tHey y’all, I made a Slack for some faster communication about questions and such:

JOIN HERE

Topics done after midterm:

- Markov Decision Process & Reinforcement Learning
- Bayes Nets
- Dynamic Bayes Nets
- Partially Observable MDPs
- Neural Nets
    - https://www.youtube.com/watch?v=aircAruvnKk
- Decision Trees
    - Can someone explain the + and - notes the professor wrote in OneNote? Thanks!
- Ethics

3600 Final Exam Study Guide

Table of Contents

Ethics

Societal Implications for AI

- Technology will follow given rules in a neutral and unbiased manner. Humans who make the rules, however, are not neutral and are biased.
- AI: we are handing off decision-making to a computer
    - Questions we should ask ourselves:
    - How do the decisions from the AI affect people? For example, if an AI chooses a song you don't like, it doesn't affect you as much. However, if a company uses an AI to determine who is a good fit for the job, it would affect you a lot (especially if there's a bug in the system)
    - Also, what should we do if the AI makes a wrong decision?
    - Are certain groups of people disadvantaged or treated differently by the AI?
    - Do people (especially the people who use AI) know how decisions are made?
    - Is technology used to avoid accountability? (e.g., "it's not on me that you weren't hired, it was the AI's decision")
- AI doesn't have to be perfect, since humans make mistakes too
    - It's not bad if technology fails, as long as there is a remedy

Four things we Talked About

1. Fairness – is the data unbiased, where is it from, how is it collected?
2. Accountability – remedy; who is responsible if errors are made?
3. Transparency – how do we know how decisions are made?
4. Ethics – what applications should we consider

Biased Data = Prejudicial Bias

2015 – Gorillas
- An AI went through social media and tried to identify images
- If the image was a cat, it said it was a cat
- If the image was a white man, it said it was a white man
- But if the image was a black man, it said gorilla
- That's because fewer darker skinned people upload pictures of themselves on social media, so the AI didn't have enough data to work with to know what a black man is

2017 – AI Sentences

- This is a project that Dr. Riedl made
- He had an AI parse through the writings of an author (I think Ernest Hemingway), so it can make paragraphs similar to the author's writing style
- In the author's writing, he uses the phrase "and then he died" a lot
- So the AI just returned a paragraph "and then he died" over and over again

2018 – Amazon Hiring

- Amazon tried to make an AI to determine which candidates they should hire based on their resumes
- They removed the "gender" category so the AI can't discriminate between men and women
- But the AI still discriminated against women, why? I don't know, you tell me
- It's because of *secondary characteristics*, for example, women are more likely to list their hobbies on their resume, and they tend to have more feminine activities
- The AI found this trend and made its decisions based off of it

2018 – Criminal Face Detector

- This was really bad
- There was a company that claimed that they have an AI that can determine who is a criminal based on their picture
- But their data was skewed
    * "good" people were pictures of CEOs, so they were dressed nicely and had proper lighting, they were smiling
    * "bad" people were pictures of mugshots, so people were not dressed nicely, there was dim lighting, and they were not smiling

2018 – Google Translate

- Turkish doesn't have pronouns like he and she
- So if you translate "she is a doctor" from Turkish to English in google translate, it returns "he is a doctor"

UPDATE : As per Professor's announcement, this issue was fixed recently

Here are some biases that AI pick up, here is a chart

FEMALE	MALE
Nurse	Doctor
Lazy	Entrepreneur
Beautiful	Hard-working
	Strong

Why do we get these Biases?

1. Data is generated by humans (it can be an innocent mistake)
2. Pattern matching is lazy (i.e., AI Sentences example)
    a. The AI finds the first pattern and goes with it

How to prevent these Biases?

1. Know the provenance – where did the data come from, how was it collected?
2. Restrict data – blank out unnecessary data (e.g., gender)
3. De-bias our data – balancing our data
4. Set rules that override the AI
5. Red team – have someone to purposely make the AI make bad decisions, so people can fix it (kind of like quality assurance)

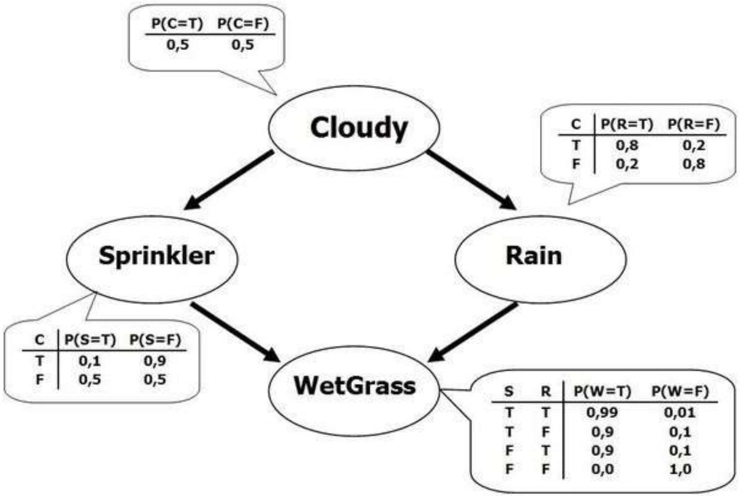
Liability: who is responsible in the eye of the law

1. AI – No, it doesn’t make sense, they are just code and anyways, they can’t give you money
2. Engineer – most rules are leaning towards “no”
3. Company or CEO (I think the European Union say that they are liable)
4. User; terms of service (but I think they are getting away from this because no one reads the terms of service)

- If a seatbelt is made according to standard and passes all of the safety tests, then according to law, the manufacturer can’t be sued if their seatbelts don’t work since they built it to standard
- So maybe there might be a standard set for AI, who knows

Bayes Nets

-infer the probability that the agent is in some particular state. Actions are not considered
-Based off of Bayesian Statistics
-A Bayesian network is a directed acyclic graph in which each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable. Formally, if an edge exists in the graph connecting random variables A and B, it means that P(B|A) is a factor in the joint probability distribution, so we must know/compute P(b|A) for all values of B and A in order to conduct valid inference.

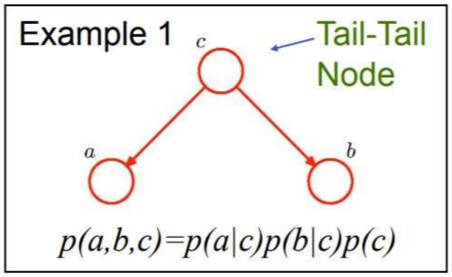


P(C=T)	P(C=F)
0,5	0,5

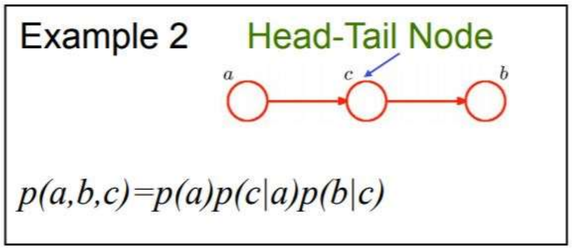
C	P(R=T)	P(R=F)
T	0,8	0,2
F	0,2	0,8

C	P(S=T)	P(S=F)
T	0,1	0,9
F	0,5	0,5

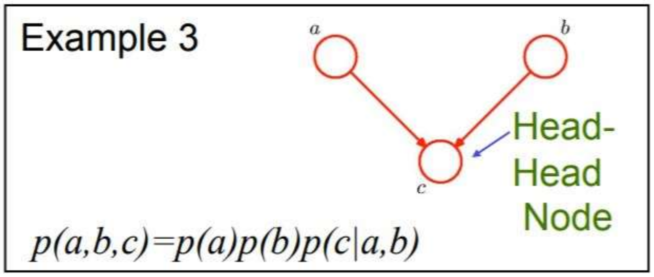
S	R	P(W=T)	P(W=F)
T	T	0,99	0,01
T	F	0,9	0,1
F	T	0,9	0,1
F	F	0,0	1,0



$p(a,b,c)=p(a|c)p(b|c)p(c)$



$p(a,b,c)=p(a)p(c|a)p(b|c)$



$p(a,b,c)=p(a)p(b)p(c|a,b)$

Dynamic Bayes Nets

-infer the probability that the world will be in a particular state given time. Actions are not considered.

- No one actually knows that these are..I assumed they were just magic

Partially Observable MDPs

-figure out the best action given sensor-stochastic and action-stochastic environment. POMDP = MDP + Dynamic Bayes Net.
-POMDPs are used for action stochastic, partially observable environments
-A sensor-stochastic environment can be action-deterministic or action-stochastic. Typically if the sensors are stochastic the environment is also action-stochastic. But with sensor-stochasticity even if actions were deterministic, the agent T300 wouldn't know with 100% confidence what state it was in, so actions might as well be stochastic for all practical purposes.

A POMDP could be used for action-stochastic or action-deterministic environments, but there must be partial observability. Partial observability can be achieved with sensor stochasticity, but can also result from other things like sensor limitations not being able to sense all parts of the world.

In a POMDP we may or may not know our starting state. The starting state would be expressed as a belief—a probability distribution over all states. The distribution can be uniform (equally likely to be in any state) or based on a strong prior (high or even 100% certainty about starting state)

Neural Nets

- Identify patterns in sample data and use these pattern to predict appropriately when given similar situations
- Approximate a large number of functions given a dataset in the form <x,y> (functions in the form of stacks of weighted fully connected bigraphs)
- Soft threshold (sigmoid perceptron): logistic/sigmoid function: S(x) = 1/(1+ e^{-x})
- Hard threshold (perceptron): binary output (1/0)
- Note: perceptron: single-layer feed-forward neural network
- **Feed-forward network**:
    - connections only in one direction (directed acyclic graph)
    - Receives input from *upstream* nodes and delivers output to *downstream* nodes
    - No loops
    - Represents a function of its current input (no internal state other than weights)
    -
- Recurrent network:
    - Feeds outputs back to its own inputs
    - Activation levels of network form a dynamical system that may reach stable state, exhibit oscillations or chaotic behavior
    - Can support short-term memory

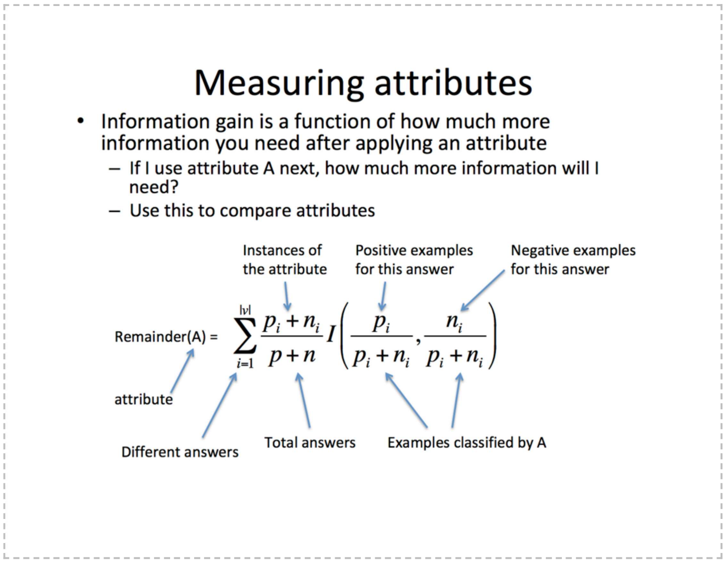
Decision Trees

Information Function

$$I(P(v_1),\ldots,P(v_n)) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

Where v1 - vn are the possible outcomes

Calculating Remainder:



Measuring attributes

- Information gain is a function of how much more information you need after applying an attribute
  - If I use attribute A next, how much more information will I need?
  - Use this to compare attributes

$$\text{Remainder}(A) = \sum_{i=1}^{|v|} \frac{p_i+n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

Instances of the attribute
Positive examples for this answer
Negative examples for this answer
attribute
Different answers
Total answers
Examples classified by A

Gain(Attribute) = Information(Parent Attribute) - remainder(Attribute)
Range of information is [0,1]
For completely homogeneous dataset (all TRUE or all FALSE values): entropy is 0
If the dataset is equally divided (same amount of TRUEs and FALSEs): entropy is 1

A BRANCH WITH ENTROPY MORE THAN 1 NEEDS SPLITTING !!!

- Root node has the maximum information gain (entropy reduction)
- Leaf nodes have entropy 0

Lower remainder is better because that is how much entropy remains in the dataset in that branch. But ultimately you want to compute gain, which is the amount of entropy/information in the remaining dataset in the branch minus the remainder. You want the attribute with the highest gain.

Why don't decision trees guarantee the best model?

"They're not necessarily the best because we're going for local maxima (highest gain) at each step in the process, which doesn't necessarily guarantee the best model overall."
The second part need a bit of nuance. If there is data or attributes missing there is still one "best" model that may or may not be found. But that best model may not be what you, the programmer/designer, want.

What is optimal and knowable to the algorithm based on data may be different than what a human knows to be optimal should infinite data with no missing attributes exist.

Markov Decision Process & Reinforcement Learning

-MDPs: figure out the best action given full-observable action-stochastic world
-MDP: a sequential decision problem for a fully observable environment, with stochastic actions, a Markovian transition model and additive rewards.
-Markovian transitions depend on the current state, s, and not the history of earlier states

POMDP (concepts)

- has the same elements as an MDP—the transition model P(s | s, a), actions A(s), and reward function R(s)—but, like the partially observable search problems of Section 4.4, it also has a sensor model P(e | s)
- P(e | s): probability of perceiving evidence, e, in state, s.
- belief state—the set of actual states the agent might be in
- belief state, b, is the probability distribution over all possible states for POMDP

$$b'(s') = \alpha\, P(e \mid s') \sum_{s} P(s' \mid s, a) b(s)$$

- where **α** is a normalizing constant that makes the belief state sum to 1

$$\begin{aligned} P(b' \mid b, a) \;&=\; P(b'|a, b) = \sum_{e} P(b'|e, a, b) P(e|a, b) \\ &= \sum_{e} P(b'|e, a, b) \sum_{s'} P(e \mid s') \sum_{s} P(s' \mid s, a) b(s)\ , \end{aligned}$$

- where $P(b'|e, a, b)$ is 1 if $b' = \text{FORWARD}(b, a, e)$ and 0 otherwise.
- Likely won't be a written problem, but conceptual is fair game.

- **Value iteration**

- Iterate n times, adjusting U(s) with each iteration
- Bellman Update: U(S_n) = R(S_n) + Z*(max(next state values), where z represents the given gamma value
- Next state value = $\sum_{num\ states}$ (prob of getting there) * (utility of the state)
- Policy for a given state = the direction with the best expected utility
- Update the utility of each state **after** each iteration
- $$U_{i+1}(s) = R(s) + \gamma \times \max_{a \in A} \left(\sum_{s' \in S} T(s, a, s') U_i(s') \right)$$

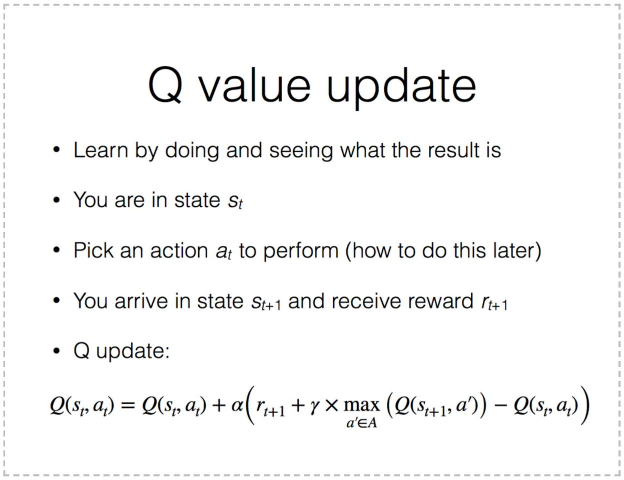
A Markov Decision Process (MDP) is really just a framework that we can use in order to establish a policy for choosing the best action in each state given the information we know. Value Iteration can be thought of as a specific way of obtaining this information such that we can develop a policy, alongside Q-Learning and other variants not covered in this course

---
As he gave it in lecture, you can think of the discount factor as a sort of "trust" that the MDP has in the information it gets from each state or iteration of the utility calculation. With discount = 1, we completely trust all the information generated by the iteration, so we can find the action that will give us the best total utility (a1). With discount = 0.9, though, we don't fully trust future states anymore, so in turn we can end up thinking that other paths (potentially longer ones) give us a better reward.

You can also think of it as a sort of measure of how much the MDP values future rewards, giving less weight to those it gets in future states. So, the net negative reward doesn't hit us as much as it did originally and the path seems better overall. Hope this helps!

- Q-learning

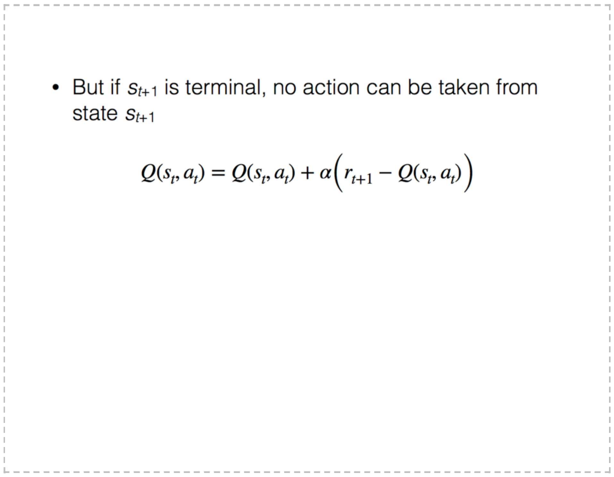
- no transition function (but you have a simulation)
- model free
- trial and error learning
- Learn a Q-table: Q(s,a)
- π(s) = argmax$_{\text{a in A}}$ Q(s,a)
    - There are 3 training approaches to creating a policy
        1) Random - Always perform a random action, regardless of whether you've already encountered the state.
            a) The benefits of this are that it will never get stuck to a single set of actions, but it's rather inefficient.
        2) On-Policy - Performs actions according to the policy, π(s) = argmax$_{\text{a in A}}$ Q(s,a)
        3) Epsilon-greedy - for random generated number, r
            a) If r < ε, pick random action, a ----- exploration
            b) If r >= ε, pick policy π(s) = argmax$_{\text{a in A}}$ Q(s,a) ----- exploitation
- q value update
    - learn by doing and seeing what the result is
    - given state s_t, action performed a_t,next state s_{t+1}



Q value update

- Learn by doing and seeing what the result is
- You are in state s_t
- Pick an action a_t to perform (how to do this later)
- You arrive in state s_{t+1} and receive reward r_{t+1}
- Q update:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\Big(r_{t+1} + \gamma \times \max_{a' \in A}\big(Q(s_{t+1}, a')\big) - Q(s_t, a_t)\Big)$$



- But if s_{t+1} is terminal, no action can be taken from state s_{t+1}

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\Big(r_{t+1} - Q(s_t, a_t)\Big)$$

Random Math That Can't Hurt

- p(a|b,c) = p(a|c) **if a & b are independent**
- p(a,b|c) = p(a|b,c)p(b|c)
- p(a|b,c,d,e) = p(a) [p(b|a) * p(c|a) * p(d|a) * p(e|a)]

$$P(X|Y) = \frac{P(X \wedge Y)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$$ **(Bayes Theorem)**

- p(x or y) = p(x) + p(y) - p(x & y)
- p(x^c) + p(x) = 1.0
- p(A & B) = P(A) x P(BIA) **if A & B are dependent**
- p(A & B) = P(A) x P(B) **if A & B are independent**
- log2(a) = ln(a)/ln(2)

In action stochastic environments you do know what state you are in at all times. You just don't know what state you will be in after choosing an action.
In sensor stochastic environments you don't know what state you are in, thus the environment must be partially observable.

Random Piazza Stuff (Mark's Answers)

State space Search: figure out the best action (and best subsequent actions because the works is fully deterministic and fully observable) to get closer to a goal

Optimization search: find a state (a configuration of the world) according to some criteria. Action is not considered.

MDP: figure out the best action given full-observable action-stochastic world

Bayes Net: infer the probability that the agent is in some particular state. Actions are not considered

- Bates nets don’t determine action. MDPs and POMDPs produced policies that determine action.

Dynamic Bayes Net: infer the probability that the world will be in a particular state given time. Actions are not considered.

POMDP: figure out the best action given sensor-stochastic and action-stochastic environment. POMDP = MDP + Dynamic Bayes Net.

-In action stochastic environments you do know what state you are in at all times. You just don’t know what state you will be in after choosing an action.

-In sensor stochastic environments you don’t know what state you are in, thus the environment must be partially observable.