Evolving Neural Networks

CSCI 250: Introduction to Robotics
Simon D. Levy
14 May 2013
Evolving Neural Nets: A Brief History

- Evolve weights in a fixed architecture
  - Back-prop vs. GA: Montana & Lawrence 1989
  - Enforced Sub-Populations (Gomez & Mikkulainen 1999)
- Evolve architecture: TWEANNs
  - Direct encoding (Miller, Todd, Hedge 1989)
  - Grammatical encoding (Kitano 1990)
  - Cellular encoding (Gruau & Whitley 1993)

Tuesday, May 14, 13
Evolving Weights Only: Early Attempts

Result: On simple tasks like XOR, some GA runs outperformed back-prop.
ESP: Enforced Subpopulations

Figure 1: The Enforced Sub-Populations Method (ESP). The population of neurons is segregated into sub-populations shown here as clusters of circles. The network is formed by randomly selecting one neuron from each subpopulation.

- Fixed # of nodes, but don’t know how many hidden nodes is optimal
- So if population fails at task, restart it with a different number of hidden nodes (so is # really “fixed”?)
Evolving Architecture via Direct Encoding

Connectivity Matrix:

from unit: 1 2 3 4 5
to unit: 1 0 0 0 0 0
2 0 0 0 0 0
3 1 1 0 0 0
4 1 1 0 0 0
5 0 0 1 1 0

chromosome: 0000000000110001100000110

How does this scale with network size?

Tuesday, May 14, 13
Grammatical Encoding

S → A B
   C D

A → cp
   ac
B → aa
   ae
C → aa
   aa
D → aa
   ab

a → 0 0
   0 0
   0 1
   0 1
b → 0 0
   0 0
   0 1
   0 1
c → 1 0
   1 0
d → 0 1
   0 1
   0 1
   0 1
e → 0 1
   0 1
   0 1
   0 1
p → 1 1
   1 1
   1 1
   1 1

Genome: SABCD Acpac Baaae Caaaa Daaab a0000 b0001 ...

S → A B
   C D
  → cp aa
     ac ae
     aa a
     aa a b
  → 1 0 1 1 0 0 0 0
     0 1 1 1 0 0 0 0
     0 0 1 0 0 0 0 1
     0 0 0 1 0 0 0 1
     0 0 0 0 0 0 0 0
     0 0 0 0 0 0 0 0
     0 0 0 0 0 0 0 0
   • A 1 at $i,i$ means node exists.
   • A 1 at $i,j$ means node $i$ connects to node $j$. 
S: Sequential division - first child inherits input links; second inherits output links
P: Parallel division - both children inherit input and output links
E: Ending - cell loses its reading head and becomes a neuron
A: Increment threshold
- : Negate threshold
Cellular Encoding
The Killer App: Pole Balancing

(Google on pole balancing demo)

• Apply force $F_t$ at time $t$ to prevent pole from falling over, cart from hitting walls
• “Easy” version: supply $h_t$, $\theta_t$ and their first derivatives (velocities) as input to network; network outputs $F_t$
• Hard version: supply $h_t$, $\theta_t$ only
• State of the art: ESP beats CE on pole-balancing - so is evolving architecture a waste of time?
NEAT: NeuroEvolution of Augmenting Topologies (Stanley & Miikkulainen 2002)

- Stanley & Miikkulainen: No, you’ve got to do it right - solve these problems:

  1. **Competing conventions**: fail to crossover different topologies in a useful way.
  2. **Premature convergence**: promising solutions die off before they have a chance to compete
  3. **Bloat**: complexity (good) leads to useless crap (bad)
Figure 1: The competing conventions problem. The two networks compute the same exact function even though their hidden units appear in a different order and are represented by different chromosomes, making them incompatible for crossover. The figure shows that the two single-point recombinations are both missing one of the 3 main components of each solution. The depicted networks are only 2 of the 6 possible permutations of hidden unit orderings.
NEAT: Genetic Encoding

**Genome (Genotype)**

<table>
<thead>
<tr>
<th>Node Genes</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor</td>
<td>Sensor</td>
<td>Sensor</td>
<td>Output</td>
<td>Hidden</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connect. Genes</th>
<th>In 1</th>
<th>In 2</th>
<th>In 3</th>
<th>In 2</th>
<th>In 5</th>
<th>In 1</th>
<th>In 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out 4</td>
<td>Out 4</td>
<td>Out 4</td>
<td>Out 5</td>
<td>Out 5</td>
<td>Out 5</td>
<td>Out 5</td>
<td>Out 5</td>
</tr>
<tr>
<td>Weight 0.7</td>
<td>Weight 0.5</td>
<td>Weight 0.5</td>
<td>Weight 0.2</td>
<td>Weight 0.4</td>
<td>Weight 0.6</td>
<td>Weight 0.6</td>
<td></td>
</tr>
<tr>
<td>Enabled</td>
<td>Disabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Innov 1</td>
<td>Innov 2</td>
<td>Innov 3</td>
<td>Innov 4</td>
<td>Innov 5</td>
<td>Innov 6</td>
<td>Innov 11</td>
<td></td>
</tr>
</tbody>
</table>

**Network (Phenotype)**

Figure 2: A genotype to phenotype mapping example. A genotype is depicted that produces the shown phenotype. There are 3 input nodes, one hidden, and one output node, and seven connection definitions, one of which is recurrent. The second gene is disabled, so the connection that it specifies (between nodes 2 and 4) is not expressed in the phenotype.
NEAT: Structural Mutation

[Diagram showing examples of mutation operations, such as adding connections and nodes.]

Tuesday, May 14, 13
1. Solving Competing Conventions via Historical Tracking

- Any two different genes started out the same at some point.
- So remember an “innovation number” telling you when they diverged.
- Crossover lines up matching innovation numbers
- Non-matching genes are either
  - **Disjoint** (D): inside range of parent’s innovation #’s
  - **Excess** (D): outside range
Crossover with Innovation Numbers
2. Solving **Premature Convergence** via Speciation

- C.f. sub-population idea from ESP, crowding distance from NSGA-II
- **Compatibility distance**: closer (more compatible) genes get put into the same species

\[
\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \frac{\overline{W}}{N}
\]

- # Excess genes
- # Disjoint genes
- Average of weight difference of matching genes

# genes in larger genome

# Excess genes
2. Solving **Premature Convergence via Speciation**

- If the $\delta$’s of two individuals differ by some threshold value $\delta_t$, they go into different species.

- **Fitness sharing**: each member of a species shares its fitness with the other members of that species. Discourages emergence of a species with many members, because “superstars” would lose their edge.
3. Solving **Bloat** via Incremental Growth from Minimal Structure

- Most GA’s start with a random initial population - introduces junk from the start
- NEAT starts with a uniform population of networks with no hidden nodes
- No structure survives unless it is useful to fitness
Results: Pole Balancing without Velocity Info (DPNV)

<table>
<thead>
<tr>
<th>Method</th>
<th>Evaluations</th>
<th>Generalization</th>
<th>No. Nets</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>840,000</td>
<td>300</td>
<td>16,384</td>
</tr>
<tr>
<td>ESP</td>
<td>169,466</td>
<td>289</td>
<td>1,000</td>
</tr>
<tr>
<td>NEAT</td>
<td>33,184</td>
<td>286</td>
<td>1,000</td>
</tr>
</tbody>
</table>
Learning How to Compute Velocity

Figure 8: A NEAT solution to the DPNV problem. This clever solution works by taking the derivative of the difference in pole angles. Using the recurrent connection to itself, the single hidden node determines whether the poles are falling away or towards each other. This solution allows controlling the system without computing the velocities of each pole separately. Without evolving structure, it would be difficult to discover such subtle and compact solutions. Starting minimally makes discovering such compact solutions more likely.
Visualizing Speciation

1 species
2 species
10 species
Extinction
Killer App #2: NERO Game

- **rNEAT**: real-time NEAT: decouple game/generation link (c.f. Overholtzer & Levy 2005)
- **Neuro Evolving Robotic Operatives**: evolve teams of agents to fight each other