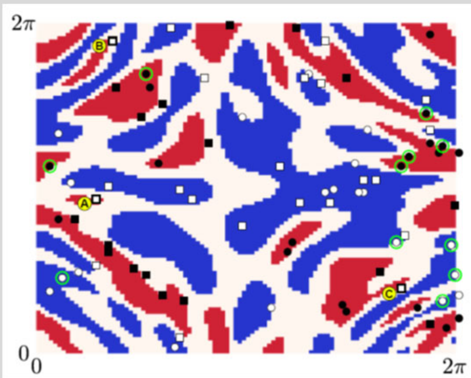


Machine Learning: Quantum SVM for Classification

ECE 592/CSC 591 – Fall 2018

NC STATE UNIVERSITY Electrical & Computer Engineering

@NCStateECE

Summary

Supervised learning with quantum enhanced feature spaces

Vojtech Havlicek[1],* Antonio D. Córcoles[1], Kristan Temme[1], Aram W. Harrow[2], Abhinav Kandala[1], Jerry M. Chow[1], and Jay M. Gambetta[1]
[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA and
[2] Center for Theoretical Physics, Massachusetts Institute of Technology, USA
(Dated: June 7, 2018)

- Applying quantum computation to Support Vector Machines
- Two approaches:
  - Quantum Variational Classification
    - Implement **feature map** as a quantum calculation, map value x to quantum state
    - Then apply **variational circuit** to implement classifier
  - Quantum Kernel Estimation
    - Use **kernel function** (inner products) instead of full feature set
    - Quantum estimation of kernel, which may be hard to compute classically
- Quantum advantage (potential):
  - Complex feature maps/kernels for better classification of high-dimensional data

Intro to Support Vector Machines

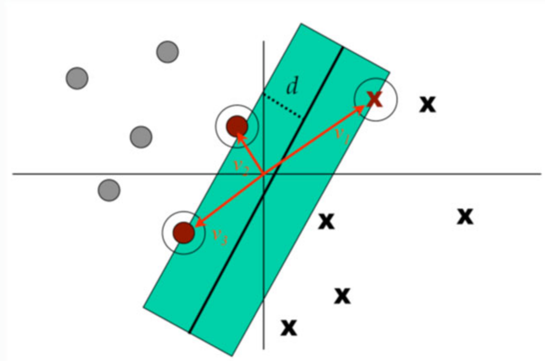
An Idiot's guide to Support vector machines (SVMs)

R. Berwick, Village Idiot

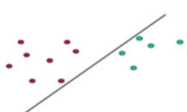
http://web.mit.edu/6.034/wwwbob/svm.pdf

Supervised learning
Construct (n-1)-dimensional hyperplane to separate n-dimensional data points

Support vectors help to determine optimal hyperplane



2-D Case



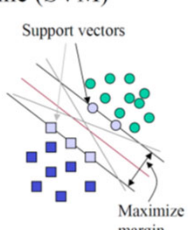
Find a,b,c, such that
$ax + by \geq c$ for red points
$ax + by \leq$ (or <) c for green points.

Which Hyperplane to pick?

- Lots of possible solutions for a,b,c.
- Some methods find a separating hyperplane, but not the optimal one (e.g., neural net)
- But: Which points should influence optimality?
  - All points?
    - Linear regression
    - Neural nets
  - Or only "difficult points" close to decision boundary
    - Support vector machines

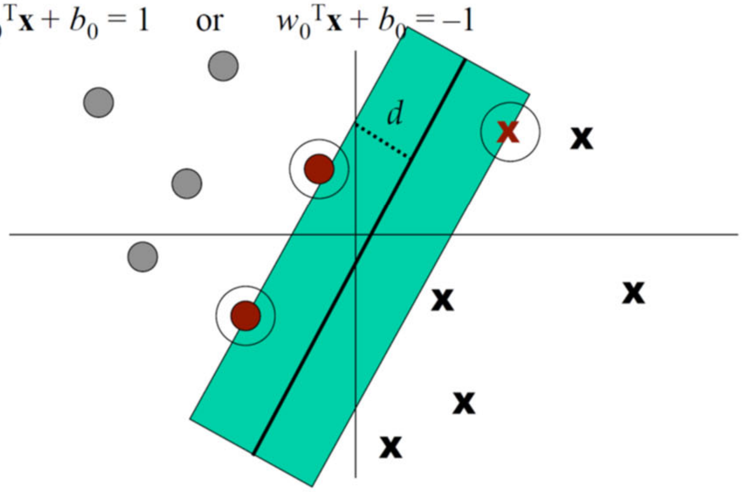
Support Vector Machine (SVM)

- SVMs maximize the margin (Winston terminology: the 'street') around the separating hyperplane.
- The decision function is fully specified by a (usually very small) subset of training samples, the support vectors.
- This becomes a Quadratic programming problem that is easy to solve by standard methods



Support Vectors: Input vectors that just touch the boundary of the margin (street) – circled below, there are 3 of them (or, rather, the 'tips' of the vectors

$w_0{}^{T}\mathbf{x} + b_0 = 1$ or $w_0{}^{T}\mathbf{x} + b_0 = -1$



Support vectors are the closest together, the hardest to classify.

Maximization problem to find hyperplane with maximum distance (d).

Primal problem:

$$\min L_P = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) + \sum_{i=1}^{l} a_i$$

s.t. $\forall i \ a_i \geq 0$

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

Dual problem:

$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2}\sum_{i=1}^{l} a_i a_j y_i y_j \boxed{\left(\mathbf{x}_i \cdot \mathbf{x}_j\right)}$$

s.t. $\sum_{i=1}^{l} a_i y_i = 0 \ \& \ a_i \geq 0$

(note that we have removed the dependence on $\mathbf{w}$ and b)

Note the inner product. Important later

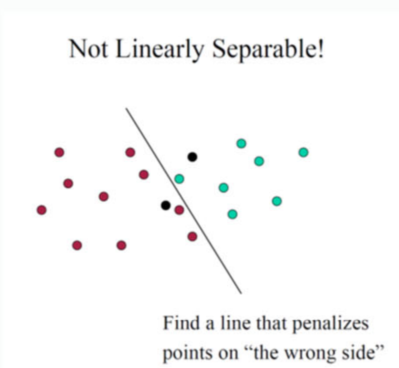
Now knowing the a_i we can find the weights **w** for the maximal margin separating hyperplane:

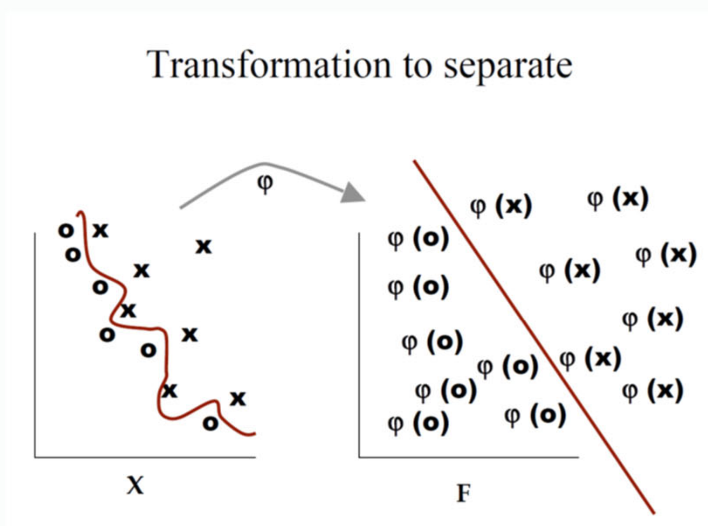
$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i$$

And now, after training and finding the **w** by this method, given an unknown point u measured on features x_i we can classify it by looking at the sign of:

$$f(x) = \mathbf{w} \bullet \mathbf{u} + b = (\sum_{i=1}^{l} a_i y_i \mathbf{x_i} \bullet \mathbf{u}) + b$$

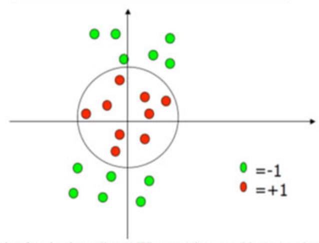
Remember: most of the weights $\mathbf{w_i}$, i.e., the a's, will be zero
Only the support vectors (on the gutters or margin) will have nonzero weights or a's – this reduces the dimensionality of the solution





This transformation is known as a **feature map**.

Problems with linear SVM



What if the decision function is not linear? What transform would separate these?

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y}+1)^p$$

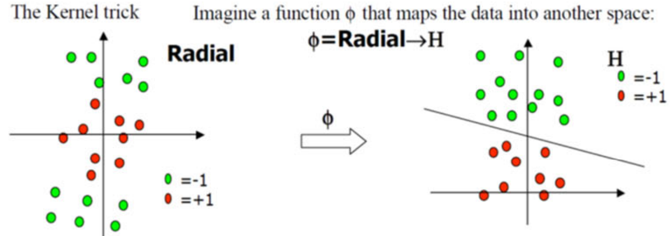
$$K(\mathbf{x},\mathbf{y}) = \exp\left\{-\|\mathbf{x}-\mathbf{y}\|^2 / 2\sigma^2\right\}$$

$$K(\mathbf{x},\mathbf{y}) = \tanh(\kappa\mathbf{x}\cdot\mathbf{y}-\delta)$$

1st is polynomial (includes x•x as special case)
2nd is radial basis function (gaussians)
3rd is sigmoid (neural net activation function)

Ans: polar coordinates!
Non-linear SVM

The Kernel trick   Imagine a function ϕ that maps the data into another space:



Remember the function we want to optimize: $L_d = \sum a_i - ½\sum a_i a_j y_i y_j (x_i \bullet x_j)$ where $(x_i \bullet x_j)$ is the dot product of the two feature vectors. If we now transform to ϕ, instead of computing this dot product $(x_i \bullet x_j)$ we will have to compute $(\phi(x_i) \bullet \phi(x_j))$. But how can we do this? This is expensive and time consuming (suppose ϕ is a quartic polynomial… or worse, we don't know the function explicitly. Well, here is the neat thing:
If there is a "kernel function" K such that $K(x_i,x_j) = \phi(x_i) \bullet \phi(x_j)$, then we do not need to know or compute ϕ at all!! That is, the kernel function defines inner products in the transformed space. Or, it defines similarity in the transformed space.

Summary of SVM

- Support vectors: small set of training vectors that are closest together
- SVs determine the optimal hyperplane for binary classification
- Non-linear SVM:
  - Feature map = mapping to higher-dimension space, which can be linearly separated
  - Kernel = function that yields inner products of vectors in the feature space, without having to directly calculate the feature map transformation

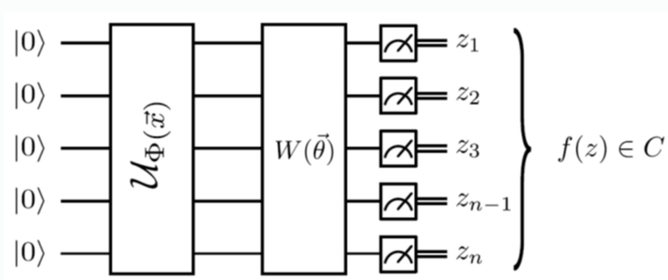
Quantum SVM

Supervised learning with quantum enhanced feature spaces

Vojtech Havlicek[1],* Antonio D. Córcoles[1], Kristan Temme[1], Aram W. Harrow[2], Abhinav Kandala[1], Jerry M. Chow[1], and Jay M. Gambetta[1]
[1]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA and
[2]Center for Theoretical Physics, Massachusetts Institute of Technology, USA
(Dated: June 7, 2018)

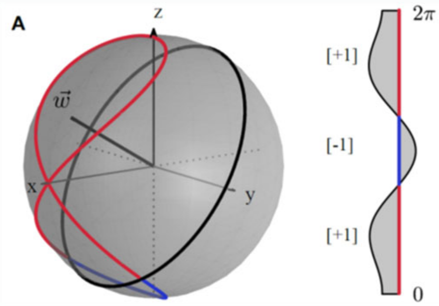
- Two approaches:
  - Quantum Variational Classification
    - Implement **feature map** as a quantum calculation, map value x to quantum state
    - Then apply **variational circuit** to implement classifier
  - Quantum Kernel Estimation
    - Use **kernel function** (inner products) instead of full feature set
    - Quantum estimation of kernel, which may be hard to compute classically

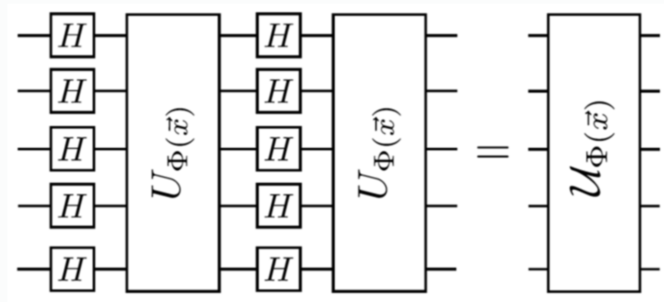
Quantum Variational Classifier



(1) Encode data vector x into quantum state $\Phi(x)$, which represents the feature map.
(2) Apply variational transform that defines the separating hyperplane.
(3) Measure the qubits to get a binary result.
(4) Multiple shots to get estimated value = classification.

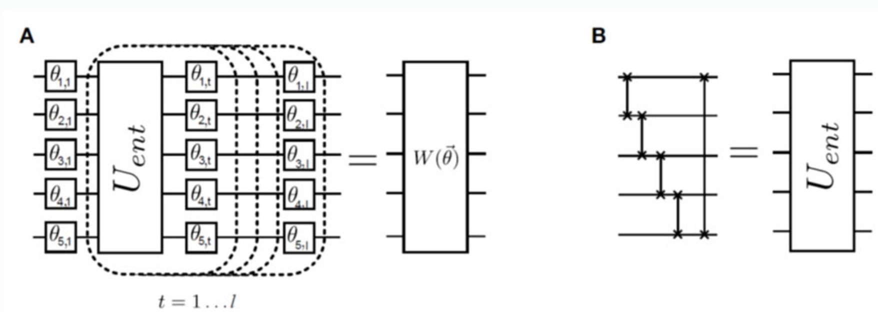
Encoding the feature map





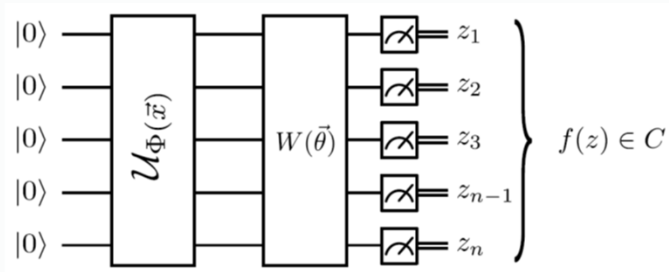
Create entangling feature map, hard to estimate kernel function classically.
Uses only single- and two-qubit gates.

Classifier circuit



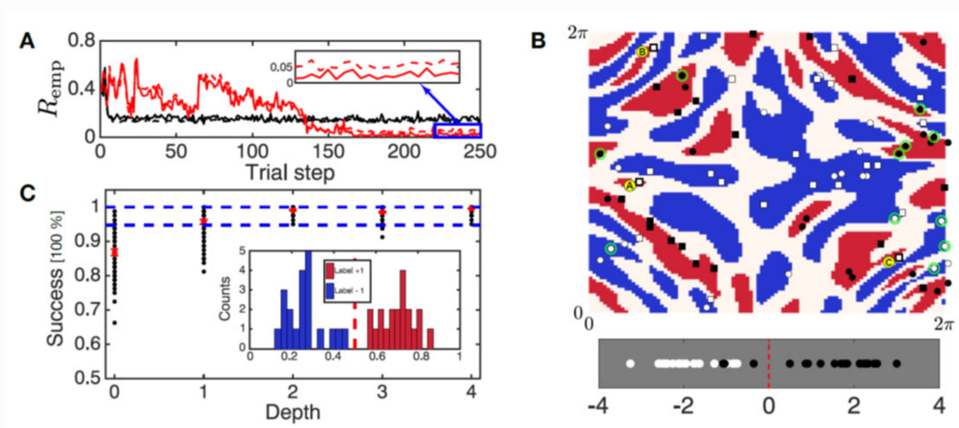
During training, values of θ are found which minimize misclassification.
During classification, θ are fixed.

Binary Classification



Construct empirical distribution $\hat{p}_y(\vec{s}) = \mathrm{r}_y R^{-1}$.
Set label $= \mathrm{argmax}_y \{\hat{p}_y(\vec{s})\}$

Empirical Results

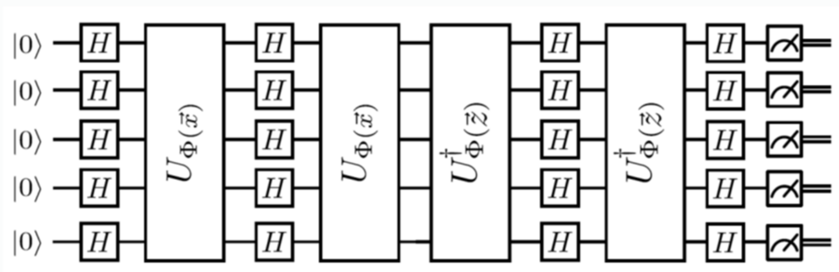


Circles are training data.
Squares are classifications.
Green circles are support vectors.

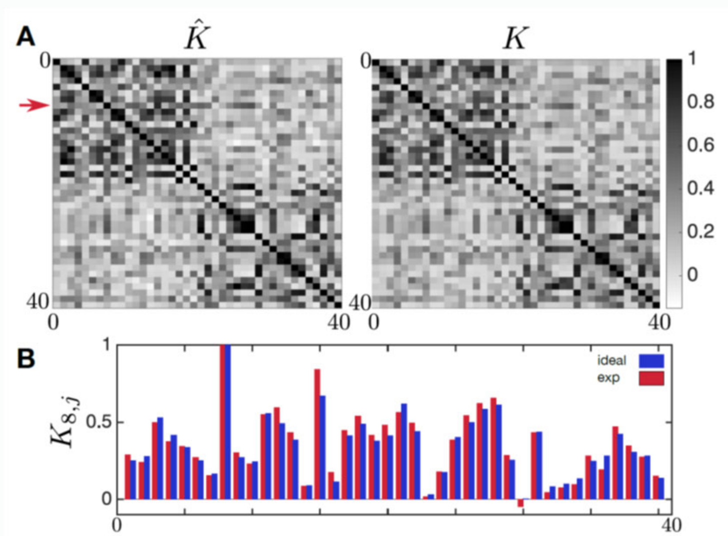
Depth of variational circuit (W).
Black dots are classification accuracies, red dot is avg.
Blue lines are accuracies of QKE (next slides).

Quantum Kernel Estimation

- Kernel is measure of similarities between vectors (inner products).
- Quantum estimator is **transition probability** between states.
- Measure, and count the number of all-zero outputs.
  The frequency of this output is estimate of transition probability.



Kernel Estimate Accuracy

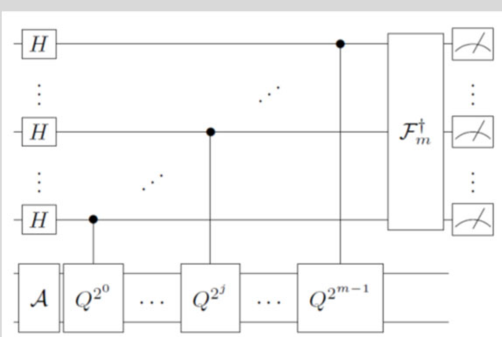


Summary

Supervised learning with quantum enhanced feature spaces

Vojtech Havlicek[1],* Antonio D. Córcoles[1], Kristan Temme[1], Aram W. Harrow[2], Abhinav Kandala[1], Jerry M. Chow[1], and Jay M. Gambetta[1]
[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA and
[2] Center for Theoretical Physics, Massachusetts Institute of Technology, USA
(Dated: June 7, 2018)

- Applying quantum computation to Support Vector Machines
- Two approaches:
  - Quantum Variational Classification
    - Implement **feature map** as a quantum calculation, map value x to quantum state
    - Then apply **variational circuit** to implement classifier
  - Quantum Kernel Estimation
    - Use **kernel function** (inner products) instead of full feature set
    - Quantum estimation of kernel, which may be hard to compute classically
- Quantum advantage (potential):
  - Complex feature maps/kernels for better classification of high-dimensional data



Quantum Risk Analysis

ECE 592/CSC 591 – Fall 2018

Summary

Quantum Risk Analysis

Stefan Woerner[1,*] and Daniel J. Egger[1]
[1]IBM Research – Zurich
(Dated: June 20, 2018)

- Given: Loss/profit probability distribution of portfolio
- Estimate various quantities:
  - Expected value, Value at risk, Conditional value at risk
- Classical Approach = Monte Carlo simulation
  - With M samples, error scales as $1/\sqrt{M}$
- Quantum Approach = Amplitude Estimation
  - Error scales as $1/M$
  - Quadratic speedup

Amplitude Estimation

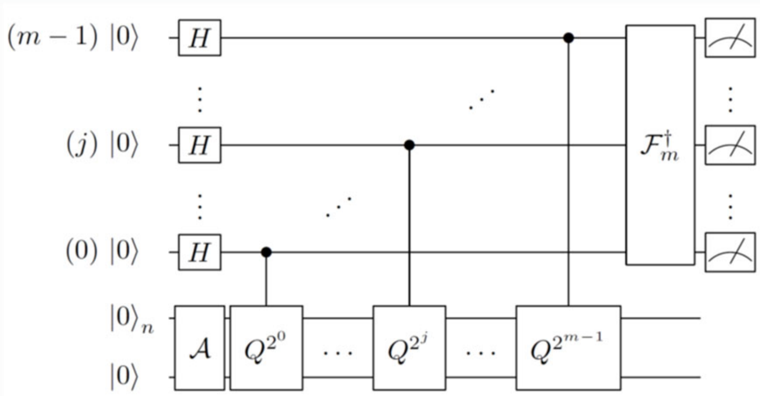
- Suppose we have a transform A, such that:

$$A|\psi\rangle_{(n+1)} = \sqrt{1-a}|\psi_0\rangle_n|0\rangle + \sqrt{a}|\psi_1\rangle_n|1\rangle$$

- Amplitude estimation provides an estimate of a, e.g., the probability of measuring a 1 in the last bit.

- Brassard, Hoyer, Mosca, and Tapp (2000).

Amplitude Estimation



Requires m additional qubits, and $M = 2^m$ applications of Q, which is related to A (next slide), and inverse QFT. Measured output is y and estimator $\tilde{a} = \sin^2(y\,\pi/M)$.

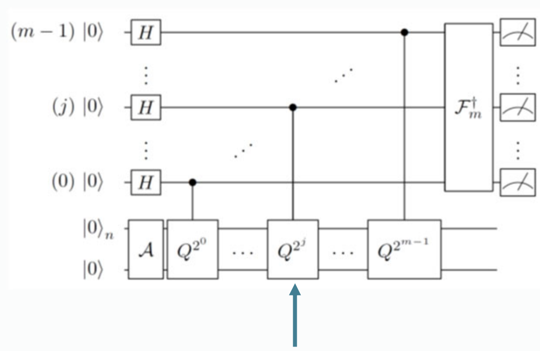
What is Q?

Appendix A: Q-Operator

For a given circuit $\mathcal{A}$ acting on $n+1$ qubits, the corresponding Q-operator used in amplitude estimation is defined as [8]

$$Q = \mathcal{A}(\mathbb{I} - 2\,|0\rangle_{n+1}\langle 0|_{n+1})\mathcal{A}^\dagger (\mathbb{I} - 2\,|\psi_0\rangle_n |0\rangle \langle\psi_0|_n \langle 0|),$$

where $\mathbb{I}$ denotes the identity operator. If $n = 0$, as e.g. considered in Sec. IV, the reflections defining Q reduce to the Pauli Z-operators and Q simplifies to $\mathcal{A}Z\mathcal{A}^\dagger Z$. In addition, if $\mathcal{A} = R_y(\theta)$ then it can be easily seen that $Q = R_y(2\theta)$.



Expected value

$$|\psi\rangle_n = \sum_{i=0}^{N-1} \sqrt{p_i}\, |i\rangle_n$$

One of the N values of a random variable X.
Represents discretized interest rate, or value of a portfolio.

$$F : |i\rangle_n |0\rangle \mapsto |i\rangle_n \left(\sqrt{1 - f(i)}\, |0\rangle + \sqrt{f(i)}\, |1\rangle \right)$$

Applied to state above...

$$\sum_{i=0}^{N-1} \sqrt{1 - f(i)} \sqrt{p_i}\, |i\rangle_n |0\rangle + \sum_{i=0}^{N-1} \sqrt{f(i)} \sqrt{p_i}\, |i\rangle_n |1\rangle$$

Amplitude estimation = $\sum_{i=0}^{N-1} p_i f(i)$ = $\mathbb{E}[f(X)]$

$f(i) = (\frac{i}{N-1})$ yields $E[\frac{X}{N-1}]$ and thus $E[X]$.

$f(i) = (\frac{i^2}{(N-1)^2})$ yields $E[X^2]$

Var(X) = $E[X^2] - E[X]^2$

Constructions

- State representing distribution $|\psi\rangle_n = \sum_{i=0}^{N-1} \sqrt{p_i}\, |i\rangle_n$
  - In general, requires 2^n gates.
    But approximations are polynomial in n for many distributions.
- Transforms for functions
  - General construction of

    $$P : |x\rangle_n |0\rangle \mapsto |x\rangle_n (\cos(p(x))\, |0\rangle + \sin(p(x))\, |1\rangle)$$

    for k-order polynomical p(x) using $O(n^{k+1})$ gates and $O(n)$ ancillas
  - Paper describes finding polynomials to enable f(x) shown on previous slide.

Constructions

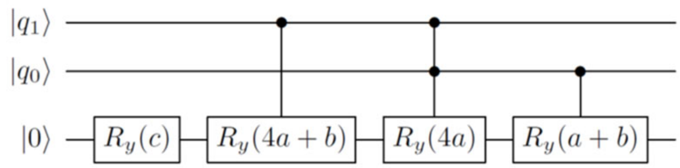


FIG. 2: Quantum circuit realizing $|x\rangle_n |0\rangle \mapsto |x\rangle_n (\cos(p(x)/2)|0\rangle + \sin(p(x)/2)|1\rangle)$ for $p(x) = (ax^2 + bx + c)$ and $x \in \{0, 1, 2, 3\}$. Exploiting $x = (2q_1 + q_0)$ and $q_i^2 = q_i$ leads to $p(x) = (4a+b)q_1 + 4aq_0q_1 + (a+b)q_0 + c$, which can be directly mapped to a circuit. R_y denotes a Y-rotation.

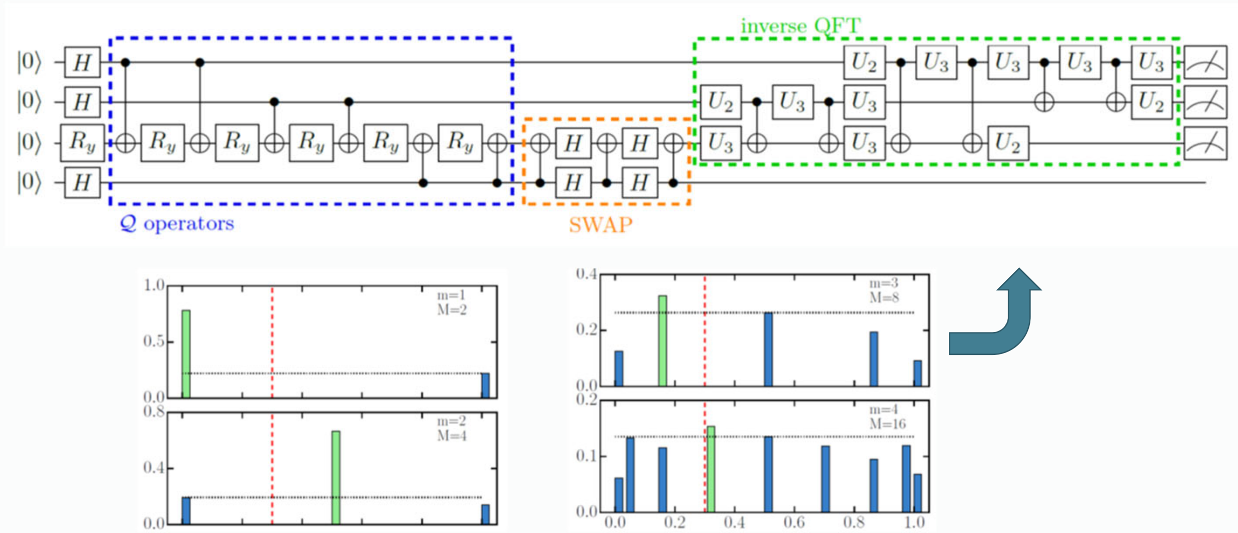
T-Bill Model, Binomial Tree

- Value of T-Bill today, given that rate may change in next time period.

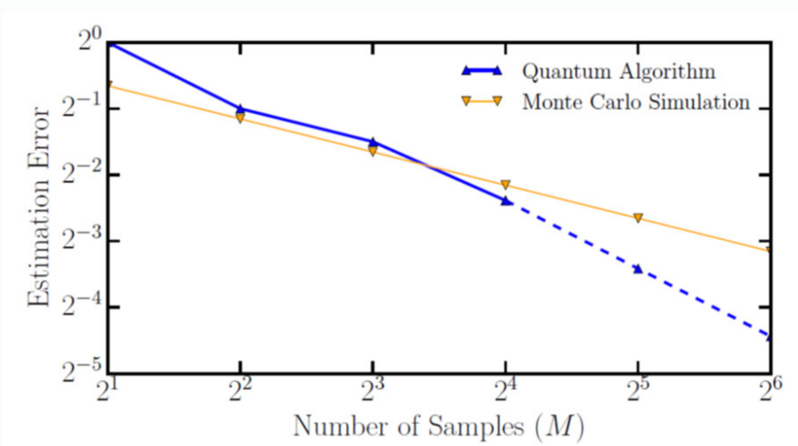
$$V = \frac{(1-p)V_F}{1+r+\delta r} + \frac{pV_F}{1+r} = (1-p)V_{\text{low}} + pV_{\text{high}}$$

- Only need one qubit to represent uncertainty.
- $A = R_y\,(\theta_p)$ where $\theta_p = 2/\sin(\sqrt{p})$

Quantum Circuit and Results



Error vs. Monte Carlo



Two-Asset Portfolio

$$V(r_1, r_2) = \frac{V_{F_1}}{1 + r_1} + \sum_{i=1}^{4} \frac{cV_{F_2}}{(1 + r_2/2)^i} + \frac{V_{F_2}}{(1 + r_2/2)^4}$$

Using 3 qubits for shift (S) and
2 qubits for twist (T) components of risk.

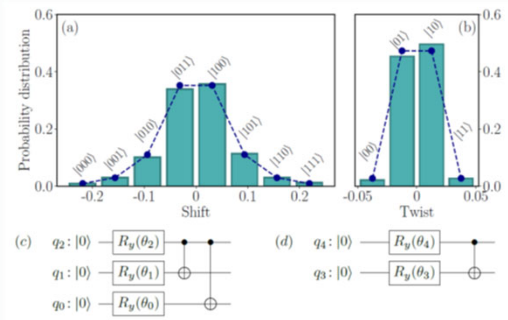


FIG. 8: (a) 8-bin histogram of historical shift data (bars) as well as fitted distribution (dashed line). (b) 4-bin histogram of historical twist data (bars) as well as fitted distribution (dashed line). In both cases the labels show the quantum state that will occur with the corresponding probability. (c) and (d) show the quantum circuits used to load the distributions of (a) and (b), respectively, into the quantum computer.

Results

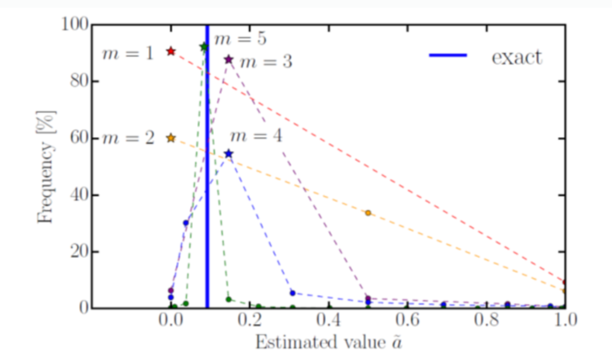


FIG. 9: VaR estimated through a simulation of a perfect quantum computer. As the number of sample qubits m is increased the quantum estimated VaR approaches the classical value indicated by the vertical blue line. The dashed lines are intended as guides to the eye. The stars indicate the most probable values.

			#CX		
m	M	#qubits	all-to-all	IBM Q 20	overhead
1	2	13	795	1'817	2.29
2	4	14	2'225	5'542	2.49
3	8	15	5'085	12'691	2.50
4	16	16	10'803	26'457	2.45
5	32	17	22'235	55'520	2.50

Summary

Quantum Risk Analysis

Stefan Woerner[1,*] and Daniel J. Egger[1]
[1] IBM Research – Zurich
(Dated: June 20, 2018)

- Given: Loss/profit probability distribution of portfolio
- Estimate various quantities:
  - Expected value, Value at risk, Conditional value at risk

- Classical Approach = Monte Carlo simulation
  - With M samples, error scales as $1/\sqrt{M}$
- Quantum Approach = Amplitude Estimation
  - Error scales as $1/M$
  - Quadratic speedup