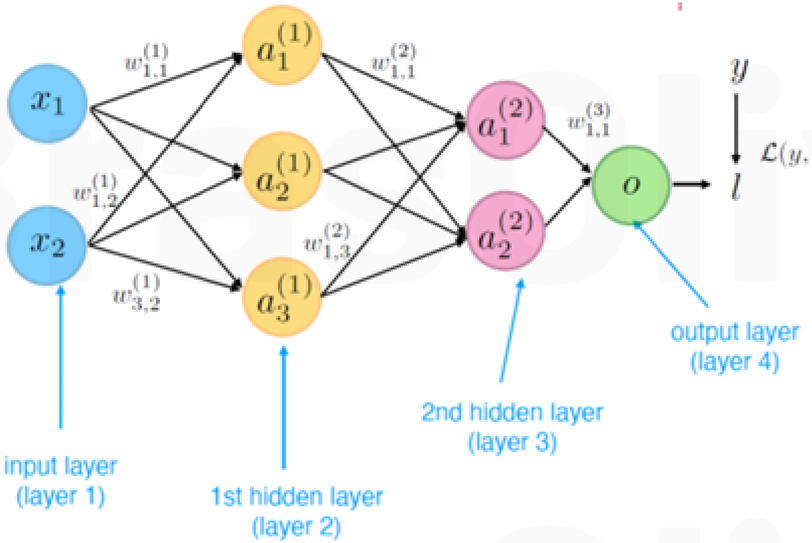
Artificial Neural Networks (ANN)

What are they?

Deep Learning (DL) models that:

Attempt to emulate the way the brain works through organized layers of neurons.



Notation:

- x_i inputs to the neural network (features).
- $z^{(l)}$ Weighted sum of inputs at layer (l)
- $a^{(l)}$ Activation output at layer (l).
- $W^{(l)}$ Weight matrix connecting layers.
- $b^{(l)}$ Bias vector added to neurons.
- $L(y, \hat{y})$ Loss function
- $\frac{\partial L}{\partial W_{ij}^{(l)}}$ Gradient of the loss
- $\delta^{(l)}$ Error term at layer (l)
- η Learning rate

Internal parameters (model specific) — Total # Weights + Total # Biases

Total # Weights = sum of the products of each pair of adjacent layers.

Total # Biases = # Hidden + Output neurons.

Forward (loss) **and Backpropagation** (gradient of loss)

Multiply inputs by weights & sum biases → Apply activation

Forward propagation

1st layer

$$z_1^{(1)} = W_{11}^{(1)} \cdot x_1 + W_{12}^{(1)} \cdot x_2 + b_1^{(1)} \qquad a_1^{(1)} = f(z_1^{(1)})$$

(...)

Output layer

$$z^{(3)} = W_{11}^{(3)} \cdot a_1^{(2)} + W_{12}^{(3)} \cdot a_2^{(2)} + b^{(3)} \qquad o = f(z^{(3)})$$

Computer errors and gradients → Weights and Biases update

Back propagation

$$\frac{\partial L}{\partial W_{ij}^{(3)}} = \delta^{(3)} \cdot a_i^{(2)} \qquad W_{ij}^{(3)} \leftarrow W_{ij}^{(3)} - \eta \cdot \delta_j^{(3)} \cdot a_i^{(2)}$$

$$\delta^{(3)} = \frac{\partial L}{\partial o} \cdot f'(z^{(3)}) \qquad b_j^{(3)} \leftarrow b_j^{(3)} - \eta \cdot \delta_j^{(3)}$$

(...)

$$\frac{\partial L}{\partial W_{ij}^{(1)}} = \delta_j^{(1)} \cdot x_i \qquad W_{ij}^{(1)} \leftarrow W_{ij}^{(1)} - \eta \cdot \delta_j^{(1)} \cdot x_i$$

1st layer

$$\delta_j^{(1)} = \left(\sum_k W_{kj}^{(2)} \cdot \delta_k^{(2)} \right) \cdot f'(z_j^{(1)}) \qquad b_j^{(1)} \leftarrow b_j^{(1)} - \eta \cdot \delta_j^{(1)}$$

Hyperparameters (user defined)

Architecture (structure and complexity of the neural network):

- Number of layers - Input layer + Hidden layers + Output layer;
- Number of neurons per layer;
- Activation function - Allows for non-linearities;
- Weights initialization - Specifies how weights are initialized.

Training Process (govern how the model learns and updates its parameters):

- Loss function - amount to be minimised during the learning process;
- Optimizer - Algorithm used for weight updates;
- Momentum - makes learning smoother and faster;
- Learning rate - pace at which the weights get updated;
- Epoch - a complete pass through the entire data set/training;
- Batch size - is the number of observations in a subset/partition.

Regularization (prevent overfitting and improve generalization):

- Dropout rate - Randomly turns off a rate of neurons during training;
- Early Stopping - Stops training when performance stops improving on validation data;
- Weight regularization - Implements a penalty for large weights.

Hyperparameter search/tuning (search for the best combination of hyperparameters).