In F. Detje, D. Dörner, & H. Schaub (Eds.), *Proceedings of the Fifth International Conference on Cognitive Modelling* (pp. 21–26). Bamberg, Germany: Universitäts–Verlag Bamberg, 2003.

The Use of Entropy for Analysis and Control of Cognitive Models

Roman V. Belavkin (r.belavkin@mdx.ac.uk)
School of Computing Science, Middlesex University, London NW4 4BT, UK

Frank E. Ritter (ritter@ist.psu.edu)
School of Information Sciences and Technology,
The Pennsylvania State University, University Park, PA16802, USA

Abstract

Measures of entropy are useful for explaining the behaviour of cognitive models. We demonstrate that entropy can not only help to analyse the performance of the model, but also it can be used to control model parameters and improve the match between the model and data. We present a cognitive model that uses local computations of entropy to moderate its own behaviour and matches the data fairly well.

Introduction

Information theory can be applied to cognitive models and help to analyse the performance of both the models and subjects they simulate. Both information theory and cognitive science are concerned with the question of information change either in a form of transfer or acquisition. Learning is an important cognitive process that allows making correct decisions and improves performance. From information theory point of view learning can be seen as a reduction of uncertainty. Thus, the amount to which the uncertainty is reduced can be an indicator of the speed of learning.

Cognitive models are sufficiently transparent allowing a researcher to examine all aspects of their internal state at any moment (although in practise it may not always be a trivial task). This extends the number of observables beyond the traditional ones (i.e. errors, reaction times, strategy choice). These additional observables can be probabilities of rules, activations of knowledge units, synaptic weights and so on. Although a good model (good in terms of correlation with data) cannot be considered literally as the representation of processes in the brain, careful analysis of the dynamics of internal variables of the model helps to explain why we observe what we observe externally in the behaviour of the model and, perhaps, in subjects.

In this paper the notion of entropy will be applied to describe a state of a cognitive model. It will be shown in the first section how to calculate the entropy using internal parameters provided by cognitive architectures, such as the conflict resolution parameters in ACT–R (Anderson & Lebiere, 1998). In the second section we shall illustrate the use of entropy in a particular ACT–R model and discuss the dynamics of entropy during the model run. In particular, it will be shown that although the entropy decreases on average, it may suddenly increase at certain stages of problem solving, such as when the model learns new production rules or when the number of errors increases due to environment changes.

In the third section we shall discuss how the amount of information gain determined by the changes of entropy may help to analyse the impact of parameter settings on learning. For example, it will be shown that an increase of noise variance in the ACT–R conflict resolution accelerates information gain. Thus, noise increase may help a problem solver at certain stages of task exploration (e.g. when the environment changes or at the beginning of problem exploration). Finally, we shall discuss the idea of dynamic parameters control in the model using entropy. We present a model with noise variance controlled dynamically by the entropy that achieves a better match with the data than a model with static settings.

Uncertainty of Success

The entropy H of a system with random states ξ is defined as

$$H\{\xi\} = -E\{\ln P(\xi)\} = -\sum_{\xi} P(\xi)\ln P(\xi)\,, \quad (1)$$

where $E\{\cdot\}$ denotes expected value, and $P(\xi)$ is the probability of state ξ. It is quite difficult to estimate the entropy of a large system with many state (e.g. a cognitive model). However, we may consider the problem from a different perspective. Let us consider a problem solver with a goal and a set of decisions, and let $\xi \in \{0,1\}$ be a set of two states with respect to achieving the goal: failure ($\xi = 0$) and success ($\xi = 1$). Now the uncertainty that the problem solver will achieve the success or failure state is:

$$H_{01} = -\left[P(0)\ln P(0) + P(1)\ln P(1)\right]\,, \quad (2)$$

where $P(1)$ and $P(0)$ are probabilities of success and failure respectively. Note that $P(0) = 1 - P(1)$. We shall call H_{01} the *entropy of success*.

Suppose that the success of a problem solver depends on decisions it makes. Thus, the probability of success can be written as

$$P(1) = \sum_i P(1,i) = \sum_i P(1 \mid i) P(i)\,,$$

where $P(1,i)$ is the joint probability of event 1 and ith decision, $P(1 \mid i)$ is the conditional probability of 1 given that ith decision has been made, and $P(i)$ is the probability of ith decision.

In order to calculate the entropy H_{01} one should establish ways of estimating probabilities $P(1 \mid i)$ and $P(i)$, and these ways may depend on specific architectural implementation. Consider as an example the ACT–R cognitive architecture (Anderson & Lebiere, 1998). In this case the decisions are applications of production rules of the model.

ACT–R records the history of successes and failures of each rule and uses this information to estimate empirically the expected probabilities P_i of success for each production rule:

$$P_i = \frac{\text{Successes}_i}{\text{Successes}_i + \text{Failures}_i}\,. \qquad (3)$$

These probabilities are empirical estimations of conditional probabilities $P(1 \mid i)$, and given that tests of rules are independent P_i, asymptotically converges to $P(1 \mid i)$. Thus, we may use ACT–R expected probabilities to calculate $P(1)$:

$$P(1) \approx \sum_i P_i\, P(i)\,.$$

Probability $P(i)$ that rule i fires is determined in ACT–R by several subsymbolic mechanisms. The most important here is the conflict resolution (a process of selecting one rule out of several matching the current goal state). In ACT–R each production rule has a *utility* U_i attached to it (also sometimes called *expected gain*). In order to resolve the conflict, ACT–R selects the rule with the highest utility. The utility of rule i is defined as:

$$U_i = P_i G - C_i + \xi(s)\;. \qquad (4)$$

Here P_i is the expected probability of success (eq. 3), G is the *goal value* (usually measured in time units), C_i is the *expected cost* representing the average effort required to achieve the goal if rule i fires (it too may be learned empirically). $\xi(s)$ is the *expected gain noise*, a random value added to utility of each rule, taken from a normal distribution with the mean value of zero and variance determined by the s parameter: $\sigma^2 = \pi^2 s^2/3$.

Thus, equation (4) describes a distribution of U_i with the mean value $P_iG - C_i$ and variance controlled by the parameter s (see Figure 1). Noise adds a non-deterministic aspect to the conflict resolution, which helps ACT–R to select from several production rules even when their evaluations $P_iG - C_i$ are equal.

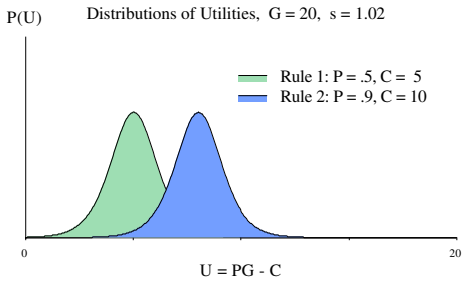

Figure 1: Example of utility distributions for two rules for $G = 20$ and noise $s = 1.02$ in eq. (4). A rule with the highest utility is preferable. Rule 2 is more likely to be chosen here.

The fact that rule i has the highest utility does not guarantee that it will fire. For example, failure to retrieve a chunk in the antecedent of a rule will lead to an instantiation failure, and the rule will not fire. Retrievals rely on many other parameters in ACT–R: chunks activations, activation noise, retrieval threshold and so on. There may be other parameters involved such as the *utility threshold*. Therefore, in order to correctly estimate the probability $P(i)$ for particular model settings, a series of intensive calculations would be required, such as the Monte–Carlo simulations.

Nevertheless, for some studies it may be sufficient to neglect the effect of mechanisms other than conflict resolution on choice of production rules. In such closed–form approximation the probability $P(i)$ that rule i will be selected is given by the Boltzmann 'soft–max' equation:

$$P(i) = \frac{e^{\bar{U}_i/\tau}}{\sum_j e^{\bar{U}_j/\tau}}, \qquad (5)$$

where $\bar{U}_i$ is the evaluation of ith rule $(P_iG - C_i)$, and τ is called the *noise temperature*, which is related to the variance and noise s parameters as $\tau^2 = 6\sigma^2/\pi^2 = 2s^2$

Using the above formula for $P(i)$ and expected probabilities P_i we can estimate the probability of success:

$$P(1) = \frac{1}{\sum_i e^{\bar{U}_i/\tau}} \sum_i P_i\, e^{\bar{U}_i/\tau}\;. \qquad (6)$$

Therefore, the entropy H_{01} of success in achieving the goal can now be calculated by equation (2).

Note that it is more convenient to use the following formula for calculating the empirical probabilities P_i:

$$P_i = \frac{\text{Successes}_i}{\text{Successes}_i + \text{Failures}_i + 1}\,.$$

This is because by default ACT–R sets initially all the probabilities $P_i = 1$ (the number of successes is set to 1 initially). Anderson and Lebiere (1998, p. 135) justify this in order to make the prospects of a new production optimistic. This approach, however, is biased and not convenient for calculating the entropy. Indeed, if at the beginning all $P_i = 1$, then the uncertainty of a success $H_{01} = 0$, which contradicts the idea that initial state should be the maximum entropy state (no experience). The above formula for the empirical probability is more suitable to estimate the entropy: it makes the initial values $P_i = 1/2$, and probability of success $P(1) = 1/2$ (see eq. 6). This corresponds to the maximum of entropy H_{01}.

Because the number of production rules may change due to learning new rules, it is convenient to use the notion of relative entropy:

$$H_{\text{rel}} = \frac{H_{01}}{H_{\max}} ,$$

where $H_{\max}$ is the maximum entropy with all states equally likely (i.e. $P(1) = P(0) = 1/2$). Redundancy, defined as $1 - H_{\text{rel}}$, can be a good estimator of the information accumulated by a system. So, we can use the reduction in entropy to test how well the model learns under different parameter settings.

Dynamics of Entropy in a Model

In this section we illustrate the dynamics of relative entropy in the Dancer model (see Belavkin, 2003, for details) — an ACT–R model of the famous "dancing mouse" experiment of Yerkes and Dodson (1908) (the experiment in which the Inverted–U effect of strength of stimulus on performance was first discovered). This work despite its respectable age is still widely cited in the literature. In addition, the experiment is an example of a typical animal learning study. In this experiment mice were placed into a discrimination chamber with two exits, and they were trained for several days to escape the chamber through one particular door based on some strategy. For example, in this study the mice had to learn to escape only through the door marked by a white card. The order of the doors was changed randomly, so the mice had to learn to choose the door based on its colour rather than on its position.

Figure 2 shows a typical trace of one such test in the model: the Dancer (simulated mouse), shown as an arrow object, first enters the left door with a black card, and after receiving aversive stimulus escapes back into the chamber and enters the right door with the white card.

It is not possible to describe here all the model's features. Nevertheless, let us outline how the model learns to choose the correct escape door. A choice of two objects is represented in the model by a chunk of a special type *choice*. The model starts with only two simple production rules matching such goal:

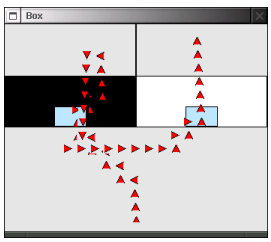

Figure 2: A typical trace of the Dancer model (Belavkin, 2003).

Choose1st:
IF the goal is a *choice* of *first* or *second*
THEN focus on *first*

Choose2nd:
IF the goal is a *choice* of *first* or *second*
THEN focus on *second*

The two conflicting rules above do not use any features of the objects, and due to the noise in the conflict resolution (eq. 4) they implement a random choice strategy. In order to choose the door correctly the model needs to learn new production rules that use features of the doors as constraints. These rules are added into the procedural memory of the model using the ACT–R *production compilation* mechanism. After entering the wrong door, the model recalls the last choice it has made, and using the information about the two objects contained in the choice chunk the model compiles a new rule. The new production rule is similar to the two rules shown above: it also matches the choice goal. However, the new rule uses additional constraints. For example, below is an example of a rule that prohibits choosing the black door:

New-Choice-Rule:
IF the goal is a *choice* of *first* or *second*
AND *first* is a *black* door
THEN focus on *second*

The rule above uses only one feature — colour. Another feature that can be used in a new rule is the door's position (left or right). Rules using only one feature implement one–dimensional learning. The model may also learn rules with both colour and position as constraints (two–dimensional learning).

In their experiment Yerkes and Dodson performed 10 tests per day with each mouse for up to 30 days (training series). The number of errors for each day was recorded, and if the mouse did not produce any errors for three consecutive days, the experimented was terminated. Yerkes and Dodson referred to this moment as the *perfect habit* formation. Figure 3 shows an example of the error curve produced by the model simulating these tests with perfect habit

formed on day 8. Note that during the first two days the mice were allowed to escape through any door (preference series denoted by letters A and B).

In a series of tests the model learns up to six new production rules, of which some prove to be more successful than the others. For example, the new–choice rule shown above is the most successful strategy in this experiment, because it prohibits choosing the black door and does not contain any redundant information (i.e. door position). Using the subsymbolic parameters learning mechanism of ACT–R the model learns statistically the probabilities P_i and costs C_i of the new rules. Selecting rules with the highest utility (eq. 4) allows the model to reduce the number of errors. Figure 4 shows the dynamics of probabilities of all the rules matching the choice goal. The corresponding dynamics of the relative entropy of success H_{rel} associated with these rules is shown on Figure 5.

Although the error curve (Figure 3) suggests that the task performance improves, this curve does not really provide much information about learning in the model. The traces of probabilities (Figure 4) give much more detailed picture: one can see that new rules are added during training days 1, 5 and 7. However, it is hard to compare the knowledge of the model, for example, between day 8 and 9. Moreover, this representation would be much more obscure if the number of production rules in the conflict set was large (e.g. several hundreds of rules).

The entropy plot (Figure 5) unites the information about probabilities into one curve. It shows exactly how much the uncertainty reduced even between day 8 and 9 when no obvious change in probabilities or behaviour is noticeable. More importantly, the dynamics of entropy emphasises some significant stages in the learning process: moments of sudden information changes when the number of errors increases and the new rules are learned, such as on day 5 and 7. An increase of entropy can be explained by changes in probabilities and addition of new rules to the conflict set. As we can see, entropy proves to be a convenient tool for the analysis of learning in the model and its behaviour.

Estimating the Knowledge

Reduced entropy of success H_{01} states that the success of a problem solver (the model) is more certain, and it takes into account not only the knowledge acquired, but also the way this knowledge is used to make decisions. Indeed, the choice probability $P(i)$ described by equation (5) was used to calculate the probability of success $P(1)$ in equation (6). However, as was noted before, the choice of rule i (and $P(i)$) depends on the architecture and its parameters. For example, in ACT–R $P(i)$ depends also on the noise temperature τ (see eq. 5). Thus, entropy H_{01} is not convenient for estimating learning in the system under different noise settings.

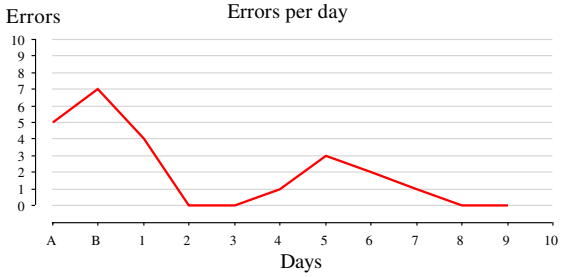

Figure 3: Error curve produced by the model in one experiment.

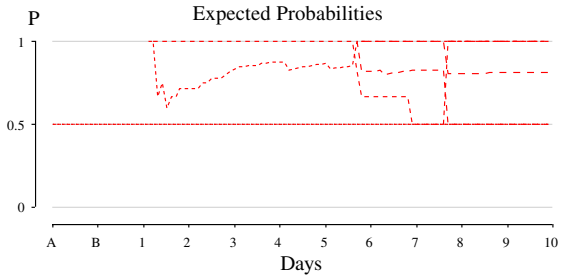

Figure 4: Dynamics of probabilities of rules matching the choice goal. The number of these rules increases due to compilation (learning) of new rules.

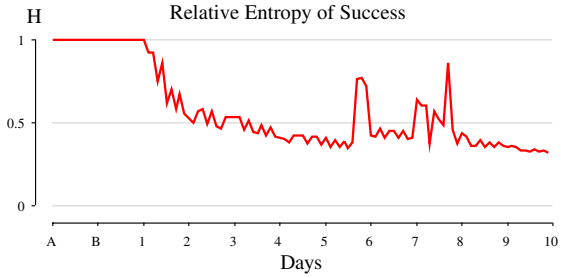

Figure 5: Relative entropy of success of the choice rules. Entropy increases with the number of errors (see Figure 3) and when new rules are learned.

To make the estimation of knowledge learned by the system independent of the decision making and other mechanisms in the architecture let us assume that the choice of a rule is completely random: $P(i) = \frac{1}{n}$, where n is the number of rules (decisions). In this case probability of a success $P(1)$ can be calculated as

$$P(1) = \frac{1}{n} \sum_i P_i \ . \qquad (7)$$

The entropy associated with this probability (calculated similarly by eq. 2) can be used to estimate the knowledge accumulated in the system in the form of

empirical probabilities P_i, because it is independent of the way the decisions are made. We shall refer to it as the *entropy of knowledge* H_{k}.

The value of H_{k} is that it decays differently under different parameters settings in the architecture. Thus, it shows what conditions facilitate the information acquisition. For example, it turns out that although noise in the ACT–R conflict resolution (eq. 4 and 5) may seem to hinder the performance of the model, it in fact helps to learn the expected probabilities of rules faster. Figure 6 illustrates the probability learning in the Dancer model for different noise settings. The left plot shows traces of probabilities during 10 simulated days with low noise $\tau = .05$ (or $T = 1\%$ of goal value G), and the right plot for high noise settings $\tau = 1$ (or $T = 20\%$).[1] Probabilities on the right plot were updated much more often, thus have more correct values. The corresponding entropy H_{k} is shown on Figure 7. One may see that by day 10 the entropy on the right plot decayed significantly more than on the left plot. Thus, by day 10 the model with a greater noise gained more information than the model with less noise.

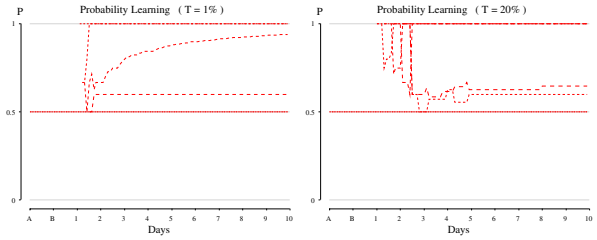

Figure 6: Probability learning under a low noise (left) and a high noise conditions (right).

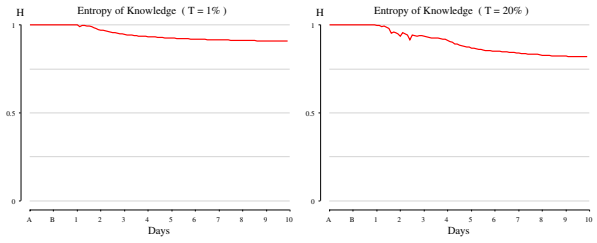

Figure 7: Dynamics of entropy under a low noise (left) and a high noise condition (right).

Use of Entropy for Control

Results of some ACT–R models suggested that the models could fit the data better if the noise variance in conflict resolution was decreasing during problem solving (Jones, Ritter, & Wood, 2000; Belavkin, 2001). Moreover, the analysis of H_{k} reduction for different noise settings in the conflict resolution lead to a speculation that subjects strategically control the variance of the noise (Belavkin, 2003). Indeed, dynamic noise may have a very useful optimisation purpose:

1. Noisy behaviour in the beginning of problem exploration supports gaining information about the task or the environment more quickly.

2. After the important information has been acquired, the reduction of noise allows narrowing the search and concentrate on more successful decisions. If the learned knowledge is correct for the task or the environment, then keeping the noise low should improve performance.

3. If the environment changes and the number of errors suddenly increases, then a noise increase widens the search and allows speeding–up the learning process again.

Note that the dynamics of noise variance described above corresponds to the dynamics of entropy in the model (e.g. Figure 5). A simple way to control noise variance by the entropy parameter has been proposed recently (Belavkin, 2003). More specifically, the relative noise temperature $T = \frac{1}{G}\tau \cdot 100\%$ ($\tau^2 = 2s^2$) was modified in time as:

$$T(t) = T_0 H_{\mathrm{rel}}(t)\,, \tag{8}$$

where t is time, $T_0 = T(0)$ is the initial value of the noise, $H_{\mathrm{rel}}(t)$ is relative entropy of success for the task–related productions. In the Dancer model these are rules that make the choice of the door.

Comparison of the Model with Data

As predicted, the above described modification significantly improved the model performance. Table 1 compares the best results of the model with static noise settings and the model with noise controlled by equation (8). The table compares models evaluated on data Set I from Yerkes and Dodson (1908) for three levels of stimulation: weak (125), medium (300) and strong (500) stimulus. One can see that both the coefficient of determination (R^2) and the root–mean–square error (RMS) have improved for the model with dynamic noise.

Figure 8 shows one to one comparison of the error curves from data for three levels of stimulation in Set I with the corresponding curves of the model with static noise. Comparison with the dynamic model is shown on Figure 9. One can see that the distributions of errors for the dynamic model fit better the experimental distributions. This pattern of improvement is consistent for different data sets.

[1] Here T is called relative noise defined as $\frac{1}{G}\tau \cdot 100\%$.

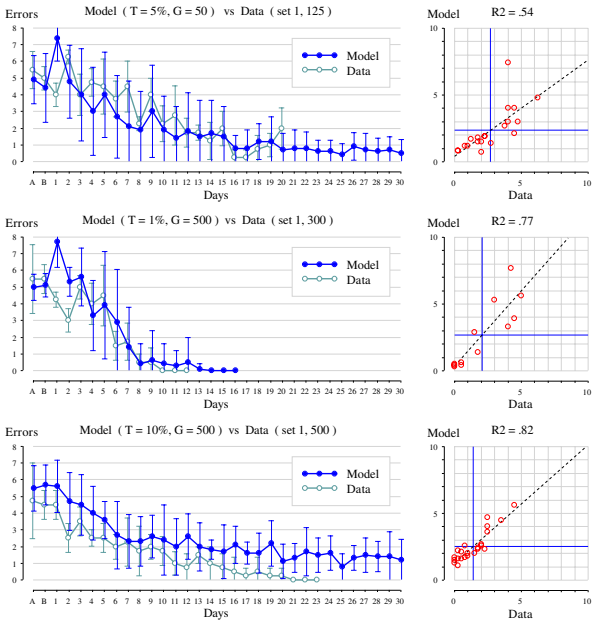

Figure 8: Static model compared with three sets of data (Yerkes & Dodson, 1908). Left: learning curves. Right: regression plots.

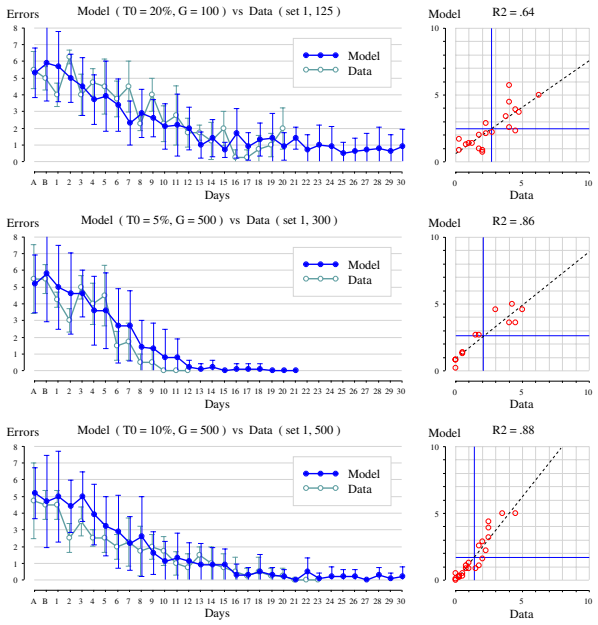

Figure 9: Dynamic model compared with three sets of data (Yerkes & Dodson, 1908). Left: learning curves. Right: regression plots.

Table 1: Comparison of models with static and dynamic noise variance to Yerkes and Dodson data.

	Static noise		Dynamic noise	
Data set	R^2	RMS	R^2	RMS
Set I–125	.54	12.2%	.64	10.1%
Set I–300	.77	13.2%	.86	8.8%
Set I–500	.82	12.4%	.88	7.1%

Conclusions

It has been shown here that some architectures provide sufficient information to estimate the value of entropy in a model. The reduction of entropy becomes a useful tool for representation and analysis of the model learning. Using entropy we demonstrated on an ACT–R model that increased noise variance on certain stages of problem solving helps the model to learn faster. In addition, we showed that entropy reduction can be used to control the decay of noise variance, which in turn significantly improved the fit of our model to data. We hope that this paper will encourage other cognitive scientists to consider using this approach to analyse and improve their model's ability to match and explain human performance.

Acknowledgements

David Elliman and David Wood provided useful comments and support for this work. We also thank the two reviewers. Belavkin was supported by the Overseas Research Studentship and ESRC Credit and Ritter by the ONR, award numbers N000140110547 and N000140310248.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought.* Mahwah, NJ: LEA.

Belavkin, R. V. (2001). Modelling the inverted–U effect in ACT–R. In E. M. Altmann, A. Cleeremans, C. D. Schunn, & W. D. Gray (Eds.), *Proceedings of the Fourth International Conference on Cognitive Modelling* (pp. 275–276). Mahwah, NJ: LEA.

Belavkin, R. V. (2003). *On emotion, learning and uncertainty: A cognitive modelling approach.* PhD Thesis, The University of Nottingham, UK.

Jones, G., Ritter, F. E., & Wood, D. J. (2000). Using a cognitive architecture to examine what develops. *Psychological Science*, *11*(2), 93–100.

Yerkes, R. M., & Dodson, J. D. (1908). The relation of strength of stimulus to rapidity of habit formation. *Journal of Comparative Neurology and Psychology*, *18*, 459–482.