Quantum Machine learning in Feature Hilbert space

The problems attributed to you:

-Training a variational quantum circuits to perform approximately QFT (quantum compilation problem)
-Build quantum circuits for evaluating “exotic” classical kernels
-Quantum classifiers with qutrits

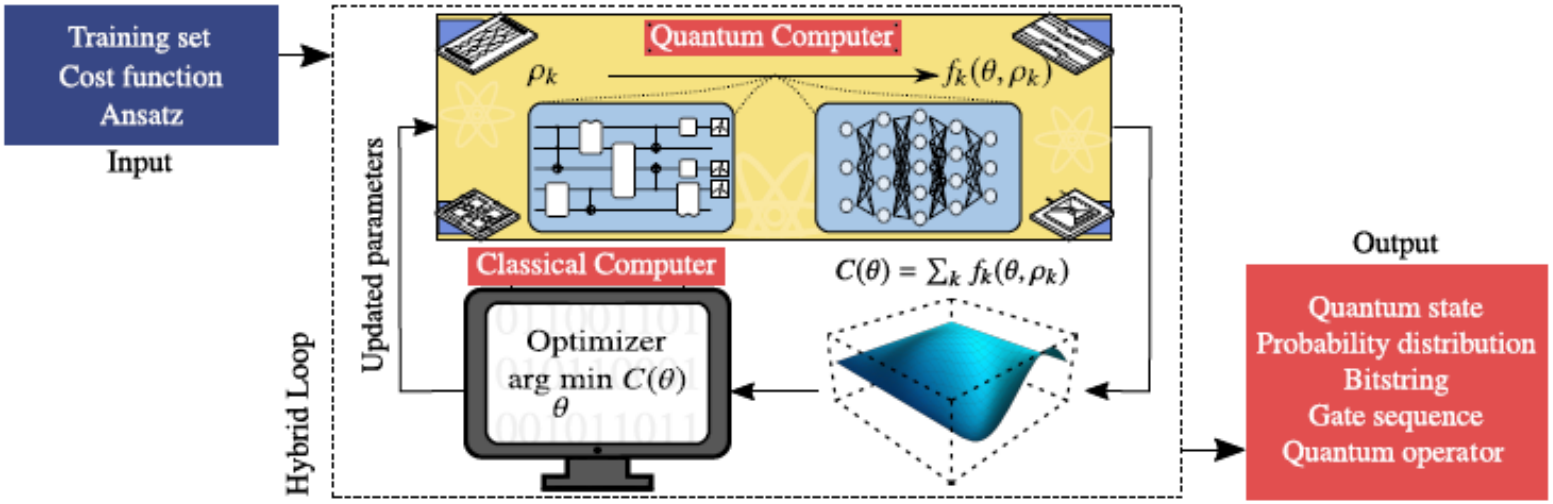
What you need:

- A well-defined problem
- A good-quality review paper on the subject
- Warm-up: reproduce some basic easy results
- creativity

Eva:

Variational quantum algorithms

M. Cerezo,[1,2,3,*] Andrew Arrasmith,[1,3] Ryan Babbush,[4] Simon C. Benjamin,[5] Suguru Endo,[6] Keisuke Fujii,[7,8,9] Jarrod R. McClean,[4] Kosuke Mitarai,[7,10,11] Xiao Yuan,[12,13] Lukasz Cincio,[1,3] and Patrick J. Coles[1,3,†]



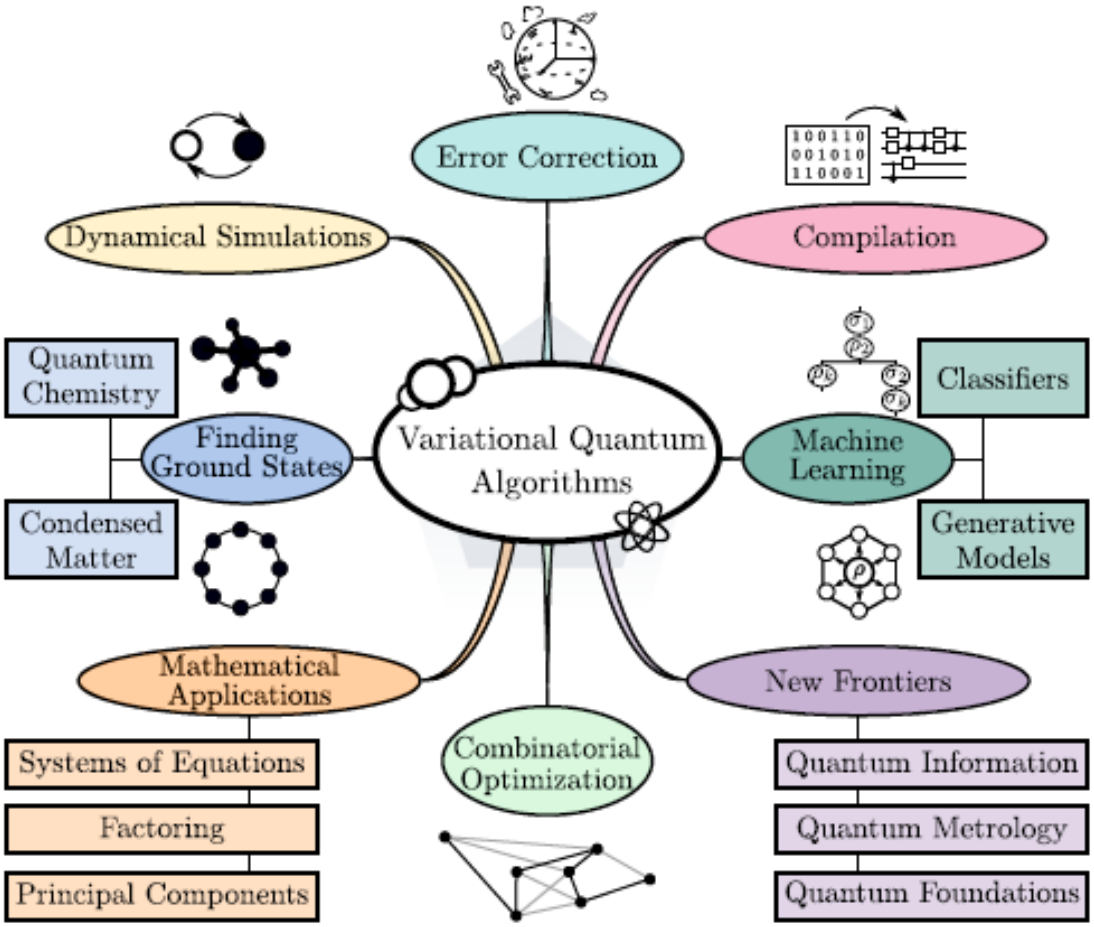
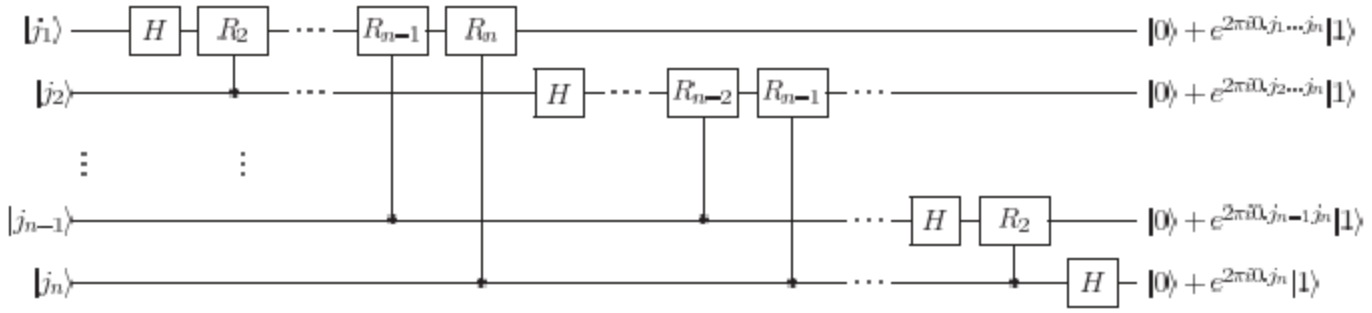


FIG. 3. **Applications of Variational Quantum Algorithms (VQAs).** Many applications have been envisioned for VQAs. Here we show some of the key applications that are discussed in this Review.

QFT



+ swaps

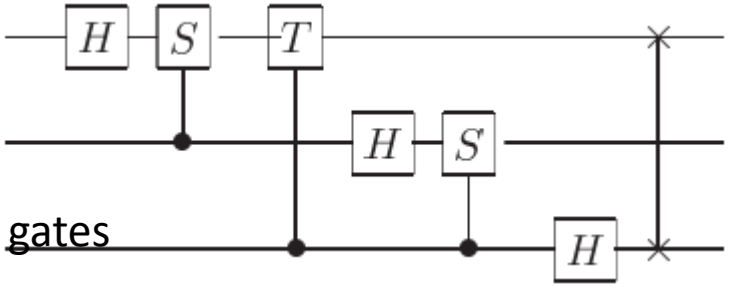
Classically: DFT $y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}$

If $|j\rangle \xrightarrow{U} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle$ with $N = 2^n$

then

$$\sum_{j=0}^{N-1} x_j |j\rangle \xrightarrow{U} \sum_{k=0}^{N-1} y_k |k\rangle$$

Can we do with less but approximately?
Let's do the counting..



Initial circuit's depth: $\Theta(n^2)$ $\frac{n^2}{2}$: controlled gates

Classical DFT: $\Theta(n2^n)$

This is not an algorithm, it is simpler because it has specific: Input-Output

We could transform to an algorithm later that is state-dependent
and therefore has less parameters

Steps to follow:
-Start with 3 qubits. (1st non-trivial case)

Cost function

Ansatzes

Express ability of circuit you use, and characteristics of the Unitary and of the generating Hamiltonian, Available Hamiltonians

Gradients

Optimizers

Can you use the IBM platform for the simulations?

U-> decomposition on 64 generators

-> only 32 participate

-> these form a subgroup

-> minimum? number of generators

11 (5 local+ **6** binary ~n^2/2)

-This gives some evidence for an ansatz for 3 qubits
(build and try this out)
- One needs to look at 2 qubit and 4 qubit in order to see if generalizes

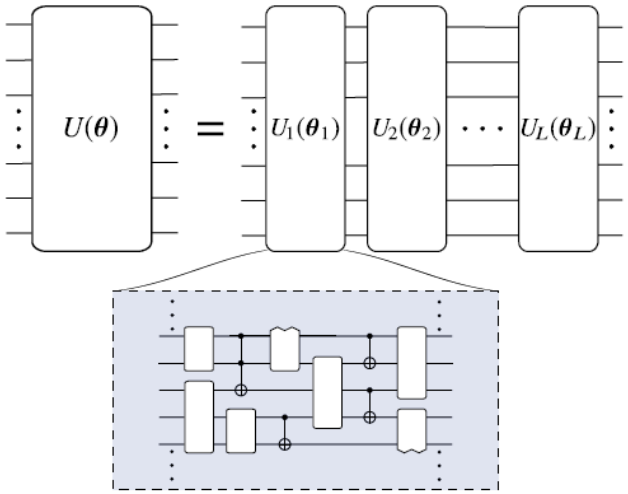
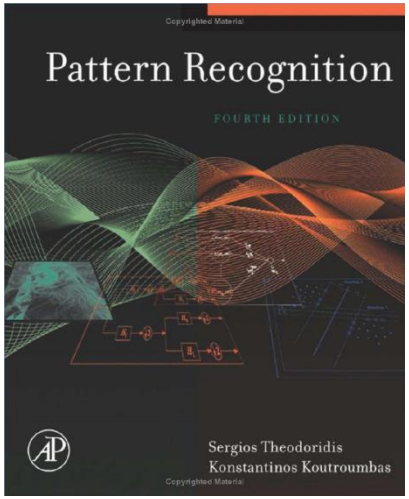


FIG. 2. **Schematic diagram of an ansatz**. The unitary $U(\boldsymbol{\theta})$, with $\boldsymbol{\theta}$ a set of parameters, can be expressed as a product of L unitaries $U_l(\boldsymbol{\theta}_l)$ sequentially acting on an input state. As indicated, each unitary $U_l(\boldsymbol{\theta}_l)$ can in turn be decomposed into a sequence of parametrized and unparametrized gates.

Parenthesis: Classification with Classical means



Linear classifiers

- Neural networks (single layer)
- Least square
- SVM

Searching for the decision hyper-plane

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$$

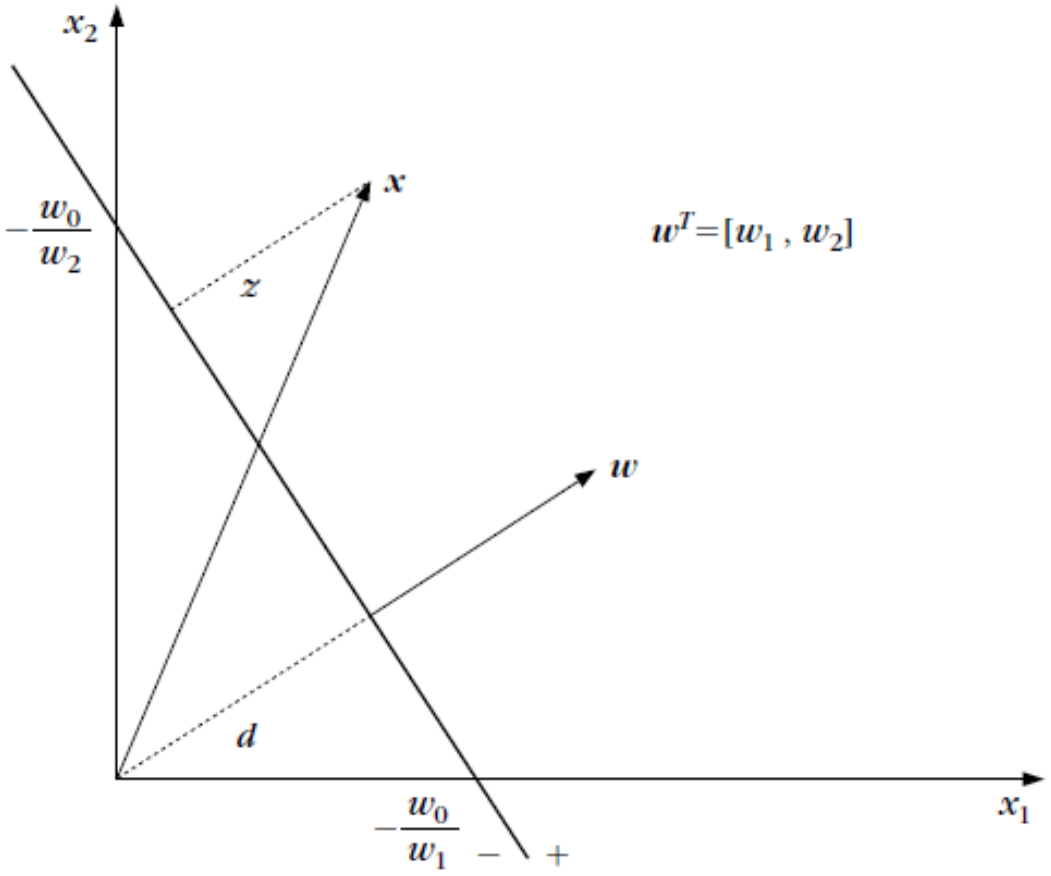
old. If $\boldsymbol{x}_1, \boldsymbol{x}_2$ are two points on the decision hyperplane, then the following is valid

$$0 = \boldsymbol{w}^T \boldsymbol{x}_1 + w_0 = \boldsymbol{w}^T \boldsymbol{x}_2 + w_0 \quad \Rightarrow$$
$$\boldsymbol{w}^T (\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0 \qquad (3.2)$$

any $\boldsymbol{x}_1, \boldsymbol{x}_2$), it is apparent from Eq. (3.2) that the vector $\boldsymbol{w}$ is *orthogonal* to the decision hyperplane.

$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}$$

$$z = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}}$$



$$\boldsymbol{w}^T = [w_1, w_2]$$

If we assume two linear separable classes w1, w2

$$w^{*T}x > 0 \quad \forall x \in \omega_1$$
$$w^{*T}x < 0 \quad \forall x \in \omega_2$$

$$x' \equiv [x^T, 1]^T$$
$$w' \equiv [w^{*T}, w_0^*]^T$$

How to find the hyperplane?

THE PERCEPTRON ALGORITHM

Cost: $J(w) = \sum_{x \in Y} (\delta_x w^T x)$ + with gradient descent

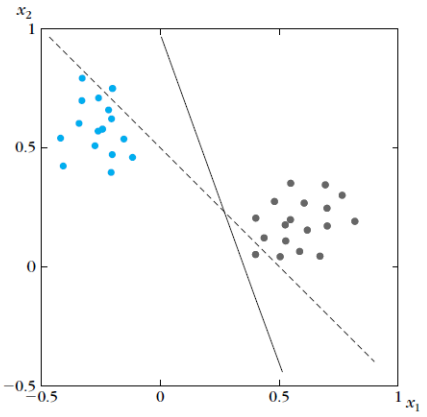


FIGURE 3.3
An example of the perceptron algorithm. After the update of the weight vector, the hyperplane is turned from its initial location (dotted line) to the new one (full line), and all points are correctly classified.

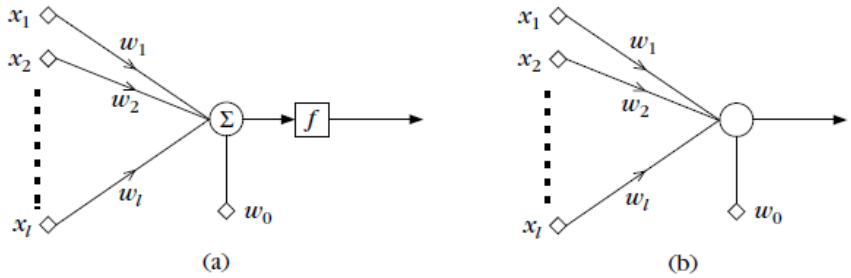


FIGURE 3.5
The basic perceptron model. (a) A linear combiner is followed by the activation function. (b) The combiner and the activation function are merged together.

Other methods:

- Least Square methods

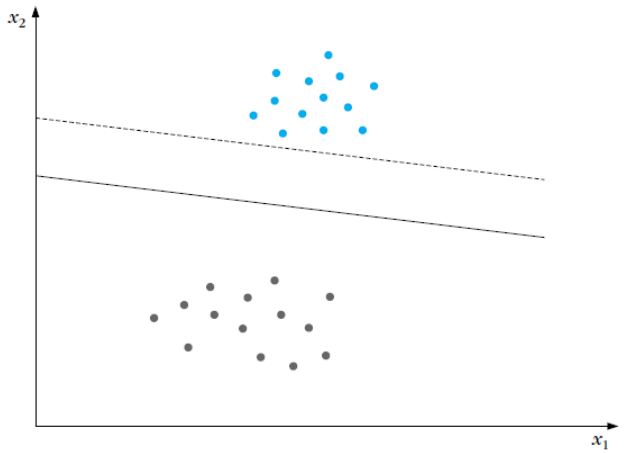
$y(\boldsymbol{x}) \equiv y = \pm 1$. The weight vector will be computed so as to minimize the mean square error (MSE) between the desired and true outputs, that is,

$$J(\boldsymbol{w}) = E[|y - \boldsymbol{x}^T \boldsymbol{w}|^2] \tag{3.26}$$

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}) \tag{3.27}$$

- SVM

"1" is $2z_1$, and the margin for direction "2" is $2z_2$. *Our goal is to search for the direction that gives the maximum possible margin.* However, each hyperplane is



$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$$

$$z = \frac{|g(\boldsymbol{x})|}{\|\boldsymbol{w}\|}$$

We can now scale $\boldsymbol{w}, w_0$ so that the value of $g(\boldsymbol{x})$, at the nearest points in ω_1, ω_2 (circled in Figure 3.10), is equal to 1 for ω_1 and, thus, equal to -1 for ω_2. This is equivalent with

1. Having a margin of $\frac{1}{\|\boldsymbol{w}\|} + \frac{1}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$
2. Requiring that

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 \geq 1, \qquad \forall \boldsymbol{x} \in \omega_1$$
$$\boldsymbol{w}^T\boldsymbol{x} + w_0 \leq -1, \qquad \forall \boldsymbol{x} \in \omega_2$$

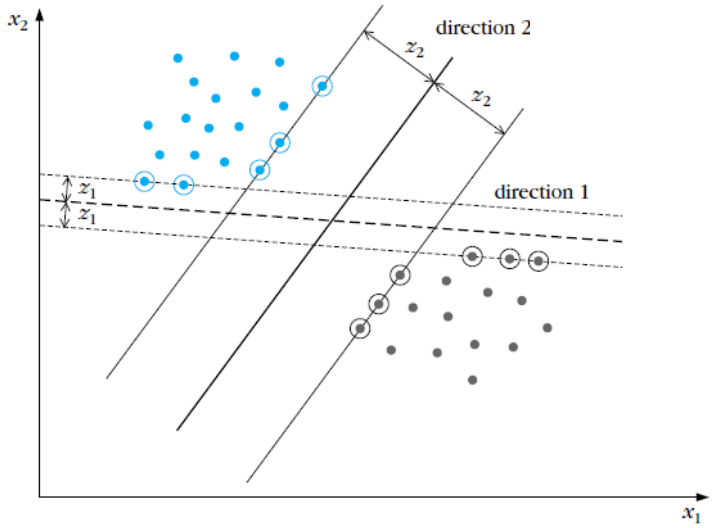


FIGURE 3.10
An example of a linearly separable two-class problem with two possible linear classifiers.

We have now reached the point where mathematics will take over. For each $\boldsymbol{x}_i$, we denote the corresponding class indicator by y_i ($+1$ for ω_1, -1 for ω_2.) Our task can now be summarized as: Compute the parameters $\boldsymbol{w}, w_0$ of the hyperplane so that to:

$$\text{minimize} \quad J(\boldsymbol{w}, w_0) \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{3.72}$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) \geq 1, \quad i = 1, 2, \ldots, N \tag{3.73}$$

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N}\lambda_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1]$$

Obviously, minimizing the norm makes the margin maximum. This is a nonlinear (quadratic) optimization task subject to a set of linear inequality constraints. The Karush–Kuhn–Tucker (KKT) conditions (Appendix C) that the minimizer of (3.72), (3.73) has to satisfy are

$$\frac{\partial}{\partial \boldsymbol{w}}\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \mathbf{0} \tag{3.74}$$

$$\frac{\partial}{\partial w_0}\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = 0 \tag{3.75}$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N \tag{3.76}$$

$$\lambda_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1] = 0, \quad i = 1, 2, \ldots, N \tag{3.77}$$

....

The Lagrange multipliers can be either zero or positive (Appendix C). Thus, the vector parameter $\boldsymbol{w}$ of the optimal solution is a linear combination of $N_s \leq N$ feature vectors that are associated with $\lambda_i \neq 0$. That is,

$$\boldsymbol{w} = \sum_{i=1}^{N_s} \lambda_i y_i \boldsymbol{x}_i \tag{3.81}$$

These are known as *support vectors* and the optimum hyperplane classifier as a *support vector machine* (SVM). As it is pointed out in Appendix C, a nonzero Lagrange multiplier corresponds to a so called active constraint. Hence, as the set of constraints in (3.77) suggests for $\lambda_i \neq 0$, *the support vectors lie on either of the two hyperplanes*, that is,

$$\boldsymbol{w}^T \boldsymbol{x} + w_0 = \pm 1 \tag{3.82}$$

In other words, they are the training vectors that are closest to the linear classifier, and they constitute the *critical elements of the training set*. Feature vectors corresponding to $\lambda_i = 0$ can either lie outside the "class separation band," defined as the region between the two hyperplanes given in (3.82), or they can also lie on one of these hyperplanes (degenerate case Appendix C).

called *Lagrangian duality*. The problem can be stated equivalently by its Wolfe dual representation form, that is,

$$\text{maximize} \quad \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) \tag{3.83}$$

$$\text{subject to} \quad \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i \tag{3.84}$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \tag{3.85}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{3.86}$$

The two equality constraints are the result of equating to zero the gradient of the Lagrangian, with respect to $\boldsymbol{w}, w_0$. We have already gained something. The training feature vectors enter into the problem via equality constraints and not inequality ones, which can be easier to handle. Substituting (3.84) and (3.85) into (3.83) and after a bit of algebra we end up with the equivalent optimization task

$$\max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \right) \tag{3.87}$$

$$\text{subject to} \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{3.88}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{3.89}$$

Non-linear classifiers

- Neural networks (more layers)
- Generalized linear classifiers (polynomial, radial basis..)
- SVM with kernels

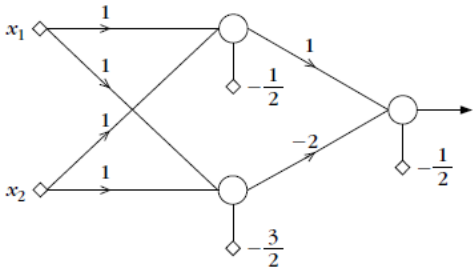


Table 4.1 Truth Table for the XOR Problem

x_1	x_2	XOR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B

Mapping to a linear separable one:

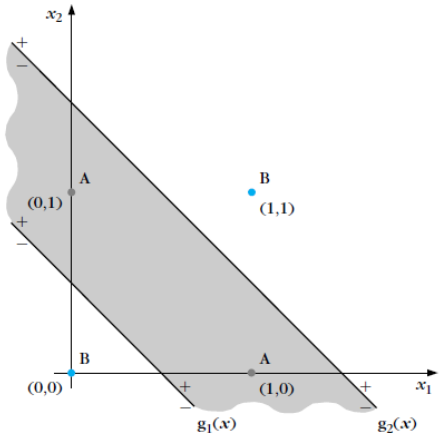


Table 4.3 Truth Table for the Two Computation Phases of the XOR Problem

1st Phase				
x_1	x_2	y_1	y_2	2nd Phase
0	0	0 (−)	0 (−)	B (0)
0	1	1 (+)	0 (−)	A (1)
1	0	1 (+)	0 (−)	A (1)
1	1	1 (+)	1 (+)	B (0)

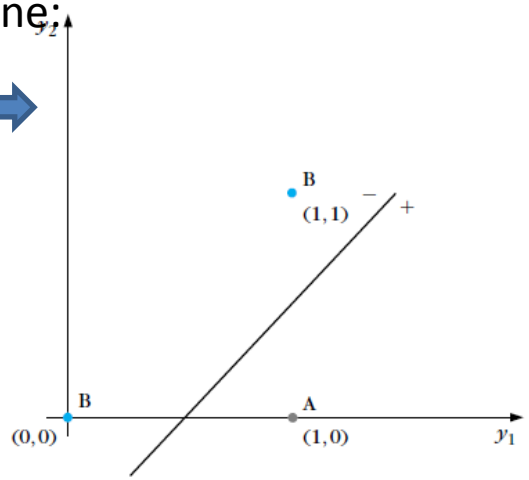


Figure 4.4 shows two such possible lines, $g_1(\boldsymbol{x}) = g_2(\boldsymbol{x}) = 0$, as well as the regions in space for which $g_1(\boldsymbol{x}) \gtrless 0, g_2(\boldsymbol{x}) \gtrless 0$. The classes can now be separated. Class A is to the right (+) of $g_1(\boldsymbol{x})$ and to the left (−) of $g_2(\boldsymbol{x})$. The region

den neurons and one output layer. We will show, constructively, that such an

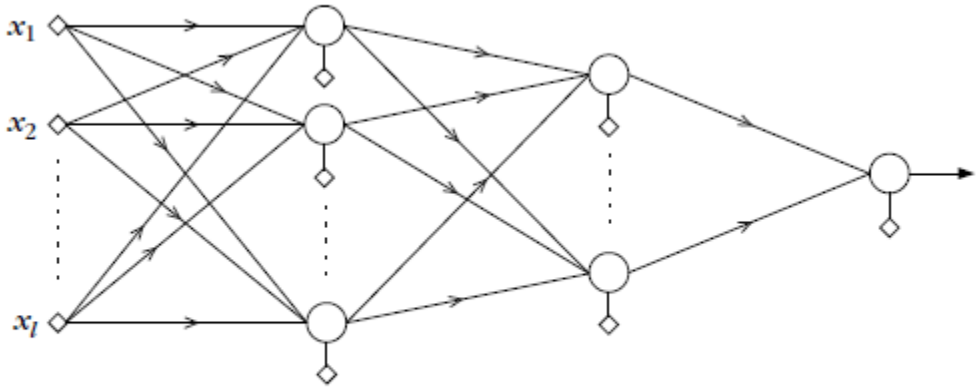


FIGURE 4.10

Architecture of a multilayer perceptron with two hidden layers of neurons and a single output neuron.

In summary, we can say that *the neurons of the first layer form the hyperplanes, those of the second layer form the regions, and finally the neurons of the output layer form the classes.*

Training:

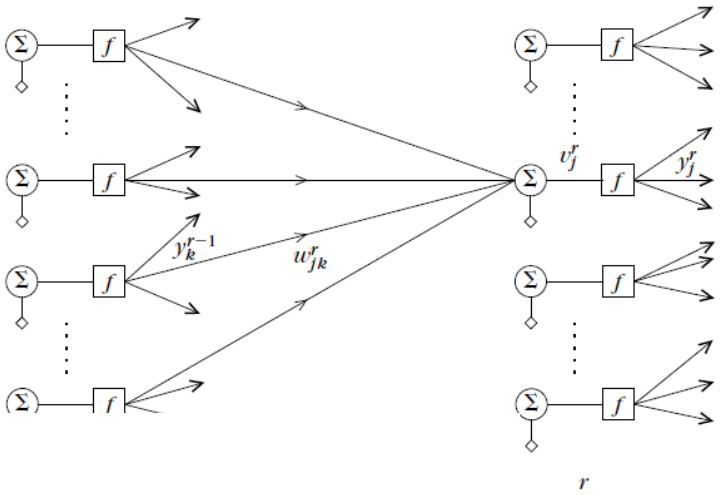
ALGORITHMS BASED ON EXACT CLASSIFICATION OF THE TRAINING SET

THE BACKPROPAGATION ALGORITHM

$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases}$$

$$\mathcal{E}(i) = \frac{1}{2}\sum_{m=1}^{k_L} e_m^2(i) \equiv \frac{1}{2}\sum_{m=1}^{k_L}(y_m(i) - \hat{y}_m(i))^2, \quad i = 1, 2, \ldots, N$$

$$J = \sum_{i=1}^{N} \mathcal{E}(i)$$



The Backpropagation Algorithm

- *Initialization:* Initialize all the weights with small random values from a pseudorandom sequence generator.
- *Forward computations:* For each of the training feature vectors $\boldsymbol{x}(i), i = 1, 2, \ldots, N$, compute all the $v_j^r(i)$, $y_j^r(i) = f(v_j^r(i)), j = 1, 2, \ldots, k_r$, $r = 1, 2, \ldots, L$, from (4.7). Compute the cost function for the current estimate of weights from (4.5) and (4.14).
- *Backward computations:* For each $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, k_L$ compute $\delta_j^L(i)$ from (4.15) and in the sequel compute $\delta_j^{r-1}(i)$ from (4.22) and (4.23) for $r = L, L-1, \ldots, 2$, and $j = 1, 2, \ldots, k_r$
- *Update the weights:* For $r = 1, 2, \ldots, L$ and $j = 1, 2, \ldots, k_r$

$$\boldsymbol{w}_j^r(\text{new}) = \boldsymbol{w}_j^r(\text{old}) + \Delta\boldsymbol{w}_j^r$$

$$\Delta\boldsymbol{w}_j^r = -\mu\sum_{i=1}^{N}\delta_j^r(i)\boldsymbol{y}^{r-1}(i)$$

n.

$$\boldsymbol{w}_j^r(\text{new}) = \boldsymbol{w}_j^r(\text{old}) + \Delta\boldsymbol{w}_j^r$$

$$\Delta\boldsymbol{w}_j^r = -\mu\frac{\partial J}{\partial \boldsymbol{w}_j^r}$$

$$\delta_j^{r-1}(i) = e_j^{r-1}(i)f'(v_j^{r-1}(i))$$

$$e_j^{r-1}(i) = \sum_{k=1}^{k_r}\delta_k^r(i)w_{kj}^r$$

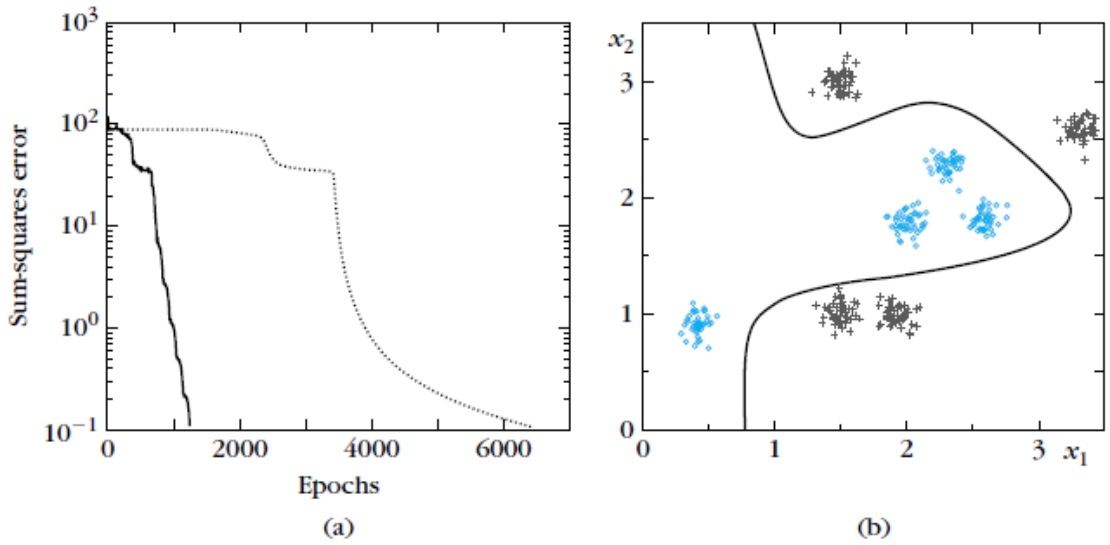


FIGURE 4.15

(a) Error convergence curves for the adaptive momentum (dark line) and the momentum algorithms. Note that the adaptive momentum leads to faster convergence. (b) The decision curve formed by the multilayer perceptron.

CHOICE OF THE NETWORK SIZE

THE COST FUNCTION CHOICE

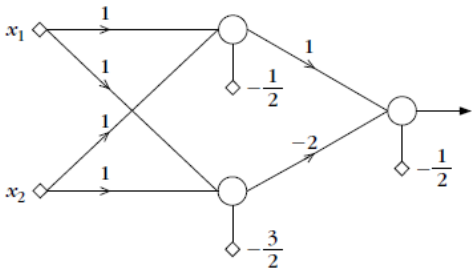
higher order network. $$f(v) = f\left(w_0 + \sum_i w_i x_i + \sum_{jk} w_{jk} x_j x_k\right)$$

circular backpropagation model $$f(v) = f\left(w_0 + \sum_i w_i x_i + w_s \sum_i x_i^2\right)$$

Where the weights are constrained from invariance requirements

GENERALIZED LINEAR CLASSIFIERS

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$$



$$\boldsymbol{x} \longrightarrow \boldsymbol{y}$$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} f(g_1(\boldsymbol{x})) \\ f(g_2(\boldsymbol{x})) \end{bmatrix}$$

Let $f_1(\cdot), f_2(\cdot), \ldots, f_k(\cdot)$ be nonlinear (in the general case) functions

$$f_i : \mathcal{R}^l \to \mathcal{R}, \quad i = 1, 2, \ldots, k$$

which define the mapping $\boldsymbol{x} \in \mathcal{R}^l \to \boldsymbol{y} \in \mathcal{R}^k$

$$\boldsymbol{y} \equiv \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_k(\boldsymbol{x}) \end{bmatrix} \tag{4.44}$$

Our goal now is to investigate whether there is an appropriate value for k and functions f_i so that classes A, B are linearly separable in the k-dimensional space of the vectors $\boldsymbol{y}$. In other words, we investigate whether there exists a k-dimensional space where we can construct a hyperplane $\boldsymbol{w} \in \mathcal{R}^k$ so that

$$w_0 + \boldsymbol{w}^T \boldsymbol{y} > 0, \quad \boldsymbol{x} \in A \tag{4.45}$$

$$w_0 + \boldsymbol{w}^T \boldsymbol{y} < 0, \quad \boldsymbol{x} \in B \tag{4.46}$$

Equivalent to searching for a non-linear hypersurface in the original space:

Assuming that in the original space the two classes were separable by a (non-linear) hypersurface $g(\boldsymbol{x}) = 0$, relations (4.45), (4.46) are basically equivalent to approximating the nonlinear $g(\boldsymbol{x})$ as a linear combination of $f_i(\boldsymbol{x})$, that is,

$$g(\boldsymbol{x}) = w_0 + \sum_{i=1}^{k} w_i f_i(\boldsymbol{x}) \tag{4.47}$$

This is a typical problem of function approximation in terms of a preselected class of *interpolation functions* $f_i(\cdot)$. This is a well-studied task in numerical analysis,

A similar expression to (4.47) expansion of $g(\boldsymbol{x})$ is known as *projection pursuit*, introduced in [Fried 81], and it is defined as

$$g(\boldsymbol{x}) = \sum_{i=1}^{k} f_i(\boldsymbol{w}_i^T \boldsymbol{x})$$

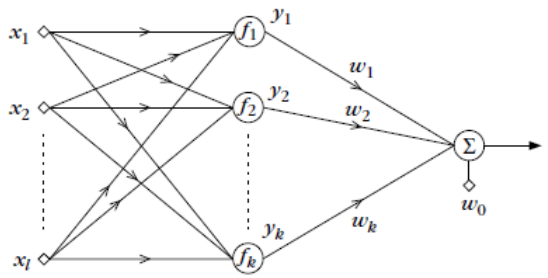


FIGURE 4.17
Generalized linear classifier.

Examples of interpolating functions:

$$g(\boldsymbol{x}) = w_0 + \sum_{i=1}^{l} w_i x_i + \sum_{i=1}^{l-1} \sum_{m=i+1}^{l} w_{im} x_i x_m + \sum_{i=1}^{l} w_{ii} x_i^2$$

$$f(\boldsymbol{x}) = \exp\left(-\frac{1}{2\sigma_i^2}\|\boldsymbol{x} - \boldsymbol{c}_i\|^2\right)$$

$$f(\boldsymbol{x}) = \frac{\sigma^2}{\sigma^2 + \|\boldsymbol{x} - \boldsymbol{c}_i\|^2}$$

$$y = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}$$

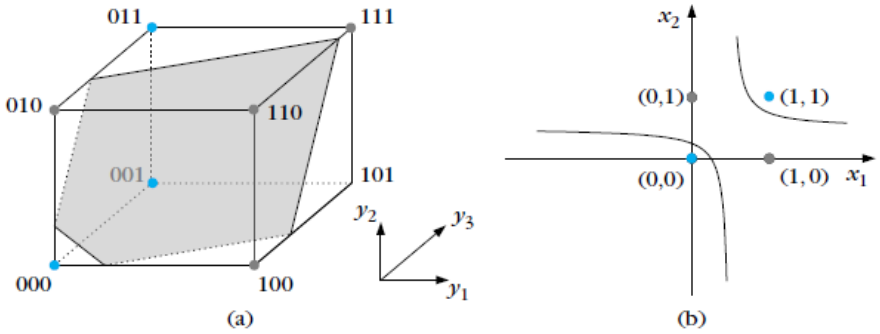


FIGURE 4.20

The XOR classification task, via the polynomial generalized linear classifier. (a) Decision plane in the three-dimensional space and (b) decision curves in the original two-dimensional space.

$$y = y(x) = \begin{bmatrix} \exp(-\|x - c_1\|^2) \\ \exp(-\|x - c_2\|^2) \end{bmatrix}$$

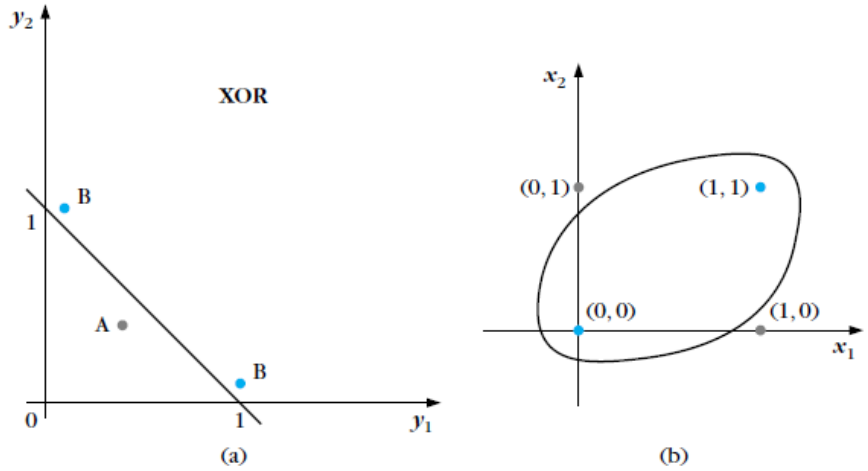


FIGURE 4.21

Decision curves formed by an RBF generalized linear classifier for the XOR task. The decision curve is linear in the transformed space (a) and nonlinear in the original space (b).

Separating capacity:

TABLE I

EXAMPLES OF SEPARATING SURFACES WITH THE CORRESPONDING NUMBER OF SEPARABLE DICHOTOMIES OF N POINTS IN m DIMENSIONS

Mapping ϕ Defined on x in E^m	Separating Surface	Degrees of Freedom of ϕ-Surface	General Position	Number of ϕ-Separable Dichotomies of N Points	Separating Capacity
$\phi(x)=x$	hyperplane through origin	m	no m points on any subspace	$C(N, m)$	$2m$
$\phi(x)=(1, x)$	hyperplane	$m+1$	no $m+1$ points on any hyperplane	$C(N, m+1)$	$2(m+1)$
$\phi(x)=(1, x, \|x\|^2)$	hypersphere	$m+2$	no $m+2$ points on any hypersphere	$C(N, m+2)$	$2(m+2)$
$\phi(x)=(x, \|x\|)$	hypercone	$m+1$	no $m+1$ points on any hypercone	$C(N, m+1)$	$2(m+1)$
$\phi(x)=$all r-wise products of components of x	rational rth-order variety	$\binom{m+r}{r}$	no $\binom{m+r}{r}$ points on any rth-order surface	$C\left(N, \binom{m+r}{r}\right)$	$2\binom{m+r}{r}$

SUPPORT VECTOR MACHINES: THE NONLINEAR CASE

Initially classification

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

$$= \sum_{i=1}^{N_s} \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + w_0$$

is + or −, where N_s is the number of support vectors. Thus, once more, only inner products enter into the scene. If the design is to take place in the new k-dimensional space, the only difference is that the involved vectors will be the k-dimensional mappings of the original input feature vectors. A naive look at it would lead to

$\boldsymbol{x} \in \mathcal{R}^l \rightarrow \boldsymbol{y} \in \mathcal{R}^k$

$$\boldsymbol{x} \in \mathcal{R}^2 \rightarrow \boldsymbol{y} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

$$\boldsymbol{y}_i^T \boldsymbol{y}_j = (\boldsymbol{x}_i^T \boldsymbol{x}_j)^2$$

Theorem *Mercer's Theorem. Let* $\boldsymbol{x} \in \mathcal{R}^l$ *and a mapping* $\boldsymbol{\phi}$

$$\boldsymbol{x} \mapsto \boldsymbol{\phi}(\boldsymbol{x}) \in H$$

where H *is a Hilbert space.*[5] *The inner product operation has an equivalent representation*

$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{z}) \rangle = K(\boldsymbol{x}, \boldsymbol{z}) \tag{4.63}$$

where $\langle \cdot, \cdot \rangle$ *denotes the inner product operation in* H *and* $K(\boldsymbol{x}, \boldsymbol{z})$ *is a symmetric continuous function satisfying the following condition:*

$$\int_C \int_C K(\boldsymbol{x}, \boldsymbol{z}) g(\boldsymbol{x}) g(\boldsymbol{z})\, d\boldsymbol{x}\, d\boldsymbol{z} \geq 0 \tag{4.64}$$

for any $g(\boldsymbol{x})$, $\boldsymbol{x} \in C \subset \mathcal{R}^l$ *such that*

$$\int_C g(\boldsymbol{x})^2\, d\boldsymbol{x} < +\infty \tag{4.65}$$

where C *is a compact (finite) subset of* $\mathcal{R}^l$. *The opposite is always true; that is, for* any *symmetric, continuous function* $K(\boldsymbol{x}, \boldsymbol{z})$ *satisfying (4.64) and (4.65) there exists a space in which* $K(\boldsymbol{x}, \boldsymbol{z})$ *defines an inner product! Such functions are also known as kernels and the space* H *as* Reproducing kernel Hilbert space (RKHS) *(e.g., [Shaw 04, Scho 02]). What Mercer's theorem does not disclose to us, however, is how to find this space. That is, we do not have a general tool to construct the mapping* $\boldsymbol{\phi}(\cdot)$ *once we know the inner product of the corresponding space. Furthermore, we lack the means to know the dimensionality of the space, which can even be infinite. This is the case, for example, for the radial basis (Gaussian) kernel ([Burg 99]). For more on these issues, the mathematically inclined reader is referred to [Cour 53].*

Once an appropriate kernel has been adopted that implicitly defines a mapping into a higher dimensional space (RKHS), the Wolfe dual optimization task (Eqs. (3.103)–(3.105)) becomes

$$\max_{\boldsymbol{\lambda}} \left(\sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right) \tag{4.69}$$

$$\text{subject to} \quad 0 \leq \lambda_i \leq C, \quad i = 1, 2, \ldots, N \tag{4.70}$$

$$\sum_i \lambda_i y_i = 0 \tag{4.71}$$

and the resulting linear (in the RKHS) classifier is

$$\text{assign } \boldsymbol{x} \text{ in } \omega_1(\omega_2) \text{ if } g(\boldsymbol{x}) = \sum_{i=1}^{N_s} \lambda_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + w_0 > (<)\, 0 \tag{4.72}$$

Typical examples of kernels used in pattern recognition applications are as follows:

Polynomials

$$K(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z} + 1)^q, \quad q > 0 \tag{4.66}$$

Radial Basis Functions

$$K(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{\sigma^2}\right) \tag{4.67}$$

Hyperbolic Tangent

$$K(\boldsymbol{x}, \boldsymbol{z}) = \tanh\left(\beta \boldsymbol{x}^T \boldsymbol{z} + \gamma\right) \tag{4.68}$$

Possible issues:

A major limitation of the support vector machines is that up to now there has been no efficient practical method for selecting the best kernel function. This is still an unsolved, yet challenging, research issue. Once a kernel function has been adopted, the so-called kernel parameters (e.g., σ for the Gaussian kernel) as well as the smoothing parameter, C, in the cost function are selected so that the error performance of the resulting classifier can be optimized. Indeed, this set of parameters, also known as hyperparameters, is crucial for the generalization capabilities of the classifier (i.e., its error performance when it is "confronted" with data outside the training set).

have access to the global optimum. The advantages of kernel-based training are therefore that we are guaranteed to find the globally optimal measurement over all possible quantum models. If the loss is convex, the optimisation problem is furthermore of a favourable structure that comes with a lot of guarantees about the performance and convergence of optimisation algorithms. But besides these great properties, in classical machine learning with big data, kernel methods were superseded by neural networks or approximate kernel methods [30] because of their poor scaling. Training involves computing the pair-wise distances between all training data in the Gram matrix of Eq. (88), which has at least a runtime of $\mathcal{O}(M^2)$ in the number of training samples M.[7] In contrast, training neural networks takes time $\mathcal{O}(M)$ that only depends linearly on the number of training samples. Can the training of variational quantum circuits offer a similar advantage over kernel-based training?

Alexandros:

PHYSICAL REVIEW LETTERS **122,** 040504 (2019)

Quantum Machine Learning in Feature Hilbert Spaces

Maria Schuld* and Nathan Killoran
Xanadu, 372 Richmond Street West, Toronto M5V 2L7, Canada

Supervised quantum machine learning models are kernel methods

Maria Schuld
Xanadu, Toronto, ON, M5G 2C8, Canada

Encoding data into quantum states is a feature map

$$|\phi(x)\rangle = U(x)\,|\psi\rangle$$

data-encoding quantum circuit $U(x)$

The data-encoding feature map gives rise to a kernel

Definition 2 (Quantum kernel). *Let ϕ be a data-encoding feature map over domain $\mathcal{X}$. A quantum kernel is the inner product between two data-encoding feature vectors $\rho(x), \rho(x')$ with $x, x' \in \mathcal{X}$,*

$$\kappa(x, x') = \mathrm{tr}\left[\rho(x')\rho(x)\right] = |\langle\phi(x')\,|\phi(x)\rangle|^2. \tag{6}$$

To justify the term "kernel" we need to show that the quantum kernel is indeed a positive definite function. The quantum kernel is a product of the complex-valued kernel $\kappa_c(x, x') = \langle\phi(x')\,|\phi(x)\rangle$ and its complex conjugate $\kappa_c(x, x')^* = \langle\phi(x)\,|\phi(x')\rangle = \langle\phi(x')\,|\phi(x)\rangle^*$. Since products of two kernels are known to be kernels themselves, we

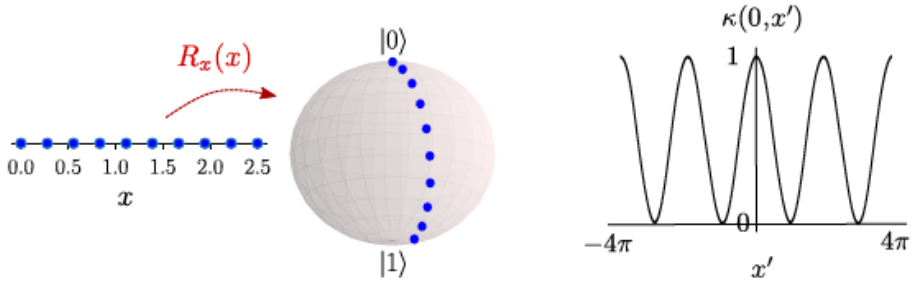


FIG. 6. **Example of a data-encoding feature map and quantum kernel.** A scalar input is encoded into a single-qubit quantum state, which is represented as a point on a Bloch sphere. The embedding uses a feature map facilitated by a Pauli-X rotation. As can be seen when plotting the quantum states encoding equidistant points on an interval, the embedding preserves the structure of the data rather well, but is periodic. The embedding gives rise to a quantum kernel κ. When we fix the first input at zero, we can visualise the distance measure, which is a squared cosine function.

EXAMPLES OF QUANTUM KERNELS

9

encoding	kernel $\kappa(x, x')$
basis encoding	$\delta_{x,x}$
amplitude encoding	$\lvert\mathbf{x}^\dagger\mathbf{x}'\rvert^2$
repeated amplitude encoding	$(\lvert\mathbf{x}^\dagger\mathbf{x}'\rvert^2)^r$
rotation encoding	$\prod_{k=1}^{N} \lvert\cos(x'_k - x_k)\rvert^2$
coherent state encoding	$e^{-\lvert\mathbf{x}-\mathbf{x}'\rvert^2}$
general near-term encoding	$\sum_{s,t\in\Omega} e^{i\mathbf{s}\mathbf{x}} e^{i\mathbf{t}\mathbf{x}'} c_{\mathbf{s},\mathbf{t}}$

"arbitrary state preparation"

Fourier representation of the quantum kernel

TABLE I. **Overview of data encoding strategies used in the literature and their quantum kernels.** If bold notation is used, the input domain is assumed to be the $\mathcal{X} \subseteq \mathbb{R}^N$.

10

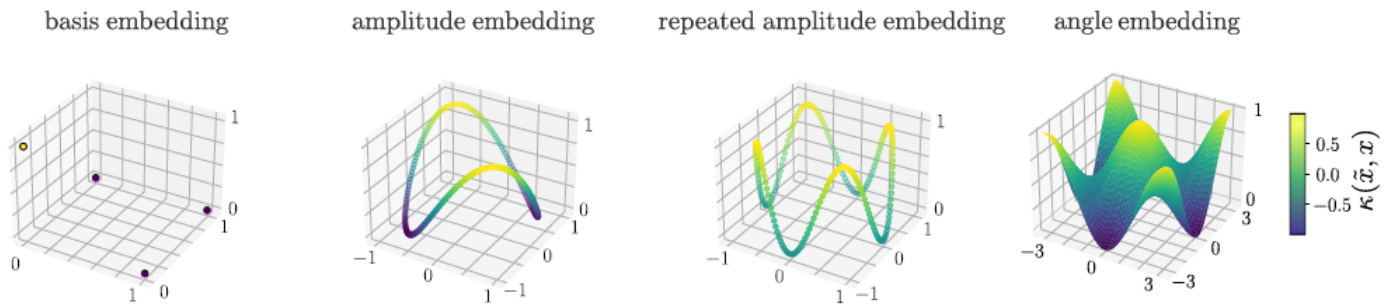


FIG. 7. **Quantum kernels of different data embeddings.** Plots of some of the functions $\kappa(\tilde{x}, x)$ for the kernels introduced above, using $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ for illustration purposes. The first entry $\tilde{\mathbf{x}}$ is fixed at $\tilde{\mathbf{x}} = (0, 0)$ for basis and rotation embedding, and at $\tilde{\mathbf{x}} = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ for the variations of amplitude embedding. The second value is depicted as the x-y plane.

Tasks:

- Optional: prove this..

$$|z\rangle = \frac{1}{\sqrt{\cosh(r)}} \sum_{n=0}^{\infty} \frac{\sqrt{(2n)!}}{2^n n!} [-e^{i\varphi} \tanh(r)]^n |2n\rangle$$

$$S(z) = \exp\left[\frac{1}{2} z a^\dagger a^\dagger - \frac{1}{2} z^* aa\right] , \qquad z = re^{i\phi} = z_1 + iz_2 . \tag{8}$$

By BCH relations the squeeze operator can be written as

$$\begin{aligned} S(z) &= \exp\left[\frac{1}{2} e^{i\phi} (\tanh r) a^\dagger a^\dagger\right] \left(\frac{1}{\cosh r}\right)^{(\frac{1}{2} + a^\dagger a)} \exp\left[-\frac{1}{2} e^{-i\phi} (\tanh r) aa\right] \qquad (9) \\ &= \exp\left[\frac{1}{2} e^{i\phi} (\tanh r) a^\dagger a^\dagger\right] (\cosh r)^{-1/2} \sum_{n=0}^{\infty} \frac{(\mathrm{sech} r - 1)^n}{n!} (a^\dagger)^n (a)^n \\ &\times \exp\left[-\frac{1}{2} e^{-i\phi} (\tanh r) aa\right] . \qquad (10) \end{aligned}$$

$$\begin{aligned} e^{-\widehat{H}} \widehat{C} e^{\widehat{H}} = \widehat{C} + \left[\widehat{C}, \widehat{H}\right] + \left[\left[\widehat{C}, \widehat{H}\right], \widehat{H}\right] /2! \\ + \left[\left[\left[\widehat{C}, \widehat{H}\right], \widehat{H}\right], \widehat{H}\right] /3! + \ldots \qquad \text{(A1)} \end{aligned}$$

- Then

$$\kappa(x, x'; c) = \prod_{i=1}^{N} \langle (c, x_i) | c, x_i' \rangle$$

with

$$\langle (c, x_i) | c, x_i' \rangle = \sqrt{\frac{\mathrm{sech}\, c\, \mathrm{sech}\, c}{1 - e^{i(x_i' - x_i)} \tanh\, c\, \tanh\, c}},$$

- Reproduce/understand this point

IV. LINEAR SEPARABILITY IN FOCK SPACE

We show here that if we map the inputs of a dataset $\mathcal{D}$ to a new dataset

$$\mathcal{D}' = \{|(c, \mathbf{x}^1)\rangle, ..., |(c, \mathbf{x}^M)\rangle\},$$

using the squeezing feature map with phase encoding from the main Letter, the data always becomes linearly separable in Fock space $\mathcal{F}$. Linear separability means that any assignment of two classes of labels to the data can be separated by a hyperplane in $\mathcal{F}$. We give a formal proof as well as numerical confirmation.

- Optional: Reproduce these results on the Implicit Approach

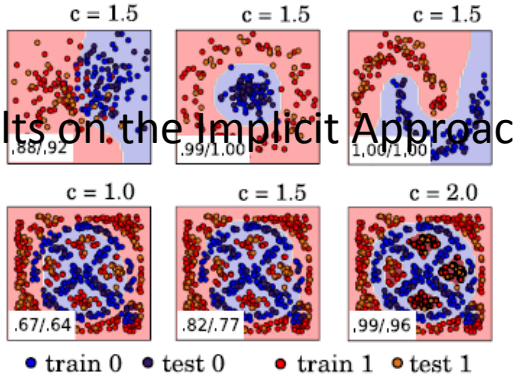
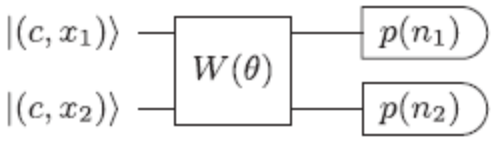


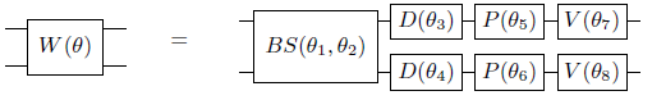
FIG. 3. Decision boundary of a support vector machine with the custom kernel from Eq. (5). The shaded areas show the decision regions for class 0 (blue) and class 1 (red), and each plot shows the rate of correct classifications on the training set or test set. The first row plots three standard two-dimensional datasets: "circles," "moons," and "blobs," each with 150 test and 50 training samples. The second row illustrates that increasing the squeezing hyperparameter c changes the classification performance. Here, we use a dataset of 500 training and 100 test samples. Training was performed with Python's *scikit-learn* SVC classifier using a custom kernel which implements the overlap of Eq. (6).

- Explicit Approach using non-Gaussian layer



$$p(2,0)/[p(2,0)+p(0,2)]$$

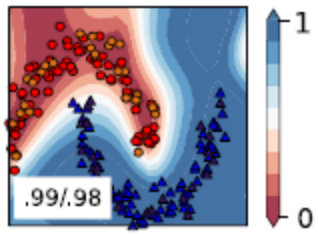
data in Fig. 5, using four repetitions of the layer in the variational circuit, and 32 parameters in total. The



$$BS(u,v) = \mathrm{e}^{u(\mathrm{e}^{iv}\hat{a}_1^\dagger \hat{a}_2 - \mathrm{e}^{-iv}\hat{a}_1 \hat{a}_2^\dagger)},$$

$$P(u) = \mathrm{e}^{i\frac{u}{2}\hat{x}^2}$$

$$V(u) = \mathrm{e}^{i\frac{u}{3}\hat{x}^3}.$$



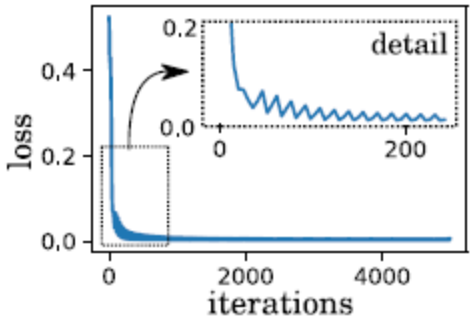


FIG. 5. Fock-space classifier presented in Fig. 4 and the text for the moons dataset. The shaded areas show the probability $p(y = 1)$ of predicting class 1. The model has been trained for 5000 steps with stochastic gradient descent of batch size 5, an adaptive learning rate and a square-loss cost function with a gentle l_2 regularization applied to all weights. The loss drops predominantly in the first 200 steps (left). The simulations were performed with the Quantum Machine Learning Toolbox (QMLT) application for the STRAWBERRYFIELDS software platform [43].

- Change parametric layer a bit and retry (other non-Gaussian operators)

$$\Sigma(\phi) = \sum_{n=0}^{\infty} e^{in(n-1)\phi/2} |n\rangle\langle n|$$

- Add a layer with free parameters before encoding
- Try as input squeezed number states and Gaussian layer

$$\begin{aligned}
S(z)|n\rangle &= e^{da^\dagger a^\dagger} \left(\frac{1}{\cosh r}\right)^{\frac{1}{2}+a^\dagger a} e^{-d^* aa}|n\rangle \\
&= e^{da^\dagger a^\dagger} \left(\frac{1}{\cosh r}\right)^{\frac{1}{2}+a^\dagger a} \sum_{j=0}^{\left[\frac{n}{2}\right]} \left[\frac{n!}{(n-2j)!}\right]^{1/2} \frac{(-d^*)^j}{j!}|n-2j\rangle \\
&= \left(\frac{1}{\cosh r}\right)^{n+1/2} (n!)^{1/2} \sum_{j=0}^{\left[\frac{n}{2}\right]} \frac{(-d^*)^j (\cosh r)^{2j}}{(n-2j)!j!} \\
&\quad \sum_{k=0}^{\infty} \frac{d^k[(n-2j+2k)!]^{1/2}}{k!}|n-2j+2k\rangle,
\end{aligned}$$

A couple of papers which worth reading:

Fock State-enhanced Expressivity of Quantum Machine Learning Models

Beng Yee Gan,[1,*] Daniel Leykam,[1] and Dimitris G. Angelakis[1,2,3,†]

[1] *Centre for Quantum Technologies, National University of Singapore, 3 Science Drive 2, Singapore 117543*
[2] *School of Electrical and Computer Engineering, Technical University of Crete, Chania, Greece 73100*
[3] *AngelQ Quantum Computing, 531A Upper Cross Street, #04-95 Hong Lim Complex, Singapore 051531*

(Dated: June 24, 2022)

Continuous-Variable Instantaneous Quantum Computing is Hard to Sample

T. Douce,[1,2,†] D. Markham,[2,3] E. Kashefi,[2,3,5] E. Diamanti,[2,3] T. Coudreau,[1] P. Milman,[1] P. van Loock,[4] and G. Ferrini[1,4,*]

[1] *Laboratoire Matériaux et Phénomènes Quantiques, Sorbonne Paris Cité, Université Paris Diderot, CNRS UMR 7162, 75013 Paris, France*
[2] *Laboratoire d'Informatique de Paris 6, CNRS, UPMC—Sorbonne Universités, 4 place Jussieu, 75005 Paris, France*
[3] *LTCI, CNRS, Télécom ParisTech, Université Paris Saclay, 75013 Paris, France*

Themis: Classification capacity of a qutrit

Data re-uploading for a universal quantum classifier

Adrián Pérez-Salinas[1,2], Alba Cervera-Lierta[1,2], Elies Gil-Fuster[3], and José I. Latorre[1,2,4,5]

In contrast to what we have seen there are adjustable parameters also in the encoding (kernels and nn: mixed up)

Core idea:

D-level quantum system versus 1 qubit + re-uploading
or versus D entangled qubits

Adopting a geometric interpretation possibly using Kernel formalism

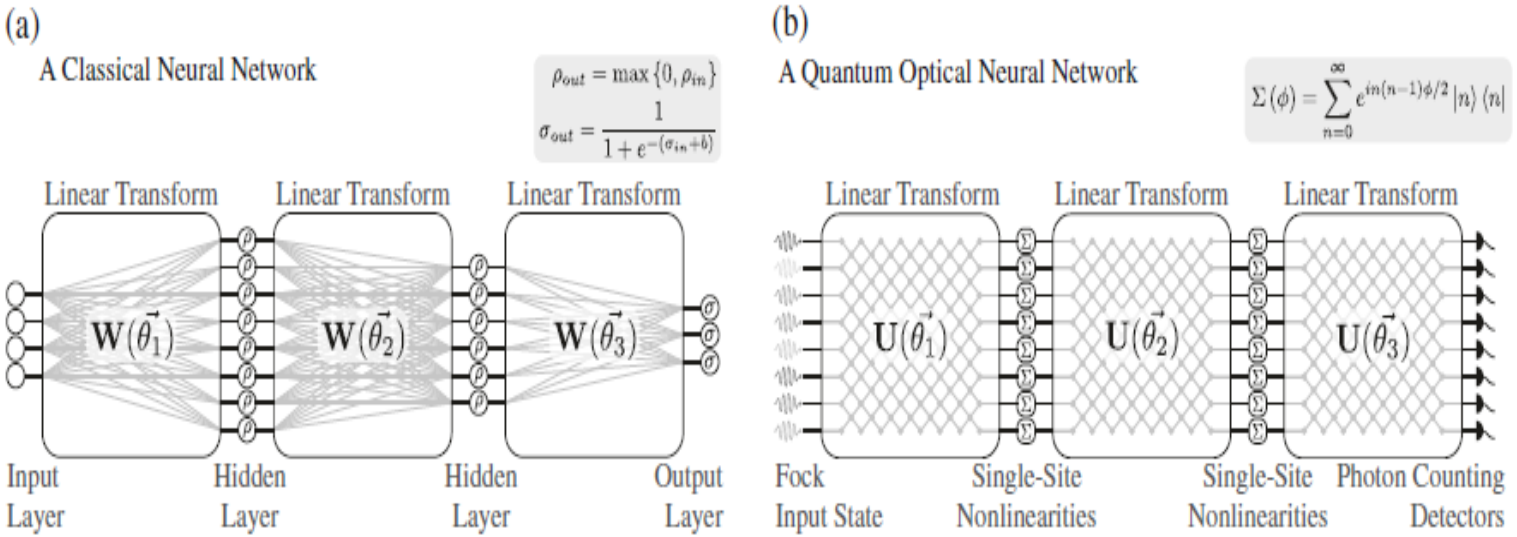


Fig. 1 Quantum optical neural network (QONN). **a** An example of a classical neural network architecture. Hidden layers are rectified linear units (ReLUs) and the output neuron uses a sigmoid activation function to map the output into the range (0, 1). **b** An example of our quantum optical neural network (QONN) architecture. Inputs are single photon Fock states. The single-site nonlinearities are given a Kerr-type interaction applying a phase quadratic in the number of photons. Readout is given by photon-number-resolving detectors, which measure the photon number at each output mode

Both encodings are performed by first compressing each of the four observation variables into $\gamma_j \in [0, \pi/2]$ ($j \in \{1, \ldots, 4\}$). For the direct encoding, each qubit q_j is set to $\cos(\gamma_j)|0\rangle_L + \sin(\gamma_j)|1\rangle_L$. For the QRAM encoding, the state over the two input qubits is set to $\frac{1}{4}\left(e^{i\gamma_1}|00\rangle_L + e^{i\gamma_2}|01\rangle_L + e^{i\gamma_3}|10\rangle_L + e^{i\gamma_4}|11\rangle_L\right)$. Finally, the QRAM encoding is given an ancilla qubit to act as phase reference.

Mapping Classical data on the Bloch hyper-sphere

$$|\psi\rangle = \sum_{k=0}^{D-1} c_k |k\rangle$$

$su(D)$ algebra for the system, spanned by D^2 generators $\{\hat{g}_i\}$

$$\hat{\rho} = \frac{1}{2}\hat{1} + \sum_{m=1}^{D^2-1} r_m \hat{g}_m$$

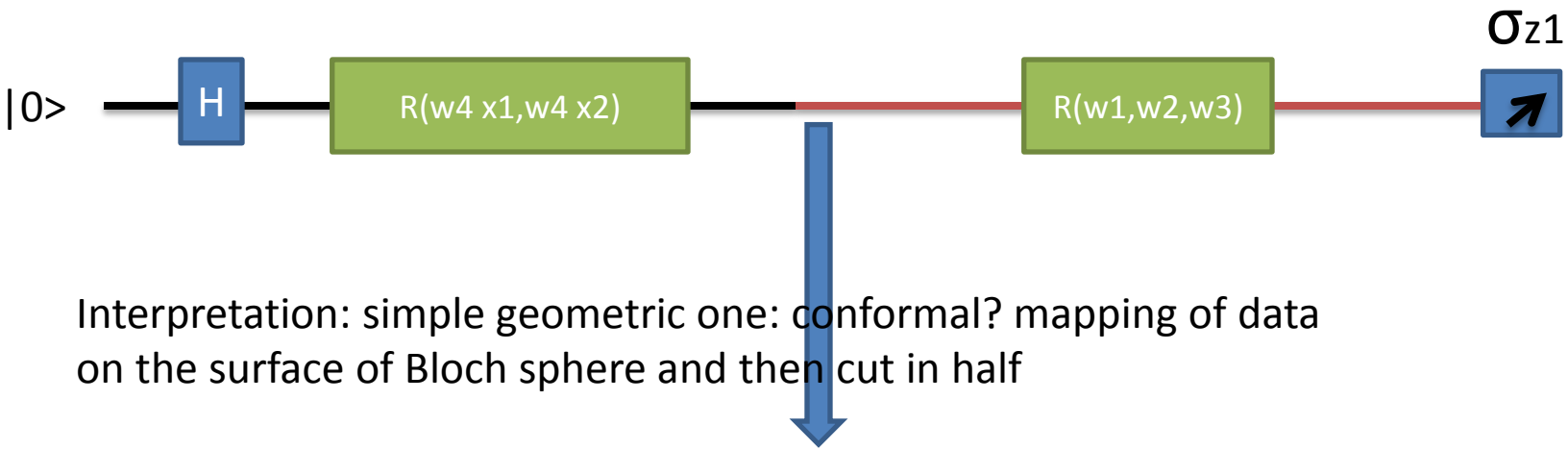
Most general operations

$$\hat{U} = e^{i\phi \sum_{m=0}^{D^2-1} n_m \hat{g}_m}.$$

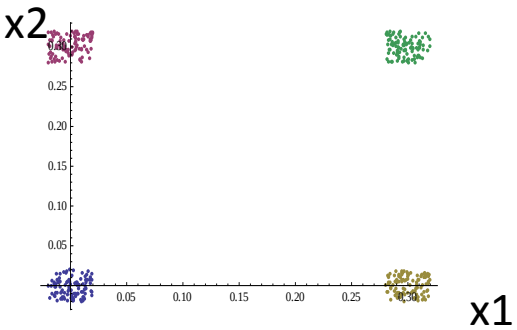
Some preliminary encouraging results:

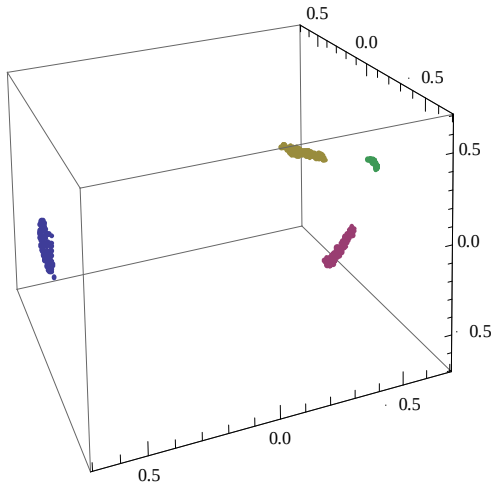
Single qubit XOR

1qubit_linear_classifiersTest_withoutCNOT.nb



Interpretation: simple geometric one: conformal? mapping of data on the surface of Bloch sphere and then cut in half





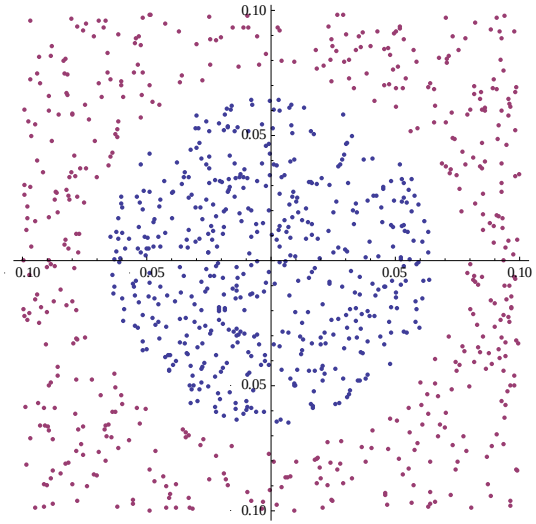
Works very well:
Train 6% of data-> classification 100%

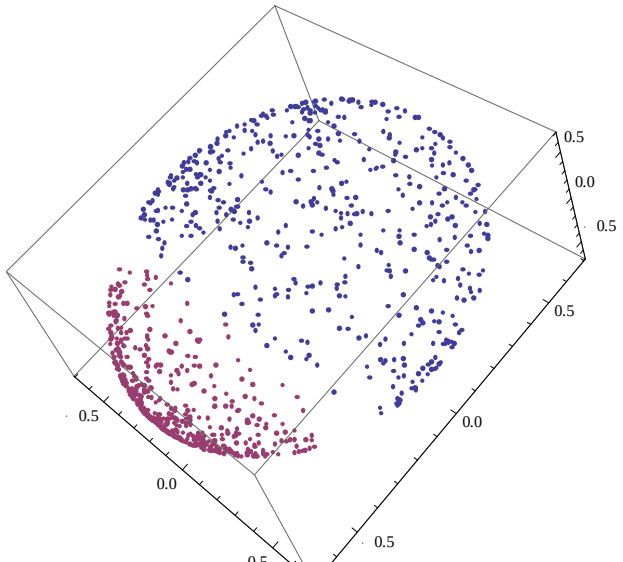
How to make it universal?
Universal approximation theorem

Single qubit circular problem

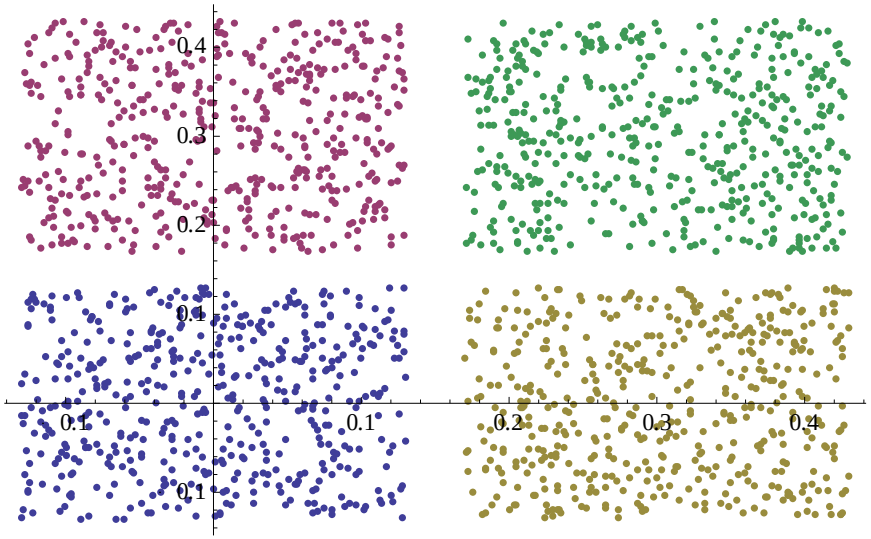
Interpretation: simple geometric one: conformal mapping of data on the surface of Bloch sphere and then cut in 2

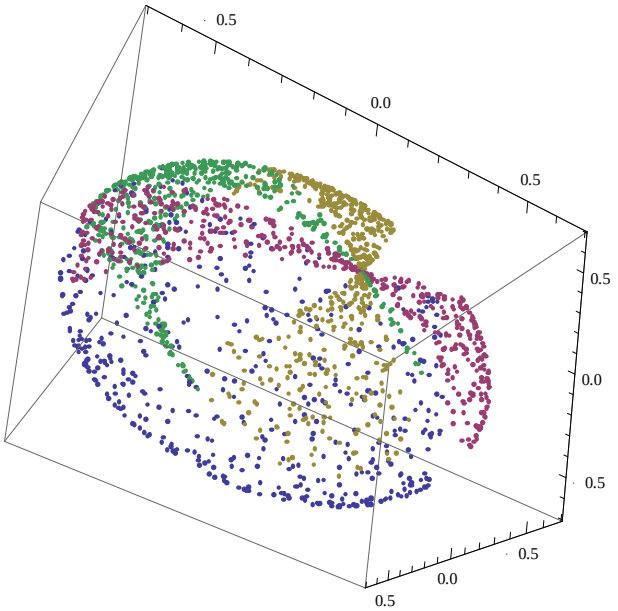
97% of success using 2% of data for training





But when I get to this case:





It does not work with a qubit even if I add one more parameter (on the angle or as a threshold

It works with a qutrit

qubit can accommodate at most 3 weights (according to our formulation and staying with classification the plane $N = 2$), one needs a higher D for more complicated problems. As a test we numerically solve the more challenging task presented in Fig.2 using a single qutrit and 8 parameters/weights. This can be fast solved with 1% of data for the training and a success ratio of classification is 86%. However I would like go back and check again if this problem could be solved with a qubit..

Tasks

- Calculate Kernels for different encodings
- Identify the separating capacity of the unit
- Break the rotations independent
- Barren plateaus
- Perform classification examples with meaning
- Try classification to more than 2 classes

General subjects of interest we can make some presentations:

Classical kernels
Kernels in Hilbert Space
Fourier representation of quantum Kernels
QRAM and arbitrary state preparation
Quantum support vector machines
Circuits for SWAP tests, overlap of unitary operations