DATA SCIENCE & MACHINE LEARNING WORKSHOP

Week 6 - Intro to Deep Learning

AGENDA

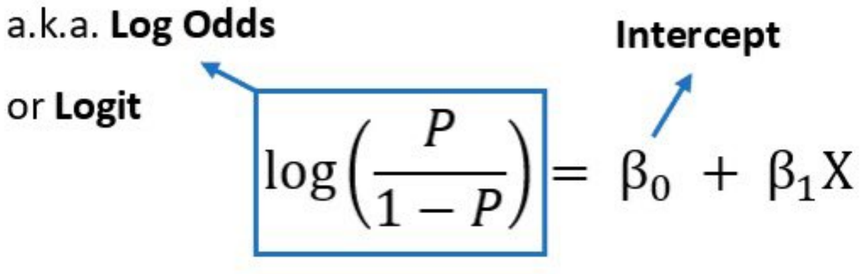
Week 6 topic:

- Motivation and Intro to Artificial Neural Network
- Types of Deep Learning Architecture
- Evaluating Deep Learning Models
- Optimisation and Gradient Descent
- Unsupervised Learning with Deep Learning

Repository:
https://github.com/AlgoSoc/Data-Science

MOTIVATION

Linear Model for classification:



In logistic regression, we model the log-odds through a linear relationship with the input space X with (beta_0 and beta_1 as the parameters). If we rearrange it to the probabilities, we'll get:

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 X$$

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 X}$$

$$p = (1 - p)\, e^{\beta_0 + \beta_1 X}$$

$$p = e^{\beta_0 + \beta_1 X} - pe^{\beta_0 + \beta_1 X}$$

$$p\,(1 + e^{\beta_0 + \beta_1 X}) = e^{\beta_0 + \beta_1 X}$$

$$p = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

Sigmoid Function

MOTIVATION

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 X$$

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 X}$$

$$p = (1 - p)\, e^{\beta_0 + \beta_1 X}$$

$$p = e^{\beta_0 + \beta_1 X} - p e^{\beta_0 + \beta_1 X}$$

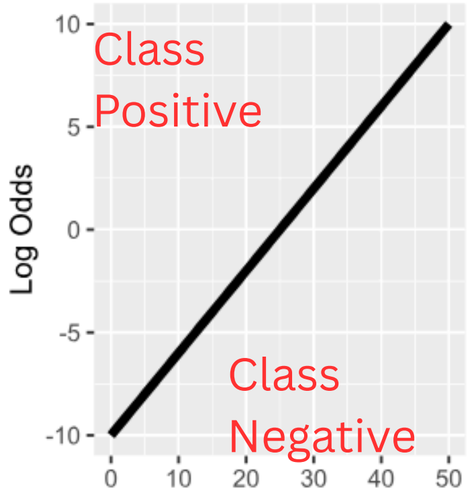
$$p\,(1 + e^{\beta_0 + \beta_1 X}) = e^{\beta_0 + \beta_1 X}$$

$$p = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

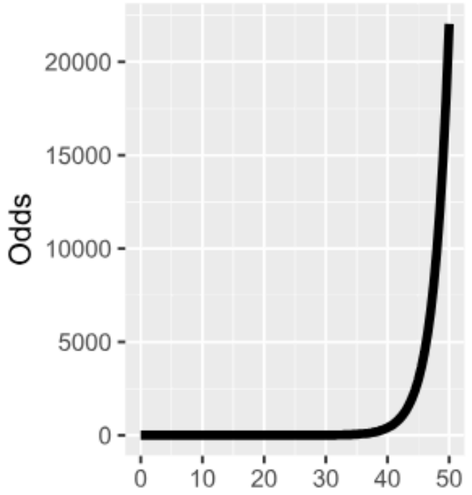
Sigmoid Function

Visually

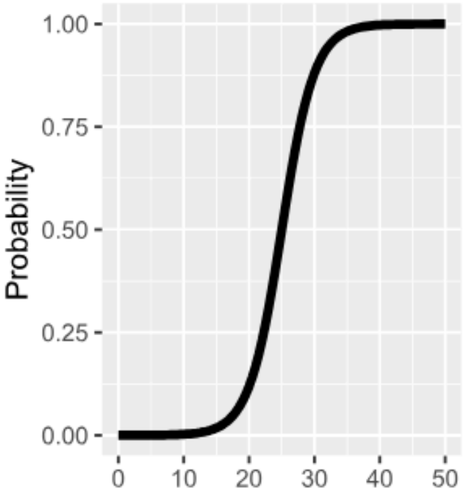
log(p/(1-p))



p/(1-p)

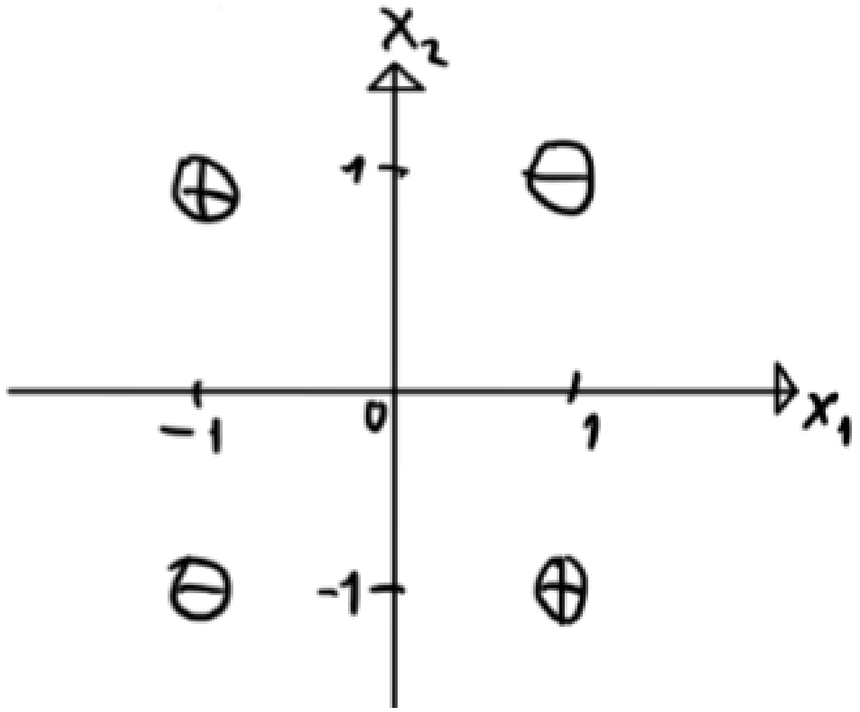


p



We can see that the decision boundary (for the log odds) is applied for linearly separable data

MOTIVATION

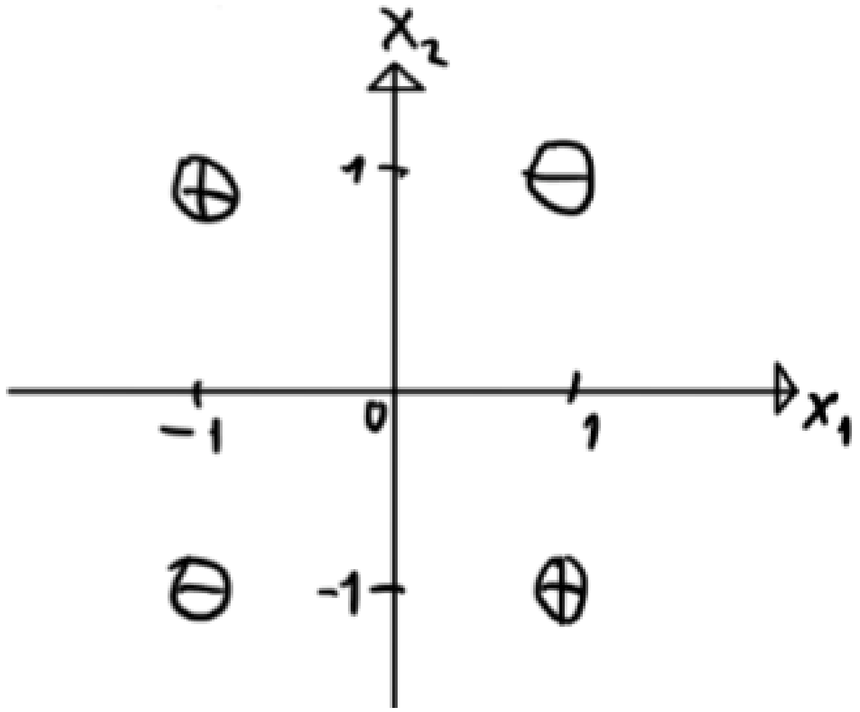


But what if the data are not linearly separable like this example?

Mental exercise: try drawing a straight line to separate the negative and positive classes

Taken from Week 1 exercise of Alexander Krull's Neural Computation module at University of Birmingham

MOTIVATION



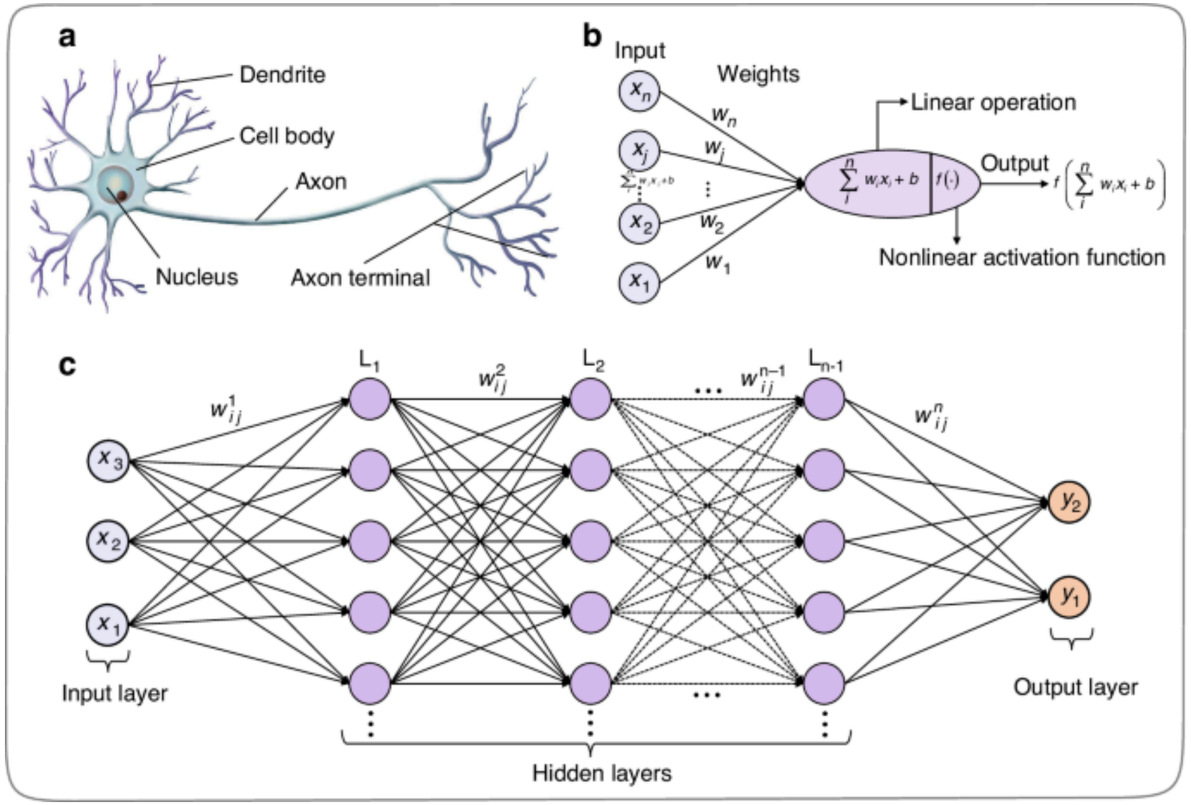


Image Source: https://www.nature.com/articles/s41377-024-01590-3

We model them with multi-layered artificial neural network (ANN), where we can use multiple linear representations to model non-linear relationships

Taken from Week 1 exercise of Alexander Krull's Neural Computation module at University of Birmingham

MOTIVATION

However in recent years, deep learning with its ability to process unstructured data (e.g., images, texts), are being considered as black boxes.

this is because, we don’t really know the representation of information they are learning inside their “hidden layers” (simplified, current models have more complex architectures)

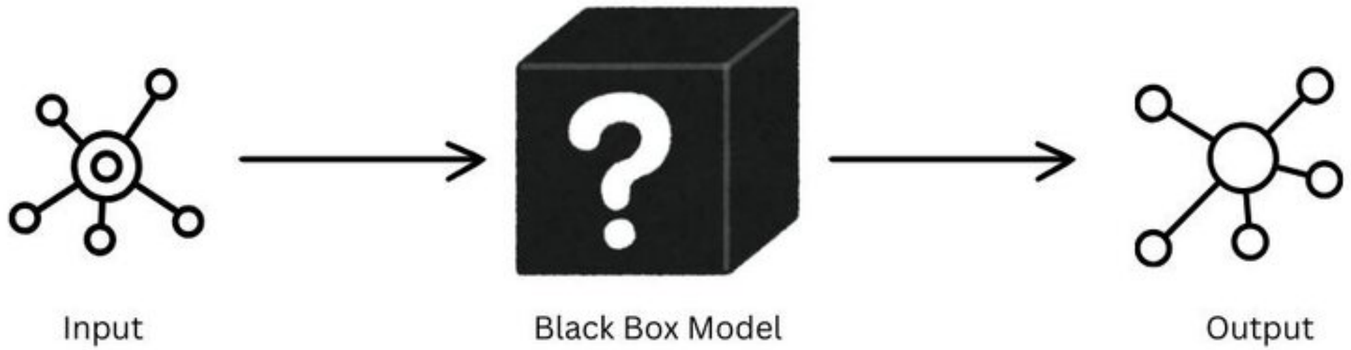


Image Source: https://www.researchgate.net/figure/The-Black-box-nature-of-deep-learning-models-Source-The-Researchers_fig3_385892716

MOTIVATION

ANNs and deep learning models can act as a basis transformation similar to SVMs but instead of using one step kernel operation, the transformation is **learned through the hidden layers.**

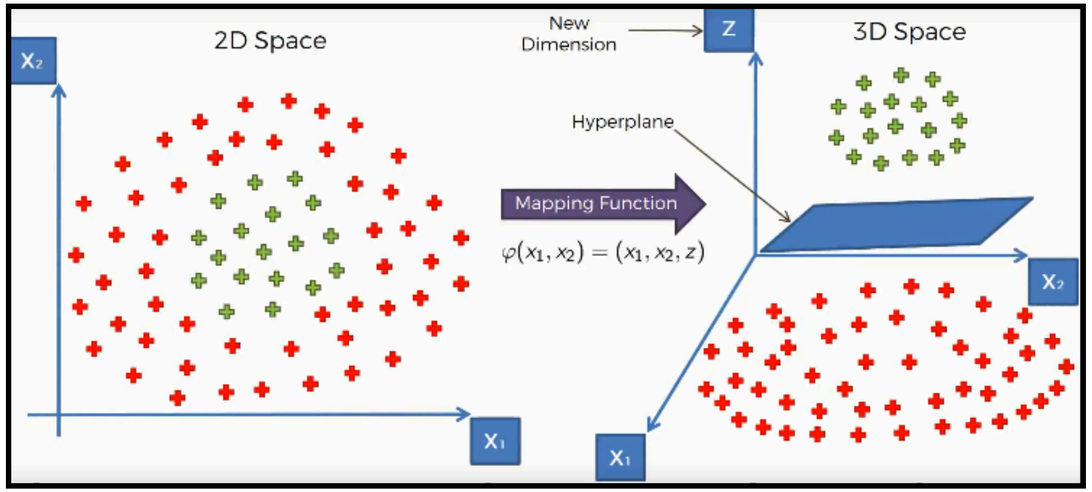


Image Source: https://medium.com/@rdhawan201455/support-vector-machines-tutorial-intuition-and-maths-beginners-88b6594fad20

$$\phi(x) \rightarrow w^\top \phi(x) + b$$

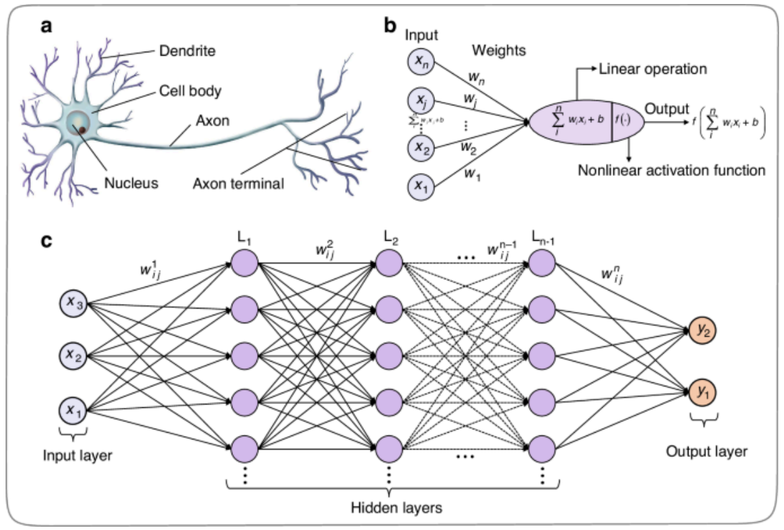


Image Source: https://mriquestions.com/deep-network-types.html

$$x \rightarrow f(W_2 f(W_1 x + b_1) + b_2)$$

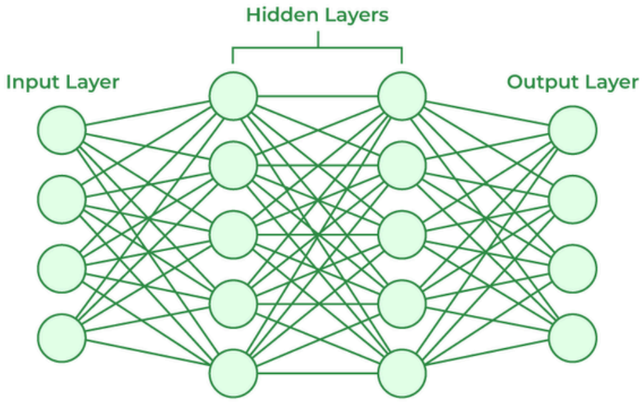
Phi(X) is a fixed function, transforming input X into a new feature space while W and b (parameters) in deep learning models are learned.

MOTIVATION

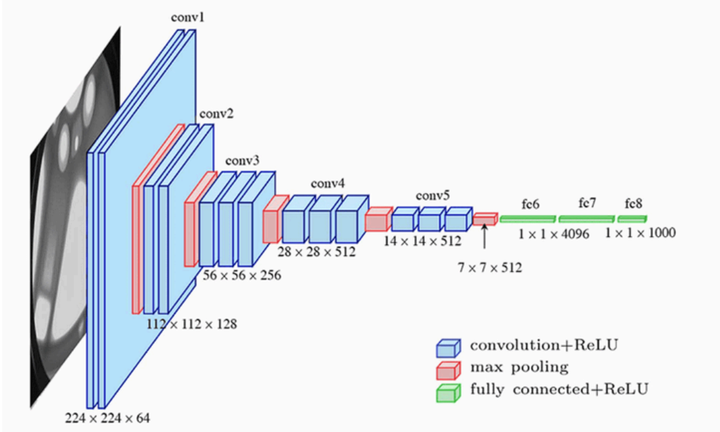
Using the adaptive ‘learning’ on extracting information of data, we can utilise deep learning models to perform operations on unstructured data (i.e., images, texts, signal).

This is because we can have the target value (e.g., class) and let the network handle the information extraction by itself where it would (hopefully) find the optimal information to be extracted.

VARIOUS DEEP LEARNING ARCHITECTURES



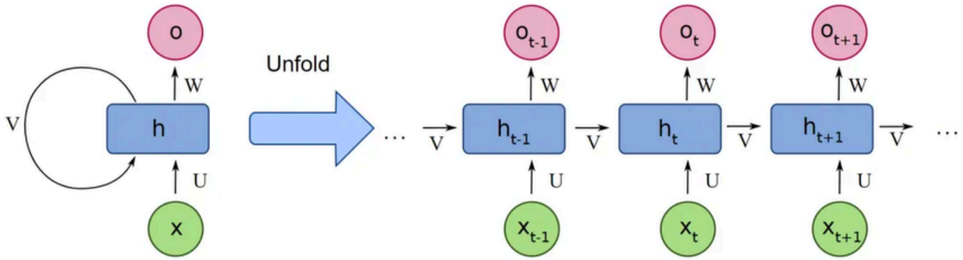
Multi-Layered Perceptron or ANN



Convolutional Neural Networks (Special Case of ANN) → work well for images

There are more types of deep learning architecture:

- Attention models
- Latent models
- Neural ODEs
- Residual Networks
- and many more...



Recurrent Neural Netowrks

- Takes input from the previous output and use the same set of weights
- Works when we have time-series data

EVALUATING DL MODELS

Similar to standard machine learning, we have **loss functions.** We will be focusing on **binary classification**

However, instead of using accuracy, we will be introducing binary cross entropy:

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

- Y_i is the true class (value would be 1 or 0)
- \hat{Y}_i is the predicted probability (value would range between 1 and 0)

The more correct our model is, the lower the loss would be.

OPTIMISATION

The goal of supervised learning is to learn a function (in our case parameters of our predictor). One way to learn it is through **gradient descent** by minimising the loss function

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

In a nutshell:

📌 Gradient Descent Algorithm (Step-by-Step)

1. **Initialize** parameters θ randomly
2. **Repeat until convergence**:

   a. Compute the gradient:

   $$g = \nabla_\theta L(\theta)$$

   b. Update parameters:

   $$\theta \leftarrow \theta - \alpha g$$

3. Stop when:
   - Gradient becomes small
   - Loss stops decreasing
   - Max iterations reached

OPTIMISATION

What if the network is too big?

We have an algorithm called backpropagation by using chain rule for derivatives

$$\frac{dy}{dx} = \frac{dy}{du} \times \frac{du}{dx}$$

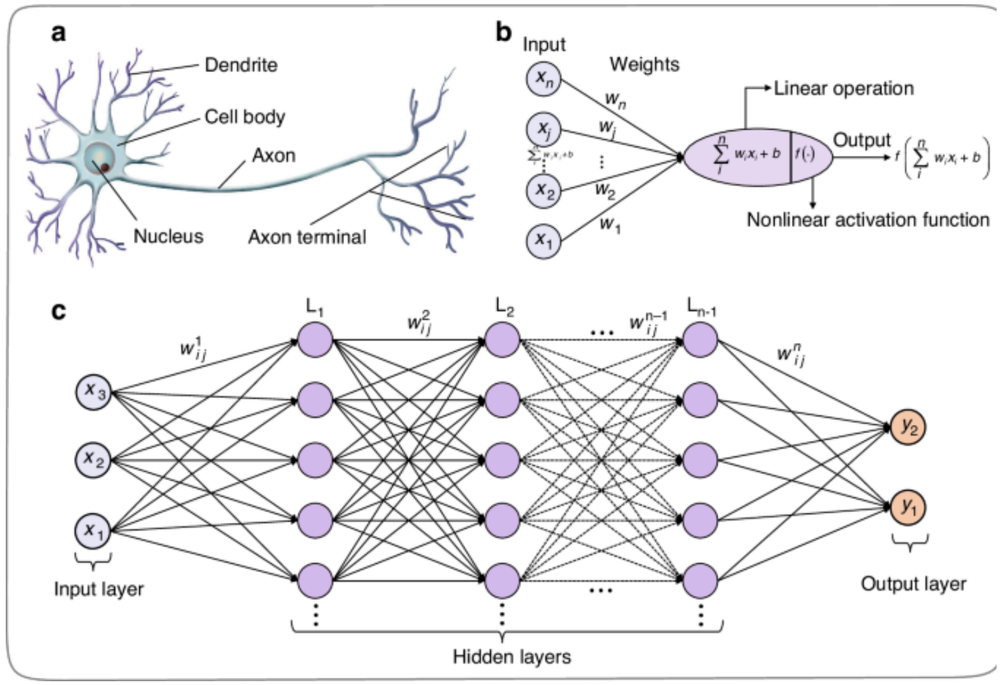


Image Source: https://mriquestions.com/deep-network-types.html

$$\frac{\partial C}{\partial w_1} = \frac{\partial z_1^{(1)}}{\partial w_1} \times \frac{\partial a_1^2}{\partial z_1^{(1)}} \times \frac{\partial C}{\partial a_1^2}$$

Partial derivative of hidden layer dot product w.r.t weight 1

Partial derivative of hidden layer activated output w.r.t dot product z

Partial derivative of Cost, C w.r.t activated 'hidden layer' output

Video explanation from 3Blue1Brown: https://www.youtube.com/watch?v=tIeHLnjs5U8

UNSUPERVISED LEARNING

Think of this case:

- We have a lot of unsupervised data because it’s expensive to label them.
- What if we can train a network to “understand data” without the label
- Answer: Autoencoders

Base

Binary Mask

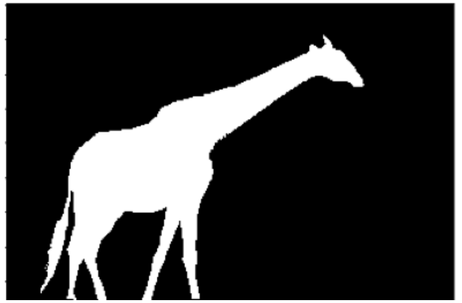


Image source: https://mxnet.apache.org/versions/1.2.1/tutorials/python/data_augmentation_with_masks.html

UNSUPERVISED LEARNING

Autoencoders can be used to learn from unlabelled data by trying to reconstruct data. The two components of autoencoders are:

- Encoder (f) → encoding data to latent space
- Decoder (g) → decoding data back

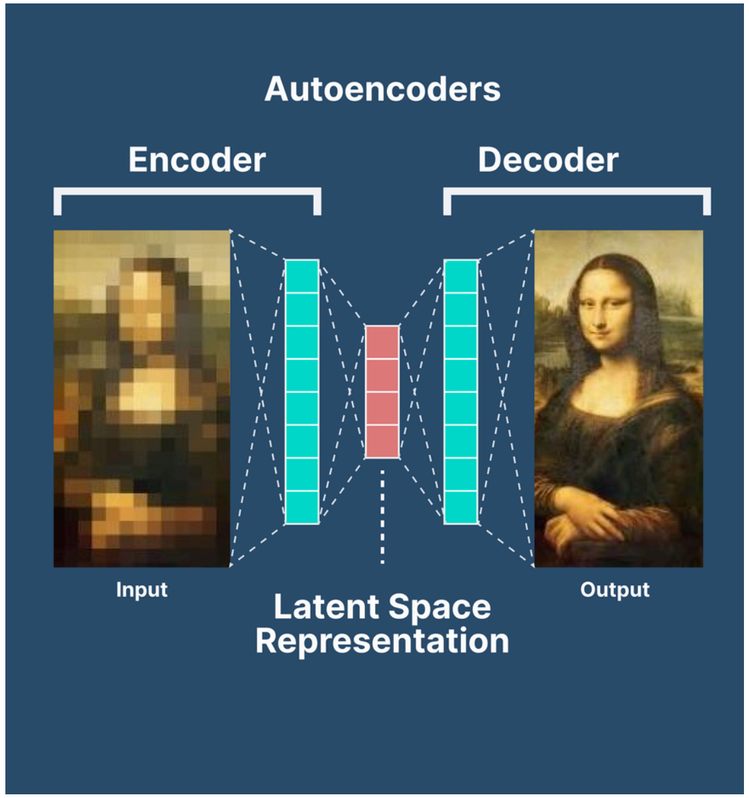


Image source: https://www.v7labs.com/blog/autoencoders-guide

UNSUPERVISED LEARNING

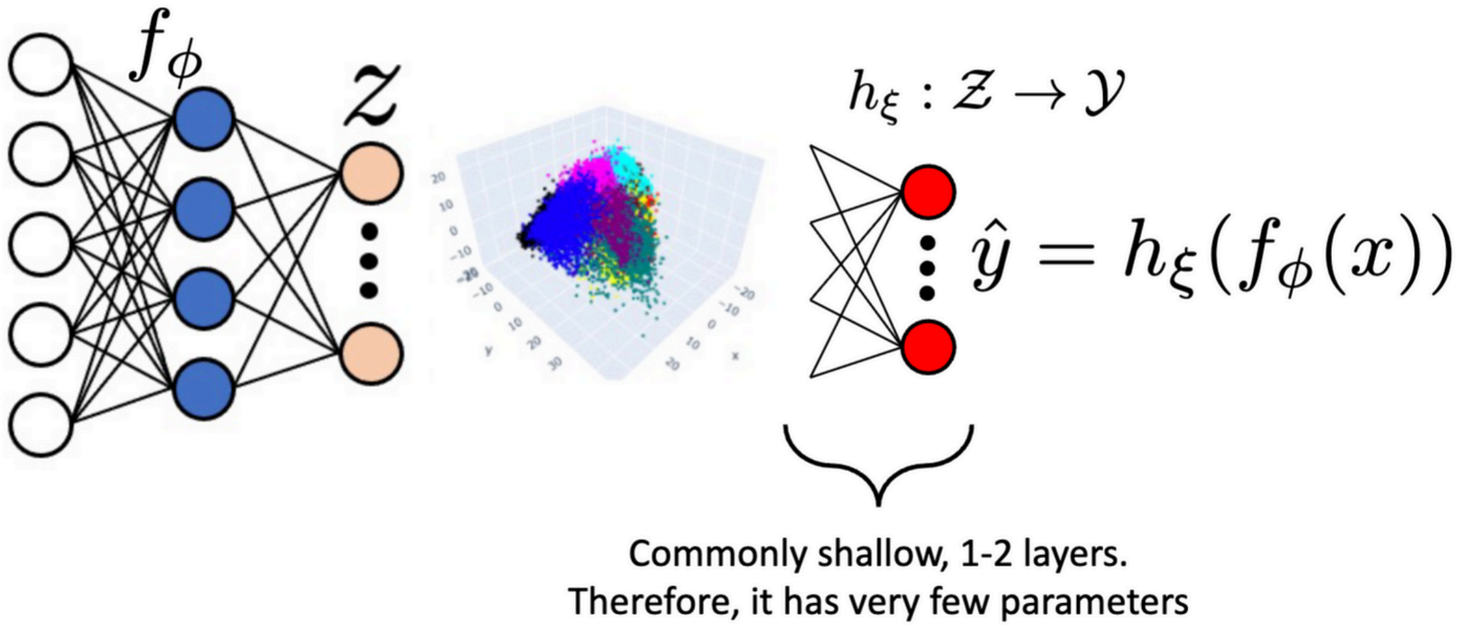
ALGOSOC
UOB

One of the loss function that we can use would be the MSE

$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

UNSUPERVISED LEARNING

We can then use the encoder to act as a feature extractor for our classification with the **limited labelled data.**



Taken from University of Birmingham, Neural Computation Module (25/26)

HANDS ON

https://bit.ly/AlgosocWk6

Special thanks to the concerned parties. A lot of materials here are inspired from University of Birmingham’s Neural Computation module, taught by the module lead, Alexander Krull in the 2025/2026 term.