Intro to Deep Learning

Big Data y Machine Learning para Economía Aplicada

Ignacio Sarmiento-Barbieri

Universidad de los Andes

Agenda

Deep Learning: Intro

- Neural networks are simple models.
- Their strength lays in their simplicity
- 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

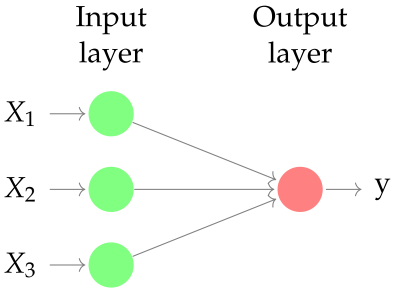
- Let's start with a familiar and simple model, the linear model

$$
\begin{aligned}
y &= f(X) + u \\
y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + u
\end{aligned} \tag{1}
$$

Deep Learning: Intro

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

$$
\begin{aligned}
y &= f(X) + u \\
y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + u
\end{aligned}
\tag{1}
$$



Agenda

Single Layer Neural Networks

- Linear Models may miss the nonlinearities that best approximate $f^*(x)$
- We can overcome these limitations of linear models and handle a more general class of functions by incorporating hidden layers.
- Neural Networks are also called deep feedforward networks, feedforward neural networks, or multilayer perceptrons (MLPs), and are the quintessential deep learning models

Single Layer Neural Networks

- A neural network takes an input vector of p variables

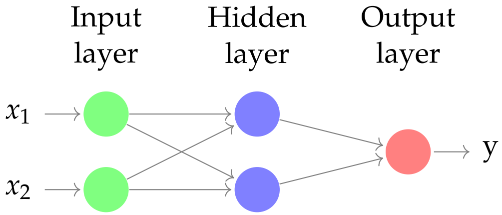
$$X = (X_1, X_2, ..., X_p) \tag{2}$$

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

$$y = f(X) + u \tag{3}$$

- What distinguishes neural networks from previous methods is the particular structure of the model.

Single Layer Neural Networks



Single Layer Neural Networks

- The NN model has the form

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) \tag{4}$$

$$= \beta_0 + \sum_{k=1}^{K} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right) \tag{5}$$

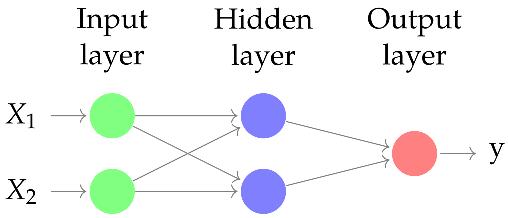
- where $g(.)$ is a activiation function specified in advance
- where the nonlinearity of $g(.)$ is essential

Worked Example I: Single Layer Neural Networks

- $p = 2$, $X = (X_1, X_2)$
- $K = 2$, $h_1(X)$ and $h_2(X)$
- $g(z) = z^2$

Worked Example I: Single Layer Neural Networks

- $p = 2$, $X = (X_1, X_2)$
- $K = 2$, $h_1(X)$ and $h_2(X)$
- $g(z) = z^2$



Worked Example I: Single Layer Neural Networks

$$f(X) = \beta_0 + \sum_{k=1}^{2} \beta_k g\left(w_{k0} + \sum_{j=1}^{2} w_{kj} X_j\right) \tag{6}$$

- Suppose we get

$$\begin{array}{lll}
\hat{\beta}_0 = 0 & \hat{\beta}_1 = \frac{1}{4} & \hat{\beta}_2 = -\frac{1}{4} \\
\hat{w}_{10} = 0 & \hat{w}_{11} = 1 & \hat{w}_{12} = 1 \\
\hat{w}_{20} = 0 & \hat{w}_{21} = 1 & \hat{w}_{22} = -1
\end{array}$$

NN Minimalist Theory

- Why not a linear activation functions?
- Let's go back to our example
  - $p = 2$, $X = (X_1, X_2)$
  - $K = 2$, $h_1(X)$ and $h_2(X)$
  - Now $g(z) = z$
- Then

$$
\begin{aligned}
f(X) &= \beta_0 + \sum_{k=1}^{2} \beta_k A_k &(7)\\
&= \beta_0 + \sum_{k=1}^{2} \beta_k h_k(X) &(8)\\
&= \beta_0 + \sum_{k=1}^{2} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right) &(9)
\end{aligned}
$$

NN Minimalist Theory

Why not a linear activation functions?

- Since $g(z) = z$ we get

$$f(X) = \beta_0 + \sum_{k=1}^{2} \beta_k \left(w_{k0} + \sum_{j=1}^{2} w_{kj} X_j \right) \tag{10}$$

- Replacing

$$f(X) = \beta_0 + \beta_1 \left(w_{10} + w_{11} X_1 + w_{12} X_2 \right) + \beta_2 \left(w_{20} + w_{21} X_1 + w_{22} X_2 \right) \tag{11}$$

- then

$$f(X) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 \tag{12}$$

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	q v p
0	0	0
0	1	1
1	0	1
1	1	0

- When exactly one of these binary values is equal to 1, the XOR function returns 1. Otherwise, it returns 0

Worked Example: The "Exclusive OR (XOR)" Function

- Let's use a linear model

$$y = \beta_0 + \beta_1 q + \beta_2 p + u \tag{13}$$

$$y = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} X = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} 1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{14}$$

- Solution $\beta_0 = \frac{1}{2}$, $\beta_1 = 0$, $\beta_2 = 0$
- Prediction $\hat{y} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$

Worked Example: The "Exclusive OR (XOR)" Function

- Let's use Single Layer NN containing two hidden units
- Activation Funcition: ReLU: $g(z) = max\{0, z\}$
- NN

$$f(X) = \beta_0 + \sum_{k=1}^{2} \beta_k g\left(w_{k0} + \sum_{j=1}^{2} w_{kj} X_j\right) \tag{15}$$

Worked Example: The "Exclusive OR (XOR)" Function

- Suppose this is the solution to the XOR problem

$$f(x) = max\{0, XW + W_0\}\,\beta + \beta_0$$

$$W = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$W_0 = \begin{pmatrix} 0 & -1 \\ 0 & -1 \\ 0 & -1 \\ 0 & -1 \end{pmatrix}$$

$$\beta = \begin{pmatrix} 1 & -2 \end{pmatrix}$$

$$\beta_0 = 0$$

Worked Example: The "Exclusive OR (XOR)" Function

- Lets work out the example step by step

$$f(x) = max\{0, XW + W_0\}\,\beta + \beta_0 \tag{16}$$

Worked Example: The "Exclusive OR (XOR)" Function

- In this example, we simply specified the solution, then showed that it obtained zero error.
- In a real situation, obviously we can't guess the solution
- What we do is to estimate the parameters via optimization methods
- All gain comes from using nonlinear activation function

Example: XOR

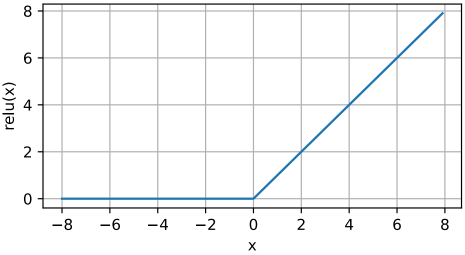
Agenda

- 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:

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

Activation Functions

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



Activation Functions

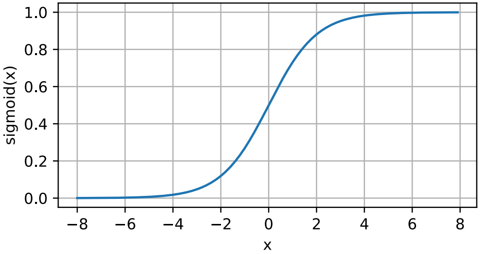
Sigmoid Function (Logit)

- 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)



- In the earliest neural networks, scientists were interested in modeling biological neurons which either fire or do not fire. Thus the pioneers of this field, going all the way back to McCulloch and Pitts, the inventors of the artificial neuron, focused on thresholding units.
- A thresholding activation takes value 0 when its input is below some threshold and value 1 when the input exceeds the threshold.
- When attention shifted to gradient based learning, the sigmoid function was a natural choice because it is a smooth, differentiable approximation to a thresholding unit.

Activation Functions

Tanh Function

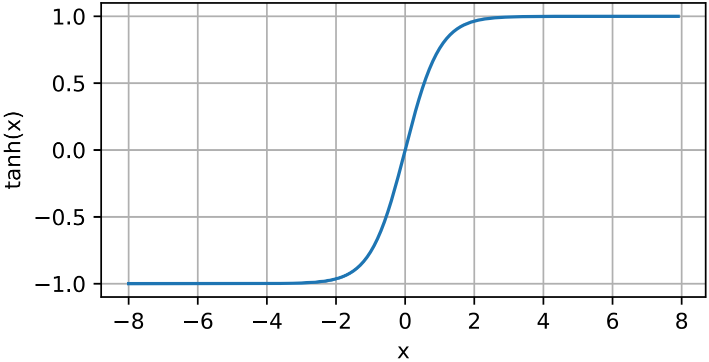
- Like the sigmoid function, the tanh (hyperbolic tangent) function also squashes its inputs, transforming them into elements on the interval between -1 and 1:

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}.$$

- Although the shape of the function is similar to that of the sigmoid function, the *tanh* function exhibits point symmetry about the origin of the coordinate system.

Activation Functions

Tanh Function



Activation Functions

- Other Activation functions
  - $h = cos(Wx + b)$ Goodfellow et al. (2016) claim that on the MNIST dataset they obtained an error rate of less than 1 percent
  - Radial basis function (RBF): $exp\left(\frac{1}{\sigma^2)}||W - x||^2\right)$
  - Softplus: $log(1 + e^x)$
  - Hard tanh: $max(-1, min(1, x))$
- 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 cost function is tightly coupled with the choice of output unit.
- Most of the time, we simply use the distance between the data distribution and the model distribution.
  - Linear $y = \beta_0 + \sum_{k=1}^{K} \beta_k h_k \to \mathbb{R}$
  - Sigmoid (Logistic) $\to$ classification $\{0, 1\}$
  - Softmax $\to$ classification multiple categories

Agenda

Example: MNIST

Agenda

Architecture Design

- Another key design consideration for neural networks is determining the architecture.
- The word architecture refers to the overall structure of the network: how many units it should have and how these units should be connected to each other.
- The universal approximation theorem guarantees that regardless of what function we are trying to learn, a sufficiently large MLP will be able to represent this function.

Architecture Design

- The universal approximation theorem (Hornik et al., 1989; Cybenko, 1989) states that:
  - A feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function (such as the logistic sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired nonzero amount of error, provided that the network is given enough hidden units.

Architecture Design

- We are not guaranteed, however, that the training algorithm will be able to learn that function.
- Even if the network is able to represent the function, learning can fail for two different reasons.
  1. The optimization algorithm used for training may not be able to find the value of the parameters that corresponds to the desired function.
  2. The training algorithm might choose the wrong function as a result of overfitting

Architecture Design

- A feedforward network with a single layer is sufficient to represent any function, but the layer may be infeasible large and may fail to learn and generalize correctly.
- In many circumstances, using deeper models can reduce the number of units required to represent the desired function and can reduce the amount of generalization error.
- The ideal network architecture for a task must be found via experimentation guided by monitoring the validation set error

Agenda

Multilayer Neural Networks

- Modern neural networks typically have more than one hidden layer, and often many units per layer.
- In theory a single hidden layer with a large number of units has the ability to approximate most functions.
- However, the learning task of discovering a good solution is made much easier with multiple layers each of modest size.

Multilayer Neural Networks: MNIST Digits

Agenda

Network Tuning

- Training networks requires a number of choices that all have an effect on the performance:
  - The number of hidden layers,
  - The number of units per layer
  - Regularization tuning parameters
  - Details of stochastic gradient descent.
- This is an active research area that involves a lot of trial and error, and overfitting is a latent danger at each step.

Agenda

When to Use Deep Learning?

- The performance of deep learning ussually is very impressive.
- The question that then begs an answer is: should we discard all our older tools, and use deep learning on every problem with data?
- Let's see an example

When to Use Deep Learning?