COMP9321:
Data services engineering

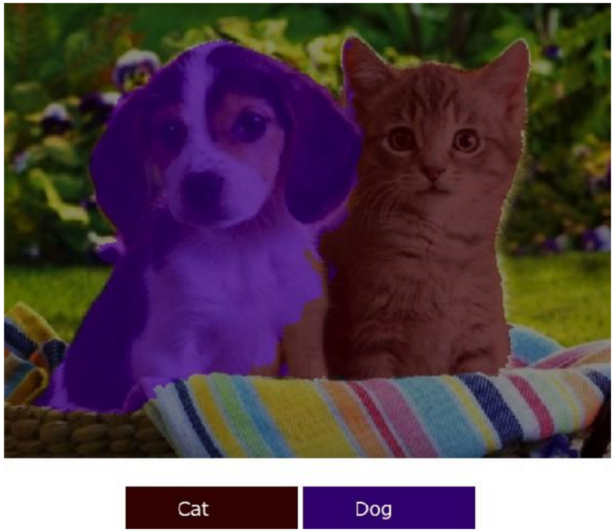
Week 10: Introduction to Deep Learning

Term 1, 2023

By Morty Al-Banna, CSE UNSW


Deep Learning


What can I help you with?


Cat
Dog

Why Deep Learning

- Deal very well with unstructured Data
- More Data = Better Learning
- More Accurate in some cases
- Extensive feature engineering is not required

When to use Deep Learning

- Deep Learning outperform other techniques if the *data size is large*. But with small data size, traditional Machine Learning algorithms are preferable.
- Deep Learning techniques need to have *high end infrastructure* to train in reasonable time.
- When there is *lack of domain understanding for feature introspection*, Deep Learning techniques outshines others as you have to worry less about feature engineering.
- Deep Learning really shines when it comes to *complex problems such as image classification, natural language processing, and speech recognition*.

Many machine learning methods inspired by biology, e.g., the (human) brain

Our brain has $\sim 10^{11}$ neurons, each of which communicates (is connected) to $\sim 10^{4}$ other neurons

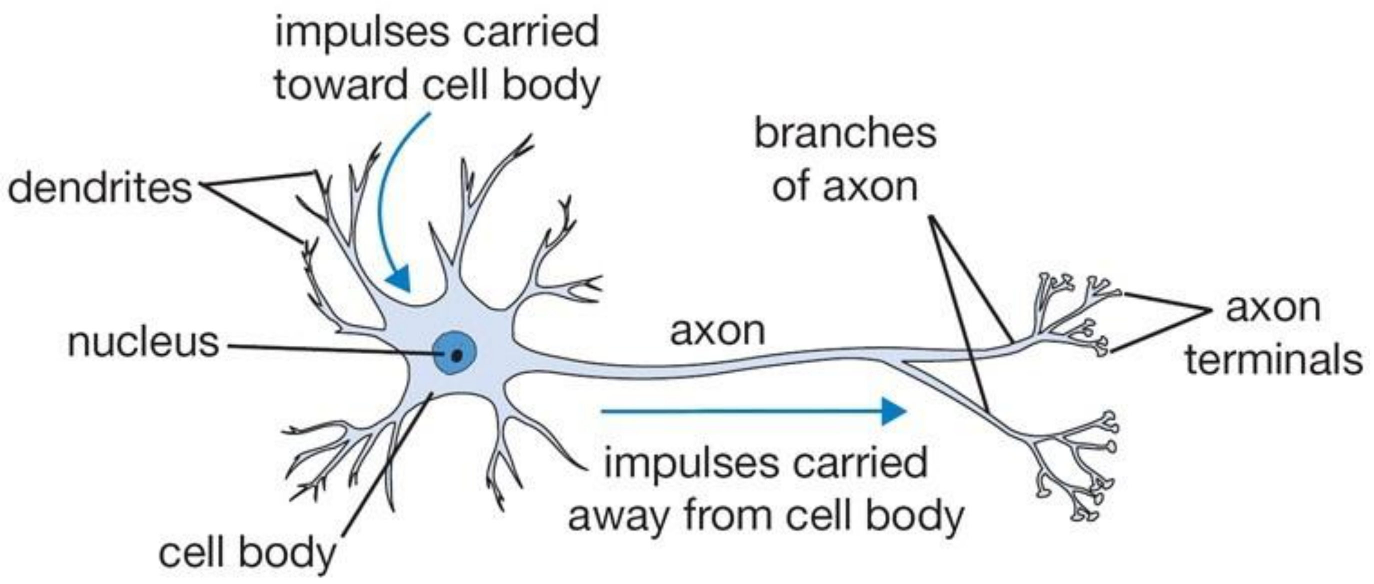


Figure : The basic computational unit of the brain: Neuron

Neural networks define functions of the inputs (hidden features), computed by neurons

Artificial neurons are called units

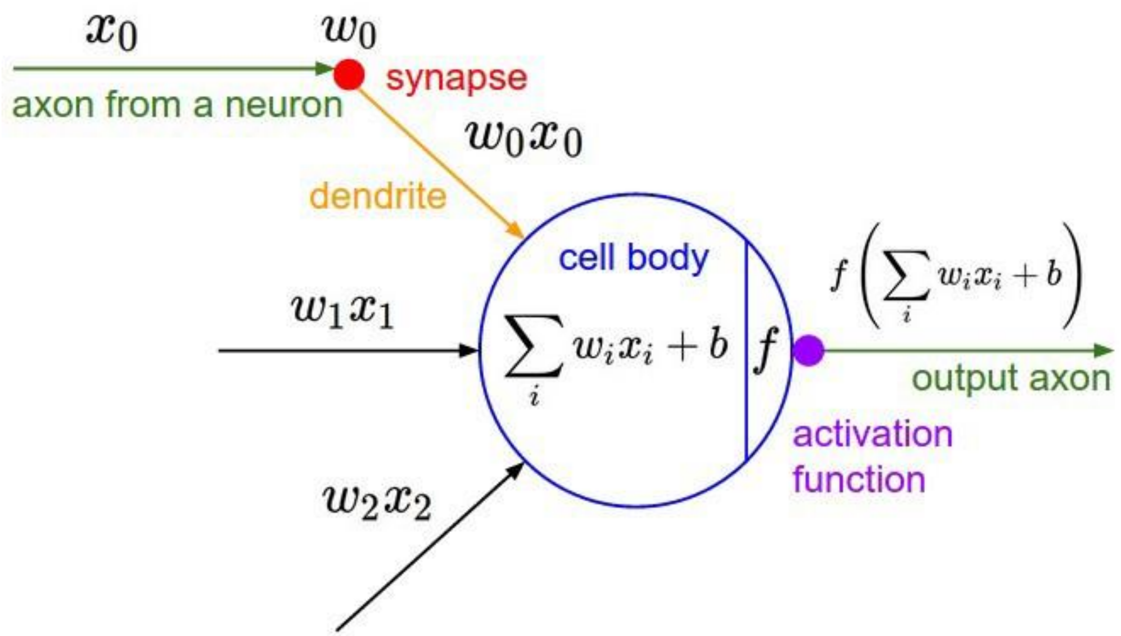


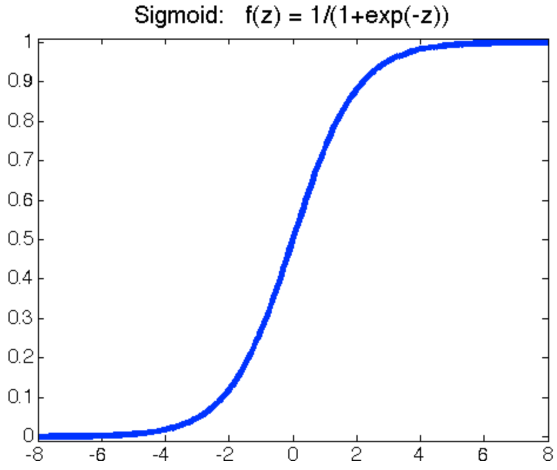
Figure : A mathematical model of the neuron in a neural network

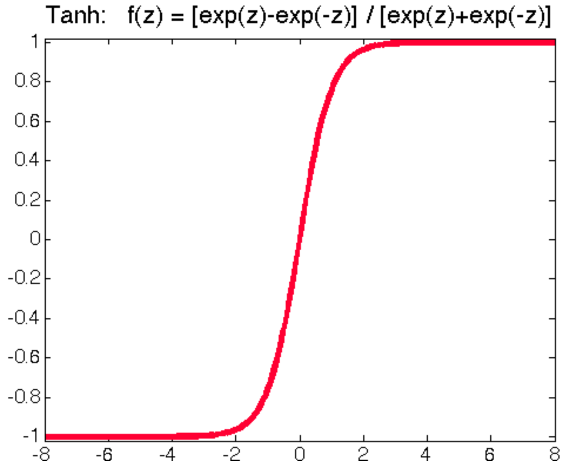
Most commonly used activation functions:

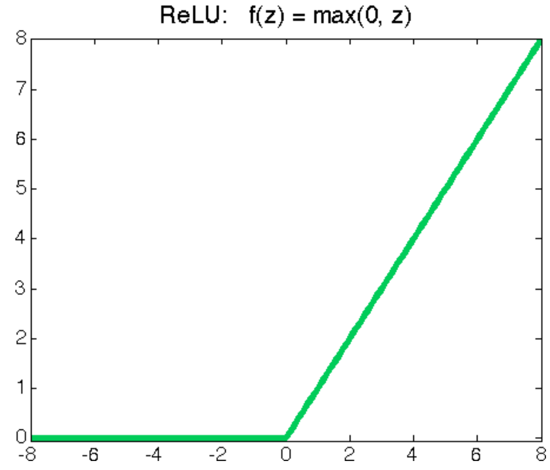
Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$

Tanh: $\tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$

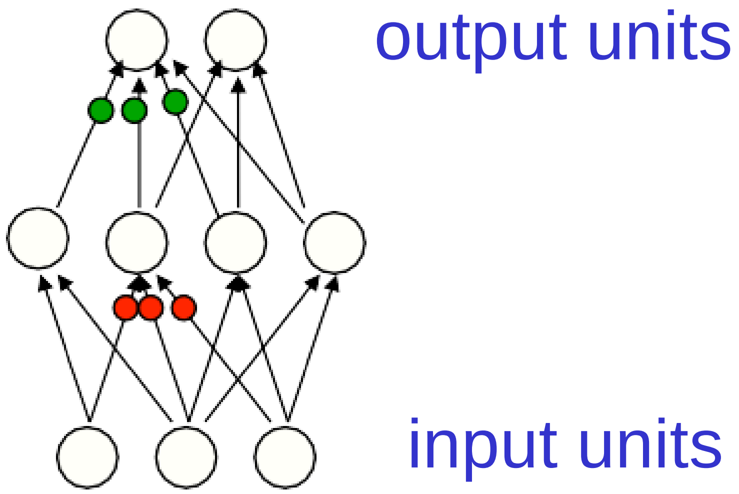
ReLU (Rectified Linear Unit): $\mathrm{ReLU}(z) = \max(0, z)$







Network with one layer of four hidden units:



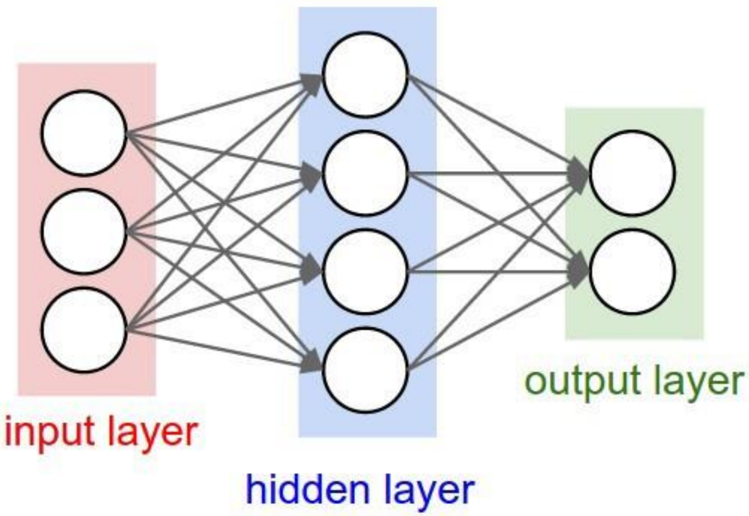


Figure : Two different visualizations of a 2-layer neural network. In this example: 3 input units, 4 hidden units and 2 output units

Naming conventions; a 2-layer neural network:

- One layer of hidden units
- One output layer
  - (we do not count the inputs as a layer)

Going deeper: a 3-layer neural network with two layers of hidden units

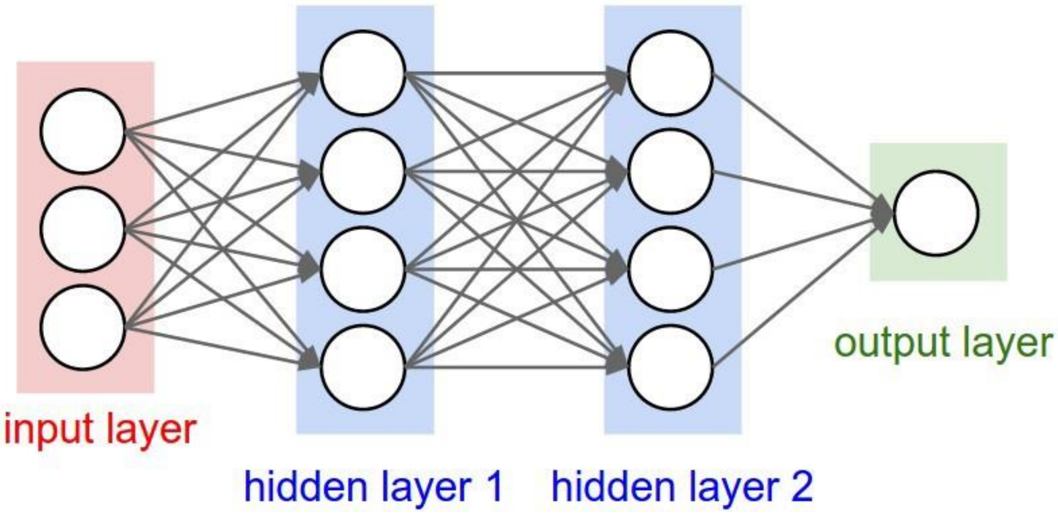


Figure : A 3-layer neural net with 3 input units, 4 hidden units in the first and second hidden layer and 1 output unit

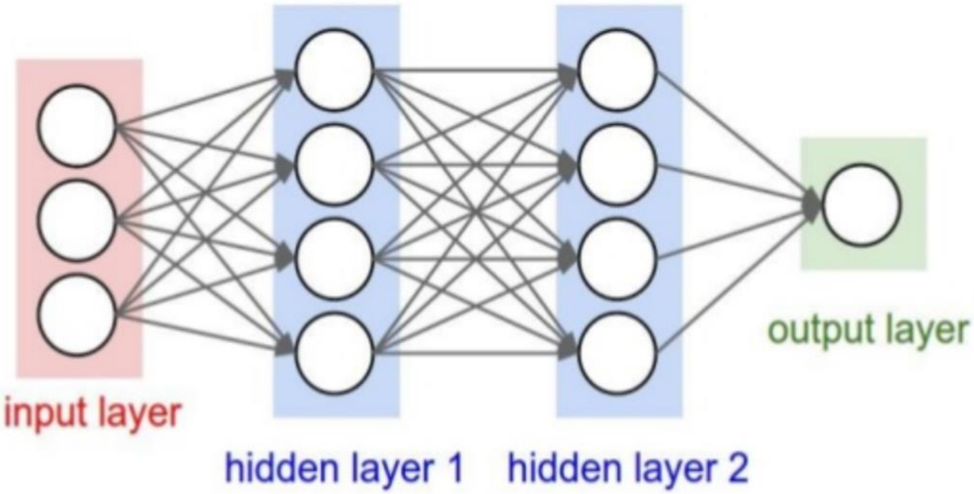
Naming conventions; a N-layer neural network:

$N - 1$ layers of hidden units

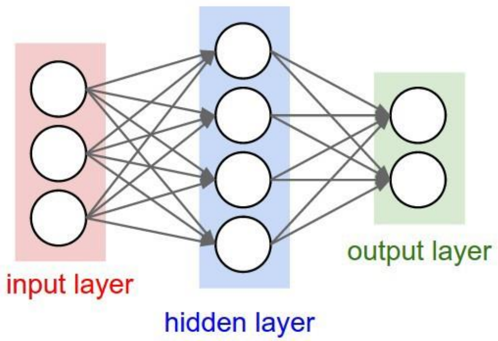
One output layer

Forward Pass (Forward Propagation)

- The input neurons first receive the data features of the object. After processing the data, they send their output to the first hidden layer.
- The hidden layer processes this output and sends the results to the next hidden layer.
- This continues until the data reaches the final output layer, where the output value determines the object's classification.
- This entire process is known as Forward Propagation, or Forward prop.



Forward Pass (Forward Propagation)



Output of the network can be written as:

$$h_j(\mathbf{x}) = f\left(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}\right)$$

$$o_k(\mathbf{x}) = g\left(w_{k0} + \sum_{j=1}^{J} h_j(\mathbf{x}) w_{kj}\right)$$

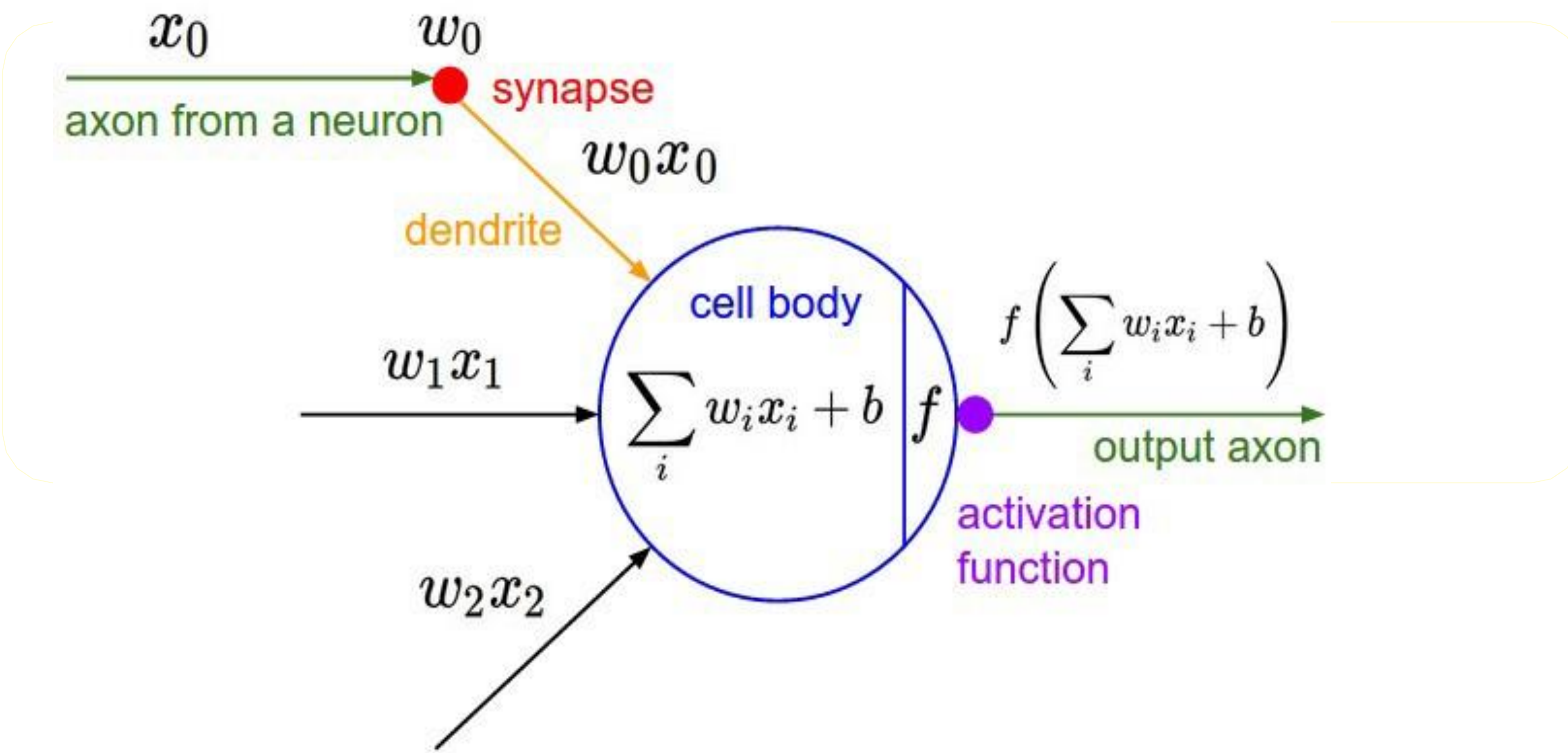
(j indexing hidden units, k indexing the output units, D number of inputs)

Activation functions f, g: sigmoid/logistic, tanh, or rectified linear (ReLU)

$$\sigma(z) = \frac{1}{1+\exp(-z)}, \quad \tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}, \quad \mathrm{ReLU}(z) = \max(0, z)$$

Backward Pass (Back-propagation)

- Back-propagation: an efficient method for computing gradients needed to perform gradient-based optimization of the weights in a multi-layer network



Backward Pass (Back-propagation)

- Back-propagation: an efficient method for computing gradients needed to perform gradient-based optimization of the weights in a multi-layer network

Training neural nets:

Loop until convergence:

- for each example n
  1. Given input $\mathbf{x}^{(n)}$, propagate activity forward ($\mathbf{x}^{(n)} \rightarrow \mathbf{h}^{(n)} \rightarrow o^{(n)}$) (**forward pass**)
  2. Propagate gradients backward (**backward pass**)
  3. Update each weight (via gradient descent)

What are we propagating Backward?

- Find weights:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \operatorname{loss}(\mathbf{o}^{(n)}, \mathbf{t}^{(n)})$$

  where $\mathbf{o} = f(\mathbf{x}; \mathbf{w})$ is the output of a neural network

- Define a loss function, eg:
  - Squared loss: $\sum_k \frac{1}{2}(o_k^{(n)} - t_k^{(n)})^2$
  - Cross-entropy loss: $-\sum_k t_k^{(n)} \log o_k^{(n)}$

- Gradient descent:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \frac{\partial E}{\partial \mathbf{w}^t}$$

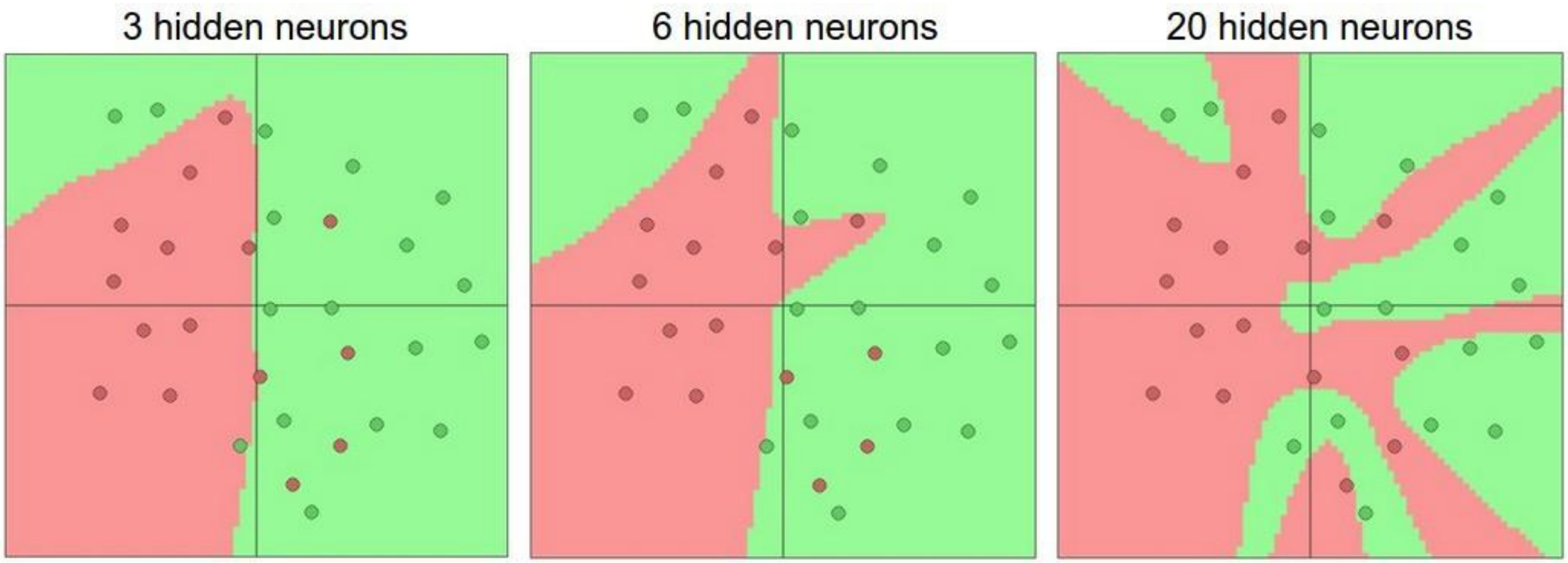
  where η is the learning rate (and E is error/loss)

Some Types of Deep Neural Networks

- **Feedforward Networks (FFN):** Best for when we have an idea of the output
- **Recurrent Neural Network (RNN):** Best for time series Data
- **Convolutional Neural Network (CNN):** Best of Image, text, voice recognition.

Neural network with at **least one hidden layer** is a universal approximator (can represent any function).

Proof in: Approximation by Superpositions of Sigmoidal Function, Cybenko, [paper]



The capacity of the network increases with more hidden units and more hidden layers

Why go deeper? Read e.g.,: Do Deep Nets Really Need to be Deep? Jimmy Ba, Rich Caruana, Paper: [paper]

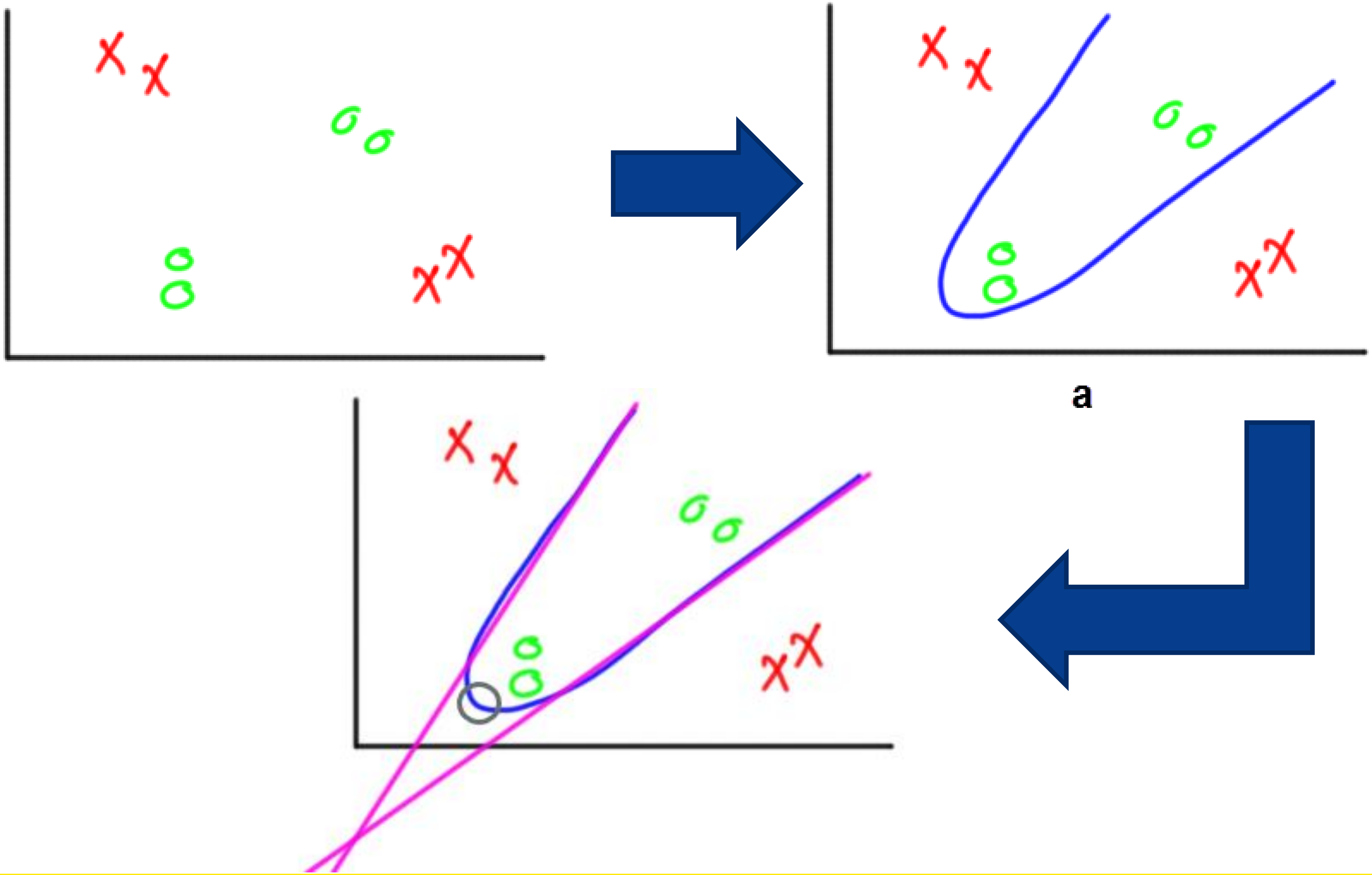
[http://cs231n.github.io/neural-networks-1/]

How to Determine Number of Layers and Number of Neurons in Each Layer?

1. Based on the data, draw an expected decision boundary to separate the classes.
2. Express the decision boundary as a set of lines. Note that the combination of such lines must yield to the decision boundary.
3. The number of selected lines represents the number of hidden neurons in the first hidden layer.
4. To connect the lines created by the previous layer, a new hidden layer is added. Note that a new hidden layer is added each time you need to create connections among the lines in the previous hidden layer.
5. The number of hidden neurons in each new hidden layer equals the number of connections to be made.

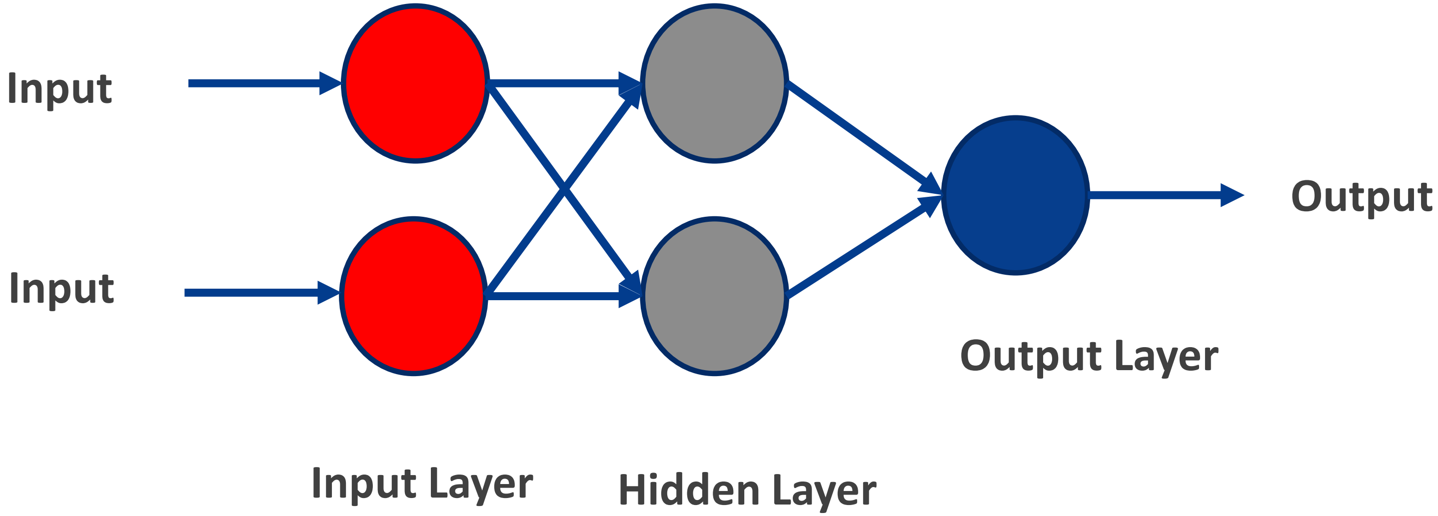
https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e

How to Determine Number of Layers and Number of Neurons in Each Layer?



https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e

How to Determine Number of Layers and Number of Neurons in Each Layer?



Deep Neural Networks Weaknesses

- Deep Learning requires a large dataset, hence long training period.
- In term of cost, Machine Learning methods like SVMs and other tree ensembles are very easily deployed even by relative machine learning novices and can usually get you reasonably good results.
- Deep learning methods tend to learn everything. It's better to encode prior knowledge about structure of images (or audio or text).
- The learned features are often difficult to understand. Many vision features are also not really human-understandable (e.g, concatenations/combinations of different features).
- Requires a good understanding of how to model multiple modalities with traditional tools.

Useful Tools

- TensorFlow
- PyTorch
- Apache MXNet
- Caffe

Useful Resources

- https://towardsdatascience.com/introducing-deep-learning-and-neural-networks-deep-learning-for-rookies-1-bd68f9cf5883
- https://www.edureka.co/blog/backpropagation/
- https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f
- https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/
- https://medium.com/datadriveninvestor/thats-not-enough-we-have-to-go-deeper-24dd16d85828

Q&A