Neural Networks II

Mark A. Austin

austin@umd.edu

ENCE 688P, Spring Semester 2026
University of Maryland

May 26, 2026

Overview

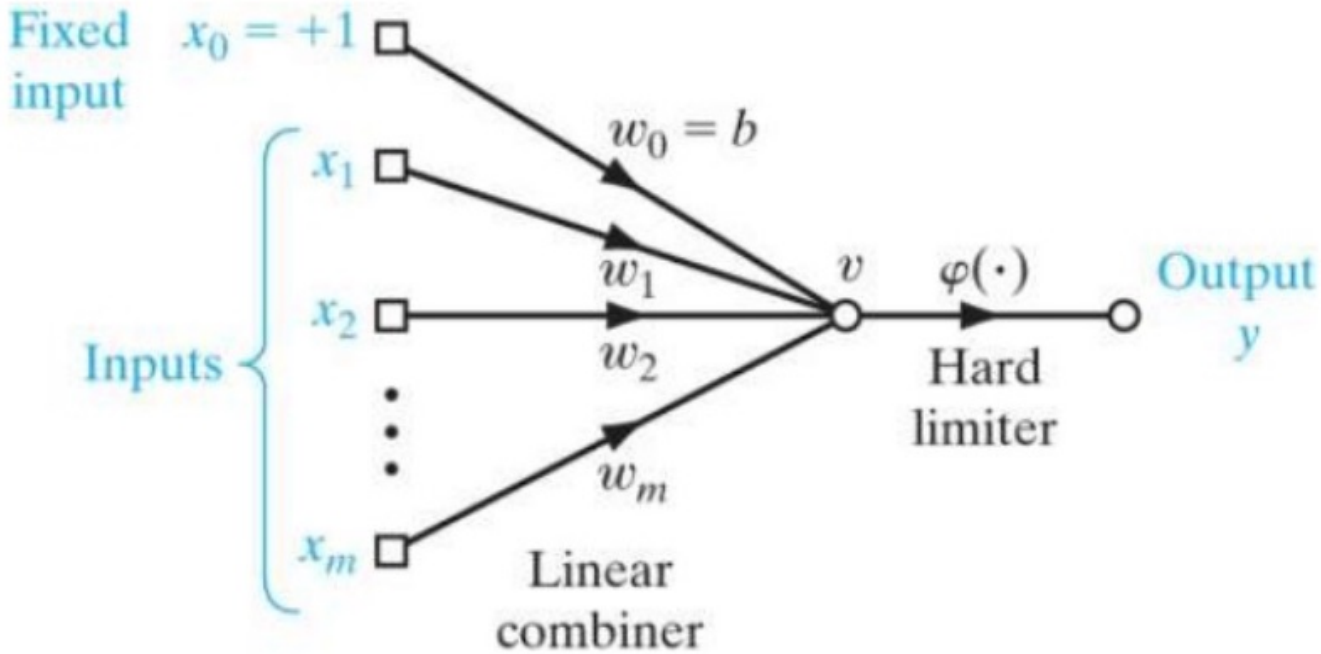
Quick Review

Mathematical Model of a Single Perceptron

Perceptron Model (Rosenblatt, 1958)

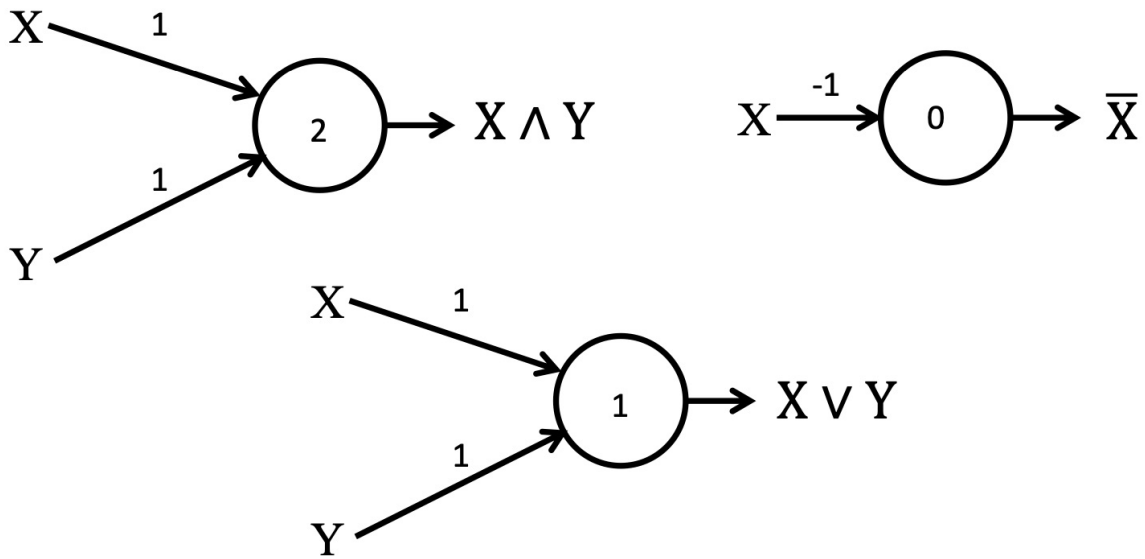
The simplest form of a neural network consists of a single neuron with adjustable synaptic weights and bias.

A nonlinear neuron consists of a linear combiner followed by a hard limiter.



Mathematical Model of a Single Perceptron

Perceptron Model for OR and AND Boolean Gates



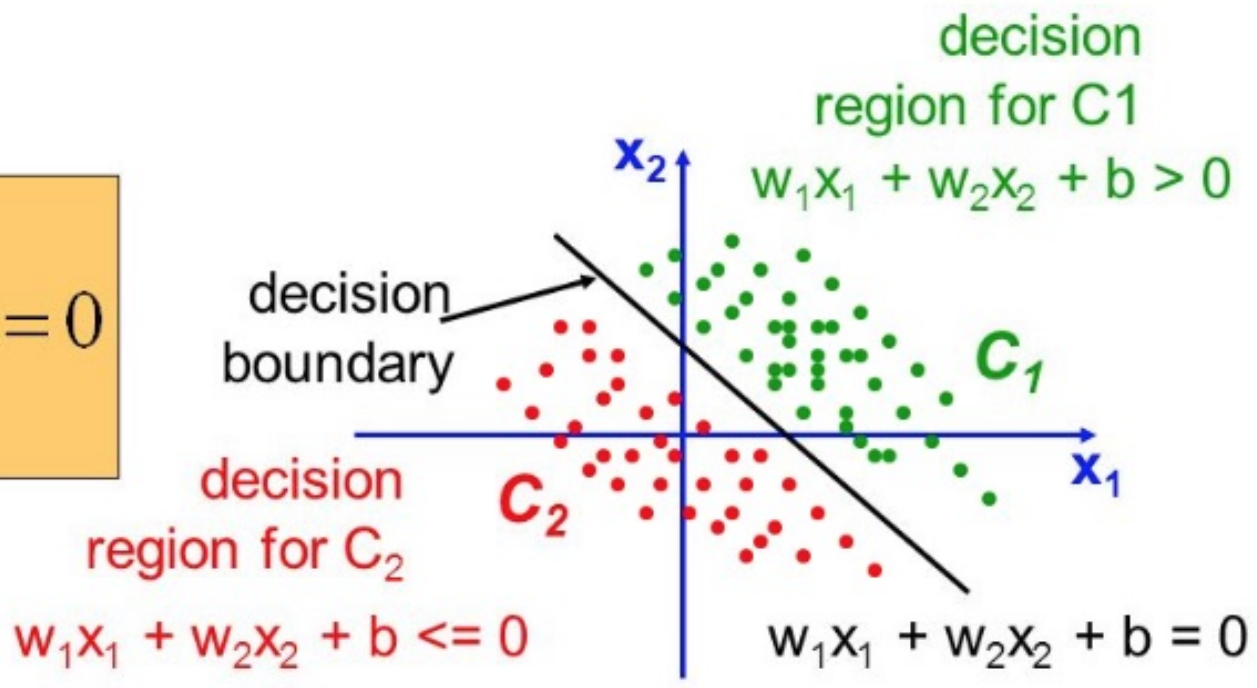
No solution for XOR Problem.

Individual elements are weak. Networked elements are required.

Perceptron Model as a Linear Classifier

Perceptron operating on real-valued vectors is a linear classifier:

$$\sum_{i=1}^{m} w_i x_i + b = 0$$

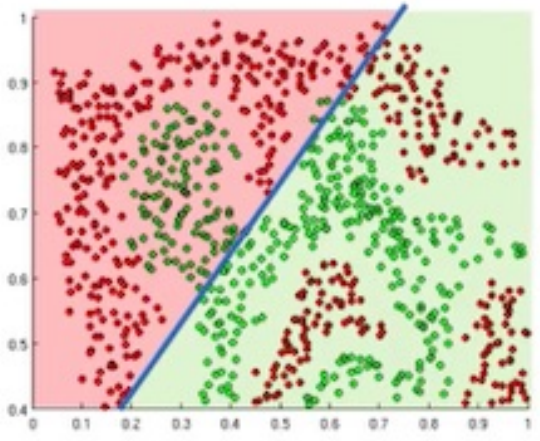


Addition of bias values expands modeling capability. No bias value $\rightarrow$ decision boundary constrained to pass through the origin.

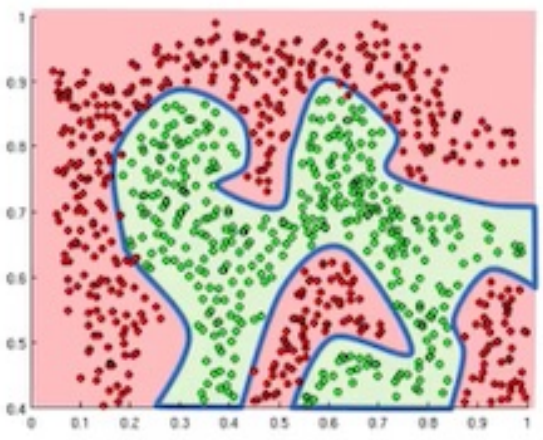
Perceptron Model as a Linear Classifier

Points to Note:

- Nonlinear decision boundaries require networks of nonlinear perceptrons.



Linear activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

- Back propogation involves calculating gradients for every single trainable parameter with respect to the loss function.

Perceptron Model as a Linear Classifier

General Classification Capabilities

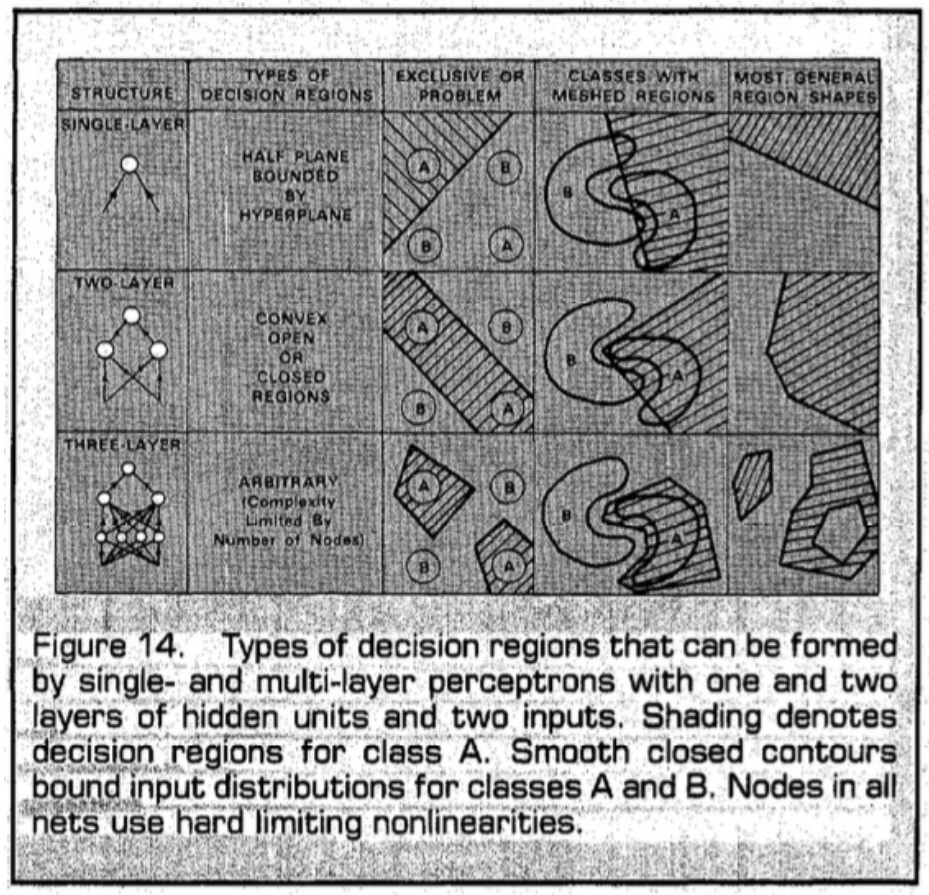


Figure 14. Types of decision regions that can be formed by single- and multi-layer perceptrons with one and two layers of hidden units and two inputs. Shading denotes decision regions for class A. Smooth closed contours bound input distributions for classes A and B. Nodes in all nets use hard limiting nonlinearities.

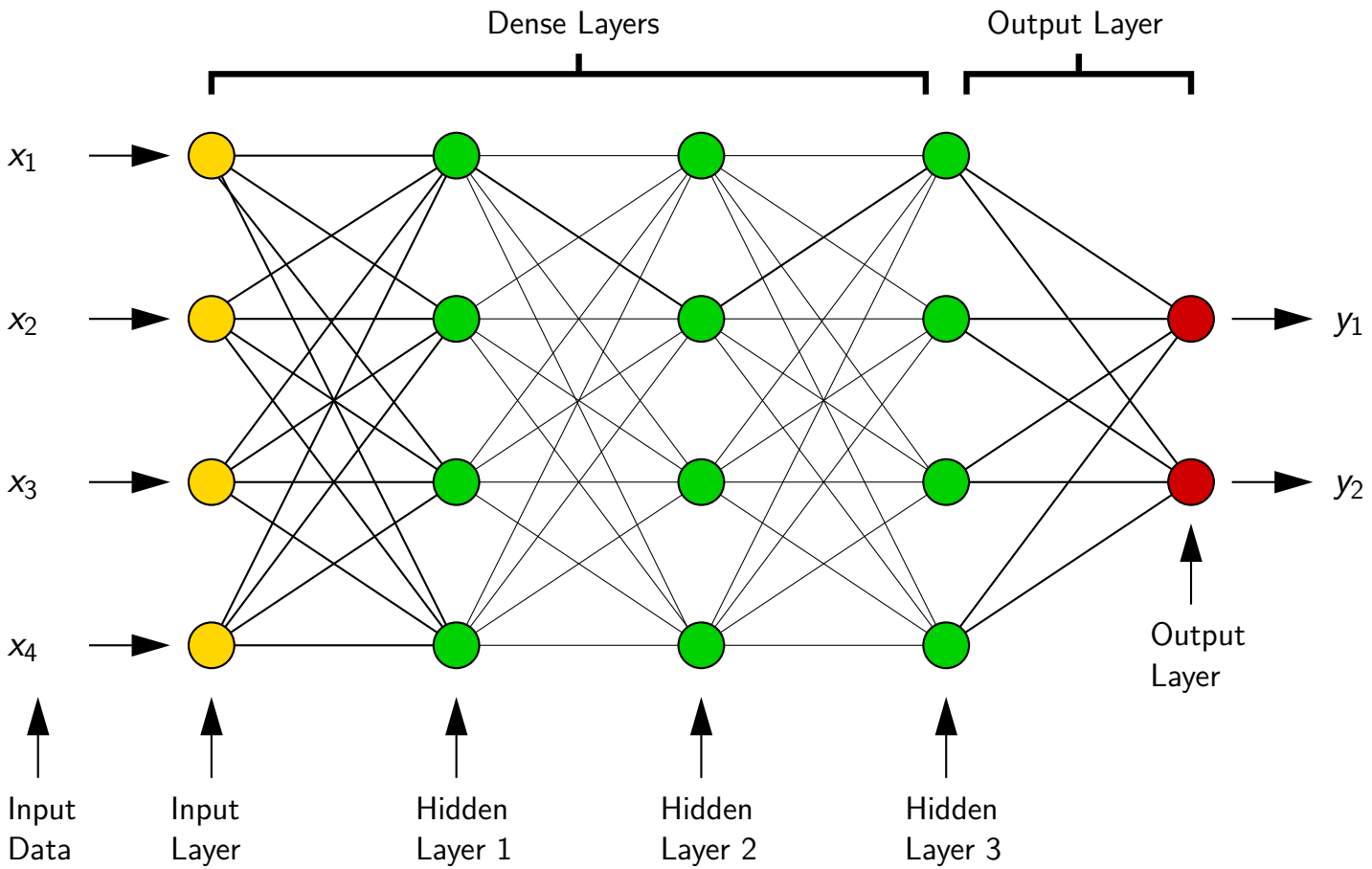
Source: Lippmann, IEEE Magazine, 1987.

Multilayer

Neural Networks

Multilayer Network Architecture

Network Architecture with 3 Hidden Layers



Multilayer Network Architecture

Assumptions

- We assume that inputs feed forward – no feedback to the inputs, either directly or indirectly.
- Assume the network architecture (see below) is capable of representing the needed function.

Network Architecture Design

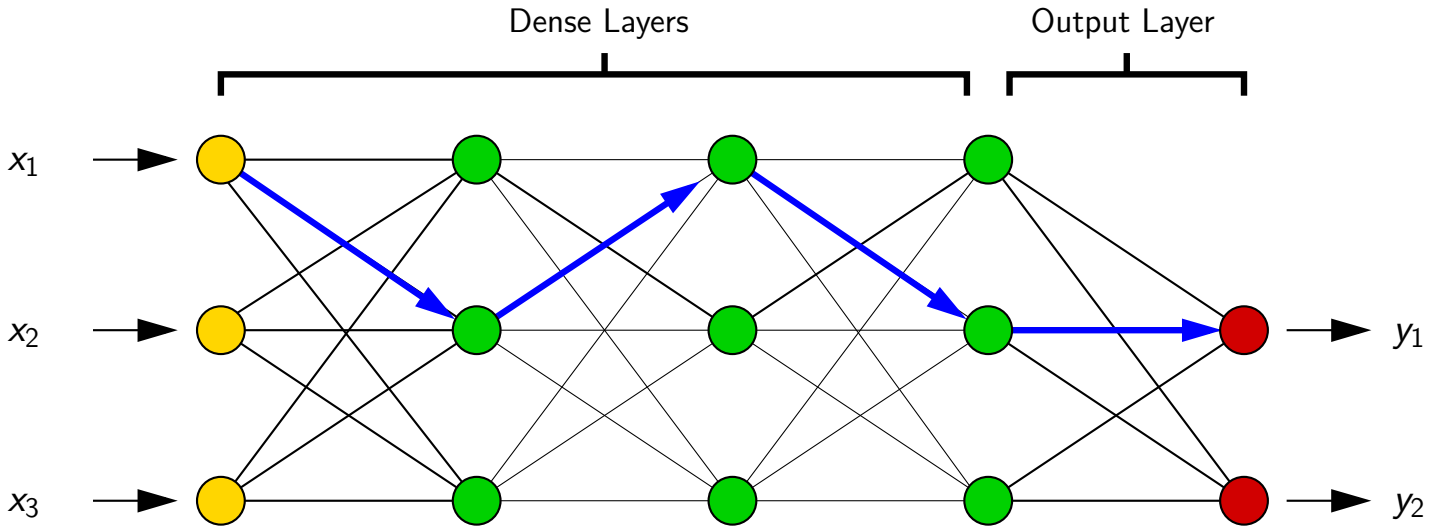
- How many inputs and outputs?
- How many layers?
- How many neurons per layer?
- Will individual neurons have biases?
- Layout of connectivity between neurons?
- Selection of training algorithms.

Multilayer Network Architecture (Deep Structures)

Network Depth

In any directed network of computational elements, with input (source) and output (sink) nodes, depth is the longest path from the source to the sink.

Example. Network depth =4.



Deep Network: depth > 2.

Multilayer Network Capabilities

Key Points:

- Multi-layer perceptron models are universal Boolean functions, universal classifiers, and universal function approximators, over any number of inputs and any number of outputs.

Computational Advantage of Deep Network Structures

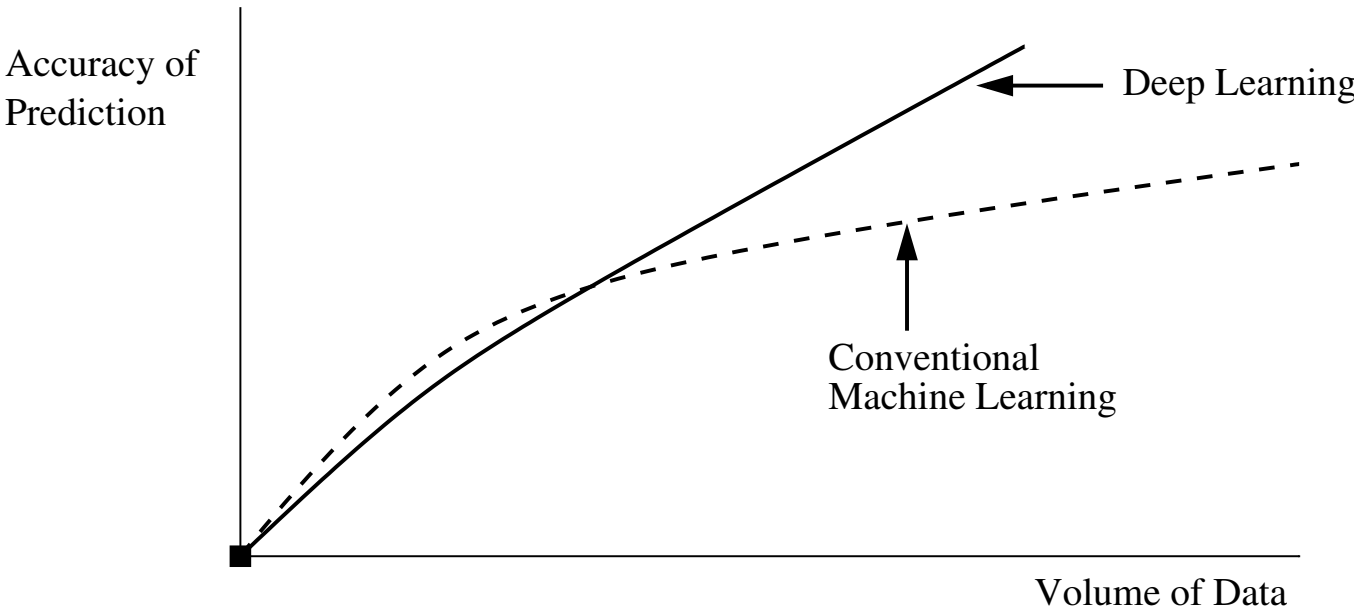
- Any truth table can be expressed by a MLP with only one hidden layer. Requires 2^{n-1} perceptrons – exponential growth.
- Can model same problem with $3(n\text{-}1)$ perceptrons, arranged into $2\log_2(n)$ layers. This is linear in N.
- Alternative view: for a given volume of data, deep learning provides higher accuracy of prediction than is possible with conventional machine learning.

Source: CMU course notes on Deep Learning.

Multilayer Network Capabilities

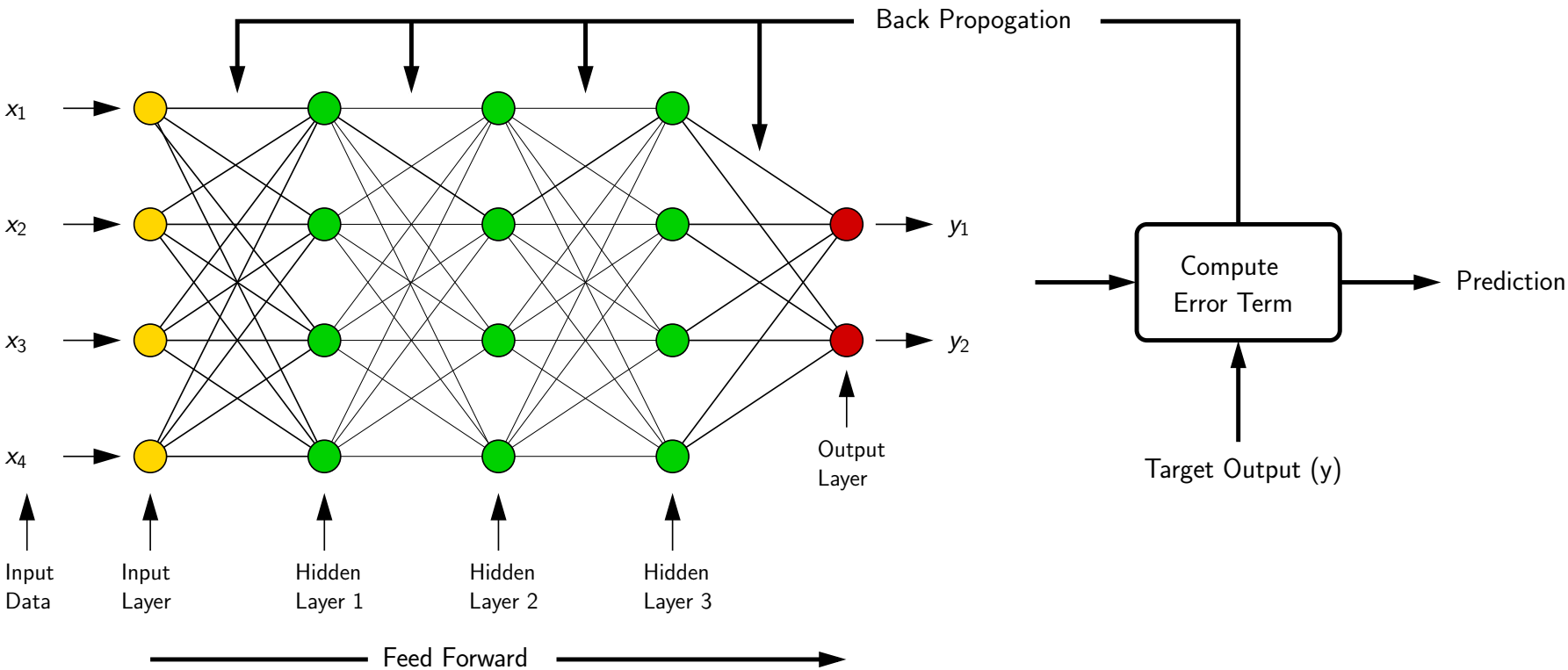
Conventional Machine Learning vs Deep Learning

Deep learning often provides higher accuracy of predictions than is possible with conventional machine learning.



Multilayer Network Training

Training Procedure: Learning the weights and biases to compute a target function (i.e., match the input-output relation of training instances drawn from the target function).



Common

Activation Functions

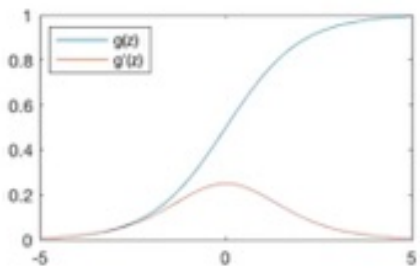
Activation Functions

Activation Function

Activations introduce nonlinearities into the neural network.

Common Activation Functions

Sigmoid Function

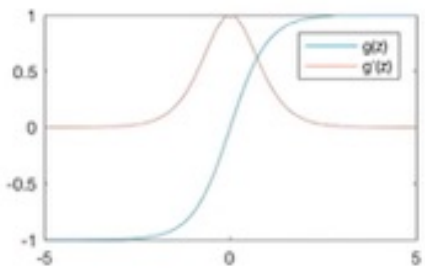


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

`tf.math.sigmoid(z)`

Hyperbolic Tangent

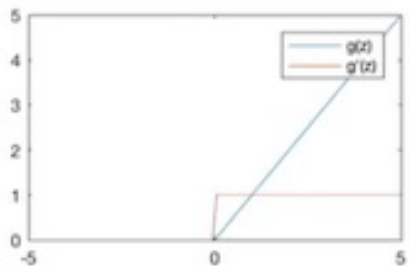


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

`tf.math.tanh(z)`

Rectified Linear Unit (ReLU)



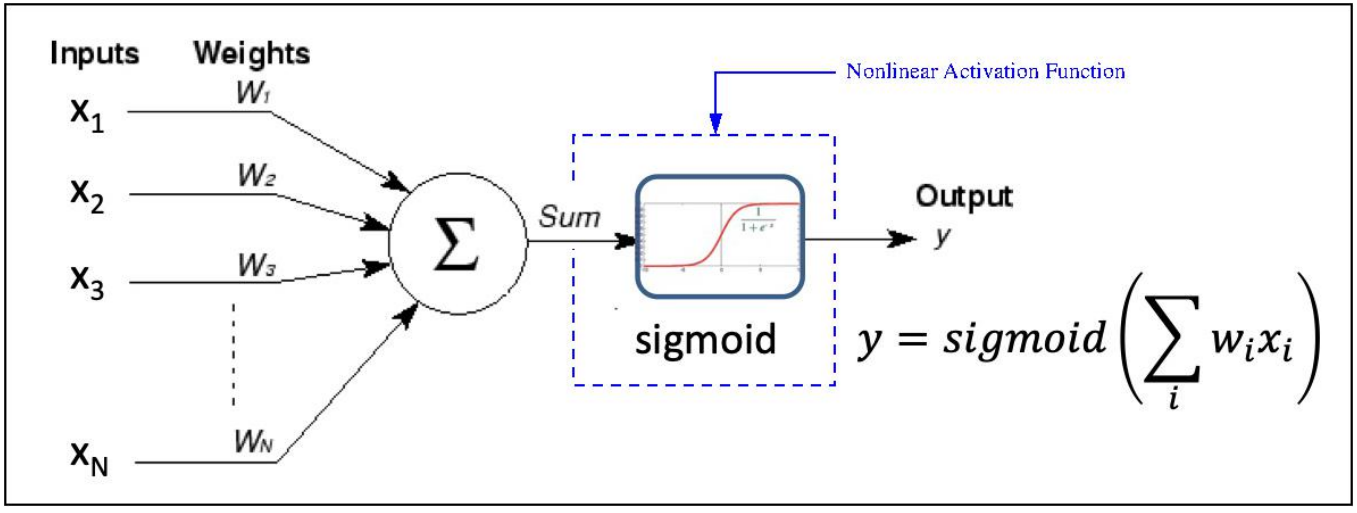
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

`tf.nn.relu(z)`

Activation Functions

Sigmoid Activation Function



To estimate network parameters, we need activation functions that are continuous, with non-zero derivatives.

Activation Functions

Pathway to Differentiable Activation

Replace step activation with sigmoid function:

$$y = f(z) = \begin{cases} 0, & z < 0, \\ 1, & z \geq 0. \end{cases} \quad \rightarrow y = \sigma(z) = \left[\frac{1}{1 + e^{-z}} \right]. \tag{1}$$

Derivative of sigmoid is easy:

$$\frac{dy}{dz} = \frac{d}{dz}\sigma(z) = \sigma(z)\left[1 - \sigma(z)\right]. \tag{2}$$

Can interpret output as P(y = 1|x).

Common

Loss Functions

Loss Functions

Loss Function

The loss of a neural network measures how well a prediction model does in terms of being able to predict the expected value (or outcome).

Regression Loss Functions

- Mean squared error loss.
- Mean squared logarithmic error loss.
- Mean absolute error loss.
- Root mean square error.

Loss Functions

Binary Classification Loss Functions

- Binary cross-entropy.
- Hinge loss.
- Squared hinge loss.

Multi-Class Classification Loss Functions

- Multi-class cross entropy loss.
- Kullback-Leibler (KL) divergence loss.

Mean Squared Error Loss

Mean Squared Loss Function

Meaned squared error (MSE) is the average of the square of the difference between actual and predicted values, summed over all data points.

- This loss function is the heart of least squares analysis.
- Mathematically:

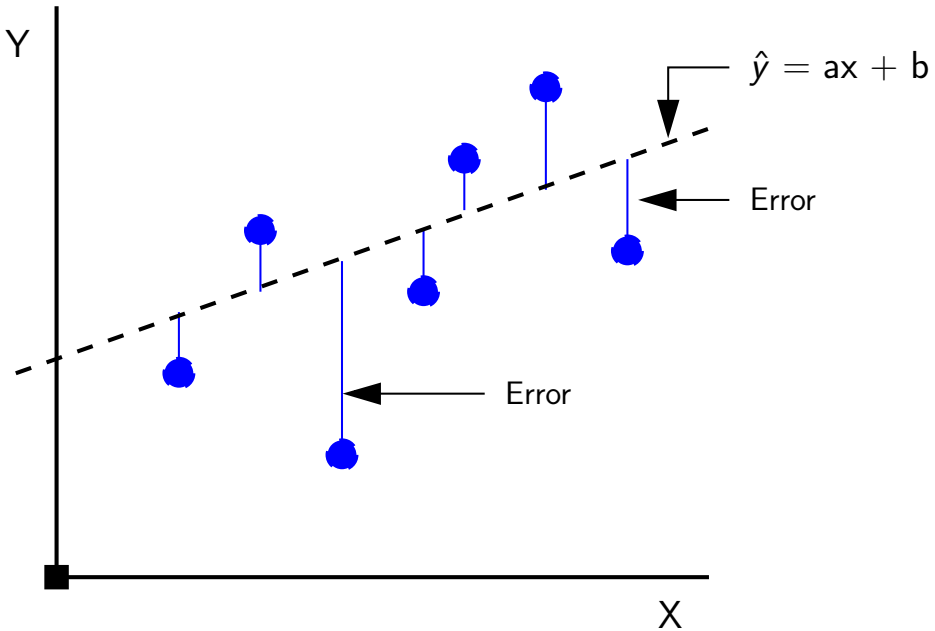
$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y - \hat{y})^2 \tag{3}$$

  Here, y is the actual value, $\hat{y}$ is the predicted value.
- Loss can be sensitive to outliers (i.e., unusually large errors).

Mean Squared Error Loss

Example. Least Squares Analysis.

Let $\hat{y}(x) = ax + b$.



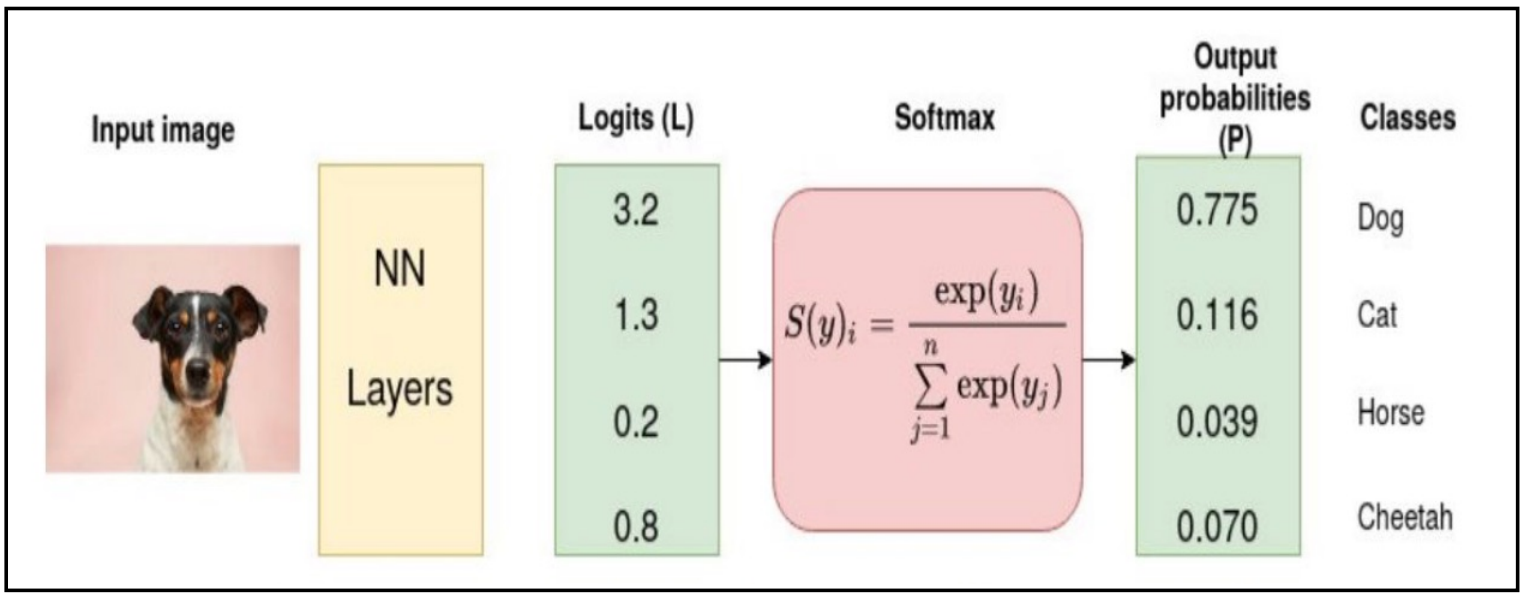
Least squares analysis involves finding coefficients a and b that minimize $L(y, \hat{y})$.

Cross-Entropy Loss

Cross Entropy Loss

A cross entropy loss function can be used with models that output a probability between 0 and 1.

Example. Animal Classification



Cross-Entropy Loss

Example. Animal Classification

- The desired output for class dog is $T = [1, 0, 0, 0]$.
- The NN model output is $S = [0.775, 0.116, 0.039, 0.070]$.

The categorical cross-entropy is computed as follows:

$$\begin{aligned} L_{ce} &= -\sum_{i=1}^{4} T_i \log_2(S_i) \\ &= -\left[1 \cdot \log_2(0.775) + 0 \cdot \log_2(0.116) \cdots\right] \\ &= -\log_2(0.775) \\ &= 0.3677 \end{aligned} \tag{4}$$

Training Procedures

Summary of Learning Procedure:

- Convert the neural network training problem into an optimization problem.
- Define a loss function and then optimize the parameter values to minumize its value.
- The loss function is the beginning of back propogation.

Training Procedures

Weights:

Training Procedures

Epochs:

Training Procedures

Back Propagation:

Optimizers

Adam Optimizer

Adam Optimizer

Adam optimizer ...

Gradient Descent Algorithms

Gradient Descent Algorithms

Gradient Descent

Gradient Descent

Gradient descent ...

Classes of Problems:

- ...
- ...
- ...
- ...

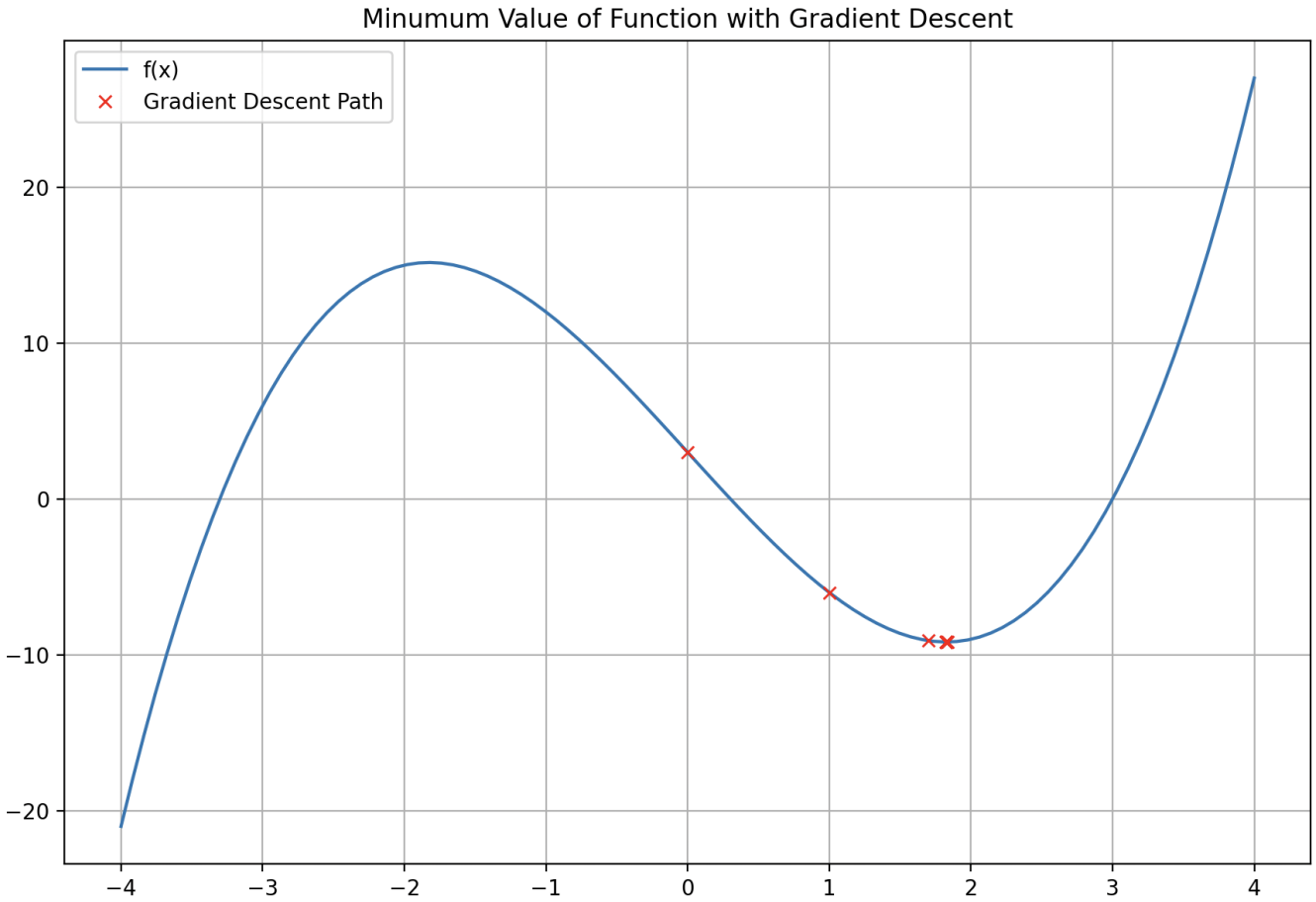
Gradient Descent

Applications:

- ...
- ...
- ...
- ...

Gradient Descent

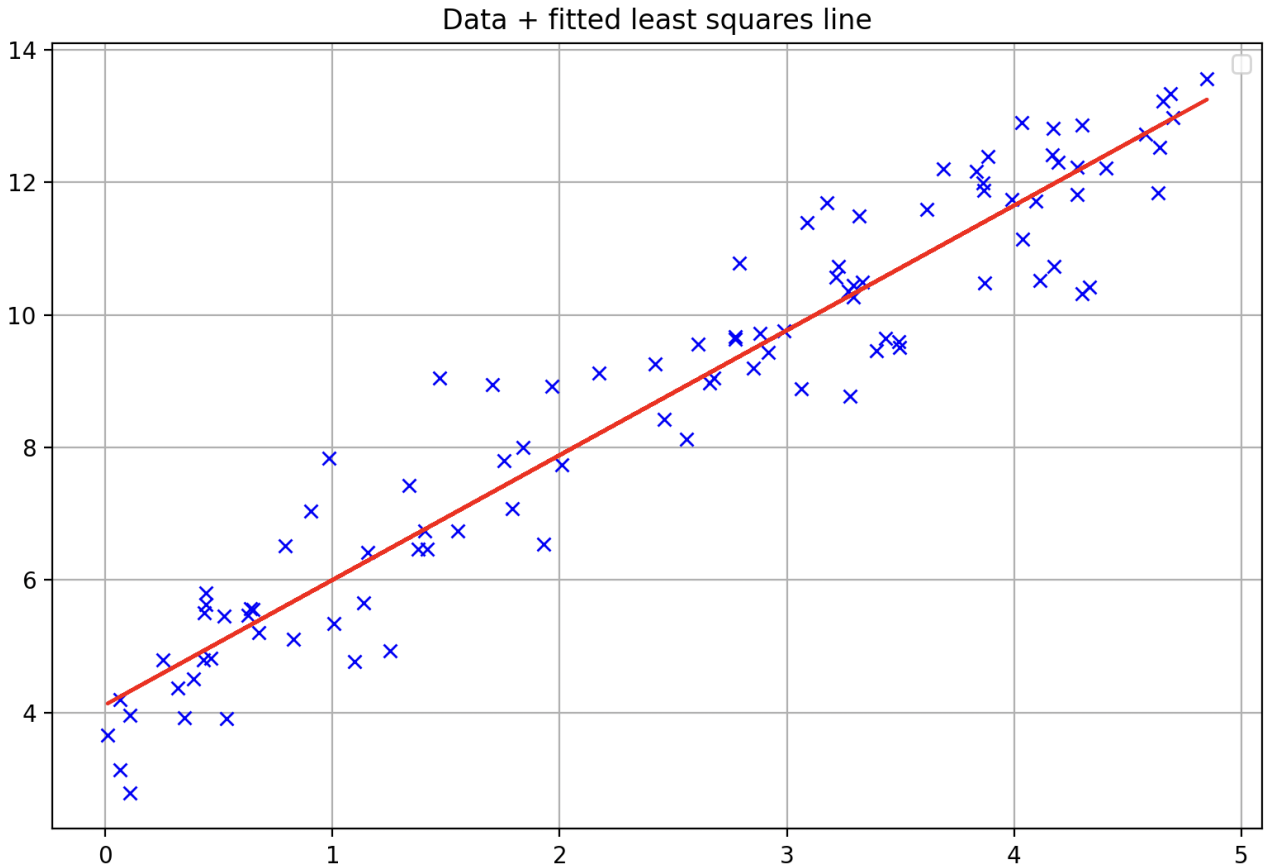
Example 1: Minimize Function



Source code: ...

Gradient Descent

Example 2: Least Squares Analysis



Source code: ...

Gradient Descent

Example 3: ...

Gradient Descent

Example 4: ...

Metrics of Evaluation

Metrics of Evaluation

Confusion Matrix

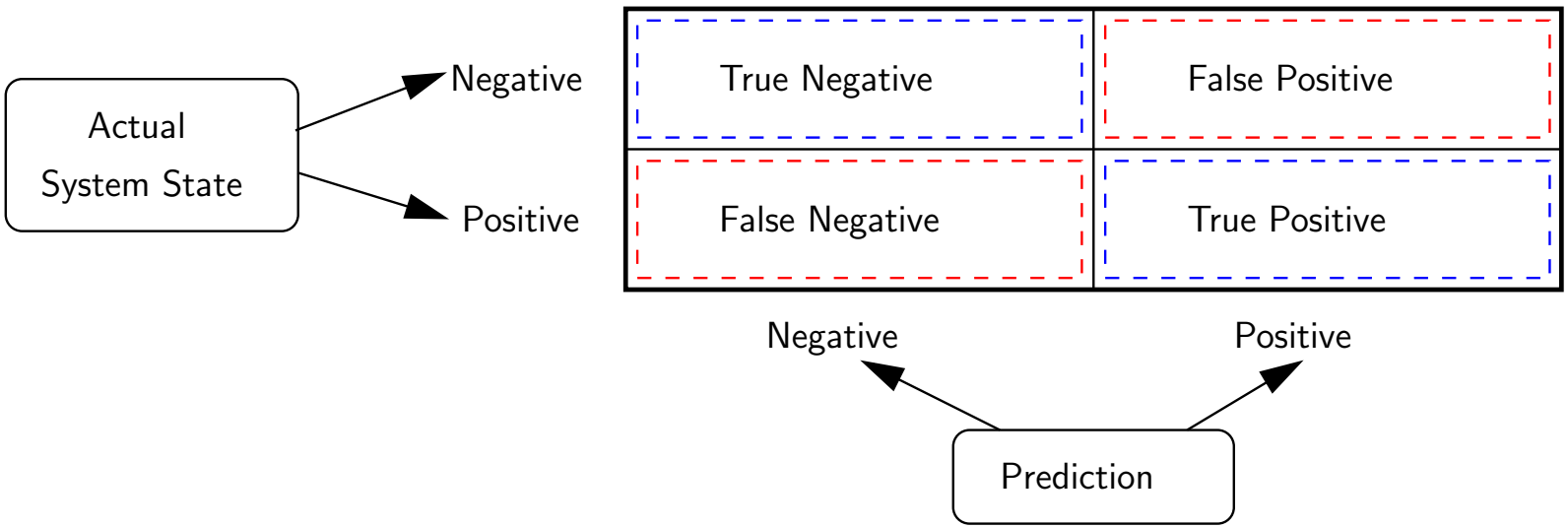
A simple metric to understand performance of a model in terms of predictions and their relationship to the actual state of a system.

Four Cases to Consider:

- True negative: The system state is negative; the model predicts negative.
- False positive: The system state is negative, but the model predicts positive.
- False negative: The system state is positive, but the model prediction is negative.
- True positive: The system state is positive and the model prediction is positive.

Metrics of Evaluation

Training Objective: We want:



- True negative and true positive numbers to be as high as possible,
- False positive and false negative to be as low possible.

Metrics of Evaluation

Accuracy:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{5}$$

Precision:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{6}$$

Recall:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{7}$$

F1 Score:

$$\text{F1} = 2\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

Metrics of Evaluation

Simple Example

Insert venn diagram ...

Assessment:

Metrics of Evaluation

Confusion Matrix

Insert description and simple example ...

Simple Matrix

Insert simple example ...

References

- Lippmann R.P., An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, April 1987.
- Bhiksha R., Introduction to Neural Networks, Lisbon Machine Learning School, June, 2018.
- Sun J., Fundamental Belief: Universal Approximation Theorems, Computer Science and Engineering, University of Minnesota, Twin Cities, 2020.