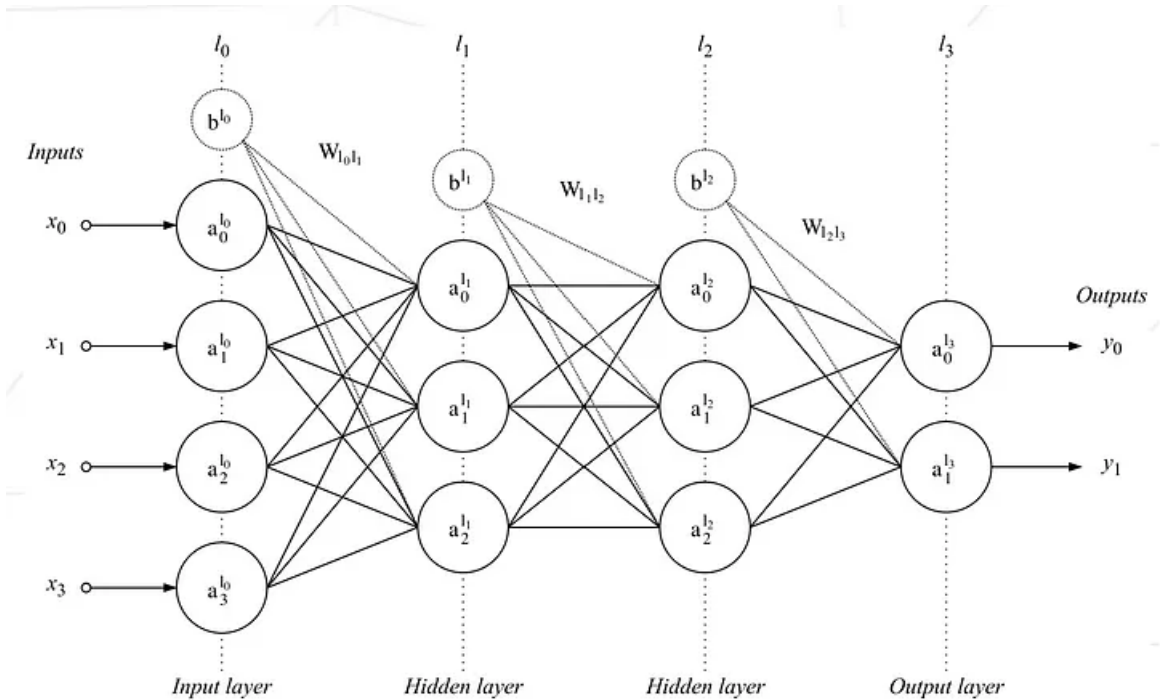


# Discovering Artificial Neural Networks with Perceptrons

Ludovic DEBEVER

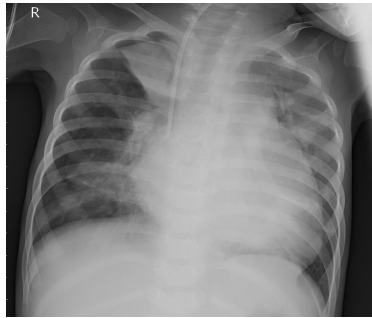
May 2024



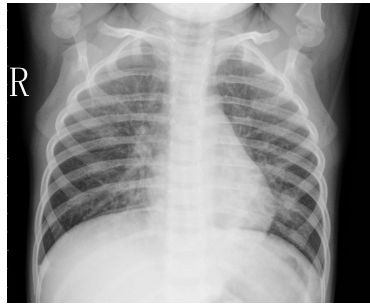
# 1 Introduction

The rise of Artificial Intelligence has had a huge impact on most domains related to Information Technology. Developers, Writers, Designers, and others have been empowered by these new solutions, which have transformed the way tasks are approached and executed, leading to increased efficiency, creativity, and innovation across various fields.

In this paper, we will explore how Artificial Intelligence can help Doctors classify diseases such as pneumonia using chest X-rays. By leveraging machine learning algorithms and deep neural networks, AI systems can analyze medical images with remarkable accuracy and speed, assisting healthcare professionals in diagnosing conditions more effectively. We will delve into the technical aspects of these AI models, examining their training processes, the data sets used, and the techniques employed to ensure reliability and precision.



Bacteria



Virus



Normal

# 2 Frank Rosenblatt's Perceptron

In 1957, Frank Rosenblatt invented the Perceptron, the simplest form of neural network and one of the first models to demonstrate machine learning. This groundbreaking model laid the foundation for future advancements in the field, as it was one of the earliest systems capable of learning from data. A model that would later be applied to fields such as image processing and speech recognition, it showcased the potential of machine learning to tackle complex tasks by improving performance through experience.

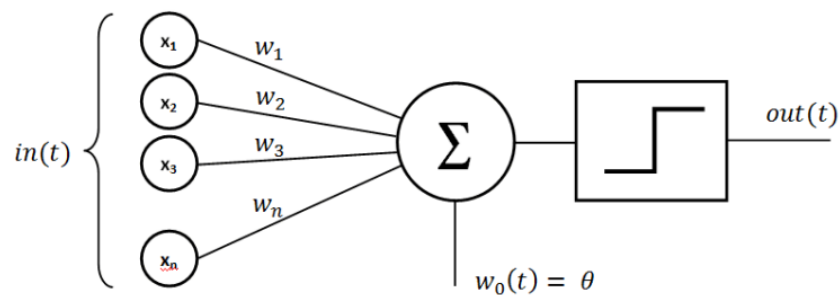
Based on a simple algorithm, the Perceptron could adapt to the input data in order to perform better over time. However, despite its innovative approach, the Perceptron had hard limitations. It could not solve non-linear problems, meaning it struggled with tasks where the relationship between input and output was not straightforward. This limitation was famously highlighted by Marvin Minsky and Seymour Papert in their 1969 book "Perceptrons," which demonstrated that the Perceptron could not handle the XOR problem, a simple example of a non-linear classification task.

## 2.1 How does a Perceptron learn?

The Perceptron involves principles that are foundational to modern artificial intelligence. It comprises key components such as Inputs and Weights, an Activation Function, and Back-propagation. At its core, the Perceptron is a single neuron directly connected to the inputs. Each input is assigned a specific weight, which represents its importance in the prediction process. To make a prediction, the Perceptron calculates the weighted sum of all inputs, which is the sum of each input multiplied by its corresponding weight.

Once the weighted sum is computed, the Perceptron employs an activation function to determine the output. In this simplified model, we use a threshold as the activation function. If the weighted sum exceeds the set threshold, the Perceptron produces a positive result; otherwise, it returns a negative result. This binary classification process allows the Perceptron to distinguish between two classes based on the input data.

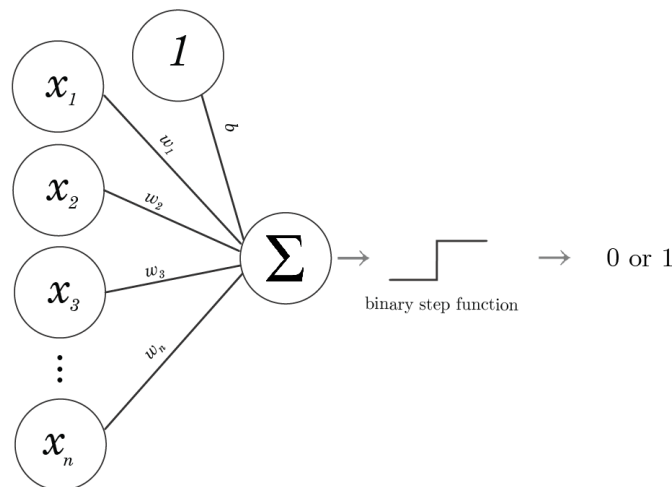
Perceptron



To improve its performance over time, the Perceptron uses a learning process called Back-propagation. During training, the Perceptron adjusts its weights based on the errors between the predicted and actual outcomes. By iteratively updating the weights to minimize these errors, it learns to make more accurate predictions.

In some rare cases we may encounter a set of inputs that sum to 0, with such input, no matter the weights the result will always be 0. To avoid this issue we use a concept called bias. It is a complementary input that we add with a fixed value of 1, so that no matter the input, the result can always be different than 0.

Perceptron with bias



## 2.2 Classifying Images using the Perceptron

Using this model, we can classify images. However, due to the Boolean nature of the Perceptron's output, it can only differentiate between two classes, which is enough in our case. Before we proceed, we need to normalize our data. Each image will be resized to a fixed number of pixels—64x64 in this case—which provides a good balance between detail and low memory usage. Then, the color ranges will be converted to gray-scale values, leaving us with a single number per pixel. Each of these gray-scale values will serve as an input to the Perceptron. The final step before starting the training process is to define the hyper-parameters, which are the human-controlled settings of the model. For a Perceptron, these include the learning rate, the activation function, the number of epochs, and the number of inputs. We will use a simple threshold for the activation function, and the number of inputs has already been established during data normalization. Therefore, we only need to define the learning rate and the number of epochs. The learning rate is crucial during the Perceptron's learning phase; after predicting a value and calculating the error, the error is multiplied by the learning rate and used to adjust the weights accordingly. The learning rate needs to be small enough to allow the back-propagation process to find precise values for the Perceptron's weights. Finally, the number of epochs define how many times the training process will go through the entire dataset. Perceptrons cannot be over-fit and are very quick to train due to its simple structure, so setting a high number of epochs should not disturb the training process.

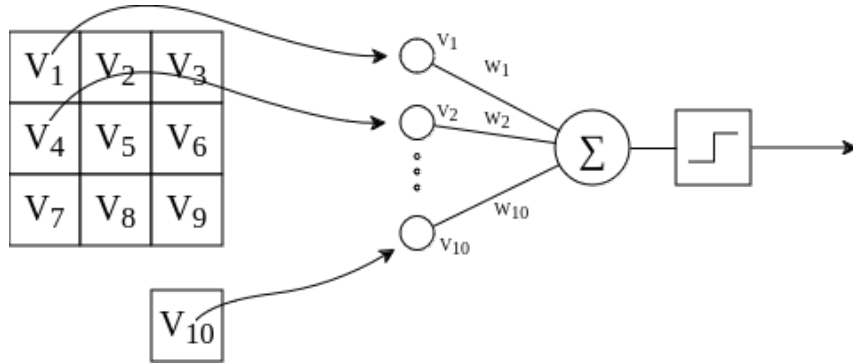
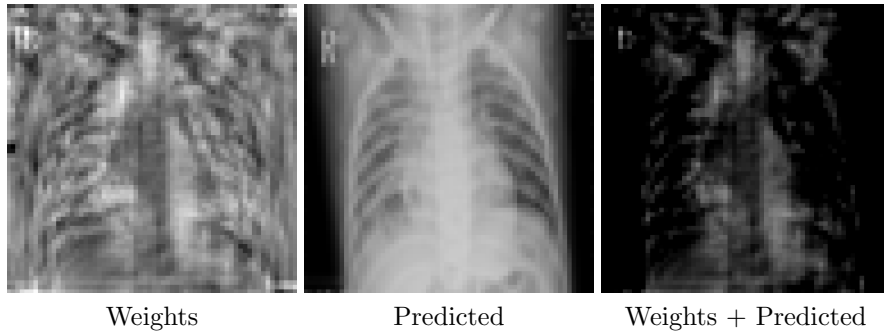
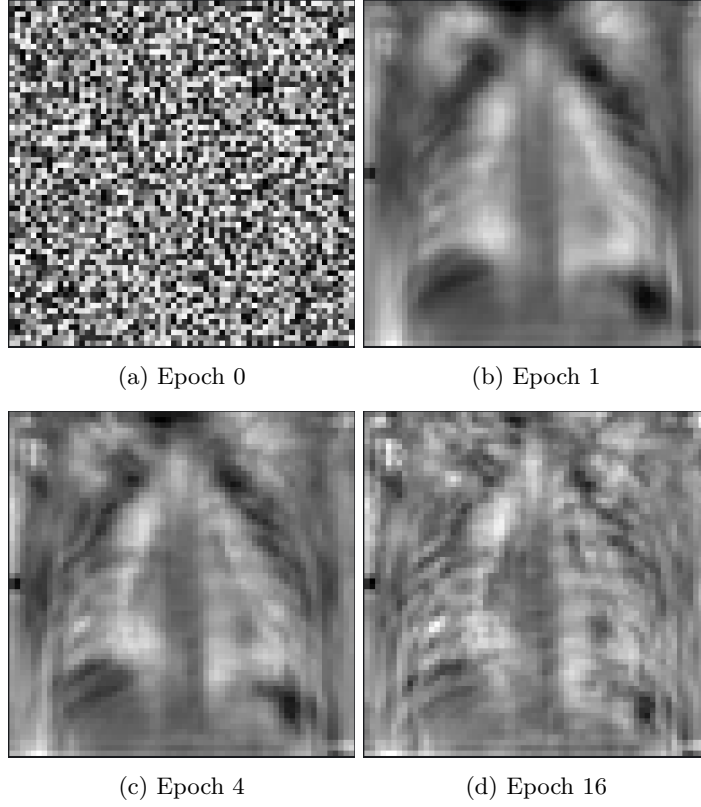


Image Classification Perceptron

Due to the simple structure of the Perceptron, visualizing its weights is quite simple. By assigning one weight per pixel, we can generate an image that mirrors the properties of our normalized training data, providing a glimpse into the Perceptron's "brain." Additionally, when predicting an image, we can combine each pixel's value with the corresponding weight from the Perceptron and normalize the result. This allows us to see what the Perceptron focuses on during image prediction.

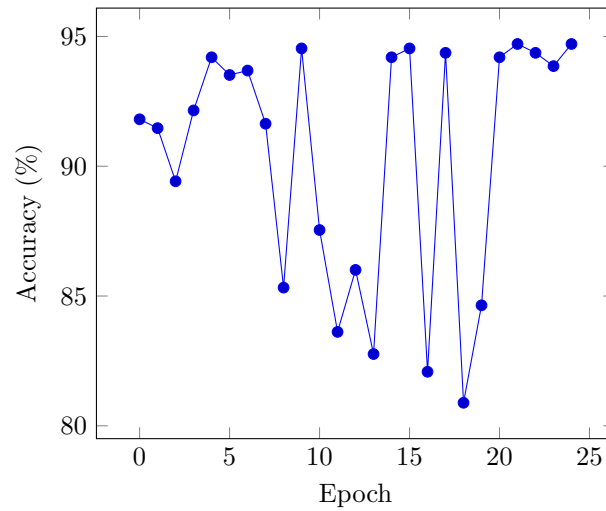
Weights of the Perceptron after 25 epochs on a single prediction.





Evolution of the Perceptron's weights

However, a problem arises when we examine the accuracy trend over multiple epochs: it fluctuates significantly. This suggests that the Perceptron is unable to find an optimal solution. The likely cause is a learning rate that is too high, leading to back-propagation overshooting as it attempts to adjust the weights based on the error.



To address this issue, we can replace the fixed learning rate with a dynamic function known as a learning rate scheduler. There are various types of schedulers, but I chose to test only three: Step Decay, Exponential Decay, and Cosine Annealing.