Comparing Direct and Indirect Encodings Using Both Raw and Hand-Designed Features in Tetris

By Lauren Gillespie, Gabby Gonzales and Jacob Schrum

gillespl@southwestern.edu  gonzale9@alumni.southwestern.edu  schrum2@southwestern.edu
Introduction

- Challenge: Use less domain-specific knowledge
  - Important for general agents
  - Accomplished using raw inputs
  - Need to be able to process with a neural network
- Why challenging?
  - Complex domains = Large input space
  - Large input space = Large neural networks
  - Large neural networks = Difficult to train
Addressing Challenges

- Deep Learning applies large NN to hard tasks†
- HyperNEAT also capable of handling large NNs
  - Indirect encoding, good with geometric inputs‡
  - Compare to direct encoding, NEAT
  - See if indirect encoding advantageous
  - Also compare with hand-designed features

Direct Vs. Indirect Encoding

Evolved network and agent network

Direct Encoding (NEAT)

Agent network

Indirect Encoding (HyperNEAT)

Evolved network
Tetris Domain

- Consists of 10 x 20 game board
- Orient tetrominoes to clear lines
- Clearing multiple lines = more points
- NP-Complete domain†
- One piece controller
  - Agent has knowledge of current piece only

† Breukelaar et al. 2004. Tetris is hard, even to approximate.
Previous Work

• Tetris Domain
  ✦ All use hand-designed features
  ✦ Reinforcement Learning:
    ✦ Policy search: Szitza & Lörincz 2006
    ✦ Approximate Dynamic Programming: Gabillon et al. 2013
  ✦ Evolutionary Computation:
    ✦ Simple EA with linear function approximator: Böhm et al. 2004
    ✦ Covariance Matrix Adaptation Evolution Strategy: Boumaza 2009
• Raw Visual Inputs
  ✦ Neuroevolution: Gauci & Stanley 2008, Verbancsics & Stanley 2010
Hand-Designed Features

- Most common input scheme for training ANNs†
- Hand-picked information of game state as input

Pros:
- Network doesn’t deal with excess info
- Smaller input space, easier to learn

Cons:
- Very domain-specific, not versatile
- Human expertise needed
- Useful features not always apparent

† Schrum & Miikkulainen. 2016. Discovering Multimodal Behavior in Ms. Pac-Man through Evolution of Modular Neural Networks.
Raw Features

- One feature per game state element
- Minimal input processing by user

**Pros:**
- Networks less limited by domain†
- Less human expertise needed

**Cons:**
- Large input space & networks
- Harder to learn, more time

NEAT

- NeuroEvolution of Augmenting Topologies†
- Synaptic and structural mutations
- Direct encoding
  - Network size proportional to genome size
- Crossover alignment via historical markings
- Inefficient with large input sets
  - Mutations do not alter behavior effectively

† Stanley & Miikkulainen. 2002. Evolving Neural Networks Through Augmenting Topologies
HyperNEAT

• Hypercube-based NEAT†
• Extension of NEAT
• Indirect encoding
  ‡ Evolved CPPNs encode larger substrate-based agent ANNs
• Compositional Pattern-Producing Networks (CPPNs)
  ‡ CPPN queried across substrate to create agent ANN
  ‡ Inputs = neuron coordinates, outputs = link weights
• Substrates
  ‡ Layers of neurons with geometric coordinates
  ‡ Substrate layout determined by domain/experimenter

† Stanley et al. 2009. A Hypercube-based Encoding for Evolving Large-scale Neural Networks
**HyperNEAT with Tetris**

- *Geometric awareness*: arises from indirect encoding
- CPPN encodes geometry of domain into agent via substrates
- Agent network can learn from task-relevant domain geometry
Raw Features Setup

- Board configuration:
  - Two input sets
    1. Location of all blocks
      - block = 1, no block = 0
    2. Location of all holes
      - hole = -1, no hole = 0
- NEAT: Inputs in linear sequence
- HyperNEAT: Two 2D input substrates
Hand-Designed Features Setup

- Bertsekas et al. features\(^\dagger\) plus additional hole per column feature
- All scaled to \([0,1]\)
  - Column height
  - Height difference
  - Tallest column
  - Number of holes
  - Holes per column

\(^\dagger\) Bersekas et al. 1996. Neuro-Dynamic Programming
Experimental Setup

- Agent networks are afterstate evaluators
- Each experiment evaluated with 30 runs
  - 500 generations/run, 50 agents/generation
  - Objectives averaged across 3 trials/agent
    - Noisy domain, multiple trials needed
- NSGA-II objectives: game score & survival time
NEAT vs. HyperNEAT: Raw Features

Game Score vs. Generation for HyperNEAT Raw and NEAT Raw.
NEAT vs. HyperNEAT: Hand-Designed Features
Raw Features Champion Behavior

NEAT with Raw Features

HyperNEAT with Raw Features
Hand-Designed Features Behavior

NEAT with Hand-Designed Features

HyperNEAT with Hand-Designed Features
Visualizing Substrates

Inputs  Hidden  Output  Result
Discussion

• Raw features: HyperNEAT clearly better than NEAT
  ✦ Indirect encoding advantageous
  ✦ NEAT ineffective at evolving large networks

• Hand-Designed: HyperNEAT has less of an advantage
  ✦ Geometric awareness less important
  ✦ HyperNEAT CPPN limited by substrate topology
Future Work

• HybrID†
  - Start with HyperNEAT, switch to NEAT
  - Gain advantage of both encodings

• Raw feature Tetris with Deep Learning

• Raw features in other visual domains
  - Video games: DOOM, Mario, Ms. Pac-Man
  - Board games: Othello, Checkers

Conclusion

- Raw features
  - Indirect encoding HyperNEAT effective
  - Geometric awareness an advantage
- Hand-designed features
  - Ultimately NEAT produced better agents
  - HybrID might combine strengths of both
Questions?

• Contact info:
  gillespl@southwestern.edu
  schrum2@southwestern.edu
  gonzale9@alumni.southwestern.edu

• Movies and Code:
  https://tinyurl.com/tetris-gecco2017
Auxiliary Slides
NSGA-II

• Pareto-based multiobjective EA optimization
• Parent population, $\mu$, evaluated in domain
• Child population, $\lambda$, evolved from $\mu$ and evaluated
• $\mu + \lambda$ sorted into non-dominated Pareto fronts
  • **Pareto front:** All individual such that
  • $v = (v_1, \ldots, v_n)$ dominates vector $u = (u_1, \ldots, u_n)$ iff
    1. $\forall i \in \{1, \ldots, n\}: v_i \geq u_i$, and
    2. $\exists i \in \{1, \ldots, n\}: v_i > u_i$.
• New $\mu$ picked from highest fronts
• Tetris objectives: Game score, time
Visualizing Link Weights
Afterstate Evaluation

- Evolved agents used as afterstate evaluators
- Determine next move from state after placing piece
- All possible piece locations determined, evaluated
- Placement with best evaluation from state chosen
- If placements lead to loss, not considered
- Agent moves piece to best placement, repeats