EMERGENT ACTIVATION FUNCTIONS FROM A STOCHASTIC BIT-STREAM NEURON

M. van Daalen, T. Kosel[†], P. Jeavons, and J. Shawe-Taylor
Connection Science and Machine Learning Group,
Royal Holloway, University of London, Egham, Surrey, UK
[†] Visiting from the University of Ljubljana, Slovenia

October 25, 1995

Abstract

In this paper we present the results of experimental work that demonstrates the generation of linear and sigmoid activation functions in a digital stochastic bit-stream neuron. These activation functions are generated by a stochastic process and require no additional hardware, allowing the design of an individual neuron to be extremely compact.

Introduction

An artificial neuron is required to calculate a single output value by applying an 'activation function' to a weighted sum of its inputs. Such neurons are intended to operate in massively parallel networks, so any electronic implementation of a neuron must provide this functionality as efficiently as possible.

One promising approach is to represent and process real valued signals using digital bit serial stochastic techniques [1, 2]. For example, the product of two stochastic bit streams that have been modulated to carry real values in the interval $[-1, 1]$ may be found by calculating their bit-wise Exclusive OR. By using stochastic bit streams, and simple logic to process them, significant hardware savings can be achieved over other approaches to neural networks [3, 4].

In order to exploit these advantages, we have developed a complete neural network design using fully digital circuitry and stochastic bit streams [3, 5]. One remarkable feature of this design is the fact that the activation function of the neuron is not built into the hardware explicitly but is generated solely by the interaction of two probability distributions.

By appropriately controlling the distribution of the threshold value supplied to each neuron, it is possible to achieve a variety of activation functions [5]. This paper describes the features of our digital stochastic neuron which give rise to these emergent activation functions, and then compares the theoretical predictions with the experimental results obtained from actual hardware designed to investigate this novel property.

The Stochastic Neuron

The circuitry required to construct a single stochastic neuron, is shown in Figure 1. A detailed description of network architectures, and surrounding support circuitry is given in [3].

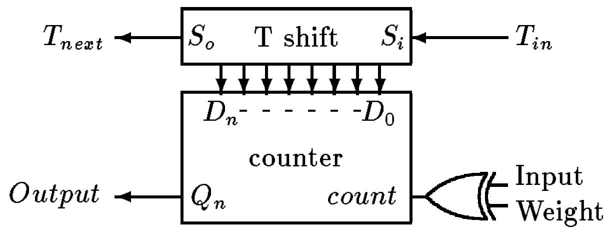

Figure 1: The elements of a single stochastic neuron

The neuron has only one physical input and weight connection, but by using time multiplexing, may have many logical connections. The core of the neuron is a k bit counter, which may be preloaded with a threshold value. The counter only increments when the *count* input is active, and this is directly controlled by the weighted input signals, as produced by the XOR gate.

Each weighted input value therefore contributes 0 or 1 to the counter, on each operational cycle. Hence, the total net input contribution to the counter, after summing the contributions due to all of the inputs to the neuron, will be an integer between 0 and the number of inputs, n, and the probability distribution of this value will depend on the weighted input values represented by the input bit-streams.

Unique threshold values are supplied to the neurons by a long shift register, with each individual neuron tapping into its own local section. The threshold value is chosen such that it will cause an overflow into the top most counter bit, when a given input count, t, is achieved or exceeded. Thus the output of the neuron is taken from the most significant bit of the counter.

Varying the distribution of the threshold value alters the probability, p_{out} that this output bit will be set, and hence alters the effective activation function computed by the neuron. For example, if the weighted input values are all equal to p_{in} and the threshold value is fixed, then

$$p_{\text{out}} = \sum_{i=t}^{n} \begin{pmatrix} n \\ t \end{pmatrix} p_{\text{in}}^{i}(1-p_{\text{in}})^{n-i} \tag{1}$$

which is a sigmoid function of p. Alternatively, if the threshold value is chosen uniformly at random over all suitable values, then we have

$$\begin{aligned} p_{\text{out}} &= \frac{1}{n}\sum_{t=1}^{n}\sum_{i=t}^{n} \begin{pmatrix} n \\ t \end{pmatrix} p_{\text{in}}^{i}(1-p_{\text{in}})^{n-i} \\ &= \frac{1}{n}\sum_{i=0}^{n} i \begin{pmatrix} n \\ t \end{pmatrix} p_{\text{in}}^{i}(1-p_{\text{in}})^{n-i} \\ &= p_{\text{in}} \end{aligned} \tag{2}$$

which corresponds to a linear activation function.

Whatever activation function is selected, the network may be trained using a modified version of the back-propagation algorithm [5].

The hardware

The hardware used to obtain the results presented in this paper, consisted of a single neuron with 5 logical inputs. These inputs were arranged to receive the same actual value, in the range [0,1], but represented as distinct stochastic bit streams.

The input bit streams were generated by 5 7-stage bit stream modulators, of the kind described in [2]. These modulators required 35 independent pseudorandom bit streams with bit probability of one half, generated by tapping a single 39 stage linear feedback shift register. The input bit streams were multiplexed onto the single physical input connection to the neuron, by a 5 stage loadable shift register.

The neuron itself consisted of a single 4 bit counter. (The XOR weight multiplier was omitted as the weights in this experiment were fixed at 1.) The output of the neuron was taken from the top bit of this counter. On each input cycle this bit was reset, and the remaining bits were initialised to a 3 bit threshold value. Finally, the output bits from the neuron were integrated by a 20 bit counter, allowing output streams of up to 1M bits to be collected.

The complete design was implemented using an FPGA operating at an effective bit rate of 27Mhz. The FPGA was interfaced to a PC slot that provided the 7 bit value used to control the input bit stream modulators, and the 3 bit threshold value, and gave access to the 20 bit result. Software written in Pascal controlled the neuron whilst appropriately varying the input and threshold values, allowing a family of activation functions to be displayed in real time.

Results

Four graphs showing sigmoid and linear activation functions obtained from this hardware are presented in Figure 2. Each function is displayed twice, as they were calculated using bit streams of length 5K bits, and then of length 100K bits.

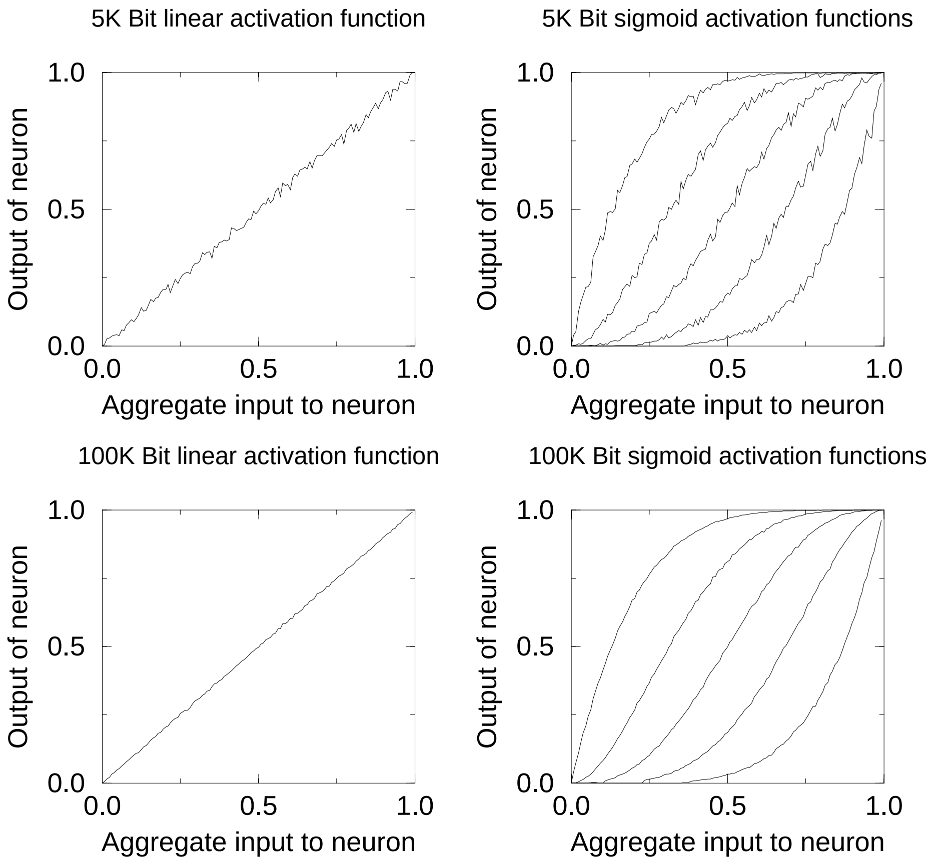

Figure 2: Sigmoid and linear activation functions

The sigmoid curves were obtained by setting a fixed threshold value in the range 3 to 7, corresponding to t values from 1 to 5. The central, symmetrical, sigmoid curve was obtained by setting a threshold value of 5 ($t = 3$). The other sigmoid curves illustrate how the activation function of the neuron varies with different values of t (Equation 1).

The linear activation functions were obtained by choosing the threshold value uniformly at random, between 3 and 7, on each operational cycle (Equation 2).

The standard deviations of the experimental results compared with the theoretical predictions for a typical experiment with a linear activation function and the central sigmoid activation function are shown in Table 1.

	5K bits	100K bits
Sigmoid	0.0109	0.00250
Linear	0.0119	0.00255

Table 1: Standard deviation from theoretical curves

Conclusion

In this paper we have demonstrated that it is possible to construct stochastic bit stream neurons with minimal digital circuitry. We have shown that such a neuron produces an emergent activation function without any additional hardware, of the predicted mathematical form.

In most applications of these neurons, it is envisaged that a sigmoid activation function would be the most useful, as this corresponds most closely to the standard neural network model. It is notable that this is also the easiest form of activation function to obtain from our design, since it results from a fixed threshold value. However, we anticipate that the capacity to obtain a linear activation function from the same hardware may also be useful, for example to re-randomise bit streams within bit stream based Hopfield networks or Boltzmann machines.

References

[1] B. R. Gaines, “Stochastic Computing Systems”, Advances in Information Systems Science, 2 1969, pp. 37–172.

[2] M. van Daalen, P. Jeavons, J. Shawe-Taylor and D. Cohen, “Device for generating binary sequences for stochastic computing”, Electronics Letters, Vol 29, No 1, Jan 1993, pp. 80–81.

[3] M. van Daalen, P. Jeavons and J. Shawe-Taylor, “A stochastic neural architecture that exploits dynamically reconfigurable FPGAs”, IEEE Workshop on FPGAs for Custom Computing Machines, April 5-7, 1993, Napa CA, pp. 202–211.

[4] M. Stanford Tomlinson Jr, D. J. Walker, M. A Silvilotti, “A Digital Neural Network Architecture for VLSI”, *IJCNN*, 1990, San Diego, II pp. 545–550.

[5] J. Shawe-Taylor, P. Jeavons and M. van Daalen, “Probabilistic Bit Stream Neural Chip : Theory”, Connection Science, Vol 3, No 3, 1991, pp. 317–328.