Quantum Neurons

Willard L. Miranker
Department of Computer Science
Yale University, TR1234

Quantum Neurons

Willard L. Miranker
Department of Computer Science
Yale University

Abstract: Several neuronal models, each as a quantum function evaluator, are defined. Synaptic inputs are qubits. Output is a quantum state, in some cases also a qubit. A quantum Hebb rule is defined, and a quantum version of Hebbian learning is formulated. A quantum Hopfield net is specified, and applications for it are made to a wide problem class. In all cases the quantum variant majorizes the computational complexity of the classical counterpart. Finally a so-called seriously entangled quantum neuron is described. Such a quantum neuron can instantiate an arbitrary number of neural nets, problem types and problem instances, simultaneously.

1. Introduction

Quantum computing (Deutsch, 1985, Steeb, 1998, Bouwmeester, Ekert, Zeilinger (Eds.), 2000) brings improvement in computational complexity to a variety of problems (Deutsch, Jozsa, 1992, Shor, 1997, Grover, 1998, Jozsa, 2000). Gains vary, in at least one case (the factorization of integers algorithm of Shor) achieving an exponential speedup with respect to problem size compared to any classical[1] computation. Of course there are other key attributes of quantum computing. These include theoretical concepts that impact the theory of computation and philosophical implications that inform our understanding of reality (Deutsch, 1997). Neural nets provide a computational paradigm with wide applicability in cognitive information processing, both real (brain circuitry modeling) and artificial (Hertz, Krogh, Palmer, 1991, Haykin, 1999).

Our object here is to combine these two areas of study. Neural nets provide quantum computing a wide set of applications and resultant cognitive constructs. Quantum computing provides neural nets with computational complexity improvement. The

---

[1] Throughout, classical means not quantum as in the distinction between classical and quantum physics.

expectation is that the combination will stimulate the production of deeper ideas for understanding cognition as well.

We begin with a brief review of quantum computing framed in a context suitable for defining neurons that process information as quantum function evaluators. We call such neurons, quantum neurons. To read the information produced by a quantum neuron may require a quantum measurement (a measurement of the spin that encodes the qubit (the quantum bit)). So three types of quantum neurons are specified. These neurons, types (*i*), (ii), (*iii*), respectively, are the quantum analogs of certain classical model neurons, namely, of linear neurons, of McCulloch-Pitts neurons, and of stochastic neurons, respectively. Type (*i*) requires no measurement, the others do.

The applications that we give in Sections 3 and 4 use neurons of type (*i*) and (*ii*) variously and in combination. The neuron of type (*i*), employing no measurement, as it does, seems to be especially advantageous. Indeed measurement has a computational cost (time), and it de-coheres (collapses) an entangled state, which in principle results in a loss of information.

Our first application is to Hebbian synaptic dynamics and involves formulation of a quantum Hebb rule. Hebbian dynamics are a widely accepted model for how living neurons adjust their function. These dynamics are an autonomous (i.e., unsupervised) method for recording information (so-called learning) in the collection of synaptic weights in a manner depending only on the neuronal activity and correlation between a neuron's inputs and its output. We shall see that the effort for updating a neuron's N synaptic weights in one step of Hebbian dynamics is reduced from O(N) operations for a classical neuron to O(1) operations for the quantum neuron.

Our second application treats Hopfield nets (recursive neural nets) that correspond to an energy (Lyapunov) function whose value decreases as the net's input-output dynamics progresses. Many interesting problems (the Traveling Salesman Problem, for example) can be cast into the form of minimizing a suitable such energy function. We shall apply a net of N quantum neurons to expedite delivery of a minimum point by the descent dynamics. We entangle all of the 2^N vertices of the N-cube on which the dynamics is set and conduct the descent simultaneously on all members of this exponentially entangled quantum state. The complexity benefits provided by this class of applications are varied, capable of exponential improvement for special problem classes.

Finally the process of entanglement motivates defining neurons each of which has multiple sets of weights. One such neuron is able to perform the computation of an entire

net. Continuing, injection of multiple sets of entangled inputs enables one neuron to operate on more that one problem at a time. Then we consider so-called seriously entangled neurons that have both types of multiplicities simultaneously. We augment the entangled members of the multiple sets of weights and inputs with labels (quantum decryption keys) that are concatenations of qubits to track the information flow.

2. Quantum Neurons

In this section we define several types of quantum neurons as quantum function evaluators. The synaptic inputs are qubits. The output is a quantum state that is only sometimes a simple qubit.

2.1 Input qubits

A neuron has N input synapses. The input at each synapse is an on-off signal. Since we employ quantum notions, such inputs are taken to be spin valued. Thus the input at the i-th synapse is the ket

2.1) $$|a_i\rangle,\ a_i \in \{0,1\},\ i = 0,\ldots,N.$$

Here

2.2) $$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix},\quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

These binary valued kets are known as qubits[2]. To account for the neuron's threshold, we clamp the ket $|a_0\rangle$. In particular, $|a_0\rangle = |0\rangle$.

The Hadamard transformation and the quantum *Not*

We shall make use of two unitary operators on qubits. The Hadamard transformation $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ and the quantum $Not = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Note that

2.3) $$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

and that

[2] Some interchange the assignment of $|0\rangle$ and $|1\rangle$ in (2.2), a source of confusion in the subject.

2.4) $\quad Not\ |0\rangle=|1\rangle \quad \text{and} \quad Not|1\rangle=|0\rangle.$

Synaptic weights, binary fixed point arithmetic

A synapse has a weight w_i, $\forall i$, associated with it. In particular,

2.5) $\quad w_0 = -\theta$

is the neuronal bias or threshold. Weights take on values from among the binary integers $0,1,\cdots,2^N-1$. A weight is encoded as the following quantum state.

2.6) $\quad w_i = \sum_{x=0}^{2^N-1} c_x^i |x\rangle, \quad c_x^i \in \{0,1\}.$

Since $2^N = 1\underbrace{0\cdots0}_{N}$ (in binary), the kets on the right in (2.6) are specified as follows.

2.7) $\quad |0\rangle = \underbrace{|0\cdots0\rangle}_{N}, \quad |1\rangle = \underbrace{|0\cdots01\rangle}_{N}, \quad \cdots, \quad |2^N-1\rangle = \underbrace{|1\cdots1\rangle}_{N}.$

So in the jargon of computer arithmetic, what we have is binary fixed point arithmetic in the quantum computation being discussed. Note that there is an overload in the notation, since $|0\rangle$ stands for both $\begin{pmatrix}1\\0\end{pmatrix}$ (as in (2-2)) and $|0\cdots0\rangle$ (as in (2.7)). Likewise $|1\rangle$ stands for both $\begin{pmatrix}0\\1\end{pmatrix}$ and $|0\cdots01\rangle$. For the sake of clarity, we shall, when necessary, replace $|0\rangle$ and $|1\rangle$ in (2.7) by $|\vec{0}\rangle$ and $|\vec{1}\rangle$, respectively.

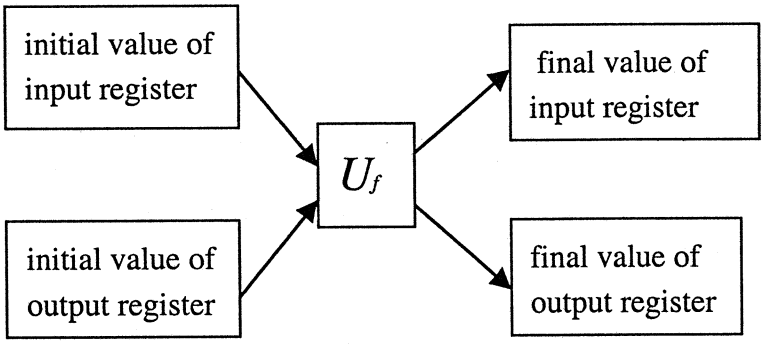
Quantum states are by definition normalized, so we shall always take that to be the case here. While this may be achieved in (2.6) by choosing the c_x^i so that only one term in the sum there survives (since each fixed point number of relevance is uniquely so represented), there are other possibilities. This scope of possible representations is indicative of the augmented power of quantum computation compared to classical.

2.2 Neuronal response

A neuron processes its input qubits. This processing will be represented as a parallel quantum function evaluation. The latter in turn is implemented by the operator U_f that

instantiates a function f as a unitary transformation. U_f has two operands (called initial values) each placed in a separate one of two registers. These registers are called the input and output registers, respectively. The result of the transformation (consisting of two parts called final values) is displayed in the same two registers. This is illustrated in (2.8).

2.8)

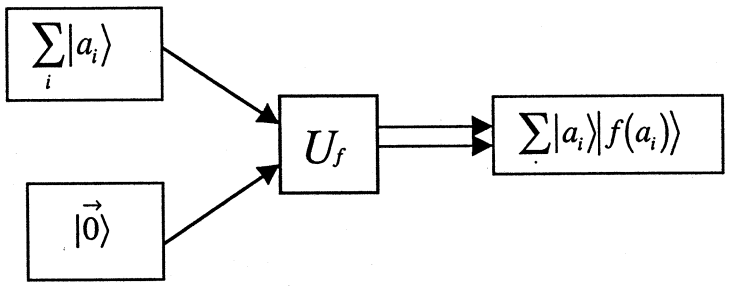


To specify the neuronal processing, place $\sum_i |a_i\rangle$ in the input register and place $|\vec{0}\rangle = |0\cdots 0\rangle$ in the output register. The output of U_f is the entangled quantum state

2.9) $$\sum_I |a_i\rangle |f(a_i)\rangle.$$

This is illustrated in (2.11).

2.10)



The function f

Define f as

2.11) $$f(a_i) = f(i) = w_i, \quad \forall i.$$

Thus f depends on a_i only through the value of the index i. With f defined as in (2.11), the output (2.10) of U_f is the following entangled quantum state.

2.12) $$u = \sum_i |a_i\rangle|w_i\rangle.$$

Using (2.1), (2.2), we write this as

2.13) $$u = \alpha|0\rangle + \beta|1\rangle,$$

where[3]

2.14) $$\alpha = \sum_{i:a_i=0} |w_i\rangle = \sum_i \delta_{a_i,0}|w_i\rangle \quad \text{and} \quad \beta = \sum_{i:a_i=1} |w_i\rangle = \sum_i \delta_{a_i,1}|w_i\rangle.$$

Note that $\delta_{a_i,0} + \delta_{a_i,1} = 1$, and that since quantum states are normalized, $\alpha^2 + \beta^2 = 1$.

2.3 Neuronal output

We shall consider three kinds of neuronal output called: (*i*) non-measured output (*ii*) threshold output, and (*iii*) collapsed output, respectively. A corresponding quantum neuron is denoted as a neuron of type (*i*), (*ii*), or (*iii*), respectively.

(*i*) Non-measured output

The classical neuronal correspondent to this is the case of a linear neuron with identity for the neuronal gain function. In this classical case the total weighted input is the output. Then the output the neuron of type (*i*) is taken to be the state of the neural output function U_f itself, that is, the quantum state specified in (2.12).

(*ii*) Threshold output

Threshold output is a quantum version of the classical McCulloch-Pitts neuron, where the neuron fires only if the total weighted input exceeds a threshold. To determine whether that is the case requires a quantum measurement of the output register in (2.10), as we shall see. A type (*ii*) neuron fires the output $|v\rangle$, where

---

[3] The order in a string of concatenated kets is not critical and is usually maintained to keep track of which ket is which. For convenience we have written u in (2.14) as a linear combination of spins $|0\rangle$ and $|1\rangle$, but at the cost of permuting the input and output kets in (2.13). Confusion should not occur. We shall, also for convenience, use this type of representation of the entangled output of the entangled output of U_f (as in (2.14)) a number of times in what follows.

2.15) $$|v\rangle = \begin{cases} |0\rangle, & |\alpha| \geq |\beta|, \\ |1\rangle, & |\alpha| < |\beta|. \end{cases}$$

So firing a qubit $|0\rangle$ or $|1\rangle$ corresponds to the McCulloch-Pitts case of firing or not firing.

To determine the output specified by (2.15), make a quantum measurement for the spin on the contents of the resultant input register of U_f, that is, a measurement for $|0\rangle$ versus $|1\rangle$. The measurement yields $|0\rangle$ or $|1\rangle$ with probability α^2 or $\beta^2 = 1 - \alpha^2$, respectively. After the measurement of the input register, the output register will have the contents α or β, as the case may be. If the content of the latter is α, say (i.e., if the measurement of the input register yielded $|0\rangle$, compute $|\alpha|^2$. If $|\alpha|^2 \geq 1/2$, set $|v\rangle = |0\rangle$. If $|\alpha|^2 < 1/2$, set $|v\rangle = |1\rangle$. If the measurement made on the input register yielded $|1\rangle$, proceed analogously to define the output $|v\rangle$.

(*iii*) Collapsed output

Collapsed output corresponds to the classical stochastic neuron, where the neuron fires with a probability depending upon the total weighted input. For a type (*iii*) neuron this probability is specified by a quantum measurement of the output register (as in (*ii*)).

We take as the neuronal output, the direct result of a quantum measurement of the entangled state u in (2.13) (in particular, measurement of the input register), measured for $|0\rangle$ versus $|1\rangle$. Then the neuron will be taken to fire the content of the input register, post measurement. Namely, the neuron fires the collapsed quantum state

2.16) $$|v\rangle = \begin{cases} |0\rangle \ \textit{with probability } \alpha^2, \\ |1\rangle \ \textit{with probability } \beta^2. \end{cases}$$

As in the case of type (*ii*) neurons, the quantities α or β can be obtained from the contents of the output register, post measurement.

Comment on the nature of the applications to follow

The applications that we give in Sections 3 and 4 use neurons of type (*i*) and (*ii*) variously and in combination. The neuron of type (*iii*), perhaps the most natural in the quantum context, is not yet used in our applications, and is included here for completeness. The neuron of type (*i*), employing no measurement, as it does, seems to be especially advantageous for applications. There is an intrinsic reason for this. Namely a

quantum measurement has a computational cost (time), and it de-coheres (collapses) an entangled state. It is the means of extracting information from a quantum state, but it does so at the cost of a loss of other information represented in that state. Thus it is particularly advantageous for a quantum computation process to defer a measurement as long as possible, preferably to when it is used to extract the information sought. This point of view characterizes the applications to follow. They each involve iterative algorithms. The entangling is exploited as a way to multiplex the computation (parallelize it), and incidence of measurements (the extraction of information) is minimized, and where possible, deferred to the end, as we shall see.

3. Hebbian Dynamics

Hebbian dynamics compose a fundamental procedure for changing a classical neuron's synaptic weights, $w_i, i = 1, \cdots, N$ (Haykin, 1999). It is a widely accepted model for how living neurons adjust their information processing. These dynamics are an autonomous (i.e., unsupervised) method for recording information (so-called learning) in the collection of those weights in a manner depending only on the neuronal activity and correlation between a neuron's inputs and its output. In this section we shall see that the effort for updating a neuron's weights in one step of Hebbian dynamics is reduced from O(N) operations for a classical neuron to O(1) operations for the quantum neuron defined in Section 2. In fact we shall make a double use of quantum neurons. The first of these (a neuron of type (*i*)) implements determination of the correlation between the input and output required for the Hebb rule. The second, the neuron whose weights we seek to update (a neuron of type (*ii*)) implements the neuronal input-output dynamics.

To proceed we define a function f_1 to compute the correlation needed.

3.1) $$f_1(a_i) = a_i, \forall i,$$

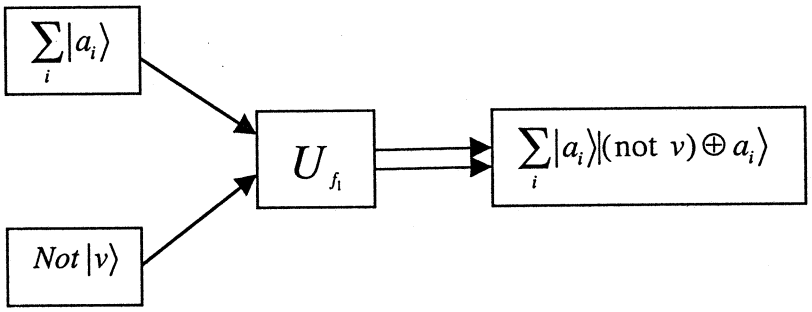
where $|a_i\rangle$ is the input to synapse i (c.f. (2.1)-(2.2)). While f_1 is simply the identity function, its quantum unitary correspondent, U_{f_1}, is hardly trivial. Indeed we shall exploit U_{f_1} in a special way that entangles the input states appropriately. First let $\oplus$ denote addition mod 2 and note the following expression of correlation between the pair of qubits $|a\rangle$ and $|v\rangle$, where the latter is the neuronal output.

3.2) $$(\text{not } v) \oplus a = \begin{cases} 0 \\ 1 \end{cases}, \text{ if } v \text{ and } a \text{ are correlated } \begin{cases} \text{positively} \\ \text{negatively} \end{cases}.$$

Here not is the Boolean not. Although there are other possibilities, (3.2) represents the way we shall model, in the context of quantum computation, the correlation of inputs and outputs at the core of Hebb's rule for synaptic weight change. Recall that $|v\rangle$ is used to denote the output qubit of a neuron of type (*ii*)

Now we define the unitary transformation U_{f_1} corresponding to the function f_1. To do this, place $\sum_i |a_i\rangle$ in the U_{f_1} input register, and place $Not\,|v\rangle$ (here we have the quantum *Not*, cf. (2.4)) in the output register. The result is illustrated in (3.3)

3.3)



The output in (3.3) may be written as follows (c.f. (2.14) and (2.15)). (See footnote 2.)

3.4) $$\sum_i |a_i\rangle|(\text{not } v)\oplus a_i\rangle = \sum_i \delta_{v,a_i}|a_i\rangle|0\rangle + \sum_i \delta_{1\oplus v,a_i}|a_i\rangle|1\rangle.$$

Let us index the variables a, v, and w with n, the clock cycle number that paces the neuronal dynamics. Now

3.5) $$a_j^{(n+1)} = v_j^{(n+1)}, \forall j,$$

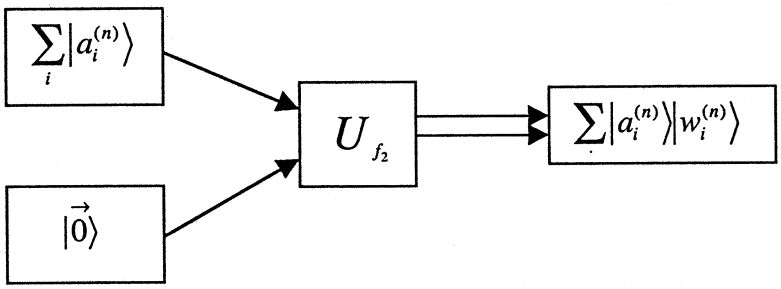
where $|v_j\rangle$ represents the output of neuron j (for the moment, we are thinking of a network of N neurons, each of which being subjected to Hebbian dynamics for synaptic weight update). So the neuron of (3.3) is a quantum neuron of type (*i*). Now to complete a step in the dynamics, we must update v and w_i, $\forall i$. That is, we must determine $v^{(n+1)}$ and $w_i^{(n+1)}$, $\forall i$. Recall that the w_i denote the synaptic weights in question.

First we update v as in Section 2.1. To do this we use the function f in (2.11) (renamed f_2 here) relevant to the neuron being updated. This will be a neuron of type (*ii*). Let

3.6) $$f_2(a_i) = |w_i\rangle.$$

Next place $\sum_i |a_i^{(n)}\rangle$ in the input register of U_{f_2} and $|\vec{0}\rangle$ in the output register. Then

3.7)



Write the output of U_{f_2} as

3.8) $$\sum_i |a_i^{(n)}\rangle |w_i^{(n)}\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where (see footnote 2)

3.9) $$\alpha = \sum_i \delta_{a_i,0} |w_i^{(n)}\rangle \quad \text{and} \quad \beta = \sum_i \delta_{a_i,1} |w_i^{(n)}\rangle.$$

Now measure the input register in (3.7) for $|0\rangle$ versus $|1\rangle$. The output register will contain α or β, accordingly. Then from (2.15)*ff*, the output we seek is

3.10) $$v^{(n+1)} = \begin{cases} |0\rangle, & |\alpha|^2 \geq 1/2, \\ |1\rangle, & |\alpha|^2 < 1/2. \end{cases}$$

So the neuron corresponding to (3.7) is a neuron of type (*ii*).

A quantum Hebb rule

Next to update $|w_i^{(n)}\rangle$, $\forall i$, we implement Hebb's law. That is, each weight $w_i^{(n)}$ is updated, augmented (or not), according to the value of (not v) $\oplus\, a_i$ (c.f. (3.2)). Then add the entangled contents of the resultant registers of (3.3) and (3.7). We obtain

3.11) $$\sum_i |a_i^{(n)}\rangle |w_i^{(n+1)}\rangle = \sum_i |a_i^{(n)}\rangle \left[|w_i^{(n)}\rangle + |(\text{not } v^{(n)}) \oplus a_i^{(n)}\rangle \right],$$

which we call a quantum Hebb rule[4]. So, the state $w_i^{(n)}$ is increased by unity (or left invariant) if $v^{(n)}$ and $a_i^{(n)}$ are positively (or negatively) correlated (c.f. (3.2)). (Of course the updated state $|w_i^{(n+1)}\rangle$ is normalized as in the discussion relevant to (2.6).)

Note that the Hebbian dynamics need not converge. A neural net may learn while the weights are still in flux. In fact, the learning process is usually monitored by the learning quality (e.g., the quality of the response of the net to an input cue) without regard for the progress of the weights. There are exceptions; for instance, weights are usually not allowed to grow without bound.

4. Hopfield Nets

Hopfield nets are recursive neural nets. The state of the net (the collection of neural outputs) evolves in a manner specified by the net's encoding (the synaptic weights). An interesting class of such nets corresponds to an energy (Lyapunov) function whose value decreases as the net's input-output dynamics progresses. The associated trajectories converge to one or another fixed-point of the dynamics (to a minimum point of the energy function).

Many interesting problems can be cast into the form of minimizing a suitable such energy function. So defining the corresponding net and running the associated descent dynamics (the input-output dynamics) provides a method for finding solutions to these problems. The NP complete Travelling Salesman Problem is a conspicuous example, but there are very many others.

---

[4] As with classical neural nets, there are other possible choices of the Hebb rule for quantum neurons.

In this section we shall apply a net of quantum neurons to expedite delivery of a fixed-point by the descent dynamics. Without loss of generality, we confine our attention to problems whose possible solution states are binary vectors, i.e., vertices of the unit N-cube. We entangle all of the 2^N vertices of the N-cube and conduct the descent on this exponentially entangled quantum state. The complexity benefits provided by this class of applications are varied, capable of exponential improvement for special problem classes.

Note that the quantum neurons we shall use here are of type (i), that is, the output is non-measured.

Descent dynamics on the quantum states

The binary vector $A = (a_1, \cdots, a_N)^T$, $a_i \in \{0,1\}$, $\forall i$, a vertex of the unit N-cube corresponds to a vector ket $|A\rangle$ of spins, where

4.1) $$|A\rangle = \left(|a_1\rangle, \cdots, |a_N\rangle\right)^T.$$

The set $\{w_{ij}\}$ of the net's synaptic weights specifies a matrix W. The input-output relation of the net is

4.2) $$|V\rangle = W|A\rangle,$$

where each component of V is specified by a quantum computation as in (2.12). The successor of A in the input-output dynamics is the updated input (type (i) neurons). Then using n to index the clock cycle, a step of the descent dynamics on the quantum net is given by

4.3) $$\left|A^{(n+1)}\right\rangle = |W\rangle\left|A^{(n)}\right\rangle, n = 0,1,\cdots.$$

The initial ket corresponds to an arbitrarily prescribed vertex $A^{(0)}$ of the N-cube. While the net implements a quantum function evaluation corresponding to multiplication by W, the output, a (normalized) quantum state is in fact (could be viewed as being) produced by multiplication by CW, where C is the diagonal matrix

4.4) $$C = \left(\left(\sum_{k=1}^{N} w_{ik}^2\right)^{-\frac{1}{2}} \delta_{ij} \right).$$

Now consider the following state $\left|B^{(0)}\right\rangle$ consisting of an entanglement of all of the vertices of the N-cube (c.f. (2.7)).

4.5) $$\left|B^{(0)}\right\rangle = \frac{1}{2^{N/2}} \sum_{x=0}^{2^N-1} |x\rangle.$$

Although this state consists of a superposition of an exponential number (2^N) of states, it can be generated by a linear number, in fact, N of applications of the Hadamard transformation H (c.f. (2.3)). Namely,

4.6) $$\left|B^{(0)}\right\rangle = \underbrace{(H \otimes \cdots \otimes H)}_{N} \underbrace{|0 \cdots 0\rangle}_{N}.$$

We use the direct product symbol $\otimes$ here to denote that the operators H are applied once and the results added. Each application is made separately to a different one of the N spins in the ket $|0 \cdots 0\rangle$. We replace (4.3) by

4.7) $$\left|B^{(n+1)}\right\rangle = |W\rangle \left|B^{(n)}\right\rangle, n = 0,1,\cdots .$$

A model case (quantum state purification)

A computationally efficacious case has a unique fixed-point A^* and certain other properties that we now describe. Partition the vertices of the N-cube into level sets where A^* has level zero, the lowest level. The nearest neighbors of A^* on the N-cube (Hamming distance unity) have level one. The next nearest neighbors have level two... In this hypothesized case a vertex following the descent dynamics must descend at least one level per step (except, of course, for A^*). Thus the length of any path on the N-cube following the descent dynamics is at most N. In the case of a classical computation, at most N steps are required to proceed from any initial guess (vertex) to A^*. The same is true of the quantum case. In fact all 2^N vertices are simultaneously (entangled) starting points of descent paths, and as the quantum algorithm (4.7) proceeds, all of these entangled paths converge to A^*. Thus after at most N descent steps, the evolving quantum state is the pure state $\left|A^*\right\rangle$. So a quantum measurement (one measurement each to disclose each of the N qubits comprising $\left|A^*\right\rangle$) after N steps delivers A^*. The $O\left(N^2\right)$ classical operations per step are reduced to O(N) operations plus N quantum function evaluations.

Summarizing, we have

4.8) $$\left|B^{(n)}\right\rangle = \left|A^*\right\rangle, n \geq N,$$

so that for all $n \geq N$, a measurement of each the N components of $|B^{(n)}\rangle$ for $|0\rangle$ versus $|1\rangle$ determines A^* and solves the problem in question.

We shall exploit this qualitative feature of quantum state purification observed in this quantum version of the descent dynamics in other applications (in the future).

Non-unique fixed point and neighbor level sets

In general there are multiple fixed-points, A_s^*, $s = 1, \cdots, S$, of the descent dynamics. Each fixed point has a basin of attraction, a subset $\beta(A_s^*)$ of the vertices of the N-cube. A trajectory of the descent dynamics terminates at A_s^* if and only if it starts in $\beta(A_s^*)$. Let us retain the hypothesis concerning the level sets. Then we see that no path in the descent dynamics exceeds $N - S$ in length. Let $\|\beta(A_s^*)\|$ denote the number of vertices in $\beta(A_s^*)$, and let

4.9) $$p_s = \frac{\|\beta(A_s^*)\|^2}{\sum_\sigma \|\beta(A_\sigma^*)\|^2}, s = 1, \cdots, S.$$

Then for $n > N - S$, a measurement of the input register will yield A_s^* with probability p_s. So the relative size of the basins of attraction grades this stochastic process of finding fixed-points by the quantum neuronal net.

Let $\theta_s = \|\beta(A_s^*)\| / 2^N$ denote the fraction of vertices of the N-cube that are in $\beta(A_s^*)$. Then (4.9) becomes

4.10) $$p_s = \frac{\theta_s}{\sum_\sigma \theta_\sigma^2}, \forall s.$$

In the special case that all of the basins of attraction are the same size, that is, if $\|\beta(A_s^*)\| = 2^N / S$ then $p_s = 1/S$, $\forall s$.

The general case

The general case has multiple fixed-points and lacks the efficacious level set property. So the descent dynamics are hardly as simple as in the previous two cases. Indeed some paths arriving at a fixed-point are exponentially long. The most we can say at this point is that a measurement will yield A_s^* (or a vertex close to it) with a probability

approaching p_s as the clock cycle number n increases. These observations motivate study of procedures for shortening paths and for adjusting basin sizes. The desired fixed point(s) should have basin(s) of attraction that are as large as possible.

Other optimization problems

It should be clear that the process of entangling all of the vertices of the N-cube and conducting a quantum state purifying descent iteration to solve a minimization problem is applicable more generally that to the Hopfield net formulation described here.

5. Seriously Entangled Neurons

The ability to exploit entanglement computationally motivates defining neurons, each of which has multiple sets of weights. This would enable one such neuron to do the computation of an entire net, for example. Another approach would be to inject multiple sets of entangled inputs. This would enable one neuron to operate on more that one problem instance at a time. Then we could consider neurons with both types of multiplicities simultaneously[5]. We call these seriously entangled neurons. Of course, the versatility gained employing such neurons comes with an increased cost to disentangle the state of the computation, by means of measurement at an appropriate stage (ideally at the end of the algorithm). To facilitate this process, we augment the members of the multiple sets of weights and inputs with labels (quantum decryption keys) that are concatenations of qubits to keep track.

Superposed weights

Suppose there are J weights at each synapse. For convenience, we shall restrict J to take on values that are powers of 2. Let

5.1) $$J = 2^{K-1}, K = 1,2,\cdots.$$

We shall store the weights as an entangled superposition. In particular, at synapse i, we have

5.2) $$W_i^J = \sum_{j=1}^{J} \left|w_i^j\right\rangle R_j^j, i = 1,\cdots,N.$$

Here R_j^j is the quantum decryption key

---

[5] Although we don't consider it here, we could multiplex sets of multiplexed weights, allowing one neuron to work on any number of different problem types simultaneously.

5.3) $$R_J^j = \left|r_1^j\right\rangle \cdots \left|r_J^j\right\rangle, \quad r_\kappa^j \in \{0,1\}, \kappa = 1, \cdots, J$$

with

5.4) $$R_0^j = 1.$$

More generally, we could replace R_J^j by R_J^{ij}, where

5.5) $$R_J^{ij} = \left|r_1^{ij}\right\rangle \cdots \left|r_J^{ij}\right\rangle, r_\kappa^{ij} \in \{0,1\}, \kappa = 1, \cdots, J.$$

Recall that i denotes the synapse and j denotes the weight set label.

Superposed inputs

Now suppose we apply L inputs to each synapse, where

5.6) $$L = 2^{M-1}, M = 1,2, \cdots.$$

The input to a synapse is encoded as an entangled superposition.

5.7) $$A_i^L = \sum_{l=1}^{L} \left|a_i^l\right\rangle S_L^l.$$

Here S_L^l is the key

5.8) $$S_L^l = \left|s_1^l\right\rangle \cdots \left|s_L^l\right\rangle, \quad s_\lambda^l \in \{0,1\}, \lambda = 1, \cdots, L$$

with

5.9) $$S_0^l = 1.$$

More generally, we could replace S_L^l by S_L^{il}, where

5.10) $$S_L^{il} = \left|s_1^{il}\right\rangle \cdots \left|s_L^{il}\right\rangle, s_\lambda^{il} \in \{0.1\}, \lambda = 1, \cdots, L.$$

Total neuronal input with both superposed weights and inputs

The total doubly superposed input is

5.11) $$u = \sum_{i=0}^{N} W_i^J A_i^L$$
$$= \sum_i \sum_{j=1}^{J} \sum_{l=1}^{L} \left|w_i^j\right\rangle R_J^{ij} \left|a_i^l\right\rangle S_L^{il}$$

$$= \sum_i \sum_j \sum_l |w_i^j\rangle |r_1^{ij}\rangle \cdots |r_J^{ij}\rangle |a_i^l\rangle |s_1^{il}\rangle \cdots |s_L^{il}\rangle$$
$$= \sum_{i;a_i^l=0} \sum_j \sum_l |w_i^j\rangle |r_1^{ij}\rangle \cdots |r_J^{ij}\rangle |0\rangle |s_1^{il}\rangle \cdots |s_L^{il}\rangle + \sum_{i;a_i^l=1} \sum_j \sum_l |w_i^j\rangle |r_1^{ij}\rangle \cdots |r_J^{ij}\rangle |1\rangle |s_1^{il}\rangle \cdots |s_L^{il}\rangle$$
$$= \sum_{j,l} \alpha_{JL}^{jl} |0\rangle + \sum_{j,l} \beta_{JL}^{jl} |1\rangle,$$

where (see footnote 2)

$$\alpha_{JL}^{jl} = \sum_{i:a_i^l=0} |w_i^j\rangle R_J^{ij} S_L^{il} = \sum_i \delta_{a_i^l,0} |w_i^j\rangle R_J^{ij} S_L^{il},$$

5.12)

$$\beta_{JL}^{jl} = \sum_{i;a_i^l=1} |w_i^j\rangle R_J^{ij} S_L^{il} = \sum_i \delta_{a_i^l,1} |w_i^j\rangle R_J^{ij} S_L^{il}.$$

References

Bouwmeester, D., Ekert, A., Zeilinger, A. Eds., (2000), *The Physics of Quantum Information,* Springer-Verlag, Berlin.

Deutsch, D., (1985), Proc. R. Soc. London A **400**, 97.

Deutsch, D., (1997), *The Fabric of Reality*, Penguin Putnam, New York.

Deutsch, D., Jozsa, R., (1992), Proc. R. Soc. London A **439**, 553.

Grover, L., (1998), Phys. Rev. Let. **78**, 325.

Haykin, S., (1999), *Neural Networks, A Comprehensive Foundation*, Prentice Hall, Upper Saddle River.

Hertz, J., Krogh, A., Palmer, R., (1991), *Introduction to the Theory of Neural Computation,* Addison Wesley, Redwood City.

Jozsa, R., Sect. 4.2 in Bouwmeester, D., Ekert, A., Zeilinger, A., Eds., (2000).

Shor, P., (1997), SIAM Journal on Computing, **26**.

Steeb, W.-H., (1998), *Hilbert Space, Wavelets, Generalized Functions and Modern Quantum Mechanics*, Kluwer Academic, Dordrecht.