Concurrent Layered Learning

Shimon Whiteson, Peter Stone
Department of Computer Science
University of Texas at Austin

CSCI2950-Z
March. 15. 2010
Presentation by Hojoon Park
Outline

- Background
- Traditional Layered Learning
- Concurrent Layered Learning
- Empirical Results
- Discussion
- Conclusion
Background

- Traditional Layered Learning
  - Hierarchical paradigm
  - Low-Level (simple) behaviors are trained prior to High-Level (complex) behaviors
  - Each learned layer is frozen before learning higher layer

- Concurrent Layered Learning
  - Same formalism with existing layered learning
  - Lower layers allow to keep learning concurrently with the training of subsequent layers
Principles of the layered learning

- Principle 1
  - A direct mapping from inputs to outputs is not tractably learnable
- Principle 2
  - A bottom-up, hierarchical task decomposition is given
- Principle 3
  - Machine learning exploits data to train and/or adapt
  - Learning occurs separately at each level
- Principle 4
  - The output of learning in one layer feeds into the next layer
Formalism for Layered Learning

\[ L_i = (\vec{F}_i, O_i, T_i, M_i, h_i) \]

- \( \vec{F}_i \) is the input vector of state features
  - ex) \( F_4: \{\text{Ball}_r, \text{Ball}_t, \text{Taker}_r, \text{Taker}_t, \text{Boundary}_r\} \)
- \( O_i \) is the set of outputs
- \( T_i \) is the set of training examples used for learning subtask
- \( M_i \) is the ML algorithm used at \( L_i \)
- \( h_i \) is the result of running \( M_i \) on \( T_i \)
  - \( h_i \) is used to construct one or more feature \( F_{i+1} \)
  - \( h_i \) is used to construct elements of \( T_{i+1} \)
  - \( h_i \) is used to prune the output set \( O_{i+1} \)
Background

- **Neuro-Evolution**
  - Machine Learning algorithm
  - Strings weights to form an individual genome
  - Evolve a population of the individual genomes
Background

Enforced Sub-populations Method (ESP)
- Advanced neuro-evolution technique
- Evolves sub-populations of neurons, not complete network
Background

- Coevolution
  - Multi-Agent ESP method to co evolve

- Delta-Code
  - Seed new population from $L_i$
Background

- Keepaway
  If the taker touches the ball or the ball touches the bounding circle, then the game ends.
Traditional Layered Learning in Keepaway

Decision Tree

Layered Learning Hierarchy
Traditional Layered Learning in Keepaway

- **L₁: Intercept**
  - Goal: Get to the ball quickly
  - Input: Two ball’s current positions and two ball’s current velocities.
  - Output: Heading and Speed

- **L₂: Pass**
  - Goal: Kick the ball at a specified angle
  - Input: Two ball’s current positions and one target angle
  - Output: Heading and Speed
**Traditional Layered Learning in Keepaway**

- **L₃**: Pass Evaluation
  - **Goal**: Analyze the current state of the game and assess the likelihood of successfully passing to a specific receiver
  - **F₃**: \{Ballᵣ, Ballᵣ, Takerᵣ, Takerᵣ, Teammateᵣ, Teammateᵣ\}
  - **O₃**: \{confidence\}
  - **M₃**: 6 inputs, 2 hidden nodes, 1 output
  - **h₃**: a trained pass evaluator

- **T₃**: Training environment
L₄: Get Open
Goal: Move to a position where it can receive a pass
F₄: \{Ballᵣ, Ballᵫ, Takerᵣ, Takerᵫ, Boundaryᵣ\}
O₄: \{Heading, Speed\}
M₄: 6 inputs, 2 hidden nodes, 1 output
h₄: a trained get open behavior
Problems in Pass evaluation
- $L_3$ and $L_4$ can get discrepancy in a real game
- Because the order of the decision tree and the layered learning hierarchy are different.
- Inverting the order of two layer($L_3$, $L_4$) cannot solve the problem

Solution
- Concurrent layered learning, by allowing the lower layer to adjust to its new environment, will get better performance.
Concurrent Layered Learning in Keepaway

Problems in Pass evaluation

L3
Near Ball?

Yes

Teammate #1 Safer?

Yes
Pass To Teammate #1

No

No

Pass To Teammate #2

Passed To?

Yes

Intercept

No

Get Open

Solution

Concurrent layered learning, by allowing the lower layer to adjust to its new environment, will get better performance.
Concurrent Layered Learning in Keepaway

Problems in Pass evaluation

- $L_3$ and $L_4$ can get discrepancy in a real game
  - Because the order of the decision tree and the layered learning hierarchy are different.
- Inverting the order of two layer($L_3$, $L_4$) cannot solve the problem

Solution

- Concurrent layered learning, by allowing the lower layer to adjust to its new environment, will get better performance.
Empirical Result

VIDEOS

- Coevolution: Before Training | After Training
- Layered Learning: Before Training | After Training
- Concurrent Layered Learning: Before Training | After Training
Empirical Results

Traditional vs. Concurrent Layered Learning (Intercepting Receivers). When learning \( L_3 \), the potential receivers use the intercept behavior.
Empirical Results

Traditional vs. Concurrent Layered Learning in Keepaway (Stationary Receivers). When learning L₃ the potential receivers are stationary until the passer decides to kick to one of them, at which point the selected receiver begins to intercept.
Empirical Results

Coevolution from Scratch vs. Concurrent Layered learning (Stationary Receivers).

Coevolution from Scratch vs. Concurrent Layered Learning (Stationary Receivers)
Discussion

- Concurrent layered learning is useful, but not in all cases.
- Traditional layered learning performs just as well as, or outperforms, concurrent layered learning in some cases due to its more aggressive use of hierarchy. However, there are many instances that it cannot create perfect training environment for the lower layers.
Conclusion

- Concurrent layered learning is an effective option

- Add more instance both in the keepaway task and other tasks

- Finding the conditions under which concurrent layered learning beneficial