MIDS W207
Applied Machine Learning

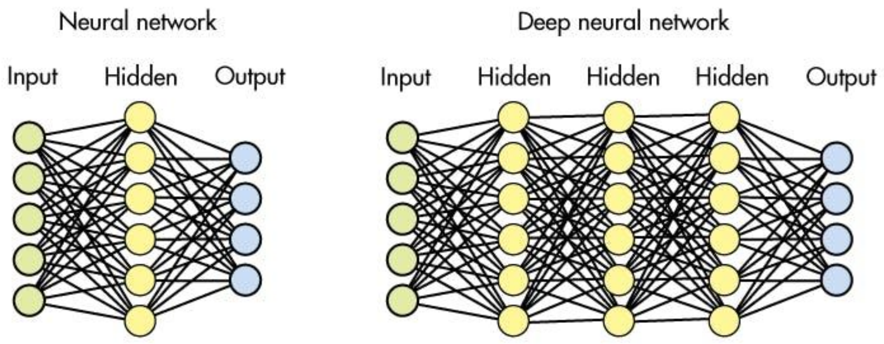
Week 6
Live Session Slides

Neural Networks

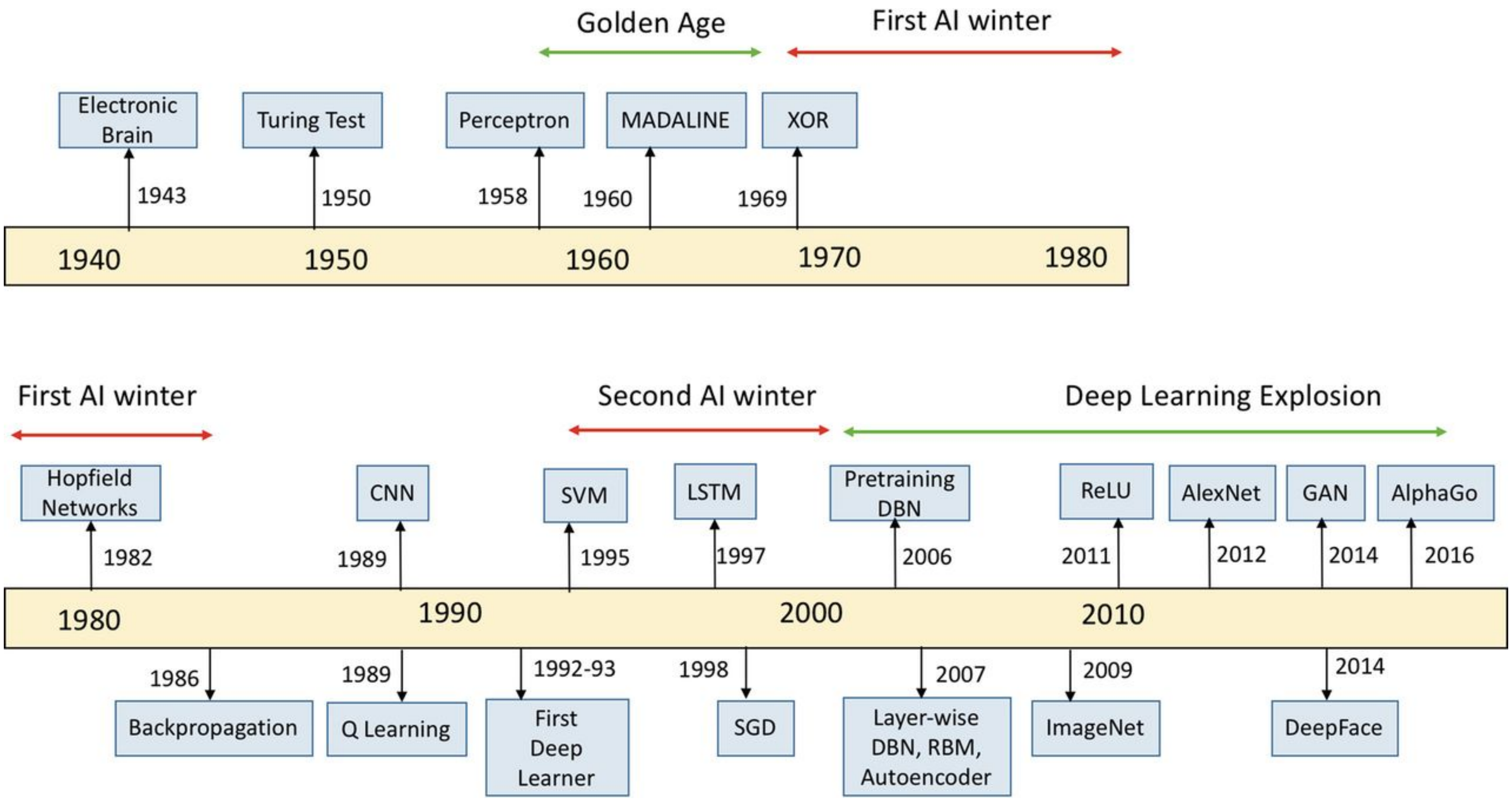
Deep learning algorithm structured similar to the organization of neurons in the brain

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

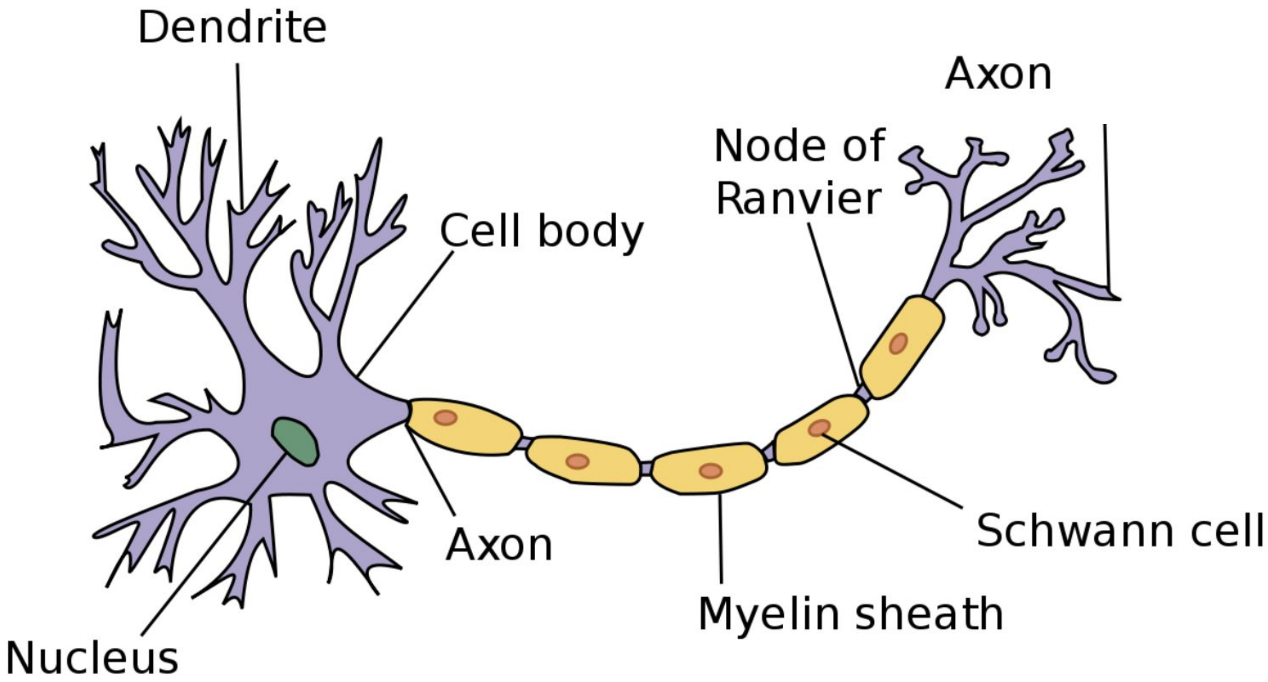
Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.



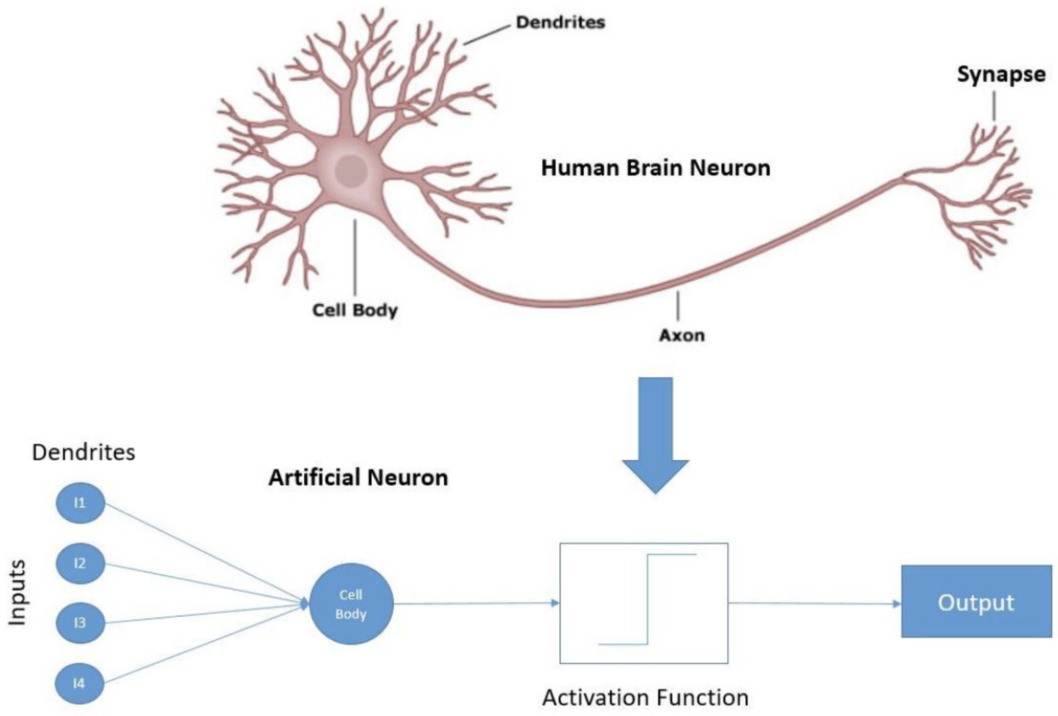
Neural Networks: History and Timeline



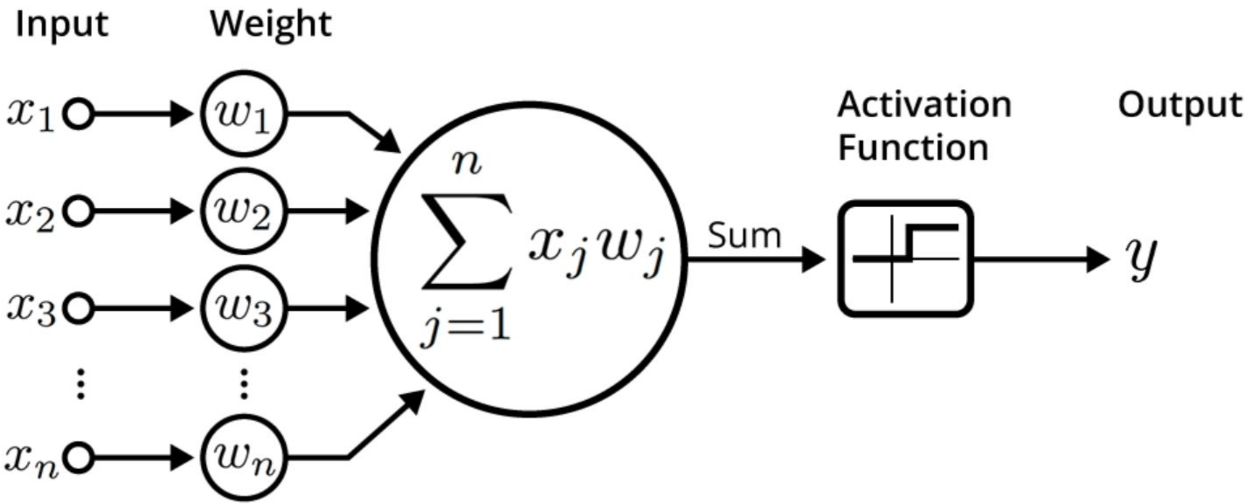
Neural Networks



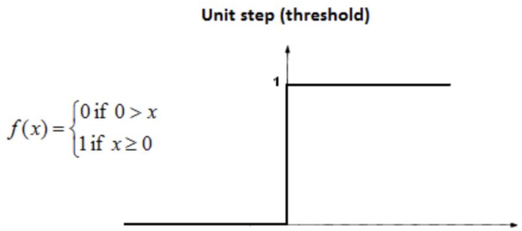
Neural Networks

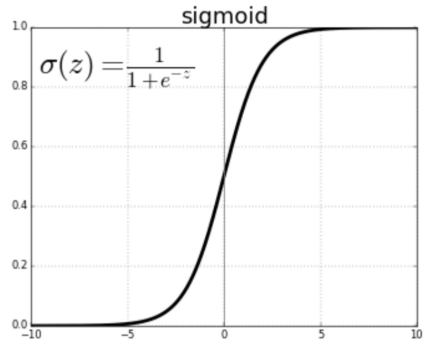


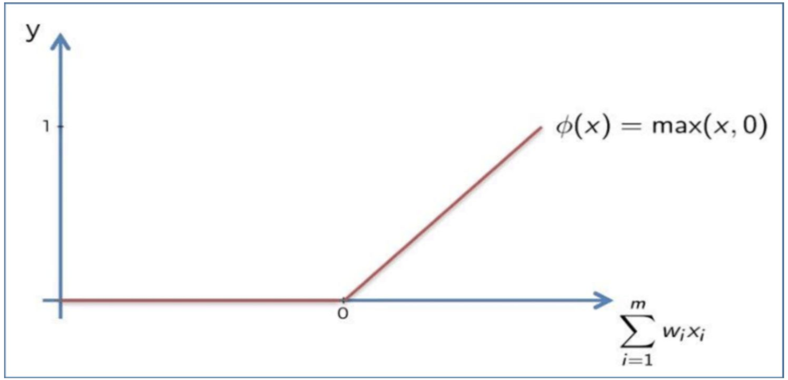
Neural Networks

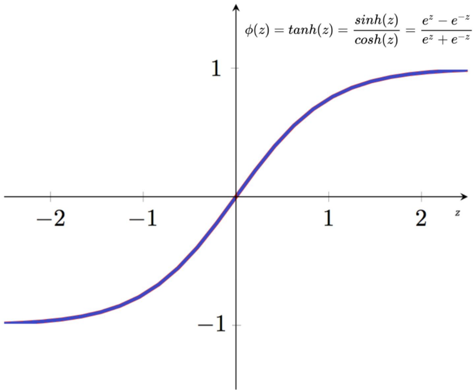


Neural Networks: Activation Functions

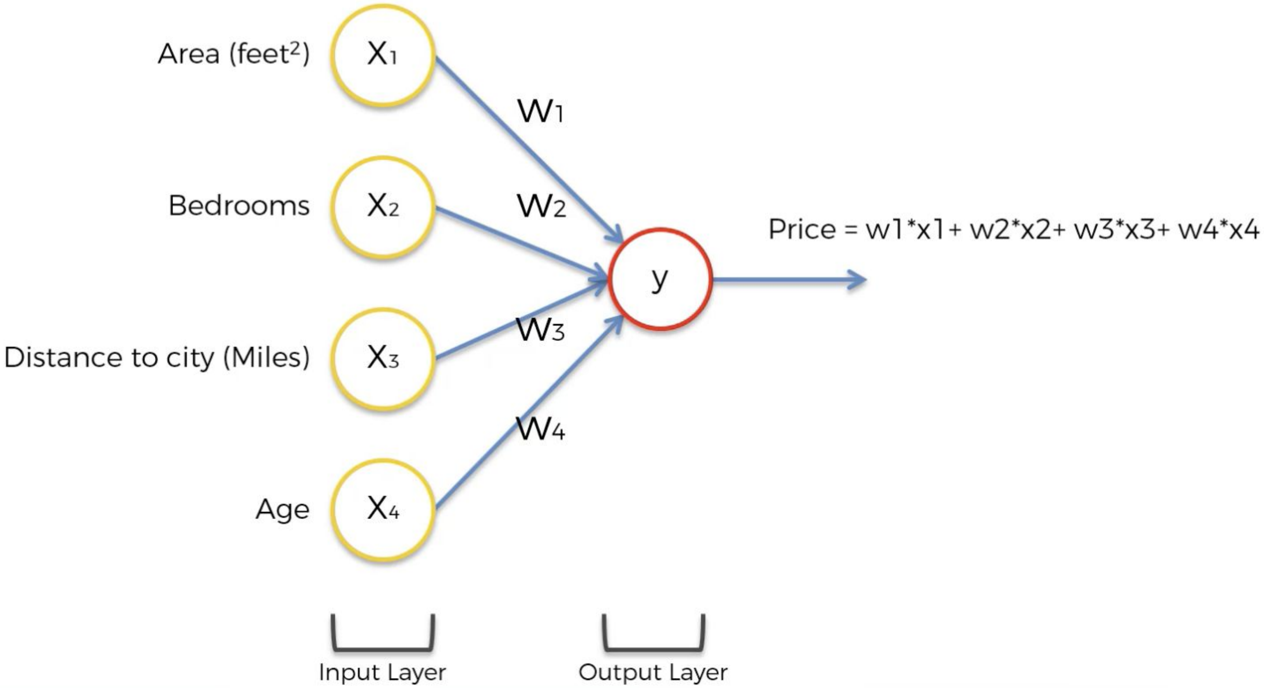




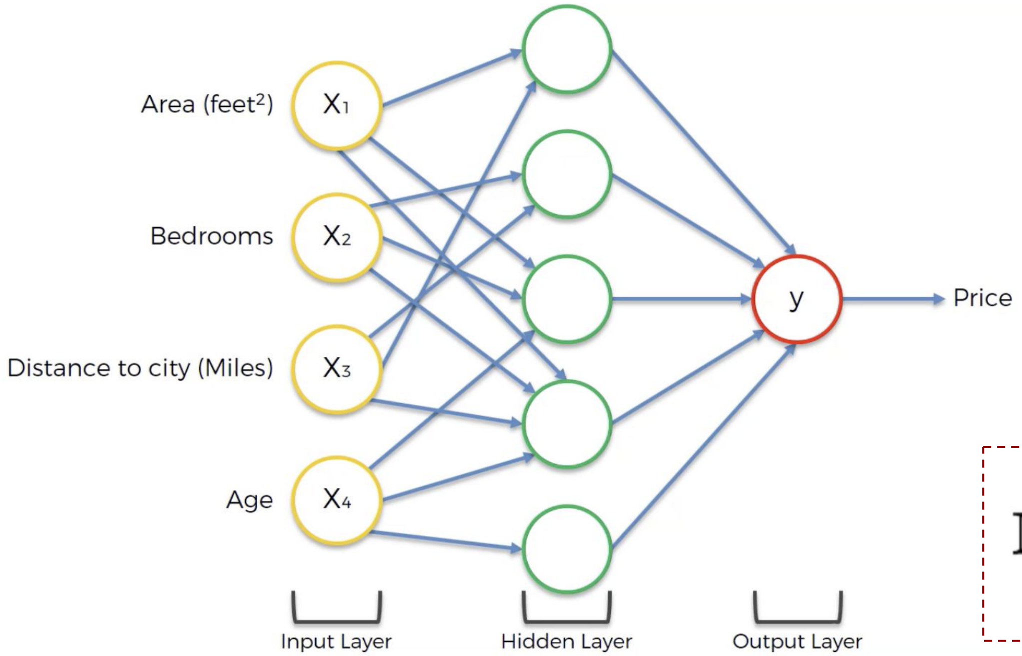




Neural Networks: Example (Property Valuation)



Neural Networks: Example (Property Valuation)

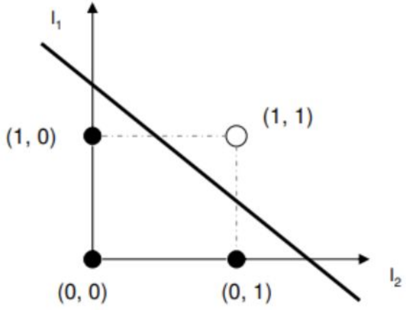


Cost Function

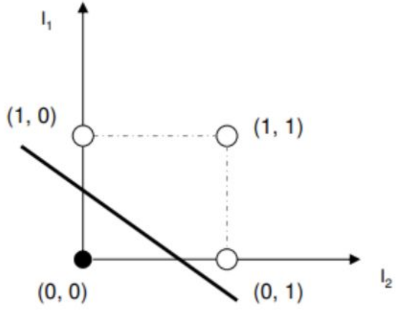
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Neural Networks: Perceptron

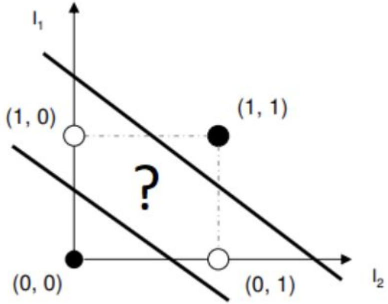
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0

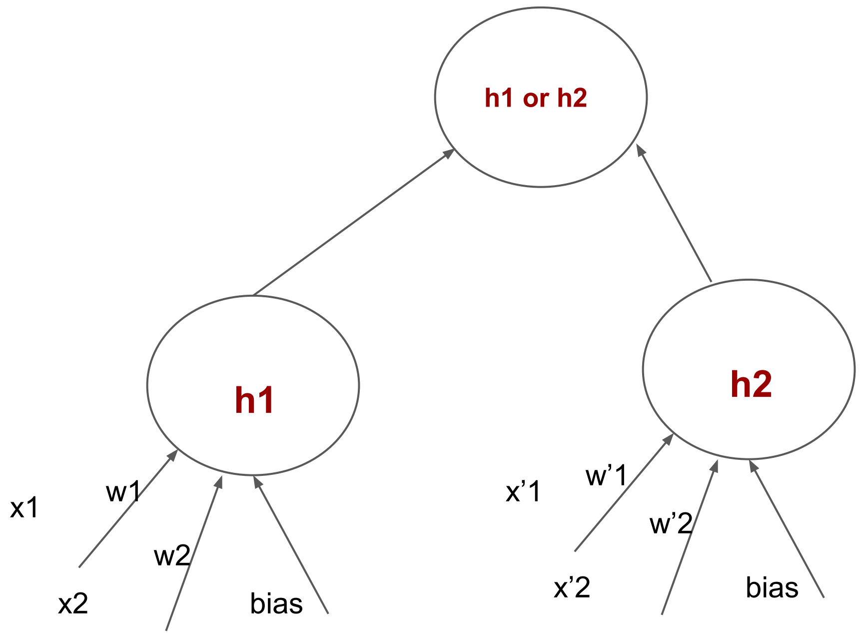


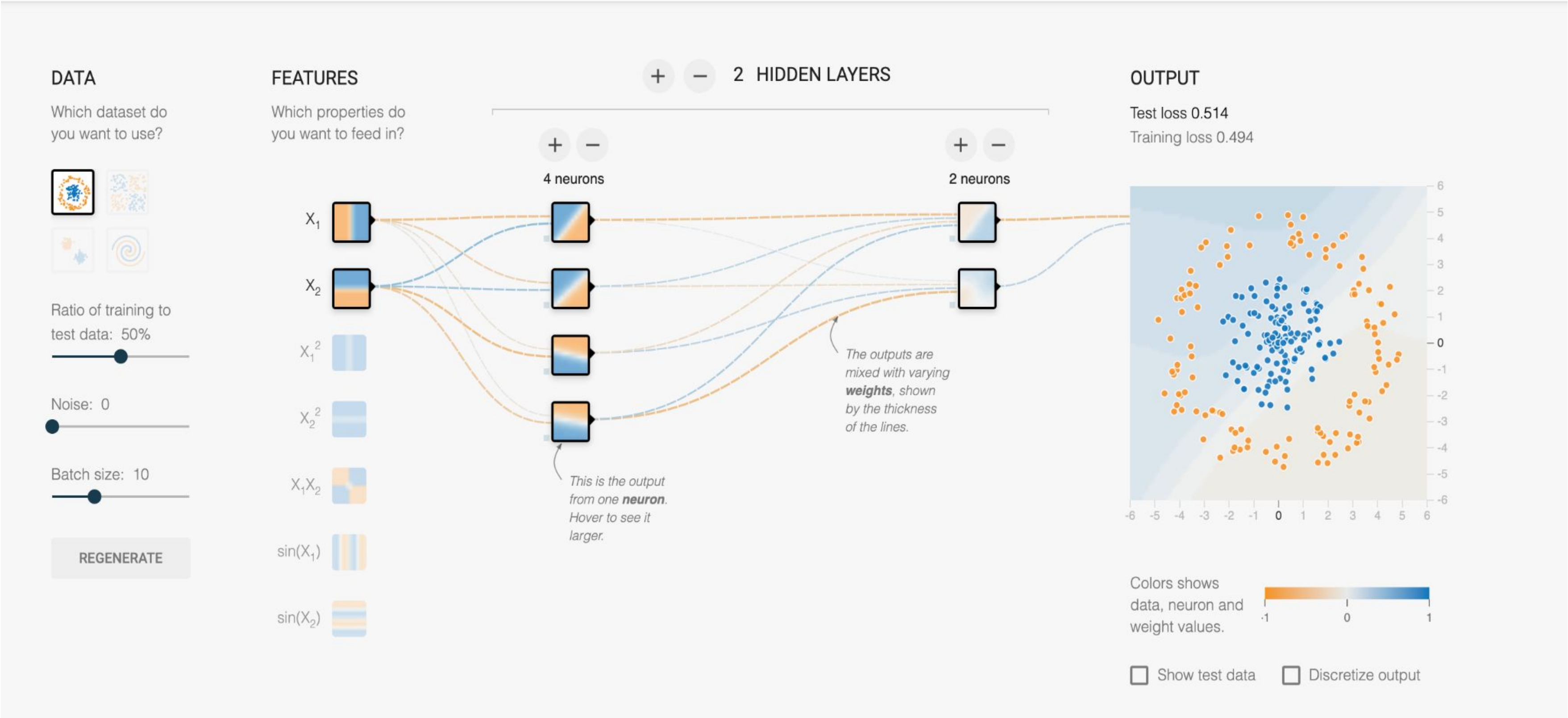
Neural Networks: Perceptron (XOR)

			h1	h2	
X1	**X2**	**Y**	**X1 AND ⌐X2**	**⌐X1 AND X2**	**h1 OR h2**
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	0	0

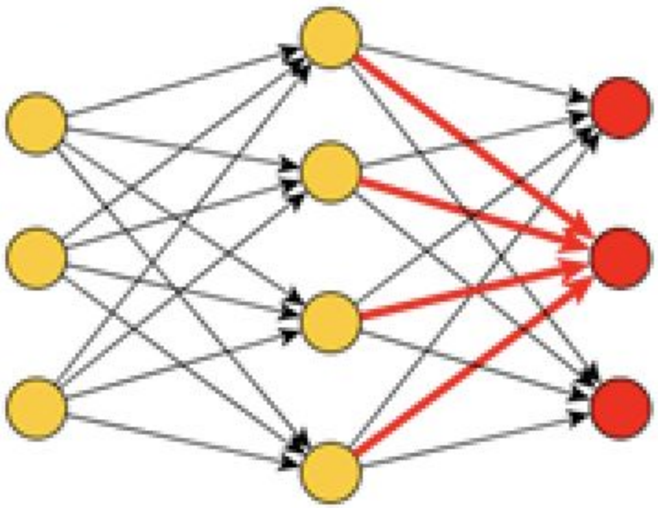
$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

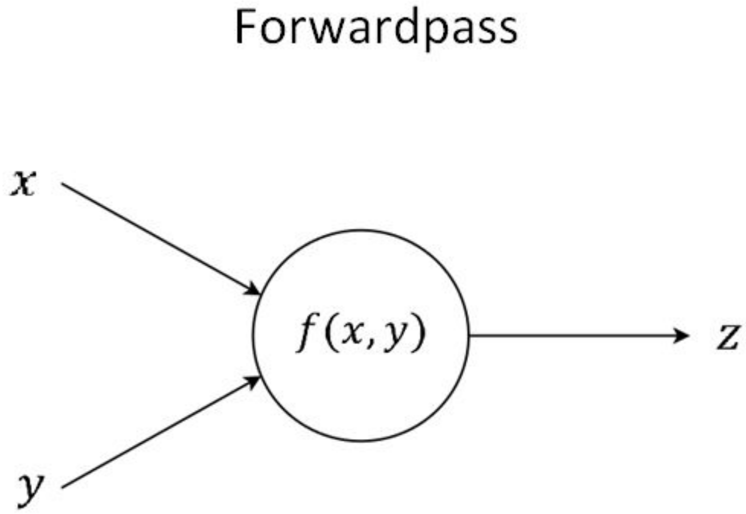
Neural Networks: Perceptron (XOR)




DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1^2
X2^2
X1X2
sin(X1)
sin(X2)
+ − 2 HIDDEN LAYERS
4 neurons
2 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.514
Training loss 0.494
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

Neural Networks: Forward Propagation





Chain Rule

Suppose you have a composite function:

$$h(x) = f(g(x))$$

and both *f* and *g* are differentiable functions.
Then the chain rule says that the derivative of *h* is the following product:

$$h'(x) = f'(g(x))g'(x)$$

The chain rule expressed with partial derivatives:

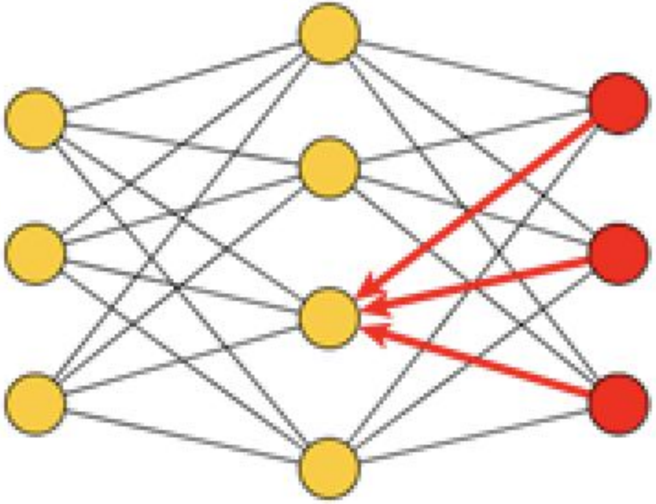
$$\frac{\partial}{\partial x} f(g(x)) = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$$

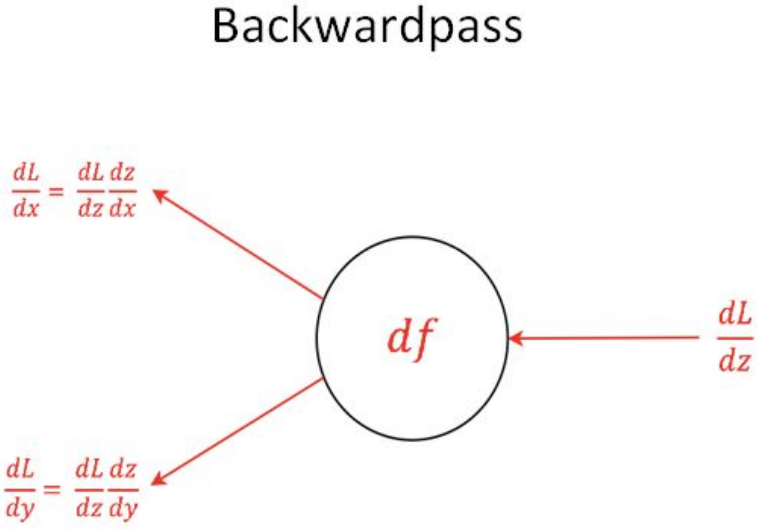
Instantaneous rate of change of *f* to *x*

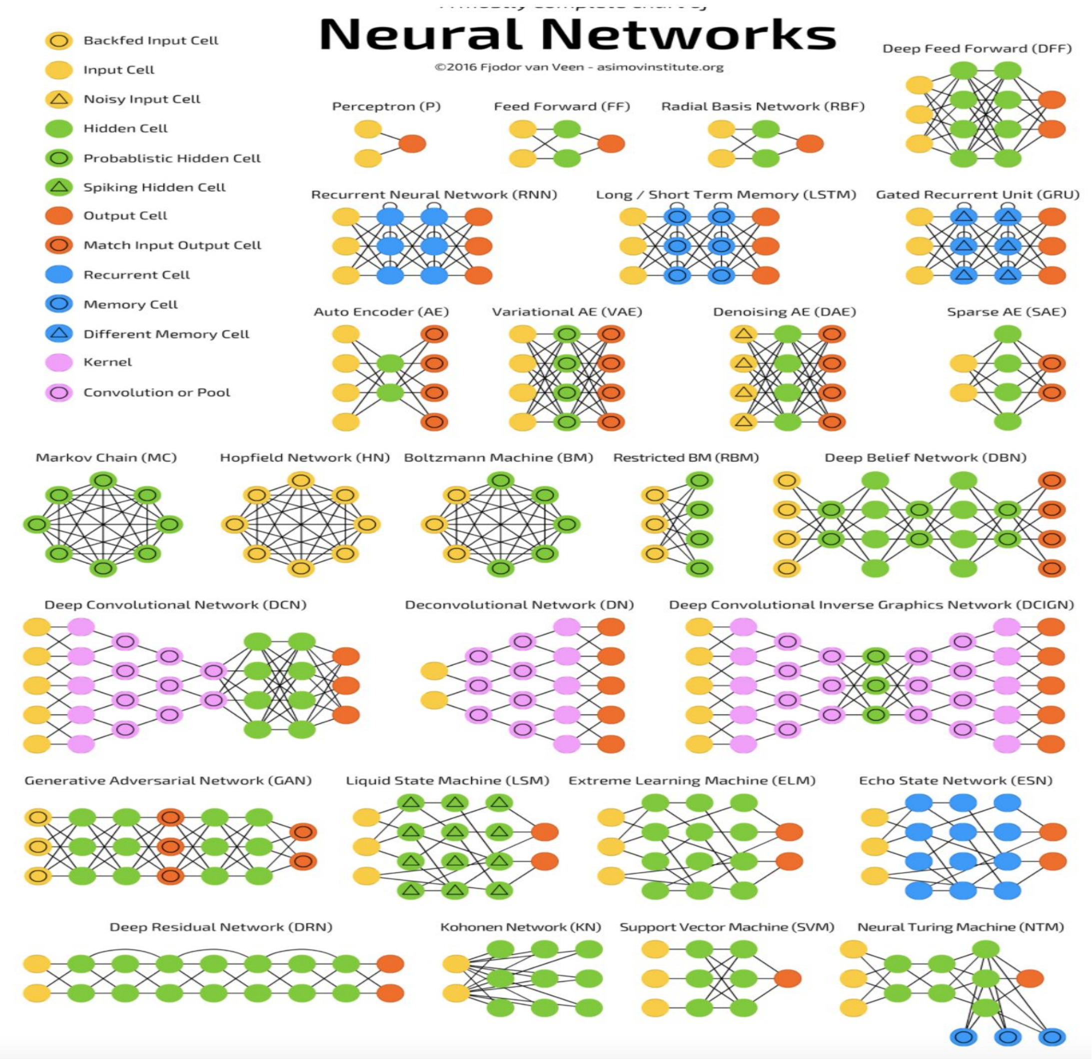
Instantaneous rate of change of *f* to *g*

Instantaneous rate of change of *g* to *x*

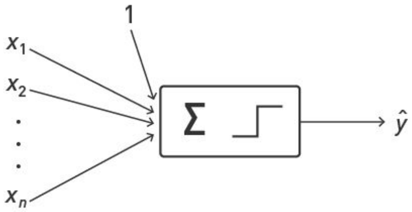
Neural Networks: Backpropagation

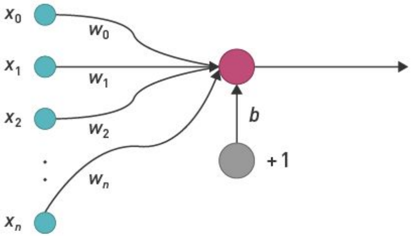





Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Neural Networks: Perceptron





- **Discriminative classifier:** learns decision boundary
- Perceptron fires if predicted value is positive

$$h(x_i) = b + \sum_{d=1}^{D} w_d x_{id}$$

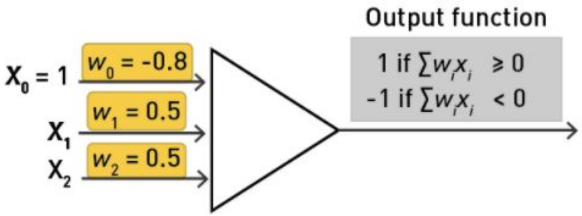
- *b* allows for nonzero threshold

$$\boxed{output = \begin{cases} 0 \ \ if \sum_j w_j x_j < threshold \\ 1 \ \ if \sum_j w_j x_j > threshold \end{cases}}$$

Neural Networks: Perceptron (AND)

A perceptron for AND:

x_1	x_2	$\Sigma w_1 x_1$	Output
1	1	0.2	1
1	0	–0.3	–1
0	1	–0.3	–1
0	0	–0.8	–1



- Two weights and intercept

$$h(x_i) = b + w_1 x_{i1} + w_2 x_{i2}$$

Neural Networks: Perceptron (XOR)

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$\rightarrow$ b ≤ 0

$\rightarrow$ b + w2 > 0

$\rightarrow$ b + w1 > 0

$\rightarrow$ b + w1 + w2 ≤ 0

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

Perceptron: Rosenblatt’s Algorithm

```
initialize weights randomly
while termination condition is not met:
      initialize Δw_j = 0
      for each training example (X_i, Y_i):
            compute predicted output Ŷ_i
            for each weight w_j:
                  Δw_j = Δw_j + η(Y_i − Ŷ_i)X_i
      for each weight w_j:
            w_j = w_j + Δw_j
```

Neural Networks: Gradient Descent

- Training rule (Rosenblatt)

$$\Delta w_j = \Delta w_j + \eta(Y_i - \hat{Y})X_i$$

- Gradient descent (stochastic)

$$\beta <- \beta + R(Y_i - \hat{Y}_i)X_i$$

- Key is the $\hat{Y}_i$.
  - Perceptron: $\hat{Y}_i$ is step function; either 0 or 1.
  - Gradient descent: $\hat{Y}_i$ is smooth function; continuous.
  - Gradient provides continuous surface