Topographic Representation for Quantum Machine Learning

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

May 16, 2019

A Introduction

This paper 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, with quantum parallelism. There are of course many technical challenges in quantum computing, but the principal theoretical challenge in quantum machine learning is the fact that quantum processes are linear whereas most approaches to machine learning depend crucially on nonlinear operations. Artificial neural networks, in particular, require a nonlinear activation function, such as a logistic sigmoid function, for nontrivial machine learning. 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.

Despite the complications of quantum machine learning, it presents a tantalizing approach to implementing large-scale machine learning in a post-Moores law technological era. However, there are many approaches to machine learning and several approaches to quantum computation (e.g., circuit model, adiabatic), and it is not obvious which combinations are most likely of success. Schuld, Sinayskiy, and Petruccione [17, 18] provide useful recent reviews of some approaches to quantum neural networks and quantum machine learning more generally. They conclude that none of the proposals for quantum neural networks succeed in combining the advantages

of artificial neural networks and quantum computing. They argue that open quantum systems, which involve dissipative interactions with the environment, are the most promising approach. Moreover, few of the proposals include an actual theory of quantum machine learning.

This paper explores an approach to the quantum implementation of machine learning involving nonlinear functions operating on information represented topographically, as is common in neural cortex. However, there are many approaches to neurocomputing, that is, to brain-inspired computing, some of which may be more amenable to quantum implementation than others, and so we must say a few words about the alternatives. For the brain may be modeled at many different levels, and models at each of these levels may provide a basis for machine learning. For example, in recent years spiking neural network models have become popular once again, largely due to their power advantages when implemented in hardware [3, 12]. Such models mimic the "all or none" action potential generation by biological neurons without addressing the detailed dynamics of the action potential. On the other hand, most contemporary applications of artificial neural networks, including those used in deep learning, use a higher level, rate-based model. That is, the real values passed between the units (neuron analogs) represent the rate of neural spiking rather than individual spikes. It has been argued that this is the appropriate level for modeling neural information processing, since there are many stochastic effects on the generation and reception of action potentials, and because the fundamental units of information processing are microcolumns comprising about 100 neurons [14, ch. 2]. Therefore it is most fruitful to view neural computation as a species of massively parallel analog computation. Since quantum computation makes essential use of complex-valued probability amplitudes, it is also fruitful to treat it as a species of analog computation, and so analog information representation provides one point of contact between quantum computation and artificial neural networks [9].

B Topographic Representation in the Brain

Another respect in which information processing in the brain differs from most artificial neural network models is that biological neural networks are spatially organized, with connectivity dependent on spatial organization. Although artificial neural networks are typically organized in layers, there is generally no spatial relationship among the neurons in each layer;¹ the exceptions are convolutional neural networks,

¹ They are numerically indexed, of course, but interchangeable in terms of their pattern of connections before learning.

which were in fact inspired by the organization of sensory cortex.

One of the most common spatial 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; that is, spatial relationships among the neurons are correlated with topological relationships in the abstract space. These maps are especially common in motor areas [11] and sensory areas [14, ch. 6]. For example, tonotopic maps have the 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. There are many other examples throughout the brain, and this is perhaps the single most common information representation used by the brain.

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.

Topographic maps provide another point of contact between artificial neural networks and quantum computation, because the computational maps in the brain are large and dense enough that they can be usefully treated mathematically as fields, that is, as continuous distributions of continuous quantity [6]. Such representations are suggestive of quantum mechanical wave functions, which are also continuous distributions of continuous quantity (the complex probability amplitude). In both cases these fields are treated mathematically as continuous functions on a continuous domain, and Hilbert spaces provide the mathematical framework for describing them [9]. In this paper we exploit this analogy to implement brain-inspired approaches to quantum machine learning.

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 represents the result. Thus activity representing \mathbf{x} in the input map will cause corresponding activity representing $f(\mathbf{x})$ 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 [1, 5, 6, 7, 8], which suggests their usefulness in quantum computation [9]. 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. (although this can be mitigated by means of *coarse coding*, by which precise values are represented by a population of broadly tuned neurons with overlapping receptive fields [15, pp. 91–96][16]). Therefore, use of topographic representations will require a reasonably scalable quantum computing technology. In this paper we explore topographic approaches to quantum computation with a focus on machine learning.

C Topographic Basis Maps

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. The states $|\mathbf{r}\rangle$ form a discrete basis or continuous pseudo-basis for the input and output quantum states. In the continuous case, the input value \mathbf{x} is represented by a state $|\mathbf{r}\rangle$, which is a Dirac unit impulse at $\mathbf{r} = \mu(\mathbf{x})$: that is, $|\mathbf{r}\rangle = \delta_{\mathbf{r}}$ where $\delta_{\mathbf{r}}(\mathbf{s}) = \delta(\mathbf{s} - \mathbf{r})$. Similarly an output value \mathbf{y} is represented by the continuous basis 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. We call such a representation (whether discrete or continuous) a topographic basis map.

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

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

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: $T|\mu(\mathbf{x})\rangle = |\mu'[f(\mathbf{x})]\rangle$ 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.²

² It is not the same as the graph kernels used in machine learning applied to graph theory.

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: $T|\psi\rangle = a|\mu'[f(\mathbf{x})]\rangle + b|\mu'[f(\mathbf{x}')]\rangle$. Therefore, continuous topographic basis maps permit nonlinear functions to be computed in linear superposition (quantum parallelism). This is a step toward quantum computation, but T might not be unitary, and we have more work to do.

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, for example, by a discrete series of sinusoids or complex exponentials of the appropriate dimension. Second, the infinite discrete basis can be approximated by a finite discrete basis, for example, by band-limited sinusoids or complex exponentials. Such an approximation is especially appropriate for neural network machine learning, which requires only low-precision calculation.

C.1 Bijection

We proceed to show several examples of nonlinear computations performed via topographic basis 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 one-dimensional continuous domains, $\Omega = [x_l, x_u]$, and discrete domains, $\Omega = \{x_1, \ldots, x_n\}$. 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.) In both the continuous and discrete 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 $n = 2^N$. This might seem to require a large number of qubits, but even in the absence of coarse coding, seven qubits would be sufficient to represent values with 1% precision, which is adequate for many machine learning applications.

We begin with a bijective scalar function $f:[-1,1] \to [-1,1]$. The hyperbolic tangent (appropriately restricted³), which is a useful sigmoid function for neural computation, is an example of such a function. The graph kernel to compute the function topographically is $T = \int_{\Omega} dx |f(x)\rangle\langle x|$. Since f is bijective, the adjoint of

³ For example, $f(x) = [\tanh(x)/\tanh(1)]|_{[-1,1]}$.

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.

Proof: Substituting the above equations, observe:

$$T^{\dagger}T = \left(\int_{\Omega'} |f^{-1}(y)\rangle\langle y| dy \right) \left(\int_{\Omega} |f(x)\rangle\langle x| dx \right)$$

$$= \int_{\Omega} \int_{\Omega'} |f^{-1}(y)\rangle\langle y| f(x)\rangle\langle x| dy dx$$

$$= \int_{\Omega} \int_{\Omega'} |f^{-1}(y)\rangle\langle y| f(x)\rangle dy \langle x| dx$$

$$= \int_{\Omega} |f^{-1}(f(x))\rangle\langle x| dx$$

$$= \int_{\Omega} |x\rangle\langle x| dx$$

$$= I_{\Omega}.$$

The fourth line follows from the "sifting" property of the Dirac delta. Likewise,

$$TT^{\dagger} = \left(\int_{\Omega} |f(x)\rangle\langle x| dx \right) \left(\int_{\Omega'} |f^{-1}(y)\rangle\langle y| dy \right)$$
$$= \int_{\Omega'} \int_{\Omega} |f(x)\rangle\langle x| f^{-1}(y)\rangle\langle y| dx dy$$
$$= \int_{\Omega'} |f(f^{-1}(y))\rangle\langle y| dy = I'_{\Omega}.$$

Therefore T is unitary.

In the discrete basis case, let $y_i = f(x_i)$; the unit vectors $|x_i\rangle$ are a basis for \mathbb{C}^n , and the unit vectors $|y_i\rangle$ are also a (possibly different) basis for \mathbb{C}^n . The graph 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 also easily proved to be unitary. More directly, T is a permutation matrix on the basis elements and therefore orthogonal.

If the input is a weighted superposition of values, $|\psi\rangle = \int_{\Omega} dx \ p(x)|x\rangle$, 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. Moreover, since the graph kernel is unitary, the adjoint is the inverse: if $|\phi\rangle = \int_{\Omega'} \mathrm{d}y \; q(y)|y\rangle$, then $T^{\dagger}|\phi\rangle = \int_{\Omega'} \mathrm{d}y \; q(y)|f^{-1}(y)\rangle$. That is, applying the adjoint to a superposition of outputs will compute a corresponding superposition of inputs.

C.2 Non-surjective injections

As a further step towards the quantum computation of arbitrary functions by means of computational maps, we consider a relatively simple case: non-surjective injections (that is, one-to-one non-onto functions). We restrict our attention to finite domains and codomains. 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}_{C} , and so the complete input space will be in $\mathcal{H}(\Omega) \otimes \mathcal{H}_{C}$. Our implementation will also generate "garbage" output in a space \mathcal{H}_{G} , and so the complete output space will be in $\mathcal{H}(\Omega') \otimes \mathcal{H}_{G}$. The input and output dimensions must be equal, and the simplest way to accomplish this is to make \mathcal{H}_{C} m-dimensional and \mathcal{H}_{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}_{C} and let $\{|v_1\rangle, \ldots, |v_n\rangle\}$ be a basis for \mathcal{H}_{G} . (We could in fact use the $\mathcal{H}(\Omega')$ basis for \mathcal{H}_{C} and the $\mathcal{H}(\Omega)$ basis for \mathcal{H}_{G} , but here we develop a more general result.)

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 can be expressed as the sum of several operators: $U \stackrel{\text{def}}{=} T + S + R + Q$. 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 $T|x_j\rangle|w_1\rangle = |f(x_j)\rangle|v_1\rangle$, and T is a bijection of the n-dimensional subspace $\mathcal{H}(\Omega)\otimes\mathcal{H}(|w_1\rangle)$. However T is not unitary since it is not a surjection. See Figure 1.

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

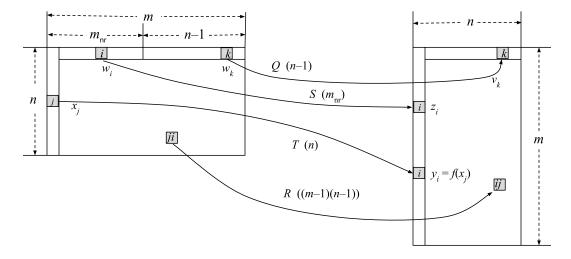


Figure 1: Permutation of basis vectors to implement non-surjective injections. After each component of the kernel, the number of basis vectors that it maps is indicated in parentheses; for example, R maps (m-1)(n-1) basis vectors. T maps function inputs to outputs, S maps zero amplitudes to non-range codomain elements, and R and Q bijectively map the remaining basis elements.

codomain elements $\{z_1, \ldots, z_{m_{\text{nr}}}\}\subset \Omega'$. Then the S component is defined:

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

Therefore, S is a bijection of an $m_{\rm nr}$ -dimensional subspace and each non-range element $|z_i\rangle|v_1\rangle$ has a unique preimage $|x_1\rangle|w_{i+1}\rangle$

Note that T transforms n basis vectors (those for which the second register is $|w_1\rangle$), and S transforms $m_{\rm nr}$ basis vectors (those for which the first register is $|x_1\rangle$ and the second register is $|w_2\rangle, \ldots, |w_{m_{\rm nr}+1}\rangle$, for a total of $n+m_{\rm nr}=m$ basis elements, but the input space has a total of mn basis elements, and the remainder must be bijectively mapped.

Therefore, to complete the unitary operator we add the following additional components:

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

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

R maps (m-1)(n-1) basis elements: those for $i \neq 1$ and $j \neq 1$, that is, those with neither $|x_1\rangle$ in the first register nor $|w_1\rangle$ in the second. Q maps the remaining n-1 basis elements: those which have $|x_1\rangle$ in the first register and $|w_{m_{nr}+1}\rangle$ to $|w_{m_{nr}+n-1}\rangle$ in the second (recall that $m=m_{nr}+n$).

Notice that U maps every input basis vector into exactly one output basis vector and vice versa (see Fig. 1). Summing the basis vector dyads for T, S, R, Q gives:

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

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}_{C} and \mathcal{H}_{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}_{C} and $|v\rangle$ be a fixed basis vector of \mathcal{H}_{G} . Then there is a unitary operator $U \in \mathcal{L}[\mathcal{H}(\Omega) \otimes \mathcal{H}_{C}, \mathcal{H}(\Omega') \otimes \mathcal{H}_{G}]$ 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 normalized superposition of the corresponding function results: $U|\psi\rangle|\omega\rangle = \sum_k p_k |f(x_k)\rangle|v\rangle$.

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 LCM(m, n). The principle is the same: a permutation of the basis vectors.

C.3 Non-injective surjections

Next we consider functions $f: \Omega \to \Omega'$ 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 $T(|x\rangle - |x'\rangle) =$ **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, 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 $|\nu\rangle \in \mathcal{N}(T)$. The $|\mu\rangle$ component is sufficient to determine the output, so there is a bijection $|\mu\rangle \leftrightarrow T|\psi\rangle$, and the $|\nu\rangle$ component preserves the information to differentiate the inputs that map to this output.

To explain how this separation 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), 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 $|\mathbf{u}_1\rangle, \dots |\mathbf{u}_m\rangle$ to be an orthonormal (ON) basis for $\mathcal{N}(T)^{\perp}$ (the row space of T), where m is the rank of T; and we will define $|\mathbf{v}_1\rangle, \dots, |\mathbf{v}_{n_o}\rangle$ 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.

An example will make this clearer. Suppose $\Omega = \{k\Delta x \mid -N < k < N\}$ and $\Omega' = \{k\Delta x \mid 0 \le k < N\}$. Let abs: $\Omega \to \Omega'$ be the absolute value function (a noninjective surjection between these sets). A basis for the nonnull space $\mathcal{N}(T)^{\perp}$ comprises $|\mathbf{u}_0\rangle = |0\rangle$ and the vectors $|\mathbf{u}_k\rangle = (|-k\Delta x\rangle + |k\Delta x\rangle)/\sqrt{2}$ (for k = 1, N - 1). (Note that $|-k\Delta x\rangle$ and $|k\Delta x\rangle$ are orthogonal vectors for $k \ne 0$.) These N basis vectors are in a one-to-one relation with the codomain elements $|k\Delta x\rangle$ (for $k = 0, \ldots, N - 1$). The nullity is $n_0 = (2N - 1) - N = N - 1$ and the basis vectors of the null space are:

$$|\mathbf{v}_k\rangle = (|-k\Delta x\rangle - |k\Delta x\rangle)/\sqrt{2}$$
 (for $k = 1, ..., N-1$).

Projection onto this space keeps the information necessary to distinguish the specific preimage that maps to a given output. In this case, it remembers the sign of the input: note that $\langle \mathbf{v}_k \mid k\Delta x \rangle = +1/\sqrt{2}$ and $\langle \mathbf{v}_k \mid -k\Delta x \rangle = -1/\sqrt{2}$. Therefore, for input $|k\Delta x\rangle$, the orthogonal components are $|\mu\rangle = |\mathbf{u}_k\rangle/\sqrt{2}$ and $|\nu\rangle = |\mathbf{v}_k\rangle/\sqrt{2}$; and for input $|-k\Delta x\rangle$, they are $|\mu\rangle = |\mathbf{u}_k\rangle/\sqrt{2}$ and $|\nu\rangle = -|\mathbf{v}_k\rangle/\sqrt{2}$. This completes the example and we return to the construction for an arbitrary non-injective surjection.

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 $y_i \in \text{Im } T$ 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 $\mathcal{N}(T)^{\perp}$:

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

The constant maps 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 $\mathbf{0} = T_i |\psi\rangle = |y_i\rangle \langle \mathbf{u}_i | \psi\rangle$, that is, if and only if $\langle \mathbf{u}_i | \psi\rangle = 0$. 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 $\{|\mathbf{u}_i\rangle\}$. Therefore $\mathcal{N}(T)^{\perp}$ is spanned by $\{|\mathbf{u}_1\rangle, \dots, |\mathbf{u}_m\rangle\}$.

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, the nonnull space. The first part generates "garbage," but is required for the operation to be invertible and hence unitary; the second part 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 $\{|\mathbf{w}_j\rangle \mid j = 1, \dots n_o\}$ be any basis for $\mathcal{H}(\Omega')^{\perp}$, the orthogonal complement of $\mathcal{H}(\Omega')$ in \mathcal{H}' , which also has dimension n-m. We 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. As a result, 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." Specifically, 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 superimposed on 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 [13, §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 unitary means for doing so.

It might seem that something like a Grover iteration [4] could be used to rotate the state vector from $s_j|y_i\rangle + |\gamma_j\rangle$ to $|y_i\rangle$, but different s_j 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 considered 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 $\Omega = \{x_1, ..., x_n\}$ and $\Omega' = \{y_1, ..., y_m\}$ be finite sets with $m \le n$. Suppose $\{|x_1\rangle, ..., |x_n\rangle\}$ is an ON basis for a Hilbert space \mathcal{H} and $\{|y_1\rangle, ..., |y_m\rangle\}$ is an ON basis for a subspace $\mathcal{H}(\Omega')$ of \mathcal{H} . For any surjection $f : \Omega \to \Omega'$ 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 $\mathcal{H}(\Omega')^{\perp}$.

C.4 Arbitrary Functions

In the preceding, we have assumed for convenience that the function is either non-injective or non-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. More directly, we can combine Prop. 3, to implement the function as a surjection onto its range, with 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 $\{|x_0\rangle, \dots, |x_n\rangle\}$. (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" from previous computations. Similarly, let the codomain $\Omega' = \{y_1, \dots, y_m\}$, where $m = |\Omega'|$, and let $\{|y_1\rangle, \dots, |y_m\rangle\}$ 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 $\{|y_0\rangle, \dots, |y_m\rangle\}$. The $|y_0\rangle$ component carries the garbage in the output state.

Let $\{r_1, \ldots, r_{m_r}\} \stackrel{\text{def}}{=} \text{Im } f$ and $\{z_1, \ldots, z_{m_{nr}}\} \stackrel{\text{def}}{=} \Omega \setminus \text{Im } f$ be the range of f and its complement, respectively; $n = m_r + m_{nr}$. As before, an n-dimensional input $|x_j\rangle$ will be projected into orthogonal subspaces (the nonnull and null spaces) of dimension m_r and $n_o = n - m_r$, with basis vectors $|\mathbf{u}_i\rangle$ and $|\mathbf{v}_k\rangle$, respectively.

An additional input quantum register will be used to provide the constant zero amplitudes for non-range elements for non-surjective functions. The m+1 dimensional state of this register will be in, for convenience, $\mathcal{H}_{\rm C}=\mathcal{H}(\Omega^*)$ with basis $\{|y_0\rangle,\ldots,|y_m\rangle\}$. There will also be an additional output quantum register to hold the null space garbage for non-injective functions. Its n+1 dimensional state is in, for convenience, $\mathcal{H}_{\rm G}=\mathcal{H}(\Omega^\circ)$ with basis $\{|x_0\rangle,\ldots,|x_n\rangle\}$. Note that both the input and output spaces have dimension (m+1)(n+1). This is because the ancillary input register is in the same space as the regular output register, and the ancillary output register is in the same space as the regular input register. This can be confusing because, as will be seen, we use the extra output vector $|y_0\rangle$ as a constant in the ancillary input register, and the extra input vector $|x_0\rangle$ appears in the ancillary output register.

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

$$U[(s|x_i\rangle + t|x_0\rangle) \otimes |y_0\rangle] = (s'|f(x_i)\rangle + t'|x_0\rangle) \otimes |\gamma\rangle,$$

for scalars s, s', t, t' and for $|\gamma\rangle \in \mathcal{H}_{G}$. That is, the input register is initialized to the input $|x_{j}\rangle$ with some positive amplitude s, possibly with superimposed garbage with amplitude t; the ancillary input register is initialized to constant $|y_{0}\rangle$. After computation, the output register will contain the functions value $|f(x_{j})\rangle$ with some positive amplitude s'; and superimposed garbage with amplitude t'. The ancillary output register may also contain garbage. In other words, the argument of f is in the first input register [corresponding to $\mathcal{H}(\Omega^{\circ})$], and its result is in the first output register [corresponding to $\mathcal{H}(\Omega^{\circ})$], 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 work of computing f is done by the graph kernel M, which will map the $|\mathbf{u}_i\rangle|y_0\rangle$ vectors into corresponding (m+1)(n+1) dimensional output vectors $|r_i\rangle|x_0\rangle$ in the output space $\mathcal{H}(\Omega^{\circ})\otimes\mathcal{H}_{G}$ (see Fig. 2). (The ancillary $|x_0\rangle\in\mathcal{H}_{G}$ output

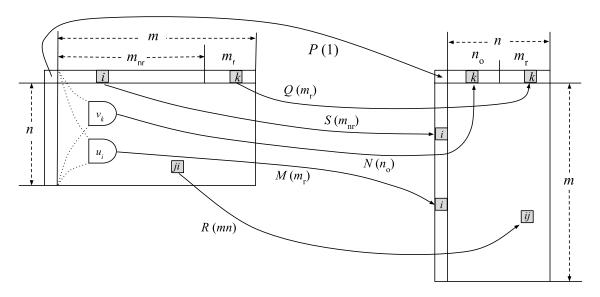


Figure 2: Permutation of basis vectors to implement arbitrary function. After each component of the kernel, the number of basis vectors that it maps is indicated in parentheses. For example, M maps mn basis vectors. The shapes labeled \mathbf{u}_i and \mathbf{v}_k represent projection onto the basis vectors of the nonnull and null spaces, respectively.

is required so that the input and output spaces have the same dimension.) To accomplish this mapping, define M as follows:

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

It maps $m_{\rm r}$ of the basis vectors of $\mathcal{H}(\Omega^{\circ}) \otimes \mathcal{H}_{\rm C}$ into $m_{\rm r}$ of the basis vectors of $\mathcal{H}(\Omega^{*}) \otimes \mathcal{H}_{\rm G}$ (see Fig. 2). Specifically it is a bijection between the nonnull input subspace of $\mathcal{H}(\Omega^{\circ}) \otimes \mathcal{H}(\{y_{0}\})$ and the range subspace of $\mathcal{H}(\Omega^{*}) \otimes \mathcal{H}(\{x_{0}\})$.

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

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

It maps them to n_o of the basis vectors $|y_0, x_k\rangle$ of the m+1 dimensional $|y_0\rangle$ subspace of the output space. 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 non-range codomain components $|z_i\rangle|x_0\rangle$:

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

This is a mapping of $m_{\rm nr}$ basis vectors between the $|x_0\rangle$ subspaces of the input and output spaces. The three operators M, N, S handle the mapping of f (and disposal of the null space).

We have to be careful, however, to handle all the $|x_0\rangle|y_i\rangle$ basis vectors since $m_{\rm nr}$ might be less than n (see Fig. 2). We map the remaining basis vectors of 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 $n_o + m_r = n = m_{nr} + m_r$, so that all these vectors are bijectively mapped.

It remains to handle the other components of the input space in a unitary way. The preceding maps have either $|x_0\rangle$ or $|y_0\rangle$, but not both, in the input register. The R operator maps the mn basis vectors with neither: $|x_j\rangle|y_i\rangle$, for $i, j \neq 0$, map into their reverses:

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

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

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

In summary, 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:

$$m_{\rm r} + n_o + m_{\rm nr} + mn + m_{\rm r} + 1 = (m_{\rm r} + n_o) + (m_{\rm nr} + m_{\rm r}) + mn + 1$$

= $n + m + mn + 1$
= $(m+1)(n+1)$.

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 $f: \Omega \to \Omega'$, where $\Omega = \{x_1, \ldots, x_n\}$ and $\Omega' = \{y_1, \ldots, y_m\}$. 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 $\{|y_0\rangle, \ldots, |y_m\rangle\}$. 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, t (with $|s|^2 + |t|^2 = 1$) and $x \in \Omega$:

$$U[(s|x\rangle + t|x_0\rangle) \otimes |y_0\rangle] = ss'|f(x), x_0\rangle + t'|y_0, \gamma\rangle,$$

where $s = 1/\sqrt{n_x}$, where $n_x = |f^{-1}\{f(x)\}|$, and $|ss'|^2 + |t'|^2 = 1$.

Proof: By construction we know:

$$U|x_{j}, y_{0}\rangle = (M+N)|x_{j}, y_{0}\rangle$$

$$= \left(\sum_{i=1}^{m_{r}} |r_{i}, x_{0}\rangle\langle \mathbf{u}_{i}, y_{0}|\right) |x_{j}, y_{0}\rangle + \left(\sum_{k=1}^{n_{o}} |y_{0}, x_{k}\rangle\langle \mathbf{v}_{k}, y_{0}|\right) |x_{j}, y_{0}\rangle$$

$$= |f(x_{j}), x_{0}\rangle\langle \mathbf{u}_{i} | x_{j}\rangle + |y_{0}\rangle \sum_{k=1}^{n_{o}} |x_{k}\rangle\langle \mathbf{v}_{k} | x_{j}\rangle$$

$$= s_{j}|f(x_{j}), x_{0}\rangle + |y_{0}, \gamma\rangle,$$

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 $s|x_j\rangle + t|x_0\rangle$ (with $|s|^2 + |t|^2 = 1$) we have:

$$U[(s|x_{j}\rangle + t|x_{0}\rangle) \otimes |y_{0}\rangle] = sU|x_{j}, y_{0}\rangle + tU|x_{0}, y_{0}\rangle$$

$$= s(s_{j}|f(x_{j}), x_{0}\rangle + |y_{0}, \gamma\rangle) + t|y_{0}, x_{0}\rangle$$

$$= ss_{j}|f(x_{j}), x_{0}\rangle + |y_{0}\rangle(s|\gamma\rangle + t|x_{0}\rangle).$$

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 $1/\sqrt{n_x}$ 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 set of corresponding outputs $f(x_j)$.

D Topographic Qubit Maps

D.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 $\Omega' = \{y_1, \ldots, y_m\}$. In these topographic qubit maps, 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 (in the brain perhaps as little as one digit [10, p. 378]). Therefore 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).

Like the topographic basis maps, these topographic qubit 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 the notation $|\{x_j\}\rangle$ for the topographic 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 2^n is the set of *n*-bit binary strings, and on the right 2^{Ω} is the powerset of Ω . Therefore, the sets are basis states and as a consequence topographic qubit maps permit *multiple* sets to 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 topographic qubit map

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

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

With the topographic qubit representation, the transformations between computational maps will be implemented by the quantum 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 following subsections), so that computational maps buy 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.

D.2 Unary Functions

An example will illustrate how to implement an arbitrary finite function $f: \Omega \to \Omega'$ by topographic qubit 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 $|\phi_i\rangle = |\psi_j\rangle$, transferring the state $|\psi_j\rangle$ of input qubit j to output qubit i. 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 $|\psi_k\rangle$. This is accomplished by the two-input OR₂ gate:

$$OR_2 \stackrel{\text{def}}{=} CCNOT(X \otimes X \otimes I),$$

where CCNOT is the conditional-conditional-not or Toffoli gate. The result of ORing the input states is:

$$OR_2(|\psi_i\rangle \otimes |\psi_k\rangle \otimes |1\rangle) = X|\psi_i\rangle \otimes X|\psi_k\rangle \otimes |\phi_i\rangle.$$

The $|1\rangle$ is an ancilla. The result of the OR is in the third output qubit, and the first two output qubits, in which the negated inputs remain, are considered garbage. If $|\psi_j\rangle = p'|0\rangle + p|1\rangle$ and $|\psi_k\rangle = q'|0\rangle + q|1\rangle$, then OR₂ transfers probability amplitudes as follows:

$$\begin{aligned}
\operatorname{OR}_{2}|\psi_{j}\rangle|\psi_{k}\rangle|1\rangle &= \operatorname{OR}_{2}(p'|0\rangle + p|1\rangle)(q'|0\rangle + q|1\rangle)|1\rangle \\
&= p'q'\operatorname{OR}_{2}|001\rangle + p'q\operatorname{OR}_{2}|011\rangle + pq'\operatorname{OR}_{2}|101\rangle + pq\operatorname{OR}_{2}|111\rangle \\
&= p'q'|110\rangle + p'q|101\rangle + pq'|011\rangle + pq|001\rangle.
\end{aligned}$$

Therefore, the third output qubit is the OR of the first two input qubits with the amplitudes shown. If we interpret the squares of the amplitudes as probabilities, then OR_2 computes the correct probabilities for the third output qubit. 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 recursively in terms of OR_2 :

$$\operatorname{OR}_{n}|\psi_{1}\rangle\cdots|\psi_{n}\rangle|1\rangle^{\otimes(n-1)}\stackrel{\text{def}}{=}\operatorname{OR}_{n-1}[(\operatorname{OR}_{2}|\psi_{1}\rangle|\psi_{2}\rangle|1\rangle)\otimes|\psi_{3}\rangle\cdots|\psi_{n}\rangle|1\rangle^{\otimes(n-2)}] \quad (n>2).$$

For completeness, we define $OR_1 = I$.

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

Proposition 5 Suppose $f: \Omega \to \Omega'$, where $\Omega = \{x_1, \ldots, x_n\}$ and $\Omega' = \{y_1, \ldots, y_n\}$. Let $m_{nr} \stackrel{\text{def}}{=} n - |\operatorname{Im} f|$ 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 $n_n = n - n_b$ be the

number of non-injective domain elements. Let $m_n = |\operatorname{Im} f| - n_b$ 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 $2n_n + n_b + m_{nr} - m_n$ dimensional unitary operator U_f that computes f by topographic qubit maps:

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

where $y_i = f(x_j)$ and $|\gamma\rangle$ is $2(n_n - m_n)$ qubits of garbage.

Proof: The inputs are the n elements of the input map, $m_{\rm nr}$ constant $|0\rangle$ ancillae (for the non-range elements), and $n_{\rm n}-m_{\rm n}$ constant $|1\rangle$ ancillae for the ${\rm OR}_2$ gates that map multiple domain elements to the same range element. The latter is because each of the non-injective range elements requires a number of ${\rm OR}_2$ gates that is one less than the number of its preimages; hence the m_n non-injective range elements require $n_{\rm n}-m_{\rm n}$ ${\rm OR}_2$ gates. Therefore, there are

$$n + m_{\rm nr} + n_{\rm n} - m_{\rm n} = (n_{\rm b} + n_{\rm n}) + m_{\rm nr} + n_{\rm n} - m_{\rm n} = 2n_{\rm n} + n_{\rm b} + m_{\rm nr} - m_{\rm n}$$

input qubits. The $m_{\rm nr}$ constant $|0\rangle$ s are passed directly to the output qubits for nonrange codomain elements. The $n_{\rm b}$ qubits for injective inputs are passed to the same number of output qubits, permuted as required. The outputs of the ORs project onto the $m_{\rm n}$ qubits that represent range values with more than one pre-image. Each OR₂ also generates two garbage qubits, for a total of $2(n_{\rm n}-m_{\rm n})$. Therefore the total number of output qubits is

$$m_{\rm nr} + n_{\rm b} + m_{\rm n} + 2(n_{\rm n} - m_{\rm n}) = 2n_{\rm n} + n_{\rm b} + m_{\rm nr} - m_{\rm n}$$

which is equal to the number of input qubits, as it should be. Next we define U_f explicitly as the tensor product of three operators:

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

We will use the notation $|1\rangle_q$ to represent a $|1\rangle$ state in qubit q, and $|0\rangle_q$ to represent a $|0\rangle$ state in qubit q.

The $U_{\rm nr}$ operator is an identity operation copying constant $|0\rangle$ ancillae into the codomain elements that are not in the range of f. Therefore, let $\{c_1, \ldots, c_{m_{\rm nr}}\} = \Omega' - \text{Im } f$ be this set, and let $|z_i\rangle$ be ancillae qubits to provide constant $|0\rangle$ s. Then $U_{\rm nr}: \mathcal{H}^{m_{\rm nr}} \to \mathcal{H}^{m_{\rm nr}}$ is defined:

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

That is, the states of the z_i input qubits (intended to be $|0\rangle$) are transferred to the c_i output qubits. This operator can be abbreviated by the following bracket notation:

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

It is just a permutation of the qubits, which might be implemented by SWAP operations.

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

$$U_{\mathbf{b}} \stackrel{\text{def}}{=} [u_1, \dots, u_{n_{\mathbf{b}}}] \leftarrow [v_1, \dots, v_{n_{\mathbf{b}}}].$$

For U_n we must OR together the domain elements corresponding to each range element with more than one preimage. Therefore we define U_n as a tensor product of operators for each such range element:

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

where these y_i have more than one preimage element; for example, $f^{-1}\{y_i\} = \{x_1, \ldots, x_{n_i}\}$, where $n_i = |f^{-1}\{y_i\}| \geq 2$. The output state $|\psi_i\rangle$ for such a range element is the OR of the input states $|\xi_i\rangle$ $(j = 1, \ldots, n_i)$ of its preimage elements:

$$|\psi_i\rangle|\gamma\rangle = \mathrm{OR}_{n_i}|\xi_1\rangle\cdots|\xi_{n_i}\rangle|1\rangle^{\otimes(n_i-1)},$$

where OR_{n_i} is a cascade of $n_i - 1$ OR_2 s and $|\gamma\rangle$ is $2n_i - 2$ dimensional garbage. This is accomplished by the operator $V_i(y_i, \{x_1, \dots, x_{n_i}\}) \in \mathcal{L}(\mathcal{H}^{2n_i-1}, \mathcal{H}^{2n_i-1})$:

$$V_i(y_i, \{x_1, \dots, x_{n_i}\}) \stackrel{\text{def}}{=} [y_i, \gamma_1, \dots, \gamma_{2n_i-2}] \leftarrow OR_{n_i}[x_1, \dots, x_{n_i}, o_1, \dots, o_{n_i-1}],$$

where o_1, \ldots, o_{n_i-1} are the qubits that provide ancillary $|1\rangle$ s for the ORs, and the garbage outputs $\gamma_1, \ldots, \gamma_{2n_i-2}$ receive the negated inputs and intermediate OR outputs. The bracket notation identifies the qubits that are the inputs and outputs of OR_{n_i} . This completes the construction of U_f .

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.

D.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 $\Omega' = \{y_1, \ldots, y_m\}$. We will represent the input to the function by a two-dimensional array of qubits for each (x_j, x_k) pair. (They do not have to be physically arranged as a two-dimensional array so long as there is a qubit for each pair of values.) 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 extra qubits (containing the inputs) in addition to the conjunction. However, these extra qubits are passed along the rows and columns to be used in other conjunctions, 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 computed by

$$|\phi_j\rangle|\psi_k\rangle|\chi_{jk}\rangle = \text{CCNOT}|\phi_j\rangle|\psi_k\rangle|0\rangle.$$

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 computes probability-like amplitudes if we interpret the squares of the amplitudes as probabilities.

Based on the foregoing, we define a unitary operator $U_{\rm OP}$ 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:

$$|\phi\rangle|\psi\rangle|\chi\rangle = U_{\rm OP}|\phi\rangle|\psi\rangle|0\rangle^{\otimes n^2},$$

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 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 $\Omega = \{0, \Delta x, \dots, (n-1)\Delta x\}$, then let $\Omega' = \{0, \Delta x, \dots, 2(n-1)\Delta x\}$. Generally, however, it is more convenient to have the codomain map 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, hsum(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, tsum $(x,y) = \min(x+y,x_n)$, where $x_n = \max \Omega$. This is the solution that we will use for illustration.

If $\Omega = \{0, \Delta x, \dots, (n-1)\Delta x\}$, 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 $(n-1)\Delta x$.

Proposition 5 shows how to implement the truncated addition operation, but it treats it as a unary function on an n^2 -dimensional space, which is wasteful since the intended output (the sum) is n-dimensional and the remaining $n^2 - n$ elements are garbage. Therefore, we implement a unitary operator that directly maps the input pairs to the corresponding outputs.

To compute the outer product we require n^2 constant $|0\rangle$ ancillae, and this computation also passes the two n-dimensional inputs through as garbage output. The qubit representing (0,0) maps bijectively to the output qubit y_1 representing 0. Each of the other n-1 output qubits y_i ($i=2,\ldots,n$) has two or more domain pairs mapping to it. As before, let n_i be the preimage multiplicity of output i and note that $\sum_{i=1}^n n_i = n^2$. Each of these non-injective outputs receives its value from an OR_{n_i} operation, which requires n_i-1 input $|1\rangle$ ancillae and generates $2n_i-2$ qubits of output garbage (n_i for the negated inputs and n_i-1 for the intermediate disjunctions). Therefore, the total number of $|1\rangle$ ancillae is

$$\sum_{i=2}^{n} (n_i - 1) = \sum_{i=2}^{n} n_i - (n-1) = (n^2 - 1) - n + 1 = n^2 - n.$$

Moreover, the complete input dimension is $2n + n^2 + n^2 - n = 2n^2 + n$. This is

also the complete output dimension, for we have n qubits for the function value, 2n qubits for the passed-through input arguments (garbage), and the garbage output from the OR gates, which is: $\sum_{i=2}^{n} (2n_i - 2) = 2(n^2 - n)$. That is, the complete output dimension is $3n + 2(n^2 - n) = 2n^2 + n$. In summary, there is a unitary operator $U_{\text{tsum}} \in \mathcal{L}(\mathcal{H}^{2n^2+n}, \mathcal{H}^{2n^2+n})$ so that

$$U_{\text{tsum}} \mid \{x\} \rangle \mid \{y\} \rangle \mid 0 \rangle^{\otimes n(n-1)} = \mid \{\text{tsum}(x,y)\} \rangle \mid \{x\} \rangle \mid \{y\} \rangle \mid \gamma \rangle,$$

where the garbage $|\gamma\rangle$ has dimension 2n(n-1) (the passed-through inputs may also be considered garbage). Based on this example, we state a more general result.

Proposition 6 Suppose $f: \Omega \times \Omega \to \Omega$ and let $n = |\Omega|$. Let $m_{nr} = n - |\operatorname{Im} f|$ be the number of codomain elements that are not in the range of f. Then there is a unitary operator $U_f \in \mathcal{L}(\mathcal{H}, \mathcal{H})$, where \mathcal{H} is $2n^2 + n + 2m_{nr}$ dimensional Hilbert space, such that:

$$U_f |\{x\}\rangle |\{y\}\rangle |0\rangle^{\otimes (n^2 + m_{\text{nr}})} |1\rangle^{\otimes (n^2 - n + m_{\text{nr}})} = |\{f(x, y)\}\rangle |\{x\}\rangle |\{y\}\rangle |\gamma\rangle,$$

where the garbage $|\gamma\rangle$ has dimension $2(n^2 - n + m_{nr})$ (the 2n passed-through inputs may also be considered garbage).

Proof: The operator is constructed very similarly to U_{tsum} , but we also have to consider non-range codomain elements for non-surjective functions, which didnt occur in that case. As before, the computation of the outer product will require n^2 ancillary $|0\rangle$ inputs and it will generate 2n qubits containing the passed-through inputs. We can consider disjoint subsets of the codomain. Codomain elements that are not in the range of f will need to be sent a $|0\rangle$ state, for which we need an additional $m_{\rm nr}$ ancillary $|0\rangle$ inputs. Let $n_{\rm b}$ be the number of input pairs that are mapped one-to-one to the corresponding outputs; they neither require ancillary constants nor generate garbage. The remaining codomain elements are range elements with $n_i \geq 2$; let $m_{\rm n} = n - m_{\rm nr} - n_{\rm b}$ be the number of them. Each of these will receive the OR of the corresponding (preimage) domain elements. As we saw previously, the OR_{n_i} operation requires $n_i - 1$ ancillary $|1\rangle$ qubits and produces $2n_i - 2$ qubits of garbage. Therefore, the total number of $|1\rangle$ qubits required for the $n^2 - n_{\rm b}$ input pairs mapping to the $m_{\rm n}$ non-injectively mapped range elements is:

$$\sum_{i=1}^{m_{\rm n}} (n_i - 1) = n^2 - n_{\rm b} - m_{\rm n} = n^2 - n + m_{\rm nr},$$

since $m_{\rm n}=n-m_{\rm nr}-n_{\rm b}$. The garbage generated by the ORs is then $\sum_{i=1}^{m_{\rm n}}(2n_i-2)=2(n^2-n+m_{\rm nr})$. The complete input dimension is 2n (arguments) + $(n^2+m_{\rm nr})$ (for

 $|0\rangle$ ancillae) + $(n^2 - n + m_{\rm nr})$ (for $|1\rangle$ ancillae) = $2n^2 + n + 2m_{\rm nr}$. The complete output dimension is 3n (arguments and result) + $2(n^2 - n + m_{\rm nr})$ (garbage) = $2n^2 + n + 2m_{\rm nr}$.

The same approach can be used for operations with more than two arguments, but the number of qubits increases exponentially with the number of arguments.

E Conversions Between Representations

Ordinary binary representations can be translated to topographic qubit maps 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:

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

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

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 topographic qubit 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 $n^{-1} \sum_{j=1}^n p_j x_j$? A normalized superposition of all the values? The answer is not apparent, so we leave the question open.⁴

F Applications to Quantum Machine Learning

Given this general ability to compute non-unitary and even nonlinear functions by means of topographic qubit maps, 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-dimensional vectors requires N multiplications and N-1 additions. If $|\Omega| = n$, then each multiplication and addition will require approximately $2n^2$ qubits, for a total of about $2N^2n^2$.

⁴ 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.

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 topographic qubit 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 about $2MN^2n^2$ qubits for the computation (not including ancillary qubits).

G 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, called a topographic basis map, 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 representing crisp sets of inputs, but there is an unavoidable data-dependent attenuation of the result (relative to a "garbage" state) that is not easily avoidable. The second approach, called a topographic qubit map, assigns a separate qubit to each location in the map, and uses the relative amplitude of the $|1\rangle$ and $|0\rangle$ states 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. In particular, computational maps enable the implementation of the functions commonly used in artificial neural networks.

References

- C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen. A large-scale model of the functioning brain. *Science*, 338:1202–1205, 2012.
- [2] E. F. Fredkin and T. Toffoli. Conservative logic. *Int. J. Theo. Phys.*, 21(3/4):219–253, 1982.
- [3] E. F. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown. Overview of the SpiNNaker system architecture. *IEEE Transactions on Computers*, 62:2454–2467, 2013.
- [4] Peter Høyer. Arbitrary phases in quantum amplitude amplification. *Phys. Rev.* A, 62:052304, Oct 2000.
- [5] 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.
- [6] Bruce J. MacLennan. Field computation in natural and artificial intelligence. *Information Sciences*, 119:73–89, 1999. Also available from web.eecs.utk.edu/~mclennan.
- [7] 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.
- [8] Bruce J. MacLennan. The promise of analog computation. *International Journal of General Systems*, 43(7):682–696, 2014.
- [9] 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.
- [10] J. L. McClelland, D. E. Rumelhart, and PDP Research Group. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models. MIT Press, Cambridge, 1986.

- [11] P. G. Morasso and V. Sanguineti. Self-Organization, Computational Maps and Motor Control. North-Holland, Amsterdam, 1997.
- [12] E. Neftci, J. Binas, U. Rutishauser, E. Chicca, G. Indiveri, and R. J. Douglas. Synthesizing cognition in neuromorphic electronic systems. *Proceedings of the National Academy of Sciences of the United States of America*, 110:E3468–E3476, 2013.
- [13] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge, Cambridge, 10th anniversary edition edition, 2010.
- [14] R. C. O'Reilly, Y. Munakata, M. J. Frank, T. E. Hazy, and contributors. Computational Cognitive Neuroscience. http://ccnbook.colorado.edu/, 2nd edition, 2014.
- [15] D. E. Rumelhart, J. L. McClelland, and PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations.* MIT Press, Cambridge, 1986.
- [16] T. D. Sanger. Probability density estimation for the interpretation of neural population codes. *Journal of Neurophysiology*, 76:2790–2793, 1996.
- [17] M. Schuld, I. Sinayskiy, and F. Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13:2567–2586, 2014.
- [18] M. Schuld, I. Sinayskiy, and F. Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56:172–185, 2015.