# Deep Learning

Dr. Sugata Ghosal
sugata.ghosal@pilani.bits-pilani.ac.in

**BITS** Pilani

Pilani Campus
Pilani Campus

# Agenda

- Introduction
- Course Objectives and Logistics
- Introduction to Perceptron and MLP
    - Approximation Capabilities
- Characteristics of Deep Learning

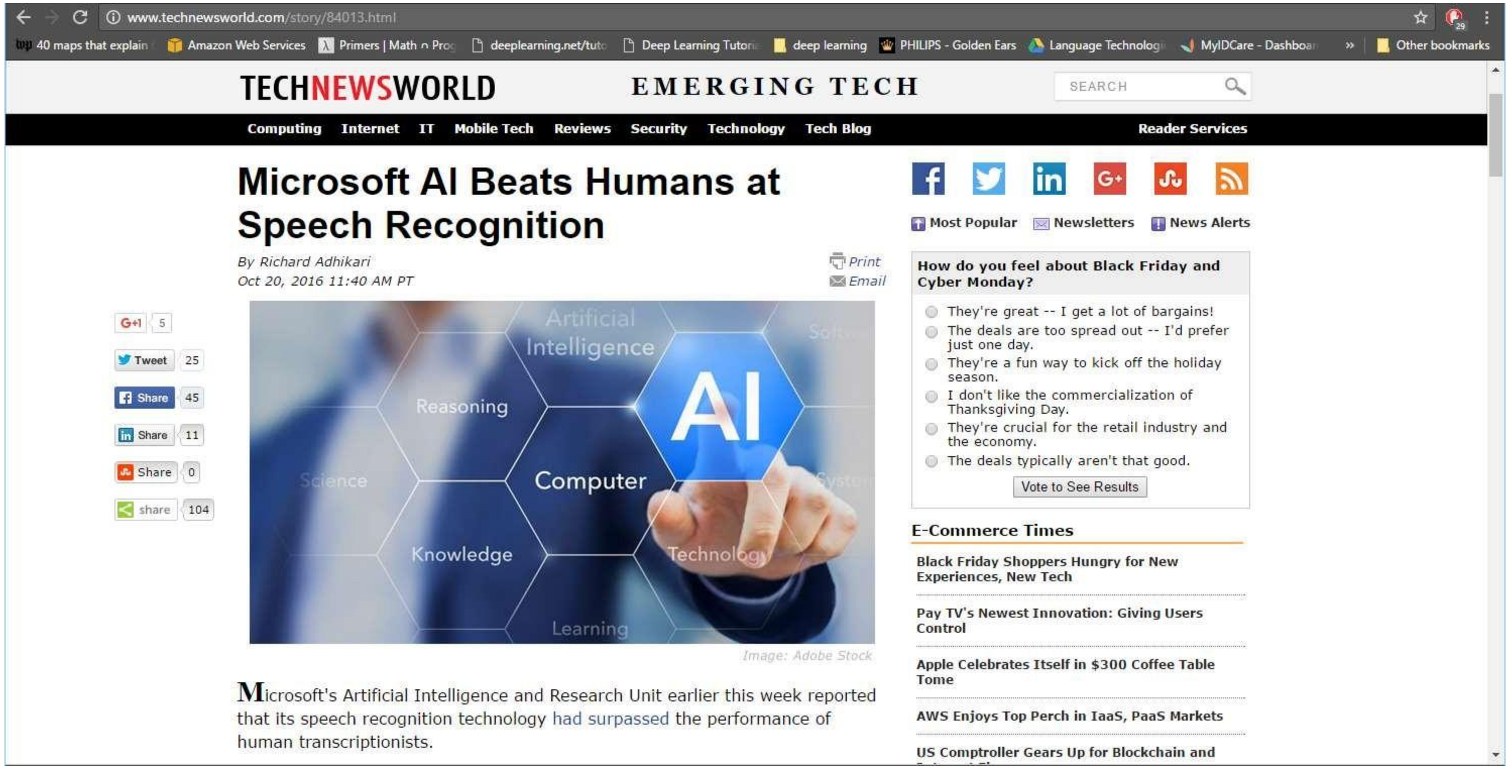
- Reading: Chapter 1 of Textbook

# Neural Networks are taking over!

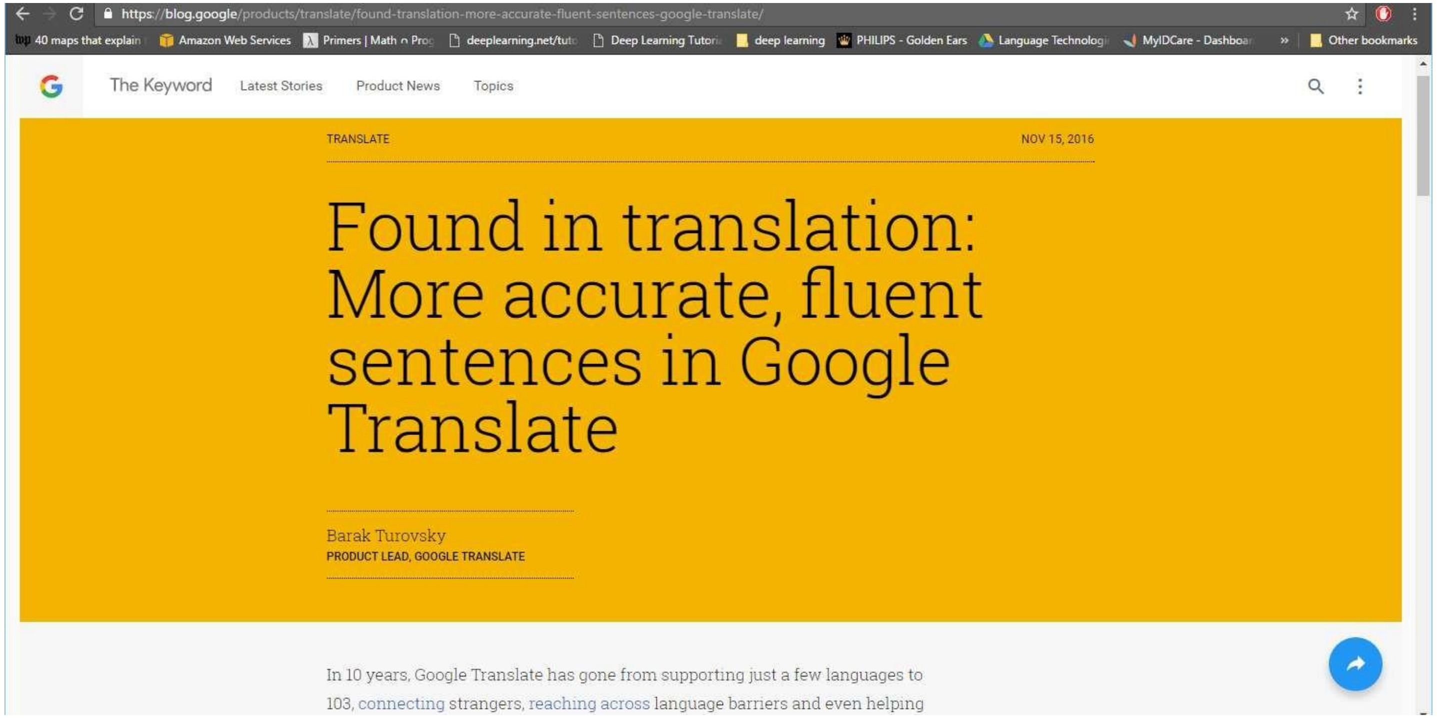
- Neural networks have become one of the major thrust areas recently in various pattern recognition, prediction, and analysis problems

- In many problems they have established the state of the art
  - Often exceeding previous benchmarks by large margins

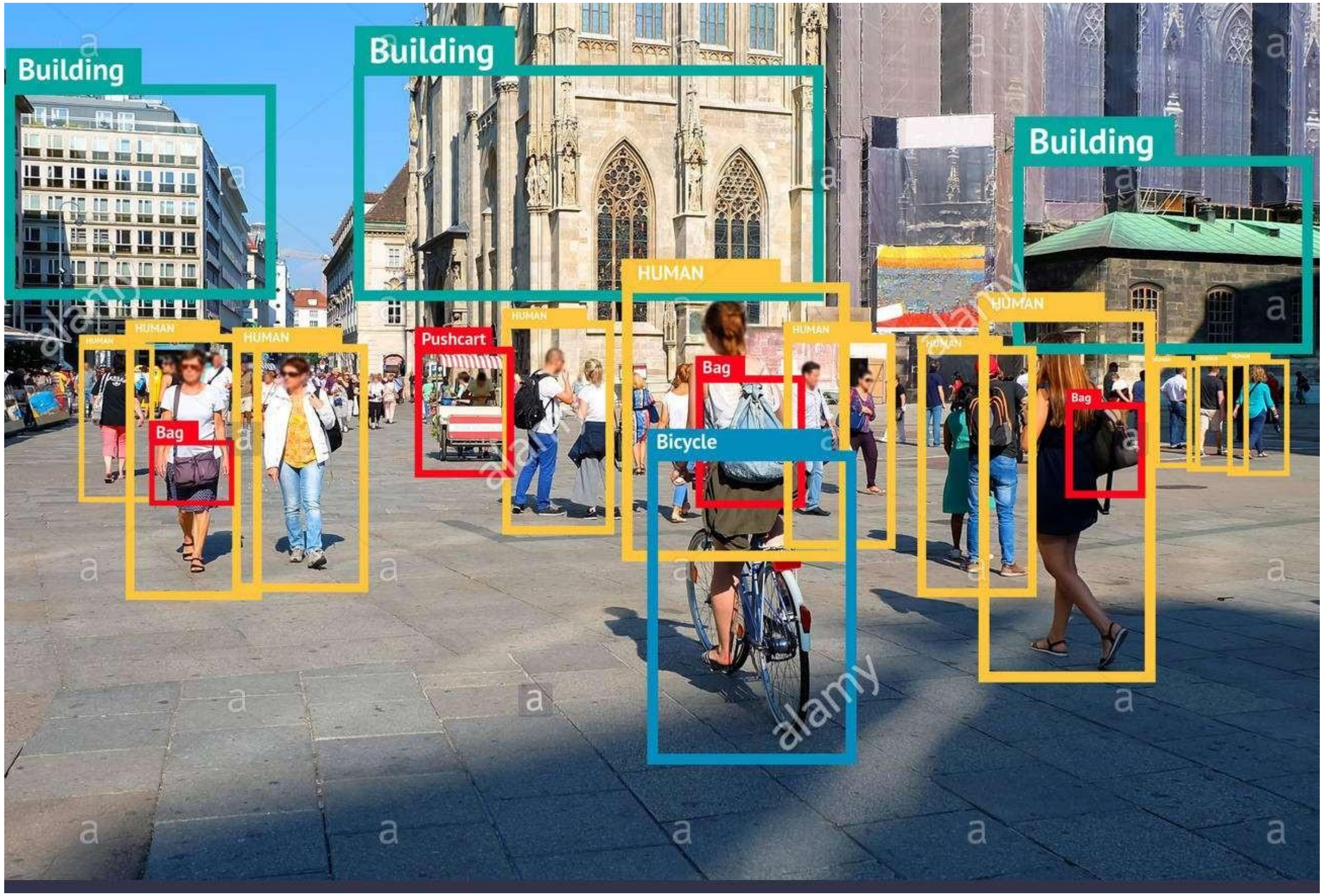
# Breakthroughs with neural networks



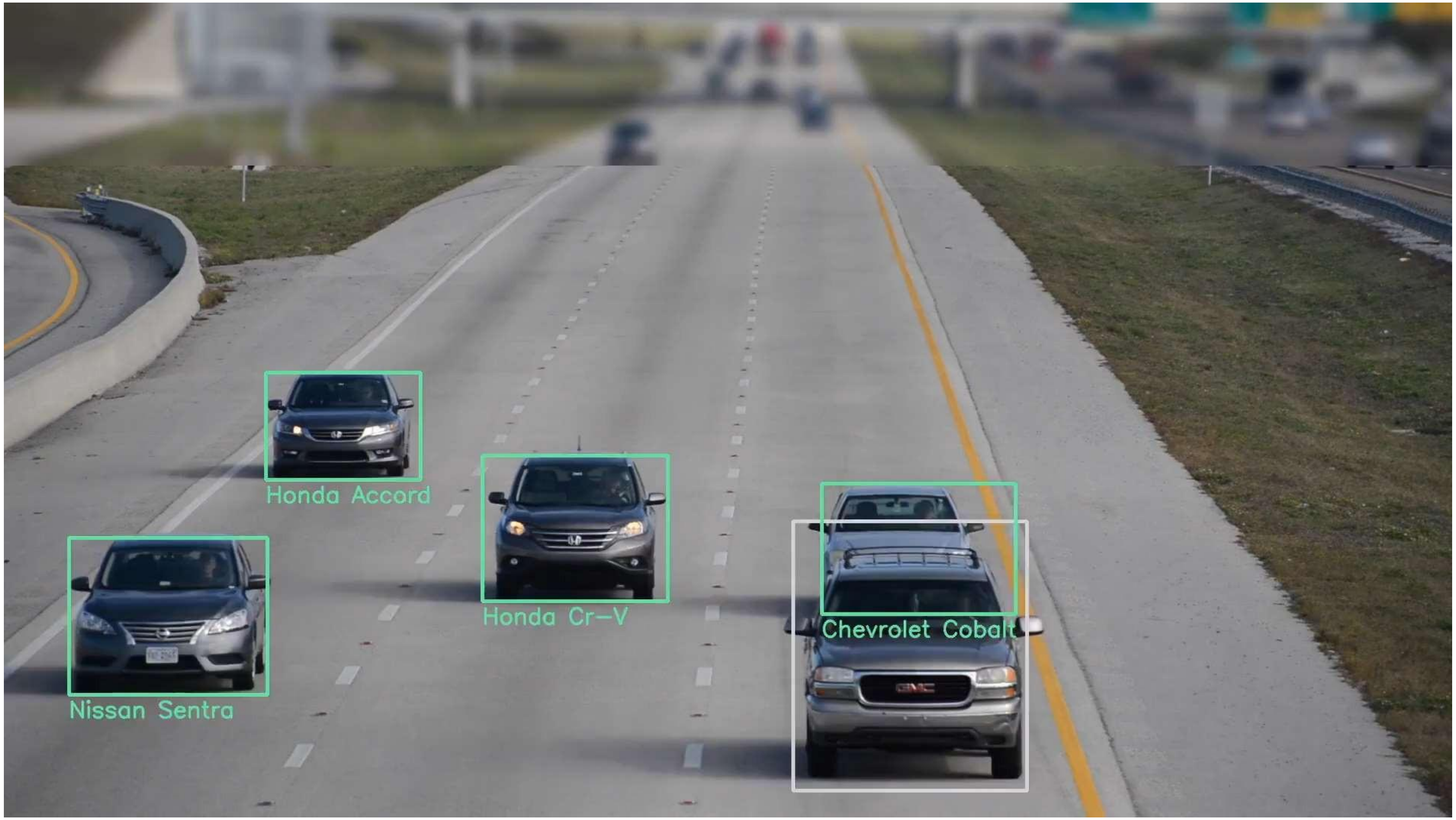
# Breakthrough with neural networks



# Image segmentation and recognition

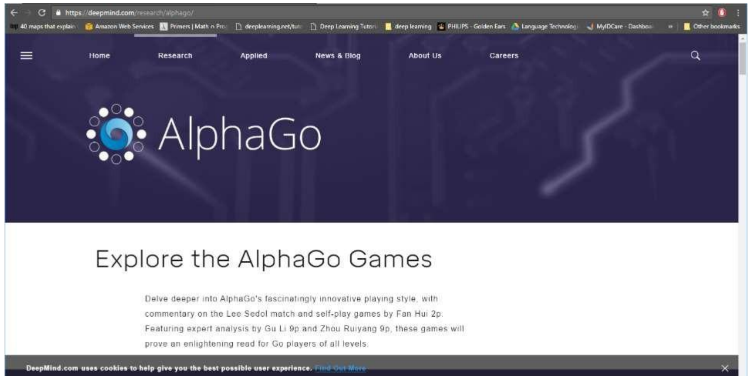


# Image recognition




https://www.sighthound.com/technology/

# Breakthroughs with neural networks



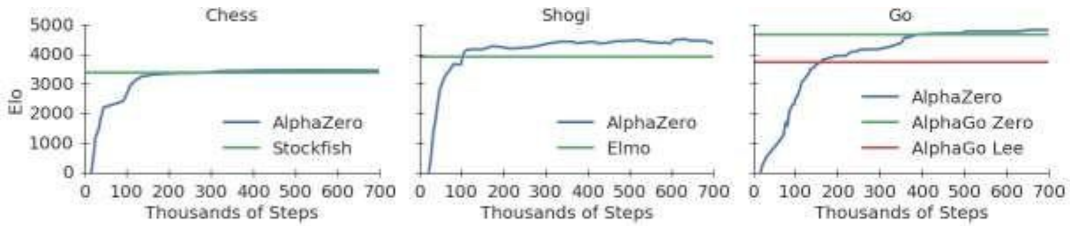
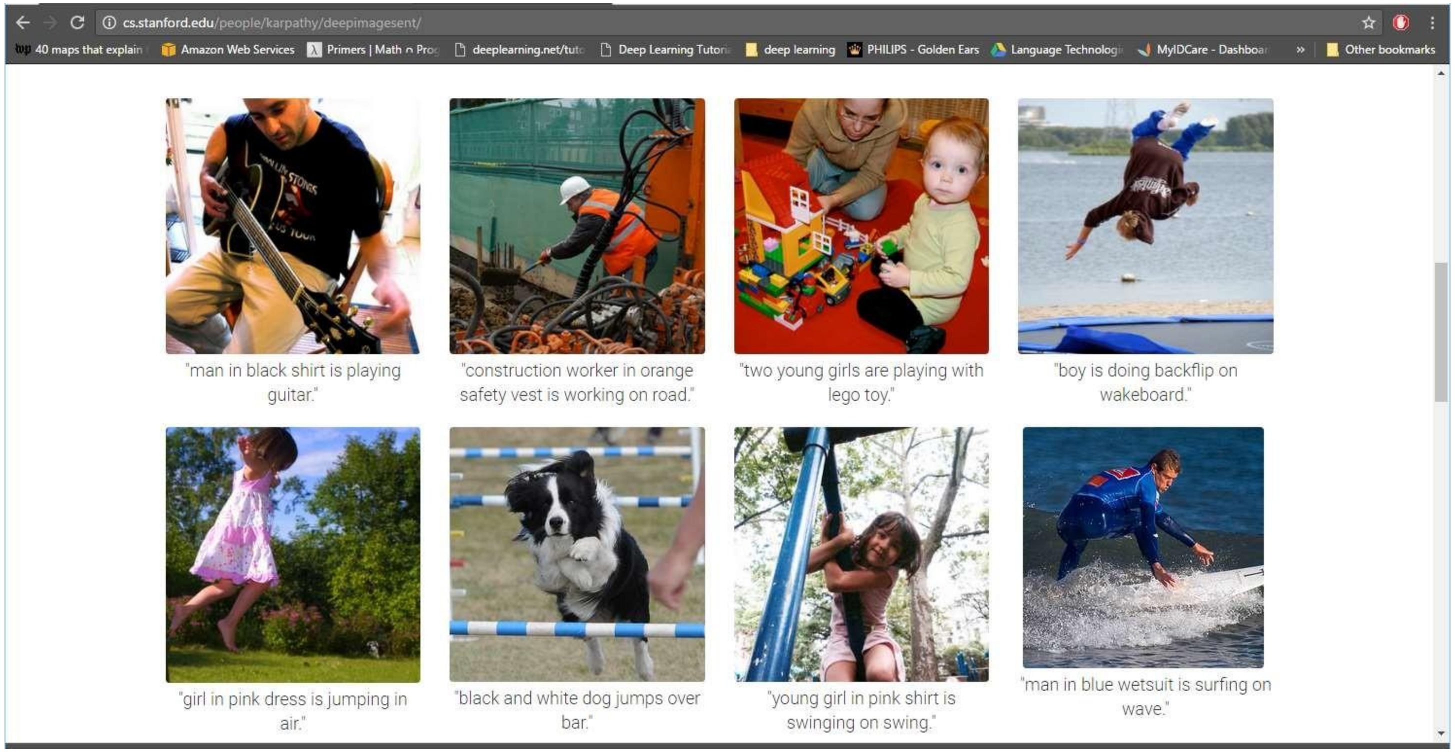




Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared to 2017 CSA world-champion program *Elmo*. **c** Performance of *AlphaZero* in Go, compared to *AlphaGo Lee* and *AlphaGo Zero* (20 block / 3 day) (29).

# Success with neural networks




- Captions generated entirely by a neural network

# Successes with neural networks

- And a variety of other problems:
  - From art to astronomy to healthcare..
  - and even predicting stock markets!

# Objectives of this course

- Understanding neural networks
- Comprehending the models that do the previously mentioned tasks
  - And maybe build them
- Design, build and train networks for various tasks

- *You will not become an expert in one course*

# Course objectives: Broad level

*Deep Dive into Artificial Neural Networks*

- Concepts
  - Types of neural networks and underlying ideas
  - Learning in neural networks
    - Training, concepts, practical issues
  - Architectures and applications
- Practical
  - Familiarity with training and parameter tuning
  - Implement various neural network architectures
- Overall: Set you up for further work in your area

# Course learning objectives: Topics

- Basic network formalisms (for classification and prediction):
  - Multi-Layer Perceptron (MLP)
  - Convolutional networks (CNN)
  - Recurrent networks (RNN)
- Some advanced formalisms (for creation)
  - Generative models: VAEs
  - Adversarial models: GANs
- Applications we will touch upon:
  - Computer vision: recognizing images
  - Text processing: modelling and generating language
  - ….

# Reading

- List of books on Canvas Course Page
  - Primary: https://www.deeplearningbook.org/
    - "Deep Learning", Goodfellow, Bengio, Courville
  - Reference: https://www.manning.com/books/deep-learning-with-python
    - "Deep Learning with Python", Francois Chollet.
  - Additional reading material will be posted on Canvas, if needed

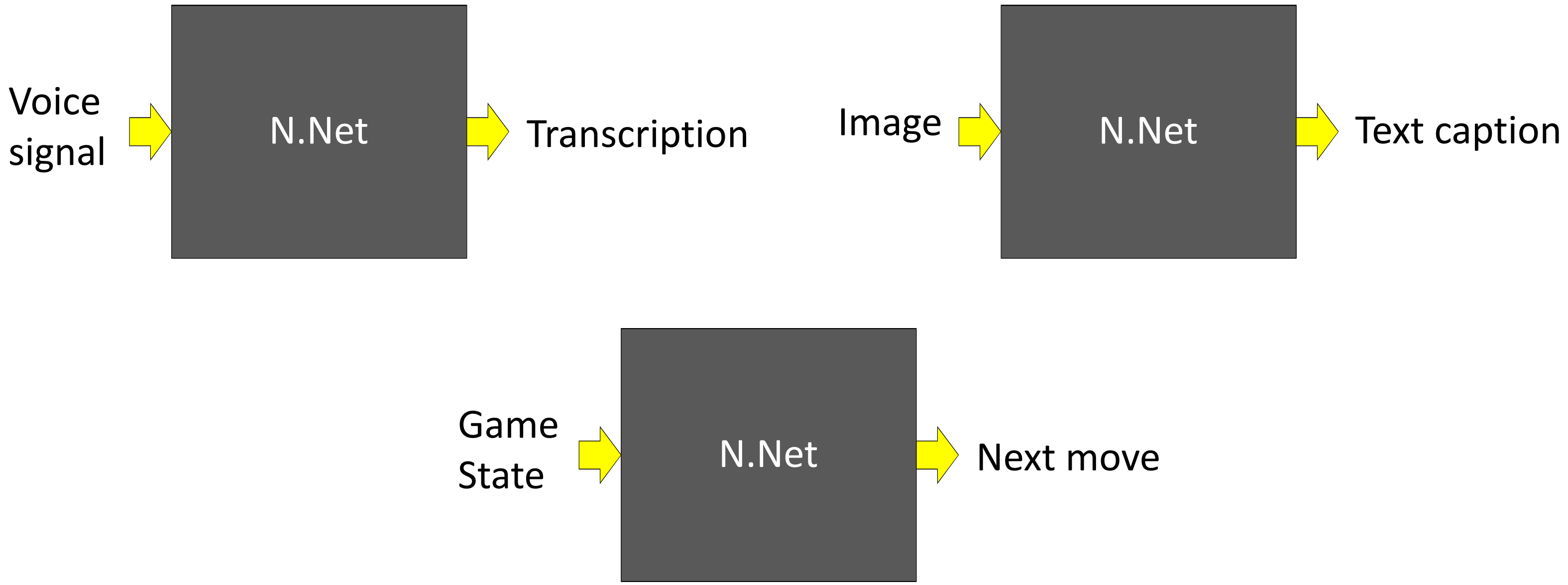
# Logistics

- Most relevant info on Canvas
  - Handout
  - Schedule of Webinars, Quiz, Assignments, ….
  - Lecture Slides
  - Lab Sheets
  - One Quiz, Two Assignments
    - Quiz, one assignment before midsem
    - One assignment after midsem
    - *submissions beyond deadline will be deducted some marks / day (unless medical emergencies)*
    - Programming using Python, Keras / Tensorflow

# Questions?

- Please post on Discussions Forum
  - TAs and instructors will answer
  - Collaborate with your fellow students

# So what are neural networks??




- What are these boxes?

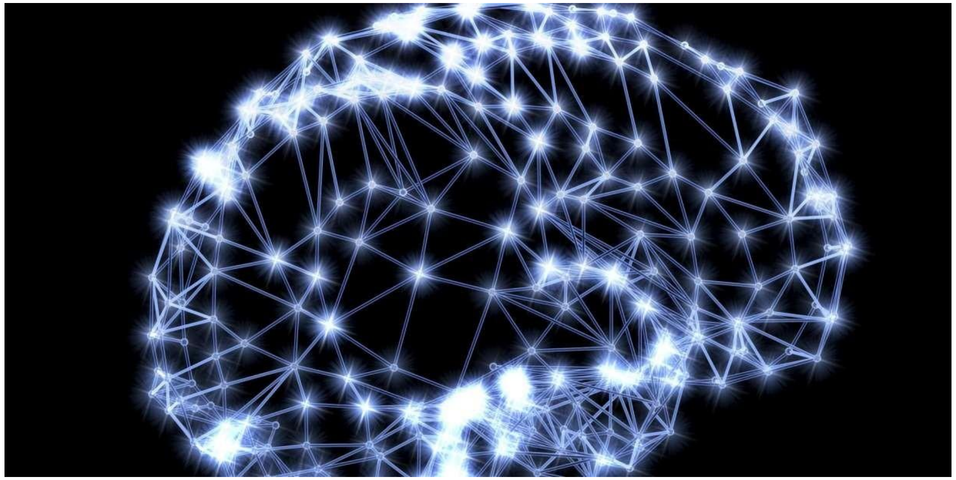
# So what are neural networks??

- It begins with this..

# Early Models of Human Cognition

- Associationism
  - Humans learn through association
- 400BC-1900AD: Plato, David Hume, Ivan Pavlov..

# Observation: ***The Brain***

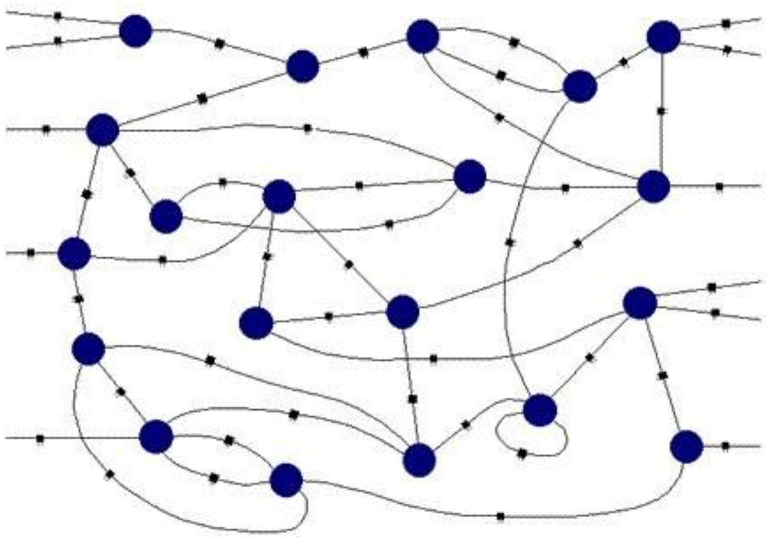


- Mid 1800s: The brain is a mass of interconnected neurons

# Brain: Interconnected Neurons

- Many neurons connect *in* to each neuron
- Each neuron connects *out* to many neurons

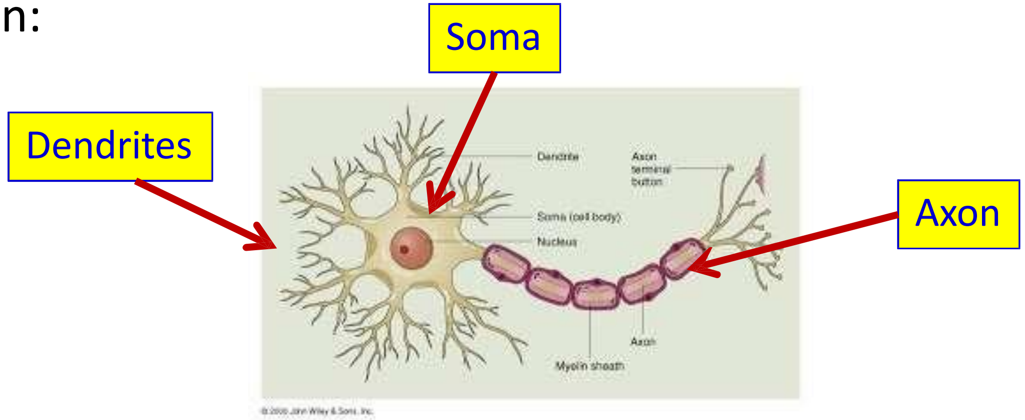
# Connectionist Machines



- Network of processing elements
- **All world knowledge is stored in the *connections* between the elements**
  - *But what are these individual elements?*

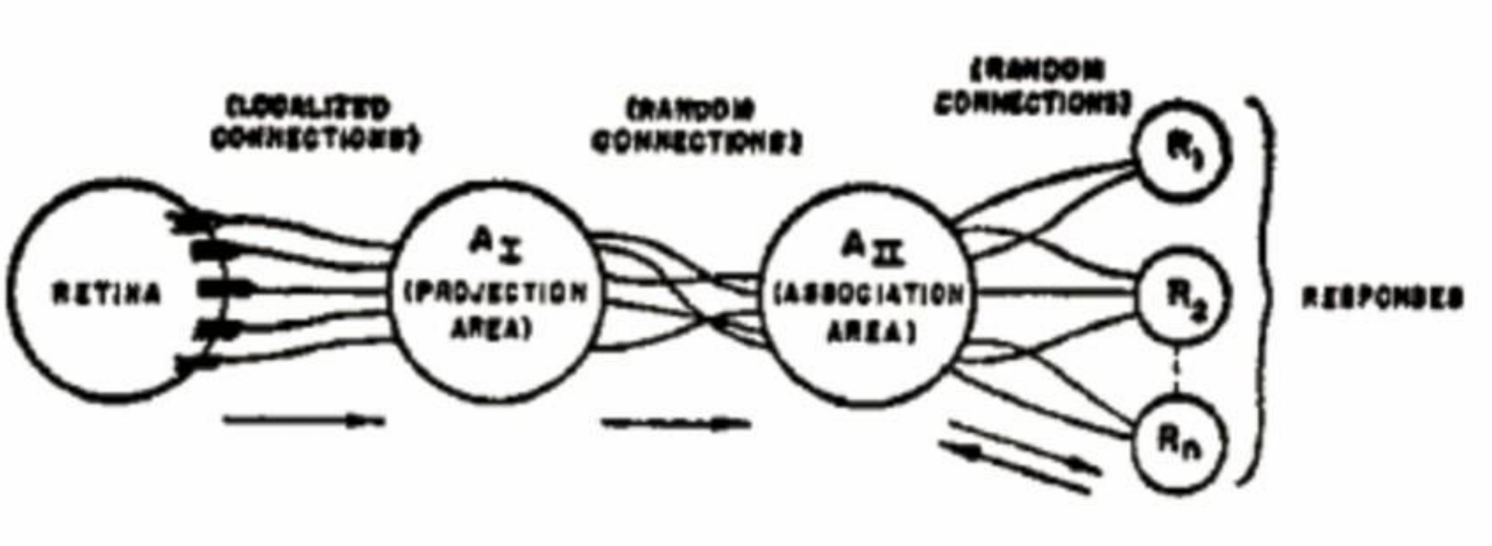
# Modelling the brain

- What are the units?
- A neuron:




- Signals come in through the dendrites into the Soma
- A signal goes out via the axon to other neurons
  - Only one axon per neuron
- Factoid that may only interest me: Neurons do not undergo cell division
  - Neurogenesis occurs from neuronal stem cells, and is minimal after birth

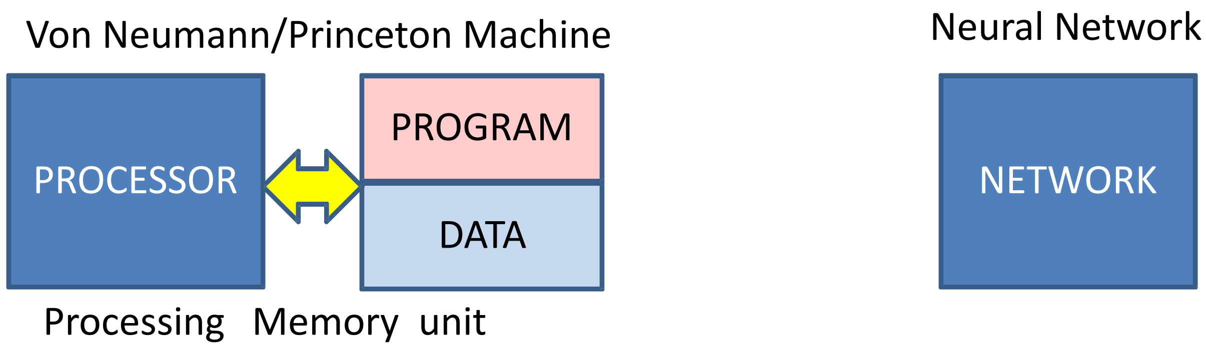
# Rosenblatt's perceptron




- Original perceptron model
  - Groups of sensors (S) on retina combine onto cells in association area A1
  - Groups of A1 cells combine into Association cells A2
  - Signals from A2 cells combine into response cells R
  - All connections may be excitatory or inhibitory

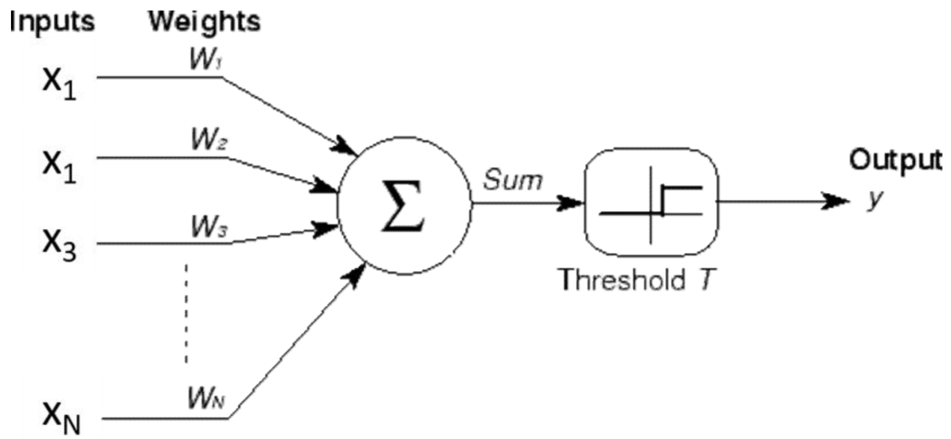
# Connectionist Machines

- Neural networks are *connectionist* machines
  - As opposed to Von Neumann Machines




- The machine has many non-linear processing units
  - The program is the connections between these units
    - Connections may also define memory

# Simplified mathematical model of Perceptron




- Number of inputs combine linearly
  - Threshold logic: Fire if combined input exceeds or equal to threshold

$$Y = \begin{cases} 1 \;\; if \sum_i w_i x_i - T >= 0 \\ 0 \;\; else \end{cases}$$

# Also provided a learning algorithm

$$\mathbf{w} = \mathbf{w} + \eta(d(\mathbf{x}) - y(\mathbf{x}))\mathbf{x}$$

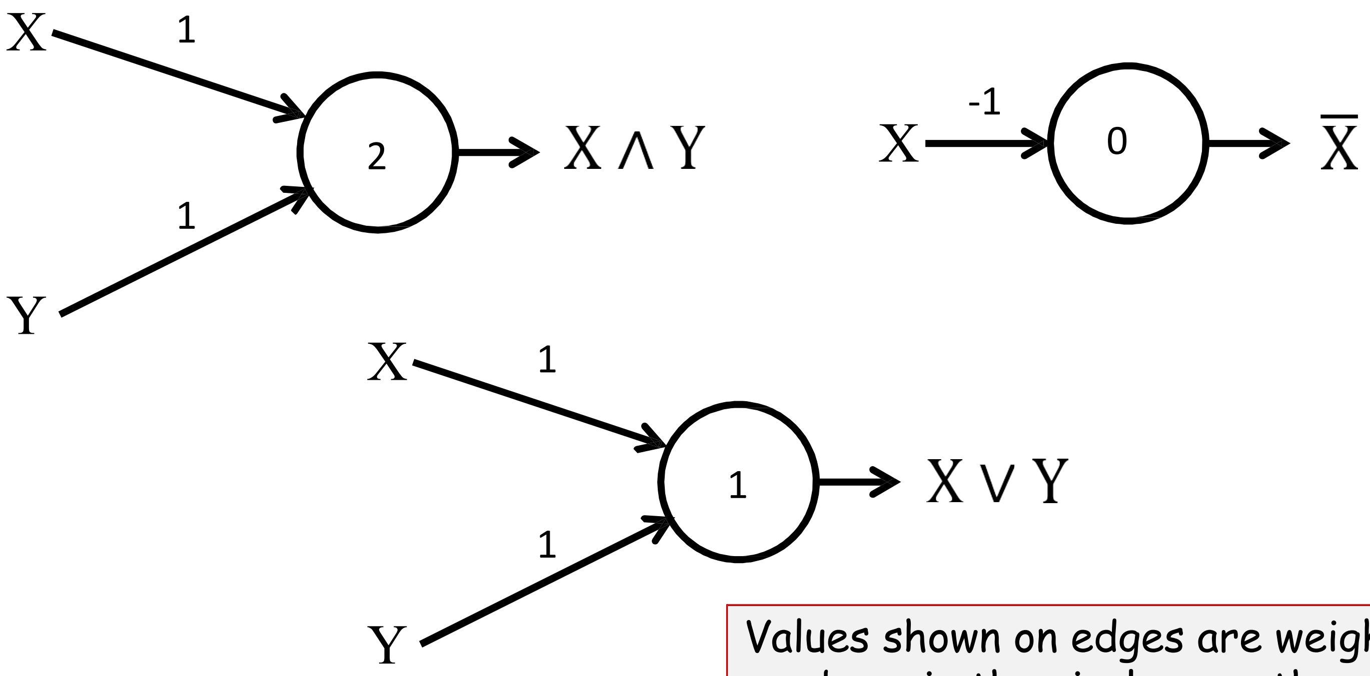
Sequential Learning:

$d(x)$ is the desired output in response to input $x$

$y(x)$ is the actual output in response to $x$

- Boolean tasks
- Update the weights whenever the perceptron output is wrong
- Proved convergence for linearly separable classes

# Perceptron



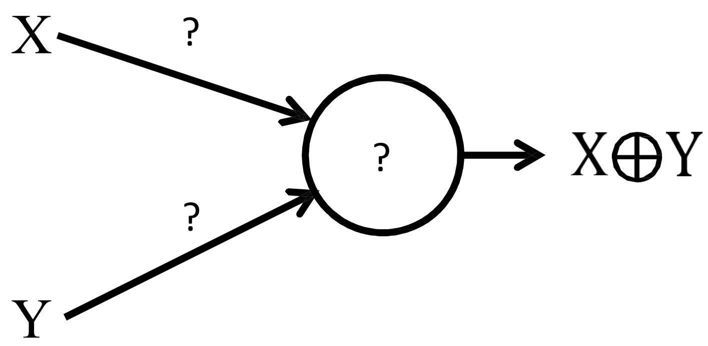

Values shown on edges are weights, numbers in the circles are thresholds

- Easily shown to mimic any Boolean gate
- But…

# Perceptron

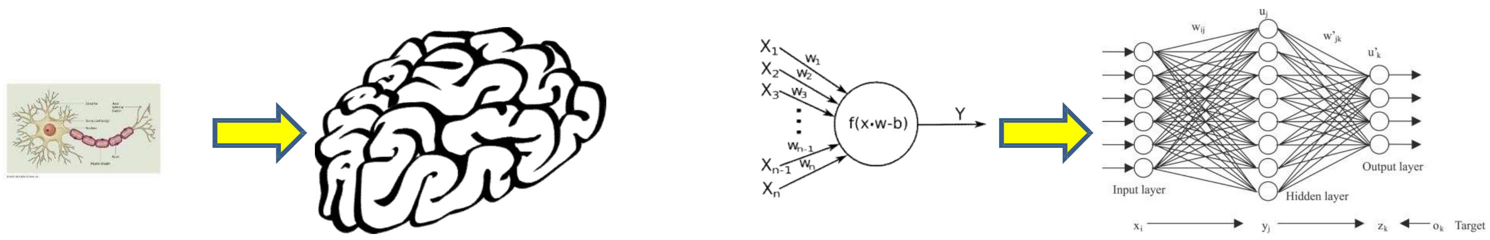
No solution for XOR!

*Not universal!*




- Minsky and Papert, 1968

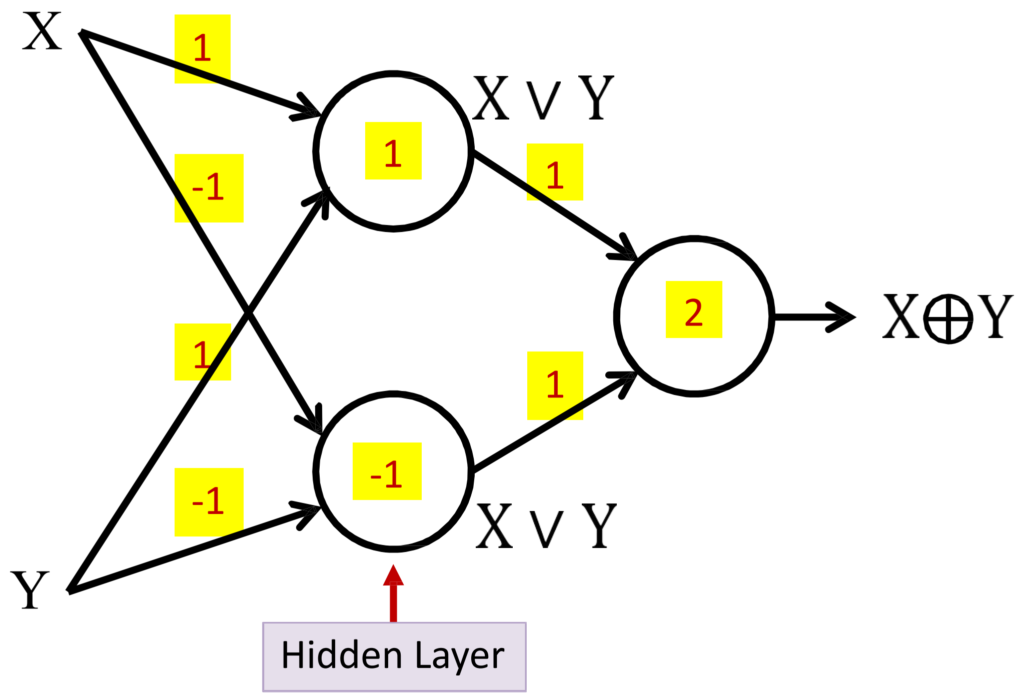
# A single neuron is not enough




- Individual elements are weak computational elements
  - Marvin Minsky and Seymour Papert, 1969, *Perceptrons: An Introduction to Computational Geometry*

- *Networked* elements are required

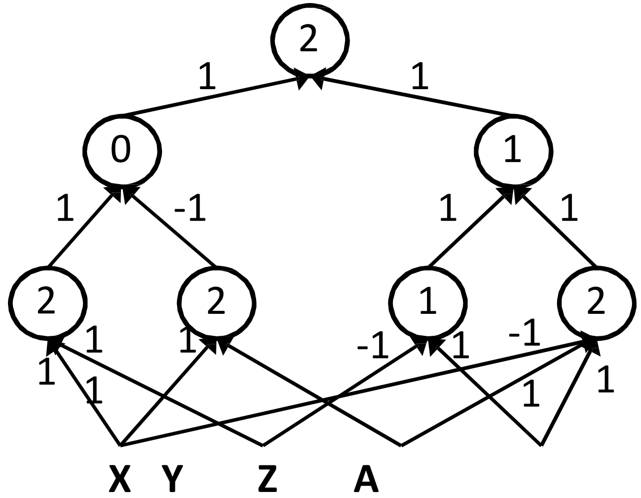
# Multi-layer Perceptron!




- XOR
  - The first layer is a “hidden” layer
  - Also originally suggested by Minsky and Papert 1968

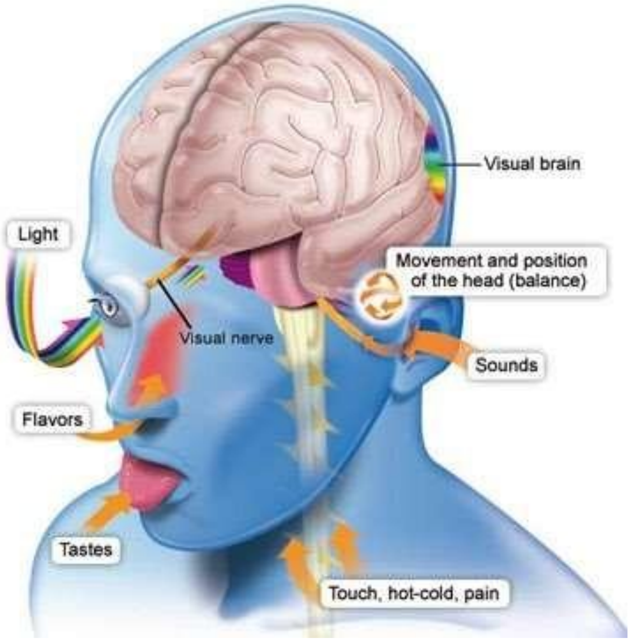
# A more generic model

$$((A\&\bar{X}\&Z)|(A\&\bar{Y}))\&((X\ \&\ Y)|\overline{(X\&Z)})$$



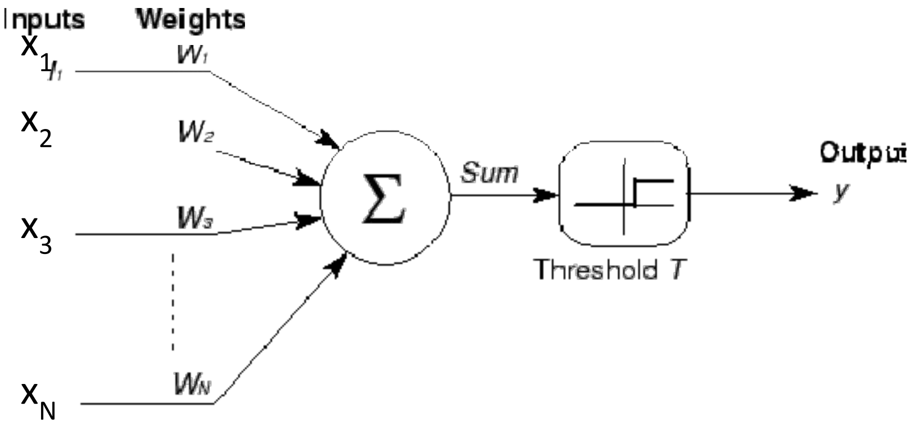

- A “multi-layer” perceptron
- Can compose arbitrarily complicated Boolean functions!
  - In cognitive terms: Can compute arbitrary Boolean functions over sensory input
  - More on this in the next class

# But our brain is not Boolean



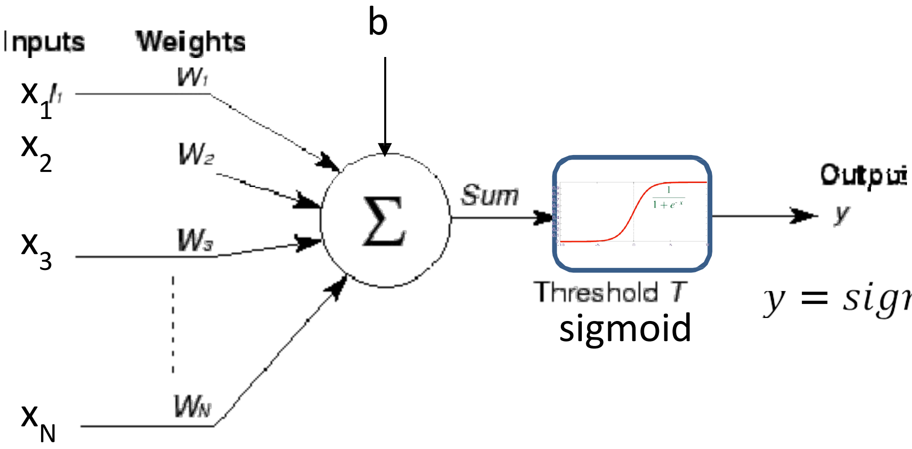

- We have real inputs
- We make non-Boolean inferences/predictions

# The perceptron with *real* inputs




- $x_1 \ldots x_N$ are real valued
- $w_1 \ldots w_N$ are real valued
- Unit "fires" if weighted input exceeds a threshold

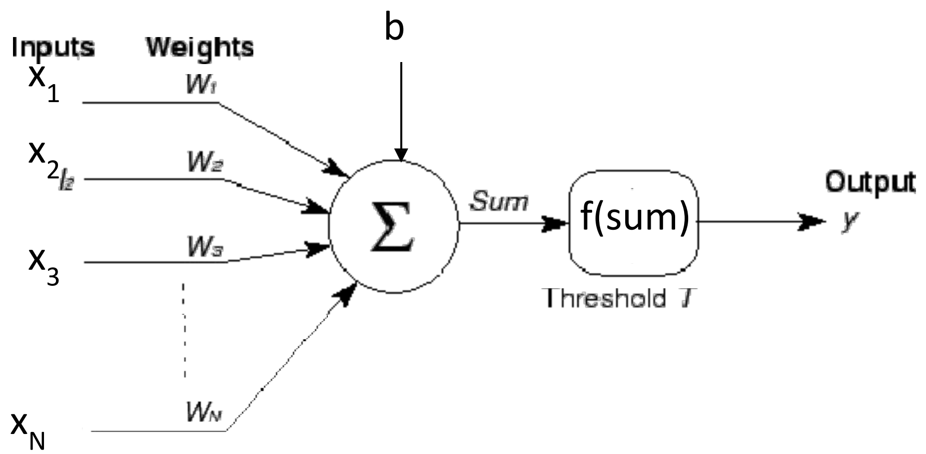
# The perceptron with *real* inputs and a real *output*




$$y = sigmoid(\sum_i w_i x_i + b)$$

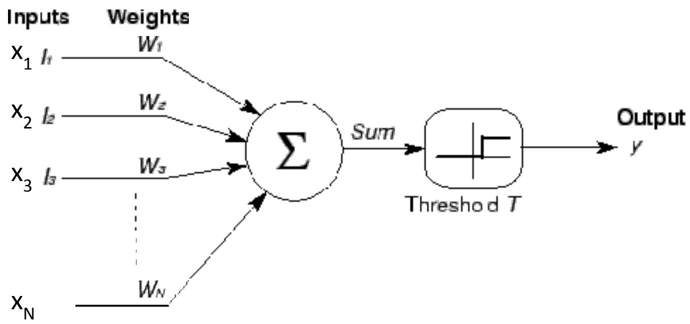
- $x_1 \ldots x_N$ are real valued
- $w_1 \ldots w_N$ are real valued
- The output $y$ can also be real valued
  - Sometimes viewed as the "probability" of firing

# The "real" valued perceptron

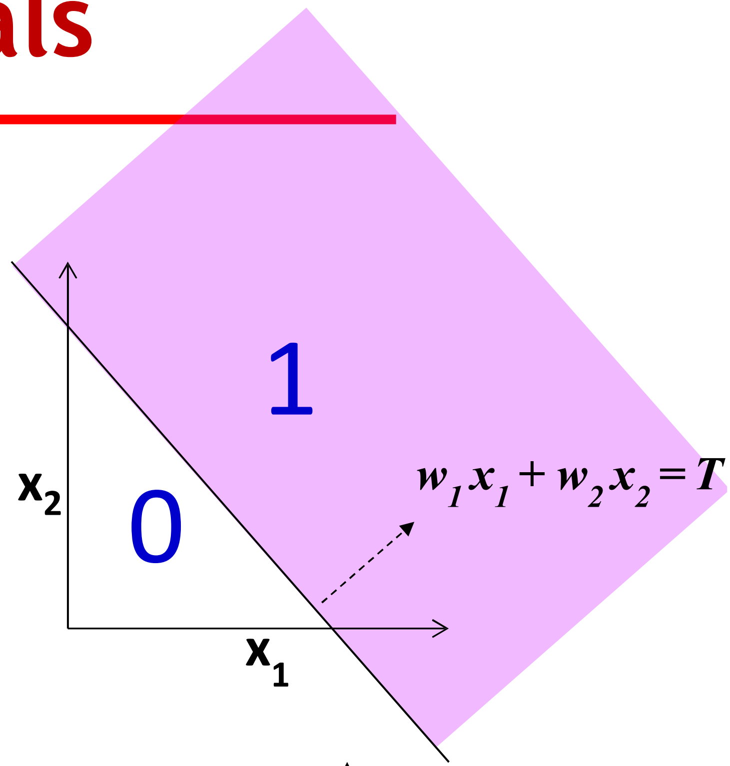


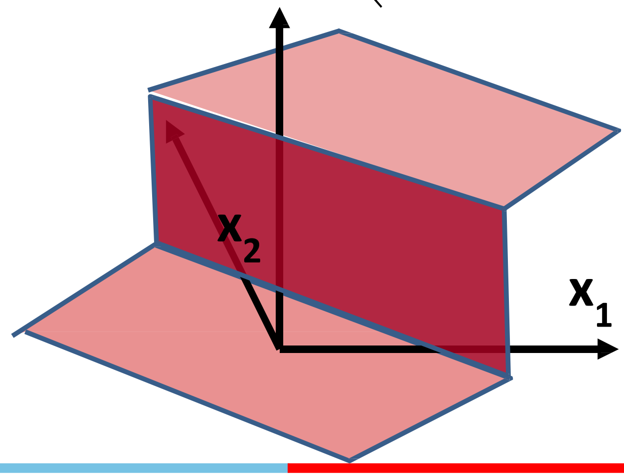

- Any real-valued "activation" function may operate on the weighted- sum input
  - We will see several later
  - Output will be real valued
- The perceptron maps real-valued inputs to real-valued outputs
- *Is useful to continue assuming Boolean outputs though, for interpretation*

# A Perceptron on Reals



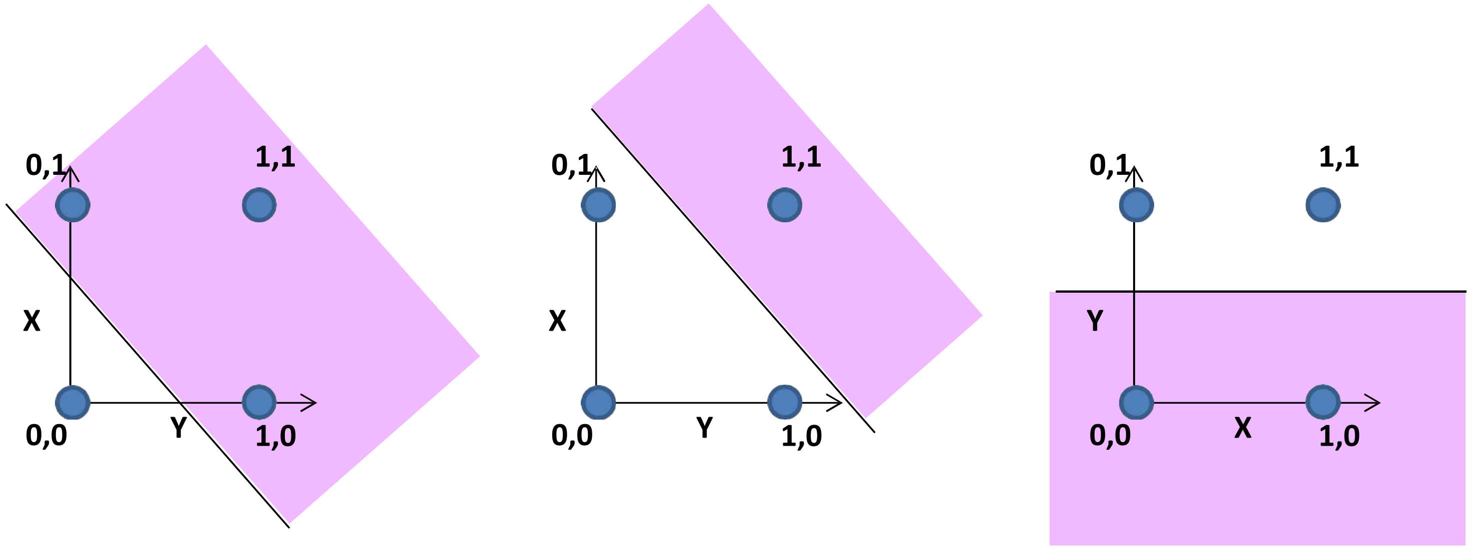

$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$



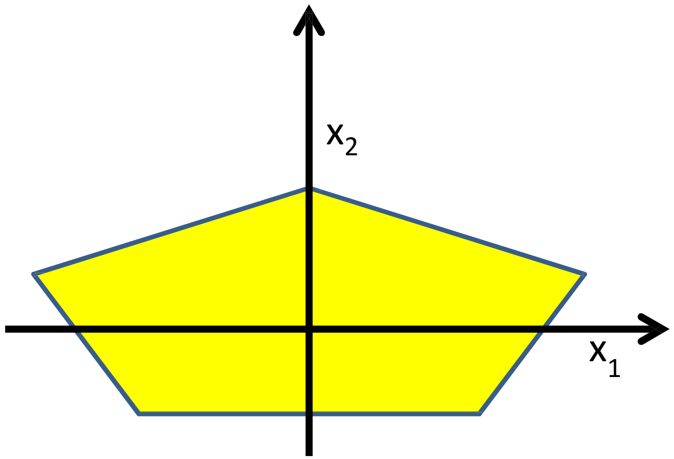

- A perceptron operates on *real*-valued vectors
  - **This is a *linear classifier***

# Boolean functions with a real perceptron




- Boolean perceptrons are also linear classifiers
  - Purple regions have output 1 in the figures
  - What are these functions
  - Why can we not compose an XOR?

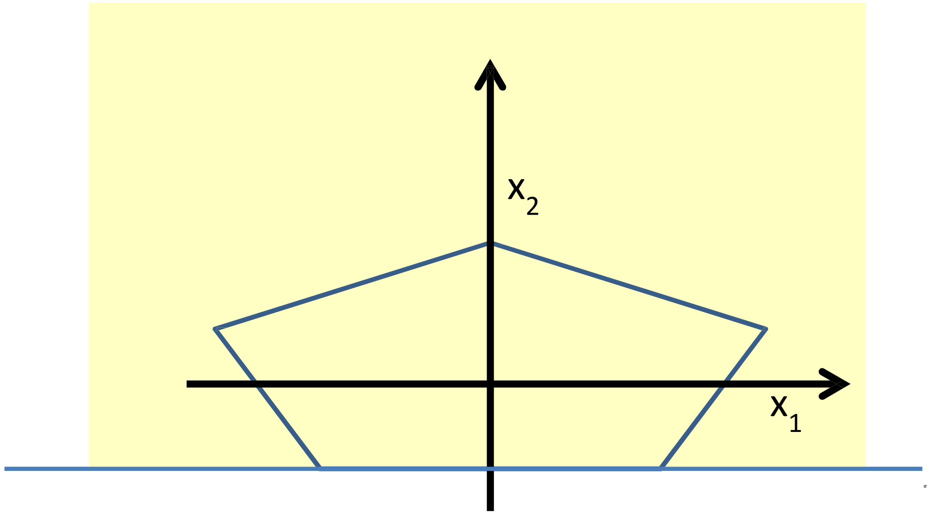
# Composing complicated “decision” boundaries

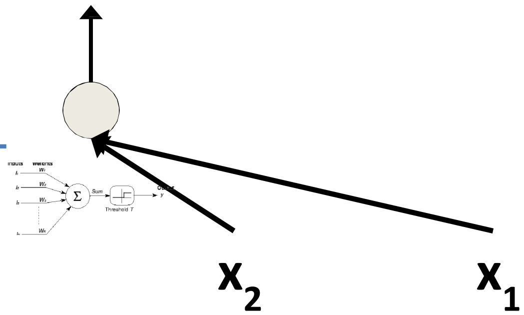



Can now be composed into “networks” to compute arbitrary classification “boundaries”

- Build a network of units with a single output that fires if the input is in the coloured area

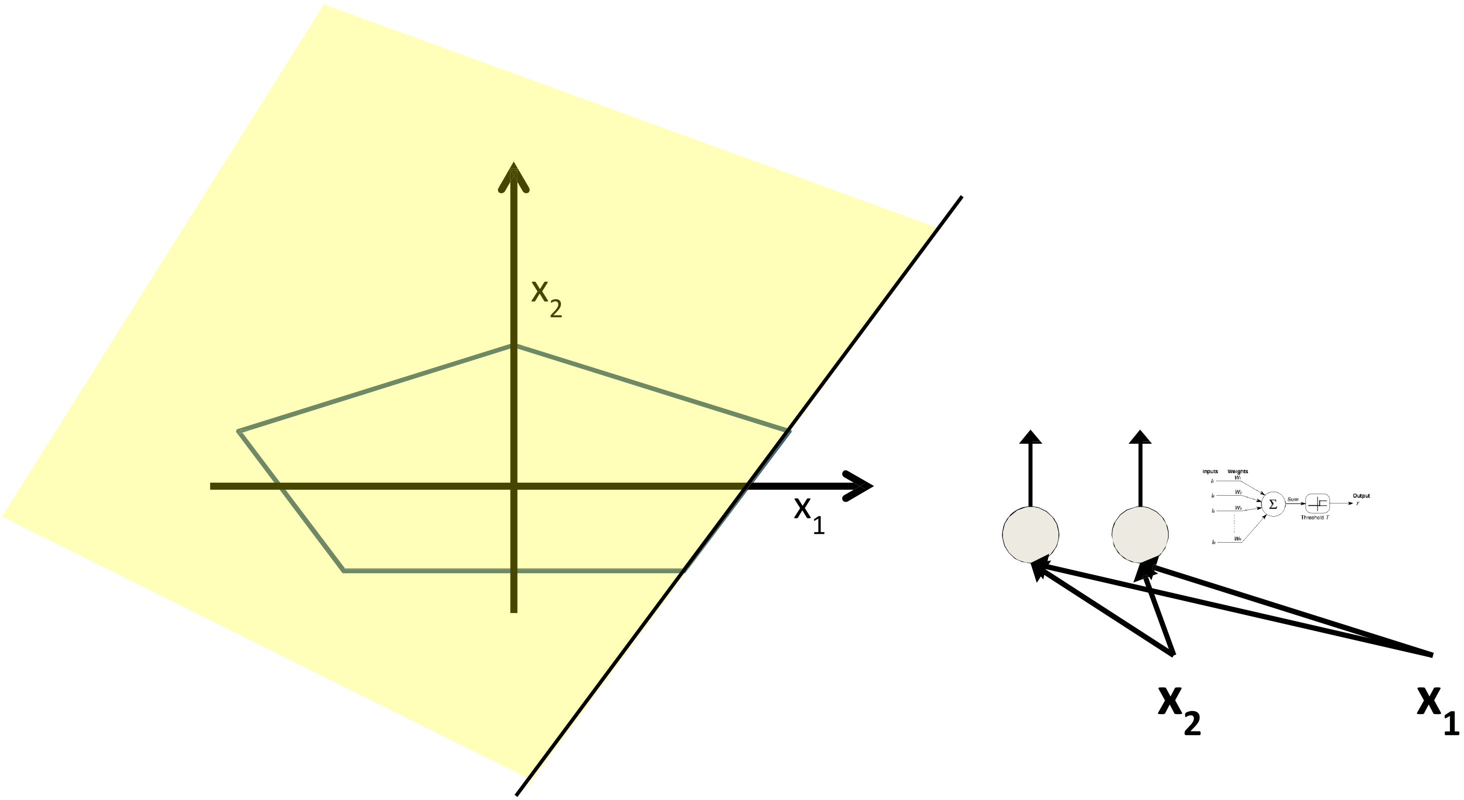
# Booleans over the reals



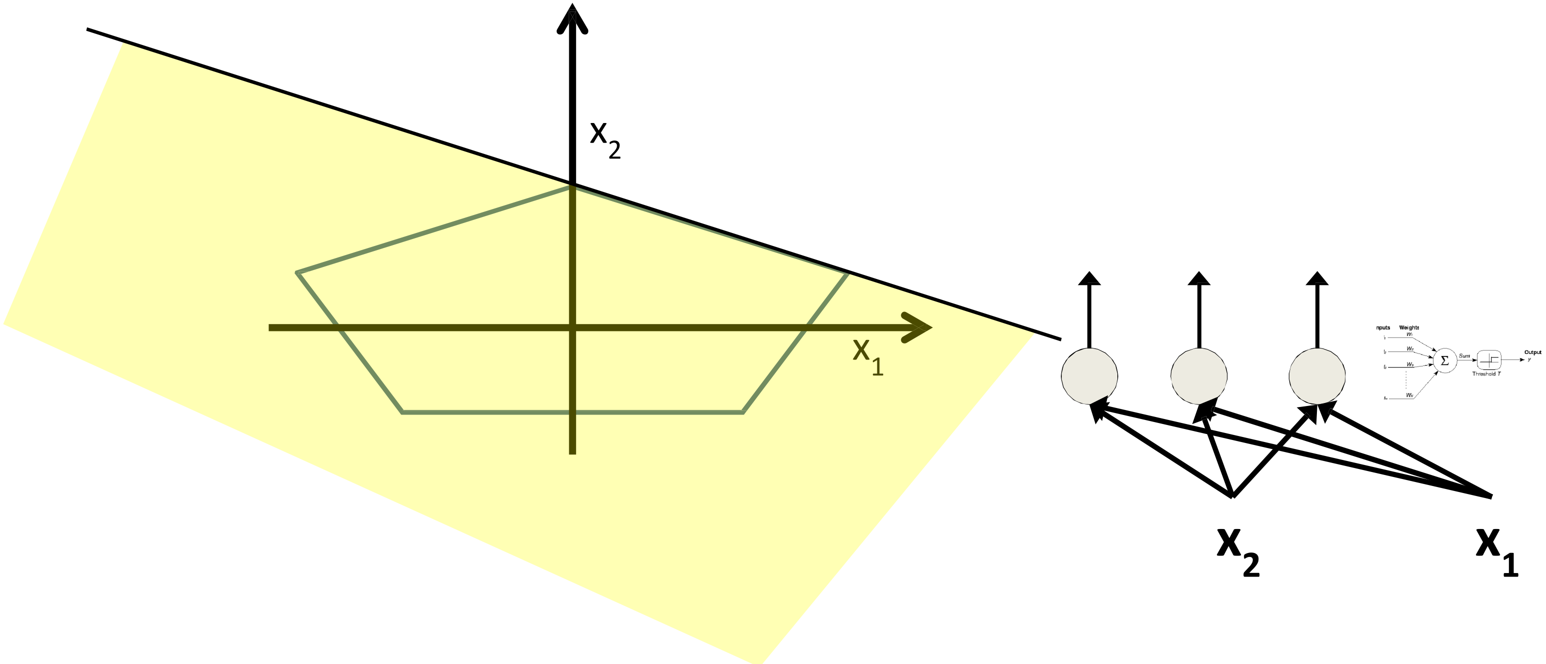

- The network must fire if the input is in the coloured area

# Booleans over the reals



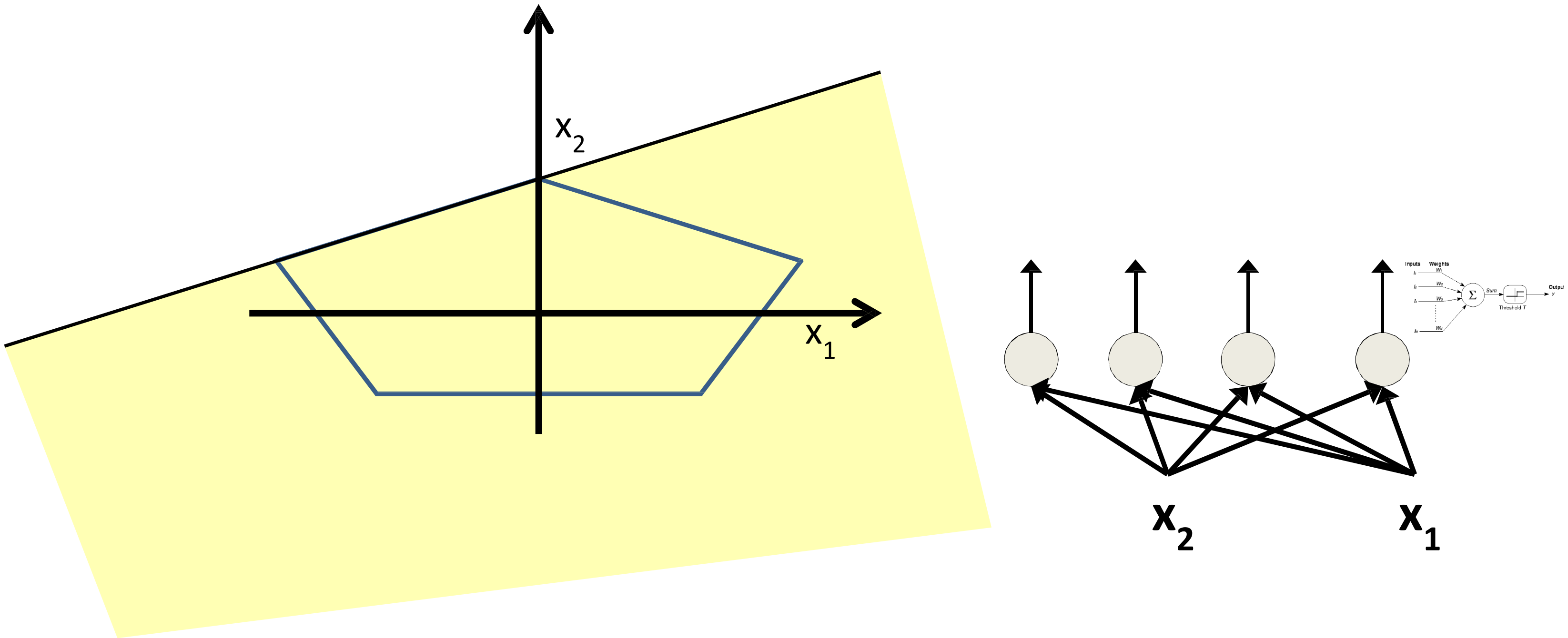

- The network must fire if the input is in the coloured area

# Booleans over the reals



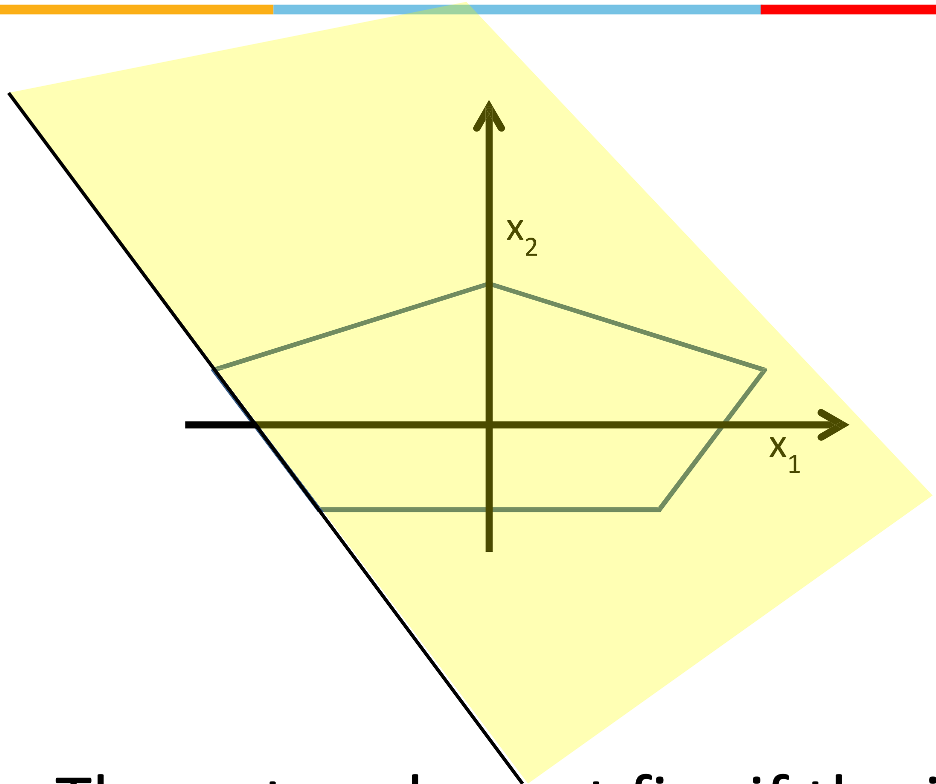

- The network must fire if the input is in the coloured area

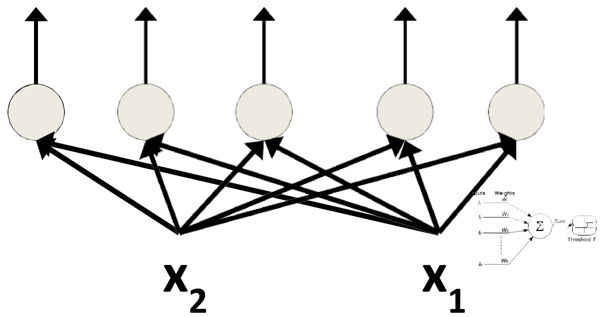
# Booleans over the reals




- The network must fire if the input is in the coloured area

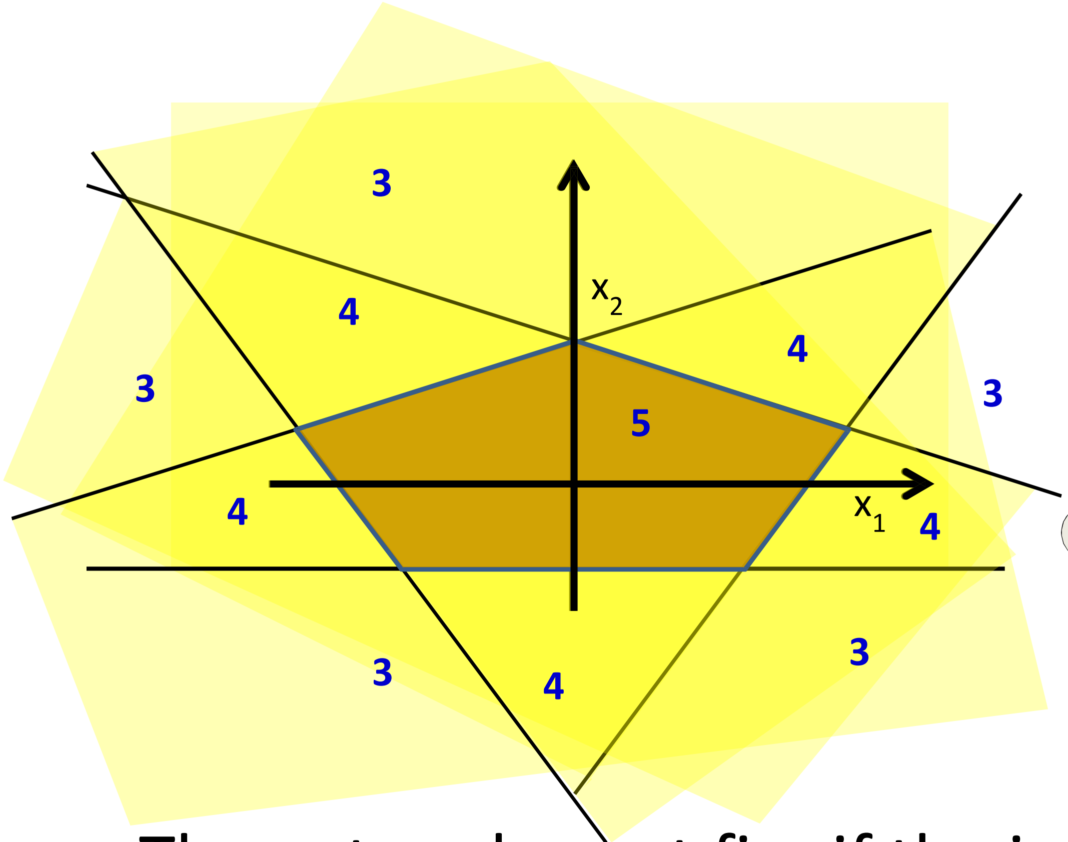
# Booleans over the reals

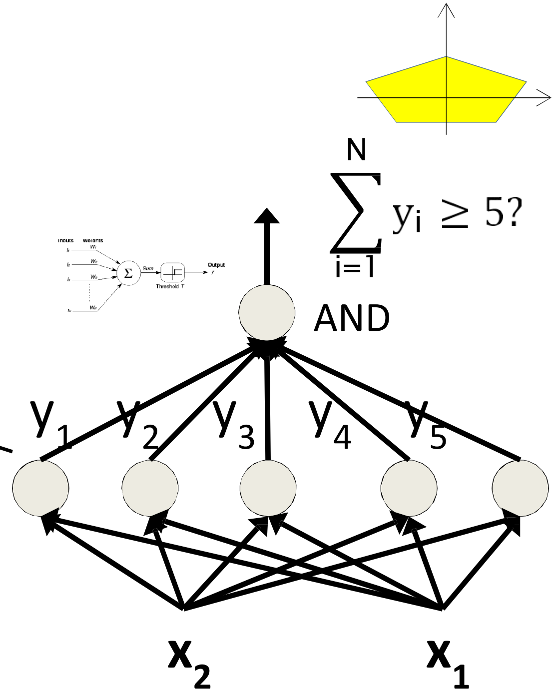



- The network must fire if the input is in the coloured area

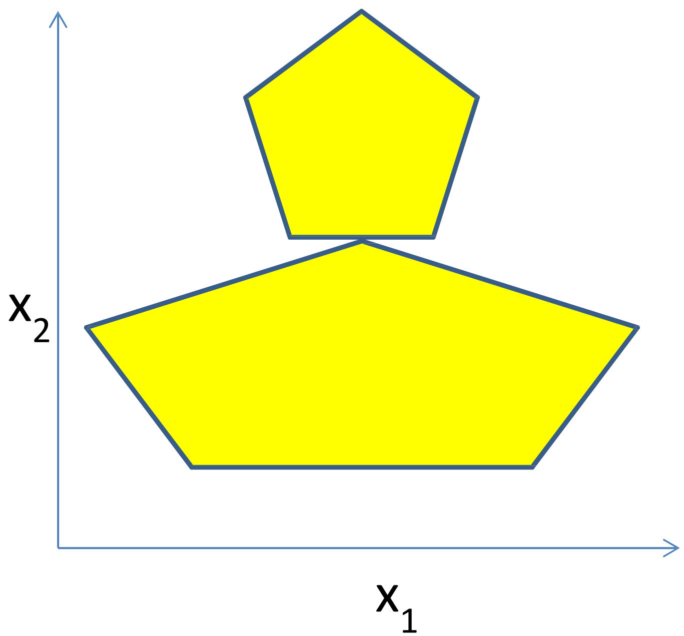
# Booleans over the reals

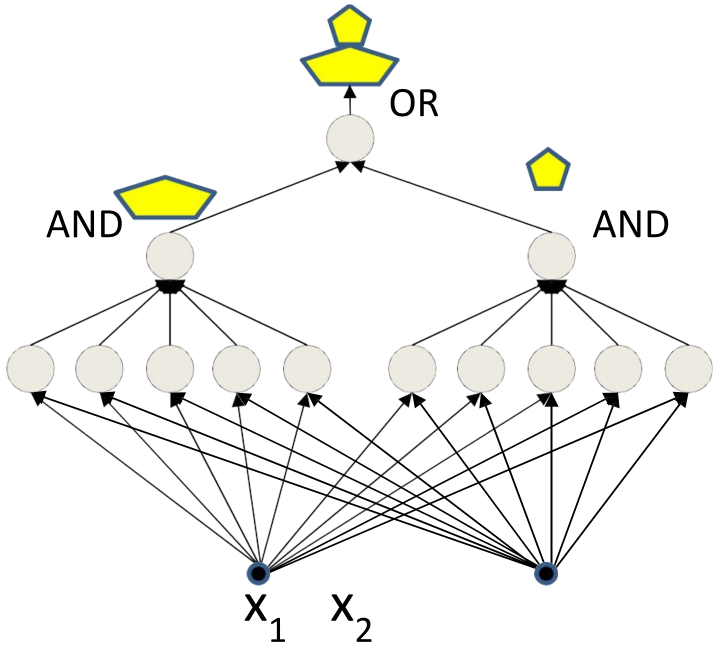



- The network must fire if the input is in the coloured area

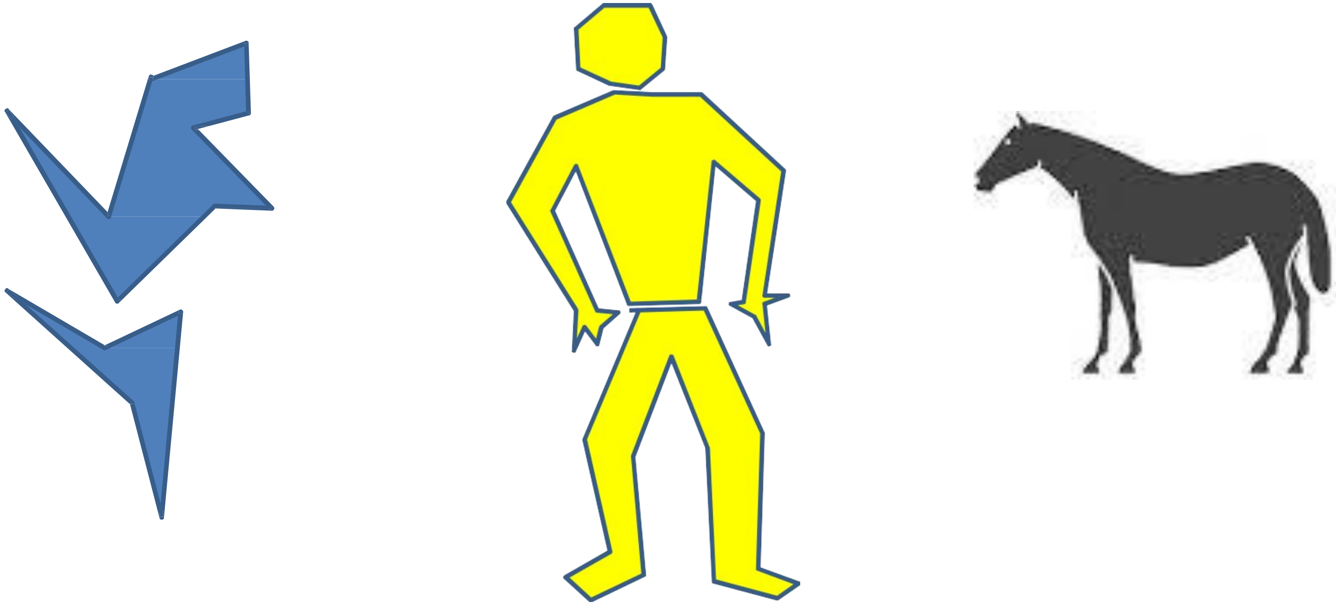
# More complex decision boundaries



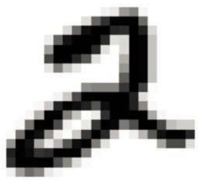

- Network to fire if the input is in the yellow area
  - "OR" two polygons
  - A third layer is required

# Complex decision boundaries

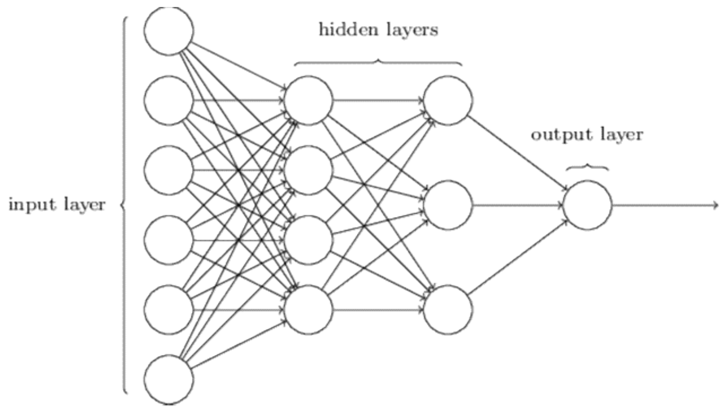


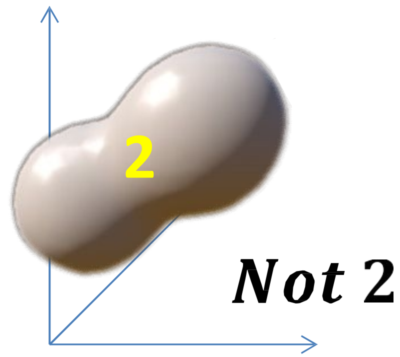
- Can compose very complex decision boundaries
  - How complex exactly?     More on this in the next class

# Complex decision boundaries



784 dimensions (MNIST)




784 dimensions

- Classification problems: finding decision boundaries in high-dimensional space
  - Can be performed by an MLP
- MLPs can *classify* real-valued inputs

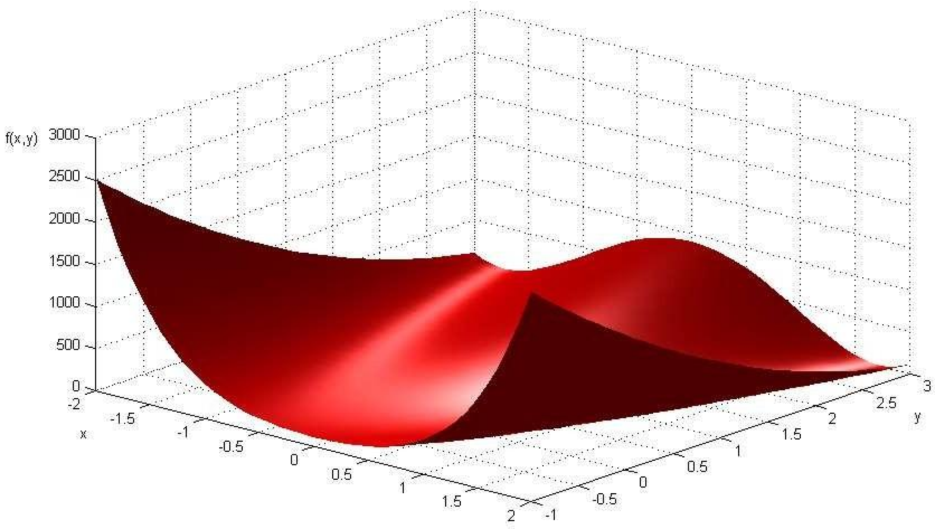
# Story so far

- **MLPs are connectionist computational models**
  - Individual perceptrons are computational equivalent of neurons
  - The MLP is a layered composition of many perceptrons

- **MLPs can model Boolean functions**
  - Individual perceptrons can act as Boolean gates
  - Networks of perceptrons are Boolean functions

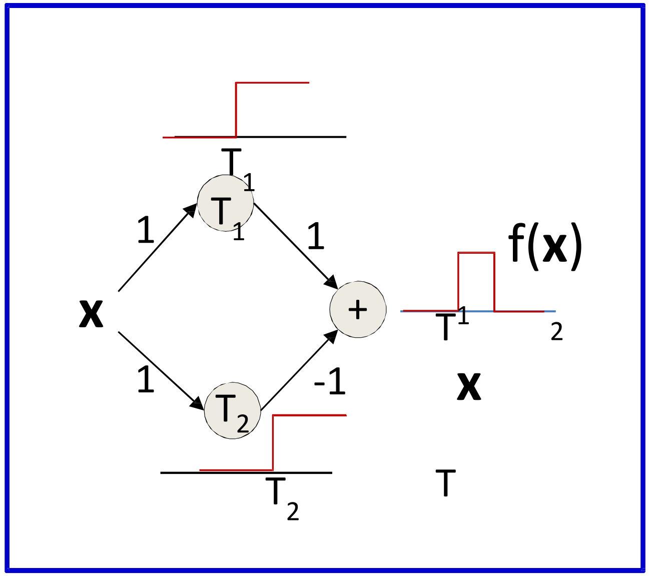
- **MLPs are Boolean *machines***
  - They represent Boolean functions over linear boundaries
  - They can represent arbitrary decision boundaries
  - They can be used to *classify* data

# But what about continuous valued *output?*



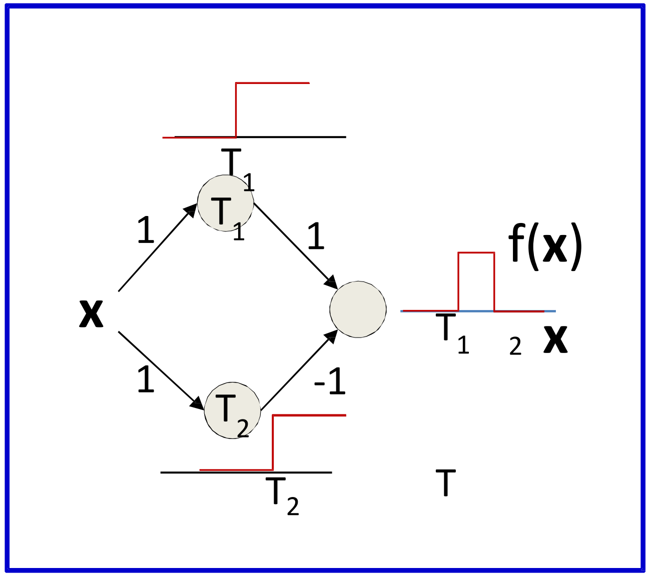

- Inputs may be real valued
- Can outputs be continuous-valued too?

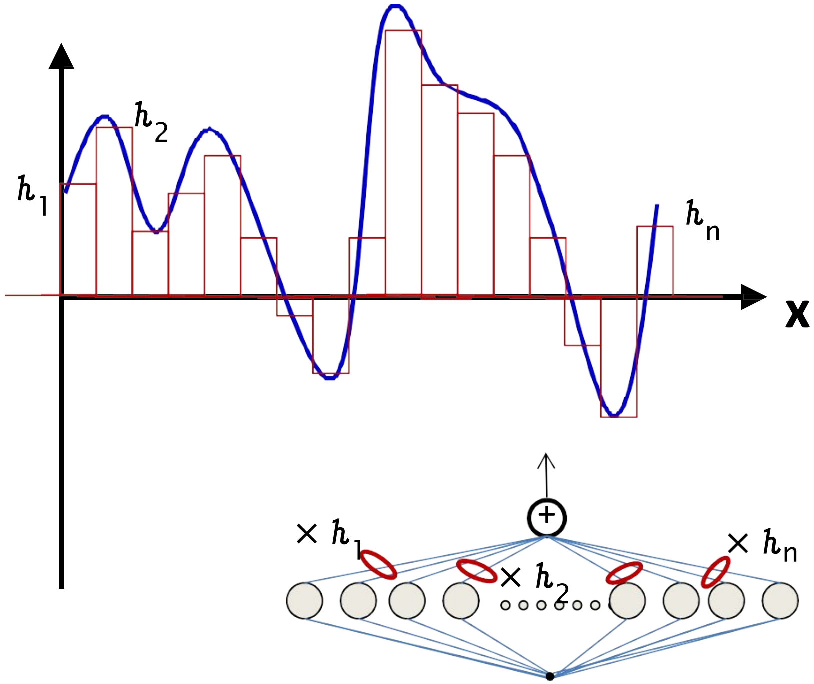
# MLP as a continuous-valued regression




- A simple 3-unit MLP with a "summing" output unit can generate a "square pulse" over an input
  - Output is 1 only if the input lies between $T_1$ and $T_2$
  - $T_1$ and $T_2$ can be arbitrarily specified

# MLP as a continuous-valued regression




- A simple 3-unit MLP can generate a "square pulse" over an input
- **An MLP with many units can model an arbitrary function over an input**
  - To arbitrary precision
    - Simply make the individual pulses narrower
- This generalizes to functions of any number of inputs

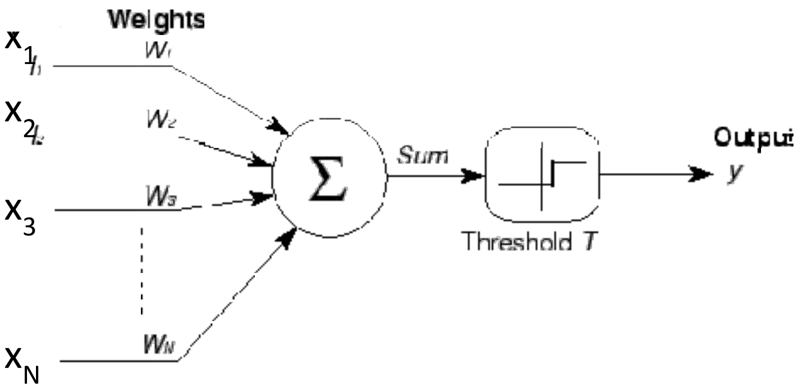
# Story so far

- **Multi-layer perceptrons are connectionist computational models**
- **MLPs are *classification engines***
  - They can identify classes in the data
  - Individual perceptrons are feature detectors
  - The network will fire if the combination of the detected basic features matches an "acceptable" pattern for a desired class of signal
- **MLP can also model continuous valued functions**

# So what does the perceptron really model?

- Is there a "semantic" interpretation?
  - Cognitive version: Is there an interpretation beyond the simple characterization as Boolean functions over sensory inputs?

# Lets look at the weights

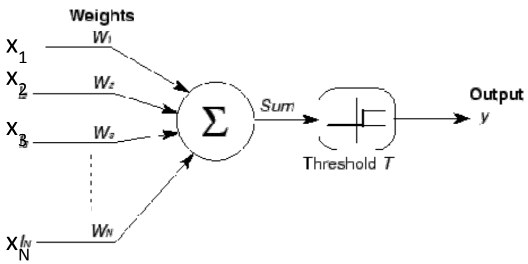



$$y = \begin{cases} 1 \ if \ \sum_i w_i x_i \geq T \\ 0 \ else \end{cases}$$

$$y = \begin{cases} 1 \ if \ \mathbf{x}^T \mathbf{w} \geq T \\ 0 \ else \end{cases}$$

- What do the *weights* tell us?
  - The neuron fires if the inner product between the weights and the inputs exceeds a threshold

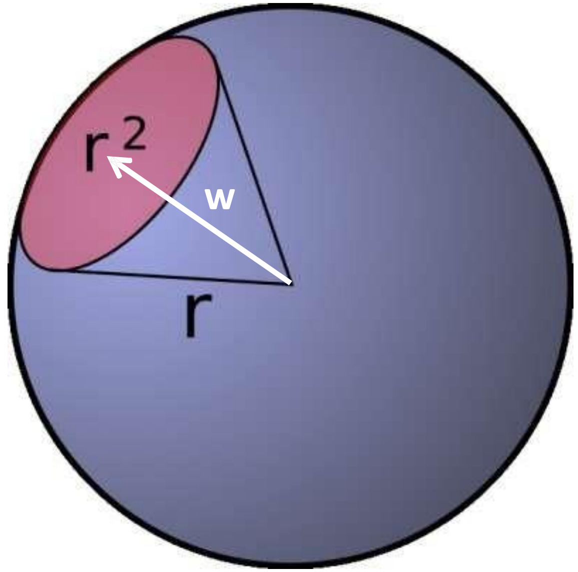
# The weight as a “template”




$$X^T W > T$$

$$\cos\theta > \frac{T}{|X|}$$

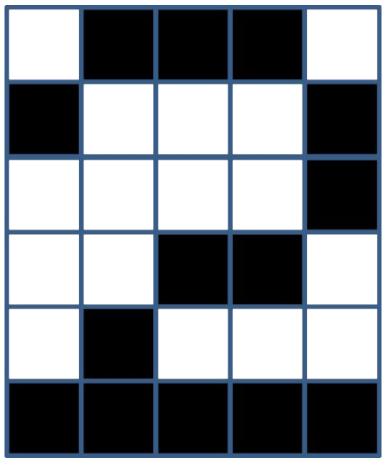
$$\theta < cos^{-1}\left(\frac{T}{|X|}\right)$$



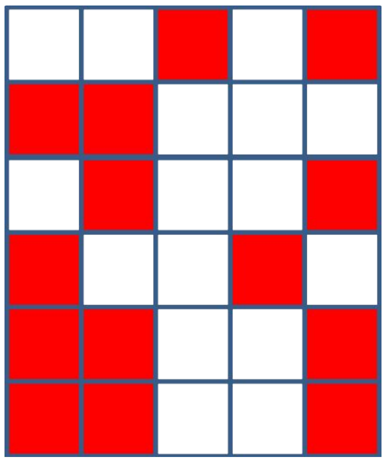

- The perceptron fires if the input is within a specified angle of the weight
- Neuron fires if the input vector is close enough to the weight vector.
  - If the input pattern matches the weight pattern closely enough

# The weight as a template

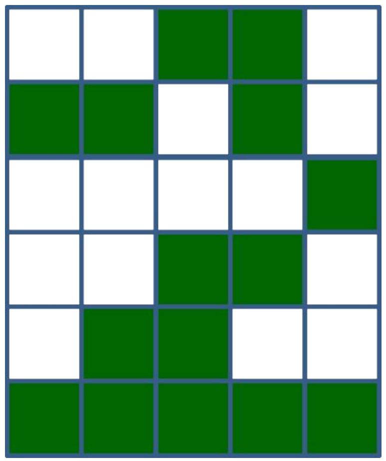
W



$$\begin{cases} 1 & \text{If } \Sigma w_i x_i \geq 0, \\ 0 & \text{otherwise} \end{cases}$$



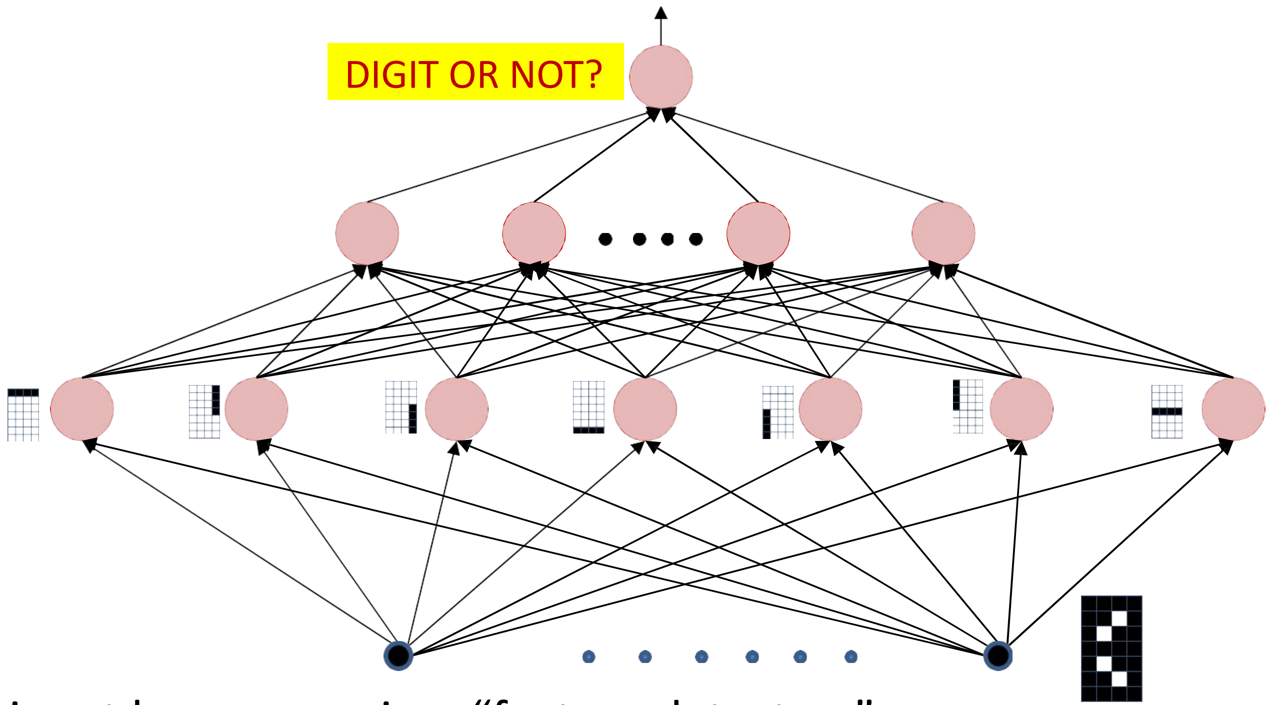
Correlation = 0.57



Correlation = 0.82

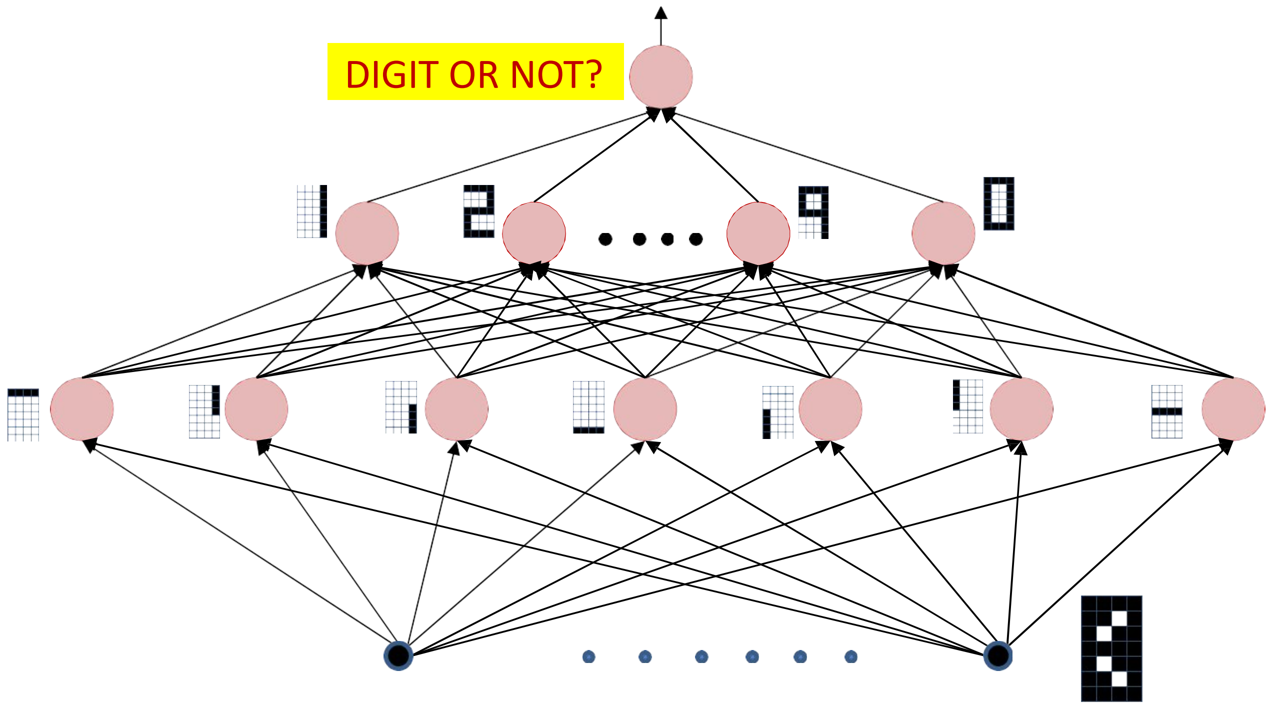
- If the *correlation* between the weight pattern and the inputs exceeds a threshold, fire
- The perceptron is a *correlation filter!*

# The MLP as a Boolean function over feature detectors




- The input layer comprises “feature detectors”
  - Detect if certain patterns have occurred in the input
- The network is a Boolean function over the feature detectors
- I.e. it is important for the *first* layer to capture relevant patterns

# The MLP as a cascade of feature detectors




- The network is a cascade of feature detectors
  - Higher level neurons compose complex templates from features represented by lower-level neurons

# Terminology ....

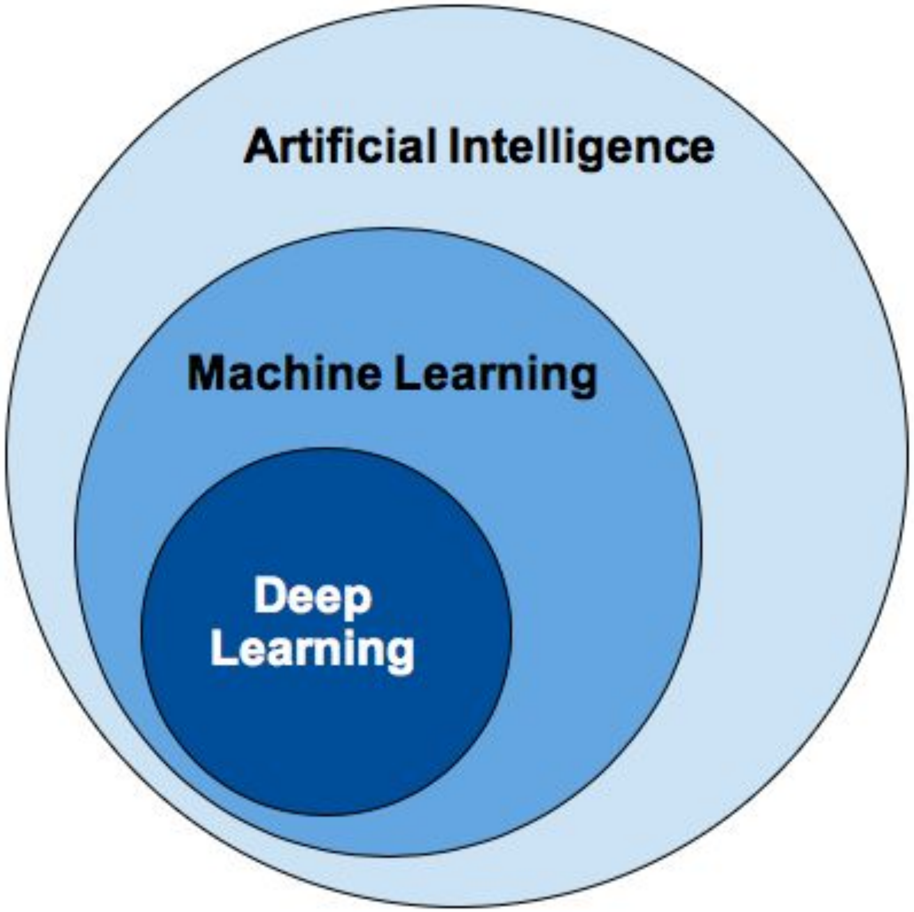



Image Credit: https://www.sumologic.com/blog/machine-learning-deep-learning/