Multiplication with noisy spiking neurons

Panagiotis Nezis and Mark C. W. van Rossum

December 7, 2008

Institute for Adaptive and Neural Computation
School of Informatics
10 Crichton Street
University of Edinburgh
Edinburgh, EH8 9AB
UK

Abstract

Multiplication is an operation which is fundamental in mathematics, but it is also relevant for many sensory computations in the nervous system. Nevertheless, despite a number of suggestions in the literature, it is not known how multiplication is implemented in neural circuitry. We propose a simple feedforward circuit that combines a rate model of neural activity and a realistic input-output relation to accurately and efficiently implement multiplication. A simulation of a network with integrate and fire neurons is used to demonstrate the feasibility of the circuit.

1 Introduction

After addition and subtraction, one of the basic computations in mathematics is multiplication. It is a core ingredient in many fields of mathematics. One example of the use of multiplication in mathematics is the Stone–Weierstrass theorem, which states that any function can be accurately approximated by polynomials. Hence once a multiplication is implemented and powers can be calculated, it becomes possible to approximate any continuous function.
Multiplication is also an essential operation in the nervous system, in particular in sensory systems. Many sensory computations such as motion detection, looming stimulus detection, and auditory processing rely on a multiplication operation. The strongest evidence that multiplication can be done accurately in the nervous system comes from the barn-owl auditory system (Peña and Konishi, 2001; Fischer, Peña, and Konishi, 2007), where it was observed that the sub-threshold responses of neurons accurately code the product of two stimulus parameters (the inter-aural level difference and the inter-aural time difference).

Nevertheless, exactly how the nervous system multiplies signals is unknown. This question is intimately connected to neural coding. For instance, if firing rates do not code the actual signals but the log of the signals, multiplication becomes a trivial addition (see below). As another example, if population codes are used, radial basis functions can be used for computations (Haykin, 1998). Like most other work, we exclusively consider multiplication of firing rates in this study, which is presumably most relevant for early sensory processing. Addition and subtraction of firing rates is easily implemented by synaptic excitation and inhibition, however it is unclear how firing rates are multiplied. Various suggestions have been made (see (Koch and Poggio, 1992) for an overview). For instance, to calculate the product of the rate of two Poisson trains, one can use that the coincidence rate of both processes rate is the product of the rates (Srinivasan and Bernard, 1976). However, this method requires that only a few inputs are simultaneously active within an integration time-constant, which is unrealistic as most neurons receive many inputs. An alternative is to use that \( x \cdot y = \exp(\log x + \log y) \), i.e. multiplication can be done by addition in the log domain. Of course this needs to be followed by an exponentiation. Evidence for this multiplication algorithm has been found in the fly visual system (Gabbiani et al., 2002). A number of researchers have explored the possibility of doing non-linear computation in a single neuron using the interaction of excitatory and inhibitory inputs on the dendritic tree (Koch, 1999), and the voltage dependence of NMDA receptors (Mel, 1992). Finally in population coding networks, recurrent inhibition can be used to multiply inputs that are additive to the single neuron (Salinas and Abbott, 1996).

In this study we propose simple feedforward circuits that multiply two firing rates. The circuits are then implemented and simulated using noisy integrate and fire neurons. Noise integrate-and-fire networks have received recent interest because, despite the presence of
noise, they can accurate transmit and compute with rate-coded information while operating in a regime presumed comparable to cortical networks in vivo (Knight, 1972; Gerstner, 2000; van Rossum, Turrigiano, and Nelson, 2002; Vogels and Abbott, 2005), see also (Yu, Giese, and Poggio, 2002).

2 Approximate multiplication

The central idea of this paper is that the product can be approximated by the minimum function. (The AND-function in the binary domain). This is illustrated in Fig. 1. As in (Peña and Konishi, 2001), the firing rate along the x-axis changes according to a Gaussian curve \( f_A(x) = e^{-x^2} \), while along the y-axis the firing rate is \( f_B(y) = \cos(y) + 1 \). In panel a the exact multiplication of these firing rates is plotted, i.e. \( f_{A \times B}(x, y) = e^{-x^2} \cos(y) + 1 \). In panel b the minimum of these functions is shown. Although there are clear differences between the minimum and the multiplication, the minimum function is a decent initial approximation to the multiplication.

The approximation of the multiplication by the minimum function is useful because the minimum function can be implemented easily in feedforward networks. Fig. 2 depicts two circuits that compute the minimum of two firing rates. The squares represent small populations of neurons, the activity of which is measured by their firing rate. By excitatory and inhibitory synapses, the various populations add or subtract the inputs \( f_A \) and \( f_B \) followed by a rectification. First, we assume that the rectification of the nodes, denoted by \( h(x) \) is simply a threshold linear relation. That is, \( h(x) = [x]_+ \), where \( [x]_+ = \max(x, 0) \). In that case the circuit in Fig. 2a calculates

\[
([f_A + f_B]_+ - [f_A - f_B]_+ - [f_B - f_A]_+)_+ = 2 \min(f_A, f_B)
\]

This is easily checked by assuming \( f_A < f_B \) or, symmetrically, \( f_A > f_B \).

The sharp thresholding used above is only an approximation of a real neuron’s input-output relation. It has been argued using data (Anderson et al., 2000) and models (Miller and Troyer, 2002; Hansel and van Vreeswijk, 2002) that noise smooths the input-output relation so that it approximates a power-law. That is, the input-output relation can be written as

\[
h(x) = ([x]_+)^n
\]
For very high firing rates one expects saturation to become important, however, in vivo firing rates seem mostly far from saturated. Interestingly, when such a power-law transfer function is used for the circuit of Fig. 2 a, the approximation of the multiplication is much better, Fig. 1c.

To measure the accuracy of the circuit we compare its output to a true multiplication. We calculate the root mean square (RMS) error, after scaling the output to have the same peak response. Numerically we find that with \( h(x) = [x]^{1.45} \), the approximation is best for the example inputs of Fig. 1. The average error of the circuit is on average 3.3% of the maximal response, and nowhere more than 13%. If, in contrast, threshold linear units are used, so that the circuit calculates the minimum, the error is 14% on average and maximally 49%.

Intuition for the optimal value for \( n \) can be gained by considering \( f_A = f_B \), in that case only the left center node in Fig. 2a contributes to the output node; the function that the network then calculates becomes \( h(f_A) = (f_A)^{1.9} \), close to the desired output \( (f_A)^2 \). Note, that when the transfer function of the first 3 nodes is \( h(x) = [x]^2 \), and of the final node is \( h(x) = [x]_+ \), the output of the network is exactly a multiplication. However, as this requires different transfer functions for different nodes, this seems less realistic.

The other circuit, Fig. 2b, calculates \( \min(f_A, f_B) \). It uses less nodes, and is the smallest circuit we could find that calculates a minimum. Nevertheless, this circuit approximates the multiplication less well when power-law transfer functions are used than the circuit Fig. 2a, (average RMS error 12%, maximal error 50%), and will not be considered further.

3 Implementation using a spiking network

To examine the multiplication in spiking networks we use small populations of noisy integrate-and-fire neurons connected in a feed-forward fashion with conductance based synapses to implement the circuit of Fig. 2a. Except for the noise, the parameters were as in (van Rossum, Turigiano, and Nelson, 2002): \( R_{in} = 100 \, M\Omega \), \( V_{thr} = -50mV \), \( V_{rest} = -60mV \), \( V_{reset} = V_{rest} \), and \( \tau_{mem} = 20ms \). We used 20 neurons per population; the inputs are also modeled as neural populations, thus the circuit depicted in Fig. 2a contains 120 neurons in total.

Apart from the synaptic input, the neurons are injected with a noisy bias current. As
a result the input-output relation is smoothed and synchronization between neurons is suppressed. With this noise, information coded in firing rates is transmitted rapidly and with little distortion (Knight, 1972; Gerstner, 2000). We used Gaussian noise with a mean of 30pA and a standard deviation of 100pA, filtered with a 2ms timeconstant (compared to a mean of 55pA, and 70pA SD in an earlier study (van Rossum, Turrigiano, and Nelson, 2002)). We reduced the bias current a bit to reduce spontaneous background rates and slightly improve the approximation of the multiplication.

Synaptic input is provided by the neurons in the previous layer via either excitatory and/or inhibitory synapses. For simplicity we assume that input populations can make both excitatory and inhibitory synapses, rather than creating excitatory and inhibitory input populations separately. AMPA-type excitatory synapses were implemented with a reversal potential of 0mV, and inhibition is implemented through GABA_A type synapses with a reversal potential equal to the resting potential. We assume the same time-constant for both excitation and inhibition, $\tau_{syn} = 5ms$. In contrast to the excitatory synapses, the reversal potential of inhibitory synapses is close to the mean membrane potential. To obtain excitatory and inhibitory synaptic currents of similar magnitude, the inhibitory conductances were set seven times larger than the excitatory ones. With this adjustment, the firing rate of a neuron receiving excitatory input at a rate $f_e$ and inhibitory input at a rate $f_i$, is approximately $h(f_e - f_i)$. In other words, the inhibitory is subtracted from the excitation before the non-linearity converts the input current to a firing rate, cf. (Holt and Koch, 1997).

4 Performance of the network

In Fig. 3A the spike trains of the inputs and the output of the network are shown. As an illustration, we injected one input population with a staircase patterned current, while the other input received a sawtooth shaped current. As expected from the injected noise, both input and output spike trains are quite noisy and asynchronous. The average response over 20 trials is shown in Fig. 3B (dots), it closely matches the true scaled multiplication (solid line). The latency in the network (not shown) is about 10ms, twice the synaptic timeconstant, as expected from earlier studies (Knight, 1972; Treves, 1993; Gerstner, 2000; van Rossum, Turrigiano, and Nelson, 2002).
Both systematic and random errors contribute to the difference between the network's output and a true multiplication. The random errors are caused by the randomness of the activity and disappear after sufficient averaging, while the systematic error is due to the network not quite implementing a multiplication. The remaining error is 5.9% RMS of the maximal response. This is close to the error of the circuit of Fig. 2b, which gives an average error of 4.8% RMS using this choice of inputs. The systematic error is plotted in Fig. 3C as a function of the parameters of the noise current: mean and standard deviation.

Both components of the bias current are important to make the multiplication as accurate as possible, while without any bias current the error is much higher. The mean current ensures that small inputs lead to some response and are not thresholded. The noise current (measured through the standard deviation) prevents synchronization of the neurons(van Roesum, Turrigiano, and Nelson, 2002). It should be noted that even without noise, but an appropriately tuned mean current, the performance is reasonable, Fig. 3C. The reason is that the network is not very deep and therefore synchronization of the firing is limited. Finally, the minimum in Fig. 3C is quite broad, i.e. for an approximate multiplication the precise values of the bias current parameters are not very critical.

5 Discussion

We explored how circuits of spiking neurons could implement a multiplication of two firing rates. We found that a simple feed-forward circuit can approximate a multiplication accurately, assuming a powerlaw neural transfer function. If the transfer function of the neurons would be threshold-linear, the circuit would implement a minimum function, but the smooth powerlaw transfer function typically observed in cortical neurons, causes the network to accurately approximate a multiplication.

A critical ingredient is the noise of the neurons which is essential to de-synchronize the neurons and smooth the transfer function. The origin of the noise could be many-fold: intrinsic noise, unspecific other inputs, or random fluctuations resulting from a chaotic network state. There has been a recent interest in networks of noisy spiking neurons, as they are assumed to be realistic models of cortical networks. These networks have been shown to have small latencies, realistic statistics (van Roesum, Turrigiano, and Nelson, 2002) and can calculate various binary functions (Vogels and Abbott, 2005). The current
work suggest that they are also useful for analog computations.

The model makes the following predictions: when inhibition is blocked the output of the circuit distorts in a predictable manner; the firing rates increase, and the multiplication becomes less accurate. When inhibition would be completely blocked, the network calculates an exponentiated sum of the firing rates \((f_A + f_B)^2\). Furthermore, the proposed network will also become less accurate when noise or bias current are changed (Fig. 3c), but these parameters are hard to manipulate experimentally.

Our result is not only relevant when sensory firing rate signals need to be multiplied. Arbitrary computations on population coded information can be done using radial basis functions (Haykin, 1998). Radial basis functions are often constructed using a multiplication operation (although the multiplication need not be very exact). A network similar to the one presented here, has been used to implement such computations (van Rossum and Renart, 2004).

The current study can be viewed from a broader perspective. The classical results by McCullough and Pitts demonstrated that a network of simple binary threshold units can implement any binary computation (McCullogh and Pitts, 1943). Similarly, using multilayer perceptrons any continuous function can be approximated (Haykin, 1998). Indeed, in barn-owl auditory processing the accurate multiplicative responses arise from summation of more diverse responses (Fischer, Peña, and Konishi, 2007). In light of this, the fact that a multiplication can be done at all, should not be surprising. However, the question remains if the proposed mechanism is biologically plausible and efficient. We believe it is.

Acknowledgements

We thanks Matthias Hennig and Alfonso Renart for discussion.

References


Figure 1: Multiplication of two firing rates. Along one axis the firing rate is defined as $e^{-x^2}$ with $x = (-3, 3)$, the other as $\cos(y) + 1$ with $y = (-5, 5)$.

a) Mathematical exact multiplication of these two functions.

b) The minimum of the two inputs approximates the multiplication, although not very accurately. The minimum is equivalent to the two layer network with rectifying neurons, with a transfer function $h(x) = [x]_+$.

c) Approximation when the nodes have a supra-linear input-output relation, $h(x) = ([x]_+)^{1.45}$. 
Figure 2: Two neural circuits for the implementation of approximate multiplication. The sharp arrows depict excitatory inputs, the stump arrows depict inhibitory inputs. The squares correspond to populations of neurons which sum and rectify the net input.
Figure 3: Implementation of the multiplication network using integrate-and-fire neurons.

a) Spike responses in the input and output populations.

b) Firing rates corresponding to panel a, averaged over 20 runs. The response in the output layer (dots, top graph) matches the true multiplication of the rates (solid line) (100ms bins).

c) The error of the network as a function of the noise parameters, revealing a broad range of parameters for which the network approximates a multiplication. The error was the mean RMS error relative to the maximum response calculated using the input and output firing rates.