ARTIFICIAL INTELLIGENCE

The Very Idea

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science, Bioinformatics & Genomics and Neuroscience
Director, Artificial Intelligence Research Laboratory
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

Is Intelligence Uniquely and Exclusively Human?

Chimpanzees

- Chimpanzees
  - Have been observed using a stick as a tool to catch and each termites
  - Can learn American Sign Language (ASL)
  - Can learn to assign word meanings to physical tokens (colored disks) and arrange them into structured sentences to communicate with their trainers

Image source: BBC

African Gray Parrots

African gray parrots

- Pass a test often used to measure logic and reasoning in young children
- When presented with two cups, one containing a hidden treat, the other empty, when shown which cup was empty, the parrot correctly inferred that the reward must be in the other cup.
  - Humans don't develop this ability until they are about two-and-a-half years old
- Can pass even more complex tests that human young pass only at the age of 5 years or older

Image source: Sandra Mikolasch

Bees

Bees

- Have symbolic language
- Use tools
- Can count
- Learn by observation

Image source: Chittka lab

Bottlenose Dolphins

Bottlenose dolphins

- Recognize themselves in the mirror
- Hold onto memories
- Utilize language

New Caledonian Crows

Crows

- Make their own tools
- Can be taught to count
- Remember human faces

Im

Image source: Wired Science

Elephants

Elephants

- Use tools
- Form friendships

Image source: Carol Buckley

Octopuses

- Octopuses
  - Solve problems
  - Use tools
  - Can be sneaky
  - Recognize individual members of another species

Ravens

- Ravens
  - Match the performance of adult apes and human toddlers on a slate of tests

Image source: CBC News

Is Intelligence uniquely human?

- Preponderance of evidence suggests that there are many examples of intelligence in the natural world
- Intelligence is by no means uniquely human
- There are many ways to be intelligent
- What is the nature of the mechanisms needed to support different types of intelligence?
- Can we design machines that behave as if they are intelligent? How?

Is Intelligence uniquely human?

- Working hypothesis of AI
  - Cognition is (or at least can be modeled by) computation
    - What computations underlie intelligent behavior?
      - We will have a theory of reasoning if we can devise algorithms that reason from assumptions to conclusions
      - We will have a theory of learning if we can devise algorithms for learning from experience
      - We will have a theory of linguistic communication if we can devise algorithms that effectively communicate using language
      - We will have a theory of cooperation if we have a computational model of multi-agent collaboration
      - We will have a theory of creativity if we can devise algorithms that exhibit creativity
    - What exactly is computation?

On Computation

On computation

- Thomas Hobbes, the British philosopher claimed that reasoning or thinking – was a form of computation
- Gottfried Leibniz, the German philosopher, mathematician and scientist envisioned machines that could reason logically
- George Boole, the British logician develops Boolean Algebra -- a language for manipulating logical propositions and deciding whether or not they were True based on assumptions
- David Hilbert, the German philosopher poses the decision problem: Is there an algorithm that can decide whether a claim is provable from (consistent) assumptions
- Alan Turing invents the Turing Machine and shows that True claim is provable from the assumptions, but a claim that is false result in the algorithm running forever – no way to know if it is because the claim is false or the machine needs more time
- Now we look at computing in a bit more detail





Boolean Algebra

- Boolean algebra is similar to algebra you learned in high school except
  - Variables in the expressions or formulae, instead of numbers, represent logical propositions that are either True or False
  - The arithmetic operations are replaced by the symbols $\wedge, \vee, \neg$ denoting logical AND, OR, and NOT operations

Semantics of AND, OR, and NOT

X	Y	$X \wedge Y$
False	False	False
False	True	False
True	False	False
True	True	True

X	Y	$X \wedge Y$
False	False	False
False	True	True
True	False	True
True	True	True

X	$\neg X$
False	True
True	True

- $X \wedge Y$ is True if *both* X and Y are True
- $X \vee Y$ is True if either X or Y or *both* are True
- $\neg X$ is True if and only if X is False

Boolean algebra at work

Suppose I want to state that "If it's sunny outside AND I have completed my work then (and only then) I will go for a run." To represent this in Boolean Algebra I introduce three Boolean variables, say, S, W, and R each of which may be `True` or `False`:

- S will represent if it is sunny outside or not
- W will represent if I have completed my work or not.
- R will represent if I go for a run or not.

Now "If it's sunny outside AND I have completed my work then (and only then) I will go for a run" can be expressed concisely as the formula $R = S \wedge W$.

Boolean Algebra at work

- "I get home early from work if I get to leave early or the traffic is light"
  - H = L OR T
  - $H = T \vee L$
- If (and only if) it is not sunny, I will play chess
  - NOT $S = C$
  - $\neg S = C$

Graphical notation for Boolean Logic operators

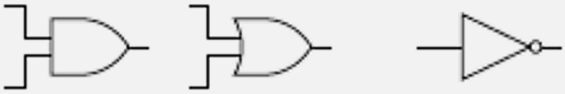


Figure 4.1: Graphical notation for 2-input AND, 2-input OR, and NOT Boolean operators

Boolean function composition

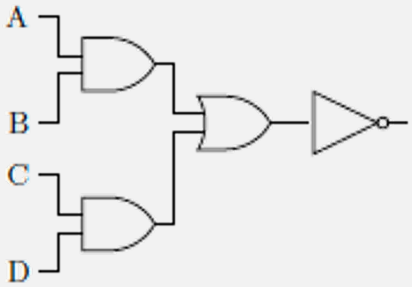


Figure 4.2: A compler Boolean function composed of $\wedge$, $\vee$ and $\neg$

$$\neg((A \wedge B) \vee (C \wedge D))$$

Boolean algebra - completeness

- The operators ∧,∨, ¬ are complete for Boolean algebra
- Any conceivable Boolean function can be expressed using only the operations ∧,∨, ¬
- The meaning of any complex Boolean formula can be understood in terms of the meanings of the Boolean variables and the meaning of the logical operators

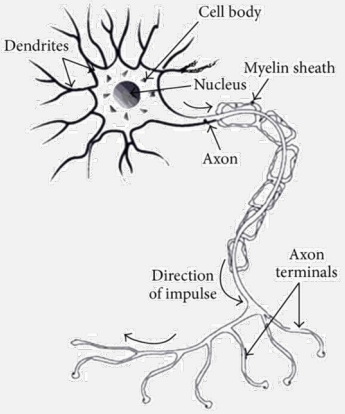
A	B	C	D	$f(A,B,C,D)$
False	False	False	False	True
False	False	False	True	True
False	False	True	False	True
False	False	True	True	False
False	True	False	False	True
False	True	False	True	True
False	True	True	False	True
False	True	True	True	False
True	False	False	False	True
True	False	False	True	True
True	False	True	False	True
True	False	True	True	False
True	True	False	False	False
True	True	False	True	False
True	True	True	False	False
True	True	True	True	False

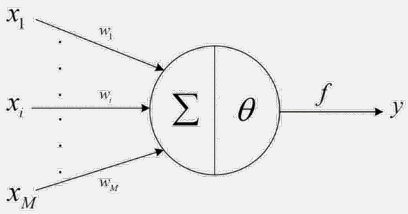
Boolean functions provide compact encodings of Truth Tables

$$f(A,B,C,D) = \neg((A \wedge B) \vee (C \wedge D))$$

Modern computer design relies on composition of complex Boolean functions from elementary Boolean functions, e.g., ∧,∨, ¬

McCulloch-Pitts Neurons

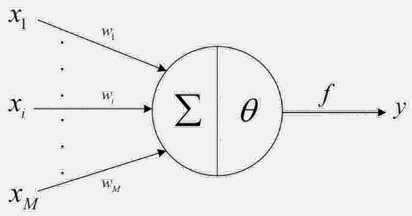




$$y = 1 \; if \sum_{i=1}^{M} w_i \, x_i \geq \theta$$

$$y = 0 \text{ otherwise}$$

McCulloch-Pitts Neurons



$$y = 1 \; if \sum_{i=1}^{M} w_i \, x_i \geq \theta$$

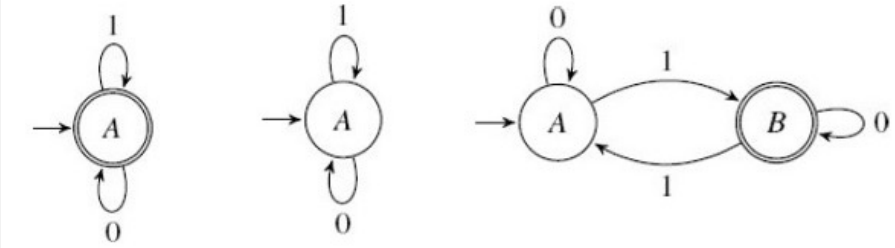
$$y = 0 \text{ otherwise}$$

- You can get the McCulloch Pitts neuron to compute the
  - Logical AND of its Boolean inputs if all the weights are set to 1 and threshold $\theta = M$
  - Logical OR of its Boolean inputs if all of the weights are set to 1 and the threshold $\theta = 1$
  - Logical NOT if $M = 1$, the single weight is -1 and $\theta = 0$
- An appropriately wired network of McCulloch Pitts neurons can implement any Boolean function

Finite automata

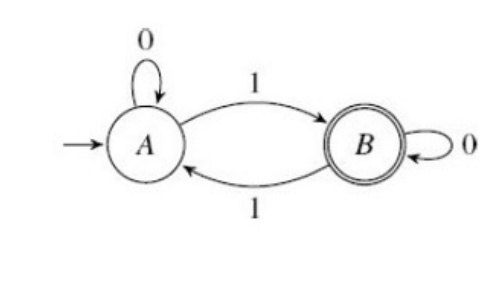
- Can be in a finite set of states which include the start state and one or more accept state(s).
- Input to the machine is a string or a sequence of symbols over a finite alphabet
- The machine starts in the start state and reads the first symbol of the input string
- A state transition table tells the machine what state to move to given the state it is currently in and the symbol it has just read
- The process halts when the machine reaches the end of its input string
- The machine `accepts' the string if the machine halts in an `accept state'. Otherwise the machine rejects the string.

Examples of finite automata



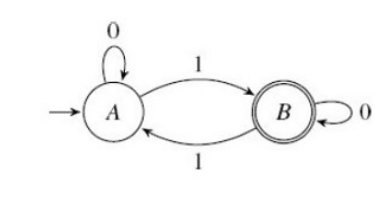
- The first automaton accepts all strings made of 0s and 1s
- The second automaton accepts no strings
- What does the third automaton do?

Exercise: What strings does the automaton accept?



- What happens with string 101 as input?
- What happens with string 100 as input?
- What about with string 1011 as input?

The automaton accepts strings with an odd number of 1s



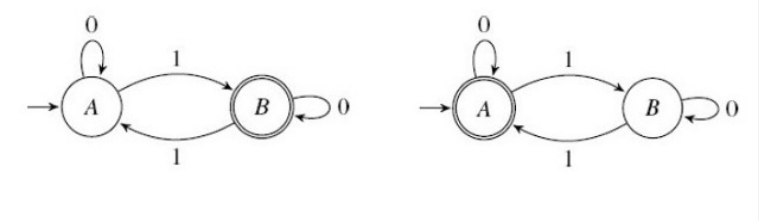
- We switch states only when the machine reads a 1
- We start in a non-accepting state A
- As long as the string is finite, the machine must end up in
  - A after reading an even number of 1s
  - B after reading an odd number of 1s
- What if the input string is infinite? The machine would run for ever.
- But we assume that the input string is finite

Language of a finite automaton

- The set of all strings accepted by a machine is called the language of the machine
- The languages accepted by finite automata, are called regular languages
- Can you give an example of a regular language?
  - The set of all binary strings
  - The set of binary strings with an odd number of 1s
  - …

Finite automaton solves decision problems

- Decision problem: Is the input string belong to the language of the automaton or not?
  - Answer can be Yes or No
  - Interchanging the accept and non-accept states of the machine corresponds to negating the question



  - There exist languages whose membership cannot be decided by finite automata but can be decided by Turing Machines

Turing Machine

- Hilbert and Ackerman (1928)
  - Decision Problem: "Is there an effective procedure, that allows us to decide, by means of finitely many operations, whether any given theorem is provable from a given set of premises?"
- Alan Turing (1936)
  - What is an effective procedure?
  - An effective procedure is an algorithm – step-by-step sequence of instructions that can be executed by a human or a machine
  - What kind of machine?
  - Turing Machine!

Turing's observations

Based on his observation of human computers at work, Turing concluded that a computer must be able to:

- Read the inputs of the computation (e.g., "267 + 856");
- Carry out the appropriate algorithm
  - or a sequence of simple steps like adding two digits in the 1's position
  - or stages in the computation
  - which Turing called the states of mind of the (human) computer
  - which later got abbreviated as simply states
- Write results to keep track of the results of each stage.

Turing's observations

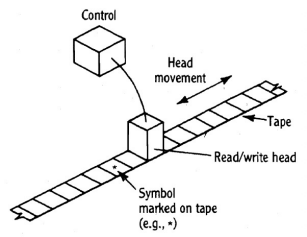
- The two-dimensional nature of the paper is irrelevant
- There is no reason to read more than one symbol at a time
- There is no reason to write more than one symbol at a time
- There is a need for the computer to step through states that keep track of the stage of the computation being carried out
  - Example: the digits in the one's position have been added and carry written

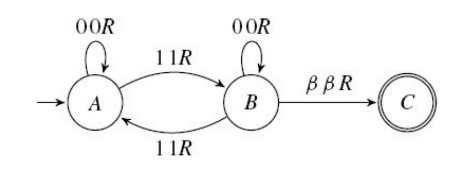
Turing's claim

- An effective procedure is simply an algorithm that tells a machine exactly what to do at any given step
- From the standpoint of executing such an algorithm human and the machine are functionally equivalent
  - Given the same input, they both can be seen as carrying out identical steps ending up with identical outputs
- The physical substrate used to implement the machine is immaterial
- All that is needed is the ability to manipulate symbols based on the syntax of the input
- One can build computers out of silicon, tinker toys, …

From Finite Automata to Turing Machine

- The finite automaton can be designed such that
  - The machine's current state and the input symbol that is read at the current position of the head
    - The symbol to be written on the tape
    - Determine the movement of the read/write head to the left or right or stays where it is
    - The next state of the machine

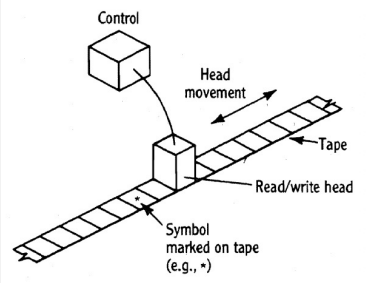




In state A, if you read a 0, write a 0, move R, and stay in A
In state A, if you read a 1, write a 1, move R, and enter state B

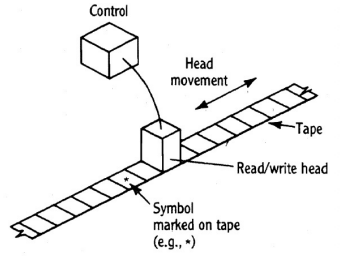
Turing Machine

- The tape is assumed to be infinite in both directions
- Ensures that there is enough space to write intermediate results
- At any given step, only a finite portion of the tape is used and the rest is blank



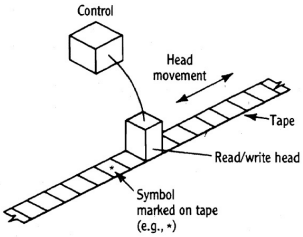
Universal Turing Machine

- Turing showed that there is a Universal Turing Machine – that can perform any computation that any other Turing Machine could possibly perform
- Input: The program of the target Turing machine, and its input
- Output: Output of the target machine



Turing Machine

- Formalizes the notion of an effective procedure
  - ✓An effective procedure is an algorithm executed by the Turing machine
- Note that this is a definition of an effective procedure
- If you agree with Turing's definition of an effective procedure, you have no choice but to accept that any algorithm whatsoever can be implemented by a Turing Machine
- If you disagree with the definition, the burden is on you to devise a better definition or at least show an example of a process that intuitively looks like an algorithm but cannot be realized by the Turing Machine



Church-Turing Thesis

- Many tried to investigate alternative formalizations of effective procedures
  - Alonzo Church came up with λ −calculus
  - Emil Post proposed Post productions
  - Andrey Markov proposed Markov Algorithms
  - Stephen Kleene proposed general recursive functions

- All of them were shown to be equivalent to Turing Machines
  - Any computation that any of the formalisms could perform could be performed by the Turing Machine and vice versa
- Church-Turing Thesis: Any function that can be computed by an algorithm can be computed by a Turing Machine
  - What does it mean to be computed by an algorithm?
  - Algorithm is a finite sequence of instructions
  - Input and output are of finite length
  - The computation must terminate after a finite number of steps

Implications of Church-Turing Thesis

- Any computer that is sufficiently powerful (Turing equivalent) can execute any program
- Program for one computer can be translated to program for another computer
- Programs written in one language can be translated into programs written in other languages
- Design of modern computers, programming languages, software .. rests on Church-Turing Thesis

Limits of computing machines

- There exist problems that are unsolvable by Turing machines and hence, any Turing-equivalent computing formalism
  - Halting problem: Deciding whether an arbitrary program eventually terminates on all inputs
  - Theorem proving in logic: Deciding whether a theorem logically follows from a set of axioms
    - If the theorem is true, there is a finite proof
    - If the theorem is false, it is possible that the prover runs for ever – there is no way to know if it is because the theorem is false or because the machine needs more time

Implications of Church-Turing Thesis for AI

- So long as intelligence has an algorithmic description, there are many ways to realize it that are functionally equivalent
  - Brains
  - Digital computers
  - Quantum computers
  - DNA computers

Implications of limits of computation for AI
Which side of the argument are you on?

- Argument 1: To the extent that intelligence can be modeled by computation, there will be some truths that are unknowable by both humans and machines
- Argument 2: Powers of mind exceed those of computers – Tantamount to saying that minds are beyond finite algorithmic description
- Argument 3: Most of our knowledge of the world comes from inductive and not deductive methods so the limits of computation which apply to deductive methods are irrelevant to AI

Can we build computers that are more powerful than Turing machines?

- We need to allow the machines to take advantage of something Turing machine can't:
  - Infinite precision numbers
    - There is no known physical substrate capable of supporting infinite precision calculations
  - Quantum computers – machines that make use of quantum as opposed to deterministic states
    - There is no evidence that quantum computers can circumvent the limitations of Turing machines
- We don't know if we can build computers that are more powerful than Turing Machines