Neuroevolution

From Wikipedia, the free encyclopedia

Neuroevolution, or **neuro-evolution**, is a form of machine learning that uses evolutionary algorithms to train artificial neural networks. It is most commonly applied in artificial life, computer games, and evolutionary robotics. A main benefit is that neuroevolution can be applied more widely than supervised learning algorithms, which require a syllabus of correct input-output pairs. In contrast, neuroevolution requires only a measure of a network's performance at a task. For example, the outcome of a game (i.e. whether one player won or lost) can be easily measured without providing labeled examples of desired strategies.

Contents

Features

There are many neuroevolution algorithms. One common distinction is whether algorithms evolve only the strength of the connection weights for a fixed network topology (sometimes called conventional neuroevolution), as opposed to those that evolve both the topology of the network and its weights (called TWEANNs, for Topology & Weight Evolving Artificial Neural Network algorithms).

A separate distinction can be made between methods that evolve the structure of ANNs in parallel to its parameters (those applying standard evolutionary algorithms) and those that develop them separately (through memetic algorithms).[1]

Other dimensions of variation include what type of neural model is employed, which ranges from simple weighted-sum units to more biologically accurate models; whether the neural network weights are fixed during evaluation or whether evolved learning rules can allow lifetime learning (i.e. plastic neural networks); and whether each element of the evolved network is directly encoded as a separate gene (called a direct encoding), or whether there is gene reuse through which one gene may encode many network elements (called an indirect encoding).

Direct and indirect encoding

Evolutionary algorithms operate on a population of genotypes (also referred to as genomes). In neuroevolution, a genotype is mapped to a neural network phenotype that is evaluated on some task to derive its fitness.

In *direct* encoding schemes the genotype directly maps to the phenotype. That is, every neuron and connection in the neural network is specified directly and explicitly in the genotype. In contrast, in *indirect* encoding schemes the genotype specifies indirectly how that network should be generated.[2]

Indirect encodings are often used to achieve several aims:[2][3][4][5][6]

- Allow recurring structures or features in the network to form (modularity and other regularities);
- compression of phenotype to a smaller genotype, providing a smaller search space;
- mapping the search space (genome) to the problem domain.

Taxonomy of embryogenic systems for indirect encoding

Traditionally indirect encodings that employ artificial embryogeny (also known as artificial development) have been categorised along the lines of a *grammatical approach* versus a *cell chemistry approach*.[5] The former evolves sets of rules in the form of grammatical rewrite systems. The latter attempts to mimic how physical structures emerge in biology through gene expression. However, this separation is somewhat superficial as indirect encoding systems often use aspects of both approaches.

Stanley and Miikkulainen[5] propose a taxonomy for embryogenic systems that is intended to reflect their underlying properties. The taxonomy identifies five continuous dimensions, along which any embryogenic system can be placed and, thus, compared with others:

- **Cell (Neuron) Fate:** the final characteristics and role of the cell in the mature phenotype. This dimension ranges from a single method for determining the fate of a cell to having many determination methods.
- **Targeting:** the method by which connections are directed from source cells to target cells. This ranges from specific targeting (source and target are explicitly identified) to only relative targeting (e.g. based on locations of cells relative to each other).
- **Heterochrony:** the timing and ordering of events during embryogeny. Ranges from no mechanisms for changing the timing of events to many mechanisms.
- **Canalization:** how tolerant the genome is to mutations (brittleness). Ranges from requiring precise genotypic instructions to a high tolerance of imprecision or mutation.
- **Complexification:** the ability of the system (including evolutionary algorithm and genotype to phenotype mapping) to allow complexification of the genome (and hence phenotype) over time. Ranges from allowing only fixed-size genomes to allowing highly variable length genomes.

Examples

Examples of neuroevolution methods (those with direct encodings are necessarily non-embryogenic):

Method	Encoding	Evolutionary algorithm	Aspects evolved
Cellular Encoding (CE) by F. Gruau, 1994[7]	Indirect, embryogenic (grammar tree using S-expressions)	Genetic programming	Structure and parameters (simultaneous, complexification)
GNARL by Angeline et al., 1994[8]	Direct	Evolutionary programming	Structure and parameters (simultaneous,

			complexification)
EPNet by Yao and Liu, 1997[9]	Direct	Evolutionary programming (combined with backpropagation and simulated annealing)	Structure and parameters (mixed, complexification and simplification)
NeuroEvolution of Augmenting Topologies (NEAT) by Stanley and Miikkulainen, 2002[10][11]	Direct	Genetic algorithm. Tracks genes with historical markings to allow crossover between different topologies, protects innovation via speciation.	Structure and parameters
Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) by Stanley, D'Ambrosio, Gauci, 2008[3]	Indirect, non-embryogenic (spatial patterns generated by a Compositional pattern-producing network (CPPN) within a hypercube are interpreted as connectivity patterns in a lower-dimensional space)	Genetic algorithm. The NEAT algorithm (above) is used to evolve the CPPN.	Parameters, structure fixed (functionally fully connected)
Evolutionary Acquisition of Neural Topologies (EANT/EANT2) by Kassahun and Sommer, 2005[12] / Siebel and Sommer, 2007[13]	Direct and indirect, potentially embryogenic (Common Genetic Encoding[2])	Evolutionary programming/Evolution strategies	Structure and parameters (separately, complexification)
Interactively Constrained Neuro-Evolution (ICONE) by Rempis, 2012[14]	Direct, includes constraint masks to restrict the search to specific topology / parameter manifolds.	Evolutionary algorithm. Uses constraint masks to drastically reduce the search space through exploiting domain knowledge.	Structure and parameters (separately, complexification, interactive)
Deus Ex Neural Network (DXNN) by Gene Sher, 2012[15]	Direct/Indirect, includes constraints, local tuning, and allows for evolution to integrate new sensors and actuators.	Memetic algorithm. Evolves network structure and parameters on different time-scales.	Structure and parameters (separately, complexification, interactive)
Spectrum-diverse Unified Neuroevolution Architecture (SUNA)	Direct, introduces the Unified Neural	Genetic Algorithm with a diversity preserving mechanism called Spectrum-diversity that scales well with	Structure and

by Danilo Vasconcellos Vargas, Junichi Murata[16] (Download code (https://github.com/zweifel/Physis-Shard))	Representation (representation integrating most of the neural network features from the literature).	chromosome size, is problem independent and focus more on obtaining diversity of high level behaviours/approaches. To achieve this diversity the concept of chromosome Spectrum is introduced and used together with a Novelty Map Population.	parameters (mixed, complexification and simplification)

See also

- Evolutionary computation
- Artificial neural network
- NeuroEvolution of Augmented Topologies (NEAT)
- Noogenesis
- HyperNEAT (A Generative version of NEAT)
- Evolutionary Acquisition of Neural Topologies (EANT/EANT2)
- Spectrum-diverse Unified Neuroevolution Architecture (SUNA)

References

1. Togelius, Julian and Schaul, Tom and Schmidhuber, Jurgen and Gomez, Faustino. Countering poisonous inputs with memetic neuroevolution. In Proceedings of Parallel Problem Solving from Nature (PPSN X), 2008. [1] (http://www.academia.edu/download/30945872/poison.pdf)
2. Yohannes Kassahun, Mark Edgington, Jan Hendrik Metzen, Gerald Sommer and Frank Kirchner. Common Genetic Encoding for Both Direct and Indirect Encodings of Networks. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007), London, UK, 1029-1036, 2007.
3. Gauci, Stanley. Generating Large-Scale Neural Networks Through Discovering Geometric Regularities. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007). New York, NY: ACM. [2] (http://eplex.cs.ucf.edu/papers/gauci_gecco07.pdf)
4. F. Gruau. Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm. PhD thesis, Ecole Normale Superieure de Lyon, Laboratoire de l'Informatique du Parallelisme, France, January 1994.
5. Stanley, Miikkulainen. A Taxonomy for Artificial Embryogeny. The MIT Press Journals, 2003. [3] (http://nn.cs.utexas.edu/downloads/papers/stanley.alife03.pdf)
6. Clune J, Stanley KO, Pennock RT, and Ofria C. On the performance of indirect encoding across the continuum of regularity. IEEE Transactions on Evolutionary Computation, 2011. [4] (http://dx.doi.org/10.1109/TEVC.2010.2104157)
7. Gruau, F. (1994). Neural network synthesis using cellular encoding and the genetic algorithm. Doctoral dissertation, Ecole Normale Superieure de Lyon, France.
8. Peter J Angeline, Gregory M Saunders, and Jordan B Pollack. An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks, 5:54–65, 1994. [5] (http://demo.cs.brandeis.edu/papers/ieeenn.pdf)
9. Xin Yao and Yong Liu. A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks, 8(3):694–713, May 1997. [6] (http://www.cs.bham.ac.uk/~axk/evoNN2.pdf)
10. http://nn.cs.utexas.edu/downloads/papers/stanley.ieeetec05.pdf
11. http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf
12. Yohannes Kassahun and Gerald Sommer. Efficient reinforcement learning through evolutionary acquisition of neural topologies. In Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN 2005), pages 259–266, Bruges, Belgium, April 2005. [7] (http://www.ks.informatik.uni-kiel.de/~yk/ESANN2005EANT.pdf)
13. Nils T Siebel and Gerald Sommer. Evolutionary reinforcement learning of artificial neural networks. International Journal of Hybrid Intelligent Systems 4(3): 171-183, October 2007. [8] (http://www.ks.informatik.uni-kiel.de/~vision/doc/Publications/nts/SiebelSommer-IJHIS2007.pdf)
14. Christian W. Rempis, Evolving Complex Neuro-Controllers with Interactively Constrained Neuro-Evolution. PhD thesis,

Osnabrück University, October 2012, urn:nbn:de:gbv:700-2012101710370 [9] (http://repositorium.uni-osnabrueck.de/handle/urn:nbn:de:gbv:700-2012101710370)

15. Gene I. Sher, Handbook of Neuroevolution Through Erlang. Springer Verlag, November 2012 [10] (http://www.springer.com/computer/swe/book/978-1-4614-4462-6)
16. Danilo Vasconcellos Vargas, Junichi Murata, IEEE Transactions on Neural Networks and Learning Systems [11] (http://dx.doi.org/10.1109/TNNLS.2016.2551748)

External links

- BEACON Blog: What is neuroevolution? (http://beacon-center.org/blog/2012/08/13/evolution-101-neuroevolution/)
- University of Texas neuroevolution page (http://nn.cs.utexas.edu/keyword?neuroevolution) (has downloadable papers on NEAT and applications)
- SharpNEAT (http://sharpneat.sourceforge.net/) is a mature Open Source neuroevolution project implemented in C#/.Net.
- ANNEvolve is an Open Source AI Research Project (http://ANNEvolve.sourceforge.net) (Has downloadable source code in C and Python for a variety of interesting problems. Also a tutorial & miscellaneous writings and illustrations)
- Web page on evolutionary learning with EANT/EANT2 (http://www.siebel-research.de/evolutionary_learning/) (information and articles on EANT/EANT2 with applications to robot learning)
- NERD Toolkit. (http://nerd.x-bot.org/) The Neurodynamics and Evolutionary Robotics Development Toolkit. A free, open source software collection for various experiments on neurocontrol and neuroevolution. Includes a scriptable simulator, several neuro-evolution algorithms (e.g. ICONE), cluster support, visual network design and analysis tools.
- DXNN (https://github.com/CorticalComputer) Source code for the DXNN Neuroevolutionary system.

Categories: Evolutionary algorithms