Evolving Neural Networks

Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
http://www.cs.utexas.edu/users/risto

Why Neuroevolution?

- Neural nets powerful in many statistical domains
  - E.g. control, pattern recognition, prediction, decision making
  - 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
  – Recurrancy in neural networks\(^{61}\) (Taylor GECCO’06)

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>589</td>
<td></td>
</tr>
<tr>
<td>Two</td>
<td>NE</td>
<td>24,543</td>
<td></td>
</tr>
</tbody>
</table>

- Difficult RL benchmark: Non-Markov Pole Balancing
- NE 2 orders of magnitude faster than standard RL
- NE can solve harder problems

Role of Neuroevolution

• Powerful method for sequential decision tasks\(^{19,40,71}\)
  – Optimizing existing tasks
  – Discovering novel solutions
  – Making new applications possible

• Also may be useful in supervised tasks\(^{35,45}\)
  – Especially when network topology important

• Unique model of biological adaptation and development\(^{41,49,67}\)
Outline

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

Neuroevolution Decision Strategies

• Input variables describe the state
• Output variables describe actions
• Network between input and output
  – Hidden nodes
  – Weighted connections
• Execution:
  – Numerical activation of input
  – Nonlinear weighted sums
• Performs a nonlinear mapping
  – Memory in recurrent connections
• Connection weights and structure evolved

Conventional Neuroevolution (CNE)

• Evolving connection weights in a population of networks

• Chromosomes are strings of weights (bits or real)
  – E.g. 100101101011011
  – Usually fully connected, fixed topology
  – Initially random

Conventional Neuroevolution (2)

• Each NN evaluated in the task
  – Good NN reproduce through crossover, mutation
  – Bad thrown away
  – Over time, NNs evolve that solve the task
• 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 Neurons

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

Advanced NE 2: Evolutionary Strategies

- Evolving complete networks with ES (CMA-ES)
- 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
- E.g. Neuroevolution of Augmenting Topologies (NEAT)
- Based on Complexification
  - Of networks: Mutations to add nodes and connections
  - Of behavior: Elaborates on earlier behaviors

How Can Crossover be Implemented?

- Problem: Structures do not match
- Solution: Utilize historical markings

How Can Innovation Survive?

- Problem: Innovations have initially low fitness
- Solution: Speciate the population
  - Innovations have time to optimize
  - Mitigates competing conventions
  - Promotes diversity

How Can We Search in Large Spaces?

- Need to optimize not just weights but also topologies
- Solution: Start with minimal structure and complexify
  - Hidden nodes, connections, input features

Generations pass...
Population of Diverse Topologies
Advanced NE 4: Indirect Encodings

- Instructions for constructing the network evolved
  - Instead of specifying each unit and connection \(^2,33,51,73\)
- E.g. Cellular Encoding (CE \(^{24}\))
- Grammar tree describes construction
  - Sequential and parallel cell division
  - Changing thresholds, weights
  - A “developmental” process that results in a network

Properties of Indirect Encodings

- Smaller search space
- Avoids competing conventions
- Describes classes of networks efficiently
- Modularity, reuse of structures
  - Recurrency symbol in CE: XOR → parity
  - Useful for evolving morphology
- Not all that powerful (yet)
- Much future work needed \(^{57}\)
  - More general L-systems
  - Developmental codings, embryogeny
  - Designing evolvable representations \(^{46}\)
    (Reisinger GECCO’06)

How Do the NE Methods Compare?

<table>
<thead>
<tr>
<th>Poles</th>
<th>Method</th>
<th>Evals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-1</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>24,543</td>
</tr>
<tr>
<td>Two-2</td>
<td>CMA-ES</td>
<td>6,061 - 25,254</td>
</tr>
<tr>
<td></td>
<td>ESP</td>
<td>7,374</td>
</tr>
<tr>
<td></td>
<td>NEAT</td>
<td>6,929</td>
</tr>
</tbody>
</table>

Two poles, no velocities, 2 different setups:
- Advanced methods better than CNE
- Advanced methods about equal
- Indirect encodings future work
- DEMO

Further NE Techniques

- Incremental evolution \(^{21,63,72}\)
- Utilizing population culture \(^4,32\)
- Evolving ensembles of NNs \(^{29,44,69}\)
- Evolving neural modules \(^{47}\)
- Evolving transfer functions and learning rules \(^{7,48,60}\)
- 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\(^{3,25}\)
  - 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\(^{41}\)
  - 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\(^{43}\)
  - 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 the Population

- Train new offspring to imitate parents/champion
  - Trained in population “culture”
- Local search around good individuals
  - Limited training: 8-20 backprop iterations
- Becomes part of the evaluation
  - Individuals evolve to anticipate training
  - Perform poorly at birth, well after training
- Evolution discovers optimal starting points for learning!

No Targets: Unsupervised Learning

- Hebbian adaptation during performance
  - E.g. handwritten character recognition
    - Evolution determines the starting point
    - Competitive learning finishes the design
- Starting points are poor recognizers
  - Only bias learning away from local minima
- Synergetic effect: Evolution utilizes learning
- Future work: Constructing developmental systems

Extending NE to Applications

- Evolving composite decision makers
- Evolving teams of agents
- Utilizing coevolution
- Real-time neuroevolution
- Combining human knowledge with evolution

Applications to Control

- Pole-balancing benchmark
  - Originates from the 1960s
  - Original 1-pole version too easy
  - Several extensions: acrobat, jointed, 2-pole, particle chasing
- Good surrogate for other control tasks
  - Vehicles and other physical devices
  - Process control
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 the same amount of fuel

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

Rocket Stability

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

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

Control Policy

Results: Apogee

Driving and Collision Warning

- Goal: evolve a collision warning system
  - Looking over the driver’s shoulder
  - Adapting to drivers and conditions
  - Collaboration with Toyota\(^2\) (Kohl GECCO’06)
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)
- Discovers optimal driving strategies (e.g. how to take curves)
- Works from range-finder & radar inputs
- Works from raw visual inputs
- DEMO

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

Applications to Robotics

- Controlling a robot arm$^{38}$
  - Compensates for an inop motor
- Robot walking$^{26,50}$
  - Various physical platforms
- Mobile robots$^{10,14,42,52}$
  - Transfers from simulation to physical robots
  - Evolution possible on physical robots
- DEMO
Robotic Soccer

- E.g. robocup soccer “Keepaway” task
- Three keepers, one (algorithmic) taker
- Includes many behaviors:
  Get-Open, Intercept, Evaluate-Pass, Pass...

Direct Evolution

- Mapping sensors directly to actions
  - Difficult to separate behaviors
  - Ineffective combinations evolve
- DEMO

Cooperative Coevolution

- Evolve multiple actions
  - Each one in a separate network
  - Decision tree to decide on actions
  - Or evolve a decision network

Cooperative Coevolution (2)

- Networks learn individual tasks
- Learn to anticipate other tasks
  - Lining up for a pass
- Cooperative coevolution of composite behavior
- 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

Competitive Coevolution

- Evolution requires an opponent to beat
- Such opponents are not always available
- Co-evolve two populations to outdo each other
- How to maintain an arms race? (Monroy GECCO’06)

Competitive Coevolution with NEAT

- Complexification elaborates instead of alters
  - Adding more complexity to existing behaviors
- Can establish a coevolutionary arms race
  - Two populations continually outdo each other
  - Absolute progress, not just tricks

Robot Duel Domain

- Two Khepera-like robots forage, pursue, evade
  - Collect food to gain energy
  - Win by crashing to a weaker robot
Early Strategies

- Crash when higher energy
- Collect food by accident
- DEMO

Mature Strategies

- Collect food to gain energy
- Avoid moving to lose energy
- Standoff: Difficult to predict outcome
- DEMO

A Sophisticated Strategy

- “Fake” a move up, force away from last piece
- Win by making a dash to last piece
- Complexification → arms race
- DEMO

Applications to Games

- Good research platform
  - Controlled domains, clear performance, safe
  - Economically important; training games possible
- Board games: beyond limits of search
  - Evaluation functions in checkers, chess\(^8,16,17\)
  - Filtering information in go, othello\(^{36,59}\)
Discovering Novel Strategies in Othello

Players take turns placing pieces
Each move must flank opponent’s piece
Surrounded pieces are flipped
Player with most pieces wins

Strategies in Othello

Positional
- Number of pieces and their positions
- Typical novice strategy

Mobility
- Number of available moves: force a bad move
- Much more powerful, but counterintuitive
- Discovered in 1970’s in Japan

Evolving Against a Random Player

Network sees the board, suggests moves by ranking
Networks maximize piece counts throughout the game
A positional strategy emerges
Achieved 97% winning percentage

Evolving Against an $\alpha$-$\beta$ Program

Iago’s positional strategy destroyed networks at first
Evolution turned low piece count into an advantage
Mobility strategy emerged!
Achieved 70% winning percentage
Example game

- Black’s positions strong, but mobility weak
- White (the network) moves to f2
- Black’s available moves b2, g2, and g7 each will surrender a corner
- The network wins by forcing a bad move

Discovering Novel Strategies

- Neuroevolution discovered a strategy novel to us
- “Evolution works by tinkering”
  - So does neuroevolution
  - Initial disadvantage turns into novel advantage

Video Games

- Economically and socially important
- Adaptation an important future goal
  - More challenging, more fun games
  - Possible to use for training people
- How to make evolution run in real time?

Real-time NEAT

- A parallel, continuous version of NEAT\textsuperscript{54}
- Individuals created and replaced every $n$ ticks
- Parents selected probabilistically, weighted by fitness
- Long-term evolution equivalent to generational NEAT
NERO: A Complex Game Platform

- Teams of agents trained to battle each other
  - Player trains agents through exercises
  - Agents evolve in real time
- New genre: Learning is the game
- Challenging platform for reinforcement learning
  - Real time, open ended, requires discovery
- DEMO

Future Challenge: Utilizing Knowledge

- Given a problem, NE discovers a solution by exploring
  - Sometimes you already know (roughly) what works
  - Sometimes random initial behavior is not acceptable
- How can domain knowledge be utilized?
  - By incorporating rules
  - By learning from examples

Incorporating Rules into NE

E.g. how to go around a wall in NERO

- Specify as a rule:
  - `wall_ahead: move_forward, turn_right`
  - `wall_45deg_left, move_forward, turn_right_slightly`
- Convert into a network with KBANN

Incorporating Rules into NE (2)

- KBANN network added to NEAT networks
  - Treated as complexification
  - Continues to evolve
  - If advice is useful, it stays
- Initial behaviors, on-line advice
- Injecting human knowledge as rules
- DEMO
Lessons from NERO

- NEAT is a strong method for real-time adaptation
  - Complex team behaviors can be constructed
  - Novel strategies can be discovered
- Problem solving with human guidance
- Coevolutionary arms race
- NE makes a new genre of games possible!

(NERO details, download: http://nerogame.org)

Numerous Other Applications

- Creating art, music
- Theorem proving
- Time-series prediction
- Computer system optimization
- Manufacturing optimization
- Process control optimization
- 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


[65] A. v. E. Conradie, R. Miikkulainen, and C. Aldrich, Intelligent process control utilizing symbiotic memetic neuro-