Reinforcement Learning of Strategies for Settlers of Catan

Michael Pfeiffer
pfeiffer@igi.tugraz.at

Institute for Theoretical Computer Science
Graz University of Technology, Austria
Motivation

- Computer Game AI
  - Mainly relies on prior knowledge of AI designer
  - inflexible and non-adaptive

- Machine Learning in Games
  - successfully used for classical board games

- TD Gammon [Tesauro 95]
  - self-play reinforcement learning
  - playing strength of human grandmasters

Figures from Sutton, Barto: Reinforcement Learning
Goal of this Work

- Demonstrate **self-play Reinforcement Learning** (RL) for a large and complex game
  - Settlers of Catan: popular board game
  - more in common with commercial computer strategy games than backgammon or chess
  - in terms of: number of players, possibilities of actions, interaction, non-determinism, ...

- **New RL methods**
  - model tree-based function approximation
  - speeding up learning

- Combination of **learning and knowledge**
  - Where in the learning process can we use our prior knowledge about the game?
Agenda

- Introduction
- Settlers of Catan
- Method
- Results
- Conclusion
The Task: Settlers of Catan

- Resources
- Production
- Construction
- Victory Points
- Trading
- Strategies
What makes Settlers so difficult?

- Huge state and action space
- 4 players
- Non-deterministic environment
- Interaction with opponents
Agenda

- Introduction
- Settlers of Catan
  - Method
- Results
- Conclusion
Reinforcement Learning

- Goal: Maximize cumulative discounted *rewards*
- Learn optimal *state-action value function* $Q^*(s,a)$
- Learning of strategies through *interaction* with the environment
  - Try out actions to get an estimate of $Q$
  - Explore new actions, exploit good actions
  - Improve currently learned policies
  - Various learning algorithms: Q-Learning, SARSA, ...
Self Play

- How to simulate opponents?
- Agent learns by **playing against itself**
- Co-evolutionary approach
- Most successful approach for RL in Games
  - **TD-Gammon** [Tesauro 95]
  - Apparently works better in non-deterministic games
  - Sufficient exploration must be guaranteed
Typical Problems of RL in Games

- **State Space** is too large
  - Value Function Approximation

- **Action Space** is too large
  - Hierarchy of Actions

- **Learning Time** is too long
  - Suitable Representation and Approximation Method

- Even **obvious moves** need to be discovered
  - A-priori Knowledge
Function Approximation

- Impossible to visit whole state space
- Need for **generalization** from visited states to whole state space

**Regression Task:** $Q(s, a) \approx F(\varphi, a, \theta)$
  - $\varphi$ ... *feature* representation of $s$
  - $\theta$ ... finite *parameter* vector (e.g. weights of linear functions or ANNs)

**Features for Settlers of Catan:**
  - 216 high-level concept features (using *knowledge*)
Choice of Approximator

- **Discontinuities** in value function
  - global smoothing is undesirable

- **Local importance** of certain features
  - impossible with linear methods

- **Learning time** is crucial

- [Sridharan and Tesauro, 00] Tree based approximation techniques learn faster than ANNs in such scenarios
Model Trees

- Q-values are targets
- Partition state space into **homogeneous** regions
- Learn **local linear regression models** in leaves
- Generalization via **Pruning**

- M5 learning algorithm [Quinlan, 92]
Pros and Cons of Model Trees

+ Discrete and real-valued features
+ Ignores irrelevant features
+ Local models
+ Feature combinations
+ Discontinuities
+ Easy interpretation
+ Few parameters

- Only offline learning
- Need to store all training examples
- Long training time
- Little experience in RL context
- No convergence results in RL context
Offline Training Algorithm

One model tree approximates Q-function for one action

1. Use current policy to play 1000 training games
2. Store game traces (states, actions, rewards, successor states) of all 4 players
3. Use current Q-function approximation (model trees) to calculate Q-values of training examples and add them to existing training set
4. Update older training examples
5. Build new model trees from the updated training set
6. Go back to step 1
Hierarchical RL

- Division of action space
- Three-layer model
- Easier integration of a-priori knowledge
- Learned information defines primitive actions
  - e.g. placing roads, trading
- Independent Rewards:
  - high level: winning the game
  - low level: reaching the behavior‘s goal
  - otherwise zero

Diagram:
- Environment
  - High Level
    - Behaviors
      - 1 2 3 4 5 6 7 8
  - Low Level 1
    - Actions
      - 1 2 3 ... 12
    - Primitive Action 1
      - A-priori Knowledge
      - Low Level 2
      - Low Level 3
      - Low Level 7
      - Low Level 8
      - Prim. Act. 2
      - Prim. Act. 3
      - Prim. Act. 11
      - Prim. Act. 12

Instructions:
- invokes
- provides knowledge
- rewards
Example: Trading

- Select which trades to **offer / accept / reject**
- **Evaluation** of a trade:
  - What increase in **low-level value** would each trade bring?
  - Select highest valued trade

- **Simplification of game design**
  - **No economical model** needed
  - Gain in value function naturally replaces prices
Approaches

- Apply **Learning** or **prior Knowledge** at different stages of learning architecture

- **Pure Learning**
  - learning at *both* levels

- **Heuristic High-Level**
  - simple *hand-coded* high-level strategy
  - learning low-level policies (= *AI modules*)

- **Guided Learning**
  - use hand-coded strategy *only during learning*
  - off-policy learning of high-level policy
Agenda

- Introduction
- Settlers of Catan
- Method
- Results
- Conclusion
Evaluation Method

- 3000 – 8000 training matches per approach
- Long training time
  - 1 day for 1000 training games
  - 1 day for training of model trees
- Evaluation against:
  - random players
  - other approaches
  - human player (myself)
  - no benchmark program
Relative Comparison

- **Combination** of Learning and prior Knowledge yields the best results

- **Low-level Learning** is responsible for improvement

- **Guided** approach:
  - worse than heuristic
  - better than pure learning
Against Human Opponent

- 3 agents vs. author
- Performance Measure:
  - Avg. maximum victory points
  - 10 VP: win every game
  - 8 VP: close to winning in every game
- Heuristic policy wins 2 out of 10 matches
- Best policies are successful against opponents not encountered in training
- Demo matches confirm results (not included here)

Victory Points vs. Human Opponent

Victory points of different strategies against a human opponent (10 games)
Agenda

- Introduction
- Settlers of Catan
- Method
- Results
- Conclusion
Conclusion

- RL works in **large and complex game domains**
  - Not grandmaster level like TD-Gammon, but pretty good
  - Settlers of Catan is an interesting testbed and closer to commercial computer games than backgammon, chess, ...

- **Model trees** as a new **approximation** methodology for RL

- Combination of **prior knowledge with RL** yields promising results
  - Hierarchical learning allows incorporation of knowledge at multiple points of the learning architecture
  - Learning of AI components
  - Knowledge speeds up learning
Future Work

- **Opponent modeling**
  - recognizing and beating certain opponent types

- **Reward filtering**
  - how much of the reward signal is caused by other agents

- **Model trees**
  - other games
  - improvement of offline training algorithm (tree structure)

- **Settlers of Catan** as game AI testbed
  - trying other algorithms
  - improving results
Thank you!
Sources

- J.R. Quinlan: *Learning with Continuous Classes*, Proceedings Australian Joint Conference on AI, 1992
Extra Slides
Approaches

- High-level behaviors always run until completion
- Allowing high-level switches every time-step (*feudal approach*) did not work

- **Module-based Approach**
  - High-level is learned
  - Low-level is learned

- **Heuristic Approach**
  - *Simple hand-coded* high-level strategy during training and in game
  - Low-level is learned
  - Selection of high-level influences primitive actions

- **Guided Approach**
  - Hand-coded high-level strategy *during learning*
  - Off-policy learning of high-level strategy for game
  - Low-level is learned
Comparison of Approaches

- **Module-based:**
  - good low-level choices
  - poor high-level strategy

- **Heuristic high-level:**
  - significant improvement
  - learned low-level clearly responsible for improvement

- **Guided approach:**
  - worse than heuristic
  - better than module-based

Victories of heuristic strategies against other approaches (20 games)
Comparison of Approaches

- Comparison of strategies in games against each other
  - all significantly better than random
  - heuristic is best
Against Human Opponent

- 10 games of each policy vs. author
  - 3 agents vs. human
- Average victory points as measure of performance
  - 10 VP: win every game
  - 8 VP: close to winning in every game
- Only heuristic policy wins 2 out of 10 matches
- Demo matches confirm results (not included here)