Closed-form Solutions to a Subclass of Continuous Stochastic Games via Symbolic Dynamic Programming

ABSTRACT

Zero-sum stochastic games provide a formalism to study competitive sequential interactions between two agents with diametrically opposing goals and evolving state. A solution to such games in the case of a discrete environment, or state, was presented by Littman [9]. The continuous state case, however, remains unsolved. In many instances this requires nonlinear parameterised optimisation, a problem for which closed-form solutions are generally unavailable. We characterise a subclass of continuous stochastic games and present a novel symbolic dynamic programming method which enables these problems to be reduced to a parameterised linear program, for which an exact closed-form solution exists. The method is used to compute exact solutions to a variety of continuous state zero-sum stochastic games.

1. INTRODUCTION

Modelling sequential competitive interactions between agents has important applications within economics and decision making. Stochastic games [13] provide framework a to model sequential interactions between multiple non-cooperative agents. Zero-sum stochastic games stipulate that the participating agents have diametrically opposing goals. Littman [9] presented a multiagent reinforcement learning solution to discrete state zero-sum stochastic games. Closed form solutions for the continuous state case, however, remain unknown. Continuous state zero-sum stochastic games provide a convenient framework with which to model many important financial and economic domains, such as option valuation on derivative markets.

In this paper we present a novel technique to calculate a closed-form solution to a subclass of continuous state zero-sum stochastic games. We show that by using symbolic dynamic programming we can calculate closed-form solutions for arbitrary horizons.

We begin by presenting Markov Decision Processes (MDPs) and value iteration [1], a commonly used dynamic programming solution. We then describe both discrete and continuous stochastic games, game theoretic generalisations of the MDP framework. Following this we show how Symbolic Dynamic Programming (SDP) [6] can be used to calculate exact solutions to a particular subclass of continuous zero-sum stochastic games. Finally, we present an empirical evaluation of our novel technique on an American option quitting game.

Appears in: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Lomuscio, Scerri, Bazzan, Huhns (eds.), May, 5–9, 2014, Paris, France.*

2. MARKOV DECISION PROCESSES

A Markov Decision Process (MDP) [7] is defined by the tuple $\langle S, A, T, R, h, \gamma \rangle$. S and A specify a finite set of states and actions, respectively. T is a transition function $T : S \times A \to S$ which defines the effect of an action on the state. R is the reward function $R : S \times A \to \mathbb{R}$ which encodes the preferences of the agent. The horizon h represents the number of decision steps until termination and the discount factor γ is used to discount future rewards. In general, an agent's objective is to find a policy, $\pi : S \to A$, which maximises the expected sum of discounted rewards over horizon h.

Value iteration [1] is a general dynamic programming algorithm used solve MDPs. For each horizon h, two functions form the basis of the algorithm: $V^h(s)$, the value of state s, and $Q^h(s, a)$, the value of taking action a in state s. The two functions satisfy the following recursive relationship:

$$Q^h(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^{h-1}(s') \quad (1)$$

$$V^h(s) = \max_{a \in A} Q^h(s, a). \quad (2)$$

The algorithm is executed by first initialising V^0 to $R(s, a)$. Then for each h, the value function for $V^h(s)$ is calculated from $V^{h-1}(s)$ until the intended h-stage-to-go value function is computed. Value iteration converges linearly in the number of iterations to the true values of $Q(s, a)$ and $V(s)$ [4].

MDPs can be used to model multiagent systems under the assumption that other agents are part of the environment and have fixed behaviour. As a result, they ignore the difference between responsive agents and a passive environment [8]. In the next section we present a game theoretic framework which generalises MDPs to situations with two or more agents.

3. DISCRETE STOCHASTIC GAMES

Discrete state stochastic games (DSGs) are formally defined by the tuple $\langle S, A_{1\ldots n}, T, R_{1\ldots n} \rangle$. S is a set of discrete states and A_i is the action set available to agent i. A represents the joint action space $A_1 \times \ldots \times A_n$. T is a transition function $T : S \times A \to \Delta$ where Δ is the set of probability distributions over the state space S. The reward function $R_i : S \times A \to \mathbb{R}$

encodes the preferences agent i. In a stochastic game a policy $\pi : S \to \sigma(A)$ yields probability distributions over the agent's actions for each state in S. As a result, the optimal policy in a stochastic game may be stochastic. The goal of each agent in a stochastic game is to maximise its expected discounted future rewards.

Zero-sum discrete stochastic games impose a condition on the reward structure of the game, whereby the goals of each agent are diametrically opposed to one another. Under this restriction, the game can be expressed using a single reward function. Each agent attempts to maximise its expected discounted future rewards in the minimax sense. Zero-sum stochastic games can be solved using a technique analogous to value iteration in MDPs [9]. Under this view the value of a state s in a stochastic game is

$$V^h(s) = \max_{\pi_{a_1} \in \sigma(A_1)} \min_{\pi_{a_2} \in \sigma(A_2)} \sum_{a_1 \in A_1} \sum_{a_2 \in A_2} Q^h(s, a_1, a_2) \cdot \pi_{a_1} \cdot \pi_{a_2}, \tag{3}$$

where a_i is an action from A_i and π_{a_i} is a policy defined over the probability distributions of A_i. It is well known that Equation 3 can be further simplified to

$$V^h(s) = \max_{\pi_{a_1} \in \sigma(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q^h(s, a_1, a_2) \cdot \pi_{a_1}. \tag{4}$$

The value of taking action a_1 against action a_2 in state s is given by

$$Q^h(s, a_1, a_2) = R(s, a_1, a_2) + \gamma \sum_{s' \in S} T(s, a_1, a_2, s') V^{h-1}(s'). \tag{5}$$

The value function for the opponent can be calculated by applying symmetric reasoning and the Minimax theorem [10].

3.1 Solution Techniques

The value function shown in Equation 4 can be solved for each $s \in S$ by reformulating it as the following linear optimisation problem:

$$\begin{aligned} \text{maximise} \quad & V^h(s) \\ \text{subject to} \quad & \pi_{a_1} \geq 0 \qquad \forall a_1 \in A_1 \\ & \sum_{a_1 \in A_1} \pi_{a_1} = 1 \\ & V^h(s) \leq \sum_{a_1 \in A_1} Q^h(s, a_1, a_2) \cdot \pi_{a_1} \qquad \forall a_2 \in A_2 \end{aligned}$$

We note that two transformations have been applied in the reformulation process. Firstly, we define $V(s)$ to be $\min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q^h(s, a_1, a_2) \cdot \pi_{a_1}$, the value of the inner minimisation. Secondly, we note that the minimum of a set is less than or equal to the minimum of all elements in the set, that is $V(s) = \min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q^h(s, a_1, a_2) \cdot \pi_{a_1} \equiv V(s) \leq \sum_{a_1 \in A_1} Q^h(s, a_1, a_2) \cdot \pi_{a_1} \quad \forall a_2 \in A_2$.

4. CONTINUOUS STOCHASTIC GAMES

Continuous state stochastic games (CSGs) are formally defined by the tuple
$\langle \vec{x}, A_{1\ldots n}, T, R_{1\ldots n} \rangle$. States are represented by a vectors of continuous variables, $\vec{x} = (x_1, \ldots, x_m)$, where $x_i \in \mathbb{R}$. The other components of the tuple are as previously defined for DSGs in Section 3.

Zero-sum continuous stochastic games can be defined by the following recursive functions

$$V^h(\vec{x}) = \max_{\pi_{a_1} \in \sigma(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q^h(\vec{x}, a_1, a_2) \cdot \pi_{a_1}. \tag{6}$$

$$Q^h(\vec{x}, a_1, a_2) = R(\vec{x}, a_1, a_2) + \gamma \int T(\vec{x}, a_1, a_2, \vec{x}') V^{h-1}(\vec{x}') d\vec{x}'. \tag{7}$$

4.1 Solution Techniques

Equation 6 can be solved using a technique analogous to that presented in Section 3.1. Namely, the CSG formulation of the value function can be re-written as the following optimisation problem:

$$\begin{aligned} \text{maximise} \quad & V(\vec{x}) && && \text{(8a)} \\ \text{subject to} \quad & \pi_{a_1} \geq 0 \qquad && \forall a_1 \in A_1 && \text{(8b)} \\ & \sum_{a_1 \in A_1} \pi_{a_1} = 1 && && \text{(8c)} \\ & V(\vec{x}) \leq \sum_{a_1 \in A_1} Q(\vec{x}, a_1, a_2) \cdot \pi_{a_1} \qquad && \forall a_2 \in A_2 && \text{(8d)} \end{aligned}$$

This optimisation problem cannot be easily solved using existing techniques due to two factors: there are infinitely many states in $\vec{x}$ and constraint 8d is bilinear in $\vec{x}$ and π_{a_1}. Solving bilinear programs optimally is known to be NP-hard [2, 11].

We note that by restricting the reward function $R(\vec{x}, a_1, a_2)$ to be piecewise constant, the previously bilinear constraint 8d is made piecewise linear. This resulting subclass of zero-sum continuous stochastic games can then be solved optimally a given horizon h using symbolic dynamic programming techniques [14].

In the next section we present an overview of symbolic dynamic programming, its implementation, and show how it can be used to calculate exact solutions to this subclass of continuous zero-sum stochastic games.

5. SYMBOLIC DYNAMIC PROGRAMMING

Symbolic dynamic programming (SDP) [6] is the process of performing dynamic programming via symbolic manipulation. In the following sections we present a brief overview of SDP and the data structures used to perform operations.

5.1 Case Representation

Symbolic dynamic programming assumes that all continuous symbolic functions can be represented in case form [6].

$$f = \begin{cases} \phi_1 : & f_1 \\ \vdots & \vdots \\ \phi_k : & f_k \end{cases}$$

ϕ_i are logical formulae defined over the state $\vec{x}$ and can include arbitrary logical combinations of boolean variables and linear inequalities over continuous variables. Each ϕ_i

will be disjoint from the other ϕ_j $(j \neq i)$ and may not exhaustively cover the state space. Hence, f may only be a partial function. The f_i may be either linear or quadratic in the continuous parameters. Operations on f_i preserve the continuous nature of the function f.

5.2 Case Operations

Unary operations on a case statement f, such as scalar multiplication $c \cdot f$ where $c \in \mathbb{R}$ or negation $-f$, are applied to each f_i $(1 \leq i \leq k)$.

Binary operations on two case statements are executed in two stages. Firstly, the cross-product of the logical partitions of each case statement is taken, producing paired partitions. Finally, the binary operation is applied to the resulting paired partitions. The "cross-sum" $\oplus$ operation can be performed on two cases in the following manner:

$$\begin{Bmatrix} \phi_1 : f_1 \\ \phi_2 : f_2 \end{Bmatrix} , \begin{Bmatrix} \psi_1 : g_1 \\ \psi_2 : g_2 \end{Bmatrix} = \begin{cases} \psi_1 \wedge \psi_1 : & f_1 + g_1 \\ \psi_1 \wedge \psi_2 : & f_1 + g_2 \\ \psi_2 \wedge \psi_1 : & f_2 + g_1 \\ \psi_2 \wedge \psi_2 : & f_2 + g_2 \end{cases}$$

"cross-subtraction" $\ominus$ and "cross-multiplication" $\otimes$ are defined in a similar manner but with the addition operator replaced by the subtraction and multiplication operators, respectively. Some partitions resulting from case operators may be inconsistent and are thus removed.

Maximisation over cases, known as casemax, is defined as:

$$\text{casemax}\left(\begin{Bmatrix} \phi_1 : f_1 \\ \phi_2 : f_2 \end{Bmatrix} , \begin{Bmatrix} \psi_1 : g_1 \\ \psi_2 : g_2 \end{Bmatrix} \right) = \begin{cases} \phi_1 \wedge \psi_1 \wedge f_1 > g_1 : f_1 \\ \phi_1 \wedge \psi_1 \wedge f_1 \leq g_1 : g_1 \\ \phi_1 \wedge \psi_2 \wedge f_1 > g_2 : f_1 \\ \phi_1 \wedge \psi_2 \wedge f_1 \leq g_2 : g_2 \\ \vdots \qquad\qquad\qquad\quad \vdots \end{cases}$$

casemax preserves the linearity of its inputs. If the f_i or g_i are quadratic then the expressions $f_i > g_i$ or $f_i \leq g_i$ will be at most univariate quadratic and any such constraint can be linearised into a combination of at most two linear inequalities by completing the square.

5.3 Extended Algebraic Decision Diagrams for Case Statements

Case statements and their operations are implemented using Extended Algebraic Decision Diagrams (XADDs) [12]. XADDs provide a compact data structure with which to maintain compact forms of $Q(\vec{x}, a_1, a_2)$ and $V(\vec{x})$. XADDs also permit the use of linear constraint feasibility checkers to prune unreachable paths in the XADD.

5.4 SDP Solution Continuous Stochastic Games

Symbolic dynamic programming can be used to find a closed-form solution to a continuous stochastic game over an arbitrary horizon.

Algorithm 1: `VI`(CSG, H) $\longrightarrow (V^h)$

```
begin
    V^0 := 0, h := 0
    while h < H do
        h := h + 1
        Q^h := ∫ V^{h-1} ⊗ T(x', x, a1, a2) d_{x'}
        Q^h := R(x, a1, a2) ⊕ (γ ⊗ Q^h)
        if V^h = V^{h-1} then
            break  // Terminate if early convergence
    return (V^h)
end
```

Algorithm 2: `Regress`$(V, a_1, a_2) \longrightarrow Q$

```
begin
    Q = Prime(V)  // All x_i → x_i'
    // Continuous regression marginal integration
    for all x_j' in Q do
        Q := ∫ Q ⊗ P(x_j'|x, a1, a2) d_{x_j'}
    return R(x, a1, a2) ⊕ (γ ⊗ Q)
end
```

6. EMPIRICAL RESULTS

We evaluated the performance of our novel solution technique on an American option quitting game. The domain and results are presented below.

6.0.1 American Option Quitting Game

American options are financial instruments which allow an investor to bet on the outcome of a yes/no proposition. The proposition typically relates to whether the price of a particular asset that underlies the option will rise above or fall below a specified amount, known as the strike price. When the option reaches maturity the investor receives a fixed pay-off if their bet was correct and nothing otherwise. An American option can be exercised by an investor at any time during its lifetime.

We analyse the valuation of an American option as an extensive form zero-sum game between an investor and the issuer of the option. The problem has two states, one continuous variable $v \in \mathbb{R}$ for the market value of the option and one discrete variable $i \in \mathbb{R}$ for the investor's inventory of options.

At each time step the investor has three binary actions, $b \in \{true, false\}$ to buy an option from the issuer, $s \in \{true, false\}$ to sell an option, and $h \in \{true, false\}$ to hold the option.

The issuer has two binary actions, $n \in \{true, false\}$ to offer a narrow bid-ask spread for the option, and $w \in \{true, false\}$ to offer a wide spread.

The joint actions of the investor and issuer, a_{inv} and a_{iss}, are assumed to have an impact on the market value of the option. For simplicity we assume that the value may increase or decrease by fixed step sizes given by u and $-u$, respectively.

$$P(v'|v, i, a_{\text{inv}}, a_{\text{iss}}) = \delta \left[v' - \begin{cases} (c) : & v \\ (e) \wedge (i > 0) : & v + u \\ (b) \wedge (s) : & v - u \end{cases} \right]$$

The transition function for inventory i is

$$P(i'|v, i, a_{\text{inv}}, a_{\text{iss}}) = \delta \left[i' - \begin{cases} (e): & 0 \\ (b) \wedge (s): & i+1 \\ otherwise: & i \end{cases} \right]$$

The Dirac function $\delta[\cdot]$ensures that the transitions are valid conditional probability functions that integrate to 1.

The reward obtained by the investor at each time step is given by,

$$R = \begin{cases} (e) \wedge (v > \text{strike-price}) \wedge (i > 0): & 1 \\ (e) \wedge (v < \text{strike-price}) \wedge (i > 0): & -1 \\ otherwise: & 0 \end{cases}$$

7. RELATED WORK

Multi-agent extensions to the MDP framework assume that agents have a common reward [5, 3] In the next section we introduce stochastic games, a multi-agent generalisation of the MDP framework which allows each agent to have its own separate reward function and choose actions independently of one another.

8. DISCUSSION

⋮

8.1 Future Work

⋮

9. REFERENCES

[1] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[2] K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Comput. Optim. Appl.*, 2(3):207–227, Nov. 1993.

[3] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[4] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[5] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI, pages 478–485, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[6] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order mdps. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, volume 1 of *IJCAI*, pages 690 – 697, Seattle, 2001.

[7] R. A. Howard. *Dynamic Programming and Markov Processes*. The M.I.T. Press, 1960.

[8] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML, pages 242–250, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[9] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning Machine Learning*, ICML, pages 157–163, San Francisco, California, USA, 1994. Morgan Kaufmann Publishers Inc.

[10] J. Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[11] M. Petrik and S. Zilberstein. Robust approximate bilinear programming for value function approximation. *Journal of Machine Learning Research*, 12:3027–3063, 2011.

[12] S. Sanner, K. Delgado, and L. Nunes de Barros. Symbolic dynamic programming for discrete and continuous state mdps. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI-11)*, UAI, pages 1–10, Barcelona, Spain, 2011.

[13] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.

[14] Z. Zamani and S. Sanner. Symbolic dynamic programming for continuous state and action mdps. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI-12)*, AAAI, pages 1–7, Toronto, Canada, 2012.