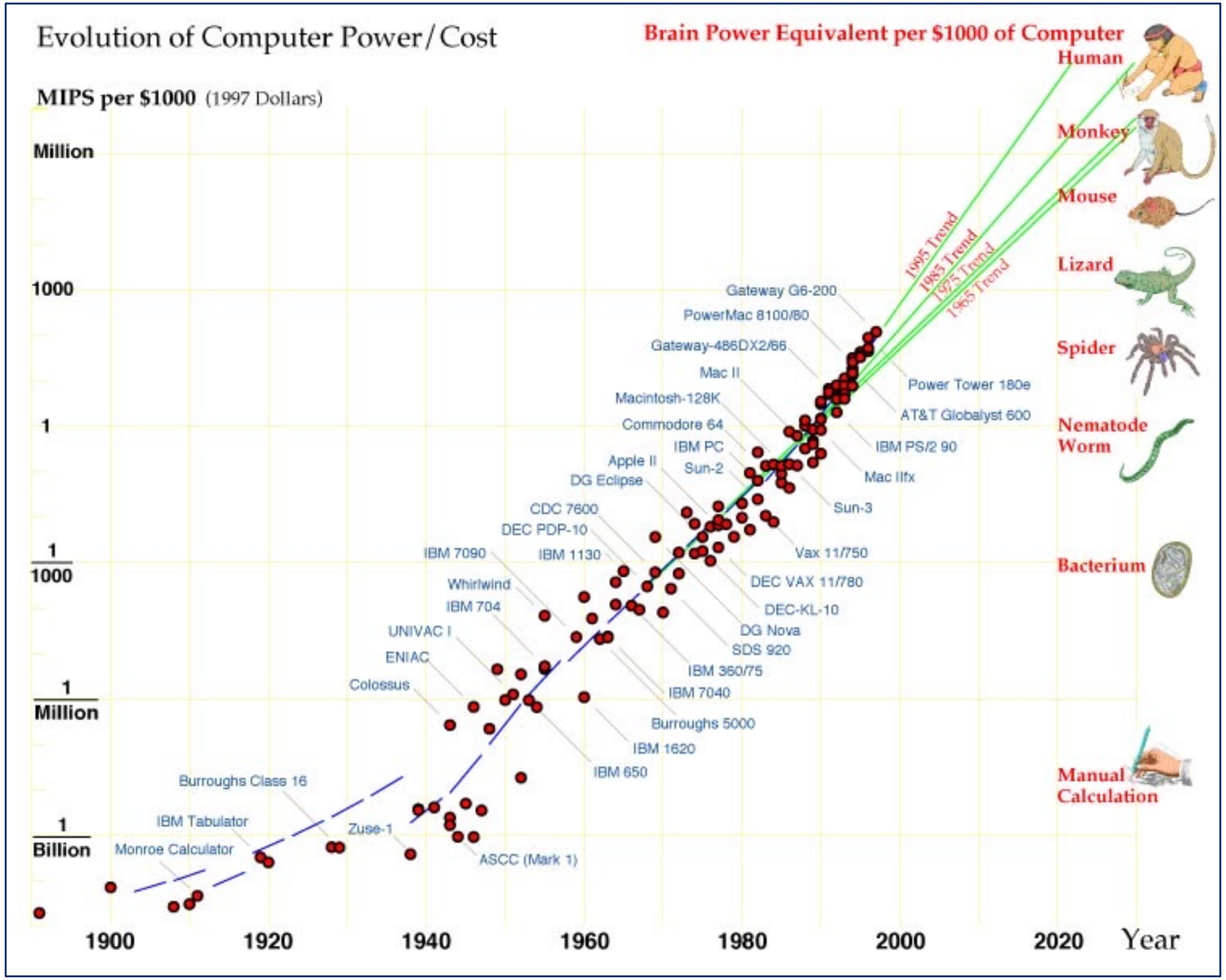
ECE408 / CS483 / CSE 408 Fall 2022

Introduction to Machine Learning



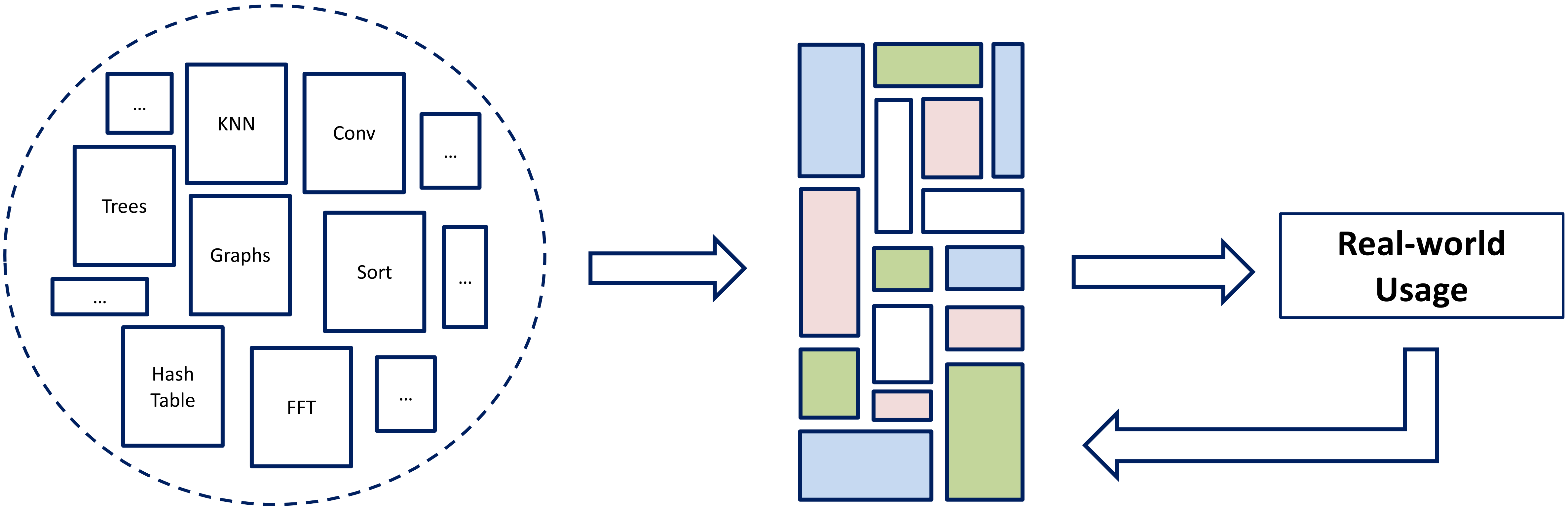
Hans Moravec, 1997

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.

Solving Hard Problems with Software (the established way)

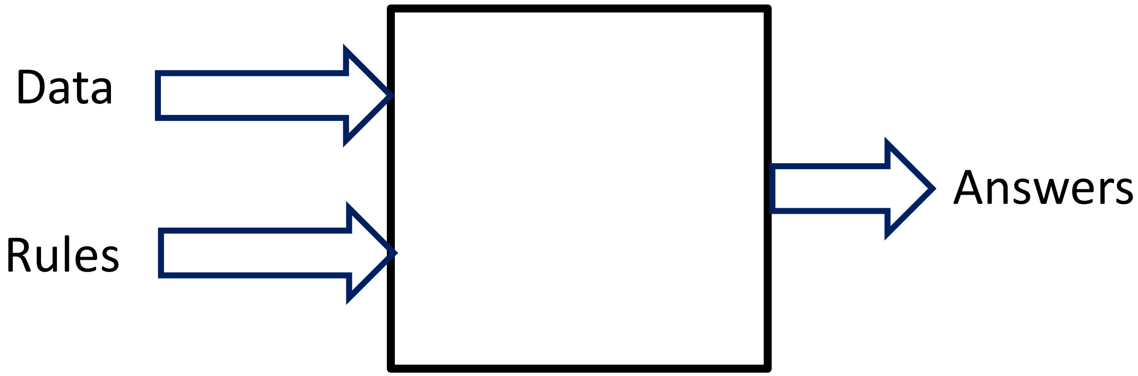


Algorithmic Building Blocks

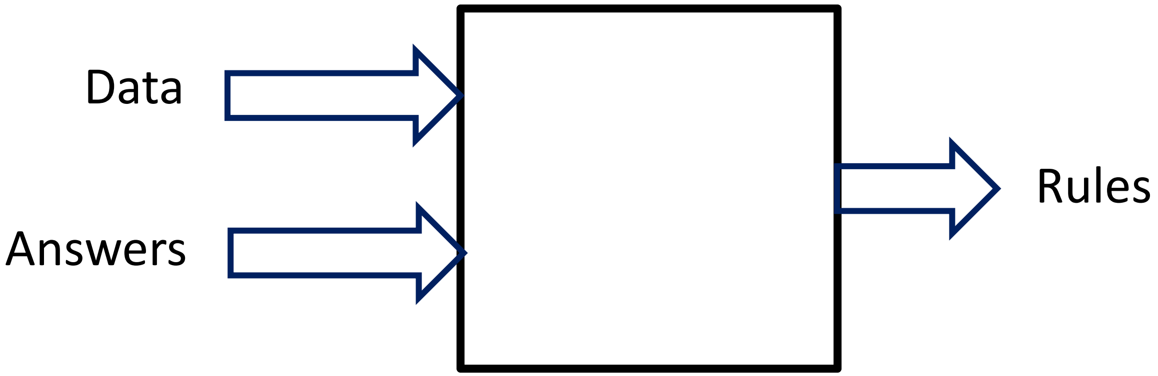
Software

Refinement

Machine Learning Restates the Problem

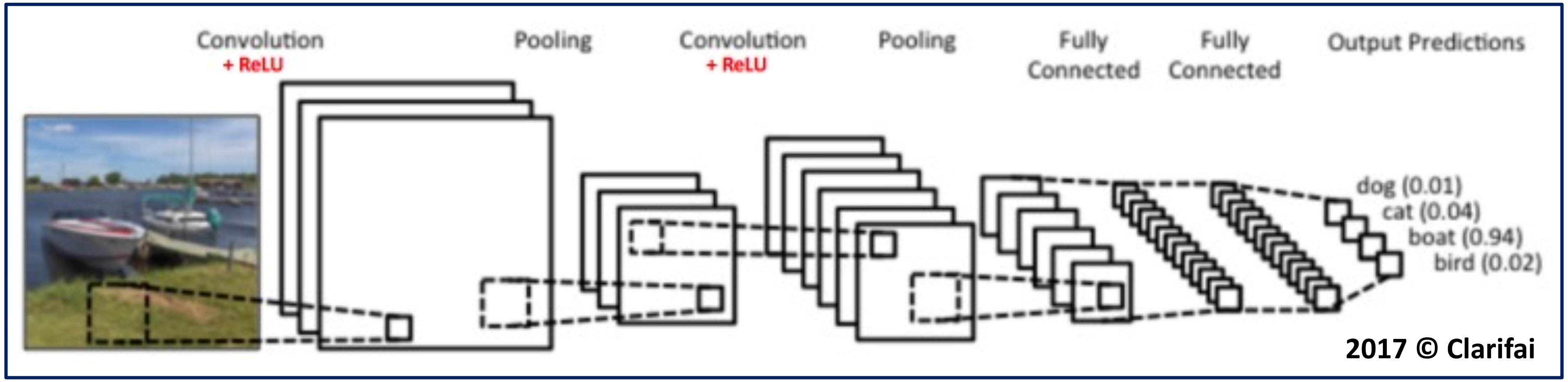


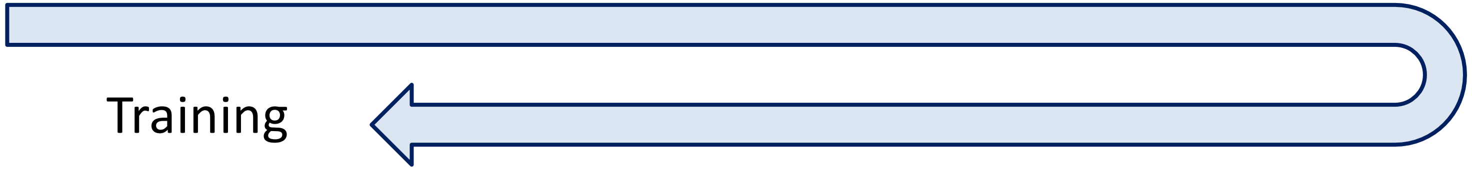
The Established Paradigm



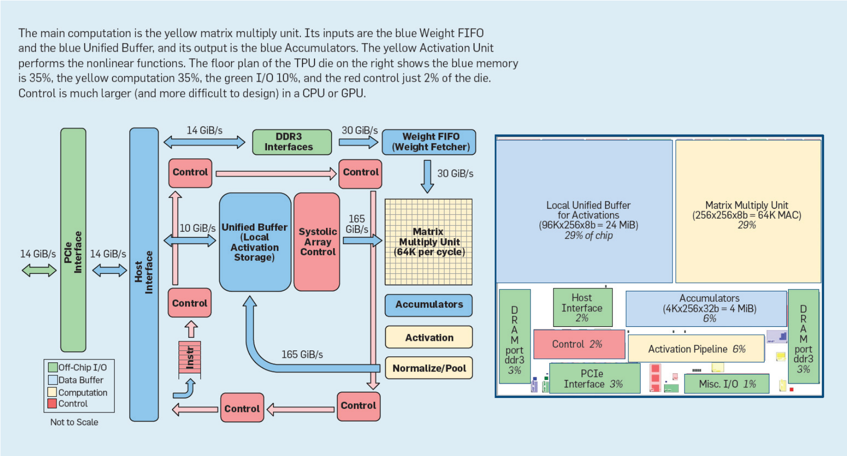
The ML Paradigm

Software that Learns:
Deep Neural Networks and the re-birth of AI

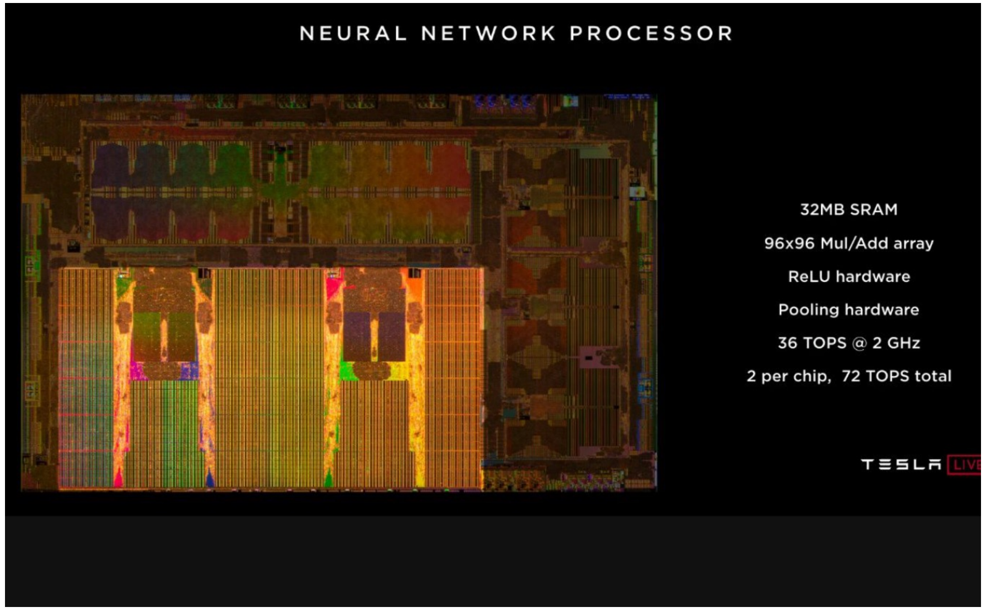




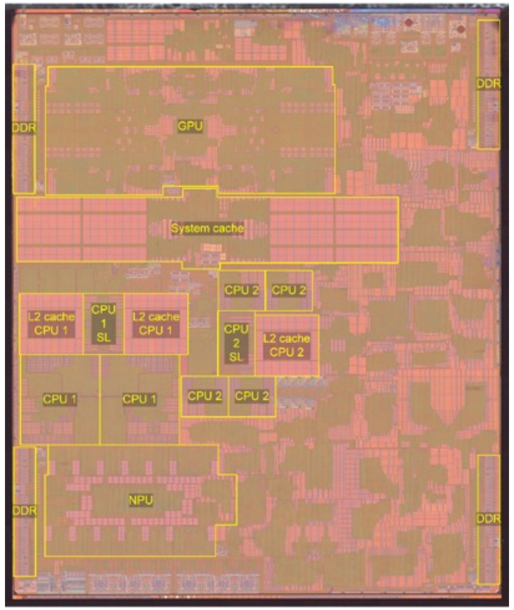
It Learns! Just Add Labeled Data. Lots of Data



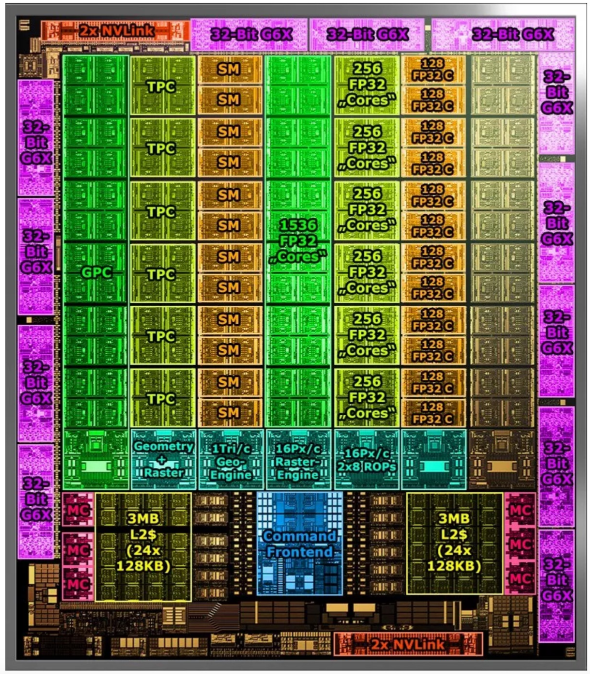
Google Tensor Processing Unit



Tesla ASIC



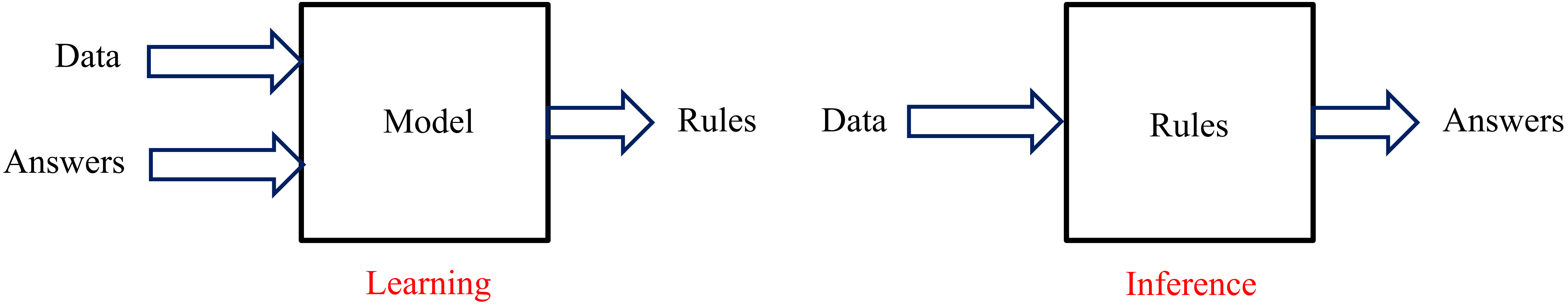
Apple A14



Nvidia Ampere

Machine Learning

- Building applications whose logic is not fully understood.
  - Use labeled data – data that come with the input values and their desired output values – to learn what the logic should be



Machine Learning Tasks (1)

- Classification
  - Which of *k* categories an input belongs to
  - Ex: object recognition
- Regression
  - Predict a numerical value given some input
  - Ex: predict tomorrow's temperature
- Transcription
  - Unstructured data into textual form
  - Ex: optical character recognition

Machine Learning Tasks (2)

- 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 a tree of grammatical structure
- Others
  - Forecasting, Anomaly detection, recommendation, synthesis, sampling, imputation, denoising, density estimation

Why Machine Learning Now?

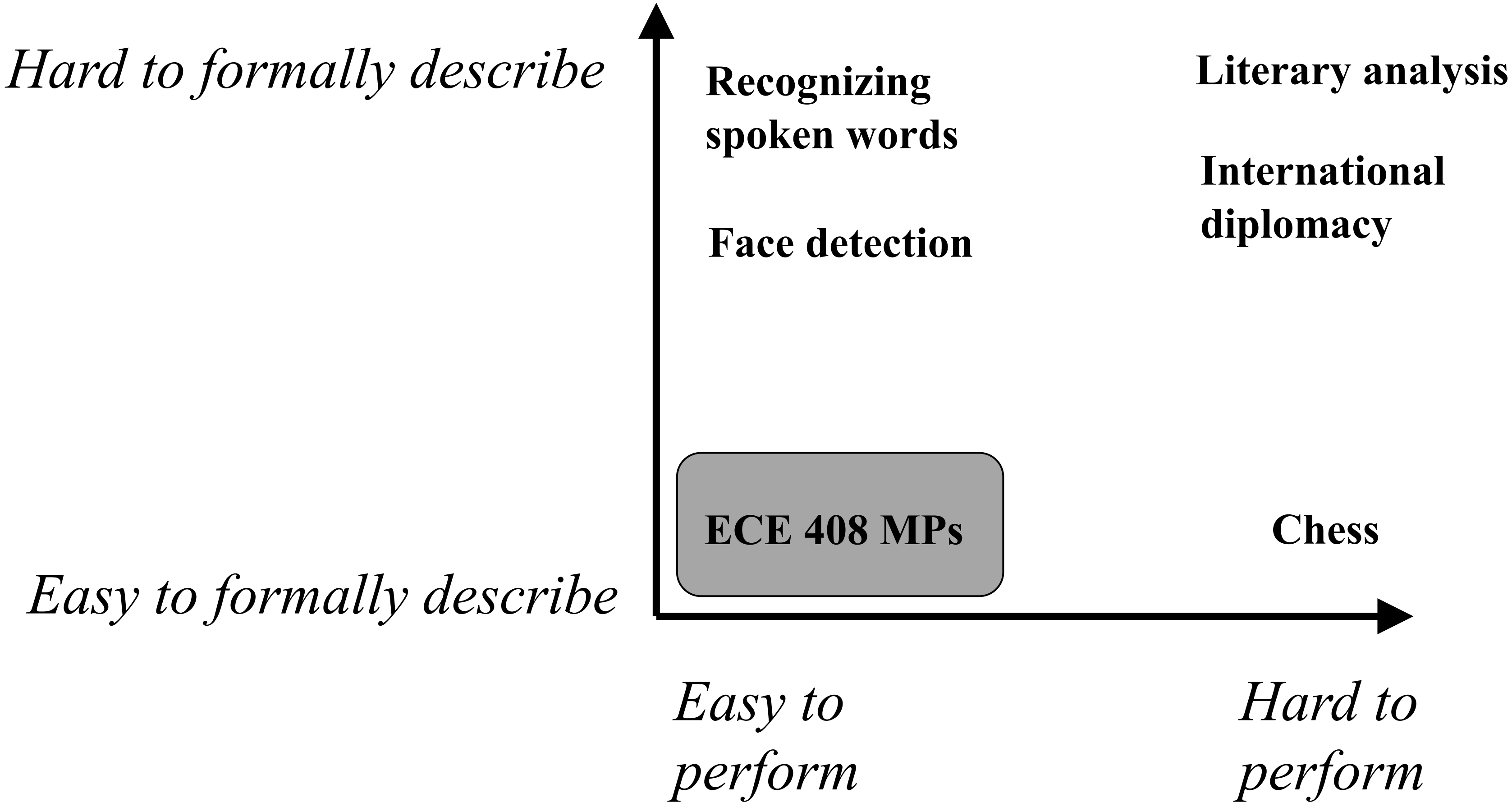
- **Deep Learning:** Advances in DL concepts, frameworks, toolkits have made DL very accessible to all

- **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..

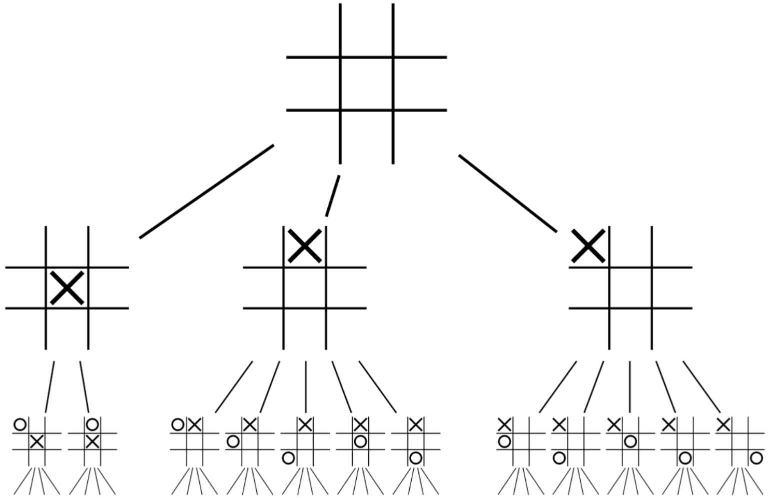
- **Needs:** Autonomous Vehicles, SmartDevices, Security, Societal Comfort with Tech, Health Care, Digital Agriculture, Digital Manufacturing

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



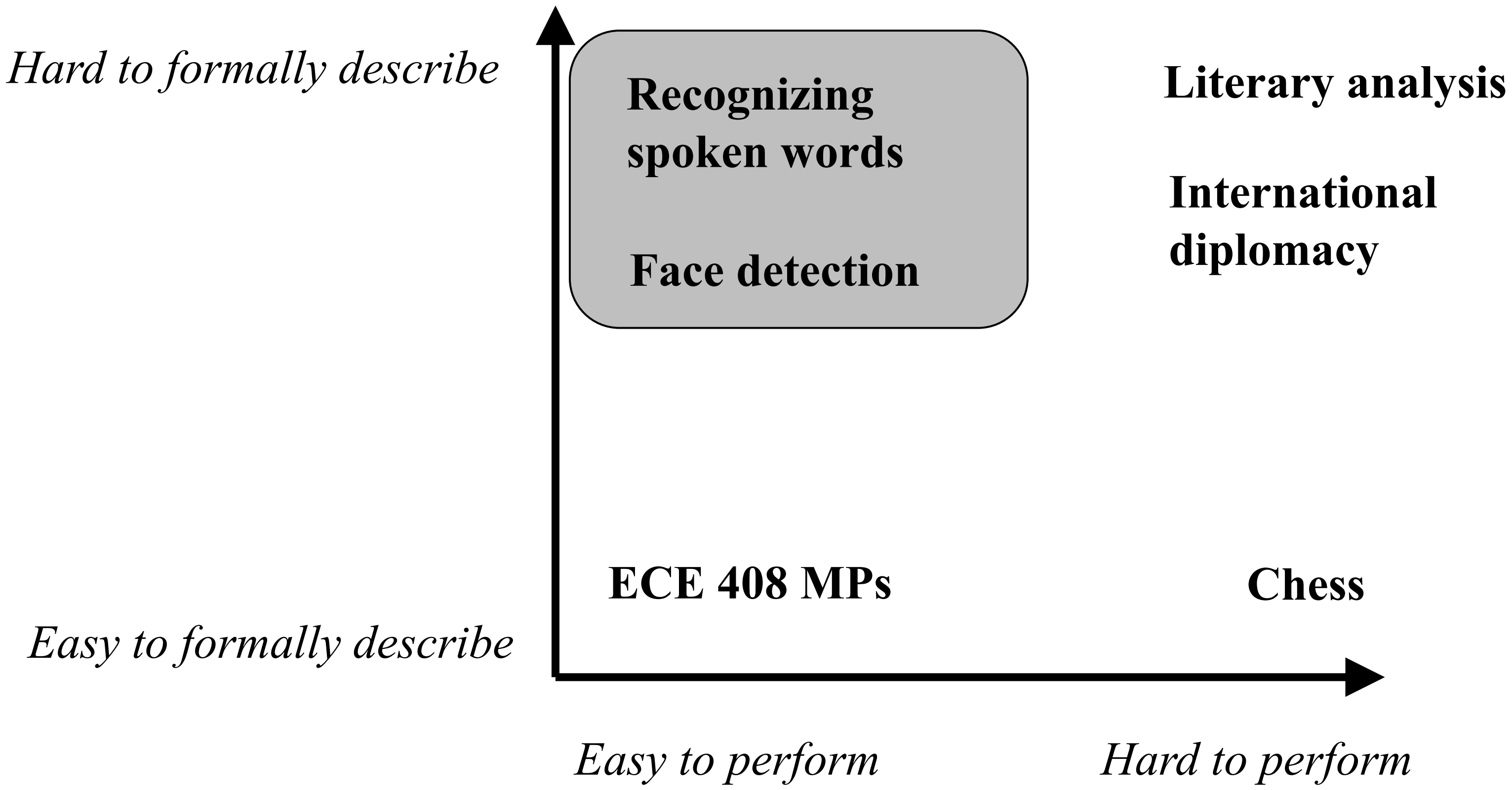
2-ply game tree for tic-tac-toe

Chess as an AI Success (2)

IBM 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

Types of Problems



The “Machine Learning” Approach

Challenge

Hard to formalize the problem.

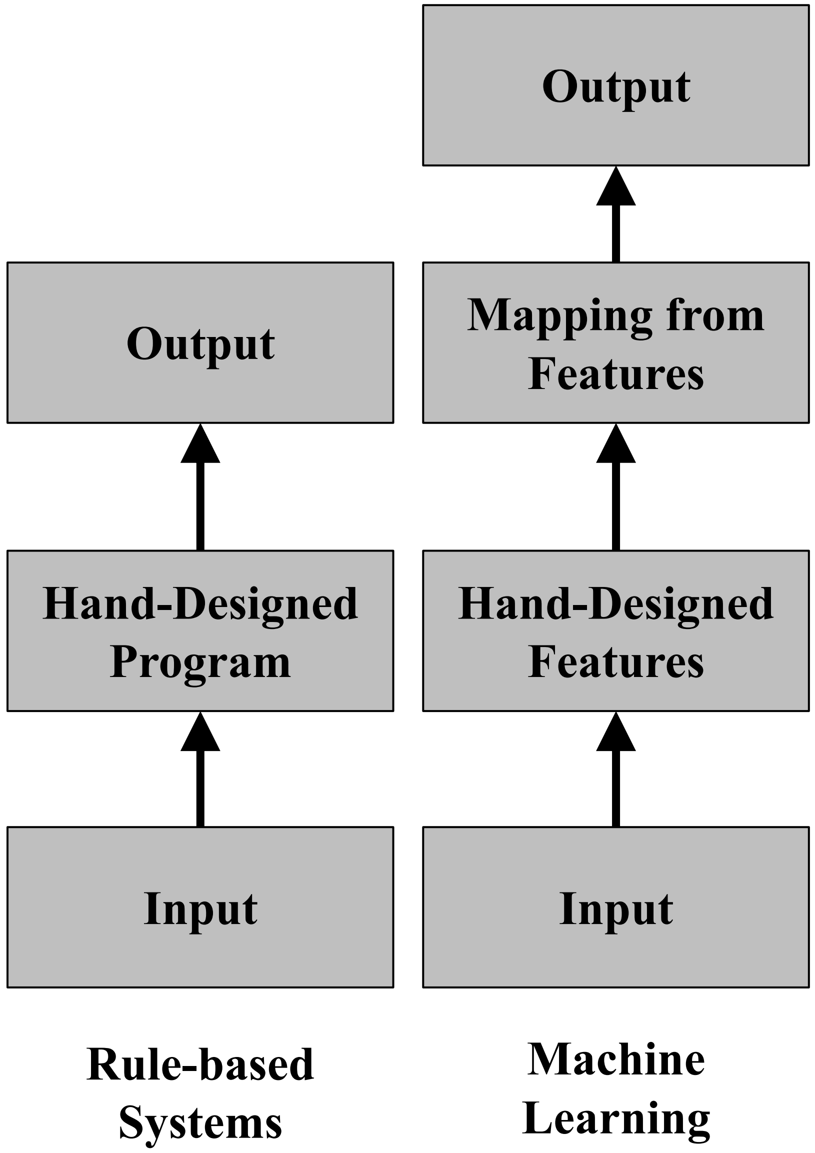
Solution

Don’t formalize the problem.

Let the machine learn from data/experience.

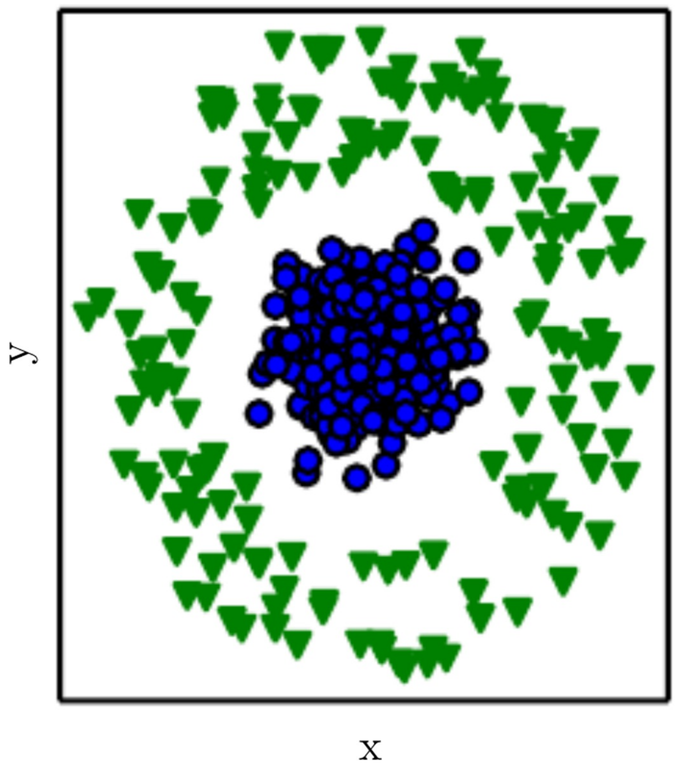
Classic Machine Learning

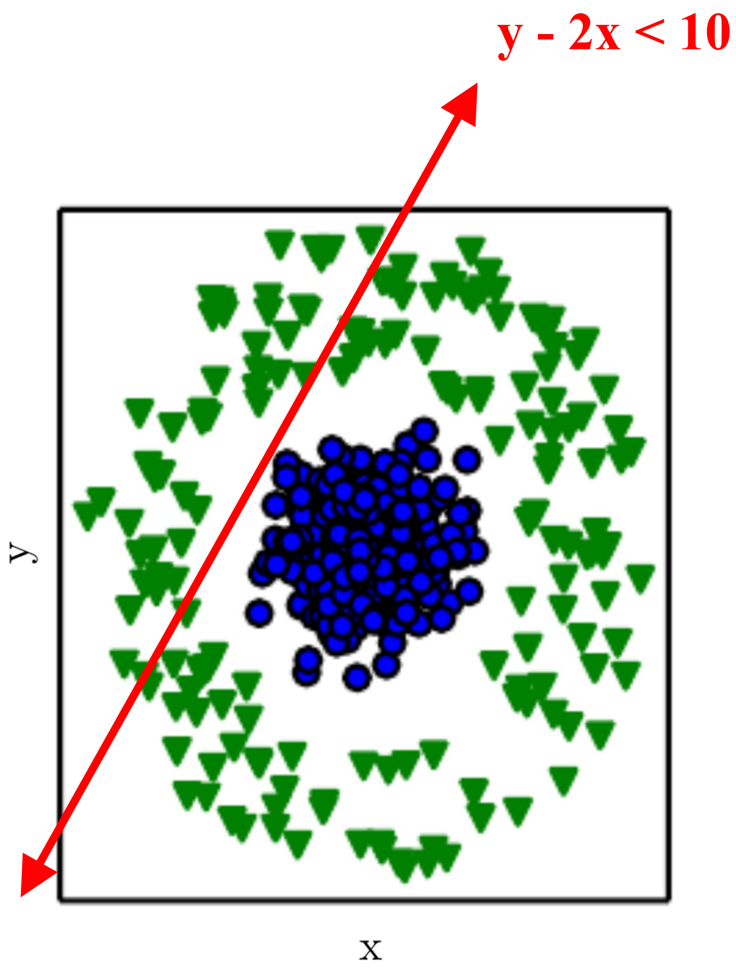
- Humans choose features
- Learn how features are associated with outputs



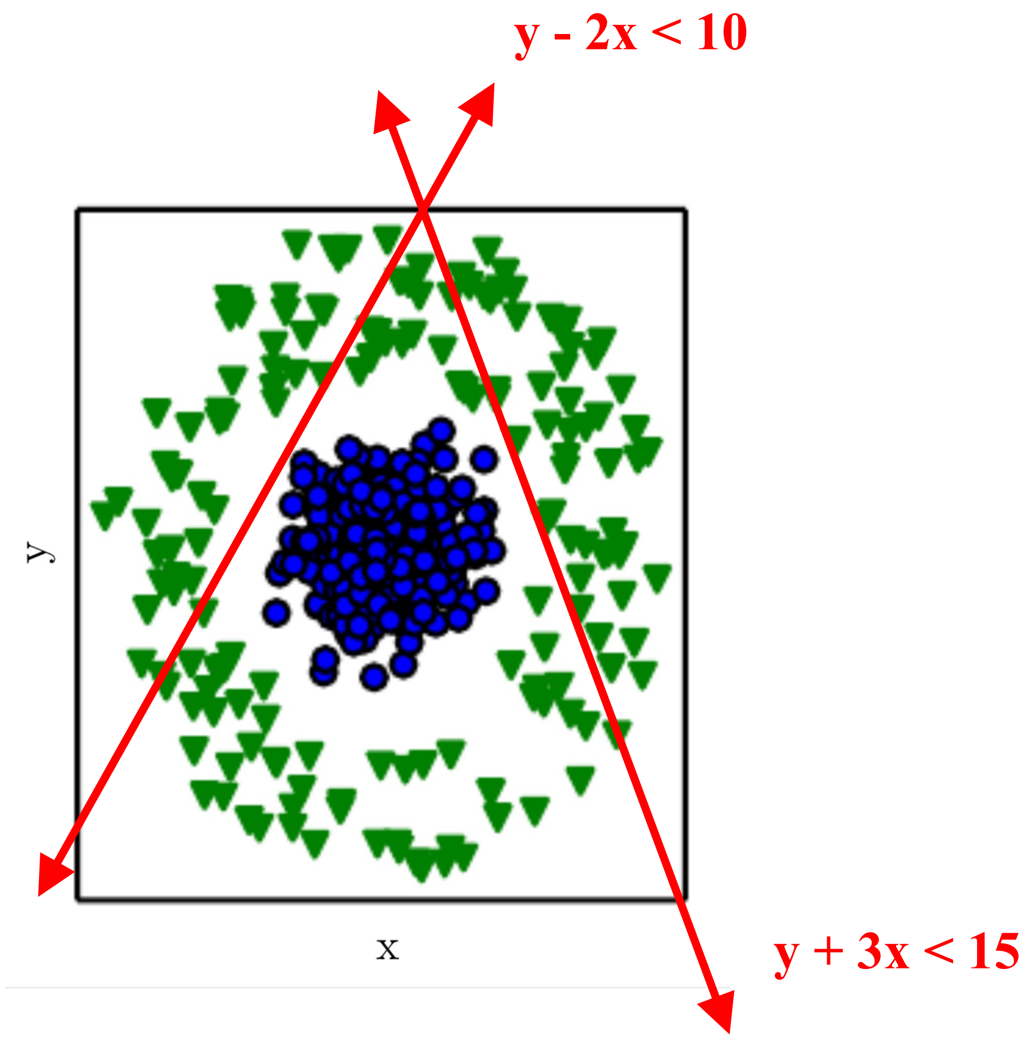
Machine Learning Building Blocks

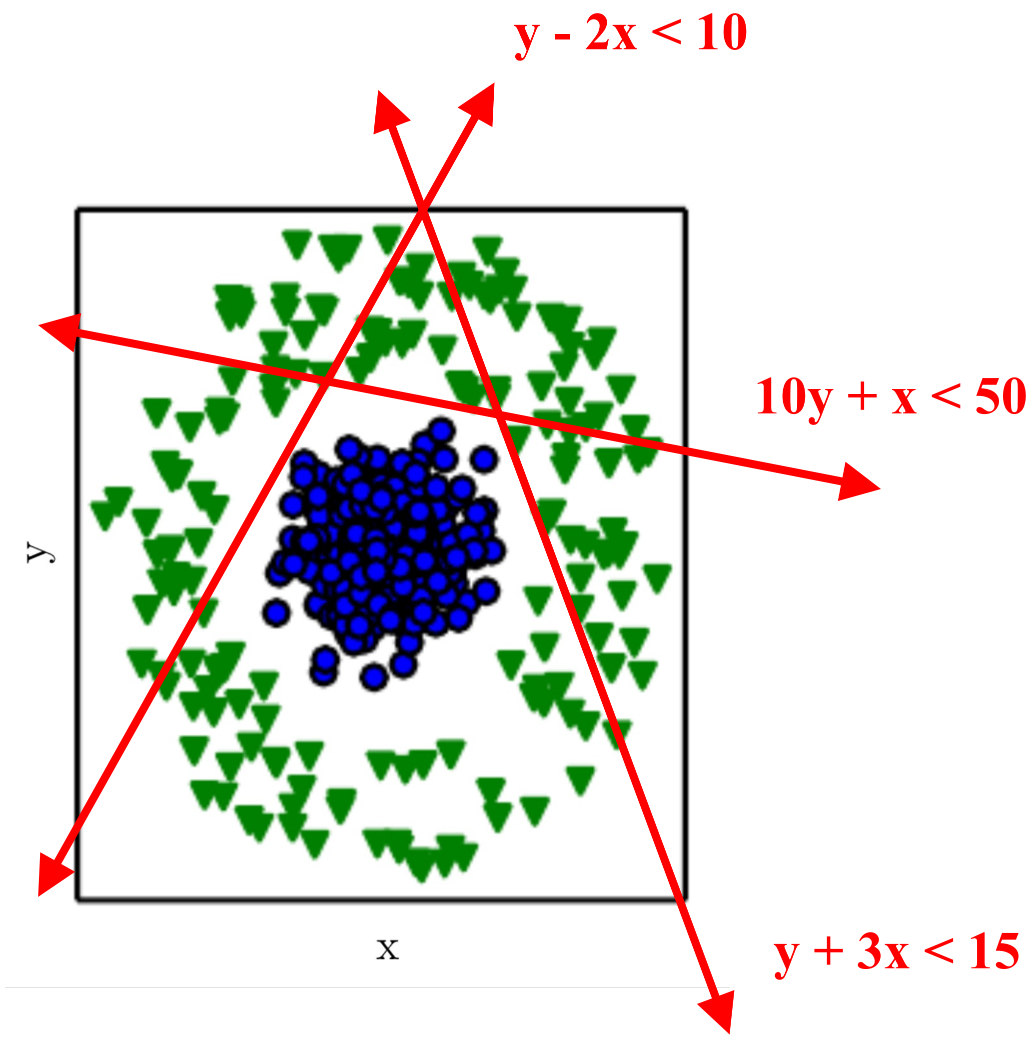
- **Naïve Bayes**
  - Independent features combined using Bayesian prediction model
- **Perceptrons**
  - "Bio/Neural" inspired method
- **Linear / Logistic Regression**
  - Feature contribution learned to perform prediction / classification
- **Support Vector Machines**
  - Large margin method
- **Decision Trees / Random Forests**
  - Space-splitting methods
- **K-Means Clustering**
  - Unsupervised technique for data analysis

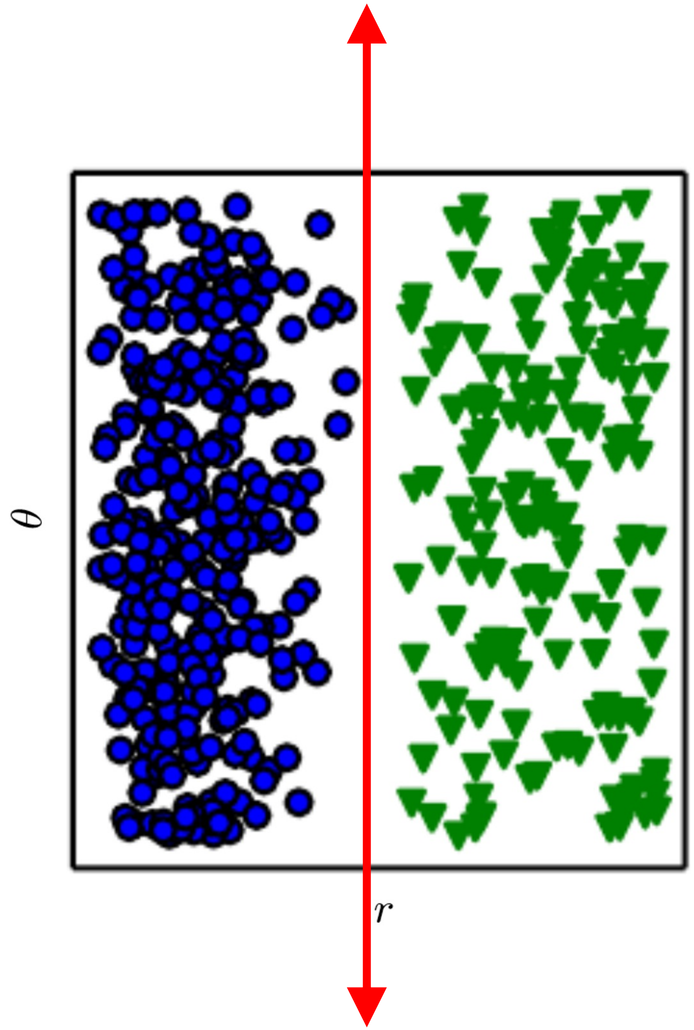

y
x



y - 2x < 10
y
x


y - 2x < 10
y
x
y + 3x < 15


y - 2x < 10
10y + x < 50
y + 3x < 15
y
x



$\Theta = \arctan(y / x)$
$r = \mathrm{sqrt}(x^2 + y^2)$

Data Representation is Important!

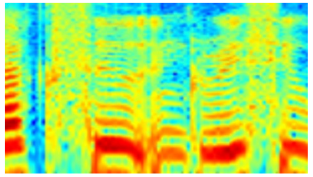
Different Features for Different Tasks

Image

Vision Features

Detection

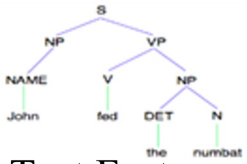
Audio



Audio Features

Identify
Speaker

Text



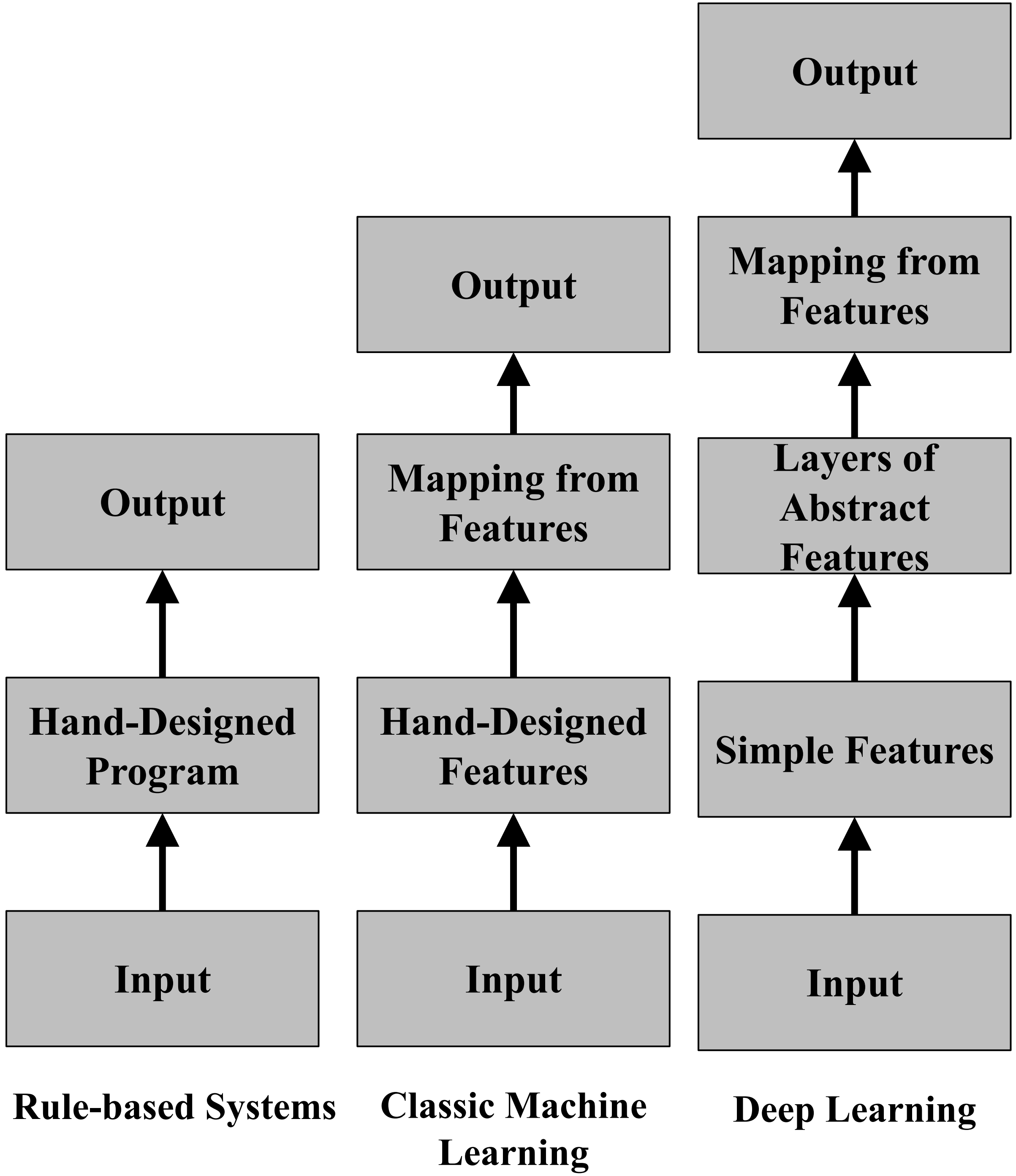
Text Features

Text classification, machine translation, information retrieval

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?
  - Black/dark rubber around metal rim?
- But what about?
  - Occlusion, perspective, shadows, different colored tires, …
- Need to treat variations in a consistent and comprehensive manner

Evolution of AI



Classification

- Formally: a function that maps an input to *k* categories

  $$f: \mathrm{R}^n \rightarrow \{1, \ldots, k\},$$

- Our formulation: a function f parameterized by Θ that maps input vector $\boldsymbol{x}$ to numeric code y

  $$y = f(\boldsymbol{x}, \Theta)$$

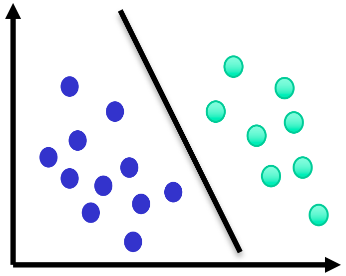
- Θ encapsulates the parameters in our network

Linear Classifier (Perceptron)

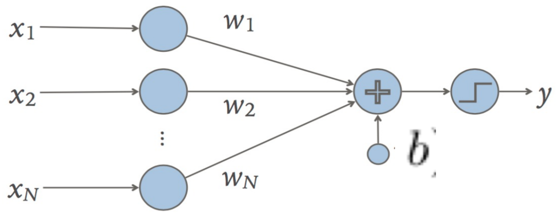
- Our formulation: $y = f(\boldsymbol{x}, \Theta)$

$$\Theta = \{W, b\}$$

$$y = sign(W \bullet \boldsymbol{x} + b)$$

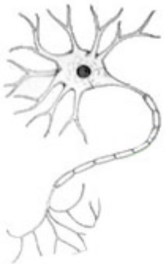


The perceptron



$\approx$

The neuron



- Dot product + Scalar addition:

output

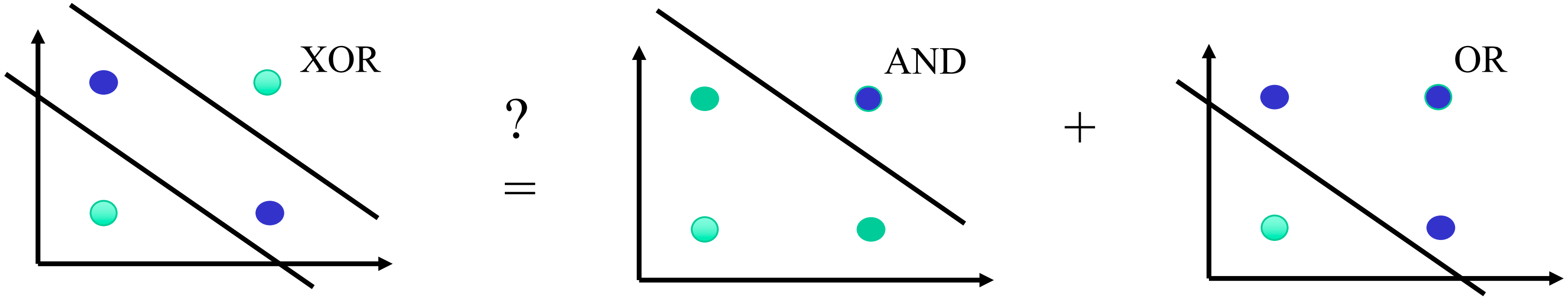
$$y = W \bullet \boldsymbol{x} + b$$

bias

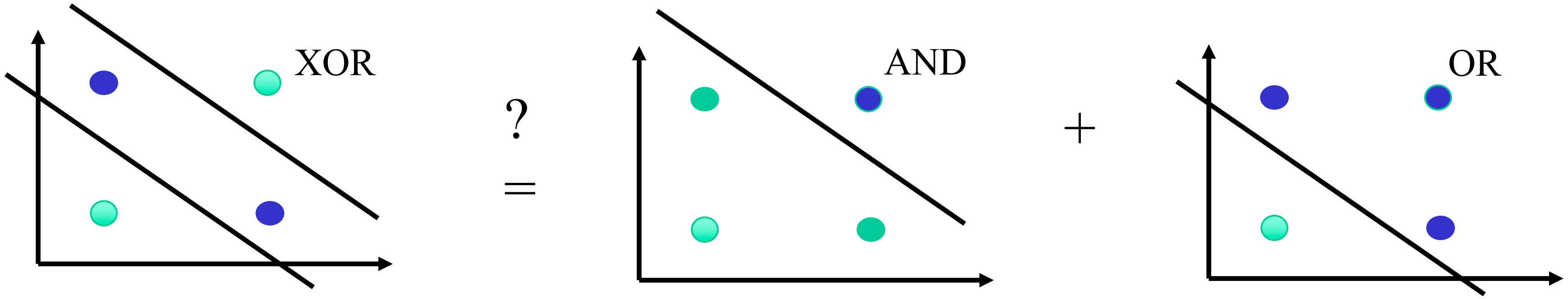
weight

input

Can we learn XOR with a Perceptron?



Perceptron



$x[0] + x[1] - 1.5 > 0$ $\qquad$ $x[0] + x[1] - 0.5 > 0$

x[1]	x[0]	AND	OR	XOR
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)

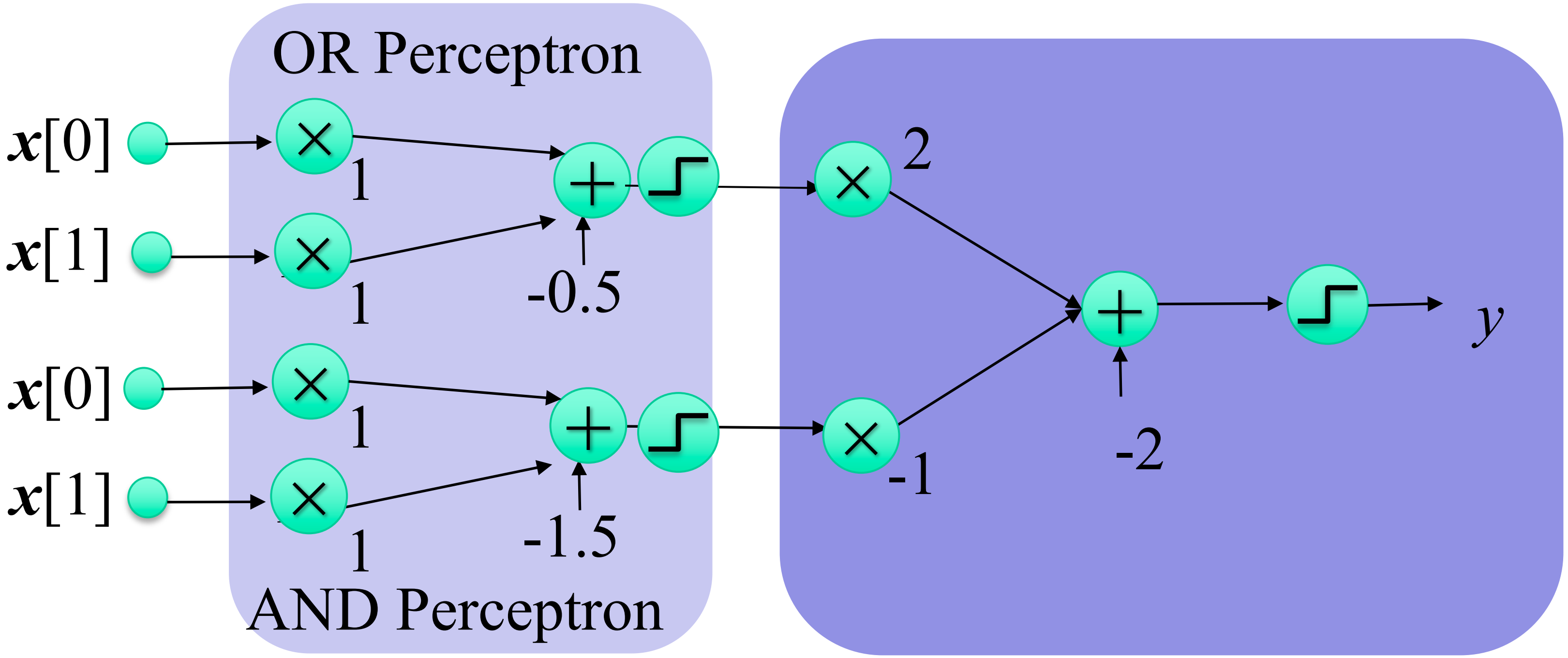
XOR is not a linear combination of AND and OR functions.

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)

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

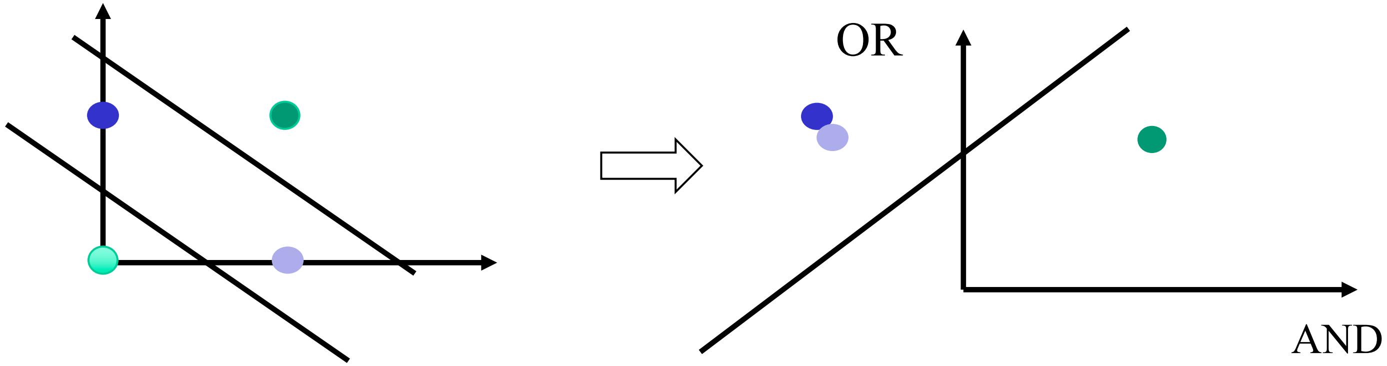
$$\text{AND} = \text{sign}(\boldsymbol{x}[0] + \boldsymbol{x}[1] - 1.5)$$

sign() function adds non-linearity to "reposition" data points for the next layer.



XOR = sign(2*OR + -1*AND - 2)

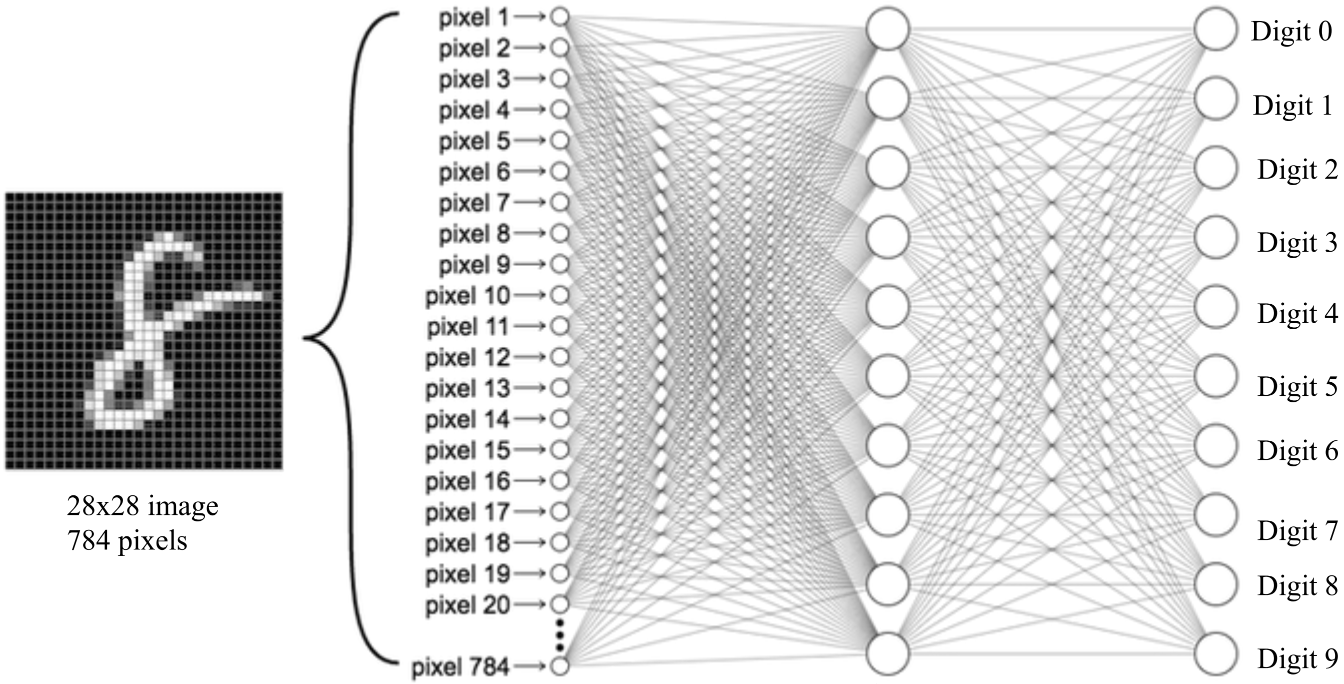
Multi-Layer Perceptron



XOR = sign(2*OR + -1*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)

MultiLayer Perceptron (MLP) for Digit Recognition



$$n_k^1 = \text{activation}(\boldsymbol{w}_k \boldsymbol{x} + \boldsymbol{b}_k)$$

This network would has

- 784 nodes on input layer (L0)
- 10 nodes on hidden layer (L1)
- 10 nodes on output layer (L2)

784*10 weights + 10 biases for L1
10*10 weights + 10 biases for L2

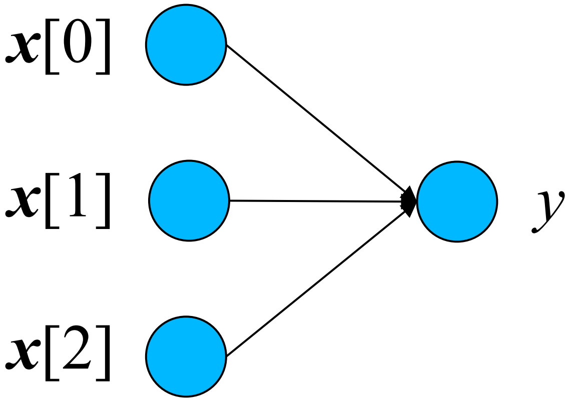
A total of 7,960 parameters

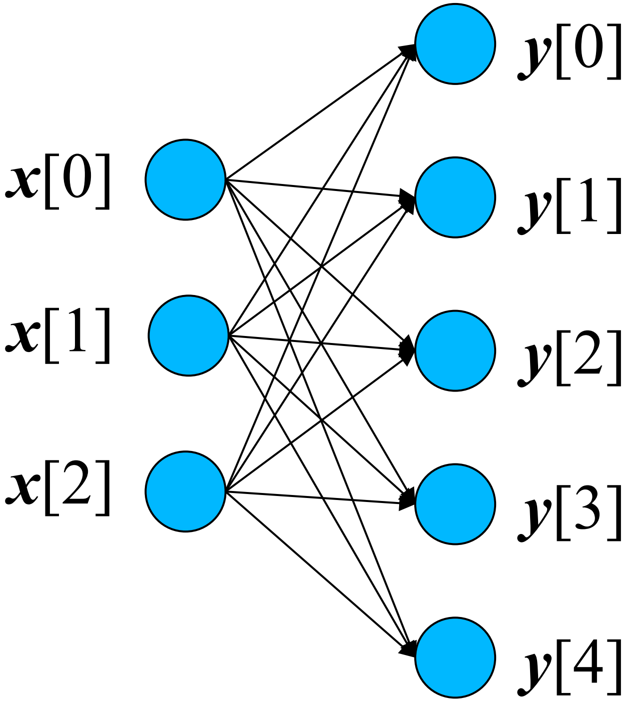
Each node represents a function, based on a linear combination of inputs + bias

Activation function “repositions” output value.

Sigmoid, sign, ReLU are common…

Generalize to Fully-Connected Layer

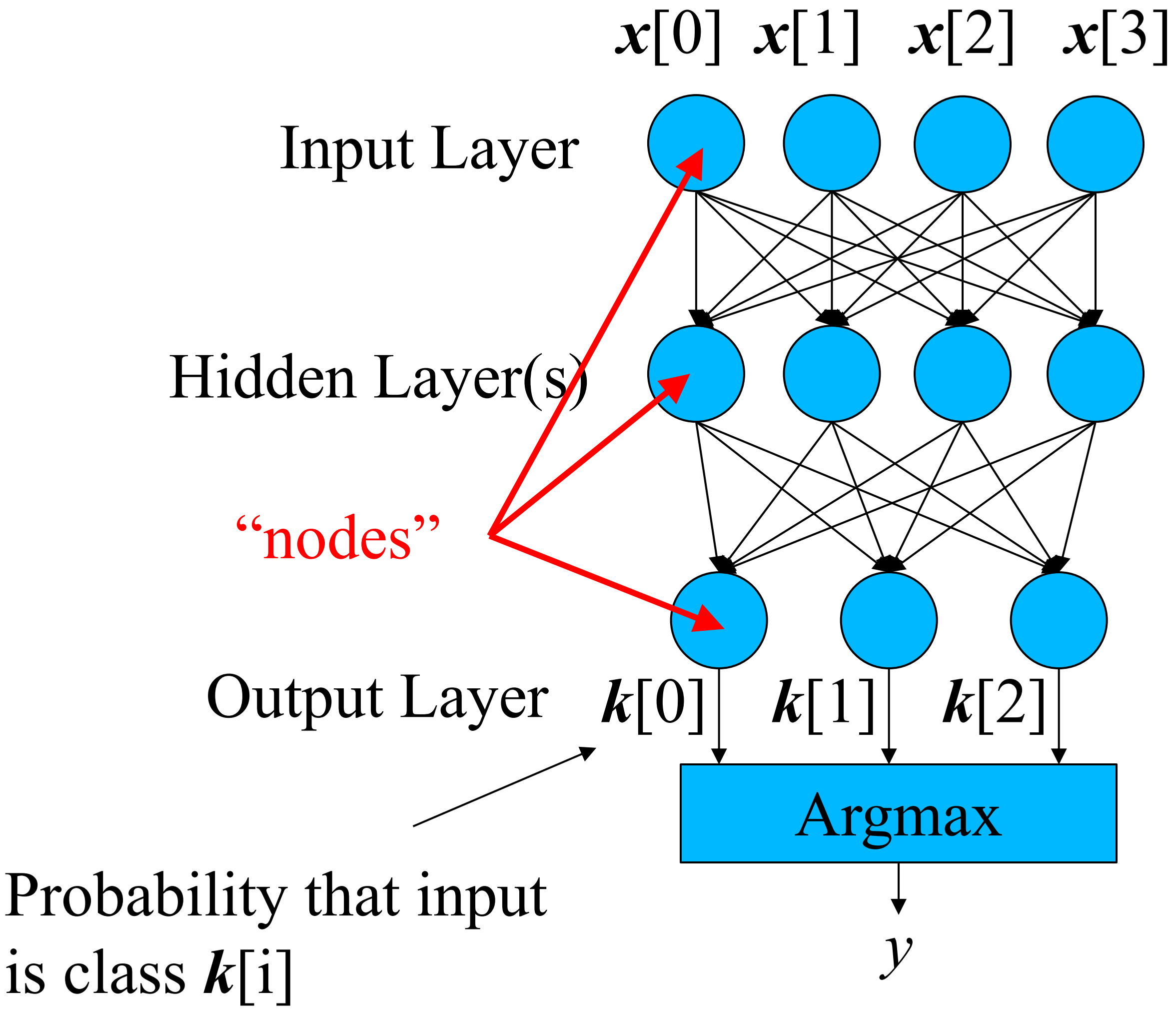




Linear Classifier:
Input vector $\boldsymbol{x} \times$ weight vector $\boldsymbol{w}$ to produce scalar output y

Fully-connected:
Input vector $\boldsymbol{x} \times$ weight <u>matrix</u> $\boldsymbol{w}$ to produce <u>vector</u> output y

Multilayer Terminology



W_1 is [4x4], b_1 is [4x1]

W_2 is [4x3], b_2 is [3x1]

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

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

Forward Propagation (Inference)

Forward:

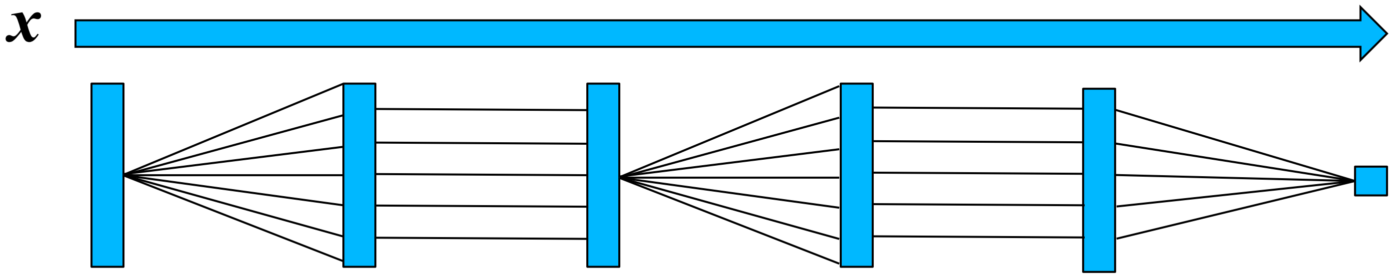
$$\boldsymbol{fc}_1 = W_1 \bullet \boldsymbol{x} + b_1$$

$$\boldsymbol{s}_1 = sigmoid(\boldsymbol{fc}_1)$$

$$\boldsymbol{fc}_2 = W_2 \bullet \boldsymbol{s}_1 + b_2$$

$$\boldsymbol{s}_2 = sigmoid(\boldsymbol{fc}_2)$$

$$y = argmax(s_2)$$

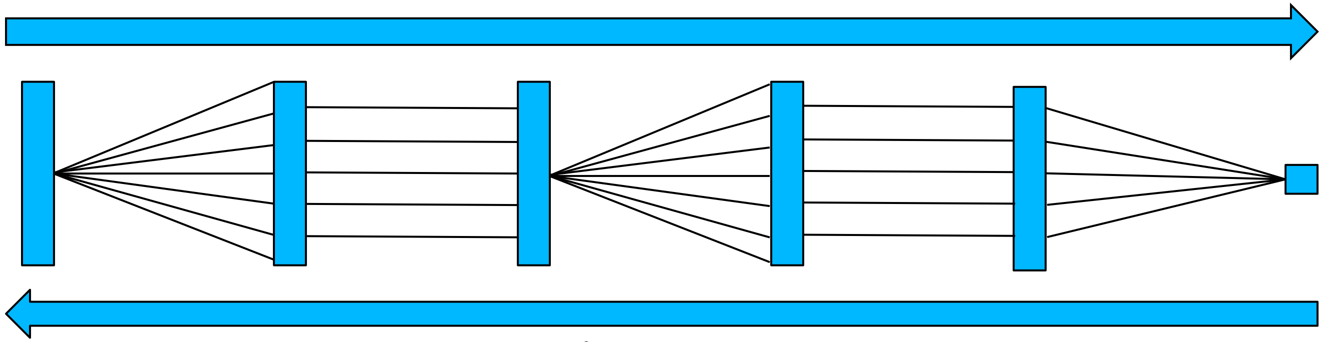


Backward Propagation (Training)

Forward:

$\boldsymbol{fc}_1 = W_1 \bullet \boldsymbol{x} + b_1$ $\quad$ $\boldsymbol{s}_1 = sigmoid(\boldsymbol{fc}_1)$ $\quad$ $\boldsymbol{fc}_2 = W_2 \bullet \boldsymbol{s}_1 + b_2$ $\quad$ $\boldsymbol{s}_2 = sigmoid(\boldsymbol{fc}_2)$ $\quad$ $y = argmax(s_2)$

$\boldsymbol{x}$



Minimize loss over entire training set:

Need $\frac{dE}{dW}$

Backward:

$\frac{dE}{dfc_1} = \frac{dE}{ds_1}\frac{ds_1}{dfc_1}$ $\quad$ $\frac{dE}{ds_1} = \frac{dE}{dfc_2}\frac{dfc_2}{ds_1}$ $\quad$ $\frac{dE}{dfc_2} = \frac{dE}{ds_2}\frac{ds_2}{dfc_2}$ $\quad$ $\frac{dE}{ds_2} = \frac{dE}{dy}\frac{dy}{ds_2}$ $\quad$ $\frac{dE}{dy}$