Quantum Support Vector Machines

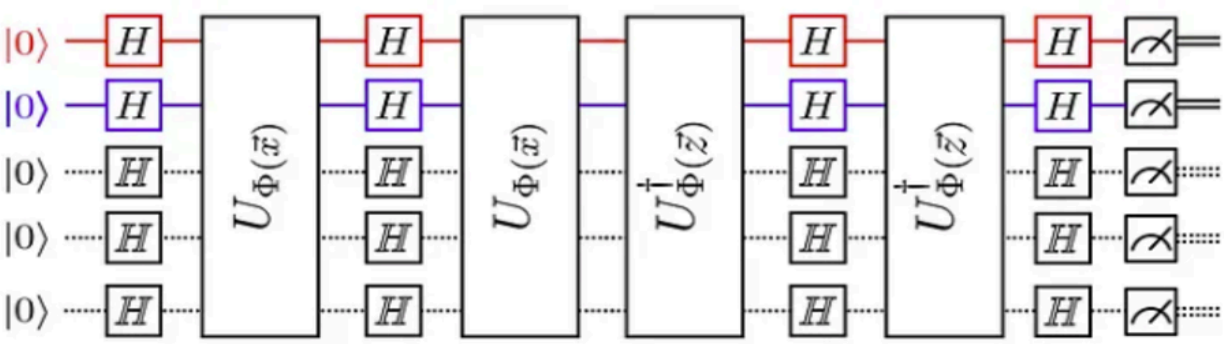
Joseph Suess

Support Vector Machines (SVM) are one of the most studied algorithms within machine learning. These highly adaptable methods assist in regression and outlier detection tasks, but the bulk of their use comes in data classification techniques. In a dataset, SVMs adapt to a target feature and create a maximum-margin hyperplane between different targets. In linear classifications, this plane is constructed in a way that minimizes as much loss as possible. In non-linear classifications, there will not exist an optimal hyperplane. A field of potential solution, which has been extensively studied in recent literature, is applying a kernel to the system to scale the datasets to a higher dimension, while not adding any new parameters to the existing set. New decision surfaces allow for a clearer hyperplane in a higher dimension as well. Many options exist for choosing a kernel, but the simplest way to scale is with dot or inner products. The equation of the kernel can be formalized as: $K(x,y) = \phi(x) \cdot \phi(y) = \langle \phi(x) \mid \phi(y) \rangle$. Some important properties should be noted, such as the result of the kernel will always be real, and positive. Additionally, the dataset should always be real.

Quantum feature spaces can be highly expressive and non-linear. Consider a dataset where classes are only separable in high-dimensional, non-linear feature spaces. Classical kernels might reach a good approximation, but a quantum kernel has the ability to see intricate relationships that might have initially been missed. To evaluate a quantum kernel, a quantum feature map must be used that takes in classical information and encodes it into the quantum Hilbert space. $|\Phi(x)\rangle = U(x) \; |0\rangle^n$ should be the considered equation that initializes an empty space and transforms it accordingly. Once encoded, measuring the kernel is simply a matter of measuring the observable, $K(x,y) = |\langle \Phi(x_i) \mid \Phi(x_j) \rangle|^2 = \; |\langle 0^n \mid U^\dagger(x_i) U(x_j) \mid 0^n \rangle|^2$, where x_i and x_j denote two classical points. Without optimization, evaluation of the kernel for the dataset takes $O(N^2)$ operations, which offers no speedup from the classical kernel.

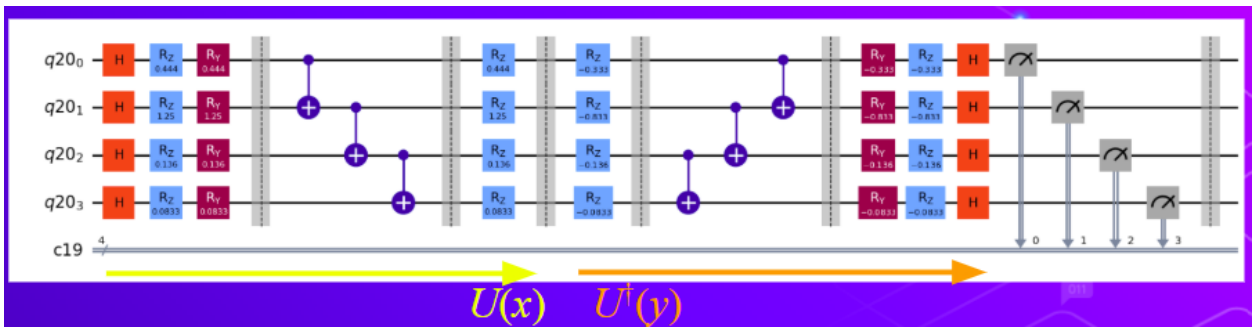
Choosing the *U(x)* encoding function is critical to the success of the kernel, and the options depend heavily on the dataset. For instance, periodic data might benefit from a *U(x)* that encodes single-qubit rotational phases with magnitude of $\cos(x_i)$. Entanglement using CNOT gates should

also be considered when encoding a dataset with features that depend on each other. These datasets, in literature, are referred to as non-linear. The dataset chosen for implementation is the Iris Flower set in the sklearn library. Some advantages of utilizing this set is that it is a combination of linear and non-linear elements, the set is pre-standardized and is a known benchmark for testing kernel applications. Creating a custom feature map for this set, also known as a "shallow circuit" can be expressed as: $U(x) = (\otimes HR_Z(x_i)R_Y(x_i)U_{ent}R_Z(x_i))$. This was adapted from the circuit from Aram Harrow, *et al.* :



Quantum Kernel

The main differences include treating the data as non-linear by entangling as well as utilizing the y, z parameters of the Bloch Sphere. Though not much difference, there was a slight increase in accuracy on this dataset when utilizing these rotational gates. Now we can express the kernel after defining U(x):



Measuring the $|0\ldots0_n\rangle$ state on the kernel implies likelihood of similar classification. At this stage, we can compute a kernel matrix, A_{ij}, for each point to not explicitly evaluate each vector, but because the dataset is relatively small (150 flowers, 3 equal sets), we can divide the data into a training and testing portion. 20 random points were selected from each flower group for a total

of 60 trained points. The remaining 90 data points were put through the kernel circuit for evaluation to be immediately classified. Classification was constructed using a custom algorithm that took the highest probability for a point to be considered in each of the 3 sets. 4 points were misclassified, resulting in a success rate of 86/90, or 95.56%. A total of 20 tests were run sequentially, and the average success rate was 94.52%. 20 classical tests were also recorded. The classical kernels (linear, poly, rbf) resulted in the subsequent accuracy scores: 94.16%, 95.83%, 80.83%. Meaning, this particular classification problem did not benefit too much from a quantum kernel of the data.

References

Akrom, M. (2024). Quantum Support Vector Machine for Classification Task: A Review. *Journal of Multiscale Materials Informatics*, *1*(2), 1–8. https://doi.org/10.62411/jimat.v1i2.10965

Joachims, T. (2011). *Support Vector Machines: Kernels CS4780/5780 -Machine Learning Fall 2011*. https://www.cs.cornell.edu/courses/cs4780/2011fa/lecture/08-svm_kernels.pdf

Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum Support Vector Machine for Big Data Classification. *Physical Review Letters*, *113*(13). https://doi.org/10.1103/physrevlett.113.130503

Suzuki, T., Hasebe, T., & Miyazaki, T. (2024). Quantum support vector machines for classification and regression on a trapped-ion quantum computer. *Quantum Machine Intelligence*, *6*(1). https://doi.org/10.1007/s42484-024-00165-0

Training a Quantum Model on a Real Dataset - Qiskit Machine Learning 0.7.1. (n.d.). Qiskit-Community.github.io. https://qiskit-community.github.io/qiskit-machine-learning/tutorials/02a_training_a_quantum_model_on_a_real_dataset.html