Some Results in Quantum Learning Theory

Faris Sbahi

3/5/19

Fine Print

- In general, quantum machine learning algorithms convert quantum input states to the desired quantum output states.
- In practice, data is initially stored classically and the algorithm's output must be accessed classically as well.
- Highlight: A practical way to make comparisons between classical and quantum algorithms is to analyze classical algorithms under ℓ^2 sampling conditions
- Tang: linear algebra problems in low-dimensional spaces (say constant or polylogarithmic) likely can be solved "efficiently" under these conditions
- Many of the initial practical applications of quantum machine learning were to problems of this type (e.g. Quantum Recommendation Systems - Kerendis, Prakash, 2016)

In search of a "fair" comparison

- How can we compare the speed of quantum algorithms with quantum input and quantum output to classical algorithms with classical input and classical output?
- Quantum machine learning algorithms can be exponentially faster than the best standard classical algorithms for similar tasks, but quantum algorithms get help through input state preparation.
- Want a practical classical model that helps its algorithms offer similar guarantees to quantum algorithms, while still ensuring that they can be run in nearly all circumstances one would run the quantum algorithm.

In search of a "fair" comparison

- How can we compare the speed of quantum algorithms with quantum input and quantum output to classical algorithms with classical input and classical output?
- Quantum machine learning algorithms can be exponentially faster than the best standard classical algorithms for similar tasks, but quantum algorithms get help through input state preparation.
- Want a practical classical model that helps its algorithms offer similar guarantees to quantum algorithms, while still ensuring that they can be run in nearly all circumstances one would run the quantum algorithm.
- Solution (Tang): compare quantum algorithms with quantum state preparation to classical algorithms with sample and query access to input.

Classical ℓ^2 Sampling Model

Definition

We have "query access" to $x \in \mathbb{C}^n$ if, given $i \in [n]$, we can efficiently compute x_i. We say that $x \in \mathcal{Q}$.

Definition

We have sample **and** query access to $x \in \mathbb{C}^n$ if

1. We have query access to x i.e. $x \in \mathcal{Q}$ ($\Rightarrow \mathcal{SQ} \subset \mathcal{Q}$)
2. can produce independent random samples $i \in [n]$ where we sample i with probability $|x_i|^2/\|x\|^2$ and can query for $\|x\|$.

We say that $x \in \mathcal{SQ}$.

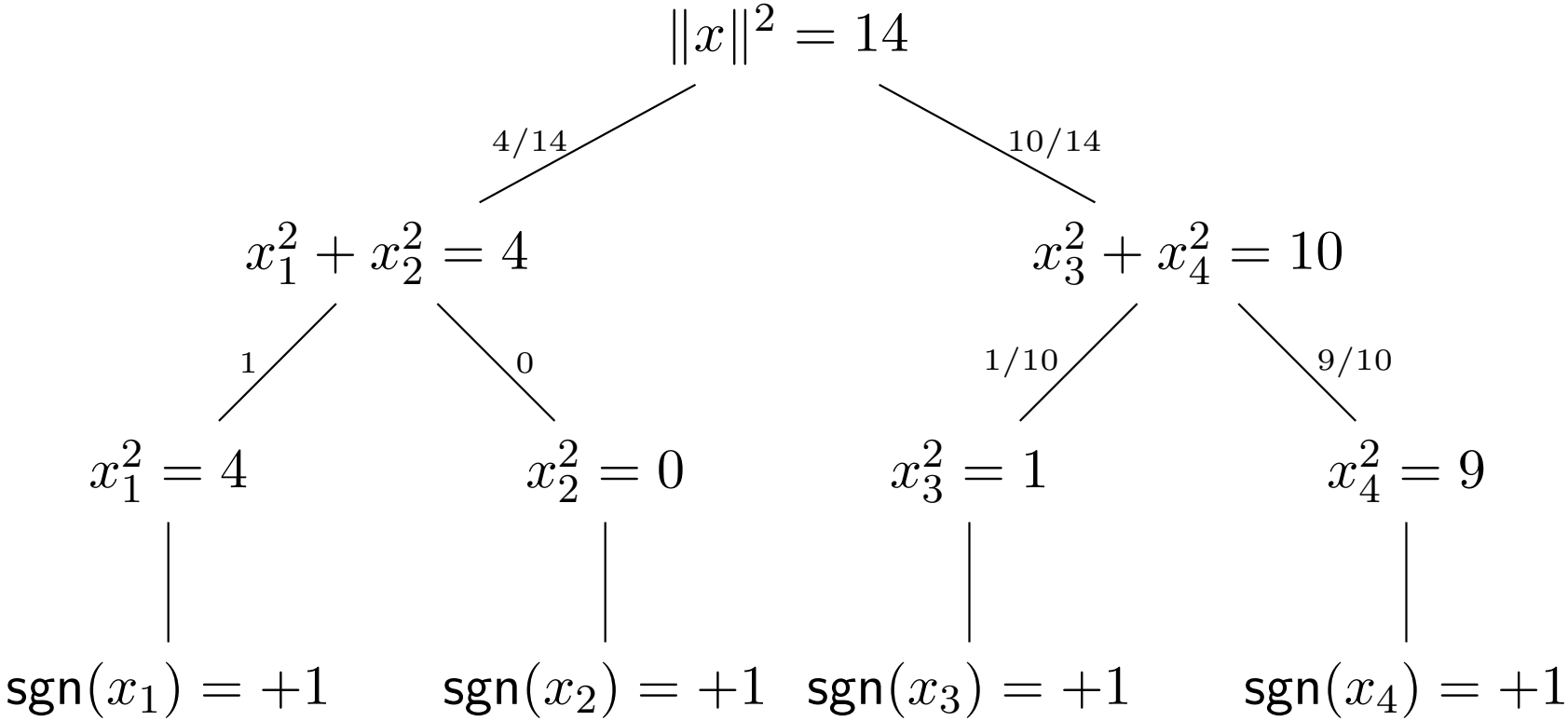
Definition

For $A \in \mathbb{C}^{m \times n}$, $A \in \mathcal{SQ}$ (abuse) if

1. $A_i \in \mathcal{SQ}$ where A_i is the ith row of A
2. $\tilde{A} \in \mathcal{SQ}$ for $\tilde{A}$ the vector of row norms (so $\tilde{A}_i = \|A_i\|$).

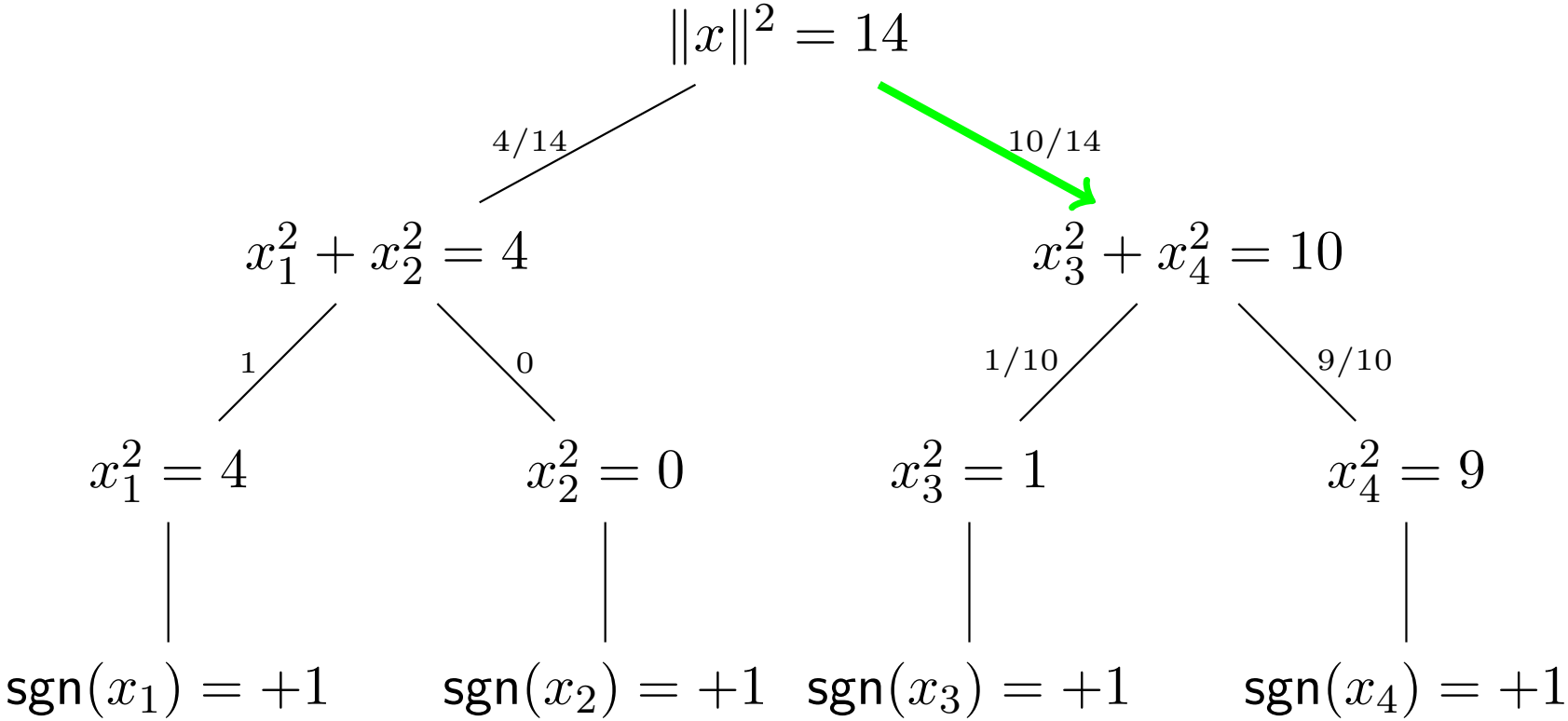
Example Data Structure

Say we have the vector $\vec{x} = (2, 0, 1, 3)$ and $\vec{x} \in \mathcal{SQ}$. Consider the following binary tree data structure.



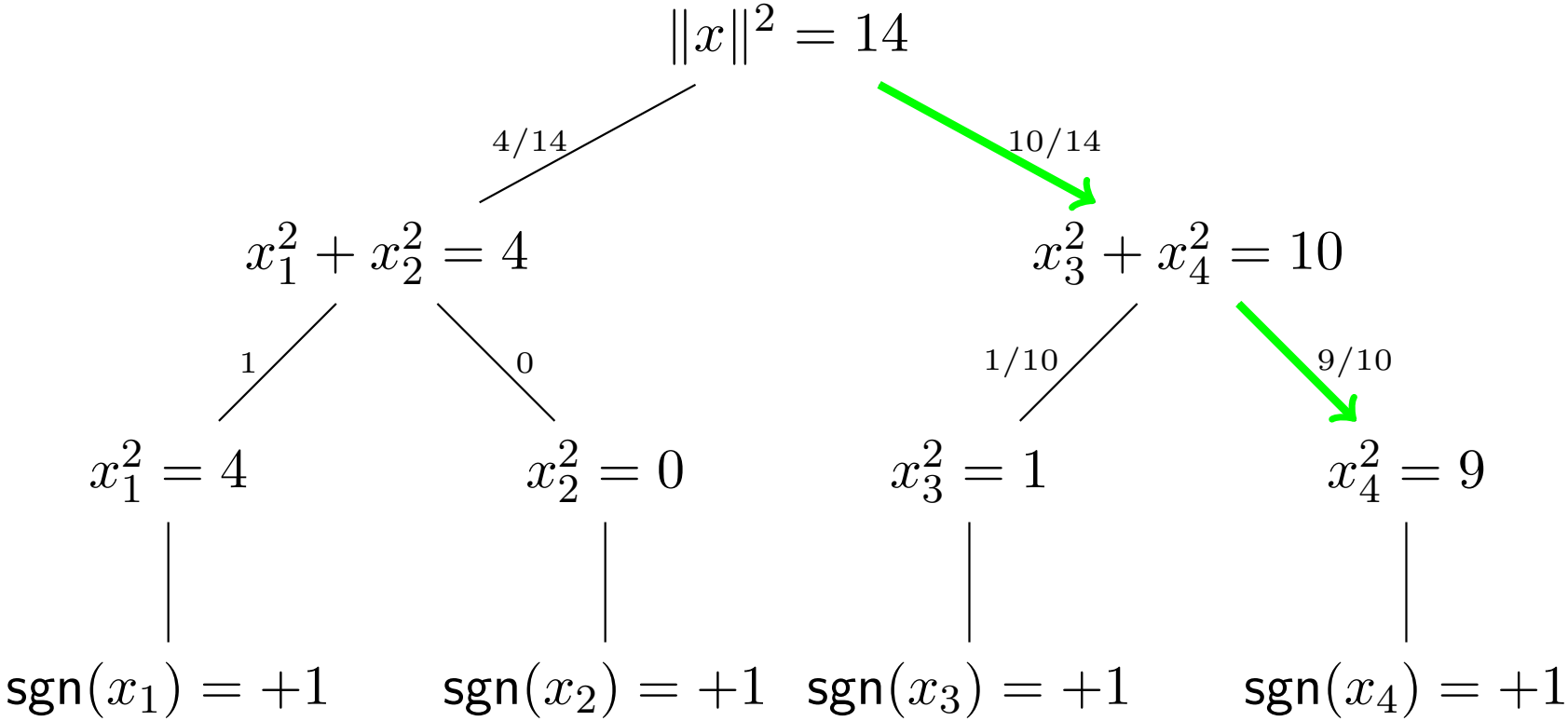
Example Data Structure

Say we have the vector $\vec{x} = (2, 0, 1, 3)$ and $\vec{x} \in \mathcal{SQ}$. Consider the following binary tree data structure.



Example Data Structure

Say we have the vector $\vec{x} = (2, 0, 1, 3)$ and $\vec{x} \in \mathcal{SQ}$. Consider the following binary tree data structure.



Dequantization Toolbox

Method 1: Inner product estimation (Tang, 2018)

- For $x, y \in \mathbb{C}^n$, if we are given that $x \in \mathcal{SQ}$ and $y \in \mathcal{Q}$, then we can estimate $\langle x, y \rangle$ with probability $\geq 1 - \delta$ and error $\epsilon \|x\| \|y\|$

Dequantization Toolbox

Method 1: Inner product estimation (Tang, 2018)

- For $x, y \in \mathbb{C}^n$, if we are given that $x \in \mathcal{SQ}$ and $y \in \mathcal{Q}$, then we can estimate $\langle x, y \rangle$ with probability $\geq 1 - \delta$ and error $\epsilon \|x\| \|y\|$
- Quantum analog: SWAP test

Dequantization Toolbox

Method 1: Inner product estimation (Tang, 2018)

Fact

For $\{X_{i,j}\}$ i.i.d random variables with mean μ and variance σ^2, let

$$Y := \underset{j \in [\log 1/\delta]}{\text{median}} \ \underset{i \in [1/\epsilon^2]}{\text{mean}} \ X_{i,j}$$

Then $|Y - \mu| \leq \epsilon\sigma$ with probability $\geq 1 - \delta$, using only $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ samples.

- In words: We may create a mean estimator from $1/\epsilon^2$ samples of X. We compute the median of $\log 1/\delta$ such estimators

Dequantization Toolbox

Method 1: Inner product estimation (Tang, 2018)

Fact

For $\{X_{i,j}\}$ i.i.d random variables with mean μ and variance σ^2, let

$$Y := \underset{j \in [\log 1/\delta]}{\text{median}} \; \underset{i \in [1/\epsilon^2]}{\text{mean}} \; X_{i,j}$$

Then $|Y - \mu| \leq \epsilon\sigma$ with probability $\geq 1 - \delta$, using only $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ samples.

- In words: We may create a mean estimator from $1/\epsilon^2$ samples of X. We compute the median of $\log 1/\delta$ such estimators
- Catoni (2012) shows that Chebyshev's inequality is the best guarantee one can provide when considering pure empirical mean estimators for an unknown distribution (and finite μ, σ)
- "Median of means" provides an exponential improvement in probability of success $(1 - \delta)$ guarantee

Dequantization Toolbox

Method 1: Inner product estimation (Tang, 2018)

Corollary

For $x, y \in \mathbb{C}^n$, given $x \in \mathcal{SQ}$ and $y \in \mathcal{Q}$, we can estimate $\langle x, y \rangle$ to $\epsilon \|x\| \|y\|$ error with probability $\geq 1 - \delta$ with query complexity $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$

Dequantization Toolbox

Method 1: Inner product estimation (Tang, 2018)

Corollary

For $x, y \in \mathbb{C}^n$, given $x \in \mathcal{SQ}$ and $y \in \mathcal{Q}$, we can estimate $\langle x, y \rangle$ to $\epsilon \|x\| \|y\|$ error with probability $\geq 1 - \delta$ with query complexity $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$

Proof.

Sample an **index** s from x. Then, define $Z := x_s y_s \frac{\|y\|^2}{|y_s|^2}$. Apply the Fact with $X_{i,j}$ being independent samples Z. $\square$

Dequantization Toolbox

Method 2: Thin Matrix-Vector (Tang, 2018)

- For $V \in \mathbb{C}^{n \times k}, w \in \mathbb{C}^k$, given $V^\dagger \in \mathcal{SQ}$ (*column*-wise sampling of V) and $w \in \mathcal{Q}$, we can simulate $Vw \in \mathcal{SQ}$ with poly(k) queries
- In words: if we can least-square sample the columns of matrix V and query the entries of vector w, then
  1. We can query entries of their multiplication (Vw)
  2. We can least-square sample from a distribution that emulates their multiplication
- Hence, as long as $k \ll n$, we can perform each using a number of steps polynomial in the number of columns of V.

Dequantization Toolbox

Method 2: Thin Matrix-Vector (Tang, 2018)

Definition

Rejection sampling

Algorithm

Input: Samples from distribution P

Output: Samples from distribution Q

- *Sample* s *from* P
- *Compute* $r_s = \frac{1}{N}\frac{Q(s)}{P(s)}$, *for fixed constant* N
- *Output* s *with probability* r_s *and restart otherwise*

Fact

Fact. If $r_i \leq 1, \forall i$, *then the above procedure is well-defined and outputs a sample from* Q *in* N *iterations in expectation.*

Dequantization Toolbox

Method 2: Thin Matrix-Vector (Tang, 2018)

Proposition

For $V \in \mathbb{R}^{n \times k}$ and $w \in \mathbb{R}^k$, given $V^\dagger \in \mathcal{SQ}$ and $w \in \mathcal{Q}$, we can simulate $Vw \in \mathcal{SQ}$ with expected query complexity $\tilde{O}((\frac{1}{\epsilon^2} \log \frac{1}{\delta}))$
We can compute entries $(Vw)_i$ with $O(k)$ queries.
We can sample using rejection sampling:

- *P is the distribution formed by sampling from $V_{(\cdot,j)}$.*
- *Q is the target Vw.*
- *Hence, compute r_s to be a constant factor of Q/P*

$$r_i = \frac{\|w^T V_{\cdot,i}\|^2}{\|w\|^2 \|V_{\cdot,i}\|^2}$$

Dequantization Toolbox

Method 2: Thin Matrix-Vector (Tang, 2018)

- Notice that we can compute these r_i's (in fact, despite that we cannot compute probabilities from the target distribution), and that the rejection sampling guarantee is satisfied (via Cauchy-Schwarz).
- Since the probability of success is $\|Vw\|^2/\|w\|^2$, it suffices to estimate the probability of success of this rejection sampling process to estimate this norm.
- Through a Chernoff bound, we see that the average of $O(\|w\|^2(\frac{1}{\epsilon^2}\log\frac{1}{\delta}))$ "coin flips" is in $[(1-\epsilon)\|Vw\|, (1+\epsilon)\|Vw\|]$ with probability $\geq 1-\delta$.

Dequantization Toolbox

Method 3: Low-Rank Approximation (Frieze, Kannan, Vempala, 1998)

- For $A \in \mathbb{C}^{m \times n}$, given $A \in \mathcal{SQ}$ and some threshold k, we can output a description of a low-rank approximation of A with poly(k) queries.
- Specifically, we output two matrices $S, \hat{U} \in \mathcal{SQ}$ where $S \in \mathbb{C}^{\ell \times n}$, $\hat{U} \in \mathbb{C}^{\ell \times k}$ ($\ell = \text{poly}(k, \frac{1}{\epsilon})$), and this implicitly describes the low-rank approximation to A, $D := A(S^\dagger \hat{U})(S^\dagger \hat{U})^\dagger$ ($\Rightarrow$ rank $D \leq k$).
- This matrix satisfies the following low-rank guarantee with probability $\geq 1 - \delta$: for $\sigma := \sqrt{2/k}\|A\|_F$, and $A_\sigma := \sum_{\sigma_i \geq \sigma} \sigma_i u_i v_i^\dagger$ (using SVD),

$$\|A - D\|_F^2 \leq \|A - A_\sigma\|_F^2 + \epsilon^2 \|A\|_F^2$$

- Note the $\|A - A_\sigma\|_F^2$ term. This says that our guarantee is weak if A has no large singular values.
- Quantum analog: phase estimation

Dequantization Toolbox

$$\left[\cdots A \cdots\right] \left[S^{\dagger}\right] \left[\hat{U}\right] \left[\hat{U^{\dagger}}\right] \left[\cdots S \cdots\right]$$

Polynomial (low-rank)

Application (Lloyd, Tang, 2018)

Problem

For a low-rank matrix $A \in \mathbb{R}^{m \times n}$ with SVD $\sum_i \sigma_i |u_i\rangle \langle v_i|$ and a vector $b \in \mathbb{R}^n$, given $b, A \in \mathcal{SQ}$, (approximately) simulate $\sum_i (\sigma_i)^m |u_i\rangle \langle v_i| b \in \mathcal{SQ}$ for any $m \in \mathbb{Z}$.

Problem

Special case: Moore-Penrose Pseudoinverse

Polynomial (low-rank)

Application (Lloyd, Tang, 2018)

Problem

For a low-rank matrix $A \in \mathbb{R}^{m \times n}$ with SVD $\sum_i \sigma_i |u_i\rangle \langle v_i|$ and a vector $b \in \mathbb{R}^n$, given $b, A \in \mathcal{SQ}$, (approximately) simulate $\sum_i (\sigma_i)^m |u_i\rangle \langle v_i| b \in \mathcal{SQ}$ for any $m \in \mathbb{Z}$.

Problem

Special case: Moore-Penrose Pseudoinverse

Algorithm

- *Algorithm: see the thesis!*

Thoughts

- Claim (Tang): For machine learning problems, $\mathcal{SQ}$ assumptions are more reasonable than state preparation assumptions.
- We discussed pseudo-inverse which inverts singular values, but in principle we could have applied any function to the singular values
- Gilyen et. al (2018) show that many quantum machine learning algorithms indeed apply polynomial functions to singular values
- Our discussion suggests that exponential quantum speedups are tightly related to problems where high-rank matrices play a crucial role (e.g. Hamiltonian simulation or QFT)

Read the Fine Print

- This poses two problems if seek to use these algorithms: the "state preparation" and "readout" problems.
- Even if we ignore the readout problem, can we at least find a state preparation routine that maintains a speedup for the discussed quantum algorithms? Open question!
- See "Quantum Machine Learning Algorithms: Read the Fine Print" by Aaronson

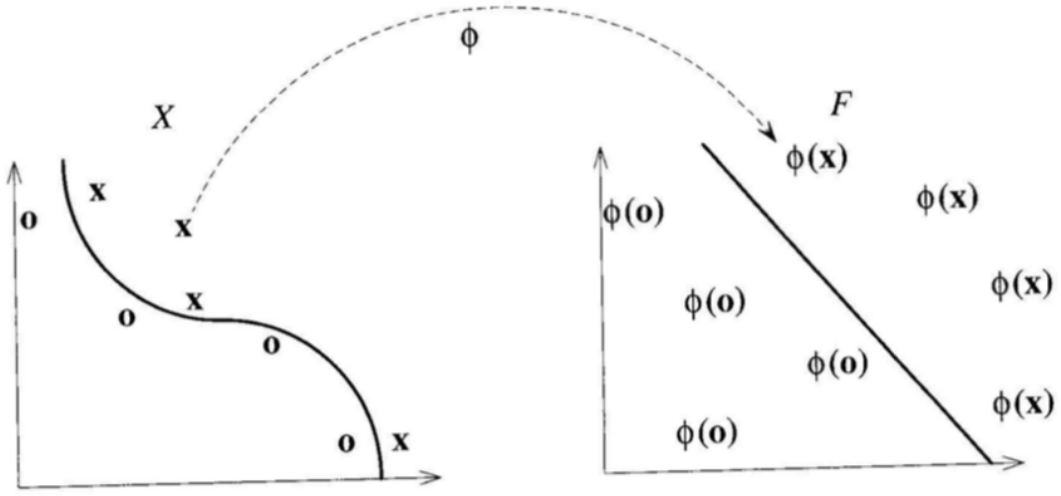
"Dequantization" (Tang)

Definition

Let $\mathcal{A}$ be a quantum algorithm with input $|\varphi_1\rangle, \ldots, |\varphi_C\rangle$ and output either a state $|\psi\rangle$ or a value λ. We say we dequantize $\mathcal{A}$ if we describe a classical algorithm that, given $\varphi_1, \ldots, \varphi_C \in \mathcal{SQ}$, can evaluate queries to $\psi \in \mathcal{SQ}$ or output λ, with similar guarantees to $\mathcal{A}$ and query complexity $\text{poly}(C)$.

Kernel Maps

- SVM: construct a separating hyperplane such that the distance to the nearest training observation (minimum margin) is maximized
- "kernel trick": implicitly maps the data to a higher dimensional space so that it is separable or approximately separable (implicit by Mercer's Theorem)
- Seek a linear function $f(x) = \vec{w} \cdot \vec{x}$ where $\vec{w}$ is a weight vector s.t. $y_i f(x_i) > 0$ for all (or many) i
- Input state preparation solution?



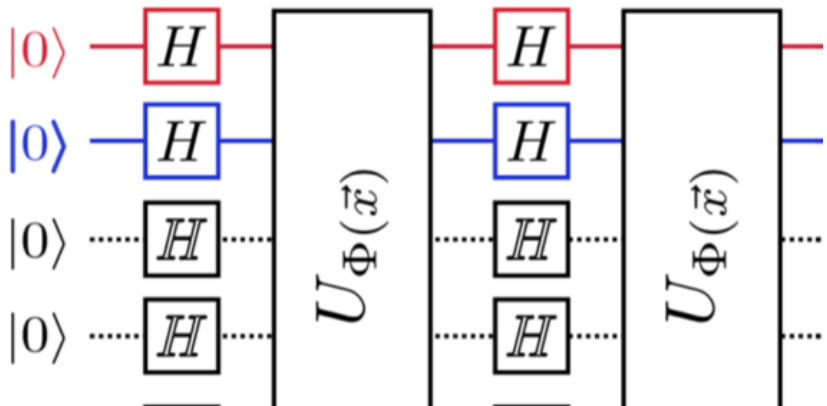
The Shifted Bent Map

Consider the feature map defined in Havlicek et. al (Nature Physics).
First define,

$$U_{\Phi(x)} = \exp\left(i \sum_{S \subseteq [n]} \varphi_S(x) \prod_{i \in S} Z_i\right) \tag{1}$$

with nonlinear coefficients $\varphi_S(x) \in \mathbb{R}$. Then, the feature map is given by

$$\mathcal{U}_{\Phi(x)} = U_{\Phi(x)} H^{\otimes n} U_{\Phi(x)} H^{\otimes n} \tag{2}$$



Quantum Variational Classification

1. Encode classical input data $\{\vec{x}_i\}$ as $\{|\Phi(x_i)\rangle\}$ $(\equiv \{\Phi(x_i)\,|0\rangle\})$ for some unitary feature map φ
2. Apply a θ-parameterized unitary $W(\theta)$ to the input state giving

$$W(\theta)\,|\Phi(x_i)\rangle$$

3. Measure multiple shots of some observables $\{M\}$. The expectation value of these observables is utilized to output a real-valued prediction function $f : \{x_i\} \to \mathbb{R}$.
4. Minimize a chosen cost function with is a function of $f(x_i)$. For example, the ℓ_2 loss

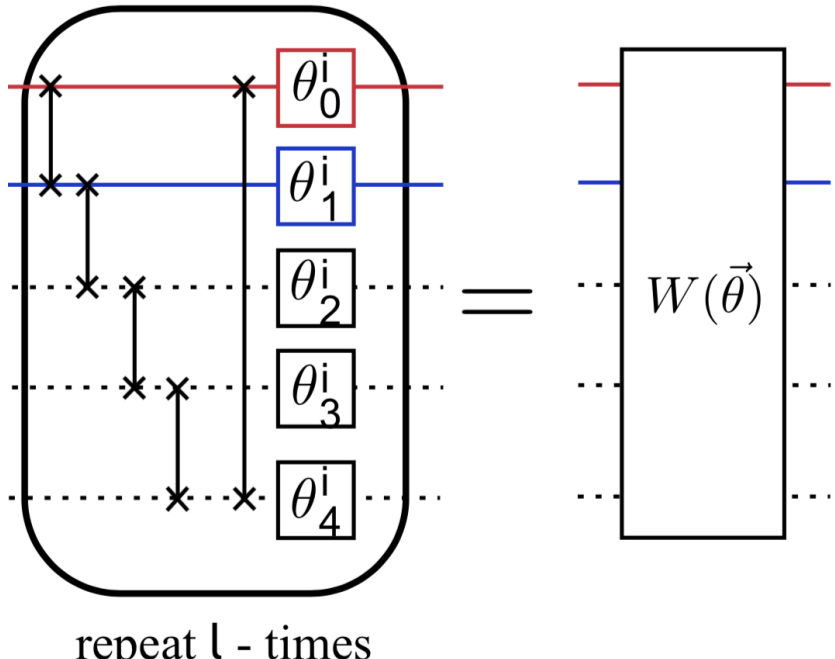
$$\sum_i (y_i - f(x_i))^2$$

5. Evaluate the performance and update θ accordingly (e.g. using gradient descent). Return to step 2, or exit if done iterating.

Quantum Variational Classification: Continued

A common choice of $W(\theta)$ (Figure 24) is the following circuit:

- Let $W(\theta)$ have l layers i.e. $W(\theta) = W_1(\theta) \cdots W_l(\theta)$.
- Now, for each $W_i(\theta)$ we construct the following circuit: label the qubits $\{1, \cdots, n\}$ and consider the nearest-neighbor pairing $N = \{(1,2), (2,3), \cdots, (n-1, n)\}$. Then, perform Control-Z on each pair in N. Next, write $\theta = (\theta_1, \cdots, \theta_n)$ where each θ_i is a vector specified by Euler angles. Hence, we can perform rotation $R(\theta_i)$ on the ith qubit.



Here, we demonstrate an experimental construction of the feature map defined in (2) considering 2 qubits for data input. We use the Strawberry Fields package for simulation [**?**].
We generate data so that it can be perfectly separated. Choose coefficients

$$\varphi_i(x) = x_i, i \in \{1, 2\}$$
$$\varphi_{\{1,2\}}(x) = (\pi - x_1)(\pi - x_2)$$

and draw a fixed random element $V \in SU(4)$. Define $\Pi = Z_1 Z_2$ to be parity on 2 qubits. Then, generate data by drawing (uniform) random $x \in (0, 2\pi]^2$ and labelling based on decision rule

$$f(x) = \langle 0 | \mathcal{U}^\dagger_{\Phi(x)} V^\dagger \Pi V \mathcal{U}_{\Phi(x)} | 0 \rangle$$

In particular, define a threshold Δ and if $|f(x)| \leq \Delta$ then draw another x. Otherwise, label by $sgn(f(x))$. We draw 40 such x and arbitrary partition the dataset into training and test subsets of size 20 each.

We then use the variational circuit described in the previous section to train our classifier. In particular, the variational circuit trains some $W(\theta)$ parameterized by θ and then constructs decision rule

$$\tilde{f}(x) = \langle 0 | \mathcal{U}^{\dagger}_{\Phi(x)} W(\theta)^{\dagger} \Pi W(\theta) \mathcal{U}_{\Phi(x)} | 0 \rangle)$$

and classifies by $sgn(\tilde{f}(x))$. In principle, then, because $W(\theta)$ can learn any $SU(4)$ gate for appropriately chosen θ, we should be able to classify to near perfect accuracy. We use the "Nesterov momentum optimizer" to train θ, an improvement of the usual stochastic gradient descent.

Below, we plot the results of a few representative sample trials exploring various values of Δ.

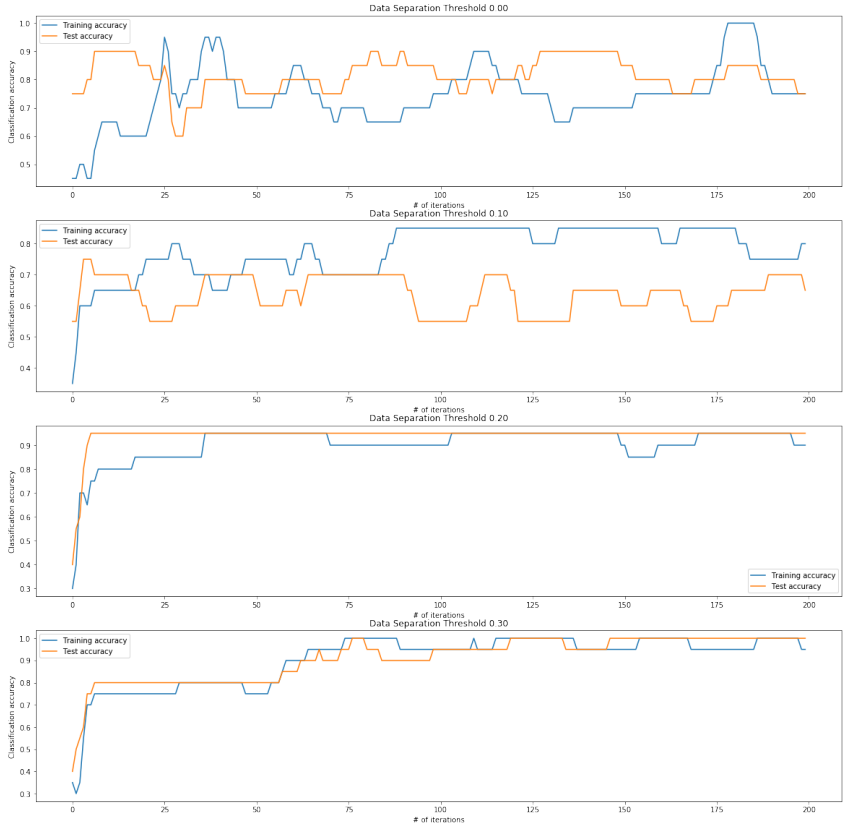


Figure: A comparison of training and test convergence across varying data separations.

Furthermore, can view the convergence of the dual ℓ^2 loss view below.

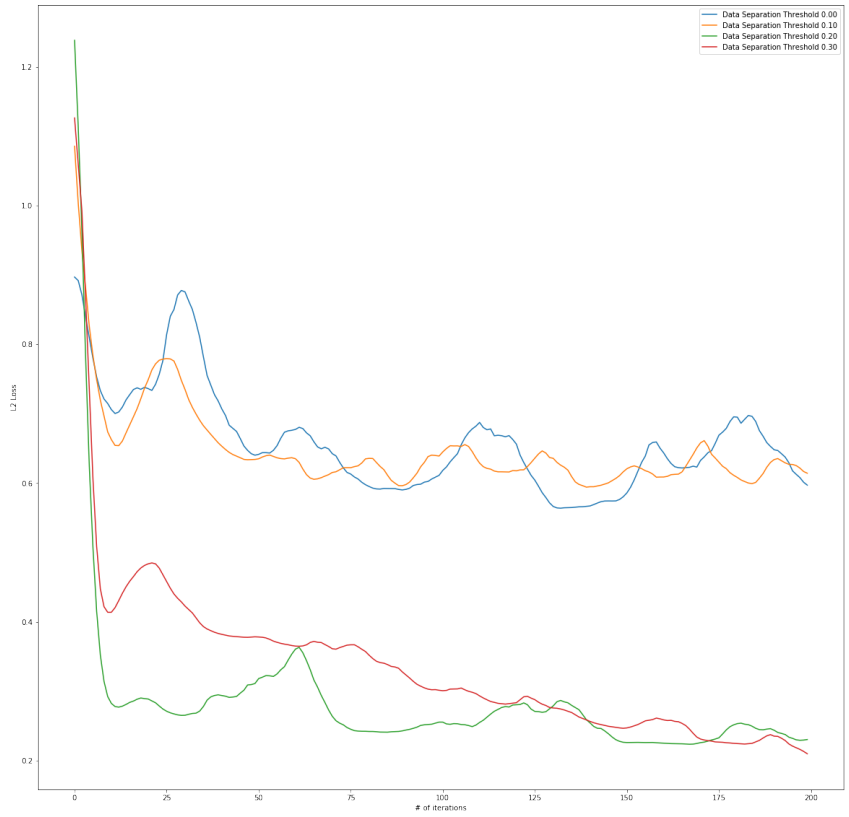


Figure: A similar comparison of ℓ^2 loss convergence across varying data separations.

Visually, generated data may look as below for some $V \in SU(4)$. We can visualize the decision boundary of the kernel by shading the regions in a color indicating the label which the classifier would output at each point.

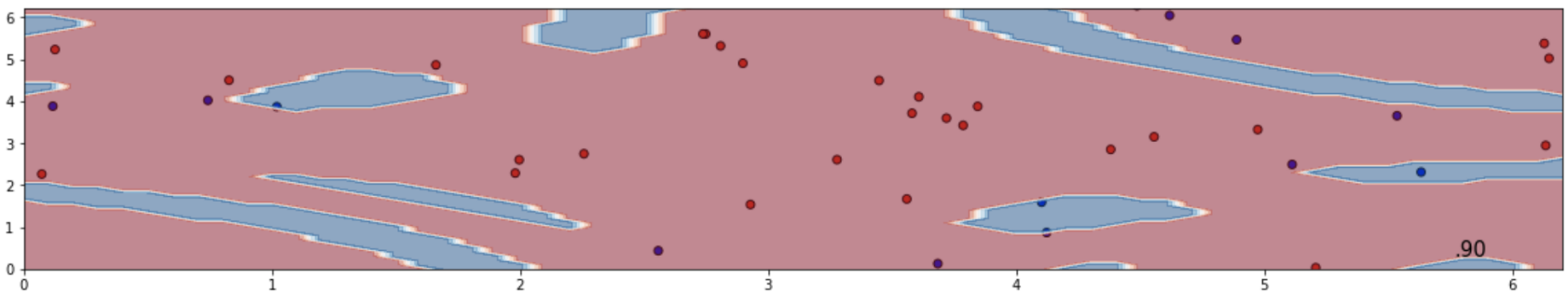


Figure: Sample output of the classifier. The colors of the regions indicate the label for which the classifier has learned for each point. The circles indicate data points in the generated set, colored by true label.

We checked convergence for two separable datasets across the range of 1-5 layers, performing 20 trials per layer depth. Based on the preceding results, data separation of 0.3 was utilized in order to speed up convergence.

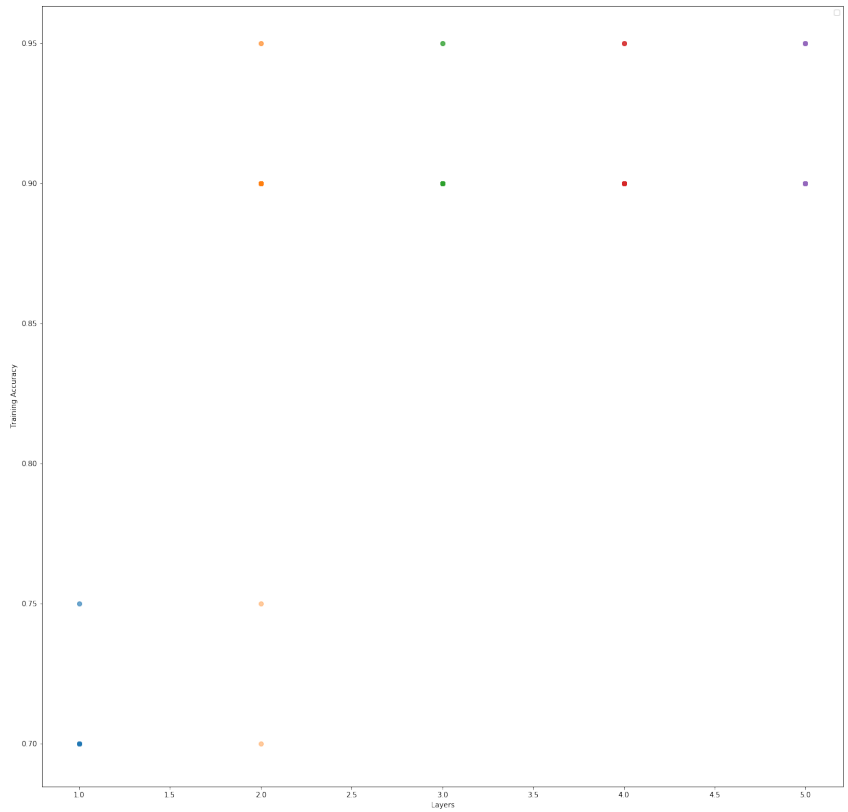


Figure: Training accuracy over repeated trials of 200 epochs varying over layer depth

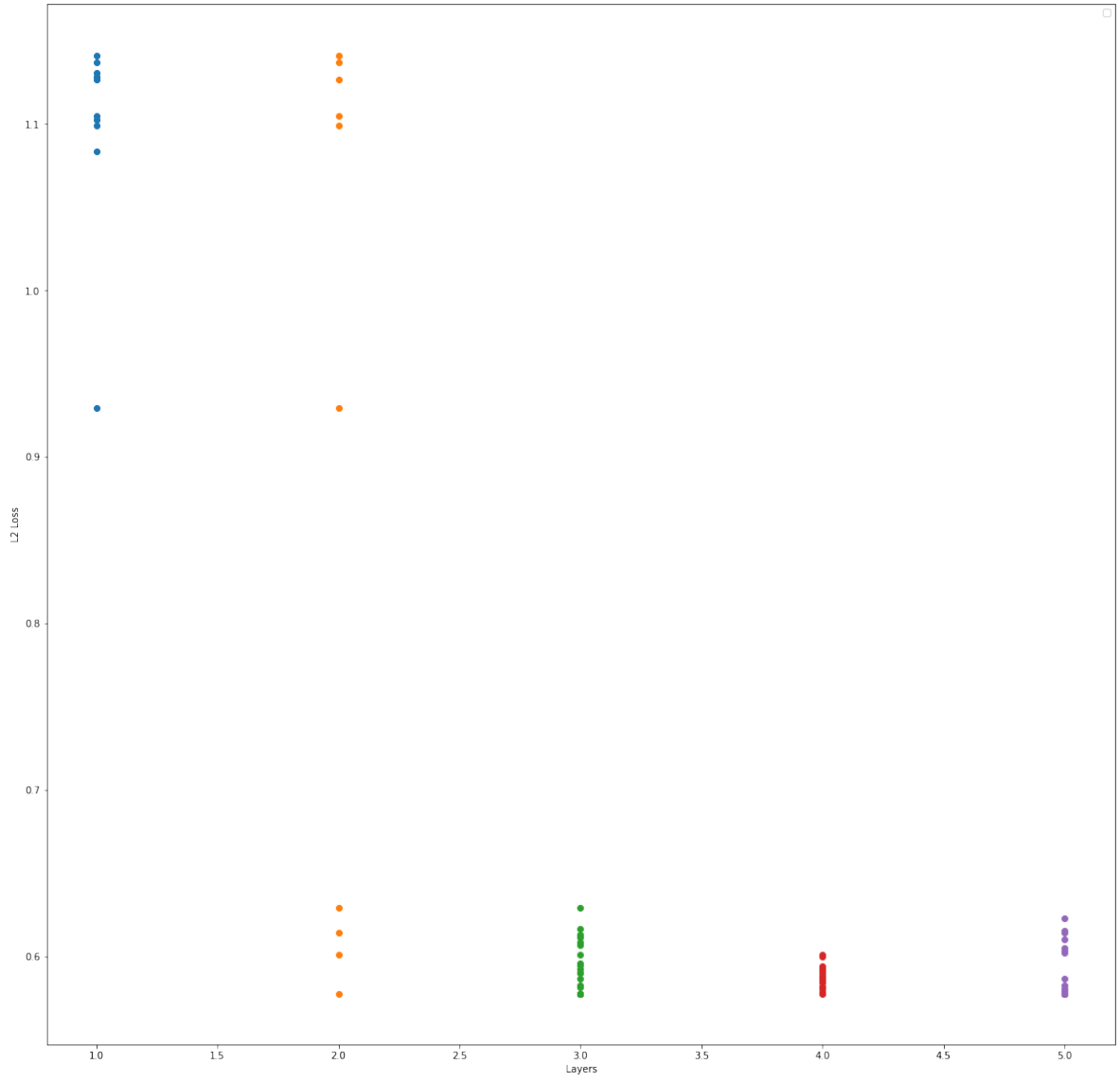
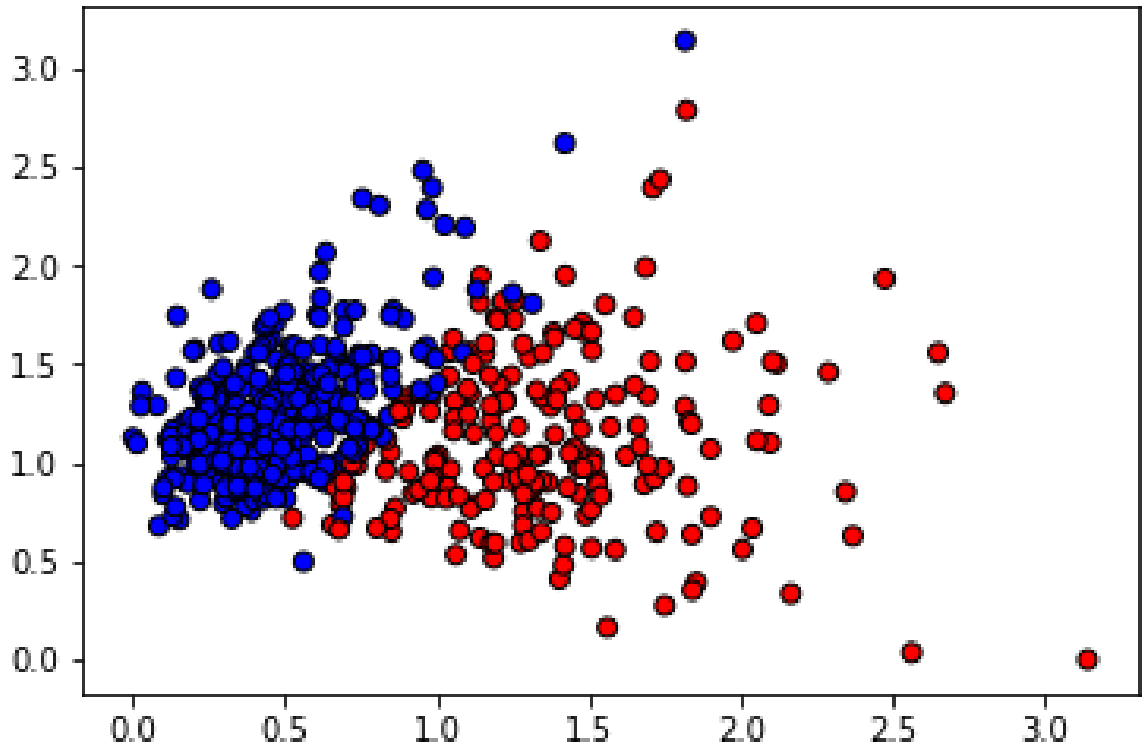
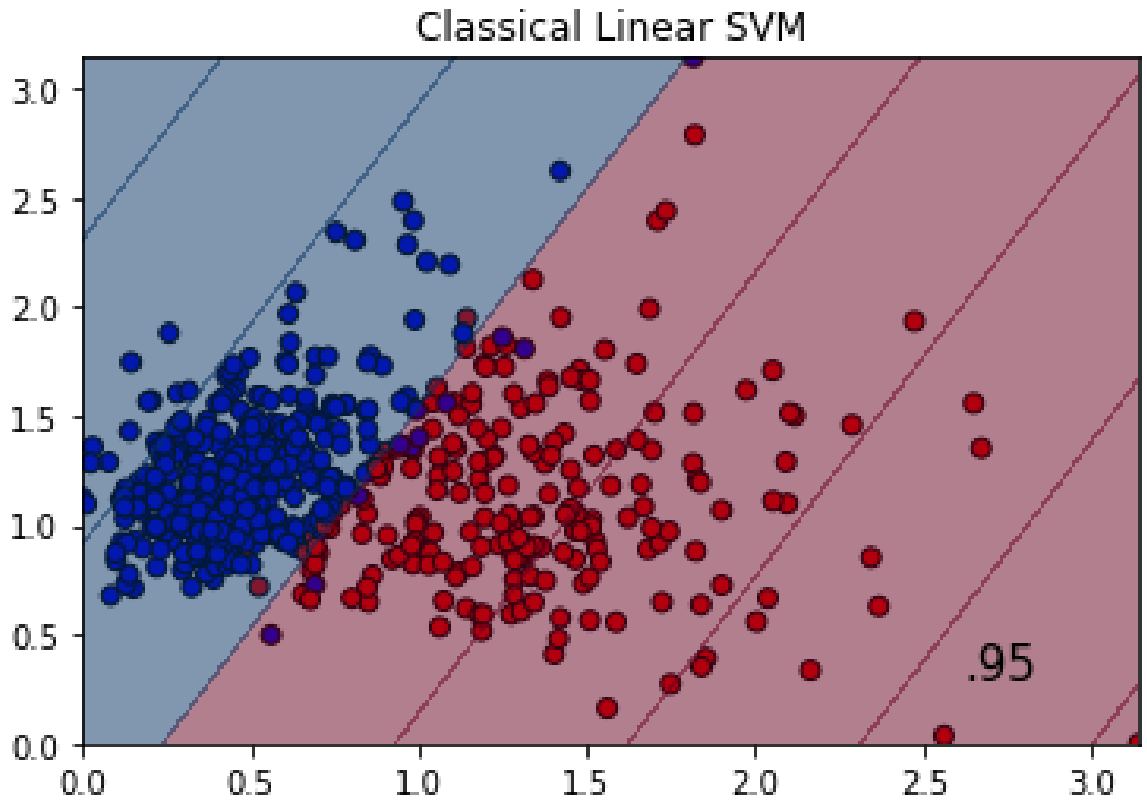


Figure: ℓ^2 loss over repeated trials of 200 epochs varying over layer depth

UCI ML Breast Cancer Dataset which contains 30 features and 569 samples. The labels correspond to whether a cell mass is malignant or benign. We apply PCA to reduce the dimensionality of the dataset to 2 and standardize to a centered Gaussian about $[0, \pi]$.




Classical Linear SVM
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
.95

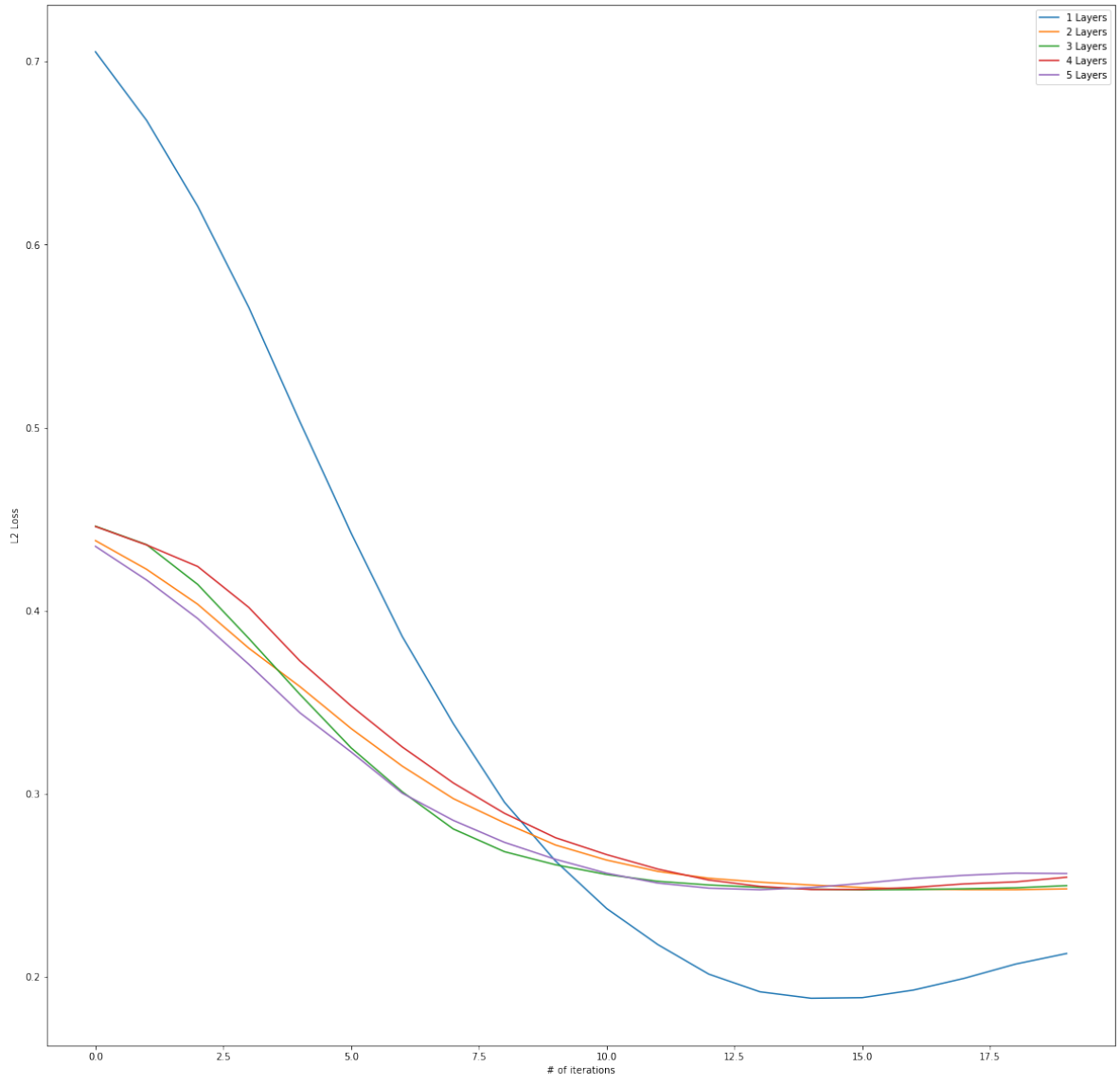


Figure: The pattern in convergence in cost on the UCI Breast Cancer dataset is similar to the one for the separable dataset.

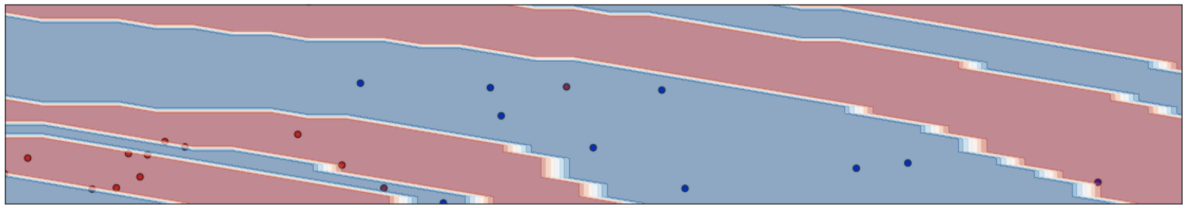


Figure: Sample output of the classifier for the UCI Breast Cancer dataset

Moons Data

We generate the data of this section by using the scikit-learn Python package and add random Gaussian noise to the $\{x_i\}$ for each trial.

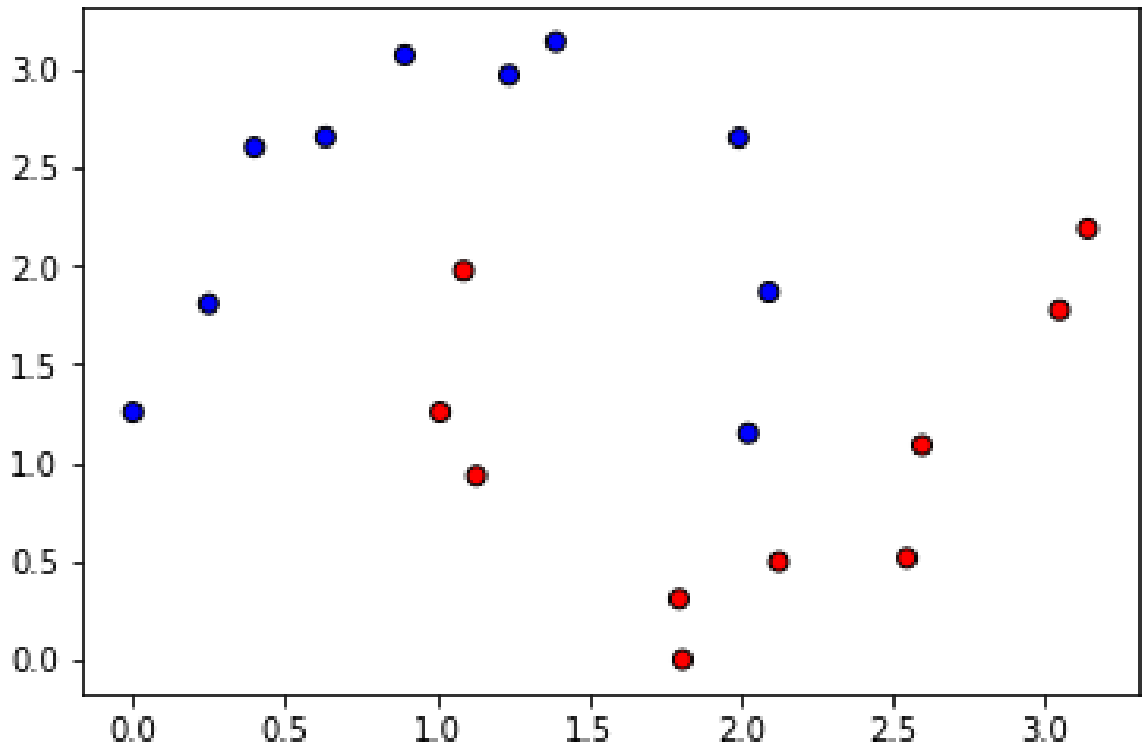


Figure: Sample Moons data for some perturbation in Gaussian noise

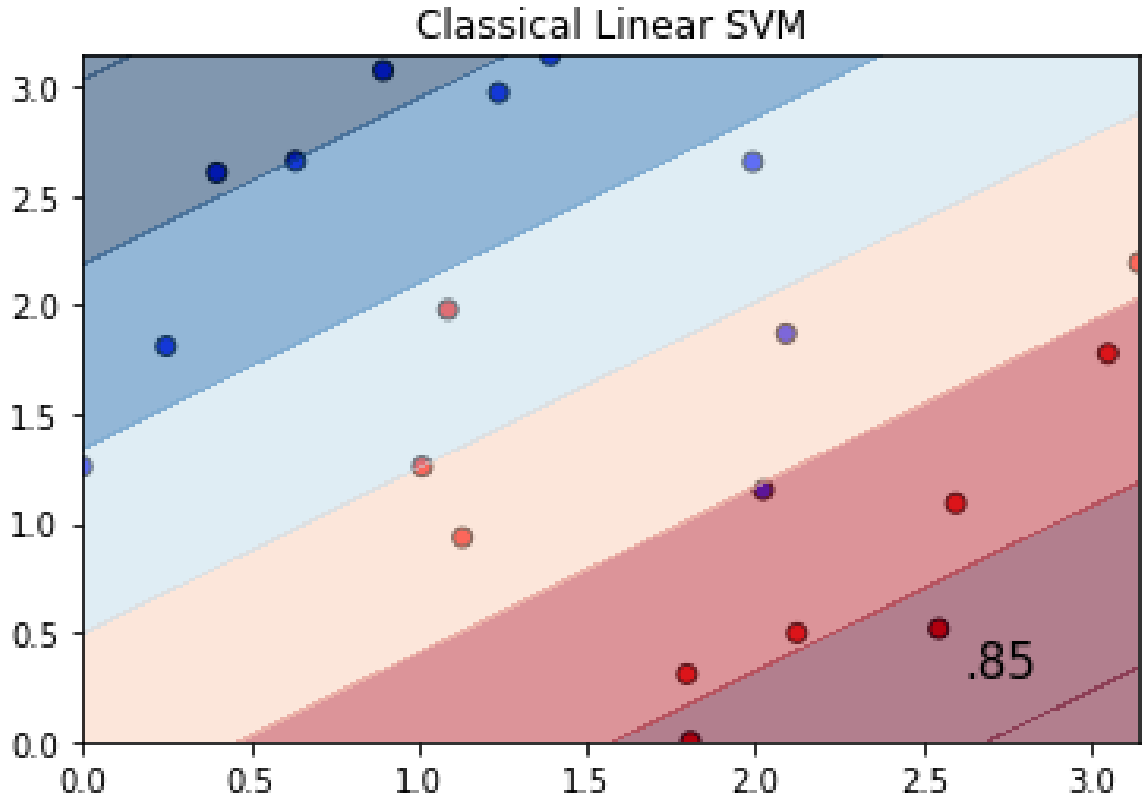


Figure: Performance of SVM with a linear kernel on the Moons dataset

To see how the quantum feature map performs, we direct your attention to the next section.

We consider introducing a set of hyper-parameters which tunes the extent to which nonlinearity enters the phase of the $U_{\Phi(x)}$ (Equation 1). We can revise the diagonal (in the Z basis) unitary within the feature map:

$$U_{\Phi(x;\vec{\gamma})} = \exp\left(i\Big[\sum_i \varphi_i(x) Z_i + \gamma_1 \sum_{i,j} \varphi_{i,j}(x) Z_i Z_j + \cdots + t\gamma_{d-1} \sum_{S\subseteq[d],|S|=d} \varphi_S(x) \prod_{i\in S} Z_i\Big]\right)$$

Figure 42 offers a representative view of how adjusting the single γ hyper-parameter for the 2-qubit map adjusts the classification circuit to which the variational algorithm converges after 200 epochs when considering the UCI Breast Cancer dataset.

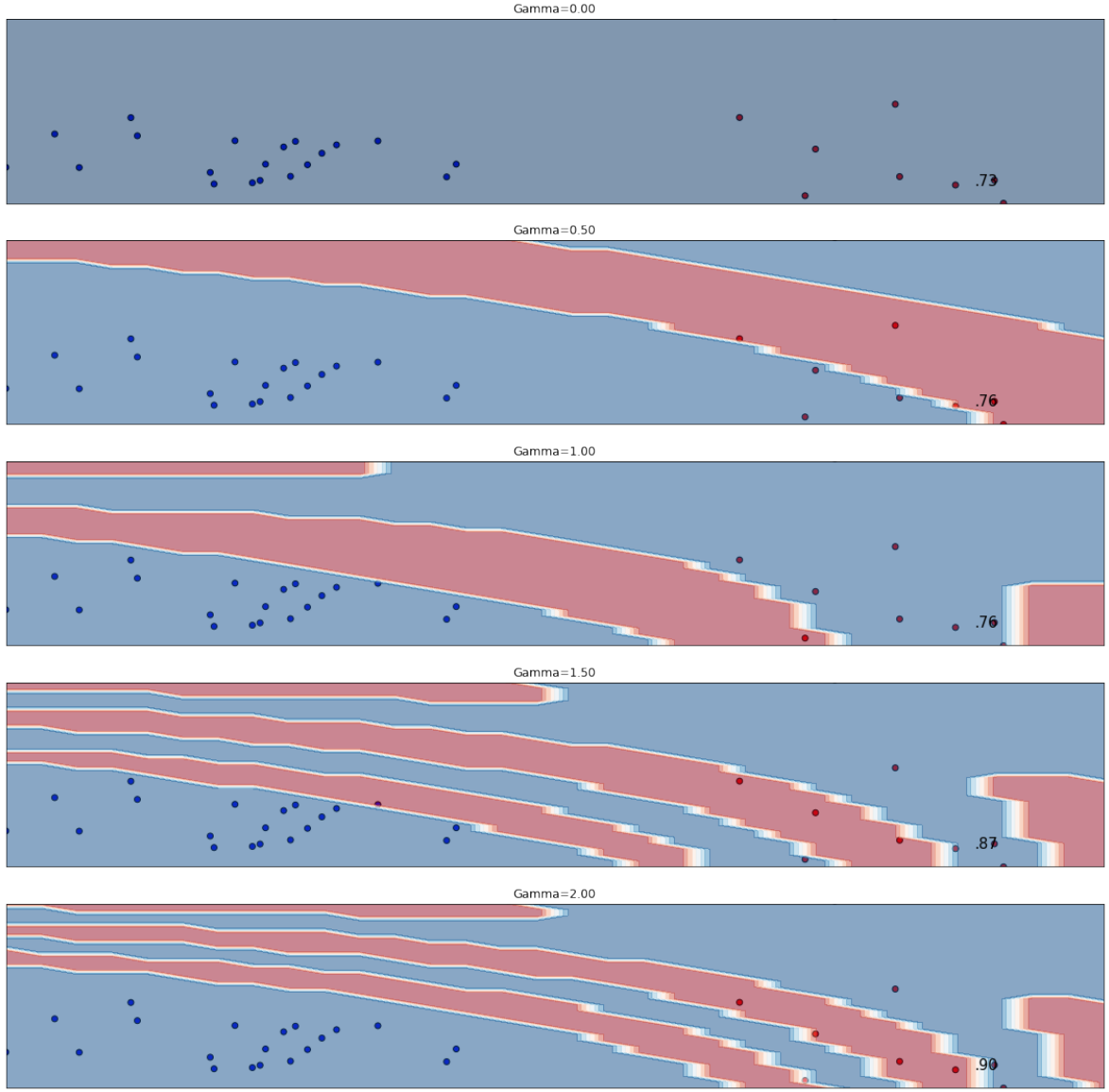


Figure: Performance of the variational classification circuit (for 200 epochs) on a random sample of the UCI Breast Cancer dataset for varying 2-point nonlinearity parameter γ.

We contribute a novel method of thinking about tuning hyper-parameters for a quantum feature map and additionally demonstrate that the method works well in practice by returning to the "moons" binary classification problem of Section 3.
First, we make the straightforward observation that the goal of the variational classifier can be restated as solving a state identification problem:
Write that a data point x is encoded as

$$|\Phi(x;\vec{\gamma})\rangle \equiv \mathcal{U}_{\Phi(x;\vec{\gamma})}\,|0\rangle$$

Then, consider the density matrices

$$\rho_+ = \frac{1}{N_+} \sum_{x_i:y_i=+1} |\Phi(x_i;\vec{\gamma})\rangle \langle\Phi(x_i;\vec{\gamma})|$$

$$\rho_- = \frac{1}{N_-} \sum_{x_i:y_i=-1} |\Phi(x_i;\vec{\gamma})\rangle \langle\Phi(x_i;\vec{\gamma})|$$

Therefore, the goal of binary classification can be restated as finding the optimal pair of POVMs so as to maximize the probability of correctly distinguishing these two density matrices. The trace distance provides an achievable (by the Helstrom measurement) upper bound for solving this state identification problem.
Hence, our method of hyperparameter tuning can be stated as follows: (1) compute the density matrices ρ_+, ρ_- above for a set of candidate $\{\vec{\gamma_i}\}$ (2) computing the trace distance $T(\rho_+, \rho_-)$ across the set of $\{\vec{\gamma_i}\}$, and (3) choose the $\vec{\gamma_i}$ which maximizes trace distance.

For the dataset of Figure 42, we indeed see that trace distance increases with gamma within the range $[0.0, 2.0]$.

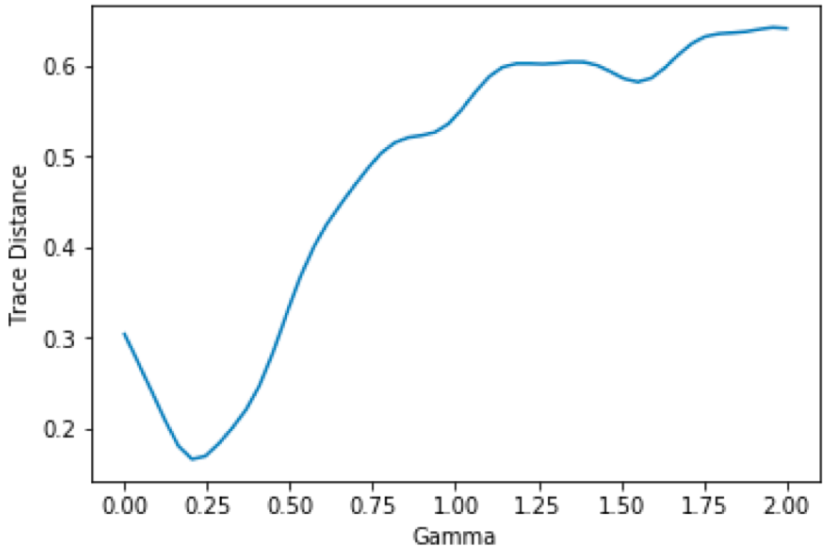
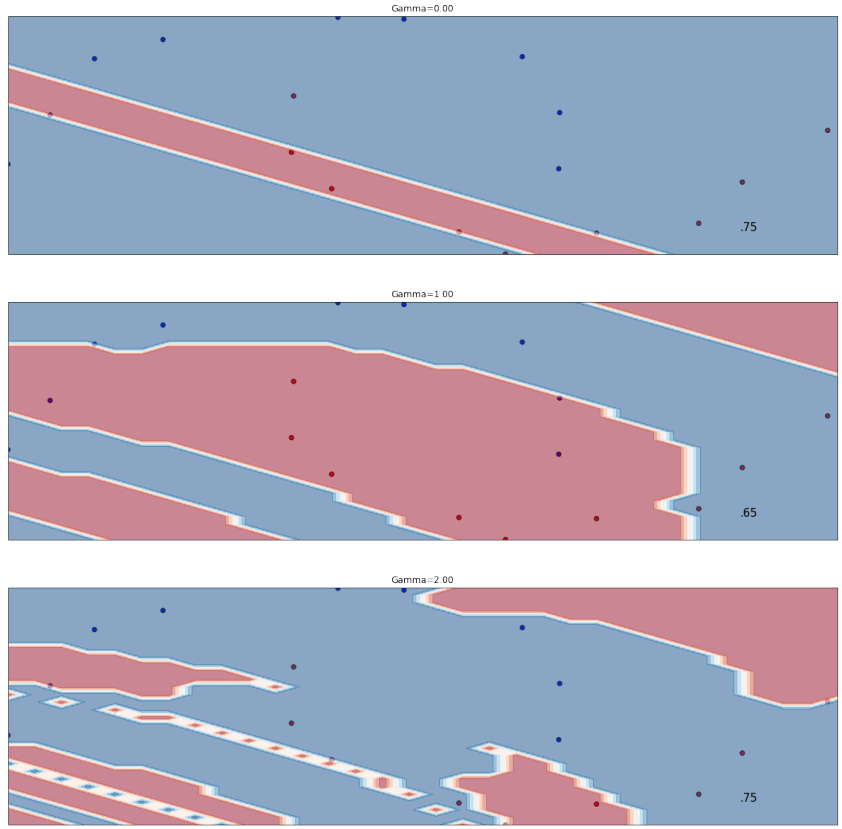


Figure: Trace distance as a function of γ for a random sample of the UCI Breast Cancer dataset

- To see that the hypothesized relationship between trace distance and the optimal choice of $\vec{\gamma}$ holds for the roughly linearly separable practical data of the previous section, we performed 20 trials of 200 epochs for linearly spaced γ on two sets of 20 random samples of the practical dataset, now requiring that the ratio between $+1$ and -1 labelled data points is balanced.
- As a result for $\gamma = \{0.0, 0.25, 0.5, 1.0, 1.5, 2.0\}$ we found that the optimal training accuracy was given by $\{0.55, 0.55, 0.60, 0.80, 0.85, 0.85\}$.

Now, we consider the Moons dataset over varying γ



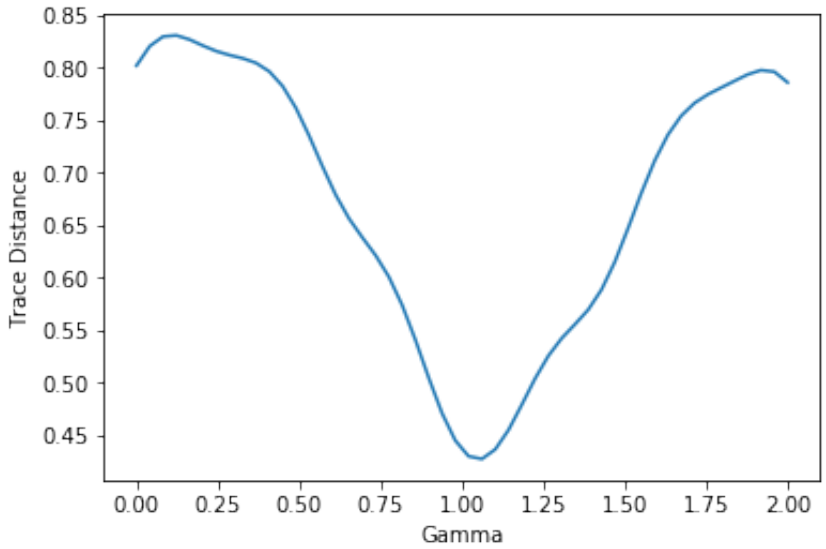
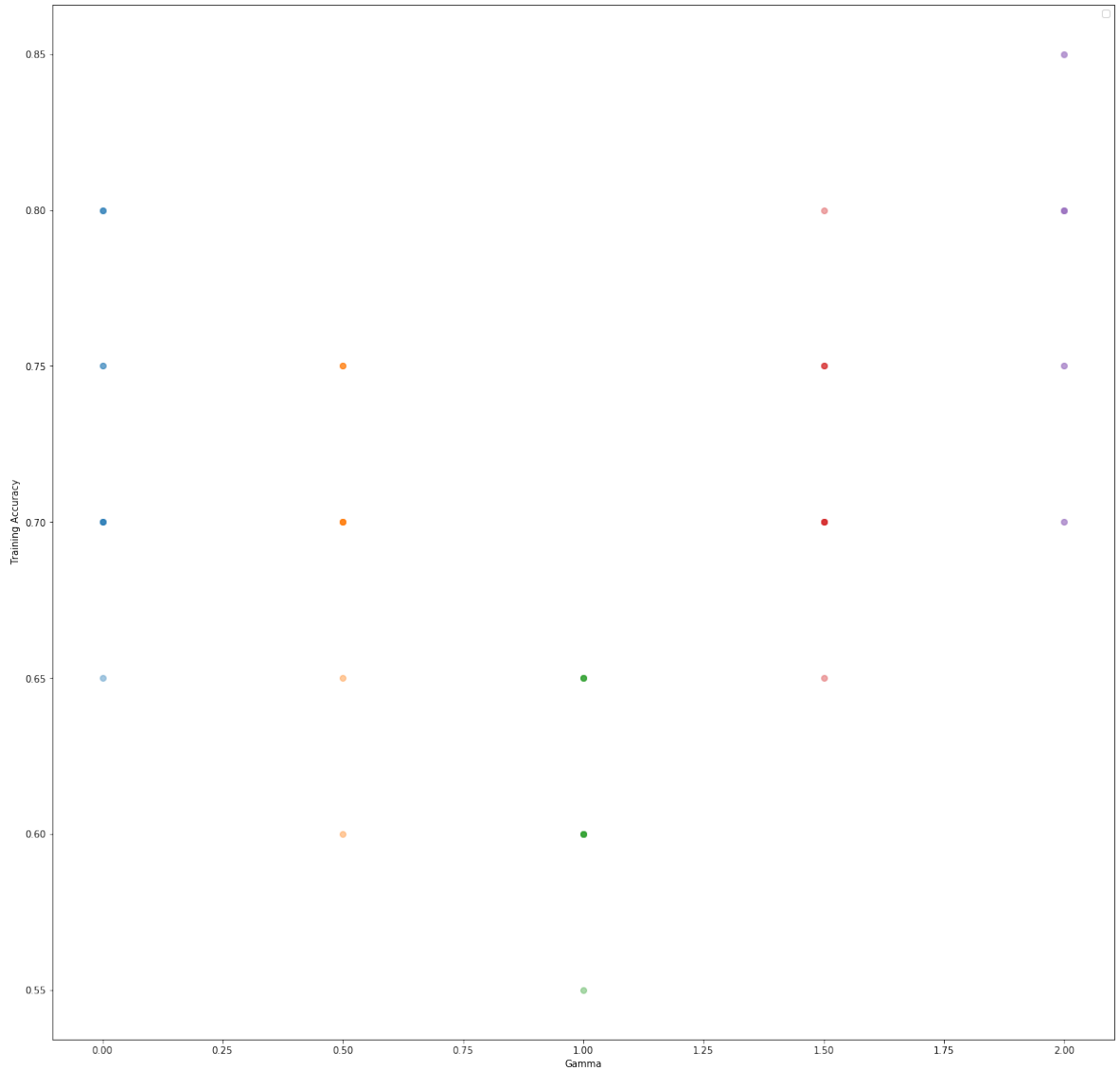


Figure: The structure of the moons dataset empirically requires a V shaped curve with the trough roughly near $\gamma = 1.0$. Note that such a γ is the default value used with no hyper-parameter tuning.

So, we expect the classifier to perform relatively poorly for $\gamma = 1.0$ and increase towards the extremities. Again, we performed 20 trials of 200 epochs for linearly spaced γ on two sets of 20 random samples of the Moons dataset with noise, now requiring that the ratio between $+1$ and -1 labelled data points is balanced.


0.85
0.80
0.75
0.70
0.65
0.60
0.55
Training Accuracy
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Gamma

Conclusion

- We hope that future works can extend these results and identify additional datasets that have a particular structure for which γ ought to be tuned and leave it to these works to verify that trace distance serves a useful purpose to this end.
- Along these lines, we have open sourced (via GitHub) the code we've used for these simulations

Thank you for listening!

- I am grateful to the committee for their enthusiasm and willingness to be a part of this defense
- Questions? fms15@duke.edu