# Aspects of Machine Learning Models

1 - input : batches = {samples, targets}, learning rate = λ, Parameters = Θ, loss = MSE, model = NN

init model(parameters);

for batch in data do

Compute output of the model, given the current parameters, using the model's forward pass:

output = NN(batch.samples, Θ)

Compute the loss between the target and output:

batch\_loss = MSE(output, batch.targets)

Compute the gradients of the parameters with respect to the loss:

gradients = gradient(batch\_loss, Θ)

Update the parameters by subtracting the gradients multiplied by the learning rate:

Θ = Θ - λ \* gradients

end

return model;

2- n = 4

Predicted = [2, 3, 5, 8]

Expected = [1, 2, 4, 5]

MSE = 1/4 \* ( (2 - 1)^2 + (3 - 2)^2 + (5 - 4)^2 + (8 - 5)^2 ) = 1.75

To minimise the MSE, the beta value should be updated such that it reduces the difference between the predicted and expected values.

This can be done using an optimization algorithm such as gradient descent, where the gradient of the MSE with respect to beta is calculated and subtracted from beta in each iteration, until a minimum is reached.

Beta = Beta - learning\_rate \* gradient(MSE, Beta)

# Neural Networks

1 - Convolutional Neural Networks (CNN)

Recurrent Neural Networks (RNN)

Convolutional Neural Networks (CNN): Image classification, object detection, image generation

Recurrent Neural Networks (RNN): Time series prediction, language modelling, speech recognition

2-An LSTM model is a type of Recurrent Neural Network (RNN) that can capture long-term dependencies in sequences. The key difference between an LSTM and a conventional RNN lies in the architecture of their hidden units. A conventional RNN has a single hidden unit that takes in information from the current input and previous hidden state, whereas an LSTM has three gates (input, forget, and output gates) that control what information to pass through to the next time step. This allows an LSTM to effectively store and retrieve information over a longer period, avoiding the vanishing or exploding gradients problem often encountered in conventional RNNs.

3-NLP tasks suitable for LSTMs:

Sentiment Analysis

Text Classification

Text Generation

4-Transformers handle sequential information by using a self-attention mechanism. Instead of processing the sequence one element at a time like RNNs and LSTMs, transformers process the entire sequence in parallel and use attention scores to weight the contribution of each element to the output. This allows transformers to effectively capture dependencies between elements regardless of their position in the sequence, making them well-suited for NLP tasks such as machine translation, text classification, and question answering.

# Neural Networks II

1-Commonly used activation functions:

Sigmoid (0 <= f(x) <= 1)

ReLU (0 <= f(x) <= x)

Tanh (-1 <= f(x) <= 1)

Leaky ReLU (f(x) >= 0 if x >= 0, f(x) = 0.01x if x < 0)

Softmax (All output values are non-negative and sum to 1)

2-The vanishing gradient problem refers to the difficulty in training deep neural networks when the gradients of the weights become very small during backpropagation. This makes it difficult for the optimization algorithm to update the weights and the model may fail to converge or learn anything useful. The problem arises because the gradients are multiplied multiple times in the backpropagation process and can become very small, effectively "vanishing" and making it hard for the optimization algorithm to make progress.

3-Finding a fitting learning rate is hard because it depends on the specific task and the model being trained. A learning rate that is too small may result in slow convergence and a long training time, while a learning rate that is too large may cause the optimization algorithm to oscillate or even diverge. A badly chosen learning rate can also lead to suboptimal model performance or a failure to converge. The problem can be tackled by using a learning rate schedule that adjusts the learning rate during training, or by using a more advanced optimization algorithm such as Adam, Adagrad, or RProp.

# Neural Networks III

1-The softmax function is used to turn a vector of real numbers into a probability distribution. The function takes a vector of logits (or raw predictions) as input and returns a vector of probabilities, with each element representing the predicted probability of a specific class. The softmax function is often used in the final layer of a neural network for multi-class classification problems, where the output of the network must be transformed into a probability distribution over the possible classes.

The softmax of the logits vector [3, 5, 6, 0] is calculated as follows:

softmax([3, 5, 6, 0]) = [e^3 / (e^3 + e^5 + e^6 + e^0), e^5 / (e^3 + e^5 + e^6 + e^0), e^6 / (e^3 + e^5 + e^6 + e^0), e^0 / (e^3 + e^5 + e^6 + e^0)]

= [0.0474, 0.8085, 0.1138, 0.0403]

The softmax of the logits vector [1, 0, 1, 2] is calculated as follows:

softmax([1, 0, 1, 2]) = [e^1 / (e^1 + e^0 + e^1 + e^2), e^0 / (e^1 + e^0 + e^1 + e^2), e^1 / (e^1 + e^0 + e^1 + e^2), e^2 / (e^1 + e^0 + e^1 + e^2)]

= [0.2113, 0.1585, 0.2113, 0.4189]

2-Overfitting occurs when a machine learning model is trained too well on the training data, capturing not only the underlying pattern but also the noise in the data. As a result, the model performs well on the training data but poorly on new, unseen data. Overfitting is caused by having too many model parameters relative to the amount of training data, leading to a model that is too complex and flexible.

Underfitting occurs when a machine learning model is not flexible enough to capture the underlying pattern in the data. This results in a model that performs poorly on both the training data and new, unseen data. Underfitting is caused by having too few model parameters, leading to a model that is too simple.

3- The network correctly classifies this data point as 1.

Data point 2: x0 = -1, x1 = 2

Input layer:

a0 = -1, a1 = 2

Hidden layer:

z1 = -3 + 4a0 + 2a1 = -3 + 4 \* -1 + 2 \* 2 = 1

z2 = -2 + 4a0 + 2a1 = -2 + 4 \* -1 + 2 \* 2 = 2

a1 = sgn(z1) = sgn(1) = 1

a2 = sgn(z2) = sgn(2) = 1

Output layer:

z = 0.5 + 1a1 + 1a2 = 0.5 + 1 \* 1 + 1 \* 1 = 2

o = sgn(z) = sgn(2) = 1

The network correctly classifies this data point as 1.

Data point 3: x0 = -3, x1 = 2

Input layer:

a0 = -3, a1 = 2

Hidden layer:

z1 = -3 + 4a0 + 2a1 = -3 + 4 \* -3 + 2 \* 2 = -5

z2 = -2 + 4a0 + 2a1 = -2 + 4 \* -3 + 2 \* 2 = -2

a1 = sgn(z1) = sgn(-5) = -1

a2 = sgn(z2) = sgn(-2) = -1

Output layer:

z = 0.5 + 1a1 + 1a2 = 0.5 + 1 \* -1 + 1 \* -1 = -2

o = sgn(z) = sgn(-2) = -1

The network correctly classifies this data point as -1.

# Language Models

1-Pre Training task for language models is typically Masked Language Modeling (MLM), where a percentage of tokens are randomly masked and the model is trained to predict the masked tokens based on their surrounding context.

2-Positional embeddings are vectors added to the input representation that encodes information about the relative or absolute position of each token in the input sequence. This is important for Transformer models as they are designed to handle variable-length input sequences, but lack the ability to retain sequential information through recurrent connections.

3-Four downstream tasks at token/text level include:

Named Entity Recognition (NER): classify named entities into predefined categories such as person, location, and organisation

Part-of-Speech Tagging (POS): assign a part-of-speech to each token in a sentence

Sentiment Analysis: classify the sentiment expressed in a text as positive, neutral, or negative

Question Answering (QA): predict the answer to a question based on a text document

4-Even the largest language models reach their limits when it comes to tasks that require extensive knowledge beyond the text they were trained on, such as reasoning, common sense understanding, and causal inference. Additionally, they may struggle with rare or unseen words, handling out-of-vocabulary words, and maintaining consistency in their outputs when generating text.