Lagging Anchor Algorithm: Reinforcement Learning with Imperfect Information and its application to poker games

Bastian Christoph bc1001.rbg.informatik.tu-darmstadt.de
Knowledge Engineering Group
Department of Computer Science
Technische Universität Darmstadt, Germany

Abstract

This paper describes how currently existing techniques for reinforcement learning with perfect information such as gradient-search based methods and linear programming methods can be extended towards the usage for solving games with imperfect information. It referes to the work of Frederick A. Dahl who developed the technique in his group and showed the application towards playing poker.

1. Introduction

Since reinforcement learning is widely developed in the case of games with perfect information there exist some well known techniques for solving games of the two-player zero-sum type in game theory.

The commonly known gradient search based techniques like TD-learning which developed over are few years and are used in a lot of tools. By changing the view to games, where not all information are visible in each game state, there has to be a technique for solving these problems in a more approximatively way. The TD-learning method is used later in the "truncated sampling"-part of the algorithm-implementation for extensive-form games.

The author analyzed the common gradient-search based method and the common linear programming method from the view of the psychological oriented human learning model and the machine learning model and used some new assumptions for changing the methods to a different way, so he showed how there can be a new technique, by him called the lagging anchor algorithm to give a quite good solution to reach the game theoretic optimal solution point.

2. Lagging Anchor Algorithm

General Description: The author describes that the fact, that simple gradient search may result in oscillations around the solution points, which can be exploited to define a better algorithm. The algorithm exhibits an exponential convergence for at least some special cases of matrix games. The general idea is to have an "anchor" for each player, which is "lagging" behind the current value of the paramter states v and w, which is dampening the oscillation seen with the basic algorithm. The anchor is drown towards the corresponding parameter value, proportionally to the distance between the parameter state and the anchor. Before he comes to this point, he gives a formal definition of the basic algorithm.

Formal Definitions: At first, the author gives the expected payoff and the strategies for the Blue and Red player. The basic algorithm is given by assuming that Blue measures the gradient of the exponential payoff with respect to the parameter set and updates the parameters in that direction. Red performs a similiar update with the opposite sign. He defines the basic algorithm by the rule (1) and the initial values, thereby using a convex projection towards a non-empty and closed convex set A. The goal is to find a minimax solution point. The necessary condition is, that neither side can improve his payoff locally. This is satisfied for a special fixed point for the mapping (1).

Definition of the Algorithm: So the scene is set for the definition of the lagging anchor algorithm. The anchors for v and w are v' and w', while the anchor drawing factor is given. The algorithm is given by (2) with the intial conditions. For a special case, there's no need for convex projections. Because convex projections are well defined, the shown sequence is determined by (2). The actual implementations demand to calculate gradients and convex projections, which my be non-trivial. In the following sections, the implementation in the matrix-form and in the extensive-form is shown. This will include the convergence result, too.

3. Usage towards Matrix-Form Games

General Description: The matrix M is given by m_{ij}, where the value is the expected outcome of the player Blue choosing i and the player Red choosing j. Linear and complete parameterization of the strategies are used. The parameter sets V and W for Blue and Red are given, while the sets are compact and convex. The interior points are sets in the affine subspaces of R^n and R^m, where the components sum to 1. The interior points there are points with positive value for all the components. We refer to the game as a fully mixed solution, if the game solution is an interior point. The information sets of the game do not need to be handled explicitly. The probability of Blue taking action i is represented as $B_v(i) = v_i$ and the probability of Red taking action j is represented as $R_w(j) = w_j$. With this, Blue has the expected payoff $E(v, w) = v^T * M * w$. The differentation of these gives the gradients, so that the learning rule given before changes to the new learning rule. These changed rule of the basic algorithm conincides the gradient update, modified by Selten. The simple algorithm for calculating convex projections is given by the Proposition 2, shown in the refered papers. It suffices to describe the computation of C_v, because the sets V and W have an identical structure. For an index set I, Q_I is defined. In the bi-matrix case, selten showed that the basic algorithm fails to converge towards fully mixed solutions. By stating that the gradient update is orthogonal to the error vector, the used Proposition 3 refines the result. The process keeps a constant distance to the solution point, when α approaches zero. For the notational convenience x^T is defined.

Lagging Anchor Algorithm: We are now moving the the lagging anchor algorithm for matrix games. The learning rule of (2) can be implemented towards (4), using the calculation procedure for C_v and C_w, which are given in the proposition (2). The proposition (4) therefore gives the convergence result. For general matrix games, the convergence has not been proven. But the algorithm has worked well on a large sample of test games. Therefore, random matrices were tested with dimensions up to 100 and the entries where uniformly distributed in the interval $[-1, 1]$. There, the exponential convergence has been

observed. The anticipatory learning rule of selten is given in the new notation. With the experiments, it giives a exponential convergence for large random matrices. In general, the idea of the algorithm is to produce approximate solutions to large games, using non-linear and incomplete parameterization. Rather than compete with existing efficient algorithms for linear programming, which are the standard for solving the matrix games. Here, the lagging anchor algorithm appears to have similarities with the interior point methods for the linear programming. It searches the interior of the set of admissible solution points. And it works to solve the primal and the dual problem simultaneously. There, the primal problem is the optimization problem of Blue, while the dual problem is the one of Red. Because the interior point methods consist of finite sequences of projections, rather than incremental search that converge to a solution, the similarities end at that point. There, the main fact is that the interior point methods can not be extended beyond the context of matrix games.

4. Usage towards Extensive-Form Games

General Description: In (Dahl, 2001) the presented agent design allows non-linear and incomplete stategy representations in extensive form games. So the algorithm can be adjusted to that context. At last the stability result is given. The learning rule in (2) contains non-trivial operation of evaluating the gradient of the players. The author explained, how this is handled and presented the algorithm in pseudocode. The learning rule contains convex projections, which are needed if the domains of the players' parameters are restricted. Because the different parameter sets require different methods, there is no ability to specify a general algorithm. So the general optimization techniques like the solution of the Kuhn-Tucker conditions should be useful.

Main changes: In general, the process of estimating the gradient is split into two. In the first process, the gradient of the expected payoff is estimated with respect to the action probabilities (dE/dB). In the second process, the gradient of the action probabilities is calculated with respect to the parameters (dB/dv). When combining the gradient using the chain rule of differentiation, we get the estimated gradient of the agent's expected payoff with respect to the parameters (dE/dv). At last the agent's parameters are updated in the gradient direction, multiplied with the step size α.

4.1 Training Patterns

For explaining the algorithm, the notion of "training patterns" is used, which is commonly used in the context of neural networks. There, the training pattern is a combination of the input and the feedback. The application of the training pattern is implemented as the gradient search step (length proportional to α) in the parameter space V towards a minimization. The parameter state of B is modified to make the output closer to the feedback of the given input. The second part of the gradient calculation (dB/dv) is integrated with the parameter update. There we are assuming that the derivative of the payoff with respect to the out of B on input is estimated to be d. The training with the pattern implements both, the calculation of the gradients and the parameter update in the gradients direction.

4.2 Sampling the consequences of the alternative actions

General Description: In general, the problem of calculating the gradients of the player's payoffs analytically is far too complex to be considered. So we want to estimate them based on the outcomes of the sample games, where the Blue and Red agents are playing. This leads us the denotion "reinforcement learning". For simplificationn of the presentation, we describe the estimation of the Blue's gradient only. The method, which is developed therefore relates to the "what-if" analysis. At first the sample game is completed, then the course of the game is analyzed. For each visited decision node, the hypothetical consequences of the different available actions are estimated. So later, the convertion of the estimates into the training patterns is shown. The estimation if the consequence of action in the descision node is equivalent to the estimation of the expected payoff for the resulting node. So is seems, that there is an contradiction. It is mentioned, that the nodes of the game tree may not have unique values. Now it seems like we are estimating this. Given current strategies of Blue and Red, we are attemping to estimate expected payoff for nodes, so we do not. Given the players Blue and Red, there's no problem on defining the value of a node as its average payoff. The simplest way of estimating the expected payoff for a node is by doing sampling. There one simulates one or more games from the nodes in question, and then takes the sample average as the estimate. So this is clearly unbiased. Consider the case, where only one additional game is completed for each decision as "canonical form" of the algorithm. We can take the outcome of the original sample game as estimated consequence of action made in game, only actions deviating from course of game need to be sampled. In the figure the author shows an illustration of the algorithm. The circular and square nodes represent the decision nodes for the Blue and Red player, while the triangle nodes represent terminal states where the game rules define the payoff of the Blue player. The vertical path represents the course of the game. In the top node, Blue has two deviating actions, leading to game states which are labelled with "X". To estimate the utility, an additonal sample game is completed from each of these states, indicated in the figure. In the central node of the figure, the Blue player has two alternatives to the action, taken in the original game, leading to the states which are labelled "Y". The numbers which are attached to the arcs in the figure are giving the estimated payoffs resulting form the action taken by the Blue player in the preceding nodes. These numbers are turned into the training patterns later.

4.3 Truncated sampling using the TD-learning

General Description: The canonical form of the algorithm requires one sample game to be completed for each decision that player Blue can make throughtout the original sample game. The number of additional games that must be completed to estimate the gradient from one sample game is less than the branching factor multiplied by the tree depth. The sampling of the results from alternative decisions slows down the algorithm by a linear factor of thre tree depth, which is no desaster from the viewpoint of the computational complexity. The computational burden from the canonical form of the algorithm can be very significant in practice. It introdues the considerable noise in the estimed gradients.

Truncated Sampling: Therefore, the "truncated sampling" is introduced by the author. The general idea is to use an external function for the estimation of the node values. The purpose is only to estimate the consequences of the alteernative decisions, not to take

partin the game playing. In the example figure 4 in (Dahl, 2002), this means that nodes which are labelled "X" and "Y" are evaluated by a function indead of being played out. Given the players Blue and Red, the course of the game is a Markov process, where the states are the game-tree nodes. The fact that the players have just imperfect information and that they act on the information sets is not a problem, as the pair of mixed strategies for Blue and Red introduces the Markov Process on the set of the nodes. The standard method for estimating the expected outcome of the Markov Process is the TD-learning, which is used here. After a sample game has been completed, the course of the game (here as sequence of visited nodes) is used to update the evaluator of the node through TD-learning. This essentially means, that the evaluation of the given node is tuned towards the weighted average of the actual payoff and the function owns the evaluations of the intermediate nodes. The use of truncated sampling throught a function tuned by TD-learning will introduce bias in the gradient estimation, unless the function produces exact evaluations of the nodes. At the beginning of the training session, the node evaluator function will normally be initialized randomly, in which case the gradient estimation will be very poor until the function starts giving meaningful evaluations. Only the experiments can determine wether the gain in speed and the reduced randomness of the truncated sampling outweight the disadvantage of the biased gradient estimation.

5. Conclusion

At the end we can reach the point, that there are some ways of changing the traditionally known methods for two-play zero-sum games with perfect information by extending the model with the result of psychological analysis. We have seen that the gradient-search based method and a linear programming method can be extended for the usage in the lagging anchor algorithm. The application to poker showed, that for this case the developed algorithm is quite useful and gave a good approximation. Looking forward to futher questions this could be a quite good example for looking at the common techniques in an extending way.

References

Dahl, F. A. (2001). A reinforcement learning algorithm applied to simplified two-player texas hold'em poker. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pp. 85–96, London, UK. Springer-Verlag.

Dahl, F. A. (2002). The lagging anchor algorithm: Reinforcement learning in two-player zero-sum games with imperfect information. *Mach. Learn.*, *49*(1), 5–37.