Parameterization of connectionist models

RAFAL BOGACZ and JONATHAN D. COHEN
Princeton University, Princeton, New Jersey

We present a method for estimating parameters of connectionist models that allows the model's output to fit as closely as possible to empirical data. The method minimizes a cost function that measures the difference between statistics computed from the model's output and statistics computed from the subjects' performance. An optimization algorithm finds the values of the parameters that minimize the value of this cost function. The cost function also indicates whether the model's statistics are significantly different from the data's. In some cases, the method can find the optimal parameters automatically. In others, the method may facilitate the manual search for optimal parameters. The method has been implemented in Matlab, is fully documented, and is available for free download from the Psychonomic Society Web archive at www.psychonomic.org/archive/.

Connectionist models are abstract models of how information processing occurs in the brain during performance of psychological tasks (Rumelhart, McClelland, & the PDP Research Group, 1986). In these models, task performance is simulated by the flow of activity between sets of processing units that comprise processing pathways. Connectionist models have been successfully used to explain a variety of effects observed in psychological experiments (e.g., Cohen, Dunbar, & McClelland, 1990; Cohen, Servan-Schreiber, & McClelland, 1992; Holroyd & Coles, 2002; Rumelhart, McClelland, & the PDP Research Group, 1986; Spencer & Coles, 1999; Usher & McClelland, 2001; Yeung, Botvinick, & Cohen, 2004). Furthermore, such models provide a natural bridge between theories about cognitive processes and their implementation in the brain. However, they often involve many parameters that must be fit to the data to be explained.

To introduce the problem of parameterization, we first describe an example of a connectionist network that models the behavior of subjects in the Eriksen flanker task (Eriksen & Eriksen, 1974). In one version of the Eriksen task, subjects are presented with one of the following visual stimuli: "< < < < <," "> > > > >," "< < > < <," or "> > < > >," and have to indicate the direction of the middle arrow by pressing the left or the right button. Stimuli in which the direction of the "flanker" arrows is the same as that of the middle arrow are called *compatible*, and those in which the direction of the flankers is opposite that of the middle arrow are called *incompatible*. A number of interesting effects are observed with this paradigm. For example, incompatible stimuli generate more errors

and have longer correct response latencies than do compatible stimuli, whereas mean response latencies for error responses are shorter than those for correct ones. Cohen et al. (1992) proposed a connectionist model that simulated many of these behavioral effects. Here we consider a simplified version of that model (shown in Figure 1).

The model consists of two layers of units: a stimulus layer and a response layer. The stimulus layer includes three pairs of units. The middle pair represents information about the direction of the middle arrow in the stimulus. If, on a given simulated trial, the middle arrow is pointing left, the input to the left unit in the pair is set to 1 and the input to the right unit is set to 0, and vice versa if the middle arrow is pointing to the right. The two other pairs of units represent the direction of the left and right flanker arrows in the stimulus in a similar way (for simplicity, there is just one pair of units representing possible values of left flankers, and one pair of units representing right flankers). The box at the bottom of Figure 1 shows how each possible stimulus is represented by the values of the inputs. The middle units in the stimulus layer also receive input from a unit representing attentional bias (the unit with an exclamation mark in Figure 1), which is always set to 1.

Once the input is turned on, activation is allowed to flow among the units of the model, along the connections shown in Figure 1. The flow of activity between the units of the model is described by a set of stochastic differential equations. The model makes a response when the activity of one of the response units exceeds a decision threshold.

In psychological experiments, subjects repeat the task many times, and various descriptive statistics can be computed regarding subjects' performance, such as error rate, mean reaction time for correct and error trials, standard deviation of the reaction times, and so forth. We will refer to these descriptive statistics simply as *statistics*. Similarly, the connectionist model of a task can be executed many times, and the same statistics can be calculated for the model. Such a repeated execution of the model in order to
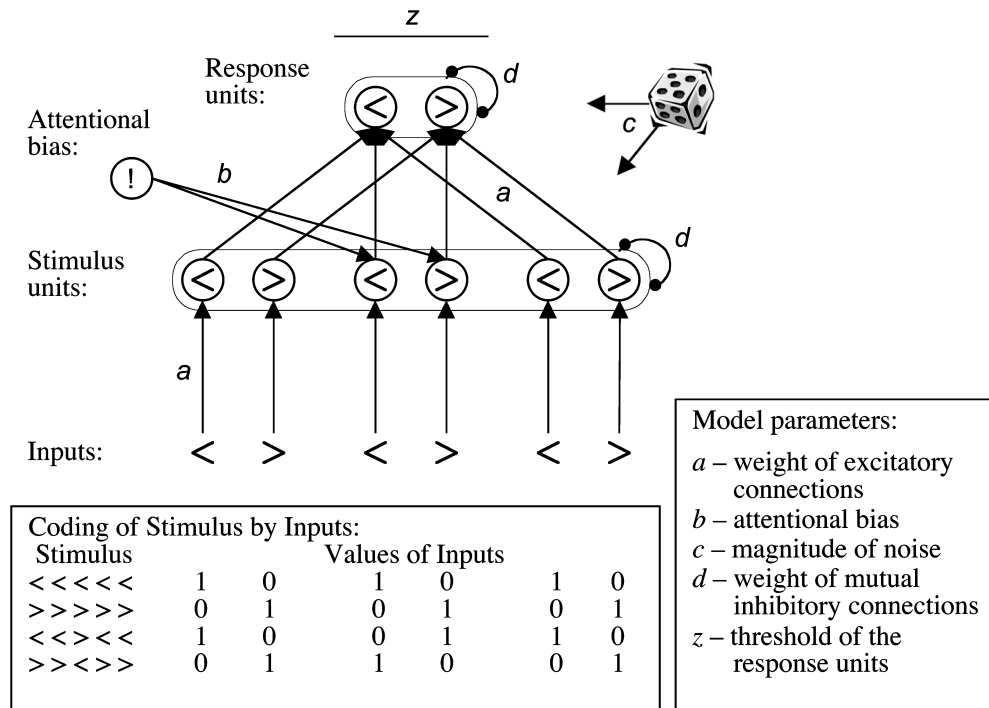
732

Figure 1. Simplified version of the Eriksen task (Eriksen & Eriksen, 1974) model (simplified from Cohen et al., 1992). Arrows denote excitatory connections, arches with circles at the end indicate that all the units in a given layer mutually inhibit one another. Box at the bottom left corner shows values of six inputs (each column corresponds to the input above) for four possible stimuli.

evaluate the statistics will be termed a *run* of the model in the remainder of this article. The behavior of the connectionist model, and thus the values of the model's statistics, are controlled by a set of parameters. For example, the network in Figure 1 is controlled by five parameters listed in the box at the bottom right corner of the figure.

Usually, in order to claim that a connectionist model can account for experimental data regarding behavior in a given task, researchers show that the statistics describing the model's performance have the same or very similar values to those for subjects in the experiment (e.g., Botvinick, Braver, Barch, Carter, & Cohen, 2001; Holroyd & Coles, 2002; Usher & McClelland, 2001). However, in order to show this, it is necessary to find the values of model parameters that result in the required behavior of the model.

Finding model parameters that produce a very close match between the model's performance statistics and human subjects' performance statistics is useful for several reasons. First, it shows that the model provides a quantitative account for the effects observed in the subjects' behavior. Second, if one tries to distinguish between two alternative models, one can parameterize both models as well as possible and then compare to what degree each matches the experimental data. Third, one can simulate additional experiments without changing the parameters to make further experimental predictions, and thereby test the generality of the model.

The parameterization of the model is usually done manually (i.e., researchers run models with different sets of parameters and search for the set resulting in the closest match). This can be extremely time consuming and may fail to yield the best set of parameters. With continued improvements in computational power, some investigators have begun to use automated procedures for parameterization of connectionist models (i.e., optimization algorithms to find parameters resulting in the best fit between the model and subjects' behavior). As yet, however, these have not appeared in published reports. Furthermore, insofar as these procedures typically have been designed for a particular model or task, they may not easily generalize to other models.

For example, Usher and McClelland (2001) automated parameterization of their two-choice model, but the optimization algorithm that they used was based on random search. Although this was appropriate for the simple two-choice model, it is likely to be too slow for larger models (i.e., models of more complex tasks with more parameters). Furthermore, they minimized a least-squares measure of difference between the model's and subjects' statistics similar to the one proposed in this article, but the weights assigned to individual statistics (and therefore the relative importance given to them) were chosen specifically for their model, and thus may not generalize to other models.

Ratcliff, Van Zandt, and McKoon (1999) automated parameterization of their diffusion model of two-choice tasks. They used the simplex optimization algorithm (Nedler & Mead, 1965)—a type of algorithm that we have also found useful for parameterization of connectionist models. Ratcliff et al. (1999) found that automatic parameterization was very useful: "it was only when the parameter space could be automatically searched . . . [that] . . . the model . . . fit the different patterns of error reaction times" (Ratcliff et al., 1999, p. 268). This gives hope that automatic parameterization procedures would similarly allow the finding of closer fits of other psychological models to experimental data.

Although there are well-described methods for optimizing certain parameters of connectionist networks (such as the back-propagation algorithm for setting connection strengths—e.g., Rumelhart, Hinton, & Williams, 1986), setting other parameters of the model (e.g., the level of noise on the input) is more challenging for several reasons. First, the analytic expressions describing how the model's statistics depend on the parameters are not fully described and can be very complex (Brown & Holmes, 2001). For example, although back-propagation can be used to find connection weights that minimize error, it has not been specified how to do so for a particular mean or distribution of reaction times.

Furthermore, connectionist models often incorporate noise, so that the statistics calculated during two different runs of the model with the same set of parameters will differ from each other. This demands that the model be run multiple times for each set of parameters to be tested, and it requires methods for comparing the statistics generated by one run with another.

This report describes an algorithm for parameterization of connectionist models that seeks to address the challenges outlined above. It can be used for any connectionist model implemented as a computer program. In this article, we focus on connectionist models, but the method can also be adapted to other psychological models that quantitatively describe behavior. In some cases, the algorithm can find the required parameters automatically. In other cases, it may help accelerate the process of manual parameterization. The algorithm employs a statistically motivated cost function, which indicates whether the model's statistics are significantly different from those obtained in the empirical experiment.

The algorithm has been implemented in Matlab and is free for download from the Psychonomic Society Web archive at http://www.psychonomic.org/archive/. It requires a user only to write a short Matlab script that executes the model and returns a vector containing the model's statistics as outputs. A user's manual for the program implementing the algorithm is contained in the appendix of the report that can be downloaded from the Web site above.

The algorithm is described in the parameterization algorithm section below. The case study section below demonstrates how the method performs in the parameterization of the sample connectionist model described above. Various issues related to use of the method are discussed in the discussion section below.

Parameterization Algorithm

The process of parameterization of connectionist models described in this report is based on least-squares estimation; that is, on minimization of a cost function that is sensitive to differences between the statistics describing the behavior of the model and those derived from empirical data concerning human subject performance. An optimization algorithm is used to find the values of the model parameters for which the value of the cost function is minimized.

The pseudocode of the algorithm is given in the Listing. The procedure consists of three phases: (1) finding the starting point of optimization by a random search, (2) finding parameters minimizing the cost function with the optimization algorithm, and (3) tuning parameters to minimize the cost function with statistics weighted by their variability. The details of these three steps in the algorithm are described in the next three subsections, followed by subsections that consider additional issues associated with these steps.

Finding a starting point of optimization. The starting point of optimization may either be specified by the user or established with random search. In the second case, the user specifies the number of search iterations S, and values p_i for each parameter i defining the range of search. Specifically, each of S sets of parameters is generated randomly, such that each parameter i is chosen as a random number from range $[0, 2p_i]$ (the user can specify the minimum and maximum values). For each set of parameters, the model is executed a number of times, corresponding to the number of trials in the experiment from which statistics are being fit, and the same statistics are calculated for the model. After each run, the cost function is evaluated. The set of parameters that resulted in the lowest value of the cost function is taken as the starting point of optimization.

Cost function. Let us denote the statistics of the model by m_i and the statistics obtained from the experiment by e_i and the number of statistics being fit by N. The cost function describes the degree to which m_i differs from e_i:

$$cost = \sum_{i=1}^{N} \left(\frac{e_i - m_i}{n_i} \right)^2. \qquad (1)$$

In Equation 1, n_i denotes the normalization factor for statistic i. This must be introduced to ensure that each statistic contributes equally to the cost function, because different statistics may have values that differ in scale (e.g., error rate may be equal to 0.1 and reaction time to 400). A natural choice for the normalization factor might be $n_i = e_i$, but we found that a somewhat different set of values is superior. As we will describe below, the method worked best when we used two different normalization factors: one for the first two phases (finding a starting

LISTING
Pseudocode of the Parameterization Algorithm

```
/*Finding starting point */
Repeat specified number of times
        Generate a random set of parameters
        Run the model for above parameters
        Compute the cost function
        If cost function the lowest so far
                Starting point = this set of parameters

/* Optimization of parameters */
Current parameters = Starting point
Repeat specified number of times
        Run the model for current parameters
        Compute the cost function
        Let optimization algorithm choose new parameters

/* Tuning of parameters */
Run model 10 times and compute standard deviation of statistics
Repeat specified number of times
        Run the model for current parameters
        Compute the normalized cost function
        Let optimization algorithm choose new parameters
Run model 10 times and test fit statistically
```

point and optimization of parameters) and another for the third phase (tuning of parameters).

If a simple normalization $n_i = e_i$ were used, statistics that have values very close to zero (e.g., an error rate for a given condition may be smaller than 1%) could contribute to the cost function much more than others because for them the denominator of Equation 1 would be very close to 0. To avoid this problem, during the two first phases (finding a starting point and optimizing parameters), the normalization factors n_i of the cost function are assigned in the following way. The user provides information for each statistic about whether it is related to error rates, reaction times, or variations in reaction times (information on the types of statistics is also given in the section on matching time units below). For each type of statistic, the average value of the empirical statistics e_i belonging to that type is calculated. The normalization factor for a given statistic is taken as the average goal statistic for its type. For example, if statistic i is of type *error rate*, the normalization factor n_i is equal to the average value of all statistics of type *error rate*. This choice was driven by the fact that the statistics belonging to the same type usually have values that are similar in scale.

Tuning of parameters. Although the method for calculating n_i described above works well for finding a starting point and the first round of optimization (Phases 1 and 2), a different method based on the statistic's variability across runs is used for tuning parameters based on a differential weighting of statistics (Phase 3). This is because some parameters are associated with statistics that vary widely from run to run, even when these parameters are held constant, whereas other parameters are associated with statistics that are more stable (i.e., have very similar values across different runs of the model with the same parameters). Therefore, for final tuning, it is best to focus on parameters associated with statistics that are stable rather than ones that differ substantially from run to run. Such emphasis can be achieved by setting the normalization factors n_i of the cost function to the standard deviations of the statistics of the model.

For these reasons, at the beginning of the third phase (tuning of parameters), the model is run 10 times for the parameters found in the second phase. For each of the statistics m_i of the model, the standard deviation across the 10 runs is calculated, and it is taken as the normalization factor for this statistic n_i in the cost function during parameter tuning in the third phase.

Normalization factors are not set to the standard deviation in the initial two phases because during the optimization in the second phase, the solution moves substantially in parameter space. As a result, the cost function is evaluated for very different values of parameters, so the model statistics may have very different values and standard deviations within the second phase.

Optimization algorithm. The same optimization algorithm is used in the second (optimization) and third (tuning) phases. We compared the performance of different optimization algorithms (including standard Matlab routines) in parameterization of a simple connectionist model. The best performance was achieved by an algorithm called subplex (Rowan, 1990). In this article, we do not describe this comparison (details may be found in the report attached to the parameterization code available for download under the address given in the introduction). Below, we briefly describe the subplex algorithm, its advantages over other algorithms, and our choice of parameters for subplex.

Subplex is a modification of the simplex algorithm (Nedler & Mead, 1965). A simplex in n-dimensional parameter space is a set of $n + 1$ points that bound part of

that space. For example, on a plane (i.e., a two-dimensional space, where the two dimensions correspond to values of the two parameters being estimated), a simplex is a triangle. The simplex defines the region in which the algorithm looks for the minimum of the cost function. The simplex algorithm starts by defining a simplex in the parameter space of a given size centered on a given point (i.e., the starting point of optimization). It evaluates the cost function for every point within the simplex (by running the model). Then, in the following steps, the set of parameters (i.e., a point of the simplex) that results in the highest value of the cost function is replaced by a new set of parameters (chosen on the basis of the values of the cost function in the current vertices of the simplex; for details, see Nedler & Mead, 1965), and the cost function for it is evaluated anew.

The subplex method (Rowan, 1990) was designed to generalize the simplex algorithm to better handle noisy functions—that is, functions that may have different values for the same set of parameters. For this reason, the subplex algorithm seems well suited for parameterizing connectionist models, which typically incorporate noise into processing. Subplex works by decomposing high-dimensioned problems into low-dimensioned subspaces that are easily handled by the simplex method (for details, see Rowan, 1990).

We also tested gradient-based optimization algorithms, but they seldom found parameters resulting in low values of the cost function. They performed poorly because they were not well suited to the noisiness of the cost function. For example, they tried to calculate the gradient at the starting point of optimization numerically by running the model for the values of the parameters in different directions in the parameter space. However, this estimate of the gradient was imprecise because the cost function is noisy, and the optimization algorithms did not take this into account. It is possible that further refinements of gradient-descent algorithms could address these issues (e.g., Kelley, 1999), and this remains an interesting challenge for future research.

An important constraint for the optimization algorithm is that it must be fast; that is, it must find the minimum of the cost function with as few runs of the model (i.e., sampling the parameter space) as possible. This feature is critical because one run of a complex connectionist model may take a significant period of time. Therefore, we did not test optimization algorithms that involve extensive random search, such as genetic algorithms and simulated annealing.

The initial size of the simplex in dimension i of the parameter space is taken as 30% of the starting value of parameter p_i in the second (optimization) phase, and 15% of p_i in the third (tuning) phase (these values were determined empirically). The number of model runs allowed to be executed by the subplex algorithm in the second and third phases is specified by the user.

Matching time units of model and experiment. Statistics describing the reaction time of the model are ex-

pressed in time units of the model (i.e., number of iterations of updating units' activations required for a decision unit to cross a given threshold), whereas in the experiment they are expressed in milliseconds. In some models, the relation between model time and real time is an explicit assumption of the model (e.g., it is assumed that one step of the model corresponds to 50 msec; Anderson, 1993). However, in most connectionist models such explicit assumptions are not made, and the reaction time statistics must be converted post hoc from model time units to milliseconds. For our present purposes, this conversion must be performed after each run of the model (i.e., repeated execution of the model in order to compute statistics) before the cost function can be evaluated, because the relationship may depend on the specific parameters used. Furthermore, the cost function includes terms expressing the difference between reaction times of the model and the experiment. In order for this to be meaningful, they need to be in the same units. This conversion is described below.

It is usually assumed that a connectionist model does not capture all information processing during a task (e.g., it may not capture early visual processing or motor execution). This can be accommodated in the following regression equation, which expresses the relationship between the reaction time in model units (RT_{units}; i.e., the number of iterations until the response is made) and experimentally observed reaction time in milliseconds RT_{msec}:

$$RT_{msec} = a \, RT_{units} + b. \qquad (2)$$

In Equation 2, slope a denotes how many milliseconds correspond to one time unit of the model, and intercept b denotes the duration of processing not captured by the model (in milliseconds; we assume for simplicity that it is constant across trials and conditions). Our goal is to find the values of a and b, minimizing the differences between the model statistics converted to milliseconds and the statistics describing human behavior in the psychological experiment. Thus a and b are "hidden" parameters that can improve the fit between the model and human data during optimization. We can find the values of a and b by performing a standard linear regression.

Some statistics of the model may correspond to differences between reaction times in different experimental conditions or standard deviations of reaction times. The relationship between such statistics in model units V_{units} and in milliseconds V_{msec} can be expressed by the following equation:

$$V_{msec} = aV_{units}. \qquad (3)$$

In Equation 3, the same conversion factor a is used as in Equation 2, but there is no intercept b because these statistics are differences between reaction times, and thus the period of processing not captured by the model is subtracted (since it is assumed to be constant across trials and conditions).

If reaction times assume an ex-Gaussian distribution (Cousineau & Larochelle, 1997; Ratcliff, 1978) with

mean μ and standard deviation σ of the Gaussian component, and exponential time constant τ, the relation between the statistic μ in model units and in milliseconds is expressed by Equation 2, and the relationship of the statistics σ and τ in model units to milliseconds is expressed by Equation 3. For simplicity, we will refer to all statistics whose relation of units is described by Equation 3 as statistics concerning variations of reaction times.

To match statistics that involve both reaction times and variations of reaction times (i.e., to ensure that they are constrained by the same value of regression parameter a), we choose the values of a and b by the following extended version of the regression. First, the statistics from the experiment concerning reaction times are denoted by RT_{exp_i}, statistics concerning variations of reaction times by V_{exp_i}, and the numbers of these statistics being matched by N_{RT} and N_V, respectively. We then choose the values of a and b that minimize the following function:

$$E = \sum_{i=1}^{N_{RT}} \left(a\,RT_{units_i} + b - RT_{exp_i} \right)^2 + \sum_{j=1}^{N_V} \left(a\,V_{units_j} - V_{exp_j} \right)^2. \quad (4)$$

In order to find the minimum for Equation 4, we have to find values of a and b for which derivatives of E are equal to 0:

$$\begin{cases} \dfrac{\partial E}{\partial a} = 0 \\[6pt] \dfrac{\partial E}{\partial b} = 0 \end{cases}. \quad (5)$$

Solving the set of Equation 5, the expressions at the bottom of this page [(6) and (7)] are obtained. Note that when we do not fit statistics concerning variations of reaction times (i.e., $N_V = 0$), Equations 6 and 7 reduce to the standard expression for regression coefficients.

If we consider fits of the model to experimental data more generally, each statistic can be assigned to one of three types: (1) those that do not require the change of units described in this section (e.g., error rates), (2) those being fit by regression with intercept (e.g., reaction times), and (3) those being fit by regression without intercept (e.g., variations of reaction times). After every run of the model, the regression coefficients are calculated accord-

ing to Equations 6 and 7. If any of the coefficients is negative, it is made equal to 0 (negative values of a and b do not make sense; they are not likely to occur for parameters found during optimization, but may occur while finding the starting point of optimization when parameters are chosen randomly). Then the model statistics involving reaction times and variations of reaction times are converted to milliseconds according to Equations 2 and 3. After the conversion, the cost function is calculated.

Statistical testing of model fit. At the end of the entire parameterization procedure, the model is run an additional 10 times to evaluate the statistical significance of the final solution. The mean values of the model statistics and their standard deviations are calculated and used to compute the final value of the cost function.

This final value of the cost function has a statistical interpretation—namely, the expression

$$z_i = \frac{e_i - \overline{m}_i}{\tilde{m}_i} \quad (8)$$

(where $\overline{m}_i$ denotes the mean model statistics i, and $\tilde{m}_i$ denotes the standard deviation of the model statistics over 10 runs) indicates how far the distribution of the values of statistics generated in different runs of the model for the final solution deviate from their goal values. If e_i and m_i come from the same distribution, then z_i has approximately a standard normal distribution (it would be normal if $\overline{m}_i$ and $\tilde{m}_i$ were the true mean and standard deviation instead of estimates). Thus, the final value of the cost function has approximately a χ^2 distribution with N degrees of freedom. However, the precise distribution of the cost function is slightly different, as is derived in the Appendix. Since the cumulative distribution F of the cost function is known, one can test whether the model and experimental statistics are significantly different. In particular, the significance of the difference between the experimental and model statistics is equal to $p = 1 - F(cost)$. The significance p is also calculated by the tool.

There are more precise methods to determine whether the model statistics differ from those of the experiment (e.g., to compare mean reaction times in a certain condition, one can use a t test for reaction times of all trials of

$$a = \frac{\displaystyle\sum_{i=1}^{N_{RT}} RT_{units_i} RT_{exp_i} + \sum_{j=1}^{N_V} V_{units_j} V_{exp_j} - \frac{1}{N_{RT}} \sum_{i=1}^{N_{RT}} RT_{units_i} \sum_{i=1}^{N_{RT}} RT_{exp_i}}{\displaystyle\sum_{i=1}^{N_{RT}} \left(RT_{units_i} \right)^2 + \sum_{j=1}^{N_V} \left(V_{units_j} \right)^2 - \frac{1}{N_{RT}} \left(\sum_{i=1}^{N_{RT}} RT_{units_i} \right)^2} \quad (6)$$

and

$$b = \frac{\displaystyle\sum_{i=1}^{N_{RT}} RT_{exp_i} - a \sum_{i=1}^{N_{RT}} RT_{units_i}}{N_{RT}} \quad (7)$$

Table 1
Comparison of the Values of the Statistics Describing the Behavior of Subjects in Yeung et al.'s (2004) Experiment and the Simplified Connectionist Model for the Best Solution Found by the Parameterization Algorithm

Statistics	Experiment	Model
Error rate for compatible stimuli	1.9%	1.3%
Error rate for incompatible stimuli	18.7%	19.1%
Mean reaction time for compatible stimuli (msec)	351.3	352.2
Mean reaction time for incompatible stimuli (msec)	403.2	402.3
Difference between mean reaction times for correct and incorrect responses	67.4	66.5
Standard deviation of reaction times for compatible stimuli	73.9	77.3
Standard deviation of reaction times for incompatible stimuli	100.9	99.3

the experiment and the model within one run). However, it is a useful property that the cost function can provide a first indication of the degree of fit.

This section has described a single session of parameterization. Since the optimization process may find only a local minimum of the cost function, the optimization sessions may be repeated. The number of such repetitions can be specified by the user.

Case Study

To evaluate how our parameterization procedure performs on an actual model, we have used it to parameterize a connectionist model of the Eriksen flanker task (Eriksen & Eriksen, 1974) described in the introduction. Recently, Yeung et al. (2004) conducted an empirical study using this task. We tried to fit the model to the values of the statistics, describing the mean performance of subjects in that experiment.

Seven statistics describing performance of the model were fit to the experimental data; they are listed in the left column of Table 1. These statistics describe the main effects observed in the Eriksen task (i.e., the difference in error rate and reaction times between compatible and incompatible trials, and the difference in reaction times between correct and error trials; see the introduction for details). Note that since the number of parameters is fewer than the number of statistics being fit, there is not a guaranteed solution.

During a single run, the model was executed for 13,056 trials to generate the statistics to be compared with the data from human subjects for a given set of parameter values (13,056 is the number of stimuli across subjects used in the Yeung et al., 2004 experiment). The order of stimuli was also the same in the simulations as in the experiment. During each optimization session, the model was run 50 times in the first phase (searching for a starting point for optimization), 150 times in the second phase (optimization), and 100 times in the third phase (tuning). The entire optimization session was repeated 200 times, which took about one day of computation using a computer with a Pentium II processor (processor speed: 450 MHz, RAM: 128 MB).

The value of the cost function for the best solution found by the parameterization procedure was 27.6. On the basis of the value of the cost function, the statistics of the model differ significantly from those of the ex-

periment with $p = .033$. Although these differences are significant, they are not large: The values of the individual statistics are compared in Table 1. One source of these differences may be our approach of fitting a single model to data averaged across subjects, thus ignoring variability between subjects.

The use of this optimization procedure may also indicate problems with the model and thus help in improving its design. For example, we attempted to simplify the model of the Eriksen task from its original form as much as possible (Cohen et al., 1992), and beyond the form shown in Figure 1, by removing the right flanker units. Thus, all flankers were represented by a single pair of units (note that the number of model parameters did not change). We were unable to parameterize this model satisfactorily; in particular, we were not able to achieve reaction times on error trials that were shorter than those on correct trials, as was the case for the empirical data. This suggests that the model with just one pair of flanker units is a less adequate model of the Eriksen task than the model with two pairs of flanker units.

Discussion

This article describes an algorithm for parameterization of connectionist models and an example of its application to a particular model. The method can be used for parameterization of any connectionist model implemented as a computer program. The optimization procedure uses a statistically motivated cost function that indicates whether the statistics of the model differ significantly from those obtained in the experiment.

Since the algorithm may find a local minimum of the cost function, it is important to perform a suitable number of optimization sessions. For example, in the simulations described above, the parameterization procedure yielded sets of parameters resulting in a low value of the cost function (i.e., resulting in $p > .01$) in only about 10% of the sessions.

We have noticed that the quality of the solution found by the algorithm strongly depends on the initial range of parameters p_i from which starting points of optimization are taken. If the method cannot find good solutions, it may simply be searching in the wrong part of the parameter space and converging to local minima. Therefore, in such situations, it may be worth trying to run the

model with different parameters to determine how they affect the behavior of the model. This may help one choose the appropriate range of parameters p_i from which starting points of optimization are taken. Of course, such manual exploration of parameters may also lend insight about the functioning of the mechanisms of the model, and the critical goal of model building in the first place.

When a run of a connectionist model takes a long time, it may take many days for our procedure to parameterize the model. In this case, it is worth trying to speed the computer program implementing the model (e.g., if the model is implemented in Matlab, it could be translated into C). In order to find an approximation of parameters, one can modify how the model is executed, reducing the execution time. For example, one can try to decrease the number of model executions in one run, or if the model is described by continuous differential equations, one can also try to make its discrete approximation coarser. Although this may reduce the precision of the estimate of the model's statistics, or make them noisier, the result may provide useful guidance in the choice of a starting point for optimization of the full but slowly executing model.

The cost function used in the algorithm has statistical meaning, and one may be tempted to use it to compare different models. For example, if two models of a given task are compared, and the parameters of the first model result in a lower value of the cost function than the second, one could claim that the first model fits the data better than the second. However, it should be remembered that the value of the cost function depends on the number of simulated trials in a run, because the number of trials influences the normalization factors n_i of the cost function. Namely, n_i are equal to the standard deviations of the statistics across trials for a given set of parameters, and the more simulated trials, the more similar are the values of statistics across trials. Hence, if the cost function is used to compare the models, the models must be executed with the same number of simulated trials per run.

The proposed method may sometimes be too powerful, in the sense that it may find parameters for two very different models, both of which match the experimental data equally well. Therefore, the fact that a model may fit the empirical data should not be treated as proof that the model is a correct abstraction of the information processing (or neural) mechanisms it is meant to simulate.

In the simulations described above, we fit the model to mean experimental statistics averaged across subjects. But one could also use the algorithm to fit the model to the behavior of individual subjects, and then explore individual differences between subjects with the model. These methods can be also used to fit neurophysiological statistics such as the amplitude or latency of ERPs or hemodynamic response measurements (such as fMRI).

The parameterization method could be further developed in a number of directions. First, the subplex optimization algorithm has a number of parameters (e.g., initial size of the simplex), which we chose on the basis of our experience. However, the optimal values of these parameters may differ for different models being optimized; hence one could investigate how to optimally set up the values of parameters of optimization. Second, we assume for simplicity that the statistics being fit are independent. However, such independence may not always be a correct assumption. For example, the reaction time in two different conditions may be correlated across subjects. If the statistics are correlated, the least-squares method should use normalization by covariance matrix[1] rather than by standard deviations, as proposed in this article (Koch, 1988, p. 180). Normalization by the covariance matrix should be explored in the future versions of the method we have proposed.

Recently, Ratcliff and Tuerlinckx (2002) analyzed the estimation of parameters for diffusion models (related to connectionist models), and they pointed out that it is crucial that the parameter estimation method be robust (i.e., insensitive to trials in which subjects' responses are not captured by the model, as occurs with very long reaction times). They noticed that their weighted least-squares method, minimizing a cost function similar to the one described here, is more robust than the standard maximum-likelihood approach.

Ratcliff and Tuerlinckx (2002) also proposed a useful framework for comparing the quality of different parameterization methods. Instead of fitting a model directly to experimental data, they fitted it to the data obtained from the simulation of the model with known parameters p_i and assessed how the fitted model parameters m_i match the original parameters p_i. It is likely that in the future, new algorithms for parameterization of psychological models will be developed. For example, Brown, Reynolds, and Braver (2003) are currently designing a tool called RT++ for parameterization of connectionist models developed under software PDP++. As such tools become available, it will become increasingly important to compare their ability to accurately and efficiently estimate the parameters of psychological models, perhaps within the framework suggested by Ratcliff and Tuerlinckx (2002).

REFERENCES

ANDERSON, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.
BOTVINICK, M. M., BRAVER, T. S., BARCH, D. M., CARTER, C. S., & COHEN, J. D. (2001). Conflict monitoring and cognitive control. *Psychological Review*, **108**, 624-652.
BROWN, E., & HOLMES, P. (2001). Modeling a simple choice task: Stochastic dynamics of mutually inhibitory neural groups. *Stochastics & Dynamics*, **1**, 159-191.
BROWN, J. W., REYNOLDS, R. R., & BRAVER, T. S. (2003). *A computational neural model of anterior cingulate cortex in response- and task-switching*. Poster presented during the Cognitive Neuroscience meeting, New York.
COHEN, J. D., DUNBAR, K., & MCCLELLAND, J. L. (1990). On the control of automatic processes. *Psychological Review*, **97**, 332-361.
COHEN, J. D., SERVAN-SCHREIBER, D., & MCCLELLAND, J. L. (1992). A parallel distributed processing approach to automaticity. *American Journal of Psychology*, **105**, 239-269.
COUSINEAU, D., & LAROCHELLE, S. (1997). PASTIS: A program for curve and distribution analyses. *Behavior Research Methods, Instruments, & Computers*, **29**, 542-548.

Eriksen, B. A., & Eriksen, C. W. (1974). Effects of noise letters upon the identification of a target letter in a nonsearch task. *Perception & Psychophysics*, **16**, 143-149.

Holroyd, C. B., & Coles, M. G. H. (2002). The neural basis of human error processing: Reinforcement learning, dopamine, and the error-related negativity. *Psychological Review*, **109**, 679-709.

Kelley, C. T. (1999). *Iterative methods for optimization* (Frontiers in Applied Mathematics, No. 18). Philadelphia: SIAM.

Koch, K. R. (1988). *Parameter estimation and hypothesis testing in linear models*. Heidelberg: Springer-Verlag.

Nedler, J. A., & Mead, R. (1965). A simple method for function minimization. *Computer Journal*, **7**, 308-313.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, **85**, 59-108.

Ratcliff, R., & Tuerlinckx, F. (2002). Estimating parameters of the diffusion model: Approaches to dealing with contaminant reaction times and parameter variability. *Psychonomic Bulletin & Review*, **9**, 438-481.

Ratcliff, R., Van Zandt, T., & McKoon, G. (1999). Connectionist and diffusion models of reaction time. *Psychological Review*, **106**, 261-300.

Rowan, T. (1990). *Functional stability analysis of numerical algorithms*. Unpublished doctoral dissertation, University of Texas at Austin.

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

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*. Cambridge, MA: MIT Press.

Spencer, K. M., & Coles, M. G. H. (1999). The lateralized readiness potential: Relationship between human data and activation in a connectionist model. *Psychophysiology*, **36**, 364-370.

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

Yeung, N., Botvinick, M. M., & Cohen, J. D. (2004). The neural basis of error detection: Conflict monitoring and the error-related negativity. *Psychological Review*, **111**, 931-959.

NOTE

1. Normalization by covariance matrix would yield the following cost function: $cost = (\mathbf{e} - \mathbf{m})^T \mathbf{C}^{-1}(\mathbf{e} - \mathbf{m})$, where $\mathbf{e}$ and $\mathbf{m}$ are the vectors of experimental and model statistics, and $\mathbf{C}$ is the covariance matrix.

ARCHIVED MATERIALS

The following materials associated with this article are retrievable from the Psychonomic Society's Norms, Stimuli, and Data archive at http://www.psychonomic.org/archive/.

To access the above files or links, search the archive for this article using the journal (*Behavior Research Methods, Instruments, & Computers*), the first author's name (Bogacz) and the publication year (2004).

File: Bogacz-BRMIC-2004.zip

Description: Compressed file including: fitparam.m, parameterization procedure for Matlab; report.pdf, a technical report including a user manual for the parameterization software; directory example, containing the example of parameterization of the Eriksen flanker model described in Section 3, README, description of all other files in the directory.

Author's e-mail address: r.bogacz@bristol.ac.uk.

Author's Web site: http://www.cs.bris.ac.uk/home/rafal/.

APPENDIX

This appendix derives the distribution of the cost function. $\bar{\bar{m}}_i$ in Equation 8 denotes the estimate of the standard deviation normalized by 10 (i.e., the number of observations n) rather than 9 (i.e., $n - 1$). Let us denote by μ_i and σ_i the true mean and standard deviation of model statistics m_i. Hence, Equation 8 may be rewritten as:

$$z_i = \frac{e_i - \bar{m}_i}{\bar{\bar{m}}_i} = \frac{\sigma_i}{\sigma_i}\left(\frac{e_i - \bar{m}_i - \mu_i + \mu_i}{\bar{\bar{m}}_i}\right)$$

$$= \frac{\sqrt{10}}{\sqrt{\dfrac{10\bar{\bar{m}}_i^2}{\sigma_i^2}}}\left(\frac{e_i - \mu_i}{\sigma} - \frac{\bar{m}_i - \mu_i}{\sigma}\right). \tag{9}$$

Three of the terms in Equation 9 have known distributions:

$$z_i = \frac{\sqrt{10}}{\sqrt{\chi^2(9)}}\left[N(0,1) - \frac{N(0,1)}{\sqrt{10}}\right] =$$

$$= \frac{\sqrt{10}}{\sqrt{9}\sqrt{\dfrac{\chi^2(9)}{9}}}\sqrt{\frac{11}{10}}N(0,1) = \sqrt{\frac{11}{9}}\frac{N(0,1)}{\sqrt{\dfrac{\chi^2(9)}{9}}}. \tag{10}$$

Hence, from the definition of F distribution:

$$\frac{9}{11}z_i^2 = F(1,9). \tag{11}$$

Thus, the final value of the cost function has the following distribution:

$$\frac{9}{11}cost = \sum_{i=1}^{N}F(1,9). \tag{12}$$

The distribution of the sum of F distributions may be calculated numerically with the equation for the distribution of the sums of distributions:

$$f_{X+Y}(z) = \int_{-\infty}^{\infty} f_X(z-u)f_Y(u)du. \tag{13}$$

The tool calculates numerically the cumulative distribution of the cost function and performs the test described in the section on statistical testing of model fit.