LEARNING WITH MONTE-CARLO METHODS

Tristan Cazenave
LAMSADE, Université Paris-Dauphine
Paris
France
cazenave@lamsade.dauphine.fr

Synonyms

Monte-Carlo Tree Search, UCT

Definition

The Monte-Carlo method in games and puzzles consists in playing random games called playouts in order to estimate the value of a position. The method is related to learning since the algorithm dynamically learns which moves are good and which moves are bad as more playouts are played. The learning is achieved by keeping statistics on the outcomes of the random games that started with a move. This algorithm is strongly linked with the area of machine learning named reinforcement learning. It has benefited from research on the multiarmed bandit problem in the area of machine learning (Auer et al. 2002).

The method is related to learning since the algorithm starts learning a position from scratch each time it has to play: it learns the moves that are good for any given position.

When there are no good heuristics for a problem, Monte-Carlo methods can learn to solve an instance of the problem. Consider for example the game of Go, this is a difficult problem for artificial intelligence since evaluating a position with rules and knowledge is very difficult, moreover there are more than 250 possible moves on average and it is necessary to look multiple moves ahead, therefore there is a combinatorial explosion of the number of positions to look at. On the other hand, in the game of Go, it is very simple to program a computer to play many random games. The system can also keep the mean result of the games that start with a given move. After many games the system can stop playing random games and choose to play the move that has the greatest associated mean. This method for choosing a move is called Monte-Carlo Go.

Monte-Carlo Tree Search is a recent refinement of the method. It consists in growing a tree of previous playouts in memory and choosing a move according to the mean result of the playouts that start with this move (Coulom 2006; Kocsis and Szepesvári 2006). It is currently the best algorithm for games such as the game of Go and the game of Hex.

The method can also be used in other games, and as the method is general and can be used without domain knowledge it is used by the best artificial general game players (Björnsson and Finnsson 2009). The principle of general game playing competitions is to give the rule of the game to the player just before they start to play. The goal is to write general programs that have more general intelligence than usual game programs that are tailored to only one game.

Theoretical Background

Monte-Carlo Tree Search is based on the theoretical analysis of multiarmed bandits. The main algorithm is UCT which means Upper Confidence Bound applied to Trees. The principle of the algorithm is to add an exploration term to the mean of an arm in order to minimize the regret of choosing an arm. The formula used to select the next arm to pull is:

mean + C * sqrt (log (t) / s)

where mean is the mean result of the arm, C is a tunable constant, t is the number of times any arm has been pulled and s is the number of times the arm has been pulled (Auer et al. 2002, Kocsis and Szepesvári 2006). The C constant has to be tuned to the problem. A great C constant favors exploration over exploitation, whereas a small C constant will more often lead to play moves that have the greatest means. Some of the best programs that combine UCT with other heuristics claim that the best constant is 0 because the heuristics are good enough to direct the search when the number of playouts is low.

Important Scientific Research and Open Questions

Monte-Carlo Tree Search is currently the best algorithm for games such as Go, Hex, Lines of Action or Amazons. It has also beaten world records in single player games such as Morpion Solitaire or SameGame. The method can be extended to other games and puzzles. It is not clear which games and which puzzles the method is well suited for, and it is the subject of active research.

A promising research is to combine the method with other methods. For example using low variance statistical estimates of moves did improve much on UCT for the game of Go (Gelly and Silver 2007), the idea here is to keep statistics on a move even if it is played anywhere in the playout and not only after the current node of the UCT tree.

Another line of research that has been effective in Lines of Action and Amazons is to replace random playouts with an evaluation function.

Other researchers are also trying to introduce knowledge in the playouts, biasing the choice of the moves to play. Such pseudo-random playouts might help the Monte-Carlo Tree Search algorithm converge faster. This is a tricky area of research since it is not yet clear which type of knowledge accelerates convergence. For example introducing knowledge so that good moves are more often played by the pseudo-random player does not necessary lead to a better overall player. However effective use of relevant knowledge in the playouts has proven very useful both in Go and Hex.

Cross-References

→ Machine learning
→ Reinforcement learning
→ Statistical learning

References

Peter Auer, Nicolò Cesa-Bianchi, Paul Fischer: Finite-time analysis of the multiarmed bandit problem. Machine learning, 47(2):235-256, 2002.

Yngvi Björnsson and Hilmar Finnsson. CadiaPlayer: A Simulation-Based General Game Player. IEEE Transactions on Computational Intelligence and AI in Games, 1(1):4–15, 2009.

Rémi Coulom: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. Computers and Games 2006: 72-83

Gelly, S., and Silver, D. 2007. Combining online and offline knowledge in UCT. In Ghahramani, Z., ed., ICML, volume 227, 273–280. ACM.

Levente Kocsis, Csaba Szepesvári: Bandit Based Monte-Carlo Planning. ECML 2006: 282-293