Alexandre St-Aubin

An Introduction to Machine Learning

FEBRUARY 21, 2024

McGill University – Directed Reading Program

Contents

1 Introduction 3

2 Neural Networks 3

2.1 Neurons 3

2.2 Weights 4

2.3 Layers 4

2.4 Activation Function 4

3 Learning 7

3.1 Cost Function 7

3.2 BackPropagation 7

1 Introduction

There are several types of Machine Learning algorithms. The main categories are divided into *Supervised learning, Semi-supervised learning, Unsupervised learning and Reinforcement learning.* Here, we will focus on *Supervised learning.*

Definition 1.1 (Supervised Learning). In supervised learning, the machine learns through examples. The machine learning algorithm is tasked with developing the strategy for achieving the specified outputs given some input. To do so, a known dataset is supplied that contains a set of inputs and associated outputs. The algorithm finds patterns in the data, learns from observations, and makes predictions. At the end of the learning process, the algorithm can be tested on data that is unknown to it, but that the operator knows the answers to, in order to test its accuracy. Then, the algorithm is adjusted if needed, and the process continues in an iterated manner.

talk about types of ML problems (classification, regression)

2 Neural Networks

Neural networks, a subset of machine learning, form the core of supervised learning and are inspired by the human brain[1].They are made of input, hidden, and output layers, and consist of interconnected *neurons* with associated *weights and thresholds*. Activated nodes transmit data to the next layer if their output surpasses the threshold. The overall network is a combination of function composition and matrix multiplication:

$$g(x) = f^L(W^L f^{L-1}(W^{L-1} \dots f^1(W^1 x) \dots)),$$

where L is the number of layers (2.3), W is the weight matrix (2.2), and f is the activation function (2.4).

[1] IBM. What are neural networks, 2024. URL `https://www.ibm.com/topics/neural-networks`. Accessed 7 Febuary 2024

2.1 Neurons

Definition 2.1 (Neurons). Neurons are the building blocks of neural networks. They receive an input, multiply it by a weight, and output it using an activation function. A simple mathematical model for a neuron's output activation is[2]

$$a_j = f\left(\sum_{i=0}^{n} w_{i,j} a_i\right)$$

where f is the activation function (2.4), a_i is the output activation of neuron i and $w_{i,j}$ is the weight on the link from neuron i to a_j. We

[2] Peter Norvig Stuart Russel. *Artificial Intelligence, A Modern Approach*, chapter 18. Pearson Education, New Jersey, 3rd edition, 2010

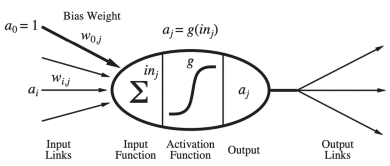

Figure 1: A neuron and its inputs and outputs.

will explore activation functions in more depth shortly.

Definition 2.2 (Artificial Neural Networks).

2.2 *Weights*

2.3 *Layers*

The layers are divided in 3 groups: the *input, hidden,* and *output* layers. See figure 2.3 for an image of the layers. Layers are a general term given to a grouping of neurons that act together in the neural network.

Definition 2.3 (Layers).

Input Layer. The first layer in a neural network, it receives the initial data for the network from the outside world. The "Entry point" of the neural network.

Hidden layers. The hidden layer(s) are where the magic happens in neural networks. Each layer is trying to learn different aspects about the data by minimizing the **cost function**. The most intuitive way to understand these layers is in the context of 'image recognition' such as a face. The first layer may learn edge detection, the second may detect eyes, third a nose, etc.[3].

Output Layer. The final layer in the neural network is the output layer. This layer is responsible for holding the final result or output of the problem. Input, such as raw images, is fed into the input layer, and the output layer produces the corresponding result.

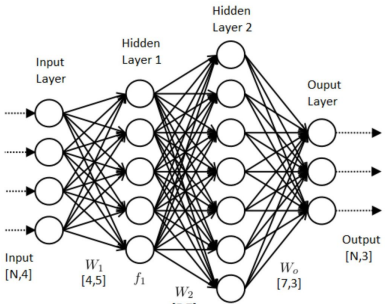

Figure 2: A graph representation of a Neural Network.

[3] cdeterman. what is a 'layer' in a neural network. Stack Overflow. URL `https://stackoverflow.com/questions/35345191/what-is-a-layer-in-a-neural-network/35347548#35347548`

2.4 *Activation Function*

An activation function in neural networks is a smooth function applied to the output of each neuron in a layer. It introduces non-linearity to the network, allowing it to learn complex patterns and relationships in the data.

The activation function determines whether a neuron should be activated or not based on the weighted sum of its inputs. In other words, it defines the output of a neuron given a set of inputs. Without activation functions, neural networks would be limited to linear transformations, and they wouldn't be able to capture the non-linearities present in many real-world datasets.

Desired Characteristics of Activation Functions [4]

There is no universal rule for choosing the best activation function, but there are some characteristics to look for, namely

[4] Ameya D. Jagtap and George Em Karniadakis. How important are activation functions in regression and classification? a survey, performance comparison, and future directions, 2022

Nonlinearity One of the most essential characteristics of an activation function is nonlinearity. In comparison to linear activation functions, the non-linearity of the activation function significantly improves the learning capability of neural networks.

Computationally Cheap The activation function must be easy to evaluate in terms of computation. This has the potential to greatly improve network efficiency.

Finite Range/Boundedness Gradient-based training approaches are more stable when the range of the activation function is finite, because pattern presentations significantly affect only limited weights.

Differentiability The most desirable quality for using gradient-based optimization approaches is continuously differentiable activation functions. This ensures that the back-propagation algorithm works properly.

Remark 2.4. The vanishing and exploding gradient problems: The variation of the inputs and outputs of some activation functions, such as the logistic function (Sigmoid), is extremely large. To put it another way, they reduce and transform a bigger input space into a smaller output space that falls between [0,1]. As a result, the back-propagation algorithm has almost no gradients to propagate backward in the network, and any residual gradients that do exist continue to dilute as the program goes down through the top layers. Due to this, the initial hidden-layers are left with no information about the gradients. For hyperbolic tangent and sigmoid activation functions, it has been observed that the saturation region for large input (both positive and negative) is a major reason behind the vanishing of gradient. One of the important remedies to this problem is the use of non-saturating activation functions, such as ReLU.

We present some commonly used activation functions.

Sigmoid Function. Its range is $[0, 1]$, and is defined as,

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

Advantage: boundedness.
Disadvantages: the vanishing gradient problem, the output not being zerocentered, and the saturation for large input values.

Hyperbolic Tangent Function. It is mostly used for regression problems, has a range of $[-1, 1]$, and is defined as,

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

Advantage: zerocentered structure.
Disadvantage: the vanishing gradient problem, i.e. once saturated, it is really challenging for the learning algorithm to adapt the parameters and learn faster.

ReLU Function. ReLU was primarily used to overcome the vanishing gradient problem. ReLU is the most common activation function used for *classification problems*. Its range is $[0, \infty)$, and is defined as

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

Advantages: Apart from overcoming the vanishing gradient problem, the implementation of ReLU is very easy and thus cheaper, unlike tanh and sigmoid, where an exponential function is needed.
Disadvantages: It has a saturation region, which can prevent the learning of the networks. In particular, ReLU always discards the negative values, which makes the neurons stop responding to the gradient-based optimizer. This problem is known as *dead or dying ReLU problem*, meaning the neurons stop outputting other than zero.

Softplus Function. It approximates the ReLU activation function in a smooth way, with a range of $(0, \infty)$, and it is defined as

$$\text{Softplus}(x) = \ln(1 + e^x)$$

Softmax It is a generalization of logistic function in high dimensions. It normalizes the output and divides it by its sum, which forms a probability distribution. The standard softmax function Softmax: $\mathbb{R}^k \to (0,1)^k$ is defined as

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \quad \text{for } i = 1, ..., k$$

In other words, it applies the standard exponential function to each element x_i of the input vector x and normalizes these values by dividing them by the sum of all these exponentials, which ensures that the sum of the components of the output vector is 1.

The following is a table summarizing the information above.

Activation Function ($\Phi(\mathbf{x})$)	**Derivatives ($\Phi'(\mathbf{x})$)**	**Range**	**Continutity**
Linear : $\Phi(x) = x$	$\Phi'(x) = 1$	$(-\infty, \infty)$	C^∞
Step : $\Phi(x) = \begin{cases} 0 & x \leq 0, \\ 1 & x > 0 \end{cases}$	$\Phi'(x) = \begin{cases} 0 & x \neq 0, \\ \text{Not defined} & x = 0 \end{cases}$	$(0, 1)$	C^{-1}
Sigmoid or Logistic : $\Phi(x) = \frac{1}{1+e^{-x}}$	$\Phi'(x) = \Phi(x)(1 - \Phi(x))$	$(0, 1)$	C^∞
Rectifier Unit (ReLU) : $\Phi(x) = \begin{cases} 0 & x \leq 0, \\ x & x > 0 \end{cases}$	$\Phi'(x) = \begin{cases} 0 & x < 0, \\ 1 & x > 0, \\ \text{Not defined} & x = 0 \end{cases}$	$[0, \infty)$	C^0
Hyperbolic Tangent : $\Phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\Phi'(x) = 1 - \Phi(x)^2$	$(-1, 1)$	C^∞
Softplus : $\Phi(x) = \ln(1 + e^x)$	$\Phi'(x) =$ Sigmoid	$(0, \infty)$	C^∞
Leaky Rectifier Unit (Leaky ReLU) : $\Phi(x) = \begin{cases} 0.01x & x < 0, \\ x & x \geq 0 \end{cases}$	$\Phi'(x) = \begin{cases} 0.01 & x < 0, \\ 1 & x > 0, \\ \text{Not defined} & x = 0 \end{cases}$	$(-\infty, \infty)$	C^0
Exponential Linear Unit (ELU) : $\Phi(x) = \begin{cases} \alpha(e^x - 1) & x \leq 0, \\ x & x > 0 \end{cases}$	$\Phi'(x) = \begin{cases} \alpha e^x & x < 0, \\ 1 & x > 0, \\ \alpha & x = 0 \end{cases}$	$(-\alpha, \infty)$	$\begin{cases} C^1 & \text{If} \alpha = 1 \\ C^0 & \text{Otherwise} \end{cases}$
Gaussian : $\Phi(x) = e^{-x^2}$	$\Phi'(x) = -2x\Phi(x)$	$(0, 1]$	C^∞
Swish ($\beta = 1$) : $\Phi(x) = x \cdot$ Sigmoid	$\Phi'(x) = x \cdot \Phi(x)(1 - \Phi(x)) +$ Sigmoid	$[0, .\infty)$	C^∞
Oscillatory : $\Phi(x) = \sin(x)$	$\Phi'(x) = -\cos(x)$	$[-1, 1]$	C^∞

Figure 3: Common activation functions, their derivatives, range, and order of continuity.

3 *Learning*

3.1 *Cost Function*

Here, we present some common loss functions[5]

3.2 *BackPropagation*

[5] Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. A survey and taxonomy of loss functions in machine learning, 2023

References

cdeterman. what is a 'layer' in a neural network. Stack Overflow. URL https://stackoverflow.com/questions/35345191/what-is-a-layer-in-a-neural-network/35347548#35347548.

Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. A survey and taxonomy of loss functions in machine learning, 2023.

IBM. What are neural networks, 2024. URL https://www.ibm.com/topics/neural-networks. Accessed 7 Febuary 2024.

Ameya D. Jagtap and George Em Karniadakis. How important are activation functions in regression and classification? a survey, performance comparison, and future directions, 2022.

Peter Norvig Stuart Russel. *Artificial Intelligence, A Modern Approach*, chapter 18. Pearson Education, New Jersey, 3rd edition, 2010.