Convolutional Neural Networks

Palma Project

Building the CNN

Steps of the CNN model:

STEP 1: Convolution

↓

STEP 2: Max Pooling

↓

STEP 3: Flattening

↓

STEP 4: Full Connection

Step 1 - Convolution

Purpose: Find features in the image using a feature detector, put them in a feature map but still preserve the special relationships between pixels.

$$\underset{\text{Input Image}}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}} \otimes \underset{\text{Feature Detector}}{\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}} = \underset{\text{Feature Map}}{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 2 & 1 \\ 1 & 4 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \end{bmatrix}}$$

[Reading: Convolution Matrix]()

Step 1 - Convolution

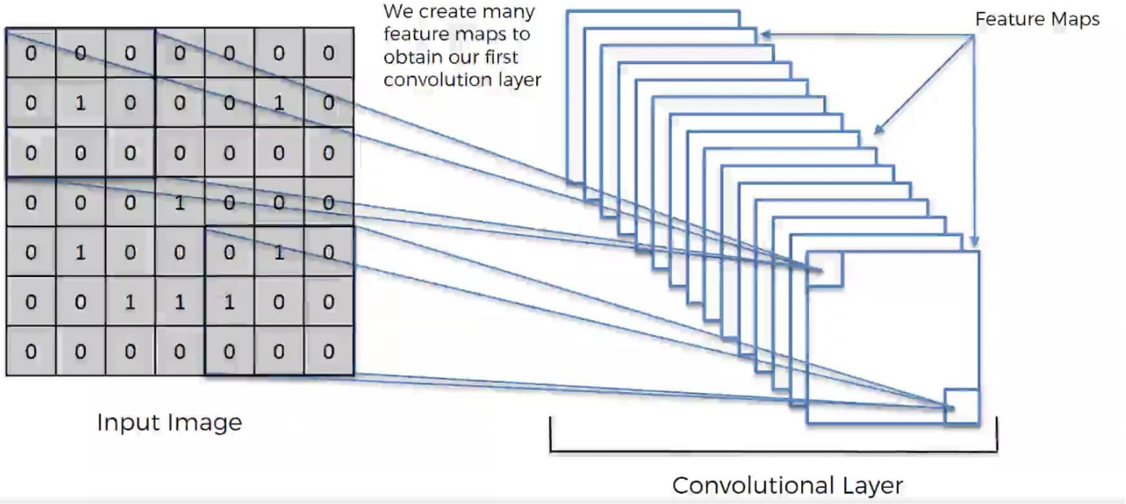
Example: Feature map of Geoffrey Hinton using a feature detector

*

1	0	-1
2	0	-2
1	0	-1

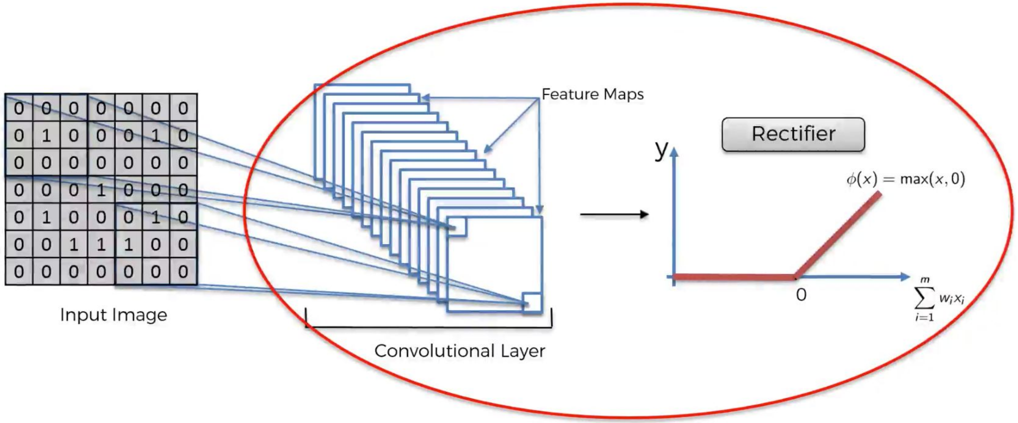
Step 1 - Convolution

<u>Purpose</u>: By using different Feature Detectors, we create many feature maps and then the first convolution layer. Each feature map corresponds to one specific feature of the image



Step 1 (B) - ReLU Layer

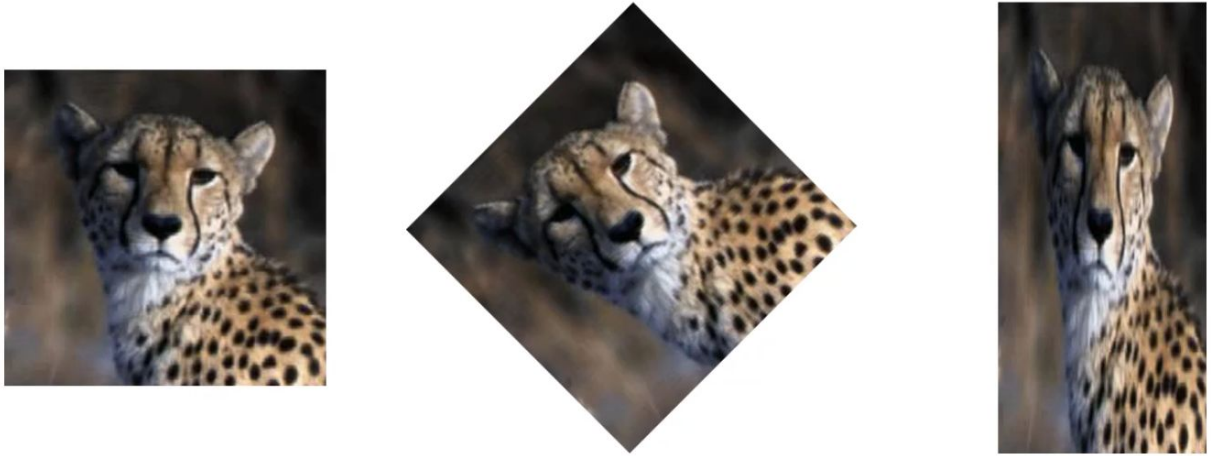
Purpose: Increase non linearity because images themseves are highly non linear.



[Reading: Understanding Convolutional Neural Networks with A Mathematical Model]

Step 2 - Pooling

We want the neural network to recognize the image even if it’s rotated or a little bit squashed



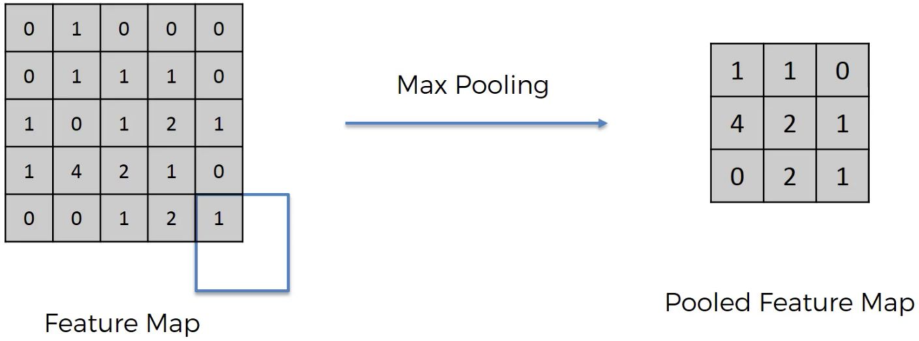
Step 2 - Pooling

We use Pooling in order to:

- Introduce Invariance
- Reduce the size of the feature map
- Remove irrelevant information and then prevent **overfitting**

Step 2 - Pooling

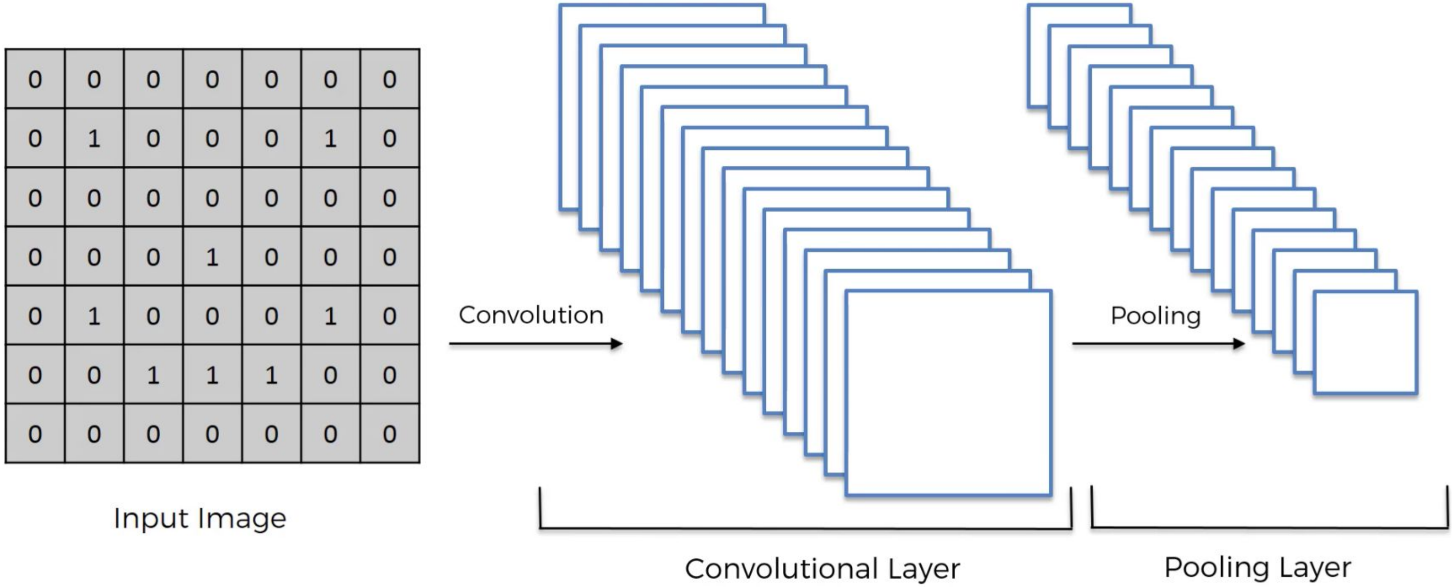
Example of Pooling: **Max Pooling**



[Reading: Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition]

Step 2 - Pooling

So far..

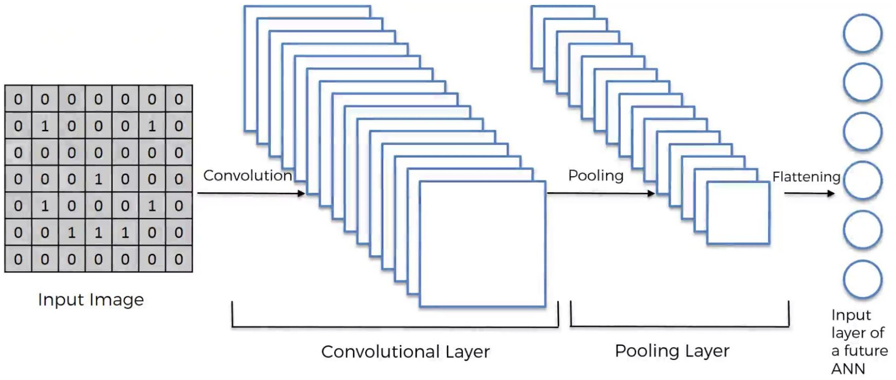


Step 3 - Flattening

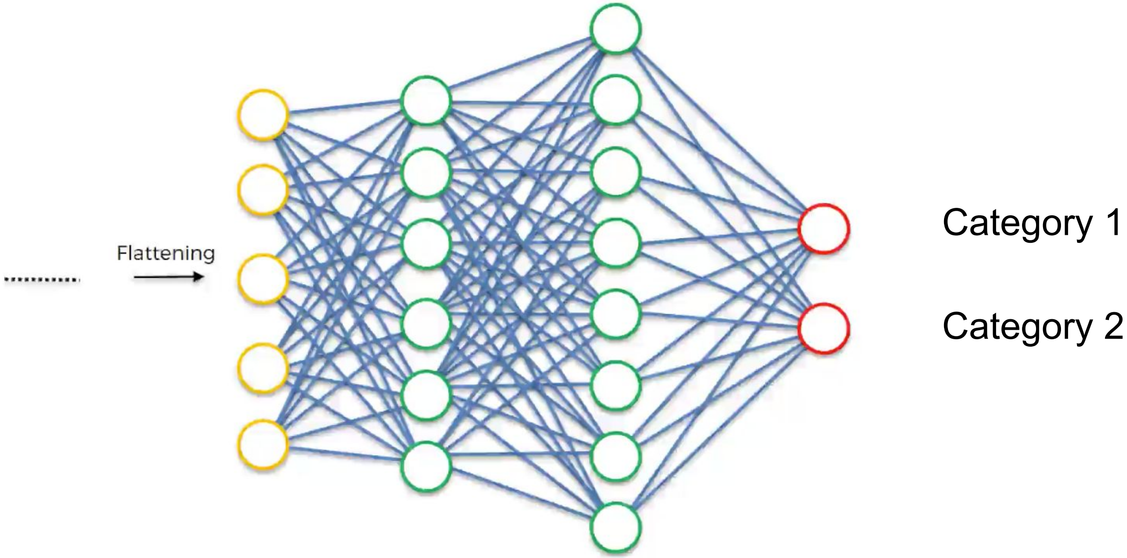
We flatten the Pooled Feature Map by taking the numbers raw by raw and put them in one column as an input to a future Artificial Neural Network.

Step 3 - Flattening

- We flatten all the pooled feature maps into a huge one-dimensional vector.
- Since each feature map corresponds to one specific feature of the image, then each node of this huge vector will represent an information of a specific feature of the input image.

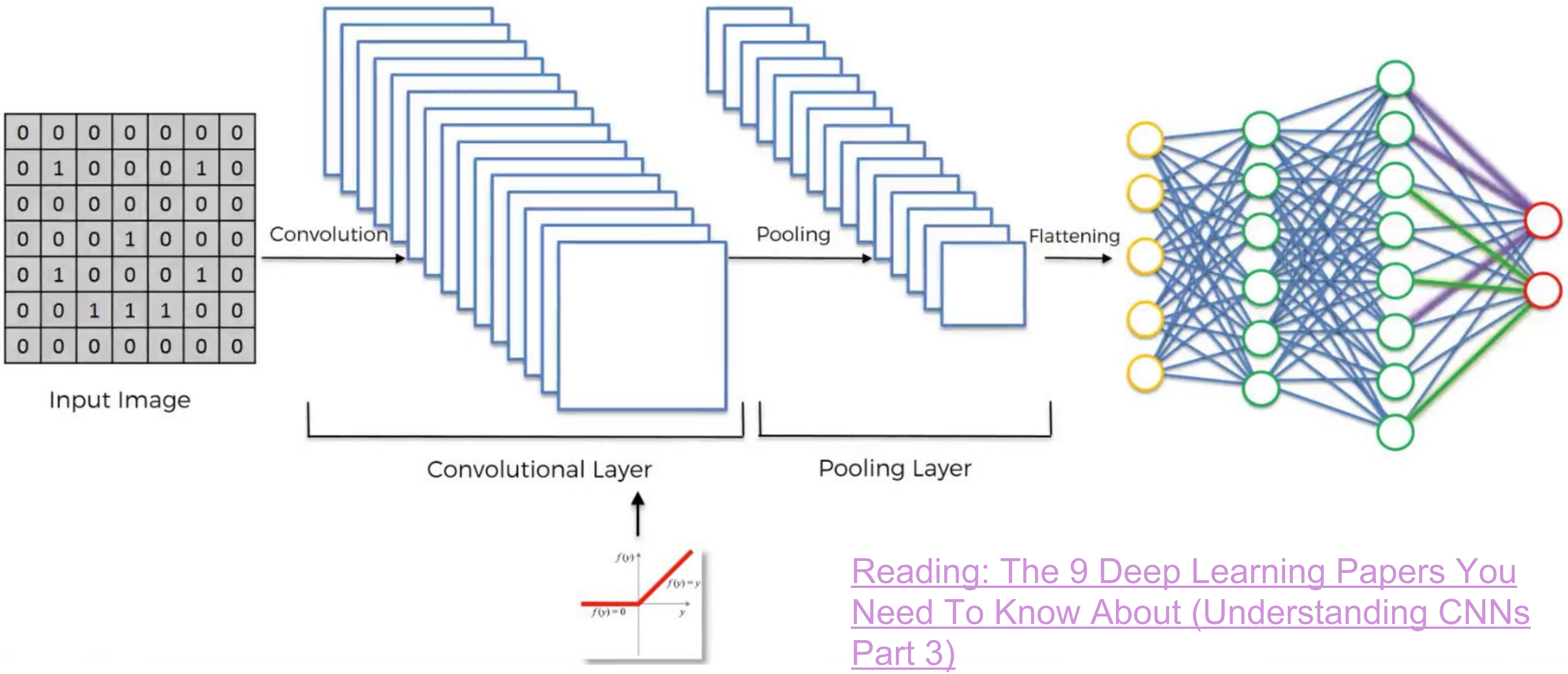


Step 4 - Full Connection



ANN

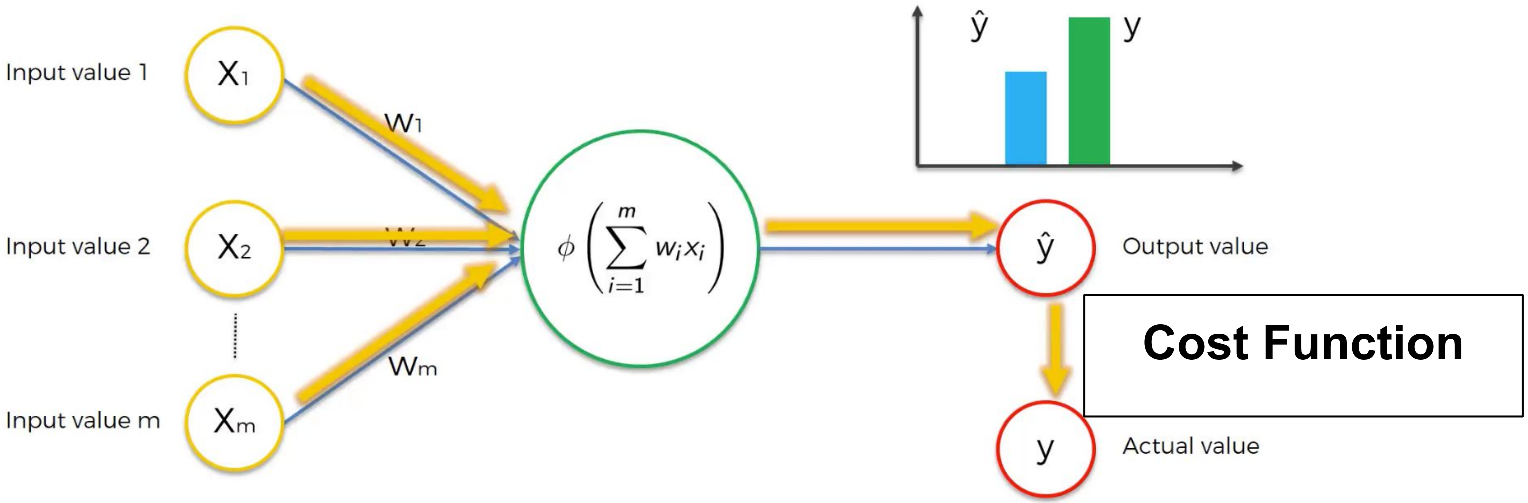
Step 4 - Summary



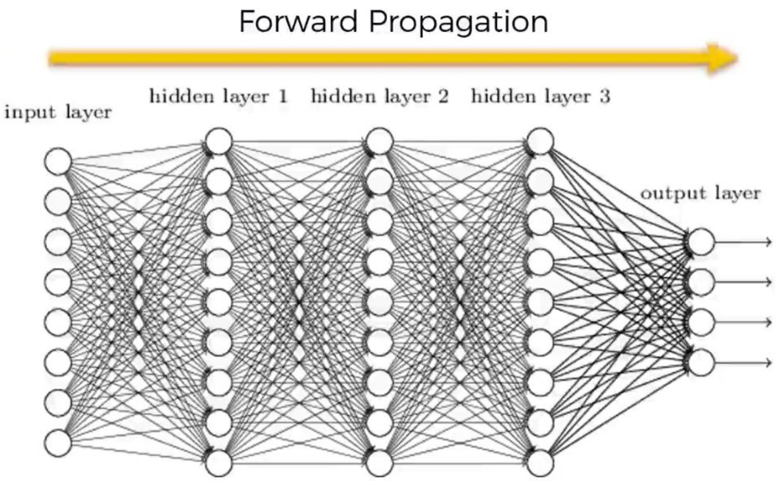
[Reading: The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)]

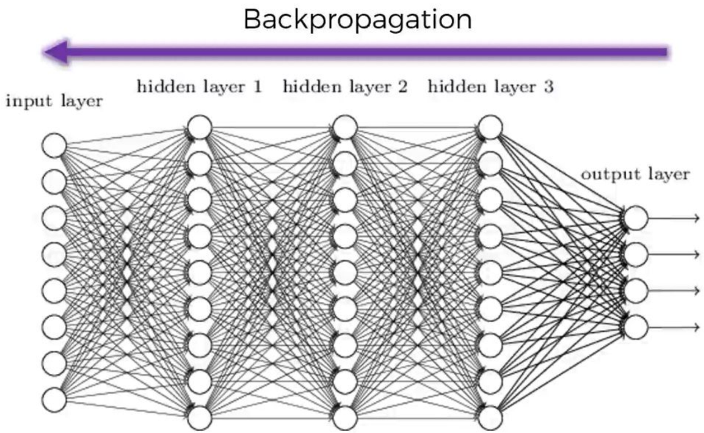
Fitting the CNN to the images

How do Neural Networks learn ?



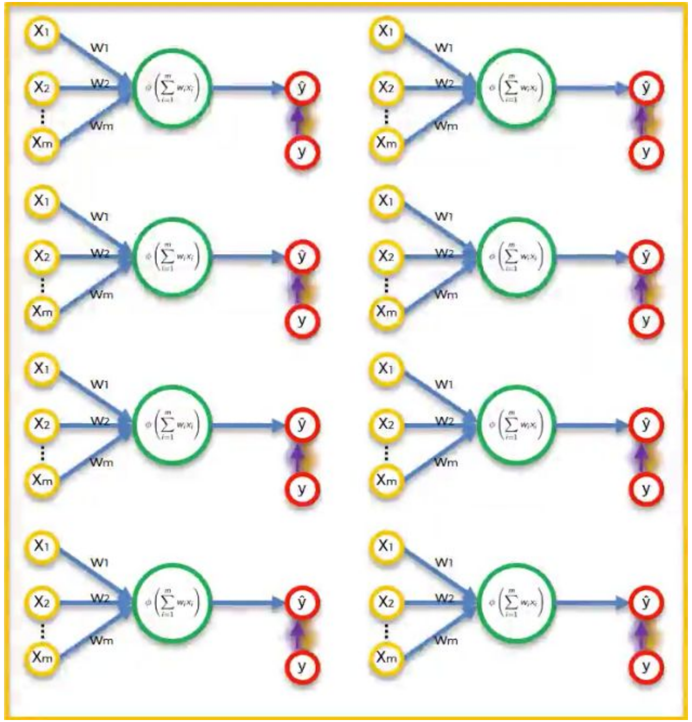
Backpropagation





Reading: How the backpropagation algorithm works

Stochastic Gradient Descent



[Reading: A Neural Network in 13 lines of Python (Part 2 - Gradient Descent)]()

Training the ANN with Stochastic Gradient Descent

STEP 1: Randomly initialise the weights to small numbers close to 0 (but not 0).

STEP 2: Input the first observation of your dataset in the input layer, each feature in one input node.

STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.

STEP 4: Compare the predicted result to the actual result. Measure the generated error.

STEP 5: Back-Propagation: from right to left, the error is back-propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 6: Repeat Steps 1 to 5 and update the weights after each observation (Reinforcement Learning). Or:
Repeat Steps 1 to 5 but update the weights only after a batch of observations (Batch Learning).

STEP 7: When the whole training set passed through the ANN, that makes an epoch. Redo more epochs.