a competitive learning process. In this lab, we implemented the Winner-Takes-All unsupervised competitive learning algorithm and applied it to a dataset of 1000 two-dimensional data points. The algorithm iteratively updates the positions of the prototypes to minimize the quantization error, which measures the discrepancy between the data points and their closest prototypes. In this report, we outline the methodology, details of the implementation, and results of our experiments with different numbers of prototypes and learning rates. We also discuss the impact of the learning rate on the convergence of the algorithm and the final quantization error. The workload was divided by Laura implementing most of the code and Rachel writing most of the report, with both parties assisting each other where needed.

Methods

Initialization: The prototypes are initialized by randomly selecting K data points from the dataset.

Training:

- For each epoch, the data points are presented in a randomized order.
- For each data point, the Euclidean distance is computed to all prototypes.
- The prototype closest to the data point (the "winner") is updated using the learning rate.
- This process is repeated for all data points in the dataset, and the quantization error HvQ is computed after each epoch.

Stopping Condition The training stops after a fixed number of epochs tmax.

Implementation:

- The number of prototypes K was set to 2 and 4.
- The learning rate η was initially set to 0.1, but smaller values (e.g., 0.01, 0.05) were also tested.
- The maximum number of epochs tmax was chosen based on the convergence of the quantization error HvQ.

Choice of parameters:

- Learning Rate (η): A learning rate of 0.1 was chosen as an initial estimate, but smaller values were tested to ensure stable convergence. Too large a learning rate can cause the prototypes to overshoot the optimal positions, while too small a learning rate can result in slow convergence.

- Maximum Epochs (tmax): The value of tmax was determined by observing when the quantization error HvQ plateaued. For our experiments, tmax was set to 100 epochs.

Experimental results

Learning curves:

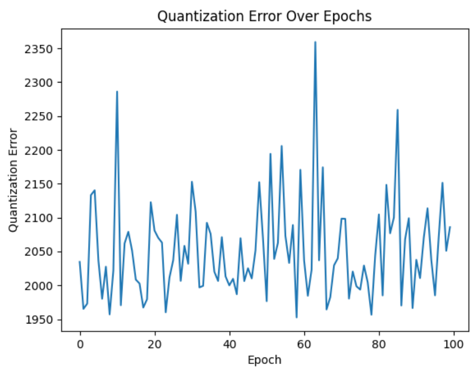

Figure 1: *Learning curve for K=2* $\eta = 0.1$

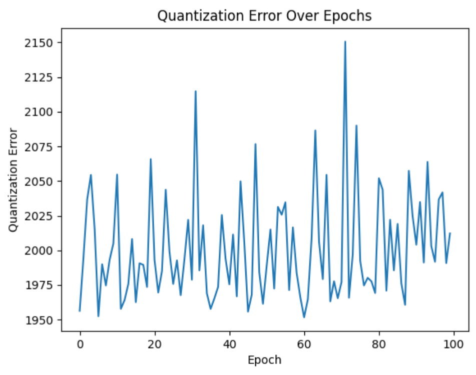

Figure 2: *Learning curve for K=2* $\eta = 0.05$

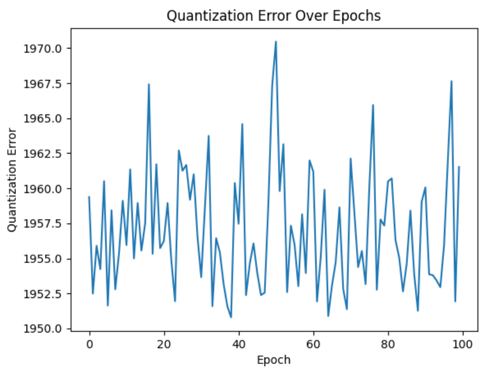

Figure 3: *Learning curve for K=2* $\eta = 0.01$

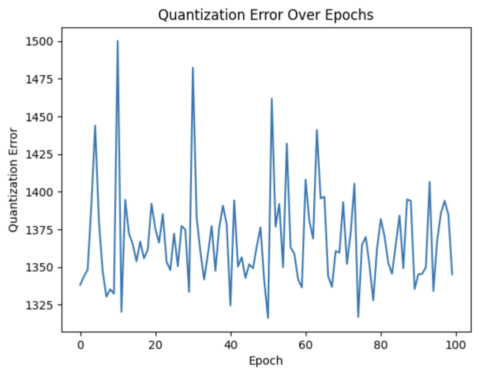

Figure 4: *Learning curve for K=4* $\eta = 0.1$

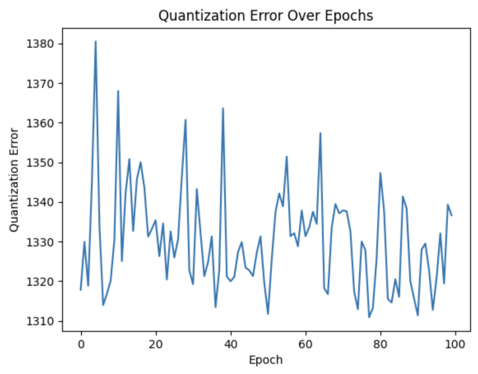

Figure 5: *Learning curve for K=4* $\eta = 0.05$

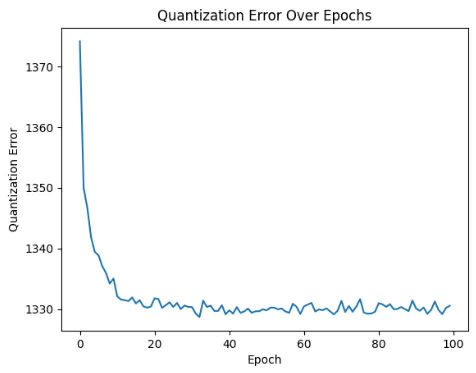

Figure 6: *Learning curve for K=4* $\eta = 0.01$

Trajectories of prototypes:

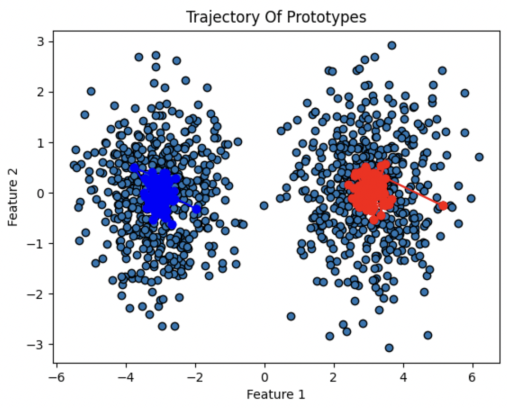

Figure 7: *Trajectory for K=2 with random initialization* $\eta = 0.1$

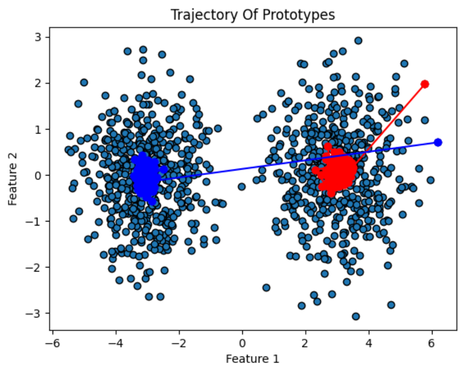

Figure 8: *Trajectory for K=2 with 'stupid' initialization* $\eta = 0.1$

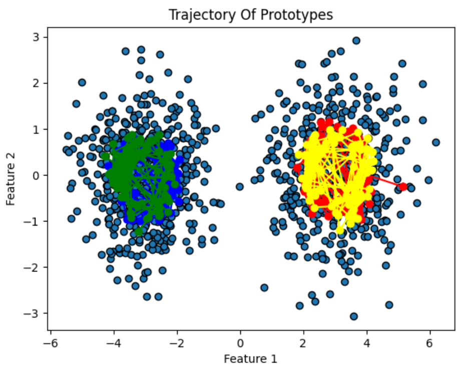

Figure 9: *Trajectory for K=4 with random initialization* $\eta = 0.1$

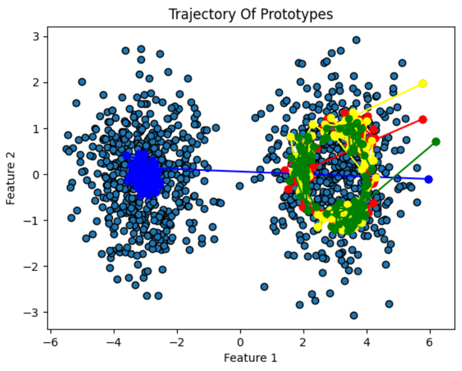

Figure 10: *Trajectory for K=4 with 'stupid' initialization* $\eta = 0.1$

Discussion

Role of the Learning Rate (η) For $\eta = 0.1$, the prototypes converged relative to their final positions relatively quickly, and the quantification error HvQ decreased steadily over the epochs. This learning rate provided a good balance between convergence speed and stability.

For $\eta = 0.05$, the convergence was slower compared to $\eta = 0.1$, but the final quantization error was similar. This suggests that a smaller learning rate can still achieve good results, even if with more training epochs.

For $\eta = 0.01$, the prototypes updated relatively slow, requiring more epochs to converge. Although the final quantization error was comparable to that of $\eta = 0.1$ the training process was inefficient due to the convergence.

Effect of Large and Small Learning Rate: If η was set to a very large value, for example 0.9, the prototypes exhibited unstable behavior. The updates to the prototypes were too large, which caused them to overshoot their optimal positions. This resulted in oscillations in the quantization error and slower convergence. If η was set to a very small value, for instance 0.001, the prototypes updated very slowly, requiring significantly more epochs to converge. A very small learning rate also increased the risk that the algorithm gets stuck in suboptimal local minima, especially for more complex data sets or higher values of initializations.

Learning Curves and Prototype Trajectories: The learning curves for different learning rates showed that: For $\eta = 0.1$, the quantization error decreased rapidly in the initial epochs and then plateaued, indicating convergence. For $\eta = 0.05$ and $\eta = 0.01$, the decrease in quantization error was more gradual, with $\eta = 0.01$ requiring significantly more epochs to reach a similar level of convergence.

The trajectories of the prototypes revealed how they moved from their initial positions to their final positions: For K = 2, the prototypes quickly moved toward the two main clusters in the data. For K=4, the prototypes initially overlapped but gradually separated to cover the four main clusters in the data.