Neural Networks and Deep Learning *

Mohamed Hesham Ibrahim Abdalla[1] [0000−1111−2222−3333], Second Author[1] [1111−2222−3333−4444], and Third Author[1] [2222−−3333−4444−5555]

German University in Cairo, Cairo, Egypt

Abstract. The abstract should briefly summarize the contents of the paper in 15–250 words.

Keywords: Neural Networks · Deep Learning · Machine Learning

1 Introduction

Artificial Neural Networks (ANNs) and preceptrons are intelligent units that have taken inspiration from biology, especially the brain (cite). ANNs work by taking labeled inputs and then trying to find a mathematical rule or function to systematically answer the question of which label belongs to which input, and later identify labels of new inputs that have never been seen before by the network. For example, inputs could be human images and the labels are the gender of the human in that particular image.

The history of ANNs and preceptrons goes back to the 50's and the 60's when the first known preceptron was created. The first preceptron was simulated on an IBM 704 computer at Cornell Aeronautical Laboratory in 1957 (cite). It works by giving(cont)

2 Mathematical Background and Concept

ANNs learn how to accomplish a task by following two main steps: forward propagation and backward propagation. Forward propagation is the process of predicting labels and computing how deviated these labels from the ones provided in the input data. On the other hand, backward propagation tries to correct the predictions by minimizing the diffrence between the input labels and the predicted labels.

2.1 Forward Propagation

ANNs consist of layers where each layer has neurons which are connected via links (Figure 2.1). Each neuron gets an input and produces an output. Each input is given a certain weight which makes that specific input has more or less priority in controlling the output of the neuron.

* Supported by organization x.

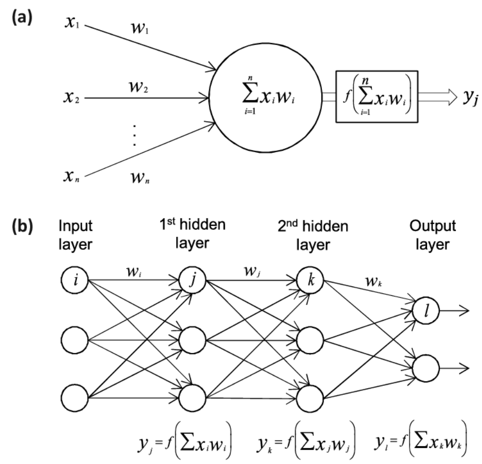


Fig. 1. (a) A single-layer ANN where the first layer has only one neuron. (b) A multi-layer ANN with 2 hidden layers.

Neurons calculate their output by multiplying their inputs by their weights and applying a bias to the multiplication. Eqaution 1 shows a linear mapping of a single input.

$$z^i = w^T x^i + b \tag{1}$$

where z^i is the linear mapping of the ith example, w^T in <- R^{1xN} is the weight vector of the form $[w_1, w_2, ...w_N]$ and x^i in <- R^{Nx1} is the input vector of the form $[x_1, x_2, ...x_N]$ of the ith example.

This mapping is then forwaded to an activation function (Eqaution 2). Action functions are used to limit the linear transformation output.

$$a^i = g(z^i) \tag{2}$$

where a^i is the output of the activation function (unit activation) on the linear mapping of the ith example.

To avoid manaully looping over each example, all of the variables are used as vectors.(add the l's correctly in the eqts)

$$Z^{[l]} = W^{T[l]} X + b^{[l]} \tag{3}$$

$$A = g(Z) \tag{4}$$

The choice of the activation function vary depending on the given data (inputs). In general, there are three main categories of the activation functions: binary, linear and non-linear. Binary or threshold functions output a binary value depending on the input. For example, the step function produces a +1 in case z^i is greater than or eqaul to 0 and -1 otherwise (Eqaution 5). Binary functions can not deal with categorical data, therefore they are not widely used.

$$f(x) = \begin{cases} +1 & z^i \leq 0 \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

Another type of activation functions is linear. Linear functions forward the input directly to the output without any transformation. This is useful in problems where the output is continuous. For example, predicting house prices.

Although all of the previous functions are useful for some sitautions, they fail to find a pattern if the data is non-linearly separable since the function will only be able to draw a linear decesion boundary that can devide the data into two groups. Therefore, other functions are used to find non-linear seprations between the data such as sigmoid, ReLU and tanh/ Hyperbolic Tangent.

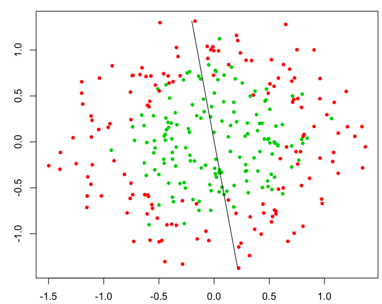


Fig. 2. Example of a linear activation function on non linearly separable data https://www.r-bloggers.com/interactive-visualization-of-non-linear-logistic-regression-decision-boundaries-with-shiny/

After calculating the activation unit for the inputs, a loss function L is calculated for each prediction to measure how different the prediction is from the real data. the formula of the loss function varies based on the type of the labels. For example, if the labels are binary values (consisting of two classes), Log Loss could be used. this loss function . Alternatively, if the labels are continuous, Mean Absolute Error (MAS) or Mean Squared Error (MSE) are used(?).

2.2 Backward Propagation

Backward propagation is the process of tunning the input weights of each neuron to correctly predict new labels. In other words, finding the weights that minimize the loss function. These weights are calculated through an algorithm called gradient descent [1]. Gradient descent is the process of iteratively updating the parameters of a function (input weights) in the direction of the steepest descent until a local minima is reached. To give a better idea, we can imagine the parameter space as a surface from where we follow the direction of the slope downhill until we reach a valley (Figure 2.2). Moreover, the slope is calculated by finding the first partial derivative of a function J in respect to every parameter of the function (Eqaution 7).

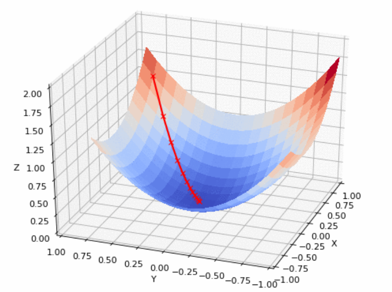


Fig. 3. Gradient descent visualization https://docs.paperspace.com/machine-learning/wiki/gradient-descent

$$w_i = w_i - \eta \frac{\partial J(W)}{\partial w_i} \tag{6}$$

in case of a multi-layer ANN with L layers, the gradient is instead calculated with the chain rule where the partial derivative is applied on each of the hidden layers till the first layer is reached(caption).

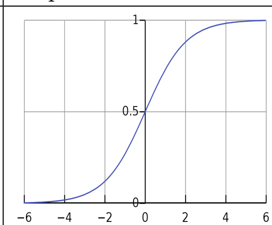
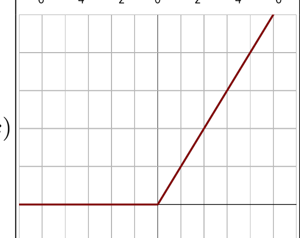
$$w_i = w_i - \eta (\frac{\partial J(g_i(W))}{\partial g_i(W)} \frac{\partial g_i(W)}{\partial W}) \tag{7}$$

Since in the multi-layer ANNs the derivative is also applied to the activation functions, the choice of which activation function to be used is crucial. For example, using tanh or sigmoid functions may result in a slower learning process, as the first derivative of these functions is not so steep on high values (inset label and refrence).

References

1. Lemaréchal, C.: Cauchy and the gradient method. Doc Math Extra **251**, 254 (2012)

Table 1. diffrent Activation functions.

Function name	Formule	Graph
Sigmoid	$h_\theta(x) = \frac{1}{1+e^x}$	
ReLU	$Relu(x) = max(0, x)$	
tanh/ Hyperbolic Tangent	$\tanh(x)$