A Competitive Approach to Game Learning

Christopher D. Rosin and Richard K. Belew

Presenter: Ersin ELBASI

Introduction

Machine learning is an area of artificial intelligence concerned with the development of techniques which allow computers to "learn".

Game learning is a branch of machine learning where computers learn to play games. Number of players and set of strategies for each player.

Introduction

- Many game learning systems use a competitive approach that repeatedly learns new strategies capable of defeating older ones.
- Goal is find out strategy learning algorithm that is able to learn strategies which defeat a given set of opponents.
- *Competitive algorithm* repeatly uses a strategy learning algorithm to discover strong strategy for the game.

Definition of Games

- A game is a function G maps two inputs h and x (first and second player strategies), G(h,x).
- First player strategy h comes from set of possible strategies. h _ H and x _ X.
- One bit output gives which player winner.
- a > b means strategy a defeats strategy b.

Structure

Exact learning: It is necessary to assume there is a perfect strategy that defeats all possible opposing strategies.

There are two main components

1. Strategy learning algorithm: reinforcement learning, heuristic search etc.
2. Competitive algorithm: Uses the strategy learning algorithms to produce new strategies. (Do not use domain specific knowledge)

How it Works?

Samuel’s original work on checkers. Games between A and B.

- A learns game from reinforcement algorithm. This is strategy learning algorithm.
- When A wins B, B replaced by A.
- The competitive algorithm uses strategy learning algorithm to find a new A strategy to win B strategy.And so on.

Complexity

- Time for a competitive algorithm refers to the total number of strategies considered by it.
- Expected complexity is the lg(H) or lg(X).
- H and X usually not cover all possible strategies; all or most.

Related Work

- Reinforcement learning. Not useful for complex domains.
- Heuristic game learning. Promising results, without domain knowledge.
- Reinforcement learning used to train neural networks for self play games.
- Genetic algorithms have been used with competitive coevolution.
- Etc.

Performance

- For games with at most c perfect strategies and specification number k, using randomized strategy learning algorithm, competitive algorithm complexity is O(k) to learn perfect strategy.
- For games G, transitive chain of length l, competitive algorithm using strategy algorithms require O(l) time to find perfect strategy. (chain is the sequence of h,x pairs)

Two Simple Competitive Algorithms

1. <u>Each Defeats the Last</u>: Algorithm obtains an initial first player strategy s, then find a second player strategy t with t>s, and so on.

This is essentially the competitive algorithm used in Samuel's checkers learning system.

Main problem is the keep choosing strategies in a cycle.

Competitive Algorithms

2. Single Counterexamples: Given a hypothesis, an equivalence query returns "yes" if the hypothesis is the target, and provides a counter example if it is not.

We can say that a counterexample is a second player strategy to defeat first player strategy.

Not sufficient to learn all games.

The Covering Competitive Algorithm

- Covering all First and Second Player Opponents: The first player strategy learning algorithm, at every step, finds a strategy that defeats all second-player strategies already seen.

- Using worst case strategy learning algorithms: This competitive algorithm performs as well as possible with worst case strategy learning algorithms.

- Using a Randomized Strategy Learning Algorithm: Define the (p,q) randomized criterion for sampling from an arbitrary set Y.

At each step of algorithm, denote by X the set of remaining feasible sets of second player strategies.

Examples

- Generalized Guessing Games: Solve the game in time polynomial n using randomized criteria.
- Concept Learning

Complexity

- Who wins? This problem solved by strategy learning algorithm in NP. Competitive algorithm using randomized criterion in lg(H) or lg(X).

- No competitive algorithm exists solves every game in polynomial time.

Future Direction

- Games with polynomial time computable outcomes are not solvable in time polynomial in lg(H) and lg(X). Open question is whether games may be solvable in time polynomial.

- The (p,q) randomization criterion is one condition that allows the covering competitive algorithm to learn perfect strategies in polynomial time.

Future Direction

- For complex games, it is unlikely that natural classes of quickly computable strategies will contain perfect strategies.
- Even when perfect strategies exist, it may be intractable to find them.

Conclusion

- Competitive algorithm is able to successfully bootstrap its way to perfect strategies for a game under general conditions.
- Covering competitive algorithm guarantees progress by ensuring that new strategies defeat all previous strategies.
- Future work should be able to extend results to approximate learning.