Machine Learning in Nuclear Spin Systems: Towards Quantum Learning with Quantum Input

Vivek Sabarad

Indian Institute of Science Education and Research Pune

October 17, 2024

Section: Introduction

Welcome to the World of Machine Learning!

- **What is Machine Learning?**
  - Imagine teaching a computer to make predictions without giving it direct instructions.
  - Just like how we learn from experiences, machine learning algorithms learn from data.
- **In Simple Terms:**
  - Feed the machine data, and it finds patterns to make decisions on its own!
- **Why Should You Care?**
  - It's everywhere! From self-driving cars to recommendations on Netflix, it's behind the scenes of our everyday tech.
  - The 2024 Nobel Prize in Physics was awarded to John J. Hopfield and Geoffrey E. Hinton for their foundational work in machine learning using artificial neural networks.

Machine learning

- The entire machine learning process can be looked at as estimation of an unknown function that a given data as follows:
- Input vectors $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^d$
- Output values $\{y_i\}_{i=1}^n \subset \mathbb{R}$
- We want to estimate the function $f^*(\mathbf{x}_i) = y_i$
- **Regression**: y_i is a continuous value, **Classification**: y_i is a discrete value.
- $f^*(\mathbf{x}_i)$ can be linear and non-linear in $\mathbf{x}_i$.

Kernel Methods

- Map input data to high-dimensional feature spaces $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$, where $\mathcal{H}$ is a high-dimensional feature space.
- A nonlinearity in the input space can be linear in the feature space.
- Requires the inner product of input vectors in the feature space to find this decision boundary.
- *Kernel Function*: A function $K(\mathbf{x}_i, \mathbf{x}_j)$ that computes the inner product in the feature space.

Kernel Methods

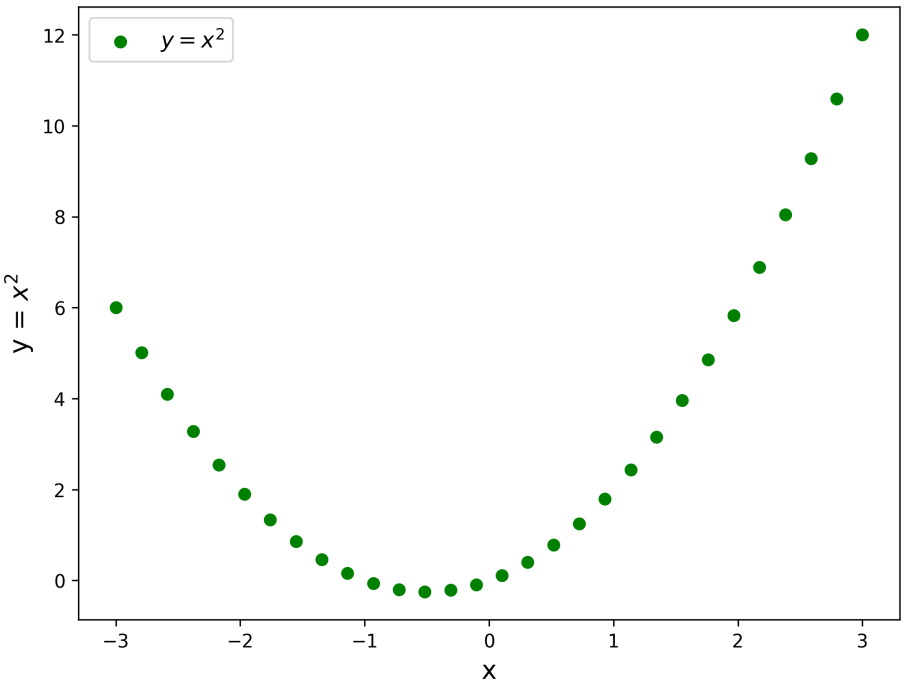


Figure: Data points tha follow $y = x + x^2$

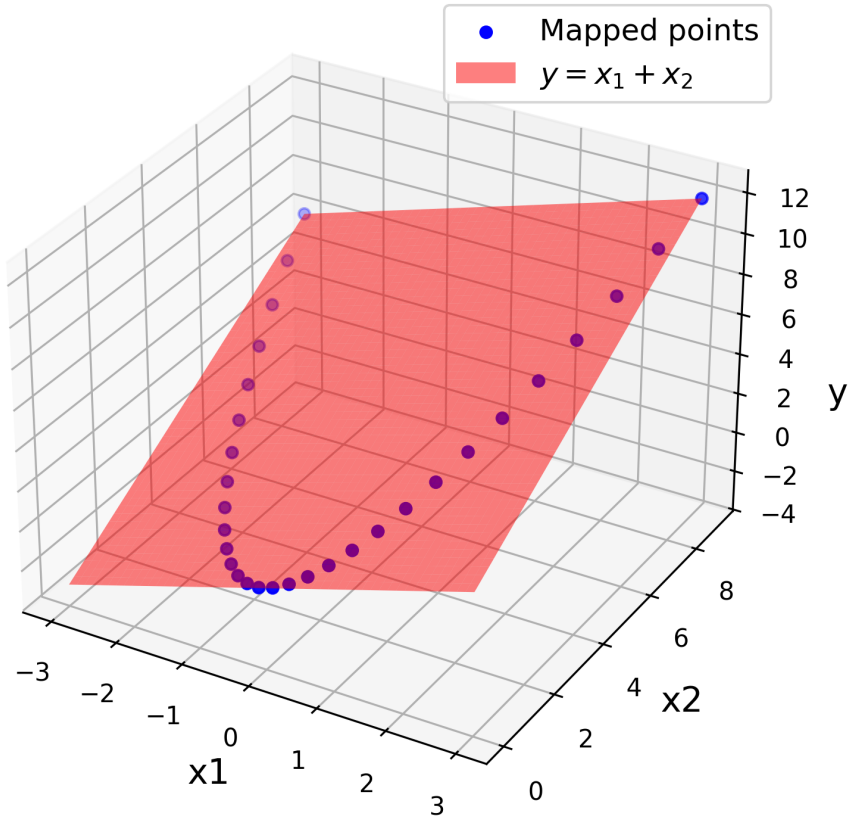


Figure: x mapped to (x, x^2)

The Four Paradigms of Quantum Machine Learning

- Quantum Machine Learning (QML) sits at the intersection of quantum computing and classical machine learning.
- Four key paradigms of QML:
  1. **Quantum-enhanced machine learning**: Accelerates classical algorithms using quantum processors.
  2. **Quantum-applied machine learning**: Classical methods are applied to quantum systems.
  3. **Quantum-inspired machine learning**: Classical algorithms inspired by quantum theory principles.
  4. **Fully quantum machine learning**: Both data and algorithms are quantum in nature.

Motivation and Objectives

- **Focus of my project:**
  - First and the last paragidms: quantum-enhanced and fully quantum machine learning.
  - Emphasis on **quantum kernel methods** within the context of NMR quantum computers.
- **Why Kernel Methods:**
  - Crucial for handling non-linear tasks in classical machine learning.
  - Quantum kernel methods extend classical kernels by embedding data into large Hilbert spaces natural to quantum systems.
- **Objective:** Devise methods to compute quantum kernels using NMR quantum computers to solve machine learning tasks.

Section: Quantum Kernel Methods

Quantum Kernel Methods

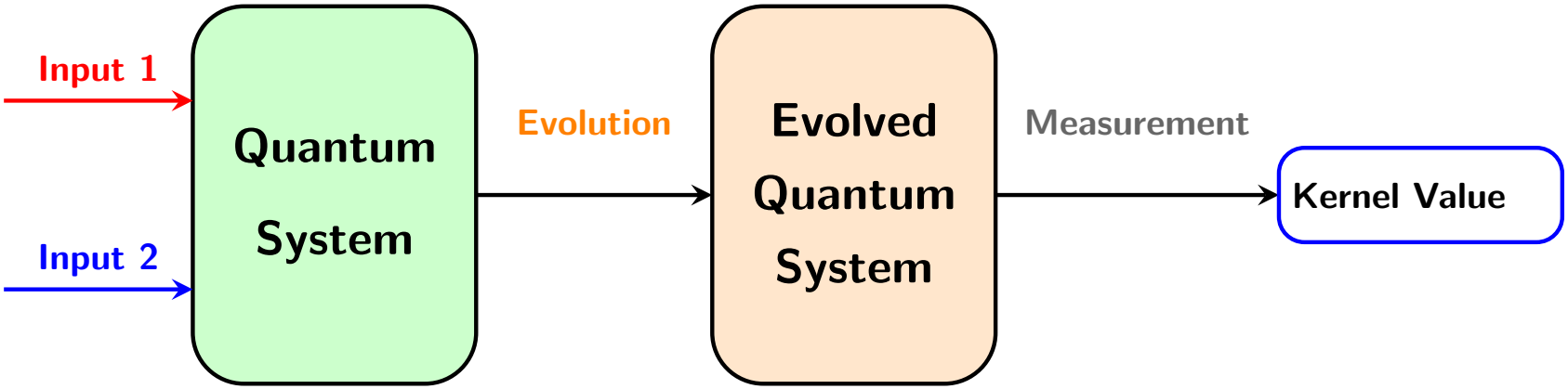
- **Classical kernels**: Predefined functions that computes the inner product in the feature space.
- Such a function has feature space mapping inherent in it.
- Examples:
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$
  - Gaussian kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$
- **Quantum kernel methods**: Map classical data into quantum states (or operators) in the exponentially large Hilbert space.
- The inner product in this Hilbert space is computed using various quantum computing techniques.

Quantum Feature Mapping

- Define a mapping from input vectors to quantum states or operators in a quantum system: $x_i \rightarrow \phi(x_i)$.
- These quantum states or operators exist in a quantum feature space - Hilbert space of the system.
- Quantum kernel $k(x_i, x_j)$ would inner product $\phi(x_i)$ and $\phi(x_j)$.
- Devise a method to compute this inner product using all the operations that a quantum system allows.

Quantum Kernel Computation

- Encode two input data point: The system is evolved using a input depended unitary transformation.
- Then the system is allowed to evolve on it's own or with some other *fixed unitary transformation* that let's the system explore the Hilbert space.
- Once input data is mapped to quantum states, the kernel function is computed by making *certain measurements* on the evolved system.



Applications of Quantum Kernel Methods

- Quantum kernel methods are used in Quantum Support Vector Machines (QSVMs), Quantum Kernel Ridge Regression (QKRR), etc.
- Our focus: Quantum kernels have the potential to handle **quantum data** for quantum tasks.
  - The input data in quantum tasks consists of quantum states or operators, which exist in the Hilbert spaces.
  - Quantum tasks include classification of unitary operators, entanglement classification, quantum state discrimination, etc.
- All this works in principle; the challenge is in the experimental realization of quantum kernels.

Section: Quantum Kernel in Nuclear Spin Systems

Quantum Computing in Liquid State NMR

- **Nuclear Spins as Qubits**: In liquid state NMR, nuclear spins of molecules in a liquid are used as qubits.
- **Quantum Gates**: Quantum operations (gates) are implemented by applying precise sequences of RF pulses.
- **Measurement**: The final state of the qubits is measured by detecting the NMR signal, which corresponds to the magnetization of the spins.
- **Advantages**: Liquid state NMR allows for high precision control and manipulation of qubits, making it suitable for small-scale quantum computations.

Density Matrix Formalism

- **Density Matrix for Pure States:**
  The density matrix ρ for a pure state $|\psi\rangle$ is defined as:
  $$\rho = |\psi\rangle\langle\psi|$$
- **Mixed States:**
  For a mixed state, which is a statistical ensemble of pure states $|\psi_i\rangle$ with probabilities p_i, the density matrix is:
  $$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$
- The action of a unitary operator U on a density matrix ρ is given by:
  $$\rho \to U\rho U^\dagger$$
- For an observable A, the expectation value in a state described by ρ is given by:
  $$\langle A\rangle = \mathrm{Tr}(\rho A)$$

Essence of the Density Matrix

- **Why the Density Matrix?**
  - Describes how to compute measurable quantities (observables) even when full information is not available.
  - Can represent statistical mixtures of quantum states, not just coherent superpositions.
- **Pure vs Mixed States:**
  - **Pure State:** Complete knowledge of the system, represented by a state vector $|\psi\rangle$.
  - **Mixed State:** Partial knowledge, or a statistical mixture of pure states, represented by a probability distribution over them.

Quantum Feature Mapping

- Classical data is encoded into quantum states of nuclear spins using a unitary transformation, $U(x_i)$.
- Consider the data-dependent operator, $A(x_i) = U(x_i) I_z U^\dagger(x_i)$ which serves as a feature map in a high-dimensional space.
  where, $I_z (= \sum_{\mu=1}^{n} I_{z,\mu}) \rightarrow$ total z-magnetization
  $I_{\alpha,\mu}$ $(\alpha = x, y, z) \rightarrow \alpha$-component for the μ-th spin.

Quantum Kernel Computation

Kernel is defined with the Frobenius inner product in this operator space:

$$\begin{aligned} k_{\text{NMR}}(\mathbf{x}_i, \mathbf{x}_j) &= \text{Tr}(A(\mathbf{x}_i)A(\mathbf{x}_j)) \\ &= \text{Tr}(U(\mathbf{x}_i)I_z U(\mathbf{x}_i)^\dagger U(\mathbf{x}_j)I_z U(\mathbf{x}_j)^\dagger) \end{aligned}$$

Using, $\rho_{\text{eq}} \approx \frac{1}{2^n}(1 + \epsilon I_z)$ as the equilibrium state:

$$k_{\text{NMR}}(\mathbf{x}_i, \mathbf{x}_j) \propto \text{Tr}(U(\mathbf{x}_j)^\dagger U(\mathbf{x}_i)\rho_{\text{eq}} U(\mathbf{x}_i)^\dagger U(\mathbf{x}_j)I_z)$$

Here, $U(\mathbf{x}_j)$ is called the encoding unitary.

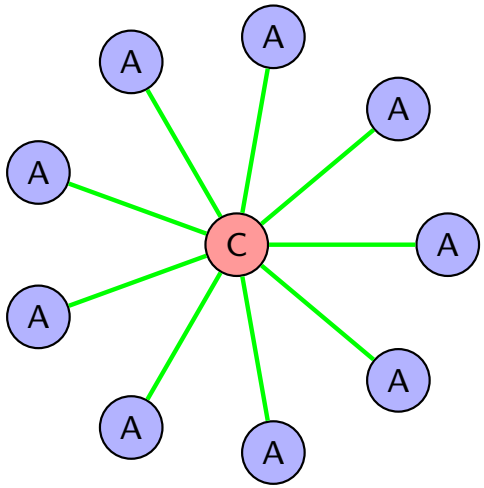
Spin System

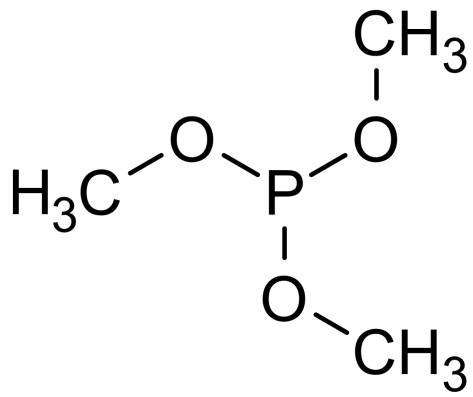
- Assuming the input vectors are one-dimensional, the encoding unitary is chosen as follows:

$$U(x_i) = e^{-ix_i I_z} U_e e^{ix_i I_z}$$

where U_e is the entangling unitary.

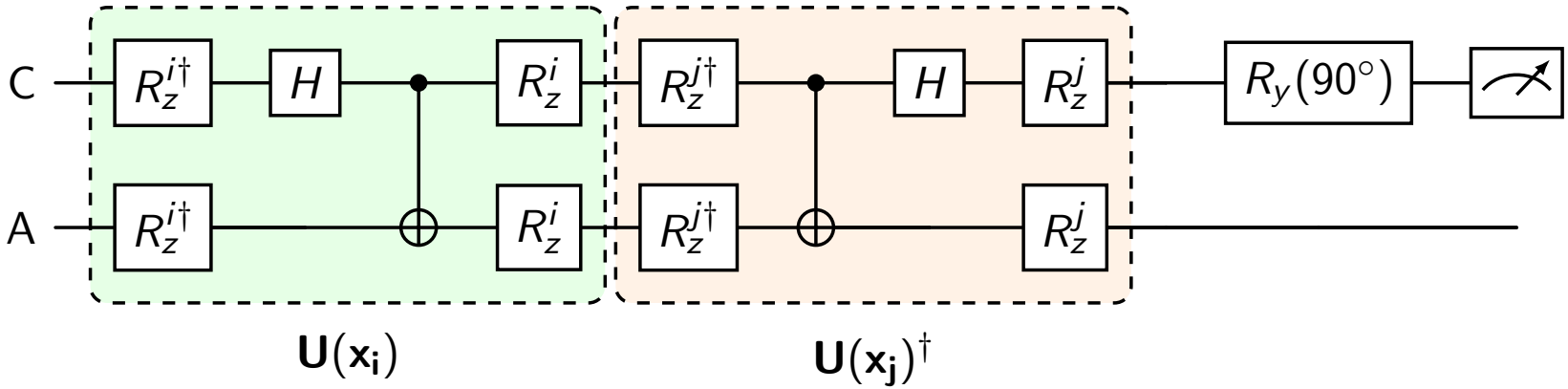
- Implemented using star systems in liquid state NMR:
  1. Both C and A spins are used to encode the data.
  2. The unitary U_e entangles all the A spins with the central spin C.
  3. The measurement is done only on the central spin C.





Quantum Circuit

- Computing the kernel between two inputs x_i and x_j: $k(x_i, x_j)$

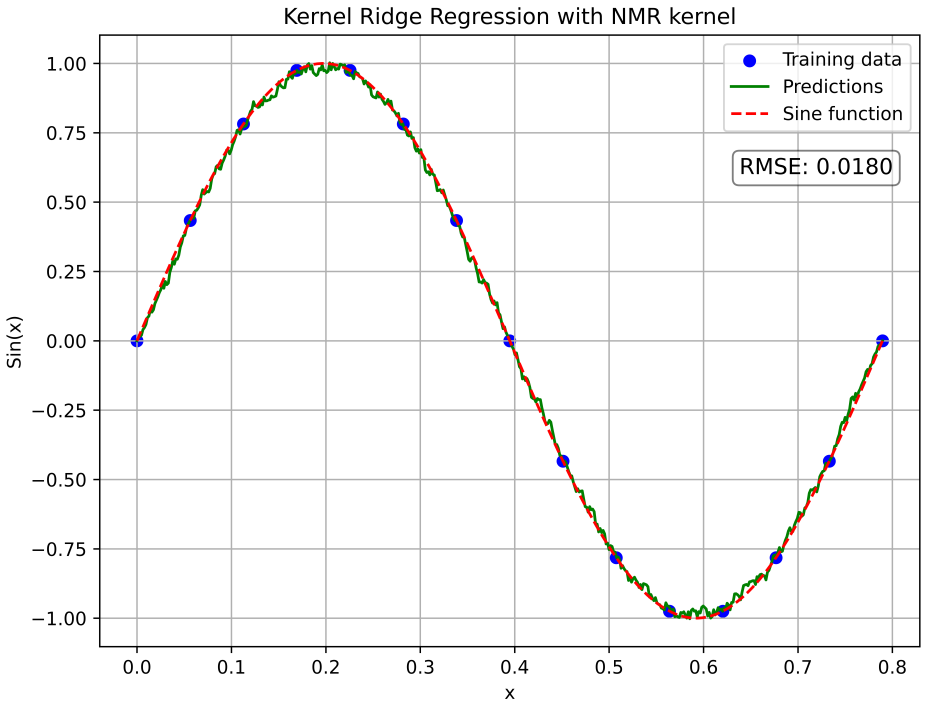


- For two-dimensional inputs: $\mathbf{x}_i = (x_i^1, x_i^2)$, the encoding unitary is given by:

$$U(\mathbf{x}_i) = e^{-ix_i^{(1)} I_z} e^{-ix_i^{(2)} I_z} U_e e^{ix_i^{(2)} I_z} e^{ix_i^{(1)} I_z}$$

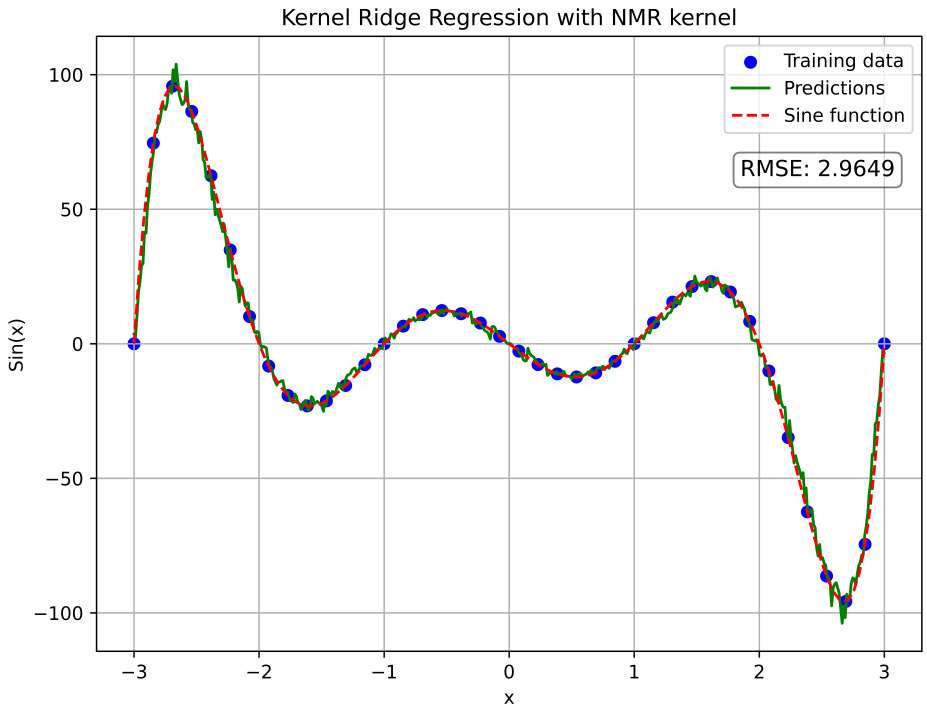
Results - 1D Regression

- One-dimensional regression task is done by kernel ridge regression.
- The regression task is performed on a sine curve for one period.
- The size of the training data is 15.



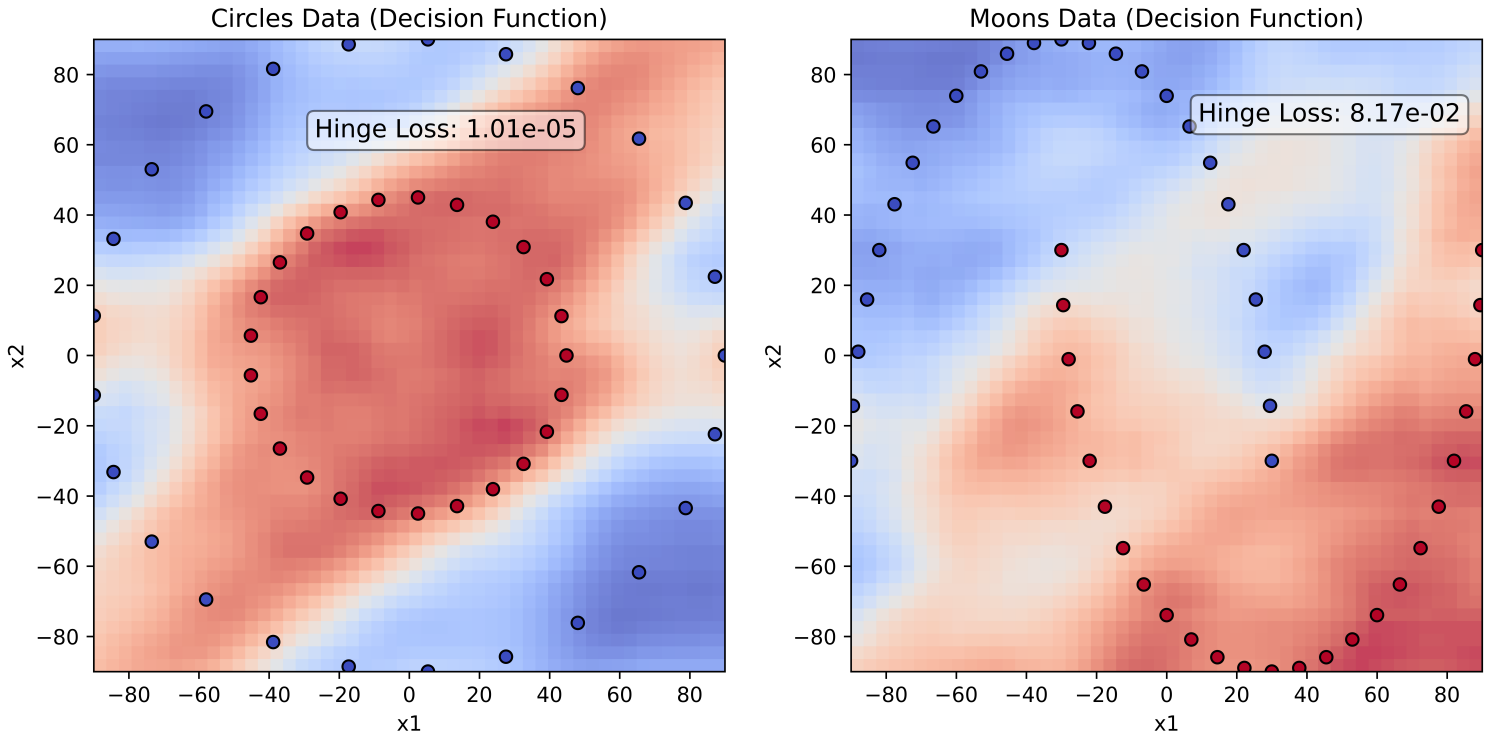
Results - 1D Regression

- This slide shows the results of regressing an 8th-degree polynomial.
- The size of the training data is 40.



Results - 2D Classification

- A two-dimensional classification task is done using a support vector machine (SVM) classifier.
- The SVM inherently uses the kernel matrix for classification.



Section: Quatnum Task: Entanglement Classification

Entanglement Classification

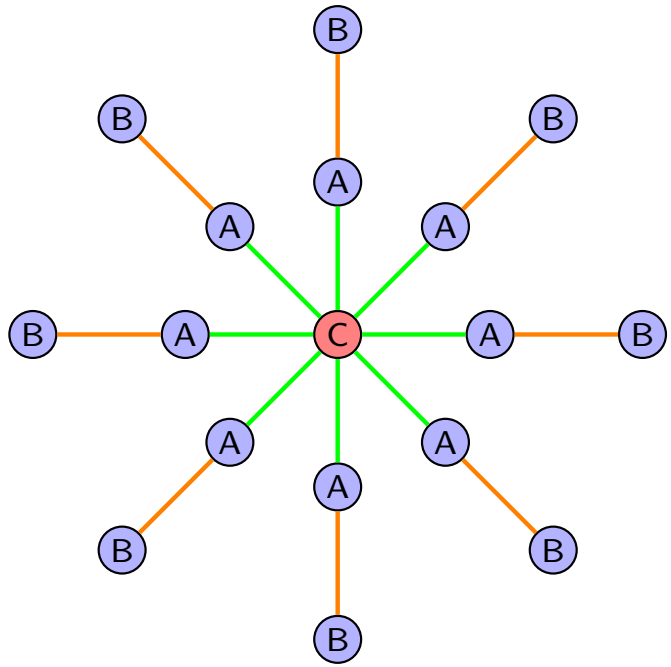
- The task is to classify entangled and non-entangled states.
- The inputs are two-qubit states and the output is binary.
- The objective is to be able to classify the states using the kind of quantum kernel methods we have developed.

Quantum Kernel Computation

- Encoding the two-qubit states into the nuclear spins:

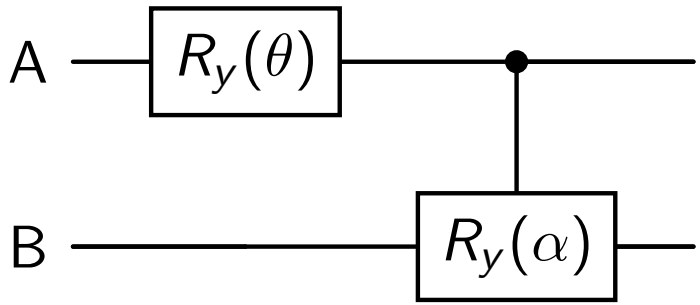
$$U(\psi_i) = U_{\psi_i} U_e U_{\psi_i}^{\dagger}$$

- The encoding unitary U_{ψ_i} is the preperation unitary of the state ψ_i: $\psi_i = U_{\psi_i}|00\rangle$.
- The entangling unitary U_e entangles the multiple two-qubit systems to the central spin.



Quantum states

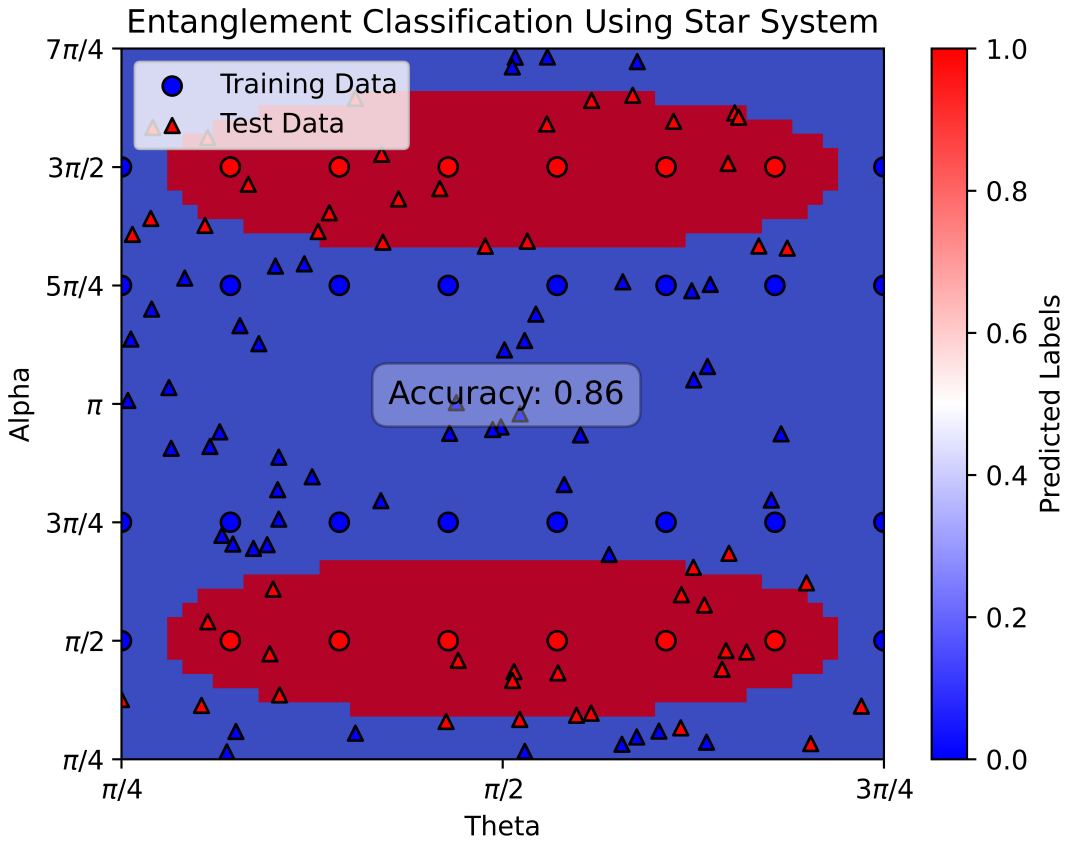
- Quantum circuit representing the preperation unitaries for the states used in the task.



- These unitaries act on the thermal states creating entangled states for some range of values of θ and α.
- The task is to classify these states as entangled or non-entangled given a training data set.

Numerical results

- Numerical results for the entanglement classification task with training data set of 32 states.



Section: Conclusion

Conclusion

- **Quantum Kernels:** We've explored the fascinating world of quantum kernels, where classical data meets quantum states, opening up new possibilities for machine learning.
- **NMR Systems:** By leveraging nuclear spin systems, we've demonstrated how quantum feature mapping and kernel computation can be practically implemented.
- **Applications:** From regression tasks to entanglement classification, quantum kernels show promise in tackling both classical and quantum problems.
- **Future Directions:** The experimental realization of quantum kernels is just the beginning. Imagine the potential as quantum computing technology continues to advance.

Section: Appendix

Linear Regression: Overview

- **Objective:** Model the relationship between dependent and independent variables.
- Finds the best-fitting linear equation to describe the data.
- Given dataset: $\{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$.

Linear Model

Linear Model:

$$y_i = f^*(x_i) + \epsilon_i$$

where:

- $f^*(x_i) = w^T x_i + b$
- $w^T x_i$ is the weighted sum of features.
- b is the bias term.

Cost Function

Mean Squared Error (MSE) Cost Function:

$$J(w, b) = \frac{1}{N} \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

- Aim is to minimize $J(w, b)$.
- Solution given by the normal equation:

$$w = (X^T X)^{-1} X^T y$$

Regularization

Regularized Cost Function (Ridge Regression):

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \lambda \|w\|^2$$

- Adds a penalty to avoid overfitting.
- λ controls the trade-off between fitting the training data and model complexity.

Kernel Ridge Regression: Feature Mapping

- Handles non-linearity by transforming input data into a higher-dimensional feature space.
- Transformation via a non-linear function $\varphi : \mathbb{R}^d \rightarrow F$.
- The relationship becomes linear in the transformed space:

$$f^*(x_i) = w^T \varphi(x_i)$$

Kernel Trick

Kernel Trick:

- Directly computes inner product in feature space without explicit transformation.
- Kernel function:

$$k(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$$

- No need to compute $\varphi(x)$ explicitly.

Derivation of the Kernel Trick

Step 1: Representing the weight vector:

$$w = \sum_{i=1}^{N} \alpha_i \varphi(x_i)$$

Step 2: Substituting in the cost function:

$$J(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \left[y_i - \sum_{j=1}^{N} \alpha_j \langle \varphi(x_j), \varphi(x_i) \rangle \right]^2 + \lambda \sum_{i,j} \alpha_i \alpha_j \langle \varphi(x_i), \varphi(x_j) \rangle$$

Step 3: Using the kernel function:

$$J(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \left[y_i - \sum_{j=1}^{N} \alpha_j k(x_j, x_i) \right]^2 + \lambda \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$

Prediction Function: Using Kernel Trick

- Once α_i values are optimized, the prediction function becomes:

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x_i, x)$$

- No need to compute the high-dimensional features $\varphi(x)$ explicitly.

Common Kernel Functions

- **Linear Kernel:** $k(x_i, x_j) = x_i^T x_j$
- **Polynomial Kernel:** $k(x_i, x_j) = (x_i^T x_j + c)^d$
- **RBF Kernel:** $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$
- **Sigmoid Kernel:** $k(x_i, x_j) = \tanh(\alpha x_i^T x_j + c)$