How to Compare Two Quantities? A Computational Model of Flutter Discrimination

Tom Verguts

Abstract

■ A task that has been intensively studied at the neural level is flutter discrimination. I argue that flutter discrimination entails a combination of a temporal assignment problem and a quantity comparison problem, and propose a neural network model of how these problems are solved. The network combines unsupervised and one-layer supervised training. The unsupervised part clusters input features (stimulus + time window) and the supervised part categorizes the resulting clusters. After training, the model shows a good fit with both neural and behavioral properties. New predictions are outlined and links with other cognitive domains are pointed out. ■

INTRODUCTION

How do we and nonhuman animals compare two quantities? A wealth of behavioral and neuroimaging studies have considerably enhanced our understanding on the topic of quantity comparison at the systems level (for a review, see Dehaene, Piazza, Pinel, & Cohen, 2003). Data at the neural level remain scarce—yet, such data are essential for a detailed understanding of the processes at the basis of comparison. Recent studies by Nieder and colleagues investigated the neural basis of numerical representations when number is presented as a collection of dots (Nieder & Miller, 2004; Nieder, Freedman, & Miller, 2002), but how these representations are used in quantity comparison has not been investigated. There is, however, one particular type of quantity comparison for which the neural correlates have been extensively documented, and that is *flutter* discrimination (mostly due to the work of Romo and colleagues, e.g., Romo, Brody, Hernández, & Lemus, 1999). Flutter refers to the sensation experienced when one touches an object that vibrates in the range from about 5 to 50 Hz. These researchers trained monkeys on a vibrotactile discrimination task. The monkey is given mechanical stimulation (*base stimulus*, f_1) of a given frequency on one finger; this is followed by a *delay* period, which is followed by mechanical stimulation of a different frequency (*comparison stimulus*, f_2) to the same finger. After the comparison stimulus, the monkey has to release a different finger from a key and press one of two buttons depending on which of the two stimuli (f_1 or f_2) has the higher frequency. During performance, the researchers recorded neurons involved with this task in different brain areas (e.g., primary somatosensory cortex [S1], prefrontal cortex [PFC]).

Behavioral data from four monkeys originally reported in Hernández, Zainos, and Romo (2002) are reproduced in Figure 1. Each of the axes represents the frequency of one of the two stimuli. Each number represents the percent accuracy for that particular stimulus combination. Figure 1A shows that monkeys can learn this task well above chance levels. Figure 1B shows accuracies when a different stimulus distribution is used. There is a distance effect in the sense that it is more difficult to compare two stimuli if the distance between them is smaller. In other words, accuracy is higher for stimuli further removed from the diagonal. However, there is no indication that performance in this task depends on the absolute magnitude of the frequencies. This is a violation of Weber's law, which holds that discrimination performance on a pair of stimuli with magnitude difference dx and mean magnitude x is a function of dx/x.

Researchers studying the vibrotactile task have not focused on behavioral results, however, but on the properties of the neurons involved in this task, and these properties are now extensively documented (for overviews, see Romo, Hernández, & Zainos, 2004; Romo & Salinas, 2003). Early in the processing stream (area S1), neurons are primarily sensory: They respond to f_1 and/or f_2 during base stimulus and comparison period, respectively, they exhibit no delay period activity, and do not integrate the information from f_1 and f_2. To formalize these properties, I use the methodology used by Romo and Salinas (2003). Consider a linear regression for each neuron of the form: firing rate = $a_1 \times f_1 + a_2 \times$

Ghent University, Belgium

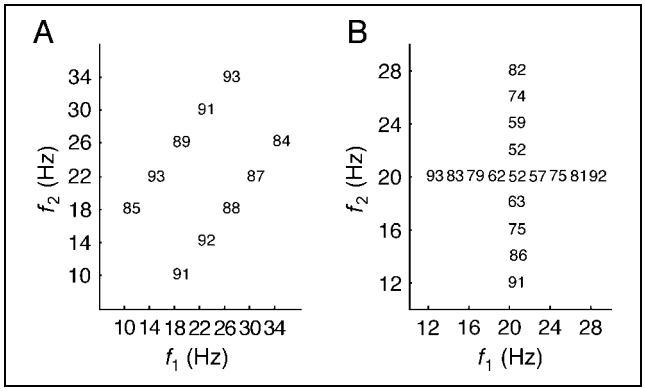

Figure 1. Behavioral data (percent accuracy) for f_1, f_2 combinations from two stimulus distributions (Hernández et al., 2002).

$f_2 + a_3$, calculated at different time points. The coefficients a_1 and a_2 indicate the dependence of the neuron on f_1 and f_2, respectively, during a particular time window. In this terminology, sensory neurons in area S1 have significant a_1 coefficients during the f_1 period, no significant coefficients during the delay period, and significant a_2 coefficients during the f_2 period. See row 1 of Figure 5 for an example. In these graphs, the coefficients are always located on one of the two main axes, meaning that the neurons respond to either f_1 or f_2, but not to both simultaneously. In this sense, these neurons do not integrate information about f_1 and f_2. Note also that the coefficients a_1 and a_2 are always positive (or zero), meaning that a higher stimulus frequency always leads to a higher firing rate.

As one proceeds further in the processing stream (secondary somatosensory cortex [S2], PFC, ventral premotor cortex [VPC]), neurons behave differently in all three stimulation periods. I summarize these neurons' response properties; they are reported more extensively in Romo, Hernández, Zainos, Lemus, and Brody (2002) for S2, in Romo et al. (1999) for PFC, and in Romo et al. (2004) for VPC. First, whereas S1 neurons are always positively tuned to f_1 ($a_1 > 0$) (Romo & Salinas, 2003), for later neurons the proportion of positively tuned neurons is only about 50%; the other 50% are negatively tuned ($a_1 < 0$). Positive and negative significant coefficients have approximately the same absolute size. Second, later neurons code f_1 during the delay period (implementing a memory process). Third, during the comparison (f_2) period, they integrate information from f_1 and f_2 (implementing a decision process). Possibly the most effective way to integrate f_1 and f_2 for comparison is to make the coefficients a_1 and a_2 of the same absolute size but opposite value, thus effectively implementing the difference $f_2 - f_1$ (or $f_1 - f_2$). This is exactly what many of these neurons do (see Figure 5, row 4, column 3, for an example).

There are, however, also differences between neurons in S2, PFC, and VPC: S2 neurons are the most ''sensory'' because they only exhibit delay activity during the early part (a few hundred milliseconds) of the delay period. Similarly, VPC neurons are more ''motor'' in that they only fire in the later part (a few hundred milliseconds) of the delay. In contrast, (a proportion of) PFC neurons possess f_1 sensitivity throughout the entire delay period. Furthermore, it is thought that the integrative behavior of S2 neurons in the comparison phase is actually due to feedback from lower downstream (PFC) neurons. For simplicity, however, here I ignore the differences between S2, PFC, and VPC neurons.

In the final stages of the processing stream (medial premotor cortex [MPC], primary motor cortex [M1]), neurons are primarily motor in that they do not respond during the base, delay, or comparison periods, but encode the difference between f_2 and f_1 in the response period (although MPC to a lesser extent than M1; see Romo et al., 2004). Presumably, these neurons collect the processed information from neurons higher upstream and send an appropriate signal to the relevant muscles.

Although this description of the neural properties provides important insights into how flutter discrimination is solved at the neural level, we also need to understand what is going on at the computational level. To do so, I consider the flutter discrimination task as one where temporal assignment is the fundamental problem to be solved. Consider the situation where one asks the price of a beer in a bar; depending on whether the bartender's answer is ''1 dollar and 50 cents'' or ''50 dollars and 1 cent,'' an adaptive organism will respond very differently. The issue here is that the response should be very different depending on which stimuli occur where in the temporal input stream. Similarly, a monkey's response to f_1 followed by f_2 should be different than the response to f_2 followed by f_1. However, if input (S1) neurons do not distinguish between f_1 and f_2 (see above), how is the organism to know? Two mathematical models have been proposed as to how monkeys solve this problem (Miller & Wang, 2006; Machens, Brody, & Romo, 2005). These represent two very sophisticated ways of how exactly the neuronal circuit may be designed so that it solves the temporal assignment problem (and hence, the task). However, the models are in a sense *too* sophisticated; they are constructed specifically to solve the vibrotactile discrimination task. In the present article, a different approach is followed. Rather than designing a fully operative model, I trained a model to solve this task. Such a training approach imposes an important constraint on a solution: If one considers each solution algorithm as a point in a parameter (weight) space, a training approach limits the search to algorithms (points) that are reachable by training. I started from a general hybrid unsupervised/supervised neural network, an approach that I have also applied in related domains (categorization: Verguts, Ameel, & Storms, 2004, and symbolic quantity processing: Verguts, Fias, & Stevens,

2005; Verguts & Fias, 2004). After training, the model was consistent with empirical data both at the behavioral and at the neural level. In addition, the resulting architecture is formally similar to recent models in other domains (e.g., visual perception: Riesenhuber & Poggio, 2000; Poggio & Edelman, 1990; sensorimotor transformations: Pouget, Deneve, & Duhamel, 2002; Salinas, 2000; see Discussion). I now discuss the two models that have been proposed earlier in the literature, followed by the new model.

Machens et al. Model

Both earlier models (Miller & Wang, 2006; Machens et al., 2005) are presented here in a simplified manner; both are actually instantiated as a large collection of integrate-and-fire neurons. Conceptually, however, they conform to the description given here. An outline of the Machens et al. (2005) model is shown in Figure 2A. S1 neurons are not modeled explicitly. S2 neurons (left in Figure 2A, input layer) respond either positively (input + cells) or negatively (input − cells) to both f_1 and f_2 frequencies. This information is transmitted to decision neurons (right in Figure 2A, output layer). Some of these output neurons represent the evidence that f_1 is larger (output + cells) and some of the output neurons represent the evidence that f_2 is larger (output − cells). In the base (f_1) period, input + cells activate output + cells, whereas input − cells activate output − cells (via solid connections in Figure 2A). Because of tonic activity sent to the output layer (provided by the node labeled A in Figure 2A) in combination with lateral inhibition, the information presented during the base period is preserved during the delay period.

The temporal assignment problem in this case consists of the fact that during the comparison period, a different set of weights has to be used, because now strong activity in input + cells is evidence for f_2 (rather than for f_1, as in the base period). This is achieved by making the model use the dashed connections in Figure 2A instead of the solid connections during the comparison period. Consequently, the input + cells now activate the output − cells, and input − cells activate the output + cells. To make sure that the model chooses the correct (solid or dashed) connections at the correct time (base or comparison period), an elaborate switching mechanism is required, shown in Figure 2B. Only the switching mechanism for the output + cell is shown; the mechanism for the output − cell is similar. The cell labeled E in Figure 2B is active during the base period; it inhibits the tonically active (inhibitory) T cell, thus allowing the input + cell to send its activation to output via the solid pathway. After the base (or delay) period, the E cell is shut off. Now the tonically active T cell inhibits the solid pathway, but the dashed pathway is no longer inhibited. Hence, this circuitry implements a switching mechanism. Although this mechanism solves the switching problem at the level of the input–output mappings, there is now an unsolved problem at the level of the E cell; it is unspecified how the organism should know when to switch off this cell. More generally, no learning mechanism for this elaborate mechanism has been specified.

Miller and Wang Model

A different way of solving the task was presented by Miller and Wang (2006) (an earlier version of the model appears in Miller, Brody, Romo, & Wang, 2003). Here I describe the linear approximation to the full model. A schematic overview of this model's architecture is shown in Figure 2C. A ''comparison'' (C) neuron responds (positively) to f_1 stimulation. If the C neuron's activity exceeds a threshold, it activates the ''memory'' (M)

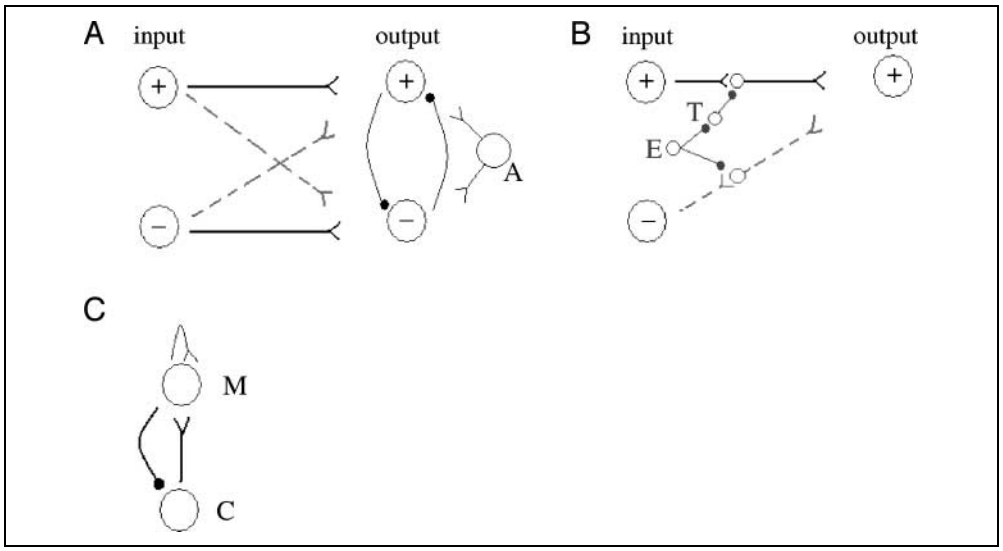

Figure 2. The two earlier models of flutter discrimination. (A, B) Machens et al. (2005) model. (C) Miller and Wang (2006) model. A fork-shaped end of a neuron stands for a positive connection; a dot-shaped end of a neuron represents an inhibitory connection.

neuron; this M neuron excites itself and inhibits the C neuron. Parameter settings are chosen in such a way that the M neuron pushes the C neuron below the threshold, but the M neuron itself remains active during the delay period (through self-excitation). The amount of inhibition sent to f_1 necessary to silence f_1 turns out to be equal to f_1. If f_2 is sufficiently strong to overcome the inhibition from the M neuron (i.e., if $f_2 > f_1$), the C neuron will become activated upon f_2 presentation. Hence, the C neuron acts as an $f_2 > f_1$ detector. In another class of neurons postulated by the authors, the C neuron responds negatively to f_1 but also activates its dedicated M neuron. By an analogous argument, such C neurons function as $f_1 > f_2$ detectors. By implementing a race between $f_2 > f_1$ detectors and $f_1 > f_2$ detectors, the correct response can then be chosen. The temporal assignment problem is solved by the fact that the f_1 frequency arrives in a ''silent'' network, whereas the f_2 frequency arrives in a network in which f_1 already had its influence, leading to a differential treatment of the two frequencies.

Like the Machens et al. (2005) model, this model is constructed without specifying how it could evolve online through task-relevant feedback. An empirical issue is that it predicts either no delay activity for the decision (C) neurons or a sign-flip for f_1 sensitivity from f_1 presentation to delay. The option of no delay activity is implausible given that decision-relevant neurons do exhibit delay period activity as a rule (e.g., Romo & Salinas, 2003). Concerning the sign-flip option, this entails that a C cell would respond positively to f_1 during f_1 presentation ($a_1 > 0$), but negatively to f_1 during the delay period ($a_1 < 0$) (i.e., a_1 flips its sign). Indeed, larger f_1 values lead to stronger inhibition from the M neuron, leading to a negative a_1 during the delay period. The sign-flip option has not been systematically investigated. Yet, the data that have been reported in sufficient detail to settle this issue speak against it. In particular, Romo et al. (1999) report that in the PFC, 95% of all neurons have the same sign during f_1 and delay (see also Hernández et al., 2002, Figure 5; and Romo et al., 2004, Figure 2).

New Model

In the hybrid unsupervised/supervised approach advocated here, learning a complex (nonlinear) mapping is broken down into two parts (subnetworks). The first subnetwork responds to input regularities by clustering co-occurring features in the input layer with an unsupervised learning rule. For example, if the combination of Neurons 1, 3, and 5 is often presented together, a cluster node will develop that codes for this regularity (i.e., it will respond to this combination of features). Applied to the present context, the central idea of the model is that the combination of stimulus input and a time point (or small time window) can constitute a cluster. The network develops nodes at the hidden layer that are sensitive to a combination of stimulus input and a time point. The result is that at different points in time, different hidden nodes will be active, thus solving the temporal assignment problem.

In the network, one set of input nodes codes for the stimulus (without differentiation between f_1 or f_2). The other input nodes code for time. See Figure 3 for a schematic overview of the model. Nodes at the level directly above the input level learn to code for combinations of the input node and particular time points. As time goes on, a Gaussian wave of activation travels across these nodes (a temporal wave; see Figure 3 for illustration and the Appendix for details). Essentially, with this method, timing consists of transforming a temporal to a spatial representation. Stimulus presentation leads to a changing pattern of activity in a collection of neurons (the time input neurons, generating the temporal wave), and the state of this collection of neurons at a given time point after stimulus presentation can be used as a timing signal by other neurons. This way of coding time is called *spectral timing* (Ivry & Spencer, 2004), and this idea has been used a number of times, for example, in models of cerebellar timing (Medina & Mauk, 2000; Buonomano & Mauk, 1994). Although the timing signals in the present context probably do not originate from the cerebellum (because the task's timescale is too long), the idea implemented

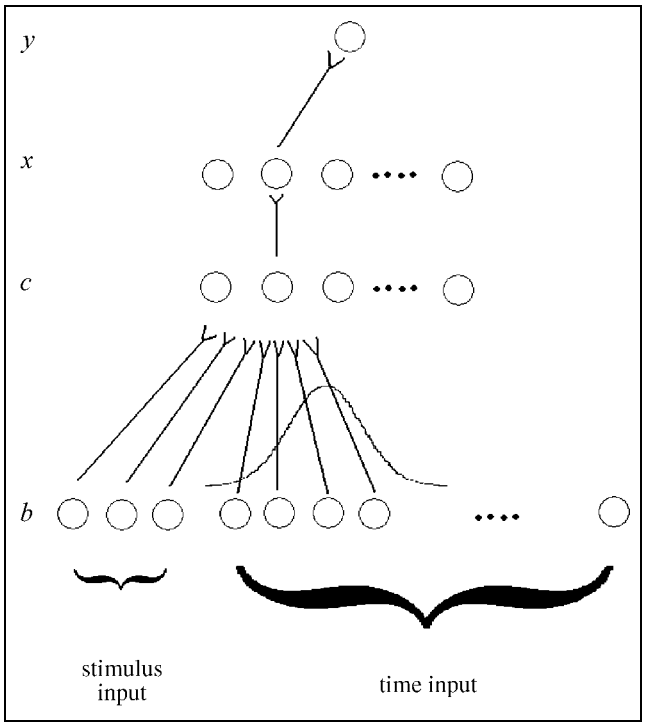

Figure 3. Schematic overview of the unsupervised/supervised learning model. For clarity, only a few selected connection weights are shown. The Gaussian curve over the time input nodes indicates the pattern of activation early in a trial across these nodes.

here is computationally the same. Thus, the model predicts the existence of neurons that are time dependent but not stimulus dependent. More importantly, such neurons are described by Brody, Hernández, Zainos, and Romo (2003) in the context of the flutter discrimination task. These authors recorded neurons in the PFC and found that of all neurons that were not stimulus dependent (i.e., did not respond to f_1 during the delay period), 52% were time dependent, meaning that they changed their firing rate during the delay period. They further observed that of all neurons in the PFC that were stimulus dependent, the great majority changed their firing rate during the delay period (97% were either early, late, or time-dependent persistent; see Brody et al., 2003, for details), suggesting that time-changing signals are critically important for solving this task. Similar time courses were described by Rainer and Miller (2002) in the context of a delayed match-to-sample task. Both Brody et al. and Rainer and Miller urged computational modelers to deemphasize the role of persistent neurons in working memory tasks (of which flutter discrimination is an example) and incorporate an important role for time-varying signals. Such an approach was followed by Singh and Eliasmith (2006), who modeled the PFC neurons described by Romo et al. (1999) with neurons that respond to both input frequency and time (although the flutter discrimination task itself was not modeled). The present model also adheres to this approach and provides computational (i.e., learning) arguments as to why such a move is important.

The second subnetwork connects the hidden nodes at the x level to responses at the y level (i.e., response $f_1 > f_2$ or $f_2 > f_1$). This mapping is trained with a supervised learning rule (the delta rule).

METHODS

To focus as clearly as possible on learning and computational issues, I used abstract ''leaky integrator'' neurons that are popular in cognitive neuroscience (e.g., Usher & McClelland, 2001). Mathematical details are provided in the Appendix. The lower part of the model ($b \rightarrow c$) learns in an unsupervised manner; it consists of a competitive network in which c-level nodes code for input patterns (presented at the lower b level). The middle part ($c \rightarrow x$) is a simple one-to-one mapping without learning. The upper part of the model ($x \rightarrow y$) is supervised in the sense that it receives feedback on the task (i.e., choosing the larger of f_1 and f_2).

The unsupervised part has two types of input. There is one set of nodes (three nodes in the implementation) coding for the input stimulus (f_1 during f_1 presentation, f_2 during f_2 presentation; see Figure 3). The activation of stimulus input node i equals $s_i \times f_1$ during f_1 presentation and $s_i \times f_2$ during f_2 presentation. The slope values s_i are randomly chosen between 0.1 and 1. Hence, these nodes constitute a population code implementing a positive linear monotonic stimulus encoding. To the extent that the linear approximation embodied in firing rate $= a_1 \times f_1 + a_2 \times f_2 + a_3$ is valid, this is consistent with what is empirically observed (Romo & Salinas, 2003). These nodes are intended to model the responses of S1 neurons because they respond to both f_1 and f_2 without discrimination and because they do not exhibit delay period activity. The second set of neurons at the input layer codes for time (by a temporal wave). In the present setup, there are 5 time points for the base period, 5 for the comparison period, and 30 for the delay period. I chose for 30 delay-period time points because in the experimental paradigm used by Romo and colleagues, the delay period was usually six times longer than the base and comparison periods (3 and 0.5 sec, respectively). There are as many time input nodes as time points in a trial (i.e., 40 time input nodes).

At each time point, activation in the input layer is transmitted to the next layer (c). At some time points, a winner-take-all competition occurs across the c neurons. In particular, the probability of a competition was equal to .99 at each time point, but there was a refractory period of three time points in which no competition could occur. This ensured that there usually was competition every four time points. The winning node at the c level (i.e., the node with the highest activation) won the competition and was allowed to change its weights. I used a simple variant of the classical updating rule for unsupervised networks (see Appendix). The winning node at the competitive (c) level sends its activation to the next level (x) in a one-to-one fashion.

The supervised part of the model consists of the (linear) $x \rightarrow y$ mapping. After the last (40th) time point, feedback is given and the $x \rightarrow y$ mapping is trained using the delta rule. There are 20 nodes at each of the c and x levels, and 1 node at the y level (see Figure 3).

Stimuli in the training phase were sampled from a ''band-shaped'' stimulus distribution, as most frequently used in the Romo studies (see Figure 1A). Twenty simulation rounds were performed. In each simulation round, I trained the model for 20,000 trials. During training, a small amount of Gaussian noise (SD = 1/500) was added to all nodes at each time point. The target values (used in the supervised learning layer) were either $f_1 - f_2$ or $f_2 - f_1$ (randomly chosen in each simulation round).

After training in each simulation round, the model was presented 1000 (new) test trials, also with stimuli from the band-shaped distribution. To avoid ceiling effects in accuracy, the noise level was somewhat lower in the test phase (SD = 1/50). There was no training in the test phase. I fitted linear regression equations of the form, activation $= a_1 \times f_1 + a_2 \times f_2 + a_3$, to the different model neurons at the last time point of each of the

three periods (base, delay, comparison). The units of measurement in the regression were the 1000 test trials. The model was tested on three stimulus distributions, the band-, cross-, and square-shaped distributions (see Figure 4A–C, respectively).

RESULTS

Behavioral

During testing, a response was considered correct if the sign of the activation of the decision node at the last time point was correct (i.e., if $f_1 > f_2$, the decision node should be positive if the model is trained on the $f_1 > f_2$ decision). The accuracies for the different conditions in the testing phase (band-, cross-, and square-shaped stimulus distributions) are shown in Figure 4A–C. Each number designates the percent correct at the corresponding (f_1, f_2) stimulus pair, aggregated over the 20 simulations. Results were similar for individual simulation rounds.

As can be seen in Figure 4A, the model achieves high accuracy (comparable to the monkey data). It is also able to generalize to stimuli it has never seen before (Figure 4B and C), and it exhibits a distance effect (stimuli are more difficult when the frequencies are more similar; Figure 4B and C). Furthermore, its performance does not depend on the overall (average) magnitude of the stimulus frequencies (violating the Weber effect as in the empirical data; Figures 1A and 4A).

To test the robustness of these findings, I conducted a new simulation study (again with 20 simulation rounds) where the stimuli in the training phase were sampled from the square-shaped distribution. The test phase results are shown in Figure 4D–F. Clearly, all results are robust with respect to the initial training distribution.

Neural

The neural data of the simulation study with a square-shaped stimulus distribution in the training phase were very similar to those of the simulation study with a band-shaped distribution in the training phase. Therefore, only the results for the simulation study with a band-shaped training distribution will be discussed explicitly.

The coefficients over the 20 simulation rounds, during the three stimulus periods (columns), are plotted in Figure 5 (a_1 on abscissa, a_2 on ordinate). The first row shows the coefficients of the stimulus input nodes. During the base presentation period, this neuron is sensitive to f_1, in the delay period it is silent (all coefficients near zero), and in the comparison period it is sensitive to f_2.

Neurons at the intermediate (c, x) level learn to cluster the input space. These nodes cluster combinations of stimulus input and time input. In particular, each (c or x) neuron will code for input frequency in combination with some time points. Note that a cluster does not represent a *specific* input frequency and a specific time window: Each cluster represents the com-

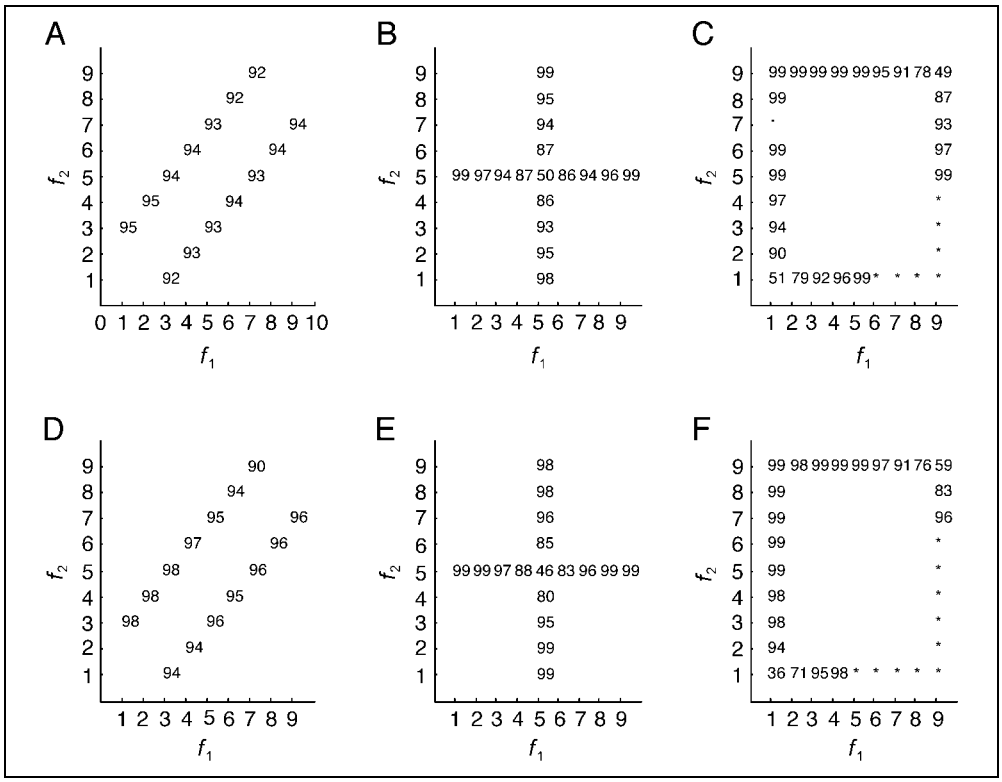

Figure 4. Simulated accuracy data. (A–C) Test phase results when the training phase used a band-shaped stimulus distribution. (D–F) Test phase results when the training phase used a square-shaped distribution. An asterisk (*) indicates 100% correct.

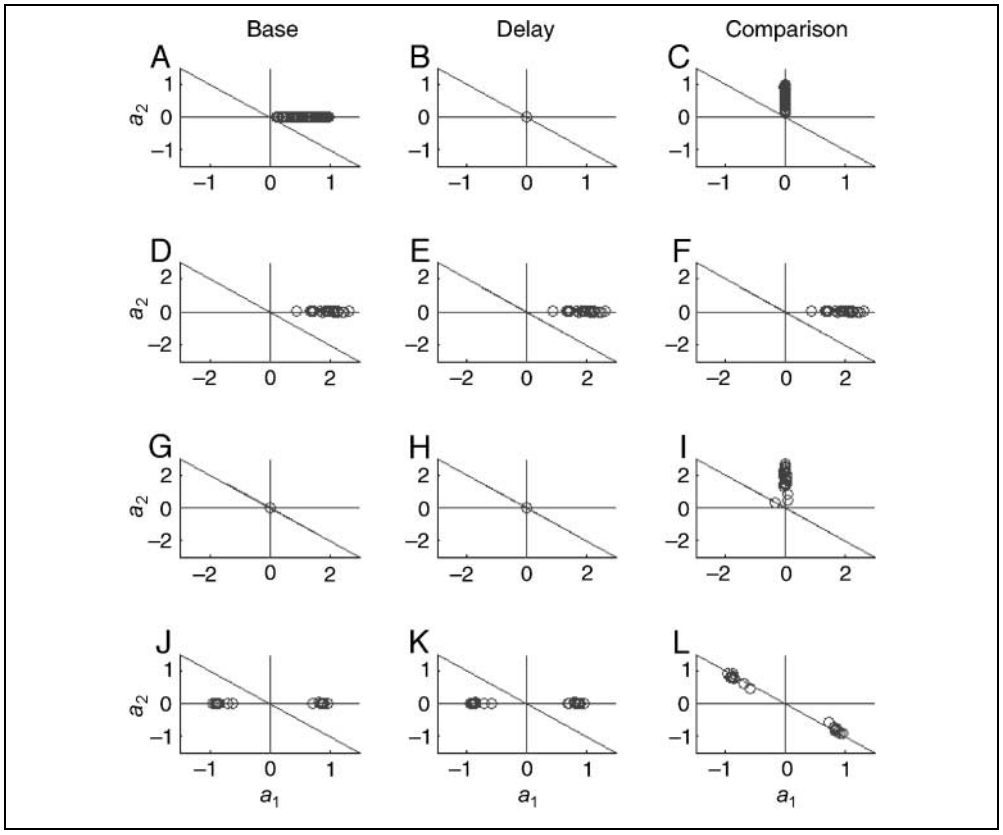

Figure 5. Simulated neural data. Each circle represents the coefficients of a particular neuron in a given period (base, delay, or comparison). (A–C) Coefficients of stimulus input (*b* level) neurons. (D–I) Hidden-level (*x* level) neurons. (J–L) Decision (*y* level) neurons.

bination of input (any input) and a specific time window. The a_1 and a_2 coefficients of the *x* neurons are depicted in rows 2 and 3. Neurons with a_1 coefficients in the base period larger than 0.3 (in absolute value) were considered f_1-sensitive and are depicted in row 2. These neurons act as memory neurons for f_1 stimuli. Neurons with a_2 coefficients larger than 0.3 (in absolute value) in the comparison period were considered f_2-sensitive and are depicted in row 3. These neurons act as memory neurons for f_2 stimuli. To illustrate, consider the model in Figure 3. The *c* neuron whose connection weights are depicted is sensitive to input frequency and an early time window (assuming that the temporal wave runs from left to right in the picture); hence, its a_1 coefficient would be positive in each period and its a_2 coefficient in none of the periods, and this neuron would appear in row 2 of Figure 5.

In the final row of Figure 5, coefficients of output (*y*) neurons are depicted. During f_1 presentation and delay, they code f_1 in either a positively or negatively monotonic manner. During f_2 presentation, they integrate f_1 and f_2 information and encode either $f_1 - f_2$ or $f_2 - f_1$ (depending on the target values used during training). The model naturally accounts for the fact that during base (f_1) presentation, one observes negatively tuned neurons in decision areas (e.g., Romo & Salinas, 2003); this is because these neurons code the difference $f_2 - f_1$. To sum up the neural results, there are three types of frequency-sensitive neurons in the network, which can (from lower to higher levels) be designated as sensory, memory, and decision neurons, respectively.

In the simulations discussed here, there were 20 hidden nodes (at both the *c* and the *x* level). I experimented with networks with different numbers of nodes, but as long as the number of hidden nodes was sufficiently high, this did not change the results substantially. The reason is that the network simply did not train superfluous neurons: Neurons were recruited to span different time windows depending on whether competition occurred in that particular time window or not. When there were nodes left when all time windows were represented at the hidden layers, remaining *x*-level neurons were not trained by the model and retained their initial (i.e., random) connection weight to the decision node (*y*).

DISCUSSION

I have proposed how a hybrid unsupervised/supervised neural network may learn to solve the temporal assignment problem as instantiated in the vibrotactile discrimination task. After training, the model was found to be consistent with both behavioral and neural data. Yet, the account given here of how monkeys learn a discrimination task is simplified in a number of ways. First, I did not discuss motor neurons. Second, one typically observes a gradual change in neural properties from

sensory to motor areas (rather than a discrete shift as in this model; Romo et al., 2004). I tentatively propose that the ''intermediate'' neurons in S2 and VPC constitute a mixture of sensory, memory, and decision neurons as observed in the model. Third, the neurons themselves are mathematical abstractions that capture only the computationally essential aspects of real neurons. These simplifications allowed me to focus more clearly on the core computational aspects of the task. I do not think that removing any of these simplifications by, for example, implementing the model as a large collection of integrate-and-fire neurons would substantially change the conclusion.

One other aspect that is presumably a simplification is how the temporal wave is implemented. With the current implementation, the monkey will generalize to only small shifts in delay, but not from a given delay (e.g., 3 sec) to a much longer one (e.g., 6 sec; Romo et al., 1999). Alternative implementations of the temporal wave that deal with this problem are possible. For example, one might envision that the temporal wave also requires stimulus (f_1, f_2) input for traveling; in this case, generalization from a 3- to a 6-sec interval would be immediate, as is the case in the Machens et al. (2005) and Miller and Wang (2006) models. However, one should be cautious in not making the model *too* flexible with respect to delay period duration. Brody et al. (2003) report that when monkeys are switched from a (typical) block containing 3-sec delay trials to a block consisting of trials with 6-sec delay intervals, learning occurs in PFC neurons. Behavioral data are not reported for this 6-sec delay block, but the fact that there is learning at the neural level indicates that the task representations learned by the monkey do not completely abstract away from the concrete delay period; otherwise, it is hard to see why learning should occur in the 6-sec delay trial block. Hence, although the present model possibly predicts not enough generalization from 3- to 6-sec delays, the Machens et al. and Miller and Wang models probably predict too much generalization. This remains an issue of future research at this time, but it would seem easier to amend a fundamentally time based model to deal with the Brody et al. delay shift data than a fundamentally abstract model.

Sensitivity and Selectivity

Romo and colleagues observed that all neurons that were sensitive to numerosity were so in a (positive or negative) monotonic manner. This is in sharp contrast with the number-sensitive neurons that were described by Nieder et al. (2002). These authors investigated the response of prefrontal and parietal neurons to numerosities presented as collections of dots (controlling for physical variables such as luminance). The neurons that were sensitive to number responded in a nonmonotonous way: In particular, if a given neuron responded most strongly to numerosity z, it also responded (somewhat weaker) to $z - 1$ and $z + 1$, even weaker to $z - 2$ and $z + 2$, and so on. Hence, whereas the Romo neurons are quantity *sensitive*, the Nieder neurons are also *selective* for a particular number.

I have proposed a computational model to describe the properties and development of Nieder neurons, and proposed that the number-selective neurons were actually neurons at a relatively late stage of processing that received input from monotonously number-sensitive (but not number-selective) neurons (Verguts & Fias, 2004). I thus predicted the existence of monotonous number neurons. The issue remains, however, why number-selective neurons were so ubiquitous in the Nieder et al. (2002) study, but not in the Romo studies. Although a number of variables differ between the two paradigms (e.g., input modality), I propose that the relevant variable is the task that was used. Nieder and colleagues used a same/different task (the monkey was to indicate whether two dot displays contained the same or a different number of dots); Romo and colleagues used a comparison task. Because comparison is a linearly separable task and same/different judgment is not, it might be an advantage to transform a number-sensitive (monotonous) stimulus code to a number-selective (nonmonotonous) code when the task is difficult (i.e., nonlinear), whereas this might not be required when the task mapping is easy (i.e., linear). To test this idea, I trained a simple feedforward network (with sigmoid nodes and learning rate 0.1), giving it two numbers as input on each trial on two tasks: number comparison and the same/different task. Initial weights were random. The input coding scheme was either number-selective (a different active input node for each number n; nonmonotonous coding) or number-sensitive (n active input node for number n; monotonous coding). For each input coding scheme/task combination, the model was trained 20 times and the number of trials required for solving the task was averaged across simulations. Because comparison is a nonlinearly separable task, whereas same/different judgment is not (Minsky & Papert, 1969), I gave the model three hidden nodes when solving the same/different task. Hence, the model is more powerful, but also slower to learn in the same/different task, and therefore the task main effect cannot be interpreted. What is interesting though is that I obtained an interaction between task and input coding scheme. For the comparison task, the monotonous encoding was more efficient (on average, 10,621 and 1151 trials were required for learning the task with nonmonotonous and monotonous encoding, respectively). On the other hand, for the same/different task, the nonmonotonous encoding was more efficient (on average, 1059 and 4347 trials were required, respectively). This confirms the idea that different input coding schemes are differentially efficient for differ-

ent tasks, and the brain might take advantage of these differences.

The Modeling Framework: Conjunctions and Categorization

In the present model, nodes at the hidden layer respond to combinations (conjunctions) of input features, and nodes in the output layer categorize these nodes. For this reason, I will call this a CC (for conjunctions + categorization) network. This general architecture is reminiscent of existing models in many related fields. For example, a dominant view in visual perception is that perception occurs through a sequence of steps in which nodes at a given level code for combinations (conjunctions) of input features at one level earlier (Riesenhuber & Poggio, 2000; Rolls & Millward, 2000). In some of these models, nodes at the last level assign a category label to the nodes from the second to last level. Similar models have been proposed in categorization (Verguts et al., 2004; Kruschke, 1992). Another field in which this idea has been applied is that of sensorimotor transformations in parietal cortex. According to the basis function theory developed by Pouget and colleagues (e.g., Pouget et al., 2002), neurons in the parietal cortex respond to combinations of signal and context input in order to compute appropriate coordinate transformations. For example, each parietal neuron might respond to a combination of a retinotopic signal input and an eye position context input. A large collection of such (basis function) neurons can then generate the appropriate (e.g., head-centered or arm-centered) coordinates. A similar approach has been followed by Salinas (2004) in his account of task switching in the PFC. He implemented a network that learned a set of sensorimotor mappings by making each neuron at the hidden layer respond to a combination of a sensory signal and a context cue. In this way, each stimulus/context combination led to a unique hidden layer activation pattern, and it sufficed to then train the mapping from hidden nodes to output with (an offline version of) the delta rule.

Within the class of CC networks, a commonality that runs across the basis function theory, the Salinas task switching model, and the present model, is that a sensory and contextual signal are nonlinearly combined to provide a unique cue to the output layer. Machens et al. (2005) already suggested that their switching mechanism might be implemented by these means, and the model presented in this article actually does this. In this way, the present model can be seen as an online learning version of the Machens et al. model. Besides the sign-flipping mechanism, one important remaining difference is how the decision is reached. Whereas the present model obtains the information to reach a decision in a ''bottom-up'' manner, in the Machens et al. model a decision is reached through a lateral inhibition mechanism. At this moment, there is no empirical reason to choose one mechanism or the other (see below); in the absence of such information, I have chosen to implement the simpler of these two.

One reason why CC networks are increasingly used in recent years is that they solve a number of computational problems posed by competing approaches. Two-layer (input and output) networks and their learning rules are simple and attractive, but they have limited computational power (Minsky & Papert, 1969). One solution is to propagate the error signal from output back to the hidden layer (i.e., backpropagation; Rumelhart, Hinton, & Williams, 1986), but this approach suffers from other computational problems (e.g., catastrophic forgetting; McCloskey & Cohen, 1989) and is not biologically plausible because the information used for weight adaptation is not locally available to the neuron but must be obtained (backpropagated) from layers further downstream. Other solutions, although mathematically different (e.g., Boltzmann machines), suffer from the same problems. It is often claimed that the Boltzmann machine learning procedure avoids the locality problem. However, learning in these models is split into two distinct phases (negative and positive phase), and information from the first phase must be retained during the second phase, leading to a temporal version of the locality problem faced by backpropagation. The CC framework avoids these problems. First, all information is locally available. Second, there is no catastrophic forgetting because each input pattern only activates a single (or few) hidden nodes. In sum, there are excellent reasons why the (monkey) brain would use CC networks for stimulus processing.

Empirical Predictions

At this time, there are three accounts of how monkeys perform flutter discrimination. Directed empirical questions can now be stated to tease them apart. The first question concerns the response properties of neurons that show delay period activity. The model presented here and the Machens et al. (2005) model both predict that a neuron with delay activity that responds positively (negatively) to f_1 during f_1 presentation will continue to respond positively (negatively) during delay. The Miller and Wang (2006) model predicts a sign-flip to occur during the delay. As noted above, at least in the PFC, only a very small proportion of task-relevant neurons exhibit this property (5%). It therefore remains to be seen what the proportion of sign-flipping neurons is in other cortical areas.

The second prediction concerns the existence of neurons that are responsive only to f_1 (not f_2) or only to f_2 (not f_1). It is clear from Figure 5 (rows 2 and 3) that my model predicts such f_1-only and f_2-only neurons at the hidden (c and x) layers. The Machens et al. (2005) model predicts such neurons also. To see this, consider Figure 2B: The neurons between the input and output layer

(those that are directly influenced by the switching mechanism) are responsive to only one of the two frequencies, because each of these two neurons is inhibited during exactly one of the stimulus periods by the switching mechanism. What distinguishes my model from the Machens et al. model is that neurons that are exclusively sensitive to f_1 should exhibit delay activity according to my model but not (necessarily) according to the Machens et al. model (depending on when the E neuron is shut off).

Finally, the new model predicts the existence of temporally selective neurons in the flutter-discrimination processing stream (i.e., neurons that respond selectively to a particular time interval). Despite a wealth of temporal processing studies (see Mauk & Buonomano, 2004, for a review), few studies have examined the temporal tuning in awake animals. One exception is the Leon and Shadlen (2003) study, but in that study, time itself was the relevant variable to be monitored. Another exception is the study of Ehrlich, Casseday, and Covey (1997), but this was performed with animals (bats) and on a timescale (tens of milliseconds) that is very different from the situation studied here. As noted above, such purely time-sensitive neurons were described by Brody et al. (2003), but a detailed analysis of their properties (other than that they were time sensitive) is lacking at this time. The Machens et al. (2005) model also predicts temporally sensitive neurons, but it predicts them to have a qualitatively different behavior. In particular, the E neuron in Figure 1B should respond during f_1 presentation (irrespective of the f_1 value) and possibly the delay period, but not later. The model presented in this article could provide an impetus toward analyzing time-sensitive neurons in the flutter discrimination task in more detail. More generally, the wealth of findings reported by Romo and colleagues has inspired at least three models of how this task may be neurally implemented. The models may now in turn inspire focused empirical tests.

APPENDIX

Here I describe the activation and learning dynamics in detail. The notations b, c, x, and y will be used to denote both levels in the model and activation values of nodes at that particular level, as in the text. Activation $b_i(t)$ of the stimulus input nodes ($i = 1, 2, 3$) is equal to

$$b_i(t) = \begin{cases} s_i f_1 & \text{during } f_1 \text{ presentation} \\ 0 & \text{during delay} \\ s_i f_2 & \text{during } f_2 \text{ presentation} \end{cases}$$

The parameter s_i is the slope of stimulus input node i (randomly chosen between 0.1 and 1). Activation of time input node i, with "preferred" time point t_i, is equal to $\exp\{-(t - t_i)^2\}$ at time point t, implementing a Gaussian wave.

As mentioned in the text, c-level neurons function as leaky integrators; their activation $c_j(t)$ at time t is equal to

$$c_j(t) = (1 - \beta) \times c_j(t - 1) + \beta \times \sum_{i=1}^{43} w_{ij}^{(1)} b_i(t)$$

where $w_{ij}^{(1)}$ is the connection weight from an input neuron i to a competitive neuron j. The parameter β equals 0.9. When there was competition at a given time point, a winner was determined at the c level: winner = $\text{argmax}_j\{c_j\}$. The winning node was allowed to change its input weight vector $\mathbf{w}_{\text{win}}$ with a simple variant of a standard unsupervised learning rule (e.g., Herz, Krogh, & Palmer, 1991):

$$\mathbf{w}_{\text{win}}^{(1)} = (1 - \alpha_1) \times \mathbf{w}_{\text{win}}^{(1)} + \alpha_1 \times I(\mathbf{b} > \theta)$$

The vector $\mathbf{b}$ is the concatenation of the stimulus input and time input nodes, and the term $\boldsymbol{I}(\mathbf{b} > \theta)$ denotes a (vector valued) indicator function $I(\cdot)$, which is (componentwise) 1 if its argument is true and 0 otherwise. The parameter θ is a (scalar) threshold value (set at 0.02). For example, with $\mathbf{b}$ = [0.4, 0.6, 0.3, 0.01, 0.3, 1, 0.3, 0.01, 0, ..., 0], the result is $\boldsymbol{I}(\mathbf{b} > \theta)$ = [1, 1, 1, 0, 1, 1, 1, 0, ..., 0]. The parameter α_1 was set at 0.9.

The winning node at the c level transmits its activation to the x level in one-to-one fashion: $x_{\text{win}}(t) = x_{\text{win}}(t - 1) + c_{\text{win}}(t)$. One could think of the x-level neurons as being of the integrator type with different β parameters than the c-level neurons; a motivation for this change is that the x-level activity evolves at a different timescale (in the implemented model, usually only every four time points). Finally, the decision neuron integrates activity from the x level:

$$y(t) = (1 - \beta) \times y(t - 1) + \beta \times \sum_{i=1}^{20} w_i^{(2)} x_i(t)$$

In this equation, $w_i^{(2)}$ is the connection from x-level node i to the decision node. Learning in the $x \rightarrow y$ mapping follows the standard delta, or Widrow–Hoff rule (e.g., Herz et al., 1991); after the final time point in a given trial, the weight vector $\mathbf{w}^{(2)}$ is thus adapted as

$$\mathbf{w}^{(2)} = \alpha_2 \times (d - y) \times \mathbf{x},$$

where d is the desired (or target) value for that particular trial. As noted in the text, for each trained network, I randomly chose whether the network was required to respond $y = 1$ when $f_1 > f_2$ (and zero otherwise) or to respond $y = 1$ when $f_1 < f_2$ (and zero otherwise). In the former case, $d = f_1 - f_2$; in the latter, $d = f_2 - f_1$. The learning rate parameter α_2 was set at 0.01.

Acknowledgments

I thank Wim Fias, Wim Notebaert, and two anonymous reviewers for helpful comments on this article.

Reprint requests should be sent to Tom Verguts, Department of Psychology, H. Dunantlaan 2, 9000 Ghent, Belgium, or via e-mail: Tom.Verguts@UGent.be.

REFERENCES

Brody, C. D., Hernández, A., Zainos, A., & Romo, R. (2003). Timing and neural encoding of parametric working memory in macaque prefrontal cortex. *Cerebral Cortex, 13,* 1196–1207.

Buonomano, D. V., & Mauk, M. D. (1994). Neural network model of the cerebellum: Temporal discrimination and the timing of motor responses. *Neural Computation, 6,* 38–55.

Dehaene, S., Piazza, M., Pinel, P., & Cohen, L. (2003). Three parietal circuits for number processing. *Cognitive Neuropsychology, 20,* 487–506.

Ehrlich, D., Casseday, J. H., & Covey E. (1997). Neural tuning to sound duration in the inferior colliculus of the big brown bat, *Eptesicus fuscus*. *Journal of Neurophysiology, 77,* 2360–2372.

Hernández, A., Zainos, A., & Romo, R. (2002). Temporal evolution of a decision-making process in medial premotor cortex. *Neuron, 33,* 959–972.

Herz, J., Krogh, A., & Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Wesley.

Ivry, R. B., & Spencer, R. M. C. (2004). The neural representation of time. *Current Opinion in Neurobiology, 14,* 225–232.

Kruschke, J. K. (1992). ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review, 99,* 22–44.

Leon, M. I., & Shadlen, M. N. (2003). Representation of time by neurons in the posterior parietal cortex of the macaque. *Neuron, 38,* 317–327.

Machens, C. K., Romo, R., & Brody, C. K. (2005). Flexible control of mutual inhibition: A neural model of two-interval discrimination. *Science, 307,* 1121–1124.

Mauk, M. D., & Buonomano, D. V. (2004). The neural basis of temporal processing. *Annual Review of Neuroscience, 27,* 307–340.

McCloskey, M., & Cohen, N. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation, 24,* 109–164.

Medina, J. F., & Mauk, M. D. (2000). Computer simulation of cerebellar information processing. *Nature Neuroscience, 3,* 1205–1211.

Miller, P., Brody, C. D., Romo, R., & Wang, X. (2003). A recurrent network model of somatosensory parametric working memory in the prefrontal cortex. *Cerebral Cortex, 13,* 1208–1218.

Miller, P., & Wang, X. (2006). Inhibitory control by an integral feedback signal in prefrontal cortex: A model of discrimination between sequential stimuli. *Proceedings of the National Academy of Sciences, U.S.A., 103,* 201–206.

Minsky, M. L., & Papert, S. A. (1969). *Perceptrons*. Cambridge: MIT Press.

Nieder, A., Freedman, D. J., & Miller, E. K. (2002). Representation of the quantity of visual items in the primate prefrontal cortex. *Science, 297,* 1708–1711.

Nieder, A., & Miller, E. K. (2004). Analog numerical representations in rhesus monkeys: Evidence for parallel processing. *Journal of Cognitive Neuroscience, 16,* 889–901.

Poggio, T., & Edelman, S. (1990). A network that learns to recognize 3-D objects. *Nature, 343,* 263–266.

Pouget, A., Deneve, S., & Duhamel, J.-R. (2002). A computational perspective on the neural basis of multisensory spatial representations. *Nature Reviews Neuroscience, 3,* 741–747.

Rainer, G., & Miller, E. K. (2002). Timecourse of object-related neural activity in the primate prefrontal cortex during a short-term memory task. *European Journal of Neuroscience, 15,* 1244–1254.

Riesenhuber, M., & Poggio, T. (2000). Models of object recognition. *Nature Neuroscience, 3,* 1199–1204.

Rolls, E. T., & Millward, T. (2000). A model of invariant object recognition in the visual system: Learning rules, activation functions, lateral inhibition, and information-based measures. *Neural Computation, 12,* 2547–2572.

Romo, R., Brody, C. D., Hernández, A., & Lemus, L. (1999). Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature, 399,* 470–473.

Romo, R., Hernández, A., & Zainos, A. (2004). Neuronal correlates of a perceptual decision in ventral premotor cortex. *Neuron, 41,* 165–173.

Romo, R., Hernández, A., Zainos, A., Lemus, L., & Brody, C. D. (2002). Neuronal correlates of decision-making in secondary somatosensory cortex. *Nature Neuroscience, 5,* 1217–1225.

Romo, R., & Salinas, E. (2003). Flutter discrimination: Neural codes, perception, memory, and decision making. *Nature Reviews Neuroscience, 4,* 203–218.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323,* 533–536.

Salinas, E. (2004). Fast remapping of sensory stimuli onto motor actions on the basis of contextual information. *The Journal of Neuroscience, 24,* 1113–1118.

Singh, R., & Eliasmith, C. (2006). Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *The Journal of Neuroscience, 26,* 3667–3678.

Usher, M., & McClelland, J. L. (2001). The time course of perceptual choice: The leaky, competing accumulator model. *Psychological Review, 108,* 550–592.

Verguts, T., Ameel, E., & Storms, G. (2004). Measures of similarity in models of categorization. *Memory & Cognition, 32,* 179–189.

Verguts, T., & Fias, W. (2004). Numerical representations in animals and humans: A neural model. *Journal of Cognitive Neuroscience, 16,* 1493–1504.

Verguts, T., Fias, W., & Stevens, M. (2005). A model of exact small-number representation. *Psychonomic Bulletin & Review, 12,* 66–80.