MACHINE LEARNING IN PHYSICS

FOUNDATIONS 1

Harrison B. Prosper

PHY6937 / PHY4936 Fall 2025

Goals of this Course

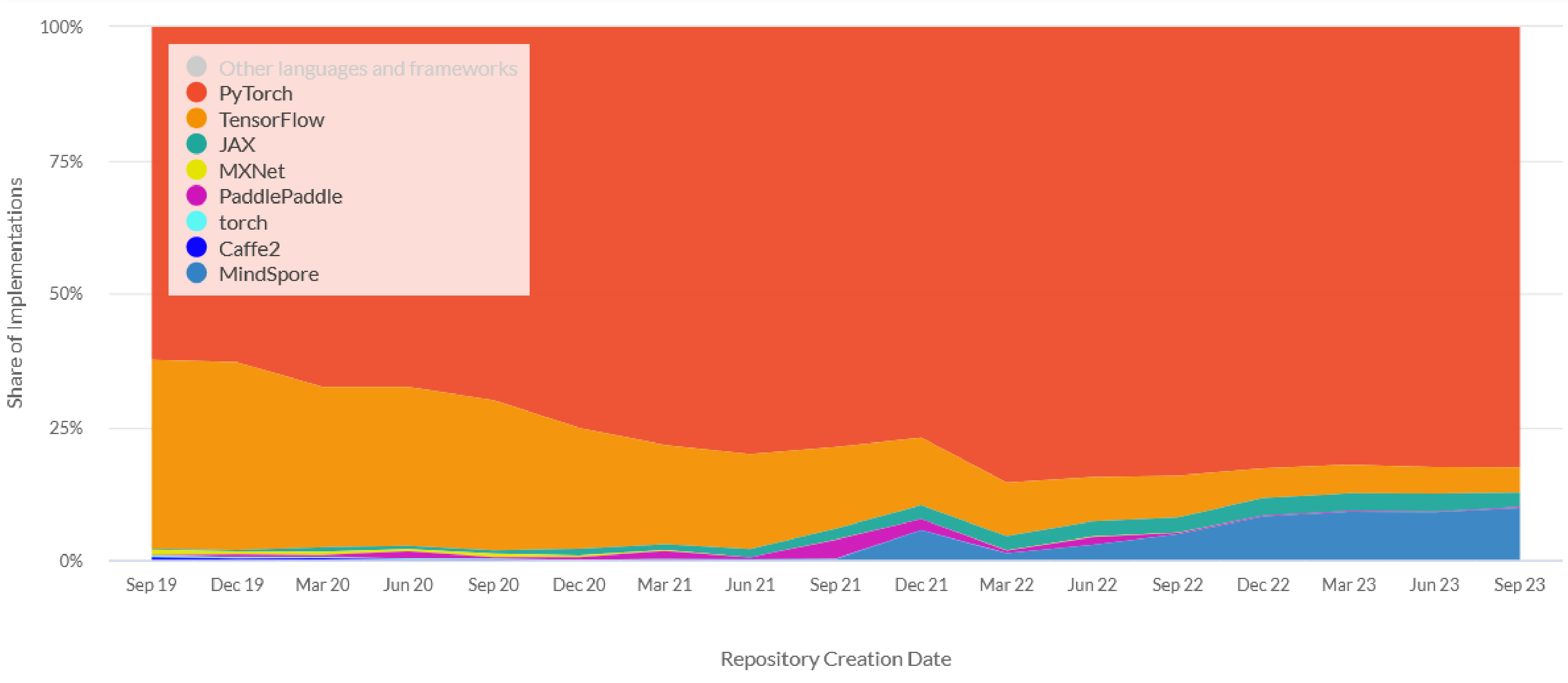
1. Gain a good understanding of the mathematical basis of machine learning (ML).

2. Gain experience building ML models using **PyTorch** to solve data science problems in physics.

3. Gain experience with different ML models.

4. Gain an appreciation of the power of ML models as well as their (current) limitations.

Why PyTorch?

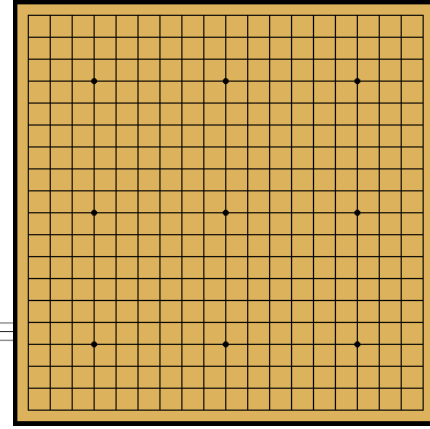


https://viso.ai/deep-learning/pytorch-vs-tensorflow/

What is Artificial Intelligence?

Artificial Intelligence
Algorithms that cause machines to exhibit *human*- or *superhuman*-level intelligence.

ARTICLE



doi:10.1038/nature24270

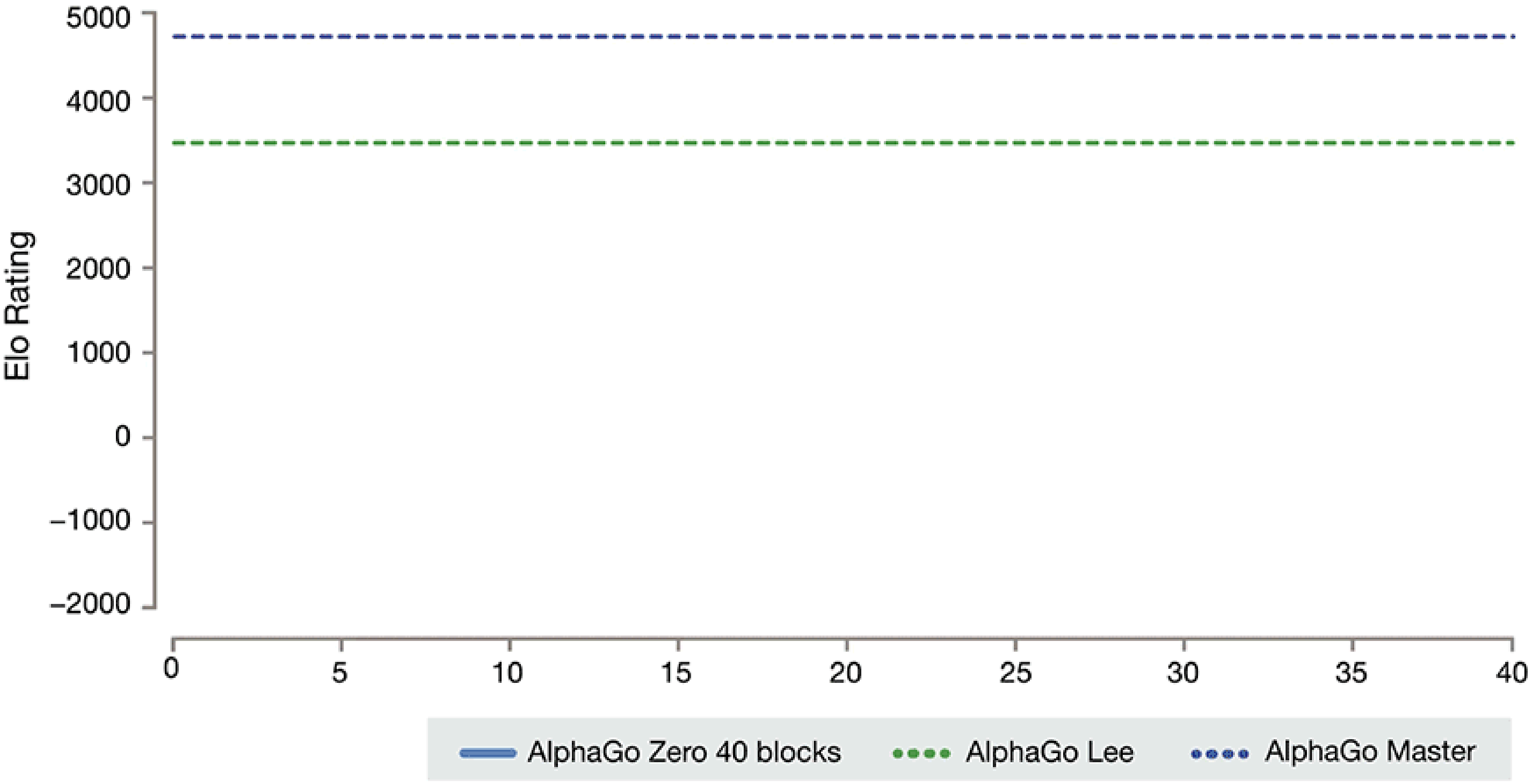
Mastering the game of Go without human knowledge

David Silver[1]*, Julian Schrittwieser[1]*, Karen Simonyan[1]*, Ioannis Antonoglou[1], Aja Huang[1], Arthur Guez[1], Thomas Hubert[1], Lucas Baker[1], Matthew Lai[1], Adrian Bolton[1], Yutian Chen[1], Timothy Lillicrap[1], Fan Hui[1], Laurent Sifre[1], George van den Driessche[1], Thore Graepel[1] & Demis Hassabis[1]

A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of the tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting *tabula rasa*, our new program AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

356 | NATURE | VOL 550 | 19 OCTOBER 2017

https://deepmind.com/blog/alphago-zero-learning-scratch/



Symbolic Mathematics

In December 2019, Guillaume Lample and François Charton* (Meta fka Facebook AI Research, Paris) claimed: "*We achieve results that outperform commercial Computer Algebra Systems such as Matlab or Mathematica*."

Lample

Charton

* G. Lample and F. Charton, Deep Learning for Symbolic Mathematics, arXiv: 1912.01412v1

MACHINE
LEARNING
Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
21 September 2022

REVISED
8 December 2022

ACCEPTED FOR PUBLICATION
12 January 2023

PUBLISHED
27 January 2023

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

PAPER

SYMBA: symbolic computation of squared amplitudes in high energy physics with machine learning

Abdulhakim Alnuqaydan[1,2,*], **Sergei Gleyzer**[3] and **Harrison Prosper**[4]

[1] Department of Physics and Astronomy, University of Kentucky, Lexington, KY, United States of America
[2] Department of Physics, College of Science, Qassim University, Qassim, Saudi Arabia
[3] Department of Physics and Astronomy, The University of Alabama, Tuscaloosa, AL, United States of America
[4] Department of Physics, Florida State University, Talllahassee, FL, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: aal700@uky.edu

Keywords: physics, high energy physics, machine learning

Abstract

The cross section is one of the most important physical quantities in high-energy physics and the most time consuming to compute. While machine learning has proven to be highly successful in numerical calculations in high-energy physics, analytical calculations using machine learning are still in their infancy. In this work, we use a sequence-to-sequence model, specifically, a transformer, to compute a key element of the cross section calculation, namely, the squared amplitude of an interaction. We show that a transformer model is able to predict correctly 97.6% and 99% of squared amplitudes of quantum chromodynamics and quantum electrodynamics processes, respectively, at a speed that is up to orders of magnitude faster than current symbolic computation frameworks. We discuss the performance of the current model, its limitations and possible future directions for this work.

ChatGPT

Developer(s)	OpenAI
Initial release	November 30, 2022 (2 years ago)[1]
Stable release	August 7, 2025 (16 days ago)[2]
Engine	GPT-5

What is Machine Learning?

Artificial Intelligence
Algorithms that cause machines to exhibit human- or *super-human* level intelligence.

Machine Learning
Algorithms for modeling data.

What is Machine Learning?

Supervised Learning

Data: (x, y)

y are labels

Task: $x \rightarrow y$

Use cases:

- Classification, regression, translation, etc.

Unsupervised Learning

Data: x

no labels

Task: find structure in, and/or model, data

Use cases:

- Clustering, data compression, solving differential equations, etc.

What is Machine Learning?

Generative Learning

Data: x

may or may not be associated with labels

Task: $x \rightarrow p(x) \rightarrow x$

Use cases:

- fast simulators, image/text generation, chatbots, etc.

Reinforcement Learning

Data: (x, a, r)

x state of the environment

a action taken on environment

r reward arising from action

Task: find optimal $x \rightarrow a$

Use cases:

- Game playing, robotics, accelerator controls, etc.

What is Deep Learning?

Artificial Intelligence
Algorithms that cause machines to exhibit human- or *super-human*-level intelligence.

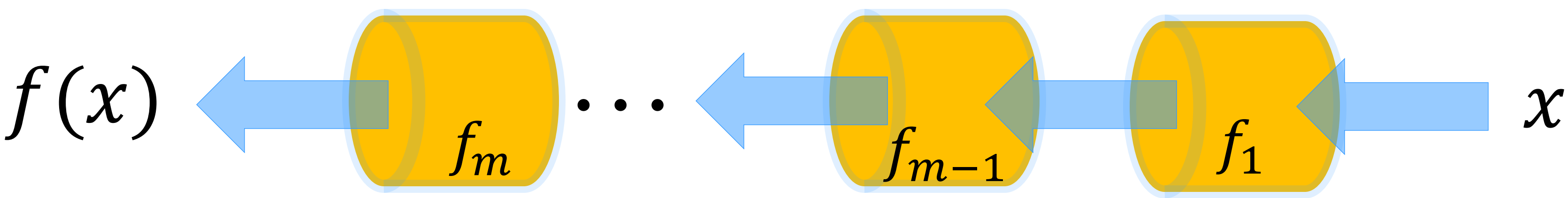
Machine Learning
Algorithms for modeling data

Deep Learning
ML using (large) neural networks

What is Deep Learning?

Deep learning is the science and art of fitting models to data using functions formed by *composing* nonlinear parameterized functions,

$$\begin{aligned} f(x) &= f_m \circ f_{m-1} \circ \cdots f_1 \\ &= f_m\big(f_{m-1}\big(\ldots f_1(x)\big)\ldots\big) \end{aligned}$$



Each of these functions is referred to as a **layer**. The ChatGPT3 function has **96** layers and **175 billion** parameters!

What is Deep Learning? Example

Here is a simple example of a quark/gluon classifier:

$$f(x) = \text{softmax}\Big(\text{dropout}(\text{linear}(\text{flatten}\big(\boldsymbol{g}(\boldsymbol{c}(\boldsymbol{h}(\boldsymbol{c}(x))))\big)))\Big)$$

Here is an algorithm-level view:

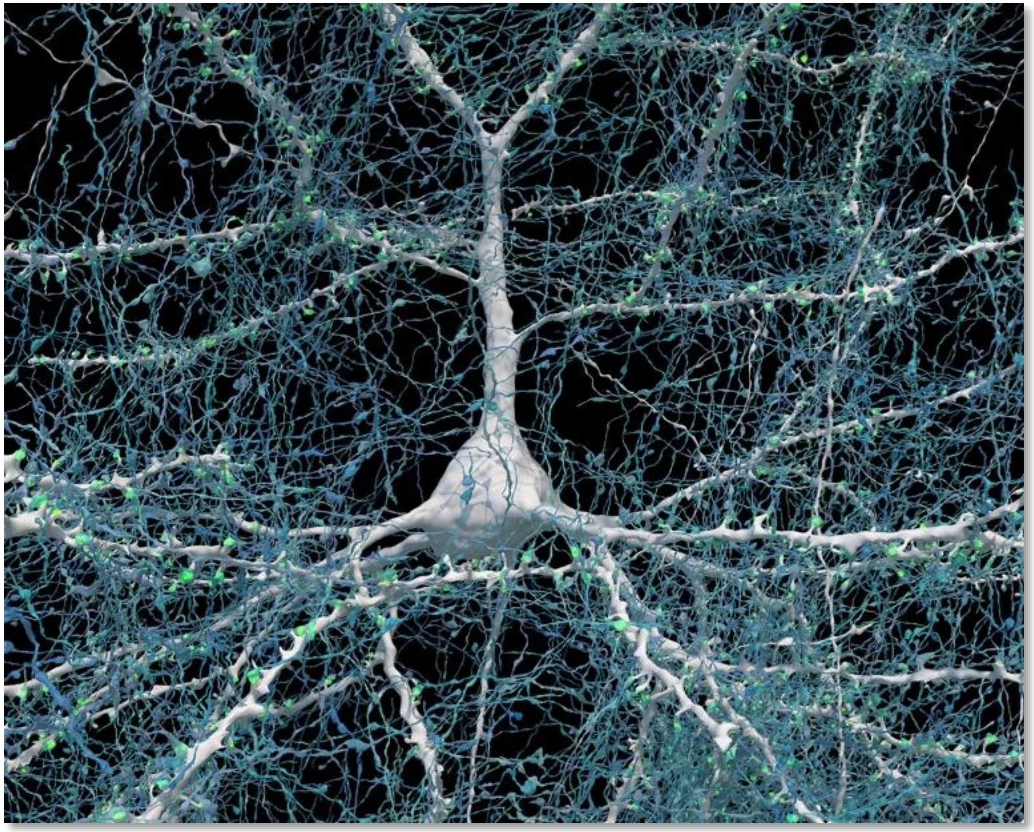
$$\begin{aligned} y &= \boldsymbol{c}(x) & y &= \text{flatten}(y) \\ y &= \boldsymbol{h}(y) & y &= \text{linear}(y) \\ y &= \boldsymbol{c}(y) & y &= \text{dropout}(y) \\ y &= \boldsymbol{g}(y) & f &= \text{softmax}(y) \end{aligned}$$

And here is a code-level view:

```
Sequential(
  (0): Conv2d(1, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False)
  (2): ReLU()
  (3): Conv2d(4, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (4): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): ReLU()
  (6): Flatten(start_dim=1, end_dim=-1)
  (7): Linear(in_features=64, out_features=2, bias=True)
  (8): Dropout(p=0.2, inplace=False)
  (9): Softmax(dim=1)
)
number of parameters: 318
```

BASIC BUILDING BLOCK:
THE PERCEPTRON

The Brain's Computational Unit



https://www.nature.com/articles/d41586-024-01387-9

Artificial Comp. Unit: The Perceptron

$$y = g(xA^T + b)$$

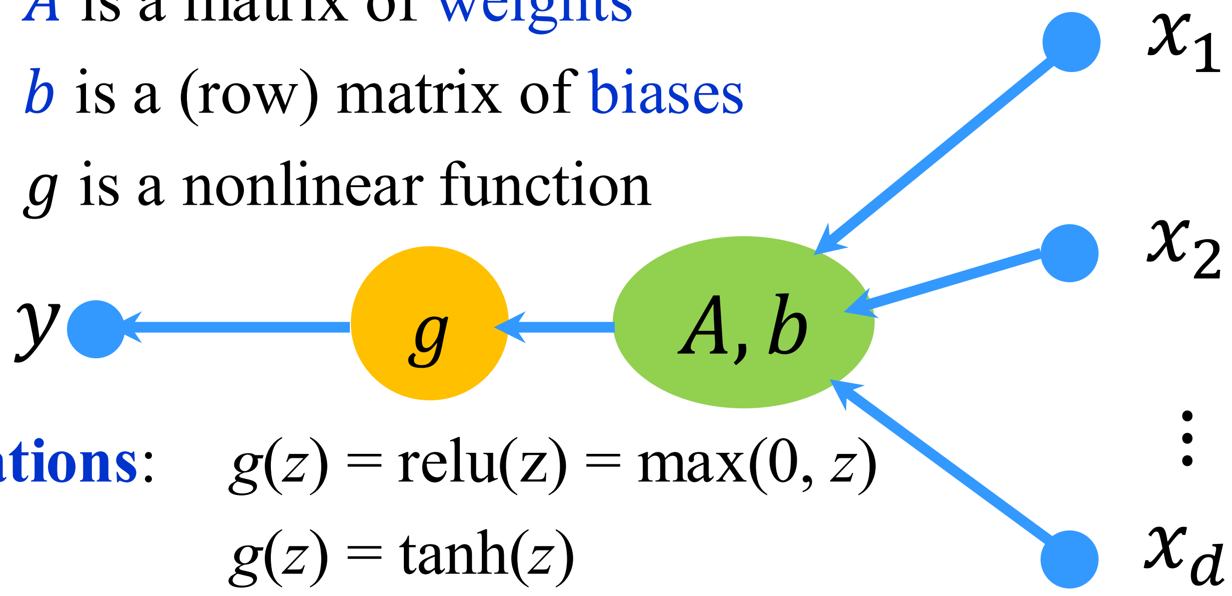
(Frank Rosenblatt, 1958)

x is a (row) matrix of input data

A is a matrix of weights

b is a (row) matrix of biases

g is a nonlinear function



Activations: $g(z) = \mathrm{relu}(z) = \max(0, z)$

$g(z) = \tanh(z)$

$g(z) = \mathrm{sigmoid}(z) = 1 / (1 + \exp(-z))$

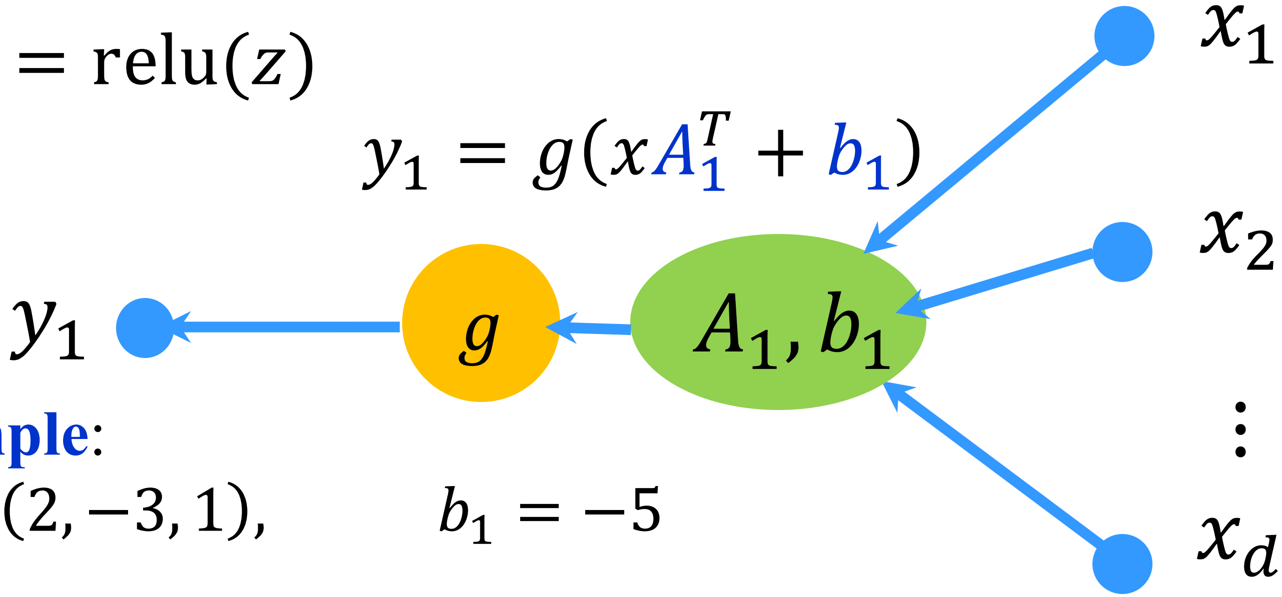
https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon

MULTI-NODE PERCEPTRON

Multi-Node Perceptron

$g(z) = \text{relu}(z)$

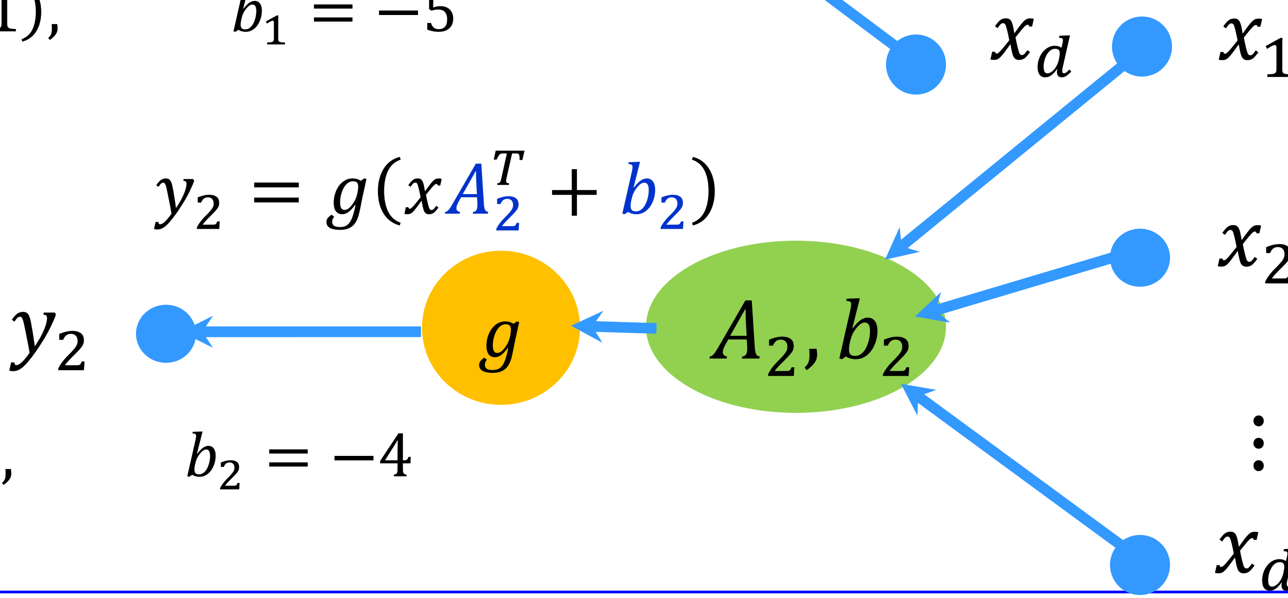
$y_1 = g(x{\color{blue}A_1^T} + {\color{blue}b_1})$



Example:

$A_1 = (2, -3, 1), \qquad b_1 = -5$

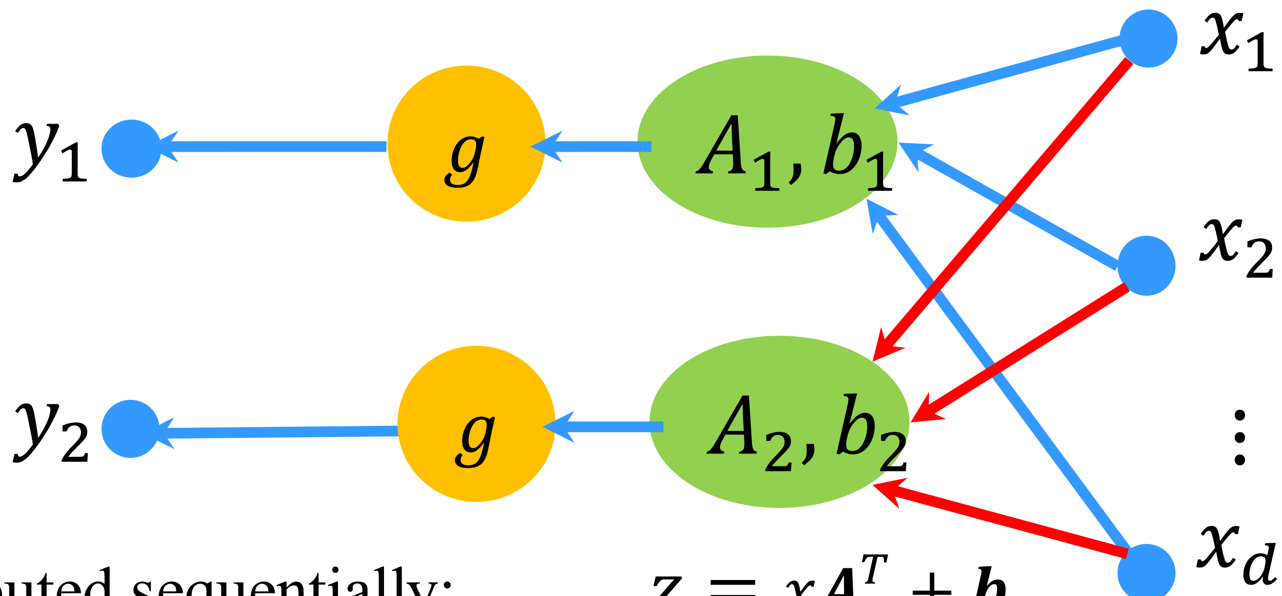
$y_2 = g(x{\color{blue}A_2^T} + {\color{blue}b_2})$



$A_2 = (1, 2, 3), \qquad b_2 = -4$

Multi-Node Perceptron

A multi-node perceptron is often drawn as follows,



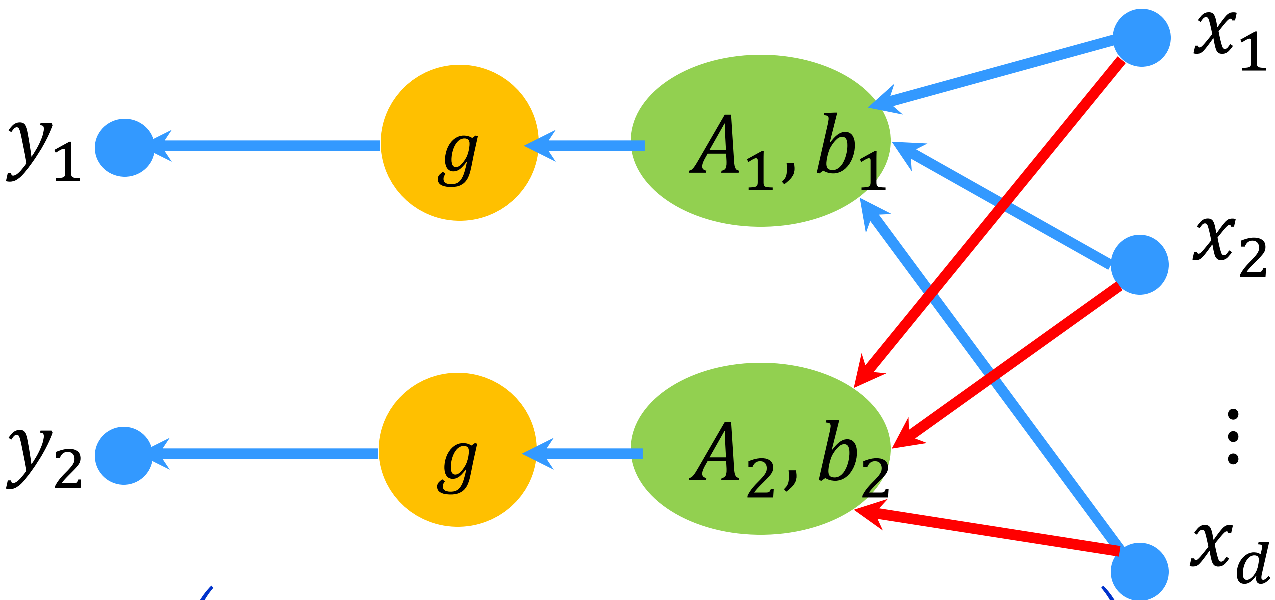
computed sequentially: $\quad z = x\boldsymbol{A}^T + \boldsymbol{b}$

$$y = g(z)$$

and, usually, the **activation function** $g(z)$ is applied *elementwise*, that is, to every element of its matrix input.

Multi-Node Perceptron: Example

$x = (-2, 1, 4)$, $\quad A = \begin{pmatrix} 2 & -3 & 1 \\ 1 & 2 & 3 \end{pmatrix}$, $\quad b = (-5, -4)$



$$y = \mathrm{relu}\left((-2,1,4) \begin{pmatrix} 2 & -3 & 1 \\ 1 & 2 & 3 \end{pmatrix}^{\mathrm{T}} + (-5, -4) \right)$$

Multi-Node Perceptron

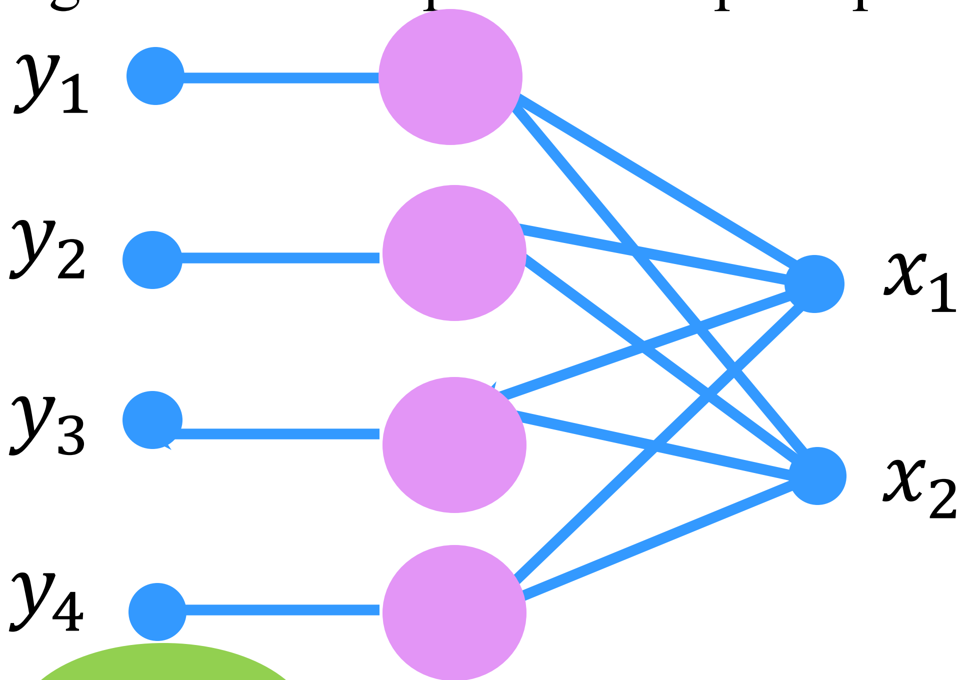
Consider the task of modeling the data triplets $D = \{(x_1, x_2, y)_{i=1}^N\}$ with a function of the form $y = f(x_1, x_2, \omega)$. Let's begin with a 2-input 4-node perceptron with $g(z) = z\sigma(z)$,

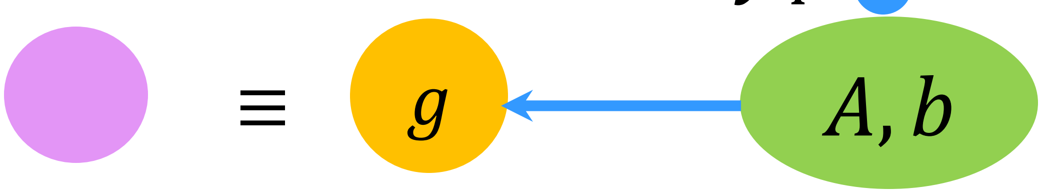
where,

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

$z\sigma(z)$ is called a silu.

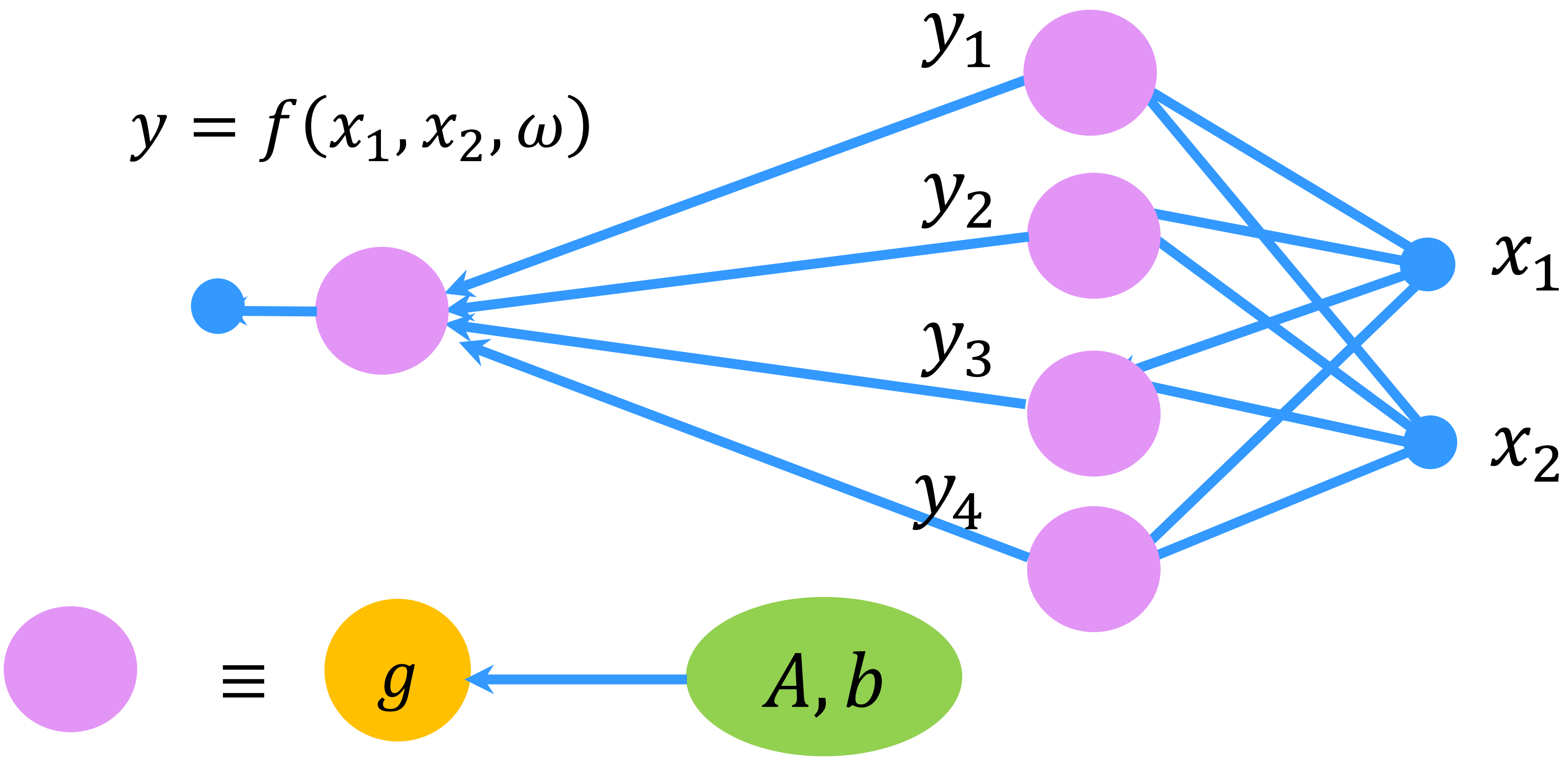
ω are free parameters.





Multi-Node Perceptron

Then, let's feed the output of the multi-node perceptron into a 4-input 1-node perceptron:



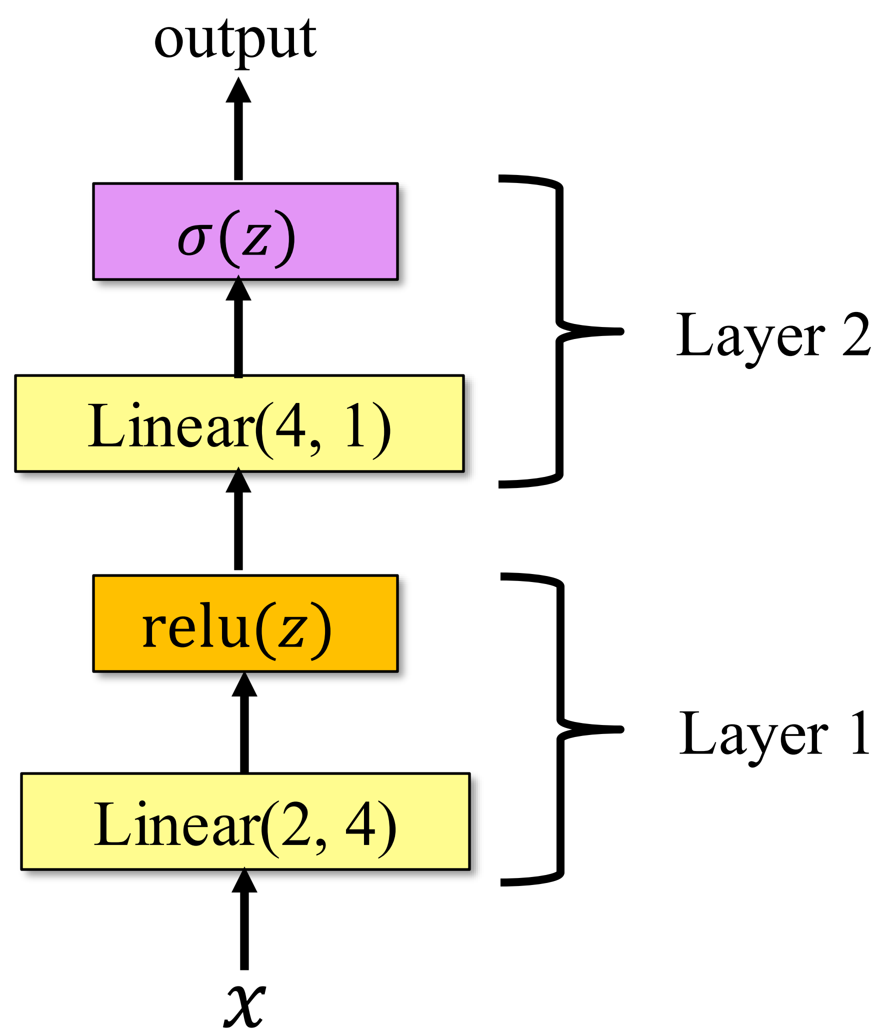
Multi-Node Perceptron

Given the complexity of ML models (i.e., ML functions), it is now common to use a graphical representation of ML models that uses higher-level components.

$f(z) = zA^T + b$

$\mathrm{relu}(z) = \max(0, z)$

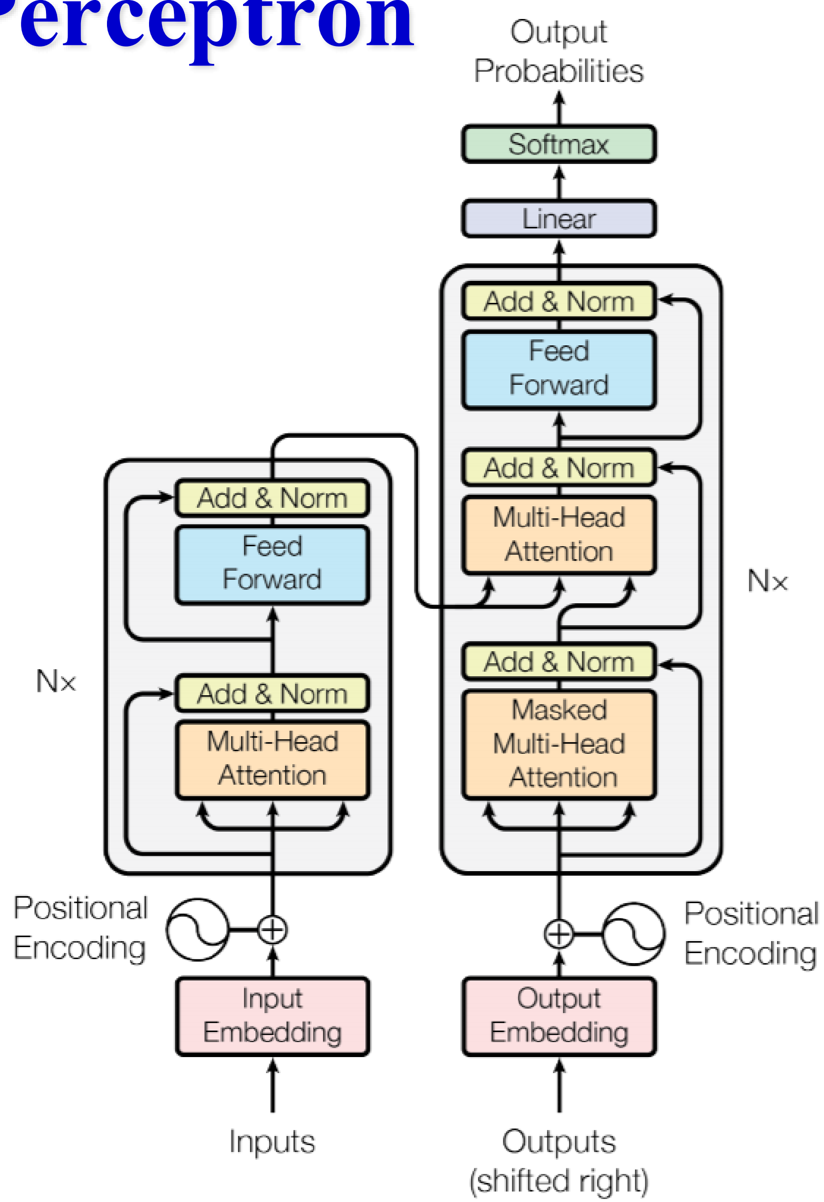
$\sigma(z) = (1 + e^{-z})^{-1}$



Multi-Node Perceptron

Here, for example, is a graphical representation of a **transformer**, the model that powers ChatGPT.

In ChatGPT 3, there are 2 x 96 of these "layers"!

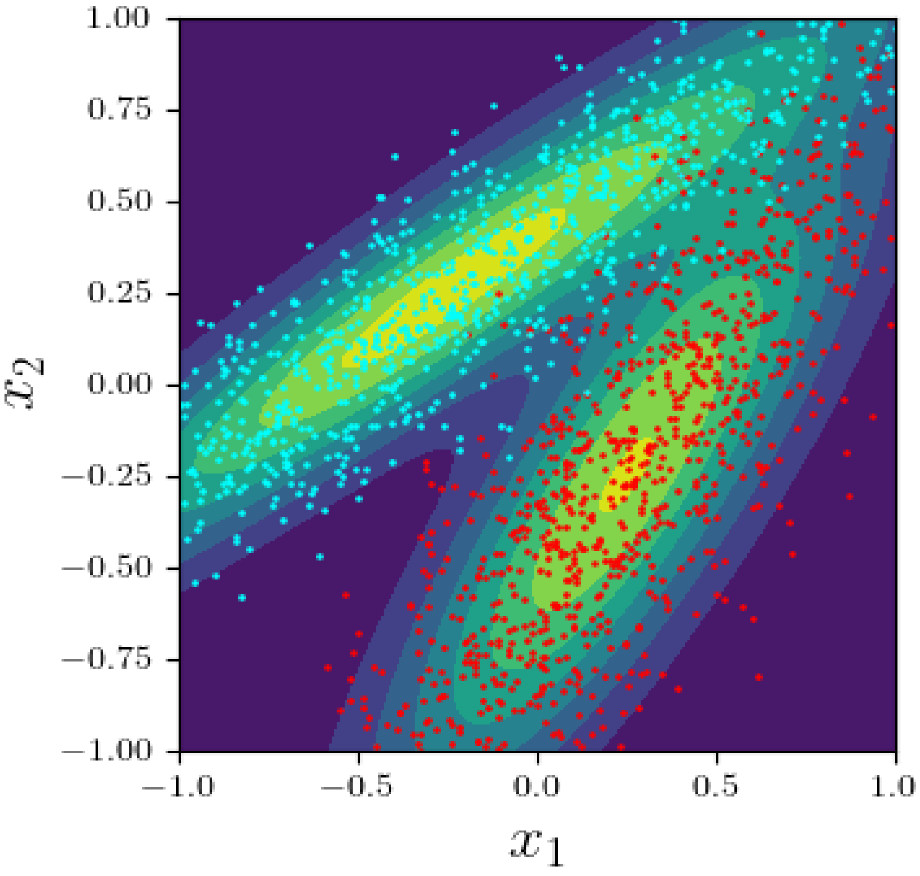


LOSS FUNCTION AND EMPIRICAL RISK

2D Dataset

- In this part of the course, we'll use a simple synthetic dataset comprising two classes of objects characterized by real numbers (x_1, x_2).

- The data are generated from two bivariate normal distributions.



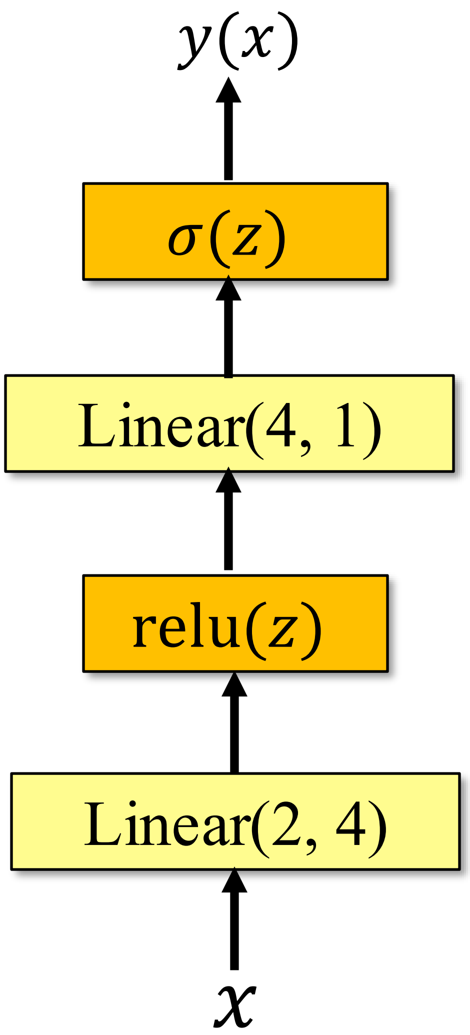
Loss Functions

Suppose our goal is to approximate an unknown function

$$y = f(x_1, x_2)$$

given a dataset $D = \{(x_1, x_2, y)_{i=1}^{N}\}$,

using the *model* on the right.



Loss Functions

A well-known approach is to approximate $f(x_1, x_2)$ with a parameterized function $f(x_1, x_2, \omega)$ and find the *best-fit values* ω^* by minimizing

$$R(\omega) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f_i)^2$$

with respect to the parameters ω, where $f_i = f(x_{1i}, x_{2i}, \omega)$.

This is referred to as a **least-squared fit**.

Loss Functions

In machine learning, the least-squares fit is generalized to

$$R(\omega) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f_i)$$

where the function, $L(y_i, f_i)$, called the **loss function**, measures the discrepancy between the desired **target** y and the model f.

The quantity $R(\omega)$ is called the **empirical risk** and the task of minimizing it is called **empirical risk minimization**.

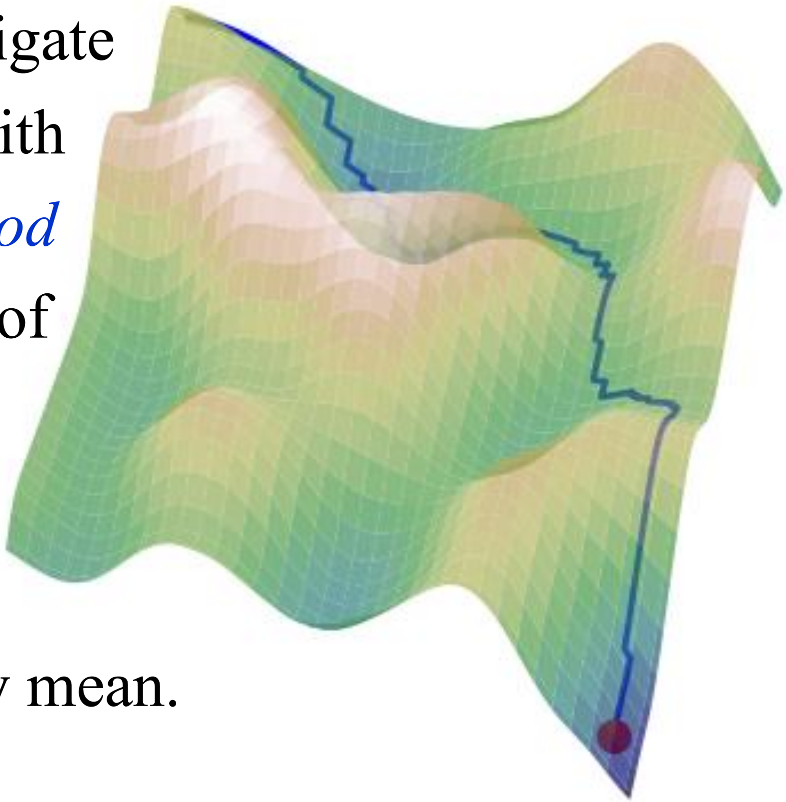
Warning: In ML, $R(\omega)$ is often referred to as the "loss".

Empirical Risk: Landscape

The empirical risk defines a highly corrugated very high-dimensional "landscape" in the space of parameters.

The goal of an **optimizer** is to navigate the landscape ($R(\omega)$) associated with a *finite* amount of data to find a *good approximation* of the lowest point of the landscape associated with an *infinite* amount of data.

When ML researchers say that a model *generalizes* this is what they mean.



Risk Functional

In mathematics, one can often gain insight by taking some limit of an expression. Let's take the limit of

$$R(\omega) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f_i)$$

as $N \to \infty$, that is, as the amount of data grows without limit.

In that limit, the empirical risk becomes the **risk functional**,

$$R[f] = \int dx \int dy \, L(y, f) \, p(x, y)$$

where $p(x, y) dx dy$ is the probability distribution of the data.

Summary (1)

- **General Approaches**
  - Supervised, unsupervised, generative, and reinforcement learning.
- **Deep Learning**
  - Uses functions constructed through deep composition.
- **The Perceptron**
  - Basic computational unit: matrix multiplication and addition and (typically) an element-wise nonlinear map.

Summary (2)

- We described the multi-node perceptron and how machine learning models are represented graphically.

- We illustrated the calculation of the output of a perceptron.

- We discussed the generalization of least-squared fitting to empirical risk minimization.