Quantum-inspired machine learning on a quantum computer

João Guilherme

Instituto Superior Técnico, Department of Physics

April 30, 2025

Abstract

This document presents...

I. Introduction

Radial Basis Function (RBF) networks are a class of machine learning models well-suited for interpolation and classification tasks. Their foundation lies in constructing a hypothesis as a weighted sum of radial basis functions—functions whose output depends solely on the distance from a central point. This property allows RBF networks to capture local structures in the data efficiently.

This report details the construction and training of an RBF network applied to image classification, particularly using the MNIST dataset. We describe the mathematical formulation of the model, data encoding strategies including quantum-inspired Gram matrix encodings, and practical training methods including both full and reduced center approaches.

II. Data Encoding

i. Gram Matrix Encoding

We consider a column vector u representing an image sample. To embed this into a higher-order representation, we construct the normalized outer product:

$$\rho = \frac{uu^T}{\mathrm{Tr}(uu^T)}$$

This yields a density-like matrix ρ with trace 1, encoding pairwise correlations between pixels. Such an encoding is inspired by quantum mechanical density operators and preserves structural information in the image.

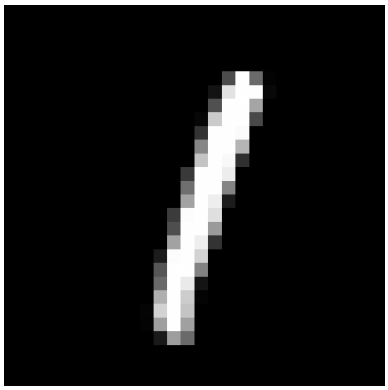

Figure 1: Example MNIST image (digit 1)

III. Radial Basis Function Network

RBF networks model the hypothesis function $h(x)$ using a radial function centered around key points (called *centers*). Each center influences the prediction based on its proximity to the input x. The standard form is:

$$h(x) = \sum_{n=1}^{N} w_n \exp\left(-\gamma \|x - x_n\|^2\right)$$

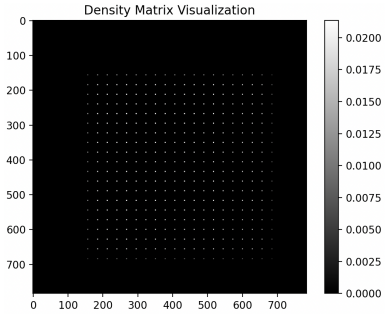

Figure 2: Gram matrix encoding ρ of the digit 1 image

where γ determines the width of the radial functions, and w_n are the learned weights.

i. Exact Interpolation

Given training data $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, we can formulate the interpolation condition:

$$\sum_{m=1}^{N} w_m \exp\left(-\gamma \|x_n - x_m\|^2\right) = y_n, \quad \forall n$$

In matrix notation:

$$\Phi \mathbf{w} = \mathbf{y}, \quad \text{where } \Phi_{n,m} = \exp\left(-\gamma \|x_n - x_m\|^2\right)$$

If Φ is invertible, we can directly solve:

$$\mathbf{w} = \Phi^{-1}\mathbf{y}$$

This yields an exact interpolating function.

ii. Classification

For classification, we apply a decision rule such as:

$$h(x) = \mathrm{sign}\left(\sum_{n=1}^{N} w_n \exp(-\gamma \|x - x_n\|^2)\right)$$

In multi-class settings, this can be extended to a softmax output over class scores.

iii. Training with Least Squares

Instead of enforcing exact interpolation, we may use a least squares approach to minimize:

$$E = \sum_{n=1}^{N} (h(x_n) - y_n)^2$$

This leads to the ridge-regularized solution:

$$\mathbf{w} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{y}$$

where λ is a regularization parameter.

IV. Choosing RBF Centers

Using every data point as a center is computationally expensive. Instead, we reduce the number of centers to $K \ll N$ by clustering the dataset using **K-Means**:

We define:

$$J = \sum_{k=1}^{K} \sum_{x_n \in S_k} \|x_n - \mu_k\|^2$$

and use Lloyd's algorithm to iteratively minimize J:

$$\mu_k = \frac{1}{|S_k|} \sum_{x_n \in S_k} x_n$$

$$S_k = \{x_n : \|x_n - \mu_k\| \leq \|x_n - \mu_j\|,\ \forall j \neq k\}$$

Repeat these steps until convergence. Since the process is sensitive to initialization, we typically run K-Means multiple times and choose the best clustering.

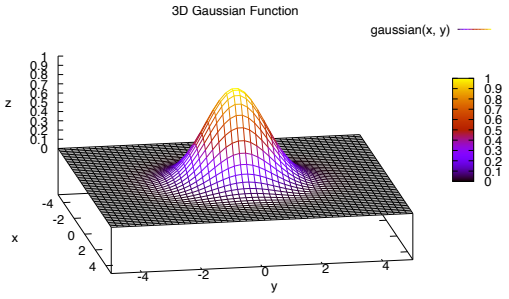

Figure 3: Example of a Gaussian RBF centered at μ

V. Implementation Notes

The RBF network was implemented in Python using NumPy and scikit-learn. To avoid memory issues, we:

- Use MiniBatchKMeans to find centers efficiently.
- Use perceptron-based or regularized least squares learning.
- Avoid computing large $\Phi^T\Phi$ matrices when possible.

VI. Conclusion

RBF networks provide an elegant and powerful framework for classification. Through Gaussian kernel construction and data-driven center selection, they can adapt to local structures in the data. When combined with efficient encodings such as Gram matrices and practical training algorithms, they scale to real-world tasks like digit recognition.