ECE 1508: Applied Deep Learning

Chapter 1: Preliminaries

Ali Bereyhi

ali.bereyhi@utoronto.ca

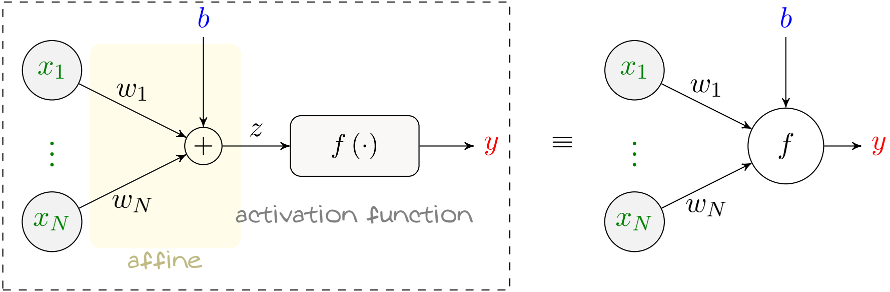
Department of Electrical and Computer Engineering

University of Toronto

Winter 2025

Artificial Neuron

Perceptron is a *special* artificial neuron. In general, an artificial neuron is



A neuron with an N-dimensional input has $N+1$ learnable parameters

- N weights, i.e., $w_1, \dots, w_N$
- a bias

From now on, when not needed, we drop them from diagram for compactness

Artificial Neuron

The output of neuron y is related to its inputs as $\boldsymbol{x}$

$$y = f\left(\mathbf{w}^{\mathsf{T}}\boldsymbol{x} + b\right)$$

where we define

- $\mathbf{w}^{\mathsf{T}} = [w_1, \ldots, w_N]$ to be the vector of weights
- b to be the bias
- $f(\cdot) : \mathbb{R} \mapsto \mathbb{R}$ to be the activation function

In perceptron, the activation function was *step function*[1]

A neuron can in general have any activation

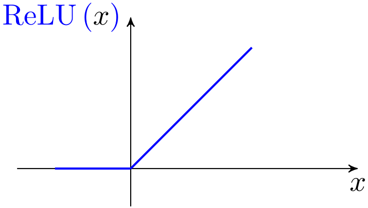
Another special case is the linear activation $f(z) = z$ by which the neuron reduces to the basic linear model

[1] It turns out soon that this was in fact a bad choice of activation!

Artificial Neuron: *Activation*

If we intend to learn nonlinear functions; then, we need nonlinear activation

Some sample activation functions: *rectified linear unit (ReLU) function*

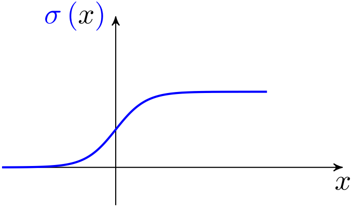


$$\mathrm{ReLU}\left(x\right) = \max\left\{x, 0\right\}$$

Artificial Neuron: *Activation*

If we intend to learn nonlinear functions; then, we need nonlinear activation

Some sample activation functions: *sigmoid function*

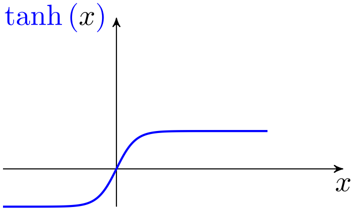


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Artificial Neuron: *Activation*

If we intend to learn nonlinear functions; then, we need nonlinear activation

Some sample activation functions: *hyperbolic tangent function*

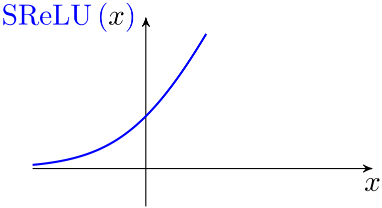


$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Artificial Neuron: *Activation*

If we intend to learn nonlinear functions; then, we need nonlinear activation

Some sample activation functions: *soft ReLU function*

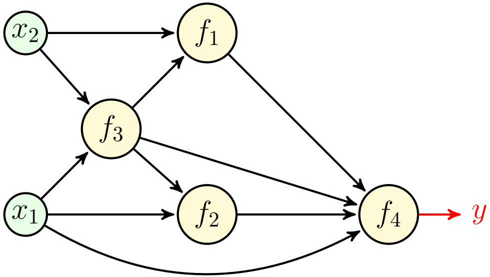


$$\mathrm{SReLU}(x) = \log(1 + e^{x})$$

Neural Network

Artificial Neural Network

Artificial neural network is a *directed graph* that connects a set of inputs to a set of outputs: *the nodes of this graph are neurons whose activation functions are known and whose weights and biases are learnable*

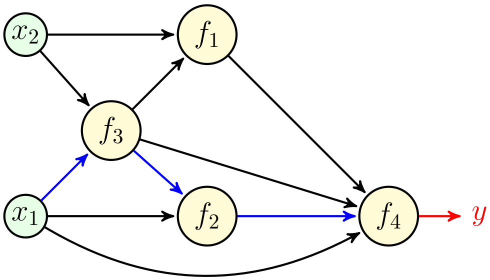


Unless we are talking to biologists, we can safely drop the term artificial ☺

Neural Network: *Depth*

Depth of a Neural Network

The longest path between an input and the output

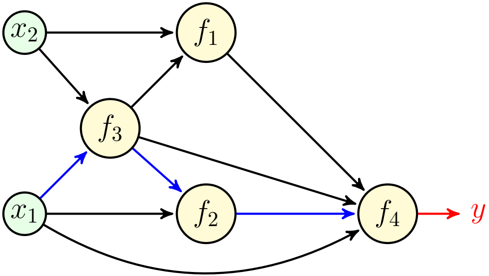


Here, the depth is 3

Deep Neural Network

Deep Neural Network

A neural network whose depth is larger than 2



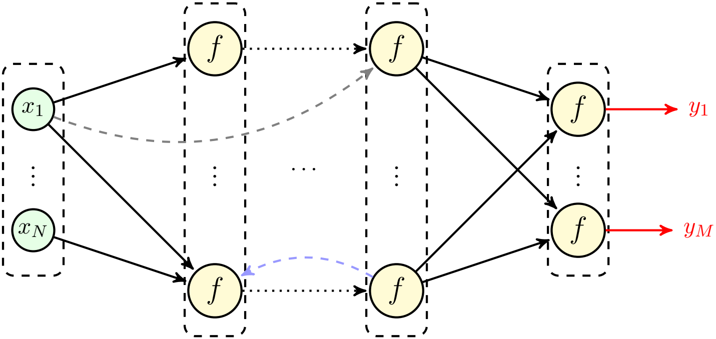
Here, the depth is 3 > 2 ⇝ it's a deep neural network

Neural Network: *Layered Architecture*

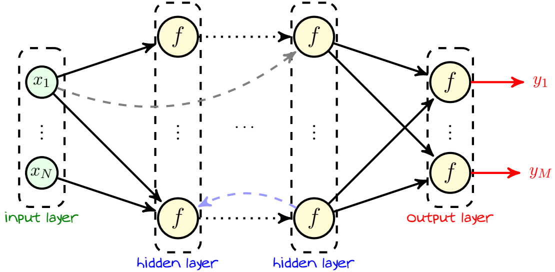
In practice, we use neural networks with layered architectures

Layer

A subset of neurons that are in the same distance from inputs



Neural Network: *Layered Architecture*

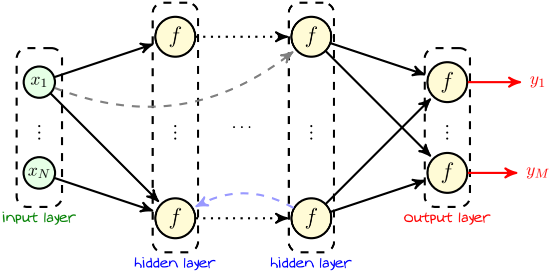


There are three types of layers

- Input layer that only contains inputs **Attention:** *No neuron here!*
- Output layer that contains the neurons computing network outputs
- hidden layers that contain neurons and are in between

It's readily seen from the definition that *depth = # hidden layers + 1*

Neural Network: *Layered Architecture*



It's readily seen from the definition that with layered architecture

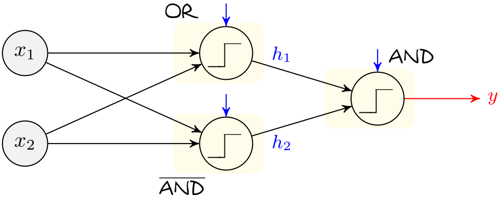
$$\boxed{\textit{depth} = \textit{\# hidden layers} + 1}$$

Hence, we could equivalently say that

a deep neural network has more than one hidden layer

Neural Network: *Layered Architecture*

Let's try our knowledge on the XOR neural network

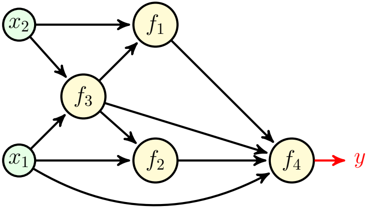


- Input layer has two inputs
- Output layer has a single neuron (perceptron)
- One hidden layer that has two neurons
  - ↳ *It's not a deep neural network, since it has only one hidden layer*
  - ↳ *It's a shallow neural network*

Neural Networks are *Models*

+ *If we talk in terms of ML components, what is a deep neural network?*
– It's a parameterized function from inputs to outputs; so, *it's a model*
+ *Well! What are then the hyperparameters and learnable parameters?*
– Anything that describes the architecture is hyperparameter
  ↳ *Number of neurons, how they are connected, depth, activations, . . .*
– The weights and biases are the learnable parameters
+ *So, it means if the architecture of the neural network is known, we explicitly know which parameters we should learn! Right?*
– Exactly! Let's look at the dummy diagram we had in previous slides!

Neural Networks are *Models*



Here, we've chosen to have 4 neurons with activations $f_1(\cdot), \ldots, f_4(\cdot)$ arranged in the above form: *these are hyperparameters*

Now that the architecture is fixed, we could say

↳ *Neurons 1, 2, 3 have two inputs: each of them has two weights and a bias*

↳ *Neurons 4 has four inputs: it has four weights and a bias*

So in total we have $3 \times (2+1) + (4+1) = 14$ learnable parameters

Deep Learning

+ *Are we finally ready to define Deep Learning?*
– Yes! There we go

Deep Learning (DL)

When we use a deep neural network to address a learning task,

we are doing deep learning

Now, let's get things a bit clear

- In ML, we talk about any model, any loss, any dataset
- In *Representation Learning*, we roughly talk about models that can describe nonlinear functions: *this includes deep neural networks*
- In DL, we have a deep neural network as the model

So, we can say DL $\subset$ Representation Learning $\subset$ ML

Starting with Deep Learning

At this point, we know

- For a given learning task, we specify the dataset, a model and a loss
- In DL, we use deep neural networks as models
- To accomplish the learning task, we need to train the model
  - ↳ *find weights and biases that achieve minimal empirical risk*

But, we yet don't know?

- How to minimize the empirical loss?
  - ↳ *what algorithm should we use?*
- What kind of hyperparameters, i.e., architecture, should we use?
  - ↳ *how many neurons do we need?*
  - ↳ *how should we connect the neurons?*

This is what we learn from now on! The only last piece of preliminary knowledge that we need is the method of gradient decent which we study next.