CAP 4630
Artificial Intelligence

Instructor: Sam Ganzfried

sganzfri@cis.fiu.edu

Schedule

- 11/16: Finish probability, focus on multiagent systems (game theory)
- 11/21, 11/28, 11/30, 12/5: Machine learning (classification, regression, clustering, deep learning)
- I will still 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

- HW3 out 10/31 due 11/14 (2:05pm in lecture or 2:00pm on Moodle)
  - https://www.cs.cmu.edu/~sganzfri/HW3_AI.pdf
  - Must be done individually (no partner)
- HW4 out this week

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

Bayes' rule for drug testing

- Even if an individual tests positive, it is more likely that they do not use the drug than that they do. Why? Even though the test appears to be highly accurate, the number of non-users is large compared to the number of users. The number of false positives outweighs the number of true positives.
- To use concrete numbers, if 1000 individuals are tested, there are expected to be 995 non-users and 5 users. From the 995 non-users, $0.01 \times 995 \simeq 10$ false positives are expected. From the 5 users, $0.99 \times 5 \approx 5$ true positives are expected. Out of 15 positive results, only 5, about 33%, are genuine.
- This illustrates the importance of base rates. Daniel Kahneman has argued that the formation of policy can be egregiously misguided if base rates are neglected when using statistics as a basis for guiding public policy.
- The importance of specificity in this example can be seen by calculating that even if sensitivity is raised to 100% and specificity remains at 99% then the probability of the person being a drug user only rises from 33.2% to 33.4%, but if the sensitivity is held at 99% and the specificity is increased to 99.5% then the probability of the person being a drug user rises to about 49.9%.

Bayesian networks

- A Bayesian network is a directed graph in which each node is annotated with quantitative probability information. The full specification is:
  1. Each node corresponds to a random variable, which may be discrete or continuous
  2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y, X is said to be a *parent* of Y. The graph has no directed cycles (and hence is a directed acyclic graph), or DAG.
  3. Each node X_i has a conditional probability distribution $P(X_i \mid \text{Parents}(X_i))$ that quantifies the effect of the parents on the node.

Bayesian networks

- The topology of the network—the set of nodes and links–specifies the conditional independence relationships that hold in the domain, in a way that will be made precise shortly. The *intuitive* meaning of an arrow is typically that X has a *direct influence* on *Y*, which suggests that causes should be parents of effects. It is usually easy for a domain expert to decide what direct influences exist in the domain—much easier, in fact, than actually specifying the probabilities themselves. Once the topology of the Bayesian network is laid out, we need only specify a conditional probability distribution for each variable, given its parents. We will see that the combination of the topology and the conditional distributions suffices to specify (implicitly) the full joint distribution for all the variables.

Bayesian networks

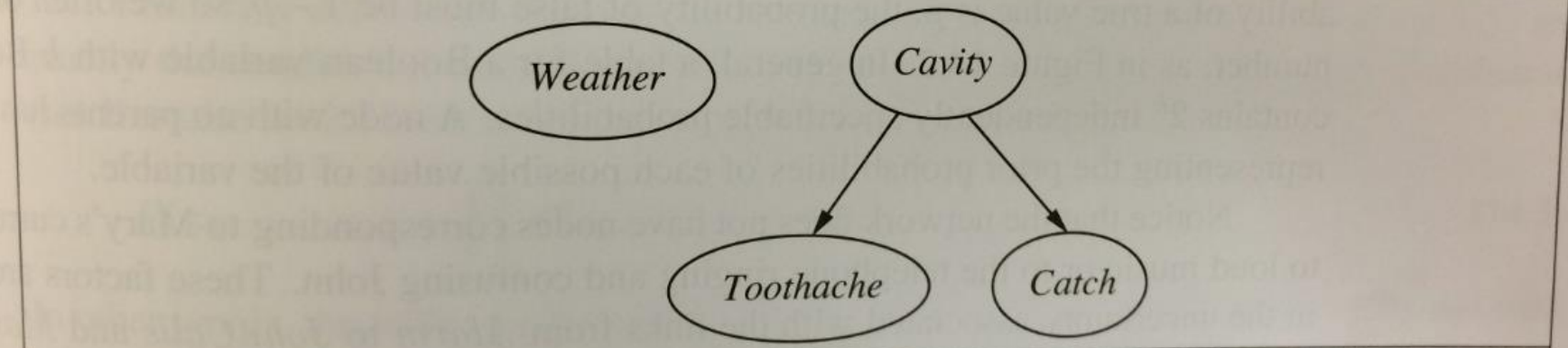


Figure 14.1 A simple Bayesian network in which *Weather* is independent of the other three variables and *Toothache* and *Catch* are conditionally independent, given *Cavity*.

Bayesian networks

- Recall the simple world consisting of the variables *Toothache*, *Cavity*, *Catch*, and *Weather*. We argued that Weather is independent of the other variables; furthermore, we argued that Toothache and Catch are conditionally independent, given Cavity. These relationships are represented by the Bayesian network structure shown above. Formally, the conditional independence of Toothache and Catch, given Cavity, is indicated by the *absence* of a link between Toothache and Catch, whereas no direct causal relationship exists between Toothache and Catch.

Bayesian network

- Now consider the following example. You have a new burglar alarm installed at home. It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes. You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm. John nearly always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, likes rather loud music and often misses the alarm altogether. Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

Bayesian network

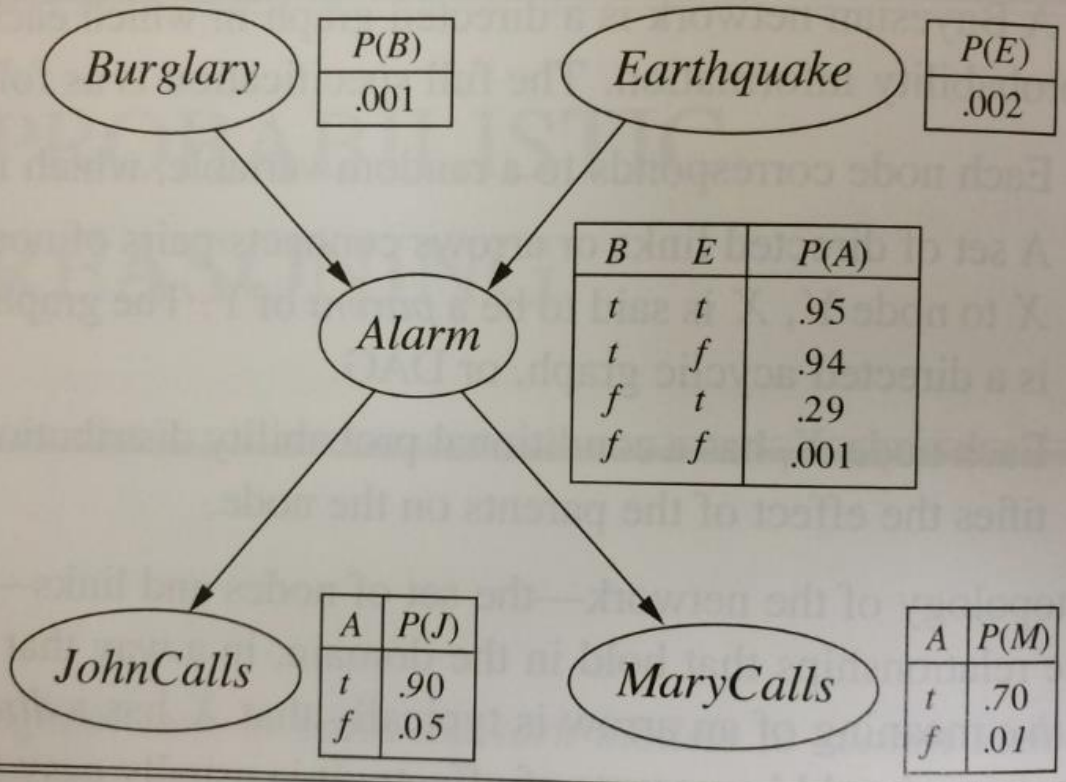


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

Bayesian network

- The network structure shows that burglary and earthquakes directly affect the probability of the alarm's going off, but whether John and Mary call depends only on the alarm. The network thus represents our assumptions that they do not perceive burglaries directly, they do not notice minor earthquakes, and they do not confer before calling.

Bayesian network

- The conditional distributions are shown as a **conditional probability table**, or CPT. Each row in a CPT contains the conditional probability of each node value for a **conditioning case**. A conditioning case is just a possible combination of values for the parent nodes—a miniature possible world. Each row must sum to 1, because the entries represent an exhaustive set of cases for the variable. For Boolean variables, once you know that the probability of a true value is p, the probability of false must be 1-p, so we often omit the second number. In general, a table for a Boolean variable with k Boolean parents contains 2^k independently specifiable probabilities. A node with no parents ha sonly one row, representing the prior probabilities of each possible value of the variable.

Bayesian network

- Notice that the network does not have nodes corresponding to Mary's currently listening to loud music or to the telephone ringing and confusing John. These factors are summarized in the uncertainty associated with the links from Alarm to JohnCalls and MaryCalls. This shows both laziness and ignorance in operation: it would be a lot of work to find out why those factors would be more or less likely in any particularly case, and we have no reasonable way to obtain the relevant information anyway. The probabilities actually summarize a *potentially infinite* set of circumstances in which the alarm might fail to go off (high humidity, power failure, dead battery, cut wires, a dead mouse stuck inside the bell, etc.) or John or Mary might fail to call and report it (out to lunch, on vacation, temporarily deaf, passing helicopter, etc.). In this way, a small agent can cope with a very large world, at least approximately. The degree of approximation can be improved if introduce additional relevant information.

Bayesian network

- There are two ways in which one can understand the semantics of Bayesian networks. The first is to see the network as a representation of the joint probability distribution. The second is to view it as an encoding of a collection of conditional independence statements. The two views are equivalent, but the first turns out to be helpful in understanding how to *construct* networks, whereas the second is helpful in designing inference procedures.

Bayesian network

- Viewed as a piece of "syntax," a Bayesian network is a directed acyclic graph with some numeric parameters attached to each node. One way to define what the network means—its semantics—is to define the way in which it represents a specific joint distribution over all the variables. To do this, we first need to retract (temporarily) what we said earlier about the parameters associated with each node. We said that those parameters correspond to conditional probabilities $P(X_i|Parents(X_i))$; this is a true statement, but until we assign semantics to the network as a whole, we should think of them just as numbers $\theta(X_i|Parents(X_i))$.

Bayesian networks

- A generic entry in the joint distribution is the probability of a conjunction of particular assignments to each variable, such as $P(X_1 = x_1 \text{ AND } \ldots \text{ AND } X_n = x_n)$. We use the notation $P(x_1,\ldots, x_n)$ as an abbreviation for this. The value of this entry is given by the formula:

$$P(x_1,..., x_n) = \prod^n_{i=1} \theta(x_i|\text{Parents}(x_i)),$$

- Where parents(X_i) denotes the values of Parents(x_i) that appear in $x_1,\ldots, x_n$. Thus, each entry in the joint distribution is represented by the product of the appropriate elements of the conditional probability tables (CPTs) in the Bayesian network.

Bayesian networks

- From this definition, it is easy to prove that the parameters $\theta(x_i|\text{Parents}(x_i))$, are exactly the conditional probabilities $P(x_i|\text{Parents}(x_i))$, implied by the joint distribution (homework exercise). Hence, we can rewrite the equation as

  $P(x_1,..., x_n) = \prod^n_{i=1} P(x_i|\text{Parents}(x_i))$.

- In other words, the tables we have been calling conditional probability tables really *are* conditional probability tables according to the semantics defined in the equation.

Bayesian network

- To illustrate this, we can calculate the probability that the alarm has sounded, but neither a burglary nor an earthquake has occurred, and both John and Mary call. We multiply entries from the joint distribution (using single-letter names for the variables):
- P(j, m, a, !b,!e) = P(j|a)P(m|a)P(a|!b AND !e)P(!b)P(!e) = 0.90 * 0.70 * 0.001 * 0.999 * 0.998 = 0.000628.
- We explained earlier that the full joint distribution can be used to answer any query about the domain. If a Bayesian network is a representation of the joint distribution, then it too can be used to answer any query, by summing all the relevant joint entries.

Bayesian networks

- Recall the equation:

  $P(x_1,..., x_n) = \prod^n_{i=1} P(x_i|Parents(x_i))$.
- The next step is to explain how to *construct* a Bayesian network in such a way that the resulting joint distribution is a good representation of a given domain. We will now show that the equation implies certain conditional independence relationships that can be used to guide the knowledge engineer in constructing the topology of the network. First, we rewrite the entries in the joint distribution in terms of conditional probability, using product rule:
- $P(x_1,..., x_n) = P(x_n \mid x_{n-1},\ldots,x_1)P(x_{n-1},\ldots, x_1)$.
- Then we repeat the process, reducing each conjunctive probability to a conditional probability and a smaller conjunction.

Bayesian networks

- We end up with one big product:
- $P(x_1,..., x_n) = P(x_n \mid x_{n-1},\ldots,x_1)\ P(x_{n-1} \mid x_{n-2},\ldots,x_1),*\ldots*$ $P(x_2|x_1)P(x_1) = \prod^n_{i=1} P(x_i|x_{i-1},\ldots,x_1)$.
- This identity is called the **chain rule.** It holds for any set of random variables. Comparing it with the previous equation, we see that the specification of the joint distribution is equivalent to the general assertion that, for every variable X_i in the network,
- $P(X_i|X_{i-1},\ldots, X_1) = P(X_i|\text{Parents}(X_i))$,
- Provided that $\text{Parents}(X_i)$ is a subset of $\{X_{i-1},\ldots, X_1\}$. This last condition is satisfied by numbering the nodes in a way that is consistent with the partial order implicit in the graph structure.

Bayesian networks

- This new equation says that the Bayesian network is a correct representation of the domain only if each node is conditionally independent of its other predecessors in the node ordering, given its parents. We can satisfy this condition with this methodology:
  - Nodes: First determine the set of variables that are required to model the domain. Now order them, $\{X_1,\ldots, X_n\}$. Any order will work, but the resulting network will be more compact if the variables are ordered such that causes precede effects.
  - Links: For i = 1 to n do:
    - Choose, from $X_1,\ldots, X_{i-1}$, a minimal set of parents for X_i, such that the equation is satisfied.
    - For each parent insert a link from the parent to X_i.
    - CPTs: Write down the conditional probability table, $P(X_i|\text{Parents}(X_i))$.

Bayesian networks

- Intuitively, the parents of node X_i should contain all those nodes in $X_1,.., X_{i-1}$ that *directly influence* X_i. For example, suppose we have completed the network in the figure except for the choice of parents for MaryCalls. MaryCalls is certainly influenced by whether there is a Burglary or an Earthquake, but not *directly* influenced. Intuitively, our knowledge of the domain tells us that these events influence Mary's calling. Formally speaking, we believe that the following conditional independence statement holds:
  - P(MaryCalls|JohnCalls, Alarm, Earthquake, Burglary) = P(MaryCalls|Alarm).
- Thus, Alarm will be the only parent node for MaryCalls.

Bayesian network construction

- Because each node is connected only to earlier nodes, this construction method guarantees that the network is acyclic. Another important property of Bayesian networks is that they contain no redundant probability values. If there is no redundancy, then there is no chance for inconsistency: *it is impossible for the knowledge engineer or domain expert to create a Bayesian network that violates the axioms of probability*.

Exact inference in Bayesian networks

- The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of **query** variables, given some observed **event**—that is, some assignment of values to a set of **evidence variables**. To simplify the presentation, we will consider only one query variable at a time; the algorithms can easily be extended to queries with multiple variables. We will use the notation: X denotes the query variable, E denotes the set of evidence variables $E_1,\ldots, E_m$, and e is a particular observed event; Y will denote the nonevidence, nonquery variables $Y_1,\ldots,Y_m$ (called the **hidden variables**). Thus, a complete set of variables is **X** = {X} Union E Union Y. A typical query asks for the posterior probability distribution P(X|e).

Inference in Bayesian networks

- In the burglary network, we might observe the event in which JohnCalls = true, and MaryCalls = true. We could then ask for, say, the probability that a burglary has occurred:
  - P(Burglary | JohnCalls = true, MaryCalls = true)
    = <0.284, 0.716> (for <true,false>).
- Now we will see exact algorithms for computing posterior probabilities and will consider the complexity of this task. It turns out that the general case is intractable, so will have to settle for approximate inference for the general case.

Bayesian network

- We saw that any conditional probability can be computed by summing terms from the full joint distribution. More specifically, a query P(X|e) can be answered using the equation, which we repeat here:
- $P(X|e) = \alpha P(x|e) = \alpha \sum_y P(x,e,y)$.
- Now a Bayesian network gives a complete representation of the full joint distribution. More specifically, we showed that the terms P(x,e,y) in the joint distribution can be written as products of conditional probabilities from the network. Therefore, *a query can be answered using a Bayesian network by computing sums of products of conditional probabilities from the network.*

Bayesian network

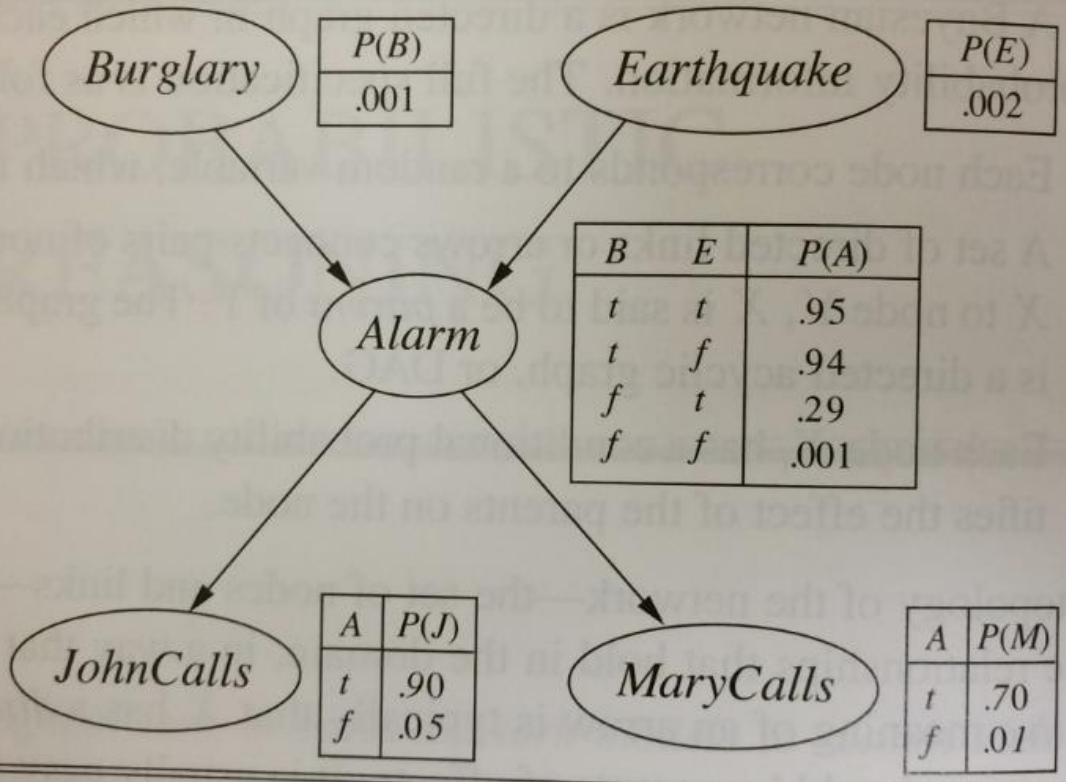


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

Bayesian network

- Consider the query P(Burglary | JohnCalls = true, MaryCalls = true). The hidden variables for this query are Earthquake and Alarm. We now see that:
- $P(B|j,m) = \alpha\, P(B,j,m) = \alpha \sum_e \sum_a P(B,j,m,e,a)$.
- The semantics of Bayesian networks then gives us an expression in terms of CPT entries. For simplicity, we do this just for Burglary = true:
- $P(b|j,m) = \alpha \sum_e \sum_a P(b)P(e)P(a|b,e)P(j|a)P(m|a)$.
- To compute this expression, we have to add four terms, each computed by multiplying five numbers. In the worst case, where we have to sum out almost all the variables, the complexity of the algorithm for a network with n Boolean variables is $O(n\, 2^n)$.

Bayesian network

- An improvement can be made from the following simple observation: the P(b) term is a constant and can be moved outside the summations over a and e, and the P(e) term can be moved outside the summation over a. Hence, we have
- $P(b|j,m) = \alpha\, P(b) \sum_e P(e) \sum_a P(a|b,e) P(j|a) P(m|a)$.
- This expression can be evaluated by looping through the variables in order, multiplying CPT entries as we go. For each summation, we also need to loop over the variable's possible values. The structure of this computation is shown in the figure. Using the numbers, we obtain $P(b|j,m) = \alpha$ <0.00059224, 0.0014919> ~= <0.284,0.716>.
- That is, the chance of a burglary, given calls from both neighbors, is about 28%.

Bayesian networks

- The evaluation process is shown as an expression tree. The ENUMERATION-ASK algorithm evaluates such trees using depth-first recursion. The algorithm is very similar in structure to the backtracking algorithm for solving CSPs and the DPLL algorithm for satisfiability.
- The space complexity of ENUMERATION-ASK is only linear in the number of variables: the algorithm sums over the full joint distribution without ever constructing it explicitly. Unfortunately, its time complexity for a network with n Boolean variables is always O(2^n) – better than the O(n 2^n) for the simple approach described earlier, but still rather grim.

Bayesian network

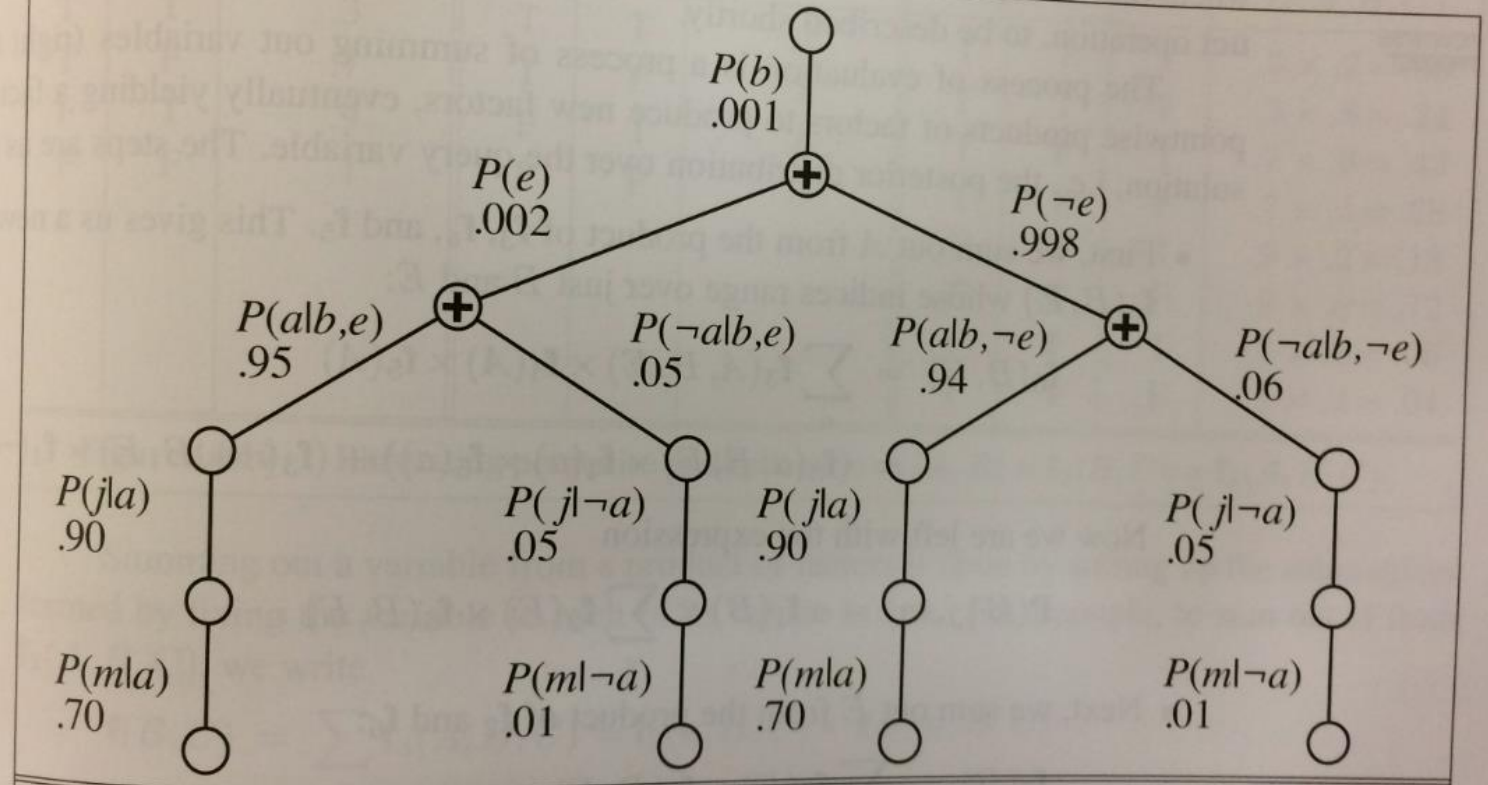


Figure 14.8 The structure of the expression shown in Equation (14.4). The evaluation proceeds top down, multiplying values along each path and summing at the "+" nodes. Notice the repetition of the paths for j and m.

Bayesian network

function ENUMERATION-ASK(X, **e**, bn) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayes net with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$ /* **Y** = *hidden variables* */

  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(bn.VARS, $\mathbf{e}_{x_i}$)
      where $\mathbf{e}_{x_i}$ is **e** extended with $X = x_i$
  return NORMALIZE($\mathbf{Q}(X)$)

---

function ENUMERATE-ALL($vars$, **e**) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  if Y has value y in **e**
    then return $P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), **e**)
    else return $\sum_y P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is **e** extended with $Y = y$

Figure 14.9 The enumeration algorithm for answering queries on Bayesian networks.

Bayesian network

- Note that the tree makes explicit the *repeated expressions* evaluated by the algorithm. The products P(j|a)P(m|a) and P(j|~a)P(m|~a) are computed twice, once for each value of e. We now describe a general method that avoids such wasted computation.

Variable elimination algorithm

- The enumeration algorithm can be improved substantially by eliminating repeated calculations of the kind illustrated above. The idea is simple: do the calculation once and save the results for later use. This is a form of dynamic programming. There are several versions of this approach; we present the **variable elimination** algorithm, which is the simplest. Variable elimination works by evaluating expressions in *right-to-left* order (that is, *bottom up*). Intermediate results are stores, and summations over each variable are done only for those portions of the expression that depend on the variable.

Variable elimination algorithm

- Let us illustrate this process for the burglary network. We evaluate the expression
- $P(B|j,m) = \alpha\, P(B) \sum_e P(e) \sum_a P(a|B,e) P(j|a) P(m|a)$.
  - Denote P(B) by f1(B)
  - Denote P(e) by f2(e)
  - Denote P(a|B,e) by f3(A,B,E)
  - Denote P(j|a) by f4(A)
  - Denote P(m|a) by f5(A)

Variable elimination

- Note that e have annotated each part of the expression with the name of the corresponding **factor**; each factor is a matrix indexed by the values of its argument variables. For example, f4(A) and f5(A) depend just on A because J and M are fixed by the query. They are therefore two-element vectors:
  - f4(A) = (P(j|a), P(j|~a)) = (0.90, 0.05)
  - f5(A) = (P(m|a),P(m|~a)) = (0.70,0.01)
  - f3(A,B,E) will be a 2x2x2 matrix.
- Query can be rewritten as
- $P(B|j,m) = \alpha\ f1(B) \times \sum_e f2(e) \times \sum_a f3(A,B,E) \times f4(A) \times f5(A)$,
- Where the "x" operator is not ordinary matrix multiplication but instead the **pointwise produce** operation, to be described shortly.

Variable elimination

- The process of evaluation is a process of summing out variables (right to left) from pointwise products of factors to produce new factors, eventually yielding a factor that is the solution, i.e., the posterior distribution over the query variable. The steps are as follows:
  - First we sum out A from the product of f3, f4, and f5. This gives us a new 2x2 factor f6(B,E) whose indices range over just B and E:
    - $F6(B,E) = \sum_a f3(A,B,E) \times f4(A) \times f5(A)$
    - $= (f3(a,B,E) \times f4(a) \times f5(a)) + (f3(\sim a,B,E) \times f4(\sim a) \times f5(!a))$.
    - Next we are left with the expression:
    - $P(B|j,m) = \alpha\ f1(B) \times \sum_e f2(e) \times f6(B,E)$

Variable elimination

- Next, we sum out E from the product of f2 and f6: f7(B) = $\sum_e$ f2(e)x f6(B,e)
- = f2(e) x f6(B,e) x f2(~e) x f6(B,~e).
- This leaves the expression P(B|j,m) = α f1(B)xf7(B),
- Which can be evaluated by taking the pointwise product and normalizing the result. Examining this sequence, we see that two basic computational operations are required: pointwise product of a pair of factors, and summing out a variable from a product of factors.
  - Textbook covers details for factor operations, variable ordering, and variable relevance. You will have homework question on these.

```
function ELIMINATION-ASK(X, e, bn) returns a distribution over X
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayesian network specifying joint distribution P(X_1, ..., X_n)

  factors ← []
  for each var in ORDER(bn.VARS) do
      factors ← [MAKE-FACTOR(var, e)|factors]
      if var is a hidden variable then factors ← SUM-OUT(var, factors)
  return NORMALIZE(POINTWISE-PRODUCT(factors))
```

Figure 14.11 The variable elimination algorithm for inference in Bayesian networks.

A	B	$\mathbf{f}_1(A,B)$
T	T	.3
T	F	.7
F	T	.9
F	F	.1

B	C	$\mathbf{f}_2(B,C)$
T	T	.2
T	F	.8
F	T	.6
F	F	.4

A	B	C	$\mathbf{f}_3(A,B,C)$
T	T	T	$.3 \times .2 = .06$
T	T	F	$.3 \times .8 = .24$
T	F	T	$.7 \times .6 = .42$
T	F	F	$.7 \times .4 = .28$
F	T	T	$.9 \times .2 = .18$
F	T	F	$.9 \times .8 = .72$
F	F	T	$.1 \times .6 = .06$
F	F	F	$.1 \times .4 = .04$

Figure 14.10 Illustrating pointwise multiplication: $\mathbf{f}_1(A,B) \times \mathbf{f}_2(B,C) = \mathbf{f}_3(A,B,C)$.

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 x S_2 x … x S_n.
  - u_i : S → 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.

Strategic-form games

- Set of strategies available to the players are not required to be finite
- A game in which strategy set of each player is finite is called a **finite game**
- We will see examples of **infinite games**
- Important: the outcome for each player depends on the strategies chosen by ALL players, not just on his strategy alone

- 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

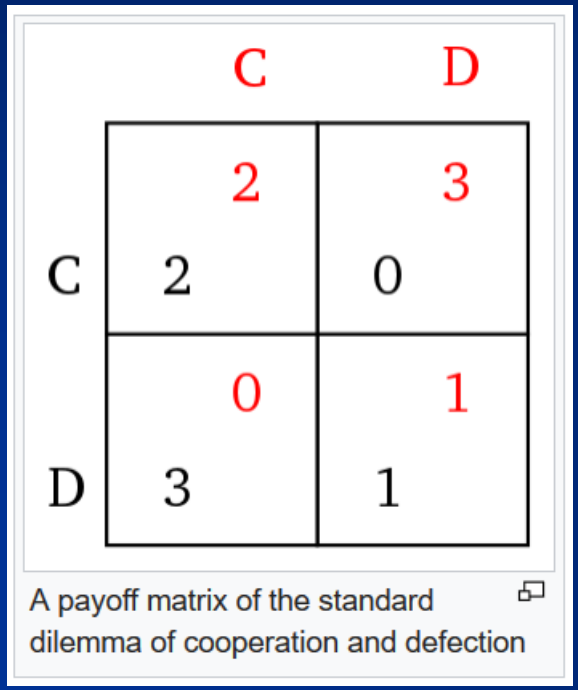
	Target #1	Target #2
Target #1	4, -3	-1, 1
Target #2	-5, 5	2, -1

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

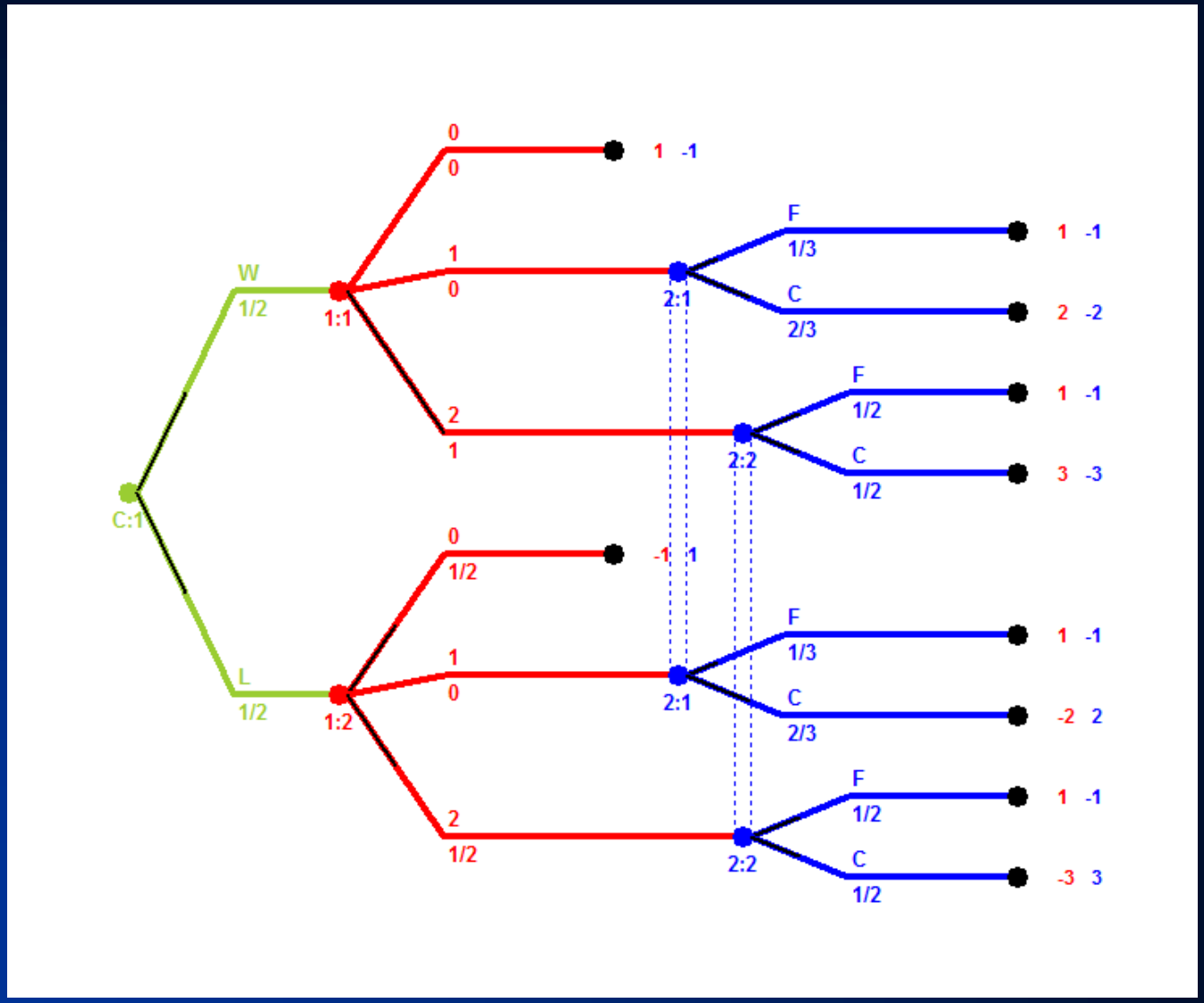
Nash equilibria in multiplayer and non-zero-sum games

- 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

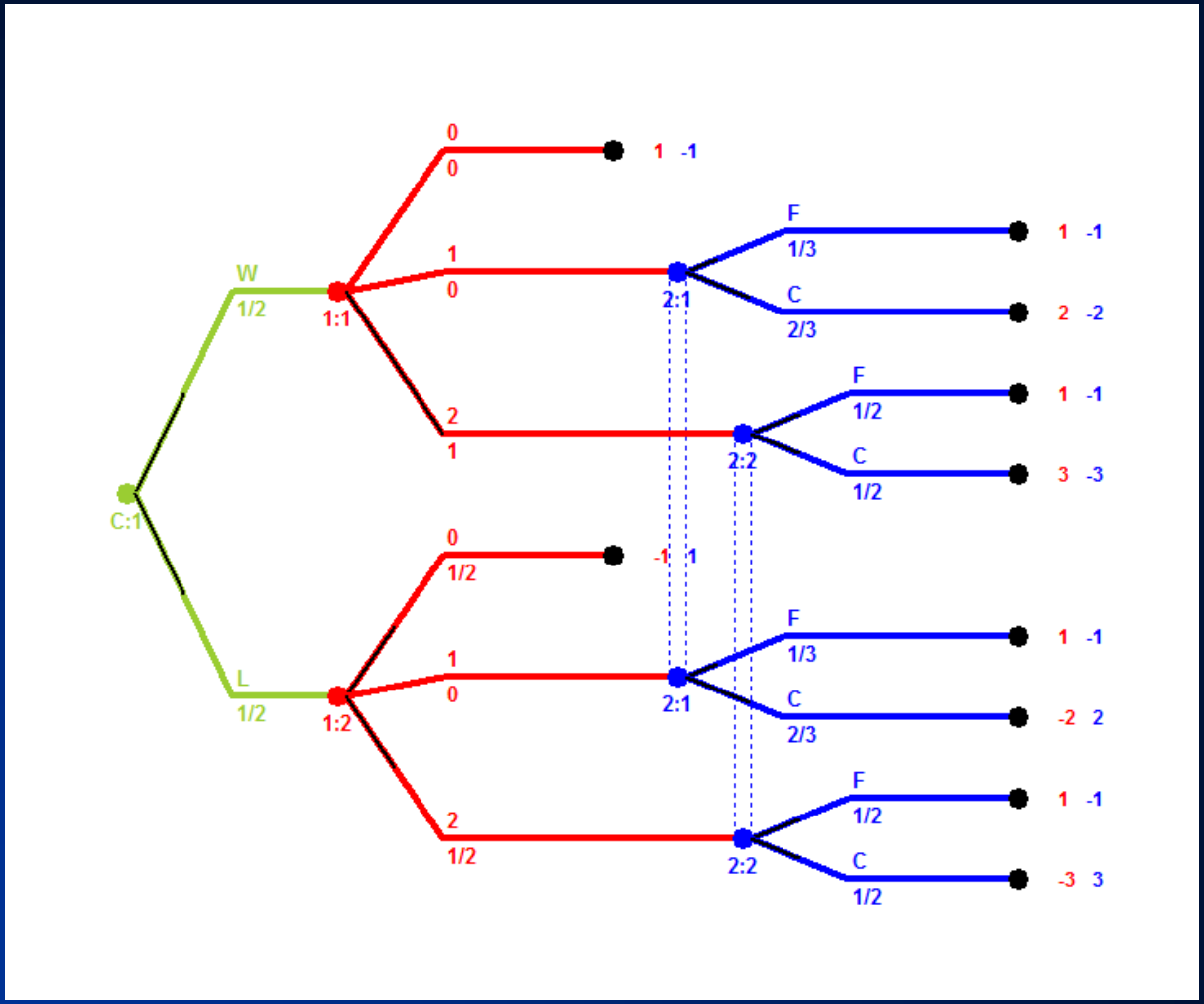
- In many important games, there is information that is private to only some agents and not available to other agents
  - In auctions, each bidder may know his own valuation and only know the distribution from which other agents' valuations are drawn
  - In poker, players may not know private cards held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

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.

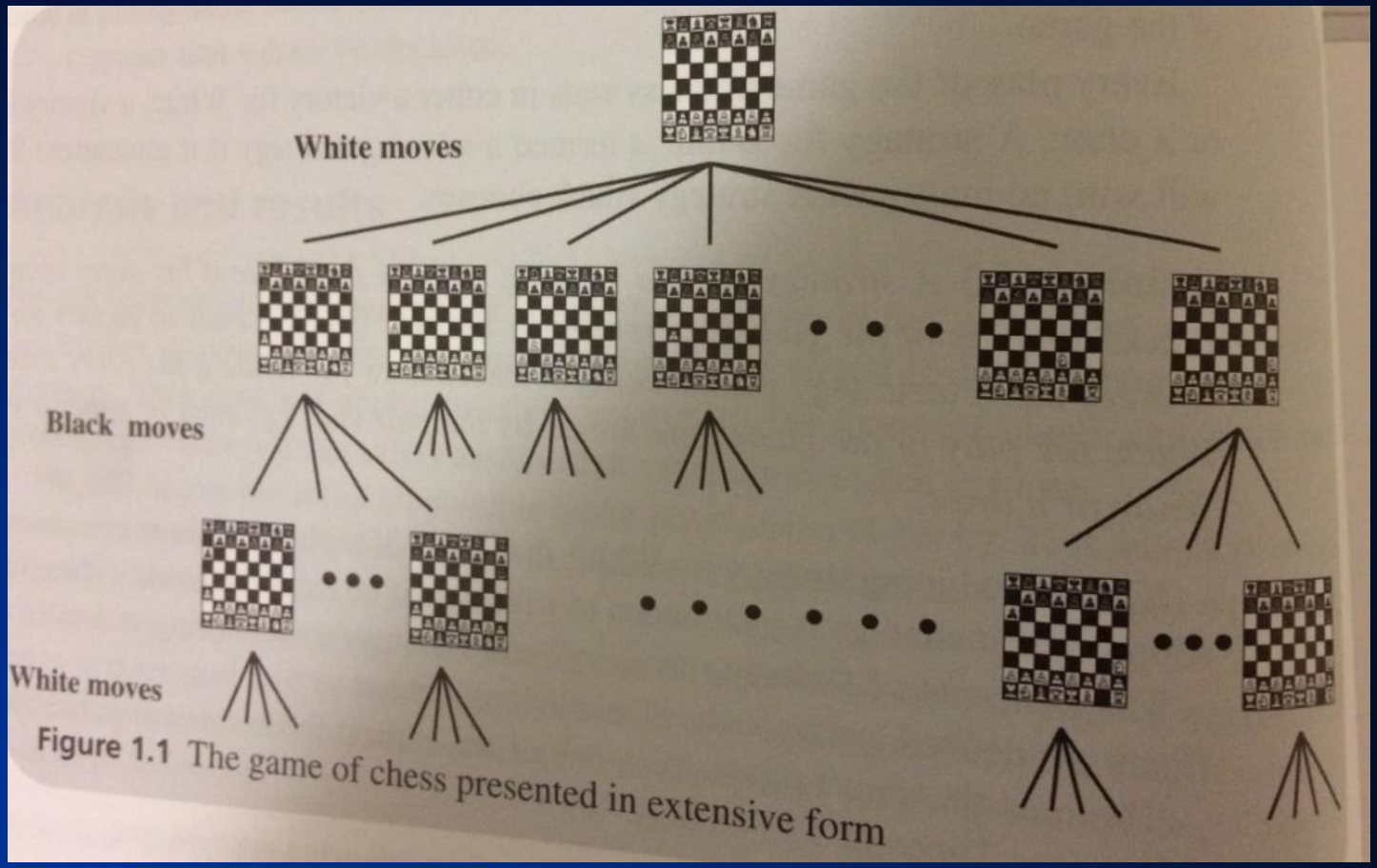


Figure 1.1 The game of chess presented in extensive form

- Every description of a game must include:
  - Set of players
  - The possible actions available to each player
  - Rules determining the order in which players make their moves.
  - A rule determining when the game ends.
  - A rule determining the outcome of every possible game ending.

Homework for next class

- Chapter 21 from Russel/Norvig
- HW3 due Tuesday 11/14
- HW4 out this week
- Next lecture: Machine learning (classification)