Topographic Representation for Quantum Machine Learning

Bruce MacLennan
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville

October 17, 2018

A Goals

This chapter proposes a brain-inspired approach to quantum machine learning with the goal of circumventing many of the complications of other approaches. The fact that quantum processes are unitary presents both opportunities and challenges. A principal opportunity is that a large number of computations can be carried out in parallel in linear superposition, that is, quantum parallelism. The challenge is that the process is linear, and most approaches to machine learning depend significantly on nonlinear processes. Fortunately, the situation is not hopeless, for we know that nonlinear processes can be embedded in unitary processes, as is familiar from the circuit model of quantum computation. This chapter explores an approach to the quantum implementation of machine learning involving nonlinear functions operating on information represented topographically, as common in neural cortex.

B Topographic Representation in the Brain

One of the most common information representations used by the brain is the *topographic representation* or *computational map*. In such representations, distinct points $\mathbf{x}$ in some abstract space $\mathcal{X}$ are mapped systematically to physical locations $\mathbf{r} = \mu(\mathbf{x})$ in a two-dimensional region of cortex. For example, *tonotopic* maps have neurons that respond to different pitches that are arranged in order by pitch. *Retinotopic* maps have a spatial organization that mirrors the organization of the retina and

visual field. Neurons in primary visual cortex that respond to edges are arranged systematically according to the orientation of the edges.

In these topographic maps, a particular value $\mathbf{x}$ is represented by an activity peak in the corresponding cortical location $\mu(\mathbf{x})$. The strength of the activity reflects the value's degree of presence or probability. Moreover, multiple simultaneous values, with differing relative strengths or probabilities, are represented by multiple simultaneous activity peaks of differing amplitudes. Therefore, such cortical maps can represent superpositions of values with various amplitudes.

Because of their spatial representation of values, topographic maps can be used to implement arbitrary functions in the brain, essentially by a kind of table lookup. Suppose the brain needs to implement a (possibly nonlinear) transformation, $\mathbf{y} = f(\mathbf{x})$. This can be accomplished by neural connections from locations $\mathbf{r} = \mu(\mathbf{x})$ in the input map to corresponding locations $\mathbf{s} = \mu'(\mathbf{y}) = \mu'[f(\mathbf{x})]$ in the output map that represent the result. Thus activity representing $\mathbf{x}$ in the input map will cause corresponding activity representing $\mathbf{y}$ in the output map. Moreover, a superposition of input values will lead to a corresponding superposition of output values. Therefore, topographic representations allow the computation of nonlinear functions in linear superposition [3, 4, 5, 6], which suggests their usefulness in quantum computation [7]. On the other hand, topographic maps make relatively inefficient use of representational resources, because every represented value has to have a location in the map. Therefore, use of topographic representations will require a reasonably scalable quantum computing technology. In this chapter we explore topographic approaches to quantum computation with a focus on machine learning.

C Topographic Quantum Computation

In the brain, the state of a topographic map is a real-valued function defined over a (typically two-dimensional) space Ω. To apply these ideas in quantum computation, we consider a quantum state $|\psi\rangle$ in which the probability amplitude $\psi(\mathbf{r})$ at location $\mathbf{r} \in \Omega$ represents the value $\mathbf{x} \in \mathcal{X}$ via the correspondence $\mathbf{r} = \mu(\mathbf{x})$. Here $\mathbf{r} \in \Omega$ may be a continuous index representing, for example, spatial location, or a discrete quantum state, such as a wavelength or the state of a qubit register. In the continuous case, the input value $\mathbf{x}$ is represented by a state $|\mu(\mathbf{x})\rangle$, which is a Dirac unit impulse at $\mathbf{r}$: $|\mathbf{r}\rangle = \delta_{\mathbf{r}}$ where $\delta_{\mathbf{r}}(\mathbf{s}) = \delta(\mathbf{s} - \mathbf{r})$. Similarly an output value $\mathbf{y}$ is represented by the quantum state $|\mu'(\mathbf{y})\rangle = |\mu'[f(\mathbf{x})]\rangle = \delta_{\mu'(\mathbf{y})}$. The states $|\mathbf{r}\rangle$ (for $\mathbf{r} \in \Omega$) form a continuous basis for the input and output quantum states.

For such a continuous basis we can define a Hilbert-Schmidt linear operator

$$T_f = \int_{\mathcal{X}} \mathrm{d}\mathbf{x}\ |\mu'[f(\mathbf{x})]\rangle\langle\mu(\mathbf{x})|,$$

where $\mathcal{X}$ is the space of input values. (We write $T = T_f$ when f is clear from context.) This operator has the desired behavior: $|\mu'[f(\mathbf{x})]\rangle = T|\mu(\mathbf{x})\rangle$ for all $\mathbf{x} \in \mathcal{X}$. In this manner the linear operator T computes the nonlinear function f via the computational maps. We call such an operator a *graph kernel* because it uses the explicit *graph* of f [that is, the set of pairs $(\mu'[f(\mathbf{x})], \mu(\mathbf{x}))$ for $\mathbf{x} \in \mathcal{X}$] to do a kind of table lookup.[1]

Notice that if the input map is a superposition of input values, $|\psi\rangle = a|\mu(\mathbf{x})\rangle + b|\mu(\mathbf{x}')\rangle$, then the output map will be a superposition of the corresponding results: $T|\psi\rangle = a|\mu'[f(\mathbf{x})]\rangle + b|\mu'[f(\mathbf{x}')]\rangle$.

The reader might question the use of a continuous basis. First, note that for separable Hilbert spaces, the continuous basis can always be replaced by an infinite discrete basis. Second, the infinite discrete basis can be approximated by a finite discrete basis, an approximation that is especially appropriate for neural network machine learning, which requires only low-precision calculation.

We proceed to show several examples of nonlinear computations performed via quantum computational maps, beginning with a simple case and proceeding to more complex ones. For simplicity, we will ignore the map μ and consider computations from one quantum state to another. We consider both discrete domains, $\Omega = \{x_1, \ldots, x_n\}$, and one-dimensional continuous domains, $\Omega = [x_l, x_u]$. In both cases the vectors $\{|x\rangle \mid x \in \Omega\}$ are an orthonormal basis (composed of unit vectors in $\mathbb{C}^n$ for the discrete case and of delta functions in $\mathcal{L}^2(\Omega)$ for the continuous case). For example, in the discrete case, the values $x_1, \ldots, x_n$ might be represented by the composite state of an N-qubit register, where $n = 2^N$.

We begin with a bijective scalar function $f : [-1, 1] \to [-1, 1]$, such as tanh (appropriately restricted)[2], which is a useful sigmoid function for neural computation. The kernel to compute the operator is $T = \int_\Omega \mathrm{d}x\ |f(x)\rangle\langle x|$. Since f is bijective, the adjoint of T is

$$T^\dagger = \int_\Omega \mathrm{d}x\ |x\rangle\langle f(x)| = \int_\Omega \mathrm{d}y\ |f^{-1}(y)\rangle\langle y|.$$

It is easy then to see that T is unitary. In general, we have:

Proposition 1 *The graph kernel T of a continuous bijection $f : \Omega \to \Omega$ is unitary.*

[1] It is not the same as the graph kernels used in machine learning applied to graph theory.

[2] For example, $f(x) = \tanh(x)/\tanh(1)|_{[-1,1]}$.

Proof:

$$\begin{aligned}
T^\dagger T &= \left(\int_\Omega |f^{-1}(y)\rangle\langle y| \mathrm{d}y\right)\left(\int_\Omega |f(x)\rangle\langle x| \mathrm{d}x\right) \\
&= \int_\Omega\int_\Omega |f^{-1}(y)\rangle\langle y \mid f(x)\rangle\langle x| \mathrm{d}y\mathrm{d}x \\
&= \int_\Omega\int_\Omega |f^{-1}(y)\rangle\langle y \mid f(x)\rangle \mathrm{d}y\ \langle x| \mathrm{d}x \\
&= \int_\Omega |f^{-1}(f(x))\rangle\langle x| \mathrm{d}x \\
&= \int_\Omega |x\rangle\langle x| \mathrm{d}x \\
&= \mathrm{I}.
\end{aligned}$$

Likewise,

$$\begin{aligned}
TT^\dagger &= \left(\int_\Omega |f(x)\rangle\langle x| \mathrm{d}x\right)\left(\int_\Omega |f^{-1}(y)\rangle\langle y| \mathrm{d}y\right) \\
&= \int_\Omega\int_\Omega |f(x)\rangle\langle x \mid f^{-1}(y)\rangle\langle y| \mathrm{d}x\mathrm{d}y \\
&= \int_\Omega |f(f^{-1}(y))\rangle\langle y| \mathrm{d}y = \mathrm{I}.
\end{aligned}$$

Therefore T is unitary.

□

In the discrete case, let $y_i = f(x_i)$; the unit vectors $|y_i\rangle$ are a basis for $\mathbb{C}^n$. The kernel is $T = \sum_{i=1}^n |y_i\rangle\langle x_i| = \sum_{i=1}^n |f(x_i)\rangle\langle x_i|$, which is unitary. More directly, T is a permutation matrix and therefore orthogonal.

If the input is a weighted superposition of values, $|\psi\rangle = \int_\Omega \mathrm{d}x\ p(x)|x\rangle$, then applying the kernel will give a corresponding superposition of the outputs: $T|\psi\rangle = \int_\Omega \mathrm{d}x\ p(x)|f(x)\rangle$. The same applies, of course, in the discrete case. Since the graph kernel is unitary, the adjoint is the inverse: if $|\phi\rangle = \int_{\Omega'} \mathrm{d}y\ q(y)|y\rangle$, then $T^\dagger|\phi\rangle = \int_{\Omega'} \mathrm{d}y\ q(y)|f^{-1}(y)\rangle$.

D Arbitrary Finite Functions

D.1 Non-surjective injections

As a step towards the evaluation of arbitrary functions by means of computational maps, we begin with a relatively simple case: non-surjective injections. Therefore, let $\Omega = \{x_1, \ldots, x_n\}$ and $\Omega' = \{y_1, \ldots, y_m\}$, where $n < m$, and consider an injection $f : \Omega \to \Omega'$. Input maps will be in an n-dimensional Hilbert space $\mathcal{H}(\Omega)$ and output maps will be in an m-dimensional Hilbert space $\mathcal{H}(\Omega')$. Since $n < m$, ancillary constants will need to be provided from a space $\mathcal{H}_{\mathrm{C}}$, and so the complete input space will be in $\mathcal{H}(\Omega) \otimes \mathcal{H}_{\mathrm{C}}$. Our implementation will also generate "garbage" output in a space $\mathcal{H}_{\mathrm{G}}$, and so the complete output space will be in $\mathcal{H}(\Omega') \otimes \mathcal{H}_{\mathrm{G}}$. The input and output dimensions must be equal, and the simplest way to accomplish this is to make $\mathcal{H}_{\mathrm{C}}$ m-dimensional and $\mathcal{H}_{\mathrm{G}}$ n-dimensional, so that our operator is an mn-dimensional Hilbert-space transformation. Let $\{|w_1\rangle, \ldots, |w_m\rangle\}$ be a basis for $\mathcal{H}_{\mathrm{C}}$ and let $\{|v_1\rangle, \ldots, |v_n\rangle\}$ be a basis for $\mathcal{H}_{\mathrm{G}}$.

Our goal will be to define a unitary U so that $U|x\rangle|w_1\rangle = |f(x)\rangle|\gamma\rangle$, where $|w_1\rangle$ is an ancillary constant and $|\gamma\rangle$ is garbage. As we will see, U can be implemented by an appropriate permutation of the input basis into the output basis, which comprises several functions $U = T + S + R + Q$. The work of the function f is accomplished by the T component:

$$T \stackrel{\text{def}}{=} \sum_{j=1}^{n} |f(x_j)\rangle|v_1\rangle\langle x_j|\langle w_1|.$$

Note that $T|x_j\rangle|w_1\rangle = |f(x_j)\rangle|v_1\rangle$, but T is not unitary.

The S component ensures that non-range elements of the codomain have preimages in the domain. Therefore, let m_{nr} be the number of codomain elements that are not in the range of f, $m_{\mathrm{nr}} = |\Omega' \setminus \operatorname{Im} f| = m - n$. Call these nonrange codomain elements $\{z_1, \ldots, z_{m_{\mathrm{nr}}}\} \subset \Omega'$. Then the S component is defined:

$$S \stackrel{\text{def}}{=} \sum_{i=1}^{m_{\mathrm{nr}}} |z_i\rangle|v_1\rangle\langle x_1|\langle w_i|.$$

To complete the unitary operator we add the following additional components:

$$R \stackrel{\text{def}}{=} \sum_{i=2}^{m} \sum_{j=2}^{n} |y_i\rangle|v_j\rangle\langle x_i|\langle w_j|,$$

$$Q \stackrel{\text{def}}{=} \sum_{j=1}^{n-1} |y_1\rangle|v_{j+1}\rangle\langle x_1|\langle w_{m_{\mathrm{nr}}+j}|.$$

Notice that every input basis vector maps into exactly one output basis vector and vice versa. Summing the basis vector dyads for T, S, R, Q gives:

$$n + m_{\text{nr}} + (m-1)(n-1) + (n-1) = m + (m-1)(n-1) + n - 1 = mn.$$

Proposition 2 *Let Ω and Ω' be finite sets with $n = |\Omega|$, $m = |\Omega'|$, and $m > n$. Let $f : \Omega \to \Omega'$ be a non-surjective injection. Let $\mathcal{H}_{\text{C}}$ and $\mathcal{H}_{\text{G}}$ be Hilbert spaces of dimension m and n, respectively (representing ancillary constant inputs and garbage outputs). Let $|\omega\rangle$ be a fixed basis vector of $\mathcal{H}_{\text{C}}$ and $|\upsilon\rangle$ be a fixed basis vector of $\mathcal{H}_{\text{G}}$. Then there is a unitary operator $U \in \mathcal{L}[\mathcal{H}(\Omega) \otimes \mathcal{H}_{\text{C}}, \mathcal{H}(\Omega') \otimes \mathcal{H}_{\text{G}}]$ such that for all $x \in \Omega$,*

$$U(|x\rangle \otimes |\omega\rangle) = |f(x)\rangle \otimes |\upsilon\rangle.$$

Proof: The proposition follows from the construction preceding the proposition.

□

If the input to U is a (normalized) superposition, $|\psi\rangle = \sum_k p_k |x_k\rangle$, then the output will be a superposition of the corresponding function results: $U|\psi\rangle|\omega\rangle = \sum_k p_k |f(x_k)\rangle|\upsilon\rangle$.

The dimension of the input and output spaces of this implementation is mn. A more resource efficient but also more complicated implementation operates on a space of dimension $\text{LCM}(m, n)$. The principle is the same: a permutation of the basis vectors.

D.2 Non-injective surjections

Next we consider functions $f : \Omega \twoheadrightarrow \Omega'$ that are surjections but not injections; that is, f maps onto Ω' but might not be one-to-one. This includes many useful functions, such as non-injective squashing functions and Gaussians, but also binary functions such as addition and multiplication (as explained later).

A non-injective function loses information, and thus it must be embedded in a larger injective function, which moreover must be unitary. In particular, if f is non-injective (e.g., $f(x) = f(x')$ for some $x \neq x'$), then the corresponding graph kernel will also be non-injective: $T|x\rangle = |y\rangle = T|x'\rangle$ for $|x\rangle \neq |x'\rangle$. Therefore $T(|x\rangle - |x'\rangle) = \mathbf{0}$, which implies that $|x\rangle - |x'\rangle$ is in the null space of T, $\mathcal{N}(T)$. Therefore there is a bijection between the orthogonal complement of the null space, $\mathcal{N}(T)^{\perp}$, and the range of the operator, $\text{Im}\, T$. Hence we can implement the non-injective operation

by decomposing the input $|\psi\rangle$ into orthogonal components $|\psi\rangle = |\mu\rangle + |\nu\rangle$, where $|\mu\rangle \in \mathcal{N}(T)^\perp$ and $|\nu\rangle \in \mathcal{N}(T)$. There is a bijection $|\mu\rangle \leftrightarrow T|\psi\rangle$.

To explain how this can be accomplished, we consider the finite-dimensional case, but it is easily extended. Let $\Omega = \{x_1, \ldots, x_n\}$ and $\Omega' = \{y_1, \ldots, y_m\}$; since f is surjective, $m \leq n$.

The desired operator $T \in \mathcal{L}(\mathcal{H}, \mathcal{H}')$, where $\mathcal{H} = \mathcal{H}(\Omega)$ is an n-dimensional Hilbert space with basis $\{|x_1\rangle, \ldots, |x_n\rangle\}$. The output space $\mathcal{H}'$ is also n-dimensional, and m of its basis vectors $|y_1\rangle, \ldots, |y_m\rangle$ are used to represent a topographic map of the function's codomain (and range), $\operatorname{Im} f = \Omega' = \{y_1, \ldots, y_m\}$. Therefore $\mathcal{H}(\Omega')$ is a subspace of $\mathcal{H}'$. Let $\{|\mathbf{w}_1\rangle, \ldots, |\mathbf{w}_{n-m}\rangle\}$ be a basis for $\mathcal{H}(\Omega')^\perp$, the orthogonal complement of $\mathcal{H}(\Omega')$ in $\mathcal{H}'$. (This subspace will represent "garbage" with no computational relevance.)

We will define $|\mathbf{u}_1\rangle, \ldots |\mathbf{u}_m\rangle$ to be an orthonormal (ON) basis for $\mathcal{N}(T)^\perp$ (the row space of T), where m is the rank of T; and $|\mathbf{v}_1\rangle, \ldots, |\mathbf{v}_{n_o}\rangle$ to be an ON basis for $\mathcal{N}(T)$, where $n_o = n - m$ is the nullity of T. These bases will determine the orthogonal components $|\mu\rangle \in \mathcal{N}(T)^\perp$ and $|\nu\rangle \in \mathcal{N}(T)$ into which any input is separated.

For each $y_i \in \Omega'$, let $f^{-1}\{y_i\} \stackrel{\text{def}}{=} \{x \mid f(x) = y_i\}$ be the inverse image of y_i; these are disjoint subsets of the domain Ω and correspond to orthogonal subspaces of $\mathcal{H}$. Let $n_i \stackrel{\text{def}}{=} |f^{-1}\{y_i\}|$ be the *preimage multiplicity* of y_i, where $n = n_1 + \cdots + n_m$. Because different $y_i \in \operatorname{Im} T$ have different preimage multiplicities, it will be convenient to separate T into m constant functions $T_i : f^{-1}\{y_i\} \to \{y_i\}$. Therefore let

$$T_i \stackrel{\text{def}}{=} \frac{1}{\sqrt{n_i}} \sum_{x_j \in f^{-1}\{y_i\}} |y_i\rangle\langle x_j| = |y_i\rangle \frac{1}{\sqrt{n_i}} \sum_{x_j \in f^{-1}\{y_i\}} \langle x_j| = |y_i\rangle\langle \mathbf{u}_i|, \tag{1}$$

where we define the normalized basis vectors of $\mathcal{N}(T)^\perp$:

$$|\mathbf{u}_i\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{n_i}} \sum_{x_j \in f^{-1}\{y_i\}} |x_j\rangle, i = 1, \ldots, m. \tag{2}$$

The T_i operate independently on orthogonal subspaces of $\mathcal{H}(\Omega)$.

Note that $\langle \mathbf{u}_i \mid x_j\rangle \neq 0$ if and only if $y_i = f(x_j)$, and in this case $T_i|x_j\rangle = \frac{1}{\sqrt{n_i}}|y_i\rangle$. (The $1/\sqrt{n_i}$ factor is required for normalization of the $|\mathbf{u}_i\rangle$.) Clearly $\{|\mathbf{u}_1\rangle, \ldots |\mathbf{u}_m\rangle\}$ is an ON set, since its elements are normalized linear combinations of disjoint sets of the basis vectors. Therefore there is a one-to-one correspondence between the output vectors $|y_i\rangle$ and the basis vectors $|\mathbf{u}_i\rangle$.

Next we characterize $\mathcal{N}(T)$ and $\mathcal{N}(T)^\perp$. Observe that $|\psi\rangle \in \mathcal{N}(T_i)$ if and only if $\mathbf{0} = T_i|\psi\rangle = |y_i\rangle\langle \mathbf{u}_i \mid \psi\rangle$, that is, if and only if $\langle \mathbf{u}_i \mid \psi\rangle = 0$. Therefore, $\mathcal{N}(T_i)$ is the

$n-1$ dimensional subspace orthogonal to $|\mathbf{u}_i\rangle$, and $\mathcal{N}(T_i)^\perp$ is the one-dimensional subspace spanned by $\{|\mathbf{u}_i\rangle\}$.

The operation T is implemented in two parts, one that handles the components representing the null space and the other that handles the components representing its orthogonal complement (and does the work of computing f).

We have $|\mathbf{v}_j\rangle \in \mathcal{H}$, the basis vectors for the $\mathcal{N}(T)$ subspace, which has dimension $n_o = n - m$. Let $\{|\mathbf{w}_j\rangle \mid j = 1, \ldots n_o\}$ be any basis for $\mathcal{H}(\Omega')^\perp$, the orthogonal complement of $\mathcal{H}(\Omega')$ in $\mathcal{H}'$. Then define $N \in \mathcal{L}(\mathcal{H}, \mathcal{H}')$ to map the nullspace components down to this n_o-dimensional space:

$$N = \sum_{j=1}^{n_o} |\mathbf{w}_j\rangle\langle\mathbf{v}_j|.$$

Since there is a one-one correspondence between basis vectors $|\mathbf{u}_i\rangle$ and output vectors $|y_i\rangle$, we implement the function f by an operator $M \in \mathcal{L}(\mathcal{H}, \mathcal{H}')$ defined

$$M = \sum_{i=1}^{m} |y_i\rangle\langle\mathbf{u}_i|.$$

This maps the n-dimensional input into an m-dimensional output subspace.

Then the operator $T \stackrel{\text{def}}{=} M + N \in \mathcal{L}(\mathcal{H}, \mathcal{H}')$ maps an input $|x\rangle$ to the correct output $|f(x)\rangle$, but with a scale factor and additional "garbage." For $x_j \in f^{-1}\{y_i\}$,

$$T|x_j\rangle = (M+N)|x_j\rangle = M|x_j\rangle + N|x_j\rangle = \frac{1}{\sqrt{n_i}}|y_i\rangle + |\gamma_j\rangle.$$

where $|\gamma_j\rangle = N|x_j\rangle$ and $\||\gamma_j\rangle\| = \sqrt{\frac{n_i - 1}{n_i}}$. Note that the garbage $|\gamma_j\rangle$ is in the same quantum register as the desired output. Subsequent computations operate on the $\mathcal{H}(\Omega') = \{y_1, \ldots, y_m\}$ subspace and ignore the orthogonal subspace, which contains the garbage (which nevertheless must be retained, since it is entangled with the computational results).

The T operator is just a transformation from the $\{|x_1\rangle, \ldots, |x_n\rangle\}$ basis to the $\{|y_1\rangle, \ldots, |y_m\rangle, |\mathbf{w}_1\rangle, \ldots, |\mathbf{w}_{n-m}\rangle\}$ basis, and is obviously unitary. Therefore it can be approximated arbitrarily closely by a combination of H (Hadamard), CNOT (conditional NOT), and T ($\pi/8$) gates [8, §4.5].

Unfortunately, the output vectors $|y_i\rangle$ have amplitudes that depend on their preimage multiplicities. That is, if $y_i = f(x_j)$, then $T|x_j\rangle = \frac{1}{\sqrt{n_i}}|y_i\rangle + |\gamma_j\rangle$, and we get different scale factors $\frac{1}{\sqrt{n_i}}$ depending on the preimage multiplicity. For $y_i = f(x_j)$,

define $s_j \stackrel{\text{def}}{=} 1/\sqrt{n_i}$ to be this scale factor, so that $T|x_j\rangle = s_j|f(x_j)\rangle + |\gamma_j\rangle$. We would like to equalize the differing amplitudes but there does not seem to be a unitary operation that will do so.

It might seem that something like a Grover iteration [2] could be used to rotate the state vector from $s_j|y_i\rangle + |\gamma_j\rangle$ to $|y_i\rangle$, but different s_i require different numbers of iterations. Something like Grover's algorithm with an unknown number of solutions could be used, but this would require trying multiple rotations. Therefore, it seems better to accept the unwanted scale factors and work with them. This means that any $|y_i\rangle$ with positive amplitudes are outputs from the computation, and therefore all positive amplitudes are treated the same.

If we ignore the relative magnitudes of positive amplitudes, then a quantum state $|\psi\rangle = \sum_j p_j|x_j\rangle$ (with $p_j \geq 0$) can be interpreted as the set of all x_j with positive amplitudes: $\{x_j \in \Omega \mid p_j > 0\}$, where we assume of course that $\sum_j p_j^2 \leq 1$. Moreover, the sum can be strictly less than one only if there are additional ancillary states that make up the difference (like $|\gamma_j\rangle$ in the previous example). Applying T to such a state computes a state representing the image of the input set. That is, $T|\psi\rangle = \sum_j p_j s_j|f(x_j)\rangle$, which represents the set $\{f(x_j) \in \Omega \mid p_j > 0\}$. If $S \subseteq \Omega$ is the set represented by $|\psi\rangle$, then $T|\psi\rangle$ represents its image $f[S]$. Since zero amplitudes will always map to zero amplitudes and positive amplitudes will map to positive amplitudes, set membership will be appropriately mapped from the domain to the codomain.

Proposition 3 *Let $\Omega = \{x_1, \ldots, x_n\}$ and $\Omega' = \{y_1, \ldots, y_m\}$ be finite sets with $m \leq n$. Suppose $\{|x_1\rangle, \ldots, |x_n\rangle\}$ is an ON basis for a Hilbert space $\mathcal{H}$ and $\{|y_1\rangle, \ldots, |y_m\rangle\}$ is an ON basis for a subspace of $\mathcal{H}$. For any function $f : \Omega \twoheadrightarrow \Omega'$ there is a unitary operator $T \in \mathcal{L}(\mathcal{H}, \mathcal{H})$ such that for any $x \in \Omega$,*

$$T|x\rangle = \frac{1}{\sqrt{n_x}}|f(x)\rangle + |\gamma\rangle,$$

where $n_x = |f^{-1}\{f(x)\}|$, $|\gamma\rangle \in \mathcal{H}(\Omega')^{\perp}$, and $\| |\gamma\rangle \| = \sqrt{\frac{n_x - 1}{n_x}}$.

Proof: As previously shown, this operator is given explicitly by

$$T = \sum_{i=1}^{m} |y_i\rangle\langle \mathbf{u}_i| + \sum_{k=1}^{n-m} |\mathbf{w}_k\rangle\langle \mathbf{v}_k|,$$

where $|\mathbf{u}_i\rangle = \frac{1}{\sqrt{n_i}} \sum_{x \in f^{-1}\{y_i\}} |x\rangle$, $n_i = |f^{-1}\{y_i\}|$, the $|\mathbf{v}_k\rangle$ are an ON basis for the orthogonal complement of the space spanned by the $|\mathbf{u}_i\rangle$, and the $|\mathbf{w}_k\rangle$ are an ON basis for $\mathcal{H}(\Omega')^{\perp}$.

□

D.3 Arbitrary Functions

In the preceding, we have assumed for convenience that the function is either injective or surjective, but not both. The solutions are easily extended to arbitrary functions. since every function can be factored as a composition of an injection and a surjection. We can use Prop. 3 to implement the function as a surjection onto its range, and then use the Prop. 2 to inject its range into its codomain. Let the domain $\Omega = \{x_1, \ldots, x_n\}$, where $n = |\Omega|$, and let $\{|x_1\rangle, \ldots, |x_n\rangle\}$ be the standard basis of $\mathcal{H}(\Omega)$. Let $x_0 \notin \Omega$ be an additional value, and define the extended domain $\Omega^\circ = \{x_0\} \cup \Omega$. Then $\mathcal{H}(\Omega^\circ)$ has basis $\{|x_0\rangle, \ldots, |x_n\rangle\}$. (For example, $\mathcal{H}(\Omega^\circ)$ may be the state space of N qubits, where $n + 1 = 2^N$.) The additional $|x_0\rangle$ dimension will carry the nullspace "garbage."

Similarly, let the codomain $\Omega' = \{y_1, \ldots, y_m\}$, where $m = |\Omega'|$, and let $\{|y_1\rangle, \ldots, |y_m\rangle\}$ be the standard basis of $\mathcal{H}(\Omega')$. Let $y_0 \notin \Omega'$ be an additional value, and define the extended domain $\Omega^* = \{y_0\} \cup \Omega'$. Then $\mathcal{H}(\Omega^*)$ has basis $\{|y_0\rangle, \ldots, |y_m\rangle\}$. The $|y_0\rangle$ component carries the garbage in the output state.

Let $\{y'_1, \ldots, y'_{m_\mathrm{r}}\} \stackrel{\mathrm{def}}{=} \mathrm{Im}\, f$ and $\{z_1, \ldots, z_{m_\mathrm{nr}}\} \stackrel{\mathrm{def}}{=} \Omega \setminus \mathrm{Im}\, f$ be the range of f and its complement, respectively; $n = m_\mathrm{r} + m_\mathrm{nr}$. As before, an n-dimensional input $|x_j\rangle$ will be projected into orthogonal subspaces (the non-null and null spaces) of dimension m_r and $n_o = n - m_\mathrm{r}$, with basis vectors $|\mathbf{u}_i\rangle$ and $|\mathbf{v}_k\rangle$, respectively.

An additional input quantum register will be used to provide the zero amplitudes for non-surjective functions. The $m + 1$ dimensional state of this register will be in, for convenience, $\mathcal{H}_\mathrm{C} = \mathcal{H}(\Omega^*)$ with basis $\{|y_0\rangle, \ldots, |y_m\rangle\}$. There will also be an additional output quantum register to hold the nullspace garbage for non-injective functions. Its $n + 1$ dimensional state is in, for convenience, $\mathcal{H}_\mathrm{G} = \mathcal{H}(\Omega^\circ)$ with basis $\{|x_0\rangle, \ldots, |x_n\rangle\}$. Note that both the input and output spaces have dimension $(m + 1)(n + 1)$.

Our goal is to define unitary $U \in \mathcal{L}[\mathcal{H}(\Omega^\circ) \otimes \mathcal{H}_\mathrm{C}, \mathcal{H}(\Omega^*) \otimes \mathcal{H}_\mathrm{G}]$ so that

$$U[(s|x_j\rangle + r|x_0\rangle) \otimes |y_0\rangle] = (s'|f(x_j)\rangle + r'|x_0\rangle) \otimes |\gamma\rangle,$$

for scalars s, s', r, r' and for $|\gamma\rangle \in \mathcal{H}_\mathrm{G}$. Therefore the argument of f is in the first input register, and its result is in the first output register, possibly with garbage in both its input and output $|x_0\rangle$ components. The input ancillary register is initialized to a constant $|y_0\rangle$.

The graph kernel will map the $|\mathbf{u}_i\rangle|y_0\rangle$ vectors into corresponding $(m+1)(n+1)$ dimensional output vectors $|y'_i\rangle|x_0\rangle$ in the output space $\mathcal{H}(\Omega^\circ) \otimes \mathcal{H}_\mathrm{G}$. To accomplish

this, define M as follows:

$$M \stackrel{\text{def}}{=} \sum_{i=1}^{m_{\text{r}}} |y'_i, x_0\rangle\langle \mathbf{u}_i, y_0|.$$

Another component of the transform will map the nullspace components $|\mathbf{v}_k\rangle|y_0\rangle$ into the $m+1$ dimensional $|y_0\rangle$ subspace:

$$N \stackrel{\text{def}}{=} \sum_{k=1}^{n_o} |y_0, x_k\rangle\langle \mathbf{v}_k, y_0|.$$

M and N together handle non-injective functions.

For non-surjective functions, zero amplitudes are copied from the ancillary register $|x_0\rangle|y_i\rangle$ into the appropriate nonrange codomain components $|z_i\rangle|x_0\rangle$:

$$S \stackrel{\text{def}}{=} \sum_{i=1}^{m_{\text{nr}}} |z_i, x_0\rangle\langle x_0, y_i|.$$

The foregoing operators M, N, S handle the mapping of f (and disposal of the nullspace).

It remains to handle the other components of the input space in a unitary way. Components $|x_j\rangle|y_i\rangle$, for $i, j \neq 0$, map into their reverses:

$$R \stackrel{\text{def}}{=} \sum_{i=1}^{m} \sum_{j=1}^{n} |y_i, x_j\rangle\langle x_j, y_i|.$$

We have to be careful to handle all the basis vectors since $n_o < n$. We map vectors in the $|x_0\rangle$ subspace that were not used in the S map to components of the $|y_0\rangle$ subspace that are unfilled by N:

$$Q \stackrel{\text{def}}{=} \sum_{k=1}^{m_{\text{r}}} |y_0, x_{n_o+k}\rangle\langle x_0, y_{m_{\text{nr}}+k}|.$$

Note that $n_o + m_{\text{r}} = n = m_{\text{nr}} + m_{\text{r}}$.

The state $|x_0\rangle|y_0\rangle$ remains, and it maps to its reverse:

$$P \stackrel{\text{def}}{=} |y_0, x_0\rangle\langle x_0, y_0|.$$

M maps m_{r} basis vectors, N maps n_o basis vectors, S maps m_{nr}, R maps mn, Q maps m_{r}, and P maps one basis vector, which accounts for all of the $(n+1)(m+1)$ basis vectors:

$$m_{\text{r}}+n_o+m_{\text{nr}}+mn+m_{\text{r}}+1 = (m_{\text{r}}+n_o)+(m_{\text{nr}}+m_{\text{r}})+mn+1 = n+m+mn+1 = (m+1)(n+1).$$

The unitary operator to compute f is the sum of these linear maps:

$$U \stackrel{\text{def}}{=} M + N + S + R + Q + P.$$

Proposition 4 *Suppose $f : \Omega \to \Omega'$. Let $n = |\Omega|$ and $m = |\Omega'|$. Let $\mathcal{H}(\Omega^\circ)$ be an $n+1$ dimensional Hilbert space with basis $\{|x_0\rangle, \ldots, |x_n\rangle\}$, and let $\mathcal{H}(\Omega^*)$ be an $m+1$ dimensional space with basis $\{y_0, \ldots, y_m\}$. Then there a unitary operator $U \in \mathcal{L}[\mathcal{H}(\Omega^\circ) \otimes \mathcal{H}(\Omega^*), \mathcal{H}(\Omega^*) \otimes \mathcal{H}(\Omega^\circ)]$ so that for scalars s, r (with $|s|^2 + |r|^2 = 1$) and $x \in \Omega$:*

$$U[(s|x\rangle + r|x_0\rangle) \otimes |y_0\rangle] = ss'|f(x), x_0\rangle + r'|y_0, \gamma\rangle,$$

where $s = 1/\sqrt{n'}$, where $n' = |f^{-1}\{f(x)\}|$, and $|ss'|^2 + |r'|^2 = 1$.

Proof: By construction we know:

$$\begin{aligned}
U|x_j, y_0\rangle &= (M+N)|x_j, y_0\rangle \\
&= \left(\sum_{i=1}^{m_{\mathrm{r}}} |y_i', x_0\rangle\langle \mathbf{u}_i, y_0|\right)|x_j, y_0\rangle + \left(\sum_{k=1}^{n_o} |y_0, x_k\rangle\langle \mathbf{v}_k, y_0|\right)|x_j, y_0\rangle \\
&= |f(x_j), x_0\rangle\langle \mathbf{u}_i \mid x_j\rangle + |y_0\rangle \sum_{k=1}^{n_o} |x_k\rangle\langle \mathbf{v}_k \mid x_j\rangle \\
&= s_j|f(x_j), x_0\rangle + |y_0, \gamma\rangle,
\end{aligned}$$

where $|\gamma\rangle = (\sum_{k=1}^{n_o} |x_k\rangle\langle \mathbf{v}_k|)|x_j\rangle$ and $\||y_0, \gamma\rangle\| = \sqrt{n_j - 1}/\sqrt{n_j}$. Furthermore, by construction,

$$U|x_0, y_0\rangle = P|x_0, y_0\rangle = |y_0, x_0\rangle.$$

Therefore, in the general case where the input register is $s|x_j\rangle + r|x_0\rangle$ (with $|s|^2 + |r|^2 = 1$) we have:

$$\begin{aligned}
U[(s|x_j\rangle + r|x_0\rangle) \otimes |y_0\rangle] &= sU|x_j, y_0\rangle + rU|x_0, y_0\rangle \\
&= s(s_j|f(x_j), x_0\rangle + |y_0, \gamma\rangle) + r|y_0, x_0\rangle \\
&= ss_j|f(x_j), x_0\rangle + |y_0\rangle(s|\gamma\rangle + r|x_0\rangle).
\end{aligned}$$

□

Therefore, the result that we want is in the first ($\mathcal{H}(\Omega^*)$) quantum register, but its $|y_0\rangle$ component is garbage and should be ignored in subsequent computations. Furthermore, the amplitude of desired result will decrease through successive computation stages through attenuation by successive $1/\sqrt{n_j}$ factors.

As discussed previously, quantum states with $|x_j\rangle$ components with positive amplitudes represent sets of the corresponding x_j ($j \neq 0$). Applying U to such a state yields a quantum state with positive amplitudes for the corresponding $|f(x_j)\rangle$, which represents the output set of $f(x_j)$.

E Alternative Approach

E.1 Representation

To further explore quantum computation by topographic maps, in this section we present an alternative representation of the maps and a circuit-based implementation of arbitrary functions on a finite domain. Therefore, suppose $f : \Omega \to \Omega'$, where the domain is $\Omega = \{x_1, \ldots, x_n\}$ and the codomain is $\Omega' = \{y_1, \ldots, y_m\}$. Typically the values would evenly spaced, for example, $\Omega = \{0, \Delta x, 2\Delta x, \ldots, (n-1)\Delta x\}$, but this is not required, and other spacings, such as logarithmic, might be useful. (Logarithmic maps are found in some sensory cortical regions.)

Here, a different quantum representation will be used for the computational maps from that used in the preceding sections. In this representation, each domain value x_j or codomain value y_i is assigned a separate qubit, whose state, for example, $|\psi_j\rangle = p'_j|0\rangle + p_j|1\rangle$, where $|p'_j|^2 + |p_j|^2 = 1$, represents the activity level of x_j by the amplitude p_j. This sort of one-out-of-n representation might seem unrealistically inefficient, but (1) we are assuming a scalable qubit implementation, which permits arrays of many thousands of qubits, and (2) neural computations typically require only low precision (maybe one digit in the brain), and so a quantity can be represented by a few tens of qubits. In other words, our m and n will typically be small ($m, n < 100$).

These quantum computational maps can also be viewed as representations of subsets of the domain; for $S \subseteq \Omega$:

$$|S\rangle = \sum_{x_j \in S} |1\rangle_j + \sum_{x_j \notin S} |0\rangle_j.$$

That is, the x_j qubit is in state $|1\rangle$ if x_j is in S and is in state $|0\rangle$ if it is not. Therefore we use $|\{x_j\}\rangle$ for the map representing just the number x_j. The set of representations of all possible subsets of Ω is then an ON basis for the 2^n-dimensional Hilbert space of these qubits. The basis can be written:

$$\{|k\rangle \mid k \in \mathbf{2}^n\} = \{|S\rangle \mid S \subseteq \mathbf{2}^\Omega\},$$

where on the left $\mathbf{2}^n$ is the set of n-bit binary strings, and on the right $\mathbf{2}^\Omega$ is the powerset of Ω. Therefore *multiple* sets can be processed in quantum superposition.

Moreover, because the sets are represented by computational basis vectors, they can be copied without violating the no-cloning theorem.

By using amplitudes other than 0 and 1, we can represent fuzzy sets. Suppose S is a fuzzy set with membership function $\mu : S \to [0, 1]$, and let $m_j = \mu(x_j)$. Then S is represented by the quantum computational map

$$|S\rangle = \sum_{j=1}^{n} m_j |1\rangle + \sqrt{1 - m_j^2}\, |0\rangle.$$

Fuzzy sets cannot, in general, be copied (nor can arbitrary superpositions of crisp sets).

With this representation, the transformations between computational maps will be implemented by the circuit model, and so one might ask whether it would be simpler to implement ordinary binary digital quantum computation. The answer is that computation on topographic maps can be implemented by a few relatively simple operations (described in the next section), so that it buys a simpler quantum implementation at the cost of greater representational expense (number of qubits). We expect topographic quantum computation to be more simply implemented than a full-scale digital quantum arithmetic-logic unit.

E.2 Unary Functions

An example will illustrate how to implement an arbitrary finite function $f : \Omega \to \Omega'$ by computational maps. For any y_i not in the range of f, we set its state $|\phi_i\rangle = |0\rangle$ supplied as an ancilla. If y_i is in the range of f, then it might be the image of a single domain element, x_j, that is, $y_i = f(x_j)$, in which case we implement directly $|\phi_i\rangle = |\psi_j\rangle$. If there are two domain values mapping into y_i, say $f(x_j) = y_i = f(x_k)$, then we make $|\phi_i\rangle$ the logical OR of $|\psi_j\rangle$ and $|\psi_k\rangle$:

$$\mathrm{X}|\psi_j\rangle \otimes \mathrm{X}|\psi_k\rangle \otimes |\phi_i\rangle = \mathrm{OR}_2(|\psi_j\rangle \otimes |\psi_k\rangle \otimes |1\rangle),$$

where the OR_2 gate is defined: $\mathrm{OR}_2 \stackrel{\mathrm{def}}{=} \mathrm{CCNOT}(\mathrm{X} \otimes \mathrm{X} \otimes \mathrm{I})$. The $|1\rangle$ is an ancilla. OR_2 transfers probability amplitudes as follows:

$$\begin{aligned}
\mathrm{OR}_2|\psi_j\rangle|\psi_k\rangle|1\rangle &= \mathrm{OR}_2(p'|0\rangle + p|1\rangle)(q'|0\rangle + q|1\rangle)|1\rangle \\
&= p'q'\mathrm{OR}_2|001\rangle + p'q\mathrm{OR}_2|011\rangle + pq'\mathrm{OR}_2|101\rangle + pq\mathrm{OR}_2|111\rangle \\
&= p'q'|110\rangle + p'q|101\rangle + pq'|011\rangle + pq|001\rangle.
\end{aligned}$$

Therefore, the third output qubit is the OR of the first two input qubits with the probability amplitudes shown. The first two output qubits are negated copies of the

inputs, which are considered garbage but must be retained, for they are entangled with the third output.

If more than two domain values map into a single codomain value, then we use the multiple argument OR_n, which can be defined in terms of OR_2. For completeness, we define $\mathrm{OR}_1 = \mathrm{I}$.

With the preceding motivation, we can give the construction for computing an arbitrary finite function by quantum computational maps:

Proposition 5 *Suppose $f : \Omega \to \Omega'$, where $\Omega = \{x_1, \ldots, x_n\}$ and $\Omega' = \{y_1, \ldots, y_n\}$. Let $m_{\mathrm{nr}} = n - |\operatorname{Im} f|$ be the number of codomain elements that are not in the range of f. Let n_{b} be the number of injective domain elements, and let $n_{\mathrm{n}} = n - n_{\mathrm{b}}$ be the number of non-injective domain elements. Let $m_{\mathrm{n}} = |\operatorname{Im} f| - n_{\mathrm{b}}$ be the number of non-injective range elements (i.e., those that are the image of two or more domain elements). Then there is an $2n_{\mathrm{n}} + n_{\mathrm{b}} + m_{\mathrm{nr}} - m_{\mathrm{n}}$ dimensional unitary operator U_f that computes f by computational maps:*

$$|\{y_i\}\rangle|\gamma\rangle = U_f|\{x_j\}\rangle|0\rangle^{\otimes m_{\mathrm{nr}}}|1\rangle^{\otimes(n_{\mathrm{n}}-m_{\mathrm{n}})},$$

where $y_i = f(x_j)$ and $|\gamma\rangle$ is $2(n_{\mathrm{n}} - m_{\mathrm{n}})$ qubits of garbage.

Proof: The inputs are the n elements of the input map, m_{nr} constant $|0\rangle$ ancillae (for the non-range elements), and $n_{\mathrm{n}} - m_{\mathrm{n}}$ constant $|1\rangle$ ancillae for the OR_2 gates that map multiple domain elements to the same range element. That is, there are $n + m_{\mathrm{nr}} + n_{\mathrm{n}} - m_{\mathrm{n}} = (n_{\mathrm{b}} + n_{\mathrm{n}}) + m_{\mathrm{nr}} + n_{\mathrm{n}} - m_{\mathrm{n}} = 2n_{\mathrm{n}} + n_{\mathrm{b}} + m_{\mathrm{nr}} - m_{\mathrm{n}}$ input qubits. The m_{nr} constant $|0\rangle$s are passed directly to the output qubits for non-range elements. The n_{b} qubits for injective inputs are passed to the same number of output qubits, permuted as required. The outputs of the ORs project onto the m_{n} qubits that represent range values with more than one pre-image. Each OR_2 also generates two garbage qubits, for a total of $2(n_{\mathrm{n}} - m_{\mathrm{n}})$. Therefore the total number of output qubits is $m_{\mathrm{nr}} + n_{\mathrm{b}} + m_{\mathrm{n}} + 2(n_{\mathrm{n}} - m_{\mathrm{n}}) = 2n_{\mathrm{n}} + n_{\mathrm{b}} + m_{\mathrm{nr}} - m_{\mathrm{n}}$.

Next we define U_f explicitly as the tensor product of three operators:

$$U_f \stackrel{\mathrm{def}}{=} U_{\mathrm{nr}} \otimes U_{\mathrm{b}} \otimes U_{\mathrm{n}}.$$

We will use the notation $|1\rangle_{x_j}$ to represent a $|1\rangle$ state in the qubit corresponding to x_j, and $|0\rangle_{x_j}$ to represent a $|0\rangle$ state in the qubit corresponding to x_j.

The U_{nr} operator is an identity operation producing the codomain elements that are not in the range of f. Therefore, let $\{c_1, \ldots, c_{m_{\mathrm{nr}}}\} = \Omega' - \operatorname{Im} f$ be this set, and

let $|z_i\rangle$ be ancillae qubits to provide constant $|0\rangle$s (numbered $n+1,\ldots,n+m_{\rm nr}$ after the input qubits), Then $U_{\rm nr}:\mathcal{H}^{m_{\rm nr}}\to\mathcal{H}^{m_{\rm nr}}$ is defined:

$$U_{\rm nr}\stackrel{\rm def}{=}\sum_{i=1}^{m_{\rm nr}}|0\rangle_{c_i}\langle 0|_{z_i}+|1\rangle_{c_i}\langle 1|_{z_i}.$$

This can be abbreviated by the following bracket notation:

$$U_{\rm nr}\stackrel{\rm def}{=}[c_1,\ldots,c_{m_{\rm nr}}]\leftarrow[z_1,\ldots,z_{m_{\rm nr}}].$$

It is just a permutation of the qubits.

The $U_{\rm b}$ operator handles the domain elements that are mapped injectively to their images. Therefore, let $\{v_1,\ldots,v_{n_{\rm b}}\}\subseteq{\rm Im}\,f$ be the injective domain elements, and let $u_i=f(v_i)$ be the corresponding range elements. Then $U_{\rm b}:\mathcal{H}^{n_{\rm b}}\to\mathcal{H}^{n_{\rm b}}$ is a permutation of this subset of the map elements:

$$U_{\rm b}\stackrel{\rm def}{=}\sum_{i=1}^{n_{\rm b}}|0\rangle_{u_i}\langle 0|_{v_i}+|1\rangle_{u_i}\langle 1|_{v_i}.$$

This can be abbreviated:

$$U_{\rm b}\stackrel{\rm def}{=}[u_1,\ldots,u_{n_{\rm b}}]\leftarrow[v_1,\ldots,v_{n_{\rm b}}].$$

For $U_{\rm n}$ we must OR together the domain elements corresponding to each range element with more than one preimage. Therefore we define:

$$U_{\rm n}\stackrel{\rm def}{=}\bigotimes_{i=1}^{m_{\rm n}}V(y_i,f^{-1}\{y_i\}),$$

where these y_i have more than one preimage element, for example, $f^{-1}\{y_i\}=\{x_1,\ldots,x_l\}$, where $l=|f^{-1}\{y_i\}|\geq 2$. The output state $|\psi_i\rangle$ for input states $|\xi_j\rangle$ $(j=1,\ldots l)$ is then

$$|\psi_i\rangle|\gamma\rangle|\xi_1\rangle\cdots|\xi_l\rangle={\rm OR}_l|\xi_1\rangle\cdots|\xi_l\rangle|1\rangle^{\otimes(l-1)},$$

where $|\gamma\rangle$ is $l-2$ dimensional garbage. This is accomplished by the operator $V(y_i,\{x_1,\ldots,x_l\})\in\mathcal{L}(\mathcal{H}^{2l-1},\mathcal{H}^{2l-1})$:

$$V(y_i,\{x_1,\ldots,x_l\})\stackrel{\rm def}{=}[y_i,\gamma_i,x'_1,\ldots,x'_l]\leftarrow{\rm OR}_l[x_1,\ldots,x_l,o_1,\ldots,o_{l-1}],$$

where $o_1,\ldots,o_{l-1}$ are the qubits that provide ancillary $|1\rangle$s for the ORs, and $x'_1,\ldots,x'_l$ receive the negated inputs ($|x'_j\rangle={\rm X}|x_j\rangle$). The bracket notation identifies the qubits that are the inputs and outputs of ${\rm OR}_l$.

□

There are more efficient ways to compute U_f, but the above construction is easier to understand.

This basic approach can be used to approximate a variety of unary functions useful in neural networks, such as sigmoid functions, including non-injective, non-surjective squashing functions. However, neural networks also require non-unary functions such as addition and multiplication, to which we now turn.

E.3 Binary Functions

In sensory cortical areas there are many topographic maps that represent two or more dimensions of a stimulus (e.g., retinal position and edge orientation); localized activity in these maps represent conjunctions of values on these dimensions. Similarly, quantum computational maps can represent conjunctions of values as inputs or outputs of functions.

Suppose we want to compute a function $f : \Omega \times \Omega \to \Omega'$, where $\Omega = \{x_1, \ldots, x_n\}$ and $\Omega' = \{y_1, \ldots, y_n\}$. We will represent the input to the function by a two-dimensional array of qubits for each (x_j, x_k) pair. This will require n^2 qubits, but we are assuming that n is small because low precision is adequate for neural networks. Therefore we expect the 2D map to comprise typically several thousand qubits. The qubits representing the (x_j, x_k) pairs are then mapped to the qubits representing the outputs $f(x_j, x_k)$ by the method described in Prop. 5.

The n^2 conjunctions are computed by n^2 CCNOT gates, each of which requires a $|0\rangle$ ancilla and generates two qubits (representing the inputs) in addition to the conjunction. However, these extra qubits are passed along the rows and columns, and so there are only $2n$ total garbage qubits. In summary, there are $2n + n^2$ input qubits (including n^2 ancillae) and $n^2 + 2n$ output qubits (including $2n$ garbage). That is, if $|\phi_j\rangle$ is the state of element j of one input map, and $|\psi_k\rangle$ is the state of element k of the other input map, then the state $|\chi_{jk}\rangle$ of element (j, k) of the two-dimensional map is given by $|\phi_j\rangle|\psi_k\rangle|\chi_{jk}\rangle = \mathrm{CCNOT}|\phi_j\rangle|\psi_k\rangle|0\rangle$. If $|\phi_j\rangle = p'|0\rangle + p|1\rangle$ and $|\psi_k\rangle = q'|0\rangle + q|1\rangle$, then

$$|\phi_j\rangle|\psi_k\rangle|\chi_{jk}\rangle = p'q'|000\rangle + pq'|100\rangle + p'q|010\rangle + pq|111\rangle.$$

The qubits are entangled, but the conjunction has probability-like amplitudes.

Based on the foregoing, we define a unitary operator U_{OP} on a n^2+2n dimensional Hilbert space that does what amounts to an outer product on two one-dimensional maps to compute a two-dimensional map: $|\phi\rangle|\psi\rangle|\chi\rangle = U_{\mathrm{OP}}|\phi\rangle|\psi\rangle|0\rangle^{\otimes n^2}$, where $|\phi\rangle$ and $|\psi\rangle$ are n-dimensional, and $|\chi\rangle$ is n^2-dimensional.

To illustrate the use of computational maps to implement binary operations, we will use a simple, useful function, addition. We want to define sum : $\Omega \times \Omega \to \Omega'$ so that $\mathrm{sum}(x, y) = x + y$, but we have a problem, since the maximum value of $x + y$ is greater than the maximums of x and y. Since the range of numbers represented by our maps is quite limited, this is a more serious problem than overflow in binary addition. One solution is to make the codomain map large enough; for example, if $\Omega = \{0, \Delta x, \ldots, (n-1)\Delta x\}$, then let $\Omega' = \{0, \Delta x, \ldots, 2(n-1)\Delta x\}$. Generally, however, it is more convenient to have the codomain maps be the same as the domain maps, since this facilitates composing functions. Therefore, another solution is to scale either the inputs or the output so that we compute, for example, $\mathrm{sum}(x, y) = (x + y)/2$; this is often useful if we know that we are going to scale the quantities anyway. A third option is to compose the operator with squashing function, so that we compute, for example, $\mathrm{sum}(x, y) = \min(x + y, x_n)$, where $x_n = \max \Omega$. This is the solution that we will use for illustration.

If $\Omega = \{0, \Delta x, \ldots, (n-1)\Delta x\}$, then the (j, k) element of the two-dimensional qubit map will represent the pair of inputs $((j-1)\Delta x, (k-1)\Delta x)$. This will be mapped to the sum $(j + k - 2)\Delta x$ if $j + k - 2 < n - 1$, and to the maximum value $(n-1)\Delta x$ otherwise. Therefore the constant $j + k$ anti-diagonals above the $j + k - 1 = n$ anti-diagonal each map to one value, $(j + k - 2)\Delta x$, and all the elements below the $j + k - 1 = n$ anti-diagonal map to $(n-1)\Delta x$. In terms of the variables in Prop. 5, the number of non-range codomain elements is $m_{\mathrm{nr}} = 0$, the number of injective domain elements is $n_{\mathrm{b}} = 1$, the number of non-injective inputs is $n_{\mathrm{n}} = n^2 - 1$, and the number of non-injective outputs is $m_{\mathrm{n}} = n - 1$. Therefore, there is a unitary operator U_{sum} will map the pairs into the appropriate (squashed) outputs:

$$|\{\mathrm{sum}(x, y)\}\rangle|\gamma\rangle = U_{\mathrm{sum}}|\{x\} \times \{y\}\rangle|1\rangle^{\otimes n(n-1)},$$

where $|\{x\} \times \{y\}\rangle$ represents the two-dimensional outer product map of $|\{x\}\rangle$ and $|\{y\}\rangle$: $|x\rangle|y\rangle|\{x\} \times \{y\}\rangle = U_{\mathrm{OP}}|\{x\}\rangle|\{y\}\rangle|0\rangle^{\otimes n^2}$. The same approach can be used for other binary operations, such as multiplication.

F Additional Considerations

Ordinary binary representations can be translated to computational maps (alternative representation) by a unitary demultiplexer U_{demux} that operates on an m-qubit binary number $|k\rangle$ and directs a $|1\rangle$ qubit to the kth of $n = 2^m$ output qubits (the remainder receiving $|0\rangle$). Let $|\{k\}\rangle$ be the resulting computational map. Then:

$$|k\rangle|\{k\}\rangle = U_{\mathrm{demux}}|k\rangle|1\rangle|0\rangle^{\otimes(n-1)}.$$

U_{demux} operates on an $m + n = m + 2^m$ dimensional Hilbert space. A demultiplexer can be implemented with CSWAP (Fredkin) gates [1].

The opposite translation, from a computational map to a binary representation, is more complicated. First, we must decide what we want it to do, for in general a computational map represents multiple values with different amplitudes, $|\psi\rangle = \bigotimes_{j=1}^{n} p'_j|0\rangle + p_j|1\rangle$, where $|p'_j|^2 + |p_j|^2 = 1$. Which x_j should it produce? The one with the maximum amplitude? (And what if several have the same maximum amplitude?) An x_j chosen probabilistically based on $|p_j|^2$? The binary representation of a weighted average $n^{-1}\sum_{j=1}^{n} p_j x_j$? The answer is not apparent, so we leave the question open.[3]

G Applications to Quantum Machine Learning

Given this general ability to compute non-unitary and even nonlinear functions, it is possible to do the operations useful for machine learning such as inner products and sigmoid nonlinearities. For example, an inner product of N-element vectors requires N multiplications and $\lceil \log_2 N \rceil$ additions. If $|\Omega| = n$, then each multiplication and addition will require approximately n^2 qubits, for a total of $(N + \lceil \log_2 N \rceil)n^2$.

For one layer of a neural network, say N neurons projecting through an $M \times N$ weight matrix into M neurons, we must do M inner products with the input. Since crisp sets are represented by computational maps that are basis vectors in the computational basis, they can be copied. Therefore, the N-dimensional input vector can be copied $M-1$ times to do the M inner products. (This requires $M-1$ CNOT gates and $(M-1)n$ ancillary $|0\rangle$ qubits. Overall, one layer requires $M(N + \lceil \log_2 N \rceil)n^2$ qubits for the computation (not including ancillary qubits).

H Conclusions

Topographic (computational) maps are widely used in the brain to implement simultaneous nonlinear vector transformations on multiple inputs. In this chapter we have explored two approaches to quantum topographic computing with a focus on brain-inspired machine learning applications. The first assigns locations in the map to state vectors in a continuous or discrete basis for a quantum Hilbert space. Arbitrary functions can be implemented on such maps, which can be interpreted as

[3] It is easy however to produce the binary representation of either the maximum or minimum number with unit amplitude ($p_j = 1, p'_j = 0$) in a map.

representing crisp sets of inputs, but there is an unavoidable data-dependent attenuation of the result (compared to a "garbage" state) that is not easily avoidable. The second approach assigns a separate qubit to each location in the map, and uses the relative amplitude of the $|1\rangle$ and $|0\rangle$ components to represent the presence of values in the (crisp or fuzzy) set represented by the map. Arbitrary functions on these maps are implemented by well-known quantum logic gates.

References

[1] E. F. Fredkin and T. Toffoli. Conservative logic. *Int. J. Theo. Phys.*, 21(3/4):219–253, 1982.

[2] Peter Høyer. Arbitrary phases in quantum amplitude amplification. *Phys. Rev. A*, 62:052304, Oct 2000.

[3] Bruce J. MacLennan. Field computation in motor control. In Pietro G. Morasso and Vittorio Sanguineti, editors, *Self-Organization, Computational Maps and Motor Control*, pages 37–73. Elsevier, 1997. Also available from `web.eecs.utk.edu/~mclennan`.

[4] Bruce J. MacLennan. Field computation in natural and artificial intelligence. *Information Sciences*, 119:73–89, 1999. Also available from `web.eecs.utk.edu/~mclennan`.

[5] Bruce J. MacLennan. Field computation in natural and artificial intelligence. In R. A. Meyers et al., editor, *Encyclopedia of Complexity and System Science*, chapter 6, entry 199, pages 3334–3360. Springer, 2009.

[6] Bruce J. MacLennan. The promise of analog computation. *International Journal of General Systems*, 43(7):682–696, 2014.

[7] Bruce J. MacLennan. Field computation: A framework for quantum-inspired computing. In Siddhartha Bhattacharyya, Ujjwal Maulik, and Paramartha Dutta, editors, *Quantum Inspired Computational Intelligence: Research and Applications*, chapter 3, pages 85–110. Morgan Kaufmann / Elsevier, Cambridge, MA, 2017.

[8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge, 10th anniversary edition edition, 2010.