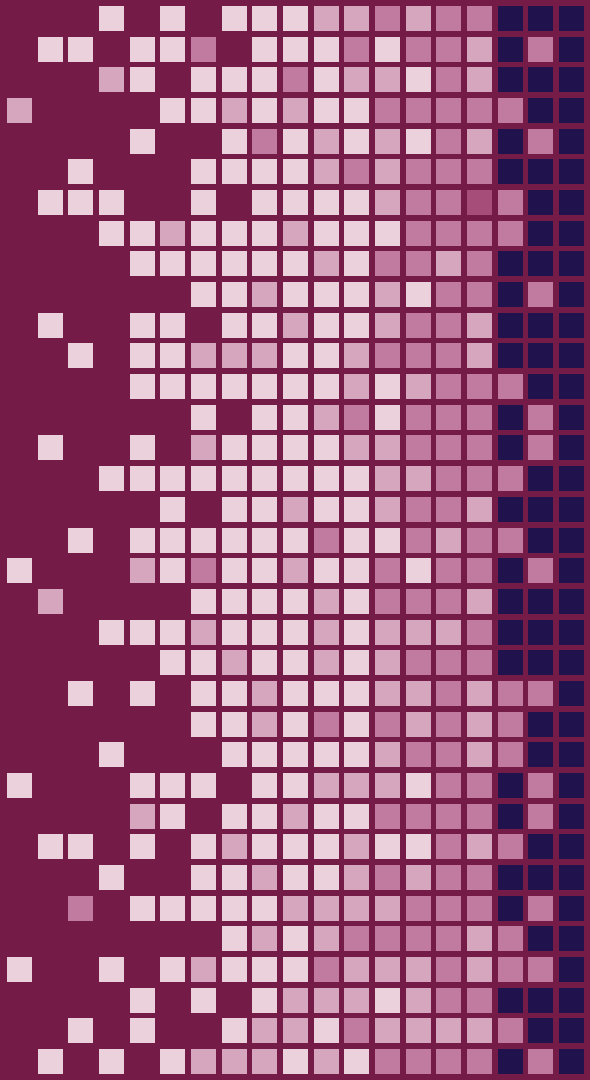
QSVM

Quantum Support Vector Machines for Breast Cancer Cell Detection on IBM's Hardware

Alice Liu

PROBLEM



BREAST CANCER: CURRENT STATE

- **3.8 million women** are diagnosed with breast cancer in US[1]
- **1 in 8** women will develop invasive breast cancer over her lifetime
- Rate for miss-classifying breast cancer cells: **20%**[2]
- Late / Misclassified Diagnosis: decreases chance of survival **90%** → **15%**
- Detecting breast cancer at early stage:
    - Better Treatment Options
    - Higher Chance of Survival

Current Key Challenges

1) Detecting breast cancer earlier (time)
2) Decreasing misdiagnosis rates (accuracy)

.

[1] Romeo , Elisa, et al. "Breast Cancer Misclassification: A ... - Journals.sagepub.com." *Sage Journals*, 2011, https://journals.sagepub.com/doi/full/10.2217/WHE.11.69.

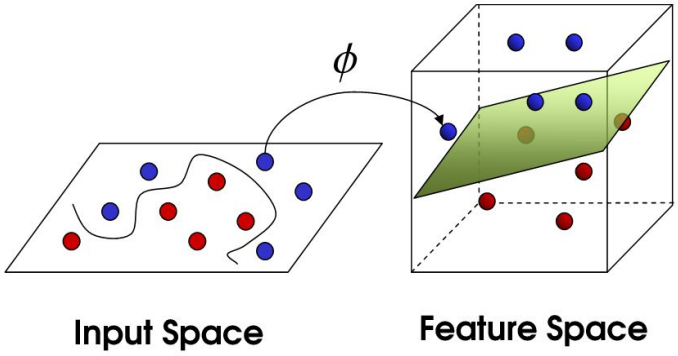
[2] "U.S. Breast Cancer Statistics." *Breastcancer.org*, 13 Jan. 2022, https://www.breastcancer.org/symptoms/understand_bc/statistics.

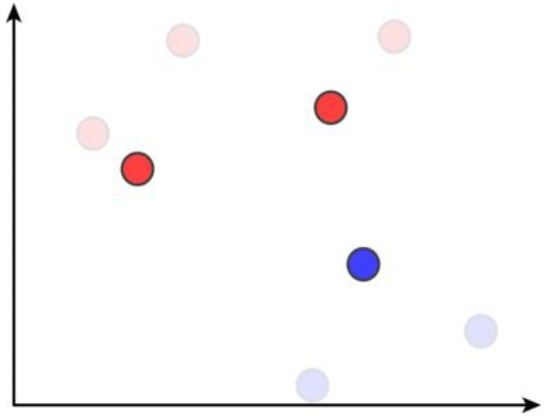
CURRENT ML: SVMs

Support Vector Machines (SVMs) are a supervised learning algorithm, where given labeled training data, will output an optimal hyperplane able to categorize new examples

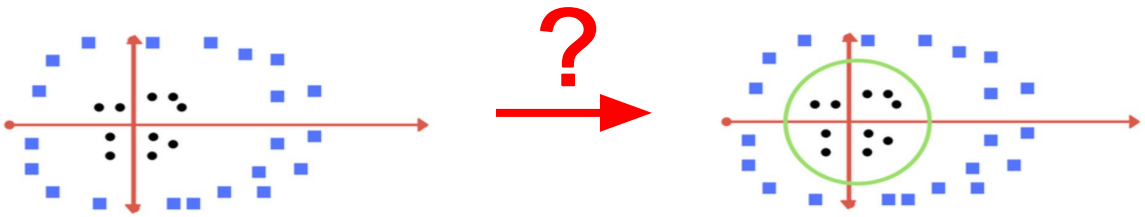
Finding the optimal hyperplane becomes more difficult as the dimensions increase.

.

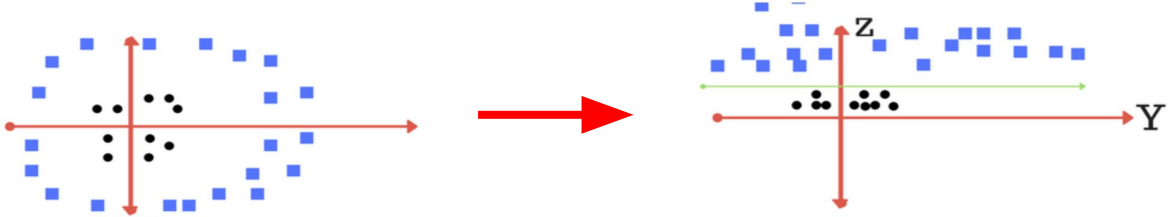




COMPUTATIONAL LIMITATIONS

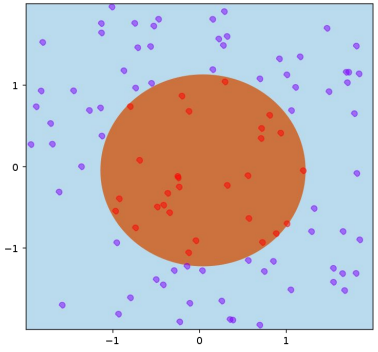


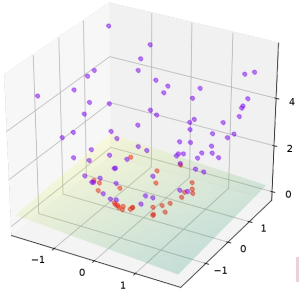
The Kernel Trick is able to map non-linear data set into a higher dimensional space where the hyperplane can be found to separate the samples.
In this 3D example, various transformations and the addition of the z-axis are able to be added to draw the plane.



CURRENT ML: SUPPORT VECTOR MACHINES

What if the data points dimensions projected keep *increasing* and the data becomes more *complex*?





Classical computers are unable to operate through these large computations without compromising time
—> can takes *weeks* to train.

Time and efficiency are both obstacles- our today's (classical) computers **lack the computational power** to effectively model classification techniques with increasing dimensionality.

QUANTUM MACHINE LEARNING

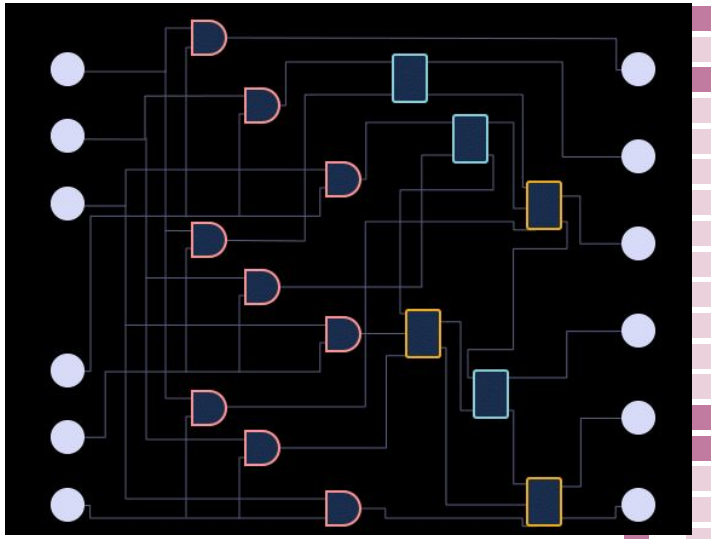
Intersection of *quantum computing and artificial intelligence* leveraging classical data and machine learning algorithms to be run on quantum processors

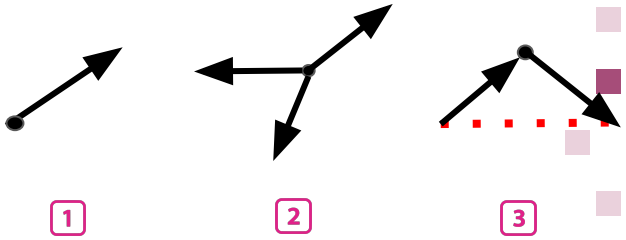
Based on the idea of **amplitude encoding** = amplitudes of a quantum state are associated with the inputs and outputs of computations.

States in qubits with amplitudes allows for an **exponentially compact representation** = associating a discrete probability distribution over random binary variables with a classical vector.

Space grows polynomially in the number of qubits, leading to **logarithmic growth** in the number of amplitudes and dimension of the input.

Amplitudes allow for new optimization routes to be found



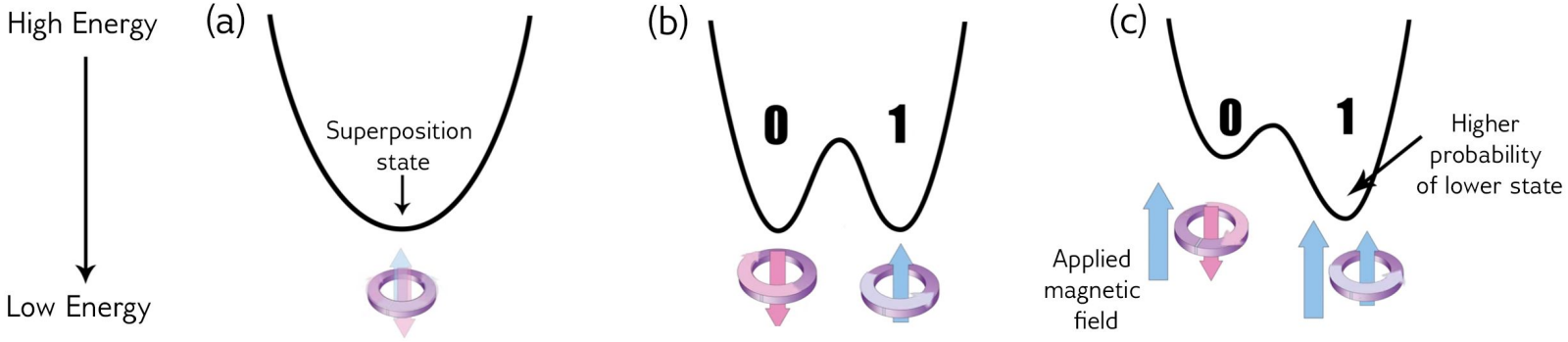


THE QUANTUM ADVANTAGE

Quantum computing = relies on quantum mechanical properties such as superposition, entanglement and tunneling, non-existent in current classical methods.

Qubits = quantum bit that can be in the state of 0, 1, or a **superposition** of 0 and 1 at the same time. (states 00, 01, 10, 11) , which allows for calculations to be run simultaneously.

n qubits = 2^n solutions able to be run at the same time → speedup is ideal for optimization



RESEARCH QUESTION

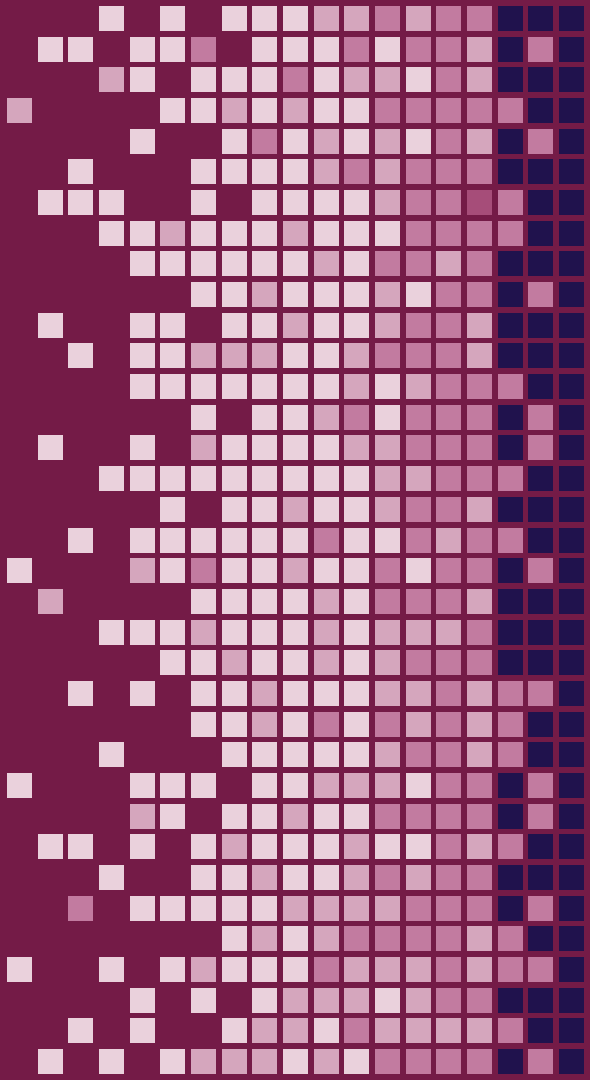
Can quantum computing methods speed up the process during binary classification, specifically when dividing breast cancer cells into benign and malignant groups?

HYPOTHESIS

The method that leverages 2 qubits with the quantum support vector machine algorithm will yield the most effective results in terms of the accuracy when classifying breast cancer cells.

This method can leverage the “quantum advantages” within the algorithm including the faster runtime, greater capacity, and higher efficiency “speedups” but will not be susceptible to noise and decoherence as experienced with quantum hardware when running on a QPU.

METHODOLOGY



EXPERIMENT OVERVIEW + DETAILS

OBJECTIVE

Using the **quantum support vector machine algorithm** on a quantum computer to explore its benefits in terms of speedups and increased accuracy (over classical computers) with the classification of cancerous cells

OVERVIEW

Tests the efficiency in the binary classification of breast cancer cells as benign or malignant in 4 different parts:

1) Classical Support Vector Machine Algorithm on a CPU (control)
2) Quantum Support Vector Machine Algorithm with a 2-Qubit SImulation on a QPU
3) Quantum Support Vector Machine Algorithm with a 4-Qubit SImulation on a QPU
4) Quantum Support Vector Machine Algorithm with an 8-Qubit SImulation on a QPU

What will be measured: *classification accuracy* (% of cells correctly classified)

All the quantum algorithm implementations will be run on IBM's quantum hardware through an API connected to the local machine

EXPERIMENT OVERVIEW + DETAILS

VARIABLES: INDEPENDENT, DEPENDENT, CONTROL

Independent: Type of Method Tested (Classical SVM Algorithm on CPU or Quantum SVM Algorithm on QPU)
Dependent: Percentage in the number of correctly classified cells
Control Variable: Classical SVM Algorithm on CPU (method that is currently being used today)

MATERIALS

- UCI Machine Learning Repository Breast Cancer Wisconsin (Diagnostic) Data Set
    - Parameters: *Texture, Perimeter, Area, Smoothness, Compactness, Concavity, Symmetry*
- IBM Quantum Cloud Service API. This is leveraged to run the algorithms and test data points on IBM's quantum hardware processors using a Python Interface.
- IBM's Qiskit Software Development Kit Version 0.23.4.
- Python Version 3.7.4
- Scikit Learn Software Package Version 0.21.3.
- Interactive web-based computational environment, ex. Jupyter Notebook was used.
- Modern Operating System consisting of x86 64-bit CPU (Intel / AMD architecture) with at least 4 GB RAM and 5 GB free disk space. For this experiment, the Mac OS Mojave Version 10.14.5 with a 1.1 GHz Intel Core m3 was used.

ALGORITHMS – SUPPORT VECTOR MACHINES

To draw the line that separates the different classes, the support vector machine algorithm finds a decision boundary with the widest distance of margin.

1. A vector w represents the decision boundary starting from the origin and a normal plane is drawn. An unknown sample *u* is included in the feature space.

2. Vector *u* is projected onto vector *w* by finding their dot product to determine which side the unknown sample lies on.

3. The decision boundary has a separation of distance from *-1 to +1* for all training samples in order to define the distance from each support vector to the decision boundary. This is represented by the equation:

$$y_i = \begin{cases} -1, & \vec{x} \in \vec{x}_+ \\ +1, & \vec{x} \in \vec{x}_- \end{cases}$$

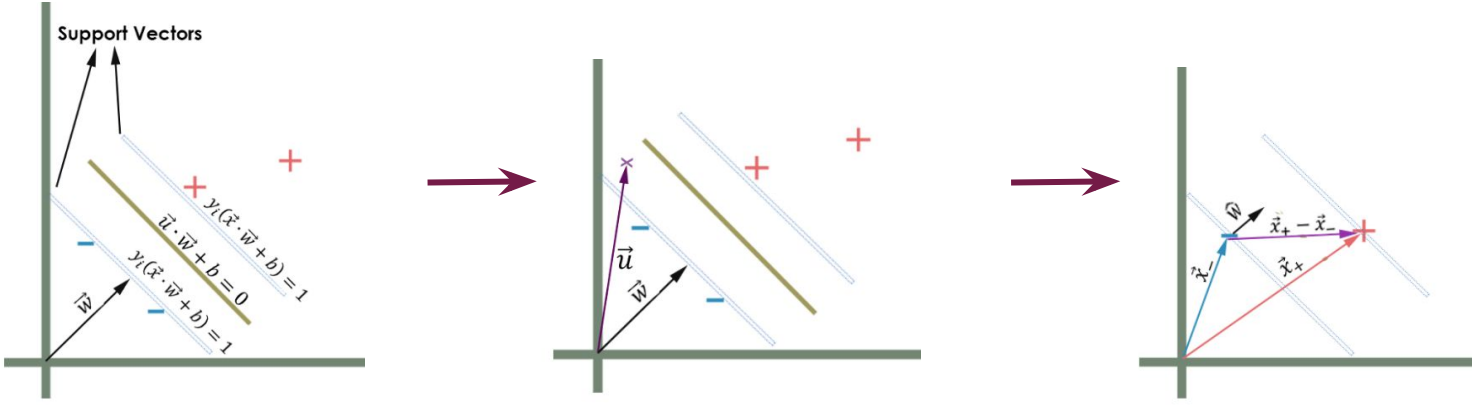
$$y_i(\vec{x} \cdot \vec{w} + b) \geq 1$$

ALGORITHMS – SUPPORT VECTOR MACHINES

4 The distance between the support vectors are equal to the projection of the difference of the vectors onto the unit vector in the direction of the normal vector *w*. The distance between the support vectors with the maximized constraints:

$$\max(width) \xrightarrow{yields} \max\left(\frac{2}{|\vec{w}|}\right) \xrightarrow{yields} \min(|\vec{w}|) \xrightarrow{yields} \min\left(\frac{|\vec{w}|^2}{2}\right)$$

5 Apply the **kernel trick.** *x* and *x'* in the input space *X*, the kernel function ***k(x,x')*** is expressed as inner product in space *V* to operate in higher-dimensional feature space.



ALGORITHMS – QUANTUM SVMs

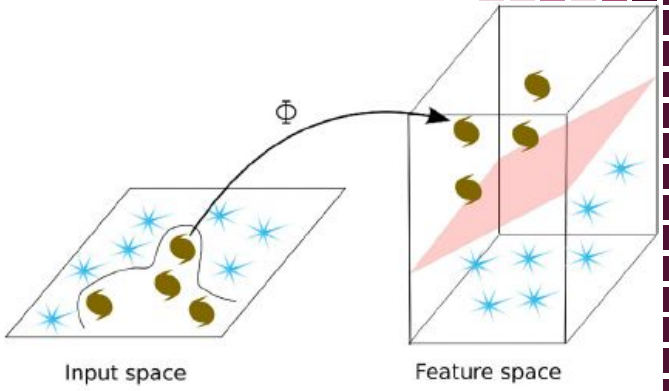
Takes the classical machine learning algorithm and performs the support vector machine on a quantum circuit in order to be efficiently processed on a quantum computer.

1. The quantum state space is used as a feature space. Data is mapped to a quantum state by applying the feature map circuit to a reference state.

2. Short-depth quantum circuit is applied to the feature state. The circuit with *l* layers is parameterized and optimized during training.

3. For a two label classification *{-1, +1}* a binary measurement is applied to the quantum state.

4. Decision rule: perform *R* repeated measurement shots to obtain the empirical distribution. The empirical risk is defined given by the error probability of assigning the incorrect label averaged over the samples in the training set *T*.



WORKFLOW PROCESS

Preprocessing

Divide the dataset into training and testing portions: 70% training and 30% testing.

Standardize the data set's features to fit into a normal distribution.

Use Principal Component Analysis (PCA) to reduce and fit the dataset dimensions into number of qubits

Scale the data between -1 and 1 to set the range for the Support Vector Machine model.

```
def breast_cancer(training_size, test_size, n, PLOT_DATA=True):
    class_labels = [r'Benign', r'Malignant']

    #testing classifier accuracy
    #training: 70%, testing: 30%
    X_train, X_test, Y_train, Y_test =
    train_test_split(cancer.data, cancer.target,
    test_size=0.3, random_state=109)

    #Standardize dataset features
    #to fit a normal distribution
    scaler = StandardScaler().fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
```

```
    #Use PCA to break down data from 30 to n dimensions
    #(finds patterns while keeping variation)
    pca = PCA(n_components=n).fit(X_train)
    X_train = pca.transform(X_train)
    X_test = pca.transform(X_test)

    # Scaling the data to be between -1 and 1
    samples = np.append(X_train, X_test, axis=0)
    minmax_scale = MinMaxScaler((-1, 1)).fit(samples)
    X_train = minmax_scale.transform(X_train)
    X_test = minmax_scale.transform(X_test)

    # Picking sample to train model from:
    training_input = {key: (X_train[Y_train == k, :])
    [:training_size] for k,key in enumerate(class_labels)}
    test_input = {key: (X_train[Y_train == k, :])[training_size:(
    training_size+test_size)] for k, key in enumerate(class_labels)}
```

WORKFLOW PROCESS

Algorithm Implementation

Set the dimensionality and number of qubits the circuit will have

Initialize the feature map in order to build the quantum SVM

Set the necessary parameters for the algorithm to train on, including the depth of the circuit, number of shots and initializing the pseudo-random number generator

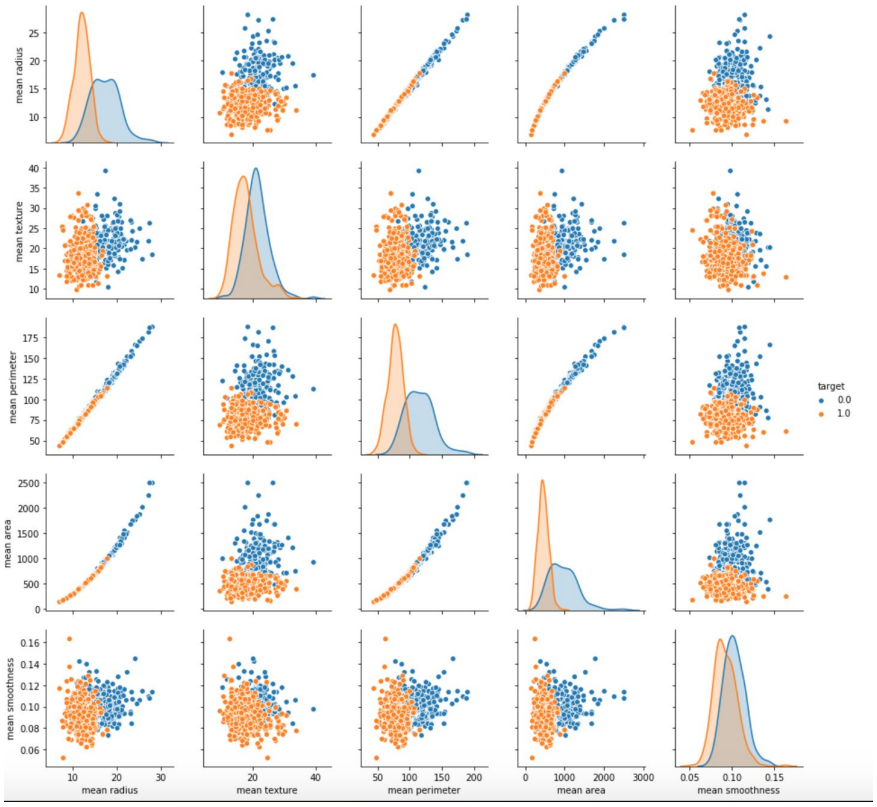
```
#split up data for algorithm input to be generated
n = 2 # number of features/qubits to use
training_dataset_size = 20
testing_dataset_size = 10

sample_Total, training_input, test_input,
class_labels = breast_cancer
(training_dataset_size, testing_dataset_size, n)

datapoints, class_to_label =
split_dataset_to_data_and_labels(test_input)
print(class_to_label)
```

```
temp = [test_input[k] for k in test_input]
total_array = np.concatenate(temp)

aqua_dict = {
    'problem': {'name': 'classification','
                random_seed': 100},
    'algorithm': {'name': 'QSVM'},
    'backend': {'provider': 'qiskit.BasicAer',
                'name': 'qasm_simulator', 'shots': 256},
    'feature_map': {'name': 'SecondOrderExpansion',
                'depth': 2, 'entanglement': 'linear'}
}

algo_input = ClassificationInput(training_input,
                                 test_input, total_array)
result = run_algorithm(aqua_dict, algo_input)

for k,v in result.items():
    print("'{}' : {}".format(k, v))
```

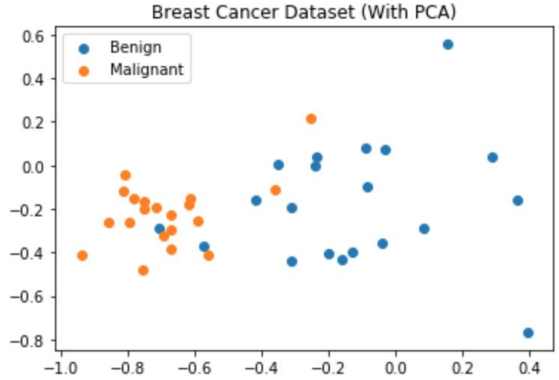
RESULTS

PREPROCESSING & PCA

CORRELATIONS BETWEEN DATA FEATURES

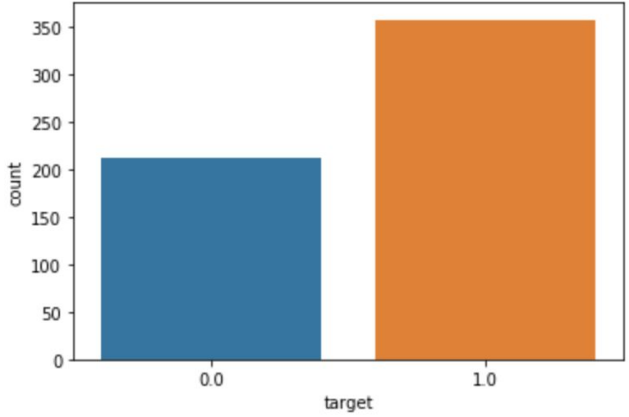


PCA REPRESENTATION




```
{'Benign': 0, 'Malignant': 1}
```

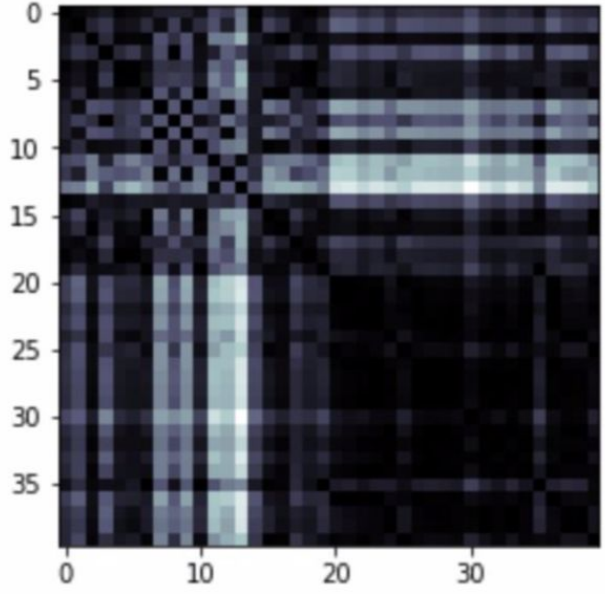
TARGET DISTRIBUTION



CLASSICAL SVM

KERNEL MATRIX

```
Testing success ratio: 0.85
```



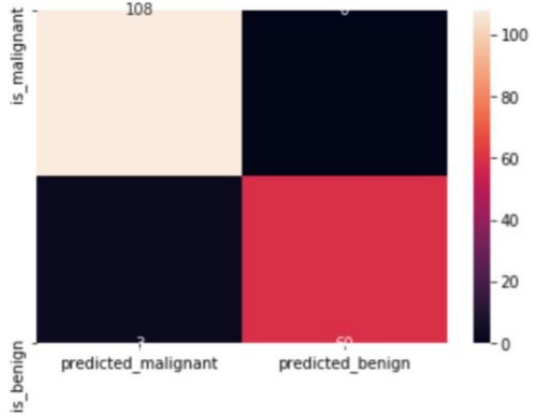
PERFORMANCE MEASURES

	precision	recall	f1-score	support
0.0	1.00	0.95	0.98	63
1.0	0.97	1.00	0.99	108
accuracy			0.98	171
macro avg	0.99	0.98	0.98	171
weighted avg	0.98	0.98	0.98	171

CORRELATION MATRIX

	predicted_malignant	predicted_benign
is_malignant	108	0
is_benign	3	60

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26ca1e10>
```

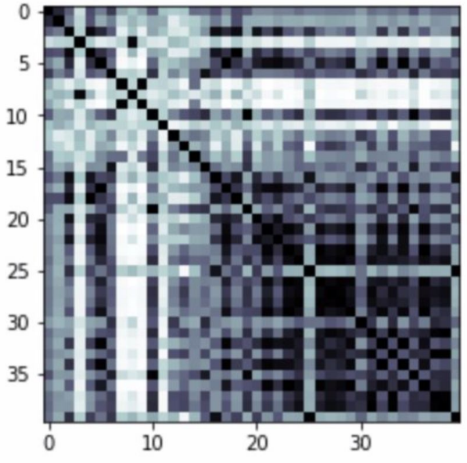


QUANTUM SVM ALGORITHM

```
'predicted_labels' : [0 1 0 0 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1]
'predicted_classes' : ['Benign', 'Malignant', 'Benign', 'Benign', 'Benign', 'Benign', 'Malign
ant', 'Malignant', 'Malignant', 'Benign', 'Malignant', 'Malignant', 'Benign', 'Malignant', 'M
alignant', 'Benign', 'Malignant', 'Malignant', 'Malignant', 'Malignant']
```

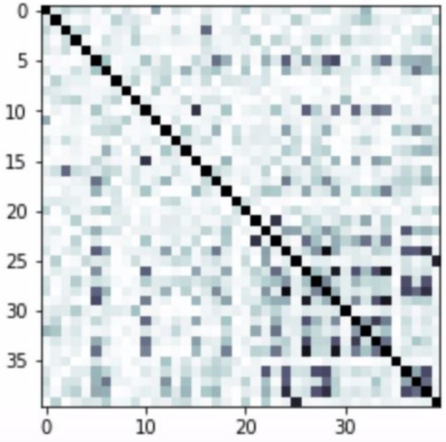
2 QUBITS KERNEL MATRIX

Testing success ratio: 0.9



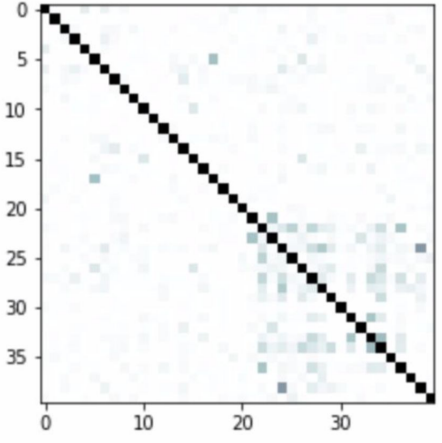
4 QUBITS KERNEL MATRIX

Testing success ratio: 0.75



8 QUBITS KERNEL MATRIX

Testing success ratio: 0.65

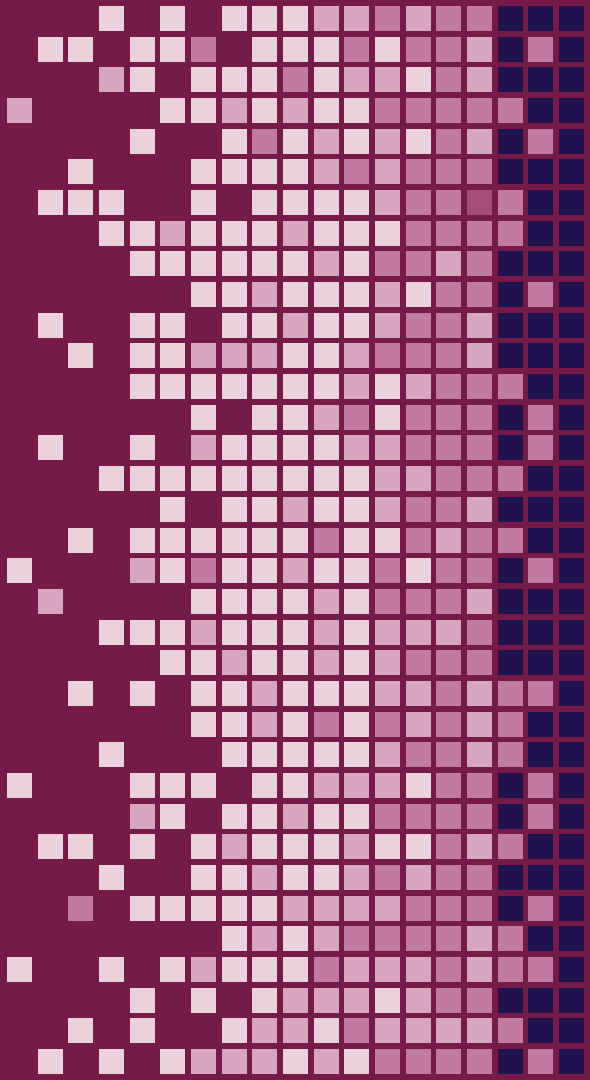


RESULTS OVERVIEW

Percentage of correctly classified breast cancer cells - Averages

	Classical SVM Algorithm	Quantum SVM - 2 Qubits	Quantum SVM- 4 Qubits	Quantum SVM- 8 Qubits
% of Correctly Classified Cells	**85%**	**90%**	**75%**	**65%**
Precision	88%	98%	78%	65%
Recall	85%	97%	72%	60%
F1 Score	79%	89%	64%	55%

DISCUSSION



RESULTS ANALYSIS

Highest number of accurately classified cells: **Quantum SVM Algorithm with a 2-Qubit Simulation**

1. The hypothesis is mostly supported by the experimental results. The quantum SVM algorithm method with a 2-qubit simulation on a QPU with IBM's quantum hardware yielded the highest accuracy when classifying breast cancer cells as benign and malignant

2. The quantum SVM algorithm with an 8-qubit simulation on a QPU with IBM's quantum hardware had the lowest yield, with the lowest classification accuracy in the breast cancer cells

3. The classical SVM algorithm ran on a CPU yielded higher classification accuracies than the 4-qubit and 8-qubit implementations with the quantum SVM algorithm, but was outperformed by the 2-qubit simulation

IMPLICATIONS

Next Steps

Implementation of other unsupervised quantum-enhanced algorithms including quantum convolutional neural networks (CNNs) for image classification and multivariate image analysis with implications in **self-diagnosis**. This project's demonstration was a proof of concept for quantum SVMs with labeled data for a binary output

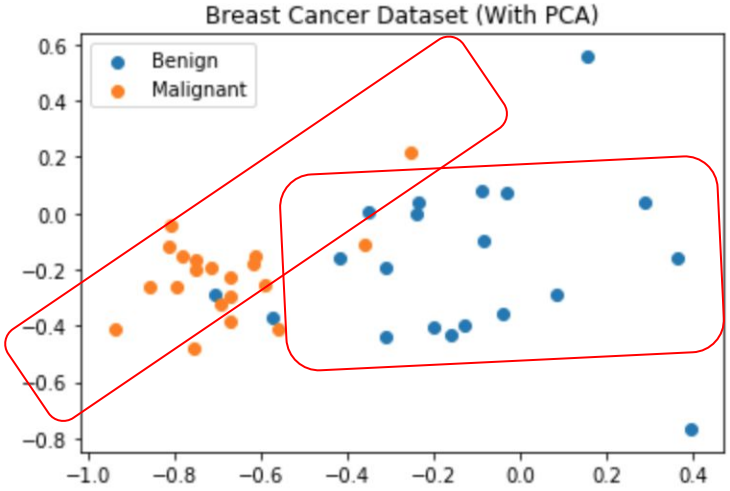
Quantum Machine Learning Applications

As hardware developments are being made for quantum error correction and noise reduction more enhanced clustering techniques for pattern recognition and image classification will be enabled. Implications range from medicine, drug discovery, genomics, and security. A few areas boosted:

- **Chemical Simulation:** Mapping out molecules and atoms for the creation of new materials
- **Quantum Matter Simulation:** Modeling molecular interactions at an atomic level, allowing new pharmaceuticals and medical research
- **Financial Applications:** simulation of the stochastic nature of financial markets for more *accurate* modeling and risk analysis.

APPENDIX

IN-DEPTH RESULTS ANALYSIS




```
{'Benign': 0, 'Malignant': 1}
```

- PCA goal: increase data interpretability + minimize information loss
- 5 parameters reduced to 2 principal components

Performance Measures

- True Positives (TP)
- True Negatives (TN)
- False Positives (FP)
- False Negatives (FN)

- Precision = TP / TP + FP
- Recall = TP / TP + FN
- F1 Score - Weighted Average of Precision and Recall

*The higher, the more accurate l

LIMITATIONS & BARRIERS

Challenge:

- Integrating IBM Quantum 2000Q Processor with local machine
- Pre-processing - encoding classical data into quantum states

Limitations:

- Current state-of-the art quantum processors subject to:
  - Decoherence - environment changes states of qubits into non-operational
  - Noise - random fluctuations of signals
  - Early-Stage Error Correction - protecting quantum information
- Production Challenges: cold temperatures (-273 degrees C), bulky, expensive, # of qubits limited → cannot be commercialized

DATASET DETAILS

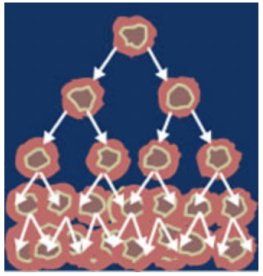
Dataset:

Breast Cancer Wisconsin (Diagnostic) Data Set

Download: Data Folder, Data Set Description

Abstract: Diagnostic Wisconsin Breast Cancer Database

Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1696313



Details:

- 31 different features for each cell nucleus
- ID number with the diagnosis type (M = malignant, B = benign)
- Diagnosis as benign / malignant used as target
- 5 / 31 features used for determining the diagnosis type when training model

QML BENEFITS

- **Main Task:** generate a quantum state describing the hyperplane with the matrix inversion algorithm, then classify the state
- **Algorithmic Complexity** - quantum SVM is logarithmic in feature size and number of training data
    - $O(\log d)$ qubits vs. $O(d)$ bits

Advantages:

- **Speed** - maximized when training data kernel matrix is dominated by a small number of principal components + Amplitude Encoding
- **Data privacy** - never requires explicit representation of all features of each of the training examples, but able to generate necessary data structure
    - Individual features of the training data are fully hidden from user once the kernel matrix is generated

SVMS PROCESSES

- **Main Task:** classify a vector into 1 / 2 classes given M training data points
- **Process:**
  - SVM finds a maximum-margin hyperplane with a normal vector that divides the 2 classes
  - The margin is given by 2 parallel hyperplanes separated by the max possible distance with no data points inside the hyperplane
- SVM fits under a class of problems where the number of equations are more than the number of unknowns → there is an overdetermined system of equations
- **Least Squares Fitting** - standard approach to approximately solve this regression analysis problem: minimizes sum of squares of residuals made in every single equation

SUPPLEMENTAL WORK

Full Code Implementation: https://github.com/aliceliu7/quantum-SVM

Implementation - Quantum SVMs for Binary Cell Classification:

https://medium.com/@aliceliu2004/quantum-support-vector-machines-a-new-era-of-ai-1262dd4b2c7e

BIBLIOGRAPHY

1) Havlicek, Vojtech, et al. "Supervised Learning with Quantum Enhanced Feature Spaces." *ArXiv.org*, 5 June 2018, arxiv.org/abs/1804.11326.
2) Mohseni, Masoud, et al. *Quantum Support Vector Machine for Big Data Classification*, American Physical Society, 1 Feb. 2014, dspace.mit.edu/handle/1721.1/90391.

CONTACT aliceliu2004@gmail.com