Applying neural pruning to NEAT

Emily Dolson and Dasol Park

5/11/12

1 Abstract

Open-ended machine learning problems require various level of flexibility from learning agents. Stanley (2004) presented in his paper that incremental complexification in neural network structure usually results in more effective learning process, while subsequent researches by James and Tucker showed simplifying the structures often results in similarly good solution. In this paper, we test various methods of allowing more flexibility in evolving neural network structures and demonstrate it is not always optimal to allow one-way complexifying evolution from the most simple structure.

2 Introduction

Artificial neural networks (ANNs) are an important tool in adaptive robotics research, both because of their practical efficacy as a tool for classification and because of their biological plausibility. The latter is relevant because it means that robots controlled by neural networks might potentially be modelling the adaptation of life, and implies that further advances in our understanding of ANNs might yield solutions to highly complex problems. One of the central questions about ANNs, currently, is that of how to determine the best topology to use for a given task. A promising method of tackling this issue is to use a genetic algorithm to evolve a topology.

One of the most successful algorithms for evolving ANN topologies is NEAT (Stanley, 2004). The NEAT method starts out with a population of simple ANNs and applies a genetic algorithm to them. The options for mutation are: adjust the weight of a connection between nodes, add or disable a connection between nodes, or add a new node to the network. Nodes are added by splitting a connection between two other nodes. Thus, the neural networks in the population become gradually more complex. Stanley argues that this complexification occurs only when necessary to solve the problem, and thus it is the only necessary form of mutation.

In sharp contrast to this constant complexification, human brains go through two developmental stages - one in early childhood and one around puberty - in which a process called neural pruning occurs. In this process, vast numbers of neurons and synapses (connections between neurons) are lost. This enables the brain to function more efficiently and is generally regarded among biologists to be vitally important to development. Although most of the inspiration for ANNs has come from observations about human brains, neural pruning is typically ignored in attempts to evolve ANN topologies, such as NEAT. We hypothesize that this is a substantial omission - that including some sort of capacity for neural pruning will improve the efficiency and perhaps even the efficacy of ANNs produced by NEAT.

Subsequent research on NEAT has found evidence that allowing complexifying and simplifying evolution to occur at the same time can yield equivalently effective neural networks in less time (James and Tucker, 2004). James and Tucker implemented simplifying evolution of a network by interchanging the rate at which connections were added with the rate at which connections were removed, and blended (both simplifying and complexifying) evolution by adding a type of mutation that removes connections. On XOR evaluation and solving Tic-tac-toe task tests, both simplification and the blended method showed encouraging implications for the success of introducing larger-scale simplification into NEAT. In either case, it should also be noted that performances of different measures obviously depend on starting topology and specific parameter settings.

Much research into algorithms for pruning neural networks has already been done. However, the vast majority of this research focuses on deterministic strategies for removing connections from standard neural networks that arent evolving. Thus, it isnt really relevant in a more adaptive context. The gains from applying such algorithms (Jorgenson and Haynes, 2008), however, do serve as further evidence that neural pruning can be beneficial to ANN performance.

3 Background

3.1 Biological Neural Pruning

Biological neural pruning, also called synaptic pruning, has been recognized as an important component of development for a long time (Huttenlocher and Courten, 1987). At birth, babies are, in some ways, like a fully-connected neural network. They have connections between all cortices of their brains, many of which are unnecessary, a lot of which will soon be severed in the first wave of pruning (Low and Cheng, 2006).

3.2 NeuroEvolution of Augmenting Topologies

AI researchers attempt to model the functional aspects of evolving biological neurons by using artificial neural networks. With a specified number of input and output nodes, solving for an optimal model involves numerous trials with different weights assigned to connections that send numerical values between nodes. Considering biological examples of genomes continuously gaining and losing portions of chromosomes, the use of ANNs with fixed topologies dramatically limits the likelihood of finding optimal solutions that may exist with different structures. This kind of problem arises frequently in many of the more complex, open-ended problems, where estimating complexity of optimal solution by heuristics is more difficult.

Stanleys NEAT method attempts to solve this problem by combining the usual neural network with dynamic, expandable structure to simulate complexification in nature. Each genetic encoding include a list of gene nodes and connection nodes, organized by innovation number, a metric that keeps track of the order in which attributes developed and which attributes are homologous. This enables genomes in the population to fairly accurately align homologous regions when they perform crossover, such that only meaningful crossover events occur. NEAT also tries to protect innovations by the process of speciation, whereby individuals are grouped together according to dynamic distance function and primarily compete within the similarity group, so as to prevent interbreeding from keeping all structures too similar to each other, and to protect structural innovation that might not seem useful at first.

3.3 Problems with one-way complexification

While NEAT opens another search space for artificial neural network to explore (thereby allowing more flexibility), it certainly comes with a cost. First, researchers still need to specify beforehand a wide range of parameters based on human reasoning, such as probability of structural changes occurring and a threshold level for innovation protection. Second, there is no guarantee that the evolved structure is indeed optimal. Since mutation of weights and addition of hidden node/connections occur randomly, complexification in NEAT - and evolved structure as a result - is always path-dependant. In other words, even if NEAT found a complexity level α solution for certain neural network, that does not mean the NEAT algorithm explored all possible neural network structure with complexity less than or equal to α. Adding a pruning method in addition to complexification also will not solve this problem, but at least this allows evolutionary algorithm to explore greater search space of neural network structures.

Another possible problem arises when there arent significant differences between the performance of ANNs with more complex structure and those with less complex structure. Stanley believes that smaller structures optimize faster than larger structures, but this might not hold true in every case especially if given problem is complex. If a problem is unsolvable, smaller structures wont necessarily optimize faster. If optimization takes a lot of steps, Stan-

leys algorithm- since it rewards newly-added structures and rewards innovation - tends to remove those simpler structures from the population (stagnated), while the algorithm hasnt fully tried sufficient candidates within smaller structure.

Lastly, while Stanley did test an alternate version of NEAT that simplified instead of complexifying network structure, his simplification algorithm was fundamentally different from what we generally observe from biological neural pruning; it started out with an unevolved, fully connected, large neural network, to which he only applied mutations that removed nodes and connections. Simplification, for the most part, failed to achieve the level of performance attained by simpler fixed level topology evolution and solely-complexifying topology evolution (NEAT). That it was ineffective should hardly come as a surprise, as part of the success of NEAT likely stems from its ability to optimize a low-dimensional neural network as much as possible to reduce the amount of computation that must be done on the less efficient, larger network. For neural pruning to be effective, it must occur between successive waves of complexifying evolution.

4 Implementation and initial experiment

We used Python NEAT package used in the Spring 2011 CS81 course, provided by Professor Lisa Meeden, and modified it by adding delete connection and delete node methods. If executed, our delete connection method randomly selects a node inside the ANN and removes it, but only in case when the removal action does not isolate any of the output nodes (as this would lead to an impossible to use toplogy). Likewise, our delete node method only applies to non-input/output nodes, and restores all the connections that were connected through the removed hidden nodes (a version that deleted the connections as well ended up being far too destructive to be practical).

We hypothesized that alternating between periods of complexification and simplification (pruning) will produce more efficient, possibly better fit genotypes, since it amelliorates the path-dependency problem and more closely resembles biological mechanisms for brain maturation. Besides this, we wanted to give more flexibility in the ways that NEAT could evolve neural net topology. Our initial goal, therefore, was to find out when to switch between the two methods without prior domain information
By starting from the most simple neural network structure, a fully-connected, 2-layer, feedforward topology, original NEAT attempts to reward simplicity of the solution while exploring various possibilities. Since the simpler structure should be preferred in cases of similar performance, it make sense to start exploring network structure space from the most simplest potential solution. In this case, our newly added pruning methods are unlikely to be actually executed early on, since there arent many valid nodes and connections to remove from.

To speed up the experiment and make clear comparisons between the experimental and control groups, we initially set up two additional parameters (α, β) to describe the window

length of two steps. As certain a level of complexity is needed for pruning to effectively make any changes, α was set up to indicate the number of generations the ANN was required to evolve in the complexifying direction before switching to pruning mode. β, on the other hand, specified the number of generations the ANN was required to keep evolving in the same direction (either complexifying of pruning) after switching. Since the probability of adding/deleting connections was generally set low (¡ 10%), this continuity was deemed necessary at first. At the end of each β generation, NEAT randomly selects one of the two directions and keeps evolving in that direction for another β generations. For example, the population with $(\alpha, \beta) = (1, 1)$ will randomly choose a direction on each generation,while if $(\alpha, \beta) = (M, k)$ for some M ¿ (maximum number of generations) will serve as our control group, as it never gets out of the complexifying-only stage.

In our later experiment, we decided to add even more flexibility to our learning algorithm. Assuming the members of each species (determined by Stanleys distance function) will similarly benefit or hurt from the chosen direction, we let each species decide which direction to choose to evolve, rather than forcing the same direction for the entire population. We implemented an approach similar to simulated annealing, by changing the direction each time the increase in fitness stagnates.
Finally, we let the probability of adding/removing nodes and connections be varied so that its negatively correlated with populations average fitness. We assumed this will effectively stabilize the speed of neural network structure changes as the population gets better (higher fitness). In the next section, we will cover in detail why these features were added.

5 Experiment

5.1 XOR

We first tried comparing the effectiveness of the original NEAT algorithm and our version of NEAT by solving the XOR problem. As XOR involves solving for a single binary output value given two binary input values, starting structure for evolution contained 2 input nodes, 1 output node connected feed-forward, without any hidden nodes or connections. The first group of ANN were required to evolve by complexification for the first 30 generations, and then given a chance to switch directions every 10 generations. (α, β) = (30,10). The second group used original NEAT, and third group randomly selected whether to complexify or simplify every single generation. We started with population of 100 individuals, evolved for 100 generations, and repeated this run 100 times to analyze the results. In the graph above, red lines indicate fitness attained by best-performing individuals in each generation,while blue lines track the average fitness of all 100 individuals in the population. In all three groups, a fitness over 99% was obtained in around 32th generation, on average with 4 hidden nodes and 13 added connections. This similarity in performance, however, does not mean all three methods found optimal solution for solving XOR. Neither NEAT nor our modified version of NEAT were able to minimize the dimension of the solution, as after all, we know that XOR

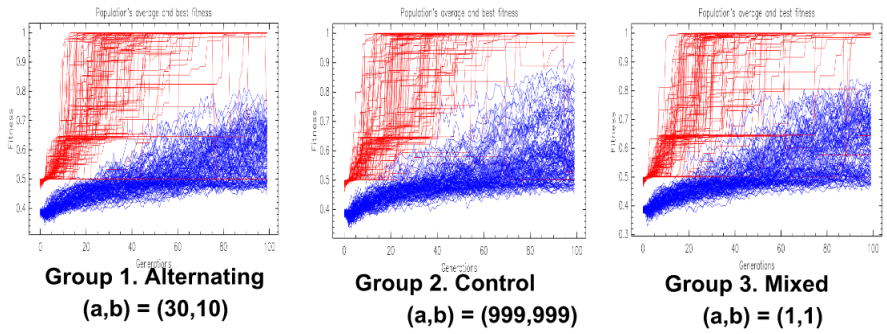

Figure 1: Performance of all three groups on the XOR task.

is solvable with one additional hidden node added to the starting ANN structure.

In fact, we could also observe that complexifying the structures for extended periods of time will often hurt overall performance. The best performing individuals were taken out of the population, as, after a certain number of generations they would be unable to improve and so be deemed stagnated. Fortunately, this problem is really just an artifact of forcing NEAT to run for far longer than it took to find a solution. Normally, the algorithm would terminate once such a high fitness was attained.

5.2 Light-seeking

In our second experiment, we compared performances of various versions of NEAT in a simplified version of Stanleys original coevolution experiment. Instead of implementing two separate agents competing for lights, we evolved a single agent whose goal was to survive for the longest time with lights serving as their energy source. Each agent was given 3 inputs (2 light sensors, 1 warning sensor activated when agent is too close to wall in front), with outputs being left and right motor values to proceed in next step.

As we found from the previous experiment NEAT provides constant pressure to keep changing structures whether or not an individual has high fitness, we made each structural change probability dependant on the average fitness of that species population. For example, if we know that a given species is performing relatively well, we reduce the probability of structural changes and focus more on adjusting connection weights. Another way to add more adaptability to our NEAT method was to let each species, rather than whole population, determine the direction of structural changes. While our previous threshold method specified phase window lengths of complexification and simplification, our new method, called Stag-Prune, focuses more on whether stagnation has occurred for a given species by continuing single-directional structural evolution. Each species is initially set to evolve in the complexifying direction, and if stagnation continues for x consecutive generations occurs (number x is set beforehand) it changes the structural evolution direction to pruning. If

the species still fails to increase its fitness and so stagnates for $2x$ generations, it gets removed from the population. Otherwise, it continues to evolve by pruning until it accrues another x generations of stagnations, at which point it switches to complexification again. The graph below compares performances of StagPrune group, control original NEAT group and Threshold (5,5) group (figure 2).

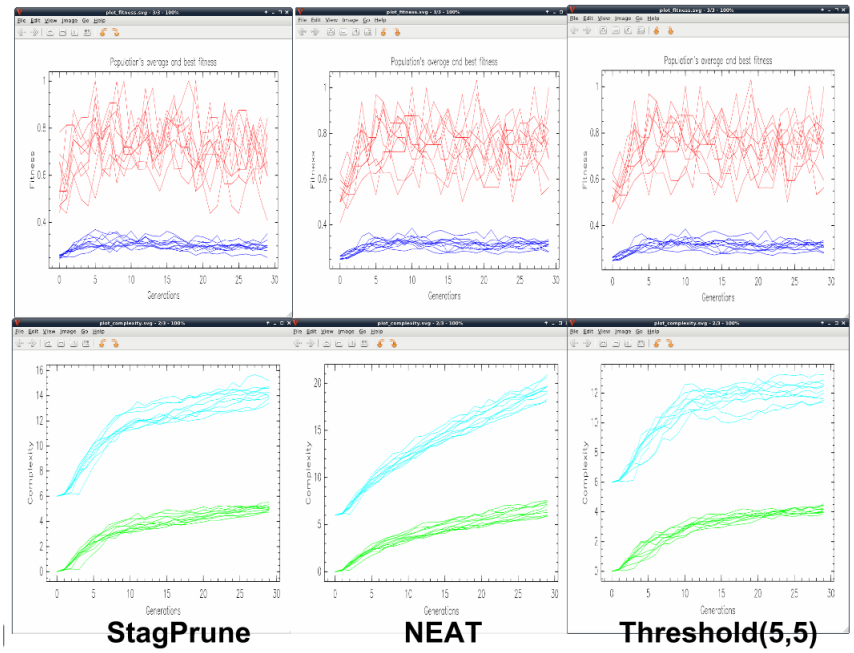

Figure 2: Performance of all three groups on the light-seeking task.

We started each evolution with 100 individuals and ran evolution for 100 generations. Red and blue lines indicate maximum and average fitness over the generations, while light blue and green lines show average numbers of hidden connections and hidden nodes. In terms of performance there werent significant differences among three groups, while average complexities of best-performing individuals from each group differ by a great amount. Original NEAT continuously increases complexities fast so that at the end of 100th generation it had 13 hidden nodes with 45 added connections. In contrast, StagPrune group had on average 7 hidden nodes and 24 added connections, while Threshold group had around 7 hidden nodes and 19 added connections.

5.3 Competititve coevolution

When Stanley first introduced NEAT (2004), he demonstrated it on a task in which two simulated robots were placed in an environment with a bunch of lights. The robots expended energy on every time step, and gained it when they ate lights. Whichever robot stayed alive

for longer won. In many ways, this is the ideal task to apply NEAT to; it gradually becomes more complex, just as the robots neural nets become more complex. It is also a hard problem; there is no single solution, and the optimal strategy is constantly changing, which makes a more complex algorithm like NEAT actually necessary. For the same reasons, coevolution is an excellent task to use to demonstrate the effects of improvements to NEAT, such as neural pruning.

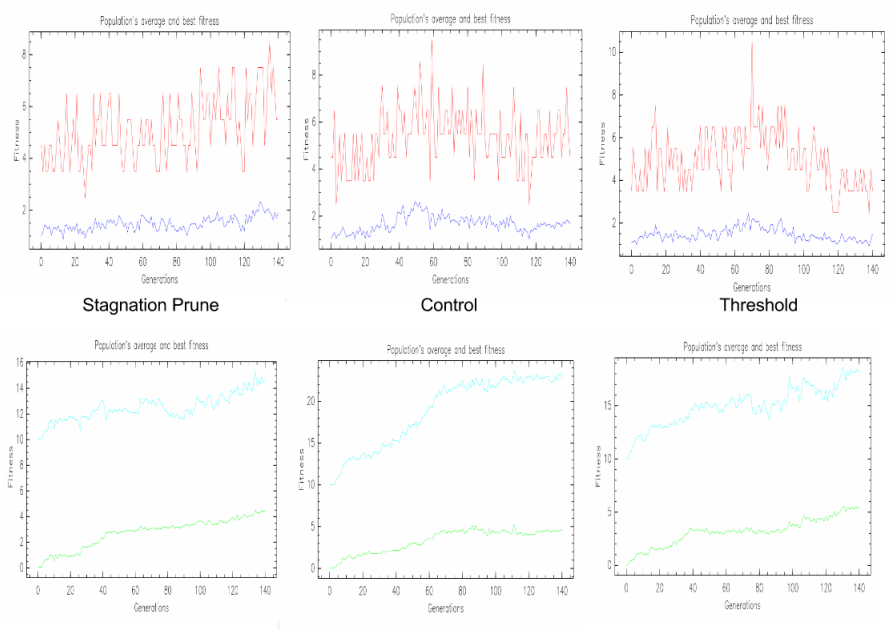

Figure 3: Performance of all three groups on the coevolution task.

Because our time and resources were more limited than Stanleys, our coevolution set-up wasnt quite the same as his. While his fitness function was based on the performance of a given agent in a series of tournaments against all of the best agents produced so far, our robots were split into two species and competed only against the best member of the opposite species. Fitness was based on the number of lights that a robot was able to eat over the course of two trials. Since both robots were competing for the same lights, this is still a form of competitive coevolution. Initially, light placement was random. Although this yielded some interesting results (see fig 3), it made fitness fluctuate semi-randomly, which interfered with the efficacy of the genetic algorithm. To make the fitness a more informative metric, light placements were standardized, as shown below (fig 4). Although this has the downside of enabling robots to memorize light locations without actually learning to seek light, understanding that light should be sought is still advantageous to outwitting other robots, so the overall task is effectively the same.

Although it is hard to interpret qualitative results, robots in the two groups that had pruning applied to them appeared to exhibit behavior that was categorically more complex.

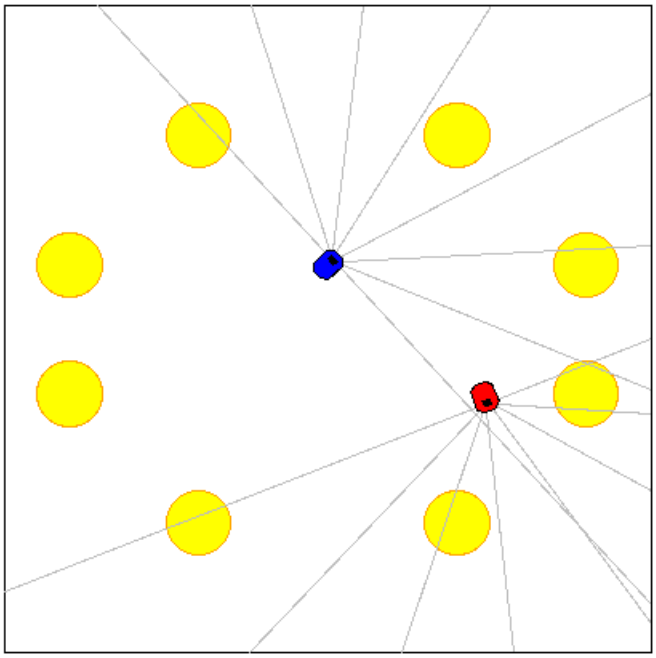

Figure 4: Fixed positions of lights

Sometimes robots would block other robots, causing them to stall and lose. Other times, both robots would try to get to all the lights as quickly as possible. At no point was there ever behavior quite as complex as that documented in the Stanley article, which is likely due to our slightly lower-pressure environment. The robots in the control group, meanwhile, would be lucky to get all of the lights between them, and had a strong propensity for spinning in circles and doing nothing. Barring some bug in our code, this would appear to suggest that pruning can lead to substantially better solutions to highly complex problems. This is particularly striking in combination with the vast differences between the topologies (see fig 5).

6 Conclusion

We have conclusively proven that adding a pruning algorithm to NEAT produces solutions which are far less complex than those produced by standard NEAT, but just as fit, or slightly more so. Even in an excessively simple problem like XOR, standard NEAT will generate very complex neural net topologies. This seems to be counter to Stanleys statement that NEAT will only increase complexity of a solution when necessary, (2004). Thus, our hypothesis that giving NEAT the ability to prune would increase the efficiency of the solutions produced was correct.

While it is hard to quantify behavioral complexity, agents evolved using either pruning

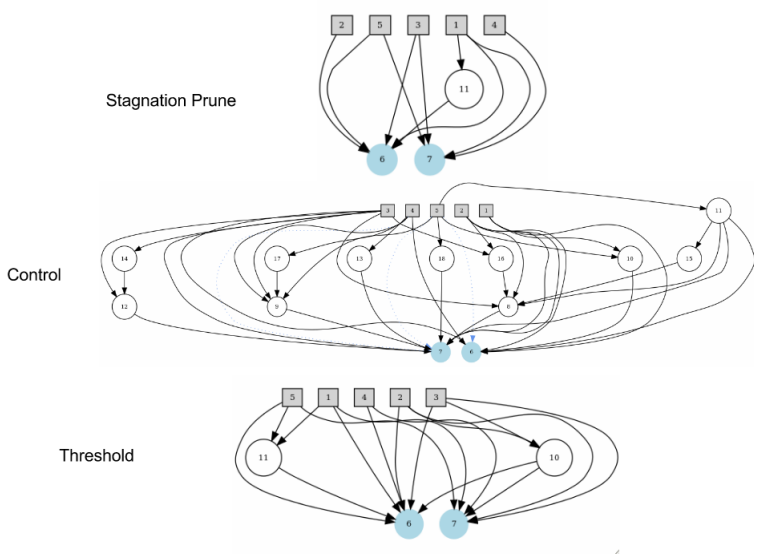

Figure 5: Best neural net topolgies for each group in the coevolution task.

method appear to categorically exhibit more strategically interesting behavior than those evolved using control. Even though this difference is only somewhat visible in the recorded fitness over time for the two groups, it is, in many ways, the most important result, as deals with the core question of the level to which a robot can adapt and the speed with which it can do so. This suggests that the second part of our hypothesis, that pruning might actually lead to better results, may also be correct.

Of the two pruning algorithms we tried, there didnt seem to be a substantial difference between them. This suggests that either the most important thing is just giving NEAT the ability to prune, so that it has a way out of random and unnecessary complexification that doesnt turn out to be useful, or that neither of these algorithms were good enough for one to be better than the other.

Further work on neural pruning in adaptive robotics should explore some more intelligent algorithms for determining when to prune. Preferentially deleting connections with low weights would potentially be an interesting concept to explore, as connection weight is sort of analogous to the amount that a connection in a biological brain is used, the primary factor for determining which connections are lost in biological neural pruning. It would also be potentially useful to know more about the relative benefits of removing nodes vs. connections. Ultimately, pruning is not necessary to solve simple problems, like XOR, although it doesnt hurt. Thus, ideally, some measure of dimensionality of a problem should be developed, so that it can be known beforehand what algorithm should be used.

7 References

P.R. Huttenlocher and C. De Courten. (1987) The development of synapses in striate cortex of man. J. Neuroscience, 6(1), 19.

James, D., and Tucker, P. (2004). A comparative analysis of simplification and complexification in the evolution of neural network topologies. Proc. of Genetic and Evolutionary Computation Conference.

Jorgensen, T. D., Haynes, B. P., and Norlund, C. (2008). Pruning artificial neural networks using neural complexity measures. International journal of neural systems, 18(5), 389.

Low, LK. and Cheng, HJ. (2006). "Axon pruning: an essential step underlying the developmental plasticity of neuronal connections". Philos Trans R Soc Lond B Biol Sci 361, 15311544.

Stanley, K. O., and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. J. Artif. Intell. Res. (JAIR), 21, 63100.