Introduction to deep learning

Tengku Muhammad Hanis Bin Tengku Mokhtar, PhD

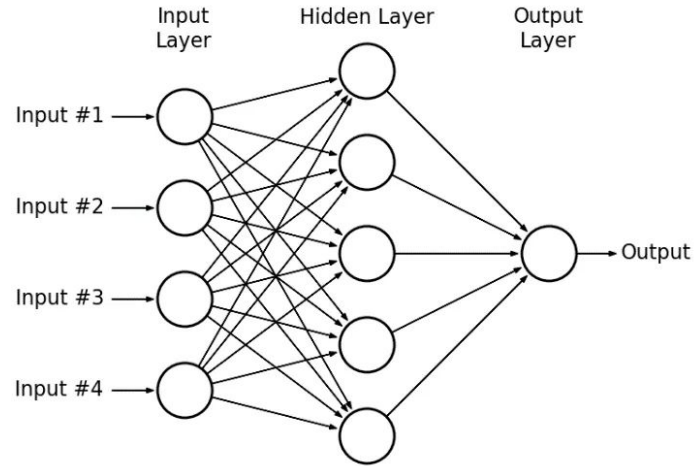**October 23, 2024**

Contents

Deep learning

- Deep learning is a subfield of machine learning
- Some consider it as a separate field
- Deep learning is not always a solution to every problem
- Most common/basic architecture based on the source of data:
  - Tabular data: Deep Neural Network (DNN)
  - Imaging data: Convolutional Neural Network (CNN)
  - Video data: CNN, Recurrent Neural Network (RNN)
  - Time series data: RNN, Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU)
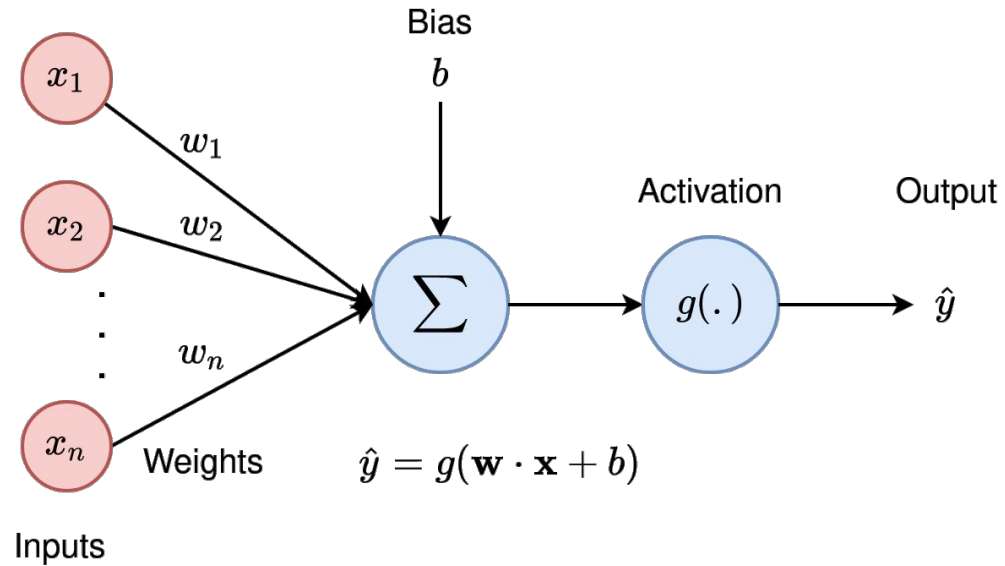 - Text data: RNN, LSTM, GRU, Transformer-based models (e.g., BERT, GPT)

- Classification and regression in DL:
  - Binary classification
  - Multiclass classification
  - Single label classification
  - Multilabel classification
  - Scalar regression
  - Vector regression

Revision - ANN

- Neural network is the basis of deep learning
- ANN can be used for regression and classification
- Simple ANN = single layer, feed-forward neural network = multilayer perceptron (MLP) = shallow neural network
- ANN is formed of:
  - Input layer
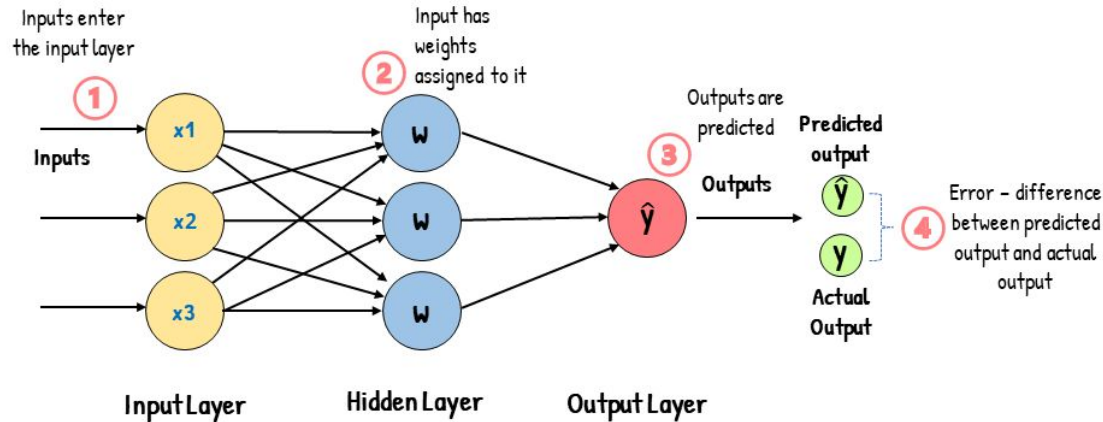 - Hidden layer
  - Output layer

- Breakdown of ANN:

Bias
b

Activation

Output

x_1

w_1

x_2

w_2

$\cdot$
$\cdot$
$\cdot$

w_n

Σ

$g(.)$

$\hat{y}$

x_n

Weights

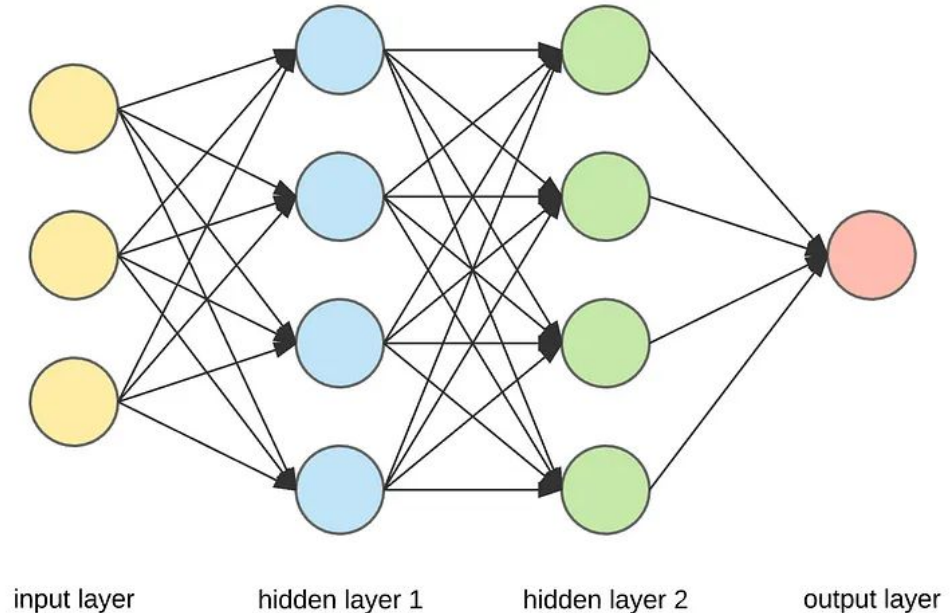$\hat{y} = g(\mathbf{w} \cdot \mathbf{x} + b)$

Inputs

- Simple ANN uses feed-forward neural network approach


Feed-Forward Neural Network

Deep Neural Network (DNN)

- Shallow neural network/ simple ANN, at most has 3 layers (input, output, and hidden layer)
- DNN has more than 3 layers:
  - "Deep" ≈ more layers

input layer   hidden layer 1   hidden layer 2   output layer

Basic concepts in DNN

Activation function

- To restrict the overall output values
- Example:
    - Sigmoid: for binary classes
    - Softmax: for multiclass
    - Rectified linear unit (ReLU)
        - Solve the issue of vanishing gradient (faster convergence)
        - Introduce non-linearity into the network - enabling the network to learn a complex pattern in the data

sigmoid
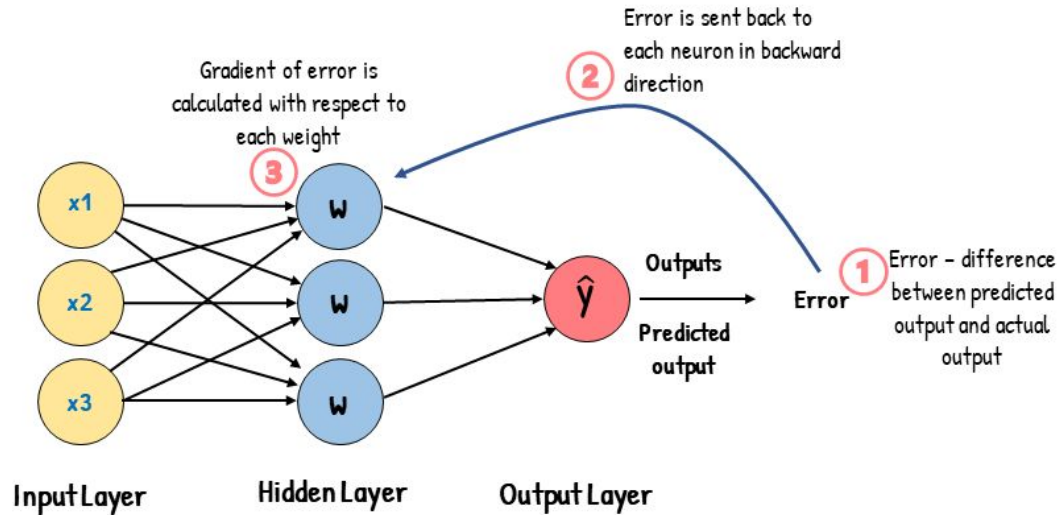
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

ReLU

$$R(z) = max(0, \; z)$$
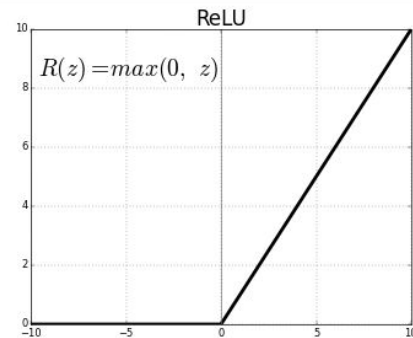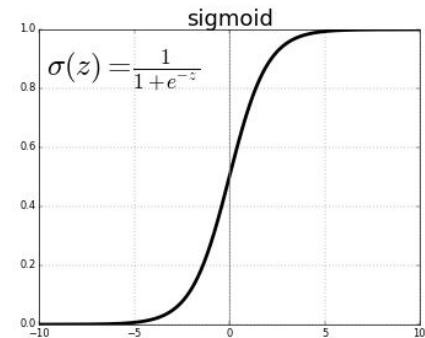
Loss/cost function

- A value to be minimised during the training process

| Problem type	Last-layer activation	Loss function
Binary classification	sigmoid	binary_crossentropy
Multiclass, single-label classification	softmax	categorical_crossentropy
Multiclass, multilabel classification	sigmoid	binary_crossentropy
Regression to arbitrary values	None	mse
Regression to values between 0 and 1	sigmoid	mse or binary_crossentropy

<u>Backpropagation</u>

- This an iterative process of fine-tuning the weights of the network using gradient descent (an optimisation algorithm)
- Process: It operates in two phases:
  - Forward Pass:
    - The input data is passed forward through the network, layer by layer, producing an output
    - The output is compared with the desired output to compute the error
  - Backward Pass:
    - The error is propagated backward through the network, adjusting the weights to minimize it

Backpropagation

Gradient of error is
calculated with respect to
each weight
③

Error is sent back to
each neuron in backward
direction
②

Input Layer

x1

x2

x3

Hidden Layer

w

w

w

Output Layer

ŷ

Outputs

Predicted
output

Error

①

Error – difference
between predicted
output and actual
output

Gradient descent

- An optimization algorithm used in DL to minimize the error between predicted and actual results
- Example:
    - RMSprop (Root Mean Square Propagation)
        - Usually works for most tasks
        - Some suggestion: γ to be set to 0.9, while a good default value for the learning rate η is 0.001
    - ADAM (Adaptive Moment Estimation)
        - Some suggest to always use ADAM (Adam usually outperforms the rest)
        - Some suggestion: 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ε

SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
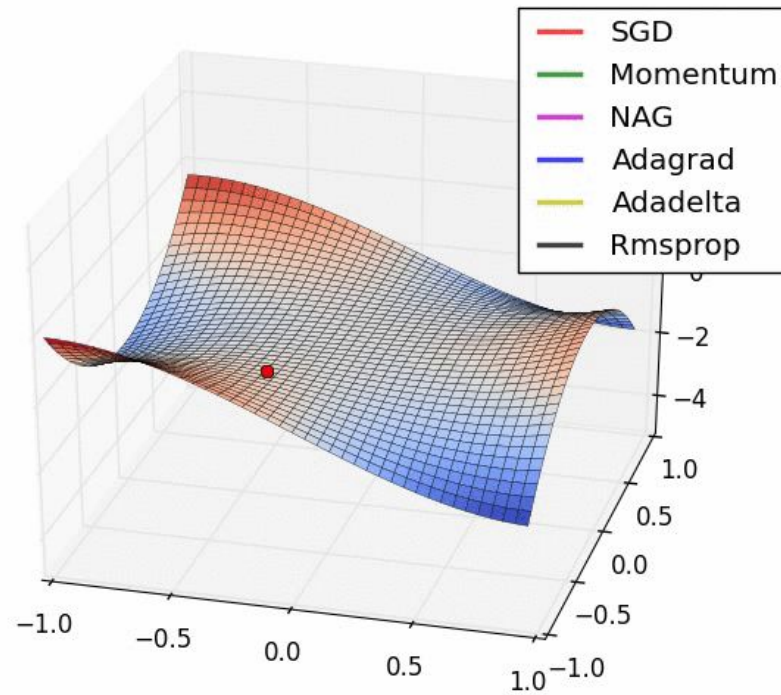
Validation approaches

- Hold-out validation - when we have a large dataset
- k-fold cross-validation - when we have a small dataset
- Iterated k-fold cross-validation with shuffling - when we have a small dataset and we need to evaluate the model as precise as possible

Approach to DL

- Set a baseline to beat
- Overfit the network on the training data first
- Then, reduce the overfitting by improving the generalisation
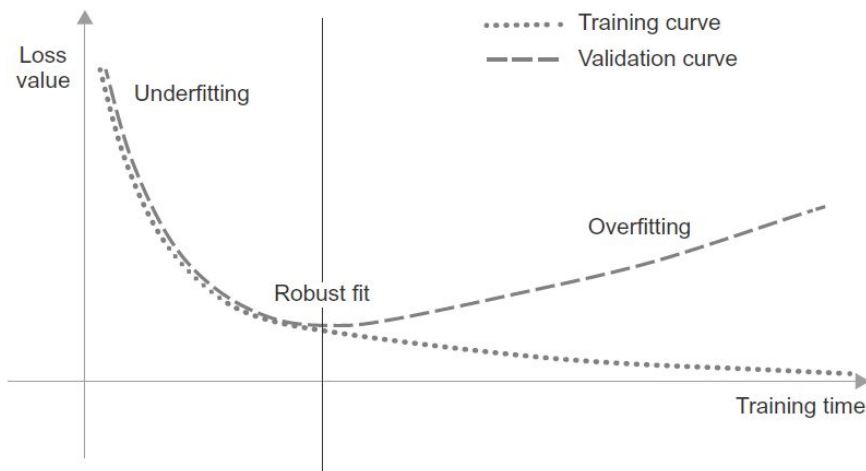
- How to overfit?
  - Add more layers
  - Make the layers bigger
  - Train for more epochs
- How to improve generalisation?
  - Reduce the layers
  - Use regularisation methods
  - Improve the data:
    - Develop better features - feature engineering
   - Remove uninformative features
    - Collect more data

Regularisation methods

- Regularisation methods are applied to avoid overfitting
- Methods:
  - Reduce the network size
  - Weight regularisation - for small deep learning model
    - L1 regularisation
    - L2 regularisation
  - Dropout layer - for large deep learning model
    - Usually between 0.2 - 0.5

DL packages in R

- Keras
- Tensorflow
- Torch
- Tabnet (basic):
  - Use torch for backend
  - Works well with tidymodels
  - For regression, classification, and time series
- Brulee (basic):
  - Use torch for backend
  - For regression and classification

DNN workflows in keras
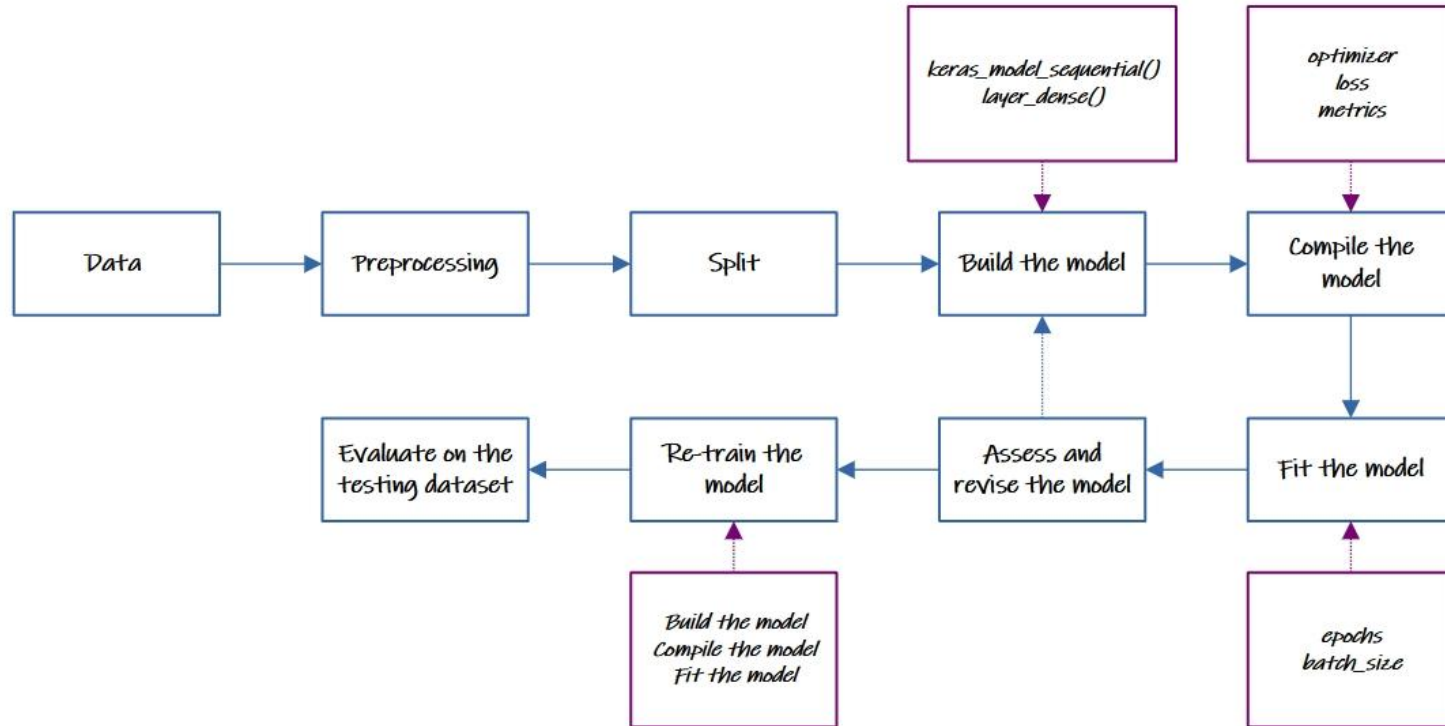
Running a DL analysis

- Bigger data
- Computational resources:
  - Laptop/PC with GPU
  - Cloud
    - [SaturnCloud](#)
    - [AWS Amazon](#)
    - [Google Colab](#)
    - Etc
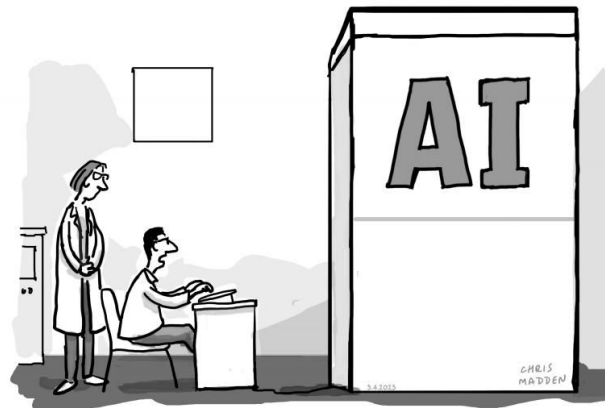
"We've got a problem. I've turned it on but I can't turn it off again."

Suggested readings/references

- Chollet, F., Kalinowski, T., & Allaire, J. J. (2022). Deep learning with R (Second edition). Manning.
- Keydana, S. (2023). Deep Learning and Scientific Computing with R Torch. Chapman and Hall/CRC.
- Kalirane, M. (2023, April 5). Gradient Descent vs. Backpropagation: What's the Difference? Analytics Vidhya. https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/
- Ruder, S. (2020, March 20). An overview of gradient descent optimization algorithms. Ruder.Io. https://www.ruder.io/optimizing-gradient-descent/

Any question?

tengkuhanismokhtar@gmail.com
jom.research.malaysia@gmail.com