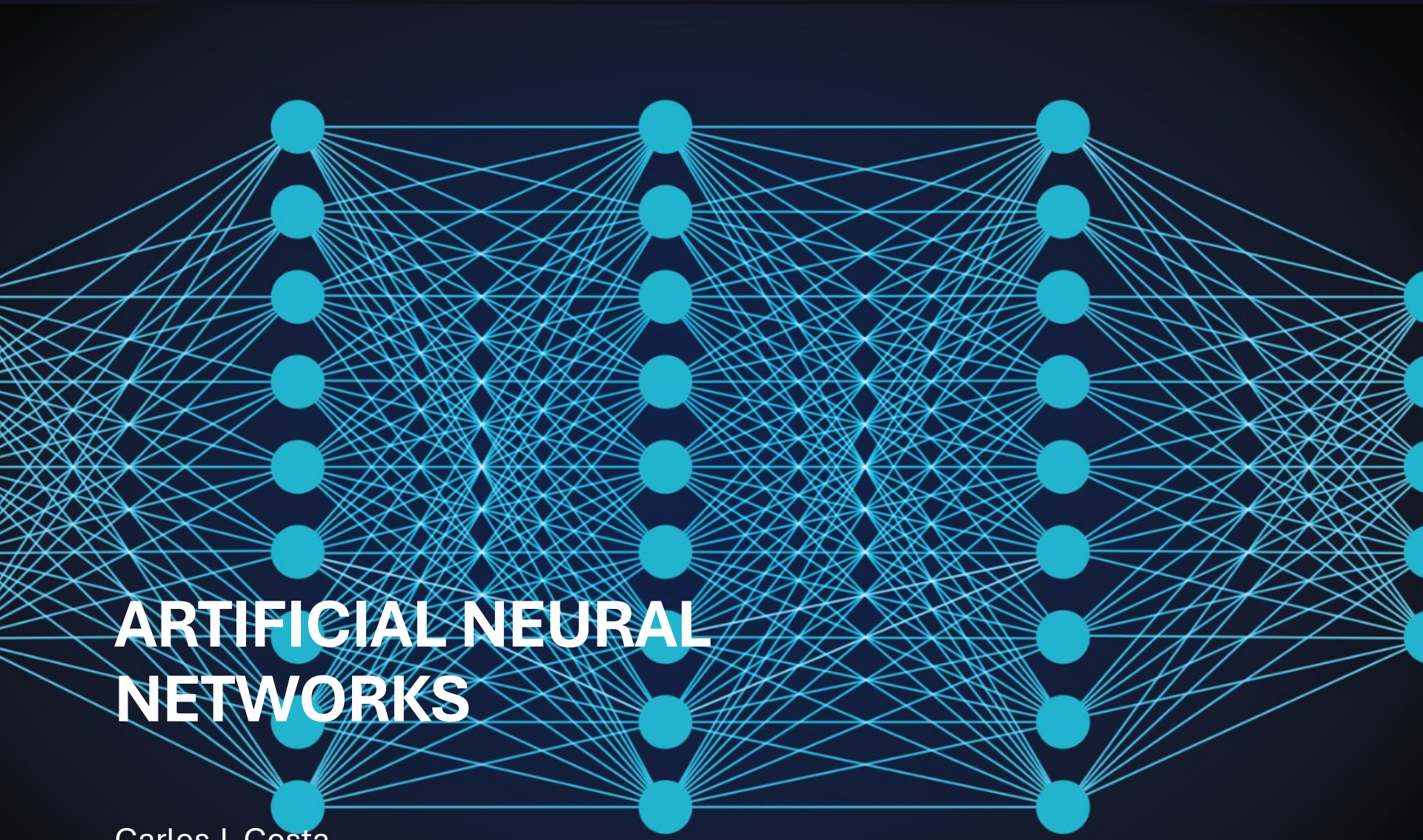
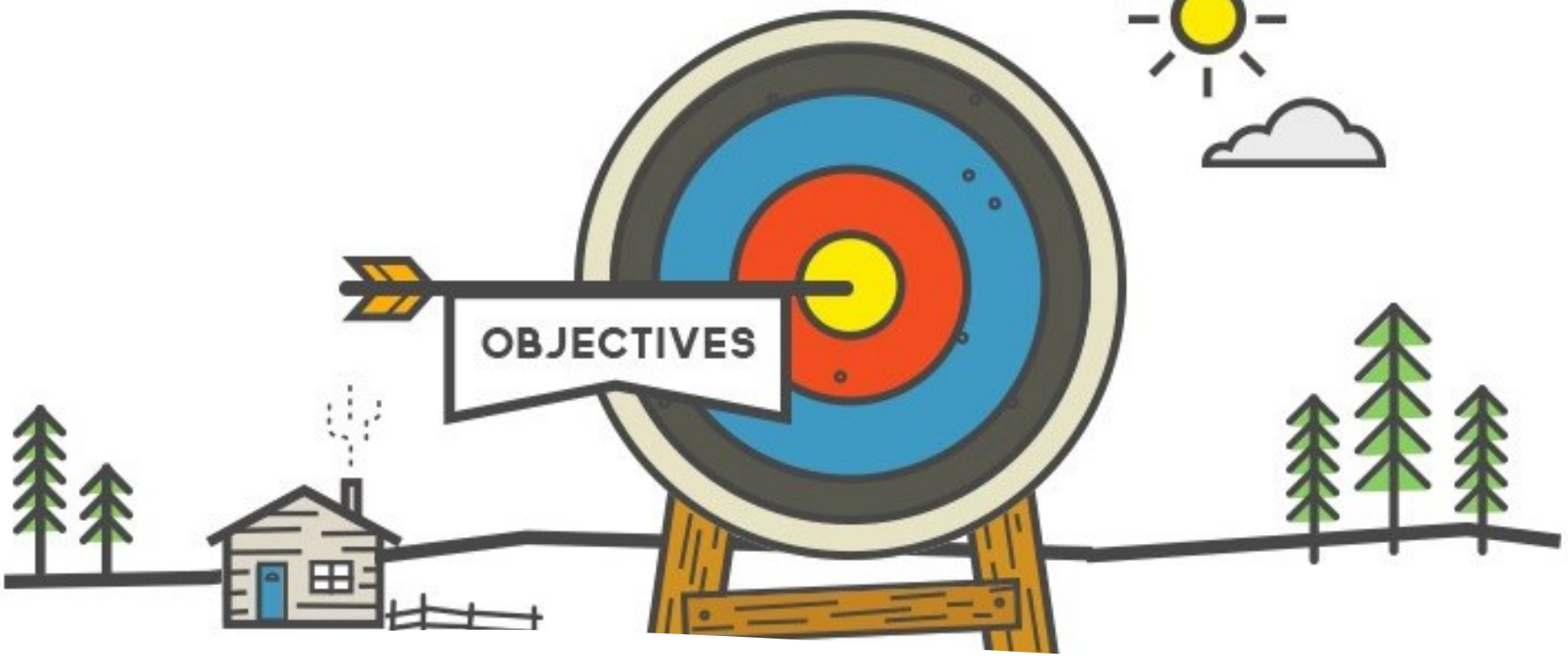


ARTIFICIAL NEURAL NETWORKS

Carlos J. Costa



Learning Goals

- Know concept of Neural Networks
- Know main components
- Apply algorithms by using python libraries

Carlos J. Costa, 2025

Agenda

- History o ANN
- Concept of neural network
- Feedforward neural networks and backpropagation
- Types of neural networks
- Implementations: TensorFlow, Keras, Scikitlearn

History of neural networks

- 1943 – Concept of Neural Network, Warren S. McCulloch and Walter Pitts
- 1958 – perceptron, Frank Rosenblatt
- 1974 - backpropagation Paul Werbos
- 1989 - Yann LeCun

Carlos J. Costa, 2025

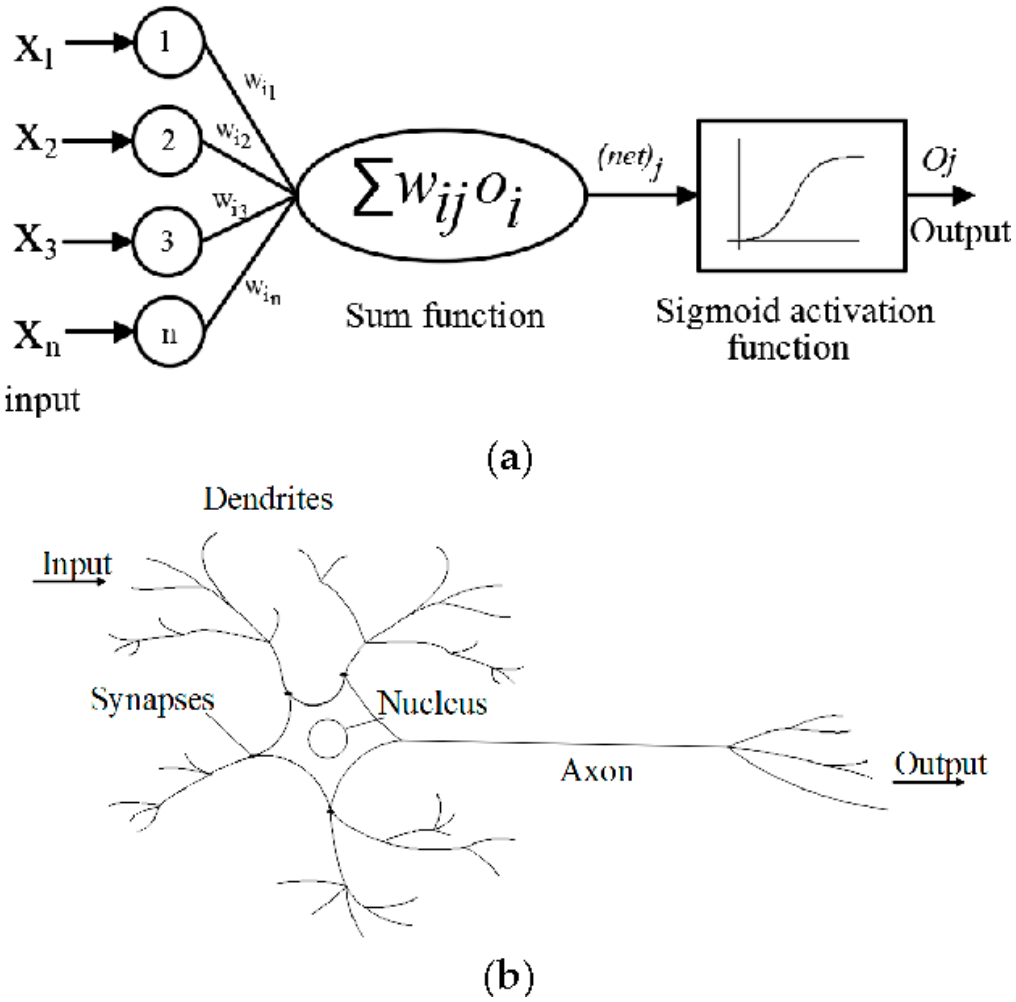
Artificial Neural Networks vs. Human Brain

Artificial neural networks (ANNs) consist of node layers, including an input layer, one or more hidden layers, and an output layer.

Each node, or artificial neuron, connects to another and has an associated weight and threshold.

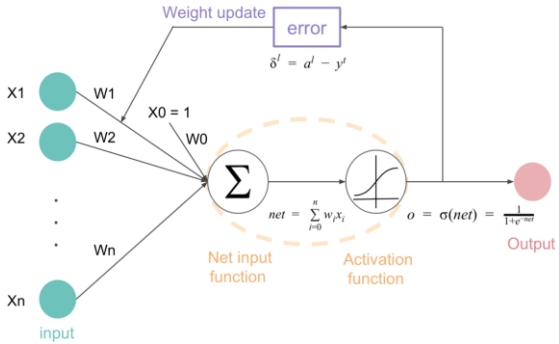
If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network.

Otherwise, no data is passed along to the next layer of the network.



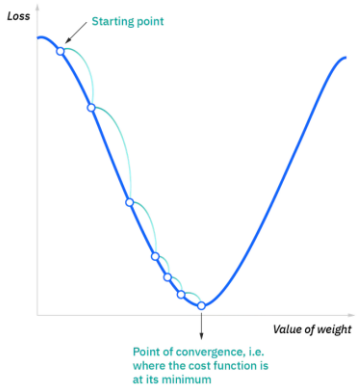
Carlos J. Costa, 2025

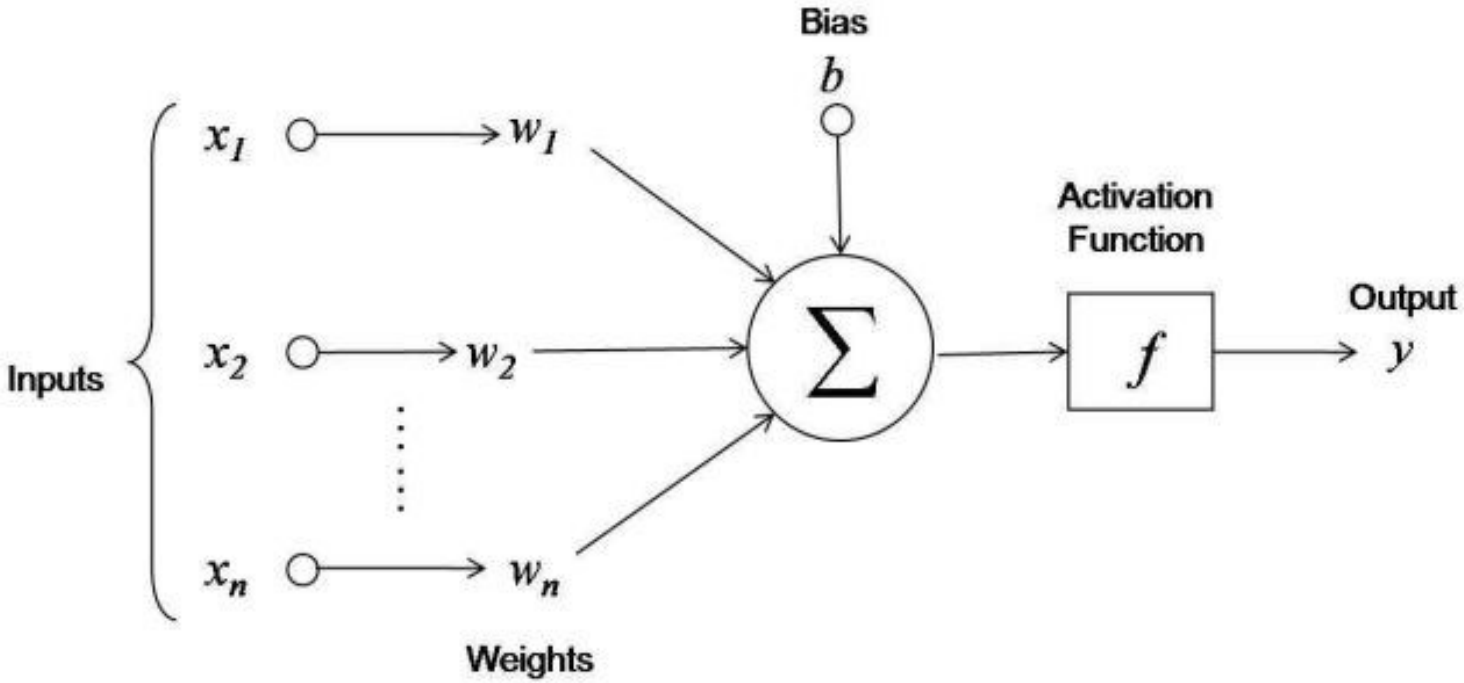
Basic Concepts



- Neuron
- Activation Function
- Loss Function
- Optimization

$$Cost\ Function = MSE = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y)^2$$



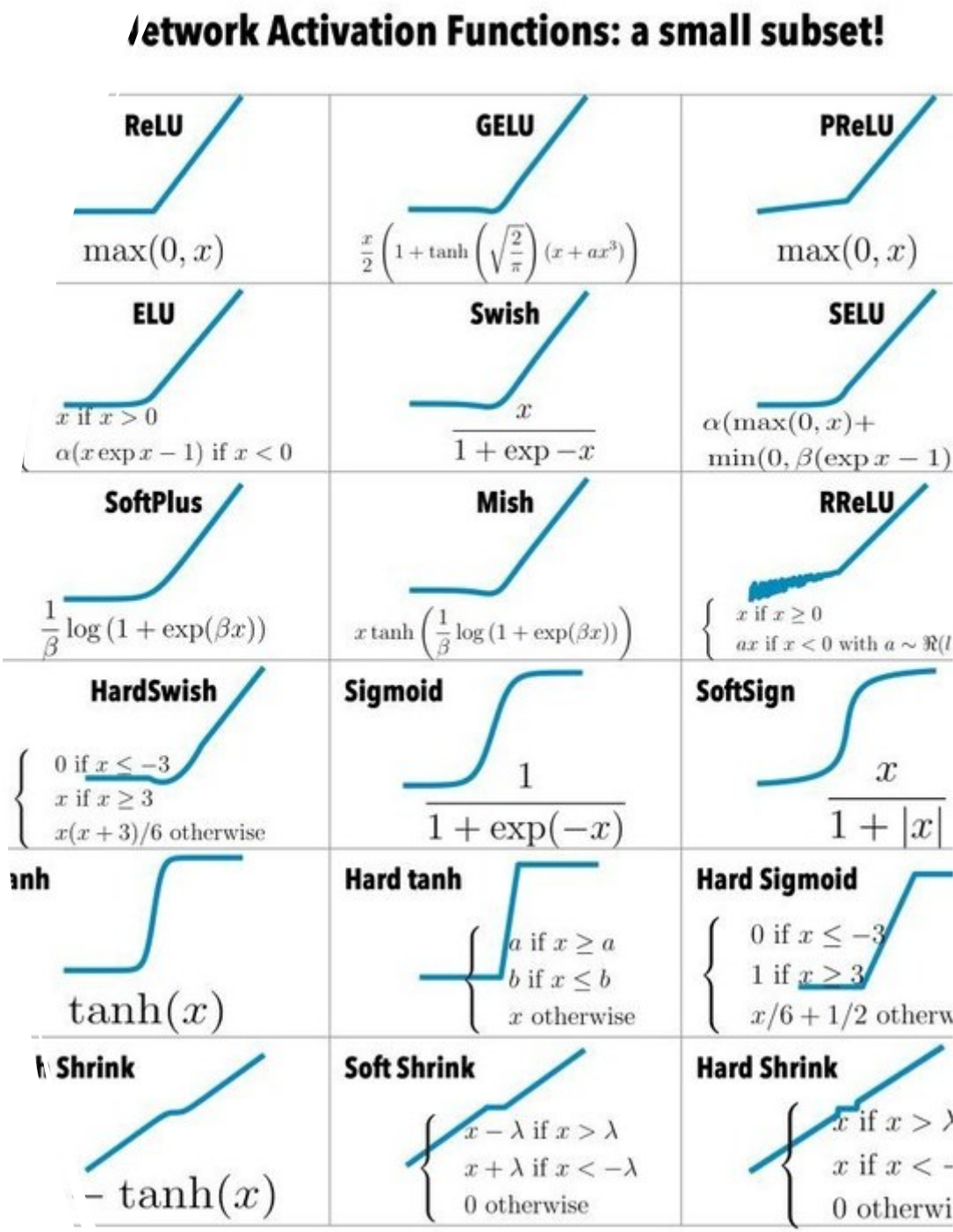


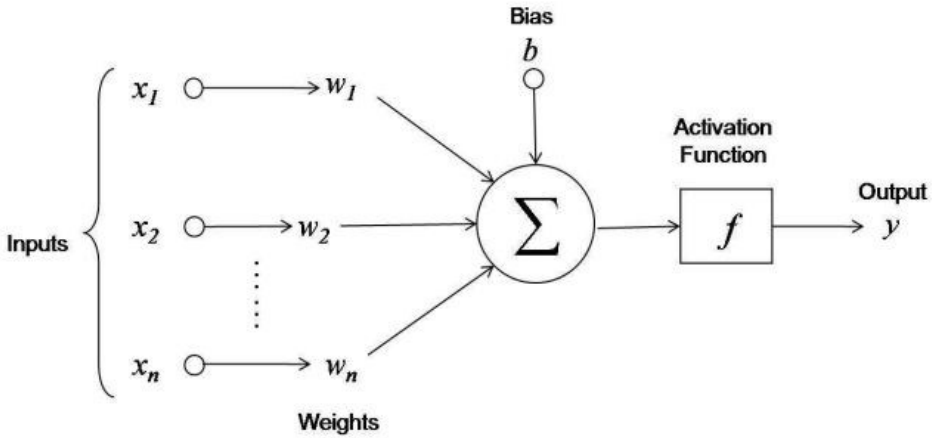
Neuron

- The basic computational unit of a neural network
- takes inputs
- applies weights and biases
- produces an output using an activation function.

Activation Function

- A function applied to the output of a neuron to introduce non-linearity, allowing the neural network to model complex relationships in the data





$$x = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

Then the activation (using the sigmoid function) is:

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

Here:

- x_i are the input features,
- w_i are the corresponding weights,
- b is the bias,
- $\sigma(x)$ is the neuron's output after applying the activation function.

For example:

Binary cross-entropy loss

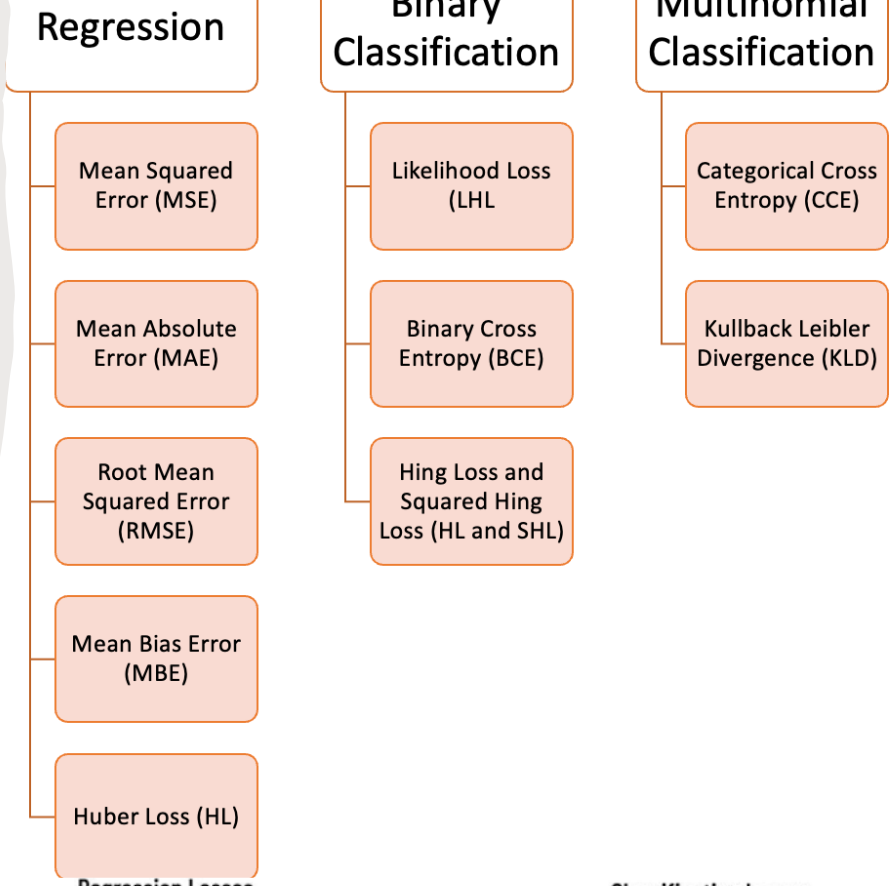
$$L = -[y_{\text{true}} \log(\hat{y}) + (1 - y_{\text{true}}) \log(1 - \hat{y})]$$

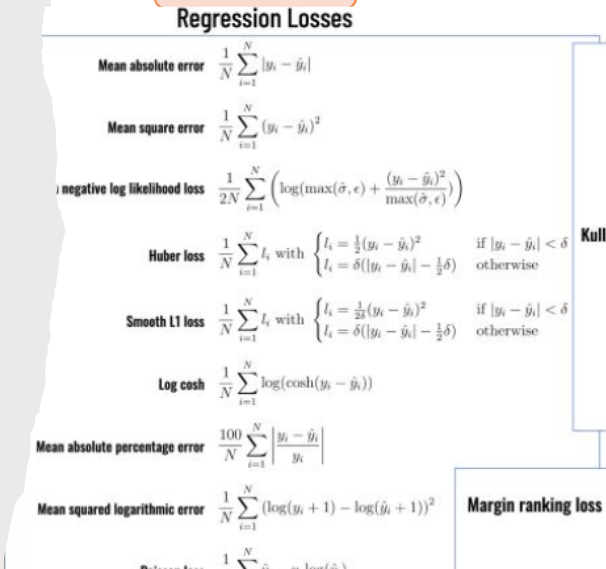
Here:

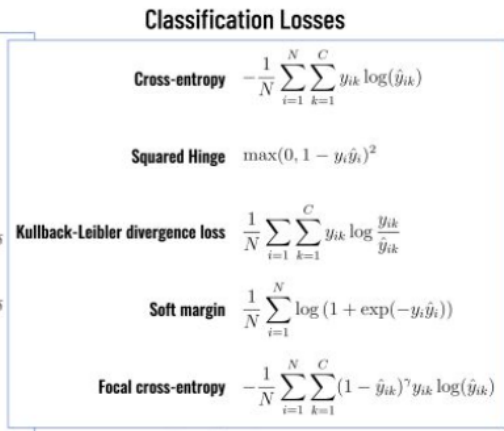
- $\hat{y}$ is the output of the activation function (e.g., sigmoid),
- y_{true} is the actual label (0 or 1),
- L is the loss that tells the model how wrong it is.

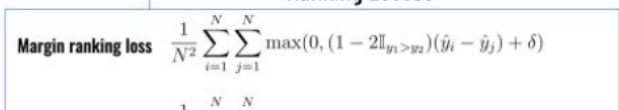
Loss Function

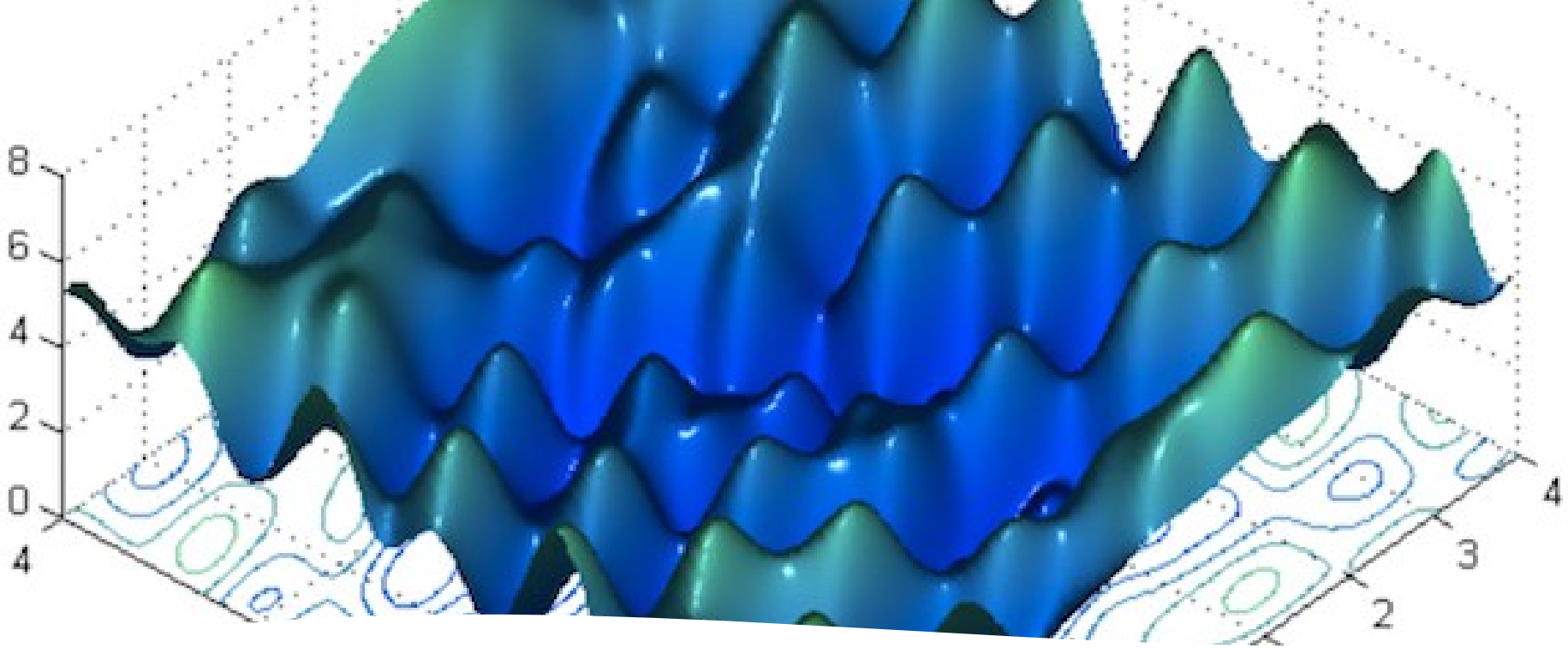
- Measures the difference between the predicted output of the neural network and the true labels
- Used in training data,
- Guides the optimization process.





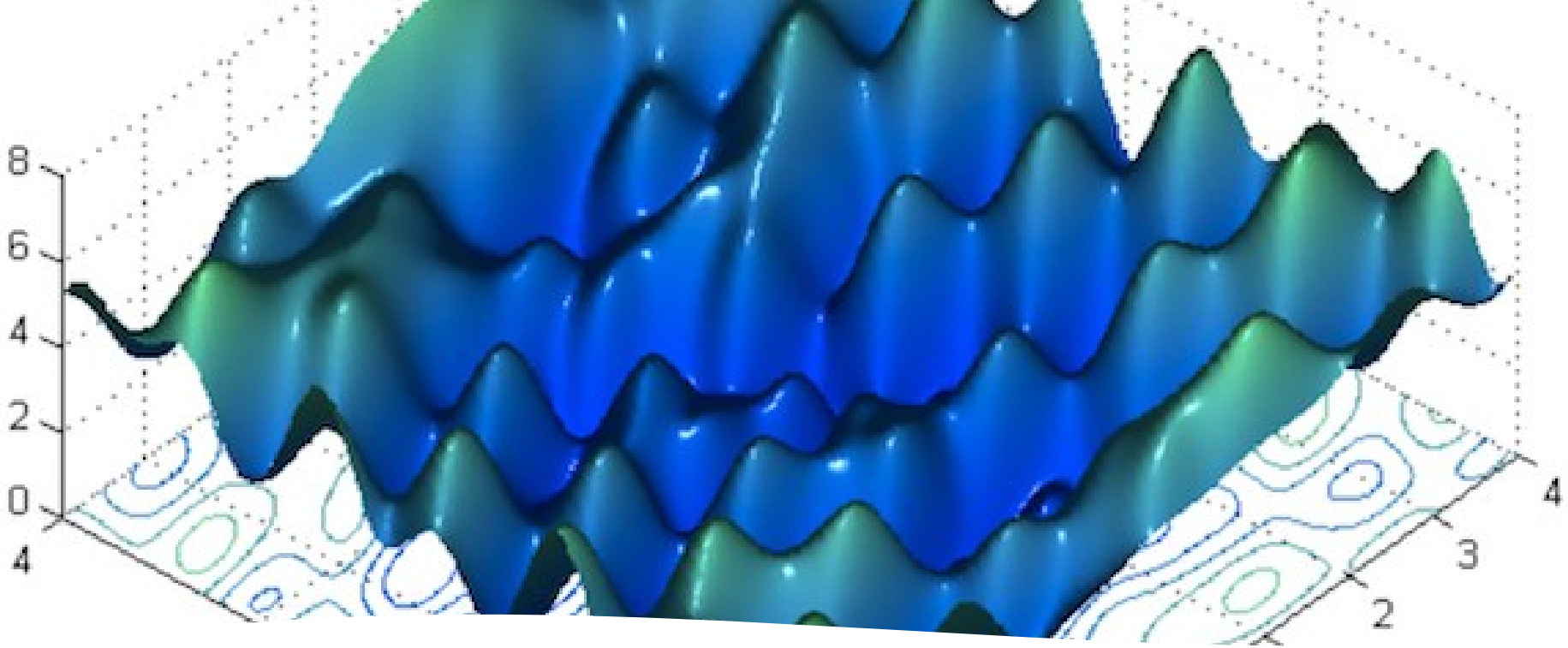






Optimization

- Optimization refers to the process of adjusting the parameters (weights and biases) of the neural network to minimize the cost function.
- This is typically done using optimization algorithms such as Gradient Descent and its variants.
- Optimization algorithms iteratively update the parameters in the direction that reduces the cost function.



Optimization

The process of adjusting the parameters (weights and biases) of the neural network to minimize the loss function, typically achieved using optimization algorithms.

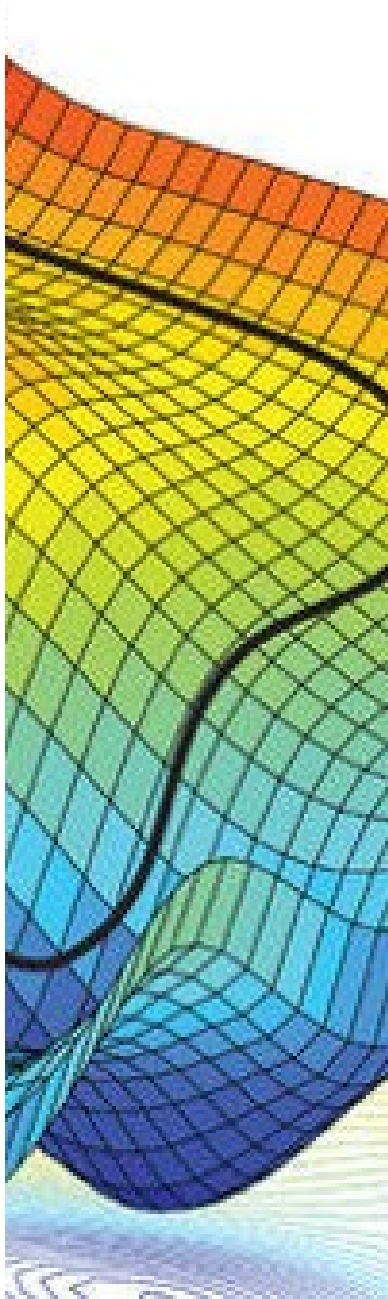
Some common optimization algorithms include:

- Gradient Descent
- Stochastic Gradient Descent (SGD
- Adam

Gradient Descent

Optimization algorithm

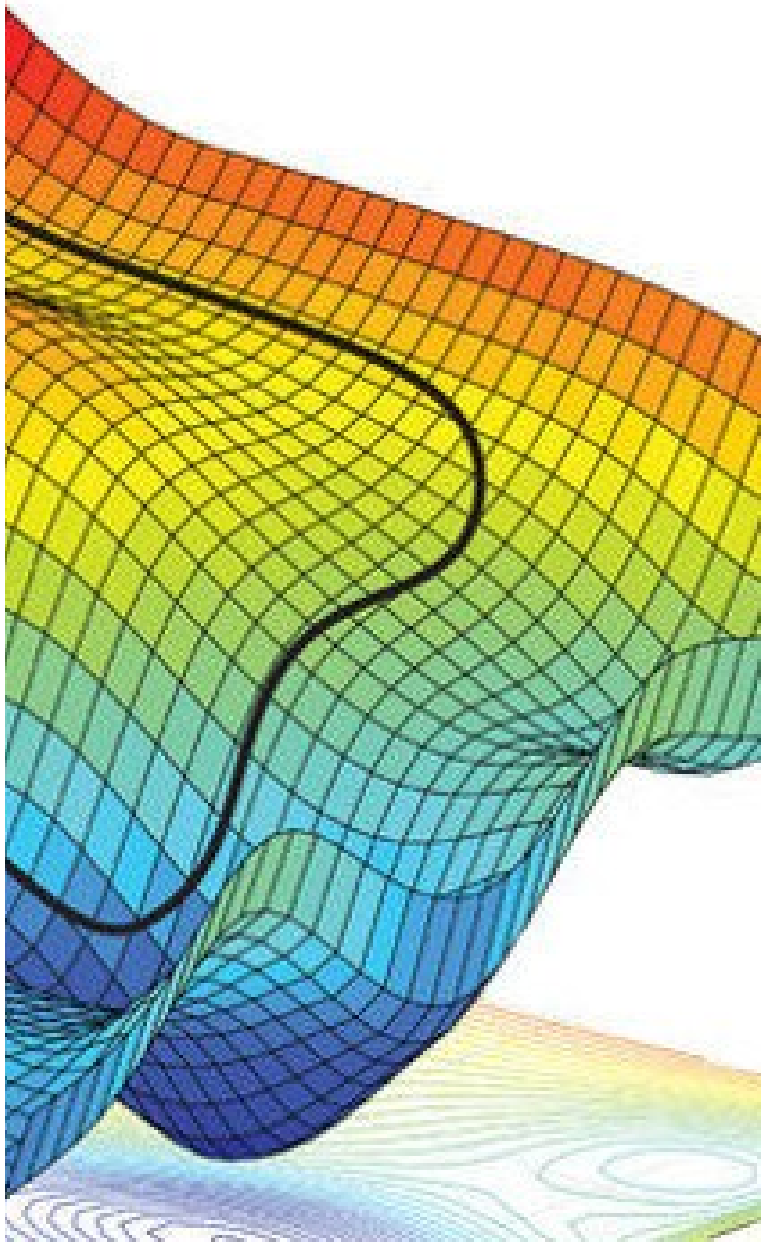
Updates the parameters of the neural network in the direction that minimizes the loss function.

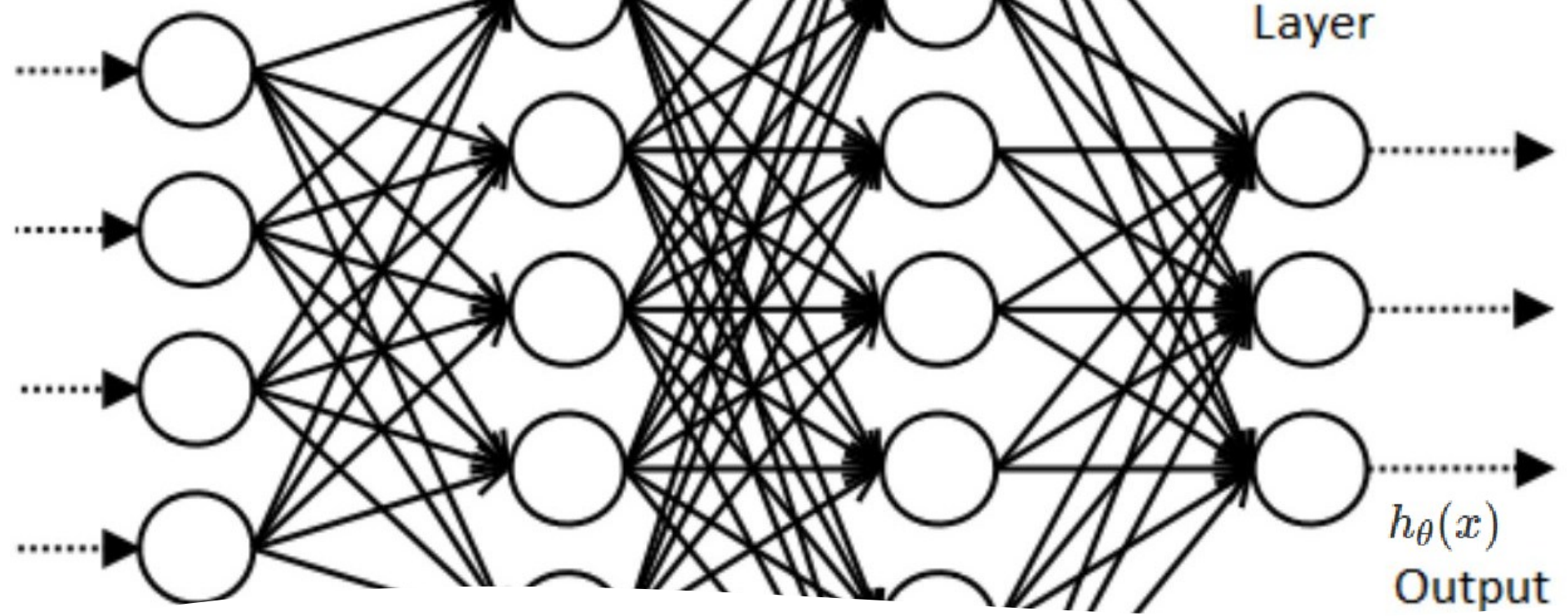


Stochastic Gradient Descent (SGD)

A variant of gradient descent

Updates the parameters using a randomly selected subset of the training data, improving training efficiency.





Layers

Neural networks consist of layers of interconnected neurons.

Each layer performs specific transformations on the input data, such as feature extraction or prediction.

Feedforward

meaning they flow in one direction only, from input to output

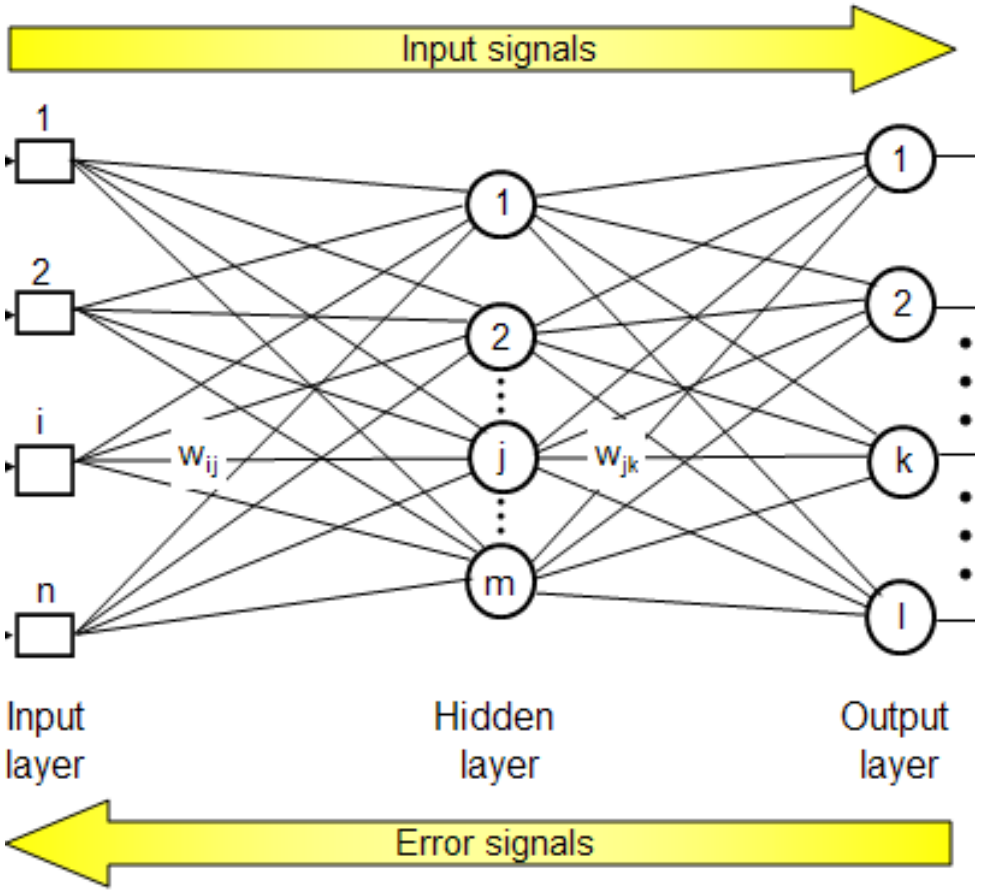
However, it is possible to train a model through backpropagation:

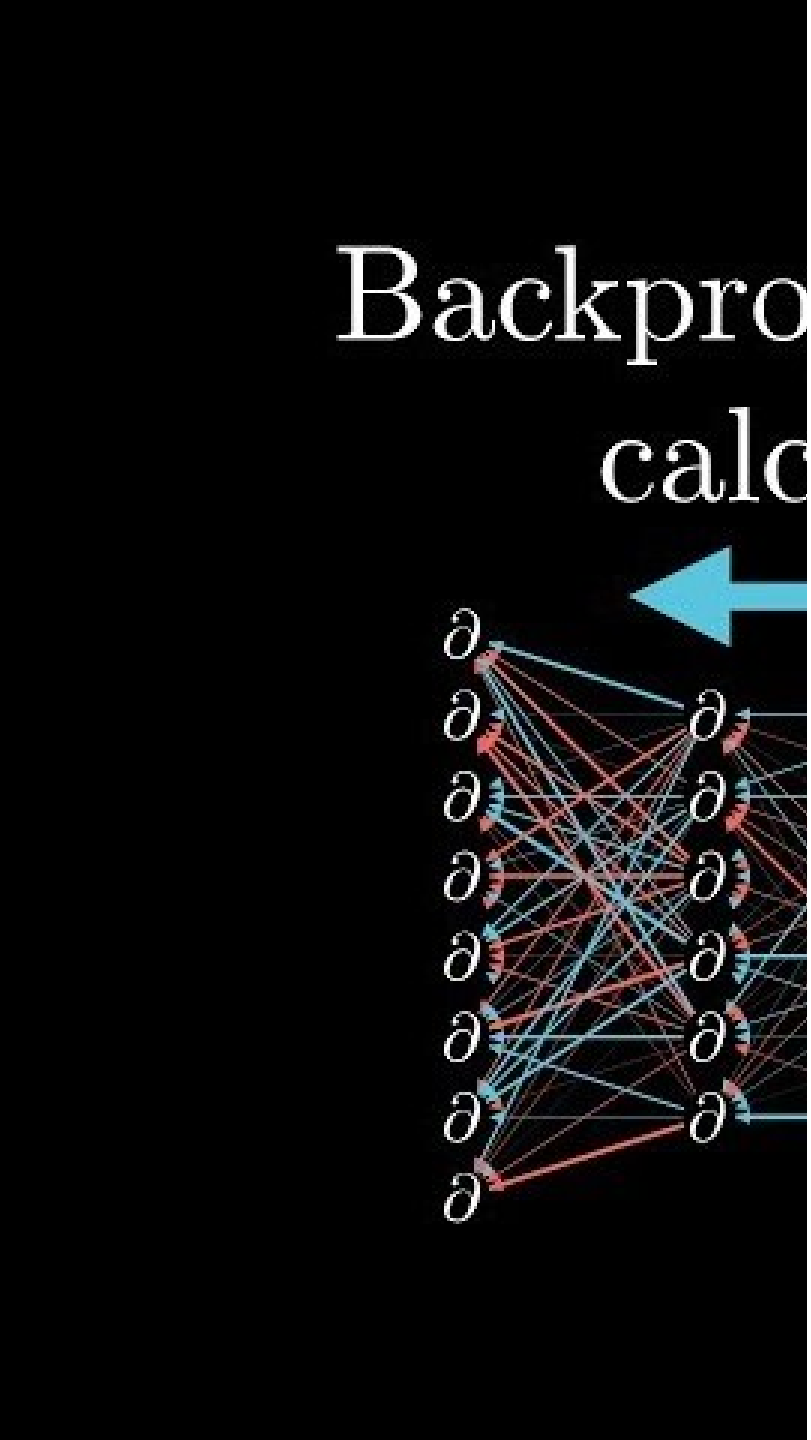
move in the opposite direction from output to input.

Backpropagation

allows to calculate and attribute the error associated with each neuron,

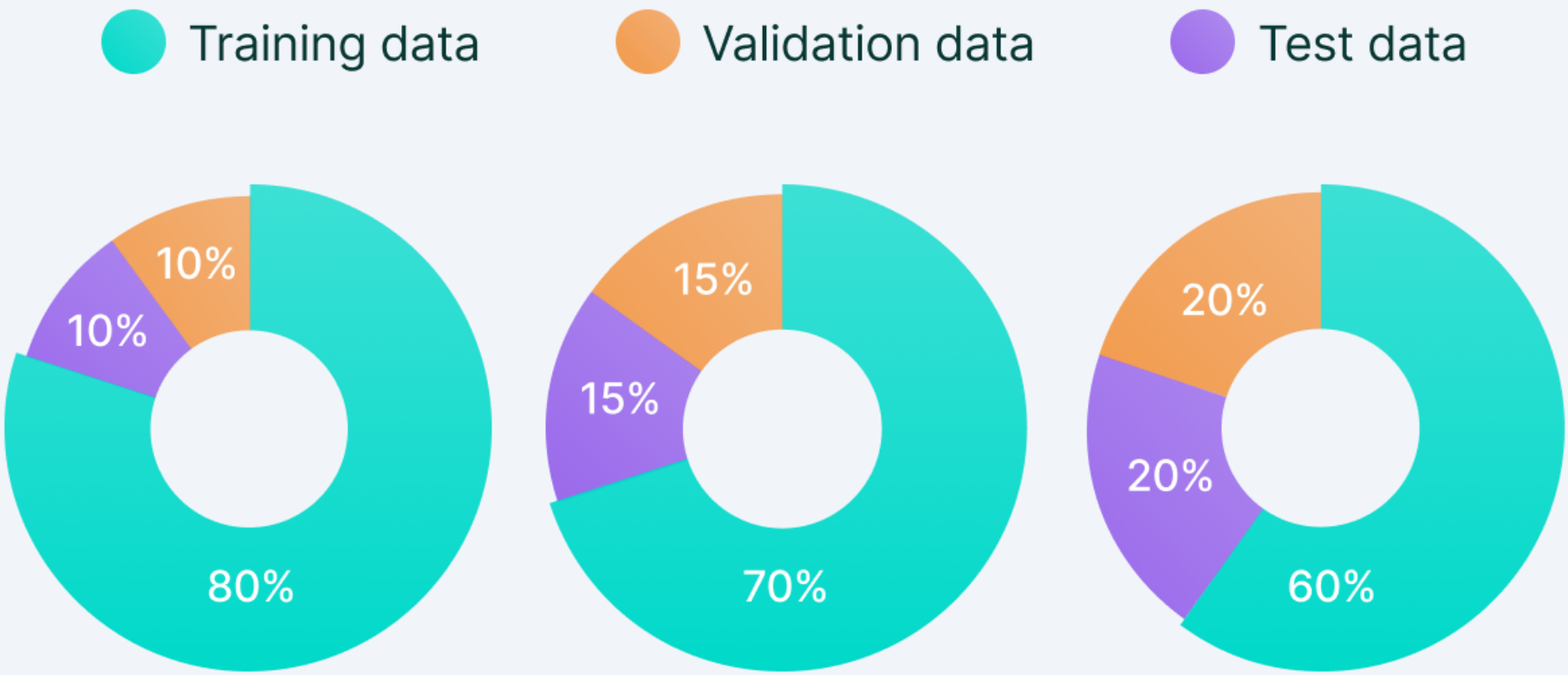
allowing to adjust and fit the parameters of the model(s) appropriately.





Backpropagation

- is an algorithm used to calculate the gradients of the loss function with respect to the weights and biases in the network.
- these gradients indicate how much the loss would change if the weights and biases were adjusted slightly.
- by propagating error signals backward through the network, backpropagation efficiently computes these gradients layer by layer.
- this process enables the network to update its parameters in a way that reduces the prediction error, leading to improved performance over time.



V7 Labs

Training, Validation, and Testing

Neural networks are

trained on a subset of the data (training set),

evaluated on a separate validation set to finetune hyperparameters,

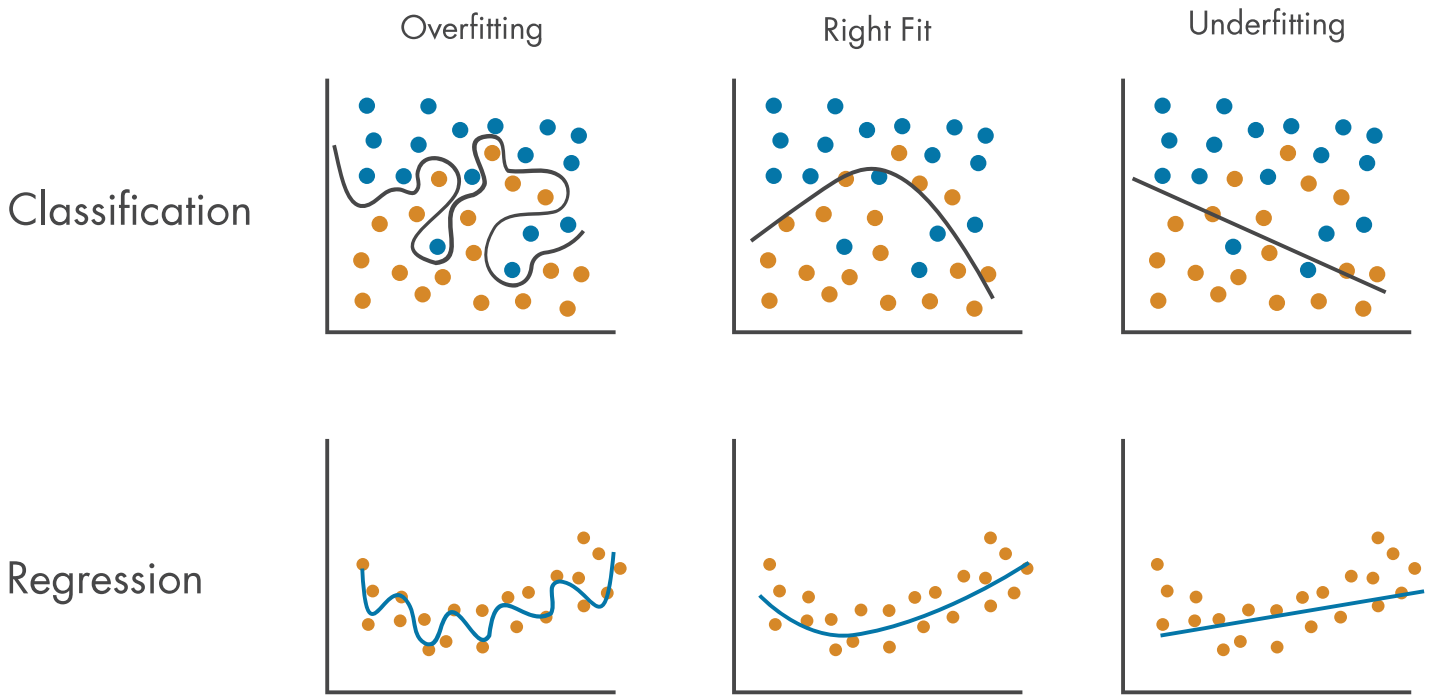
tested on a separate test set to assess generalization performance.

Carlos J. Costa, 2025

Overfitting and Regularization

Overfitting occurs when a model learns to memorize the training data instead of generalizing from it.

Regularization techniques, such as dropout and L1/L2 regularization, are used to prevent overfitting by penalizing overly complex models.



Carlos J. Costa, 2025

Hyperparameter Tuning

Hyperparameters are settings that control the architecture and optimization process of the neural network, such as the learning rate, number of layers, and batch size.

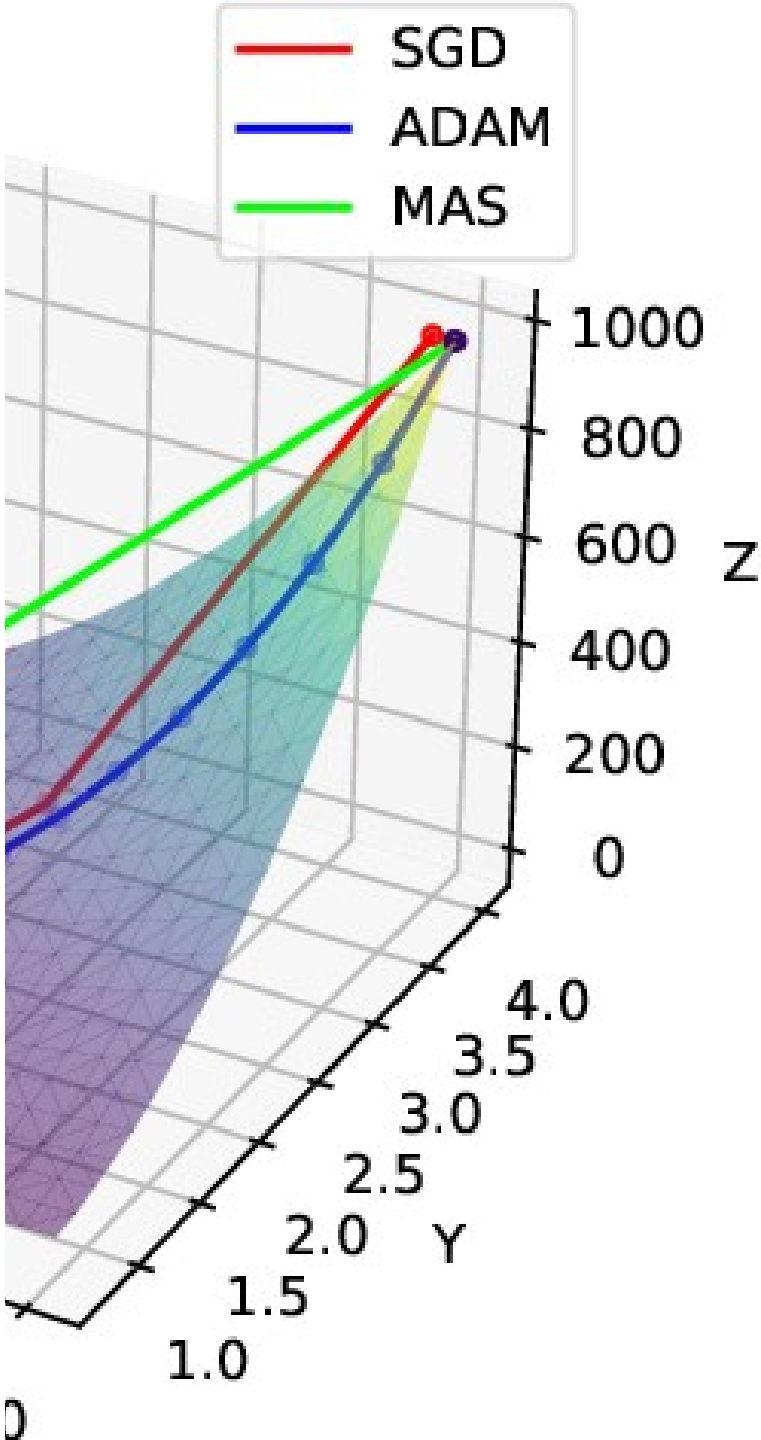
Tuning these hyperparameters can significantly impact the model's performance.

Adam Optimization:

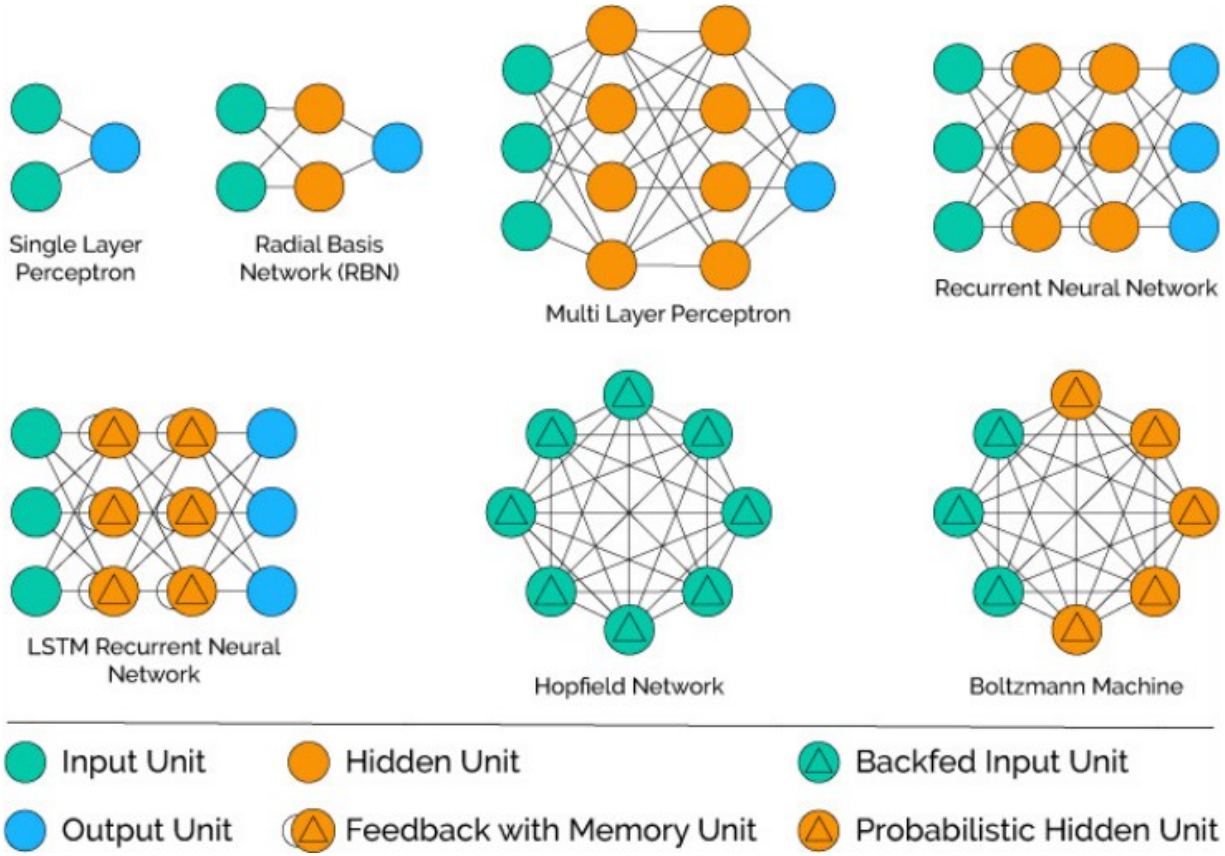
Adaptive learning rate optimization algorithm

Combines the advantages of both AdaGrad and RMSProp

Commonly used for training deep neural networks.

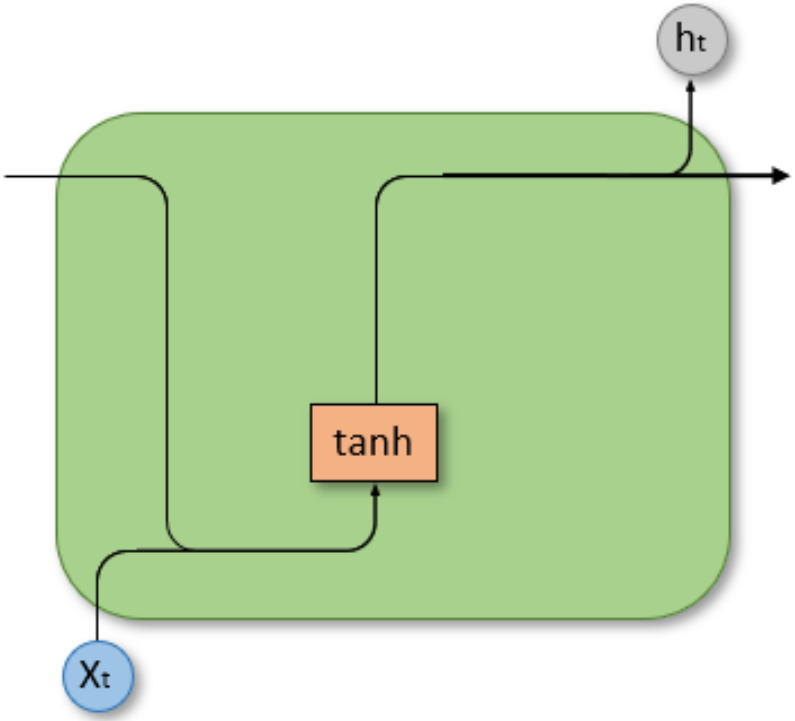


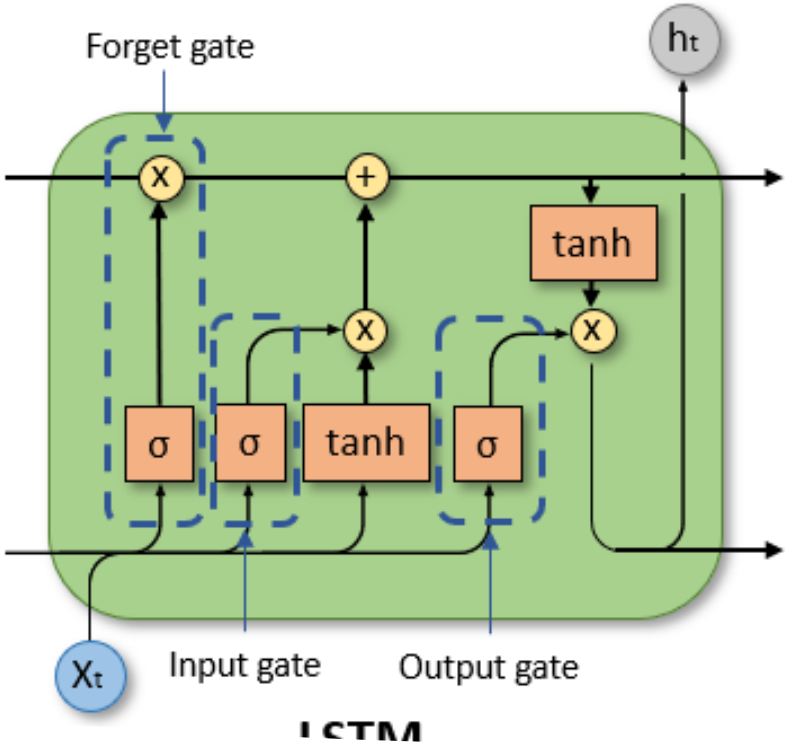
Types of Artificial Neural Networks



Types of Artificial Neural Networks

Type	Example references	Variants	Network structure	Applications
BP	Rumelhart[33]	RBF GRNN	Input layer Output layer Hidden layer	Data fitting Pattern recognition Classification
CNN	LeCun[34] Krizhevsky[35]	LeNet, AlexNet VggNet	Input layer Convolution layer Pooling layer Full connected layer	Image processing Speech signal Natural Language Processing
RNN	Mikolov[36] Sundermeyer[37]	LSTM	Input layer Hidden layer Output layer	Time series analysis Emotion analysis Natural Language Processing
GAN	Goodfellow[28]	DCGAN	Discrimination model Generation model	Image generation Video generation





Types of Artificial Neural Networks

Recurrent Neural Network (RNN)

- Through the feedback loop the output of one RNN cell is also used as an input by the same cell.
- Each cell has two inputs: the past and the present.
- Using information of the past results in a short-term memory.

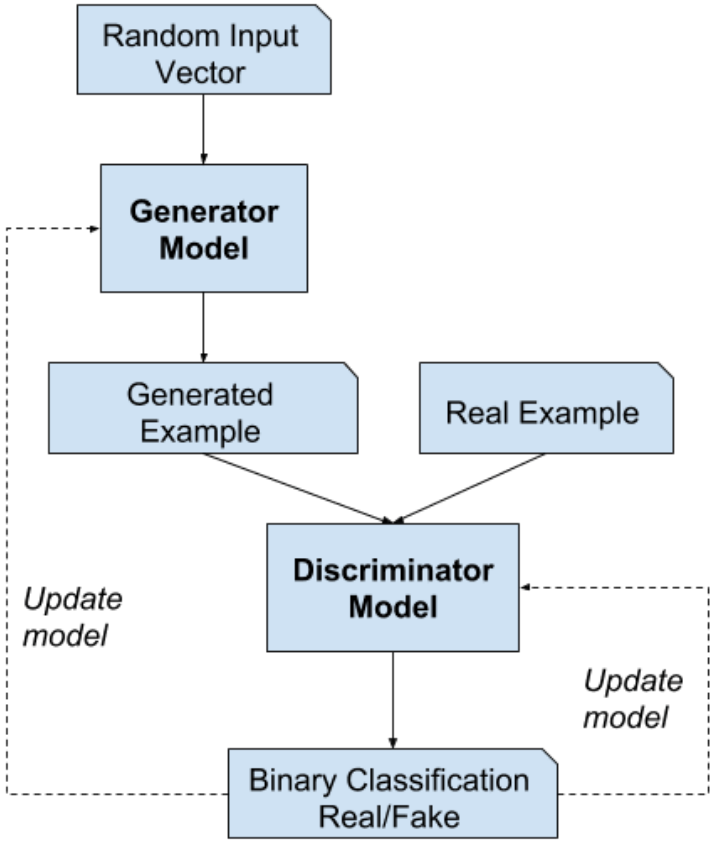
Long-Short-term-Memory Recurrent Neural Network (LSTM)

- LSTMs are a special type of RNNs which tackle the main problem of simple RNNs, the problem of vanishing gradients, i.e., the loss of information that lies further in the past.
- The key to LSTMs is the cell state, which is passed from the input to the output of a cell.
- The cell state allows information to flow along the entire chain with only minor linear actions through three gates.

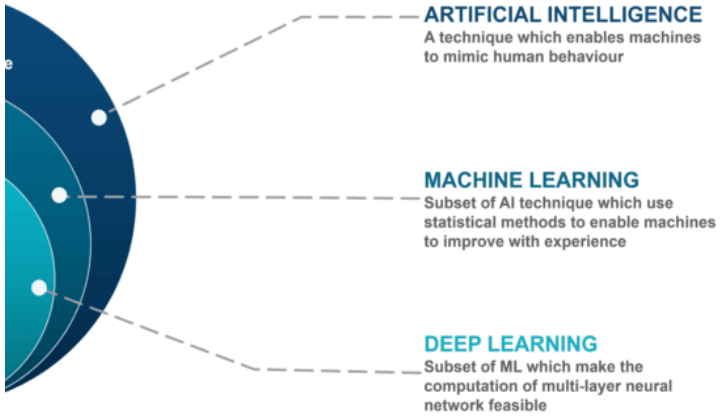
Carlos J. Costa, 2025

GAN

- Generative Adversarial Networks
- s a type of artificial intelligence model framework used in unsupervised machine learning.
- GANs is to train two neural networks in a competitive setting: generator and the discriminator.
- **Generator**-
  - generates new data instances
  - starts with random noise as input and tries to produce data samples that resemble the training data.
- **Discriminator**:
  - evaluates the generated samples and tries to distinguish between real data from the training set and the fake data produced by the generator.



Machine Learning Deep Learning

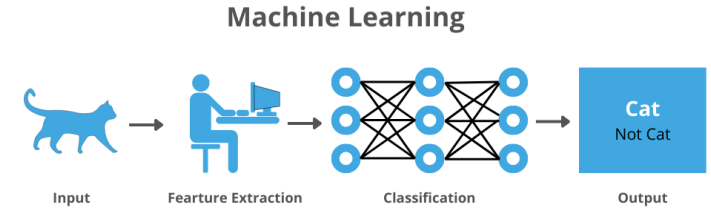


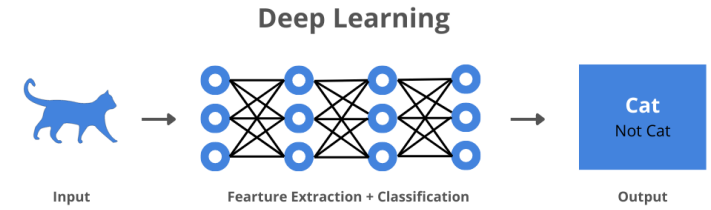
Machine Learning (ML):

- Involves algorithms that improve their performance on a specific task through experience or exposure to data, without being explicitly programmed for that task.
- ML algorithms learn patterns or representations from data to make predictions or decisions.
- includes various techniques such as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

Deep Learning:

- subset of ML
- focuses on neural networks with multiple layers (hence "deep") to automatically learn intricate patterns from data.
- ability to handle large volumes of data and automatically extract hierarchical features.
- Deep Learning architectures include convolutional neural networks (CNNs) for image processing, recurrent neural networks (RNNs) for sequential data, and transformers for natural language processing.
- require substantial computational resources for training due to the complexity of the network architecture and the large amount of data involved.





TensorFlow

- is a free and open-source software library for ML.
- particular focus on training and inference of deep neural networks
- symbolic math library based on dataflow and differentiable programming
- Google
- Apache License 2.0 since 2015
- Repository:
  - https://github.com/tensorflow/tensorflow
  - https://www.tensorflow.org/

TensorFlow

Keras

- Interface for the TensorFlow library.
- Open source
- Repository:
  - https://github.com/keras-team/keras
  - https://keras.io/
  - https://keras.io/guides/

```python
# Regression Example With Boston Dataset: Standardized and Wider
from pandas import read_csv
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
# load dataset
file="https://raw.githubusercontent.com/masterfloss/data/main/housing.csv"
dataframe = read_csv(file, delim_whitespace=True, header=None)
dataset = dataframe.values
# split into input (X) and output (Y) variables
X = dataset[:,0:13]
Y = dataset[:,13]
# define wider model
def wider_model():
        # create model
        model = Sequential()
        model.add(Dense(20, input_dim=13, kernel_initializer='normal', activation='relu'))
        model.add(Dense(1, kernel_initializer='normal'))
        # Compile model
        model.compile(loss='mean_squared_error', optimizer='adam')
        return model
# evaluate model with standardized dataset
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasRegressor(build_fn=wider_model, epochs=100, batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
kfold = KFold(n_splits=10)
results = cross_val_score(pipeline, X, Y, cv=kfold)
print("Wider: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
def wider_model():
        # create model
        model = Sequential()
        model.add(Dense(20, input_dim=13, kernel_initializer='normal', activation='relu'))
        model.add(Dense(6, kernel_initializer='normal', activation='relu'))
        model.add(Dense(1, kernel_initializer='normal'))
        # Compile model
        model.compile(loss='mean_squared_error', optimizer='adam')
        return model
# evaluate model with standardized dataset
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasRegressor(build_fn=wider_model, epochs=100, batch_size=5, verbose=0)))
```

Scikitlearn implements ANN

- Multi-layer Perceptron
- https://scikit-learn.org/stable/modules/classes.html#module-sklearn.neural_network
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

Carlos J. Costa, 2025

Summary

- Main Concepts
- Types of Neural networks
- Implementation

References

- Costa, C. J. (2024). Neural Networks: A Comprehensive Overview of Their History, Development, and Future in AI. *OAE – Organizational Architect and Engineer Journal*. https://doi.org/10.21428/b3658bca.13fccc0e
- Costa, C. J. (2025). Generative AI Models: A Comprehensive Review. *OAE – Organizational Architect and Engineer Journal*. https://doi.org/10.21428/b3658bca.d5d1872f
- Krizhevsky, A; Sutskever, I; Hinton, G.. (2017-05-24). "ImageNet classification with deep convolutional neural networks" . Communications of the ACM. 60 (6): 84–90. doi:10.1145/3065386.
- LeCun *et al. (1989)*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, 1, pp. 541–551.
- *McCulloch W, Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". Bulletin of Mathematical Biophysics. **5** (4): 115–133.*
- *Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., & Ranzato, M. A. (2014). Learning longer memory in recurrent neural networks. arXiv preprint arXiv:1412.7753.*
- *Rosenblatt F (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization in the Brain". Psychological Review. **65** (6): 386–408.*
- Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (1986). "Learning representations by back-propagating errors". *Nature*. 323 (6088): 533–536.
- Werbos P (1975). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Harvard University