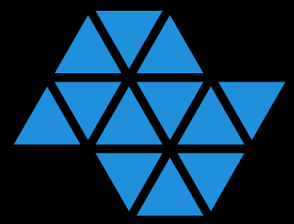


Neural Networks

Raphael Cóbe
raphael.cobe@sprace.org.br

Neural Networks

Neural Networks

- Neurons as structural constituents of the brain [Ramón y Cajál, 1911]
- Five to six orders of magnitude slower than silicon logic gates
- In a silicon chip happen in the nanosecond (on chip) vs millisecond range (neural events)
- A truly staggering number of neurons (nerve cells) with massive interconnections between them.

Neural Networks

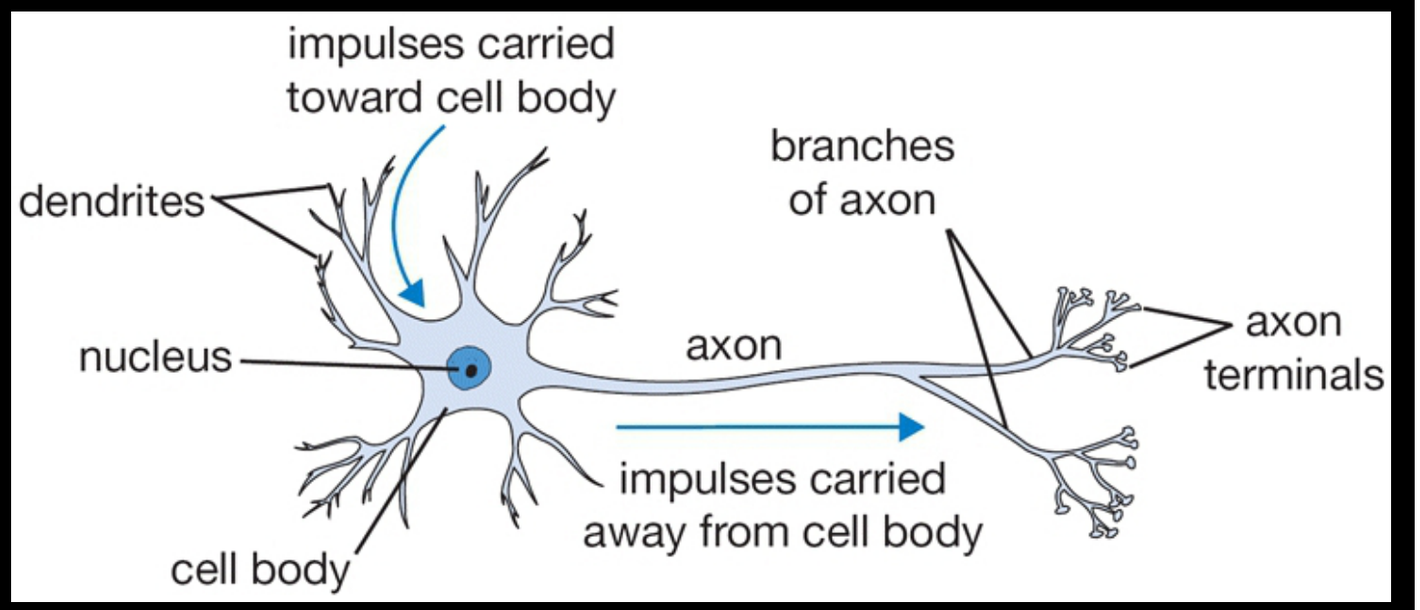
- Receive input from other units and decides whether or not to fire.

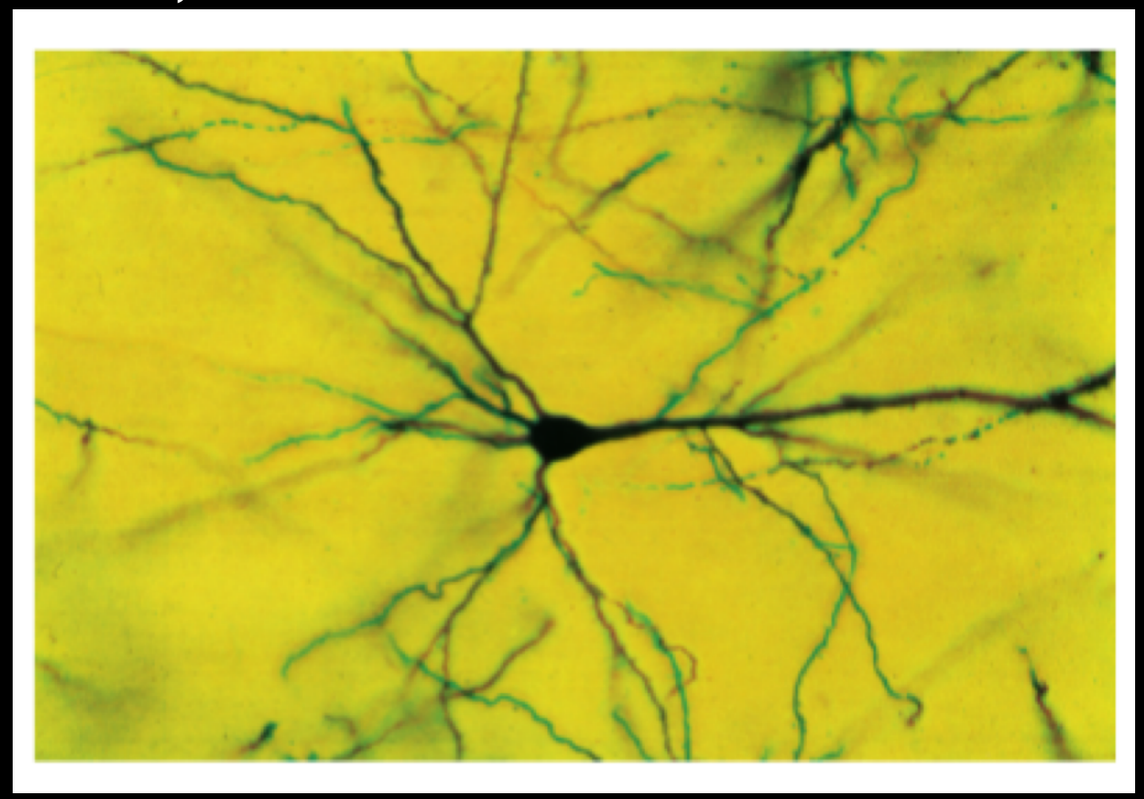
- approximately 10 billion neurons in the human cortex, and 60 trillion synapses or connections [Shepherd and Koch, 1990]

- Energy efficiency of the brain is approximately 10^–16 joules per operation per second against ~ 10 ^ –8

Neural Networks

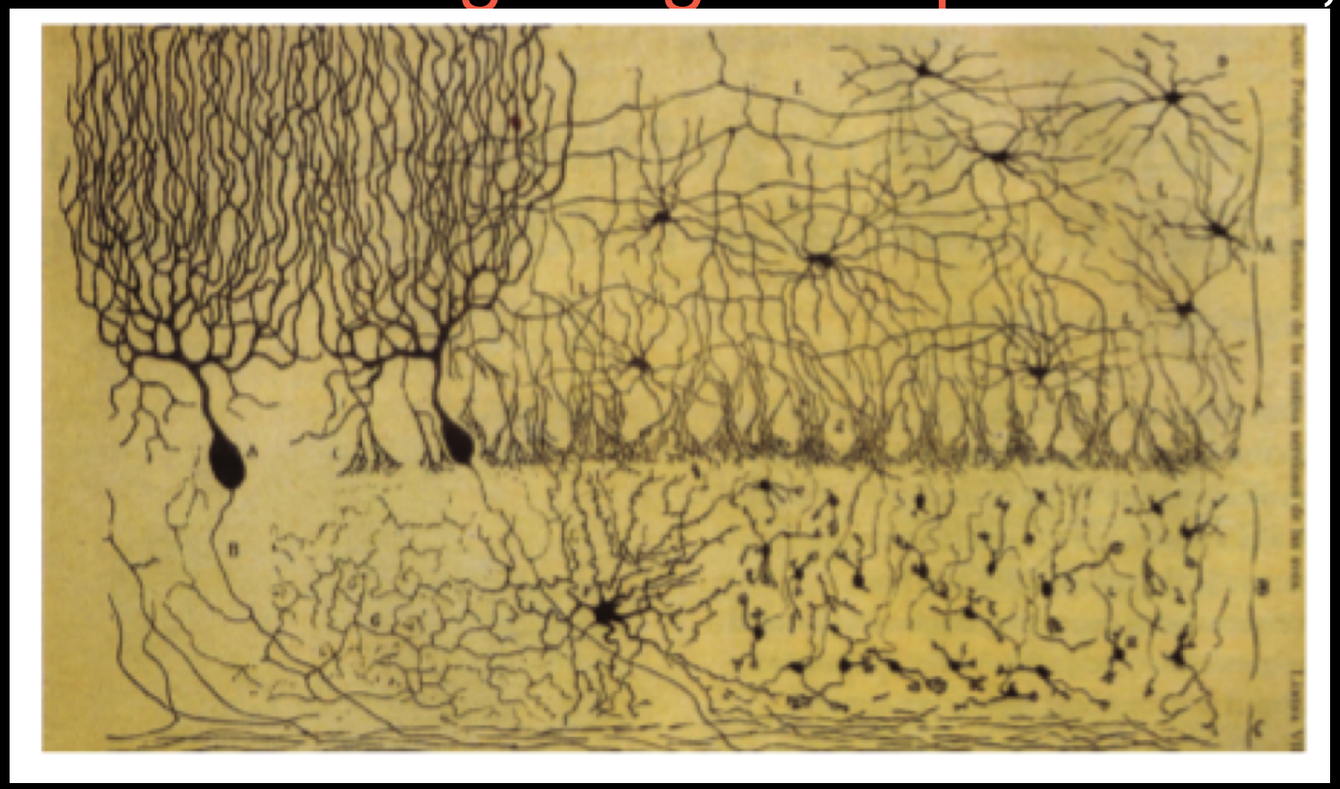
- input signals from its dendrites;
- output signals along its (single) axon;





Neural Networks

- It appears that one reason why the human brain is so powerful is the sheer complexity of connections between neurons;
- The brain exhibits huge degree of parallelism;

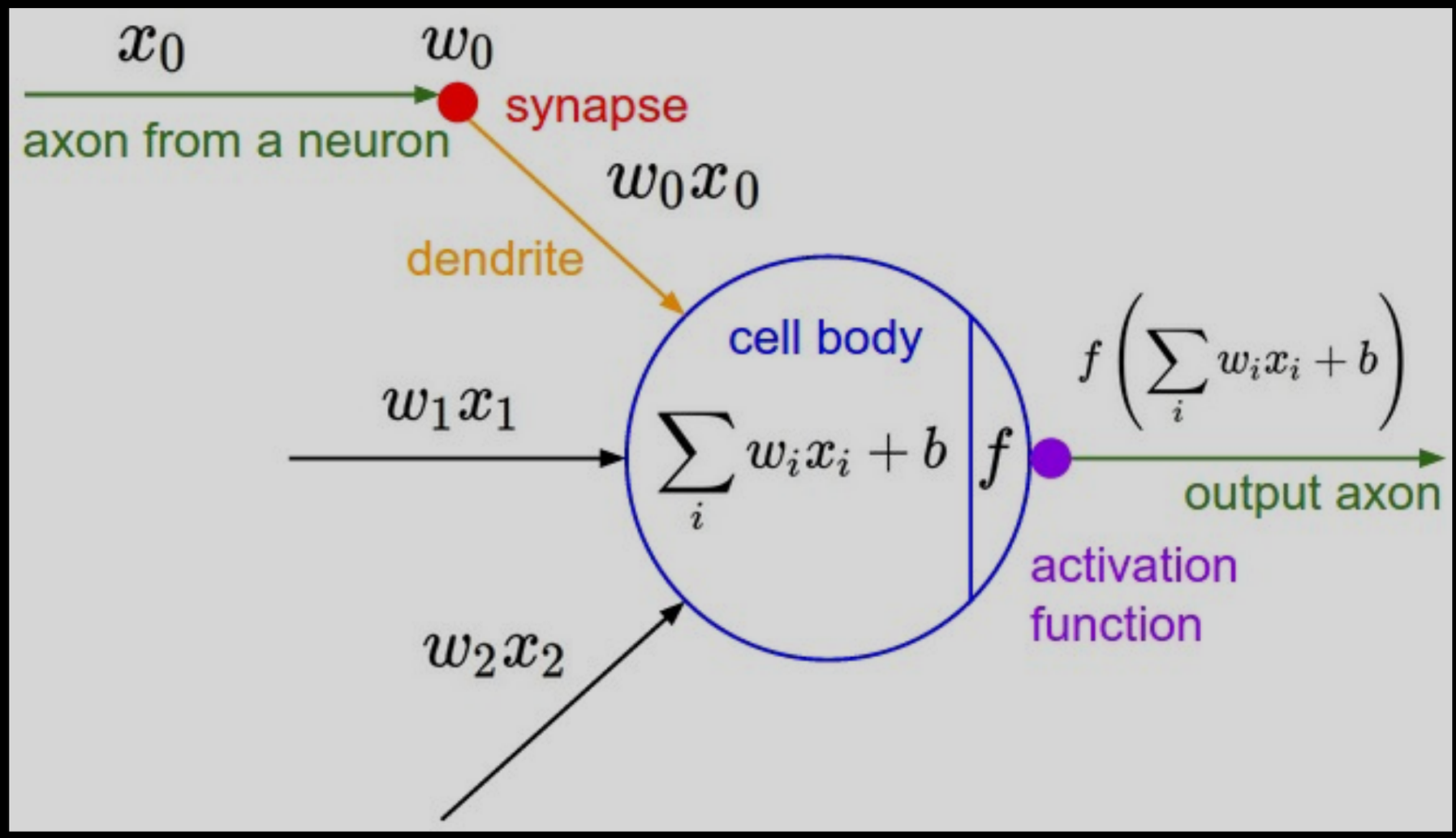


Sesquicentennial of the birthday of Santiago Ramon y Cajal, the father of modern neuroscience.
Trends of Neuroscience, 25(9), 481–484 (2002).

Neural Networks

- control the influence from one neuron on another:
  - excitatory when weight is positive; or
  - inhibitory when weight is negative;
- nucleus is responsible for summing the incoming signals;
  - if the sum is above some threshold, then fire!

Neural Networks



Images from Stanford CS231

Neural Networks

- model each part of the neuron and interactions;
- interact multiplicatively (e.g. $w_0 x_0$) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g. w_0);
- learn synapses strengths;

Neural Networks

- Function approximation machines;
- $y = f^*(x)$
  - *maps* x *input to a category* y
- $y = f(x; w)$
  - *learn the value of the parameters* w

(Artificial) Neural Networks

- can be seen as a directed graph with units (or neurons) situated at the vertices;

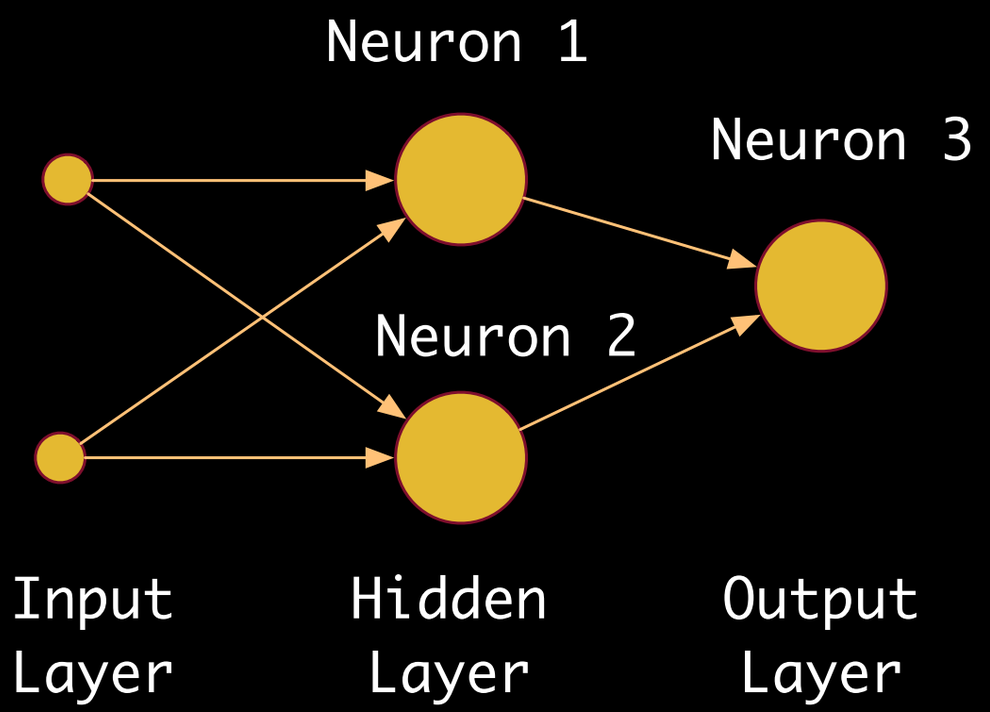
- Some are input units
    - receive signal from the outside world;

- The remaining are named computation units;

- Each unit produces an output
    - transmitted to other units along the arcs of the directed graph;

Neural Networks

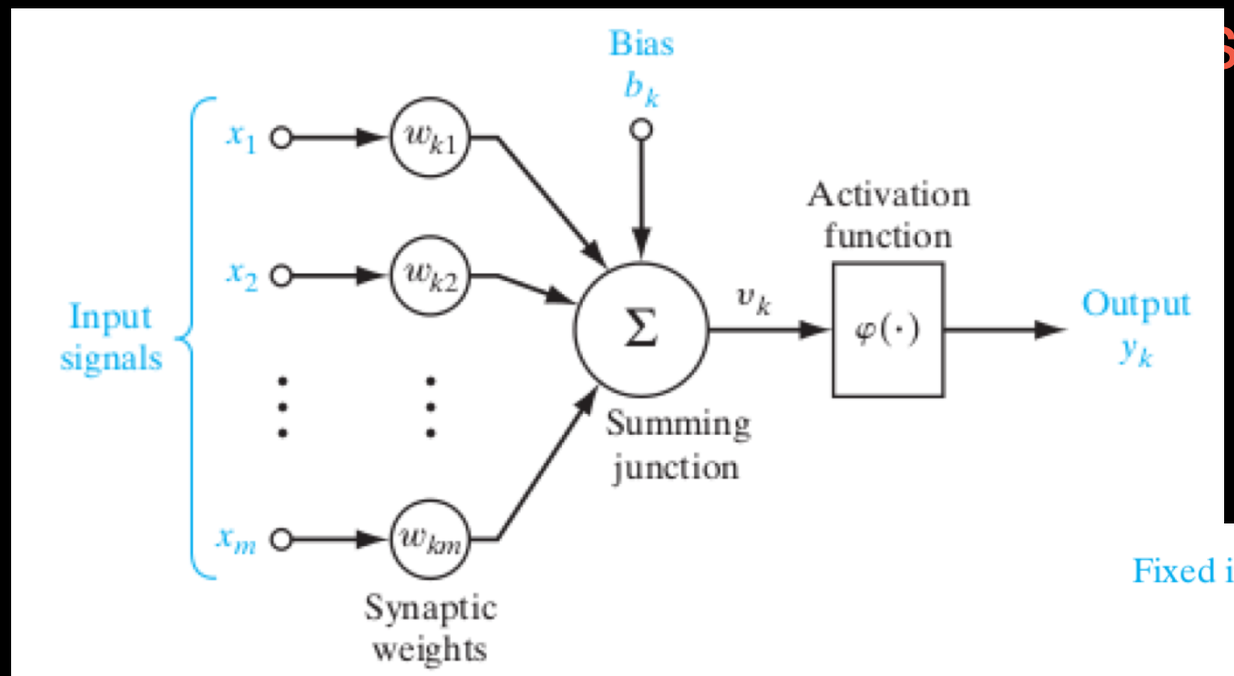
- Input, Output, and Hidden layers;
- Hidden as in "not defined by the output";
- Approximate $y = f(x; w)$ to $y = f^*(x)$ (training)

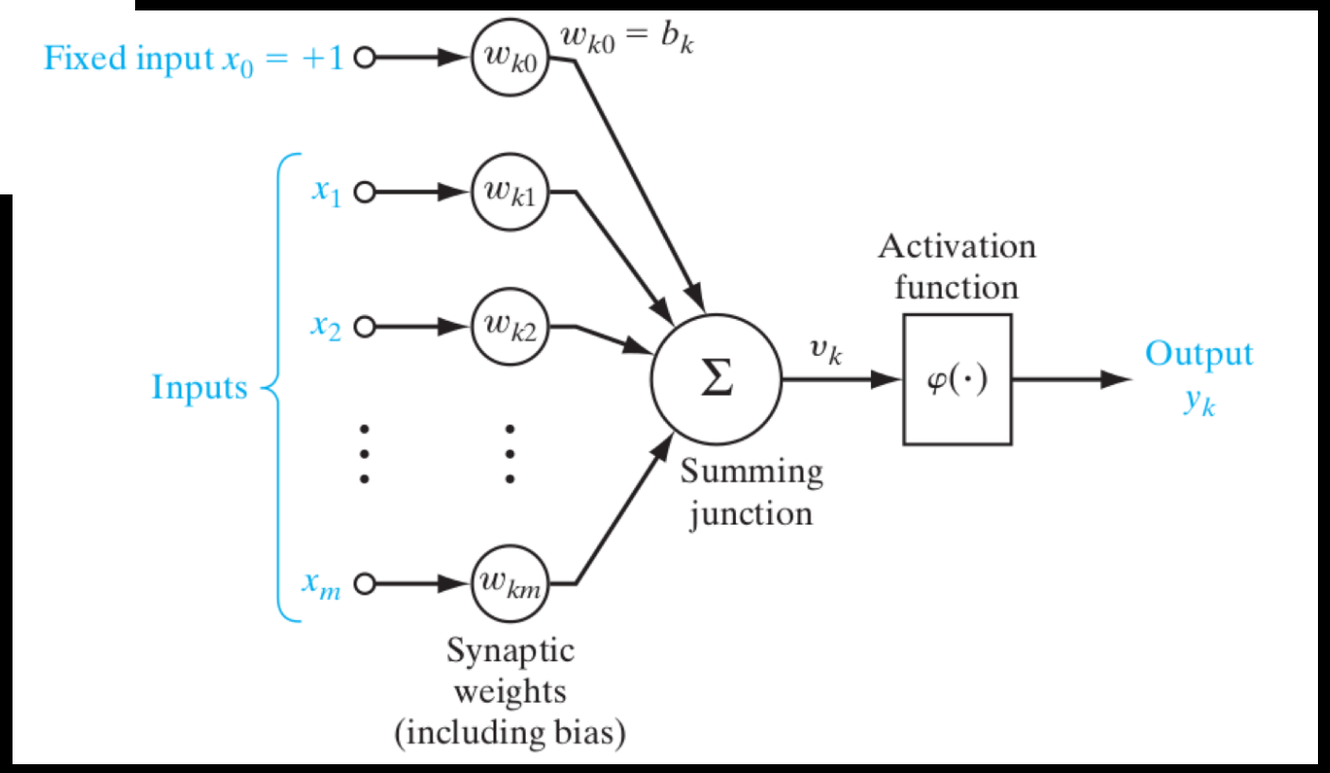


Neural Networks

- Activation Function:
    - Describes whether or not the neuron fires, i.e., if it forwards its value for the next neuron layer;
    - multiply the input by its weights, add the bias and apply the non-linearity;
    - Sigmoid, Hyperbolic Tangent, Rectified Linear Unit;

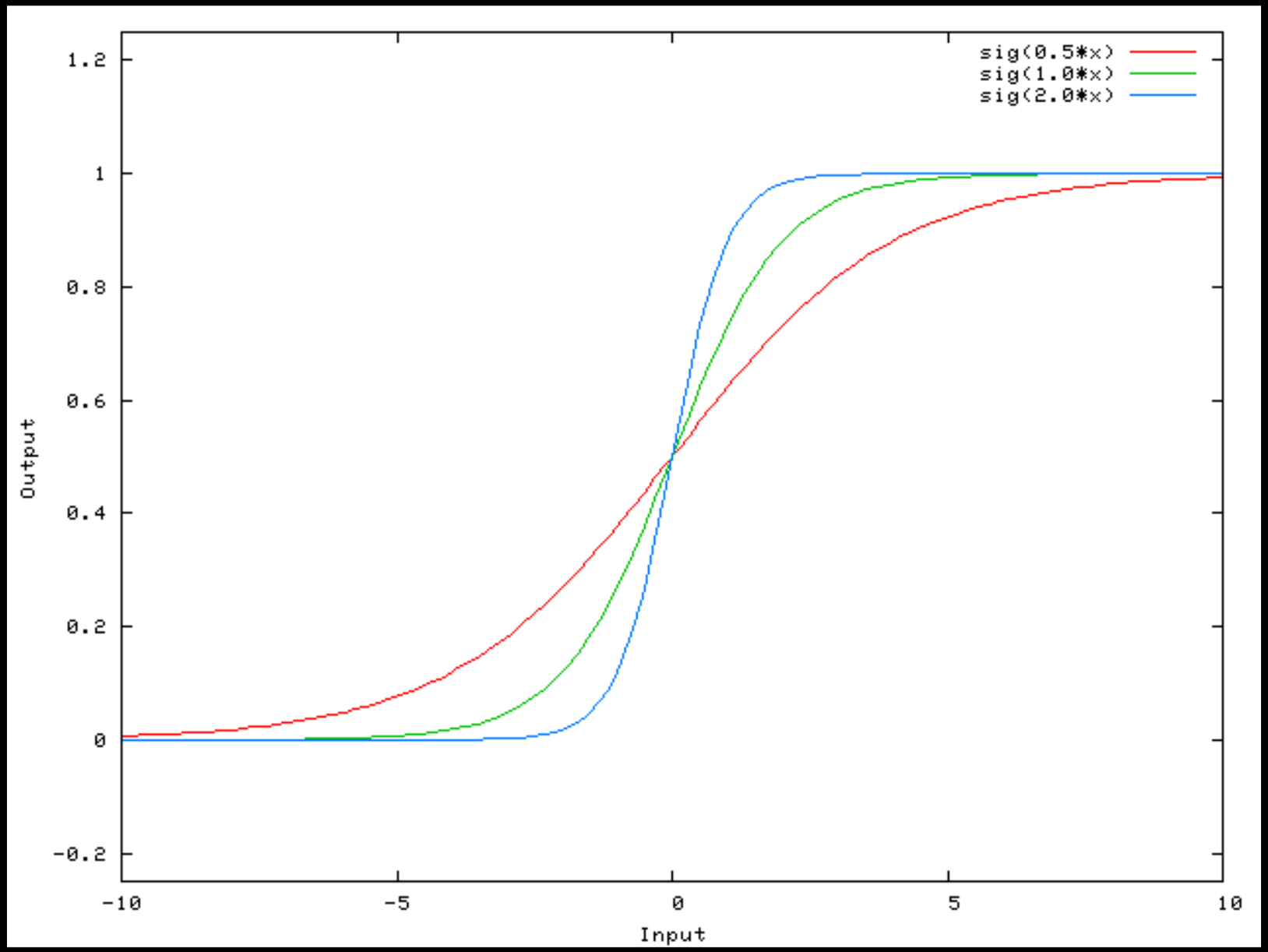
Neural Networks





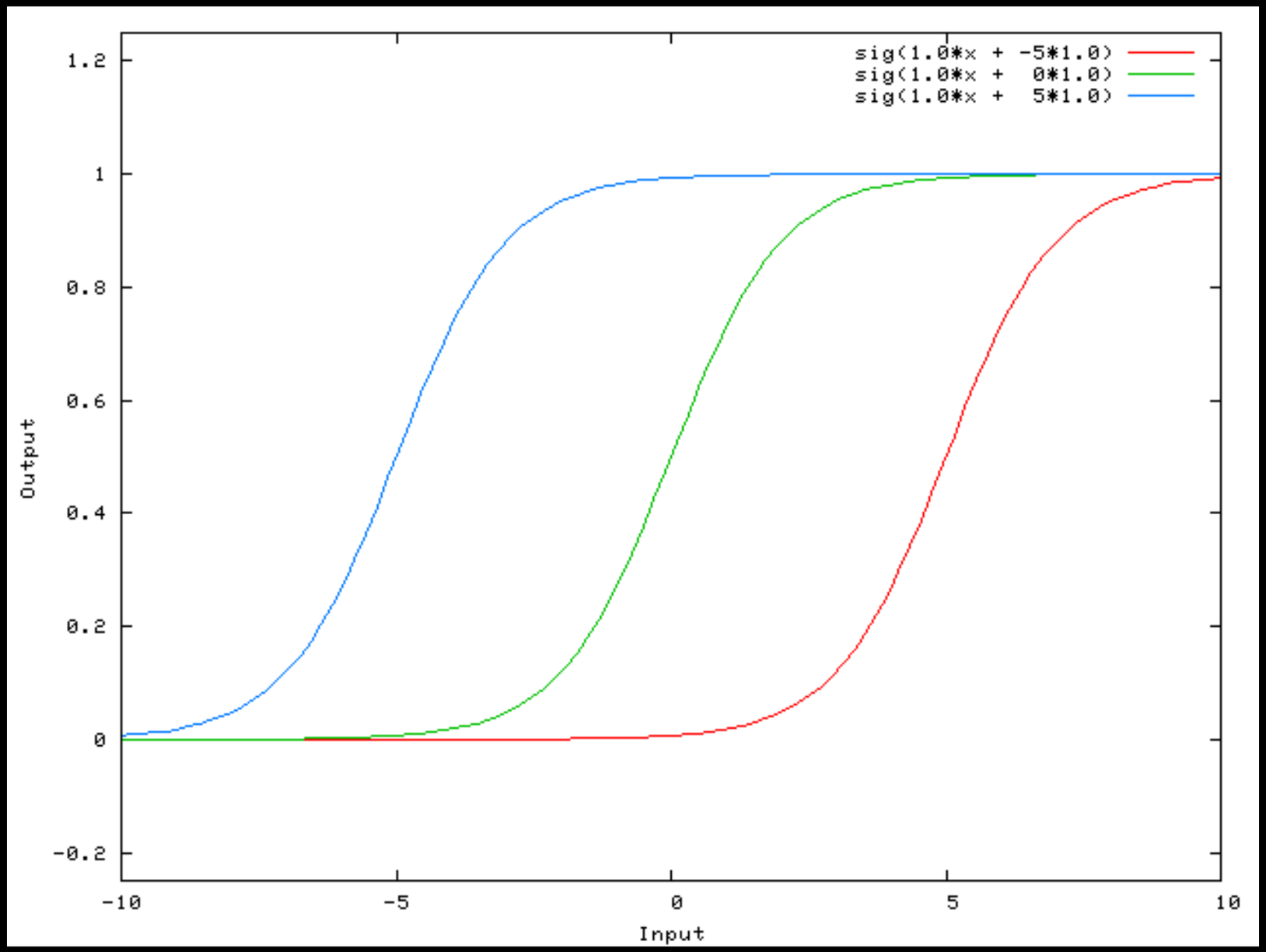
Neural Networks

- The Artificial Neuron (the bias)



Neural Networks

- The Artificial Neuron (the bias)



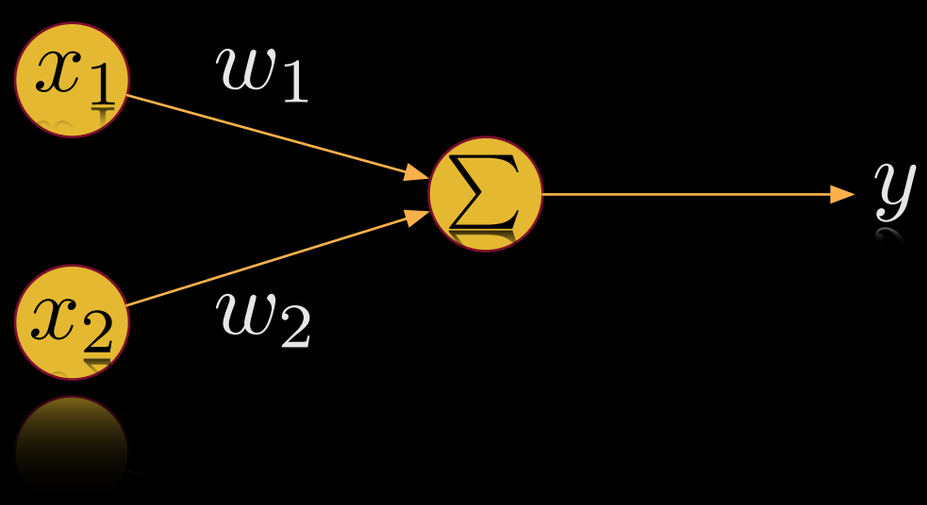
Perceptron

Perceptron

- In 1958, Frank Rosenblatt proposed an algorithm for training the perceptron.
- Simplest form of Neural Network;
- One unique neuron;
- Adjustable Synaptic weights;
- Simple activation function;

Perceptron

- Classifies inputs into two classes, with neuron output of either -1 or 1;



Perceptron

- Where ϕ is the activation function and J_1 the number of inputs

- Always converge on linearly separable classes;

$$net = \sum_{i=1}^{J_1} w_i x_i - \theta = w^T x - \theta$$

$$y = \phi(net)$$

Perceptron

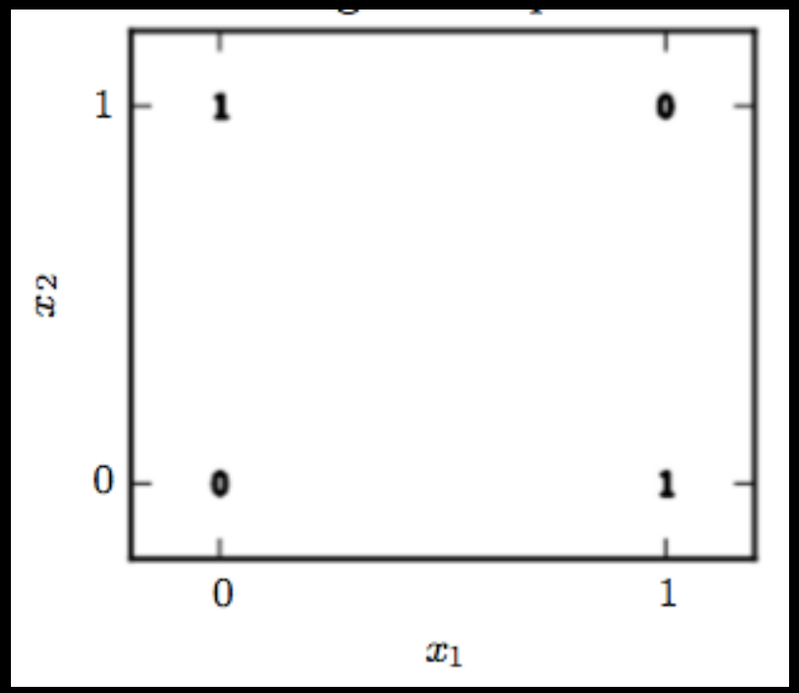
- Training Procedure:
- η is the learning rate;
- finds a linear function that separates the classes;

$$e_p = y_p - \hat{y}_p$$

$$w_i(t+1) = w_i(t) + \eta x_{p,i} e_i$$

Perceptron

- Can’t do:
  - separate non linear classes;
- XOR function:



Feedforward networks

Feedforward Networks

- typically represented by composing many different functions:

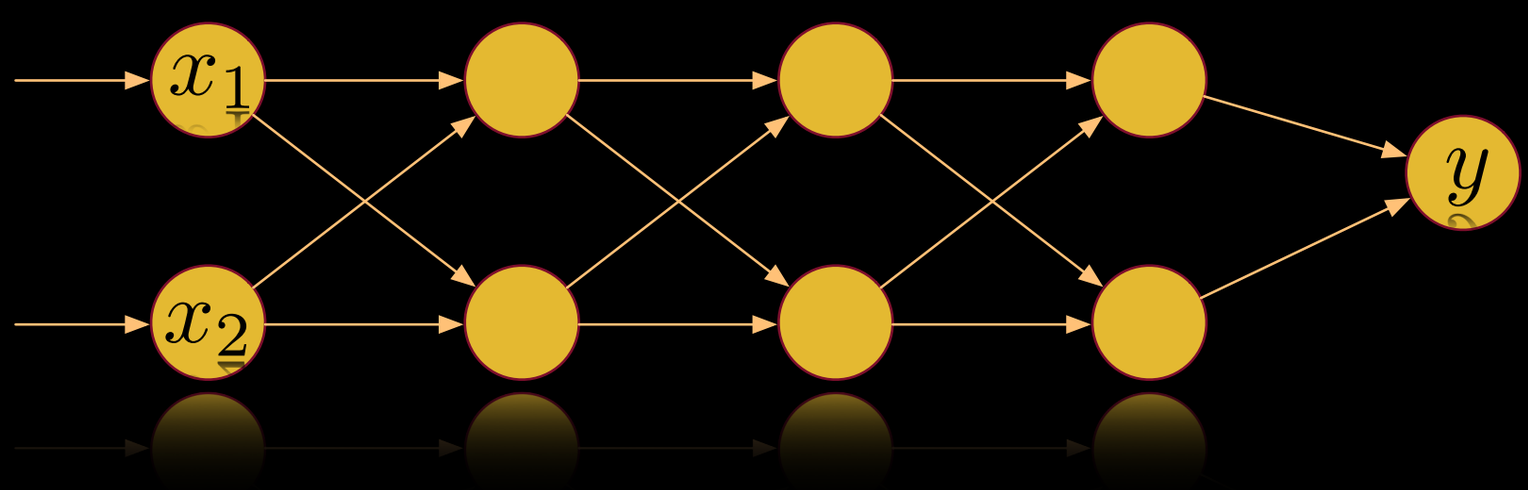
$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$$

- the **depth of the network - the deep in deep learning! (-;**

Feedforward Networks

- Information flows from x , through f computations and finally to y

- No feedback!



Feedforward Networks

- Train using a back-propagation algorithm from 1969;
    - fixes the weights in an output-to-input direction;
- each level plays a specific role in the classification;
- detect the features in the input patterns;

Feedforward Networks

- The output of a Feedforward Network:

$$\hat{y}_p = o_p^{(M)},\ o_p^1 = x_p$$

- The output of the m layer:

$$net_p^{(m)} = [W^{(m-1)}]^T o_p^{(m-1)} + \theta^{(m)}$$

$$o_p^{(m)} = \phi^{(m)}(net_p^{(m)})$$

Feedforward Network

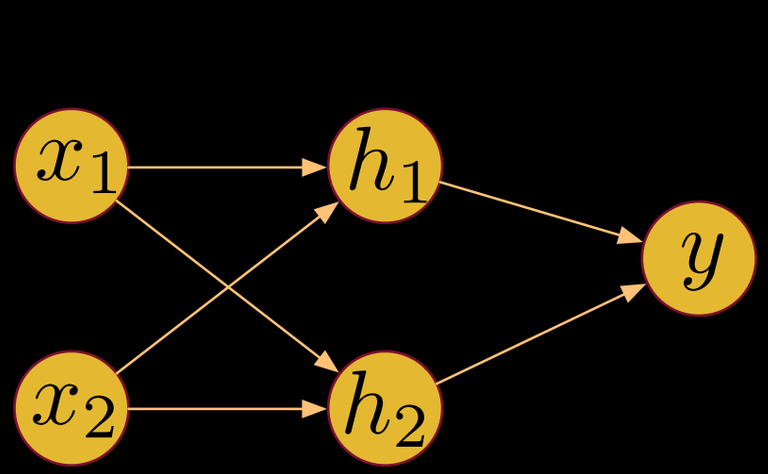
- suppose we want to learn the behavior of the binary XOR operator:
- the input will be a pair of signals with value either 0 or 1.
- The output should be classified also into 0 or 1;

Feedforward Network

- The dataset:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

The network:



Feedforward Network

- The architecture: 2 layer network (one hidden and one for output)
- Rectified linear Unit as activation function, i.e.

$$\phi(net_p^{(1)}) = max(net_p^{(1)}, 0)$$

- Why not a linear function?

Feedforward Network

- The full expanded solution:

$$y = W_2^T(max(0, W_1^T x + \theta_1)) + \theta_2$$

Feedforward Network

- Running the example:

$$W_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \theta_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \qquad X = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \qquad \theta_2 = 0$$

Feedforward Network

- Running the example:

$$net^{(1)} = W_1^T X + \theta_1 = \begin{bmatrix} 0 & 1 & 1 & 2 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

$$\phi(net^{(1)}) = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Feedforward Network

- Running the example:

$$o^{(2)} = net^{(2)} = W_2^T o^{(1)} + \theta_2 = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

$$o = \phi(net^{(2)})$$

Feedforward Networks

- most popular learning rule for performing supervised learning tasks;

- Not only used in Feedforward learning;

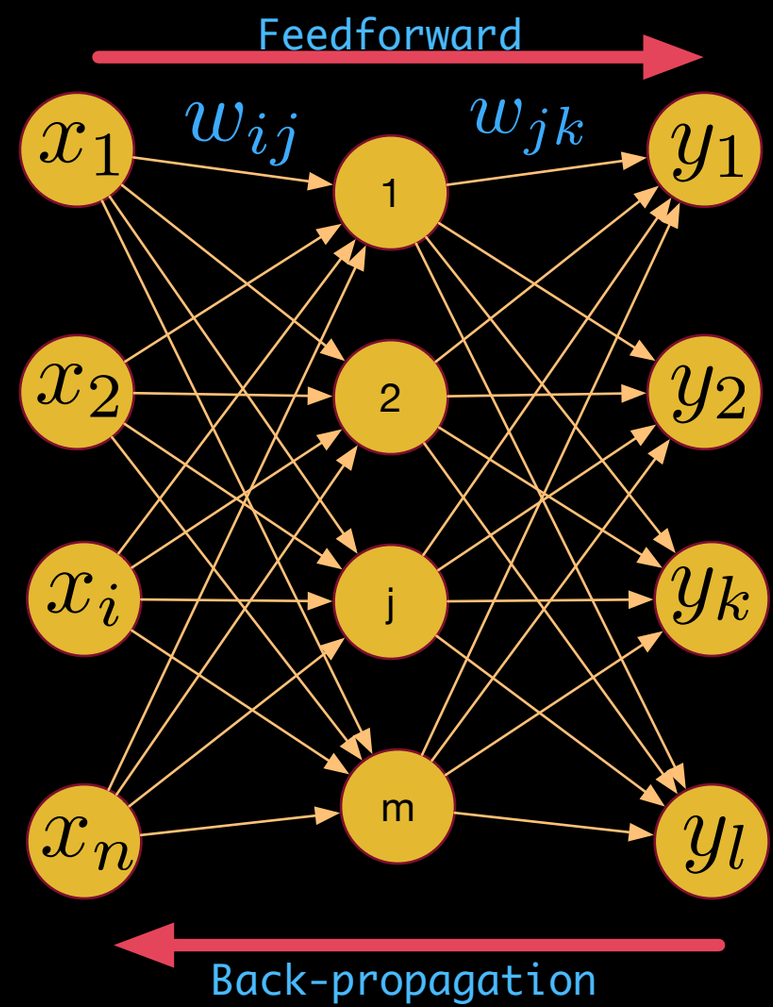
- propagates backward the error between the signal and the network output through the network;

- continuous, nonlinear, differentiable activation function

  - sigmoid functions, hyperbolic tangent;

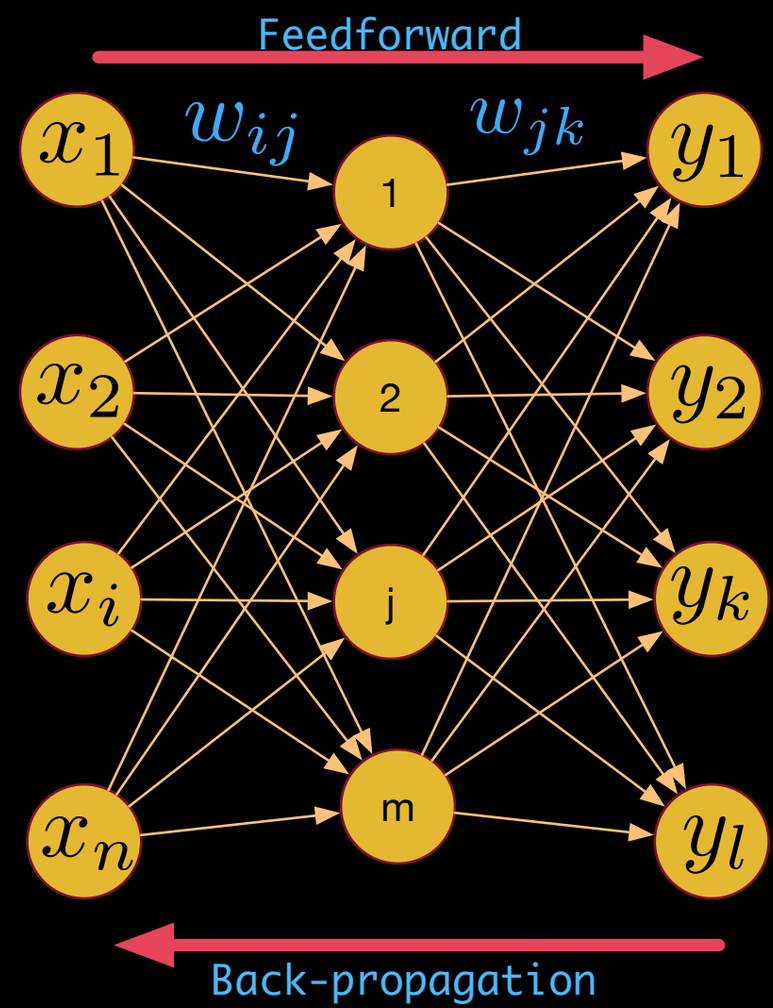
Feedforward Networks

- Indexes i,j,k refer to neurons in input, hidden and output layers;



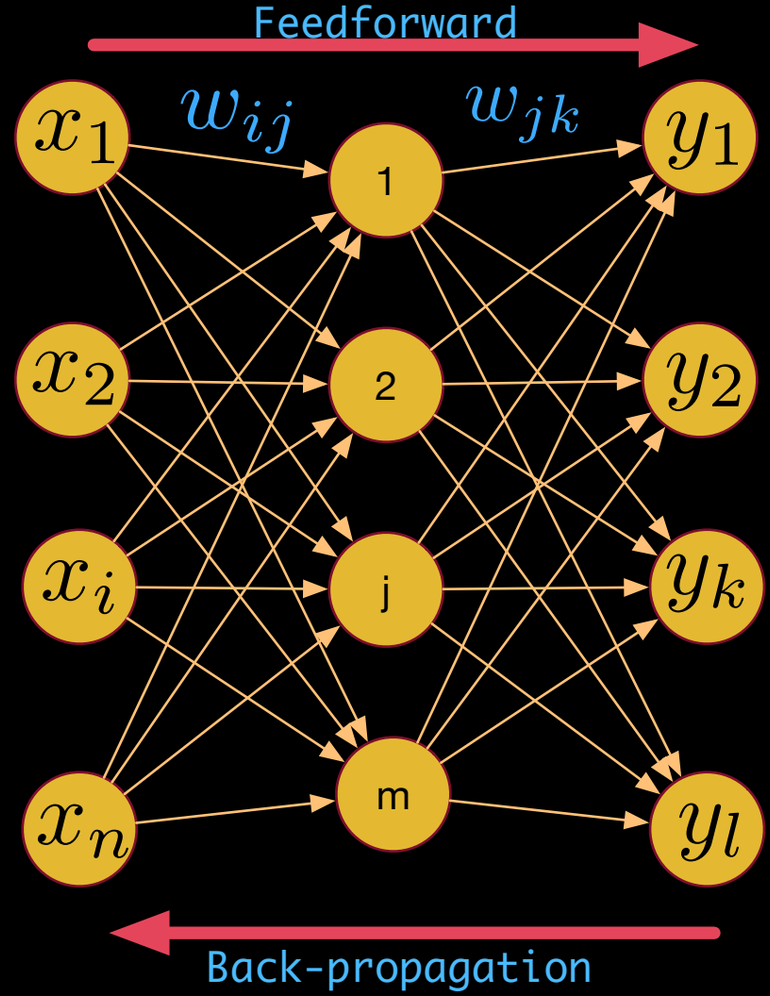
Feedforward Networks

- Input signals flow from left to right and error signals from right to left;



Feedforward Networks

- w_{ij} represents the weight that connects the input neuron i and the neuron in the hidden layer j
- w_{jk} the weight between the neuron j in the hidden layer and the neuron k in the output layer



Feedforward Networks

- objective function for optimization is defined as the MSE between the y_p and the desired output $\hat{y}_p$:

$$e_{p,k} = \hat{y}_{p,k} - y_{p,k}$$

Feedforward Networks

- first step is to correct the weight between j and k by minimizing the error;
- error gradient, learning rate;

$$w_{p+1,jk} = w_{p,jk} + \Delta w_{p,jk}$$

$$\Delta w_{p,jk} = \alpha y_{p,j} \delta(\hat{y}_{p,k})$$

$$\Delta w_{p,ij} = \alpha x_{p,i} \delta(\hat{y}_{p,j})$$

Convolutional Neural Networks

Convolutional Neural Nets

- image processing:
  - taking an input image and outputting a class (a cat, dog, etc)
- one of the first skills we learn from the moment we are born;
- being able to quickly recognize patterns, generalize from prior knowledge;

Convolutional Neural Nets

- computer "sees" an array of pixel values;

What We See

What Computers See

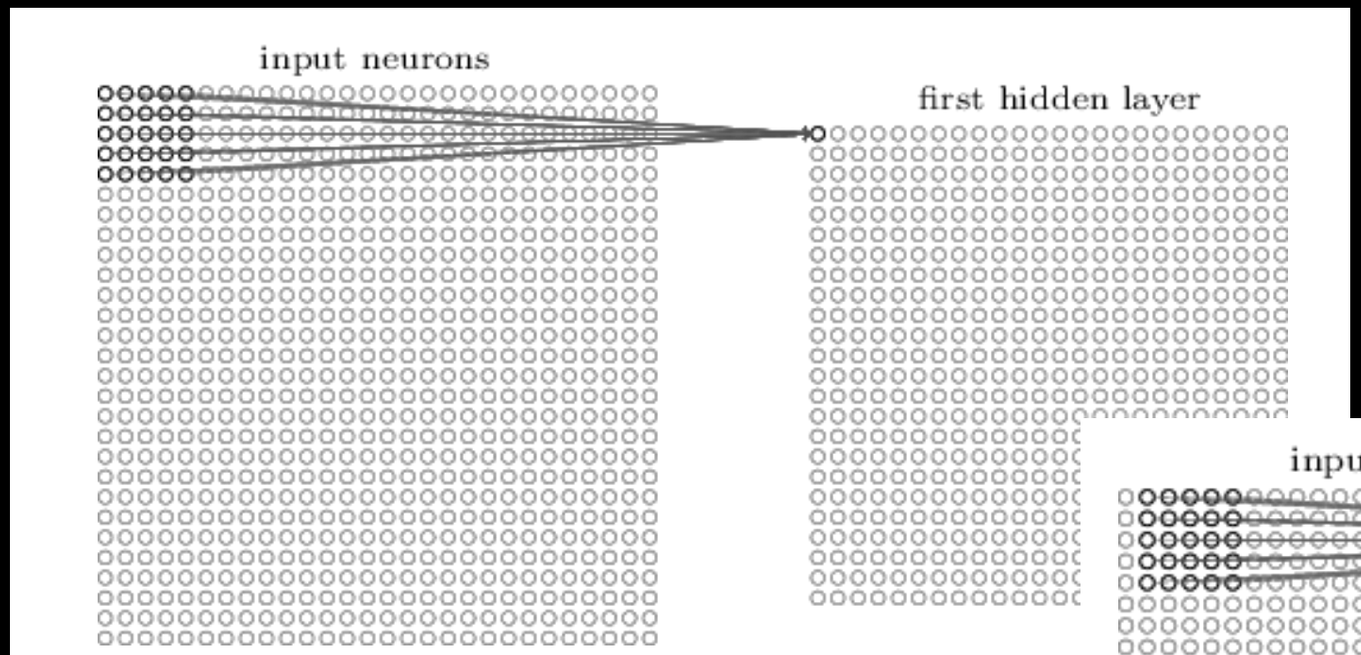
- give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class;

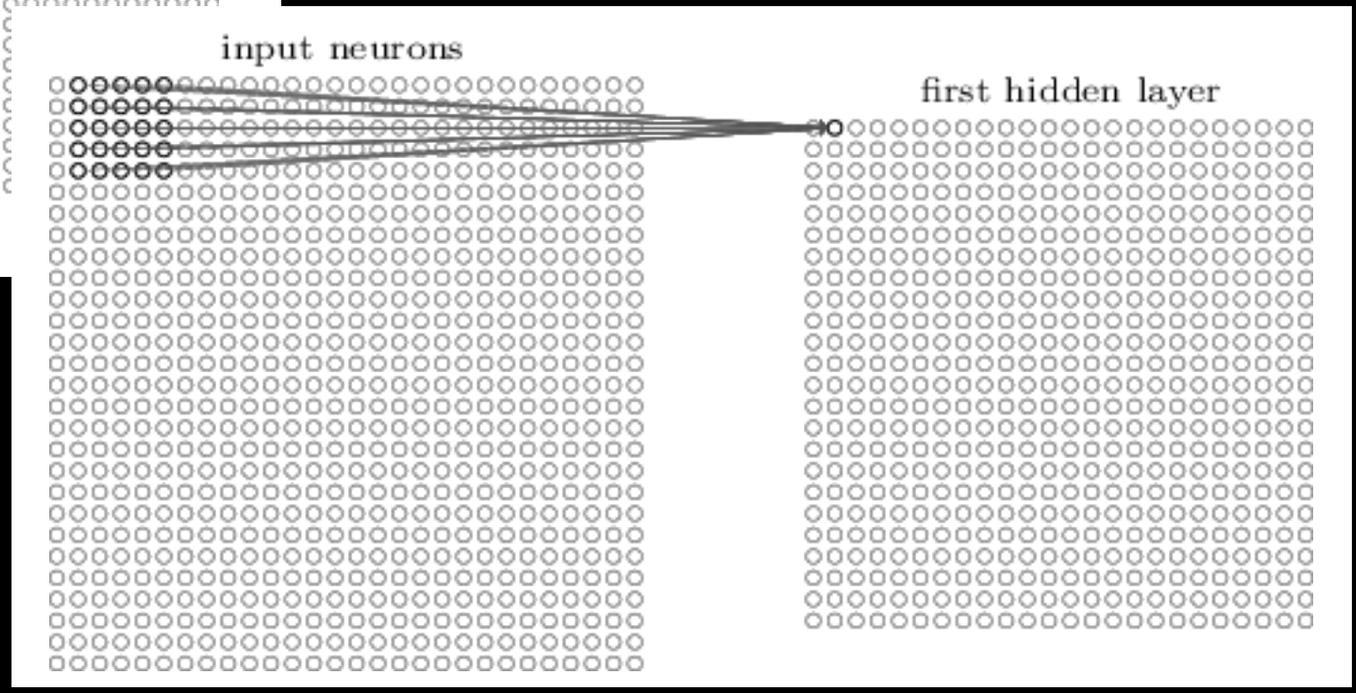
Convolutional Neural Nets

- Convolutional Layer:
  - filter sub-images;
  - slide over the image;
  - this filter is also an array of numbers (the numbers are called weights or parameters);
  - has the same depth as the input;
- receptive field;

Convolutional Neural Nets

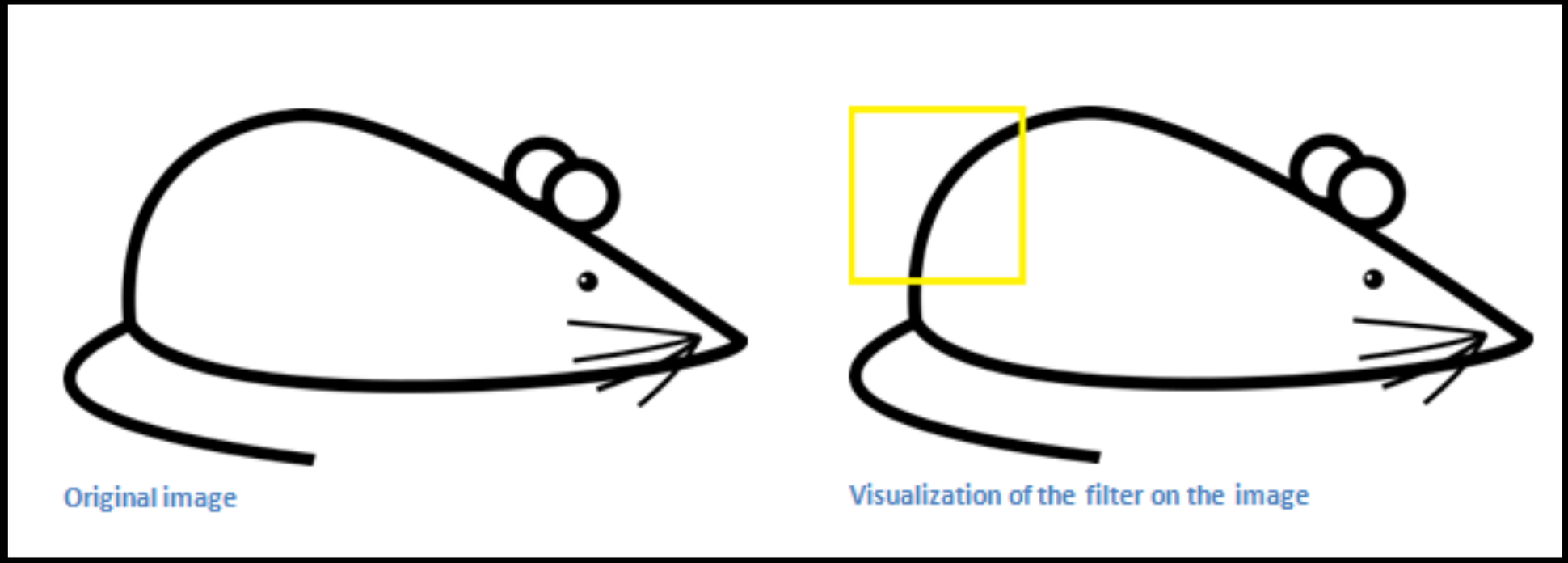
- Sliding the receptive field:





Convolutional Neural Nets

- Sliding the receptive field:



Convolutional Neural Nets

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

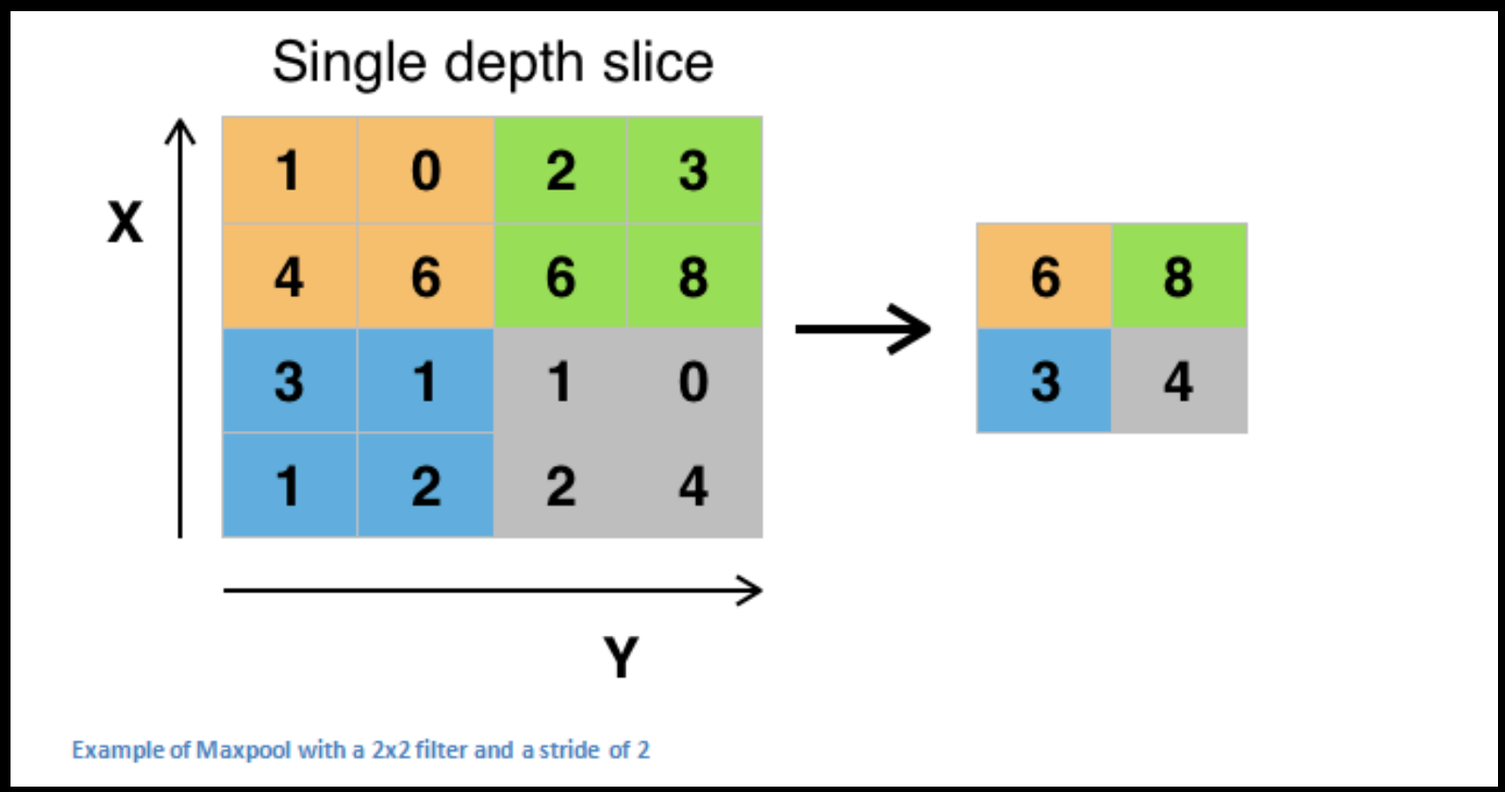
Multiplication and Summation = 0

Convolutional Neural Nets

- Stride: amount of pixels the filter slides over at a time;
    - Watch for integer output instead of fractions;
- Padding;
- Introduce Non Linearity (activation function);

Convolutional Neural Nets

- Pooling for downsampling;
  - MaxPooling, Average;
- drastically reduces the spatial dimension



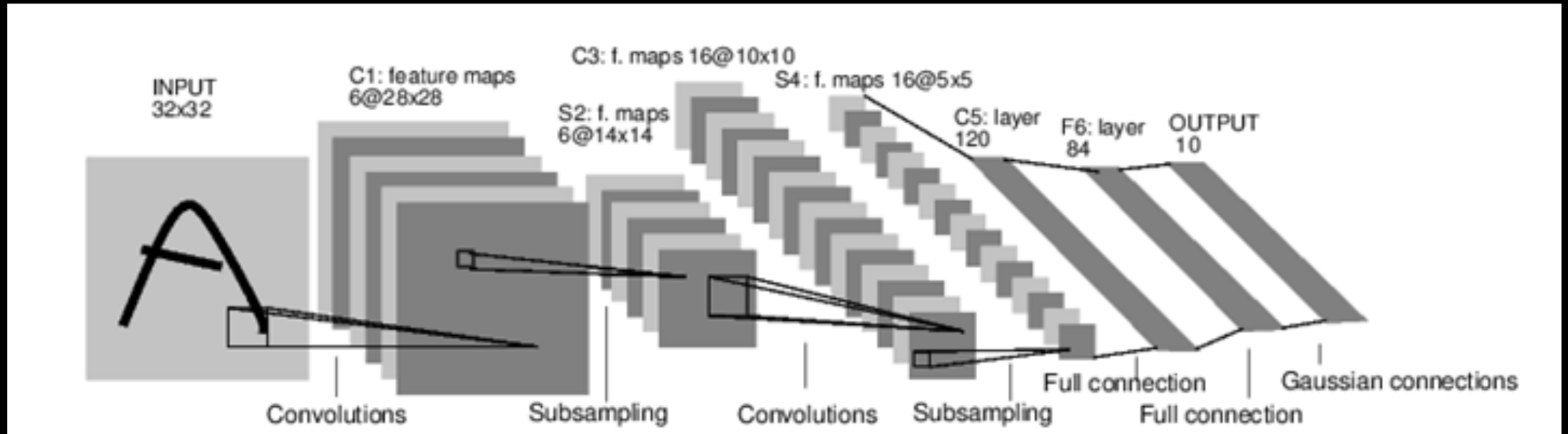
Convolutional Neural Nets

- Dropout layers:
  - “drops out” a random set of activations in that layer by setting them to zero;
  - forces the network to be redundant;
  - the net should be able to provide the right classification for a specific example even if some of the activations are dropped out;
  - Prevents overfitting;

Convolutional Neural Nets

- Fully Connected Layers:
  - Classification;
  - looks at the output of the previous layer and determines which features most correlate to a particular class;

Convolutional Neural Nets



A Full Convolutional Neural Network (LeNet)

Artificial Neural Networks

- It gets uglier!
  - Recurrent NN;
  - Long Short-Term Memory NN;
  - Generative Adversarial NN;
  - Radial bases function NN;
  - …

Questions?