

Lecture 1: More on stochastic games

Lecturer: Peter Bro Miltersen

Scribe: Janne Falden Andersen

1 Recap

Shapley's stochastic games:

- N - number of positions
- m_k - number of actions for I in position k
- n_k - number of actions for II in position k
- a_{ij}^k - reward to I if I chooses i and II chooses j in position k
- p_{ij}^{kl} - probability of going to position l when players choose i, j in position k , where $\sum_l p_{ij}^{kl} = 1$

For the case of discounted payoff games, we let λ denote the discount factor.

We have two models. One with discounted payoffs and one with undiscounted payoffs. There are as following:

Discounted payoff model:

If actual reward sequence is $\alpha_0, \alpha_1, \dots$, the payoff to I is

$$\lambda \sum_{t=0}^{\infty} (1 - \lambda)^t \alpha_t$$

Where λ is a very small number and the sum convert so we have a finite number.

Undiscounted payoff model:

If actual reward sequence is $\alpha_0, \alpha_1, \dots$, the payoff to I is

$$\liminf_{t \rightarrow \infty} \frac{\alpha_0 + \alpha_1 + \dots + \alpha_{t-1}}{t}$$

The "inf" makes it welldefined. When we use randomized strategies, we consider the expectation of this number, i.e.,

$$E[\liminf_{t \rightarrow \infty} \frac{\alpha_0 + \alpha_1 + \dots + \alpha_{t-1}}{t}].$$

We have also the following notation::

- S_1 - full strategy space of player I (including mixed strategies)
- S_2 - full strategy space of player II (including mixed strategies)

- \tilde{S}_1 - behavior strategies for I
- \tilde{S}_2 - behavior strategies for II
- $u_1^k(x, y)$ = expected payoff to I when x is played against y in discounted payoff model starting in position k .

2 Shapley's maximin theorem

As usual, the central theorem for this class of zero-sum game is a maximin theorem.

Theorem 1 (Shapley's maximin theorem)

$$\begin{aligned}
\forall k : \quad & \max_{x \in \tilde{S}_1} \min_{y \in \tilde{S}_2} u_1^k(x, y) \\
& = \min_{y \in \tilde{S}_2} \max_{x \in \tilde{S}_1} u_1^k(x, y) \\
& := \text{val}(k) \\
\exists x^* \in \tilde{S}_1 : & \forall k : \min_{y \in \tilde{S}_2} u_1^k(x^*, y) = \text{val}(k)
\end{aligned}$$

We call such an x^* a universal behavior maximin.

The proof occupies the rest of this section. To do the proof, we introduce a central concept for stochastic games

Definition 2 Given a stochastic game, its value iteration operator $T: \mathbb{R}^N \rightarrow \mathbb{R}^N$ is defined as

$$(T(\alpha_0, \alpha_2, \dots, \alpha_N))_k = \text{val}[A^{(k)}(\alpha)]$$

where $A^{(k)}(\alpha) = (\tilde{a}_{ij}^k)_{i=1, j=1}^{m_k, n_k}$ and $\tilde{a}_{ij}^k = \lambda a_{ij}^k + (1 - \lambda) \sum_{l=1}^N p_{ij}^{kl} \alpha_l$.

Informally, value iteration is the following process: We pretend that we already know the values (i.e., maximin and minimax expected payoffs) and then we recompute them, using the intuition that after we have played one move and find ourselves in a new position, we expect that the rewards we will accumulate from now on is given by the value of that position (adjusted by discounting). We can formally justify that this intuition is sound by using the Banach fixed point theorem (BFPT) to show that the iteration has a unique fixed point and that this is the vector of maximin/minimax values.

The following auxillary lemma relates the values of two matrix games, given by matrices A and B of the same dimensions.

Lemma 3 $\|\text{val}(A) - \text{val}(B)\| \leq \max_{ij} \|a_{ij} - b_{ij}\|$

Proof Let $\delta = \max_{ij} \|a_{ij} - b_{ij}\|$. We prove that $\text{val}(A) \geq \text{val}(B) - \delta$. Indeed, suppose that Player 1 plays the maximin strategy for matrix B in the game defined by A . For any play of the game, whatever payoff he obtains in B , he will obtain that payoff within δ in A . In particular, he is guaranteed at least $\text{val}(B) - \delta$, so the lower value of A and hence the value is at least $\text{val}(B) - \delta$. The proofs that $\text{val}(B) \geq \text{val}(A) - \delta$, $\text{val}(A) \leq \text{val}(B) + \delta$, and $\text{val}(B) \leq \text{val}(A) + \delta$ are symmetric. \square

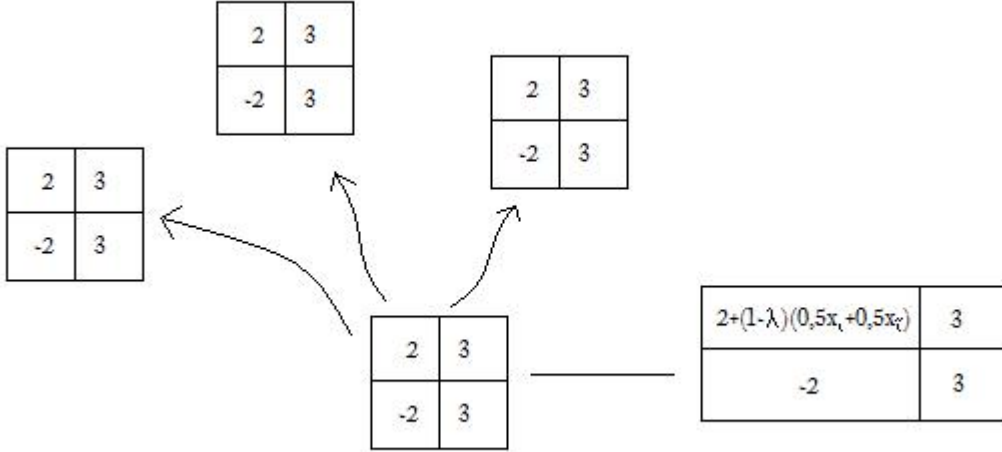


Figure 1: Illustrating value iteration

Lemma 4 T is a contraction (wrt. $\|\cdot\|_\infty$)

Proof Take $\alpha, \beta \in \mathbb{R}^N$, $\alpha \neq \beta$. Then

$$\begin{aligned}
 \|T(\alpha) - T(\beta)\|_\infty &= \max_k |\text{val}[A^{(k)}(\alpha)] - \text{val}[A^{(k)}(\beta)]| \\
 &\leq \max_k \max_{i,j} |\lambda a_{ij}^k + (1-\lambda) \sum_{l=1}^N p_{ij}^{kl} \alpha_l - (\lambda a_{ij}^k + (1-\lambda) \sum_{l=1}^N p_{ij}^{kl} \beta_l)| \\
 &\leq \max_k \max_{i,j} (1-\lambda) \sum_{l=1}^N p_{ij}^{kl} |\alpha_l - \beta_l| \\
 &\leq (1-\lambda) \max_{i,j,l} |\alpha_l - \beta_l| \\
 &= (1-\lambda) \|\alpha - \beta\|_\infty
 \end{aligned}$$

□

By BFPT, T has a unique fixed point, $FP(T)$. Also, by the proof of the BFPT, we have $\lim_{t \rightarrow \infty} T^{(t)}(\vec{0}) = FP(T)$. By easy induction (omitted; it is essentially the same as the backwards induction procedure for perfect information games), we can prove the intuitive fact that $T^{(t)}(\vec{0})$ is the value of a finite game G_t where the stochastic game is played for t rounds and then stopped (formally, in the expression of the payoff for G_t , we simply replace the \sum^∞ in the payoff for G with \sum^t).

We now want:

$$\max_{x \in S_1} \min_{y \in S_2} u_1^k(x, y) = \min_{y \in S_2} \max_{x \in S_1} u_1^k(x, y) = (FP(T))_k$$

but we shall first prove something slightly weaker, namely:

$$\sup_{x \in S_1} \inf_{y \in S_2} u_1^k(x, y) = \inf_{y \in S_2} \sup_{x \in S_1} u_1^k(x, y) = (FP(T))_k$$

By symmetry it is enough to prove that $\sup_{x \in S_1} \inf_{y \in S_2} u_1^k(x, y) = (FP(T))_k$ and actually even enough to only prove that $\sup_{x \in S_1} \inf_{y \in S_2} u_1^k(x, y) \geq (FP(T))_k$ since if

$$\begin{aligned} \sup \inf &\geq z, \\ \inf \sup &\leq z \text{ and} \\ \sup \inf &\leq \inf \sup \end{aligned}$$

we get that

$$z \leq \sup \inf \leq \inf \sup \leq z \Rightarrow z = \sup \inf = \inf \sup = z$$

It is enough to prove that for all $\epsilon > 0$, we have $\sup_x \inf_y u_1^k(x, y) \geq (FP(T))_k - \epsilon$. Pick sufficiently large t . Let x be a maximin strategy for player I in G_t augmented to be a strategy in G , by making arbitrary moves after time t . The guarantee to Player I when he plays this strategy in G is at least $T^{(t)}(\vec{0}) - (1 - \lambda)^t \sum_{j=1}^{\infty} \lambda M (1 - \lambda)^j = T^{(t)}(\vec{0}) - (1 - \lambda)M$. Since $(1 - \lambda)M < \frac{\epsilon}{2}$ for t large enough, we get that for t large enough, $T^{(t)}(\vec{0}) - (1 - \lambda)M > (FP(T))_k - \frac{\epsilon}{2}$.

This completes the proof that the sup inf is equal to the inf sup. In particular, we now have a well defined value vector and we know that it is the unique fixed point of T .

Finally, we should find a behavior strategy x^* achieving the sup. We let $x^* :=$ vector of maximin strategies of $A^{(k)}(FP(T))$ and show that this has the right property.

Due to time constraints, we only sketched the proof of correctness. We look at the family finite games (one for each value of t) where we play for t rounds and then stop with final reward $(FP(T))_j$ rewarded (where j is the current position). We first prove that x^* is maximin in *all* these games, and then we argue, similarly to the argument above that when t is large enough, we lose arbitrarily little if we play optimally in the finite game instead of the infinite game. Thus, x^* is also maximin in the infinite game: Since we lose less than ϵ for all $\epsilon > 0$, we actually lose 0.

We now have the existence of x^* achieving the sup inf value. In particular, the ‘‘sup inf’’ is really a ‘‘max inf’’, with the max begin the same if restricted to behavior strategies. To complete the proof, we just need to show that the ‘‘max inf’’ is really a ‘‘max min’’. To do this, we consider Player I’s strategy frozen and consider the resulting 1-player game for Player II. By negating rewards, we can consider him a maximizing player. But we just saw that even in a 2-player game, the maximizing player’s supremum guarantee can be achieved. So, for fixed strategy of Player 1, the inf is a min, and in particular, the max inf is a max min, and we have completed the proof.

3 Algorithms for solving discounted stochastic games

We now turn to the algorithmic problem for discounted stochastic games: Given game and λ , compute universal maximin strategy and vector of values. Annoyingly, values and maximin behavior probabilities might be irrational, even when all numbers of the input, a_{ij}^{kl} , p_{ij}^{kl} , λ are rational (See example in Ferguson, chapter 6). This means that we will in general will not go for an algorithm that computes the values and strategies exactly. There are, however, special cases that can be solved exactly.

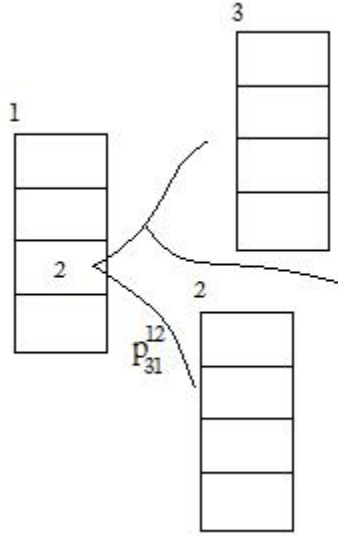


Figure 2: One player game - Markov decision process

3.1 The 1-player case

An interesting special case of a stochastic game is the *1-player case*. We say that the game is a 1-player game (with Player I being the only player) if in every position, Player II only has one possible action. That is, all matrices associated with the positions are simply column vectors, as illustrated in Figure 2. A 1-player stochastic game is also called a *Markov decision process*. We shall see that we can solve such games in polynomial time using linear programming. As a byproduct, we get a proof that when all parameters of the problem are rational numbers, so are values and maximin behavior probabilities.

To compute the value vector using linear programming, we use the fact that the value vector (v_1, v_2, \dots, v_N) with $v_i = \text{val}(i)$ is the unique fixed point of the operator T . That is, the value vector satisfies the equations

$$v_k = \max_i \{ a_i + (1 - \lambda) \sum_{l=1}^N p_{ij}^{kl} v_l \}$$

and it is the unique vector doing so. We can easily express this property by a linear program, having v_i as variables:

$$\begin{aligned} & \text{minimize } v_1 + v_2 + \dots + v_N \\ & \forall k, i : v_k \geq a_i + (1 - \lambda) \sum_{l=1}^N p_{ij}^{kl} v_l \end{aligned}$$

The optimal solution to this linear program is the unique fixed point of T . That is, we have a polynomial time algorithm for computing the value vector. Also, if the parameters of the game are rational, this value vector is rational, by the properties of linear programming. Having found the

value iteration, we can now easily find the maximin behavior strategy x^* (for a 1-player game, we also call this an *optimal policy*). We recall from the proof of Shapley’s minmax theorem that this is $x^* :=$ vector of maximin strategies of $A^{(k)}(FP(T))$. In the present 1-player case, this is a pure strategy, as we just have to select the largest entry of each column vector $A^{(k)}(FP(T))$. That is, we in fact has a universal *positional* maximin strategy.

It is a very interesting open problem if a *strongly* polynomial time algorithm for solving 1-player stochastic games exists. There is no evidence that this should be as hard as solving linear programs in general in strongly polynomial time. On the contrary, the strategy iteration algorithm (described below for the 2-player case) is a very good candidate for a strongly polynomial time algorithm for the 1-player case.

3.2 Value iteration algorithm

The value iteration operator gives rise to a very natural algorithm, Algorithm 1, known as the value iteration algorithm, for approximating the value of a 2-player stochastic game:

Algorithm 1 Value iteration

```

1:  $\alpha = (0, 0, \dots, 0)$ 
2: while true do
3:    $\alpha := T(\alpha)$ 
4: end while

```

By the proof of the BFPT, the algorithm provides better and better approximations to the value vector. A real life implementation stops at some point and output an approximation. We shall not analyze the convergence rate for the general case of stochastic games (at least not for now) but we remark that the convergence rate is tied to the Lipschitz constant $1 - \lambda$ of the contraction T . That is, if λ is large (high inflation), then the convergence is fast, but if λ is very close to 0, the convergence can be very slow.

We can use the value iteration algorithm to also compute an *approximate* maximin strategy. By an approximate maximin strategy we mean one that guarantees the value of the game within some additive error ϵ . In the last step of the loop before we terminate, we augment the computation of T , which involves solving N matrix games, with also computing a maximin strategies of these. Glueing these together, we obtain a behavior strategy x which approximates (by continuity of linear programming) an actual maximin behavior strategy x^* and by continuity of the payoff function, it therefore approximately achieves the optimal guaranteee.

3.3 Strategy iteration

The strategy iteration algorithm, Algorithm 2, is a somewhat less obvious algorithm than value iteration. Informally, in strategy iteration, we repeatedly update a strategy x for Player I as follows. We compute a *best reply* y by Player II. We compute the expected rewards when x and y are played out, and, pretending that these are the actual values, we recompute a new strategy for Player 1. At first sight, this may seem like just a convoluted reformulation of value iteration. In fact, there is a crucial difference, as can be seen by actually making such as convoluted reformulation. Such a reformulation is given as Algorithm 3. Note that the reformulation is slightly different from the

Algorithm 2 Strategy iteration

- 1: $x :=$ arbitrary behavior strategy for Player I
 - 2: **while** true **do**
 - 3: $y :=$ minimax strategy in the 1-player game for Player II where Player I is frozen to play x .
 - 4: $\forall k : \alpha_k := u_1^k(x, y)$
 - 5: $\forall k : x_k := \text{maximin}[A^{(k)}(\alpha_k)]$
 - 6: **end while**
-

original version as the starting vector will not $(0, 0, \dots, 0)$ but from some other vector - but this does of course not change the stated property of value iteration.

Algorithm 3 Value iteration rephrased

- 1: $x :=$ arbitrary behavior strategy for Player I
 - 2: $\alpha = (0, 0, \dots, 0)$
 - 3: **while** true **do**
 - 4: $y :=$ vector of best replies in the matrix games $A^{(k)}(\alpha_k)$
 - 5: $\forall k : \alpha_k := u_1^k(x, y)$
 - 6: $\forall k : x_k := \text{maximin}[A^{(k)}(\alpha_k)]$
 - 7: **end while**
-

We see that the crucial difference between value iteration and strategy iteration is the more intelligent computation of y in strategy iteration.

To show that the values and strategies computed in strategy iteration converges to the correct maximin strategies and values, we need a lemma.

Lemma 5 *Entries of (α_k) never decrease.*

Proof We must show that the “new” x , let us call it x' , achieves at least as good a guarantee as the “old” x , for every position of the game. Let us denote this property by $x \preceq x'$. Suppose y is a behavior strategy and z is an arbitrary strategy. Denote by $y^k\{z\}$ the strategy where one plays y for the first k rounds and then switches to z , pretending that the play has just started. It is easy to see that for all strategies z and u for which $z \preceq u$, we also have $y\{z\} \preceq y\{u\}$, for arbitrary behavior strategy y . By construction of the algorithm, we have that $x = x\{x\} \preceq x'\{x\}$. Indeed, the computed maximin strategy is by definition the best possible guarantee for playing the first round in a game in which one is afterwards forced to play x . But then we also have $x'\{x\} = x'\{x\{x\}\} \preceq x'\{x'\{x\}\} = (x')^2\{x\}$ and similarly $(x')^2\{x\} \preceq (x')^3\{x\}$. By induction, we have $x \preceq (x')^k\{x\}$ for all k . As the game is discounted, the guarantee of $(x')^k\{x\}$ approaches the guarantee of x' as k approaches infinity. That is, we have $x \preceq x'$ and we are done.

The following diagram illustrates the flow of the proof.

$$\begin{aligned} & \boxed{x} \rightarrow \boxed{x} \rightarrow \boxed{x} \rightarrow \dots \rightarrow \boxed{x} \\ \leq & \boxed{x'} \rightarrow \boxed{x} \rightarrow \boxed{x} \rightarrow \dots \rightarrow \boxed{x} \\ \leq & \boxed{x'} \rightarrow \boxed{x'} \rightarrow \boxed{x} \rightarrow \dots \rightarrow \boxed{x} \\ \leq & \boxed{x'} \rightarrow \boxed{x'} \rightarrow \boxed{x'} \rightarrow \dots \rightarrow \boxed{x} \end{aligned}$$

□