Evolving Neural Networks

Risto Miikkulainen

Department of Computer Science
The University of Texas at Austin
http://www.cs.utexas.edu/~risto

Why Neuroevolution?

- Neural nets powerful in many statistical domains
  - E.g. control, pattern recognition, prediction, decision making
  - Where no good theory of the domain exists
- Good supervised training algorithms exist
  - Learn a nonlinear function that matches the examples
- What if correct outputs are not known?

Sequential Decision Tasks

- POMDP: Sequence of decisions creates a sequence of states
- No targets: Performance evaluated after several decisions
- Many important real-world domains:
  - Robot/vehicle/traffic control
  - Computer/manufacturing/process optimization
  - Game playing

Forming Decision Strategies

- Traditionally designed by hand
  - Too complex: Hard to anticipate all scenarios
  - Too inflexible: Cannot adapt on-line
- Need to discover through exploration
  - Based on sparse reinforcement
  - Associate actions with outcomes
Standard Reinforcement Learning

- AHC, Q-learning, Temporal Differences
  - Generate targets through prediction errors
  - Learn when successive predictions differ
- Predictions represented as a value function
  - Values of alternatives at each state
- Difficult with large/continuous state and action spaces
- Difficult with hidden states

NEuroevolution (NE) Reinforcement Learning

- NE = constructing neural networks with evolutionary algorithms
- Direct nonlinear mapping from sensors to actions
- Large/continuous states and actions easy
  - Generalization in neural networks
- Hidden states disambiguated through memory
  - Recurrency in neural networks \(^{76}\)

How well does it work?

<table>
<thead>
<tr>
<th>Poles</th>
<th>Method</th>
<th>Evals</th>
<th>Succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>VAPS</td>
<td>(500,000)</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>SARSA</td>
<td>13,562</td>
<td>59%</td>
</tr>
<tr>
<td></td>
<td>Q-MLP</td>
<td>11,331</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NE</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>Two</td>
<td>NE</td>
<td>3,416</td>
<td></td>
</tr>
</tbody>
</table>

- Difficult RL benchmark: Non-Markov Pole Balancing
- NE 3 orders of magnitude faster than standard RL \(^{27}\)
- NE can solve harder problems

Role of Neuroevolution

- Powerful method for sequential decision tasks \(^{17,27,51,89}\)
  - Optimizing existing tasks
  - Discovering novel solutions
  - Making new applications possible
- Also may be useful in supervised tasks \(^{47,56}\)
  - Especially when network topology important
- Unique model of biological adaptation and development \(^{52,61,84}\)
Outline

- Basic neuroevolution techniques
- Advanced techniques
  - E.g. combining learning and evolution
- Extensions to applications
- Application examples
  - Control, Robotics, Artificial Life, Games

Neuroevolution Decision Strategies

- Input variables describe the state
- Output variables describe actions
- Network between input and output:
  - Nonlinear hidden nodes
  - Weighted connections
- Execution:
  - Numerical activation of input
  - Performs a nonlinear mapping
  - Memory in recurrent connections

Conventional Neuroevolution (CNE)

- Evolving connection weights in a population of networks
- Chromosomes are strings of connection weights (bits or real)
  - E.g. 10010110101100101111001
  - Usually fully connected, fixed topology
  - Initially random

Conventional Neuroevolution (2)

- Parallel search for a solution network
  - Each NN evaluated in the task
  - Good NN reproduce through crossover, mutation
  - Bad thrown away
- Natural mapping between genotype and phenotype
  - GA and NN are a good match!
Problems with CNE

- Evolution converges the population (as usual with EAs)
  - Diversity is lost; progress stagnates
- Competing conventions
  - Different, incompatible encodings for the same solution
- Too many parameters to be optimized simultaneously
  - Thousands of weight values at once

Advanced NE 1: Evolving Partial Networks

- Evolving individual neurons to cooperate in networks
- E.g. Enforced Sub-Populations (ESP)
  - Each (hidden) neuron in a separate subpopulation
  - Fully connected; weights of each neuron evolved
  - Populations learn compatible subtasks

Evolving Neurons with ESP

- Evolution encourages diversity automatically
  - Good networks require different kinds of neurons
- Evolution discourages competing conventions
  - Neurons optimized for compatible roles
- Large search space divided into subtasks
  - Optimize compatible neurons

Evolving Partial Networks (2)

- Extend the idea to evolving connection weights
- E.g. Cooperative Synapse NeuroEvolution (CoSyNE)
  - Connection weights in separate subpopulations
  - Networks formed by combining neurons with the same index
  - Networks mutated and recombined; indices permutated
- Sustains diversity, results in efficient search
Advanced NE 2: Evolutionary Strategies

- Evolving complete networks with ES (CMA-ES\textsuperscript{34})
- Small populations, no crossover
- Instead, intelligent mutations
  - Adapt covariance matrix of mutation distribution
  - Take into account correlations between weights
- Smaller space, less convergence, fewer conventions

Advanced NE 3: Evolving Topologies

- Optimizing connection weights and network topology\textsuperscript{31,72,191}
- E.g. Neuroevolution of Augmenting Topologies (NEAT\textsuperscript{69,71})
- Based on Complexification
- Of networks:
  - Mutations to add nodes and connections
- Of behavior:
  - Elaborates on earlier behaviors

Why Complexification?

- Problem with NE: Search space is too large
- Complexification keeps the search tractable
  - Start simple, add more sophistication
- Incremental construction of intelligent agents

Advanced NE 4: Indirect Encodings

- Instructions for constructing the network evolved
  - Instead of specifying each unit and connection\textsuperscript{3,17,46,67,91}
- E.g. Cellular Encoding (CE\textsuperscript{29})
- Grammar tree describes construction
  - Sequential and parallel cell division
  - Changing thresholds, weights
  - A “developmental” process that results in a network
Indirect Encodings (2)

- Encode the networks as spatial patterns
- E.g. Hypercube-based NEAT (HyperNEAT\textsuperscript{12})
- Evolve a neural network (CPPN) to generate spatial patterns
  - 2D CPPN: \((x, y)\) input \(\rightarrow\) grayscale output
  - 4D CPPN: \((x_1, y_1, x_2, y_2)\) input \(\rightarrow\) \(w\) output
  - Connectivity and weights can be evolved indirectly
  - Works with very large networks (millions of connections)

Properties of Indirect Encodings

- Smaller search space
- Avoids competing conventions
- Describes classes of networks efficiently
  - Recurrency symbol in CE: XOR \(\rightarrow\) parity
  - Repetition with variation in CPPNs
  - Useful for evolving morphology

Properties of Indirect Encodings

- Not fully explored (yet)
  - See e.g. GDS track at GECCO
- Promising current work
  - More general L-systems; developmental codings: embryogeny\textsuperscript{72}
  - Scaling up spatial coding\textsuperscript{1322}
  - Genetic Regulatory Networks\textsuperscript{57}
  - Evolution of symmetries\textsuperscript{79}

How Do the NE Methods Compare?

<table>
<thead>
<tr>
<th>Poles</th>
<th>Method</th>
<th>Evals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two</td>
<td>CE</td>
<td>(840,000)</td>
</tr>
<tr>
<td></td>
<td>CNE</td>
<td>87,623</td>
</tr>
<tr>
<td></td>
<td>ESP</td>
<td>26,342</td>
</tr>
<tr>
<td></td>
<td>NEAT</td>
<td>6,929</td>
</tr>
<tr>
<td></td>
<td>CMA-ES</td>
<td>5,091</td>
</tr>
<tr>
<td></td>
<td>Cosyne</td>
<td>3,416</td>
</tr>
</tbody>
</table>

Two poles, no velocities, damping fitness\textsuperscript{27}

- Advanced methods better than CNE
- Advanced methods still under development
- Indirect encodings future work
Further NE Techniques

- Incremental, multiobjective, novelty evolution \(^{40;64\ 25;38;39;64;78;90}\)
- Utilizing population culture \(^{5;44}\)
- Evolving NN ensembles and modules \(^{41;55;58;86}\)
- Evolving transfer functions and learning rules \(^{8;60;75}\)
- Evolving value functions \(^{87}\)
- Combining learning and evolution

Combining Learning and Evolution

- Good learning algorithms exist for NN
  - Why not use them as well?
- Evolution provides structure and initial weights
- Fine tune the weights by learning
- Lamarckian evolution is possible
  - Coding weight changes back to chromosome
- Difficult to make it work
  - Diversity reduced; progress stagnates

Baldwin Effect

- Learning can guide Darwinian evolution \(^{4;31}\)
  - Makes fitness evaluations more accurate
- With learning, more likely to find the optimum if close
- Can select between good and bad individuals better
  - Lamarckian not necessary
- How can we implement it?
  - How to obtain training targets?

Targets from a Related Task

- Learning in a related task is sufficient
- E.g. foraging for food in a microworld \(^{52}\)
  - Network sees the state, outputs motor commands
  - Trained with backprop to predict the next input
  - Training emphasizes useful hidden-layer representations
  - Allows more accurate evaluations
Evolving the Targets

- Evolve extra outputs to provide targets
- E.g. in the foraging task\(^5\)4
  - Motor outputs and targets with separate hidden layers
  - Motor weights trained with backprop, targets evolved
  - Targets do not correspond to optimal performance: Direct system towards useful learning experiences

Targets from Humans

- Humans can demonstrate desired behavior
- E.g. fine tuning game agents\(^7\)
  - Human observer identifies suboptimal behavior
  - Drives the NPC with a joystick
  - Agent placed in the same input situation
  - Backpropagate from human actions

Extending NE to Applications

- Control
- Robotics
- Artificial life
- Gaming

Issues:
- Combining evolution with human knowledge\(^7,16,93\) (Karpov GECCO’11)
- Making evolution run in real-time\(^70\)
- Utilizing coevolution\(^59,73\)
- Utilizing problem symmetry and hierarchy\(^36,79,81\)
- Evolving multimodal behavior\(^63,64,86\)
- Evolving teams of agents\(^6,70,92\)

Applications to Control

- Pole-balancing benchmark
  - Originates from the 1960s
  - Original 1-pole version too easy
  - Several extensions: acrobat, jointed, 2-pole, particle chasing\(^55\)
- Good surrogate for other control tasks
  - Vehicles and other physical devices
  - Process control\(^82\)
Controlling a Finless Rocket

Task: Stabilize a finless version of the Interorbital Systems RSX-2 sounding rocket

- Scientific measurements in the upper atmosphere
- 4 liquid-fueled engines with variable thrust
- Without fins will fly much higher for same amount of fuel

Rocket Stability

(a) Fins: stable
(b) Finless: unstable

Active Rocket Guidance

- Used on large scale launch vehicles (Saturn, Titan)
- Typically based on classical linear feedback control
- High level of domain knowledge required
- Expensive, heavy

Simulation Environment: JSBSim

- General rocket simulator
- Models complex interaction between airframe, propulsion, aerodynamics, and atmosphere
- Used by IOS in testing their rocket designs
- Accurate geometric model of the RSX-2
Rocket Guidance Network

Results: Control Policy

Results: Apogee

Applications to Robotics

- Controlling a robot arm
  - Compensates for an inop motor
- Robot walking
  - Various physical platforms
- Mobile robots
  - Transfers from simulation to physical robots
  - Evolution possible on physical robots

DEMO
**Multilegged Walking**

- Navigate rugged terrain better than wheeled robots
- Controller design is more challenging
  - Leg coordination, robustness, stability, fault-tolerance, ...
- Hand-design is generally difficult and brittle
- Large design space often makes evolution ineffective

---

**ENSEO: Symmetry Evolution Approach**

- Symmetry evolution approach
  - A neural network controls each leg
  - Connections between controllers evolved through symmetry breaking
  - Connections within individual controllers evolved through neuroevolution

---

**Robust, Effective Solutions**

- Different gaits on flat ground
  - Pronk, pace, bound, trot
  - Changes gait to get over obstacles
- Asymmetric gait on inclines
  - One leg pushes up, others forward
  - Hard to design by hand
- DEMO

---

**Transfer to a Physical Robot**

- Built at Hod Lipson’s lab (Cornell U.)
  - Standard motors, battery, controller board
  - Custom 3D-printed legs, attachments
  - Simulation modified to match
- General, robust transfer
  - Noise to actuators during simulation
  - Generalizes to different surfaces, motor speeds
  - Evolved a solution for 3-legged walking!
- DEMO
Driving and Collision Warning

- Goal: evolve a collision warning system
  - Looking over the driver’s shoulder
  - Adapting to drivers and conditions
  - Collaboration with Toyota

The RARS Domain

- RARS: Robot Auto Racing Simulator
  - Internet racing community
  - Hand-designed cars and drivers
  - First step towards real traffic

Evolving Good Drivers

- Evolving to drive fast without crashing (off road, obstacles)
- An interesting challenge of its own
- Discovers optimal driving strategies (e.g. how to take curves)
- Works from range-finder & radar inputs
- Works from raw visual inputs (20 × 14 grayscale)

Evolving Warnings

- Evolving to estimate probability of crash
- Predicts based on subtle cues (e.g. skidding off the road)
- Compensates for disabled drivers
- Human drivers learn to drive with it!
- DEMO
Transferring to the Physical World?

- Applied AI Gaia moving in an office environment
  - Sick laserfinder; Bumblebee digital camera
  - Driven by hand to collect data
- Learns collision warning in both cases
- Transfer to real cars?
- DEMO

Applications to Artificial Life

- Gaining insight into neural structure
  - E.g. evolving a command neuron
- Emergence of behaviors
  - Signaling, herding, hunting...
- Future challenges
  - Emergence of language
  - Emergence of community behavior

Emergence of Cooperation and Competition

- Predator-prey simulations
  - Predator species, prey species
  - Prior work single pred/prey, team of pred/prey
- Simultaneous competitive and cooperative coevolution
- Understanding e.g. hyenas and zebras
  - Collaboration with biologists (Kay Holekamp, MSU)
- DEMO

Open Questions

- Role of communication
  - Stigmergy vs. direct communication in hunting
  - Quorum sensing in e.g. confronting lions
- Role of rankings
  - Efficient selection when evaluation is costly?
- Role of individual vs. team rewards
- Can lead to general computational insights
Applications to Games

- Good research platform\textsuperscript{45}
  - Controlled domains, clear performance, safe
  - Economically important; training games possible
- Board games: beyond limits of search
  - Evaluation functions in checkers, chess\textsuperscript{9;19;20}
  - Filtering information in go, othello\textsuperscript{48;74}
  - Opponent modeling in poker\textsuperscript{42}

Video Games

- Economically and socially important
- GOFAI does not work well
  - Embedded, real-time, noisy, multiagent, changing
  - Adaptation a major component
- Possibly research catalyst for CI
  - Like board games were for GOFAI in the 1980s

Video Games (2)

- Can be used to build “mods” to existing games
  - Adapting characters, assistants, tools
- Can also be used to build new games
  - New genre: Machine Learning game

BotPrize Competition

- Turing Test for game bots: $10,000 prize
- Three players in Unreal Tournament:
  - Human confederate: tries to win
  - Software bot: pretends to be human
  - Human judge: tries to tell them apart!
- DEMO
Killer App: Evolving an Unreal Bot

- Evolve basic strategies
  - Battle, chase, get-unstuck...
  - Can be extended to many other low-level behaviors
- Best bots judged 25-30% human
  - Vs. humans at 35-80%
- Fascinating challenges remain:
  - Judges can still differentiate in seconds
  - Judges lay cognitive, high-level traps

NERO Gameplay

- Teams of agents trained to battle each other
  - Player trains agents through exercises
  - Agents evolve in real time
  - Agents and player collaborate in battle
- New genre: Learning is the game
  - Challenging platform for reinforcement learning
  - Real time, open ended, requires discovery
- Try it out:
  - Available for download at http://nerogame.org
  - Open source research platform version at opennero.googlecode.com

A New Genre: Machine Learning Games

- E.g. NERO
  - Goal: to show that machine learning games are viable
  - Professionally produced by Digital Media Collaboratory, UT Austin
  - Developed mostly by volunteer undergraduates

Real-time NEAT

- A parallel, continuous version of NEAT
- Individuals created and replaced every $n$ ticks
- Parents selected probabilistically, weighted by fitness
- Long-term evolution equivalent to generational NEAT
NERO Player Actions

- Player can place items on the field
  - e.g. static enemies, turrets, walls, rovers, flags
- Sliders specify relative importance of goals
  - e.g. approach/avoid enemy, cluster/disperse, hit target, avoid fire...
- Networks evolved to control the agents
- DEMO

Numerous Other Applications

- Creating art, music, dance...\(^{10,15,32,65}\)
- Theorem proving\(^{14}\)
- Time-series prediction\(^{43}\)
- Computer system optimization\(^{24}\)
- Manufacturing optimization\(^{28}\)
- Process control optimization\(^{82,83}\)
- Measuring top quark mass\(^{88}\)
- Etc.

Evaluation of Applications

- Neuroevolution strengths
  - Can work very fast, even in real-time
  - Potential for arms race, discovery
  - Effective in continuous, non-Markov domains
- Requires many evaluations
  - Requires an interactive domain for feedback
  - Best when parallel evaluations possible
  - Works with a simulator & transfer to domain

Conclusion

- NE is a powerful technology for sequential decision tasks
  - Evolutionary computation and neural nets are a good match
  - Lends itself to many extensions
  - Powerful in applications
- Easy to adapt to applications
  - Control, robotics, optimization
  - Artificial life, biology
  - Gaming: entertainment, training
- Lots of future work opportunities
  - Theory not well developed
  - Indirect encodings
  - Learning and evolution
  - Knowledge and interaction
References


--End of Document--