CAP 4630
Artificial Intelligence

Instructor: Sam Ganzfried

sganzfri@cis.fiu.edu

Schedule

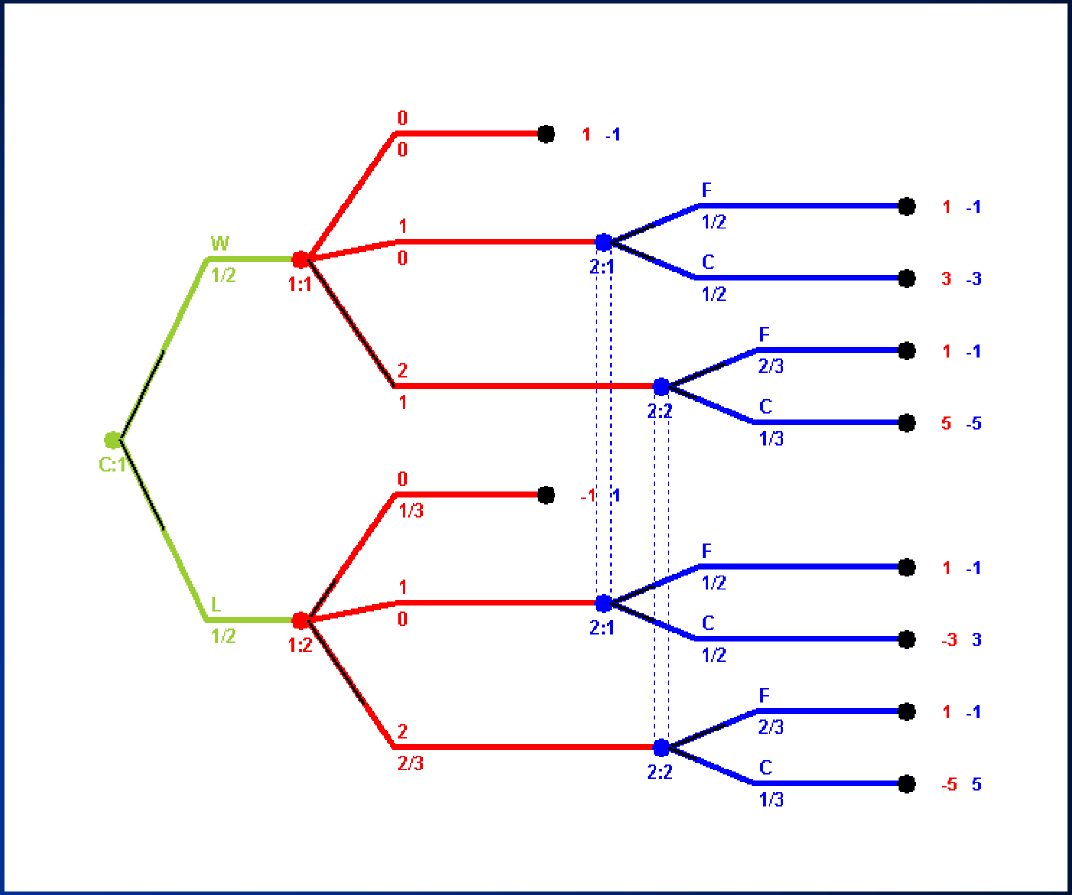
- 11/21: Wrap up multiagent systems (game theory)/elaborate on class project. Start machine learning (classification).
- 11/28, 11/30, 12/5: Continue machine learning (regression, clustering, deep learning)
- I will still try to discuss topics in Markov decision processes and reinforcement learning as they relate to the above topics.
- 12/7: Project presentations and class project due
  - Project code due Monday 12/4 at 2PM on Moodle.
- Final exam on 12/14

Announcements

- HW4 out last week (final homework assignment) due 12/1 (2:05pm in lecture or 2:00pm on Moodle)
  – https://www.cs.cmu.edu/~sganzfri/HW4_AI.pdf

Gambit

- http://gambit.sourceforge.net/gambit15/gui.html



Class project

- For the class project students will implement an agent for 3-player Kuhn poker. This is a simple, yet interesting and nontrivial, variant of poker that has appeared in the AAAI Annual Computer Poker Competition. The grade will be partially based on performance against the other agents in a class-wide competition, as well as final reports and presentations describing the approaches used. Students can work alone or in groups of up to 3.
- Link to play against optimal strategy for one-card poker:
  - http://www.cs.cmu.edu/~ggordon/poker/
- Paper on Nash equilibrium strategies for 3-player Kuhn poker
  - http://poker.cs.ualberta.ca/publications/AAMAS13-3pkuhn.pdf
- https://moodle.cis.fiu.edu/v3.1/mod/forum/discuss.php?d=21801

Multiagent systems (game theory)

- Strategic multiagent interactions occur in all fields
  - Economics and business: bidding in auctions, offers in negotiations
  - Political science/law: fair division of resources, e.g., divorce settlements
  - Biology/medicine: robust diabetes management (robustness against "adversarial" selection of parameters in MDP)
  - Computer science: theory, AI, PL, systems; national security (e.g., deploying officers to protect ports), cybersecurity (e.g., determining optimal thresholds against phishing attacks), internet phenomena (e.g., ad auctions)

- Theorem (von Neumann): In chess, one and only one of the following must be true:
  i. White has a winning strategy
  ii. Black has a winning strategy
  iii. Each of the two players has a strategy guaranteeing at least a draw.
- Applies to ALL chess matches, not a particular match
- Theorem is significant because a priori it might have been the case that none of the alternatives was possible; one could have postulated that no player could ever have a strategy always guaranteeing a victory, or at least a draw.

Checkers is Solved (Science '07)

- The game of checkers has roughly 500 billion possible positions (5×10^{20}). The task of solving the game, determining the final result in a game with no mistakes made by either player, is daunting. Since 1989, almost continuously, dozens of computers have been working on solving checkers, applying state-of-the-art artificial intelligence techniques to the proving process. This paper announces that checkers is now solved: Perfect play ……

- The game of checkers has roughly 500 billion possible positions (5×10^{20}). The task of solving the game, determining the final result in a game with no mistakes made by either player, is daunting. Since 1989, almost continuously, dozens of computers have been working on solving checkers, applying state-of-the-art artificial intelligence techniques to the proving process. This paper announces that checkers is now solved: Perfect play by both sides leads to a draw. This is the most challenging popular game to be solved to date, roughly one million times as complex as Connect Four. Artificial intelligence technology has been used to generate strong heuristic-based game-playing programs, such as Deep Blue for chess. Solving a game takes this to the next level by replacing the heuristics with perfection.

Connect Four

Connect Four

- The solved conclusion for Connect Four is first player win. With perfect play, the first player can force a win, on or before the 41st move by starting in the middle column. The game is a theoretical draw when the first player starts in the columns adjacent to the center. For the edges of the game board, column 1 and 2 on left (or column 7 and 6 on right), the exact move-value score for first player start is loss on the 40th move, and loss on the 42nd move, respectively. In other words, by starting with the four outer columns, the first player allows the second player to force a win.

2-player limit Hold'em poker is solved (Science 2015)

Heads-up Limit Hold'em Poker is Solved

Michael Bowling,[1*] Neil Burch,[1] Michael Johanson,[1] Oskari Tammelin[2]
[1]Department of Computing Science, University of Alberta,
Edmonton, Alberta, T6G2E8, Canada
[2]Unaffiliated, http://jeskola.net

*To whom correspondence should be addressed; E-mail: bowling@cs.ualberta.ca

Poker is a family of games that exhibit imperfect information, where players do not have full knowledge of past events. Whereas many perfect information games have been solved (e.g., Connect Four and checkers), no nontrivial imperfect information game played competitively by humans has previously been solved. Here, we announce that heads-up limit Texas hold'em is now essentially weakly solved. Furthermore, this computation formally proves the common wisdom that the dealer in the game holds a substantial advantage. This result was enabled by a new algorithm, CFR$^+$, which is capable of solving extensive-form games orders of magnitude larger than previously possible.

Heads-up Limit Hold 'em Poker is Solved

- Play against Cepheus here http://poker-play.srv.ualberta.ca/

Strategic-form games

- A game in **strategic form (**or in **normal form**) is an ordered triple G = (N, (S_i) i in N, (u_i) i in N), in which:
  - N = {1,2,…,n} is a finite set of players.
  - Si is the set of strategies of player i, for every player i in N. Denote the set of all vectors of strategies by $S = S_1 \times S_2 \times \ldots \times S_n$.
  - $u_i : S \rightarrow R$ is a function associating each vector of strategies s = (s_i), i in N, with the **payoff (utility)** $u_i(s)$ to player i, for every player i in N.

- Games in strategic form are sometimes called **matrix games**
- When n = 2, we call the games **bimatrix games**, as they are given by two matrices, one for the payoff of each player.

Chicken

- The game of chicken models two drivers, both headed for a single-lane bridge from opposite directions. The first to swerve away yields the bridge to the other. If neither player swerves, the result is a costly deadlock in the middle of the bridge, or a potentially fatal head-on collision. It is presumed that the best thing for each driver is to stay straight while the other swerves (since the other is the "chicken" while a crash is avoided). Additionally, a crash is presumed to be the worst outcome for both players. This yields a situation where each player, in attempting to secure his best outcome, risks the worst.

Chicken

	Swerve	Straight
Swerve	Tie, Tie	Lose, Win
Straight	Win, Lose	Crash, Crash

Fig. 1: A payoff matrix of Chicken

Chicken

	Swerve	Straight
Swerve	0, 0	-1, +1
Straight	+1, -1	-10, -10

Fig. 2: Chicken with numerical payoffs

Security game

- Random strategy:
  - *Increase cost/uncertainty to attackers*

Adversary

Defender

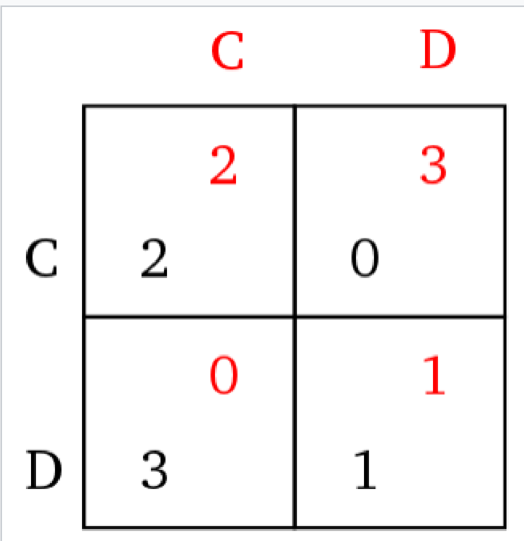
	Target #1	Target #2
Target #1	4, -3	-1, 1
Target #2	-5, 5	2, -1

Rock-paper-scissors

	rock	**paper**	**scissors**
Rock	0,0	-1, 1	1, -1
Paper	1,-1	0, 0	-1,1
Scissors	-1,1	1,-1	0,0

Prisoner's dilemma

	Prisoner B stays silent (*cooperates*)	Prisoner B betrays (*defects*)
Prisoner A stays silent (*cooperates*)	Each serves 1 year	Prisoner A: 3 years Prisoner B: goes free
Prisoner A betrays (*defects*)	Prisoner A: goes free Prisoner B: 3 years	Each serves 2 years



A payoff matrix of the standard dilemma of cooperation and defection

Canonical PD payoff matrix

	Cooperate	Defect
Cooperate	*R*, *R*	*S*, *T*
Defect	*T*, *S*	*P*, *P*

T > R > P > S

Battle of the sexes

- Imagine a couple that agreed to meet this evening, but cannot recall if they will be attending the opera or a football match (and the fact that they forgot is common knowledge). The husband would prefer to go to the football game. The wife would rather go to the opera. Both would prefer to go to the same place rather than different ones. If they cannot communicate, where should they go?

	Opera	Football
Opera	3,2	0,0
Football	0,0	2,3

Battle of the Sexes 1

	Opera	Football
Opera	3,2	1,1
Football	0,0	2,3

Battle of the Sexes 2

Strategic-form game examples

- Chicken
- Security game
- Rock-paper-scissors
- Prisoner’s dilemma
- Battle of the sexes

- We saw von Neumann’s theorem in the special case of two players and three possible outcomes: victory for White, a draw, or victory for Black.
- Central question of game theory: what “will happen” in a given game?

Central question of game theory

1. An empirical, descriptive interpretation: How do players, in fact, play in a given game?
2. A normative interpretation: How “should” players play in a given game?
3. A theoretical interpretation: What can we predict will happen in a game given certain assumptions regarding “reasonable” or “rational” behavior on the part of the players?

Descriptive game theory

- Observations of the actual behavior of players, both in real-life situations and in artificial laboratory conditions where they are asked to play games and their behavior is recorded.
  - Behavioral economics, psychology

Normative interpretation

- Appropriate for a judge, legislator, or arbitrator called upon to determine the outcome of a game based on several agreed-upon principles, such as justice, efficiency, nondiscrimination, and fairness.
- Best suited for the study of cooperative games, in which binding agreements are possible, enable outcomes to be derived from “norms” or agreed-upon principles, or determined by an arbitrator who bases his decisions on those principles.

Theoretical interpretation

- After we have described a game, what can we expect to happen?
- What outcomes, or set of outcomes, will reasonably ensue, given certain assumptions regarding the behavior of the players?

- For each of the five example games we discussed:
  - How will real players act?
  - How "should" players act?
  - How would theoretically perfectly rational players act?
- Golden Balls: Split or Steal? https://www.youtube.com/watch?v=S0qjK3TWZE8

Game theory background

	rock	paper	scissors
Rock	0,0	-1, 1	1, -1
Paper	1,-1	0, 0	-1,1
Scissors	-1,1	1,-1	0,0

- Players
- Actions (aka pure strategies)
- Strategy profile: e.g., (R,p)
- Utility function: e.g., u_1(R,p) = -1, u_2(R,p) = 1

Zero-sum game

	rock	paper	scissors
Rock	0,0	-1, 1	1, -1
Paper	1,-1	0, 0	-1,1
Scissors	-1,1	1,-1	0,0

- Sum of payoffs is zero at each strategy profile: e.g., $u_1(R,p) + u_2(R,p) = 0$
- Models purely adversarial settings

Mixed strategies

- Probability distributions over pure strategies
- E.g., R with prob. 0.6, P with prob. 0.3, S with prob. 0.1

Best response (aka nemesis)

- Any strategy that maximizes payoff against opponent's strategy
- If P2 plays (0.6, 0.3, 0.1) for r,p,s, then a best response for P1 is to play P with probability 1

Nash equilibrium

- Strategy profile where all players simultaneously play a best response
- Standard solution concept in game theory
  - Guaranteed to always exist in finite games [Nash 1950]
- In Rock-Paper-Scissors, the unique equilibrium is for both players to select each pure strategy with probability 1/3

Minimax Theorem

- Minimax theorem: For every two-player zero-sum game, there exists a value v* and a mixed strategy profile σ* such that:
  a. P1 guarantees a payoff of at least v* in the worst case by playing σ^*_1
  b. P2 guarantees a payoff of at least -v* in the worst case by playing σ^*_2
- v* (= v_1) is the *value* of the game
- All equilibrium strategies for player i guarantee at least v_i in the worst case
- For RPS, v* = 0

Exploitability

- Exploitability of a strategy is difference between value of the game and performance against a best response
  - Every equilibrium has zero exploitability
- Always playing rock has exploitability 1
  - Best response is to play paper with probability 1

Nash equilibria in two-player zero-sum games

- Zero exploitability – "unbeatable"
- Exchangeable
  - If (a,b) and (c,d) are NE, then (a,d) and (c,b) are too
- Can be computed in polynomial time by a linear programming (LP) formulation

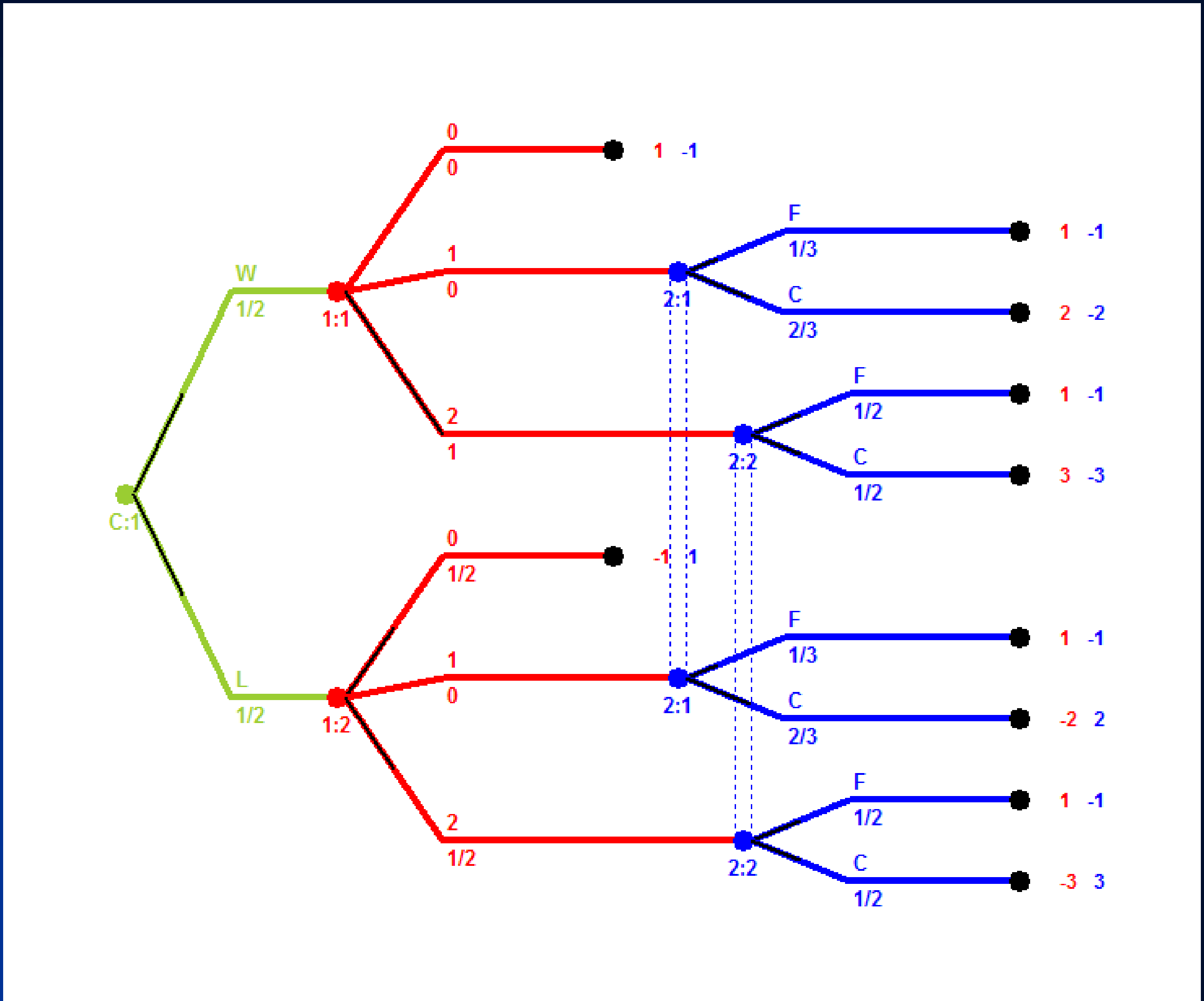
Nash equilibria in multiplayer and non-zero-sum games

- None of the two-player zero-sum results hold
- There can exist multiple equilibria, each with different payoffs to the players
- If one player follows one equilibrium while other players follow a different equilibrium, overall profile is not guaranteed to be an equilibrium
- If one player plays an equilibrium, he could do worse if the opponents deviate from that equilibrium
- Computing an equilibrium is PPAD-hard

Imperfect information

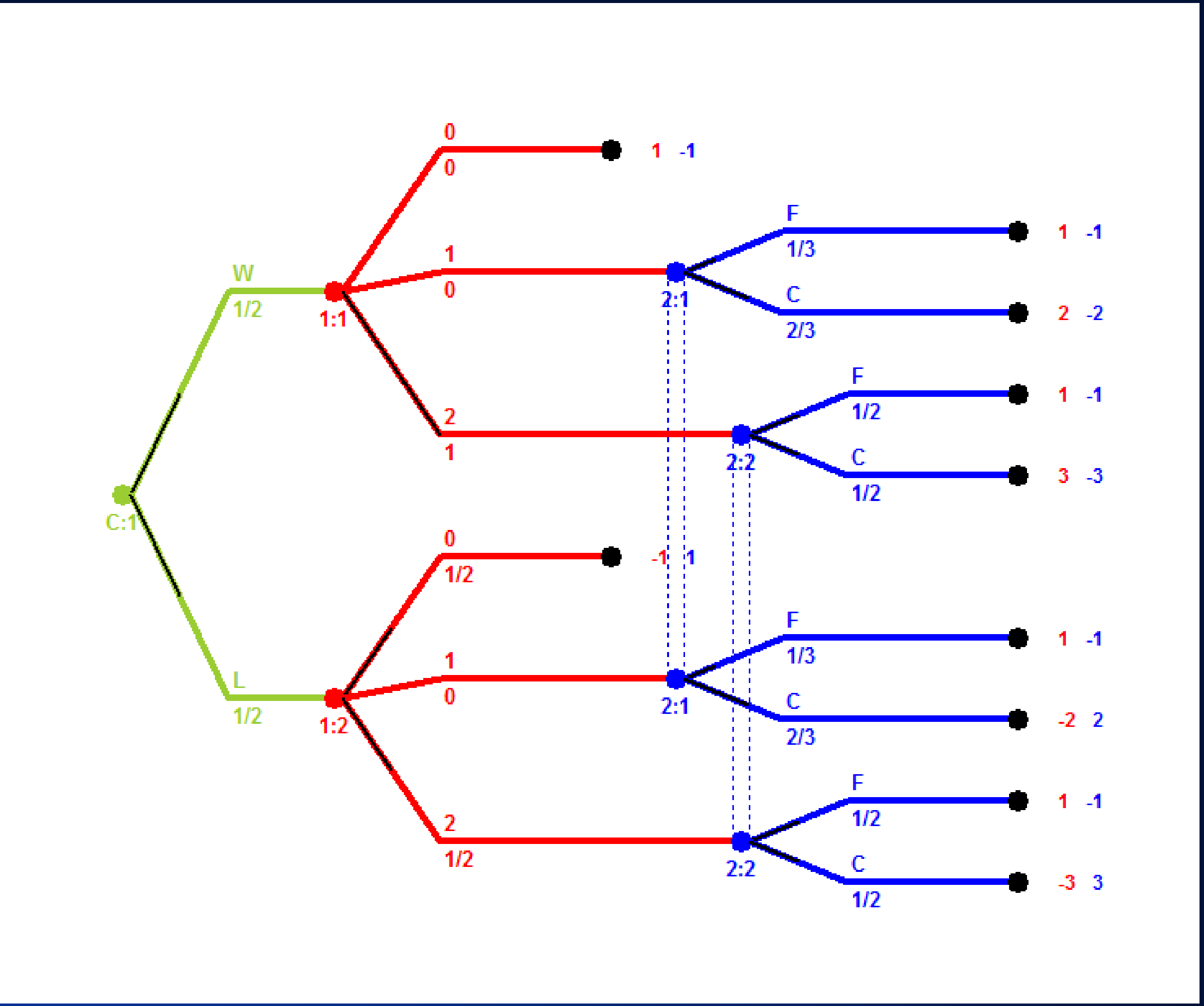
- In many important games, there is information that is private to only some agents and not available to other agents
  - In auctions, each bidder may know his own valuation and only know the distribution from which other agents’ valuations are drawn
  - In poker, players may not know private cards held by other players

Extensive-form representation



Extensive-form games

- Two-player zero-sum EFGs can be solved in polynomial time by linear programming
  - Scales to games with up to 10^8 states
- Iterative algorithms (CFR and EGT) have been developed for computing an ε-equilibrium that scale to games with 10^{17} states
  - CFR also applies to multiplayer and general sum games, though no significant guarantees in those classes
  - (MC)CFR is self-play algorithm that samples actions down tree and updates regrets and average strategies stored at every information set



C:1
W
1/2
L
1/2
1:1
1:2
0
0
1
0
2
1
0
1/2
1
0
2
1/2
2:1
2:2
2:1
2:2
F
1/3
C
2/3
F
1/2
C
1/2
F
1/3
C
2/3
F
1/2
C
1/2
1 -1
1 -1
2 -2
1 -1
3 -3
-1 1
1 -1
-2 2
1 -1
-3 3

WL/12	CC	CF	FC	FF
00	0	0	0	0
01	-0.5	-0.5	1	1
02	-1	1	-1	1
10				
11				
12				
20				
21				
22				

Extensive-form game

- A game in **extensive form** is given by a *game tree*, which consists of a directed graph in which the set of vertices represents positions in the game, and a distinguished vertex, called the *root*, represents the starting position of the game. A vertex with no outgoing edges represents a terminal position in which play ends. To each terminal vertex corresponds an outcome that is realized when the play terminates at that vertex. Any nonterminal vertex represents either a chance move (e.g., a toss of a die or a shuffle of a deck of cards) or a move of one of the players. To any chance-move vertex corresponds a probability distribution over edges emanating from that vertex, which correspond to the possible outcomes of the chance move.

Perfect vs. imperfect information

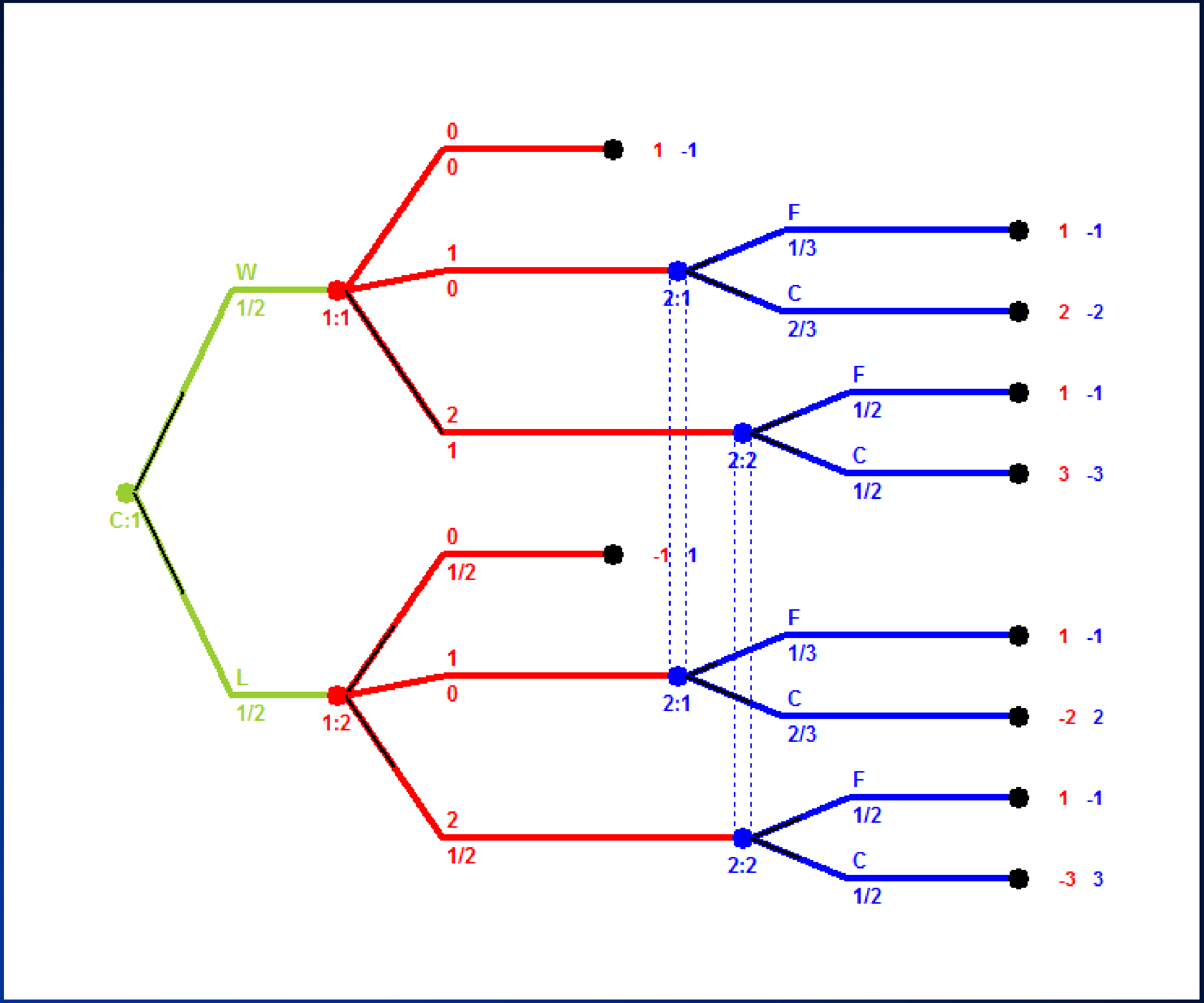
- To describe games with imperfect information, in which players do not necessarily know the full board position (like poker), we introduce the notion of *information sets*. An information set of a player is a set of decision vertices of the player that are indistinguishable by him given his information at that stage of the game. A game of *perfect information* is a game in which all information sets consist of a single vertex. In such a game whenever a player is called to take an action, he knows the exact history of actions and chance moves that led to that position.

- A *strategy* of a player is a function that assigns to each of his information sets an action available to him at that information set. A path from the root to a terminal vertex is called a *play* of the game. When the game has no chance moves, any vector of strategies (one for each player) determines the play of the game, and hence the outcome. In a game with chance moves, any vector of strategies determines a probability distribution over the possible outcomes of the game.

Strategies in imperfect-information games

- A strategy of player i is a function from each of his information sets to the set of actions available at that information set.
- Just as in games with chance moves and perfect information, a strategy vector determines a distribution over the outcomes of a game.

- Every extensive-form game can be converted to an equivalent strategic-form game, and therefore all the prior concepts and theoretical results (e.g., domination, security level, mixed strategies, Nash equilibrium, Minmax Theorem) will apply. However, this conversion produces a strategic-form game that has size that is exponential in the size of the original game tree, and is infeasible for large games. Therefore, we would like do develop algorithms that operate directly on extensive-form games and avoid the conversion to strategic form games.

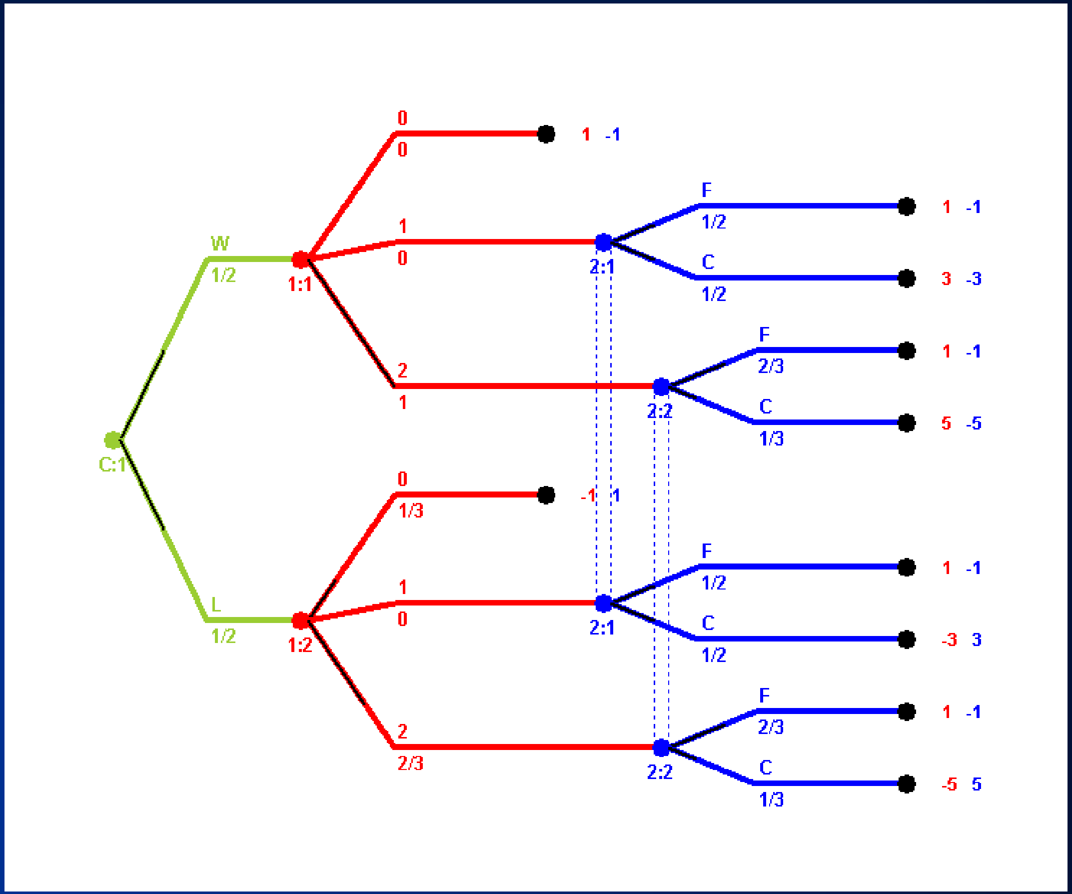


W
1/2
L
1/2
C:1
1:1
1:2
0
0
1
0
2
1
0
1/2
1
0
2
1/2
1 -1
-1 1
2:1
2:2
2:1
2:2
F
1/3
C
2/3
F
1/2
C
1/2
F
1/3
C
2/3
F
1/2
C
1/2
1 -1
2 -2
1 -1
3 -3
1 -1
-2 2
1 -1
-3 3

WL/12	CC	CF	FC	FF
00	0	0	0	0
01	-0.5	-0.5	1	1
02	-1	1	-1	1
10				
11				
12				
20				
21				
22				

Gambit

- http://gambit.sourceforge.net/gambit15/gui.html



Algorithms for game solving

- Two-player zero-sum games: there exists a linear programming formulation and it can be solved in polynomial time.
- For two player "general-sum" and games with more than two players, it is PPAD-hard (though not NP-hard), and widely conjectured no efficient algorithms exist.
- For two-player zero-sum extensive-form games, there also exists a linear-programming formulation, despite the fact that converting it to normal-form would involve an exponential blowup in size of the game tree.

Computing Nash equilibria of two-player zero-sum games

- Consider the game $G = (\{1,2\}, A_1 \times A_2, (u_1, u_2))$.
- Let U^*_i be the expected utility for player i in equilibrium (the value of the game); since the game is zero-sum, $U^*_1 = - U^*_2$.
- Recall that the Minmax Theorem tells us that U^*_1 holds constant in all equilibria and that it is the same as the value that player 1 achieves under a minmax strategy by player 2.
- Using this result, we can formulate the problem of computing a Nash equilibrium as the following optimization:

Minimize U^*_1

Subject to $\Sigma_{k \text{ in } A2} u_1,(a^j_1, a^k_2) * s^k_2 <= U^*_1$ for all j in A_1

$\Sigma_{k \text{ in } A2} s^k_2 = 1$

$s^k_2 >= 0$ for all k in A_2

Minimize U^*_1

Subject to $\Sigma_{k \text{ in } A2}\, u_1(a^j_1, a^k_2) * s^k_2 <= U^*_1$ for all j in A_1

$\Sigma_{k \text{ in } A2}\, s^k_2 = 1$

$s^k_2 >= 0$ for all k in A_2

- Note that all of the utility terms $u_1(*)$ are constants while the mixed strategy terms s^k_2 and U^*_1 are variables.

$$\text{Minimize } U^*_1$$

$$\text{Subject to } \Sigma_{k \text{ in } A2}\, u_1(a^j_1, a^k_2) * s^k_2 <= U^*_1 \qquad \text{for all } j \text{ in } A_1$$

$$\Sigma_{k \text{ in } A2}\, s^k_2 = 1$$

$$s^k_2 >= 0 \qquad \text{for all } k \text{ in } A_2$$

- First constraint states that for every pure strategy j of player 1, his expected utility for playing any action j in A_1 given player 2's mixed strategy s_1 is at most U^*_1. Those pure strategies for which the expected utility is exactly U^*_1 will be in player 1's best response set, while those pure strategies leading to lower expected utility will not.
- As mentioned earlier, U^*_1 is a variable; we are selecting player 2's mixed strategy in order to minimize U^*_1 subject to the first constraint. Thus, player 2 plays the mixed strategy that minimizes the utility player 1 can gain by playing his best response.

Minimize U^*_1

Subject to $\Sigma_{k \text{ in } A2}\, u_1(a^j_1, a^k_2) * s^k_2 <= U^*_1$ for all j in A_1

$\Sigma_{k \text{ in } A2}\, s^k_2 = 1$

$s^k_2 >= 0$ for all k in A_2

- The final two constraints ensure that the variables s^k_2 are consistent with their interpretation as probabilities. Thus, we ensure that they sum to 1 and are nonnegative.

Learning in games

- In game theory, **fictitious play** is a learning rule first introduced by George W. Brown. In it, each player presumes that the opponents are playing stationary (possibly mixed) strategies. At each round, each player thus best responds to the empirical frequency of play of their opponent. Such a method is of course adequate if the opponent indeed uses a stationary strategy, while it is flawed if the opponent's strategy is non-stationary. The opponent's strategy may for example be conditioned on the fictitious player's last move.

Fictitious play

- Simple “learning” update rule
- Initially proposed as an iterative method for computing Nash equilibria in zero-sum games, not as a learning model!
- Brown, G.W. (1951) “Iterative Solutions of Games by Fictitious Play”

- Algorithm:

  Initialize beliefs about the opponent’s strategy

  Repeat:

  1) Play a best response to the assessed strategy of the opponent

  2) Observe the opponent’s actual play and update beliefs accordingly

- In fictitious play, the agent believes that his opponent is playing the mixed strategy given by the empirical distribution of the opponent's previous actions. That is, if A is the set of the opponent's actions, and for every a in A we let w(a) be the number of times that the opponent has played action a, then the agent assess the probability of a in the opponent's mixed strategy as
  - $P(a) = w(a) / \sum_{a' \text{ in } A} w(a')$

- For example, in a repeated Prisoner's Dilemma game, if the opponent has played C, C, D, C, D in the first five games, before the sixth game he is assumed to be playing the mixed strategy (0.6, 0.4).
- In general the tie-breaking rule chosen has little effect on the results of fictitious play.
- On the other hand, fictitious play is very sensitive to the players' initial beliefs. This choice, which can be interpreted as action counts that were observed before the start of the game, can have a radical impact on the learning process. Note that one must pick some nonempty prior belief for each agent; the prior beliefs cannot be (0,…0), since this does not define a meaningful mixed strategy.

	Heads	Tails
Heads	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs	2's beliefs
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2	T	H	(2.5,3)	(2,3.5)
3	T	H	(3.5,3)	(2,4.5)
4	H	H	(4.5,3)	(3,4.5)
5	H	H	(5.5,3)	(4,4.5)
6	H	H	(6.5,3)	(5,4.5)
7	H	T	(6.5,4)	(6,4.5)
⋮	⋮	⋮	⋮	⋮

Table 7.1: Fictitious play of a repeated game of Matching Pennies.

- As the number of rounds tends to infinity, the empirical distribution of the play of each player will converge to (0.5,0.5). If we take this distribution to be the mixed strategy of each player, the play converges to the unique Nash equilibrium of the normal form stage game, that in which each player plays the mixed strategy (0.5,0.5).

Machine learning

- An agent is **learning** if it improves its performance on future tasks after making observations about the world. Learning can range from the trivial, as exhibited by jotting down a phone number, to the profound, as exhibited by Albert Einstein, who inferred a new theory of the universe.
- We will start by concentrating on one class of learning problem, which seems restricted but actually has vast applicability: from a collection of input-output pairs, learn a function that predicts the output for new inputs.

Machine learning

- Why would we want an agent to learn? If the design of the agent can be improved, why wouldn’t the designers just program in that improvement to begin with? There are three main reasons.

- First, the designers cannot anticipate all possible situations that the agent might find itself in. For example, a robot designed to navigate mazes must learn the layout of each new maze it encounters.

- Second, the designers cannot anticipate all changes over time; a program designed to predict tomorrow's stock market prices must learn to adapt when conditions change from boom to bust.

- Third, sometimes human programmers have no idea how to program a solution themselves. For example, most people are good at recognizing the faces of family members, but even the best programmers are unable to program a computer to accomplish that task, except by using learning algorithms.

Supervised learning

- The task of supervised learning is this: Given a **training set** of N example input-output pairs $(x_1, y_1),(x_2, y_2),\ldots,(x_N, y_N)$,
- Where each y_j was generated by an unknown function $y = f(x)$, discover a function h that approximates the true function f.
- Example: x_i, can be True/False for whether email says "Prize" in it, and y_i can be True/False for whether or not it is Spam.
- x and y can be any value, they need not be numbers.
  - E.g., x can be {red, green, blue} for jacket color, and y can be price.
- The function h is a **hypothesis**. Learning is a search through the space of possible hypotheses for one that will perform well, even on new examples beyond the training set.

Supervised learning

- To measure the accuracy of a hypothesis we give it a **test set** of examples that are distinct from the training set.
  - What would happen if we tested on the examples that were trained on?
- We say a hypothesis **generalizes** well if it correctly predicts the value of y for novel examples. Sometimes the function f is stochastic—it is not strictly a function of x, and what we have to learn is a conditional probability distribution, P(Y|x).

Supervised learning

- When the output y is one of a finite set of values (such as *sunny*, *cloudy*, or *rainy*), the learning problem is called **classification**, and is called Boolean or binary classification if there are only two values. When y is a number (such as tomorrow's temperature), the learning problem is called **regression**. (Technically, solving a regression problem is finding a conditional expectation or average value of y, because the probability that we have found *exactly* the right real-valued number for y is 0).

Supervised learning

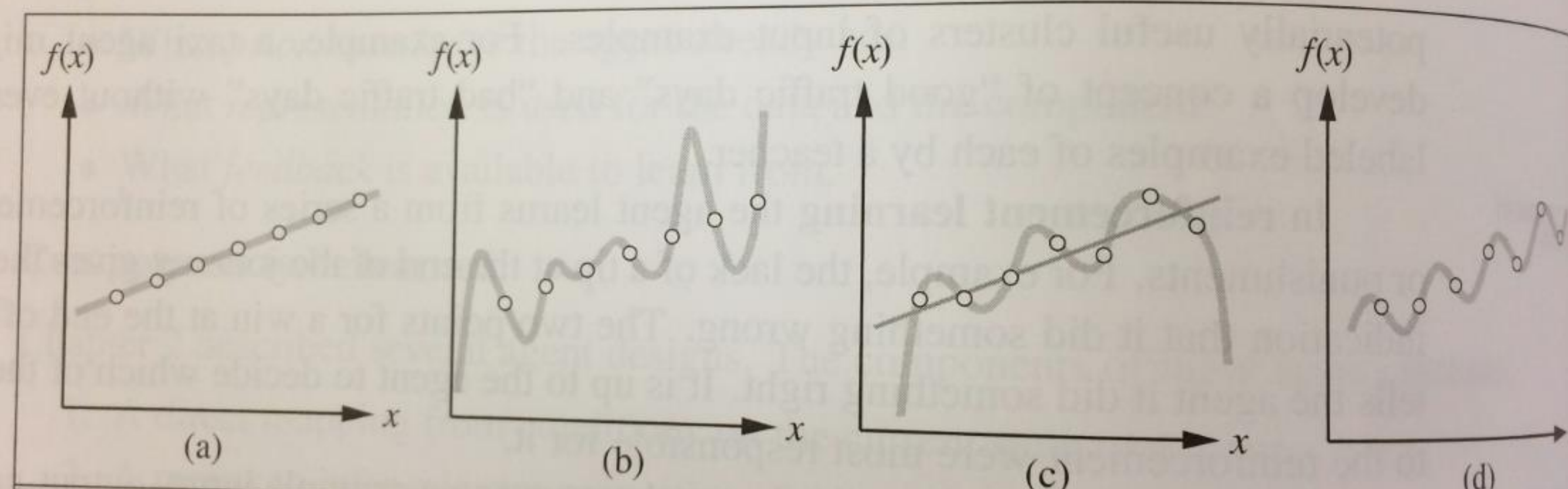


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

Supervised learning

- The figure shows a familiar example: fitting a function of a single variable to some data points. The examples are points in the (x,y) plane, where y = f(x). We don't know what f is, but we will approximate it with a function h selected from a **hypothesis space**, H, which for this example we will take to be the set of polynomials such as x^5 + 3x^2 + 2. Figure a shows some data with an exact fit by a straight line (the polynomial 0.4x + 3). The line is called a **consistent** hypothesis because it agrees with all the data. Figure b shows a high-degree polynomial that is also consistent with all the data. This illustrates a fundamental problem in inductive learning: *how do we choose from among multiple consistent hypotheses*? The answer is to prefer the *simplest* hypothesis consistent with the data. This principle is called **Ockham's razor**, after the 14^{th}-century English philosopher William of Ockham, who used it to argue sharply against all sorts of complications. Defining simplicity is not easy, but it seems clear that a degree-1 polynomial is simpler than a degree-7 polynomial, and thus (a) should be preferred to (b). We will make this intuition more precise later.

Supervised learning

- Figure c shows a second data set. There is no consistent straight line for this data set; in fact, it requires a degree-6 polynomial for an exact fit. There are just 7 data points, so a polynomial with 7 parameters does not seem to be finding any pattern in the data and we do not expect it to generalize well. A straight line that is not consistent with any of the data points, but might generalize fairly well for unseen values of x, is also shown in c. *In general, there is a tradeoff between complex hypotheses that fit the training data well and simpler hypotheses that may generalize better*. In figure d we expand the hypothesis space H to allow polynomials over both x and sin(x), and find that the data in c can be fitted exactly by a simple function of the form ax + b + csin(x). This shows the importance of the hypothesis space.

Homework for next class

- Chapter 22 from Russel/Norvig
- HW3 due Tuesday 11/14
- HW4 out last week due 12/1
- Next lecture: Continue machine learning (classification/regression)