NEAT and HyperNEAT a summary of neuroevolution algorithms

Chinmay Yalameli[1]
Research Scholar, Intel Student Ambassador For AI , Banglore , India

B. E (Information Science *)*

Chinmaysy@gmail.com

Abstract

Neuroevolution is a type of artificial intelligence that uses evolutionary algorithms to generate artificial neural networks (ANN), parameters, topology, and rules. We can find many different variants, but those who gain prominence are the NeuroEvolution of Augmenting Topologies (NEAT) and its extension, Hypercube-based NEAT HyperNEAT. In my paper, I am trying to discuss various outcomes, results, and drawbacks of these algorithms. I am also going to present results comparing HyperNEAT and NEAT analyzing and comparing different test results.

Keywords : Neuroevolution ,NEAT, Hyperneat, ANN

1. Introduction

1.1 Neuroevolution

In complex reinforcement learning and Q learning task, Neuroevolution (NE), the artificial evolution of neural networks using genetic algorithms, has shown substantial commitments. The common question asked is how to gain an advantage from evolving neural network topologies along with weights. In natural selection, an essential mechanism of biological evolution, individuals who are more fit for the conditions in which they live are “chosen” to reproduce, by being more likely than some fit individuals to remain to maturity and find a mate. These algorithms use a search-based method where individual entities are offered as a solution.

The first set of entity solutions are generated randomly, and a fitness function is used to access the fitness of each candidate and determine the next subset. These entities mate with each other by exchanging some values, in turn, generate objects with high competence. This process gets usually stopped after a stop condition is reached. As in biology, the genotype is a genetic representation of a being; the phenotype is the actualized physical representation of the creature. Evolutionary algorithms always heavily mirror biology, neuroevolution being no different in this respect. The question of encoding comes from the question of how do we wish to represent individuals genetically in our algorithm. There is no restriction on encoding genotype unless the encodings are searchable. Encoding in a string of bits is a common approach and its named phenotype. Neuroevolution algorithms are used in many fields namely economics, cryptanalysis, video games, system design, and logistics

1.2 Artificial Neural Networks

Artificial neural networks (ANN) are computational models motivated by, but not similar biological neural networks. They work as the interconnection between groups of nodes or neurons identical to the actual brain. They can "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules. They can be interpreted as graphs and can search for patterns and approximate continuous functions.

They can be understood in terms of layers. They primarily contain three layers, a layer of input, an output layer, and a layer of hidden neurons between input and output layers. The mechanism works as described; neural networks start by taking input values from the input layer. Each edge in the graph,or connection, has a weight assigned to it, neurons compute costs and send them to the output layer. Some advanced learning techniques like Reinforcement learning take feedback from the output layer. Neat and HyperNeat evolve their own topologies.

2.NEAT (NeuroEvolution of Augmenting Topologies)

NeuroEvolution of Augmenting Topologies known as NEAT is one of the neuroevolution algorithms which was developed at the University of Texas at Austin (Stanley & Mikkulainen) in 2002. Usually, the network topology is a single hidden layer of neurons, with each hidden neuron connected to every network input and every network output. Evolution searches the space of connection weights of this fully connected topology by allowing high-performing networks to reproduce[1]. The structure of neat contains a set of neuron genes and a set of connecting genes. Neuron genes are used to assign each neuron an identification number. They define whether output,input, or hidden nodes, while connect genes are used to check the connectivity, assigned weights, flags, and evidence of recurrence. The check for redundancy is done whenever a new link is added. Based on results, it is decided to add a unique identification number or the previous one.

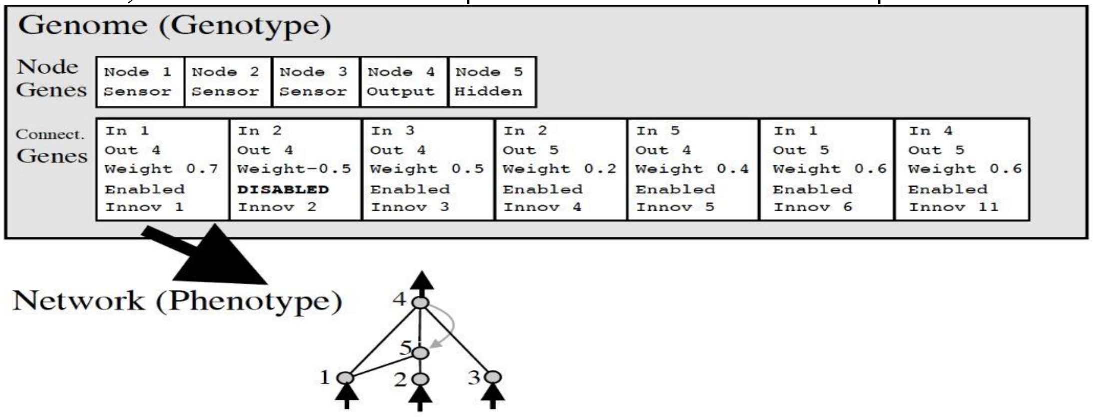
Genome (Genotype)
Node Genes
Node 1 Sensor
Node 2 Sensor
Node 3 Sensor
Node 4 Output
Node 5 Hidden
Connect. Genes
In 1 Out 4 Weight 0.7 Enabled Innov 1
In 2 Out 4 Weight-0.5 DISABLED Innov 2
In 3 Out 4 Weight 0.5 Enabled Innov 3
In 2 Out 5 Weight 0.2 Enabled Innov 4
In 5 Out 4 Weight 0.4 Enabled Innov 5
In 1 Out 5 Weight 0.6 Enabled Innov 6
In 4 Out 5 Weight 0.6 Enabled Innov 11
Network (Phenotype)
4
5
1
2
3

Figure 1. Genotype and Phenotype

Neat follows a rule that searches should begin from the smallest space and gradually expand to the search space. In the first generation, every input node is connected to every output node. It is usual in neuroevolution that the new nodes added will do poorly and can perish before they showcase there full potential to prevent this NEAT uses biological speciation. There are several problems that is faced by NEAT. Few are illustrated below.

2.1 Solutions to problems faced by NEAT

Although NEAT defines one of the prominent approach , it is vulnerable to various situation. The thing that makes it special is the way the solutions are provide.

Encoding- As we have already discussed, genotypes and phenotypes as genric, actualized physical representations. The question that arises is about there representation. A direct encoding will directly specify everything about an entity. If it represents a neural network, this means that each gene will instantly be linked to some node, connection, or property of the system. It can be a binary encoding of 1s and 0s, a graph encoding (connecting various nodes by weighted connections), or something even more difficult. Indirect encoding is governed by rules,parameters for processing and are more compact in nature.An direct encoding methodology is choosen by NEAT algorithm chooses a because of this, its representation is a more complex than a simple graph or binary encoding

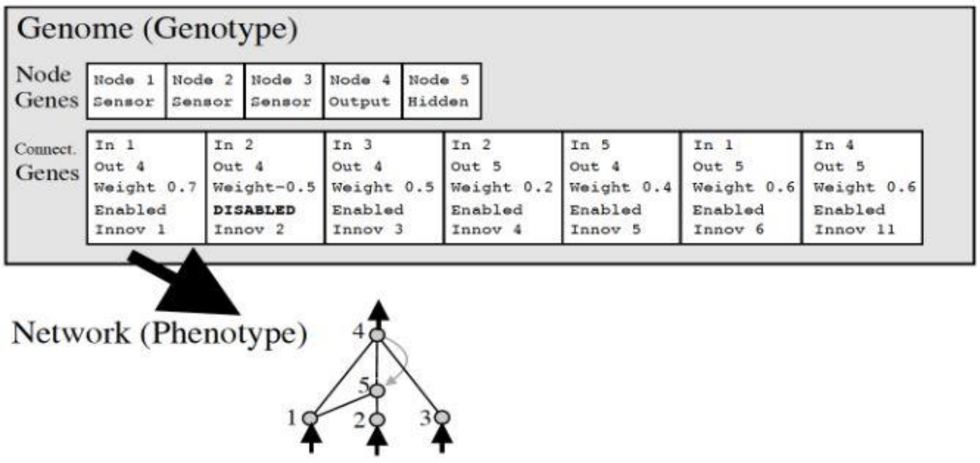
Genome (Genotype)
Node Genes
Node 1 Sensor
Node 2 Sensor
Node 3 Sensor
Node 4 Output
Node 5 Hidden
Connect. Genes
In 1 Out 4 Weight 0.7 Enabled Innov 1
In 2 Out 4 Weight-0.5 DISABLED Innov 2
In 3 Out 4 Weight 0.5 Enabled Innov 3
In 2 Out 5 Weight 0.2 Enabled Innov 4
In 5 Out 4 Weight 0.4 Enabled Innov 5
In 1 Out 5 Weight 0.6 Enabled Innov 6
In 4 Out 5 Weight 0.6 Enabled Innov 11
Network (Phenotype)
4
5
1
2
3

Figure 2. Encoding

Mutation- In NEAT, mutation occurs in different ways.It can mutate new connections or it will add new structure to the network.If a new connection is inserted between a start and end node, a random weight is assigned to it. The added new node is placed between two nodes that are already connected. The previous connection is impaired (though still exists in the genome). The previous start node is connected to the new node with the weight that of the old connection and the new node is connected to its previous end with a weight of 1. It was found to help alleviate issues with new structural additions.

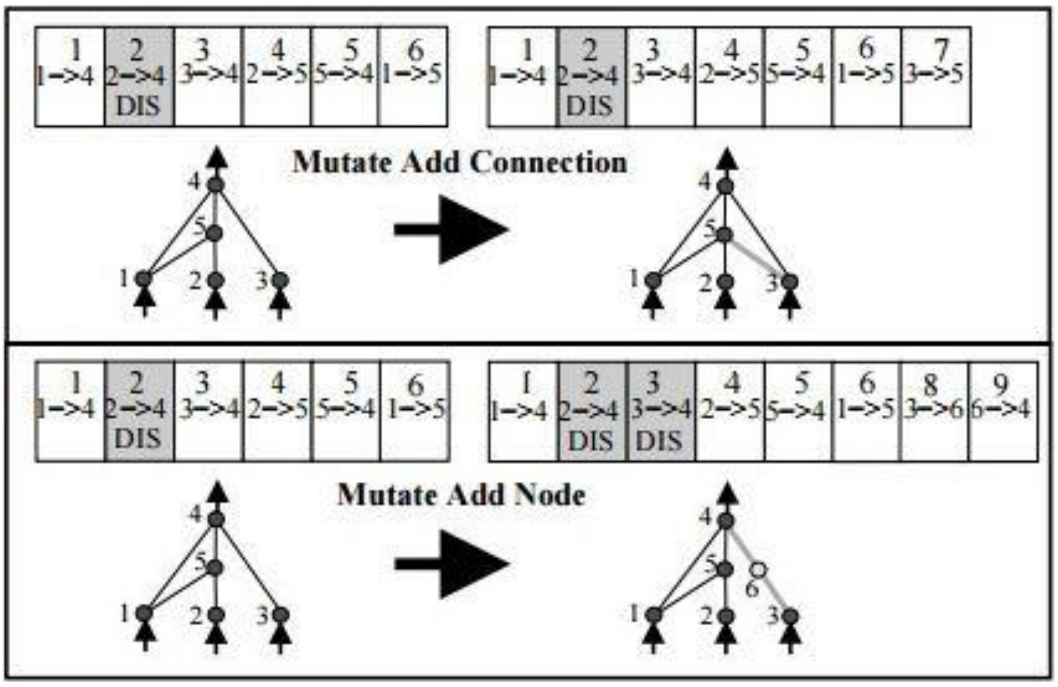

Figure 3. Mutation

Speciation- The fact that adding a new connection or node before any optimization of weights have occurred often leads to a lower performing individual puts its structure into disadvantage. The solution is given by speciation. Speciation clearly splits up the population into various species based on the similarity of connections and topology. If the competing convention problem still persists, this would be very difficult to measure! However, as NEAT uses historical markings in its principal of encoding, it becomes much easier to measure and act. A mentod for deciding how to speciate is mentioned in the paper, but the important point to note is that individuals in any population only have to compete with other individuals within that species.

3. HyperNEAT

In biological, genetic encoding, the mapping between phenotype and genotype is indirect. The phenotype typically contains orders of magnitude more structural components than the genotype contains genes. For example, the human brain of 30,000 genes can encode 30,000 genes [2].cDevelopmental encoding is an encoding scheme borrowed from the ideas of biology. This is the idea states that all genes should be able to be reused at any point in time during their developmental process and at any location within the entity. Compositional Pattern Producing Networks (CPPNs) is an abstraction of this concept that has been shown to be able to reciprocate and create Q patterns for repeating structures in Cartesian space.

In a CPPN, the inputs are Cartesian coordinates in n-dimensional space, where the number of inputs is defined by n, and are passed into a function f. A difference between typical artificial neural networks (ANNs) and CPPNs is that ANNs usually only contain sigmoid functions, while CPPNs can contain various types of tasks (Stanley et al., 2009). While CPPNs can be developed through genetic algorithms just as ANNs can, in HyperNEAT, CPPNs are not being developed as an end product, but they are being used to encode ANNs. Since a CPPN can produce a spatial pattern, and an ANN is a connectivity pattern, the algorithm uses a mapping between spatial and connectivity patterns. The input into the CPPN is couple points rather than one, and the couple points represent two nodes in the ANN, with the output of the CPPN's function being the weight of that link between the two nodes. An algorithm is designed for the utilization of the HyperNeat concept. The algorithm determines this ANN's fitness just as NEAT does with its ANNs, and evolves the second generation of CPPNs according to the NEAT strategy. It repeats as any neuroevolution algorithm would until a solution is achieved.

4. Results of Various tests on (Neat Vs HyperNEAT)

A group of researchers at MIT examined the performance of NEAT and HyperNEAT on a class of problems which are referred to in the literature as “fractured” problems (Kohl & Mikkulainen, 2009), by testing them on the example benchmark problem of keepaway soccer [3]. They used package called as SharpNEAT.This package contains C# implementations of NEAT and HyperNEAT using the Microsoft .NET framework and Keepaway soccer (Whiteson et al, 2005; Stone et al, 2006) which is a machine learning benchmark problem that requires high-level strategic decision-making and has a fractured decision space as discussed in section 1.6 (Kohl & Mikkulainen, 2009).

The results were as follows,
A group of researchers at MIT examined the performance of NEAT and HyperNEAT on a class of problems which are referred to in the literature as “fractured” problems (Kohl & Mikkulainen, 2009), by testing them on the example benchmark problem of keepaway soccer [3]. They used package called as SharpNEAT. This package contains C# implementations of NEAT and HyperNEAT using theframework of Microsoft .NET and Keepaway soccer (Whiteson et al, 2005; Stone et al, 2006) which is a machine learning benchmark problem that requires high-level strategic decision-making and has a fractured decision space as discussed in section 1.6 (Kohl & Mikkulainen, 2009).

The results were as follows,
1. Using One Configuration NEAT could achieve a score of 41 and HyperNEAT achieved a score of 48. Which stated that, HyperNEAT outperforms NEAT by 17%, which lends support to our hypothesis that HyperNEAT's ability to exploit geometric information improves its performance on this problem [1].

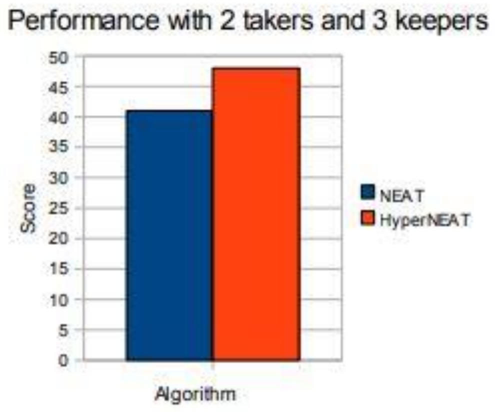

Figure 4. Neat vs Hyper Neat [1]

2. In both cases of using NEat and HyperNeat, the worst candidates achieved scores of about 35, and only one to two candidates scored above 40. This suggests that the algorithms are performing similarly for the most part, but HyperNEAT's top Illustration 3: Comparison of NEAT and HyperNEAT with the configuration of 2 takers and 3 keepers. Performance with 2 takers and 3 keepers. Comparing NEAT and HyperNEAT Algorithms Score, candidates from HyperNeat are able to break away from the pack to a greater degree than NEAT's .

3.They ran NEAT and HyperNEAT using a set of 16 different game configurations – all possible combinations of 2-5 takers and 3-6 keepers. The score achieved by each algorithm was the average of the scores for the 16 configurations. In this experiment, NEAT achieved a score of 58 and HyperNEAT achieved a score of 52 – NEAT performed 11.5% better than HyperNEAT.

In conclusion it was determined that based on experiments conducte HyperNEAT improves upon NEAT for a relatively easy and simple fractured problem, but that the benefit gradually disappeared as the problem was complicated. Thus it is advised to use each algorithms appropriately.

References

[1] Jessica Lowell, Kir Birger, and Sergey Grabkovsky 'Comparison of NEAT and HyperNEAT on a Strategic Decision-Making Problem'

[2] Kenneth O. Stanley ,David D'Ambrosio , Jason Gauci A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks

[3] Neat : An awesome approach to Neuroevolution by Hunter Heidenreich

[4] HyperNeat : Powerful,indirect approach of neuroevolution by by Hunter Heidenreich

[5] McBride, J., Lowell, J., Snorrason, M., Eaton, R., Irvine, J. Automated rapid training of ATR algorithms. Proceedings of SPIE. Vol. 7335.

[6] Richards, N., Moriarty, D.E, Mikkulainen, R. Evolving neural networks to play Go. Applied Intelligence. Vol. 8:1. pp 85-96. Stanley, K.O. 2004.

[7]Efficient evolution of neural networks through complexification. Doctoral dissertation. University of Texas at Austin, Department of Computer Science. Stanley, K.O., 2007.

[8]Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines: Special Issue on Developmental Systems. Vol. 8:2. pp 131-162. Stanley, K.O., D'Ambrosio, D., Gauci, J., 2009.

[9] A Hypercube-based indirect encoding for evolving large-scale neural networks. Artificial Life. Vol. 15:2. pp 185-212.