

Bookkeeping

- Today's class
 - What's on the midterm?
 - A quick review of topics covered
 - An overview of the class project
 - More project brainstorming
- I cannot speak above a whisper, so I cannot lecture. 😞 Please read the lecture notes I have typed up!

I tried exploring using text-to-speech to speak these notes aloud for the classroom, and it was just incredibly annoying. Part of what was scheduled for class is a quick overview of the midterm—what kinds of topics and questions you should expect – and for that I have done my best to type up notes describing what I would have said during lecture.

I will answer questions on Discord during office hour and lecture hours (only). The rest of the time, Discord is for you to use for team formation, discussions, etc.

What Will the Exam Be Like?

- Closed book, no calculator needed
- Broadly:
 - Turn a problem description into a formulation
 - Work through a problem to reach a solution
 - Demonstrate a conceptual grasp of the material
- Be able to go from concepts to/from algorithms and implementations
- Basic idea: you need to **understand the ideas** behind the material we have covered, and be **able to apply** them to solving problems.
- **Generally** easier than the homeworks (but please don't get complacent)

Broadly speaking, the exam is pretty standard—it's closed book, and the goal is to demonstrate mastery of material from the lectures **and** from the readings. At a high level, you'll be asked to demonstrate a conceptual grasp of the material; solve problems; and in some cases, turn a problem description into a formulation, so like for example, "How would I formulate this description as a search problem" would be the kind of thing you might see. I care a lot about whether you're getting the underlying concepts, and less about whether you can write Python on the fly or do arithmetic. You won't need a calculator, since nobody owns one and you can't use your phone during the exam. Generally, the exam questions will be either easier than the homeworks or, for simpler homework problems, equivalently easy.

What Kind of Questions?

- T/F, multiple choice, fill in the blank
- Work through an {algorithm | solution type | problem}
- Draw something – search trees, states, Bayes nets, paths through a map, ...
- Write a **short** answer to English questions
 - E.g.: “What approach would you use to solve this problem?”
 - E.g.: “We know these are independent. Why?”
- Write a **medium length** essay (half a page or less)
- Write a **short** Python function that performs a task (but see next slide)

Some kinds of questions that may appear include standard exam questions like fill in the blanks or multiple choice, but most of the exam credit will be in meatier question types like “work something through”—the homeworks are good examples of that; or something where you do something that requires more thought, like drawing a tree or network, writing a short essay, things that require a slightly deeper engagement with the material. You may be asked to write a short function that demonstrates knowledge of how an algorithm works; however...

What Do I Need To Do?

- Small coding questions (we don't care about minor syntax mistakes, etc.)
 - We're looking for "I understand this well," not "I know Python well"
 - Please don't study Python
- Look at **homeworks**
- Look at **sample problems** in lectures
- Look at lectures' "Why?" questions

...if there are coding questions in the exam, they will be intended to capture whether you know how something works, not whether you can write Python on the fly. So please don't focus on programming practice.

If you're looking for sample problems to study from, homeworks are a good source; however, homeworks are not exhaustive—we can't cover all the topics in depth and have homeworks that take a sane amount of time. So look at the questions in the lectures, as well. In the lecture slides, there are a lot of sort of small sample problems and places that I ask a question, like "Why is this the case?" These can be useful to study.

Scoring

- Follow directions.
- You start with a perfect score that is marked down for mistakes
 - What this means: If I ask for 2 examples, and you give 3, one of which is wrong, it's -1/2
- Read carefully.
 - You have time – my exams aren't super long
 - "I didn't see the part that said..." ← 😞
- You will need to know the terminology – we will not define things you should know to understand questions

Just as a note, don't try to acquire points by giving a ton of examples or answers; if I ask for two answers and you give me four, you will lose points for the wrong ones, not gain ones for the right ones. So don't try to scrape up points by filling up the page—instead, focus on exactly answering the specific questions asked. This also leads into the next point, which is please to read the instructions carefully. You will have time, my exams are not intended to be painfully long, and it's always really sad to lose points because you didn't read the exam instructions carefully. Also, we do talk a lot about terminology in this class, and you will need to know that terminology—during the exam we won't define terms that take more than a one-sentence answer, since that's more like teaching during the exam.

Topics: AI

- What is intelligence?
- What is AI?
- What is it used for? Good for?
- What are the tasks, targets, and goals?

Okay, so what have we talked about this semester so far? At a high level, we've talked a lot about intelligence, and about AI. We've talked about some of the problems we want to target using AI technology, and some of the goals of the field. You may need to be able to speak coherently about goals for AI, or what makes something an AI problem, for example.

Topics: Agents

- Agents
 - What kinds are there?
 - What do they do?
 - How do we characterize them (what traits do they have)?
 - Autonomy, rationality, ..?
 - How do they interact with an environment?
- Environments
 - What's an environment?
 - How is it characterized?

We started addressing these questions by defining agents—software or even robots that do something. Agents are kind of the core mechanism for demonstrating intelligent action. And in the space of agents, we talked about how to categorize them, what they are for, what traits they have like performance or actions they can take. And as well as their traits, we've talked about the environments in which they can operate, mostly in the context of looking at a significant number of examples.

Topics: Search

- What is it for?
- Elements of a search problem
 - State spaces, actions, costs, ...
 - How do state spaces pertain to search?
 - To problem-solving?
- Exploring search space (selecting, expanding, generating, evaluating)
- Specific algorithms: How do they work? What are they good for? What are their weaknesses?

We spent a lot of time on search, and we did that for a reason. Many, many AI problems, ranging from machine learning in large language models all the way down to tic tac toe, can be considered as search problems. You need to know what a search problem is, how to characterize one, and how to formalize a search problem in terms of the problem you're trying to solve. We covered a lot of search algorithms, which you should be familiar with. So let's look at some specifics...

Topics: Formalizing Search

- What are the elements of a search problem?
 - “Express [X] as a search problem.” What does that mean?
- **States:** every state a puzzle can be in
- **Actions/Operations:** how you get between states
- **Solutions:** you need a goal test (and sometimes a heuristic, or estimate of distance from goal)
 - Sometimes we care about path (planning), sometimes just goal (identification). Can you say which, for a given problem?
- **Costs:** not all solutions or actions are equal


So how do we formalize search? Well, what are the elements of a search problem? Broadly speaking, problems have state spaces, that is, a representation of what the problem can look like at different stages in the course of operating on it; actions an agent can take, which usually cause a change of state, and which may have associated costs; and solution states, which means we need a goal test to see if a state is a solution, and sometimes an estimate of how far a state is from a goal state. And both actions and solutions can have costs. So a pretty common question type might be a description of a problem, like a game or a navigating robot or something, for you to formulate as a search problem – describing its state space and possible operations and so on.

Topics: Uninformed Search

- Why do uninformed search?
- Come up with some examples of uninformed search problems
- Important algorithms: BFS, DFS, iterative deepening, uniform cost
- A likely question: “What would be the best choice of search method for *[problem]*, and **why?**”
- Characteristics of algorithms
 - Completeness, optimality, time and space complexity, ...

We’ve done three broad classes of search problem: Uninformed search is the first, where you don’t know anything too specific about the problem space, you just know how to go from state to state – that is, you know what actions you can take and what each action changes in the state space. And the slide has examples of the algorithms that are most important for this – some of them, like breadth-first search and depth-first search, are pretty simple, while uniform-cost search is capable of taking things like action costs into account. And it’s important to be able to answer questions like why something is a good algorithm for a particular problem, not just regurgitate the search definition.

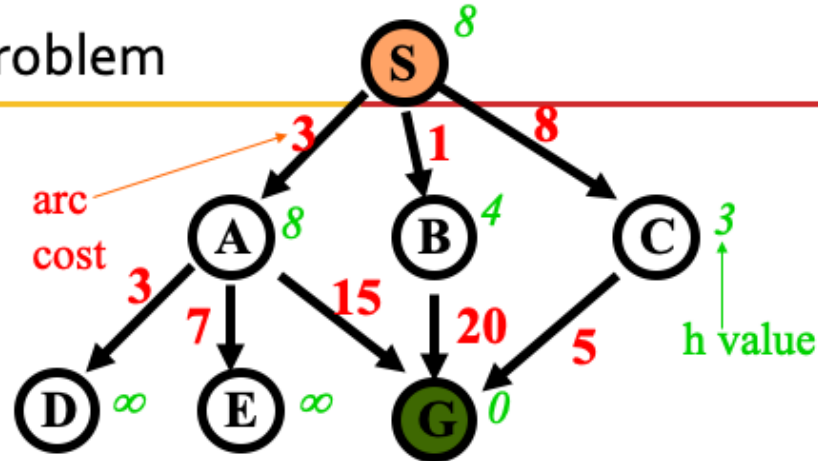
Topics: Informed Search

- Some external or pre-existing information says what part of state space is **more likely** to have a solution
- **Heuristics** encode this information: $h(n)$ 
 - What does $h(n) = 0$ mean?
- Admissibility & Optimality
 - Some algorithms can be optimal when using an admissible heuristic
- Algorithms: best-first, greedy search, A*, IDA*, SMA*
- What's a good heuristic for a problem? Why?

A heuristic applies to a **node/state**, and can give optimal solution with the right **algorithm**

Informed search, by contrast, is a kind of search where you have some idea “where” to look for a solution – that is, given a state space, you have a heuristic you can apply to that state space that says, roughly, whether it's closer to a solution than another state space. And this heuristic – which may be *admissible*, and important concept to understand – enables a new set of search algorithms, of which A* is the core one to know. And again, you may need to be able to speak to why something is a good search approach.

Sample Problem



Apply the following to search this space. At each search step, show: current node being expanded, $g(n)$ (path cost so far), $h(n)$ (heuristic estimate), $f(n)$ (evaluation function), and $h^*(n)$ (true goal distance).

Depth-first search
Uniform-cost search

Breadth-first search
Greedy search

A* search

So you may recall this search tree vaguely from the lectures. It has everything we need to do informed search – action costs and heuristic values at each node. So I would recommend making sure this makes sense and that you remember how to do different kinds of search over something that looks like this. And to bring it back to what we were talking about earlier, it's also important to be able to talk about how the nodes (states) and arcs (actions) tie into the definition of a search space.

Topics: Local Search

- Idea: Keep a single “current” state, try to improve it
 - Don’t keep path to goal
 - Don’t keep entire search in memory
 - Go to “successor states”
- Concepts: local maxima/minima, random restarts
- Important algorithms: hill climbing, local beam search, simulated annealing

The last high level search approach we talked about is *local search*, where we don’t care about the path to a goal, but rather about the goal state itself. (Our canonical example is the n -queens problem, where it doesn’t matter what sequence of moves got you to a particular board state, only what that state is.) This enables some much more memory efficient approaches, but also means you can get stuck in a loop in the search space, or stuck at a local optimum instead of the global optimum. And for this it’s important to understand local optima and some of the approaches we use to find a solution, like hill climbing and simulated annealing.

Topics: Search

- How many states are there?
- What operations fully encode this search problem?
 - That is: how can you reach every state?
- Are there loops?
- How many states does pure DFS visit?
 - If there are loops?
- What's a good algorithm? A bad one? (For a specific problem?)

Here are some (non-exhaustive!) examples of the kinds of questions I might ask about a particular search problem.

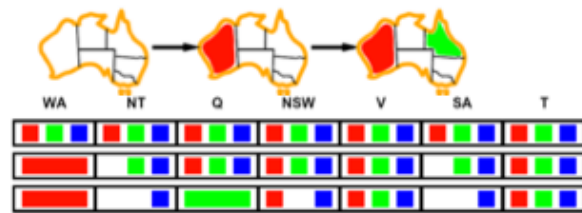
Topics: CSPs

- Constraint Satisfaction: subset of search problems
 1. **State** is defined by **random variables** X_i
 2. With values from a **domain** D
 3. Knowledge about problem can be expressed as **constraints** on what values X_i can take
- Special algorithms, esp. on constraint networks
- **How would you express [something] as a CSP?** As a search? How would you represent the constraints?
 - E.g.: “Must be alphabetical”

Our next major topic after search was constraint satisfaction problems, which are a particular constrained kind of search problem. And you can handle CSPs by defining the state as a set of variables that you are trying to find values for, and which have some constraints on their possible values. So the first thing you need to know is, how might you express a problem as a CSP? And everyone did fine on that question for the homework.

Topics: Constraint Networks

- **Constraint propagation:** constraints can be propagated through a constraint network
 - Goal: maintain **consistency** (constraints aren't violated)
 - Concepts: Variable ordering, value ordering, fail-first
 - Important algorithms:
1. **Backtracking:** DFS, but at each point:
 - Only consider a single variable
 - Only allow legal assignments



We went through a few different ways of solving CSPs, where the basic algorithm is backtracking, and forward checking is a variation on the backtracking algorithm, but where the search space is reduced by using local consistency checking.

Topics: Games

- Why play games? What games, and how?
- Characteristics: zero-sum, deterministic, perfect information (or not)
- What's the search tree for a game? How big is it?
- How would you express game $[X]$ as a search? What are the states, actions, etc.? How would you solve it?
- Algorithms: **(expecti)minimax, alpha-beta pruning**
 - Many examples on slides

Now, we've talked about games a lot as part of search, but the actual formalization of game playing matters, too. And in talking about games, we talked about characteristics games can have—like being fair, being zero sum, being deterministic versus stochastic, and so on. But a lot of what we talked about is how to do a search tree for adversarial games, which might or might not have a chance element. So as a thought experiment, how would you draw a three player game as a search tree? What if it had a dice roll associated with it? And you should know, for example, how to draw an expectiminimax tree, or how alpha-beta pruning works.

Topics: Basic Probability

- What is uncertainty?
- What are sources of uncertainty in a problem?
 - Non-deterministic, partially observable, noisy observations, noisy reasoning, uncertain cause/effect world model, continuous problem spaces...
- World of all possible states: a complete assignment of values to random variables
- Joint probability, conditional probability

We spent half a lecture revisiting basic concepts in probability, and in justifying the need for probabilistic reasoning. And a lot of what we've done since then revolves around understanding joint probabilities (e.g., $P(A,B)$) or conditional probabilities (e.g., $P(A|B)$).

Topics: Basic Probability

- Independence: A and B are independent
 - $P(A) \perp\!\!\!\perp P(B)$ iff $P(A \wedge B) = P(A) P(B)$
 - A and B do not affect each other's probability
- Conditional independence: A and B are independent given C
 - $P(A \wedge B \mid C) = P(A \mid C) P(B \mid C)$
 - A and B don't affect each other if C is known

We talked about independence, and the different ways two nodes in a graph can be independent, or have a dependency induced by observing another variable, or be made independent by observing another variable.

Topics: Basic Probability

- $P(a | b) = \frac{P(a \wedge b)}{P(b)}$
- $P(a \wedge b) = P(a | b) P(b)$

$P(\text{smart} \wedge \text{study} \wedge \text{prep})$	smart		$\neg \text{smart}$	
	study	$\neg \text{study}$	study	$\neg \text{study}$
prepared	.432	.16	.084	.008
$\neg \text{prepared}$.048	.16	.036	.072

- What is the prior probability of smart?
- What is the conditional probability of prepared, given study and smart?
- Is prepared independent of study?

So make sure you understand both the terminology of basic probability and the examples we worked through in class of conditional probabilities, and understanding how this kind of probability table works, because it's foundational to...

Topics: Probabilistic Reasoning

- Concepts:
 - Posteriors and Priors; Bayesian Reasoning; Induction and Deduction; Probabilities of Events
 - [In]dependence, conditionality, marginalization
- What is Bayes' Rule and what is it useful for?

$$P(H_i | E_j) = \frac{P(E_j | H_i)P(H_i)}{P(E_j)}$$

$$P(cause | effect) = \frac{P(effect | cause)P(cause)}{P(effect)}$$

...reasoning about probability and using Bayes' rule to reason about evidence and hypotheticals. And you don't need to memorize much for this exam, but you should know Bayes' rule and how it works.

Topics: Joint Probability

- What is the **joint probability** of A and B?
- $P(A,B)$
- The probability of any set of legal assignments.
- Booleans: expressed as a matrix/table

	alarm	¬alarm
burglary	0.09	0.01
¬burglary	0.1	0.8

≡

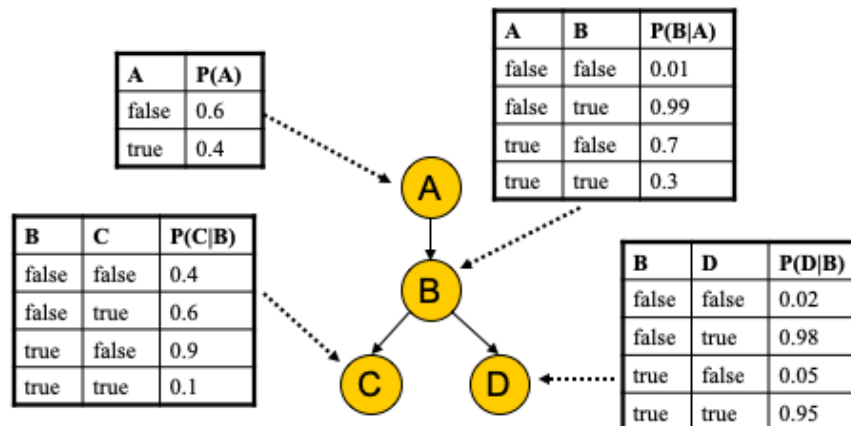
A	B	
T	T	0.09
T	F	0.1
F	T	0.01
F	F	0.8

- Continuous domains → probability functions

So make sure you understand joint probability, and how joint probabilities are expressed as conditional probability tables, or CPTs...

Conditional Probability Tables

- For X_i , CPD $P(X_i | \text{Parents}(X_i))$ quantifies effect of parents on X_i
- Parameters** are probabilities in conditional probability tables (CPTs):



Example from web.engr.oregonstate.edu/~wong/slides/BayesianNetworksTutorial.ppt

...and how that applies to **Bayesian networks**. Bayesian Belief Networks basically took up three lectures, because reasoning under probabilistic or uncertain conditions is a core part of artificial intelligence as applied to a huge variety of real world problems.

Simple Bayesian Network



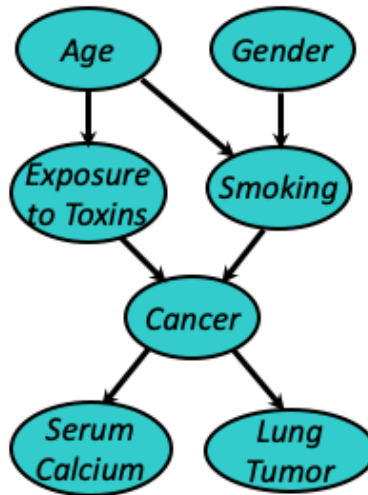
$P(S=no)$	0.80
$P(S=light)$	0.15
$P(S=heavy)$	0.05

Smoking=	no	light	heavy
$P(C=none)$	0.96	0.88	0.60
$P(C=benign)$	0.03	0.08	0.25
$P(C=malign)$	0.01	0.04	0.15

Slides from Dr. Oates, UMBC

So given some examples of a Bayesian network, can you reason about probabilistic statements?

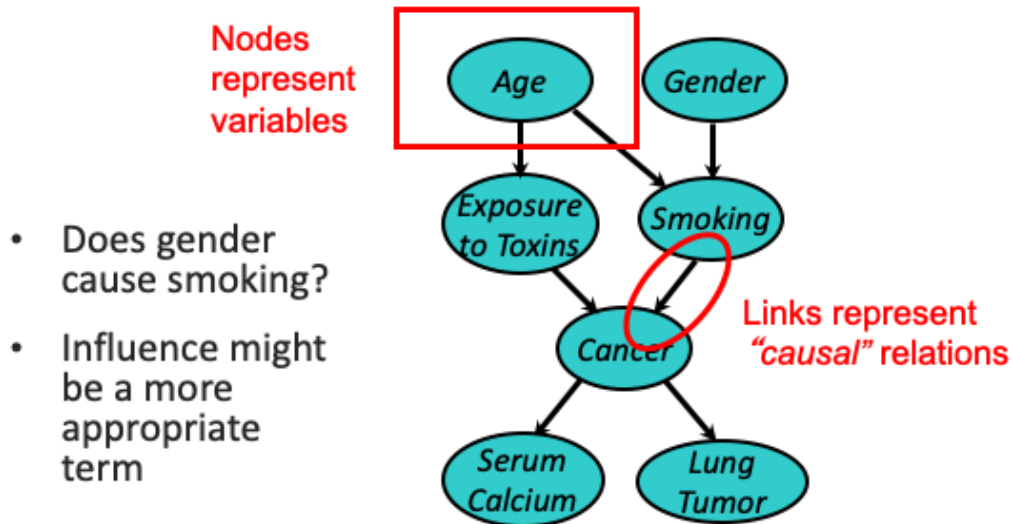
More Complex Bayesian Network



Slides from Dr. Oates, UMBC

Bayes' nets can be arbitrarily complex, although obviously the complexity of what you can reason over during an exam time is limited. However, things like understanding independence makes it much more feasible.

More Complex Bayesian Network



Slides from Dr. Oates, UMBC

So make sure you really understand how a BBN works. A common question I might ask is, given a set of factors in the world, draw a BN and justify each element of it—that exact question may not appear, but that level of understanding of where BNs come from is necessary.

Independence

Age

Gender

Age and Gender are independent.

$$P(A, G) = P(G) * P(A)$$

$$P(A | G) = P(A)$$

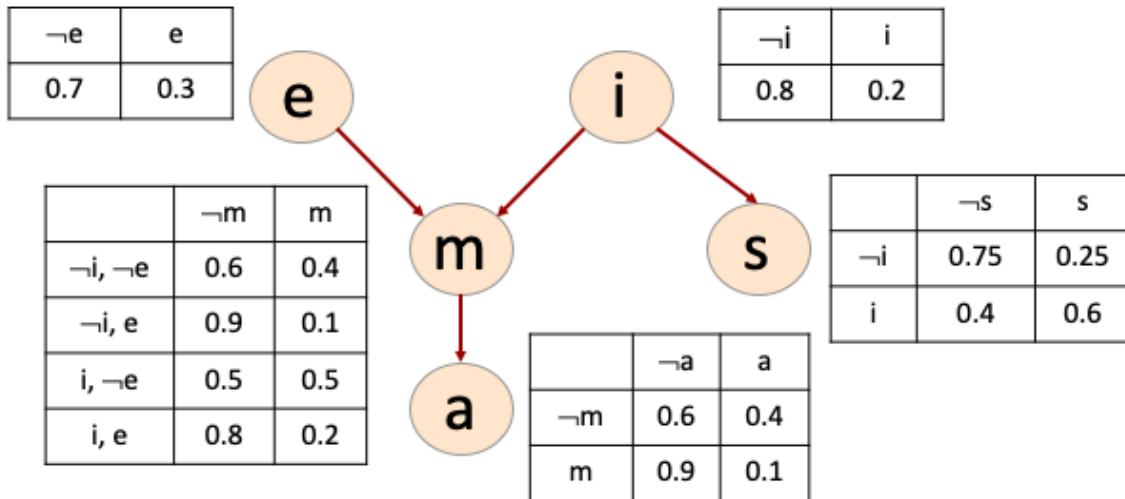
$$P(G | A) = P(G)$$

Slides from Dr. Oates, UMBC

And again, independence really matters for keeping reasoning tractable. So here are some statements pertaining to independence—if these are true, A and G are independent.

Review: Bayes' Nets

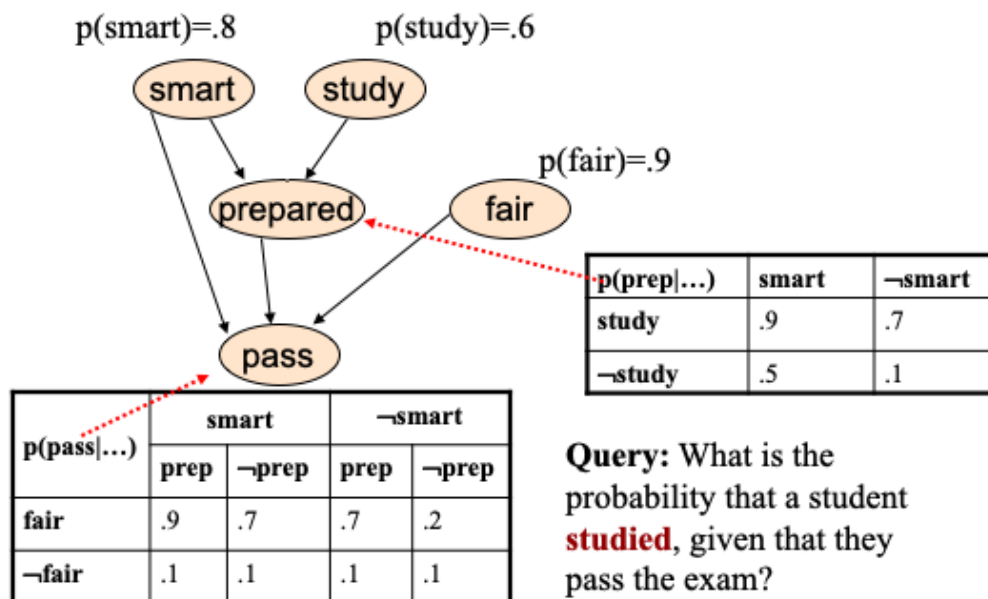
- $P(a, m, i, e, s) = P(a \mid m) * P(m \mid i, e) * P(i) * P(e) * P(s \mid i)$



www.upgrad.com/blog/bayesian-network-example/

So remember this network? We did some examples of reasoning over it, and this is the kind of thing you should be able to answer questions about—everything from independence, to the probability of a statement of probability like “What is the probability of A given not-E”.

Exercise: Variable Elimination



Here's another way of writing the smart/studied/prepared problem case, but this time as a BN and with associated values. Can you work out what this BN says and how to answer questions about it?

Topics: Reasoning Under Uncertainty

- How is the world represented over time?
 - Concepts: timesteps, world, observations
 - Transition model captures how the world changes
 - Sensor model capture what we see, given some world
 - Markov assumption (first-order) makes it all tractable
- What can we do with it?
 - Concepts: Filtering, predicting, smoothing, explaining

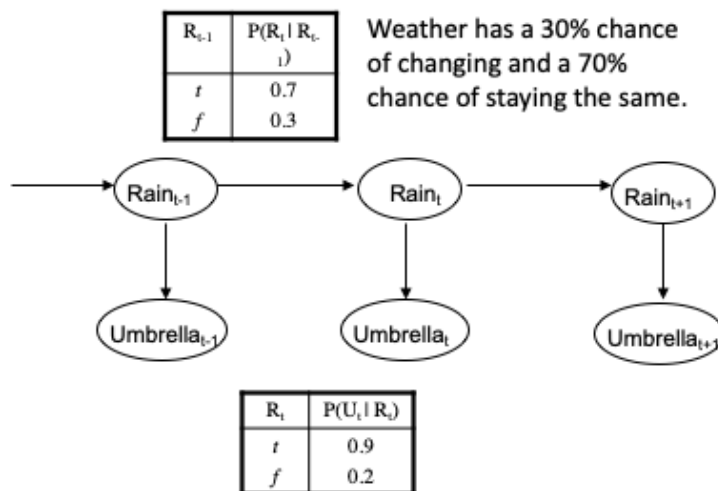
From Bayes' nets we went on to reasoning over time and reasoning over uncertain observations. And one of the core ideas here that you should hang onto is the idea that the world is split into *timesteps*, or a series of sequential states where at each step we have evidence about the state of the world (which we call observations), and hidden variables where we don't know their state, but can reason about it based on the evidence we do have. And this enables a whole suite of capabilities.

Topics: Reasoning Under Uncertainty

- How would you represent this problem as a network and set of conditional probability tables?
 - The weather has a 30% chance of changing and a 70% chance of staying the same.
 - If it's raining, the probability of seeing someone carrying an umbrella is 90%; if it's not raining, it's 20%.
- I saw umbrellas Monday and Tuesday, but not today. What is the most likely weather pattern for those days?

So here's an example of what I mean when I say you might need to be able to formulate a problem. And we did formulate this exact problem in class, as a Markov chain.

Example



BBN from: www2.isye.gatech.edu/~yxie77/isye6416_17/Lecture6.pdf

And it looks roughly like this. And make sure you understand why CPTs don't add up to one as written. It's just because we're simplifying what we write down, because we know they have to add up to one. So here are the full explanations of the BN shown:

Full matrix for R

0.9 0.1
0.2 0.8

if it's raining, the probability of carrying an umbrella given that value of rain is .9

if it's NOT raining, the probability of carrying an umbrella given that value of rain is .1

if it's raining, the probability of NOT carrying an umbrella given that value of rain is .2

if it's NOT raining, the probability of NOT carrying an umbrella given that value of rain is .8

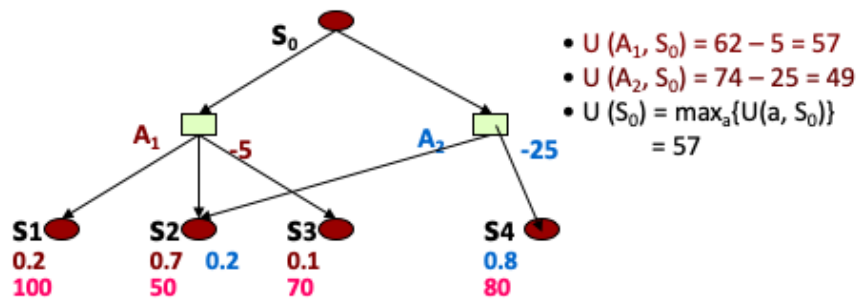
Topics: Utility

- How should rational agents make decisions?
- Concepts: rationality, utility functions, value functions, expected value, satisficing, preferences
- Utility is a function of world states
- Must have some preferences that pertain to perceived needs or wants

But in order to make meaningful choices, we need to take into account that what exactly we need (or want) is probably not fixed; instead it is more complex. So this leads us into utility, and the whole question of decision theory, which describes how a so-called rational agent should make decisions. And one thing to remember about utility proper is that it is a function over states the world can be in, not over actions or agents or anything else.

Topics: Decision Theory

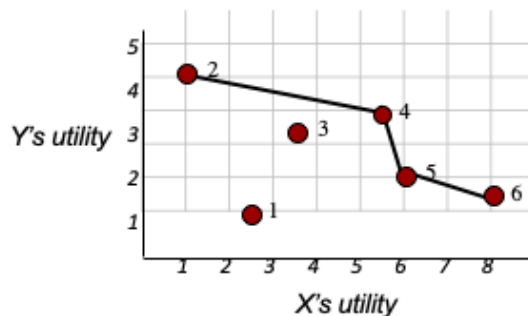
- What is the expected utility of an action?
 - Broadly: its probability times its value
 - The sum of that for all possible outcomes
- Maximum Expected Utility principle



And to reason about the utility of states as a search tree, we can express the possible states an agent can reach as nodes on a tree (which is not new), and associate those nodes with the actions that get us there, the probability of a set of possible outcomes, the utility of the eventual states, and the costs of taking actions. And that lets us make decisions about what actions to take, based on the *maximum expected utility*.

Topics: Pareto Optimality

- An outcome is Pareto optimal if there is no other outcome that all players would prefer.
- S is a Pareto-optimal solution iff
 - $\forall s' (\exists x U_x(s') > U_x(s) \rightarrow \exists y U_y(s') < U_y(s))$
 - I.e., if X is better off in s' , then some Y must be worse off



Example questions:

Which solutions are Pareto-optimal?

Which solution(s) maximize global utility (social welfare)?

Game theory also has a few ways of describing states in the world as it pertains to two or more players' preferences. And the two big ones we talked about are joint optimality and equilibria. Pareto optimality is one way in which the world can be 'optimal' given the combined needs of two players—it's a state where there is nothing state that represents an improvement for *all* players; that is, some other node that everyone would agree to move to because it increases that player's utility.

Topics: Nash Equilibrium



- Occurs when each player's strategy is optimal, **given** strategies of the other players
- No player benefits by **unilaterally** changing strategy while others stay fixed
- Example questions:
 - What strategy should you choose? Why?
 - What strateg(ies) are in a Nash equilibrium?

		<i>C</i>	
		Confesses	Denies
<i>B</i>	Confesses	(3 , 3)	(0, 5)
	Denies	(5, 0)	(1 , 1)

Whereas an equilibrium is a state where no player is willing to be the first to change their strategy, because they will not benefit from unilaterally changing, once they have chosen a strategy. And you should be able to identify Nash equilibria and Pareto-optimal states, but you should also be able to use those to justify answer to questions about “what strategy should player X actually use?”

Various Reminders

- **Everything in the readings** is fair game.
- Go over the slides — I wouldn't have lectured about it if I didn't think it was important.
- Look at the additional writeups I have posted to the schedule covering, e.g., filtering and variable elimination.
- Look at homeworks, sample problems in lectures.
- Look at lectures' "Why?" questions.
- Slides are a good source of **conceptual** understanding.
- Book goes into **detail** and explains more deeply.

Expect the problems on the exam to be similar in scope to those from the homeworks, but remember the homeworks only sample from the space; everything we've covered in class could appear. And please use the lecture slides as a resource for higher-level, conceptual understanding of the material, while the book goes into much more depth.