Computation with populations codes in layered networks of integrate-and-fire neurons

Mark C. W. van Rossum, A. Renart

ANC, University of Edinburgh, 5 Forrest Hill, Edinburgh EH1 2QL, UK

Brandeis University, Department of Biology, 415 South Street, Waltham, MA 02454-9110

Abstract

We discuss feed-forward architectures that compute with population codes. Using radial basis functions we implement a layered network of noisy integrate-and-fire neurons which computes the sum of two population coded quantities. The network performs the computation robustly, accurately and quickly.

Key words: Population codes, integrate-and-fire neurons, multiplication, radial basis functions

In many systems of the brain, information is carried not by single neurons but by population codes. In population codes neurons have wide, overlapping tuning curves such that a single stimulus causes a response of a large number of neurons. Population codes in the brain are for example found in the direction selectivity of V1 neurons, the coding of motor commands in the motor cortex, and hippocampal place cells [1,2,3].

Email address: mvanross@inf.ed.ac.uk, arenart@brandeis.edu (A. Renart).

Many theoretical studies have been undertaken to study the properties of population codes. Computations with population codes can be performed using radial basis functions [4,5]. Here we show that a feed-forward network of integrate-and-fire neurons can implement such computations. Furthermore, despite the presence of noise in each neuron, the signal quality is to a high level preserved.

Model

The basis for this study is the layered architecture introduced in [6]. The network consists of layers of integrate-and-fire neurons connected in a feed-forward fashion. It was shown that firing rates can propagate rapidly and accurately through many layers, provided that a noisy bias current is injected to all neurons in the network. The net component of the current brings the neurons close to threshold, while the noise prevents synchronization of the activity (the synchronization is known as syn-fire behavior, see other papers in this volume).

In our original study the connectivity was (mostly) uniform and all-to-all, so that any population coded llinformation was lost. This begged the questions 1) whether population codes could also be rapidly and accurately be transmitted through these networks and 2) how computations are implemented in such networks.

By connecting the layers through a center-surround connectivity, the loss of information in the population code can be remarkably small and the population code can be transmitted through many layers. The conditions under which

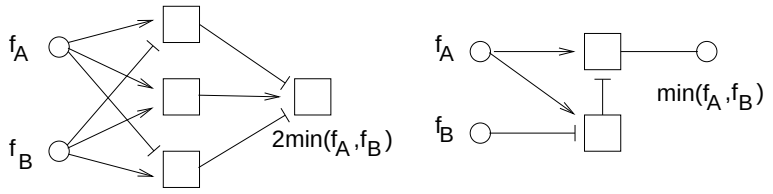

Figure 1. Two examples of architectures for linear rectifying neurons that calculate the minimum of two firing rates, $min(f_A, f_B)$. The square denotes half-wave rectifying units. The populations of integrate-and-fire neurons in our network can be accurately modeled as half-wave rectifiers. The sharp arrows denote excitatory input, the blunt arrows inhibitory input. Although both networks calculate the minimum function, they have each particular advantages: The left one is symmetric in its inputs (important for rapid dynamic inputs); the right one requires less units.

this happens are that, the connectivity profile is about as wide as the activity profile and some, but not too much, inhibition is present. This is further detailed in [7]. The other question, how networks can implement computations, is discussed here.

Computation

We show how computations such as coordinate transforms can be done in integrate-and-fire networks. In general computation on population coded quantities can be done using radial basis functions [4]. Suppose one wants to calculate a function $f(x, y)$ of the variables x and y (x and y might for instance represent the head direction and the eye direction. A typical transformation is to calculate the sum of these angles. When quantities are populated coded, already a simple sum is actually a non-linear operation. Using radial basis functions such a sum can be calculated as follows: One first creates a 2-dimensional layer with an activity equal to the (outer) product of the population activity

in x and y. Next, a projection from this layer to an output layer implements the output function $z = f(x, y)$. Under the quite general conditions this allows for the calculation of arbitrary functions.

The first problem we encounter using radial basis functions is that we need to implement a multiplication in our network. We do not use any detailed biophysical mechanism to do a multiplication, but use the fact that the neurons (or small populations of neurons) approximately act as linear rectifiers, $out = \max(0, in) = [in]_+$. This can be used to implement a minimum operation, as is shown in Fig. 1. The minimum operation is a sufficient approximation of a multiplication for our purposes. Furthermore, when the neurons are slightly supra-linear, the shown circuits approach a multiplication operation more precisely.

Based on the left network in Fig. 1 we implement a computing network. There are two one-dimensional input layers which carry the population coded inputs (30 neurons each). Four two-dimensional layers are used to calculate the product of the activities (900 neurons in each layer). The output layer is again one-dimensional. All neurons received a noisy bias current.

When a layered network has to transmit population codes, the information is best maintained when center-surround profiles are used to connect the layers [6,7]. Here the connections between the layers are center-surround profiles for excitatory connections, and purely center for inhibitory connections. The synapses are conductance based. Connection strengths are tuned such that activity levels in all layers are comparable (a peak firing rate of some 50 Hz).

In Fig. 2 we show the activity in our network. We have connected two one-dimensional population coding layers to three intermediate two-dimensional

layers. These intermediate layers implement the multiplication. The layer shown on top calculates $[f_A(x) - f_B(y)]_+$, the middle layer simply adds $f_A(x) + f_B(y)$, and the layer shown at the bottom calculates $[-f_A(x) + f_B(y)]_+$. The activity of the next two-dimensional layer combines the three previous layers and has an activity $f(x,y) = [-[f_A(x) - f_B(y)]_+ - [-f_A(x) + f_B(y)]_+ + f_A(x) + f_B(y)]_+ = min(f_A(x), f_B(y)) \approx f_A(x).f_B(y)$. The activity of this layer is projected onto the diagonal and forms the input to the final one-dimensional layer. The input to unit k in the output layer is

$$input_k = \sum_{i,j=1}^{N} f(i,j) \text{ with } mod(i+j, N) = k$$

where N is the network size. The output carries a population code for the sum of A and B. Because of the periodic boundary conditions we imposed, the addition is modulo the network size. That is, the firing rate f_C encodes $C = mod(A + B, N)$.

Simulation results

The activity shown in Fig. 2 is produced by using static stimuli and averaging over a long time. To demonstrate the accuracy and speed of our network, we used two dynamic inputs. The first input jumps every 500 ms, the second input moves continuously, Fig. 3. Every 25 ms the position coded in the output layer is estimated (circles). The solid line is the expected result. The network performs the task as expected. The latency is small (on the order of the synaptic time-constant, 5 ms in our network).

The output of the network fluctuates around the desired output. These fluctu-

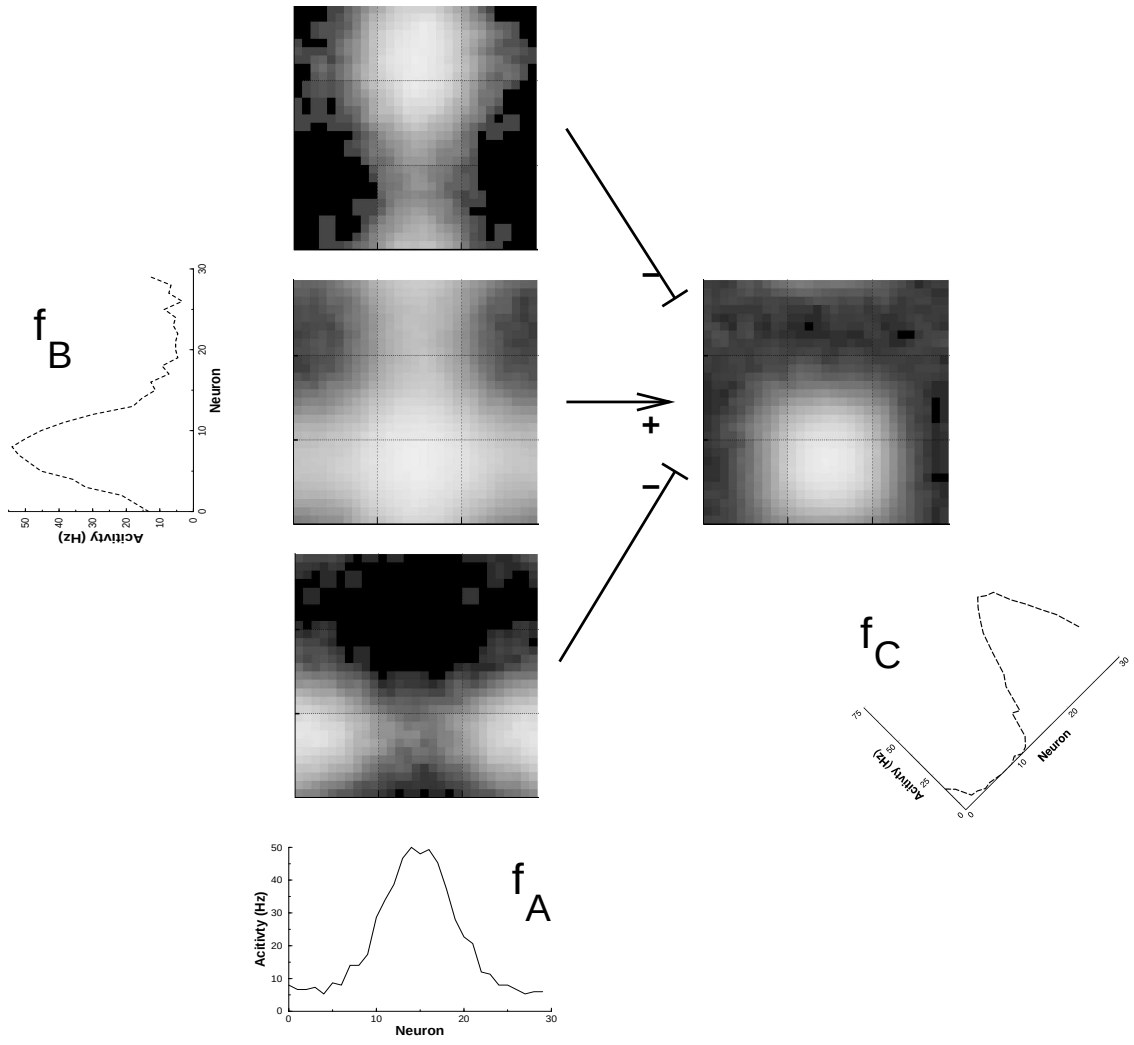

Figure 2. The activity in a network that calculates the sum of its two inputs using radial basis functions. Two quantities, let's say angles, A and B are encoded in a population code by the firing rates f_A and f_B. Radial basis function require a two-dimensional activity with activity $f_A(x).f_B(y)$. This product is approximated by the network in 2 steps, compare Fig.1 left. Lighter grays represent higher firing rates. The final step projects the activity onto the diagonal. This activity codes $A + B$ in a firing rate f_C.

ations are caused by the discrete, random nature of the spikes in the network. Importantly, the fluctuations are not much larger in the output layer than they are in the input layer, indicating that the computation does not add

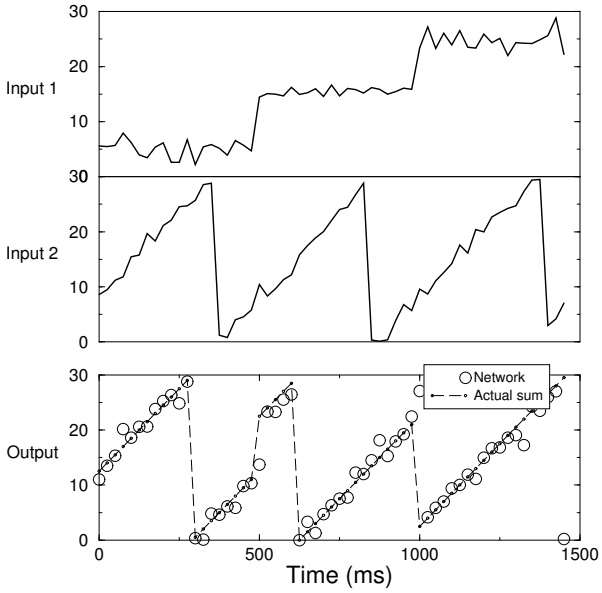

Figure 3. The network of integrate-and-fire neurons calculates the sum of the two inputs (modulo the size of the network). The upper two panels are the inputs to the network. The inputs appear noisy because the information was estimated from the actual spikes in the input layers. The lower panel shows the actual output (circles) and the expected output given by the sum modulo N (line). The output approximates the desired output. The output is of course noisy, but the noise is not much worse than in the input. The network had 30 units, while the time bins were only 25 ms. Longer integration times or more units improve the accuracy.

much noise. The fluctuations could be reduced by either increasing the integration time bin (not always an option in biology when fast reaction times are required), or by increasing the maximal firing rate, or by increasing the number neurons in the network.

Discussion

Earlier we have shown that layered networks of integrate-and-fire neurons can accurately and rapidly transmit rate coded information, provided a noisy bias

current is present in all neurons [6]. This paper shows that computations based on population codes can be implemented in a feed-forward network of noisy integrate-and-fire neurons. In contrast to other studies, it does not rely on recurrent connections [5]. The contribution of this study are that: 1) It shows that radial basis function computation can be implemented in these networks. 2) It shows that the product operation required for radial basis functions can be easily implemented with noisy integrate-and-fire neurons. 3) The loss of information caused by the noisiness of the neurons is small. 4) The latency of the computation is small.

The network computation is robust, as it actually uses the noise in the neurons. The multiplication is also robust as it does not depend on sensitive biophysical mechanisms. Furthermore, population codes are inherently robust against neuron failure.

In this paper we have considered rather simplistic connectivities. Currently we are researching what the optimal circuitry is for doing these computations. A straight forward method would be to do a numerical optimization of the connections between the neurons, but as this is rather intensive numerically, we hope that a more fundamental approach can yield insight.

References

[1] C. Lee, W. H. Rohrer, D. L. Sparks, Population coding of saccadic eye movements by neurons in the superior colliculus, Nature 332 (1988) 357–360.

[2] M. A. Paradiso, A theory for the use of visual orientation information which exploits the columnar structure of striate cortex, Biol. Cybern. 58 (1988) 35–49.

[3] P. S. Churchland, T. J. Sejnowksi, The Computational Brain, MIT Press, 1994.

[4] A. Pouget, T. J. Sejnowki, Spatial transformations in the parietal cortex using basis functions, J. Cogn. Neurosci. 9 (1997) 222–237.

[5] S. Deneve, P. E. Latham, A. Pouget, Efficient computation and cue integration with noisy population codes, Nature Neurosci. 4 (2001) 826–831.

[6] M. C. W. van Rossum, G. G. Turrigiano, S. B. Nelson, Fast propagation of firing rates through layered networks of noisy neurons, J. Neurosci. 22 (2002) 1956–1966.

[7] M. C. W. van Rossum, A. Renart, Optimal transmission of population codes with center-surround architectures, in prepatation.