

# COMP3220 — Document Processing and Semantic Technologies

## Week 04 Lecture 1: Deep Learning for Text Classification

Diego Mollá

Department of Computer Science  
Macquarie University

COMP3220 2022H1

# Programme

- 1 Deep Learning
- 2 Classification in Keras

## Reading

- Deep Learning Book (2nd edition), Chapters 2, 3, and 4.

## Additional Reading

- Jurafsky & Martin, Chapter 7 "Neural Networks and Neural Language Models".

# Programme

- 1 Deep Learning
  - A Neural Network
  - Deep Learning
- 2 Classification in Keras

# What is Deep Learning?

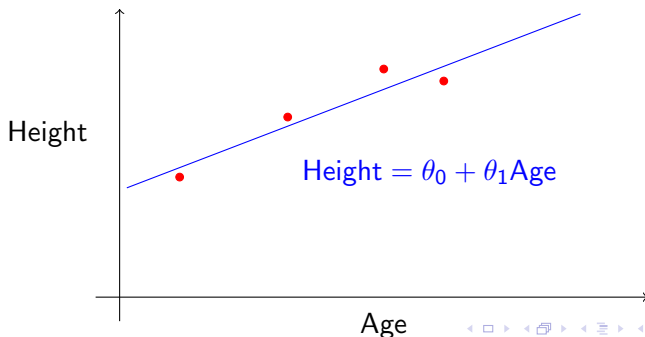
- Deep learning is an extension to the neural networks first developed during the late 20th century.
- The main differences between deep learning and the early neural networks are:
  - 1 A principled manner to combine simple neural network architectures to build complex architectures.
  - 2 Better algorithms to train the architectures.
- Besides improvements in the theory, three main drivers of the success of deep learning are:
  - 1 The availability of large training data.
  - 2 The availability of much faster computers.
  - 3 Massive parallel methods that use specialised hardware.
    - Graphic Processing Units.

# Programme

- 1 Deep Learning
  - A Neural Network
  - Deep Learning
- 2 Classification in Keras

# Linear Regression: The Simplest Neural Network

- Linear regression is one of the simplest machine learning methods to predict a numerical outcome.
- For example, we want to predict the height of a person based on its age.
- Based on the training data, linear regression will try to find the line that best fits the training data:

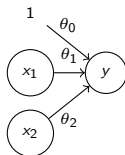


# Linear Regression with Multiple Variables

- For example, we want to predict the value of a house based on two features:
  - $x_1$  Area in squared metres.
  - $x_2$  Number of bedrooms.
- We can predict the value based on a **linear combination** of the two features:

$$f(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

- Where all  $\theta_i$  are learnt during the training stage.



# Supervised Machine Learning as an Optimisation Problem

- The machine learning approach will attempt to learn the parameters of the learning function that **minimise the loss** (prediction error) in the training data.

$$\Theta = \operatorname{argmin}_{\Theta} L(X, Y)$$

Where

- $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$  is the training data, and
- $Y = \{y^{(1)}, y^{(2)}, \dots, y^{(n)}\}$  are the labels of the training data.
- In linear regression:
  - $f(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_p x_p^{(i)}$
  - $L(X, Y) = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - f(x^{(i)}))^2$   
This loss is the **mean squared error**.



# Optimisation Problems in Other Approaches



## Logistic Regression

Logistic regression is commonly used for classification

- $$f(x^{(i)}) = \frac{1}{1 + e^{-\theta_0 - \theta_1 x_1^{(i)} - \dots - \theta_p x_p^{(i)}}}$$
- $$L(X, Y) = -\frac{1}{n} \sum_{i=1}^n \left( y^{(i)} \times \log f(x^{(i)}) + (1 - y^{(i)}) \times \log (1 - f(x^{(i)})) \right)$$

This loss is called **cross-entropy**.

## Support Vector Machines

Initially, SVM was formulated differently but it can also be seen as:

- $$f(x^{(i)}) = \text{sign} p(x^{(i)})$$
$$p(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_p x_p^{(i)}$$
- $$L(X, Y) = \frac{1}{n} \max\{0, 1 - y^{(i)} \times p(x^{(i)})\}$$

This is called the **hinge loss**.

# Solving the Optimisation Problem



- A common approach to find the minimum of the loss function is to find the value where the **gradient of the loss function is zero**.
- This results in a system of equations that can be solved.

## System of equations in linear regression

$$\frac{\partial}{\partial \theta_0} \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \dots - \theta_p x_p^{(i)})^2 = 0$$

$$\frac{\partial}{\partial \theta_1} \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \dots - \theta_p x_p^{(i)})^2 = 0$$

...

$$\frac{\partial}{\partial \theta_p} \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \dots - \theta_p x_p^{(i)})^2 = 0$$

# Gradient Descent



- Solving the system of equations  $\frac{\partial L}{\partial \theta_0} L(X, Y) = 0, \frac{\partial}{\partial \theta_1} L(X, Y) = 0, \dots$  can be too time-consuming.
- e.g. in linear regression, the complexity of computing the formula that solves the system of equations is  $O(n^3)$ .
- Some loss functions are very complex (e.g. in deep learning approaches) and it is not practical to attempt to solve the equations at all.
- **Gradient descent** is an iterative approach that finds the minimum of the loss function.

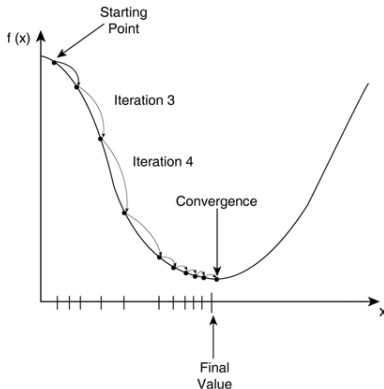
# Gradient Descent Algorithm



- 1 Assign initial random values to  $\theta_0, \dots, \theta_p$
- 2 Repeat until convergence:

For  $j = 1, 2, \dots, p$ :

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(X, Y)$$



# Batch Gradient Descent

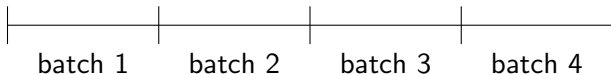


- There are automated methods to compute the derivatives of many complex loss functions.
  - This made it possible to develop the current deep learning approaches.
- Note, however, that every step of the gradient descent algorithm requires to process the **entire** training data.
- This is what is called **batch gradient descent**.

# Mini-batch Gradient Descent



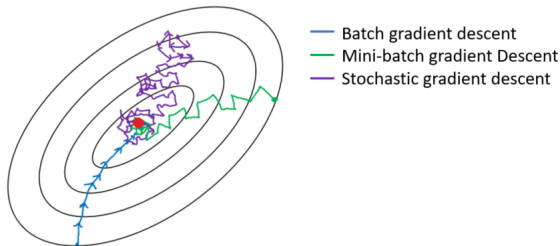
- In **mini-batch gradient descent**, only part of the training data is used to compute the gradient of the loss function.
- The entire data set is partitioned into small batches, and at each step of the gradient descent iterations, only one batch is processed.
  - If the batch size is 1, this is usually called **stochastic gradient descent**.
- When all batches are processed, we say that we have completed an **epoch** and start processing the first batch again.



# Mini-Batch Gradient Descent Algorithm



- 1  $\theta_0 = 0, \dots, \theta_p = 0$
- 2 Repeat until (near) convergence:
  - 1 Shuffle  $(X, Y)$  and split it into  $n$  mini-batches  $(X_0, Y_0), \dots, (X_n, Y_n)$ .
  - 2 For every mini-batch  $(X_i, Y_i)$ :
    - 1 For  $j = 1, 2, \dots, p$ :
$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(X_i, Y_i)$$



# Batch vs. Mini-Batch Gradient Descent



## Batch Gradient Descent

- At each iteration step we take the most direct path towards reaching a minimum.
- The algorithm converges in a relatively small number of steps.
- Each step may take long to compute (if the training data is large).

## Mini-batch Gradient Descent

- At each iteration step some random noise is introduced and we take a path roughly in the direction towards the minimum.
- The algorithm reaches **near convergence** in a larger number of steps.
- Each step is very quick to compute.



# Programme

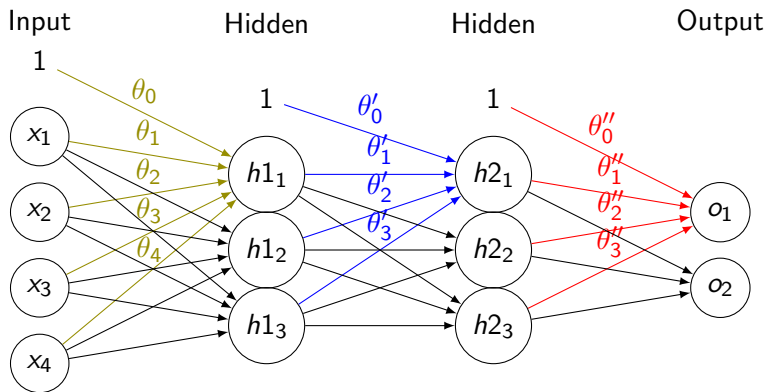
- 1 Deep Learning
  - A Neural Network
  - Deep Learning
- 2 Classification in Keras

# A Deep Learning Architecture

- A deep learning architecture is a large neural network.
- The principle is the same as with a simple neural network.
  - 1 Define a complex network that generates a complex prediction  $f(x_1, x_2, \dots, x_p)$ . This is normally based on simpler building blocks.
  - 2 Define a loss function  $L(X, Y)$ . There are some popular loss functions for classification, regression, etc.
  - 3 Determine the gradient of the loss function. This is done automatically.

# A feedforward neural network

a.k.a. multilayer perceptron (MLP)



- $h1_1 = f_{h11}(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4)$
- $h2_1 = f_{h21}(\theta'_0 + \theta'_1 h1_1 + \theta'_2 h1_2 + \theta'_3 h1_3)$
- $o_1 = f_{o1}(\theta''_0 + \theta''_1 h2_1 + \theta''_2 h2_2 + \theta''_3 h2_3)$

# Programme

- 1 Deep Learning
- 2 Classification in Keras

# Classification in Keras

This section is based on the jupyter notebooks provided by the Deep Learning book: <https://github.com/fchollet/deep-learning-with-python-notebooks>

- Binary classification of movie reviews.
- Multi-class classification of news wires.

Study these notebooks carefully since they contain important information about how neural networks are constructed and how they operate. The notebooks also introduce important terminology that you need to understand.

# Take-home Messages

- 1 Understand the general process in deep learning.
- 2 Understand the jargon in deep learning: activation, loss, batches, epochs, ...
- 3 Implement and evaluate a feedforward network in Keras for text classification.

# What's Next

## Week 5

- Embeddings and Text Sequences.

## Reading

- Deep Learning book, chapter 6.

## Additional Reading

- Jurafsky & Martin's book, chapter 9 (9.4 will be covered in week 6).