Algorithmic Game Theory and Applications

Content

Lecture 1 What is game theory

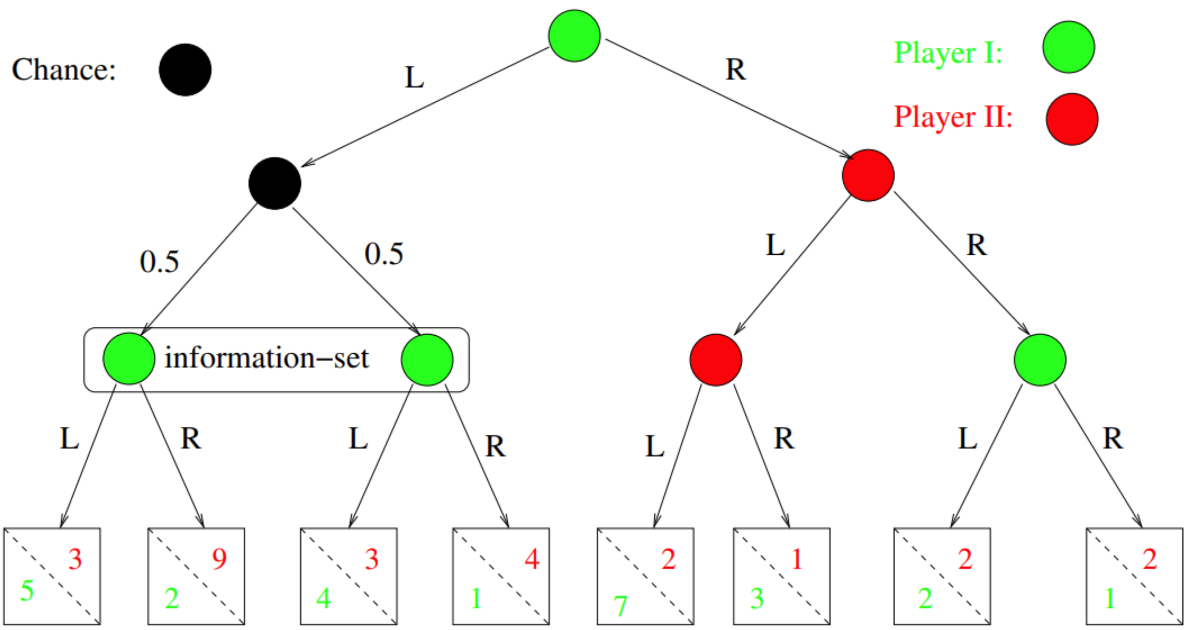
1. Dominant Strategy: No matter what other players do, it is the optimal strategy. For example in Prisoner's dilemma, defection is the dominant strategy.

2. Nash Equibria: A profile of stratefies for n players such that no player can benefit by unilaterally deviating from its strategy. E.g., in prisoner's dilemma (defect, defect) is a pure NE. In Rock-Paper-Scissors ((1/3, 1/3, 1/3), (1/3, 1/3, 1/3)) is a mixed NE.

3. Game types:

   1. Strategic game: All players make decisions simultaneously, without knowing other players' decisions
   2. Extensive game: Players make decisions sequentially with knowledge about others' previous decisions. Extensive game can be represented by a tree.

4. Chance and Information



   1. Some nodes in a tree can be chance nodes (probabilistic).

2. Not all information can be available to a player. A player employs a move at a node in a information set.
3. A game where every information set has only one node is called a game with perfect information

5. Theorem about extensive game: Any finite n-person extensive game of perfect information has an equibrium in pure strategies.

6. Mechanism design

   1. Auctions:
      1. Ascending-bid auctions (English auctions): The seller gradually raises the price. Bidders drop out and the last bidder wing the object at this final price.
      2. Descending-bid auctions (Dutch auctions): The seller lowers the price from a high initial price until some bidder accepts and pays the current price.
      3. First-price sealed-bid auctions: Bidders submit sealed bids simultaneously and the highest bidder wins and pays the value of his bid.
      4. Second-price sealed-bid auctions (Vickrey auctions): Submit simultaneously and the highest bidder wins, but pays the second highest price.

7. Applications

   1. Games in AI: modeling "rational agents" and their interactions.
   2. Games in Modeling and analysis of reactive systems: computer-aided verification.
   3. Games in Algorithms: several GT problems have a very interestion algiorithmic status.
   4. Games in Logic in CS: GT characterizations of logics, including modal and temporal logics.
   5. Games in Computational Complexity: Many computational complexity classes are definable in terms oof games.
   6. Games, the Internet and E-commerce.

Lecture 2 Mixed Strategies, Expected Payoffs, and Nash Equibrium

1. A finite strategic game consists of:

   1. A set N = {1,...,n} players
   2. For each $i \in N$, a finite set $S_i = \{1,...,m_i\}$ of (pure) strategies. $S = S_1 \times S_2 \times ... \times S_n$ be the set of possible combinations of (pure) strategies.
   3. For each $i \in N$, a payoff (utility) function: $u_i : S \mapsto \Re$, describes the payoff $u_i(s_1,...,s_n)$ to player i under each combination of strategies.
   4. The key assumption is that each player wants to maximize its own payoff.

2. Mixed (Randomized) strategies:

   1. the probability distribution over all pure strategies. For strategies $S = \{1,...,m_j\}$, $x_i(m_j)$ denotes the probability of player i taking strategy m_j. $\Sigma^j x_i(m_j) = 1$.
   2. Let X_i be the set of mixed strategies for player i. For an n-player game, $X = X_1 \times ... \times X_n$, denote the set of all possible combinations, or **profiles** of mixed strategies.

3. Expected payoffs

   1. The expected of a player i is $U_i(x) := \Sigma_{s \in S} X(s) * u_i(s)$, the weighted average of payoff for each strategy over its probability.

4. Some notations
   1. Given a mixed strategy $x = (x_1,...,x_n) \in X$, let $x_{-i} = (x_1 , ... , x_{i-1} ,\text{empty}, x_{i+1} , ... , x_n)$
   2. $(x_{-i} ; y_i)$ is the new profile where others' strategies remain the same while the i-th player change his strategy to y_i

5. Best response: the best response z_i of player i to other players' strategies, $U_i (x_{-i} ; z_i) \geq U_i (x_{-i} ; y_i)$. If every player employs best response, it is a NE. If every best response is a pure strategy, it is a pure NE.

6. Nash's theorem: Every finite n-person strategic game has a mixed Nash Equilibrium.

7. Brouwer fixed point theorem: Every continuous function $f : D \to D$ mapping a <u>compact</u> and <u>convex</u>, nonempty subset $D \subseteq \Re^m$ to iteslf has a "fixed point", i.e., there is a $x^* \in D$ such that $f(x^*) = x^*$

8. Prove (see Lecture3 page4)

9. Pareto optimal (Pareto efficient): Cannot improve any player's payoff without hurting others' payoff. A profile is $x \in X$ is **pareto efficient** if there is no other x' such that $U_i(x) \leq U_i(x')$ for all player i, and $U_k(x) < U_k(x')$ for some player k.

10. Evolution helps arrive a NE. As a result, a mixed strategy can be viewed as percentages in a population that exhibit different behaviors.

11. Symmetric game: all players can take the same actions and for all $s_1,s_2 \in S$, $u_1(s_1,s_2) = u_2(s_2,s_1)$

12. Evolutionarily Stable Strategy (ESS): a mixed strategy x_1^* is an ESS, if:
    1. x_1^* is a best response to itself, i.e., $x^* = (x_1^*, x_1^*)$ is a symmetric NE, and
    2. If $x_1' \neq x_1^*$ is another best response to x_1^*, then $U_1(x_1', x_1') < U_1(x_1^*, x_1')$
    3. Every sysmmetric game has a symmetric NE, (x_1^*, x_1^*), but not every symmetric game has a ESS.
    4. Examples:
       1. In Hawk-Dove game, (5/8, 3/8) is a NE and ESS
       2. In Rock-Paper-Scissors, (1/3, 1/3, 1/3) is a NE but not a ESS.
    5. Finding a ESS is NP-hard and coNP-hard

Lecture4 2-player zero-sum games, and the minimax Theorem

1. 2-person zero-sum games:
   1. 对任意一个profile两个player的payoff相加等于0。
   2. player1可以采用m_1个strategy， player2可以采用m_2个strategy，则 player i的payoff可以用一个$m_1 \times m_2$的矩阵来表示。

$$A_1 = \begin{bmatrix} u_1(1,1) & \dots\dots & u_1(1,m_2) \vdots & \vdots & \vdots \vdots & \vdots & \vdots u_1(m_1,1) & \dots\dots & u_1(m_1,m_2) \end{bmatrix}$$

   3. $A_2 = -A_1$. Then we assume $u(s_1, s_2)$ is given <u>as one matrix</u>, $A = A_1$.

4. Thus, a 2-player zero-sumgame can be described by a single $m_1 \times m_2$ matrix, where $a_{i,j} = u_1(i, j)$

$$A = \begin{bmatrix} a_{1,1} & \ldots\ldots & a_{1,m_2} \vdots & \vdots & \vdots\vdots & \vdots & \vdots a_{m_1,1} & \ldots\ldots & a_{m_1,m_2} \end{bmatrix}$$

5. Player 1 wants to maximize u(i, j) whereas Player 2 wants to minimize it (cause negative)

2. Notation of matrix and vector

   1. A > B: 每个值都比他大
   2. 矩阵满足乘法结合律，但是不满足交换律
   3. 矩阵转置(transpose): $(B^T)_{i,j} = B_{j,i}$

3. Matrix view of zero-sum game

   1. 每一个mixed strategy都用一个column vector表示。
   2. $x_1^T A x_2 = U_1(x) = - U_2(x)$

4. minmaximizing strategies: 在对方最大化自己payoff的基础上最大化自己的payoff, 也就是让自己的最小收入最大化

Definition: $x_1^* \in X_1$ is a **minmaximizer** for Player 1 if

$$\min_{x_2 \in X_2} (x_1^*)^T A x_2 = \max_{x_1 \in X_1} \min_{x_2 \in X_2} (x_1)^T A x_2$$

Similarly, $x_2^* \in X_2$ is a **maxminimizer** for Player 2 if

$$\max_{x_1 \in X_1} (x_1)^T A x_2^* = \min_{x_2 \in X_2} \max_{x_1 \in X_1} x_1^T A x_2$$

Note that

$$\min_{x_2 \in X_2} (x_1^*)^T A x_2 \le (x_1^*)^T A x_2^* \le \max_{x_1 \in X_1} x_1^T A x_2^*$$

5. Minimax theorem(冯诺依曼的)：对于一个2p-zs game, 存在一个唯一的值v*, 对于x* = (x_1*, x_2*) ∈ X, 满足：

   1. $((x_1^*)^T A)_j \geq v^*$, for j = 1,...,m_2
   2. $(A x_2^*)_j \leq v^*$, for j = 1,...,m_1
   3. Thus, $v^* = (x_1^*)^T A x_2^*$, and

$$max_{x_1 \in X_1} min_{x_2 \in X_2} (x_1)^T A x_2 = v^* = min_{x_2 \in X_2} max_{x_1 \in X_1} (x_1)^T A x_2$$

   4. In face, the above conditions hold when $x^* = (x_{1}^, x_{2}^)isaNE.x_{1}^ isandminmaximizerandx_{2}^$ is a maxminimizer.
   5. 也就是说，$x_{1}^保证了Player1最少获得v^的payoff。x_{2}^保证了Player2最多损失v^$
   6. $x^=(x_{1}^,x_{2}^*$)被称为**minimax profile**
   7. v^* 被称为**minimax value**
   8. Obviously, Player1的最大payoff ≤ Player2的最大损失

6. Minimax theorem的证明(see Lecture4 p11)。

7. We deal we minimax as an optimization problem

   1. **Maximize** v
   2. **Subject to constraints**:
      1. $(x_{1}^TA)*{j} \geq v for j = 1, … , m*{2}$
      2. $x_1(1)+...+x_1(m_1)=1$
      3. $x_1(j) \geq 0 for j = 1, ..., m_1$
   3. The optimal solution $(x_{1}^, v^)$ *willgivetheminimaxvalue* v^ , *andaminimaximizer* x_{1}^$ for Player1

Lecture5 Introduction to Linear Programming

1. A linear program is defined by three parts:

   1. A linear objective function
   2. An optimization criteria, maximize or minimize
   3. A set of m linear constraints or linear inequilities/equalities.
   4. K(C) 就是所有constraint的交集。K(C) not empty就是有解，就说C is feasible.

2. 可能的情况:

   1. K(C)是空的
   2. 没有上界，但是你要maximize
   3. 找到了optimal solution，这个解一定是有理数解

Lecture6 The Simplex Algorithm

1. Geometric idea of Simplex: 在feasible area随机选一个顶点，然后沿着edge换到一个能让结果变好的neighbour vertex

   ```
   While (x has a "neighbour vertex", x', with f(x') > f(x)):
      Pick such a neighbtor x'. Let x := x'
      (If the "neighbor" is at "infinity", outpout "Unbounded")
   ```

   Note: 不会stuck in local optimal，因为 K(C) is convex. On a convex region, a "local optimum" of a linear objective is always a "global optimum"

2. Slack Variables: By adding a slack variable y_i to each inequality, we get an equivalent LP with only equalities. LP in this form is called a dictionary: $$ Maximize\ c_{1}x_{1}\ +\ c_{2}x_{2}\ +\ ...\ +\ c_{n}x_{n}\ +\ d\ \ Subject\ to: \ a_{1,1}x_{1}\ +\ a_{1,2}x_{2}\ +\ ...\ +\ a_{1,n}x_{n}\ +\ y_{1}\ =\ b_{1} \ a_{2,1}x_{1}\ +\ a_{2,2}x_{2}\ +\ ...\ +\ a_{2,n}x_{n}\ +\ y_{2}\ =\ b_{2} \ ... \ a_{m,1}x_{1}\ +\ a_{m,2}x_{2}\ +\ ...\ +\ a_{m,n}x_{n}\ +\ y_{m}\ =\ b_{m} \

   x_{1},\ ...\ ,\ x_{n} \geq 0;\ y_{1},\ ...\ ,\ y_{m}\ \geq 0 $$

   1. Every equality constraint has at least one variable with coefficient 1 that doesn't appear in any other equality

2. Picking one such variable, y_i, for each equality, we obtain a set of m variables B = $\{y_1, \dots, y_m\}$ called a ***Basis***
3. Objective f(x) involves only non-Basis variables

3. Basic Feasible Solutions (BFS): Rewrite the LP as $$ Maximize\ c_{1}x_{1}\ +\ c_{2}x_{2}\ +\ ...\ +\ c_{n}x_{n}\ +\ d\ \ Subject\ to: \ x_{n+1}\ =\ b_{1}\ -\ a_{1,1}x_{1}\ -\ a_{1,2}x_{2}\ -\ ...\ -\ a_{1,n}x_{n}\ \ x_{n+2}\ =\ b_{2}\ -\ a_{2,1}x_{1}\ -\ a_{2,2}x_{2}\ -\ ...\ -\ a_{2,n}x_{n}\ \ ... \ x_{n+m}\ =\ b_{m}\ -\ a_{m,1}x_{1}\ -\ a_{m,2}x_{2}\ -\ ...\ -\ a_{m,n}x_{n}\ \

   x_{1},\ ...\ ,\ x_{n+m} \geq 0 $$

   1. This is a feasible dictionary. ($y_i = x_{n+i} = b_i$)
   2. We then have a feasible solution by letting x_i := 0 for i = 1, ..., n
   3. Then the objective value is f(0) = d
   4. This is a BFS, with basis B (different Bases B may yield the same BFS)
   5. A BFS corresponds to a vertex

4. Pivoting
   1. Current feasible dictionary is B = $\{x_{i_1}, \dots, x_{i_m}\}$, with x_{i_r} is the variable on the left of constraint r.
   2. Pivoting is to add x_j and remove x_{i_r} from basis B.
      1. Assuming C_r involves x_j, rewrite C_r as x_j = α (一个xj的表达式)
      2. 把x_j的表达式带入到其他有x_j的constraint C_l中，得到C_l'
      3. The new constraint C', have a new basis B' := (B \ $\{x_{i_r}\}$) ∪ $\{x_j\}$.
      4. Substitute α for x_j in f(x), so that f(x) again only depends on variables not in the new basis B' (which is a possible neighbor of B). However, not every such basus B' is eligible.

5. Sanity check for pivoting (eligibility)
   1. The new constraint b_i' remain ≥ 0, so we retain a "feasible dictionary", and thus B' yields a BFS.
   2. The new BFS must improve, or at least must not decrease, the value d' = f(0) of the new objective function. (all non-basic variables are set to 0 in a BFS. f(BFS) = f(0))
   3. (a) Suppose all variables in f(x) have negative coefficients. Then any increase from 0 in these variables will decrease the objective. We are then at an optimal BFS x*
      (b) Suppose a variable x_j in f(x) has coefficient c_j > 0, and coefficient of x_j in every constraint C_r is ≥ 0. Then we can increase x_j and objective to infinity, without violating the constraints. So it is unbounded.

6. Dantzig's Simplex algorithm:
   1. Check if we are at an optimal solution. If so, output the solution
   2. Check i infinity neighbor. If so, output unbounded.
   3. Otherwise, choose an eligible pivot pair of variables, and Pivot

7. Problem and solution: we can cycle back to the same basis forever, never strictly improving by pivoting. Solutions include:

1. Choose rules for pivoting. For example, Bland's rule: For all eligible pivot pairs (xi, xj), where xi is being added to the basis and xj is being removed, choose the pair such that, first, i is as small as possible, and second, j is as small as possible.
2. Choose randomly among eligible pivots. You will definitely get out.
3. Penturb the constraints slightly to make the LP "non-degenerate". (implement this using, e.g., the "lexicographic method")

8. Checking feasibility via simplex $$ Maximize\ -x_{0}\ \ Subject\ to: \ a_{1,1}x_{1}\ +\ a_{1,2}x_{2}\ +\ ...\ +\ a_{1,n}x_{n}\ -\ x_{0}\ \leq \ b_{1} \ a_{2,1}x_{1}\ +\ a_{2,2}x_{2}\ +\ ...\ +\ a_{2,n}x_{n}\ +\ x_{0}\ \leq \ b_{2} \ ... \ a_{m,1}x_{1}\ +\ a_{m,2}x_{2}\ +\ ...\ +\ a_{m,n}x_{n}\ +\ x_{0}\ \leq \ b_{m} \

   x_{0},\ ...\ ,\ x_{n} \geq 0; $$

   1. This LP is feasible: let x_0 = -min{$b_1, \dots, b_m, 0$ }, x_j = 0, for j = 1, ... , n. We can also get a feasible dictionary, and thus initial BFS, for it by adding slack variables.
   2. The original LP is feasible if and only if in an optimal solution to the new LP, x_0^* = 0

9. Complexity

   1. Each pivoting iteration can be performed in O(mn) arithmetic operations. Also, coefficients stay polynomial-sized, as long as retional coefficients are kept in reduced form (e.g., removing common factors from numerator and denominator). So, each pivot can be done in "polynomial time"
   2. How many iterations are required to reach the optimal solution? Can be exponentially many.
   3. The worst case that force exponentially many iterations (e.g., Klee-Minty(1972)). But very efficient in practive, requiring O(m) pivots on typical examples.

10. Whether exist a pivoting rule that achieves polynomially many pivots on all LPs?

    1. A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots.
    2. In every LP, is there a polynomial-length path via edges from every vertex to every other? The best known bound is $\leq m^{O(log_n)}$
    3. Khachian'79 proved LP has a polynomial time algorithm, using a completely different appraoch, The "Ellopsoid Algorithm". It is theoretically important but not practical.
    4. Karmarkar'84 gave a algorithm using the interior-point method.

Lecture7 LP Duality Theorem

1. Matrix notation of LP

   1. Primal Form $$ Maximize: c_{1}x_{1}\ +\ c_{2}x_{2}\ +\ ...\ +\ c_{n}x_{n}\ Subject\ to: \ a_{1,1}x_{1}\ +\ a_{1,2}x_{2}\ +\ ...\ +\ a_{1,n}x_{n}\ \leq \ b_{1} \ a_{2,1}x_{1}\ +\ a_{2,2}x_{2}\ +\ ...\ +\ a_{2,n}x_{n}\ \leq \ b_{2} \ ... \ a_{m,1}x_{1}\ +\ a_{m,2}x_{2}\ +\ ...\ +\ a_{m,n}x_{n}\ \leq \ b_{m} \

      x_{1},\ ...\ ,\ x_{n} \geq 0 $$

   2. By defining: $$ (m\times n)matrix\ A:\ (A)*{i,j}\ =\ a*{i.j}\ x = [x_{1},\ ...\ ,\ x_{n}]^{T}\ b = [b_{1},\ ...\ ,\ b_{m}]^{T}\ x = [c_{1},\ ...\ ,\ c_{n}]^{T}

$$WerewriteLPas:$$

Maximize: c^Tx\ Subject\ to:\ Ax\leq b\ x\geq 0 $$

2. Advesary

    1. Suppose an adversary comes along with a m-vector $y \in R^m,\ y \geq 0$ such that $c^T \leq y^T A$
    2. For any solution x, we then have:

    $$c^T x\ \leq (y^T A)x\ =\ y^T(Ax) \leq y^T b$$

    3. The adversary is then written as: (i.e. to optimize **the DUAL LP**)

    $$Minimize: b^T y\ Subject\ to:\ \ A^T y \geq c\ y \geq 0$$

    if the primal LP is:

    $$Maximize: c^T x\ Subject\ to:\ \ Ax \leq b\ x \geq 0$$

3. The LP Duality Theorem:

    1. Weak Duality: If $x^*in R^nandy*^\in R^m$ are optimal feasible solutions to the primal and dual LPs, then $c^Tx^*\leq b^Ty*^$. When x* and y* are optimal, equality holds.
    2. Strong Duality: One of the followin situations holds:
        1. Both the primal and adual LPs are feasible, and for any optimal solutions x* of the primal and y* of the dual: $$ c^Tx^*=b^Ty^* $$
        2. The primal is infeasible and the dual is unbounded
        3. The primal is unbounded and the dual is infeasible
        4. Both LPs are infeasible

4. Complementary Slackness: solutions x* and y* to the primal and dual LPs are both optimal if and only if both of the following hold:

    1. For each primal constraint, $(Ax)*{i}\leq b*{i}, $i = 1, \ldots, m, either$ (Ax^*){i}=b{i}ory_{i}^*=0$ or both
    2. For each dual constraint, $(A^Ty)*{j}\geq c*{j}, $i = 1, \ldots, m, either$ (A^Ty^*){j}=c{j}orx_{j}^*=0$ or both

5. General recipe for LP duals
   if the primal is: $$ Maximize: c^Tx\ Subject\ to:\ (Ax)*{i}\leq b*{i}\ ,\ i=1,...,d,\ (Ax)*{i}= b*{i}\ ,\ i=d+1,...,m\ x\geq 0

   $$Then the dual is:$$

   Minimize: b^Ty\ Subject\ to:\ (A^Ty)*{j}\geq c*{j}\ ,\ j=1,...,r,\ (A^Ty)*{j}= c*{j}\ ,\ j=r+1,...,n\ y\geq 0 $$

6. LP for Minimax in a zero-sum game $$ Maximize: v\ Subject\ to:\ v-(x^TA)*{j}\leq 0\ for\ j=1,...,m*{2}\ x_{1}+...+x_{m2}=1\ x\geq 0

   $$Then the dual is:$$

   Minimize: u\ Subject\ to:\ u-(Ay)*{i}\leq 0\ for\ i=1,...,m*{1}\ y_{1}+...+y_{m2}=1\ y\geq 0 $$ According to minimax Theorem, v and u are exactly the same