Neural Networks

Astrini Sie

asie@seattleu.edu

Week 7a

ECEGR4750 - Introduction to Machine Learning
Seattle University

October 31, 2023

Overview

1. Neural Networks
   - Perceptron
   - Stacking Neurons
   - Fully Connected Multi-layer Neural Network
   - Forward Pass
   - Backpropagation
   - Training a Neural Network

Linear Classifiers

All the stuff that we learned so far (Linear and Logistic Regression) are linear classifiers

- Make predictions based on linear combinations of input features
- Decision boundaries are linear in the feature space (hyperplane in the n-dimensional feature space)

Linear Classifiers

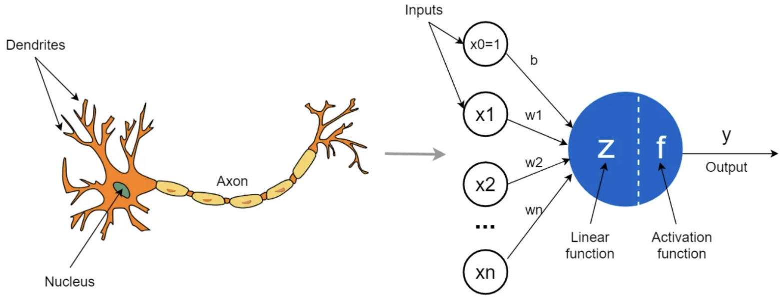
I thought all real world stuff is non-linear?

Linear Classifiers

I thought all real world stuff is non-linear?

Neural networks are excellent for capturing complex, non-linear relationships in data!

Perceptron

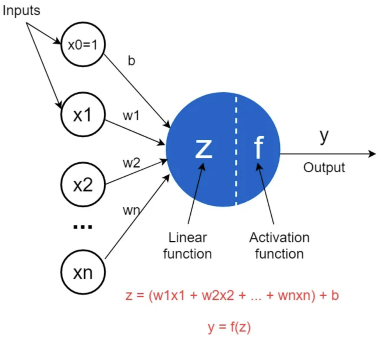


Left source and Right source

- Mimics biological neurons in the brain.
- Concept started in the 1940s, and got popularized in the 90s, and repopularized in the 2010s after GPU.

The structure of a perceptron

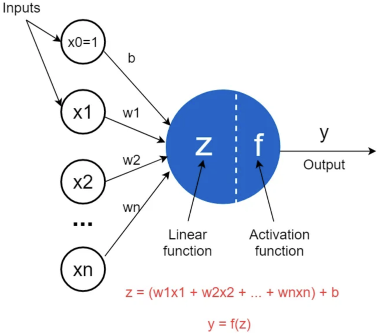
Inputs, Outputs, and Parameters



source

Inputs, Outputs, and Parameters

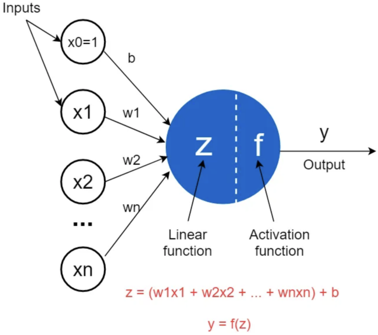
1. **Inputs**: $x_0, x_1, \ldots, _n$
   Raw data, or output from other neurons



source

Inputs, Outputs, and Parameters

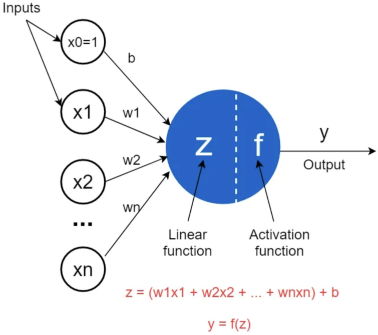
1. **Inputs**: $x_0, x_1, \dots, _n$
   Raw data, or output from other neurons
2. **Weights**: $w_1, w_2, \dots, w_n$
   Control the level of importance of each input -¿ higher the weight, more important the input.



source

Inputs, Outputs, and Parameters

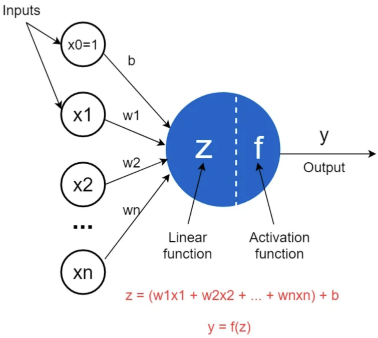
1. **Inputs**: $x_0, x_1, \ldots, {}_n$
   Raw data, or output from other neurons
2. Weights: $w_1, w_2, \ldots, w_n$
   Control the level of importance of each input -¿ higher the weight, more important the input.
3. **Bias**: b
   To ensure that the activation function does not get a zero value in case all $x1, \ldots, x_n$ are zero.



source

Inputs, Outputs, and Parameters

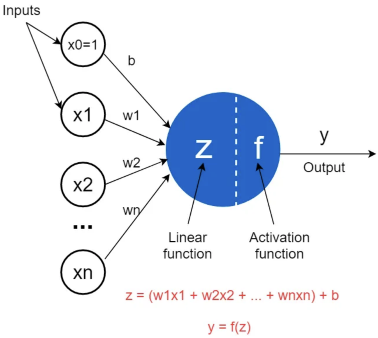
1. **Inputs**: $x_0, x_1, \ldots, _n$
   Raw data, or output from other neurons
2. **Weights**: $w_1, w_2, \ldots, w_n$
   Control the level of importance of each input -¿ higher the weight, more important the input.
3. **Bias**: b
   To ensure that the activation function does not get a zero value in case all $x1, \ldots, x_n$ are zero.
4. **Output**: $y = f(z)$



source

Functions

1. Linear Function:
   $z = w_1 x_1 + \ldots + w_n x_n$
   Weighted sum of inputs (linear combination).



source

Functions

1. Linear Function:
   $z = w_1x_1 + \ldots + w_nx_n$
   Weighted sum of inputs (linear combination).
2. Activation Function: f
   Non-linear component of a perceptron.

Linear Function

Linear Function of a Perceptron

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

where the weights w and bias b are parameters of the perceptron.

This is just like the model in a linear regression case!

Activation Function

The linear function of a perceptron is fed into a non linear function, called the activation function f, yielding the output y.

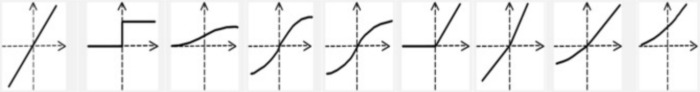
Activation Function									
	Identity	Binary Step	Logistic	TanH	ArcTan	ReLU	PreLU	ELU	SoftPlus
PLOT									
EQUATION	$f(x) = x$	$f(x) = \begin{cases} 0 \; for \; x<0 \\ 1 \; for \; x\geq 0 \end{cases}$	$f(x) = \frac{1}{1+e^{-x}}$	$f(x) = tangh(x) = \frac{2}{1+e^{-2x}} - 1$	$f(x) = tan^{-1}(x)$	$f(x) = \begin{cases} 0 \; for \; x<0 \\ x \; for \; x\geq 0 \end{cases}$	$f(x) = \begin{cases} \alpha x \; for \; x<0 \\ x \; for \; x\geq 0 \end{cases}$	$f(x) = \begin{cases} \alpha(e^{-x}-1) \; for \; x<0 \\ x \; for \; x\geq 0 \end{cases}$	$f(x) = log_e(1+e^x)$
DERIVATE	$f'(x) = 1$	$f'(x) = \begin{cases} 0 \; for \; x\neq 0 \\ ? \; for \; x=0 \end{cases}$	$f'(x) = f(x)(1-f(x))$	$f'(x) = 1-f(x)^2$	$f'(x) = \frac{1}{x^2+1}$	$f'(x) = \begin{cases} 0 \; for \; x<0 \\ 1 \; for \; x\geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha \; for \; x<0 \\ 1 \; for \; x\geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x)+\alpha \; for \; x<0 \\ 1 \; for \; x\geq 0 \end{cases}$	$f'(x) = \frac{1}{1+e^{-x}}$

Image from ScienceDirect

Activation Function

- Sigmoid and tanh are less and less used these days partly because their are bounded from both sides and the gradient of them vanishes as z goes to both positive and negative infinity.
- ReLU is the most commonly used activation function because calculation of its derivative is efficient, yielding to a much faster training time.
- Softplus is not used very often either in practice and can be viewed as a smoothing of the ReLU so that it has a proper second order derivative.
- GELU and leaky ReLU are both variants of ReLU but they have some non-zero gradient even when the input is negative.

Activation Function

What about a linear activation function? Why can't we choose a linear activation function?

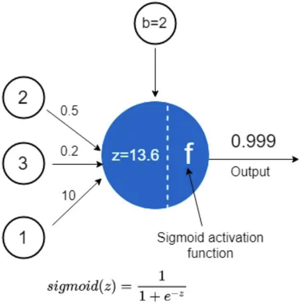
Activation Function

What about a linear activation function? Why can't we choose a linear activation function?

Applying a linear function to another linear function will result in a linear function over the original input. This loses much of the representational power of the neural network as often times the output we are trying to predict has a non-linear relationship with the inputs. Without non-linear activation functions, the neural network will simply perform linear regression.

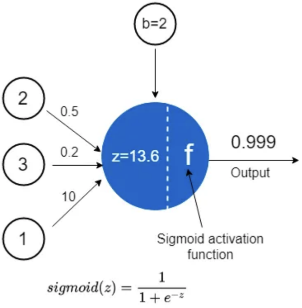
Forward-pass in a single neuron

Let's look at this example and solve the forward pass from input to output for this single neuron:



source

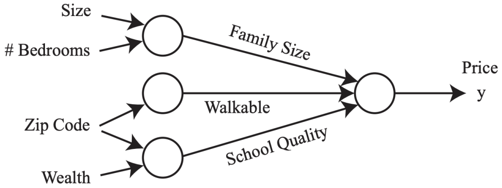
Forward-pass in a single neuron



Inputs: $x_1 = 2, x_2 = 3, x_3 = 1$
Weights: $w_1 = 0.5, w_2 = 0.2, w_3 = 10$
Bias: $b = 2$
Linear Function: $w_1x_1 + w_2x_2 + w_3x_3 + b = 13.6$
Activation Function: $f(z) = \frac{1}{1+e^{-z}}$
Output: $y = f(z) = 0.999$

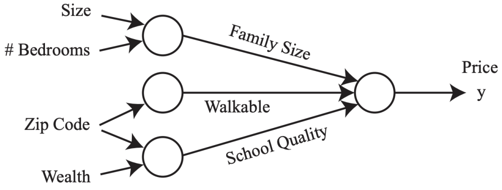
Stacking Neurons

We can stack neurons, taking the output of one neuron and passing it as input to another neuron, resulting in a more complex function.



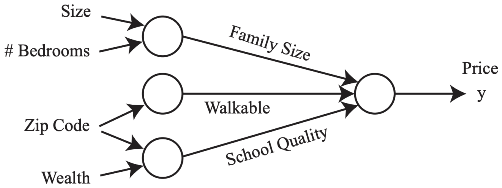
source

Stacking Neurons



1. **Input Features:** $x_1, x_2, x_3, x_4 =$ Size, Bedrooms, Zip Code, Wealth

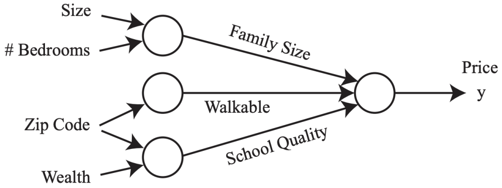
Stacking Neurons



1. **Output of the First Layer (Hidden Units):** $a_1, a_2, a_3 =$ Family Size, Walkable, School Quality
   Assuming ReLU activation function:

$$a_1 = \text{ReLU}(w^1_{11}x_1 + w^1_{12}x_2 + b^1_1)$$
$$a_2 = \text{ReLU}(w^1_{23}x_3 + b^1_2)$$
$$a_3 = \text{ReLU}(w^1_{33}x_3 + w^1_{34}x_4 + b^1_3)$$

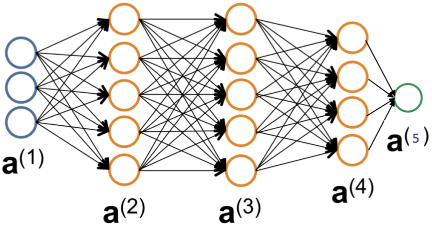
Stacking Neurons



1. **Output of the Second Layer (Final Output):** y = Price
   Assuming sigmoid activation function:

$$y = h_w(x) = \text{sigmoid}(w^2_{11}a_1 + w^2_{12}a_2 + w^2_{13}a_3 + b^2_1)$$

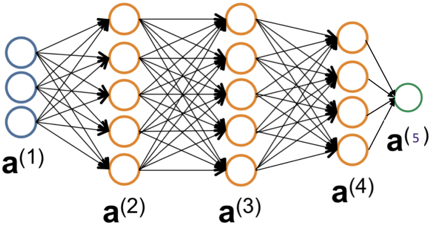
Multi-layer Neural Network



source

By expressing inputs x, weights w, hidden layers activation a, and outputs y as vectors, we can write down the equations for a forward pass in a neural network.

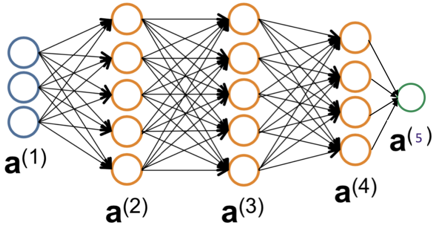
Multi-layer Neural Network



where:
$z_i^{(k)}$ = Output of unit i in layer k after going through the linear function prior to the activation function
$a_i^{(k)}$ = Output of the activation layer of unit i in layer k
$w_{ij}^{(k)}$ = Weight of connection from unit i in layer k to unit j in layer $k-1$

If in layer k there are m_k units of neurons, $a^{(k)} \in \mathbf{R}^{m_k}$, $w^{(k)} \in \mathbf{R}^{m_k \times m_{k-1}}$, $b^{(k)} \in \mathbf{R}^{m_k}$

Forward Pass



$$a^{(1)} = x \quad \text{(Input layer, no activation function)}$$
$$z^{(2)} = w^{(1)} \cdot a^{(1)} + b^{(1)}$$
$$a^{(2)} = f(z^{(2)}) = f(w^{(2)} \cdot a^{(1)} + b^{(1)})$$
and so on

References

Chris Re, Andrew Ng, and Tengyu Ma (2023)
CSE229 Machine Learning
Stanford University