Video 20

Thursday, October 29, 2020 4:40 PM

Single neuron

$$y = \vec{w}^T \vec{x} + b = \sum_i w_i x_i + b$$

w_i: i^{th} weight component in vector $\vec{w}$

x_i: each feature i of the input $\vec{x}$

(here the number of features is 3)

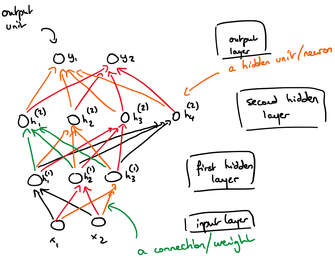
Combined with an activation function:

$$y = g\left(\vec{w}^T \vec{x} + b\right)$$

g: logistic function

Deep neural networks

- connect lots of neurons (units) together in a directed acyclic graph
- we obtain a feed-forward neural network
- if graph has cycles, instead we call the architecture a recurrent neural network (covered later in the course.)
- units are grouped together in layers (neurons)



Fully connected layer

- Each layer has N input units and M output units
- All output units are connected to all input units. (later, we will study other layer types where not all connections are made)
- Here we have an $M \times N$ weight matrix
- Output units are a function of the input units:

$$\vec{y} = \phi\left(W\vec{x} + \vec{b}\right)$$

$\vec{y}$: M dimensional vector (each component is an output neuron)

ϕ: activation function

W: $M \times N$ weight matrix.

$\vec{b}$: bias vector (M dimensional)

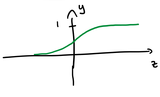
$\vec{x}$: input of this layer (N dimensional vector)

! not necessarily the input of the model

- A deep neural network composed of multiple fully-connected layers is called a multilayer perceptron

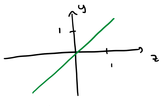
! This has nothing to do with the perceptron we studied earlier (this is a bad name choice)

Activation functions



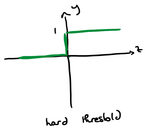
Logistic

$$y = \frac{1}{1+e^{-z}}$$



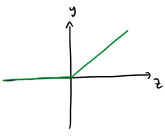
Linear

$y = z$



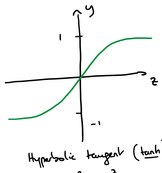
hard threshold

$$y = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z \leq 0 \end{cases}$$



ReLU

$y = \max(0, z)$



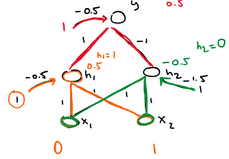
Hyperbolic tangent (tanh)

$$y = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Small neural network to solve XOR:

$1 \oplus 1 = 0$ ✓

$0 \oplus 1 = 1$ ✓



All neurons apply the hard threshold activation function

Multilayer neural network as a composition of functions:

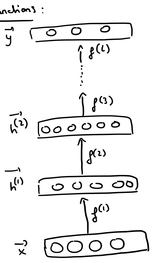
$$\vec{h}^{(1)} = f^{(1)}\left(\vec{x}\right)$$

$$\vec{h}^{(2)} = f^{(2)}\left(\vec{h}^{(1)}\right)$$

$$\vec{h}^{(3)} = f^{(3)}\left(\vec{h}^{(2)}\right)$$

$$\vdots$$

$$\vec{y} = f^{(L)}\left(\vec{h}^{(L-1)}\right)$$



We can write this as:

$$\vec{y} = f^{(L)} \circ f^{(L-1)} \circ \dots \circ f^{(2)} \circ f^{(1)}\left(\vec{x}\right)$$

Composition of functions represented by the deep neural network

Pro: This provides modularity: we can implement each layer's computations as a black box.

We can also interpret deep neural nets as performing feature learning

ψ: feature map defined by the first layers of the deep neural net

$$\psi = h^{(2)} \circ h^{(1)}$$