Neuroevolution

Inteligencia Artificial en los Sistemas de Control Autónomo
Máster Universitario en Ingeniería Industrial

Departamento de Automática

Objective

- Fusion ANN and Evolutionary Algorithms
- Identify application areas of Neuroevolution in Robotics

Bibliography

1. A. Tettamanzi, M. Tomassini. *Soft Computing. Integrating Evolutionary, Neural, and Fuzzy Systems*. Springer-Verlag. 2001
2. D. Floreano, P. Dürr, C. Mattiussi. *Neuroevolution: from architectures to learning*. Evolutionary Intelligence, Vol. 1, No. 1, pags. 47-62. Springer-Verlag. 2008.
3. S. Risi, J. Togelius. *Neuroevolution in Games: State of the Art and Open Challenges*. IEEE Trans. on Computational Intelligence and AI in Games. Vol 9, No. 1. 2017

Table of Contents

Introduction

Motivation

Problems with traditional learning algorithms

- Fixed topology
- Local mimina and other learning limitations

In Nature ...

- Global neural system architecture given by evolution
- Details (synapsys) given by learning

EAs good avoiding local maxima and searching complex search spaces

Introduction

Definition (I)

Neuroevolution (NE)

NE refers to the generation of artificial neural networks (their connections weights and/or topology) using evolutionary algorithms

Risi and Togelius

NE features

- Record-beating performance
- Broad applicability
- Scalability
- Diversity
- Open-ended learning

Problem: NE does not provide explicative models

Introduction

Definition (II)

When ANN meets EAs

- Evolve weights
- Evolve topology

Key elements to take into account

- Evaluation (straight or hybrid) (not accepted terms)
- Representation (direct or indirect)

ANN evaluation

Straight approach

Straight: Build ANN and assess it

- Cumulative error on test set
- Simulate robot behavior
- Observe robot behavior

Some problems

- Pretty slow
- Does not explot gradient if available

ANN evaluation

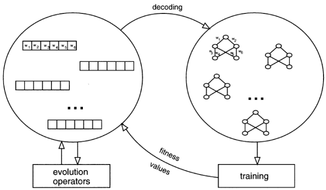
Hybrid approach

Hybrid: Join evolution and learning

- EAs have good exploration
- Training has good explotation
- Gradient-based methods sensible to initial weights

Evolve a population of ANN, then train them (backprop, ...)

- Trainning does not change genotype



A. Tettamanzi, M. Tomassini. *Soft Computing. Integrating Evolutionary, Neural, and Fuzzy Systems.* Springer-Verlag. 2001

Direct codification

Direct codification of ANNs

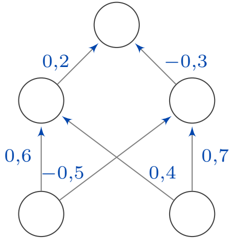
Each gene represents a weight

- Fixed topology

Can be used any EA

- Typically, GA or ES

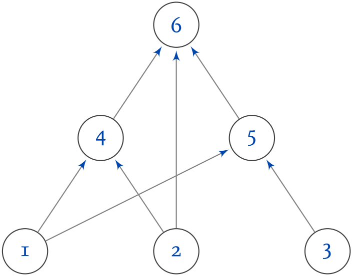
Several complex neuroevolution specific codifications



(0.2, -0.3, 0.6, -0.5, 0.4, 0.7)

Direct codification

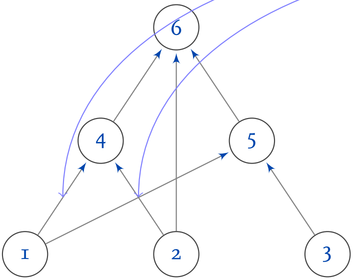
Alternative direct codification of ANNs



	1	2	3	4	5	6
1	0	0	0	1	1	0
2	0	0	0	1	0	1
3	0	0	0	0	1	0
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

Direct codification

Alternative direct codification of ANNs



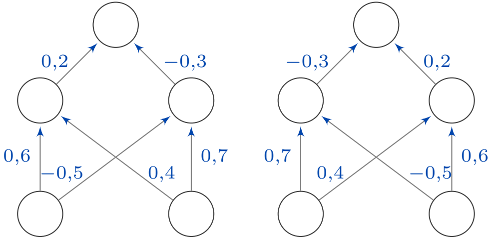
	1	2	3	4	5	6
1	0	0	0	1	1	0
2	0	0	0	1	0	1
3	0	0	0	0	1	0
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

Direct codification

Permutation problem

Permutation problem (also known as compeling convenions)

- Multiple genotypes with same phenotype



(0.2, -0.3, 0.6, -0.5, 0.4, 0.7) (-0.3, 0.2, 0.7, 0.4, -0.5, 0.6)

Direct codification

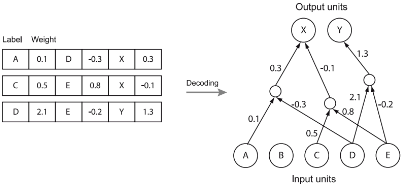
Symbolic, adaptive, neuro-evolution (SANE)

SANE evolves single neurons

- Population of neurons
- Connections and weights
- Fixed topology: One hidden layer

Evaluation

1. Build random ANNs with sampled neurons
2. Compute ANNs fitness
3. Fitness of a neuron is the average fitness of all the ANNs it has participated in



D. Floreano, P. Dürr, C. Mattiussi. *Neuroevolution: from architectures to learning*. Evolutionary Intelligence, Vol. 1, No. 1, pags. 47-62. Springer-Verlag. 2008.

Direct codification

Neuro-evolution of Augmenting Topologies (NEAT)

Quite used NE algorithm

- Weights and topologies
- Grows ANN complexity

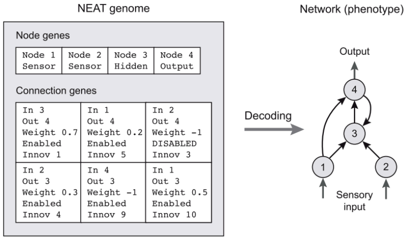
Two types of genes

- Nodes and connections

Genetic operators

- Meaningful crossover
- Gene enable/disable mutation
- Gene insert operator

(Video Torcs) (Video Mario)



D. Floreano, P. Dürr, C. Mattiussi. *Neuroevolution: from architectures to learning*. Evolutionary Intelligence, Vol. 1, No. 1, pags. 47-62. Springer-Verlag. 2008.

Indirect codification

Introduction

Problems with direct codification

- Scalability
- No reuse

Indirect encoding try to grow networks

- Evolve generation rules instead of individual weights
- Try to reuse basic building blocks
- Closer to biological systems

Indirect codification

Kitano's method (I)

Kitano used rewriting rules

- Terminals, a symbol
- Non-terminals, a rewriting rule

Grammar example

```
 <digit>  ⟶  0|1|2|3|4|5|6|7|8|9
<number>  ⟶  <digit>
<number>  ⟶  <number><digit>
```

Rather standard GA evolve rules

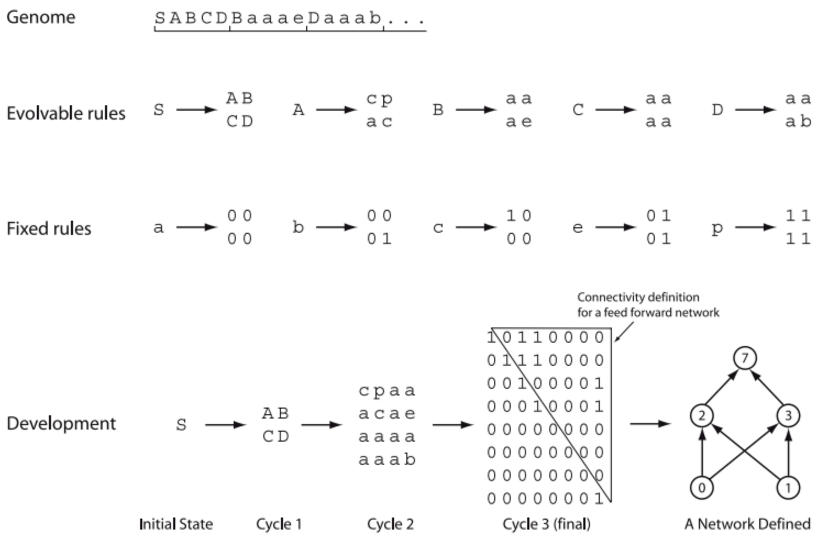
- Fitness proportionale, elitism, variable mutation rate, single crossover
- Evaluation of the network trained with backpropagation

Cromosomes composed by

- Fixed and evolvable rules

Indirect codification

Kitano's method (II)



D. Floreano, P. Dürr, C. Mattiussi. *Neuroevolution: from architectures to learning*. Evolutionary Intelligence, Vol. 1, No. 1, pags. 47-62. Springer-Verlag. 2008.

Indirect codification

Kitano's method (III)

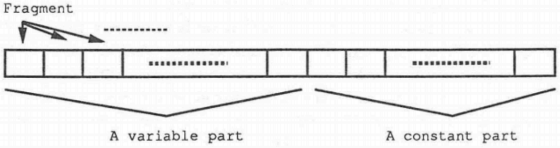


Figure 5: A variable and a constant part of a chromosome.

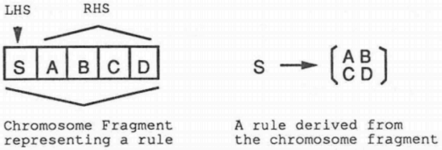


Figure 6: Grammar encoding on chromosome.

H. Kitano. *Designing Neural Networks Using Genetic Algorithms with Graph Generation System.* Complex Systems 4: 461-476. 1990.