Lecture 1
Introduction to AI and CS 486/686

Alice Gao

November 2, 2021

Contents

1 Applications of Artificial Intelligence

In this section, I'm going to talk about applications of artificial intelligence.

State of Art of AI

Arguably, the grand goal of AI is to build a general intelligence agent. If I were to summarize the state of art of AI, I would say the following. There hasn't been much success towards this grand goal. There are, however, lots of progress in many restricted domains.

The best way to get a sense of what artificial intelligence is to look at some applications. Let me describe some examples of problems in AI research. This will be a biased introduction to AI since time is so limited. For every example, I will describe the problem, the main result, and give you a high-level summary of the techniques used to solve the problem.

Chess

Example:

- Search; evaluation function; Grandmaster game database.
- Campbell, M., Hoane Jr, A. J., & Hsu, F. H. (2002). Deep blue. Artificial

intelligence, 134(1-2), 57-83.

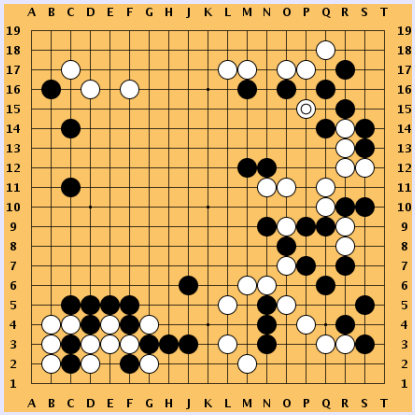
Let me start by talking about Chess. I grew up with this story and I still remember the glory of the seminal match from my childhood. However, if you are one of my CS 486 students at the University of Waterloo, this match was likely before you were born.

In 1997, IBM's DeepBlue program played Gary Kasparov, the world champion in Chess at the time. DeepBlue defeated Kasparov in a 6-game match. This match was the first time that a computer program defeated a world Chess champion in a tournament.

DeepBlue used several strategies. It performed lookahead search. It made use of a complex evaluation function to evaluate the likely game outcome of a board position. The evaluation function was handcrafted with 8000 features. DeepBlue also had a large Grandmaster game database and it was encouraged to play moves that appeared in the database.

Go

Example:



- Deep neural networks; supervised learning; reinforcement learning.
- AlphaGo: Silver et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.
- AlphaGo Zero and Alpha Zero

Next, let's look at the game of Go or Weiqi. This game was invented in China about 2500 years ago. Two players take turns putting black or white stones on a 19 by 19 grid board. The goal is to occupy a larger territory on the board.

I remember watching my dad and my grandfather playing this game for hours when I was a child. They also watched TV shows of experts going through a game of Go and explaining the moves. My understanding of Go still remained at a extremely basic level. Suppose that

we want to surround one space. In the middle of the board, surrounding one space requires four stones. If we do this on a side, we need three stones. Finally, if we do this in a corner, we need two stones only. This is why players tend to start by putting stones in the corner and on the sides in a game and only moving to the middle towards the end of the game.

Similar to Chess, Go is a game of perfect information. Everything is visible on the board. Theoretically, we can solve Go by exhaustive search. Unfortunately, the enormous search space makes exhaustive search infeasible. Try to picture a search tree, where the root is at the top and the leaf nodes are at the bottom. Each node has around 250 child nodes, and the depth of the search tree is around 150 levels. This is an enormous search tree.

The early Go programs relied on tree search algorithms and achieved strong amateur play only. Around 2015, Google DeepMind achieved a breakthrough. They developed a program called AlphaGo, which defeated top professional Go players. In 2015, AlphaGo defeated Fan Hui (5-0), who was the European Go champion for a few years. In 2016, AlphaGo defeated Lee Sedol (4-1), who is a 9-dan professional player and won 18 international titles. In 2017, AlphaGo beat Ke Jie, who was the number one ranked Go player in the world at the time.

How does AlphaGo work? AlphaGo was still based on tree search. However, it made use of two deep convolutional neural networks to reduce the effective depth and breadth of the search tree. AlphaGo trained the two networks using supervised learning on human data and reinforcement learning through self-play.

The first network is called the policy network, which maps a board position to a probability distribution over actions. AlphaGo uses the policy network to sample actions when simulating a game.

The other network is called the value network, which maps a board to a numeric estimate of the game's outcome. AlphaGo uses the value network to predict the game outcome without simulating the game until the end.

After the breakthrough, DeepMind kept working on the game of Go and developed programs that are superior to AlphaGo.

AlphaGo Zero no longer made use of human data. It learned to play the game using reinforcement learning only and defeated AlphaGo by 100 games to 0.

AlphaZero used one general-purpose algorithm to learn to play three games well (chess, shogi - Japanese chess, and Go). This was a significant step towards developing a general game-playing algorithm.

If you are interested, I encourage you to check out the papers on the three programs. They are published in Nature and Science, which are the two of the most prestigious journals in the world.

Poker

Example:

- Play with uncertainty. Must model opponent(s).
  Care about long-term payoff.
- Bowling, M., Burch, N., Johanson, M., & Tammelin, O. (2015). Heads-up limit hold'em poker is solved. Science, 347(6218), 145-149.
- Brown, N., & Sandholm, T. (2019). Superhuman AI for multiplayer poker. Science, 365(6456), 885-890.

The next story is about Poker. Poker is challenging to solve for two reasons. First, it is a game of imperfect information. Each player cannot see their opponents' cards. Second, it is not a one-shot game. During a tournament, each player must strategize to maximize their chips over many games.

Two research teams have made significant progress towards developing a program to play Poker.

The team led by Professor Bowling at the University of Alberta solved heads-up limit Texas Hold'em. This is the game with two players and there is a upper limit on the bet amount in each round. They modeled poker as an extensive-form game and solved for an approximation of the Nash equilibrium using an algorithm called counterfactual regret minimization. They declared that the game is essentially weakly solved, which means that a human lifetime of play is not sufficient to establish with statistical significance that the strategy is not an exact solution. Their results confirm that the game is a winning game for the dealer. Extensive-form game and Nash equilibrium are game-theoretic concepts. If you want to learn more, search them online or consider taking a course on Game Theory.

Around the same time, a team led by Professor Sandholm at Carnegie Mellon University was working on poker with more than two players. For a two-player poker game, any player following a Nash equilibrium strategy is guaranteed to not lose in expectation no matter what the opponent does. This claim is no longer true if the game has three or more players. Therefore, the CMU team focused on developing strategies to beat top professional poker players without solving for Nash equilibrium strategies.

The CMU team developed a program called Pluribus, which defeated elite human professionals in six-player no-limit Texas hold'em poker. Pluribus developed its strategies as follows. It first simplified the game representation by eliminating some actions and combining some

decision points. Next, Pluribus spent 8 days developing a blueprint strategy. It started by playing randomly and improved by learning to beat earlier versions of itself. After playing each action, Pluribus evaluated every other action by using counterfactual reasoning: if I chose this action instead, how much better or worse would I have done? Finally, Pluribus also used real-time search during a game.

Be sure to check out the two Science articles if you want to learn more.

Atari Games

Example:

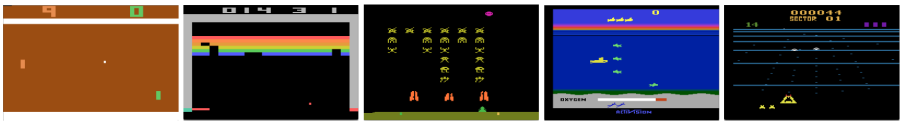


Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

- Outperforms previous approaches; Surpasses a human expert.
- Reinforcement learning; Convolutional neural network; High-dimensional sensory input.
- Mnih et al. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.
- Video: https://youtube.com/watch?v=V1eYniJ0Rnk

Let's look at some video games. The next story is on Atari 2600 games. Checkout the screenshots of the games from the paper. I remember playing Breakout and Space Invaders when I was a child.

The goal is to create a single program that is able to play as many Atari 2600 games as possible. Just like a human player, the program learns to play the game solely from the video input, the reward, and the set of possible actions.

The program was tested on seven Atari games. It outperformed all previous reinforcement learning algorithms on six of the seven games. It also surpassed a human expert on three games.

The program learned to play the Atari games using reinforcement learning, specifically Q-learning. Q-learning relies on a value function to choose an action in each state. The value function estimates the agent's expected reward in the long term by taking the action in the state.

The main contribution of this paper was learning a complex value function by training

a convolutional neural network. In prior work, the value function was created by using handcrafted features, and this was infeasible for the high-dimensional video input in the Atari games. Instead, the program used a convolutional neural network to represent the value function. The convolutional neural network automatically extracted high-level features from the video input.

Starcraft 2

Example:

Image from https://snl.no/StarCraft_II

- Multi-agent problem; Imperfect information;
  Large action and state space; Delayed credit assignment.
- Vinyals et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782), 350-354.
- Video: https://www.youtube.com/watch?v=jtlrWblOyP4

StarCraft is one of the most challenging real-time strategy games and one of the longest-played e-sports of all time. Set in a distant Milky Way galaxy, the game revolves around three intelligent species fighting for dominance.

StarCraft emerged as a grand challenge for AI research for several reasons. First, it is a multi-agent problem. Several players compete for influence and resources. Each player controls hundreds of units, which need to collaborate to achieve a common goal. Second, it is a game of imperfect information. Each player can only observe the game via a local camera. Third, the action space is vast and diverse. There are approximately 10^26 possible choices at each step. Fourth, the game lasts for tens of thousands of time steps, and the player's strategy must balance short-term payoffs and long-term gains.

In 2019, Google DeepMind developed a program AlphaStar to play StarCraft. AlphaStar played anonymously against human players on Battle.net. AlphaStar achieved Grandmaster level for all three StarCraft races and placed above 99.8% of all the ranked human players.

How did AlphaStar learn to play the game? AlphaStar trained one agent for each race. First, the agent was trained to predict human play by supervised learning. Next, the agent was trained as part of a league of several types of agents. The main agents focus on learning rapidly through self-play. Other agents try to identify exploits in some agents or systemic weaknesses of the entire league. Each agent tried to maximize the win rate against a non-

uniform mixture of other agents through reinforcement learning.

Check out the paper and the YouTube video for more details.

Jeopardy

Example:

"AI for $100, Alex."
"This popular TV quiz show is the latest challenge for IBM."
"What is Jeopardy?"

- In 2011, IBM's Watson beat the two highest ranked players in a two-game Jeopardy! match.
- Questions from a broad domain. Must answer questions with high precision and with accurate confidence. Fast answering.
- Ferrucci et al. (2010). Building Watson: An Overview of the DeepQA Project. AI Magazine, 31(3), 59-79.
- Ferrucci et al. (2013). Watson: beyond jeopardy!. Artificial Intelligence, 199, 93-105.

The next story is about Jeopardy!. Jeopardy! is a popular TV game show. In each game, three contestants compete against each other, trying to understand and answer rich natural language questions very quickly. For each question, the contestants compete for the first chance to answer via a handheld buzzer.

What's special about Jeopardy! is that the questions and answers are reversed. For example, suppose that the player chose the category "AI for $100". The host provides a clue in the form of an answer: This popular TV quiz show is the latest challenge for IBM. The contestant must phrase their response in the form of a question: "What is Jeopardy!?"

In 2007, IBM Research took on the grand challenge of building a computer system that can perform well on open-domain question answering, specifically, a system that can win the game of Jeopardy!. In 2011, a system called Watson beat the greatest players of all time, Ken Jennings and Brad Rutter, in a two-game Jeopardy! match.

To succeed at Jeopardy!, players must overcome several challenges: First, questions come from a broad domain and use rich and varied natural language expressions. Second, players must answer questions with high precision and high confidence. Third, players must answer the questions very quickly. On average, champion players must correctly answer at least 85% of the questions they buzz in for and they must buzz in for at least 70% of all the questions.

How did Watson become so good at playing Jeopardy!? The architecture behind Watson is called DeepQA. Contrary to some popular misconceptions, DeepQA does not simply look

up the answer in a database. It makes use of sophisticated natural language processing and search algorithms to answer a question.

First, DeepQA tries to understand what the question is asking. Then, it finds some potential answers. DeepQA stores 200 million pages of information, including Wikipedia articles and relational databases, and is not allowed to access the Internet. For each potential answer, DeepQA uses hundreds of algorithms to study the evidence and assigns a score to the answer. Finally, DeepQA generates a ranked list of answers.

The Jeopardy! challenge was a means to an end. Subsequently, IBM adapted Watson for healthcare. By 2012, two healthcare organisations had started piloting Watson.

Self-Driving Cars

Example:

- DARPA Grand/Urban Challenges
- The perception system
  - Tasks: Locate car in the environment; map static obstacles; map moving obstacles; lane detection; traffic sign detection.
  - Algorithms: supervised learning, for example, SVM and convolutional neural networks
- The decision-making system
  - Tasks: route and path planning; choosing driving behaviour; avoiding obstacles.
  - Algorithms: search algorithms (Dijkstra, A*), finite state machines, Markov decision processes.
- Badue et al. (2021). Self-driving cars: A survey. Expert Systems with Applications, 165, 113816.

For our next application, let's look at self-driving cars.

To stimulate the research and development of self-driving cars, DARPA (the Defense Advanced Research Projects Agency) organized three competitions in the last decade.

In 2004, the first DARPA Grand Challenge required self-driving cars to navigate an 142-mile course in the Mojave Desert, USA within 10 hours. All competing cars failed within the first few miles.

The DARPA grand challenge was repeated in 2005. This time, the course was 132-mile long and contained flats, dry lake beds, mountain passes, narrow tunnels and lots of sharp turns. Four cars completed the route within the time limit. Stanford University's car Stanley

claimed first place. Two cars from CMU, Sandstorm and Highlander, finished in second and third places.

In 2007, the DARPA Urban Challenge took place in the former George Air Force Base in California. Cars needed to complete a 60-mile course in a simulated urban environment, interacting with other self-driving and human driven cars, within 6 hours. The first three places went to CMU's car Boss, Stanford's car Junior, and Virginia Tech's car Odin.

So, what does it take to build a self-driving car? A self-driving car consists of two main parts: The perception system and the decision making system. The perception system tries to understand the environment, whereas the decision making system determines what the car should do next.

The perception system needs to perform tasks such as locating the car in the environment, recognizing static and moving obstacles, detecting lane markings on the road, and understanding traffic signs. Many of these tasks can be accomplished using supervised learning algorithms, such as support vector machines and convolutional neural networks.

The decision making system needs to perform tasks such as planning a route, determining what to do next, and avoiding obstacles. These tasks require algorithms for searching and planning, for example, search algorithms, such as Dijkstra and A*, finite-state machines, and Markov decision processes.

If you are interested, check out the survey paper for a comprehensive literature review on self-driving cars.

Other applications of AI

Example:

- Solving partial differential equations
- Antibiotic Discovery
- Hide and Seek Games
- How AI is Changing Science

There are so many other applications of AI that I don't have time to talk about. Here are some examples: solving partial differential equations, discovering new types of antibiotics, and playing hide and seek games. Be sure to follow the links if you want to learn more.

Summary

That's everything on the applications of Artificial Intelligence. Which one was your favourite application? Please feel free to let us know by posting on Piazza.

2 Topics in CS 486/686

In this section, I'm going to talk about the topics that I will cover in this course.

This course is a broad and shallow course. The academic calendar requires me to cover a large number of topics in AI. There is very limited time for each topic. I would love to discuss certain topics in more detail, but unfortunately, I don't have that freedom.

My goal is to give you a broad overview of AI. This course prepares you to explore some topics on your own if you wish to do so. It's just like a Chinese saying. It says that, the master can only guide you into the door. After that, it is up to you to keep learning and exploring yourself.

This course consists of 24 lectures over 12 weeks. There are roughly 2 lectures per week. Let's look at the lecture topics.

Introduction to AI and CS486/686
1: Introduction to AI and CS 486/686
2: Definitions of AI

Lectures 1 and 2 introduce this course.

In lecture 1, I will describe several applications of AI, mostly in games, and describe the course outline.

In lecture 2, I will discuss four definitions of Artificial Intelligence.

Search:
2: Uninformed Search
3: Heuristic Search
4: Constraint Satisfaction Problems
5: Local Search

Lectures 2 to 5 are on search algorithms.

Lecture 2 covers uninformed search algorithms. These algorithms explore the search space systematically but blindly.

In lecture 3, I will introduce heuristic functions and use them to search for optimal solutions.

In lecture 4, I will discuss solving constraint satisfaction problems with backtracking search and the arc-consistency algorithm.

In lecture 5, I will discuss local search algorithms. Local search algorithms do not explore the search space systematically and are not guaranteed to find the global optimum, but they can find reasonably good solutions quickly.

Supervised Learning:
6: Machine Learning, Decision Trees 1
7: Decision Trees 2

8: Neural Networks 1
9: Neural Networks 2

After that, lectures 6 to 9 will discuss machine learning, specifically, supervised learning.

Lectures 6 and 7 cover decision trees.

Lectures 8 and 9 discuss artificial neural networks. Both algorithms are powerful. However, decisions trees are simple and intuitive, whereas neural networks are complex and resemble black boxes.

Reasoning Under Uncertainty:
10: Uncertainty and Probability
11: Semantics of Bayesian Networks
12: Testing Independence, Constructing Bayesian Networks
13: Variable Elimination Algorithm
14: Hidden Markov Models 1
15: Hidden Markov Models 2

So far, the first half of the course focuses on algorithms for a world without uncertainty.

In the second half of the course, we will solve problems in a world with uncertainty. In lectures 10 to 15, I will introduce Bayesian networks to model uncertainty and describe an algorithm for performing exact inference in Bayesian networks.

Lecture 10 reviews probabilities.

Lecture 11 reviews the definitions of unconditional and conditional independence.

After that, I will introduce Bayesian networks. We can use Bayesian networks to represent a probability distribution compactly by making use of the independence relationships.

In lectures 12, I will discuss testing independence relationships in a Bayesian network and constructing multiple correct Bayesian networks for the same distribution.

In lecture 13, I will show you how to calculate any probability exactly given a Bayesian network using the variable elimination algorithm.

Finally, lectures 14 and 15, I will introduce a hidden Markov model, which is a special type of Bayesian network. Because of the special structure in a hidden Markov model, we can perform inference more efficiently.

Decision Making Under Uncertainty:
16: Decision Theory and Decision Networks 1
17: Decision Networks 2
18: Markov Decision Process 1
19: Markov Decision Process 2
20: Reinforcement Learning 1
21: Reinforcement Learning 2

In a world with uncertainty, performing inference is not enough. We also need to take actions.

In lectures 16 to 21, I will discuss how to make decisions under uncertainty.

In lectures 16 and 17, I will model a decision making scenario using a decision network and show you how to solve for the optimal policies using the variable elimination algorithm.

In lectures 18 and 19, I will consider problems that may go on for an indefinite number of time periods. I will show you how to model the problem using a Markov decision process and solve for the optimal policies using the value iteration algorithm or the policy iteration algorithm.

In lectures 20 and 21, I will build on the Markov decision process to develop reinforcement learning algorithms, including the adaptive dynamic programming algorithms and the Q-learning algorithms.

Multi-agent Systems:
22: Game Theory 1
23: Game Theory 2
24: Conclusion

In lectures 22 and 23, we will consider a world in which there are multiple intelligence agents. I will introduce a two-player normal-form game, the simplest model of a strategic scenario, and analyze the players' behaviour with several game theoretic solution concepts: dominant-strategy equilibrium, Nash equilibrium, and Pareto optimality.

The last lecture, lecture 24 will be a break for you.

That is a brief introduction of the topics in this course. What are you looking forward to learning in this course? Please let us know by posting on Piazza.

3 Course Outline

Do I need to read the course outline?

Yes, I expect you to read the course outline word by word from start to finish. You are responsible for knowing all the contents. Please read the course outline now. You do not want to be surprised by any contents in the course outline halfway through the term.

What is the best way to reach Alice?

The best way to reach me (Alice) is to post a message on Piazza and not by sending me an email. Piazza allows me to separate your messages from other emails (research, advising, etc.) You will often get a faster response by posting on Piazza. During weekdays, I tend to respond on the same day. During weekend, I tend to respond within 2 days. If I haven't responded in a few days, it doesn't mean that I am ignoring you. I simply needed more time to think about your question. Do not make a new post. Instead, create a follow-up question to get my attention and let me know that you're anxious to get an answer.

Should I (a CS 486 student) work on the project?

There are a few things to consider.

On one hand, the project is a great way for you to challenge yourself and to learn something deep about artificial intelligence. You will gain research skills; you will get experience working with others, and you will get hands on experience implementing something that's likely not taught in the course.

On the other hand, the project can require a lot of time and effort. You must be juggling many different things — courses, searching for Co-op or full-time job. That's one reason I made the project optional for undergraduate students. Also, it's worth at most 10% of your final mark.

Here's my advice to you. Choose to work on the project if and only if you are extremely motivated to do it well. Of course, you have to find other similar-minded people to work on the project with you.

What is the minimum requirement for passing this course?

First, your final mark must be at least 50%. In addition, there are several other requirements.

For the undergraduate students, to pass this course, your average test mark must be at least 50%, and you must complete at least 20 out of the 24 lecture quizzes. If either condition is not satisfied, you will fail this course and your final mark will be at most 46%.

For the graduate students, your average project mark must be at least 50% to pass this course. Otherwise, you will fail this course and your final mark will be at most 46%.

As a professor, I do not enjoy seeing any of my students fail. My goal is to help you learn and thrive and to see that you get the mark that you deserve. However, since I've set these rules at the beginning of this course, following these rules is the only way for me to treat all the students equally and fairly.

It is your responsibility to monitor your marks and come talk with me if you are in danger of failing this course. If you do this early, I may be able to help you. If you come talk with me when you are already failing the course, I won't bend the rules just because you are asking or complaining.

What are the assignments like?

The assignments tend to be long and challenging. I design them this way since I believe that you learn the most by doing the assignments and not by going over lecture materials. Assignments give you the opportunity to apply the algorithms that you learned in solving real problems. Each assignment has 1 written question and 1 programming question. The written question can be time-consuming, because you may need time to review the course materials and think about how to solve it. The programming question as you should know is time-consuming not just because you have to write the code and also because you need time to test and to debug.

If you always wait until a few days before the deadline to start an assignment, you will be stressed, and you won't have time to think about the questions thoroughly and enjoy the process of solving the problems. My advice is the following: Schedule a few hours every week to work on the assignment. You can work on the written question in one week, and the programming question in the next week. If you follow this advice, you will enjoy the course much more.

An important note regarding academic integrity:

The course outline has an academic integrity statement. Some important points from the statement: you should not share your work, including your code, with anyone at any point; do not show the code to someone else via screen share; do not send your code to anyone even if they promise that they won't copy it; do not post the assignment, your solutions and the sample solutions on any public or private forum or website. If you send your solution to someone else and we catch it, both of you will be punished for violating academic integrity. If your friend comes asks you for your code, smile and say "Sorry, I am not comfortable sharing my code with you." Or simply, stop responding or change the subject.

Next point, you should not look for solutions anywhere; do not search for them online; do not ask your friends who took the course in previous terms to share the solutions.

All of these can result in serious cases of academic misconduct. The standard penalty is zero on the assignment and an additional 5

The academic integrity statement tells you what you should not do. What can you do when you're working on the assignment? This is described in the collaboration policy. You can talk with your friend and study all the lecture examples, notes, and videos together regarding the assignment. You can talk about high-level ideas and strategies. However, you should not keep any written or electronic records from your discussion. In the end, you will write out the solutions on your own. Make sure that everything you submit is your own work.

To make sure that everyone follows the academic integrity rules, we are going to compare your submissions with those of the other students. In particular, for the programming question

on the assignment, we are going to run MOSS on your program. MOSS stands for Measure of Software Similarities. It compares your code with those of other students who are taking the course this term and those of the students who took the course in the past. If the two programs are too similar, I will be in touch, and you do not want to get an email from me for this reason.

Some final words about academic integrity. Cheating is an attempt to take a shortcut in life and there are no real shortcuts in life. Sooner or later, you will pay for the shortcuts you took. If you cheat, you didn't learn the materials for real, you will not be able to apply it in your jobs. Let me quote a friend of mine. This friend was an undergrad with me at UBC, and now they're working at a large tech company in the Bay area. This friend said to me recently that "Alice, I wish I studied harder in school. I thought studies didn't matter that much, but I see now that they do." Don't take the shortcut now and hurt yourself in the long run.

How should I prepare for the tests?

How should you prepare for the tests? The tests are open book. In an open book exam, there is no point asking you to remember anything directly from the lectures, since you can look it up. For this reason, open-book exams tend to be much more challenging. They often contain questions that ask you to apply the concepts in new problems, and there may not be enough time to look up things at your leisure.

As a result, to prepare for the tests, there is no need to memorize everything, but you should understand the materials well enough so that you can look them up fairly quickly.

My (Alice's) favourite type of test question is to ask you to execute an algorithm on a problem. Given this, you should study for the tests by doing exactly this: take the examples from lectures or construct new problems. Execute the algorithms on these problems and write down the execute process and the final result. Get together with your friends and test each other. One of the best ways of testing whether you understood something is to explain it to someone else. Also, pay attention to the details — the corner cases, the special scenarios. The details are important and are often where people lose marks.

Why do we have to complete the lecture quizzes?

The primary purpose of lecture quizzes is to force you to keep up with the lecture schedule. In each week, there are two lectures: one available on Monday and the other one available on Wednesday.

With online learning, it's very easy to fall behind. Once you fall behind, it takes quite a bit of time to catch up, and you will have a lot of trouble with the assignments and the tests.

In short, lecture quizzes are not worth very much themselves, but their purpose is to force you to keep up with the lectures. If you do, you will have a much easier time with the other much more important assessments.

I've made the lecture quizzes as easy as possible in several ways. First, they're taken straight from the lecture videos. Second, they're worth only 10% in total. Third, you have unlimited

time for a quiz and three attempts for each quiz. Finally, you can miss four lecture quizzes and still get a perfect mark on the lecture quizzes.

The catch is that each lecture quiz is only available for 48 hours. You must complete each quiz within 48 hours. There are no extensions and no exceptions since I'm already dropping the lowest four quiz marks.

The most common complaint regarding lecture quizzes is the following: the schedule is too rigid, can we have a few days to complete each quiz, say until the weekend.

Unfortunately, my answer is no. Here is my reasoning. In normal times, you need to attend two lectures each week, correct? Hopefully, you pay attention in each lecture and absorb the basic things. Then, all I'm asking you to do is to spend 20 minutes after each lecture reviewing the lecture materials — in the form of a lecture quiz. If you paid attention during the lecture, this should be a breeze for you. If you didn't pay attention during the lecture, you would have to spend the extra time to understand the materials anyway, probably before working on an assignment or a test. Either way, it is worthwhile spending this time to learn the materials. Please take advantage of the lecture quizzes and keep up with the lecture schedule.

4 Definitions of Artificial Intelligence

We can define artificial intelligence in many ways. In this course, I will discuss the four definitions in Artificial Intelligence: A Modern Approach by Russell and Norvig. This table summarizes the four definitions.

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

Table 1: Definitions of Artificial Intelligence

First, let's look at the two columns. The two columns differ by how we want to measure the performance of the system. Do we want to measure the performance against humans, or do we want to measure the performance against rationality? Rationality is an ideal concept of intelligence that we are going to develop.

If we use different performance measures, we will tackle the research questions using various techniques. For example, if we use humans as the performance measure, we need to understand how humans think and act. This goal would likely require putting people in the lab, observing what they do, forming a hypothesis, and conducting experiments to verify the hypothesis. These processes are part of many branches of empirical science.

If we use rationality as the benchmark, then we need to define rationality as a mathematical model. This model leads to branches of mathematics and engineering where we can develop theoretical models, analyze the theoretical models to understand the underlying principles of intelligence. We could also perform experiments to verify the theoretical model's predictions.

Next, let's look at the two rows. The two rows differ by our goals. Do we aim to model reasoning, or do we aim to model behavior? One significant difference between the two goals is whether we can observe them. We cannot observe thoughts directly, but we can easily observe external behavior.

I will discuss the four definitions one by one. The goal of this discussion is not to argue that one of the four definitions is correct. All four definitions are valid and useful. Researchers have studied all four definitions in the past.

4.1 Cognitive Modeling

For the first definition, Cognitive Modeling, our goal is to develop a system that thinks like humans.

Why do we want to use humans as the benchmark? In our world, we do not have a lot of examples of intelligence. Human is one of a few examples of intelligence, and we may as well use it.

To pursue Cognitive Modeling, we need to understand how humans think and build a system that mimics human thinking and reasoning. How do we understand how humans think?

First, we examine our thoughts — reflecting on our thinking and reasoning process by introspection.

Second, we could also conduct psychological experiments:

- Bring people into the lab.
- Observe what they do.
- Try to infer what thought processes led to their behaviour.

Third, we can also observe a person's brain in action using technology such as MRI. For example, we can see which area inside the brain is active when trying to do something.

These approaches led to a field called Cognitive Science. The goal of this field is to develop precise and testable theories of the human mind.

To recap, the first definition of AI, Cognitive Modeling, aims to develop a system that thinks like humans. This definition uses humans rather than rationality as the benchmark and aims to simulate thoughts rather than behavior.

4.2 Turing Test

For the second definition, Turing Test, our goal is to develop a system that acts like humans.

Alan Turing, the father of computer science, proposed the Turing Test around the 1950s. The Turing Award, the highest honor in Computer Science, is named after Alan Turing. When Alan Turing first proposed the Turing Test, he called it by a different name. Do you know this alternative name of the Turing Test? I will reveal the answer at the end of this video.

How does the Turing Test work? The person on the top left is the interrogator. The interrogator is communicating with the entity that could be a human or a computer program. The interrogator may ask the entity any questions through a text interface. The interrogator could also send visual signals using the light bulb or send some objects to the entity. The entity wants to prove to the interrogator that the entity is intelligent. If the entity behaves so that the interrogator cannot distinguish the entity from a human, then the entity passes the Turing Test and is considered intelligent.

The Turing Test is a simple yet powerful idea. Is it useful for Artificial Intelligence?

Some people claim that the Turing Test is not useful. Although the Turing Test gives us a way to recognize if an entity is intelligent, it doesn't tell us how to build an intelligence system.

There are lots of arguments claiming that the Turing Test is very useful. Turing Test gave rise to several important areas in artificial intelligence. Let me give you a few examples. If an entity wants to pass the Turing Test, what kinds of things should it be able to do? First, it needs to understand natural language — This led to the development of natural language processing. Second, it needs to represent and store knowledge — This led to the area of knowledge representation. Third, it needs to reason, inference, and learn–this led

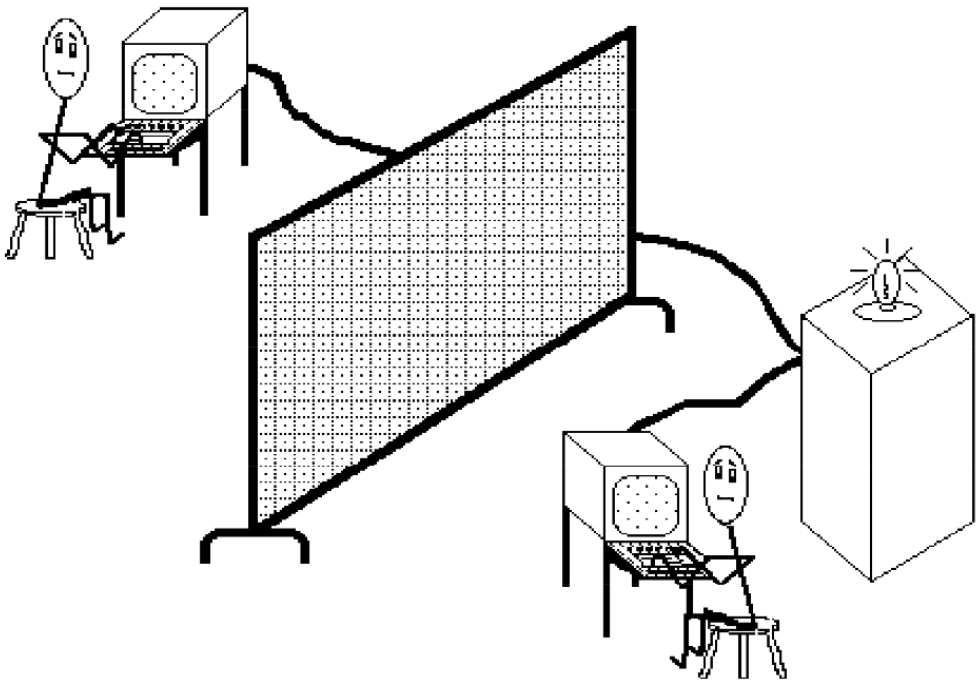


Figure 1: Turing Test

to machine learning. Fourth, it needs to perceive objects — This led to computer vision. Finally, it needs to move and manipulate objects — This led to the field of robotics. These are some prominent research areas in Artificial Intelligence. I find it incredible that the Turing Test motivated researchers to explore these critical research questions.

To recap, the second definition of AI, the Turing Test, aims to develop a system that acts like humans. This definition uses humans as the benchmark and aims to model behavior rather than thoughts.

4.3 Rationality

So far, I have discussed two definitions: Cognitive Modeling — thinking like humans, and Turing Test — acting like humans. Both definitions use humans as the benchmark.

The next two definitions will use rationality as the benchmark. Rationality is an ideal concept of intelligence, and this is a concept we can define mathematically. Roughly speaking, a system is rational if it does the right thing given what it knows.

4.4 Laws of Thought

For the third definition, our goal is to build a system that thinks rationally. This is called the Laws of Thought approach.

Aristotle, the Greek philosopher, first attempted to define what it means to think correctly formally. He defined syllogisms, which are patterns of argument structures such that given

correct premises, one can draw correct conclusions. Check out a famous example of a syllogism: Every person is mortal, Socrates is a person; therefore, Socrates is mortal. Syllogism is an example of a Law of Thought. The idea of syllogism inspired people to develop the field of logic.

Logic is a precise language we can use to express statements. The development of logic led to the logicist tradition. The logicist tradition proposes using a logical system to describe and store all of our knowledge-the objects and their relationships. If the system has all of our knowledge, theoretically, it can solve any problem. Based on this belief, the primary goal in AI was to build such a system for a long time.

There are several problems with this approach. Let me give you two examples. First, logic is a rigid language, and it is challenging to translate natural language into logic. Given this, how can we hope to encode all of our knowledge using logic?

Second, let's assume that we managed to encode all of our knowledge using logic in the system. How can we solve a problem using this system? First, we must search through the logical statements to find a useful statement for solving the problem. Since the system must be enormous, any brute force search will be incredibly slow.

These two problems tell us that: First, it's difficult to build such a system. Second, even if we build such a system, it is challenging to solve a problem efficiently using this system.

To recap, the third definition of AI, Laws of Thought, aims to develop a system that thinks rationally. This definition uses rationality rather than humans as the benchmark, aims to model thoughts rather than behavior.

4.5 Rational Agent

For the fourth definition, Rational Agent, our goal is to build a system that acts rationally.

The word "agent" comes from a Latin word, which means "todo." An agent is an entity that acts. What is a rational agent? Roughly speaking, the rational agent aims to achieve the best expected outcome by taking the expectation over our uncertainty. What behavior should be considered rational? I will spend a lot more time discussing this in the course. For now, let me give you some high-level descriptions. The agent should create and pursue goals, operate autonomously, perceive the environment, learn and adapt to changes in the environment.

To recap, the fourth definition of AI, Rational Agent, aims to develop a system that acts rationally. This definition uses rationality rather than humans as the benchmark and seeks to model behavior rather than thoughts.

4.6 Which definition would you use?

The four definitions of Artificial Intelligence are summarized in the table below.

Here is a question for you. If you were an artificial intelligence researcher, which of the four definitions would you use and why? This question does not have a correct answer. All the

Cognitive Modeling Systems that think like humans	**Laws of Thought** Systems that think rationally
Turing Test Systems that act like humans	**Rational Agent** Systems that act rationally

Table 2: Four Definitions of AI

definitions are valid and researchers have studied all four definitions in the past.

This question aims to allow you to choose an answer and come up with justifications for your answer. Pause the video. Choose an answer and come up with some rationale. Then, keep watching.

Example: If you were an Artificial Intelligence researcher, which of the following definitions of intelligence would you adopt?

(A) Systems that think like humans

(B) Systems that act like humans

(C) Systems that think rationally

(D) Systems that act rationally

Here is the "official" answer. This course will focus on the Rational Agent definition–building a system that acts rationally. This definition requires us to use rationality as the benchmark and aim to model behavior.

4.6.1 Why should we use rationality humans as the benchmark?

Let me share two reasons.

First, humans often act in ways that we do not consider to be intelligent. You may find this funny, but you also know that it's true. Let me recommend a book that I like, "Predictably Irrational" by Dan Ariely. This book describes several ways in which humans behave irrationally but in a predictable way. It's a fun read, and I highly recommend it.

Second, rationality is mathematically well defined and completely general. Since we can define rationality mathematically, we can explore it scientifically. We can develop theoretical models, analyze the theoretical models to produce predictions, then perform experiments to verify these predictions. In contrast, using humans as the benchmark makes it challenging to define and scientifically study human behavior.

These are two reasons why we should use rationality rather than humans as the benchmark.

To understand the difference between using rationality and humans as the benchmark, let's

think about the two approaches we used to develop modern aircraft.

Here is the first approach. Birds can fly. All the birds have wings. We can hypothesize that the winds must have allowed the birds to fly. Given that, we can build aircraft to mimic birds with wings.

Approach two: we will try to understand the principles that allow the birds to fly. This idea led to the field of aerodynamics, which eventually led to the development of modern aircraft.

This analogy mimics our discussion regarding humans versus rationality. Mimicking human behavior is like developing flying machines by mimicking birds with things. Developing rational behavior is like understanding the principles behind flying and then using that to create modern aircraft.

4.6.2 Why should we aim to model behaviour?

First, rational behavior is more general than rational thoughts. Thinking correctly and performing correct inference is only one way to achieve rational behavior.

For example, in real life, we often face questions that do not have a correct answer. Nevertheless, we still have to take action. In this case, correct thinking cannot help us since it does not exist.

Consider another example. When we face immediate danger, we often rely on reflexes to avoid trouble. In this case, we do not have the time to think and reason, and we must act quickly to survive.

These two examples show that rational behavior is more general than rational thoughts.