CSCI 5481 Final Project

Brian Cooper

Quantum Machine Learning

Introduction – Project Overview

- For my final project, I wanted to compared quantum algorithms (algorithms designed to run on a quantum computer) to classical algorithms (algorithms designed to run on a "classical" computer)
  - **Classical computers:** binary logic (discrete state of 0 or 1)
  - **Quantum computers:** quantum bit (“qubit”) logic (superposition of states between and including 0 and 1 until observed, then they “collapse” into a discrete state)
- Classical computers have been used extensively to solve machine learning problems, but applied quantum computing is a relatively new and smaller space
- **Question of interest:** Are quantum computing algorithms useful and practical for working with bioinformatics/genomics machine learning problems?

Introduction – Quantum Computing

- Quantum computing is very much in its infancy, but there have been strides from major companies such as IBM and Microsoft to open the field to interested individuals
  - D-Wave Ocean[2]
  - IBM Q[3] (and Qiskit[5])
  - Microsoft Q#[4]

- For this project, I went with Qiskit
  - Qiskit is a Python SDK designed to simulate quantum processes
  - Allows ability to directly access physical quantum computers through IBM Q
  - Includes OpenQASM[6] ("Open Quantum Assembly Language"): implements quantum circuitry close to the hardware

Introduction – Quantum Computing

- There are not yet many options for testing quantum computing from a home computer, especially in the artificial intelligence/machine learning scene
- However, Qiskit includes a *quantum support vector machine* (QSVM) algorithm, as well as a classical support vector machine (SVM) algorithm for reference
- Qiskit is a very young project, and some of the documentation is lacking (blank pages, filler text, etc.)
  - To learn about it, I recommend using a combination of articles, the official documentation, looking through the source code (publicly hosted on Github), and the official Qiskit tutorials[8]
- More about quantum computing itself can be learned at Martin Giles' writeup on MIT Technology Review titled *What Is a Quantum Computer?*[1]

Introduction – Quantum Computing

- The qubits in the system used for this project are “entangled” with each other
  - Entanglement is characterized by a lack of independence among the components in a system
  - In this project, the system is composed of two qubits
  - Observing the state of one qubit provides information about the other
  - Entanglement provides the groundwork for *topological* quantum computing[9], which is currently largely theoretical
    - However, much research is being performed in this field
  - Entanglement is also harnessed for “quantum annealing,” another approach to quantum computing that minimizes an objective function
  - The University of Minnesota has recently started researching topological quantum computing[10]

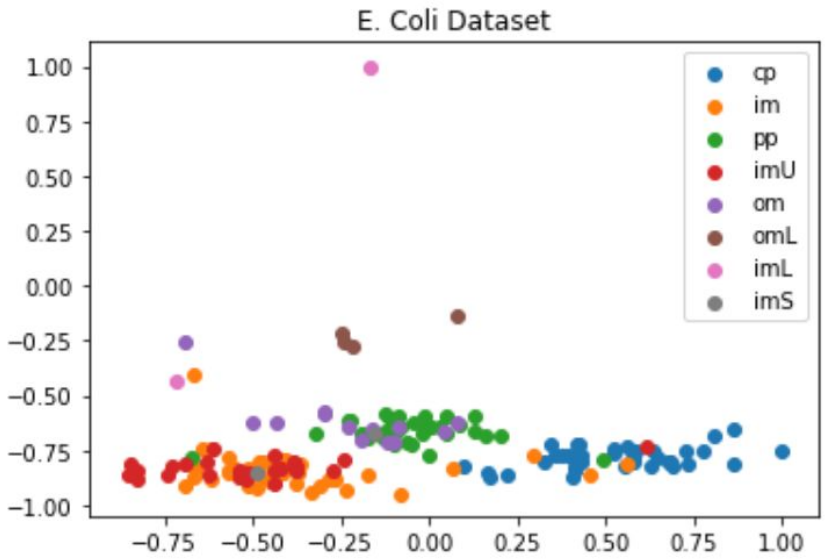
Data

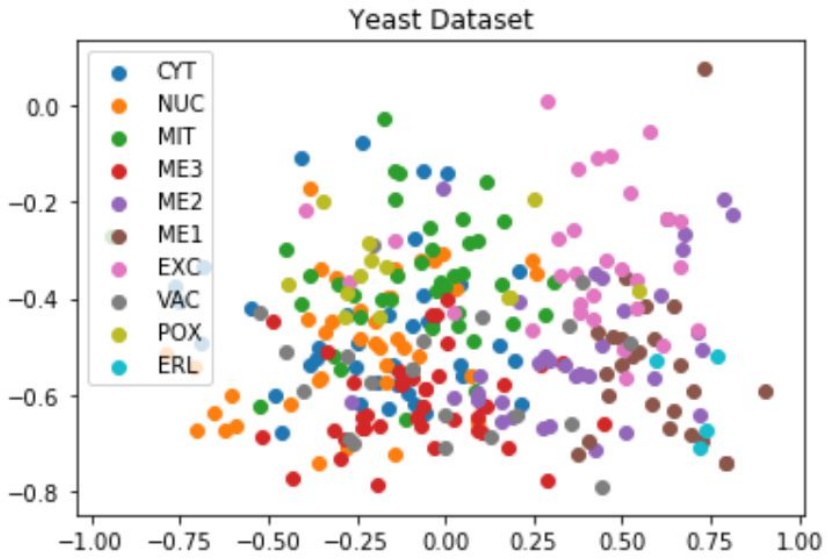
- I applied the SVM algorithms to three datasets from the UCI Machine Learning Repository[7]:
  1. Ecoli: https://archive.ics.uci.edu/ml/datasets/Ecoli
  2. Yeast: https://archive.ics.uci.edu/ml/datasets/Yeast
  3. Mouse Protein Expression: https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression
- All of the above datasets have been previously used for classification training and testing with various methods, including (but certainly not limited to) various kernel SVM methods

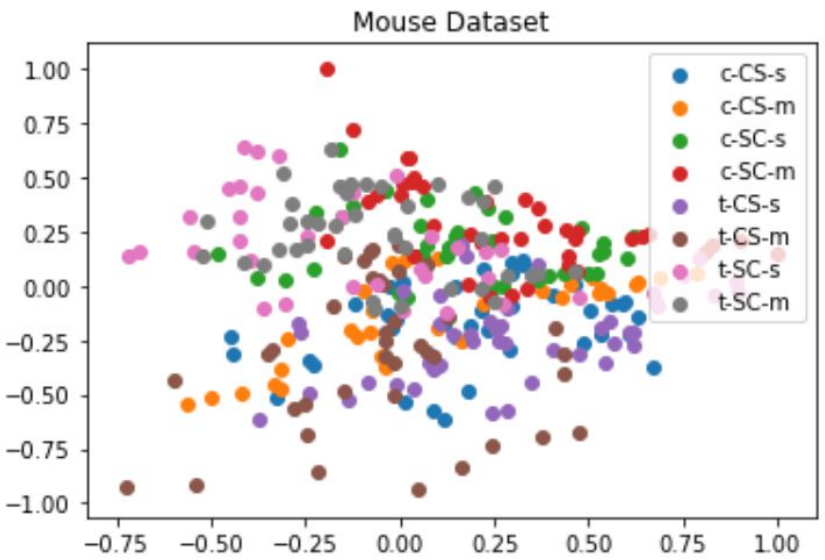
Data

- I reduced the dimensionality of each dataset using principal component analysis to match the number of qubits used in quantum algorithm (2 qubits; 2 dimensions)
- The entire datasets are used to build a model for each algorithm (QSVM and SVM with RBF kernel)
  - 70% training data split, 30% testing split to analyze model

Data – Visualization (PCA-dim. Reduced)







Experiments

Applying QSVM

- QSVM uses a quantum processor to directly estimate the kernel in the feature space
- I used the built-in simulator (QASM simulator) for code portability, but an IBM Q physical backend could be used instead (should produce similar results, possibly faster processing speed depending on cloud queue) – using a physical backend is explained at the Qiskit IBMQ Provider Github repository[11]

Applying SVM

- SVM includes RBF kernel built-in
- I used this algorithm as a reference to check the performance and output of QSVM

Comparing SVM and Quantum SVM

- I used the accuracy results to measure the performance of the models
- Since the experiments were simulated (quantum mechanical computation on a classical computer), time and space complexity would not be useful measurements (inaccurate)
  - They would be useful if real quantum hardware were used

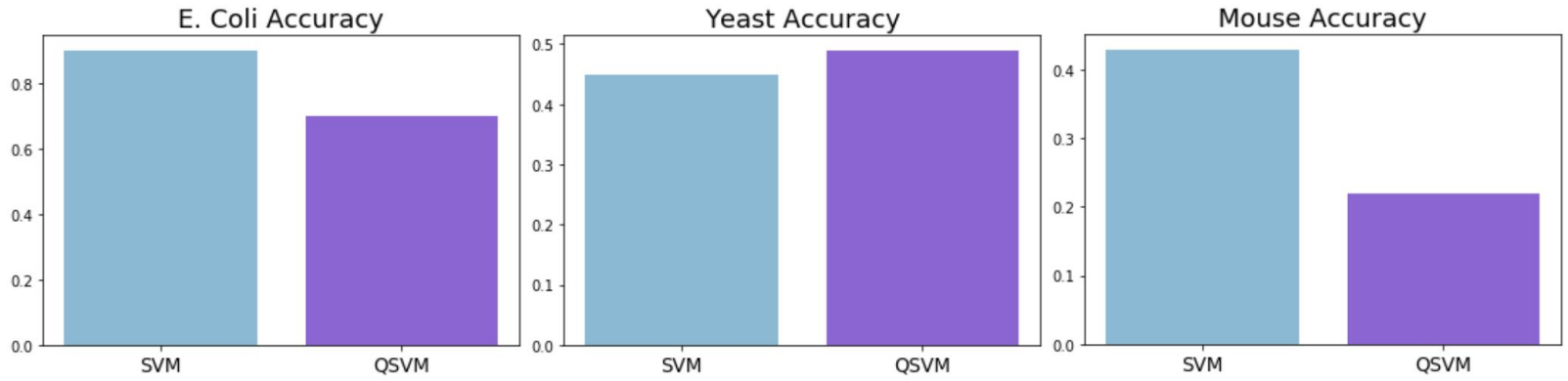
```
   E. Coli Dataset Accuracy
------------------------------
SVM (Radial Basis):       0.90
Quantum SVM:              0.70
------------------------------

    Yeast Dataset Accuracy
------------------------------
SVM (Radial Basis):       0.45
Quantum SVM:              0.49
------------------------------

    Mouse Dataset Accuracy
------------------------------
SVM (Radial Basis):       0.43
Quantum SVM:              0.22
------------------------------
```

Comparing SVM and Quantum SVM

	Classical SVM	Quantum SVM
E. Coli	90%	70%
Yeast	45%	49%
Mouse	43%	22%



Discussion

- A parameter that can be specified when constructing an algorithm in Qiskit is the number of *shots*, which specifies how many times a quantum circuit is repeated
  - Used to gradually build up distribution statistics of the logical circuit results
  - Set to 1024 by default in Qiskit, but I reduced the value to 256 to speed up computation time
- Even when reducing the number of shots to 256, the quantum algorithms still took a very long time compared to classical algorithms
  - The three quantum algorithm runs (with 256 shots) took several hours
  - The three classical algorithm runs took less than one minute (overall)
- The experiments were performed on a simulator rather than physical quantum devices, and the performance would likely improve on a physical quantum backend
- Qiskit also supports GPU augmentation and offloading ("QCGPU Provider") for processing, this would likely improve the performance as well

Discussion

- More SVM kernels can be used on the data, as well as other classification algorithms
  - I used QSVM and SVM with radial basis kernel to stay within the Qiskit environment and provide an A/B test
  - Other kernels/algorithms may perform better on the data, especially considering their drastically varying distributions
- The quantum implementation of SVM created a slightly model with better accuracy on the yeast dataset with the same kernel (49% vs. 45%)

Discussion

- Only accuracy was compared on these datasets, as that is all I could find natively in Qiskit to analyze model performance with the SVM algorithms
- In the future, I'm sure more metrics will include native support, and more metrics could be added manually since the class predictions and true class values are contained within the Qiskit SVM result structure
- As Qiskit is a young project, many of the algorithms contained within are largely proof-of-concept
  - As the project matures, the algorithms will be optimized and will naturally become more performant
  - The documentation will concurrently be improved, especially considering that Qiskit is an open-source project
  - Other quantum computing projects will follow suit: over time, utilizing quantum technology will become more accessible and robust
  - This is an exciting and rapidly-changing field that may be very useful in all sorts of applications. Even though the experiments only involved comparing the two SVM algorithms currently shipped with Qiskit, this demonstration shows that it is possible to already use quantum algorithms without too much difficulty
  - Qiskit in particular abstracts away from the physics and digital logic quite highly, at least for quickly spinning up an algorithm without too much tuning

Discussion

- Overall, I think the quantum computing space is definitely something to consider for future genomics work
- Quantum computers will supposedly be excellent at solving optimization problems, which are at the core of artificial intelligence and machine learning problems
  - Thus, I see genomics and bioinformatics as a whole enjoying a surge of innovation as quantum computers establish their place in society

References

1. Giles, Martin. “Explainer: What Is a Quantum Computer?” *MIT Technology Review*, https://www.technologyreview.com/s/612844/what-is-quantum-computing/

2. *D-Wave’s Ocean Software.* https://ocean.dwavesys.com/.

3. “IBM Research AI.” *IBM Research AI*, 5 June 2018, https://www.research.ibm.com/ibm-q/.

4. *The Q# Programming Language.* https://docs.microsoft.com/en-us/quantum/language/.

5. *Qiskit | Quantum Information Science Kit.* https://qiskit.org.

6. Cross, Andrew W., et al. *Open Quantum Assembly Language.* July 2017. arxiv.org, https://arxiv.org/abs/1707.03429v2.

7. *UCI Machine Learning Repository.* https://archive.ics.uci.edu/ml/.

8. *A Collection of Jupyter Notebooks from the Community and Qiskit Developers Showing How to Use Qiskit: Qiskit/Qiskit-Tutorials.* 2017. Qiskit, 2019. GitHub, https://github.com/Qiskit/qiskit-tutorials.

9. Lahtinen, Ville, and Jiannis K. Pachos. *A Short Introduction to Topological Quantum Computation.* May 2017. arxiv.org, doi:10.21468/SciPostPhys.3.3.021.

10. *University to Lead $2.25 Million Grant for Developing next-Generation Quantum Computer | College of Science and Engineering.* https://cse.umn.edu/college/news/university-lead-225-million-grant-developing-next-generation-quantum-computer.

11. *Qiskit Provider for Accessing the Quantum Devices and Simulators at IBMQ: Qiskit/Qiskit-Ibmq-Provider.* 2018. Qiskit, 2019. GitHub, https://github.com/Qiskit/qiskit-ibmq-provider.