Insert Title Here

by

PARBHU Varun

Project Supervisor: Professor (Dr) BHURUTH MUDDUN OSK

Project submitted to the Department of Mathematics,
Faculty of Science, University of Mauritius,
as a partial fulfillment of the requirement
for the degree of

M.Sc. Mathematical and Scientific Computing (Part time)

December 2022

Contents

List of Figures

List of Tables

Acknowledgement

Placeholder

Abstract

Placeholder

Terms and Definitions

α	Learning Rate
$\mathbf{I}$	Identity Matrix

Placeholder

Placeholder

Chapter 1

Introduction

- Discussion around Cryptocurrency and frequency of changes
- Discussion around Twitter and the API (amount of data)
- Discussion around the advancement Compute processing speed and Deep Learning algorithms

Chapter 2

Methodology

2.1 Introduction

Deep learning is a multidisciplinary field ranging from linear algebra, calculus and probability theory. With major advancement made in compute processing power; deep learning is commonly used in the industry now.

2.2 Linear Algebra

We introduce briefly the realm of linear algebra to get a better understanding of more complex concepts in neural networks. The focus is set on mainly the properties useful in deep learning.
Linear algebra is the study of linear equations of the form:

$$x_1a_1 + \cdots + x_na_n = b \tag{2.1}$$

or as a linear map defined by the following:

$$(x_1, \ldots, x_n) \mapsto (a_1x_1 + \cdots + a_nx_n) \tag{2.2}$$

and their representation, manipulation and operation using vectors and matrices within a set of predefined axioms.

2.2.1 Vectors

A vector is quantity that has both a magnitude and direction. It can be represented by an ordered set of numbers or graphically using an arrow. Consider the vector **u** in $\mathbb{R}^2$ given by:

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.3}$$

The vector can be expressed graphically as shown below.

Image Placeholder vector in R^2

The length of the arrow is an indication of the magnitude of a vector and its orientation given by the direction in which the arrow is pointing. The magnitude of the above vector can be computed as such:

$$||\mathbf{u}|| = \sqrt{u_1^2 + u_2^2} \tag{2.4}$$

The direction of the vector with respect to the x-axis is given by:

$$\theta = \tan^{-1}\left(\frac{u_2}{u_1}\right) \tag{2.5}$$

A vector space is a set of vectors which satisfy the following axioms:

Placeholder - Table of Axioms for Vector Space

Geometry representation of addition and subtraction of vectors in $\mathbb{R}^2$

Vector addition and subtraction is performed componentwise along each element of two vectors. Consider the two vectors **a** and **b** where:

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.6}$$

Addition

The addition of vector $\mathbf{u}$ and $\mathbf{v}$ is given by:

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \end{bmatrix} \tag{2.7}$$

The geometrical effect of adding $\mathbf{u}$ and $\mathbf{v}$ together is shown below.

Image Placeholder vector addition in R^2

We can observe from (image ref) above that adding $\mathbf{u}$ to $\mathbf{v}$ has the effect of rotating $\mathbf{u}$ towards $\mathbf{v}$.

Subtraction

The subtraction of vector $\mathbf{v}$ from $\mathbf{u}$ is given by:

$$\mathbf{u} - \mathbf{v} = \begin{bmatrix} u_1 - v_1 \\ u_2 - v_2 \end{bmatrix} \tag{2.8}$$

Image Placeholder vector subtraction in R^2

We can observe from (image ref) above that subtracting $\mathbf{v}$ from $\mathbf{u}$ has the effect of rotating $\mathbf{u}$ in the opposite direction of $\mathbf{v}$.

The effect of rotating two vectors though addition and subtraction are useful properties utilized in deep learning.

2.2.1.1 Inner Product

The inner product or dot product of:

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.9}$$

is given by $\mathbf{u}.\mathbf{v}$

$$\mathbf{u}.\mathbf{v} = u_1 v_1 + u_2 v_2 \tag{2.10}$$

$$= ||\mathbf{u}|| \, ||\mathbf{v}|| \cos\theta \tag{2.11}$$

where θ is the angle between vectors $\mathbf{u}$ and $\mathbf{v}$.

Image Placeholder inner product in R^2

The dot product of **u.v** equals **v.u**. The order does not make a difference.

2.2.2 Matrices

A matrix is an $m \times n$ array of numbers, where m is the number of rows and n is the number of columns; the matrix is said to be of dimension $(m \times n)$.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \tag{2.12}$$

Similar to vectors, there are a couple of matrix operations that can be done under certain condition.

Matrix Operations

Matrix Addition

The sum of two matrices **A** and **B** is given by the element wise sum of the elements provided that they both have the same dimension.

Consider two matrices **A** and **B**:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \tag{2.13}$$

the sum **A** + **B** is given by

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix} \tag{2.14}$$

Scalar Matrix Multiplication

The product $\alpha\mathbf{A}$, where $\alpha \in \mathbb{R}$ and **A** is a matrix, is calculated by multiplying every entry of **A** by α.

Consider a matrix $\mathbf{A}$ and constant α:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{2.15}$$

the scalar multiplication $\alpha\mathbf{A}$ is given by

$$\alpha\mathbf{A} = \alpha \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} \\ \alpha a_{21} & \alpha a_{22} \end{bmatrix} \tag{2.16}$$

Matrix Transpose

The transpose of a $(m \times n)$ matrix $\mathbf{A}$ is calculated by swapping the rows into columns or the columns into rows. This operation result in a $(n \times m)$ denoted as $\mathbf{A}^T$.
Consider a matrix $\mathbf{A}$, the transpose is given by $\mathbf{A}^T$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix} \tag{2.17}$$

Matrix Multiplication

The multiplication operation between two matrices is only defined is the number of columns of the left matrix is the same as the number of rows in the right matrix. If $\mathbf{A}$ is a $(m \times n)$ matrix and $\mathbf{B}$ is a $(n \times p)$ matrix, then their product $\mathbf{AB}$ is the $(m \times n)$ matrix whose entries are given by inner product of the corresponding row of $\mathbf{A}$ and the corresponding column of $\mathbf{B}$.
Consider two matrices $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \tag{2.18}$$

the matrix multiplication is given by **AB**

$$\mathbf{AB} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \tag{2.19}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{bmatrix} \tag{2.20}$$

2.2.2.1 Outer Product

The outer product of two vectors **u** and **v**:

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.21}$$

is given by $\mathbf{u} \otimes \mathbf{v}$

$$\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_1v_1 & u_1v_2 \\ u_2v_1 & u_2v_2 \end{bmatrix} \tag{2.22}$$

The outer product of $\mathbf{u} \otimes \mathbf{v}$ is not equal to $\mathbf{v} \otimes \mathbf{u}$; the order does matter. However, the transpose of latter outer product are the same:

$$\mathbf{u} \otimes \mathbf{v} = (\mathbf{v} \otimes \mathbf{u})^T \tag{2.23}$$

2.3 Differential calculus

Differential calculus is the study of the definition, properties, and applications the derivative of a function. The process of finding the derivative is called differentiation. It is the also referred to as the study of rate of change and slopes of curves. In optimization and deep learning, major concepts of differential calculus are used.

2.3.1 Derivatives

Derivatives are important concept for understanding deep learning. The derivative of a function at a particular point is the rate of change of the output of the function with respect to the input at that point. It is also an indication of the slope of the function at a particular point.
The mathematical definition of the derivative of a function $f = f(x)$ at a point a is given by

$$f^{'}(a) = \frac{df}{dx}(a) \tag{2.24}$$

$$= \lim_{\Delta \to 0} \frac{f(a + \Delta) - f(a - \Delta)}{2 \times \Delta} \tag{2.25}$$

Image Placeholder derivative of a function f at a in $\mathbb{R}^2$ and $\mathbb{R}^3$

Since the derivative of a function is an indication of the slope of a function; we can deduce that a minimum or maximum point exists when the derivative is equal to 0 (may also be indication of an inflection point).

Derivative of Multivariate Functions

2.3.2 Directional Derivatives

2.3.3 Chain Rule

The chain rule is a concept in calculus to help us understand and calculate the derivative for composite functions which can be made up of two or more functions chained together.
Suppose that we have two function $f(x)$ and $g(x)$ which are both differentiable and define the composite function $h(x) = f(g(x))$ then the derivative of $h^{'}(x)$ is given by

$$h^{'}(x) = \frac{dh}{dx} \tag{2.26}$$

$$= f^{'}(g(x))g^{'}(x) \tag{2.27}$$

Consider that we have $y = f(u)$ and $u = g(x)$, a different definition of the composite function $y = f(g(x))$, then the derivative of y is,

$$h'(x) = \frac{dh}{dx} \tag{2.28}$$
$$= \frac{dh}{du}\frac{du}{dx} \tag{2.29}$$

2.3.4 Hessian

2.4 Optimization

The optimization problem is a computational problem in which the objective is to find the best of all possible solutions. Deep neural networks is a form of an optimization problem to find the best possible set of weights in order to reduce the error in a network.

A generic form of an optimization problem is given by

$$\begin{array}{ll} \underset{x}{\text{minimize/maximize}} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_j(x) = 0, \quad j = 1, \dots, p \end{array}$$

where

$f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective/loss function to be minimized,

$g_i(x) \leq 0$ are called inequality constraints,

$h_j(x) = 0$ are called equality constraints, and

$m \geq 0$ and $p \geq 0$

If $m = p = 0$, the problem is an unconstrained optimization problem.

2.4.1 Optimization algorithms

The solutions to the optimization problem are vital in modern machine learning and artificial intelligence algorithms, which includes weight optimization in deep learning. There are a number of popular optimization algorithm currently developed to solve the problem. Hence, choosing the right algorithm can be challenging as well. We explore few of the gradient-based solution to the optimization below.

Gradient-based methods are iterative methods that use the gradient information of the objective function during iterations.

Newton's Method

For minimizing $f(x)$, $x \in \mathbb{R}$, we need to solve $g(x) = f^{'}(x) = 0$. Newton's iteration is given by

$$x_{n+1} = x_n - \frac{g(xn)}{g^{'}(x_n)} \tag{2.30}$$

$$= x_n - \frac{f^{'}(xn)}{f^{''}(x_n)} \tag{2.31}$$

For multivariate functions we need to minimize $f(\mathbf{x})$ over $\mathbf{x} \in \mathbb{R}^n$, that it

$$\underset{\mathbf{x} \in \mathbb{R}^n}{min} f(\mathbf{x}), \quad \mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n \tag{2.32}$$

The Newton's iteration for multivariate function is given by

$$\mathbf{x_{n+1}} = \mathbf{x_n} - H(\mathbf{x_n})^{-1} \nabla f(\mathbf{x_n}) \tag{2.33}$$

where

$H(\mathbf{x_n})$ is the Hessian matrix of $f(\mathbf{x})$

We can observe from (equation ref) that calculating the inverse of the Hessian matrix can be computationally very expensive for higher dimensions. Replacing $H(\mathbf{x_n})^{-1}$ by $\alpha\mathcal{I}$ where $\mathcal{I}$ is the identity matrix; we get the **method of steepest descent** given by

$$\mathbf{x_{n+1}} = \mathbf{x_n} - \alpha \mathbf{I} \nabla f(\mathbf{x_n}) \tag{2.34}$$

where $\alpha \in (0, 1)$

2.5 Deep Learning

Deep learning is part of Machine Learning which deals mainly with computers that can learn either on their own or supervised. The latter can solve problem in the industry ranging from computer vision (image), natural language processing (text), automatic speech recognition (audio), time-series prediction amongst others. Deep learning primarily uses the concept of artificial neural networks, which derivatives from how the human brain works, to solve complex linear and non-linear problem.

We start by introducing how a simple perceptron works, and gradually increase the complexity of the network until we reach a deep neural network that can solve non-linear problems.

2.5.1 Perceptron

A perceptron is an analogy to the human neuron. It is a computational model that takes an input (scalar or vector) and learns to classify the latter between two classes (binary classifier). It consists of an input, a weighed sum and an activation function. The mathematical definition a perceptron that maps its input $\mathbf{x}$ to an output value $f(\mathbf{x})$ using a step function as activation function is given by

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}.\mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.35}$$

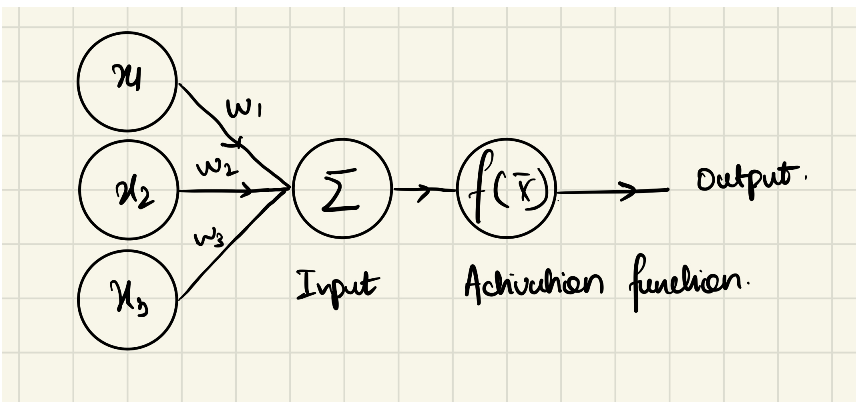

Figure 2.1: Perceptron graphical representation

From the perceptron mathematical definition 2.35, we can observe that the dot product is dependent on the angle between the weight vector ($\mathbf{w}$) and the input vector ($\mathbf{x}$). Using dot product (2.11), addition (2.7) and subtraction (2.8) of vector property; we can define a learning rule for modifying the perceptron to learn how to classify a set of input.

2.5.1.1 Perceptron Learning Rule

Consider the two sets of point **A** and **B**

$$\mathbf{A} = \left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\} \; \mathbf{B} = \left\{ \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right\} \tag{2.36}$$

and fitting the perceptron function (2.35) to learn how to classify points in each respective set. Let set **A** and **B** be class 1 and 0 respectively.

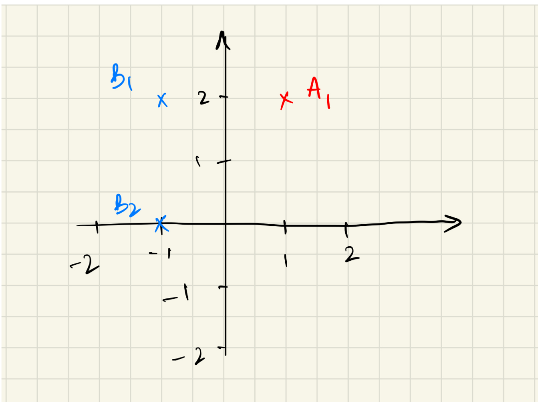

Figure 2.2: Caption

From 2.2, we can separate the two classes by a single line. Consider the random separating line passing through the origin (this will set the value of b in (2.35) to 0).

The equation of the random line is given by

$$x_1 = -x_2 \tag{2.37}$$

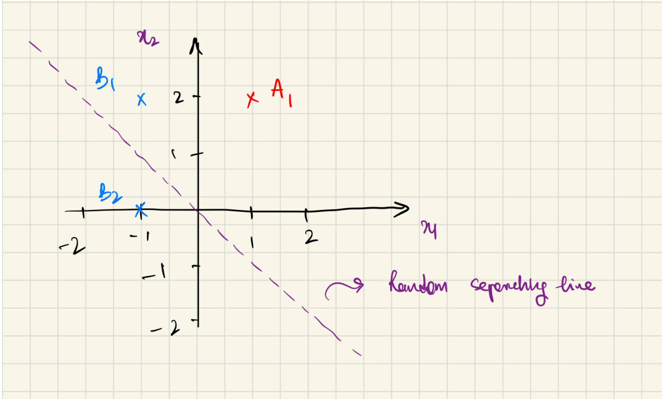

Figure 2.3: Caption

Comparing to the equation (2.35); we can deduce that

$$w_1 = 1 \quad w_2 = 1 \tag{2.38}$$

Visually, we can already see that the separating line does not split the points properly. The computation of the binary classification using the random line is given by

$$f(A_1) = f\left(\begin{bmatrix}1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\2\end{bmatrix}\right) = f(3) = 1 \tag{2.39}$$

$$f(B_1) = f\left(\begin{bmatrix}1\\1\end{bmatrix} \cdot \begin{bmatrix}-1\\2\end{bmatrix}\right) = f(1) = 1 \tag{2.40}$$

$$f(B_2) = f\left(\begin{bmatrix}1\\1\end{bmatrix} \cdot \begin{bmatrix}-1\\0\end{bmatrix}\right) = f(-1) = 0 \tag{2.41}$$

The computation confirms the visual representation and groups $\mathbf{A}_1$ and $\mathbf{B}_1$ together. We can either alter the separating line (by moving the weight vector) with respect to $\mathbf{A}_1$ and/or $\mathbf{B}_1$.

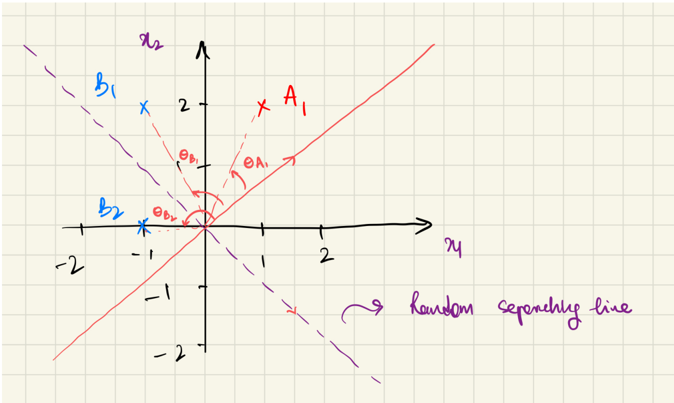

Figure 2.4: Caption

Using the property of subtraction of vectors; we move the weight vector $(\mathbf{w})$ away from $\mathbf{B}_1$ and check the classification again. We also introduce a learning rate α to control how much rotation we want. Let $\alpha = 0.5$

$$\mathbf{w}^{(1)} = \mathbf{w} - \alpha \mathbf{B}_1 \tag{2.42}$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0.5 \begin{bmatrix} -1 \\ 2 \end{bmatrix} \tag{2.43}$$

$$= \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} \tag{2.44}$$

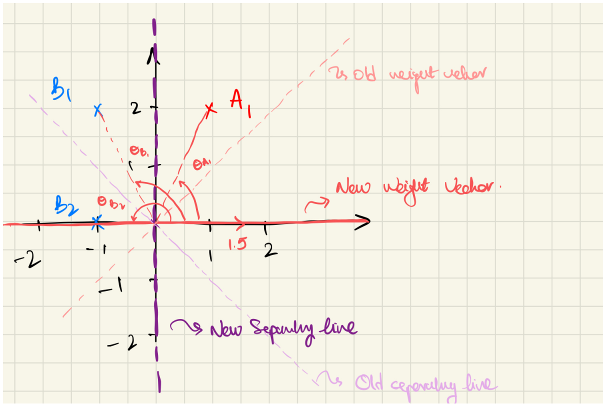

Figure 2.5: Caption

Based on the new separating line, we can observe visually that the points are correctly classified. The computation of the binary classification using the new separating line $\mathbf{w}^{(1)}$ is given by

$$f(A_1) = f\left(\begin{bmatrix}1.5\\0\end{bmatrix}.\begin{bmatrix}1\\2\end{bmatrix}\right) = f(1.5) = 1 \tag{2.45}$$

$$f(B_1) = f\left(\begin{bmatrix}1.5\\0\end{bmatrix}.\begin{bmatrix}-1\\2\end{bmatrix}\right) = f(-1) = 0 \tag{2.46}$$

$$f(B_2) = f\left(\begin{bmatrix}1.5\\0\end{bmatrix}.\begin{bmatrix}-1\\0\end{bmatrix}\right) = f(-1.5) = 0 \tag{2.47}$$

Thus, the perceptron learned how to classify the two sets of points. The function $f(\mathbf{x})$ could be re-used to classify new points added to the sets.

2.5.1.2 Perceptron Update Rule

We can define a set number of rules for the perceptron to find the optimal weight vector to classify the points. Based on the properties of the dot product; the classification of the points are dependent on the inner angle between the normal of the separating line and the vector to be classified. If the inner angle is less than 90° then the point belongs to class 1 else to class 0.

Target Classes and Errors

Let the target(actual) classes be $y = 1$ and $y = 0$. Then the network error ϵ is given by the difference between the true value and the predicted value.

$$\begin{aligned}
\text{Network Error} \quad & \epsilon(\mathbf{x}) = y - f(\mathbf{x}) \\
\text{Correct Classification} \quad & y = 1, f(\mathbf{x}) = 1 \rightarrow \epsilon(\mathbf{x}) = 0 \\
& y = 0, f(\mathbf{x}) = 0 \rightarrow \epsilon(\mathbf{x}) = 0 \\
\text{Incorrect Classification} \quad & y = 1, f(\mathbf{x}) = 0 \rightarrow \epsilon(\mathbf{x}) = 1 \\
& y = 0, f(\mathbf{x}) = 1 \rightarrow \epsilon(\mathbf{x}) = -1
\end{aligned}$$

We find that updating the weight vector can be written as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \alpha\epsilon(\mathbf{x})\mathbf{x} \tag{2.48}$$

Perceptron Learning Algorithm

The steps for the perceptron to learn how to perform binary classification is given by the steps below. Let t be the t^{th} iteration when learning and b be in an input bias.

Assignment $\rightarrow$ Assign $t = 0$, $\mathbf{w}^{(0)} = (0, 0, \ldots, 0)$, $b^{(0)} = 0$

Start $\rightarrow$ For $t = 0, 1, 2, \ldots$ until convergence

Step 1 $\rightarrow$ Randomly choose a vector $\mathbf{x}^{(t)}$ with the corresponding $y^{(t)}$ (known)

Step 2 $\rightarrow$ Compute $\mathbf{z} = \mathbf{w}^{(t)}.\mathbf{x}^{(t)} + b^{(t)}$

Step 3 $\rightarrow$ Compute $\epsilon^{(t)}(\mathbf{z}) = y^{(t)} - f(\mathbf{z})$

Step 4 $\rightarrow$ Update weight $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \alpha\epsilon^{(t)}(\mathbf{z})\mathbf{x}^{(t)}$

Step 5 $\rightarrow$ Update bias $b^{(t+1)} = b^{(t)} + \alpha\epsilon^{(t)}(\mathbf{z})$

The current perceptron we have defined is for a binary classification; we can also define multiclass perceptron and/or use different types of activation functions.

2.5.2 Neural Network

A neural network is a set of perceptron connected together which takes an input, manipulate the information to learn from it and outputs a prediction. A neural network attempts to learn a mapping from an input to an output. The goal is to reduce the error between the prediction by the network compared to the true value as much as possible.

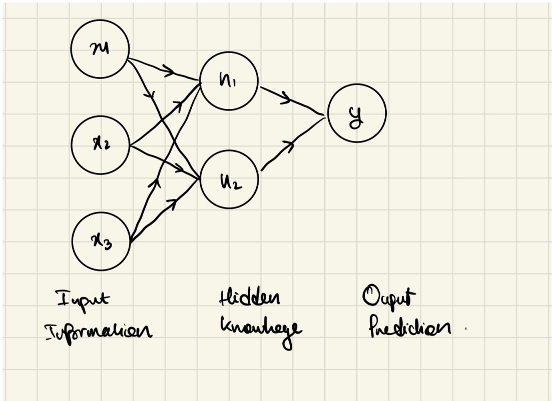

Figure 2.6: Caption

2.5.3 Deep Neural Network

- Explanation on Deep Neural Network

- Example on how it solves the OR and XOR function

2.5.4 Back propagation

2.5.5 Overfitting

2.5.6 Dropout

2.5.7 Activation Function

2.5.8 Convolutional Layer

Chapter 3

PyTorch

Chapter 4

Sentiment analysis

Chapter 5

Tweet Data

5.1 Tweets

5.1.1 Tweepy

5.1.1.1 Scrapping using Tweepy

5.1.1.2 Cleaning using RegEx

5.2 Tweet Count

5.3 Tweet Search Volume Index (SVI)

5.4 Feature Engineering

Bibliography

AuthorName (Year*a*), Conferencetitle, *in* ‘BookTitle’.

AuthorName (Year*b*), ‘Title of article’, *Journal* .

BookAuthors (Year), *BookTitle*, Publisher.