Artificial Evolution of Syntactic Aptitude

John Batali
Department of Cognitive Science
University of California at San Diego
La Jolla, CA 92093-0515
batali@cogsci.ucsd.edu

Abstract

Populations of simple recurrent neural networks were subject to simulations of evolution where the selection criterion was the ability of a network to learn to recognize strings from context free grammars. After a number of generations, networks emerged that use the activation values of the units feeding their recurrent connections to represent the depth of embedding in a string. Networks inherited innate biases to accurately learn members of a class of related context-free grammars, and, while learning, passed through periods during which exposure to spurious input interfered with their subsequent ability to learn a grammar.

Introduction

Human languages are quite complex, yet children are able to learn to understand and speak their native languages before reaching the age of four. One of the most popular (albeit controversial) responses to this puzzle is to hypothesize that humans are born with innate biases to be able to recognize certain kinds of linguistic structures. These innate biases may be specific to language (Chomsky, 1987), or may be part of general cognitive mechanisms. In either case, the proposal of innate biases raises difficult questions. Precisely what sorts of biases are there; how could they be encoded genetically; how could they evolve?

We are exploring these issues by means of computational simulations of the evolution of neural networks capable of recognizing and generating strings from formal languages. The class of "context-free" languages (Hopcroft & Ullman, 1979) is the simplest class which exhibits recursively embedded structure. In our simulations, networks emerge which are capable of recognizing strings generated by simple context-free languages. The networks use some of their recurrent connections to represent the depth of embedding in the strings. The members of a population of networks trained on strings generated by a class of related context-free languages eventually evolve a bias to efficiently learn languages from the class.

Networks, Grammars, Evolution

Recurrent neural networks (as shown in figure 1) have been shown to be capable of recognizing and generating strings generated by formal grammars (Serven-Schreiber, et al., 1988; Elman, 1990). A typical training regimen (and the one used in the experiments reported here) consists of presenting the symbols from a string generated by the grammar sequentially to the input units of the network. Activation is then fed forward through the network. The activations of the output units are taken as a prediction of the next symbol in the string. The prediction is compared with the actual symbol that appears next, and the error between the predicted and correct value is backpropagated through the network to adjust the connection weights between units.

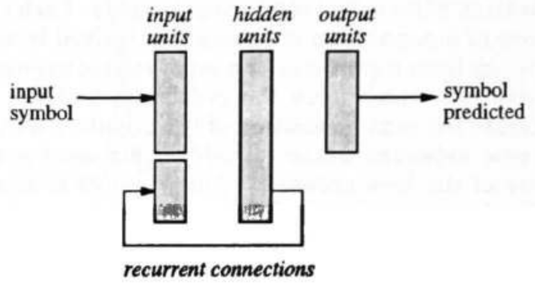

Figure 1: A simple recurrent network for recognizing strings generated by a grammar. Symbols from the string are presented sequentially to the input units. The network is trained to predict the next symbol in the string. Recurrent connections from hidden units feed back to the input layer. Each unit in a layer has connections to each of the units in the next layer; these connections are not shown. The specific numbers of input units, hidden units, output units, and recurrent connections varied in the different experiments.

It has been shown that, in principle, a recurrent neural network can recognize any class of grammar (Pollack, 1991; Siegelmann, 1994). However it is not clear if a network has actually been trained to recognize a language generated by a context-free grammar, as opposed to a simpler approximation to the language. Giles, et al., (1990), have shown that networks equipped with an external stack can learn to recognize context-free grammars. To demonstrate that an unaccessorized simple recurrent network is really recognizing strings from a context-free language, one must show that the network has somehow been trained to keep track of the depth of embedding in the string.

Researchers in the nascent fields of "artificial life" and "evolutionary programming," have explored computational simulation of evolution by natural selection, both to better understand the mechanisms of evolution, and to use those mechanisms to discover solutions to difficult computational problems (Langton, 1989). A compelling intuition (first presented by Baldwin, 1896) is that evolution and learning can work synergistically: the members of a species can inherit biases from which learning can efficiently and reliably derive superior solutions to difficult problems. This intuition has been explored by simulating the evolution of populations of neural networks where the selection criterion is the networks' ultimate ability to learn some task (Hinton & Nowlan, 1987; Belew, et al., 1991).

In our simulations, the "genome" of a network is a specification of its initial connection weights. Each of the members of a population of networks is trained to recognize strings from a grammar or a set of related grammars. The networks which learn the grammars best are used to generate the next generation of the population by creating new networks whose initial weights are the same as those of the best networks. The genomes of some of the offspring networks are then "mutated" by modifying some of the weights randomly before the members of the next generation are trained on the grammars.

Learning a Simple Context-Free Grammar

A very simple context-free grammar consists of strings of the form $\mathbf{a}^n\mathbf{b}^n$, that is: some number of tokens of the symbol **a**, followed by the same number of tokens of the symbol **b**. The pattern exhibited by strings from this language corresponds to what are called "center-embedded" constructions in natural languages, where some constituent, for example a noun phrase, must be matched with another constituent, for example a verb phrase, across some intervening material, for example a relative clause, that might exhibit the same structure.

Any machine that can recognize whether a string is in this language or not must somehow count up the number of **a**'s it sees, and count down each time it sees a **b**. A recognizer for this language must also keep track of which of two states it is in: an initial state corresponding to the sequence of **a**'s, and a second state corresponding to the sequence of **b**'s. During the period the machine is in the initial state, it may see either an **a** or a **b**; if it sees an **a**, it remains in the initial state and increments the counter; when it sees a **b**, it enters the second state and decrements the counter. While in the second state, it must only see **b**'s, and must decrement the counter for each one. While a pushdown stack is needed to recognize an arbitrary context-free language, any device capable of recording a value that can be incremented, decremented, and compared with zero, will suffice for this simple grammar.

The networks used for learning this grammar had 3 input units, 10 hidden units, 3 output units and 7 recurrent connections from the hidden units to the input layer. Training consisted of presenting the networks strings from the $\mathbf{a}^n\mathbf{b}^n$ grammar preceded and terminated by a 'space' character. Prediction error was backpropagated through the network to update the connection weights. (The backpropagation learning rate for all of these experiments was 0.1. No momentum term was used.) Each network was trained for a total of 500,000 characters, which works out to about 33,000 strings. (The strings ranged in length from 4 to 26 characters, with an average length of 15 characters.)

To assess the difficulty networks would have learning this grammar, 513 randomly initialized networks (initial weights were uniformly distributed between -1 and $+1$) were trained on strings from the grammar. Their performance on a set of test strings was then measured. The average of the networks' prediction errors per character was was 0.244, with a standard deviation of 0.031. The best network in this set achieved an average prediction error of 0.168.

The performance of a randomly initialized network after training on strings from the $\mathbf{a}^n\mathbf{b}^n$ language is shown in figure 2. The values plotted are the activation values of the network's output units, after the characters from the string are input to the network. The network correctly predicts the appearances of the **a** character for the first half of the string, and once the first **b** is seen, the prediction for **a** drops to zero. In the second half of the string, the prediction for **b**, initially high, begins to drop off, and the the prediction for the 'space' character begins to rise. Although these predictions accord with the statistics of the training strings, the network has hardly learned the grammar very well. It should be predicting only **b** until as many **b**'s as **a**'s have been seen, at which point it should predict only the 'space' character. The behavior of the network shown in figure 2 is representative of virtually all of the randomly initialized networks.

The evolutionary simulation was begun by creating a a population of 24 recurrent networks. Each network was given random initial weights. For each generation of the simulation, each network was trained on strings from the language, seeing a total of 500,000 characters, as above. The network was assigned a fitness value that depended on the average prediction error it obtained when the network was then tested on strings from the grammar. At the end of each generation, the 8 best networks were retained into the next generation and their connection weights reset to the values stored in their genomes. In addition, the genomes of those top networks were copied into the remaining 16 networks and modified by a mutation operator which changed some of the initial weights by small random value. The networks of this new generation were then trained on the grammar again.

By the 155th generation of this simulation, the best network had an error of 0.151 and the average error for all of the networks in the population was 0.179. This is 2.2 standard deviations better than randomly initialized networks.

Figure 3 shows the activation values of some of the units of a network that evolved in this experiment, after training on strings from the grammar, while the network is shown strings of various lengths. For each string, the network strongly predicts the character **a** until it sees a

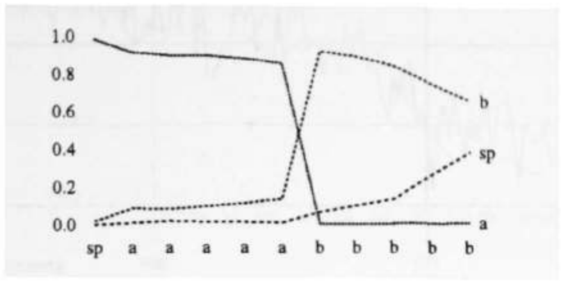

Figure 2: Performance of a randomly initialized network after training on 500,000 characters from strings in the $a^n b^n$ language. Values plotted are the activation values of output units, after the characters from the string are presented to the network. The symbol 'sp' stands for the string-delimiting 'space' character. This network had an average prediction error of .251 for the set of test strings — approximately the median value for a set of 513 randomly initialized networks.

b, at which point it predicts b until the end of the string, where it predicts the 'space' character.

The plots labeled rec on the graphs in figure 3 show the value passed from a unit in the hidden layer of the network back to the input layer via one of the network's recurrent connections. These values show that this unit seems to be behaving like a counter: Each a increases its activation until b is seen, after which the value decreases. When the activation value of the unit approaches zero, the network predicts the sp character. The behavior of this unit is precisely what is required for the network to recognize strings of the form $a^n b^n$.

Learning From a Class of Grammars

The population of networks in the first experiment was able to combine evolutionary search with backpropagation learning to learn a particular language. Obviously an ideal solution would have been for a network to need no training at all — it would be "born" with its connections already appropriate to recognize the target language. This solution is not available to humans. Children are able to learn whatever language they are exposed to, no matter what language their parents spoke. There are apparently no innate biases to learn *particular* human languages. It has been suggested, however, that all human languages share certain abstract structural features, and children are born with an innate bias to learn languages with those structural features (Chomsky, 1987).

A second evolutionary simulation was performed to explore this idea. In this experiment, each member of a population of networks was trained on a language from a class of 36 related context-free languages. The network's ability to learn the language was used to compute its fitness value, and hence whether it survived into the next generation and reproduced. But in the next generation the network and its offspring would, in general, face different languages. So to do well, networks must develop biases not for a specific language, but for the class of

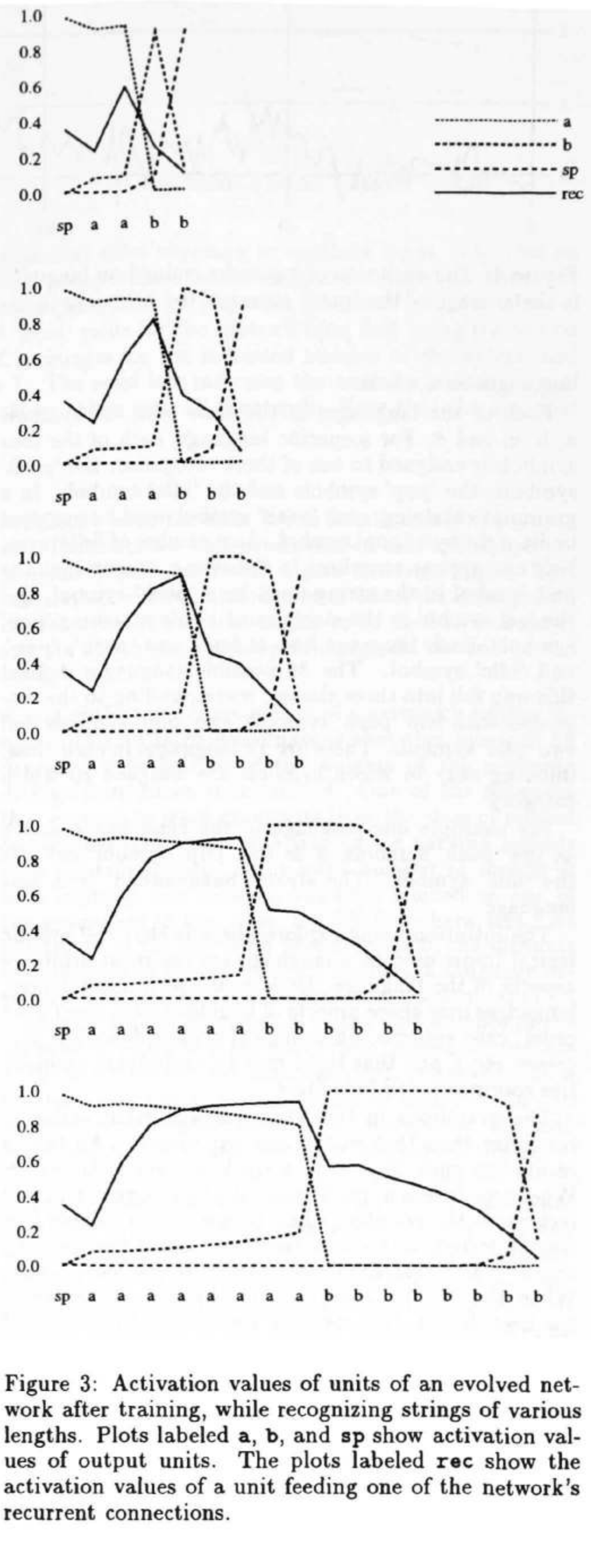

Figure 3: Activation values of units of an evolved network after training, while recognizing strings of various lengths. Plots labeled a, b, and sp show activation values of output units. The plots labeled rec show the activation values of a unit feeding one of the network's recurrent connections.

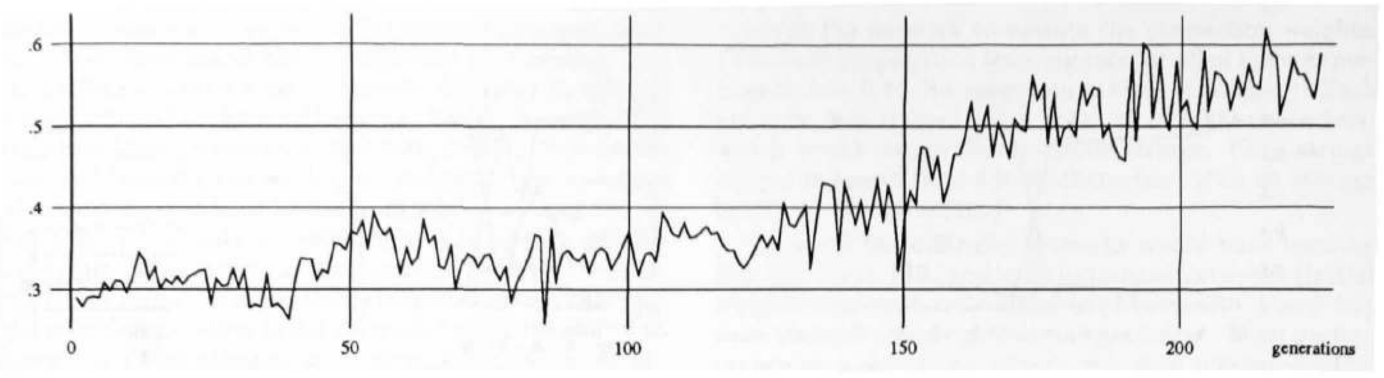

Figure 4: The evolution of networks trained on languages from the class of context-free grammars. The value plotted is the average of the 'pred' scores of the networks in each generation.

languages as a whole.

Each of the languages in the class uses the symbols **a**, **b**, **c**, and **d**. For a specific language, each of the four symbols is assigned to one of three categories: the 'push' symbols, the 'pop' symbols and the 'idle' symbols. In a grammatical string, each 'push' symbol must be matched on its right by a 'pop' symbol. Any number of 'idle' symbols can appear anywhere in the string, except that the first symbol in the string must be a 'push' symbol, and the last symbol in the string must be its matching 'pop' symbol. Each language had at least one 'push', 'pop', and 'idle' symbol. The 36 possible languages defined this way fall into three classes, corresponding to the languages with two 'push' symbols, two 'pop' symbols and two 'idle' symbols. There are 12 languages in each class, differing only in which symbols are assigned to which category.

For example one language in the class has **a** and **b** as the 'push' symbols, **d** as the 'pop' symbol and **c** is the 'idle' symbol. The string '**baadcadcdd**' is in this language.

The intuition being explored here is that the specific lexical items used in a language are the most arbitrary aspects of the language. Underneath lexical differences, languages may share aspects of linguistic structure (word order, case systems, phonological or morphological processes, etc.), and that there may be underlying regularities common to all languages.

The grammars in this class are somewhat easier to recognize than that in the first experiment. All that a recognizer must do is to keep track of one counter value. Whenever it sees a 'push' symbol, the recognizer should increment the counter; when it sees a 'pop' symbol, it should decrement the counter; and when it sees an 'idle' symbol, it should keep the counter at the same value. When the counter reaches zero, the end of the string has been found. The network must learn to do this, of course, as well as to learn the mappings from the actual symbols to their categories.

In this experiment, networks with 5 inputs, 10 hidden layers, 5 outputs and 1 recurrent connection were used. The single recurrent connection was used in order to make the language-learning task as difficult as possible for the networks, given the relative simplicity of these grammars compared to the $\mathbf{a}^n\mathbf{b}^n$ grammar.

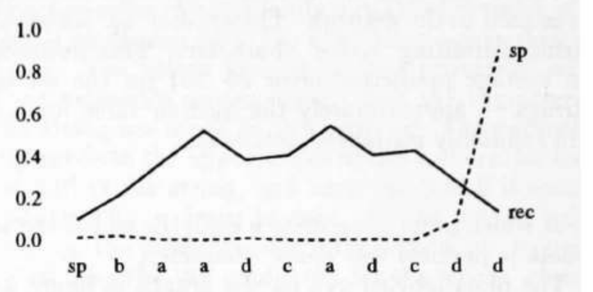

Figure 5: Activation values of units in one of the networks that evolved to learn languages from the class of context-free grammars, after being trained on a specific language from the class. For this language, the symbols **a**, and **b** are 'push' symbols, **d** is the 'pop' symbol, and **c**, is the 'idle' symbol. The solid line plots the activation value of the unit feeding into the single recurrent connection of the network after the given character is seen. The dotted line plots the network's prediction of the end of string 'space' character.

The networks were trained on strings of characters as in the first experiment. However in this simulation, the performance of a network was assessed by computing the average value of its 'space' output unit at the end of each of a set of test strings. This value will be referred to as the 'pred' value of the network. Ideally, it ought to be 1.0 — indicating that the network has correctly predicted the end of each of the strings. This value is more informative than the average per-character error for these languages, as the the specific character that can follow another within a string is much less constrained than in the earlier grammar. As before, the per-character prediction error was used for backpropagation training.

To see how well networks do recognizing this grammar with no evolutionary search, 436 randomized networks with varying numbers of hidden units and recurrent connections were trained for 500,000 characters each on grammars from the class. The average 'pred' value was 0.375 with a standard deviation of 0.117. The best network had a value of 0.827. Of 103 networks with a single recurrent connection, the average 'pred' value was

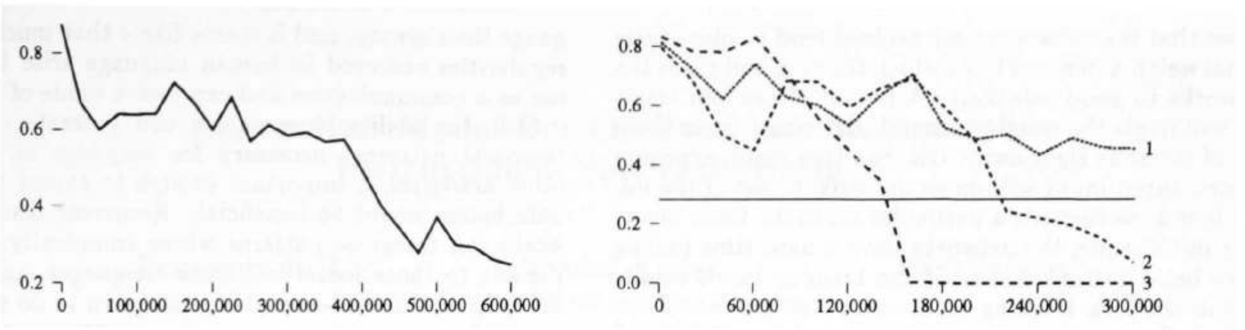

Figure 6: Performance of an evolved network in learning a grammar after exposure to spurious input. The plot on the left illustrates the final 'pred' value for the network after first being trained on the indicated number of characters from random sequences, and then trained on 500,000 characters generated by a specific grammar from the class of context-free grammars. The plot on the right shows the final 'pred' value for the network after first being trained on strings from a language from each of the three subclasses of languages for the indicated number of characters, and then trained on 500,000 characters from a language of class 1. The solid line indicates the average performance of randomly initialized networks. Both plots display average values taken over 20 runs each. Note that the scales of the abscissas are different in the two plots.

0.280 with a standard deviation of 0.057; the best network had a value of 0.457.

The evolutionary simulation was organized in a similar fashion as the first experiment, with the 'pred' value of each network used to compute its fitness. Figure 4 shows a record of the simulation. The value plotted at each generation is the average 'pred' value for the networks in the population. After an initial period of relatively aimless search, the members of the population steadily improve their aptitude at learning the grammars. (This particular run took seven days on a Sun Sparc Station 10. It was terminated due to the submission deadline for this paper.)

Figure 5 illustrates the performance of one of the networks that evolved. The network was trained for 500,000 characters from strings of a specific language in the class. The values plotted are the activation of the 'space' output unit and the unit feeding the recurrent connection after the indicated character is presented to the network. As in the first experiment, the unit feeding the recurrent connection is used as a counter: the 'push' symbols a and b increment its activation value; the 'pop' symbol d decrements it, and the 'idle' symbol c modifies it only slightly. When the activation value decreases far enough, the network signals the end of the string.

After 150 generations, the average performance of the entire population is better than the best performance for the randomly initialized networks. This illustrates that the networks are indeed developing an innate bias towards learning the languages in the class.

A network from a late generation of the simulation was tested on a string from a grammar without having been trained on any input at all. For each character in the test string, the networks 'space' output unit is strongly activated — it is almost 1.0 for each character of the string — and the recurrent activation is very close to 0.0 throughout the string. Thus the "newborn" network is apparently hypothesizing the minimal language consistent with the data it has seen: namely the empty set. Furthermore, this prediction of the 'space' symbol is innately associated with a zero recurrent input. This association will be reinforced later when the unit feeding the recurrent connection begins to be used as a counter, and its activation reaching zero signals the end of the string.

A straightforward consequence of a network's having innate biases to learn grammars is that this bias can be degraded substantially if the weights of the networks diverge from those that evolved. One of the networks that evolved to learn grammars from the class of related context-free grammars was trained for varying periods with random strings. After some number of rounds of such training, the network was then trained on one of the grammars in the class, for 500,000 characters. As illustrated in the left hand plot of figure 6, the ultimate performance of the network on learning a language decreases with the length of exposure to random input.

More dramatic degradation in the performance of the network is observed if the network is trained on one language for a while, and then switched to another, as is illustrated in the right hand plots of figure 6. In this experiment, the network was first trained for the indicated number of characters on a language from one of the three subclasses of languages: class 1 had two 'push' symbols; class 2 had two 'pop' symbols, and class 3 had two 'idle' symbols. After this initial training, the network was trained for 500,000 characters on a language from class 1. (If the initial training was on a class 1 language, a different class 1 language was used for the subsequent training.) As can be seen, the different language classes differ in their effect on the subsequent learnability of another language. If the languages were similar (class 1 first, then a different class 1 language), the degradation in performance was relatively small. Languages which were more different had a more substantial effect.

Apparently what is happening here is that the innate biases that the networks have evolved tend to place their initial weights in zones from which training will guide the networks to good solutions. A period of random training will jiggle the weights around, but won't move them out of the zone right away. On the other hand, exposure to structured input will move the weights out of the initial bias zone towards a particular solution. Once out of that initial zone, the networks have a hard time finding ways back into it. Indeed if the training input results in the network learning to recognize properties of the strings that are not relevant for recognizing the second language (and to fail to recognize those that are), the final performance of the network can be below that of randomly initialized networks. These effects are reminiscent of the "critical periods" observed in the acquisition of some cognitive abilities, in particular, language (Newport, 1990).

Discussion

Further work is needed to analyze the initial weights of the networks that evolve to understand exactly how they encode the networks' innate biases to learn specific grammars. We have found in experiments in which networks are evolved and trained to recognize classes of simpler temporal patterns, that the networks eventually inherit initial weight values which implement operations useful in recognizing any specific member of the class.

We are also interested in attempting to evolve networks that can recognize more complex syntactic structures. Another avenue of experiment will involve training populations of networks to exchange strings with each other in a communication task, to see what sorts of syntactic regularities arise spontaneously.

The enormous investment in computational power illustrated by searches like that shown in figure 4 make it questionable whether evolutionary searches really are a good way to obtain networks which can recognize strings from grammars. However the point is not to construct individual networks capable of recognizing specific languages. Instead, these results demonstrate that the members of a species can evolve biases to learn different, but related, languages reliably and efficiently. These biases would be crucial for the members of a species whose subgroups face different, but related, communicative situations.

In our simulations, the initial weights of the networks are fully specified by the networks "genotype." This is not biologically accurate — the information available in the human genome is capable of specifying only a minute fraction of the actual connectivity in the brain. On the other hand, it is possible that even such a small fraction of specified initial connectivity could strongly influence the solutions that a general-purpose learning mechanism would find. We are exploring sparser representations of the genotype and more realistic models of the interactions between genetics, development, and learning.

While this research was motivated by properties of human languages and the debate over the question of innate biases, it is important not to read to much into these results. For one thing, there is a lot more to language than syntax, and it seems likely that much of the regularities observed in human language arise from its use as a communicative and expressive mode of action.

Still, the ability to recognize and generate complex temporal patterns, necessary for language as well as other activities, is important enough to expect that innate biases would be beneficial. Recurrent neural networks can recognize patterns whose complexity is comparable to those found in human languages, and, with the help of evolved biases, they can learn to do so when exposed to examples.

Acknowledgment

I thank Jeff Elman for helpful discussions and expert spotting assistance.

References

Baldwin, J. M. (1896). A new factor in evolution. *American Naturalist,* 30, pp. 441–451.

Belew, R. K., McInerney, J., and Schraudolph, N. N. (1991). Evolving networks: Using the genetic algorithm with connectionist learning. In *Artificial Life II, SFI Studies in the Sciences of Complexity, Volume X.* C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, (eds.), Addison-Wesley, pp. 511–547.

Chomsky, Noam. (1987). *Knowledge of language: Its nature, origin, and use.* New York: Praeger.

Giles, G. L., Sun, G. Z., Chen, H. H., Lee, Y. C., and Chen, D. (1990). Higher order recurrent networks & grammatical inference. In *Advances in Neural Information Processing Systems 2.* D. S. Touretzky (ed), San Mateo CA: Morgan Kaufman.

Elman, Jeffrey, L. (1990). Finding structure in time. *Cognitive Science,* 14, pp. 179–211.

Hopcroft, J. E. & Ullman, J. D., (1979). *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley.

Hinton, G., & Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems,* 1, pp. 495–502.

Kolen, J. F., & Pollack, J. B. (1990). Back propagation is sensitive to initial conditions. *Complex Systems,* 4, pp. 269–280.

Langton, C. G. (1989). Artificial Life. In *Artificial Life.* C. G. Langton (ed.) Reading, MA: Addison-Wesley. pp. 1–48.

Newport, Elissa (1990) Maturational constraints on language learning. *Cognitive Science,* 14, pp. 11–28.

Pollack, J. B. (1991) The induction of dynamical recognizers. *Machine Learning,* 7, pp. 227–252.

Siegelmann, Hava Tova. (1993). *Foundations of Recurrent Neural Networks.* PhD Dissertation. Rutgers University, Graduate Program in Computer Science.

Serven-Schreiber, D., Cleeremans, A., and McClelland, J. L. (1988). *Encoding sequential structure in simple recurrent networks.* CMU Technical Report CMU-CS-335-87). Carnegie-Mellon University, Department of Computer Science.