Using the EPAM Theory to Guide Cognitive Model Rule Induction

Bradley J. Best
Micro Analysis & Design
4949 Pearl E. Parkway, Suite 300
Boulder, Colorado
303-442-6947
bbest@maad.com

Keywords: cognitive model, learning, induction, EPAM, ACT-R

ABSTRACT: *The isomorphism between rule-based systems and decision trees provides an opportunity to use induction methods associated with decision trees and data mining as the basis for the formation of rules in a cognitive architecture. The EPAM theory of learning, an exemplar decision tree method, provides the core processes for a theory of production formation within the ACT-R architecture which is applied to a traditional concept formation task, the 5-4 category task. Rule creation using this method provides a good quantitative fit to the human performance data. This method is impasse driven, provides generalization, and is tolerant to noisy examples. Excess rules, rather than being pruned, are withered away through utility learning.*

1. Inducing Rules from Data

The goal of this project is to exploit the isomorphism between decision trees and rule-based representations in an exploration of the use of decision-tree induction (learning) methods in general, and the EPAM theory (Elementary Perceiver and Memorizer, Feigenbaum & Simon, 1984) in particular, for developing cognitive models within the ACT-R cognitive architecture (Anderson & Lebiere, 1998).

This work applies to task domains that involve a mapping of continuous, categorical, or symbolic features onto discrete categories. Tasks typical of this type of domain include fault diagnosis, categorization tasks, and strategy selection. One exemplar task, the 5-4 Categorization Task (also known as the Brunswick Faces Task), will be described in detail below to illustrate the application of decision tree learning to a rule-based representation. In the 5-4 Categorization Task, the goal is to use the presented features of the stimulus to decide which of two groups it belongs to. To accomplish this, the cognitive model must determine which features actually are discriminative, and must learn to rely on them to perform the task.

The methodology discussed here is inherently supervised: training examples are given with their classifications. However, "supervised" is meant in this narrow technical sense, and the method is also equally applicable to developing models from streams of human performance data, where the supervision may be far less than perfect, and may simply consist of observing performance that includes both environmental cues and observable actions.

Rules as used in many cognitive modeling systems consist of a pairing of cues, or context, with an operator, or action to provide the basis for intelligent behavior. These rules are frequently described simply as condition-action pairs, but can also be thought of as condition-operator pairs in the current context.

An often asked question of rule-based cognitive models is: Where do rules come from? The variety of mechanisms proposed include many methods for combining existing rules such as chunking (Newell & Rosenbloom, 1983), compilation (e.g., Taatgen & Lee, 2003), and composition, but each of these begs the question of where those rules that are being combined themselves came from. Recently, work has shifted to learning initial rules directly from instructions and encoding instances as specific rules (Taatgen & Lee, 2003). However, this work provides little guidance on the formation of new rules that combine operators with variable cue structures. The current proposal explores the alternative of an always-on production learning mechanism based on the EPAM theory of learning and recognition.

EPAM is, at its core, a decision tree learning process. Although there are many other decision tree learning paradigms in current use, particularly in the area of machine learning, EPAM, as a cognitive theory, provides several specific benefits for this project. One advantage is that cognitive methods are often computationally simpler than AI methods due to their need to reduce the computations to those performable by a person in a limited amount of time (Best, 2005). Another advantage is that cognitive methods depend on a higher amount of pruning of search spaces (e.g., see studies of chess experts by Gobet & Simon, 1998), and as a result may scale more effectively. Yet another

advantage is that cognitive methods must be extremely adaptive, and must be able to learn from limited data with potentially changing base-rates, making these systems ideal for deployment in changing, uncertain environments (exactly those environments where more traditional AI systems tend to become brittle and founder).

1.1 Decision Trees

A decision tree is a rooted tree used for sorting cases, where leaf nodes represent case labels while paths from non-leaf nodes represent individual feature tests. Thus, as a case is sorted down the net, each decision made moves the decision process into a smaller partition of the decision space.

Decision tree methods have their roots in the early days of information processing (Feigenbaum & Simon, 1961). They are now more often used in statistical classification (Breiman, Friedman, Olshen, & Stone, 1984), machine learning and data mining domains (Mitchell, 1997; Quinlan, 1993).

One way to think of a decision tree is as an attempt to construct a hypothesis that accounts for all of the given data. Thus, decision tree induction is a search of the hypothesis space for the best tree given the data. Since the features and categories are known a-priori, the hypothesis space itself is complete (i.e., the hypothesis, if it exists, must be in the space considered by the search process). Because this space is infinite, however, the search must also be guided heuristically. Also, although it is possible to perfectly fit any data set, to be truly useful a decision tree should generalize and therefore must balance the benefits of fitting the data against the risks of over-fitting the data (and compromising generalization to new data).

Although decision tree induction takes many forms, there are several commonly accepted practices in the machine learning community. Many currently used methods such as CART (Breiman, Friedman, Olshen, & Stone, 1984), ID3, and C4.5 (Quinlan, 1993), look either at the whole data set, or if that is impractical due to size, take a statistical sample of it (a window), and extract statistical properties from that data for use in inducing a near-optimal decision tree. These methods place decisions on features with greater discriminative power towards the top of the tree thereby employing ideal partitions of the data set. There is also a generally accepted bias towards growing trees with shorter hypotheses (Occam's razor). Finally, convention is to overgrow the decision tree and then prune it back, removing unnecessary nodes, clarifying its meaning to users of the system, and reducing the risk of overfitting (Quinlan, 1993).

Decision tree representations are exact isomorphs to rule-based systems: a decision tree can be represented by an equivalent set of production rules (Mitchell, 1997; Quinlan, 1993). Despite their underlying equivalence, however, the different representations appear to present different challenges in interpretation and understanding. Decision tree representations seem to facilitate understanding of learning and induction processes (i.e., a decision tree may be easier to think about when constructing inductive methods), while the production representation facilitates understanding of performance and meaning of the representation. For example, Quinlan (1993) translates C4.5 decision trees to production rules explicitly for the purpose of improving intelligibility.

Decision tree induction has also been explored in the context of modeling human performance. In the context of cognitive modeling, however, a slightly different set of accepted practices have emerged. First, learning only occurs on the current item, in a trial-by-trial fashion. This is primarily a requirement of the domain: to simulate learning, the learning program must not look ahead into the future and take the whole learning experience all-at-once, but instead must process it in (simulated) real-time. This has the side benefit of accounting for learning against changing base-rates, which is impossible to do if the data must be presented completely before learning commences.

Much of the tradition within cognitive modeling involving decision tree learning has focused on the development of the EPAM family of theories, starting with Simon & Feigenbaum (1961, 1962) and continuing with the development of the CHREST theory (Gobet, Lane, Croker, Cheng, Jones, Oliver, & Pine, 2001). Although the use of these models as cognitive models is not without controversy, these models do account for a substantial portion of learning effects found in various lab and field experiments. For example, EPAM and CHREST account for effects such as the shape of the serial position curve, the impact of familiarity and meaningfulness on learning, conditions that enable one-trial vs. continuous learning, effects of similarity, expert pattern perception, and many others (Feigenbaum & Simon, 1984; Gobet & Simon, 1998).

1.2 EPAM

EPAM, like the methods described above, is a decision tree method. The aim of the EPAM theory, however, is not to provide best-fits to statistical classifiers or functionality for data-mining applications, but to model the way people learn to recognize stimuli. Objects in EPAM are either a list of individual features, or a list of sub-objects which may themselves be composed of sub-

objects or features. This report will focus on the representation using lists of features, but will discuss the extension of the current work to handle sub-objects as well.

An EPAM decision tree consists of a set of test-nodes, including the root, and zero or more leaf nodes containing recognized (memorized) information. The tree itself is an n-ary tree, and can therefore have any number of tests (branches) located at a particular node. In its initial state, an EPAM network consists simply of the root node with no tests or branches present. In the case we are considering here, a classification task, EPAM is then presented with a series of trials in which it is given a stimulus and asked to make a classification among the possible categories of responses (i.e., EPAM assumes some knowledge of the instructions as given in a typical classification experiment).

Interacting with a stimulus engages two distinct processes in EPAM. First, repeated exposure leads to storing more of the stimulus features through the familiarize process in the image at the leaf node. Second, any misclassification of a stimulus results in engaging the discriminate process, whereby a difference is noted between the current stimulus and the stored image, and a corresponding test is added to the decision node to allow sorting the discrepant case to a new (as yet unconstructed) leaf node. These two primary processes are exactly those that will be used to drive production induction in the current work.

The discriminate process is conservative in that it adds at most one test at a time based on noticing a single difference (this lazy learning also provides the basis for a computationally efficient learning process). The EPAM theory, as a result of being rooted in the verbal learning tradition, specifies the order of consideration of features thereby accounting for a large portion of the body of data from word and non-word learning studies. Specifically, EPAM starts with features at the ends of lists, especially the front end (thus discriminating words first on their first letters). However, this ordering, due to an idiosyncrasy of the data set modeled (the first feature is the most informative), provided an unfair advantage to the system in performing the task, and was therefore disabled in the described system.

EPAM departs from the decision tree methods discussed above in one key way: it processes items one at a time and learns what it can without reviewing all of the data available. This quality is a result of its aim to model human memorization (rather than to optimize classification accuracy). This is a strength in two ways: 1) it may improve the simulated fit to human data despite decreasing accuracy, and 2) it supports performance in dynamic, changing environments where there is no option to 'look ahead' at the remainder of the trials.

1.3 Rule Based Cognitive Architectures

Given the equivalence between decision trees and production rules, a candidate system for an EPAM-based production induction mechanism must itself use productions. Cognitive architectures such as ACT-R and Soar use productions as a basis for their procedural knowledge, and as such, are candidate targets for this method. In addition, each of these architectures has existing learning methods. In Soar, the procedural learning primarily takes the form of chunking (Newell & Rosenbloom, 1983), which is similar to production compilation in ACT-R. ACT-R also provides mechanisms for learning the usefulness of a rule from experience in applying it for use in future decisions, and as a result provides a unique platform for exploring the interaction of a rule-induction method with a utility learning mechanism.

This is not to imply that production utility learning is the sole learning method in ACT-R. In fact, many other methods have been used to map continuous and categorical inputs onto continuous and categorical outputs within the ACT-R architecture. Methods that map continuous inputs to categorical decisions using procedural representations are described by Best & Lebiere (2003). Methods mapping continuous inputs to categorical decisions using declarative representations are described in Best and Lovett (2006). Methods mapping continuous inputs to continuous outputs using declarative representations are described by Wallach & Lebiere (2002).

The current work aims to map a finite set of input features, whose potential range of values are initially unknown, onto a categorical decision whose potential values are known initially (i.e., given in the task instructions), and to do so through a production induction mechanism.

1.4 ACT-R

ACT-R is a unified architecture of cognition developed over the last 30 years at Carnegie Mellon University. At a fine-grained scale it has accounted for hundreds of phenomena from the cognitive psychology and human factors literature. The version employed here, ACT-R 5.0, is a modular architecture composed of interacting modules for declarative memory, perceptual systems such as vision and audition modules, motor systems such as a manual module, all synchronized through a central production system. This modular view of cognition is a reflection both of functional constraints and of recent advances in neuroscience concerning the

localization of brain functions. ACT-R is also a hybrid system that combines a tractable symbolic level that enables the easy specification of complex cognitive functions, with a subsymbolic level that tunes itself to the statistical structure of the environment to provide the graded characteristics of cognition such as adaptivity, robustness and stochasticity.

The central part of the architecture is the production module. A production can match the contents of any combination of buffers, including the goal, which holds the current context and intentions, the retrieval buffer which holds the most recent chunk retrieved from declarative memory, visual and auditory buffers that hold the current sensory information, the manual buffer which holds the current state of the motor module (e.g. walking, firing, etc), as well as buffers defined specially for the task. The highest-rated matching production is selected to effect a change in one or more buffers, which in turn trigger an action in the corresponding module(s). This can be an external action (movement, firing, etc) or an internal action such as requesting information from memory.

Although the production matching mechanism is rule-based, it also has many desirable "soft" properties such as adaptivity to changing circumstances, generalization to similar situations, and variability. The key mechanism that supports this capability is the utility learning mechanism, which allows the probabilistic selection of productions based on their learned cost and likelihood of success.

During the matching phase, production rules whose conditions match perfectly to the current state of various information buffers (goal, memory retrieval, perceptual, etc) qualify to enter the conflict set. Since ACT-R specifies that only one production can fire at a time, the rule with the highest expected utility is selected from the conflict set as the one to fire. The utility of a production rule is learned by a Bayesian mechanism as a function of its past history to reflect the probability and cost of achieving its goal. This is reflected in the following formula for expected gain of a production:

$$E = PG - C$$

Expected gain, then, is the product of the probability of a production's success and the value of the goal, minus the expected cost of executing the production. The probability of success is given by the following formula:

$$P = \frac{Successes}{Successes + Failures}$$

The probability of success, according to this formula, is the ratio of successes to the total number of experiences. Successes and failures are both defined in terms of a set of priors and the subsequent learning:

$$Successes = \alpha + m$$
$$Failures = \beta + n$$

These quantities all combine to determine the likelihood of a production's selection in the conflict resolution phase. That likelihood is given by the following:

$$P(production_i) = \frac{e^{E_i/t}}{\sum_j e^{Ej/t}}$$

In this equation, t represents the value of the noise used. Noise is an essential component in allowing for sampling from productions that are currently less likely to succeed based on past history. This is an essential feature of systems that operate in environments characterized by changing base rates, and has been used to explain human probability matching behavior (e.g., Holland, 1975).

This utility learning conflict resolution method can be used to implicitly prune a decision tree: though the productions are still there, they come to be dominated by other productions that are more useful, and thus they become effectively pruned (which is made obvious when inspecting the learned parameters attached to the productions). This method, then, could be called *withering*, rather than pruning.

The impact of production utility learning on a decision tree while it is being induced is quite complex, however. The main reason for this is that the discriminate process is triggered by making mistakes, while mistakes are triggered by failure to discriminate. Thus, if the system learns too quickly, it will do so by using the familiarize process to form exact representations of the input, and therefore fail to generalize. The learning must be slow enough to allow sufficient mistakes to be made early in the learning process (near the root of the decision tree) so as to drive discrimination among the features. This will be made clearer in the next section, where the learning method is applied to a concrete task.

2. Modeling Study

This section will first start with an explanation of the task environment. It will then move on to a detailed

explanation of the model and then move on to discuss the quantitative results.

The 5-4 Categorization Task is a task where an assignment it to be made to one of two categories based on four features. The training set contains five exemplars of the first category and four of the second category and one item from each category has a contradictory value for the otherwise most diagnostic feature. The transfer set consists of a set of items which have not previously been seen, but share some attributes with the training set. Thus, the task structure includes both general properties and exceptions, and tests generalization to new situations. The stimuli for the 5-4 Categorization Task, from Smith & Minda (2000), are given in table 1.

Table 1: Stimuli for the 5-4 Categorization Task

	Feature			
	F1	F2	F3	F4
Stimulus				
Category A				
A1	1	1	1	0
A2	1	0	1	0
A3	1	0	1	1
A4	1	1	0	1
A5	0	1	1	1
Category B				
B6	1	1	0	0
B7	0	1	1	0
B8	0	0	0	1
B9	0	0	0	0
Transfer Set				
T10	1	0	0	1
T11	1	0	0	0
T12	1	1	1	1
T13	0	0	1	0
T14	0	1	0	1
T15	0	0	1	1
T16	0	1	0	0

This task has been widely studied as a human concept attainment task. Smith & Minda (2000) present a meta-analysis of 29 studies which provides convenient data for modeling. In addition, Gluck, Staszewski, Richman, Simon, & Delahanty (2001) have already modeled this data using EPAM, making this an ideal test case for a production system induced using EPAM-like mechanisms.

The overall model structure consists of an iterative cycle of choosing whether to make a decision (i.e., apply an operator) based on the encoded features, or choosing to encode more features. As subsequent features are encoded, more productions might apply to classifying the particular case being decided upon (i.e., the more general productions corresponding to higher nodes in the decision tree still apply, while more specific ones may also now apply). If a decision is made to encode a particular feature that process is completed before the decision whether to make a decision or obtain more information is revisited.

The initial productions are not, in fact, capable of doing task – they just handle feature encoding and giving a response given that one was selected. There is one feature encoding production for each possible feature as is shown below for the first feature:

```
(p encode-feature-1
  =goal>
  isa face
  f1 ?
  ==>
  !bind! =f1 (get-feature 'F1)
  =goal>
  f1 =f1
  !output! "Encoding feature 1"
  )
```

In plain English, the meaning of this production is that, if a decision has been made to encode feature 1, then get that feature from the display. Note that this production would also be trivial to induce (and that this would be desirable for a larger set of features).

Thus, initially, the decision tree consists of an empty root node. On the first trial, since there are no applicable productions, the induction mechanism proposes a guess – blindly choose one of the available responses, either A or B. (Although the categories, and therefore the operators, are fixed for this example, adding new operators to the list dynamically is trivial.) This new classification rule is then added to the ruleset. Here is the first induced production for one run:

```
(p Induced-Production5098
   =goal>
      isa FACE
    - state Encoding
      guess nil
 ==>
   =goal>
      guess B
 )
```

The meaning of this production is that, if the task is to categorize a face, and the system is not currently encoding a feature, and if a decision has not been made, decide on category B. This is sufficient to bootstrap the operation of the system. Further induction depends on the two EPAM derived processes, familiarize and discriminate (note that anecdotal experience suggested that neither process by itself was sufficient to produce the desired learning effects).

If the categorization is correct, no learning will happen (or, more to the point, none is necessary since performance is perfect). However, upon making a mistake (signaled through explicit feedback), the familiarize and discriminate processes are invoked to induce productions to improve the judgment (redundant productions are not created). *Discriminate* compares the encoded features to the features of the face on the display and finding a feature that has not been previously encoded (i.e., selecting an unencoded feature to encode). This results in the following information-seeking production:

```
(p Induced-Production5102
   =goal>
      isa FACE
    - state Encoding
      f2 nil
      guess nil
 ==>
   =goal>
      state Encoding
      f2 ?
 )
```

This production, created upon incorrect categorization, seeks to encode feature 2. *Familiarize*, on the other hand, takes the feedback (which gives the correct category) and combines that with the known cues to form the following classification production:

```
(p Induced-Production5101
   =goal>
      isa FACE
    - state Encoding
      guess nil
 ==>
   =goal>
      guess A
 )
```

That is, the feedback could mean that everything in the world is an A, or it could mean that there is some structure in the cues and that they should be attended. In the Brunswick faces task, inevitably, the system will make another mistake and again invoke these two processes, resulting in more information seeking for cues that help correctly predict category. This process continues throughout training.

While discrimination is only invoked by errors (i.e., it is impasse driven), familiarization is invoked by all trials (see below for a discussion of why this does not overwhelm the system with rules). The familiarize process takes features that were not used in the decision process, but that were encoded, and combines them into new rules. This captures the transition from dealing with general rules to eventual memorization of specific examples – in a complex domain with exceptions that must be noted by their individual features such rules are needed to successfully complete the task. The following rule was induced to handle one such exception in the Brunswick faces task:

```
(p Induced-Production5194
   =goal>
      isa FACE
    - state Encoding
      guess nil
      f1 1
      f2 1
      f3 0
      f4 0
 ==>
   =goal>
      guess B
 )
```

This captures the fundamental nature of the symbolic level of learning in this system. However, interesting learning also occurs at the sub-symbolic level. In this case, what is learned are utilities and costs. As discussed above, utility, which is the ratio of successes to total trials, biases production selection in favor of productions that have had more successes. The utility for the production to blindly select category B for the above run is presented below:

```
Parameters for production Induced-Production5098:
 :Chance  1.000
 :Effort  0.250
 :P  0.571
 :C  0.750
 :PG-C 10.679
 :Successes  (8.0)
 :Failures  (6.0)
 :Efforts (10.5)
```

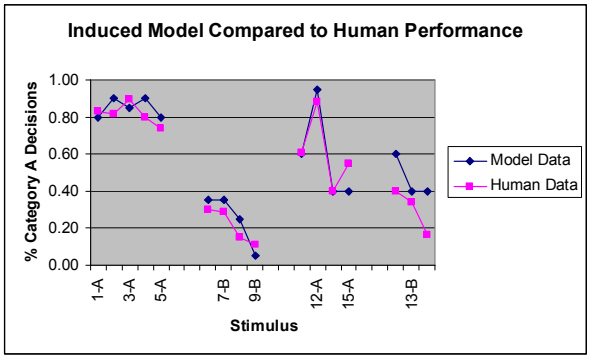
This production, whose initial expected gain was 19.75, now has an expected gain of 10.7 (after 8 successes and 6 failures). The value of the expected gain, other than being positive tells us little – it is its relative value compared to competing productions that matters. Here is the utility of the competing production that attempts to encode feature 2:

```
Parameters for production Induced-Production5102:
 :Chance  1.000
 :Effort  0.250
 :P  0.774
 :C  1.453
 :PG-C 14.019
 :Successes (82.0)
```

:Failures (24.0)
:Efforts (154.0)

Given the higher expected gain for encoding this feature, this production will dominate the production that simply guesses blindly. This was, however, achieved through setting a bias towards success for feature encoding. This is touches upon a critical issue in strategy selection, especially in developmental terms: new strategies are preferentially selected before they are capable of being executed successfully (e.g., Siegler, 1989). This is accounted for in the present work by setting a bias towards success for the information-seeking strategy (see Lovett, 1998, for a discussion of initial biases in strategy selection): each new information-seeking production is given credit for 30 successes and 10 failures, which gives a 75% probability of success. In any environment that is not completely random, collecting more information to act on will eventually result in correct decisions, so this bias does have a clear basis in reality.

The goal here is an initial qualitative fit, based on learning the structure of the task, with the purpose of exploring the induction mechanism. Given that, we will start with the overall results and then explore the induced model and the induction process in more detail.



The model results show the correspondence between the proportion of category A decisions for the stimuli made by human participants in the Smith and Minda meta-analysis and 20 individual runs of the model induced using the method developed here. Overall, the induced model captures not only the qualitative nature of the human data, but also an unexpectedly good quantitative fit (R^2=.89).

4. Discussion

In developing this system, stress was placed on fixing as many ACT-R parameters as possible at their default value (including noise and the value of goal). The main exception was the default action time. This was set to 0.25 to represent the coarse nature of the feature encoding productions and response production, which actually represent eye movements, attention shifts, and motor preparation and execution (250 ms is generally accepted as lower limit on the shortest time to process and respond to a stimulus in a lab experiment). Since the cost estimate is based on this block of activities, the learning would be identical whether one 250ms production executes, or five 50 ms productions execute, so the learning model described here will accommodate finer-grained productions. However, since the learned expected cost of an action does depend on time, this parameter is significant.

Generalization is clearly predicted and demonstrated by the current model: initial rules are maximally general. Specialization is only achieved by first developing and then incorrectly applying more general productions, and those productions higher up in the decision tree still exist. That is, productions are never removed from the current system. Thus, if a production is too specific for a new case, the more general production may have an opportunity to match even if it is not typically as successful as the more specific one.

Initially, there was a concern that the 'always on' nature of the familiarization process would lead to an out-of-control cycle of rule creation that would not allow for sufficient experience with any of the rules. However, since duplicate rules are not created, the familiarize process often proposes the creation of a rule that already exists, and rule creation slows down significantly with experience.

This method looks promising, and suggests a variety of extensions that could be accomplished through further development. One obvious improvement is to extend the induction to produce fine-grained productions that move attention, encode visual locations, press-keys, etc. In addition, the parallels between several strategy selection models and the current model are somewhat striking. In particular, given the surprising similarity in model structure and content, it appears straightforwardly possible to induce the canonical Building Sticks Task model (Lovett, 1998) using the present method. Another potential extension would be broadening the scope to include hierarchical representations (EPAM is naturally recursive and suggests this); an ACT-R slot could be encoded and recognized based on re-entering the recognition process – in this case a "feature" would actually become a particular chunk.

Potentially as interesting as method refinements are application to other domains. For example, this method is suited to any learning-by-example paradigm that involves categorical inputs and outputs. Student modeling could be achieved by treating the information

that the student has access to as the input data (e.g., features on a computer screen), and the responses they make (e.g., key presses) the output data. The learning model then would attempt to make the same mapping as the student. Expert modeling could follow an identical process, by using the expert's data stream instead, or, alternatively, by using perfect feedback to train the system.

In all of these cases, overfitting can become an issue, since the method does generate a significant number of productions. One way to deal with this is to enable the decay of experiences with time. In this case individual noisy examples will not produce much learning, while the more general learning that gets repeated and practiced would solidify with time.

5. Conclusions

The EPAM family of models is somewhat out of vogue in the cognitive modeling community, yet it represents one of the most thoroughly validated models of learning and concept attainment produced by the field. Rather than speculate why this might be the case, it is perhaps more useful to point out that this may well be a mistake. One clear conclusion from this project is that there is great value in applying this past work – the hard part involving the determination of effective mechanisms is often done already, and in this case, by leveraging a long tradition of experimental validation, there is some expectation of cognitive validity of the resulting models.

There are certainly alternative methods that exist for solving problems like this, including but not limited to statistical methods, connectionism, nearest-neighbor classifiers, case-based reasoning systems, etc. Many of these methods, however, lack the computational efficiency of decision-tree methods, which in part explains their popularity in data-mining applications.

The combination of utility learning with decision tree induction provides robust behavior – it makes the process noise tolerant, since many sub-optimal productions can be learned and withered away. Further, an inductive bias towards shorter decision-tree hypotheses is forced in this domain by the cost (in time) of encoding more features (i.e., Occam's razor has a cognitive cost explanation here). The learning that does occur, though based on the EPAM model of learning, when applied to productions looks surprisingly like SOAR style impasse-driven learning (Newell, 1990). In this case, there is no specific knowledge about which operators might be more applicable, and instead possible operators are chosen randomly, but knowledge guided selection of operators for new rules is certainly a viable extension to the current method.

The developed system demonstrates always-on learning: the learning slows with experience, but does not interfere with regular processing so there is no need to propose different modes of operation (i.e., learning mode vs. performance mode). Its learning scale is also consistent with human cognition, in that it requires tens of trials, not hundreds or thousands of trials, to learn to perform tasks like the one demonstrated here.

There are several open questions about how to best extend this method, and especially whether it is possible or practical to extend it to continuous outputs. Some rule-based systems have achieved this kind of functionality through the parallel firing of actions, and the averaging of their effects. It might also be possible to link the action of an ACT-R production to a blended retrieval, thereby achieving the same effect.

In general, however, this approach appears promising, both as a reliable means of constructing agents that behave appropriately, and as a means for producing behavior models with very little programming: the model emerges from the task and the architecture.

6. References

Anderson, J. R., & Betz, J. (2001). A hybrid model of categorization. Psychonomic Bulletin and Review, 8, 629-647.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y . (2004). An integrated theory of the mind. Psychological Review 111, (4). 1036-1060.

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

Anderson, J. R. & Matessa, M. (1998). The rational analysis of categorization and the ACT-R architecture. In M. Oaksford & N. Chater (Eds.) Rational models of cognition, pp. 197-217. Oxford: Oxford University Press.

Best, B. J. (2005). A Model of Fast Human Performance on a Computationally Hard Problem. In Proceedings of the Twenty-seventh Annual Conference of the Cognitive Science Society.

Best, B. J. & Lebiere, C. (2003). Spatial Plans, Communication, and Teamwork in Synthetic MOUT Agents. In Proceedings of the 12th Conference on Behavior Representation In Modeling and Simulation.

Best, B. J. & Lebiere, C. (2006). Cognitive agents interacting in real and virtual worlds. In R. Sun (ed.), Cognition and Multi-Agent Interaction:

From Cognitive Modeling to Social Simulation. Cambridge University Press; 186-218.

Best, B. J. & Lovett, M. (2006). Inducing a Cognitive Model from Examples Provided by an Optimal Algorithm. In Proceedings of the 7th International Conference on Cognitive Modeling.

Best, B. J., Lebiere, C., and Gacy, M. (2005). Leveraging a Cognitive Architecture to Help a Robot Walk and Chew Gum at the Same Time. In Proceedings of the 12th Conference on Behavior Representation In Modeling and Simulation.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Belmont, CA: Wadsworth International Group.

Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science*, 8, 305-336.

Gluck, K. A., Staszewski, J. J., Richman, H., Simon, H. A., & Delahanty, P. (2001). The right tool forthe job: Information processing analysis in categorization. Proceedings of the 23rdAnnual Meeting of the Cognitive Science Society. London: Erlbaum.

Gobet, F., Lane, P.C.R., Croker, S., Cheng, P. C-H., Jones, G., Oliver, I., and Pine, J. M. (2001). Chunking mechanisms in human learning. Trends in Cognitive Science, 5:236-243.

Gobet, F., and Simon, H. A. (1998). Expert chess memory: Revisiting the chunking hypothesis, Memory, 6:225-255.

Holland, J. (1975). Adaptation in Natural and Artificial Systems, University Press, Ann Arbor, Michigan.

Lee, F. J. & Anderson, J. R. (1997). Learning to Act: Acquisition and optimization of procedural skill. In Proceedings of the 19th Annual Conference of the Cognitive Science Society, pp. 418-423. Mahwah, NJ: Erlbaum.

Lovett, M. C. (1998). Choice. In J. R. Anderson, & C. Lebiere (Eds.). The atomic components of thought, 255-296. Mahwah, NJ: Erlbaum.

Mitchell, T. (1997). Machine Learning. McGraw Hill.

Newell, A. (1990). Unified thoeries of cognition. Harvard Press.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the lawof practice. In J. R. Anderson (Ed.), Cognitive skills and their acquisition (pp. 1-55).Hillsdale, NJ: Erlbaum.

Quinlan, R. (1993): C4.5: Programs for Machine Learning. Morgan Kaufmann, San Diego.

Siegler, R., and Jenkins, E. (1989). "Strategy Discovery and Strategy Generalization," in How Children Discover New Strategies. Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 1-20.

Smith, J. D., & Minda, J. P. (2000). Thirty categorization results in search of a model. Journal of Experimental Psychology: Learning, Memory, and Cognition, 26, 3-27.

Taatgen, N.A. (2003). Learning rules and productions. In: L. Nadel (Ed.), Encyclopedia of Cognitive Science, vol. 2 (pp. 822-830). London: MacMillan.

Taatgen, N.A. & Lee, F. J. (2003). Production Compilation: A simple mechanism to model Complex Skill Acquisition. Human Factors, 45(1), 61-76.

Wallach, D., & Lebiere, C. (2002). On the role of instances in complex skill acquisition. In Proceedings of the 43rd Conference of the German Psychological Association.

Author Biographies

Brad Best is a Staff Research Scientist at Micro Analysis and Design. He received his Ph.D. in Cognitive Psychology from Carnegie Mellon University, where his focus was on the development of computational models of cognition, and his M.S. in Computer Science from Central Michigan University where he worked on connectionist models of vision. At MA&D, Dr. Best has applied cognitive modeling techniques to domains with spatial and temporal dimensions including robotics and virtual environments, and has worked to integrate cognitive models with mainstream AI approaches including machine-learning, statistical clustering, and planning.