Intro to Deep Learning

Big Data y Machine Learning para Economía Aplicada

Ignacio Sarmiento-Barbieri

Universidad de los Andes

Agenda

Deep Learning: Intro

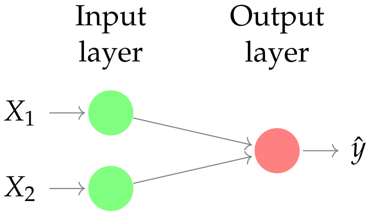
- Linear Models may miss the nonlinearities that best approximate $f^*(x)$
- Neural networks are simple models.
- The model has **linear combinations** of inputs that are passed through **nonlinear activation functions** called nodes (or, in reference to the human brain, neurons).

Deep Learning: Intro

- Let's start with a familiar and simple model, the linear model

Deep Learning: Intro

- Let's start with a familiar and simple model, the linear model



Single Layer Neural Networks

- A neural network takes an input vector of p variables

$$X = (X_1, X_2, ..., X_p) \tag{1}$$

- and builds a nonlinear function $f(X)$ to predict the response y .

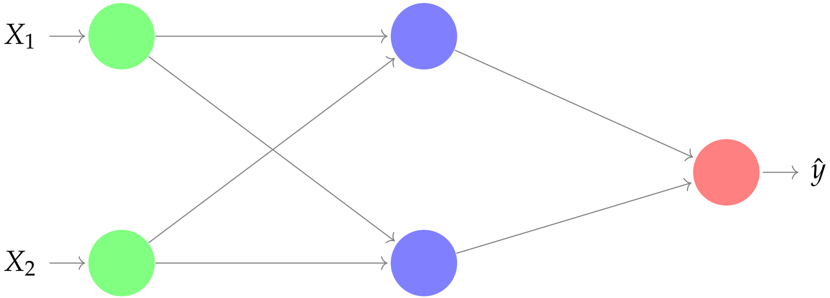
$$y = f(X) + u \tag{2}$$

- The Single layer NN model has the form

$$f(X) = f^{(output)}(g(X)) \tag{3}$$

- where g is the activation function in the hidden unit
- the second layer, $f^{(output)}$ is the output layer of the network

Single Layer Neural Networks



Single Layer Neural Networks

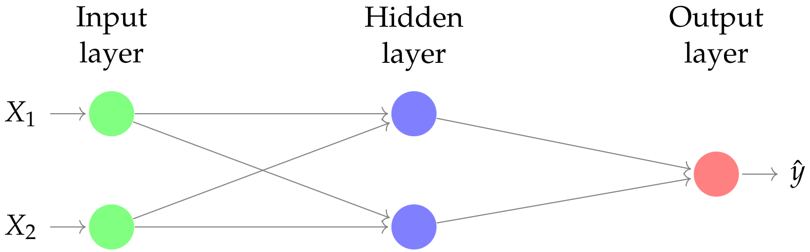
- NN are made of **linear combinations** of inputs that are passed through **nonlinear activation functions**

Worked Example I: Single Layer Neural Networks

- 2 Predictors: $p = 2$, $X = (X_1, X_2)$
- 2 Nodes: $K = 2$, $A_1(X)$ and $A_2(X)$
- Non-linear activation function $g(z) = z^2$
- Want to predict a number $\in \mathbb{R}$: identity output function ($f^{(output)} : \mathbb{R} \to \mathbb{R}$ such that $f^{(output)}(x) = x$)

Worked Example I: Single Layer Neural Networks

- 2 Predictors: $p = 2$, $X = (X_1, X_2)$
- 2 Nodes: $K = 2$, $A_1(X)$ and $A_2(X)$
- Non-linear activation function $g(z) = z^2$
- Want to predict a number $\in \mathbb{R}$: identity output function ($f^{(output)} : \mathbb{R} \to \mathbb{R}$ such that $f^{(output)}(x) = x$)



Worked Example I: Single Layer Neural Networks

- 2 Predictors: $p = 2$, $X = (X_1, X_2)$
- 2 Nodes: $K = 2$, $A_1(X)$ and $A_2(X)$
- Non-linear activation function $g(z) = z^2$
- Want to predict a number $\in \mathbb{R}$: identity output function ($f^{(output)} : \mathbb{R} \to \mathbb{R}$ such that $f^{(output)}(x) = x$)

$$f(X) = f^{(output)}(g(X)) \tag{4}$$

Worked Example I: Single Layer Neural Networks

- 2 Predictors: $p = 2$, $X = (X_1, X_2)$
- 2 Nodes: $K = 2$, $A_1(X)$ and $A_2(X)$
- Non-linear activation function $g(z) = z^2$
- Want to predict a number $\in \mathbb{R}$: identity output function ($f^{(output)} : \mathbb{R} \to \mathbb{R}$ such that $f^{(output)}(x) = x$)

$$f(X) = f^{(output)}(g(X)) \tag{4}$$

$$f(X) = \beta_0 + \sum_{k=1}^{2} \beta_k \left(w_{k0} + \sum_{j=1}^{2} w_{kj} X_j \right)^2 \tag{5}$$

Why not linear activation functions?

Worked Example II : The ”Exclusive OR (XOR)” Function

- The exclusive disjunction of a pair of propositions, (p, q), is supposed to mean that p is true or q is true, but not both
- It’s truth table is:

q	p	y
0	0	0
0	1	1
1	0	1
1	1	0

Agenda

Activation Functions

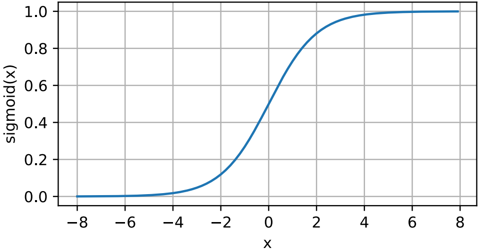
Sigmoid Function (Logit)

- The sigmoid function transforms its inputs, for which values lie in the domain $\mathbb{R}$, to outputs that lie on the interval (0, 1).
- For that reason, the sigmoid is often called a squashing function: it squashes any input in the range (-inf, inf) to some value in the range (0, 1):

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}.$$

Activation Functions

Sigmoid Function (Logit)



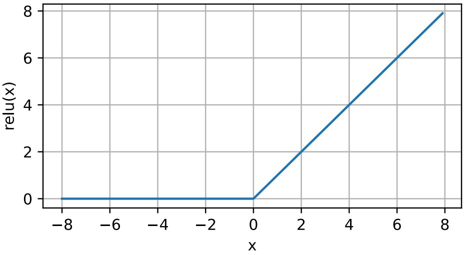
- When attention shifted to gradient based learning, the sigmoid function was a natural choice because it is smooth and differentiable.

- ReLU Function
  - The most popular choice, due to both simplicity of implementation and its good performance on a variety of predictive tasks, is the rectified linear unit (ReLU).
  - ReLU provides a very simple nonlinear transformation. Given an element x, the function is defined as the maximum of that element and 0:

$$\text{ReLU}(x) = \max\{x, 0\}$$

Activation Functions

- ReLU function retains only positive elements and discards all negative elements by setting them to 0.
- It is piecewise linear.



Neural Networks: Activation Functions

- Sigmoid$(x) = \frac{1}{1+\exp(-x)}$
- ReLU$(x) = \max\{x, 0\}$
- Among others (see more here)
- Hidden unit design remains an active area of research, and many useful hidden unit types remain to be discovered

Agenda

Output Functions

- The choice of output unit is related to the problem at hand
  - Regression
  - Classification
    - Binary
    - Multiclass

Agenda

Training the network

- El objetivo es

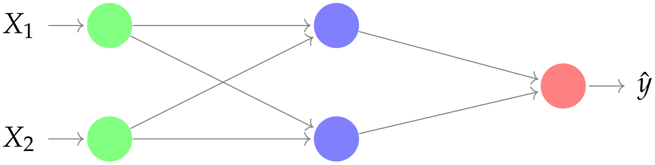
$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} L(y, f(X;\Theta)) \right\} \tag{6}$$

- SNN

$$f(X,\beta,w) = f\left[\beta_0 + \sum_{k=1}^{K} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right)\right] \tag{7}$$

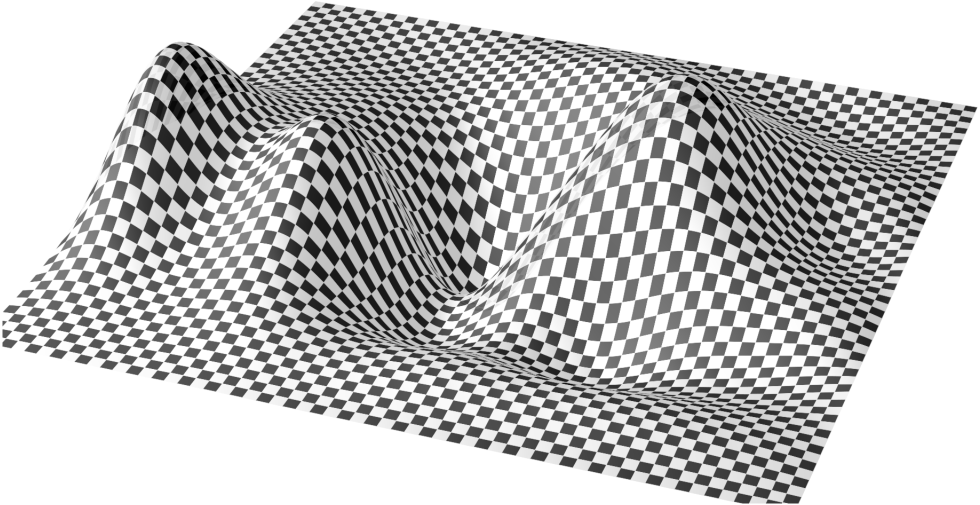
Training the network

Example: House Prices

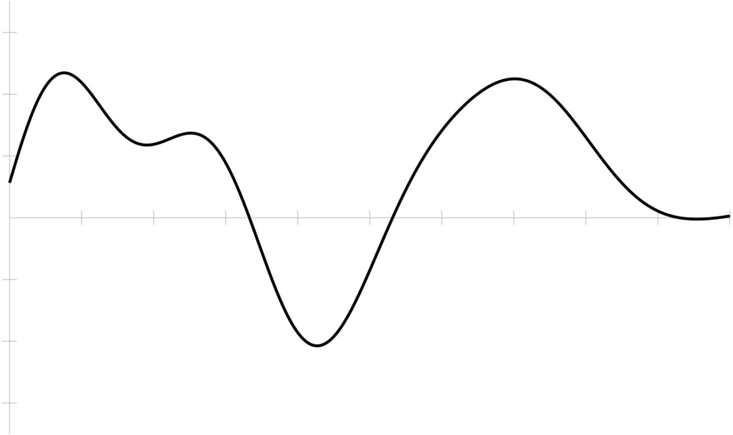


- Equations
  - Hidden Layer sigmoid (logistic):
    - $A_1 = \sigma(w_{11} \cdot X_1 + w_{12} \cdot X_2 + w_{10})$
    - $A_2 = \sigma(w_{21} \cdot X_1 + w_{22} \cdot X_2 + w_{20})$
  - Output Layer, identity output function:
    - $\hat{y}_i = \beta_0 + \beta_1 \cdot A_1 + \beta_2 \cdot A_2$
- Loss Function $\Rightarrow$ MSE: $\frac{1}{n} \sum_i^n (y_i - \hat{y}_i)^2$

Training the network



Training the network



Training the network

Example: House Prices