

# Extra Mathematical Notes for Lectures and Classes\*

Parley Ruogu Yang<sup>†</sup>

This version: Friday 28<sup>th</sup> January, 2022

## Abstract

This document consists of notes for Lectures (section 2) and Classes (section 3).

## Contents

<b>1</b>	<b>General notes</b>	<b>2</b>
1.1	Notations . . . . .	2
1.2	Activation functions . . . . .	2
<b>2</b>	<b>Lectures</b>	<b>3</b>
2.1	Lecture 3: Optimisation algorithm . . . . .	3
<b>3</b>	<b>Classes</b>	<b>4</b>
3.1	Class 1: Linear and logistic regressions . . . . .	4
3.1.1	Linear regression and MSE loss . . . . .	4
3.1.2	Gradient of linear regression with MSE loss . . . . .	4
3.1.3	Logistic regression model and binary cross entropy . . . . .	4
3.2	Class 2: Perceptron and the XOR Problem . . . . .	5
3.2.1	Perceptron . . . . .	5
3.2.2	The XOR Problem statement . . . . .	5
3.2.3	Theoretical result . . . . .	5
3.3	Class 3: Gradient Descent and Stochastic Gradient Descent . . . . .	6
3.3.1	Gradient Descent . . . . .	6
3.3.2	Stochastic Gradient Descent . . . . .	6
3.4	Class 3: Option Pricing . . . . .	7
3.4.1	Background . . . . .	7
3.4.2	Class 3 Notebook 1 . . . . .	7
3.4.3	Class 3 Notebook 2 . . . . .	7
3.4.4	Class 3 Homework . . . . .	7

---

\*Latest version: <https://parleyyang.github.io/ST456/index.html>

<sup>†</sup>Faculty of Mathematics, University of Cambridge

# 1 General notes

## 1.1 Notations

The default meaning of  $\mathbb{N}$  is the set of integers greater or equal to 1. For  $n \in \mathbb{N}$ , denote  $[n] := \{1, 2, \dots, n-1, n\} = [1, n] \cap \mathbb{N}$ . When  $x \in \mathbb{R}^n$  is written,  $x_i$  stands for the  $i$ -th entry of  $x$ . If  $\rho : \mathbb{R} \rightarrow \mathbb{R}$  is well-defined, then for  $y \in \mathbb{R}^n$ ,  $\rho(y) := (\rho(y_1), \dots, \rho(y_n))$ , also known as element-wise operation.

$N(\mu, \sigma^2)$  refers to a normal distribution with mean  $\mu$  and variance  $\sigma^2$ , while a standard normal distribution refers to the case when  $\mu = 0$  and  $\sigma^2 = 1$ .

Where  $\varepsilon$  or  $\varepsilon_i$  are written, the default meaning is that they are drawn from iid  $N(0, \sigma^2)$  distribution with  $\sigma^2$  unknown.

NN stands for Neural Networks

## 1.2 Activation functions

Let  $\rho^{\text{sigmoid}} : \mathbb{R} \rightarrow \mathbb{R}$  be the sigmoid function, it is defined by

$$x \mapsto (1 + \exp(-x))^{-1} \quad (1.2.1)$$

Let  $\rho^{\text{thr}} : \mathbb{R} \rightarrow \mathbb{R}$  be the threshold function, it is defined by

$$x \mapsto \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{else} \end{cases} \quad (1.2.2)$$

This function is also commonly written as  $\mathbb{1}[x \geq 0]$

## 2 Lectures

### 2.1 Lecture 3: Optimisation algorithm

### 3 Classes

#### 3.1 Class 1: Linear and logistic regressions

##### 3.1.1 Linear regression and MSE loss

Let  $x \in \mathbb{R}^m$  be the input variable. Let  $y \in \mathbb{R}$  be the output variable.

We consider  $y = f(x) + \varepsilon$  where  $f(x) = x^T w + b$

If we have data  $\{(x_i, y_i) : i \in [n]\}$ , the MSE loss takes the following form:

$$l(w, b) = n^{-1} \sum_{i \in [n]} (y_i - f(x_i))^2 = n^{-1} \sum_{i \in [n]} (y_i - x_i^T w - b)^2 \quad (3.1.3)$$

##### 3.1.2 Gradient of linear regression with MSE loss

It will be useful later in subsection 3.3 to have the gradient  $\nabla l$  in hand. In particular:

$$\nabla_w l(w, b) = n^{-1} \sum_{i \in [n]} (2x_i)(f(x_i) - y_i) \quad (3.1.4)$$

$$\nabla_b l(w, b) = n^{-1} \sum_{i \in [n]} 2(f(x_i) - y_i) \quad (3.1.5)$$

##### 3.1.3 Logistic regression model and binary cross entropy

Let  $\rho$  be the sigmoid function, then we consider  $y = f(x) + \varepsilon$  where  $f(x) = \rho(x^T w + b)$

In the event of binary classification problem, in which  $y \in \{0, 1\}$ , we clearly do not have  $\varepsilon$  as a Normally distributed error. In this occasion, with data  $\{(x_i, y_i) : i \in [n]\}$ , we consider the binary cross entropy as

$$l(f) = -n^{-1} \left( \sum_{i \in [n]} y_i \log(f(x_i)) + (1 - y_i) \log(1 - f(x_i)) \right) \quad (3.1.6)$$

## 3.2 Class 2: Perceptron and the XOR Problem

### 3.2.1 Perceptron

With an activation function  $\rho : \mathbb{R} \rightarrow \mathbb{R}$  and a feature engineering  $\phi$ , we have a single-layer NN as  $x \mapsto \rho(\phi(x)^T w + b)$

For the rest of the class (as well as in the lecture), we ignore  $\phi$ , or equivalently replace it by an identity map. A feed-forward NN with depth  $L$  can be written as

$$y = h_L \circ h_{L-1} \circ \dots \circ h_1(x) \quad (3.2.7)$$

where  $h_l(x) = a_l(W^{(l-1)}x + b^{(l-1)})$  for all  $l \leq L-1$  and  $h_L(x) = W^{L-1}x + b^{L-1}$ .

### 3.2.2 The XOR Problem statement

Consider  $x \in \mathbb{R}^2$  and  $y \in \mathbb{R}$ , in particular, our data is as follows:

$$D = \{((-1, -1), -1), ((-1, 1), 1), ((1, -1), 1), ((1, 1), -1)\} \quad (3.2.8)$$

The objective is to separate the points, mathematically one uses

$$L(f) = \sum_{i \in [4]} \max(-y_i f(x_i), 0) \quad (3.2.9)$$

### 3.2.3 Theoretical result

**Theorem 1** (Failure of linear functions compared against single-layer NN). *Let  $\mathcal{L}$  be the class of all non-zero linear functions  $\mathbb{R}^2 \rightarrow \mathbb{R}$  and let*

$$\mathcal{N}(\rho) = \{f : \mathbb{R}^2 \rightarrow \mathbb{R} : f(x) = \rho(w_1 x + b_1)^T w_2 + b_2, w_1 \in \mathbb{R}^{2 \times 2}, w_2, b_1 \in \mathbb{R}^2, b_2 \in \mathbb{R}\} \quad (3.2.10)$$

where  $\rho$  is the threshold function. Then

$$\min_{f \in \mathcal{L}} L(f) > 0 = \min_{f \in \mathcal{N}(\rho)} L(f) \quad (3.2.11)$$

*Proof.* The left hand side can be proved by a 2-D diagram, or analytically via the diagram-induced geometry. The right hand side can be proved by showing an element  $f \in \mathcal{N}(\rho)$  satisfies  $L(f) = 0$ , which is equivalent to show  $y_i = f(x_i) \forall i$ . Consider

$$b_1 = (0, 0), b_2 = -1, w_2 = (-2, 2) \\ w_1 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

which offers one specification that works. □

Remarks:

1.  $\mathcal{N}(\rho)$  can also be thought as the class of all two-layer NNs with architecture as  $(2, 2, 1)$  and activation function as the threshold function.
2. Note that the loss function can be 0 if  $f(x_i) = 0 \forall i$ . This is a bug of the loss function, hence when considering linear function, we restrict to the non-linear ones.

### 3.3 Class 3: Gradient Descent and Stochastic Gradient Descent

#### 3.3.1 Gradient Descent

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be a differentiable function, a gradient descent sequence  $\{x_n\}_{n=0}^\infty$  with learning rate scheduling  $\{\eta_n\}_{n=0}^\infty$  and initialisation  $x_{ini}$  is defined as

$$x_0 = x_{ini} \tag{3.3.12}$$

$$x_n = x_{n-1} - \eta_{n-1} \nabla f(x_{n-1}) \quad \forall n \in \mathbb{N} \tag{3.3.13}$$

#### 3.3.2 Stochastic Gradient Descent

## 3.4 Class 3: Option Pricing

### 3.4.1 Background

A (European) call option at maturity  $T$  gives the owner the right to buy an underlying asset at strike price  $K$ . This price of such an option is denoted as  $V(S_t, t; K)$  at time  $t \in [0, T]$ , where  $S_t$  is the price of the underlying asset at time  $t$ . It is natural to relate this to various parameters in the market: in the Black-Scholes model, we relate this to the interest rate  $r$  and volatility  $\sigma$ . A PDE expression is provided as

$$\partial_t V + rS\partial_S V + \frac{1}{2}\sigma^2 S^2 \partial_S^2 V = rV \quad (3.4.14)$$

The solution of this is complicated and non-linear:

$$V(S_t, t; K) = S_t N(d_1) - K e^{r(T-t)} N(d_2) \quad (3.4.15)$$

where  $d_1 = (\sigma\sqrt{T-t})^{-1}(\log(S_t K^{-1}) + (r + \frac{\sigma^2}{2})(T-t))$  and  $d_2 = d_1 - \sigma\sqrt{T-t}$

### 3.4.2 Class 3 Notebook 1

In this notebook, we keep other parameters the same and study the relationship between strike price  $K$  and the associated price of call option  $V$ . In particular, we select a number of strike prices, denoted  $x_1, \dots, x_n \in \mathbb{R}$  and generate the call option prices  $y_1, \dots, y_n \in \mathbb{R}$  in accordance with Equation 3.4.15. The dataset is hence  $\{(x_i, y_i) : i \in [n]\}$  and that we would like to approximate a function  $f : \mathbb{R} \rightarrow \mathbb{R}$  as we generate our data  $y_i = f(x_i) \quad \forall i$

### 3.4.3 Class 3 Notebook 2

In practice, one would be asked for the implied volatility  $\sigma$  given the data they receive — in this notebook, we fix 16 different strike prices and collect their corresponding call prices: for now, assume no noise. Then, for each  $y_i = \sigma_i \in \mathbb{R}$ , we have a 16-dimensional data  $x_i \in \mathbb{R}^{16}$ , so the dataset is  $\{(x_i, y_i) : i \in [n]\}$  and that we would like to approximate a function  $f : \mathbb{R}^{16} \rightarrow \mathbb{R}$  as we generate our data  $y_i = f(x_i) \quad \forall i$

### 3.4.4 Class 3 Homework

Realistically, the data contains noise. In the Homework, we will work with noisy data, in particular, we consider the same function  $f : \mathbb{R}^{16} \rightarrow \mathbb{R}$  as was in Notebook 2, but that we generate  $\varepsilon_i \sim N(0_{16}, \sigma^2 I_{16 \times 16}) \forall i \in [n]$ , and observe  $\tilde{x}_i = \max\{x_i + \varepsilon_i, 0\}$  instead of  $x_i$ . The maximum is in place because the practical world would not accept a negative prices on an option — so whilst there are noises, there is an obvious truncation.

So, we are still in the business of approximating  $f$ , but this time we have data  $\{(\tilde{x}_i, y_i) : i \in [n]\}$ .