Forecasting & Futurism

December 2013 – Issue 8

A NEAT Approach to Neural Network Structure

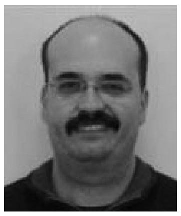
By Jeff Heaton

Neural networks are a mainstay of artificial intelligence. These machine-learning algorithms can be used for regression and classification. Typical feed forward neural networks require you to specify an exact structure of layers and neurons. This article will introduce you to three recent innovations in neural network design that alleviate you of some of the structural decisions typically required of the neural network practitioner. These three related neural networks are named NEAT, HyperNEAT and HyperNEAT-RS. Kenneth Stanley of Houston University is the primary researcher behind NEAT and many of its adaptations.

NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

NEAT is a neural network structure developed by Ken Stanley in 2002 while at the University of Texas at Austin. (Stanley 2002) NEAT uses a genetic algorithm to optimize both the structure and weights of a neural network. The input and output of a NEAT neural network are identical to a typical feed-forward neural network. For a review of feed-forward neural networks, refer to "Predictive Modeling" in the July 2013 issue of *Forecasting and Futurism Newsletter* (Xu 2013).

A NEAT network starts out with only a bias neuron, input neurons and output neurons. There are no initial connections between any of the neurons! Of course, such a completely unconnected network is useless. NEAT makes no assumptions. What if one of the input variables is statistically independent of the output? NEAT will often discover this by never evolving optimal genomes to connect that statistically independent input neuron to any other part of the network.


Jeff Heaton

Another very important difference between a NEAT network and an ordinary feed-forward neural network is that NEAT networks do not have clearly defined layers. NEAT networks do have a clearly defined input and output layer. However, the hidden neurons do not organize themselves into clearly delineated layers. A similarity between NEAT and feed-forward networks is that NEAT uses a sigmoid activation function, similar to feed-forward networks.

NEAT networks make extensive use of genetic algorithms. For a review of genetic algorithms see "Genetic Algorithms—Useful, Fun and Easy!" in the December 2012 issue of *Forecasting and Futurism Newsletter* (Snell 2012). NEAT uses a typical genetic algorithm that includes both crossover and mutation. Both mutation and crossover are operations that involve one or more parents to produce children. The crossover operation uses sexual reproduction to produce a child from two parents. The mutation operation uses asexual reproduction to produce a child from one parent.

NEAT Mutation

NEAT mutation consists of several mutation operations that can be performed on the parent genome. These operations are discussed here.

- **Add a neuron:** A neuron is added by first selecting a random link. This link is replaced by a new neuron and two links. The link is effectively split by the new neuron. The weights of each of the two new links are selected so as to provide nearly the same effective output as the link being replaced.

- **Add a link:** Two random neurons are chosen: a source and destination. The new link will be between these two neurons. Bias neurons can never be a destination. Output neurons cannot be a source. There will never be more than two links in the same direction between the same two neurons.

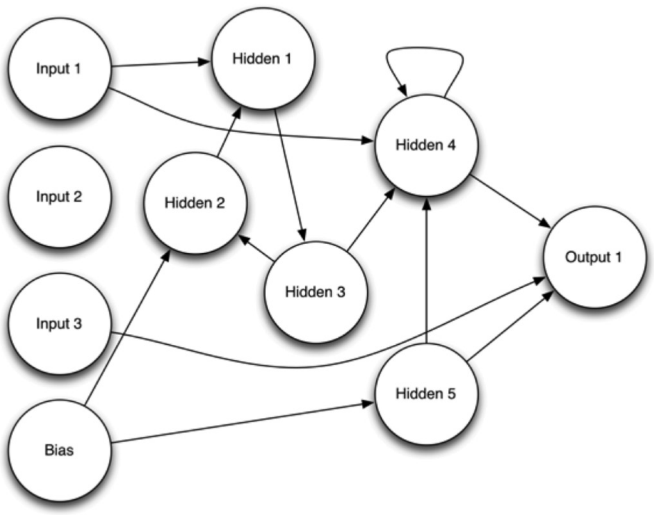
- **Remove a link:** Links can be randomly selected for removal. Hidden neurons can be removed if there are no remaining links interacting with them. A hidden neuron is any neuron that is not input, output or the single bias neuron.

- **Perturb a weight:** A random link is chosen. Its weight is then multiplied by a number from a normal random

Jeff Heaton is EHR informatics scientist at RGA Reinsurance Company in Chesterfield, Mo. He can be reached at jheaton@rgare.com.

distribution with a gamma of one or lower. Smaller random numbers will usually cause a quicker convergence. A gamma value of one or lower will specify that a single standard deviation will sample a random number of one or lower.

The mutations are weighted so that the weight perturbation occurs the most frequently. This allows fit genomes to vary their weights and further adapt through their children. The structural mutations happen with much less frequency. The exact frequency of each operation can be adjusted by most NEAT implementations. The following diagram shows a typical NEAT genome.



You can see from the above that input 2 was disregarded. You can also see that the layers are not clearly defined. There are recurrent connections, and even connections that skip directly from input to output.

NEAT Crossover

NEAT crossover is more complex than many genetic algorithms. Most genetic algorithms assume that the number of genes is consistent across all genomes in the population. This is not the case with NEAT. The NEAT genome is an encoding of the neurons and connections that make up an individual genome. Child genomes that result from both mutation and crossover may have a different number of genes than their parents. This requires some ingenuity when implementing the NEAT crossover operation.

NEAT keeps a database of all the changes made to a genome through mutation. These changes are called innovations. Innovations are created to implement mutations. However, the relationship between innovations and mutations is not one to one. It can take several innovations to achieve one mutation. There are only two types of innovation: creating a neuron and a link between two neurons. One mutation might result from multiple innovations. A mutation might also have no innovations. Only mutations that add to the structure of the network will generate innovations. The following list summarizes the innovations potentially created by the previously mentioned mutation types.

- **Add a neuron:** One new neuron innovation and two new link innovations.
- **Add a link:** One new link innovation.
- **Remove a link:** No innovations.
- **Perturb a weight:** No innovations.

It is important to note that NEAT will not recreate innovation records if such an innovation has already been attempted. Additionally, innovations do not contain any weight information; innovations only contain structural information.

Innovations are numbered. This allows NEAT crossover to determine what pre-requisite innovations are needed for a later innovation. Crossover for two genomes occurs between innovations. This allows NEAT to ensure that all prerequisite innovations are also present. A naïve crossover, such as is used by many genetic algorithms, would potentially combine links with nonexistent neurons.

NEAT Speciation

Crossover is a tricky proposition. In the real world crossover only occurs between members of the same species. This is actually a bit of a circular definition. Real-world species are defined as members of a population that can produce

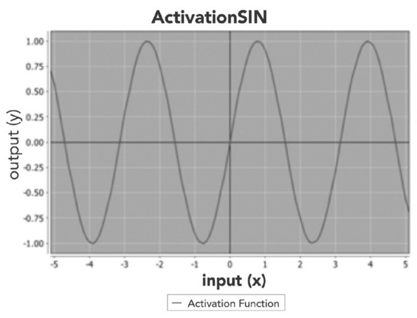
CONTINUED ON **PAGE 24**

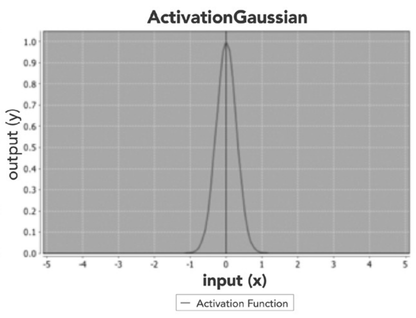
A NEAT APPROACH … | FROM PAGE 23

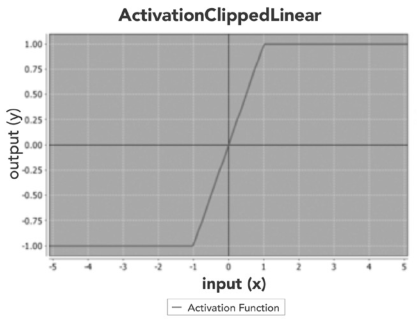
viable offspring. Attempting crossover between a horse and humming bird genome would be catastrophically unsuccessful. Yet a naïve genetic algorithm would certainly try!

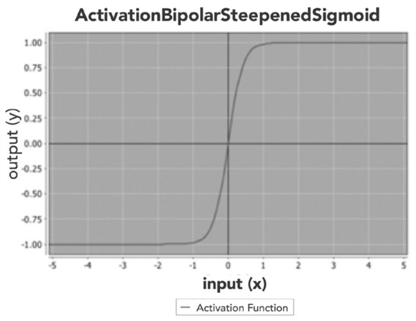
NEAT uses a type of k-means clustering to group the population into a predefined number of clusters. The relative fitness of each species is then determined. Each species is then given a percentage of the next generation's population count. The members of each species then compete in virtual tournaments to determine which members of the species will be involved in crossover and mutation for the next generation.

A tournament is an effective way to select parents from a species. A certain number of trials are performed. Typically we use five trials. For each trial two random genomes are selected from the species. The more fit of each genome advances to the next trial. This is very efficient for threading, and is also biologically plausible. You don't have to beat the best genome in your species, just the best genome you encounter in the trials! A tournament is run for each parent needed. One parent is needed for mutation and two for crossover.









In addition to the trials, several other factors determine the species members chosen for mutation and crossover.
One or more elite genomes are always carried directly to the next species. The number of elite genomes is configurable.
Younger genomes are given a bonus so they have a chance to try new innovations.

Interspecies crossover will occur with a very low probability.

All of these factors together make NEAT a very effective neural network type. NEAT alleviates you of the need to define a neural structure. Additionally, the non-level, recurrent nature of a NEAT network gives it some additional potential over regular feed-forward networks.

HYPERNEAT

Kenneth Stanley developed HyperNEAT in 2009. (Stanley 2009) HyperNEAT recognizes one very biologically plausible fact. In nature genotypes and phenotypes are not identical. What is the difference between a genotype and phenotype? The genotype is the DNA blueprint for an organism. The phenotype is what actually results from that plan.

HyperNEAT Genomes

A genome is the instructions for producing a much more complex phenotype. This is not the case with regular NEAT. The NEAT genome describes exactly, link for link, neuron for neuron, how to produce the phenotype. This is not the case with HyperNEAT. HyperNEAT creates a population of somewhat special NEAT neurons. These genomes are special in two ways. First, whereas regular NEAT always uses a sigmoid activation function, HyperNEAT can use any of the following activation functions:

- Clipped linear
- Bipolar steepened sigmoid
- Gaussian
- Sine

You can see these activation functions here.

The second difference is that these NEAT genomes are not the final product. They are not the phenotype. However, these NEAT genomes do know how to create final products. The end-result phenotype is a regular NEAT network with a sigmoid activation function. The above four activation functions are only used for the genomes. The ultimate phenotype always has a sigmoid activation function.

HyperNEAT Phenotype

You might be wondering how a neural network can be used to create another neural network. This requires one additional structure, called a substrate. The substrate defines the final structure of the phenotype. The genome neural network is then queried to determine the weight for each connection specified in the genome. There are many different substrate structures. Most HyperNEAT implementations allow you to construct substrate structures of your own. Substrate structures are very similar to traditional neural network structures in that the structure is fixed. The genome neural network's structure is dynamic, just as in NEAT. However, the phenotype structure is fixed. The genome neural network defines the weights.

The substrate is a 3D model contained in a cube. The bias and input neurons are typically embedded on one face of the cube. The output neurons are all embedded on the opposite face. Hidden neurons, if there are any, are placed in the 3D space between these two faces. Links are defined between these 3D neurons.

The substrate does not need to be a 3D cube. The substrate is actually a hypercube. In geometry, a hypercube is an n-dimensional analogue of a square (n = 2) and a cube (n = 3). Very often substrates consist of three dimensions. However it is also possible to use substrates of lower or higher dimensionality.

Why Use HyperNEAT?

You might be wondering why we would want to introduce the additional complexity of creating a genome neural network just to produce the resulting phenotype neural network. The reason is training time. When dealing with problems with a very large number of input and output neurons, training times can be very large. HyperNEAT networks are scalable.

Regular neural networks require a total retrain if you add or remove neurons. HyperNEAT allows you to change the number of neurons. One genome neural network can generate any number of phenotype neural networks of higher scale. This allows you to train at low scale and actually use the neural network at a higher scale. This allows the lower-scale training to progress much faster than the higher-scale neural network we plan to actually use.

HyperNEAT makes it possible to deal with neural networks that may ultimately have tens of thousands of input neurons. Such large neural networks could take an enormous amount of time to train. This is compounded by the fact that a population of such large neural networks would not fit into the memory of most computers.
For a real-world , consider an that generates trading signals for real-time financial data. Such a program may be ultimately used on minute or second HLOC bars. However, training can occur on larger time durations. This will speed training, but allow the neural network to take advantage of higher resolution data when back testing or actually in use.

ES HyperNEAT

NEAT's primary feature is that we no longer have to think about the structure of a neural network. HyperNEAT's primary feature is that we can quickly train with lower-scale test data and use the resulting neural network on higher-scale data. However, HyperNEAT takes away the primary advantage that NEAT gave us. With HyperNEAT we now have a substrate to architect. We are right back to designing neural network structure.

ES HyperNEAT was developed in 2010 to solve this issue. (Risi 2010) ES stands for "evolvable substrate." ES HyperNEAT is an extension of HyperNEAT that allows the substrate to evolve as well. This allows every aspect of the neural network to evolve. The end result is still a genome

CONTINUED ON **PAGE 26**

A NEAT APPROACH … | FROM PAGE 25

neural network that can create a regular NEAT network at any scale.

CONCLUSIONS

The three different types of NEAT network presented in this article represent some very recent research into neural networks. ES HyperNEAT is at the pinnacle of current NEAT research. ES HyperNEAT provides a practical way to model problems with a very large number of inputs. Kenneth Stanley is the primary researcher behind this type of neural network. To learn more about all three of these neural network types, you can visit his university home page at the following URL.

http://www.cs.ucf.edu/~kstanley/

CITATIONS

- Stanley, K., and R. Miikkulainen 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10(2): 99–127.

- Xu, R. 2013. Predictive Modeling. *Forecasting and Futurism Newsletter*. July, pp. 12–16.

- Snell, D. 2012. Genetic Algorithms—Useful, Fun and Easy! *Forecasting and Futurism Newsletter*. December, pp. 7–15.

- Stanley, K., D. D'Ambrosio and J. Gauci. 2009. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life* 15(2): 185–212.

- Risi, S., J. Lehman and K. Stanley. 2010. Evolving the Placement and Density of Neurons in the HyperNEAT Substrate. *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO-2010). New York, N.Y., pp. 563–570. ▼