
Using MCMC to Compare Network Models in Cognitive Science

Woojae Kim, Daniel J. Navarro, Mark A. Pitt, In Jae Myung

Department of Psychology

Ohio State University

{kim.1124,navarro.20,pitt.2,myung.1}@osu.edu

Abstract

Despite the popularity of network models in cognitive science, their performance can often be difficult to evaluate. Inspired by the geometric approach to statistical model selection, we introduce a conceptually simple method to examine the global behavior of a network model, by counting the number and type of response patterns it can simulate. We describe an MCMC algorithm constructed to find these patterns, and demonstrate the approach with regard to two classic models of the phonemic perception.

1 Introduction

Neural network models are popular in some areas of cognitive science, especially language processing. One reason for this is that they provide a means of instantiating the fundamental principles of a theory into a readily testable computational form. For example, in localist networks, levels of mental representation are mapped onto layers of nodes in a network. Information flow between levels is then defined by the types of connection (e.g., excitatory and inhibitory) between layers. The soundness of the theoretical assumptions are then evaluated by studying the behavior of the network in simulations and testing its predictions experimentally.

Although such modeling has enriched our understanding of human cognition, the consequences of the choices made in the design of a model can be difficult to evaluate. While good simulation performance is assumed to support the model and its underlying principles, a drawback of this testing methodology is that little attention is paid to a model's complexity or the reasons why a competing model might simulate human data equally well.

These concerns are part and parcel of the well-known problem of model selection. A great deal of progress has been made in solving it for statistical models (i.e., those that can be described by a family of probability distributions; Rissanen, 1996, 2001). Neural networks, however, are a computationally different beast. The current paper introduces a technique that can be used to assist in evaluating and choosing between network models of cognition.

2 On the Complexity of Network Models

The ability of a network model to simulate human performance well should not be taken as evidence that the network architecture is a good approximation of the human cognitive system that generated the data. For instance, it would be unimpressive if it turned out that the model could also simulate many non-human-like patterns. Accordingly, we need a “global” view of the network’s behavior, to discover all of the data patterns the it can produce.

A model’s ability to reproduce diverse patterns of data describes its complexity. Complexity is an intrinsic property of a model that arises from the interaction of its parameters and functional form. For statistical models, it can be calculated by integrating the determinant of the Fisher information matrix over the parameter space of the model. Although derived by Rissanen (1996) as a codelength, the measure is sometimes called the geometric complexity, because it is equal to the logarithm of the ratio of two Riemannian volumes. Viewed from this geometric perspective, the measure has an elegant interpretation as a count of the number of “distinguishable” distributions that a model can generate (Balasubramanian, 1997; Myung, Balasubramanian & Pitt, 2000). Unfortunately, geometric complexity cannot be applied to network models, since these models rarely possess a likelihood function, much less a well-defined Fisher information matrix. Also, in many cases a learning (i.e., model-fitting) algorithm for finding optimal parameter values is not proposed along with the model, further complicating matters.

A conceptually simple solution to the problem, albeit a computationally demanding one, is to first discretize the data space in some properly defined sense and then identify all of the data patterns a network model can generate. This approach provides the desired global view of the model’s capabilities and its definition resembles that of geometric complexity: The complexity of a network model is defined as the logarithm of the number of data patterns the model can produce. As such, this reparametrization-invariant complexity measure can be used for virtually all types of network models provided that the discretization of the data space is both justifiable and meaningful.

A challenge in implementing this solution lies in the enormity of the search space, which may contain a truly astronomical number of data patterns. Only a small fraction of these might correspond to the model’s predictions, so it is essential to use an efficient search algorithm, one that will find most or all of these patterns in a reasonable time. Inspired by two localist models of speech perception, we developed an algorithm that uses Markov Chain Monte Carlo (MCMC) to solve such problems. The algorithm is tailored to exploit the kinds of search spaces that we suspect are typical of cognitive models, and we evaluate its performance in this context.

3 Localist Models of Phoneme Perception

A central issue in the field of human speech perception is how prior knowledge of words influences the perception of speech. That is, how does knowing the word you are hearing influence how you hear the smaller units that make up the word (i.e., its phonemes)? Two localist models have been proposed that represent opposing theoretical positions. Both models were motivated by different interpretations of the same set of experimental results. Proponents of TRACE (McClelland and Elman 1986) argue for bi-directional communication between network layers whereas proponents of MERGE (Norris, McQueen and Cutler, 2000) argue against it. The models are shown schematically in Figure 1. Each contains two main layers. Phonemes are represented in the first layer and words in the second. Activation flows from the

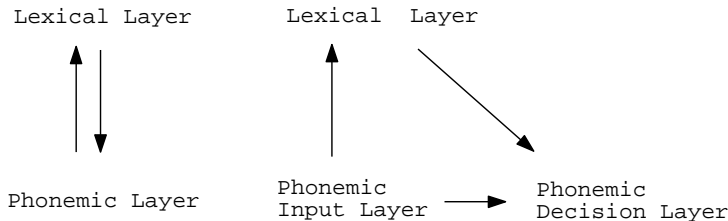


Figure 1: Network architecture for TRACE (left) and MERGE (right).

first to the second layer in both models. At the heart of the controversy is whether activation also flows in the reverse direction, directly affecting how the phonemic input is processed. In TRACE it can. In MERGE it cannot. Instead, the processing performed at the phonemic level in MERGE is split in two, with an input stage and a phonemic decision stage. The second, lexical layer cannot directly affect phoneme activation. Instead, the two sources of information (phoneme and lexical) are integrated only at the phoneme decision stage.

Although the precise details are unnecessary for the purposes of this paper, it will be useful to sketch out a few technical details of the models. (List the parameters, what they do, and highlight nonlinear dynamics. Response defined as the state after k iterations, details later). A parameter set θ was only considered valid if the final state satisfied certain decision criteria, listed in Table 1.

Despite the differences in motivation, these models are comparable in their ability to simulate key experimental findings in the literature (Norris et al 2000). When these results are considered, it may seem quite challenging if not impossible to distinguish between them experimentally. Yet surely the models are not identical? What are the functional differences between the two? Is one more complex than the other?

In order to address these questions, we consider a situation in which the two models had previously been shown to simulate human data similarly well (Norris et al., 2000). In the experiments, monosyllabic words were presented in which the last phoneme from one word was replaced by one from another word (through digital editing) to create blends that formed words and nonsense words. The six types of blends are listed on the left of Table 2. Listeners had to categorize the last phoneme in one task (phonemic decision) and categorize the entire utterance as a word or a nonsense word in the other task (lexical decision). The response choices in each task are listed on the right. Three response choices were used in lexical decision to test the models' ability to distinguish between words, not just words and nonwords. The asterisks in each cell indicate the responses that listeners chose most often. Both TRACE and MERGE can simulate this pattern of responses.

The profile of responses decisions (phonemic and lexical) over the six experimental

Table 1: Constraints.

	Choose /b/ if...	Choose job if...	Choose "nonword" if...
Weak	/b/ > 0.4 & others < 0.4	job > 0.4 & jog < 0.4	both < 0.4
Strong	/b/ > 0.45 & others < 0.25 /b/ - max(others) > 0.3	job > 0.45 & jog < 0.25 (job - jog) > 0.3	both < 0.25 abs(difference) < 0.15

Table 2: Experimental design. Asterisks denote human responses.

Condition	Example	Phonemic Decision				Lexical Decision		
		/b/	/g/	/z/	/v/	job	jog	non-word
$w_1 w_1$	JO b + jo B		*			*		
$w_2 w_1$	JO g + jo B		*			*		
$n_2 w_1$	JO v + jo B		*			*		
$n_1 n_1$	JO z + jo Z				*		*	
$w_2 n_1$	JO g + jo Z				*		*	
$n_2 n_1$	JO v + jo Z				*		*	

conditions provides a natural definition of a data pattern that the model could produce. Accordingly, there are $4^6 \times 3^6 = 2,985,984$ possible data patterns. In the current test, we are interested in determining how many patterns (besides the human-like pattern each model can generate. As previously discussed, these counts will serve as a measure of model complexity.

4 The Search Algorithm

The search problem that we need to solve differs slightly from the standard Monte Carlo counting problem. Ordinarily it is used to discover how much of the search space is covered by some region, which is solved by counting how often co-ordinates are discovered that belong to the region. In our problem, a high-dimensional parameter space has been partitioned into an unknown number of regions, with each region corresponding to a single data pattern. Our aim is to find all such regions, irrespective of their size. How do we solve this problem? Given the dimensionality of the space, brute force searches are impossible. Previous experience suggested that the Simple Monte Carlo (SMC) approach also fails, because it does not capitalize on the structure of the search space.

The types of spaces that we consider appear to possess at least three regularities, illustrated schematically in Figure 2. Firstly, on many occasions, the network does not converge on a state that meets the decision criteria, so some proportion of the space does not correspond to any data pattern at all. Secondly, the size of the regions vary a great deal. Some data patterns are elicited by a wide range of parameter values, whereas others can be produced only by a small number of values. Thirdly, small regions tend to cluster together. In these models, there are likely to be regions where the model consistently chooses the dominant phoneme and makes the correspondingly appropriate lexical decision. However, there will also be large regions in which the models always choose “nonword” irrespective of whether the stimulus is a word. Along the borders between regions, however, there might be lots of smaller “transition regions”, and these regions will tend to be near one another.

The consequence of this structure is that the size of the region in which the process is currently located provides extensive information about the size of nearby regions. In a small region, there will probably be other small regions nearby, so a fine-grained search is required in order to find them. However, a fine-grained search process will get stuck in a large region, taking tiny steps when great leaps are required. Our algorithm exploits this structure, by using MCMC to estimate a different sampling distribution $p(\theta|i)$ for every region i that it encounters, and then cycling through these distributions in order to sample parameter sets. The procedure can be reduced to three steps:

1. Set $i = 0$, $m = 0$. Sample θ from $p(\theta|0)$, a uniform distribution over the space. If θ does not generate a valid data pattern, repeat Step 1.

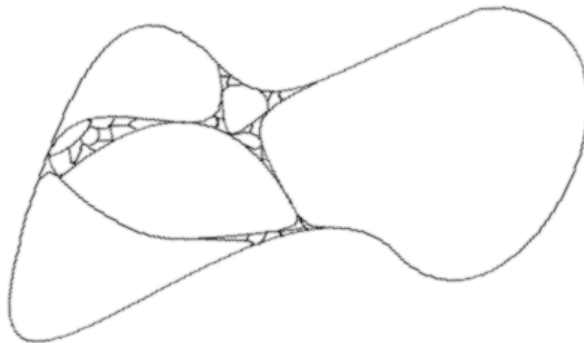


Figure 2: An uneven distribution of regions within the parameter space. Small regions tend to cluster together.

2. Set $m = m + 1$ and then $i = m$. Record the new pattern, and use MCMC to estimate $p(\theta|i)$.
3. Sample θ from $p(\theta|i)$. If θ generates a new pattern, return to Step 2. Otherwise, set $i = \text{mod}(i, m) + 1$, and repeat Step 3.

The process of estimating $p(\theta|i)$ is a fairly straightforward application of MCMC (e.g. Gilks, Richardson & Spiegelhalter 1995). We specify a uniform jumping distribution over a small (hyper)sphere centered on the current location in the parameter space, accepting points if and only if they produce the same pattern as θ . After collecting enough samples, we calculate the mean and variance-covariance matrix for these observations, and use this to estimate an ellipsoid around the mean, as an approximation to the i -th region. However, since we want to find points in the bordering regions, the the estimated ellipsoid is deliberately oversized. The sampling distribution $p(\theta|i)$ is simply a uniform distribution over the ellipsoid.

Unlike SMC (or even a more standard application of MCMC), our algorithm has the desirable property that it focuses on each region in equal proportion, irrespective of its size. Not only that, since the parameter space is high dimensional, the vast majority of the distribution $p(\theta|i)$ will actually lie near the edges of the ellipsoid: that is, the area just outside of the i -th region. Consequently, we search primarily along the edges of the regions that we have already discovered, paying closer attention to the small regions. The overall sampling distribution $p(\theta)$ is essentially a mixture distribution that assigns higher density to points known to lie near many regions.

5 Results

6 Implications for Modeling Phoneme Perception

What are the implications of these results for TRACE and MERGE? Within this particular experimental setting, MERGE is more complex than TRACE . How much more complex depends on the constraints imposed for phoneme and word recognition. When the constraints amounted to a simple threshold for recognition (low constraint), as was adopted by Norris et al (2000), MERGE yielded 21 (40%) more patterns than TRACE . In the second test, the constraints that had to be satisfied for recognition were stronger and amounted to also incorporating a variant of Luce's

(1959) choice rule, as McClelland and Elman (1986) have done. The recognized phoneme/word had to exceed an activation threshold, and exceed the activation level of its competitors by a fixed amount. With these constraints, MERGE generated 40 (148%) more data patterns than TRACE and all 27 TRACE patterns were a subset of MERGE's 67 patterns, making TRACE's performance nested within MERGE. Despite differences in complexity between the two models, it is important to remember that the behavior of each is highly constrained, coming nowhere close to being able to produce the 2,985,984 possible data patterns in the experimental design.

Analyses of the data patterns that the two models produced provides a means of learning how their design differences affect performance. In the first analysis, we investigated the degree of mismatch between these data patterns and that produced by a human listener. That is, does each model's output veer far from human performance? Each data pattern from the two models was compared with the human data across all twelve experimental conditions (6 phoneme and 6 word). The number of mismatches served as the distance measure. The proportion of patterns at each distance is shown in Figure 2. The results are quite similar and orderly for both models. Mismatching data patterns are rarely highly similar to and never grossly discrepant from the human pattern, but fall in between these extremes, producing patterns that mismatch from 3-7 of the experimental conditions. On average, TRACE produced slightly fewer mismatches than MERGE (3.15 vs. 3.60). These results are for the low-constraint criterion, but those for the high-constraint criterion are quite similar.

The second analysis built on the first by asking *how* each model's data patterns mismatched the human data. The number of mismatches in each experimental condition was tallied separately for the two tasks, and the frequency counts are plotted in Figure 3 (again, using only the low-constraint data). The distribution of mismatches is quite similar across conditions, with both models producing no mismatches in some conditions (e.g., w_1w_1 -phoneme identification, n_2n_1 -lexical decision) and many in others (e.g., w_2w_1 -lexical decision). Inspection of the relative frequency of errors across tasks reveals a substantial behavioral difference between the models. In lexical decision, TRACE and MERGE produced the same profile of mismatches across conditions and a somewhat comparable number of mismatches (108 vs. 124). In phoneme identification, not only are the profiles different but MERGE produced a substantial number of mismatches in conditions that TRACE did not (e.g., n_2w_1 , n_2n_1). When considered together, there is an asymmetry in the frequency of mismatches across tasks. For MERGE phoneme identification responses are similar to lexical decision responses (139 vs. 124). For TRACE, the number of mismatches in phoneme identification is less than half of that found in lexical decision (56 vs. 108).

The mismatch asymmetry is probably caused by the similarities and differences in the models' architectures (Figure 1). Word processing in the two models is virtually identical. Information from the phoneme level feeds into the word level, from which a lexical decision response is made. Consequently it is not surprising that the types of mismatches and their frequency are similar in this task. In contrast, phoneme processing is split between two layers in MERGE but confined to one in TRACE. This difference in design explains why phoneme identification performance differed substantially between the models. The two layers dedicated to phoneme processing provide MERGE an added degree of flexibility (i.e., complexity) in generating data patterns. This shows up in many ways, not just in MERGE's ability to produce mismatches in more conditions than TRACE. For example, these mismatches yield a wider range of phoneme responses. Shown above each bar in Figure 3 is the phoneme that was misrecognized in the given condition. TRACE only misrecognized

the phoneme as /g/ whereas MERGE misrecognized it as /g/, /z/, and /v/.

These analyses describe a few consequences of dividing processing between two layers, as in MERGE, and in doing so creating a more complex model. On the basis of performance (i.e., fit) alone, this additional complexity is unnecessary for modeling phoneme perception because the simpler architecture of TRACE simulates human data as well as MERGE. If MERGE's design is to be preferred, the additional complexity must be justified for other reasons (Norris et al, 2000).

7 Conclusions

The results of this preliminary evaluation suggest that algorithm performs well. Though inspired by TRACE and MERGE, this algorithm may be broadly applicable whenever the search space exhibits this kind of structure. Thus it could be a general tool for designing, comparing, and evaluating network models of human cognition, suitable for both global analyses (e.g. finding all patterns) and local analyses (e.g. parameter optimization). Nevertheless, some caution is required, until the approach has been applied in a greater variety of contexts. Future work will aim to extend the approach to different situations, and incorporate different dependent measures, such as response times.

Acknowledgements

The authors were supported by NIH grant R01-MH57472 awarded to IJM and MAP. DJN was also supported by the OSU Graduate Incentive Committee in Learning and Memory. We thank Cheongtag Kim and Yong Su for helpful discussions.

References

- Balasubramanian, V. (1997). Statistical inference, Occam's razor and statistical mechanics on the space of probability distributions. *Neural Computation*, 9, 349-368.
- Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1995). *Markov Chain Monte Carlo in Practice*. London: Chapman and Hall.
- McClelland, J. L. & Elman, J. L. (1986). The TRACE model of speech perception. *Cognitive Psychology*, 18, 1-86.
- Myung, I. J., Balasubramanian, V., & Pitt, M. A. (2000). Counting probability distributions: Differential geometry and model selection. *Proceedings of the National Academy of Sciences USA*, 97, 11170-11175.
- Norris, D., McQueen, J. M. & Cutler, A. (2000). Merging phonetic and lexical information in phonetic decision-making. *Brain & Behavior Sciences*.
- Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Transactions on Information Theory* 42, 40-47.
- Rissanen, J. (2001). Strong optimality of the normalized ML models as universal codes and information in data. *IEEE Transactions on Information Theory* 47, 1712-1717.