



ECE408/CS483/CSE408 Fall 2022

# Applied Parallel Programming

## Lecture 10: Machine Learning and Deep Learning

# Course Reminders

- Lab 4 out, it is due this week
- Midterm 1 is on Tuesday, October 11<sup>th</sup>
  - On-line, everybody will be taking it at the same time
    - Tuesday, Oct. 11<sup>th</sup> 7:00pm-8:20pm US Central time
    - Wednesday, Oct. 12<sup>th</sup> 8:00am-9:20am Beijing time
  - Includes materials from Lecture 1 through Lecture 10
- Project Milestone 1: Baseline CPU implementation is due Friday October 14<sup>th</sup>
  - Project details to be posted next week

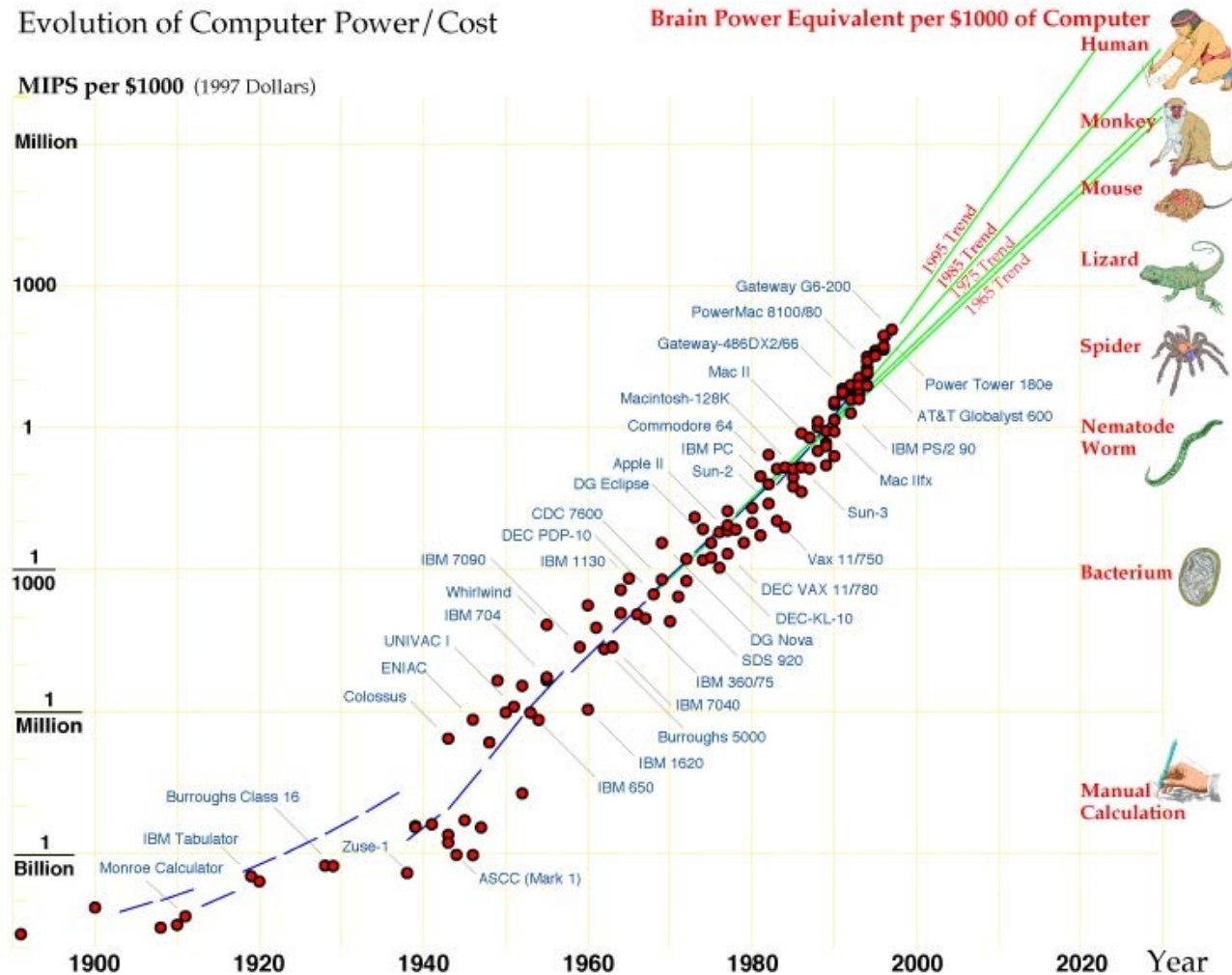
# Objective

- To understand the application areas for machine learning.
- To learn the basic strategy for machine learning applications.
- To understand the extension to deep learning (mostly a research pitch).
- To learn about a Multi-Layer Perceptron.

# Perspective is Important

- **Chips are cheaper than ever**
- Unlike humans, digital systems offer
  - **high-speed computation**,
  - **low capital investment**  
(purchase vs. training a human), and
  - **negligible operations cost** (no salary!)
- **If computer outperforms** (or even matches) a **human, use a computer**
- **Industry has done so for 40-50 years now**

# Evolution of Computer Power/Cost



Hans Moravec, 1997

Computing has evolved under the premise that some day, computing machines will be able to mimic general human intelligence.

From a computing power perspective, Moore's Law has fueled the idea of the intelligent machine. Hardware has gotten 2x faster every 18 months.

The software, though, has been a vexing open question.

<https://jetpress.org/volume1/moravec.htm>

# What is Machine Learning?

- **Machine learning**: important method of building applications whose logic is not fully understood
- Typically **by example**:
  - **use labeled data** (matched input-output pairs)
  - **to represent** desired **relationship**
- **Iteratively adjust program logic** to produce desired/approximate answers (called **training**)

# Types of Learning Tasks

- classification
  - Map each input to a category
  - Ex: object recognition, chip defect detection
- regression
  - Numerical prediction from a sequence
  - Ex: predict tomorrow's temperature
- transcription
  - Unstructured data into textual form
  - Ex: optical character recognition

# More Advanced Learning Tasks

- translation
  - Convert a sequence of symbols in one language to a sequence of symbols in another
- structured output
  - Convert an input to a vector with important relationships between elements
  - Ex: natural language sentence into grammatical structure
- others
  - Anomaly detection, synthesis, sampling, imputation, denoising, density estimation, genetic variant calling



# Why Machine Learning Now?

- **Computing Power**

- GPU computing hardware and programming interfaces such as CUDA has enabled very fast research cycle of deep neural net training

- **Data**

- Lots of cheap sensors, cloud storage, IoT, photo sharing, etc..

- **Needs**

- Autonomous Vehicles, Smart Devices, Security, Societal Comfort with Tech, Health Care

# Test Cycle Time is Important

You've all written code...

- code, test, code, test, code, test
- integrate, test, test, test
- and test again!

**But how long is the code, test cycle?**

**Depends what you're building.**

**What's your longest?**

# Your Cycle Times are Probably Small

- In college, **10k lines** took **½ hour** to compile on my PC.
- In grad. school, **100k lines** took
  - **½ hour** to compile on my workstation, or
  - **2 minutes** on our cluster (research platform)
- In ECE 435 (networking lab), students needed
  - **½ hour** to reinstall Linux after a bad bug
  - (Ever had a good bug?)
- Gene sequencing / applications can take **two weeks**

**We're all a little spoiled...**

# Why Machine Learning Again?

- In 2007, **programmable GPUs accelerated the training cycle**
- Today, **new chip designs** for learning applications **have further accelerated**
- Led to a resurgence of interest
  - in Computer Vision, Speech Recognition, Document Translation, Self Driving Cars, Data Science...
  - all tasks that **human brains solve regularly, but** for which **we** have **struggled to express solutions** systematically.

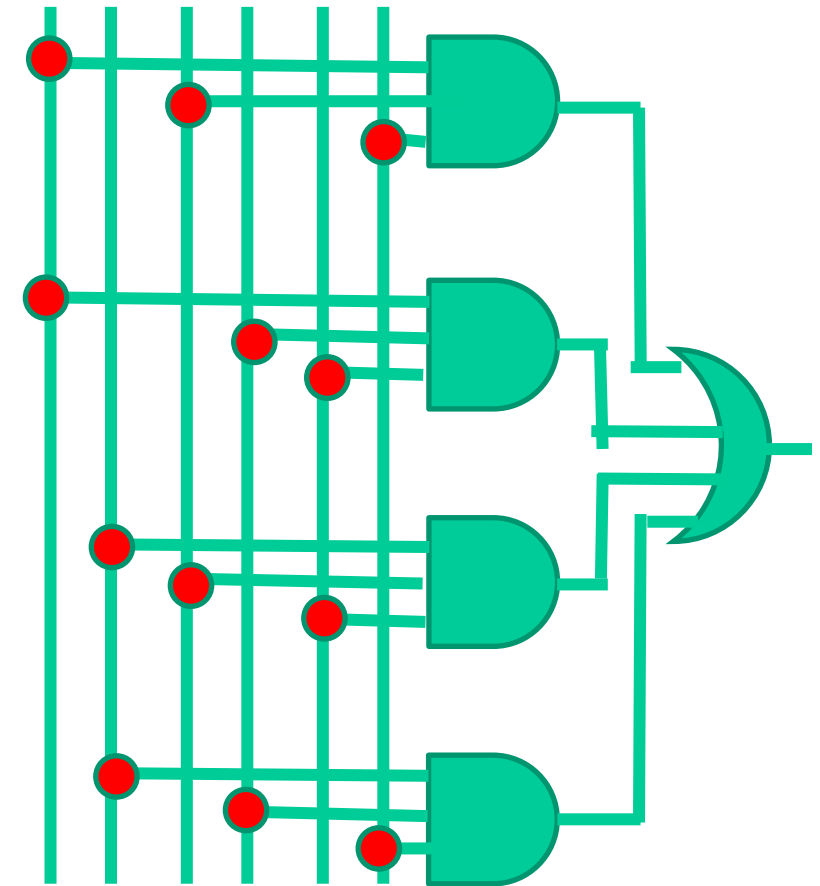
# Many Problems are Still Hard

- **Speed is not a panacea**
- Many **tasks still require human insight**
  - for network structure and feature selection
  - for effective input and output formats, and
  - for production of high-quality labeled data.
- Other trends sometimes help: ubiquitous computing enables crowdsourcing, for example.

# Many Problems Have Systematic Solutions

Example: building a Boolean function from a truth table

Input			output
a	b	c	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



# What if We Lack a Truth Table?

- Make enough observations to construct a rule
  - $000 \rightarrow 0$
  - $011 \rightarrow 0$
  - $100 \rightarrow 1$
  - $110 \rightarrow 0$
- If we cover all input patterns,  
we can construct a truth table!

# Many Problems are Too Large

- The logic formulation of a 32x32-pixel (small) image recognition problem involves
  - 1024\*8 bit input,
  - which will have a truth table of  $2^{8196}$  entries
- If we managed to collect and label 1 billion ( $\sim 2^{32}$ ) images as training data
  - We cover only  $2^{32} / 2^{8196} = 1 / 2^{8164}$  of the truth table
  - Solution - learning processes that exploits features



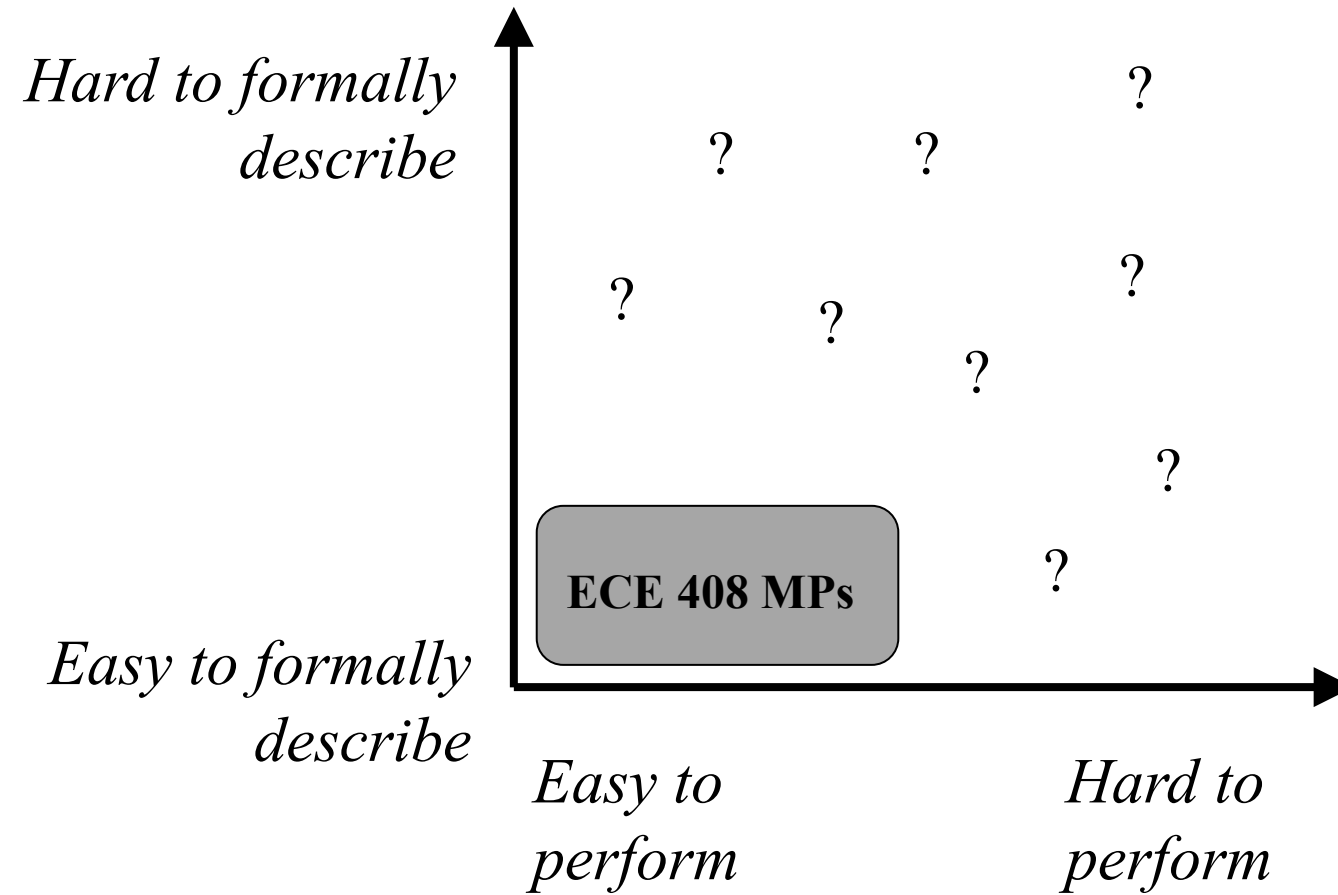
# Features in our logic example

Input			output
a	b	c	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

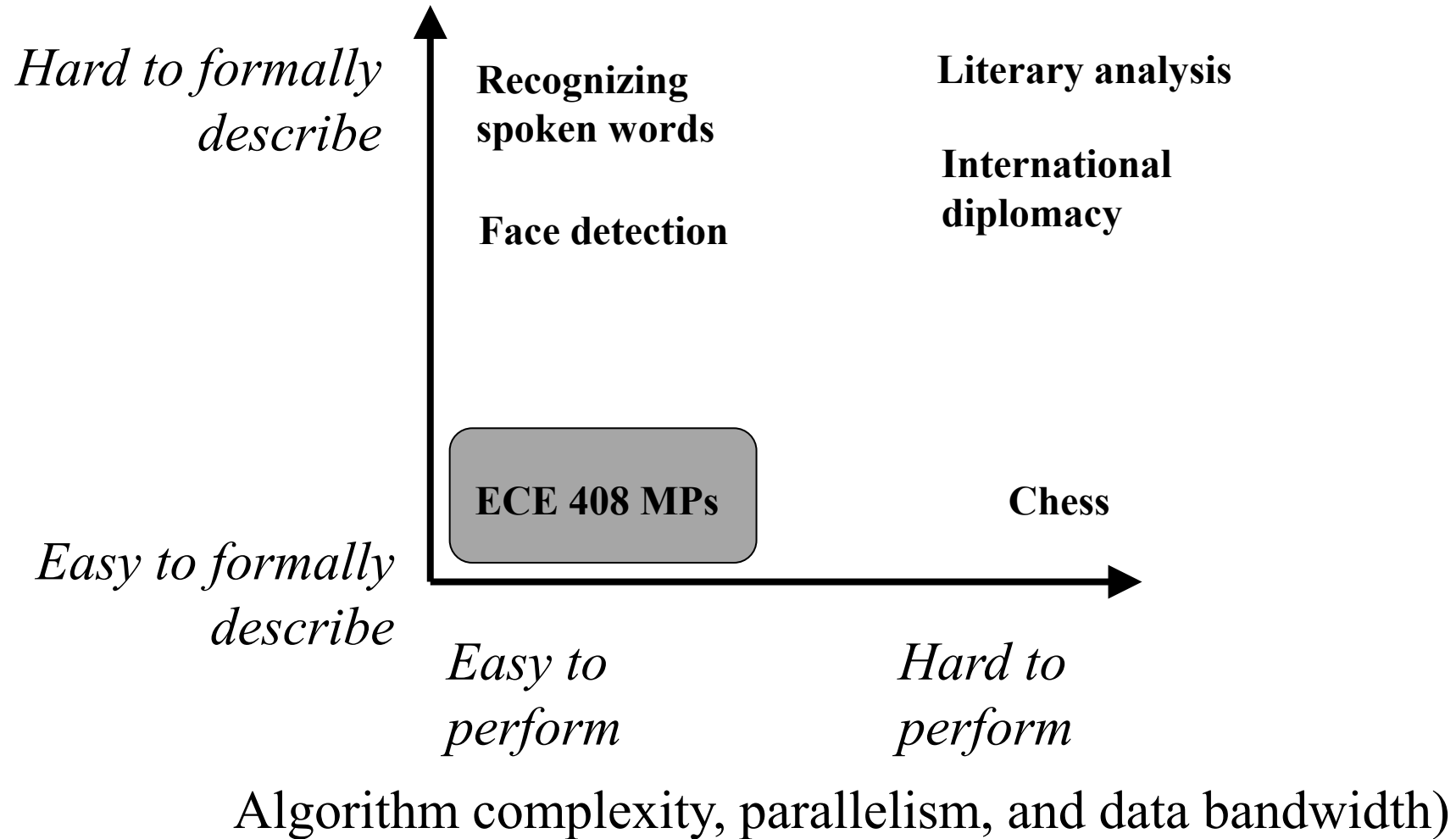
Feature 1: bit patterns with odd number of 1's result in output 1

Feature 2: bit patterns with even number of 1's result in output 0

# Types of Problems

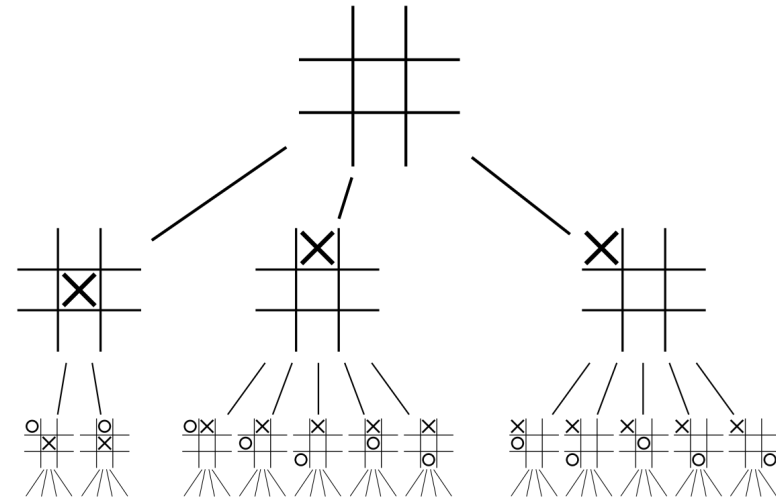


# Types of Problems



# Chess as an AI Success (1)

- Easy to formalize
  - 64 locations, 32 pieces
  - Well-defined, allowable moves
- Score each leaf in a tree of possible board positions
- Proceed down path that results in best position



2-ply game tree for tic-tac-toe

# Chess as an AI Success (2)

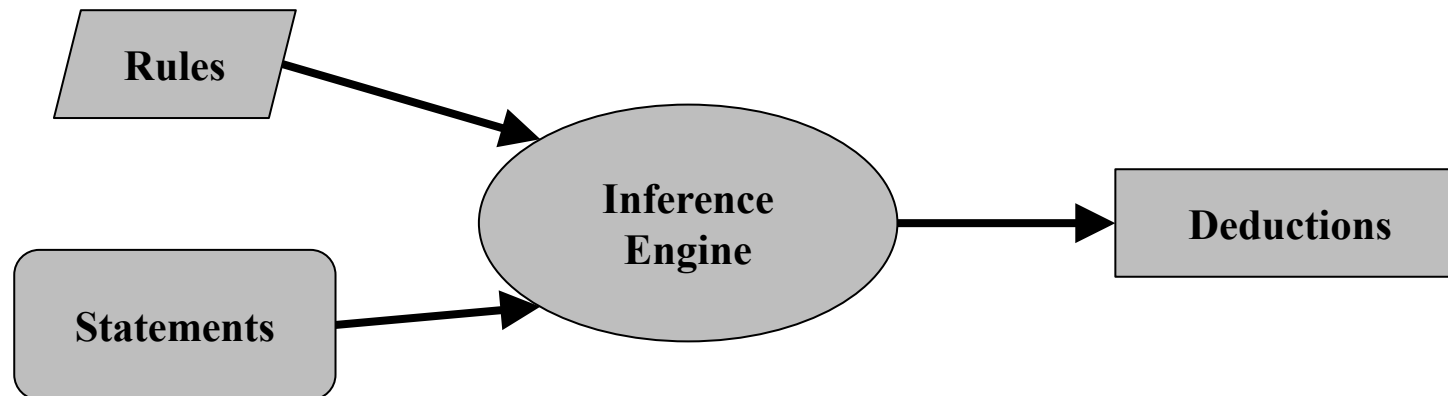


Deep Blue defeated Gary Kasparov  
in 1997

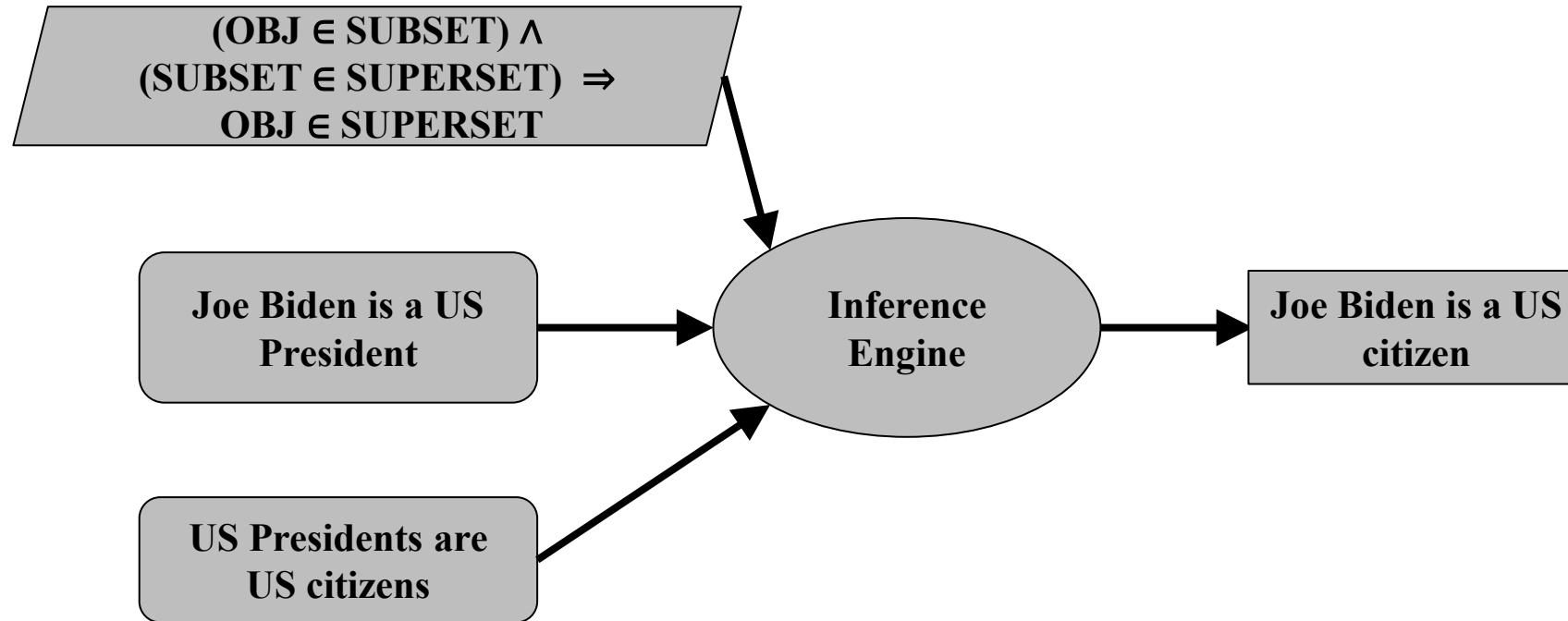
- Hard to perform
  - ~30 legal moves per position
  - 1,015 moves for 10-ply lookahead
  - 30 years of compute at 1M positions/sec
- Heuristics, pruning, parallel search, fast computers

# Cyc: Extending Rule-based Systems to the Real World

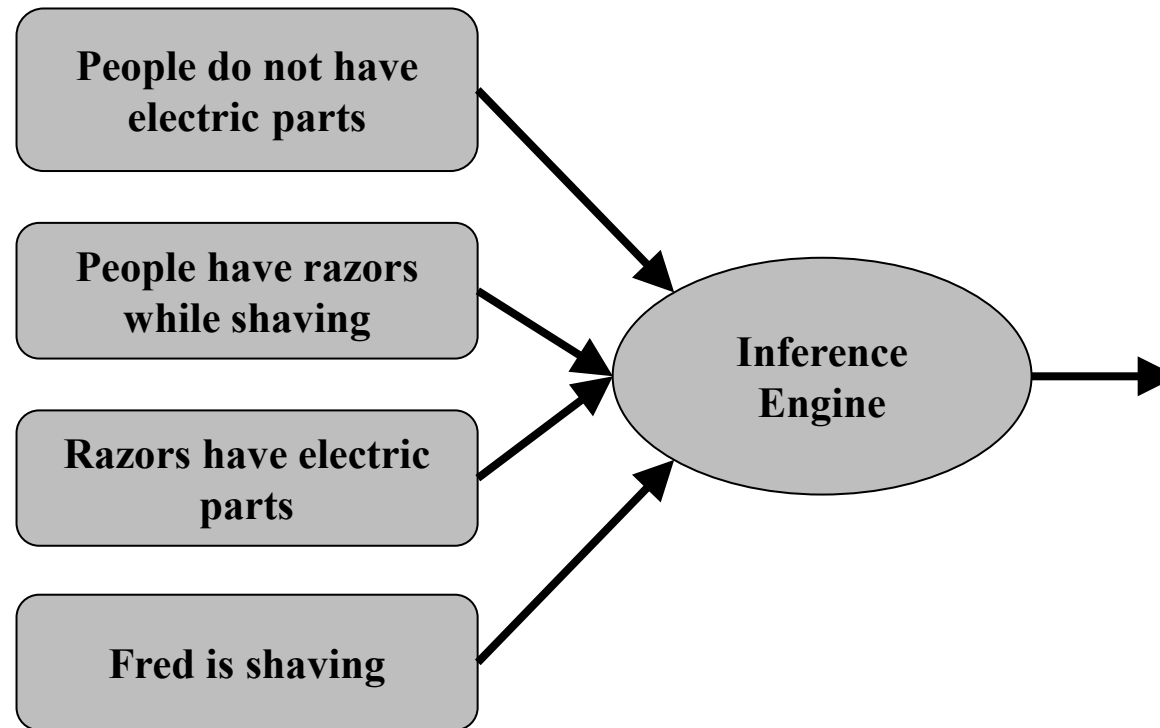
- Comprehensive ontology and knowledge base of common sense
- Cyc reasons about formal statements about the world



# Cyc: A Simple Example

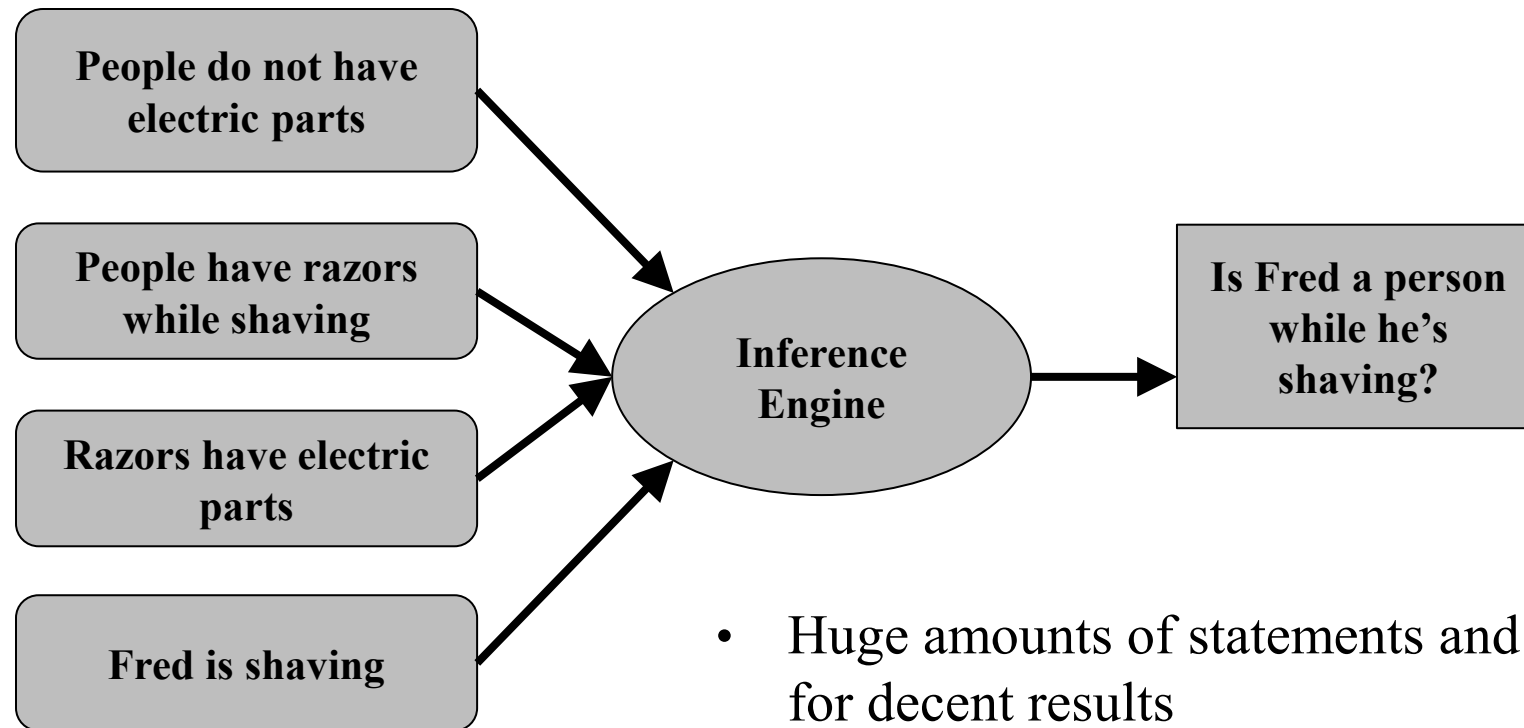


# Cyc: FredWhileShaving



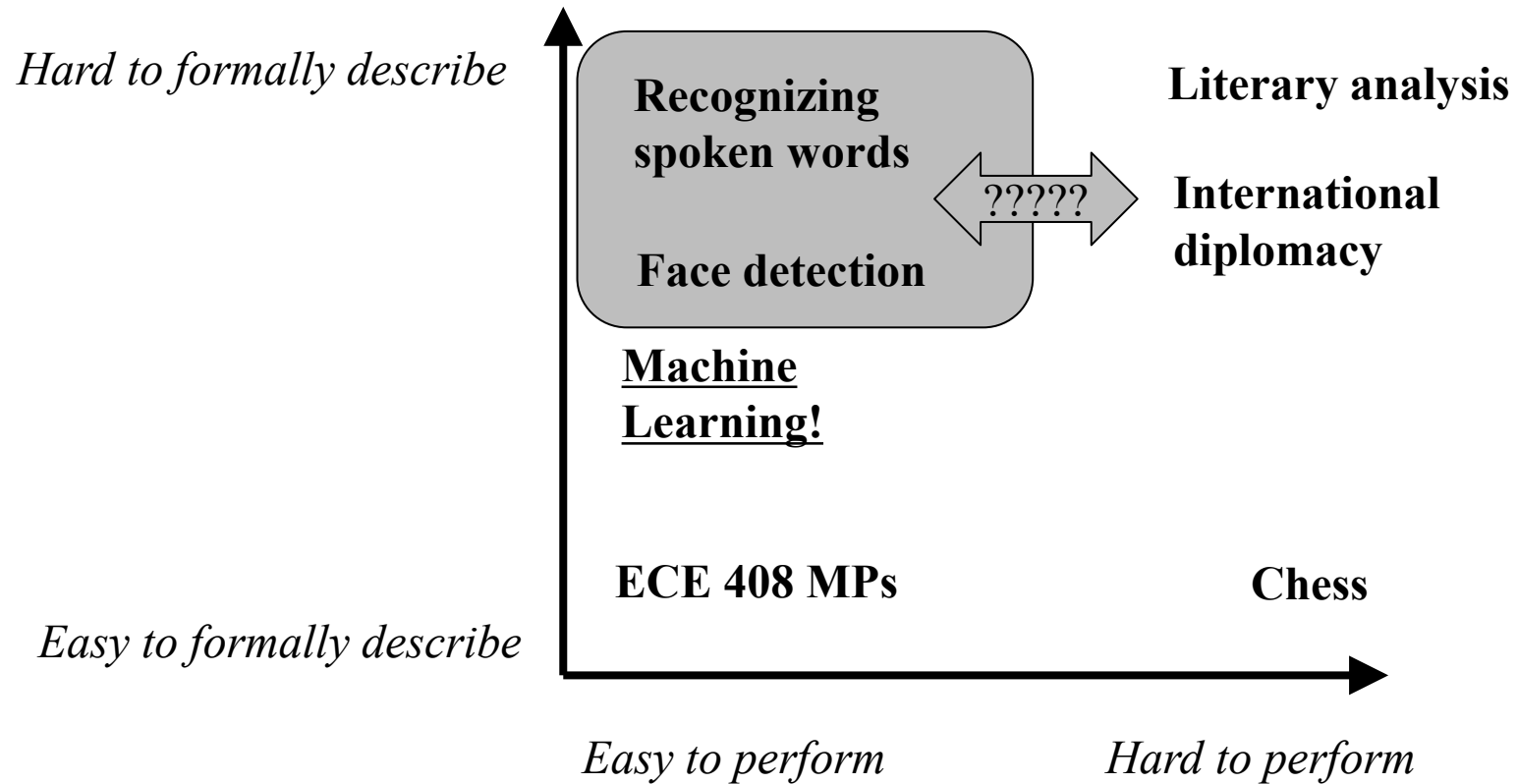


# Cyc: FredWhileShaving



- Huge amounts of statements and rules for decent results
- Cannot learn new rules or statements on its own

# Types of Problems



# The “Machine Learning” Approach

## Challenge

Hard to formalize the problem.

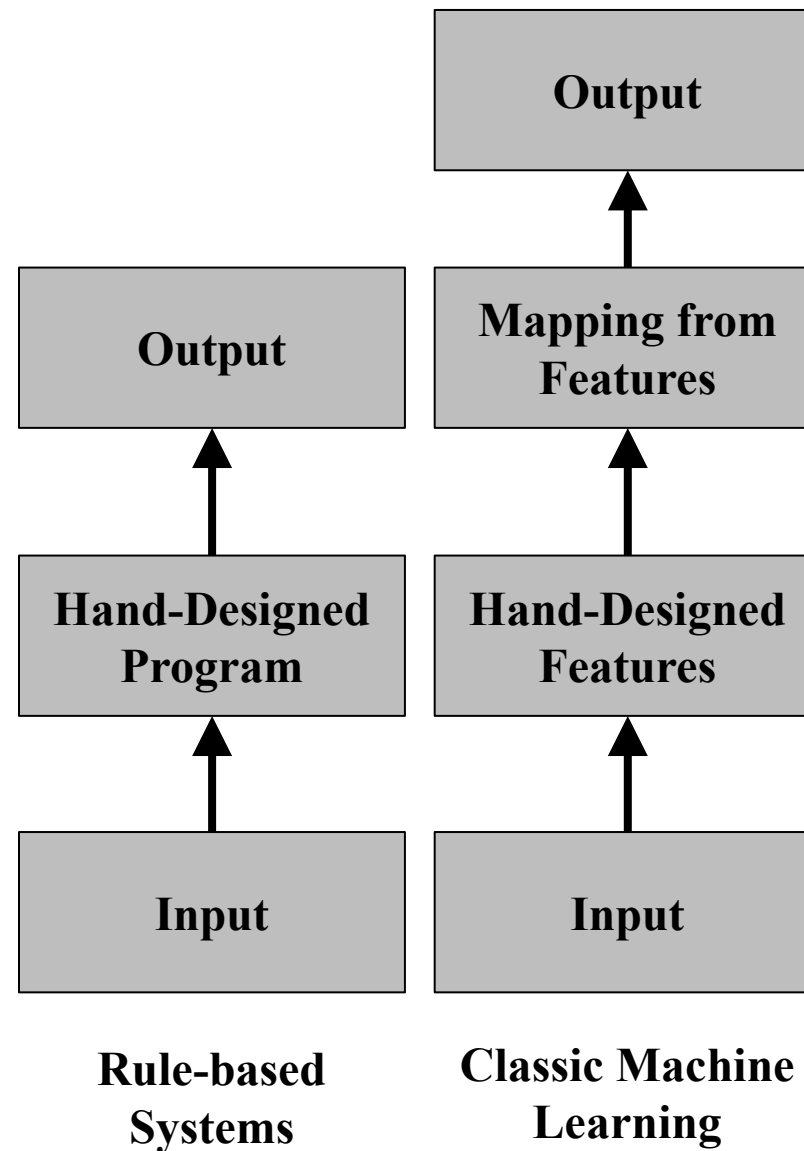
## Solution

Don't formalize the problem.

Let the machine learn from experience.

# Classic Machine Learning

- Humans choose features
- Learn how features are associated with outputs

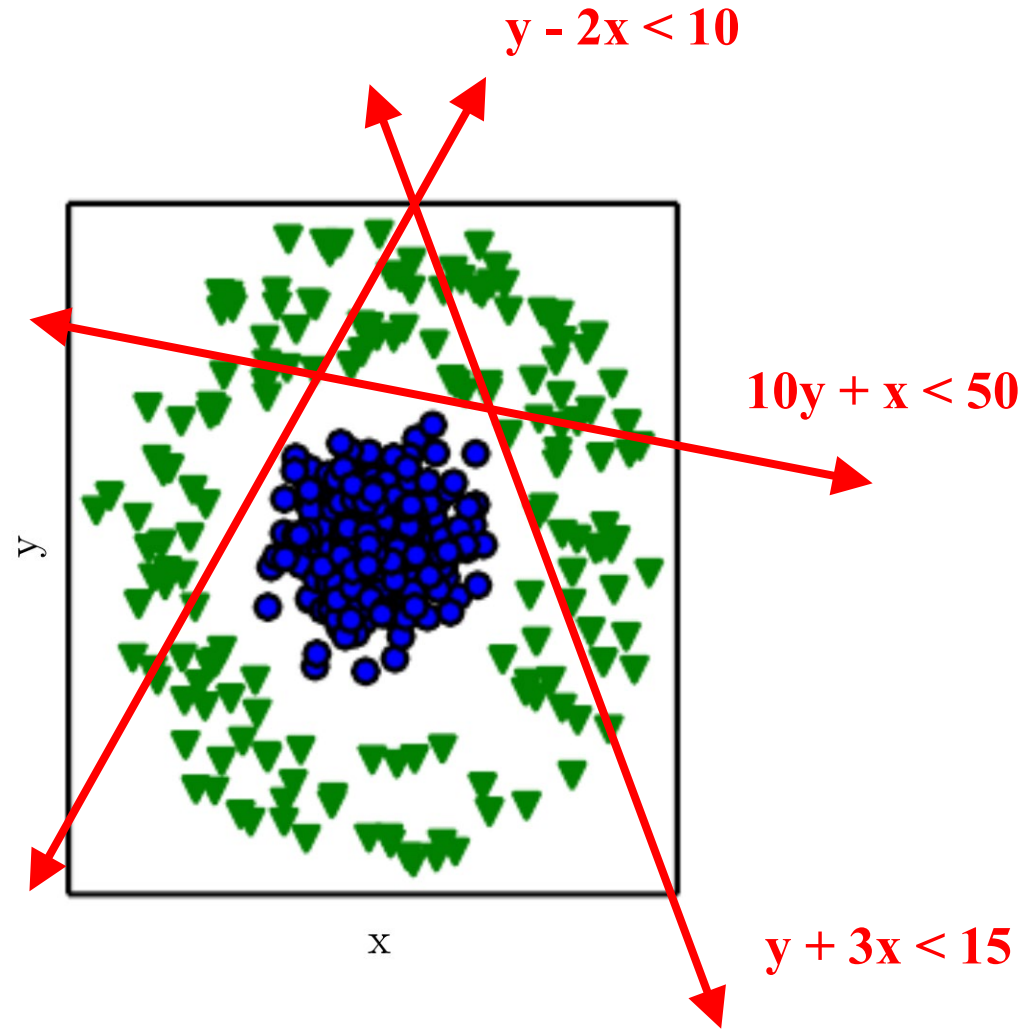


# You may have heard of...

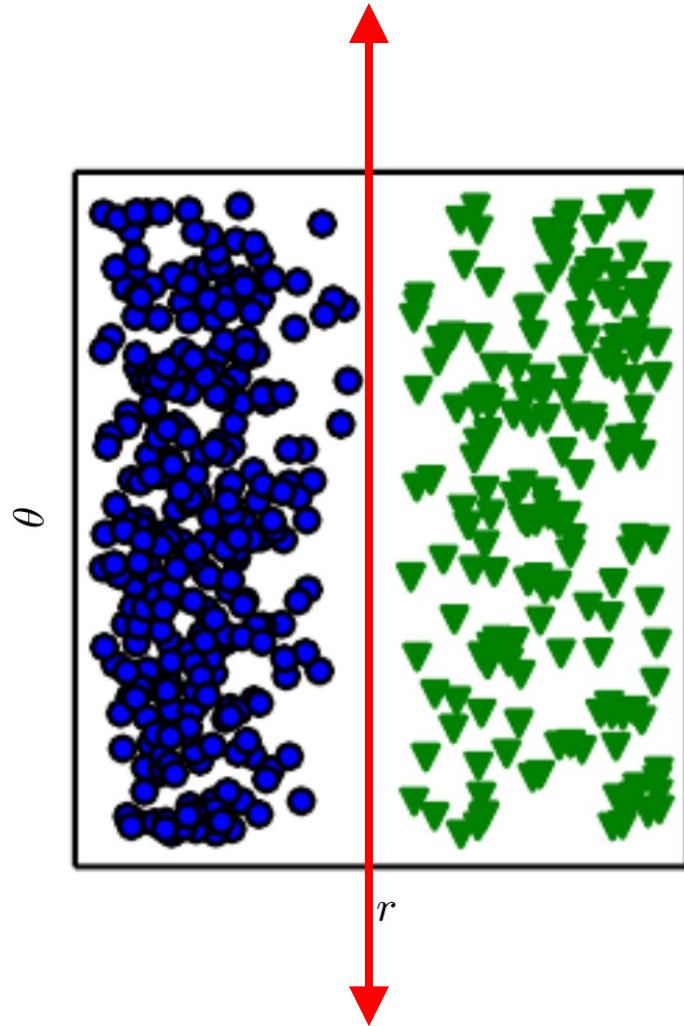
- Naïve Bayes:  
features as independent contributors to output
- Logistic Regression:
  - learn how to weight each feature's contribution to output,
  - usually through gradient descent\*

\*more on this topic later in these slides

# Data Representation is important!



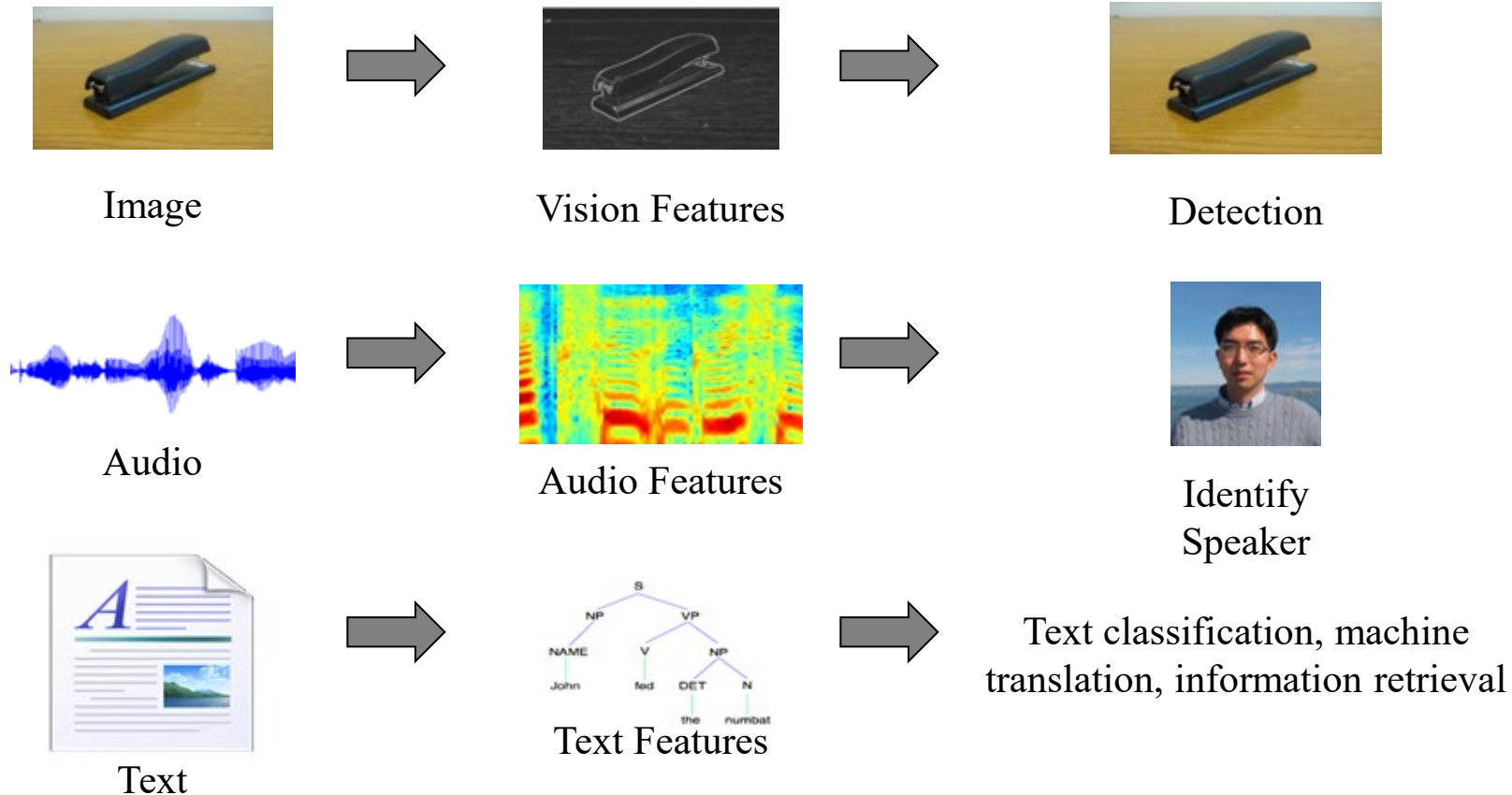
# Data Representation is important!



$$\Theta = \arctan( y / x )$$

$$r = \text{sqrt}( x^2 + y^2 )$$

# Different Features for Different Tasks





# Which Data Features are Relevant

- Detecting a car in an image
- Cars have wheels ➡ presence of a wheel?
- Can we describe pixel values that make up a wheel?
  - Circle-shaped?
  - Dark around perimeter?
- But what about?
  - Occlusion, perspective, shadows, white-walled tires, ...

# Identify Factors of Variation that Explain Data

- Unobserved objects or forces that affect observed quantities
- Mental constructs that provide simplifying explanations or inferred causes
- Ex: speech
  - Age, sex, accent, words being spoken
- Ex: car
  - Position, color, angle of sun
- Many factors influence each piece of observed data

# Representation Learning Approach

## Challenge

Which data features are relevant?

## Solution

Learn the features too!

(Looking ahead)

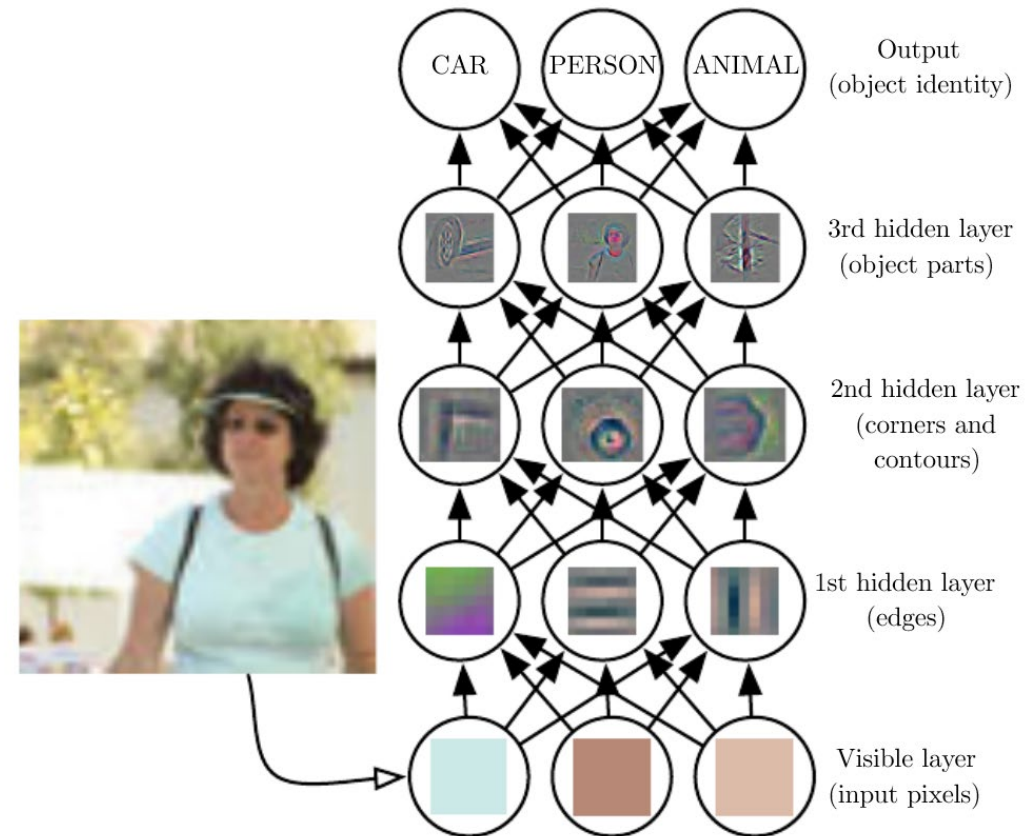
Deep Learning: a deep hierarchy of features

# Machine Learning

- Ability to acquire knowledge by extracting patterns from data

# Deep Learning

- A type of representation learning
- Representations expressed in terms of other representations



# Deep Learning Approach

## Challenge

Hard to formalize the problem?

Which data features are relevant?

## Solution

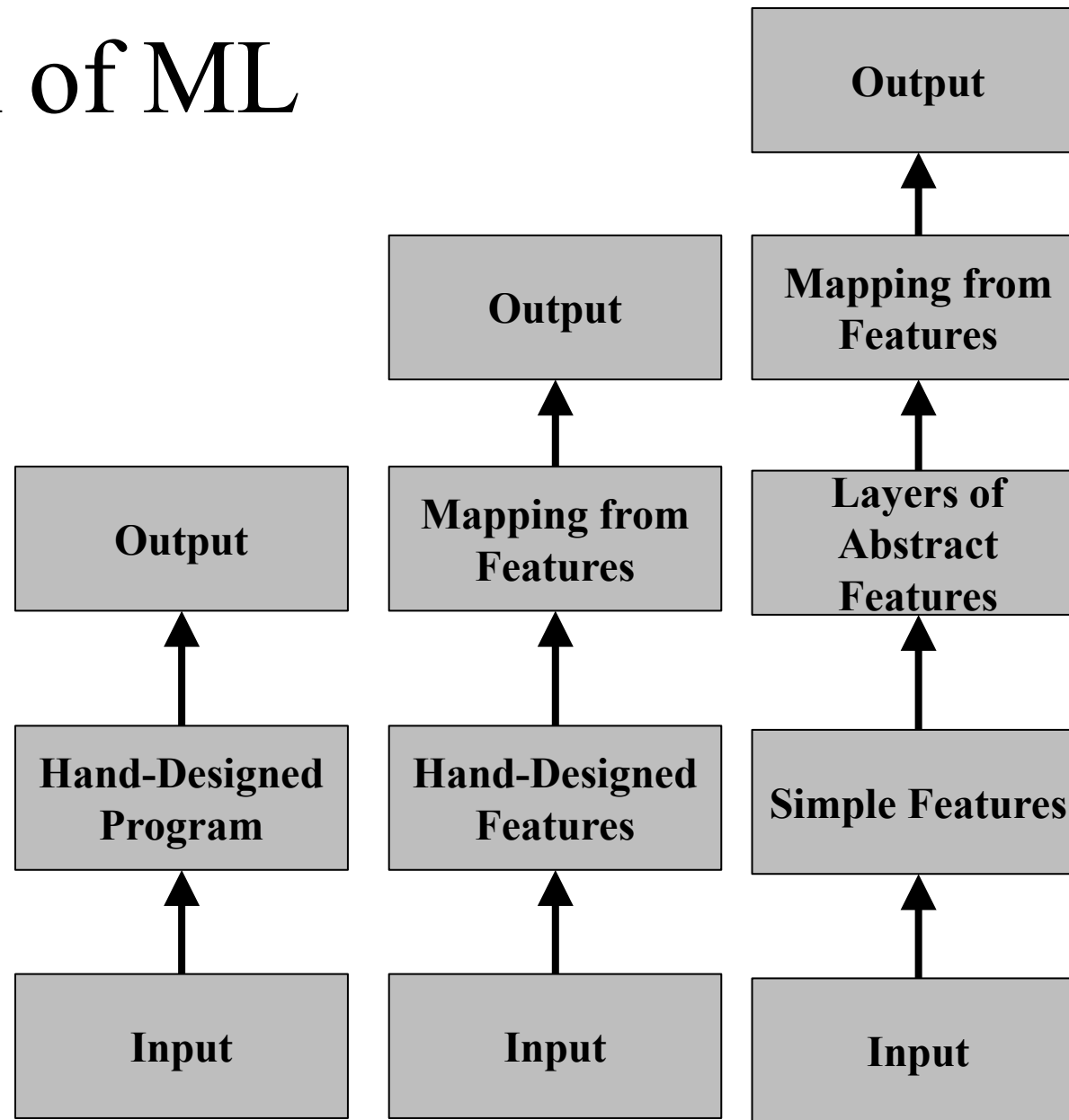
Don't formalize the problem

Let the machine learn from experience

Hierarchy of concepts to capture simple and complicated features

Learn the hierarchy too!

# Evolution of ML



**Rule-based Systems**

**Classic Machine  
Learning**

**Deep Learning**

# Let's Look at Classification

In a **classification problem**, we model

- a function mapping an input vector to a set of  **$C$**  categories:  **$F : \mathbb{R}^N \rightarrow \{1, \dots, C\}$** ,
- where the function  **$F$  is unknown**.

We **approximate  $F$  using a set of functions  $f$**

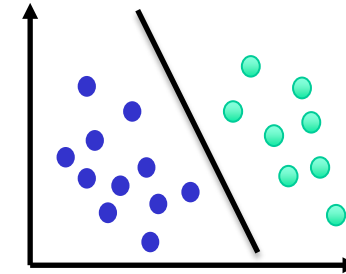
- parametrized by a (large) set of weights,  **$\theta$**
- that map from a vector of  **$N$**  real values\* to an integer value representing a category:
- for category  $i$ ,  **$\text{prob}(i) = f(x, \theta)$**

\*floating-point values



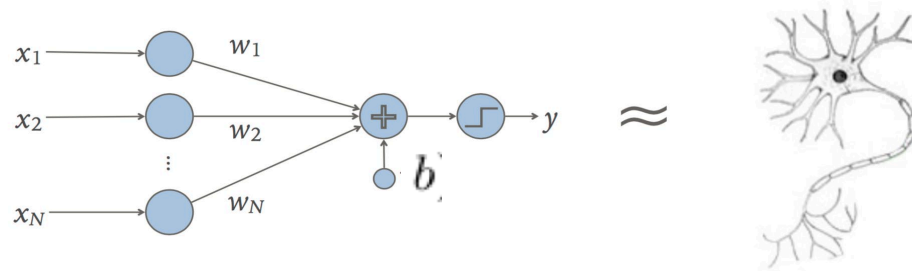
# Linear Classifier (Perceptron)

- our formulation:  $y = f(\mathbf{x}, \Theta)$   
 $\Theta = \{W, b\}$   
 $y = \text{sign}(W \cdot \mathbf{x} + b)$



The perceptron

The neuron



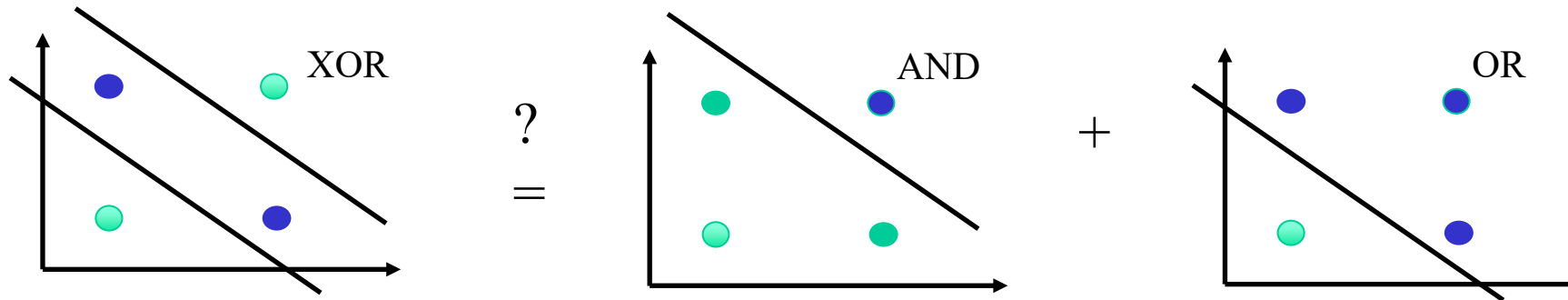
- Dot product + Scalar addition:

$$y = W \cdot \mathbf{x} + b$$

Diagram illustrating the equation  $y = W \cdot \mathbf{x} + b$  with labels for the components:

- $y$ : output
- $W$ : weight
- $\mathbf{x}$ : input
- $b$ : bias

# Can we learn XOR with a Perceptron?

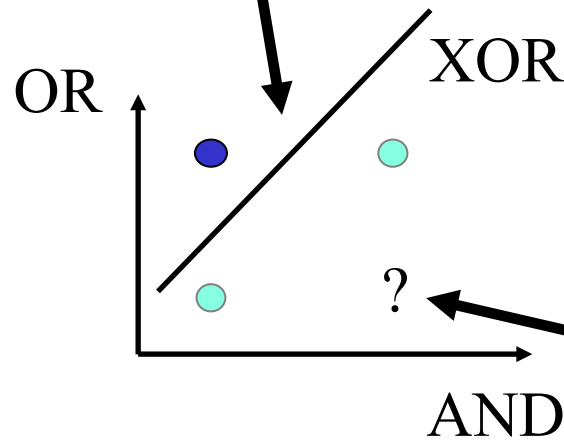


# Multiple Layers Solve More Problems

**What if input dimensions are AND and OR?**

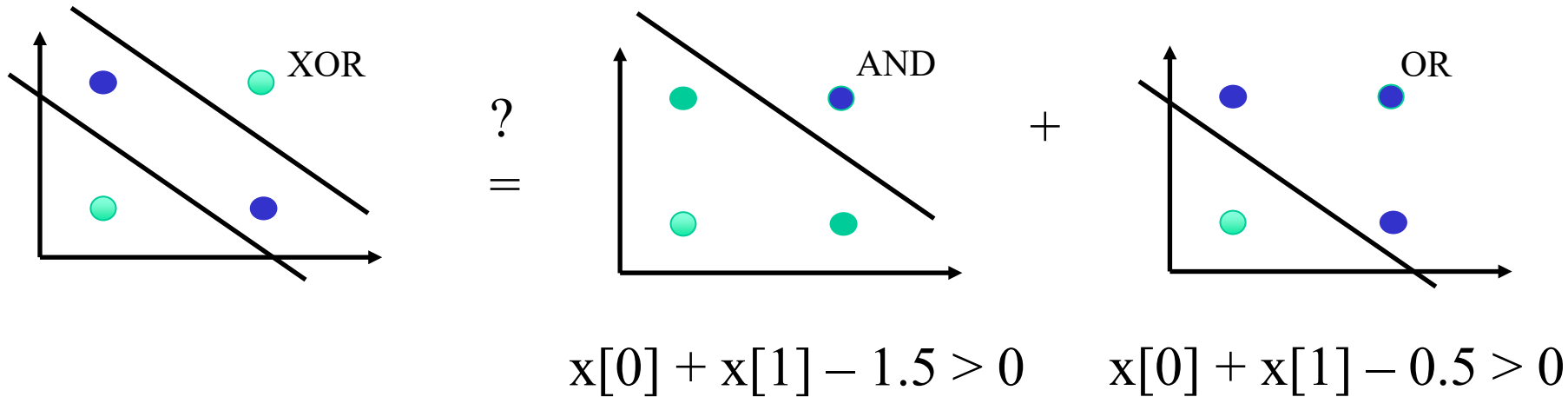
Now we can divide  
with one line.

● FALSE  
● TRUE



This combination  
is impossible!

# Perceptron



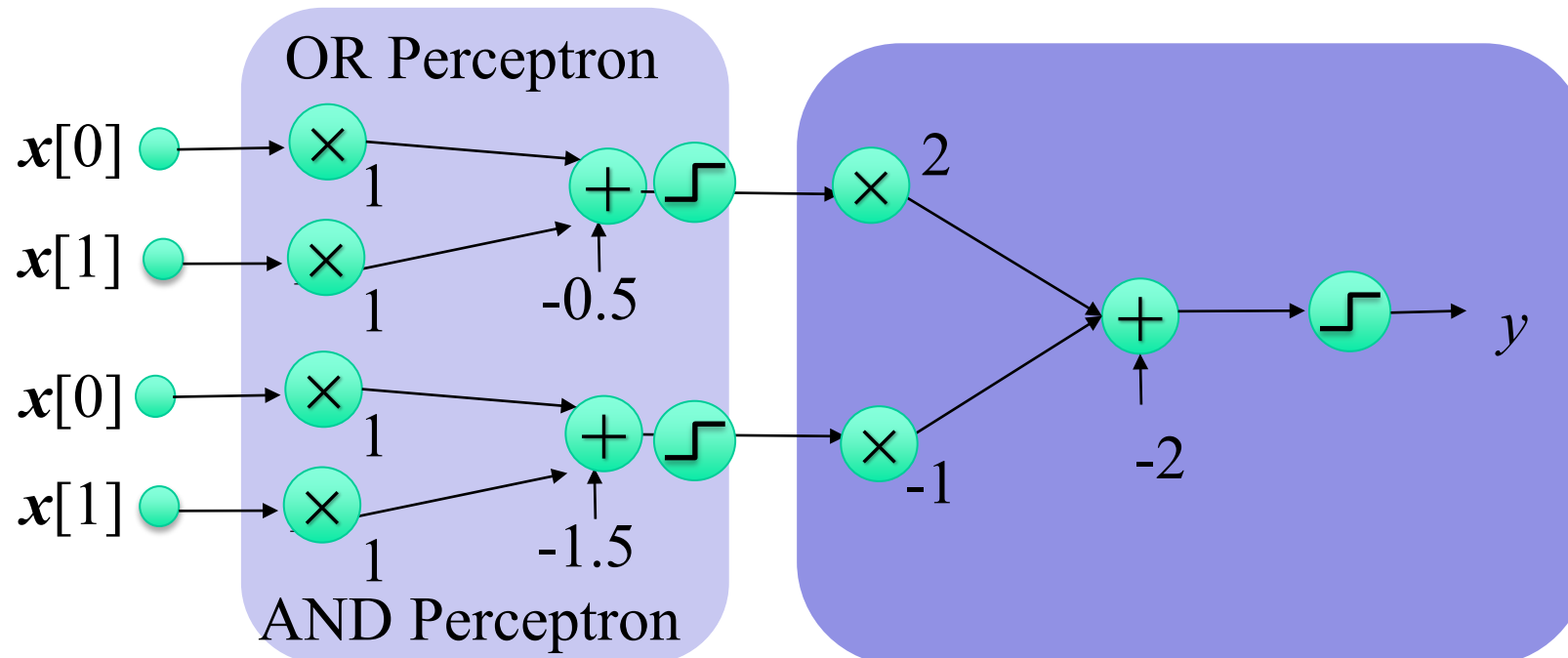
<b>x[1]</b>	<b>x[0]</b>	<b>AND</b>	<b>OR</b>	<b>XOR</b>
0	0	-1 (-1.5 < 0)	-1 (-0.5 < 0)	-1 (-2.0 < 0)
0	1	-1 (-0.5 < 0)	1 ( 0.5 > 0)	?
1	0	-1 (-0.5 < 0)	1 ( 0.5 > 0)	?
1	1	1 ( 0.5 > 0)	1 ( 1.5 > 0)	1 (2.0 > 0)

XOR is not a linear combination of AND and OR functions.

$x[1]$	$x[0]$	AND	OR	XOR
0	0	-1	-1	-1 (-3 < 0)
0	1	-1	+1	1 (1 > 0)
1	0	-1	+1	1 (1 > 0)
1	1	+1	+1	-1 (-1 < 0)

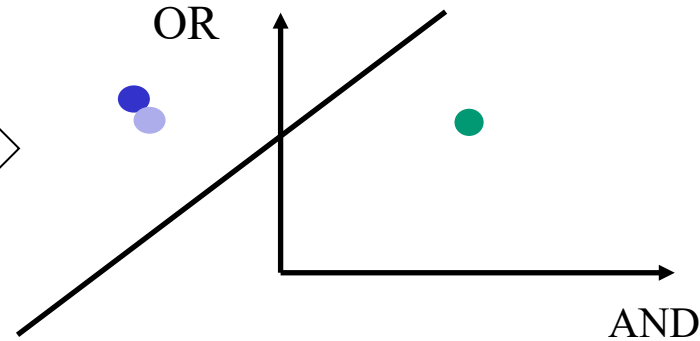
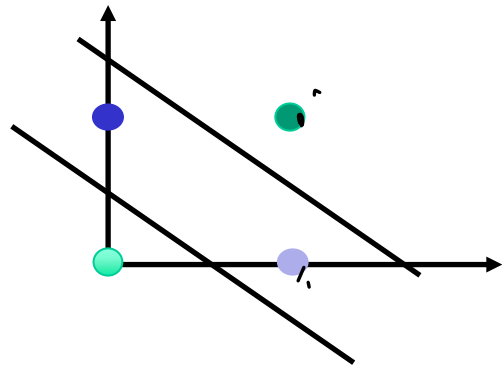
$OR = \text{sign}(x[0] + x[1] - 0.5)$   
 $AND = \text{sign}(x[0] + x[1] - 1.5)$

$\text{sign}()$  function adds non-linearity to  
 “reposition” data points for the next layer.



$$XOR = \text{sign}(2*OR + -1*AND - 2)$$

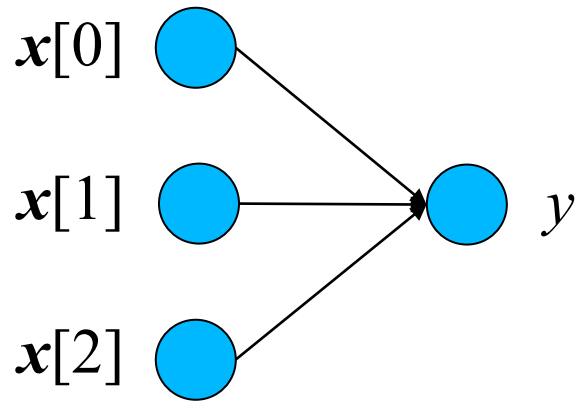
# Multi-Layer Perceptron – data repositioning



  $\text{XOR} = \text{sign}(2 * \text{OR} + -1 * \text{AND} - 2)$

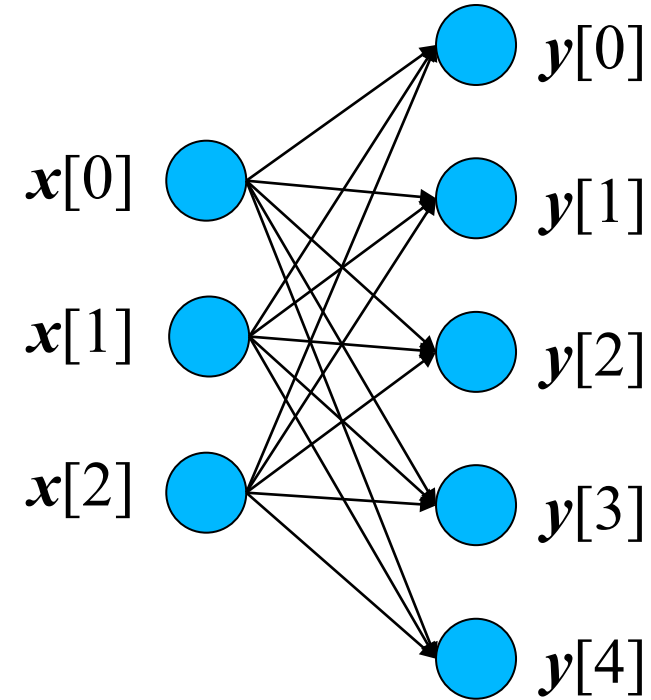
x[1]	x[0]	AND	OR	XOR
0	0	-1	-1	-1 (-3 < 0)
0	1	-1	+1	1 (1 > 0)
1	0	-1	+1	1 (1 > 0)
1	1	+1	+1	-1 (-1 < 0)

# Generalize to Fully-Connected Layer



Linear Classifier:

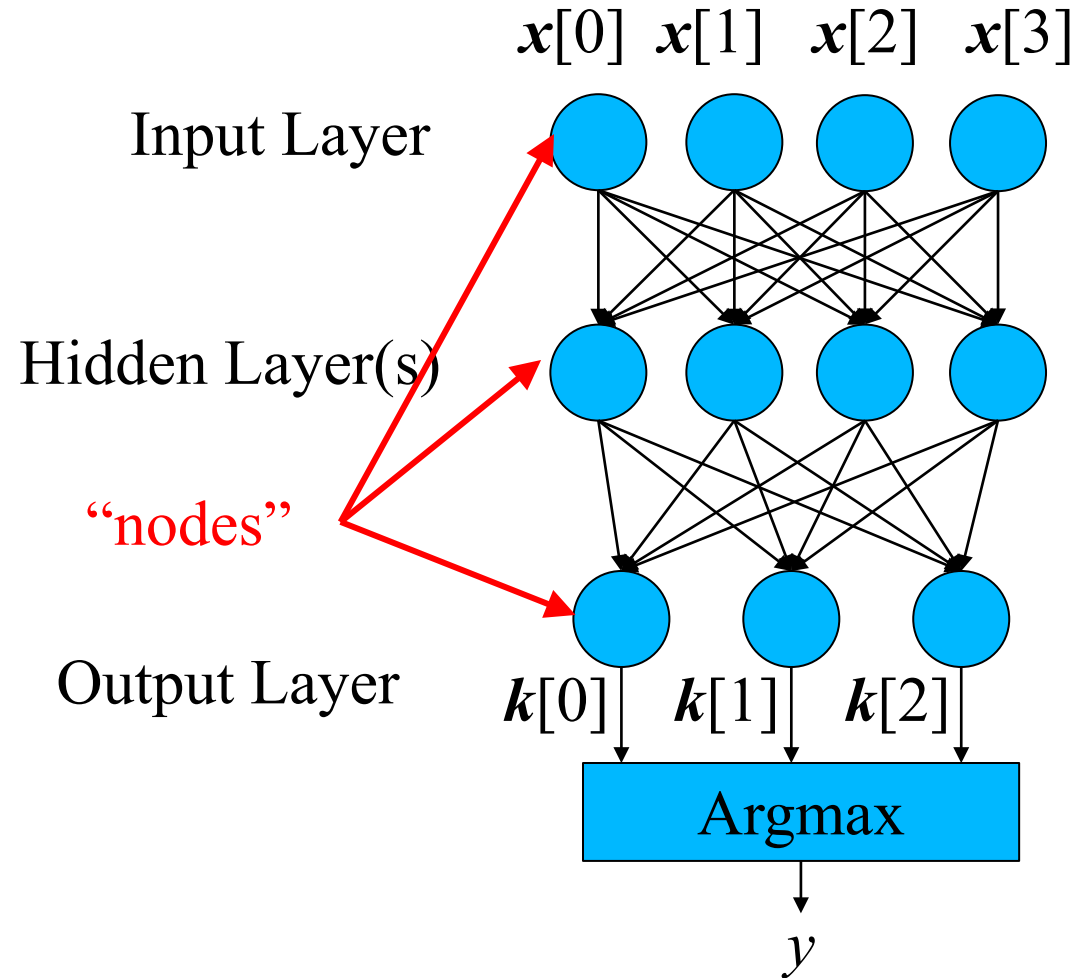
Input vector  $\mathbf{x}$   $\times$  weight vector  $\mathbf{w}$  to produce scalar output  $y$



Fully-connected:

Input vector  $\mathbf{x}$   $\times$  weight matrix  $\mathbf{w}$  to produce vector output  $y$

# Multilayer Terminology



$W_k[i, j]$  : weight between  $i^{\text{th}}$  input and  $j^{\text{th}}$  output of the  $k^{\text{th}}$  layer

$W_1$  is  $[4 \times 4]$ ,  $b_1$  is  $[4 \times 1]$

$W_2$  is  $[4 \times 3]$ ,  $b_2$  is  $[3 \times 1]$

Probability that input is class  $k[i]$



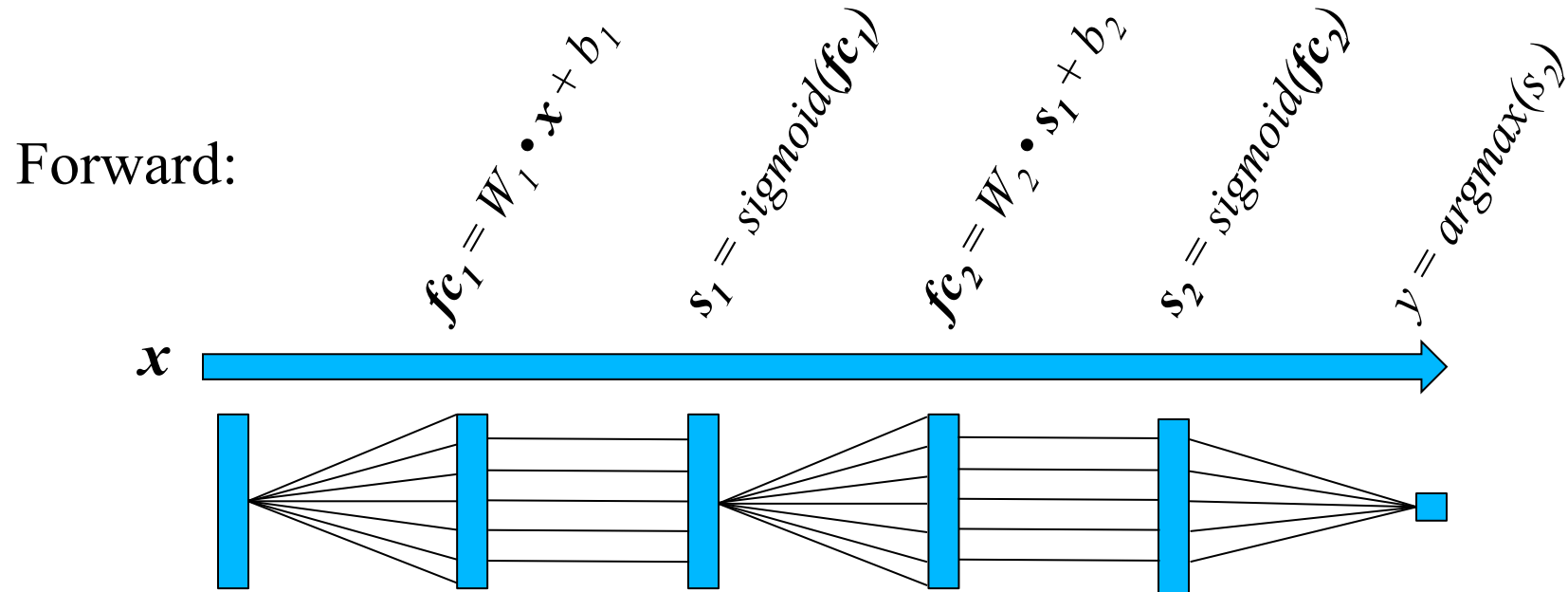
# How to determine the weights?

- Look at observational data to determine the weights?
- Pick some random values?
- Start with something that partially works?
- With enough *labeled* data, we can automatically *encode* the relationship between inputs and outputs.

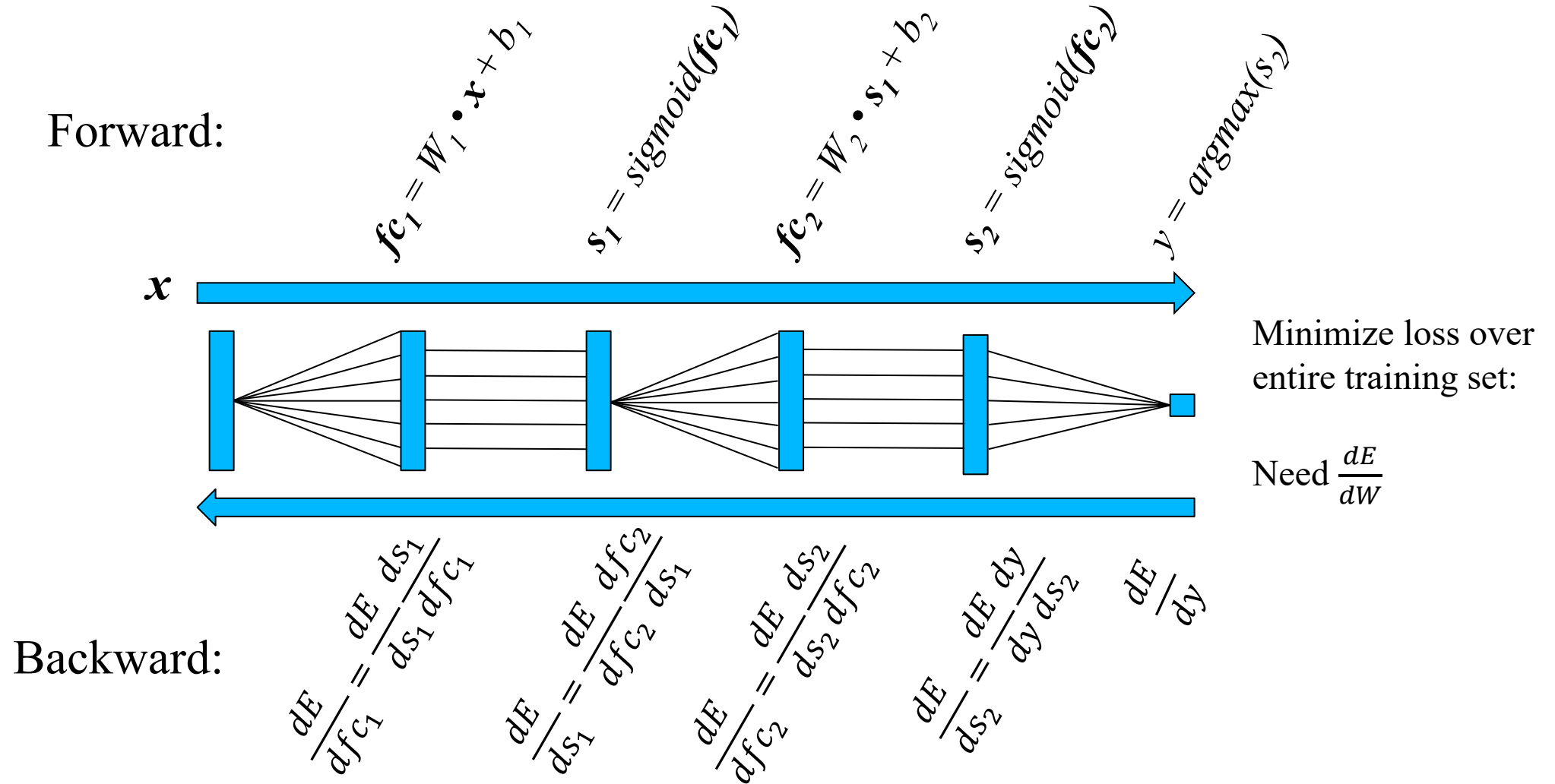
# Forward and Backward Propagation

- Forward (inference)
  - Given parameters  $\Theta$  and input  $\mathbf{x}$ , produce label  $y$
- Backward (training)
  - Need a way to assess correctness (loss function)
  - Example:  $(x - y)^2$
  - Find  $\Theta$ , such that loss is minimized over all input data

# Forward Propagation (Inference)



# Backward Propagation (Training)



Two vertical lines, one blue and one orange, are positioned on the left side of the slide.

# **ANY MORE QUESTIONS?**