Fully Observable + Multiple Agents

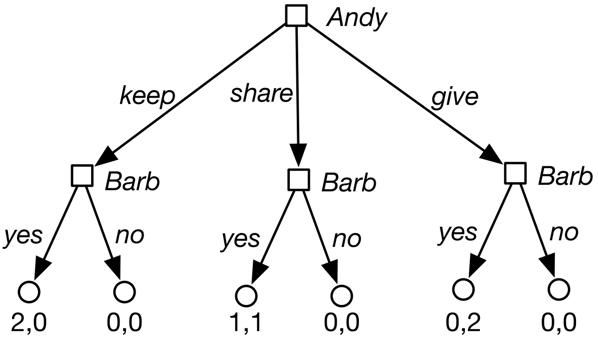
- If agents act sequentially and can observe the state before acting: Perfect Information Games.
- Can do dynamic programming or search:
  Each agent maximizes for itself.
- Multi-agent MDPs: value function for each agent.
  each agent maximizes its own value function.
- Multi-agent reinforcement learning: each agent has its own Q function.

Fully-observable Game Tree Search

procedure *GameTreeSearch*(n)
    Inputs
        n a node in a game tree
    Output
        A pair: value for each agent for node n, path that gives this value
    if n is a leaf node **then**
        return $\{i : evaluate(i, n)\}, None$
    else if n is controlled by agent i **then**
        $max := -\infty$
        for each child c of n **do**
            $score, path := GameTreeSearch(c)$
            if $score[i] > max$ **then**
                $max := score[i]$
                $res := (score, c : path)$
        return res

Extensive Form of a Game

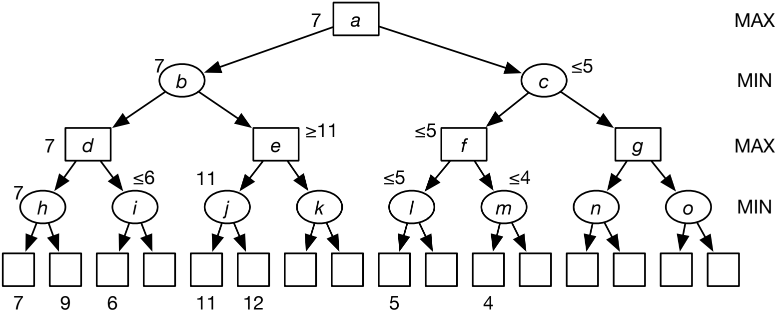
What happens with this game? Payoff is for *Andy*, *Barb*



What if the 2,0 payoff was 1.9,0.1?
Should Barb be rational / predictable?
What should Andy do if Barb threatens to not do her best action?

Pruning Dominated Strategies

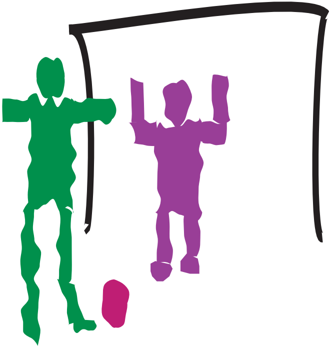
Special case: two person, competitive (zero sum) game. Utility for one agent is negative of utility of other agent. $\Longrightarrow$ minimax.



- square MAX nodes controlled by maximizing agent score
- round MIN nodes are controlled by a minimizing adversary
- leaves without a number do not need to be evaluated.

  $\longrightarrow$ α-β pruning.

Partial Observability and Competition



		goalkeeper	
		left	right
kicker	left	0.6	0.2
	right	0.3	0.9

Probability of a goal.

- Each agent decides what to do without seeing the other agent's action.
- What should each agent do?

Strategy Profiles

- Assume an n-player game in normal form
- A strategy for an agent is a probability distribution over the actions for this agent.
- A strategy profile is an assignment of a strategy to each agent.
- A strategy profile σ has a utility for each agent.
  Let $utility(\sigma, i)$ be the utility of strategy profile σ for agent i.
- If σ is a strategy profile:
  σ_i is the strategy of agent i in σ,
  σ_{-i} is the set of strategies of the other agents.
  Thus σ is $\sigma_i \sigma_{-i}$

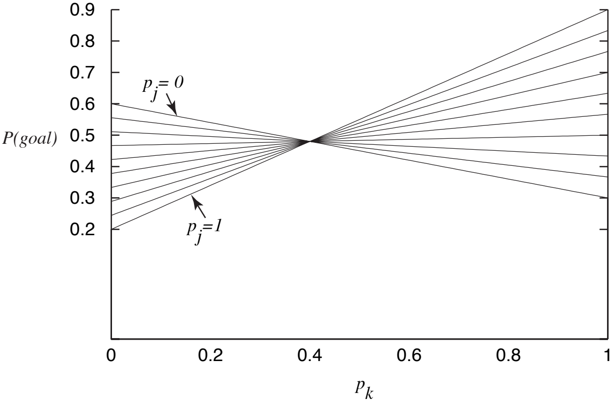
Nash Equilibria

- σ_i is a best response to σ_{-i} if for all other strategies σ'_i for agent i,

$$utility(\sigma_i \sigma_{-i}, i) \geq utility(\sigma'_i \sigma_{-i}, i).$$

- A strategy profile σ is a Nash equilibrium if for each agent i, strategy σ_i is a best response to σ_{-i}. That is, a Nash equilibrium is a strategy profile such that no agent can do better by unilaterally deviating from that profile.
- Theorem [Nash, 1950] Every finite game has at least one Nash equilibrium.

Stochastic Policies



		goalkeeper	
		left	right
kicker	left	0.6	0.2
	right	0.3	0.9

Probability of a goal.

p_k is $P(kicker = right)$
p_j is $P(goalkeeper = right)$

Multiple Equilibria

Hawk-Dove Game:

		Agent 2	
		dove	hawk
Agent 1	dove	R/2,R/2	0,R
	hawk	R,0	-D,-D

D and R are both positive with $D >> R$.

Coordination

Just because you know the Nash equilibria doesn't mean you know what to do:

		Agent 2	
		shopping	football
Agent 1	shopping	2,1	0,0
	football	0,0	1,2

Prisoner's Dilemma

Two strangers are in a game show. They each have the choice:

- Take \$100 for yourself
- Give \$1000 to the other player

This can be depicted as the playoff matrix:

		Player 2	
		take	give
Player 1	take	100,100	1100,0
	give	0,1100	1000,1000

Tragedy of the Commons

Example:

- There are 100 agents.
- There is an common environment that is shared amongst all agents. Each agent has $1/100$ of the shared environment.
- Each agent can choose to do an action that has a payoff of $+10$ but has a -100 payoff on the environment
  or do nothing with a zero payoff
- For each agent, doing the action has a payoff of
  $10 - 100/100 = 9$
- If every agent does the action the total payoff is
  $1000 - 10000 = -9000$

Computing Nash Equilibria

To compute a Nash equilibria for a game in strategic form:

- Eliminate dominated strategies
- Determine which actions will have non-zero probabilities. This is the support set.
- Determine the probability for the actions in the support set

Eliminating Dominated Strategies

		Agent 2		
		d_2	e_2	f_2
	a_1	3,5	5,1	1,2
Agent 1	b_1	1,1	2,9	6,4
	c_1	2,6	4,7	0,8

- Can prune c_1 becuase it is dominated by a_1
- Can prune f_2 becuase it is dominated by $[0.5 : d_2, 0.5 : e_2]$
- Next prune b_1 then e_2
- Single Nash equilibrium is (a_1, d_2)

Computing probabilities in randomized strategies

Given a support set:

- Why would an agent will randomize between actions $a_1 \ldots a_k$? Actions $a_1 \ldots a_k$ have the same value for that agent given the strategies for the other agents.
- This forms a set of simultaneous equations where variables are probabilities of the actions
- A solution with all probabilities in range (0,1) is a Nash equilibrium.

Search over support sets to find a Nash equilibrium

Example: computing Nash equilibrium

		goalkeeper	
		left	right
kicker	left	0.6	0.2
	right	0.3	0.9

Probability of a goal.

When would goalkeeper randomize?
Let p_k be probability the kicker will kick right.

$$\begin{aligned} P(goal \mid jump\ left) &= p(goal \mid jump\ right) \\ p_k * 0.3 + (1 - p_k) * 0.6 &= p_k * 0.9 + (1 - p_k) * 0.2 \\ 0.6 - 0.2 &= (0.6 - 0.3 + 0.9 - 0.2) * p_k \\ p_k &= 0.4 \end{aligned}$$

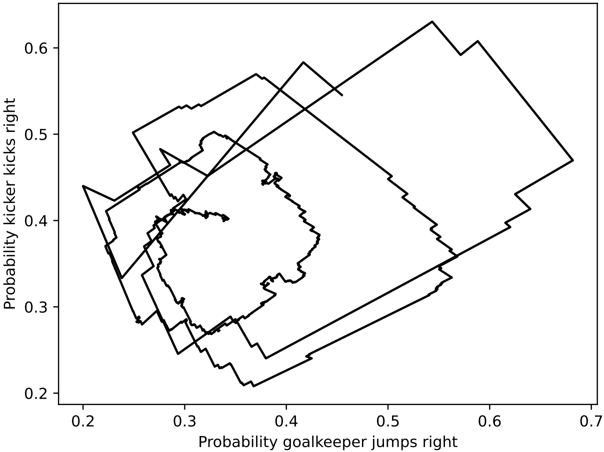
Similarly for goal keeper: $P(jump\ right) = 0.3$
Probability of a goal is:
$(0.6*0.7)*0.6+(0.6*0.3)*0.2+(0.4*0.7)*0.3+(0.4*0.3)*0.9 = 0.48$

Fictitious Play

- Collect statistics of the other player.
- Assuming those statistics reflect the stochastic policy of the other agent, play a best response.
- Both players using fictitious play converges to a Nash equilibrium for many types of games (including two-player zero-sum games).
- If an opposing agent knew the exact strategy (whether learning or not) agent A was using, and could predict what agent A would do, it could exploit that knowledge.

controller $Stochastic\_policy\_iteration(S, A, \alpha, \gamma, q\_init, p\_init)$
    Inputs
        S is states, A is actions, α is step size, γ discount
        q_init and p_init (> 0) are initial Q and P values
    Local
        $P[S, A]$ unnormalized $P(a \mid s)$ ▷ Dirichlet
        $Q[S, A]$ estimate of value of doing A in state S
    $P[s, a] := p\_init$; $Q[s, a] := q\_init$ for each $s \in S$ and $a \in A$
    observe state s; **select** action a at random
    repeat
        $do(a)$
        observe reward r, state s'
        select action a' based on $P[s', a'] / \sum_{a''} P[s', a'']$
        $Q[s, a] := Q[s, a] + \alpha * (r + \gamma * Q[s', a'] - Q[s, a])$
        $a\_best := \arg\max_a(Q[s, a])$
        $P[s, a\_best] = P[s, a\_best] + 1$
        $s := s'$; $a := a'$
    until termination

Stochastic Policies



Repeated playing goal-kick game with single state ($\alpha = 0.1$, $\gamma = 0$, $q\_init = 1$, $p\_init = 5$).
AIPython: `masLearn.py`

Stochastic Policies

AlphaZero – plays world-class chess, shogi, and Go. Improvement of program that beat Lee Sedol in 2016.

- implements modified policy iteration
- uses a deep neural network:
  - Input:the board position
  - Output: the value function and a stochastic policy
- To get a better estimate of the current state, it does stochastic simulation (forward sampling) of the rest of the game, using the stochastic policy with the upper confidence bound (UCB).
- This relies on a model to restart the search from any point.
- It was trained on self-play, playing itself for tens of millions of games.