Probability Density Methods for Smooth Function Approximation and Learning in Populations of Tuned Spiking Neurons

Terence David Sanger
Massachusetts Institute of Technology Department of Brain and Cognitive Sciences, Cambridge, MA 02139, and Boston Children's Hospital Department of Neurology, Boston, MA, U.S.A.

This article proposes a new method for interpreting computations performed by populations of spiking neurons. Neural firing is modeled as a rate-modulated random process for which the behavior of a neuron in response to external input can be completely described by its tuning function. I show that under certain conditions, cells with any desired tuning functions can be approximated using only spike coincidence detectors and linear operations on the spike output of existing cells. I show examples of adaptive algorithms based on only spike data that cause the underlying cell-tuning curves to converge according to standard supervised and unsupervised learning algorithms. Unsupervised learning based on principal components analysis leads to independent cell spike trains. These results suggest a duality relationship between the random discrete behavior of spiking cells and the deterministic smooth behavior of their tuning functions. Classical neural network approximation methods and learning algorithms based on continuous variables can thus be implemented within networks of spiking neurons without the need to make numerical estimates of the intermediate cell firing rates.

1 Introduction

As electrophysiological recording data from multiple cortical cells become available, it is increasingly important that neural network models be able to interpret the information and computations performed by the neural code (Sejnowski, 1995). However, an important gap exists between the physiological representation of information and much of the large body of theoretical work on neural networks. Many neural network models are defined in terms of smooth function approximation (Poggio & Girosi, 1990, for example) under the assumption that continuous variables can be represented in the firing rate of cells (Judd & Aihara, 1993). Cell behavior is described by a tuning curve (Snippe, 1996; Rieke, Warland, van Steveninck, & Bialek, 1997, for discussion) that gives the average firing rate as a function of some external variable. It is assumed that once firing rates are known, they can be added, multiplied, and otherwise manipulated according to the rules of continuous mathematics. In fact, this assumption is so basic to such models that it is rarely stated explicitly. Simulated implementations use continuous variables as inputs and hidden units, and compute continuous functions at the outputs without attempting to relate these variables to the actual underlying spike data.

The problem with this approach is that it can be difficult to implement in neural hardware. In order to extract the continuous rate variable from the spike data, we must count the number of spikes that occur within some fixed time interval. Since a rapidly (and regularly) firing cell might fire at most 100 spikes per second, we would need to count over at least 1 second in order to have an estimated error less than 1 percent (although Tovee, Rolls, Treves, & Bellis, 1993, suggest that shorter time intervals may be possible). If the cell's firing is described by a Poisson process with an average rate of 100 spikes per second, then we will need to count over a significantly longer interval in order to make an accurate estimate of the rate (Stein, 1967). In either case, the error in rate estimation means that variables can be only approximately represented or represented with low precision (Softky, 1996). This situation becomes worse for cells with low firing rates or if the average rate is chaotic (van Vreeswijk & Sompolinsky, 1996) or varying with time (August & Levy, 1996; Gabbiani & Koch, 1996). As experimenters, we frequently have the option of observing a biological network over a long period of time under static or repeating conditions, so we can in fact compute accurate average rates. But a behaving animal cannot afford to wait 1 second or longer at each stage of computation.

There are several ways in which a neural implementation of a continuous model might address this problem. One is to assume that there are many copies of each cell, so that we can use spatial averaging or a population code to increase the accuracy of rate estimation for short time windows (Shadlen & Newsome, 1994; Maass, 1995; Shadlen, Britte, Newsome, & Movshon, 1996; Theunissen, Roddey, Stufflebeam, Clague, & Miller, 1996). Although this appears to be wasteful of neural resources, it might be feasible given the large number of cortical cells available. Population averaging may not lead to optimal estimates if there is significant correlation between the output of different cells (Zohary, Shadlen, & Newsome, 1994). Other methods note that under certain circumstances, there is a remarkable predictability of spike firing times (Bair & Koch, 1996), and one can thus use the interspike interval (Judd & Aihara, 1993; Softky, 1994) or the filtered spike train (Bialek & Rieke, 1992; August & Levy, 1996; Gabbiani & Koch, 1996) as an "instantaneous" estimate of the relevant variable. There is considerable theoretical and experimental evidence to support both rate codes (Shadlen & Newsome, 1994, 1995) and temporal codes (Murthy & Fetz, 1994; Hopfield, 1995; Mainen & Sejnowski, 1995; Konig, Engel, & Singer, 1996), and some comparisons are given in Stein (1967), Gerstner and van Hemmen (1994), and Softky (1995, 1996). A detailed review can be found in Rieke et al. (1997).

The underlying problem for rate codes is that although the firing rates are indeed continuous and often deterministic functions of the cell inputs, it is difficult to calculate these rates explicitly from the pattern of spike data. This article proposes a new method based on rate coding, in which smooth network approximations can be represented and manipulated by populations of spiking neurons. Similarly to other techniques, this method assumes that information is represented in the instantaneous average rate (or firing probability), but it avoids the need to compute this rate explicitly. Neural populations map directly onto other neural populations in such a way that the firing rate automatically computes desired functions. The mathematical formulation assumes that spike rate is a measure of the probability of firing and that data are encoded in the instantaneous probability of firing. I show that it is possible to connect a network of randomly spiking neurons such that the firing probability of the output neurons is a desired smooth function of the firing probability of the input neurons.

For example, consider computing the sum of two functions $f(x)$ and $g(x)$, where f and g are the average probability of firing of two different neurons s_f and s_g in response to different stimulus values x. In order to do this computation using explicit firing rates, the network could wait for several seconds and count the input neuron spikes and then use the total spike count to modulate another neuron's rate. Alternatively, many copies of each of the input neurons could be provided, and the total number of spikes over the input population could determine the output neuron's rate. In any finite time period (or finite number of copies of the input neurons), these two methods can only approximate the firing rates $f(x)$ and $g(x)$, and thus the calculation of the output rate $f(x) + g(x)$ will not be exact. I propose instead that if the output neuron is connected so that it fires every time either of the input neurons fires, then the average firing rate of the output neuron is the sum of the input neuron average firing rates (so long as two spikes do not coincide within the refractory period of the output neuron). If the average firing rate of one of the input neurons changes, so does the firing rate of the output neuron. The probability of firing and the output cell tuning curve are proportional to $f(x) + g(x)$ from the instant of stimulus onset, although it may not be possible to measure this until at least one spike has occurred.

The above example illustrates an important and subtle point that is the foundation for the rest of this article: it is possible to compute an output cell's tuning curve deterministically using connections that transmit only the randomly firing spikes. This is because the probability of a spike is a continuous deterministic variable that is only implicitly linked to whether a spike actually occurred. Since the tuning curve is the average firing rate for each value of a measured variable, it will be proportional to the conditional probability of firing given that variable. Each cell has an implicit probability of firing, and computations on the probabilities can be performed without ever needing to estimate these probabilities from the pattern of spike firing. Only at the final output stage of the system, when the cells must actually drive the muscles, is it necessary to use time or space averaging to estimate the firing probabilities from the spike data (Stein, 1967). In practice, if a cell fires rarely, then the implicit probability may be irrelevant since no information is transmitted in the absence of spikes. This would be an even greater problem for a rate-averaging system, since it would take a long time to estimate the average spike rate.

2 Assumptions of the Poisson Spike Model

The mathematical foundations and fundamental assumptions of the computational method proposed here come from the probability density estimation method for interpretation of experimental results from population codes (Sanger, 1996). Probability density estimation was not intended to model the extraction of signals within the brain, but rather to indicate the extent to which information is implicit within the neural code. This implicit information is essential to understanding the method proposed here. Related theoretical results on maximum likelihood (ML) estimation have been proposed (Snippe, 1996; Seung & Sompolinsky, 1993; Paradiso, 1988).

To summarize the results in Sanger (1996), consider a very simple model for neural firing described by

$$P[\text{spike} \mid x] \propto \sigma(x), \tag{2.1}$$

meaning that the probability of firing is proportional to a smooth function of an externally measurable variable x. The function $\sigma(x)$ is referred to as the tuning curve of the cell with respect to the variable x, and it describes the average firing rate for each value of x. If the probability of firing within any time interval of the same length is the same, then firing is governed by a Poisson distribution,

$$P[n \text{ spikes} \mid x] = \frac{\sigma(x)^n e^{-\sigma(x)}}{n!}. \tag{2.2}$$

This model is quite general, since x can be a vector if the cell is dependent on multiple variables, and it can include time if the firing rate is time dependent. The model does not include any representation of refractory periods or the possibility of explicitly coding elements of x using the relative spike times or interspike intervals.

From Bayes' law, for an arbitrary number of spikes n, we have

$$P[x \mid \text{spike}] = \sigma(x)^n e^{-\sigma(x)} P[x]/\mathcal{N}, \tag{2.3}$$

where $P[x]$ is the prior probability distribution of the external variable x (usually controlled by the experimenter) and $\mathcal{N}$ is a normalization constant calculated to make the total probability equal to 1. For $n = 1$, this equation states what we learn about the variable x from a single spike. Under this model, the conditional density $P[x \mid \text{spike}]$ completely describes the behavior of this cell.

The probability density estimation method (Sanger, 1996) makes the assumption that the spike-generating units are independent (see also Maass, 1995). In other words, for any given value of x, whether cell i with tuning curve $\sigma_i(x)$ fires is independent of whether any other cell fired:

$$P[\text{spike}_i \text{ and spike}_j \mid x] = P[\text{spike}_i \mid x]P[\text{spike}_j \mid x], \tag{2.4}$$

for all values of x. If each cell i fires s_i times during some time interval, then we can immediately calculate the conditional density of x given the entire population of cells as (Sanger, 1996)

$$P[x \mid \text{population}] = \frac{1}{\mathcal{N}} P[x] \prod_i e^{-\sigma_i(x)} \sigma_i(x)^{s_i}. \tag{2.5}$$

The conditional density of x is implicit in the population-firing response, and this is the *most* information about x that can be extracted from this population. The importance of this result for the following discussion is that the complete information in the population is encapsulated by a polynomial in the tuning curves σ_i. Other methods for estimating x or the conditional density have been proposed based on linear estimation (Georgopoulos, Kettner, & Schwartz, 1988; Salinas & Abbott, 1994), basis function approximation (Anderson, 1995; Zemel, Dayan, & Pouget, 1998), maximum likelihood (Snippe, 1996; Seung & Sompolinsky, 1993), and iterative network convergence (Pouget & Zhang, 1997; Pouget, Zhang, Deneve, & Latham, 1998).

3 Implicit Computation with Spiking Neurons

Under the models given by equations 2.1 and 2.2, a cell's behavior is completely described by its tuning curve, and the tuning curve can (eventually) be extracted as the long-time average firing rate of the cell for each value of x. Therefore, the long-time spike data and the tuning curve essentially are two different ways of describing the same cell. This is the basis of rate coding. In order to create a cell with any particular behavior, we need only to choose a desired tuning curve and synthesize a cell with this tuning curve using the spike data from existing cells. For example, a motor neuron that controls a muscle intended to fire for movements that extend a particular joint needs to be connected to other joint-related neurons (with a variety of different tuning curves) in such a way that its tuning curve is a monotonically increasing function of joint angle. I now describe how simple neural connectivity can be used to synthesize rate-coded cells with almost arbitrary tuning curves.

Consider a neuron k that is a coincidence detector for neurons i and j and thus has a nonzero probability of firing only if i and j both fire within a small time window Δt. In this situation, we can write

$$E[s_k] = \lambda s_i s_j, \tag{3.1}$$

where $E[s_k]$ is the expected number of output spikes for neuron k within time Δt, s_i and s_j are the number of spikes in Δt for each input neuron, and λ is a scaling factor. Since Δt is small, s_i and s_j will usually be either zero or one, so the probability of firing $E[s_k]$ is usually a boolean function of s_i and s_j.

Assume now that cell k is in fact a true multiplier, so that equation 3.1 also holds for s_i and s_j greater than 1 (more than one spike in Δt). Then the tuning curve for fixed s_i and s_j is given by

$$\sigma_k = \frac{E[s_k]}{\Delta t} = \frac{\lambda s_i s_j}{\Delta t},$$

where σ_k is the instantaneous average firing rate in spikes per second for neuron k. Note that σ_k is a function of the two random variables s_i and s_j, and thus the event of neuron k firing is a Poisson process modulated by a random variable. The expected firing rate given x is

$$E[\sigma_k(x)] = (\lambda/\Delta t)E[s_i s_j \mid x], \tag{3.2}$$

and since $P[s_i s_j \mid x] = P[s_i \mid x]P[s_j \mid x]$ by the independence assumption, we have

$$E[\sigma_k(x)] = (\lambda/\Delta t)E[s_i \mid x]E[s_j \mid x] = \lambda \Delta t \sigma_i(x)\sigma_j(x), \tag{3.3}$$

where $E[s_i \mid x]$ is the expected number of spikes in time Δt given x, and $\sigma_i(x)$ is the average number of spikes per second for each value of x. We need to use the expected value $E[\sigma]$ here since the firing rate σ is now a random variable dependent on s_i and s_j.

The expected tuning curve (and thus the average firing rate) for cell k is proportional to the product of the tuning curves for cells i and j. The Δt term in the product represents the fact that the firing rate is lower if the spike coincidence (or product) must occur within a shorter time window. The continuous product relationship between the tuning curves was created simply by making cell k compute the integer product of the number of input spikes occurring within a small time interval. For sufficiently small Δt, cell k functions as a coincidence detector. There is no need to compute the firing rates or tuning curves of cells i or j explicitly.

Figure 1 shows a simulated example. The tuning curves for neurons A and B are given by shifted raised cosine functions with a nonzero "dc" component. The maximum firing rate is 100 spikes per second, and spikes are generated randomly using Poisson statistics according to equation 2.2. Neuron C is a spike coincidence detector during 10 msec intervals, and the

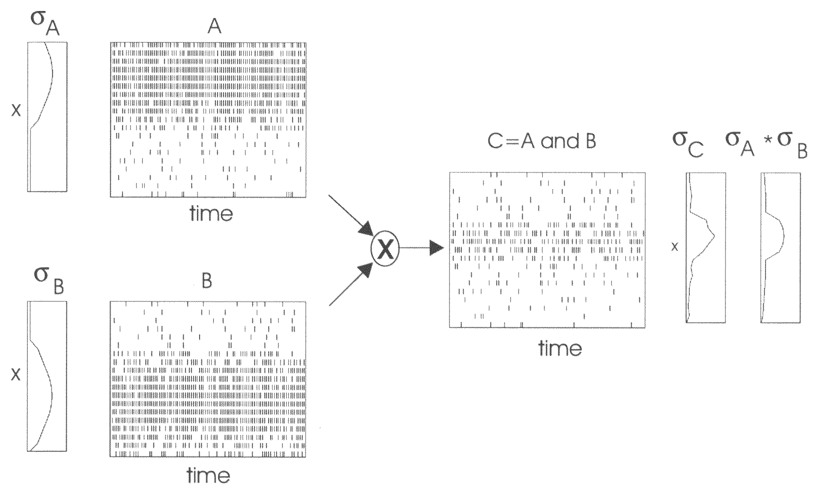

Figure 1: Simulation of tuning curve multiplication. The probability that cell C fires is nonzero only when both cells A and B fire simultaneously. Each row of the simulated spike plots shows the simulated spikes over time for one fixed value of x. $\sigma_A(x)$ and $\sigma_B(x)$ are the tuning curves for cells A and B, and $\sigma_C(x)$ is the tuning curve for C estimated from the spike data. On the right side of the figure, σ_C is compared to the product of σ_A and σ_B.

resulting measured tuning curve σ_C is compared to the product of the input tuning curves $\sigma_A * \sigma_B$.

Suppose that neuron k has nonzero probability of firing λ only if neuron i fires exactly n times during some time interval Δt. Then, from equation 2.2,

$$E[\sigma_k(x)] = \lambda P[i \text{ fires } n \text{ times } \mid x] = \frac{\lambda}{n!}\sigma_i^n \Delta t^n e^{-\sigma_i \Delta t}. \tag{3.4}$$

For fixed and small Δt, this expression becomes proportional to the nth power of σ_i, which shows how an output neuron can approximately compute powers of the input neuron tuning curves.

Now consider a cell whose firing rate is a weighted linear combination of the spike outputs of a set of other cells, so that the expected number of spikes within Δt is

$$E[s_k] = \sum_i w_i s_i, \tag{3.5}$$

and the average firing rate in spikes per second is given by

$$\sigma_k = \sum_i w_i s_i / \Delta t. \tag{3.6}$$

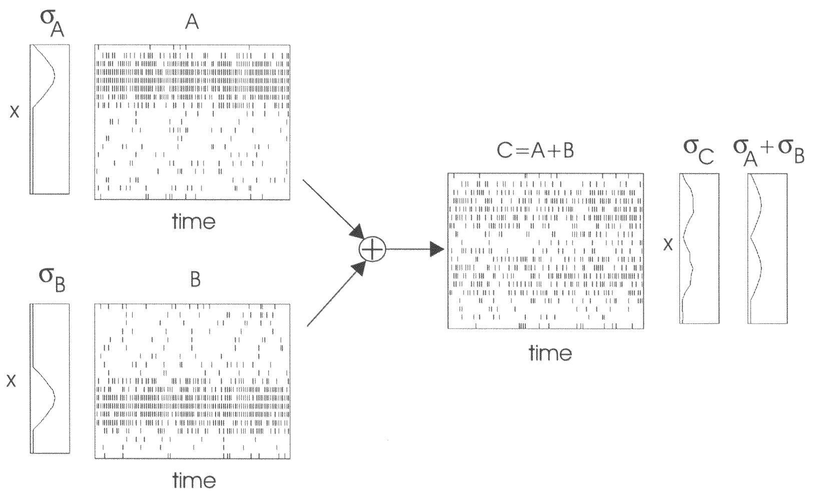

Figure 2: Simulation of tuning curve addition. The probability that cell C fires is proportional to the sum of the total number of spikes of cells A and B in any time interval. On the right side of the figure, $\sigma_C(x)$ is compared to the sum of σ_A and σ_B.

The "synaptic weights" w_i are real valued and indicate the incremental change in average firing rate for neuron k when neuron i fires. Then

$$E[\sigma_k(x)] = \sum_i w_i E[s_i \mid x]/\Delta t = \sum_i w_i \sigma_i(x), \tag{3.7}$$

so the average firing rate is a linear combination of the firing rates of the input neurons $\sigma_i(x)$. Note that the tuning curve σ_k was produced as a linear combination of the σ_i's without ever calculating the actual firing rates of the input neurons. In fact, σ_k itself remains implicit and cannot be extracted without time averaging.

Figure 2 shows a simulated example. Again, the input neurons A and B have tuning curves that are shifted raised cosines, and they fire Poisson-distributed spikes with a maximum average firing rate of 100 spikes per second. Neuron C fires according to Poisson statistics with an instantaneous firing probability proportional to the sum of the number of spikes $s_A + s_B$ within the previous 10 msec window. The resulting sample tuning curve σ_C is compared to the sum of the input tuning curves $\sigma_A + \sigma_B$.

Since we can now calculate linear combinations, powers, and products of input tuning curves, we have the ability to calculate polynomials and thus can approximate any polynomial function of the input probability distributions. (Polynomial computations and multiplicative interactions in neural networks are discussed in Koch & Poggio, 1992; Mel, 1993; Tal & Schwartz, 1997.) The output neurons are Poisson-distributed by assumption, although they are modulated by time-varying rates σ that depend on the particular input spike firing times. Succeeding neural layers can make use of the same type of computation; however, if the input neurons are Poisson, then the output neurons will continue to satisfy an independent interval property (for which the probability of n spikes in any Δt is independent of the number of spikes in any other time interval) and thus they will be Poisson distributed at all time scales.

It is important to note some interesting properties of this form of implicit computation. The actual spike rates do not need to be calculated until they are used at the output stage to control muscles. The probability values propagate implicitly. Although spikes are random binary events, the underlying probabilities are continuous variables that can be computed exactly. The computational delay is equal to the synaptic delay; an output cell has the correct probability of firing as soon as an input cell spike could have affected it, even if the input cell has not yet actually fired. This is often a moot point, since the cell cannot communicate information until it fires. But over many trials, it would be predicted to be tuned immediately after stimulus onset. Time or population averaging may be needed to read out the information or to control muscles, but it is not needed for intermediate computations. An alternative model that has very short synaptic delays and is based on relative spike timing can be found in Maass (1994, 1996, 1997).

4 Learning Algorithms

In the previous section, I showed how it is possible to perform smooth function approximation on the implicit firing probability of cells. The question naturally arises as to whether desired functions can be learned from examples. In this section, I show that it is possible to find spike-equivalent forms of common neural network algorithms such that the underlying implicit probabilities converge smoothly according to the network algorithms, while the synaptic weights change in response to cell spiking. The spike algorithms are closely related to neural network models of stochastic convergence, since it turns out that the expected change in weights from the spike-based algorithms is approximately equal to the weight change for the smooth, implicit algorithms. There are many examples in biology of adaptation in response to spike coincidence, although the relationship to current neural network models can be difficult to elucidate (Magee & Johnston, 1997; Markram, Lubke, Frotscher, & Sakmann, 1997).

First, I show an example of a supervised learning algorithm whose average behavior approximates the Widrow-Hoff LMS algorithm (Widrow & Hoff, 1960; Widrow, McCool, Larimore, & Johnson, 1976). Let p be a "teaching" neuron with tuning curve $\sigma_p(x)$, and let k be a neuron that we would like to train so that $E[\sigma_k(x)] = \sigma_p(x)$. The firing rate of k is a linear combination of the spikes of other cells i according to equation 3.6. The training algorithm has access only to the spike data, and it must adapt the synaptic weights w_i from the set of input neurons i to the output neuron k. Let s_p, s_k, and s_i be the number of spikes in cells p, k, or i during time Δt, and let γ be the learning rate. Consider the adaptation algorithm

$$\Delta w_i = \gamma (s_p - s_k) s_i. \tag{4.1}$$

The expected change in weight during time Δt is

$$\begin{aligned} E[\Delta w_i \mid x] &= \gamma E[(s_p - s_k)s_i \mid x] && (4.2)\\ &= \gamma (E[s_p s_i \mid x] - E[s_k s_i \mid x]) && (4.3)\\ &= \gamma (E[s_p \mid x] E[s_i \mid x] - E[s_k s_i \mid x]), && (4.4) \end{aligned}$$

since neurons p and i have independent firing by assumption. Neurons k and i are not independent, since the firing rate of k is directly modulated by whether neuron i fired. We have

$$\begin{aligned} E[s_k s_i \mid x] &= E[s_k \mid s_i, x] E[s_i \mid x] && (4.5)\\ &= \left(w_i + \sum_{j \neq i} w_j E[s_j \mid x] \right) E[s_i \mid x] && (4.6)\\ &= \left(w_i (1 - E[s_i \mid x]) + \sum_j w_j E[s_j \mid x] \right) E[s_i \mid x] && (4.7)\\ &= (w_i (1 - E[s_i \mid x]) + E[s_k \mid x]) E[s_i \mid x], && (4.8) \end{aligned}$$

and combining gives

$$\begin{aligned} E[\Delta w_i \mid x] &= \gamma (E[s_p \mid x] - E[s_k \mid x]) E[s_i \mid x] \\ &\quad - \gamma w_i (1 - E[s_i \mid x]) E[s_i \mid x] && (4.9)\\ &= \gamma (\Delta t)^2 \Bigg((\sigma_p(x) - \sigma_k(x)) \sigma_i(x) \\ &\qquad - w_i \left(\frac{1}{\Delta t} - \sigma_i(x) \right) \sigma_i(x) \Bigg). && (4.10) \end{aligned}$$

The first term on the right of equation 4.9 is the Widrow-Hoff supervised linear learning rule. The last term is always opposite in sign to w_i and gives a decay term for w_i that is most significant if the average firing rate for neuron i is close to $1/2\Delta t$. The factor of $(\Delta t)^2$ shows that the learning rate decreases if spike coincidence must occur in a shorter time window.

An illustrative example of convergence of the tuning curve when synapses are trained according to equation 4.1 is shown in Figure 3. Twenty input cells

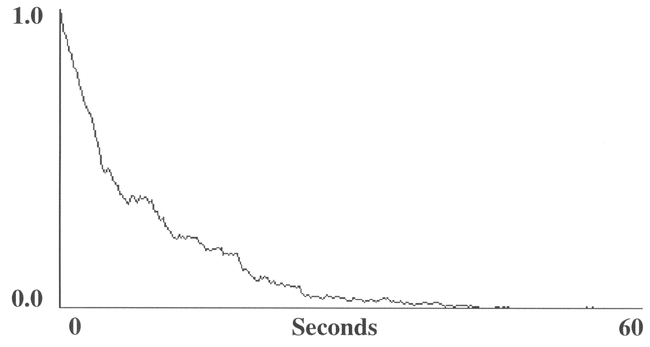

Figure 3: The normalized mean squared error as a function of time for a neuron trained with the supervised spike learning algorithm. There were 20 input cells with a maximum spike rate of 50 / second and randomly chosen smooth tuning curves. The target output cell tuning curve was also chosen randomly. The learning rate is 0.001, and spike coincidence was calculated within 20 msec time windows.

and a target output cell were simulated with randomly generated smooth tuning curves and a maximum firing rate of 50 spikes per second. Learning occurred in 20 millisecond time bins ($\Delta t = 0.02$ second), and convergence is shown for a total of 3000 trials (60 seconds).

I now show an example of an unsupervised Hebbian learning algorithm that causes the tuning curves to converge to the eigenfunctions of the input probability density. Previous examples of Hebbian learning for spike data are found in Brown, Zador, Mainen, & Claiborne (1992) and Gerstner, Kempter, van Hemmen, and Wagner (1996), and the relationship between principal components analysis and information maximization for spike trains is elegantly derived in Fry (1995).

Assume there are multiple outputs σ_k for many values of k. Let

$$\sigma_k = \frac{1}{\Delta t} \sum_i w_{ki} s_i,$$

and consider the learning algorithm,

$$\Delta w_{ki} = \gamma \left(s_i - \sum_{j \le k} w_{ji} s_j \right) s_k. \tag{4.11}$$

Here s_j is an output neuron. Assume the spike generators for $s_j, j \neq k$ are almost independent of s_k in the sense that $P[s_j s_k \mid x] \approx P[s_j \mid x] P[s_k \mid x]$. (This is not strictly true, since s_j and s_k may have common inputs, but the inaccuracy will be small if the number of inputs is large.) Then we can compute the average weight update during Δt as

$$E[\Delta w_{ki} \mid x] = \gamma E\left[\left(s_i - \sum_{j\le k} w_{ji}s_j\right) s_k \mid x\right] \tag{4.12}$$

$$\begin{aligned} &= \gamma\left(E[s_i \mid x] - \sum_{j\le k} w_{ji}E[s_j \mid x]\right) E[s_k \mid x] \\ &\quad + \gamma w_{ki}(1 - E[s_i \mid x])E[s_i \mid x] \end{aligned} \tag{4.13}$$

$$\begin{aligned} &= \gamma(\Delta t)^2\left(\sigma_i(x) - \sum_{j\le k} w_{ji}\sigma_j(x)\right)\sigma_k(x) \\ &\quad + \gamma(\Delta t)^2 w_{ki}\left(\frac{1}{\Delta t} - \sigma_i(x)\right)\sigma_i(x). \end{aligned} \tag{4.14}$$

It should be noted that the first term of equation 4.14 describes the generalized Hebbian learning algorithm that finds the principal components of the input distribution (Sanger, 1989). The second term in equation 4.14 is proportional to w_{ki} and introduces a bias not present in the generalized Hebbian algorithm. Figure 4 shows an example of the convergence of the unsupervised spike algorithm (see equation 4.11) for simulated data. The graph gives the diagonalization score, which is a measure of the extent to which the matrix of cross-products of network outputs $E[\sigma_k(x)\sigma_m(x)]$ is diagonal. The score is the sum of the magnitudes of the off-diagonal elements divided by the sum of the magnitudes of the diagonal elements. Low numbers indicate a more diagonal matrix, and therefore a set of relatively uncorrelated tuning curves. There are 20 input cells with smooth, randomly generated tuning curves and maximum firing rate of 50 spikes per second. There are 4 output cells, which were trained for 1000 iterations, with each step occurring within a 20 millisecond bin, for a total learning time of 20 seconds. The learning rate was annealed from 0.015 to 0.0055 during the training.

Figure 5 shows the results of another simulation of the generalized Hebbian algorithm for 20 input cells (only 8 are shown) and 8 output cells. The other parameters are the same as for Figure 4. The figure shows 1-second spike tracings for the input cells on the left and the output cells on the right, demonstrating that output cells can have tuning curves that are more broadly or more narrowly tuned than the inputs. Because of the nature of the generalized Hebbian algorithm, the tuning curves for the output cells are orthogonal, and they proceed from greatest to least average firing rate in order.

Both the supervised and unsupervised algorithms make use of random spike data to perform learning whose long-time average behavior approxi-

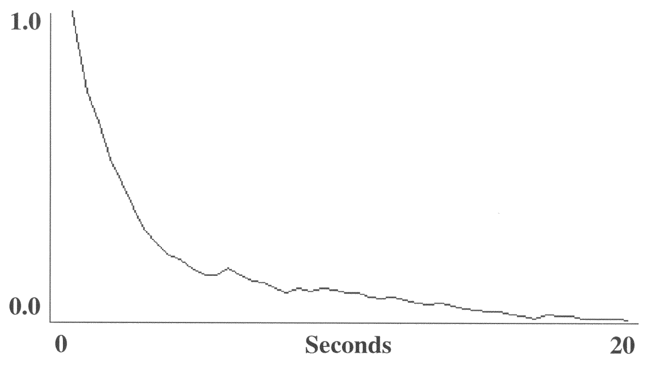

Figure 4: The diagonalization score as a function of time for the spike unsupervised generalized Hebbian algorithm. There are 20 input cells, 4 output cells, and 1000 time steps of 20 msec each. The learning rate decreased from 0.015 to 0.0055 during training.

mates smooth learning algorithms. These algorithms demonstrate the possibility of performing standard learning tasks in the single-spike domain.

5 Unsupervised Learning and Information

After the unsupervised algorithm has been trained, the tuning curves are orthogonal. In this section I show that this implies that the spike trains then become independent. This is important, since it implies that orthogonality of the tuning curves is the most separate that we can make the behavior of two cells. To see this, note that the mean of the conditional density $P[s_i \mid x]$ is given by

$$\int P[s_i \mid x]P[x]dx = P[s_i],$$

and define the zero-mean function

$$f_i(x) = P[s_i \mid x] - P[s_i],$$

for which

$$\int f_i(x)P[x]dx = 0.$$

Under the independence assumption we can write

$$P[s_i s_j] = \int P[s_i s_j \mid x]P[x]dx \tag{5.1}$$

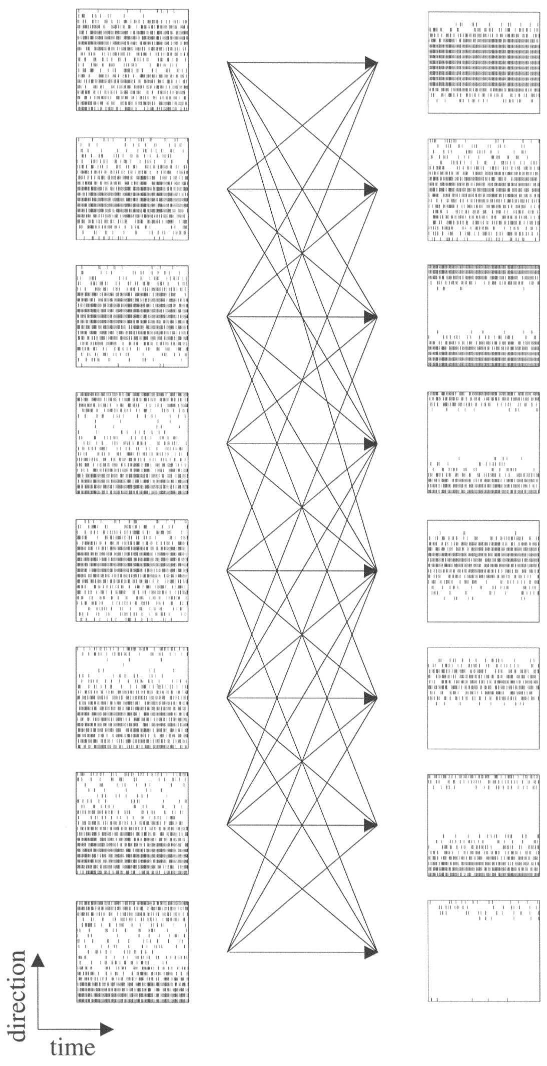

Figure 5: Spike tracings for input and output neurons after training with the generalized Hebbian algorithm (see text for details). Input neurons are shown in the column on the left and output neurons on the right. The lines in the middle illustrate the connectivity. Of the 20 input neurons, only the first 8 are shown. The output neurons are in decreasing order of average spike rate. Each row of the spike plots gives 1 second of simulated spike data for a fixed value of movement direction. Different rows of each plot show the neuron's spike response for different movement directions.

$$= \int P[s_i \mid x]P[s_j \mid x]P[x]dx \tag{5.2}$$

$$= \int \left(f_i(x) + P[s_i]\right)\left(f_j(x) + P[s_j]\right) P[x]dx \tag{5.3}$$

$$= P[s_i]P[s_j] + \int f_i(x)\, f_j(x)P[x]dx, \tag{5.4}$$

and we see that $P[s_i s_j] = P[s_i]P[s_j]$ (meaning that the spikes are independent) if and only if

$$\int f_i(x)\, f_j(x)P[x]dx = 0,$$

so that the (zero-mean) tuning curves are orthogonal. The converse also holds, so that maximum dependence occurs for maximally cross-correlated tuning curves.

The mutual information between x and the spikes s is given by

$$I[X; S] = H[X] - H[X \mid S],$$

and since $P[x \mid s]$ is given by equation 2.3, we have

$$\begin{aligned} H[X \mid S] &= \frac{1}{\mathcal{N}} \int (n \log \sigma(x) - \sigma(x) + \log P[x])P[x]dx \\ &= (H[X] - P[s] + H[\sigma])/\mathcal{N}, \end{aligned} \tag{5.5}$$

so the mutual information is maximized when the probability of firing $P[s]$ is large and the entropy of the tuning curve $H[\sigma]$ is small. For tuning curves that are gaussian, the entropy is minimized when the variance is small.

There is thus a parallel between linear properties of the zero-mean tuning curves such as cross-correlation, and nonlinear properties of the spike trains such as independence. The existence of such relationships shows that simple linear algorithms on the underlying probability distributions can lead to optimal nonlinear behavior at the level of the spikes. This fact can considerably simplify the types of calculations that are necessary and gives neural networks an important set of tools for performing nonlinear computations.

6 Duality

The results above demonstrate a close link between the behavior of spiking neurons and the underlying implicit tuning curves that describe their conditional probability of firing for different values of x. This link is a natural and unavoidable property of probabilistic models of spiking neurons. It is not specific to Poisson firing statistics, but will arise for any neuron model in which the probability of firing is smoothly modulated by a continuous tuning curve. The spike behavior and the tuning curve properties are two different yet completely equivalent ways of describing a neuron's response, and we can take advantage of their different mathematical properties to simplify certain types of computations.

The equivalence of these two descriptions leads to a duality relationship between the space of probabilistic events (spikes) and the space of deterministic bounded integrable functions (tuning curves) of a random variable x. (See Zemel et al., 1998, for an alternate discussion of this type of duality.) Independence in the spike space is related to orthogonality in the tuning curve space. Polynomial functions in the two spaces are equivalent (under the assumption of independent spike generators). I hypothesize that for any learning algorithm in the tuning curve space, there is a spike learning algorithm with similar convergence properties, as I have shown above for the LMS and generalized Hebbian algorithms.

7 Conclusion

The analysis given above shows the possibility of a direct link between familiar continuous neural network algorithms and learning algorithms performed by spiking neurons. It is not necessary to postulate that unit activity levels in a neural network are somehow formed from average neural firing rates (which require a long time to estimate), since these activity levels can be directly represented by the instantaneous probability of firing. Computations can be performed directly on firing probabilities without the necessity of first estimating these probabilities by estimating the average firing rate. This saves considerable time in synaptic transmission and provides a better model for the rapid transmission of information in large biological neural networks.

The model still begs the question of how the data are eventually read out. Stein (1967) gives a detailed discussion of approximation error for reading out rate codes. As for any neuronal processing, the code must necessarily be mapped from the final layer of processing onto the relevant muscles. The significant advantage of the model proposed here is that the rate code needs to be interpreted only at the final step of computation, so that it does not contribute to approximation errors at every step.

This model is a form of rate coding that provides a rapid computational method based on average firing rates, without explicitly computing those rates. It is different from codes that base processing on explicit differences in spike firing times or interspike intervals. For example, Softky (1996) argues for the use of a binary pulse code for maximizing the transmission of information. Such a code is definitely superior at the time of read-out, but it has considerable disadvantages in terms of the complexity of intermediate computations and learning. The distinction between "simple" rate codes and "efficient" binary spike codes (Softky, 1995) is blurred by the model proposed here, since we have the advantages of simple computation based on average firing rate without the disadvantage of slow and error-prone spike counting.

It is important to realize that these results are inherent properties of probabilistic models of spiking neurons. Whether a system chooses to make use of these properties, they are necessarily present. A cell with a tuning curve has an implicit conditional probability of firing, and when connected to other cells will have a direct effect on their probabilities of firing. This effect requires only the synaptic transmission time and is not dependent on temporal integration of spike rates. There is evidence that cell tuning is present within the first few spikes after stimulus onset, thus suggesting that rapid processing of rate information is indeed occurring in biological systems (Celebrini, Thorpe, Trotter, & Imbert, 1993; Tovee et al., 1993).

This work has shown probabilistic methods for interpreting the patterns of activity in populations of spiking neurons, computing new neuron activities with desired tuning curves, and learning in both supervised and unsupervised modes. It is hoped that such methods may form the basis for a probabilistic "neural calculus" that can allow interpretation of biological neural systems and explain some of the ways in which those systems perform internal computations.

Acknowledgments

I thank Emilio Bizzi, Simon Giszter, Alex Pouget, Peter Dayan, Barak Pearlmutter, and the reviewers for helpful comments and discussions. I was supported by a McDonnell-Pew Postdoctoral Fellowship. Dr. Bizzi's laboratory is supported by grants from the National Institutes of Health (NS09343) and the Office of Naval Research (N00014/90/J/1946).

References

Anderson, C. H. (1995). Unifying perspectives on neuronal codes and processing. In *Proc. 19th Int'l Workshop on Condensed Matter Theories.* Caracas, Venezuela.

August, D. A., & Levy, W. B. (1996). A simple spike train decoder inspired by the sampling theorem. *Neural Computation, 8,* 67–84.

Bair, W., & Koch, C. (1996). Temporal precision of spike trains in extrastriate cortex of the behaving Macaque monkey. *Neural Computation, 8,* 1185–1202.

Bialek, W., & Rieke, F. (1992). Reliability and information transmission in spiking neurons. *Trends in Neuroscience, 15*(11), 428–434.

Brown, T. H., Zador, A. M., Mainen, Z. F., & Claiborne, B. J. (1992). Hebbian computations in hippocampal dendrites and spines. In T. McKenna, J. Davis, & S. F. Zornetzer (Eds.), *Single neuron computation* (pp. 81–116). San Diego: Academic Press.

Celebrini, S., Thorpe, S., Trotter, Y., & Imbert, M. (1993). Dynamics of orientation coding in area V1 of the awake primate. *Visual Neuroscience, 10,* 811–825.

Fry, R. L. (1995). Observer-participant models of neural processing. *IEEE Trans. Neural Networks, 6*(4), 918–928.

Gabbiani, F., & Koch, C. (1996). Coding of time-varying signals in spike trains of integrate-and-fire neurons with random threshold. *Neural Computation, 8*, 44–66.

Georgopoulos, A. P., Kettner, R. E., & Schwartz, A. B. (1988). Primate motor cortex and free arm movements to visual targets in three-dimensional space. II. Coding of the direction of movement by a neuronal population. *J. Neurosci, 8*(8), 2928–2937.

Gerstner, W., & van Hemmen, J. L. (1994). How to describe neuronal activity: Spikes, rates, or assembiles? In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processin systems 6* (pp. 463–470). San Mateo, CA: Morgan Kaufmann.

Gerstner, W., Kempter, R., van Hemmen, J. L., & Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature, 383*, 76–78.

Hopfield, J. J. (1995). Pattern recognition computation using action potential timing for stimulus representation. *Nature, 376*, 33–36.

Judd, K. T., & Aihara, K. (1993). Pulse propagation networks: A neural network model that uses temporal coding by action potentials. *Neural Networks, 6*, 203–215.

Koch, C., & Poggio, T. (1992). Multiplying with synapses and neurons. In T. McKenna, J. Davis, & S. F. Zornetzer (Eds.), *Single neuron computation* (pp. 315–345). San Diego: Academic Press.

Konig, P., Engel, A. K., & Singer, W. (1996). Integrator or coincidence detector? The role of the cortical neuron revisited. *Trends in Neuroscience, 19*, 130–137.

Maass, W. (1994). On the computational complexity of networks of spiking neurons. In G. Tesauro, D. Touretzky, & D. Leen (Eds.), *Advances in neural information processing systems 7* (pp. 183–190). Cambridge, MA: MIT Press.

Maass, W. (1995). On the computational power of noisy spiking neurons. In G. Tesauro, D. Touretzky, & D. Leen (Eds.), *Advances in neural information processing systems 7* (pp. 183–190). Cambridge, MA: MIT Press.

Maass, W. (1996). Lower bounds for the computational power of networks of spiking neurons. *Neural Computation, 8*, 1–40.

Maass, W. (1997). Fast sigmoidal networks via spiking neurons. *Neural Computation, 9*, 279–304.

Magee, J. C., & Johnston, D. (1997). A synaptically controlled, associative signal for Hebbian plasticity in hippocampal neurons. *Science, 275*, 209–213.

Mainen, Z. F., & Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science, 268*, 1503–1506.

Markram, H., Lubke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science, 275*, 213–215.

Mel, B. W. (1993). Synaptic integration in an excitable dendritic tree. *J. Neurophysiology, 70*(3), 1086–1101.

Murthy, V. N., & Fetz, E. E. (1994). Effects of input synchrony on the firing rate of a three-conductance cortical neuron model. *Neural Computation, 6*, 1111–1126.

Paradiso, M. A. (1988). A theory for the use of visual orientation information which exploits the columnar structure of striate cortex. *Biological Cybernetics, 58*, 35–49.

Poggio, T., & Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science, 247*, 978–982.

Pouget, A., & Zhang, K. (1997). Statistically efficient estimation using cortical lateral connections. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems 9*. Cambridge, MA: MIT Press.

Pouget, A., Zhang, K., Deneve, S., & Latham, P. E. (1998). Statistically efficient estimation using population code. *Neural Computation, 10*, 373–401.

Rieke, F., Warland, D., van Steveninck, R., & Bialek, W. (1997). *Spikes: Exploring the neural code*. Cambridge, MA: MIT Press.

Salinas, E., & Abbott, L. F. (1994). Vector reconstruction from firing rates. *J. Computational Neuroscience, 1*, 89–107.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks, 2*, 459–473.

Sanger, T. D. (1996). Probability density estimation for the interpretation of neural population codes. *J. Neurophysiology, 76*(4), 2790–2793.

Sejnowski, T. F. (1995). Time for a new neural code? *Nature, 376*, 21–22.

Seung, H. S., & Sompolinsky, H. (1993). Simple models for reading neuronal population codes. *Proc. Natl. Acad. Sci USA, 90*, 10749–10753.

Shadlen, M. N., Newsome, W. T. (1994). Noise, neural codes and cortical organization. *Current Opinion in Neurobiology, 4*, 569–579.

Shadlen, M. N., Newsome, W. T. (1995). Is there a signal in the noise? *Current Opinion in Neurobiology, 5*, 248–250.

Shadlen, M. N., Britten K. H., Newsome, W. T., & Movshon, J. A. (1996). A computational analysis of the relationship between neuronal and behavioral responses to visual motion. *J. Neuroscience, 16*(4), 1486–1510.

Snippe, H. P. (1996). Parameter extraction from population codes: A critical assessment. *Neural Computation, 8*, 511–529.

Softky, W. R. (1994). Sub-millisecond coincidence detection in active dendritic trees. *Neuroscience, 58*(1), 13–41.

Softky, W. R. (1995). Simple codes versus efficient codes. *Current Opinoin in Neurobiology, 5*, 239–247.

Softky, W. R. (1996). Fine analog coding minimizes information transmission. *Neural Networks, 9*(1), 15–24.

Stein, R. B. (1967). The information capacity of nerve cells using a frequency code. *Biophysical Journal, 7*, 797–826.

Tal, D., & Schwartz, E. L. (1997). Computing with the leaky integrate-and-fire neuron: Logarithmic computation and multiplication. *Neural Computation, 9*.

Theunissen, F., Roddey, J. C., Stufflebeam, S., Clague, H., & Miller, J. P. (1996). Information theoretic analysis of dynamical encoding by four identified primary sensory interneurons in the cricket cercal system. *J. Neurophysiology, 75*(4), 1345–1364.

Tovee, M. J., Rolls, E. T., Treves, A., & Bellis, R. P. (1993). Information encoding and the responses of single neurons in the primate temporal visual cortex. *J. Neurophysiology, 70*(2), 640–654.

van Vreeswijk, C., & Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science, 274*, 1724–1726.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. In *IRE WESCON Conv. record, Part 4* (pp. 96–104).

Widrow, B., McCool, J. M., Larimore, M. G., & Johnson, C. R. (1976). Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proc. IEEE, 64*(8), 1151–1162.

Zemel, R. S., Dayan, P., & Pouget, A. (1998). Probabilistic interpretation of population codes. *Neural Computation, 10(2)*, 403–430.

Zohary, E., Shadlen, M. N., & Newsome, W. T. (1994). Correlated neuronal discharge rate and its implications for psychophysical performance. *Nature, 370*, 140–143.

Received March 26, 1997; accepted December 10, 1997.