Machine Learning I

Yu-Chung Wang

Core Knowledge

ML - Neural Network pipeline

- **How does machine calculate?**
- **Set training, validating, testing**
- Evaluation Matrix (Next week)
- Overfitting/Underfitting, Regularization (Next week)
- How to connect to ***probrability and utility***?

Various Applications and modern ML models

- How can we find new ML methods?
- XAI

Large Language Model

- Core method on LLM
- Retrieval-Augmented Generation (RAG)

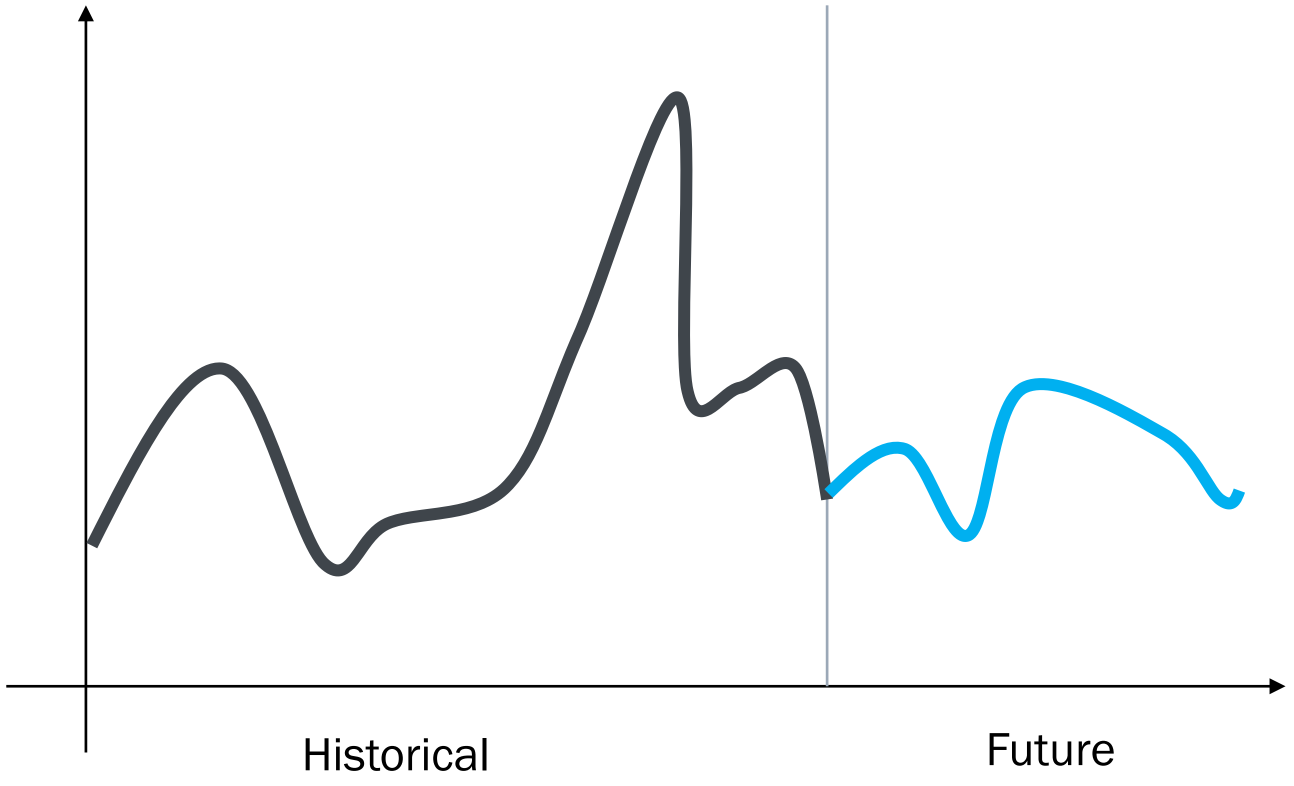
ML model basically does only **one** thing

Prediction

“ The current wave of advances in artificial intelligence doesn’t actually bring us intelligence but instead a critical component of intelligence: prediction. ”

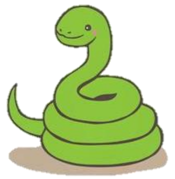
Agrawal-Gans-Goldfarb, “Prediction Machines”

These predictions are not just about the future

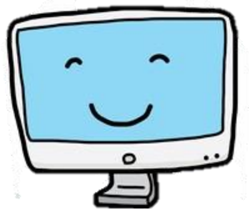


Using historical time series data to predict the future is, of course, called forecasting

Making accurate judgments in situations that haven't been encountered before is also called 'prediction'

Question	Answer
	Formosan black bear
	Snake
	?

I have never seen this photo before, but I know it's a **Formosan black bear**!



Predictor machine

Same Concept

Input:

Questions

Text

Image

Voice

Different ML Models

GPT-5
Gemini2.0 Ultra
DeepSeek R1
Claude3.7

DALL-E
Imagen
Stable Diffusion
………

Output:

An answer

Answers

Image

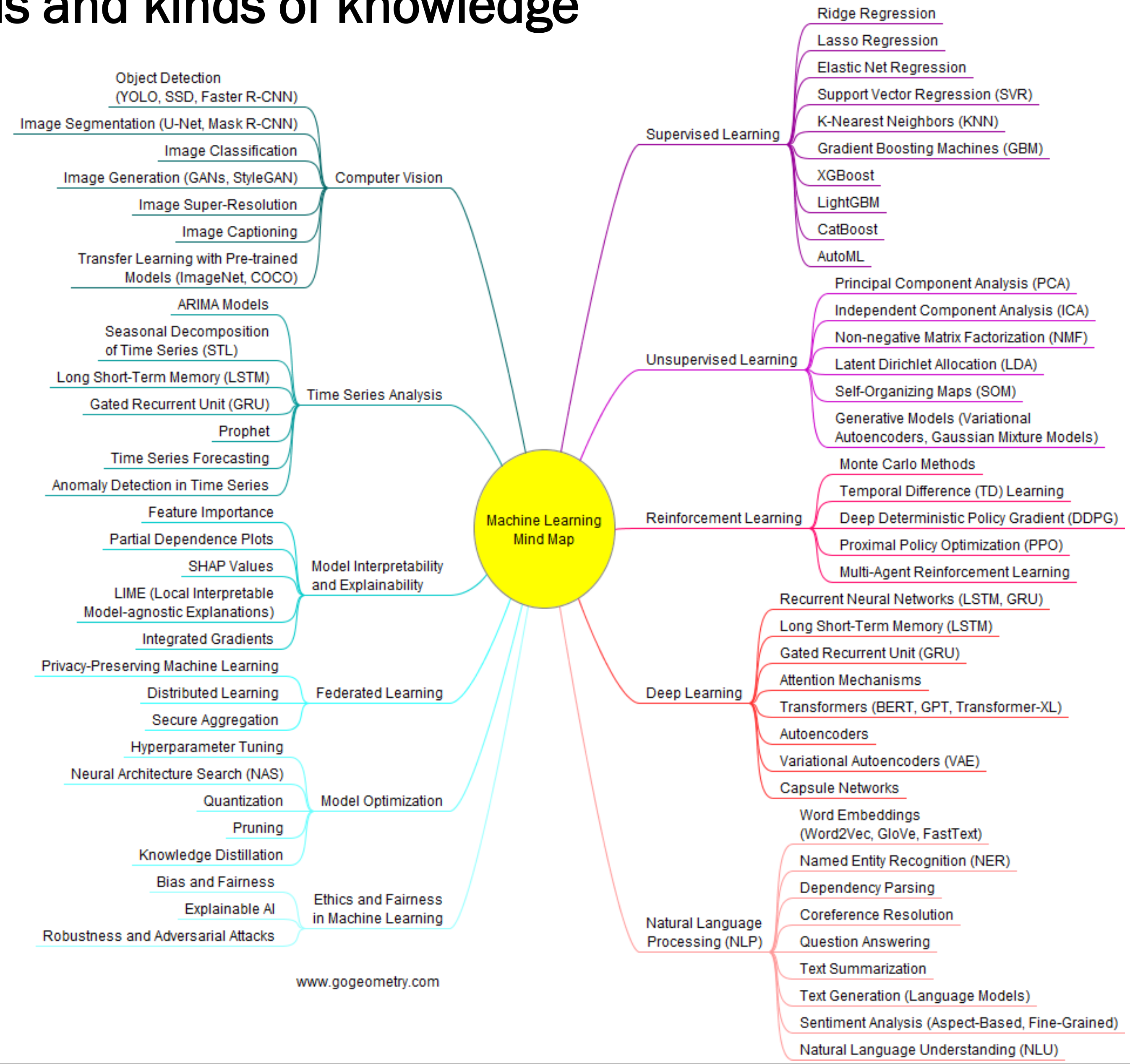
Text

Video

Voice

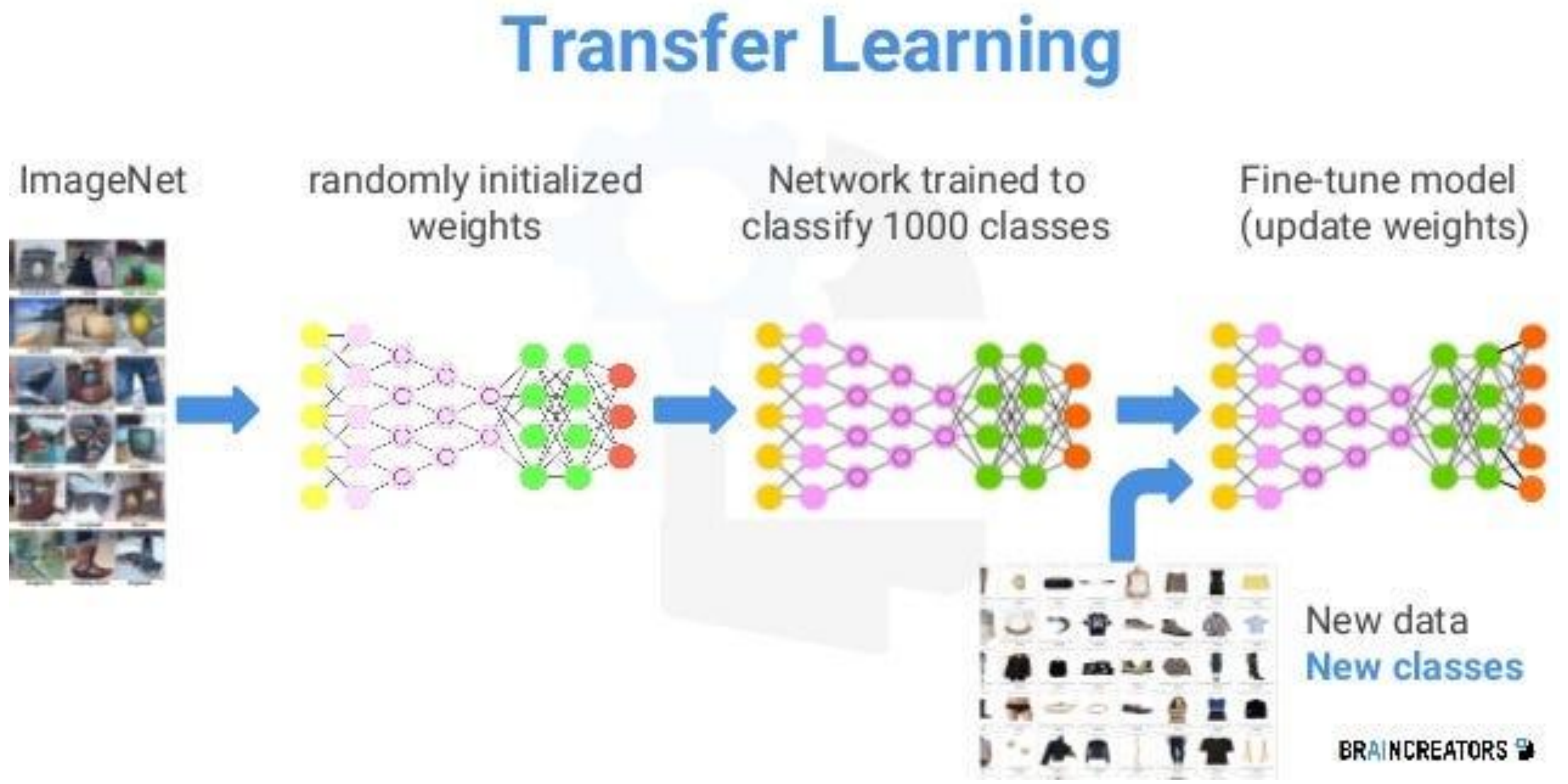
There are many different ML methods and kinds of knowledge

1. What should we do?
2. How should we start?



Machine Learning Mind Map
Supervised Learning
Ridge Regression
Lasso Regression
Elastic Net Regression
Support Vector Regression (SVR)
K-Nearest Neighbors (KNN)
Gradient Boosting Machines (GBM)
XGBoost
LightGBM
CatBoost
AutoML
Unsupervised Learning
Principal Component Analysis (PCA)
Independent Component Analysis (ICA)
Non-negative Matrix Factorization (NMF)
Latent Dirichlet Allocation (LDA)
Self-Organizing Maps (SOM)
Generative Models (Variational Autoencoders, Gaussian Mixture Models)
Reinforcement Learning
Monte Carlo Methods
Temporal Difference (TD) Learning
Deep Deterministic Policy Gradient (DDPG)
Proximal Policy Optimization (PPO)
Multi-Agent Reinforcement Learning
Deep Learning
Recurrent Neural Networks (LSTM, GRU)
Long Short-Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Attention Mechanisms
Transformers (BERT, GPT, Transformer-XL)
Autoencoders
Variational Autoencoders (VAE)
Capsule Networks
Natural Language Processing (NLP)
Word Embeddings (Word2Vec, GloVe, FastText)
Named Entity Recognition (NER)
Dependency Parsing
Coreference Resolution
Question Answering
Text Summarization
Text Generation (Language Models)
Sentiment Analysis (Aspect-Based, Fine-Grained)
Natural Language Understanding (NLU)
Computer Vision
Object Detection (YOLO, SSD, Faster R-CNN)
Image Segmentation (U-Net, Mask R-CNN)
Image Classification
Image Generation (GANs, StyleGAN)
Image Super-Resolution
Image Captioning
Transfer Learning with Pre-trained Models (ImageNet, COCO)
Time Series Analysis
ARIMA Models
Seasonal Decomposition of Time Series (STL)
Long Short-Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Prophet
Time Series Forecasting
Anomaly Detection in Time Series
Model Interpretability and Explainability
Feature Importance
Partial Dependence Plots
SHAP Values
LIME (Local Interpretable Model-agnostic Explanations)
Integrated Gradients
Federated Learning
Privacy-Preserving Machine Learning
Distributed Learning
Secure Aggregation
Model Optimization
Hyperparameter Tuning
Neural Architecture Search (NAS)
Quantization
Pruning
Knowledge Distillation
Ethics and Fairness in Machine Learning
Bias and Fairness
Explainable AI
Robustness and Adversarial Attacks
www.gogeometry.com

With a solid grasp of the basics, you can easily learn any new machine learning technique.


Transfer Learning
ImageNet
randomly initialized weights
Network trained to classify 1000 classes
Fine-tune model (update weights)
New data
New classes
BRAINCREATORS

Prelimary knowlege

What is a function?

A function is essentially an answer key.

For every question, there is an answer.

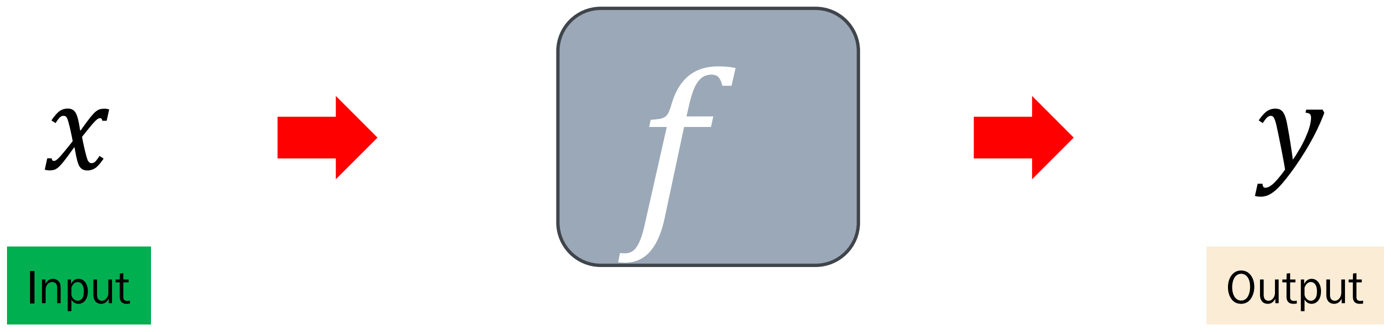
$\boxed{f}$

Prelimary Knowledge for ML

Prelimary knowledge

Convert a problem into a function

Then find out this function:



Def: Function

Mathematical Definition:

Let $X, Y \in \mathbb{R}$.
A function f is a mapping from a set $\boldsymbol{X}$ to another set $\boldsymbol{Y}$. The mapping of $\boldsymbol{x}$ to $\boldsymbol{y}$ is written as $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{y}$.

To be a function, it must satisfy two conditions:

Let X, Y be two sets

$$f\colon X \rightarrow Y$$

Prelimary knowledge

Def: Function

1

For all $x \in X, \exists y \in Y$, such that $f(x) = y$

For every x in the set X, there exists **one** y in the set Y, such that $f(x) = y$.

Quiz

Exercise 1.

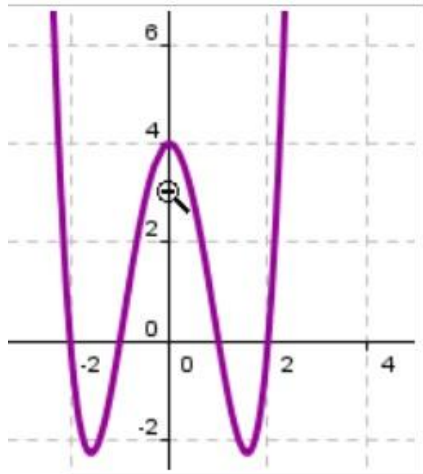
Prelimary knowledge

Def: Function

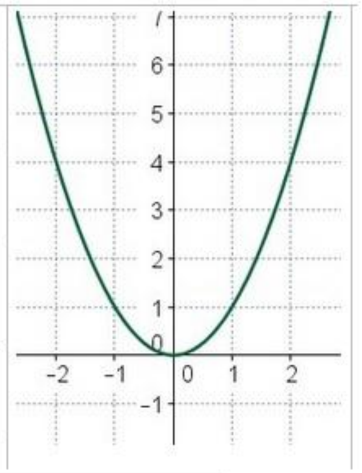
2

$\boldsymbol{X} \in \mathbb{R}^1, \boldsymbol{Y} \in \mathbb{R}^1$

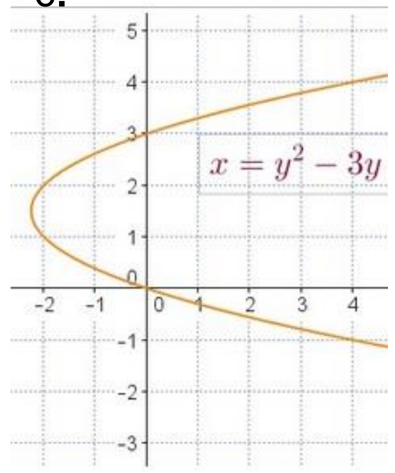
a.



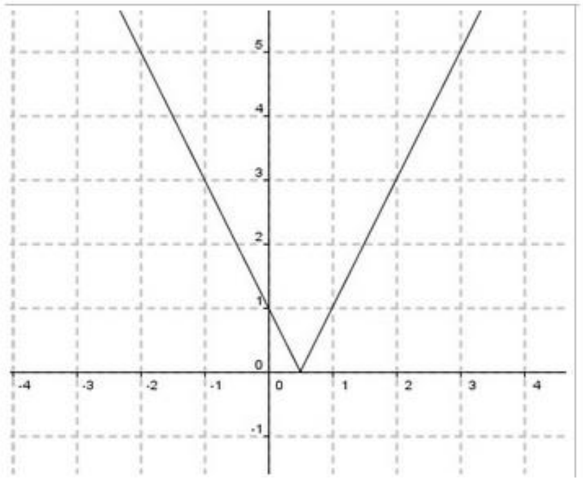
b.



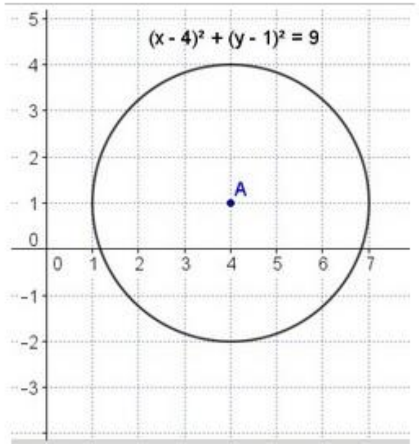
c.



d.



e.



Prelimary knowledge

Def: Function

2

Let $x_1, x_2 \in X$, such that $x_1 = x_2$,
$f(x_1) = f(x_2)$

Exercise 2.

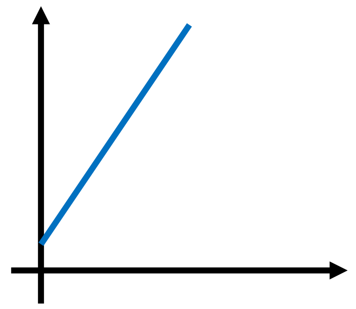
Prelimary knowledge

From Mathematical Function to Learned Function — What Machine Learning Really Does

Mathematical Function

Definition: $f: X \to Y$

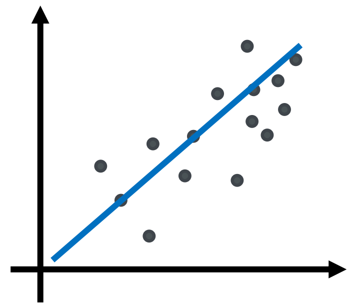
Example: $f(x) = 2x + 1$



Machine Learning

Use data to **learn an approximation** $\hat{f}_\theta(x) \approx f^*(x)$.

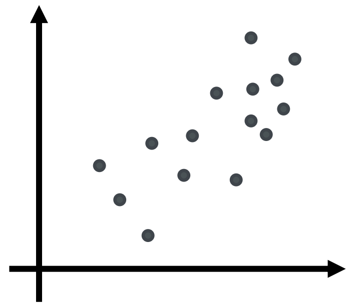
Neural network = parametric function
$\longrightarrow$ weights θ are adjusted to minimize error.



$$\hat{f}_\theta(x) = 1 \times x + 0 = x$$

Real World

We only observe samples: $(x_1, y_1), (x_2, y_2), \ldots$
but the true mapping $f^*(x)$ is unknown.



Same Concept

Input:

Questions

Text

Image

Voice

Different ML Models

GPT-5
Gemini2.0 Ultra
DeepSeek R1
Claude3.7

DALL-E
Imagen
Stable Diffusion
………

Output:

An answer

Answers

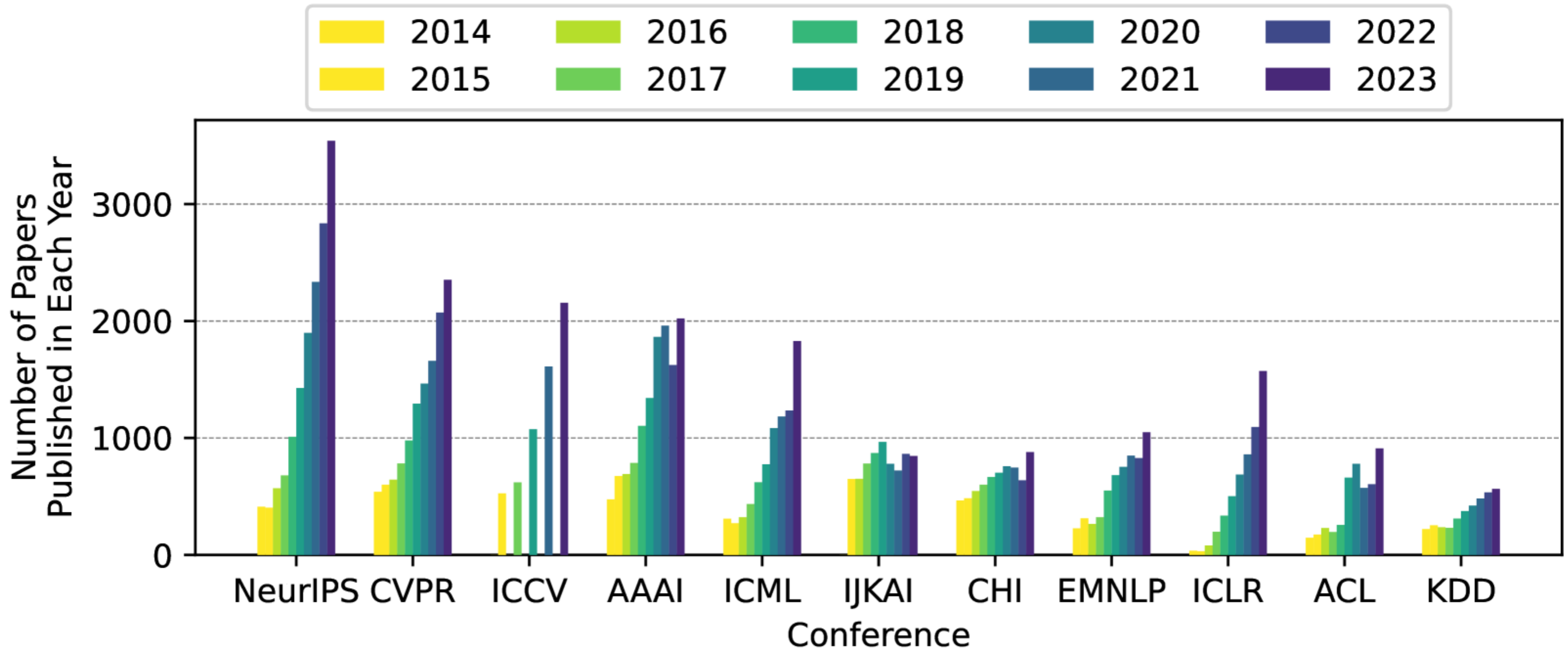
Image

Text

Video

Voice

AI/ML Publications from 2014 to 2023



From Resistance to Reflection — How Humans Adapt to AI

Ke Jie

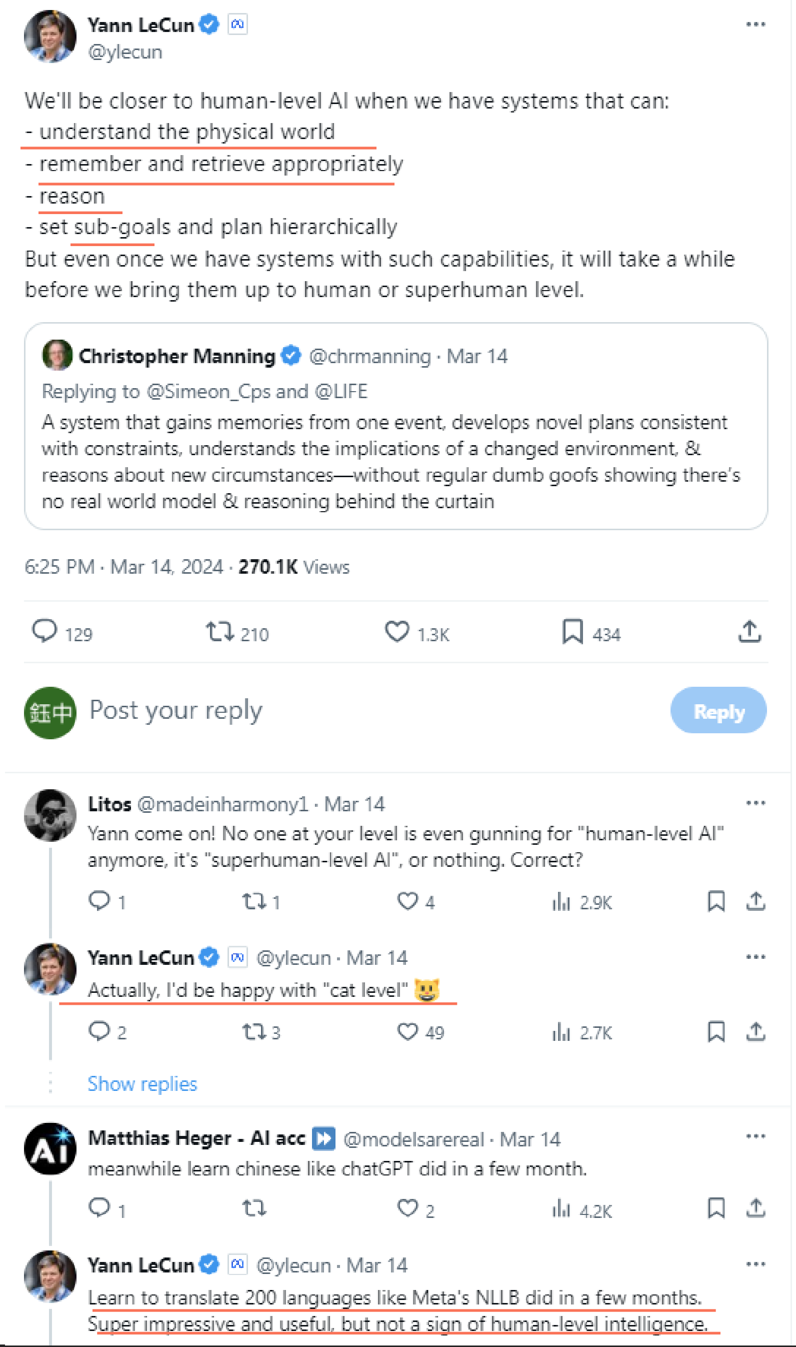
Before playing with alphago

2016: Ke Jie said, **“It can’t beat me.”**
2016: “I won’t play against AlphaGo — **I don’t want it to learn my divine-level strategies.”**

After playing with alphago

2016: AlphaGo is truly **the most powerful opponent I have ever faced!**

2025: “There’s no need to worry about AI collecting our data. Our skill level is so low — why would it even want our data? **It would just pollute its database.”**



Yann LeCun @ylecun

We'll be closer to human-level AI when we have systems that can:
- understand the physical world
- remember and retrieve appropriately
- reason
- set sub-goals and plan hierarchically

But even once we have systems with such capabilities, it will take a while before we bring them up to human or superhuman level.

> Christopher Manning @chrmanning · Mar 14
> Replying to @Simeon_Cps and @LIFE
> A system that gains memories from one event, develops novel plans consistent with constraints, understands the implications of a changed environment, & reasons about new circumstances—without regular dumb goofs showing there's no real world model & reasoning behind the curtain

6:25 PM · Mar 14, 2024 · **270.1K** Views

129 · 210 · 1.3K · 434

Post your reply · Reply

Litos @madeinharmony1 · Mar 14
Yann come on! No one at your level is even gunning for "human-level AI" anymore, it's "superhuman-level AI", or nothing. Correct?

1 · 1 · 4 · 2.9K

Yann LeCun @ylecun · Mar 14
Actually, I'd be happy with "cat level" 😺

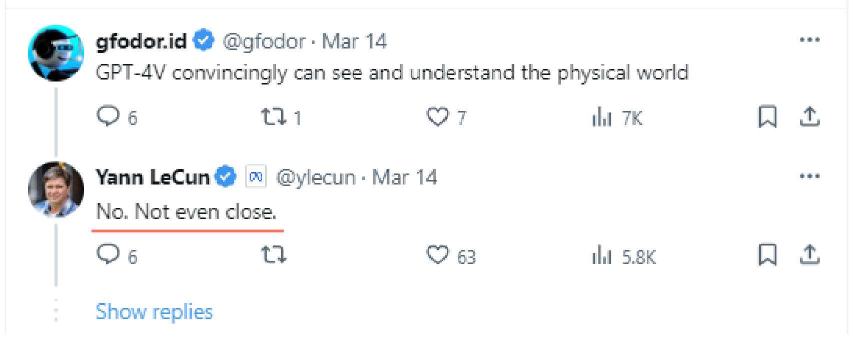
2 · 3 · 49 · 2.7K

Show replies

Matthias Heger - AI acc ⏩ @modelsarereal · Mar 14
meanwhile learn chinese like chatGPT did in a few month.

1 · 2 · 4.2K

Yann LeCun @ylecun · Mar 14
Learn to translate 200 languages like Meta's NLLB did in a few months.
Super impressive and useful, but not a sign of human-level intelligence.



gfodor.id @gfodor · Mar 14
GPT-4V convincingly can see and understand the physical world

6 · 1 · 7 · 7K

Yann LeCun @ylecun · Mar 14
No. Not even close.

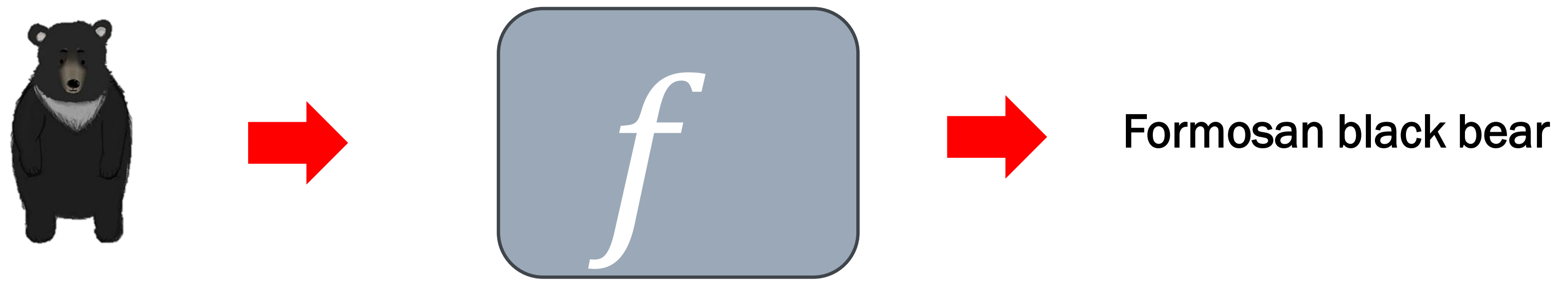
6 · 63 · 5.8K

Show replies

Prelimary knowledge

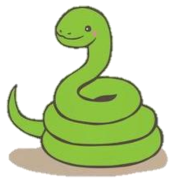
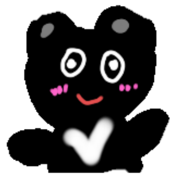
Convert a problem into a function

Then find out this function



Prelimary knowledge-

We have partial answers

Question	Answer
	Formosan black bear
	Snake
	?

Take note of this example, it's impossible for us to gather all possible scenarios!

Prelimary knowledge-

Create a 'Function Learning Machine' to fully master functions!

We're building a 'Function Learning Machine' to master this function!
(The method is to continuously work on 'past exam questions').

Prelimary knowledge-

In order to have the computer compute, we will require...

1

$$X \in \mathbb{R}^n, \; Y \in \mathbb{R}^m$$

Prelimary knowledge-

In order to have the computer compute, we will require...

2

Note that both the input and output must be numbers of fixed dimensions (such as matrices, tensors, etc.).

Prelimary knowledge-

In order to have the computer compute, we will require...

2

Scalar	$x = 9$	0th-order tensor
Vector	$x = [9,4,8,7]$	1st-order tensor
Matrix	$x = \begin{bmatrix} 9 & 4 \\ 8 & 7 \end{bmatrix}$	2nd-order tensor

Prelimary knowledge-

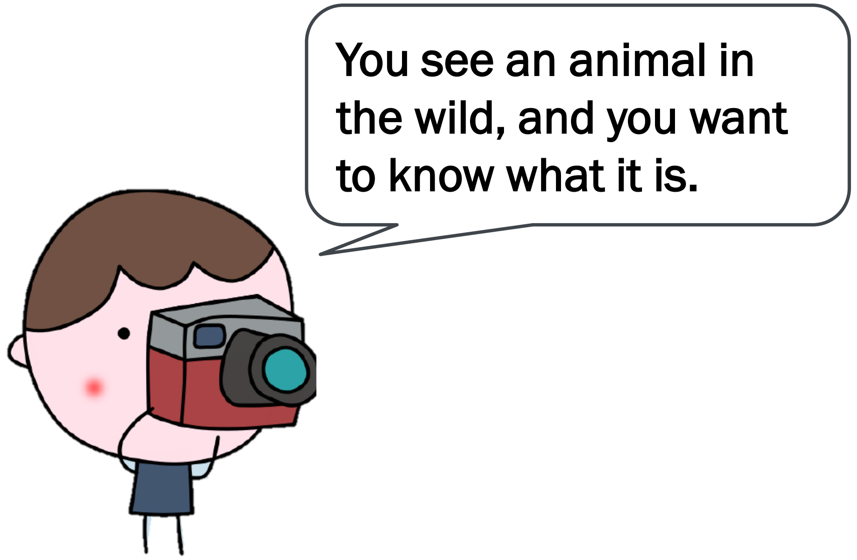
How to implement an ML model

Hard

Easy

Pipeline

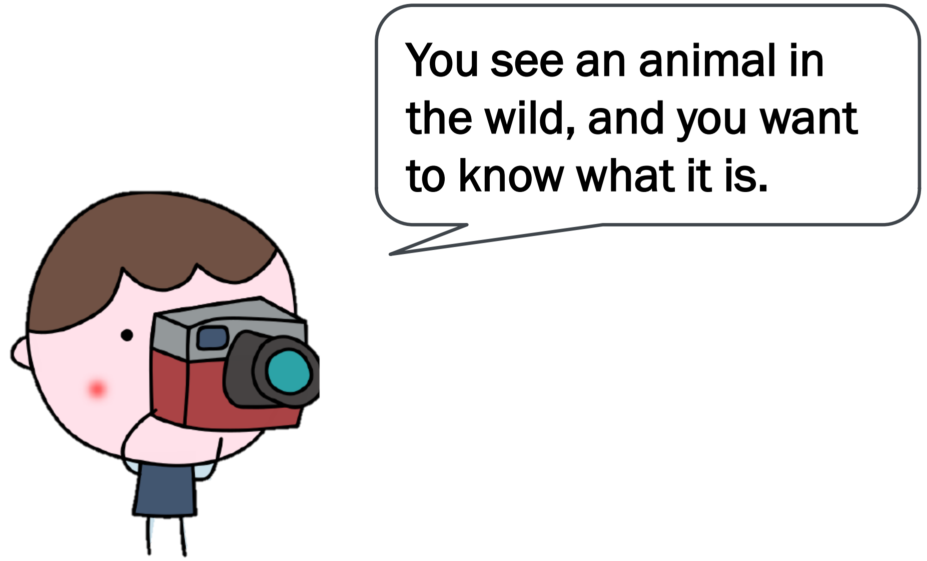
1. Ask a question
2. Convert the question into a functional form
3. Prepare data
4. Build the "function learning machine“
5. Train the model
6. Make predictions


You see an animal in the wild, and you want to know what it is.

Prelimary knowledge-

How to implement an ML model

1. Ask a question


You see an animal in the wild, and you want to know what it is.

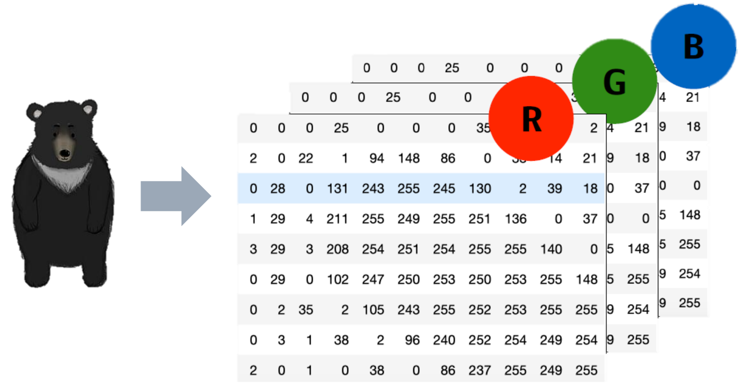
Prelimary knowledge-
How to implement an ML model

2. Convert the question into a functional form

$$\begin{tikzcd}\text{Input} \arrow[r] & \boxed{\hat{f}_\theta(x)} \arrow[r] & ?\end{tikzcd}$$

Prelimary knowledge-

How to implement an ML model



3. Prepare data

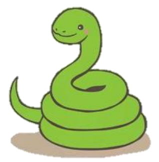
$$\begin{array}{l}\begin{bmatrix}1\\0\\0\end{bmatrix}\end{array}\begin{array}{l}\text{Formosan black bear}\\ \text{Snake}\\ \text{Pigeon}\end{array}$$

1

Formosan black bear

$\begin{bmatrix}1\\0\\0\end{bmatrix}$

2



Snake

$\begin{bmatrix}0\\1\\0\end{bmatrix}$

3

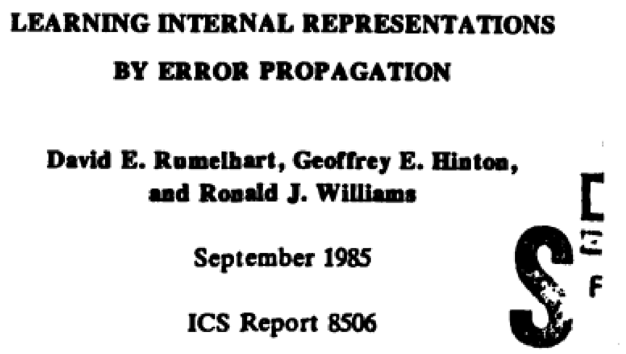
Pigeon

$\begin{bmatrix}0\\0\\1\end{bmatrix}$

This method is called one-hot encoding
(Or you can use Label encoding)

If the output is **not a number**, we'll assign it one ourselves and give it a number!

One-hot encoding

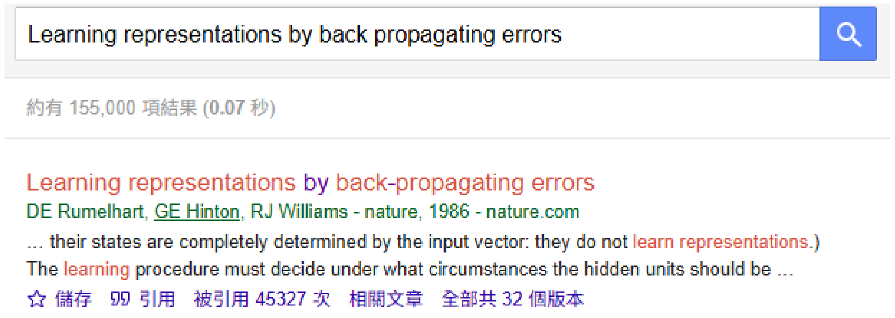


LEARNING INTERNAL REPRESENTATIONS
BY ERROR PROPAGATION

David E. Rumelhart, Geoffrey E. Hinton,
and Ronald J. Williams

September 1985

ICS Report 8506

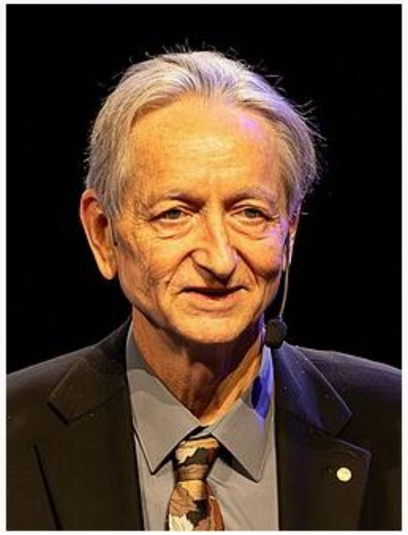


Learning representations by back propagating errors

約有 155,000 項結果 (0.07 秒)

Learning representations by back-propagating errors
DE Rumelhart, GE Hinton, RJ Williams - nature, 1986 - nature.com
... their states are completely determined by the input vector: they do not learn representations.)
The learning procedure must decide under what circumstances the hidden units should be ...
☆ 儲存 99 引用 被引用 45327 次 相關文章 全部共 32 個版本

Geoffrey Hinton

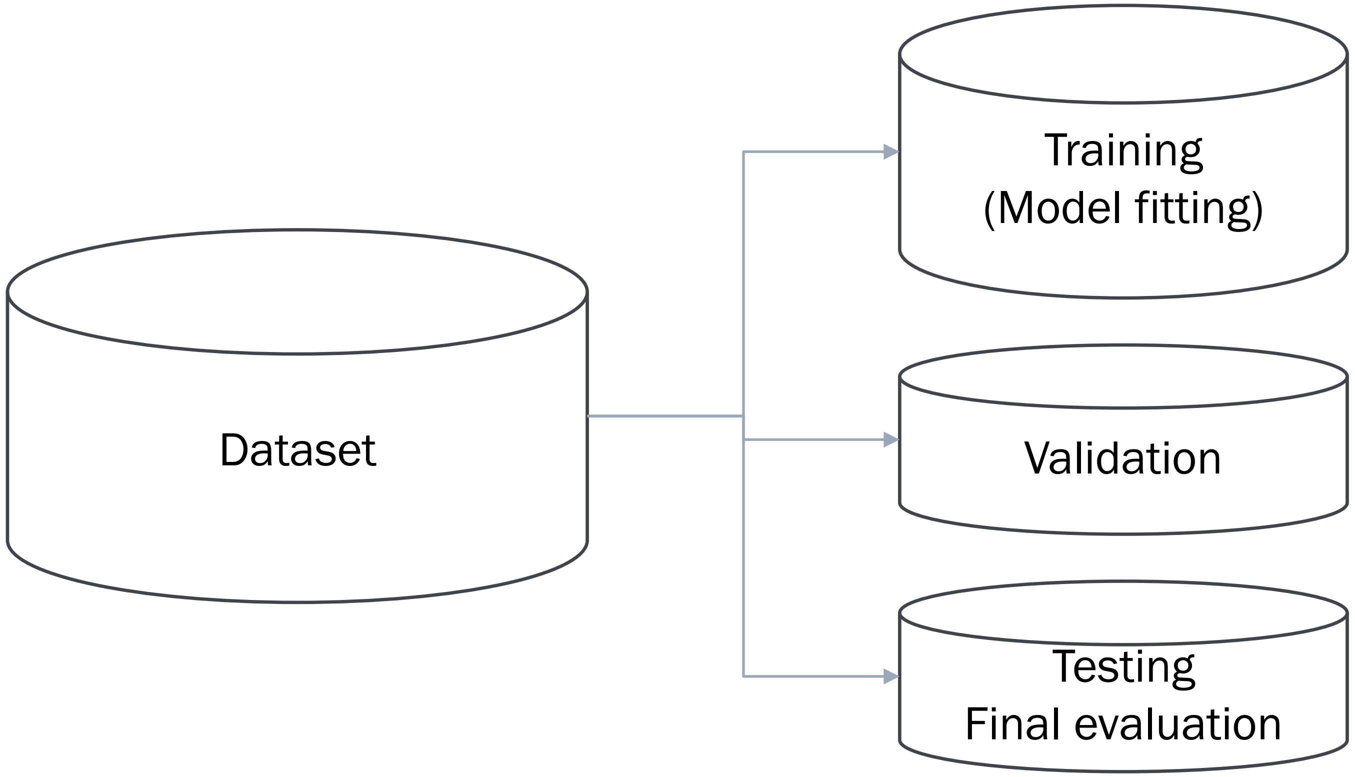


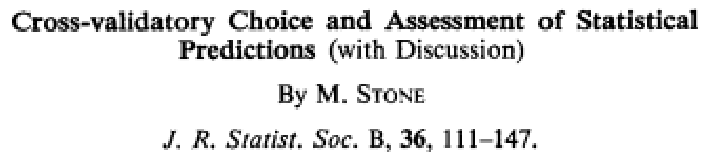
Backpropagation,
Boltzmann Machine,
Alexnet

Prelimary knowledge-

How to implement an ML model

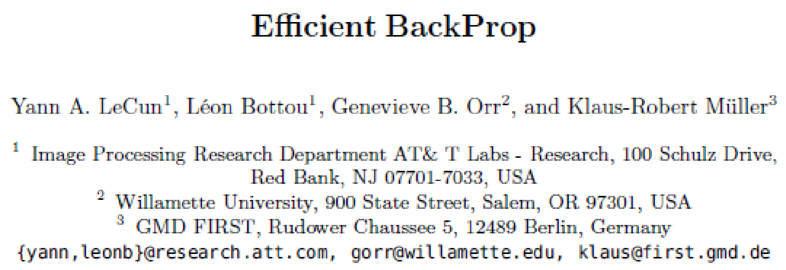
3. Prepare data




Cross-validatory Choice and Assessment of Statistical Predictions (with Discussion)

By M. Stone

J. R. Statist. Soc. B, **36**, 111–147.


Efficient BackProp

Yann A. LeCun[1], Léon Bottou[1], Genevieve B. Orr[2], and Klaus-Robert Müller[3]

[1] Image Processing Research Department AT& T Labs - Research, 100 Schulz Drive, Red Bank, NJ 07701-7033, USA
[2] Willamette University, 900 State Street, Salem, OR 97301, USA
[3] GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany
{yann,leonb}@research.att.com, gorr@willamette.edu, klaus@first.gmd.de

Must shuffle training set

Prelimary knowlege-

How to implement an ML model

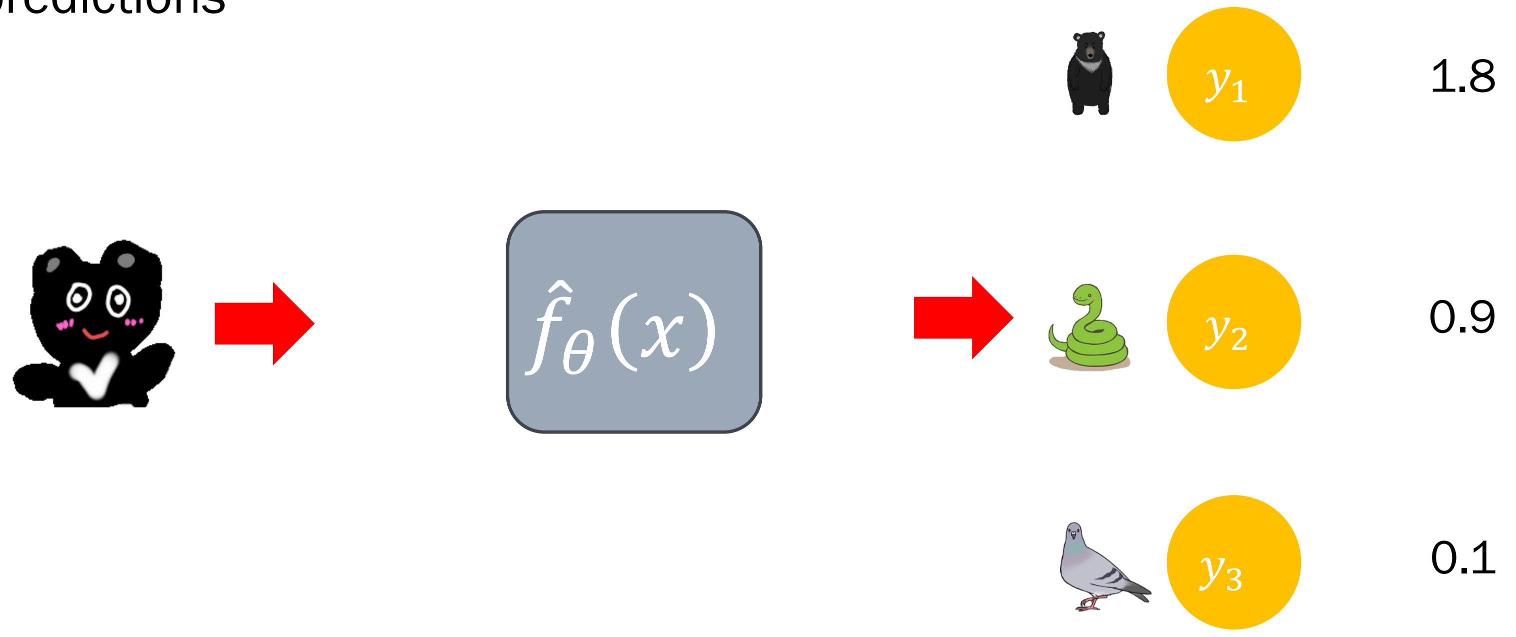
4. Build the "function learning machine“
5. Train the model

What you will learn this week

Prelimary knowlege-
How to implement an ML model

In our example, after applying **one-hot** encoding, the function learning machine will learn to take an input (a photo) and output three numbers, such as:

6. Make predictions



We would know that the computer has determined the most likely result is a **Formosan black bear.**

Three major types of neural networks

Neural networks (NN)

Convolution neural networks (CNN)

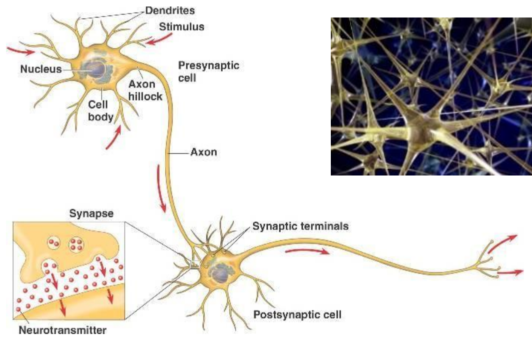
- Image identification
- Dimension reduction

Recurrent neural networks (RNN)

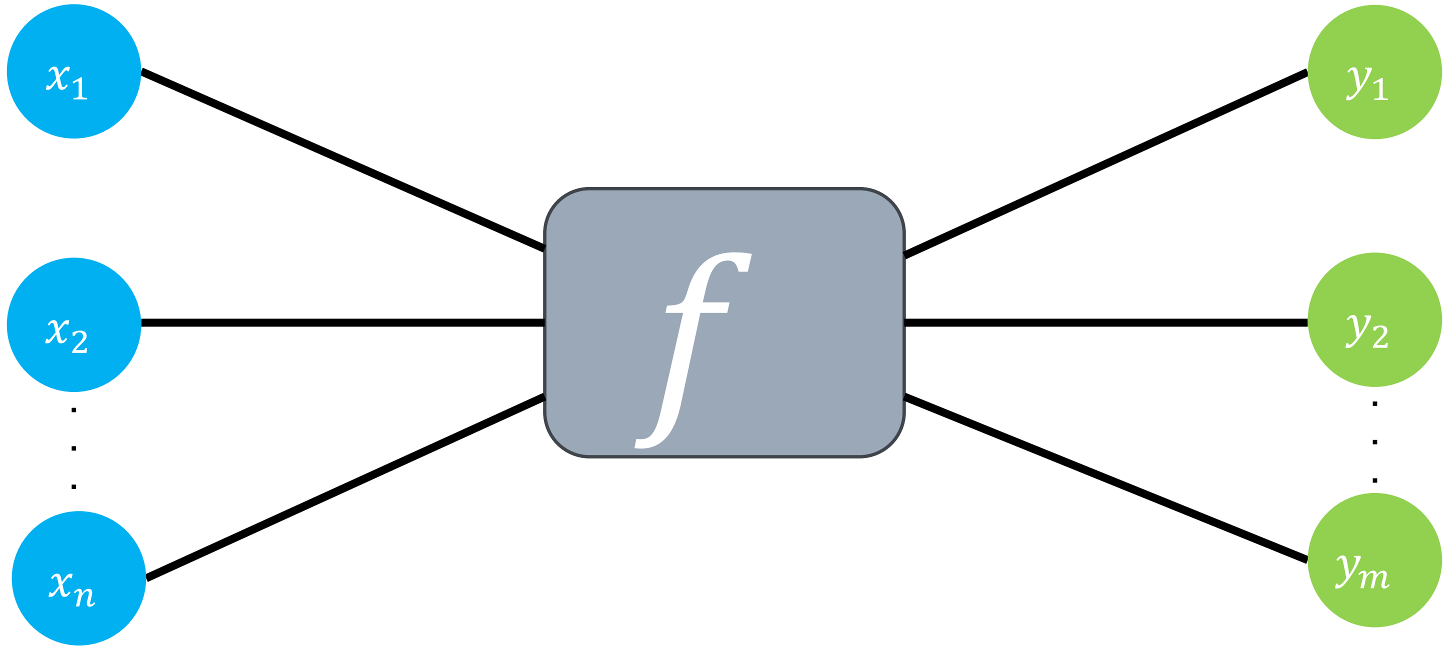
- Time series data

Let’s start from

Neural networks (NN)



Remember, we're just trying to learn a function.

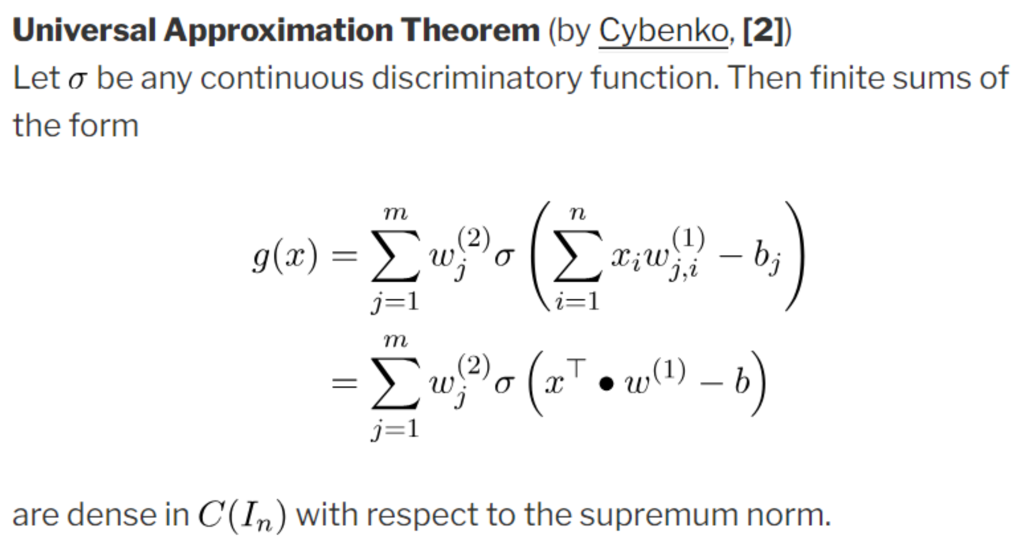


The Power of Neural Networks Proven!

It tells us that neural networks have a strong ability to **approximate functions**, even with just a **single hidden layer**.

Universal Approximation Theorem

The Power of Neural Networks Proven!


Universal Approximation Theorem (by Cybenko, **[2]**)
Let σ be any continuous discriminatory function. Then finite sums of the form

$$g(x) = \sum_{j=1}^{m} w_j^{(2)} \sigma \left(\sum_{i=1}^{n} x_i w_{j,i}^{(1)} - b_j \right) = \sum_{j=1}^{m} w_j^{(2)} \sigma \left(x^\top \bullet w^{(1)} - b \right)$$

are dense in $C(I_n)$ with respect to the supremum norm.

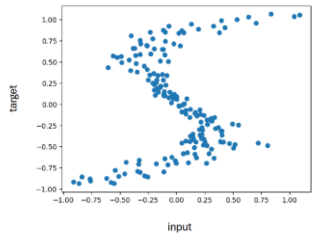
Universal Approximation Theorem

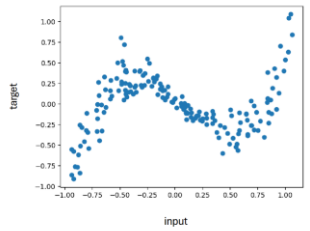
The Universal Approximation Theorem states that (potentially) a **feedforward neural network with a single hidden layer** containing a **sufficient number of neurons** and using a **non-linear activation function** can approximate **any continuous real-valued function** to an **arbitrarily small error.**

https://www.youtube.com/watch?v=Ln8pV1AXAgQ

Neural Network Method

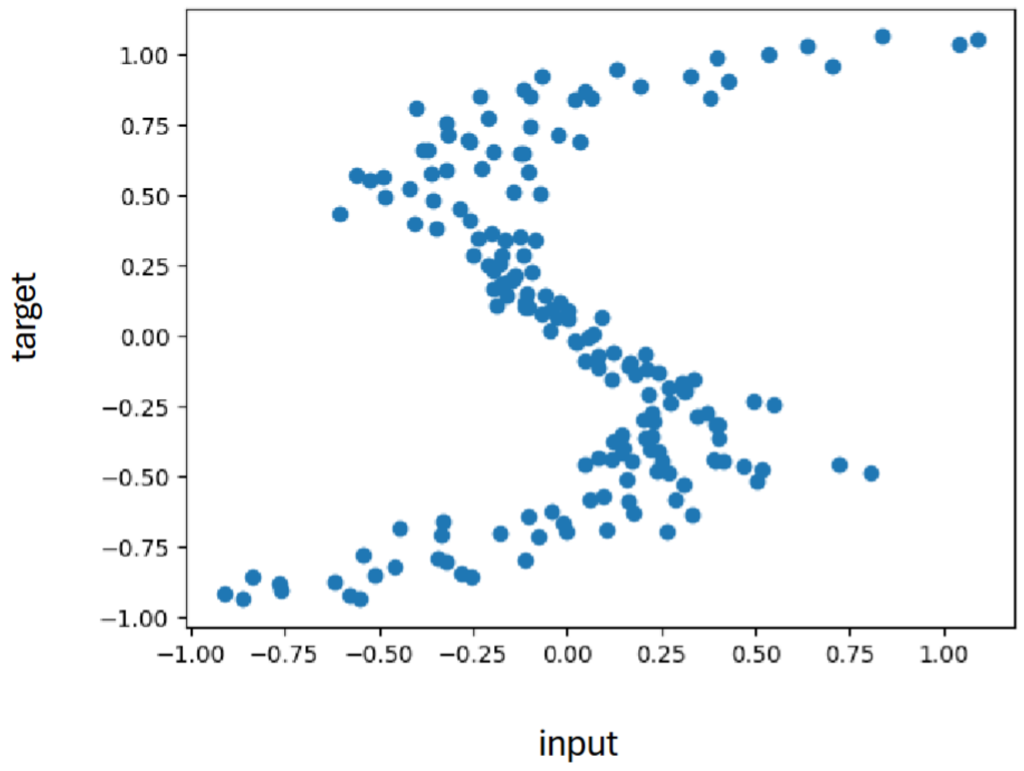
And we **don’t** even need to know what the function looks like (whether it's linear, polynomial, etc.)!

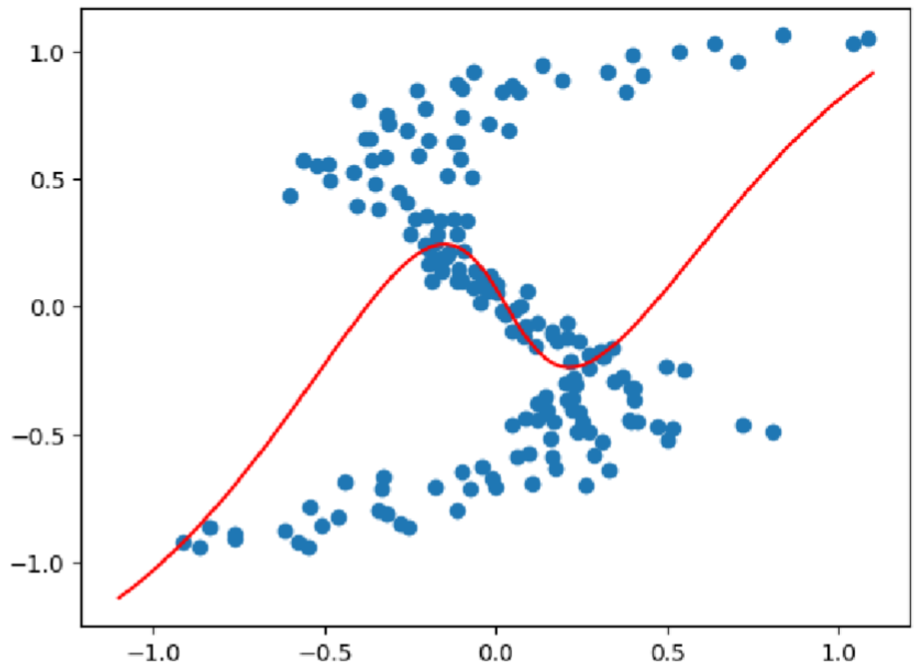




An Example:

```
tfk = tf.keras
forward_model = tfk.Sequential([
  tfk.layers.Dense(4, activation='tanh'),
  tfk.layers.Dense(4, activation='tanh'),
  tfk.layers.Dense(1, activation=None),
])
```



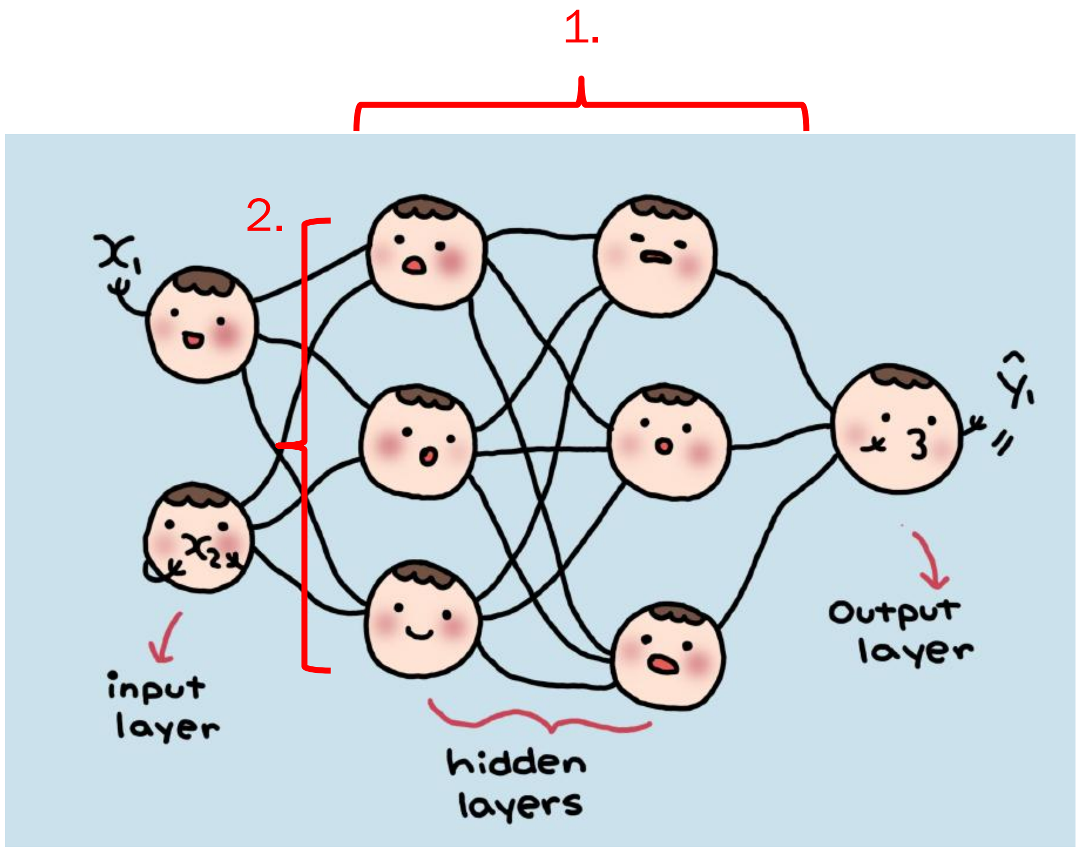


(Fully Connected) Neural Network

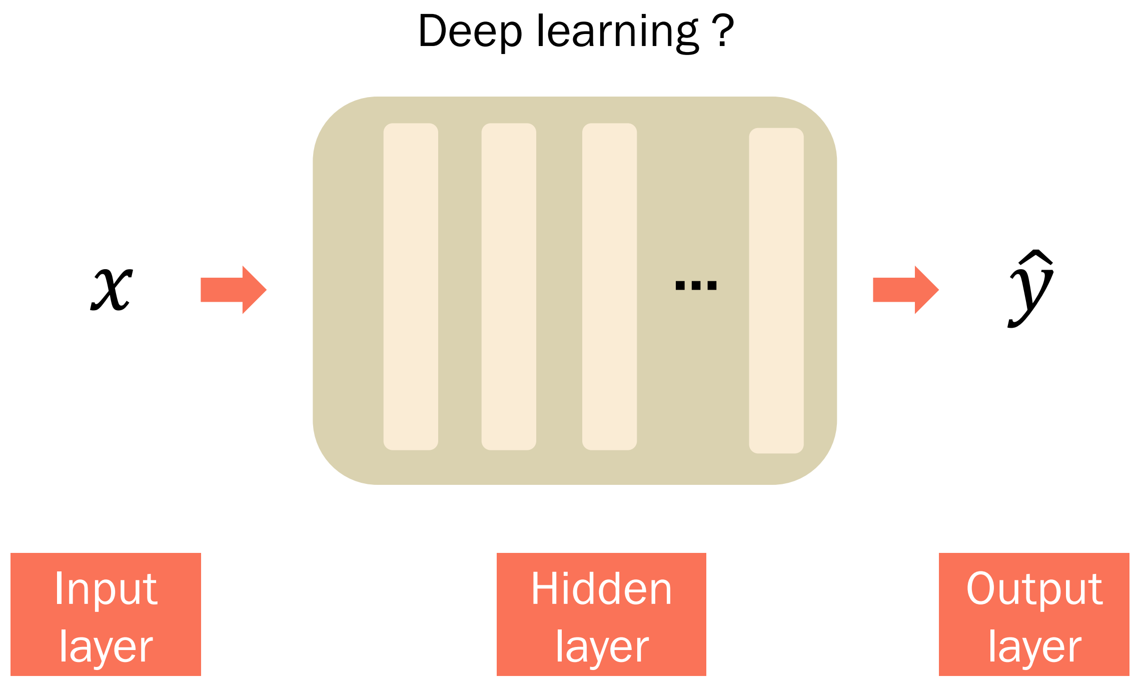
Only need to decide **two sets of numbers**, and the NN is determined!

1. The number of **hidden layers**
2. The number of **neurons each layer**

The neurons between layers are **fully connected.**



NN is all about building layer upon layer of "hidden layers."

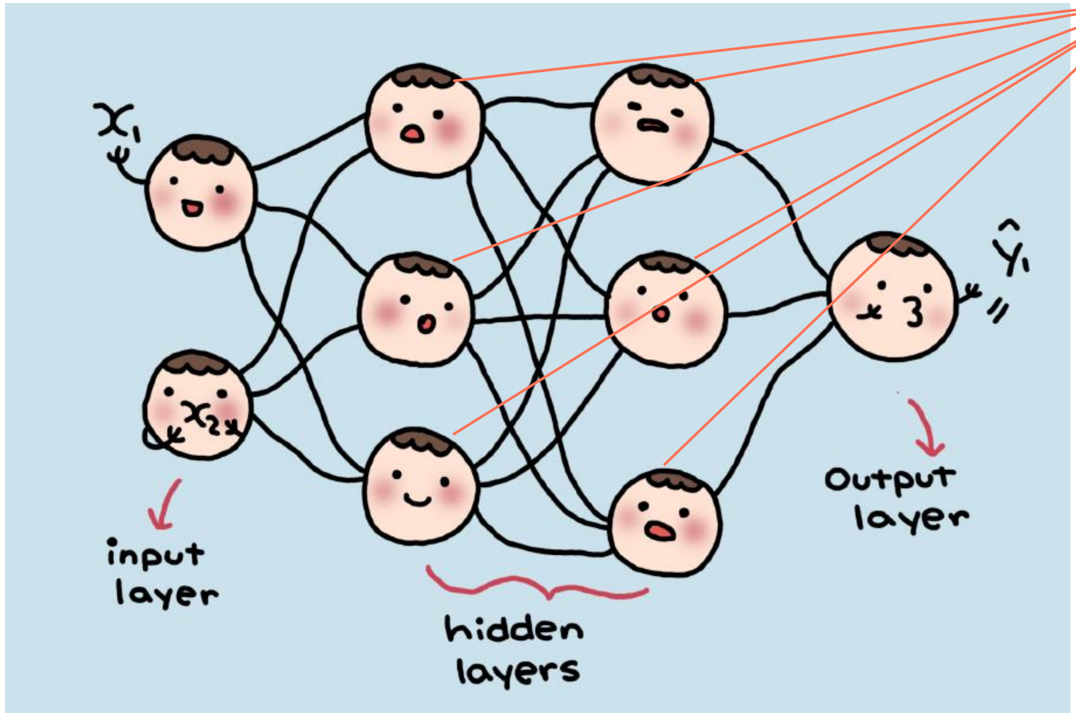


In NN, CNN, and RNN, each layer can be designed using any of these three methods for the hidden layers.

Examples
Basic MLP
LeNet-5 (5 layers), AlexNet (8 layers), VGG-16 (16 layers), ResNet-50 (50 layers)
BERT(12 layer), GPT-3(96 layers)

Exercise 3.

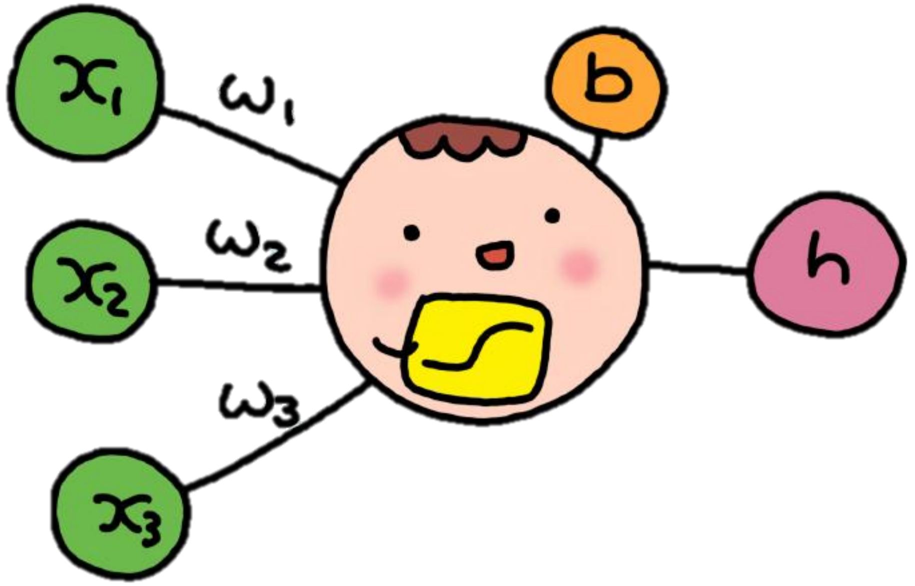
How a Neuron Works?



We are doing the same thing!

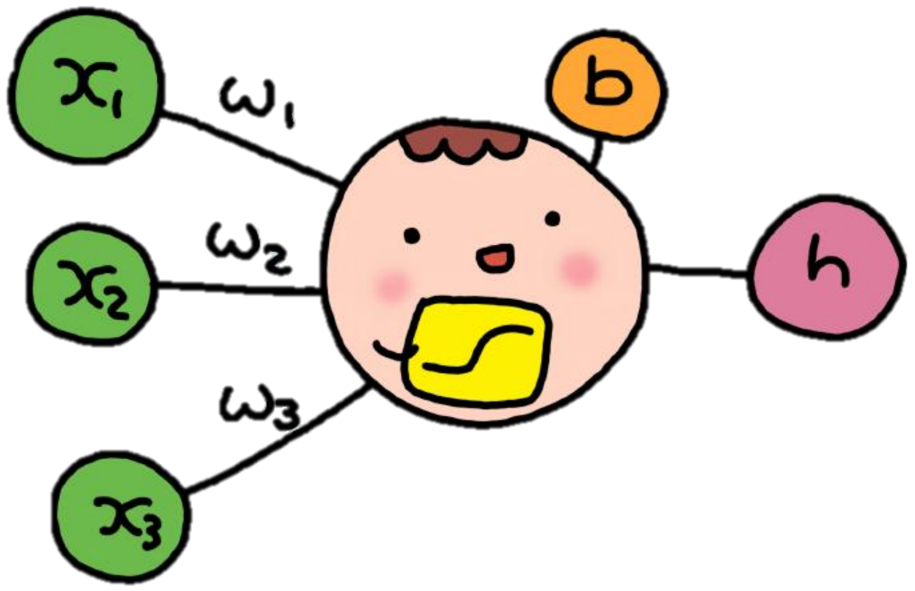
No matter which type of neural network, neurons are the basic computational units, and **each neuron operates in the same way (Only for NN) !**

How a Neuron Works?



Each neuron receives several input signals and then sends out one output signal.

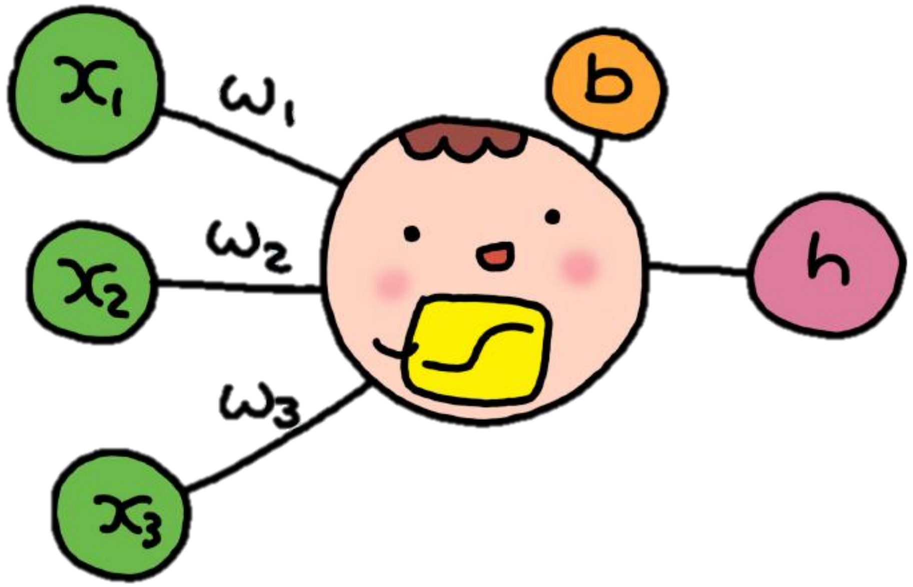
How a Neuron Works?



We first need to calculate the total output that this neuron receives.

$$\sum_{i=1}^{3} x_i w_i$$

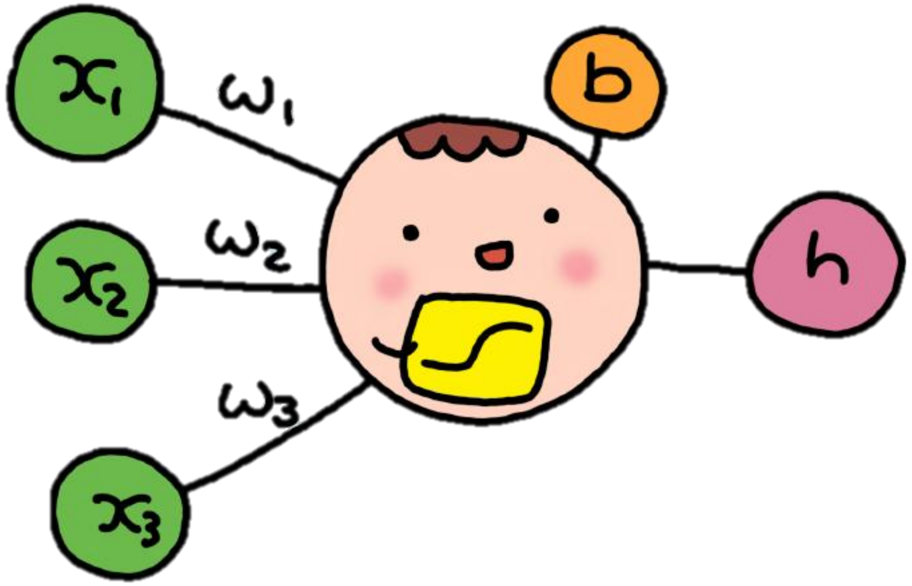
How a Neuron Works?



Next, we add the bias to make a baseline adjustment. Both the weights and the bias are learned through training! Let's call this the adjusted total output for now.

$$\sum_{i=1}^{3} x_i w_i + b$$

How a Neuron Works?

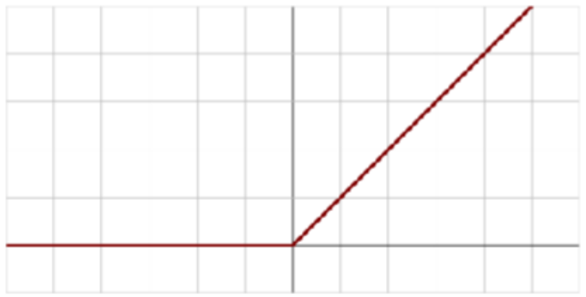


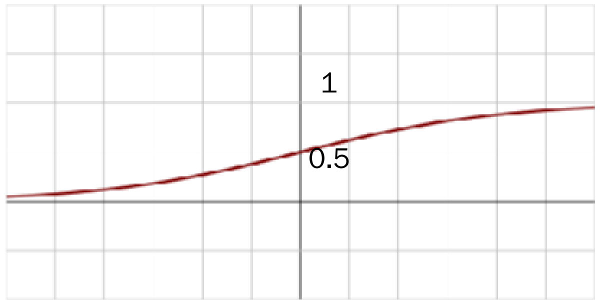
The current calculations are all linear, and even after passing through all the neurons, they remain linear. Therefore, we need to apply an activation function to transform the output before sending it out.

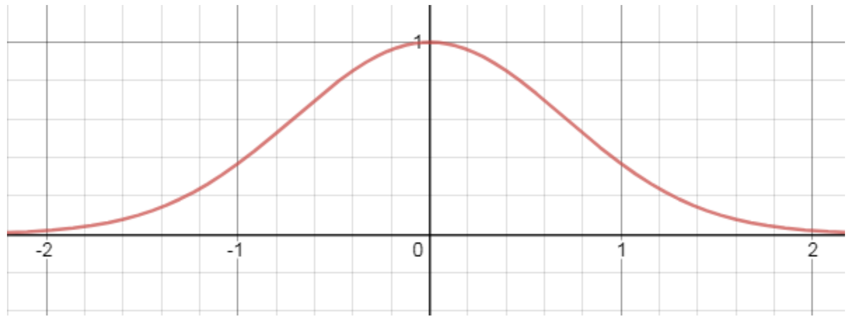
$$\varphi\left(\sum_{i=1}^{3} x_i w_i + b\right) = h$$

Activation function

Several well-known activation functions







Rectified linear unit:

$$\mathrm{ReLU}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Sigmoid:

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

Gaussian

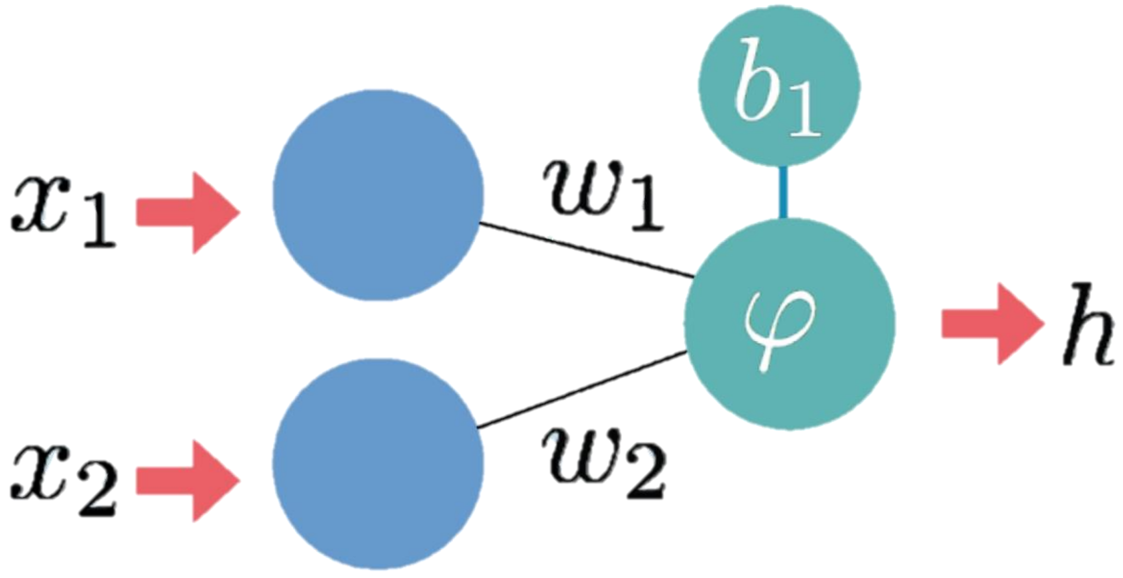
The new favorite of this century !

It seems very close to the behavior of human neurons.

Rarely used now

Exercise 4.

For an example.



Both the **weights** and **biases** are learned, but initially, we assign them random values.

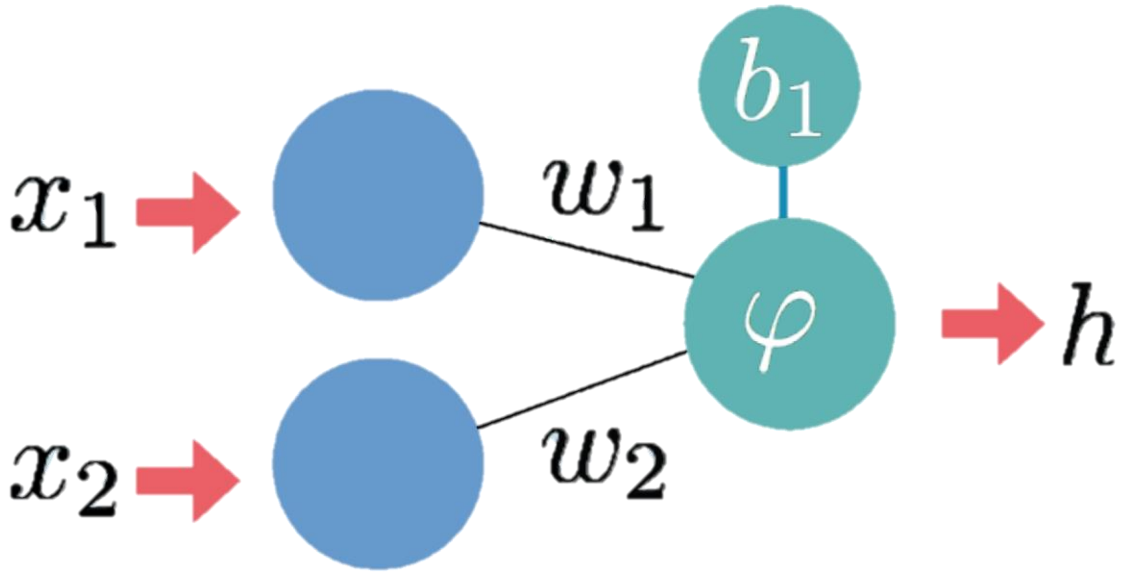
Assume the following initialization results:

$w_1 = 1$

$w_2 = 3$

$b_1 = 1$

For an example (Quiz)



Assume the following initialization results:

$w_1 = 1$

$w_2 = 3$

$b_1 = 1$

$$\boldsymbol{x} = (x_1, x_2) = (1,3)$$

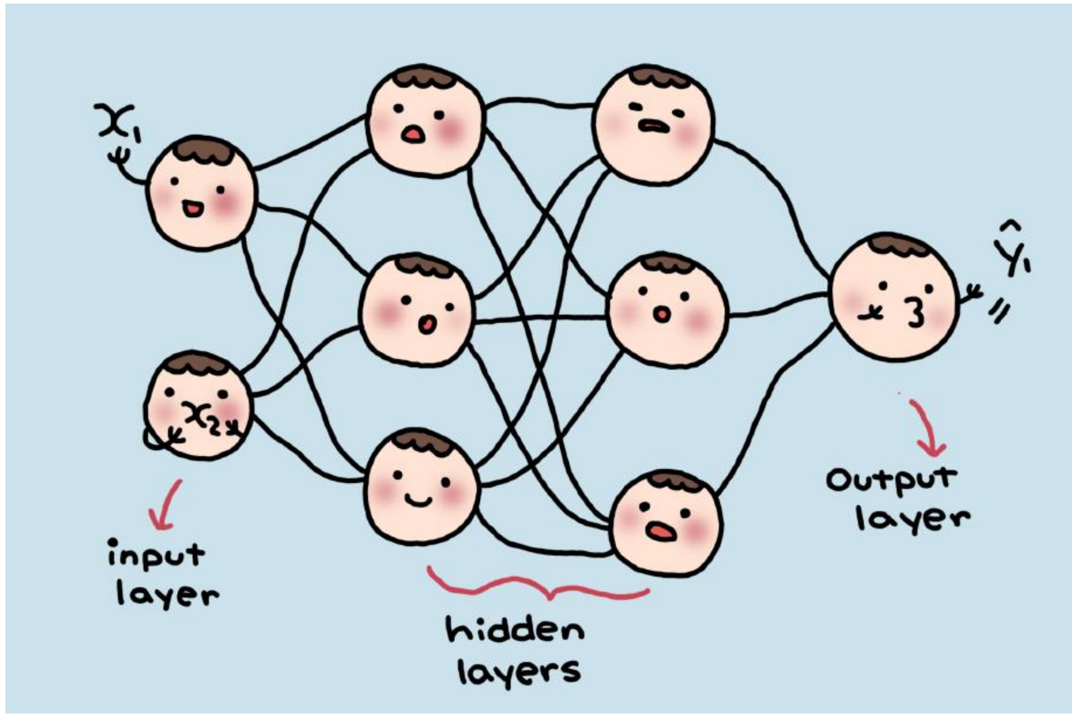
Rectified linear unit:

$$\mathrm{ReLU}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$\boldsymbol{\varphi(x_1 w_1 + x_2 w_2 + b_1) = h}$$

Caluculate the neural output ?

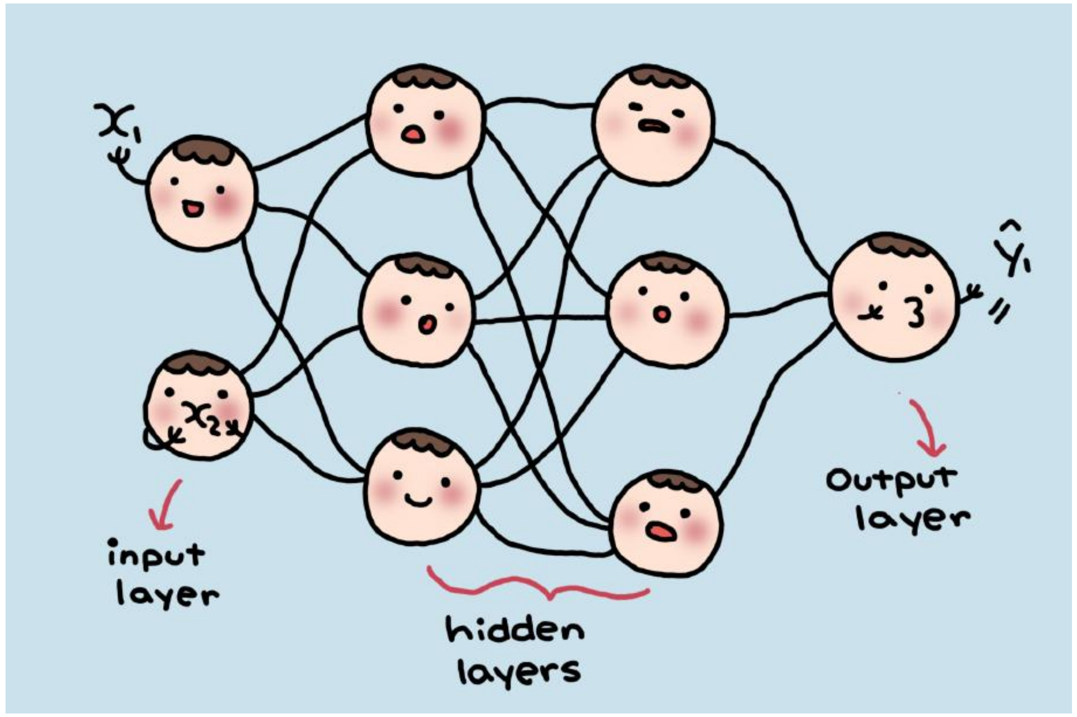
Function learning machine built!

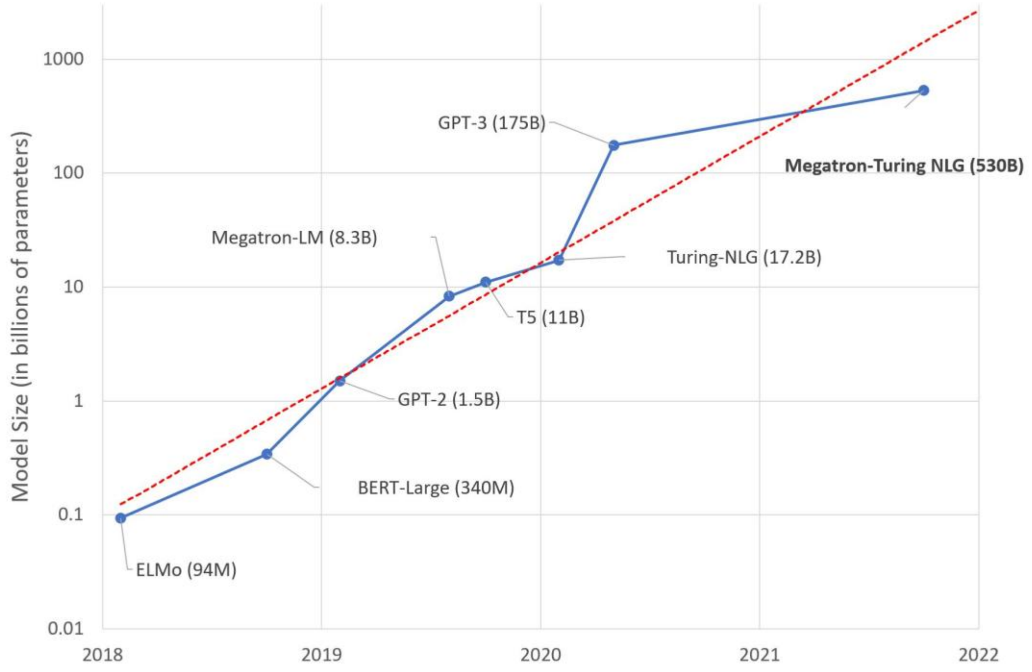


We will combine all the parameters to be learned, including all the weights and biases, and call them:

$$\theta$$

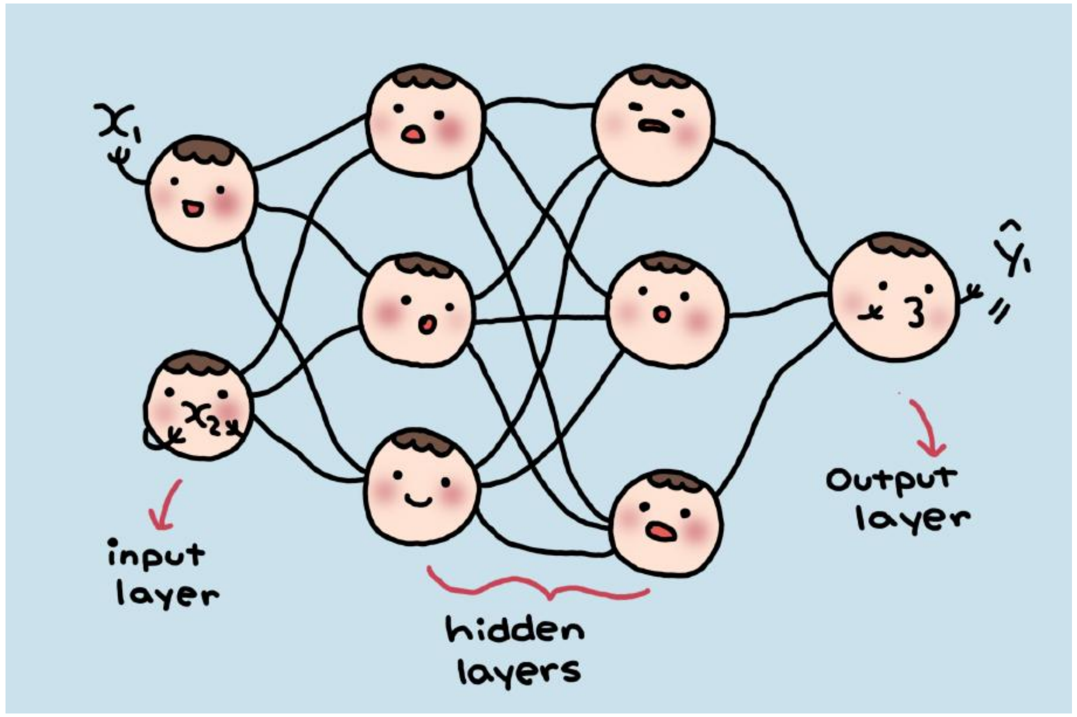
Function learning machine built!



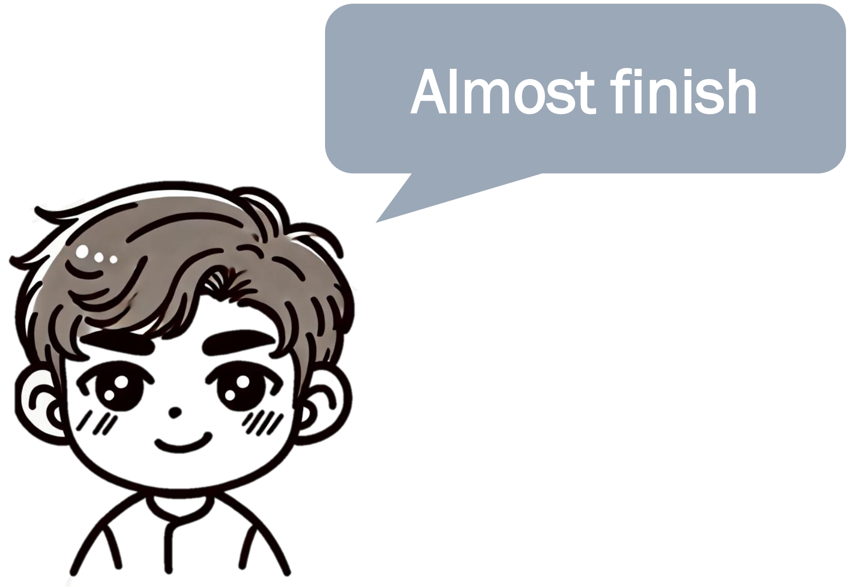


How many parameters in this neural network?

Function learning machine built!



Once parameters are determined, meaning a set of values is established, we can input any values, and the neural network will give us an output!



How does a neural network learn?

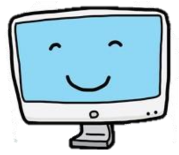
Training Our Neural Network

Neural networks need to be trained! The method is to repeatedly present our training data to the neural network for learning.
This learning process is called backpropagation.

Training Our Neural Network - Function space

After building a function learning machine with a neural network, **a set of parameters (weights and biases)** defines a specific function. Collecting all possible functions creates a function space.

$\boxed{f_\theta}$

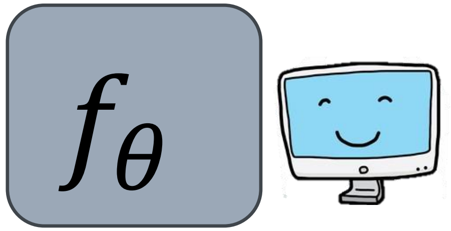


$$\{f_\theta\}$$

Training Our Neural Network - Function space

$$\theta^*$$

Our goal is to select the best set θ^*, meaning the one that makes the resulting function f_{θ^*} as close as possible to the target (goal) function.



Training Our Neural Network - Loss Function

We use a loss function to calculate how far off our neural network's output is from the correct answer in the training data.

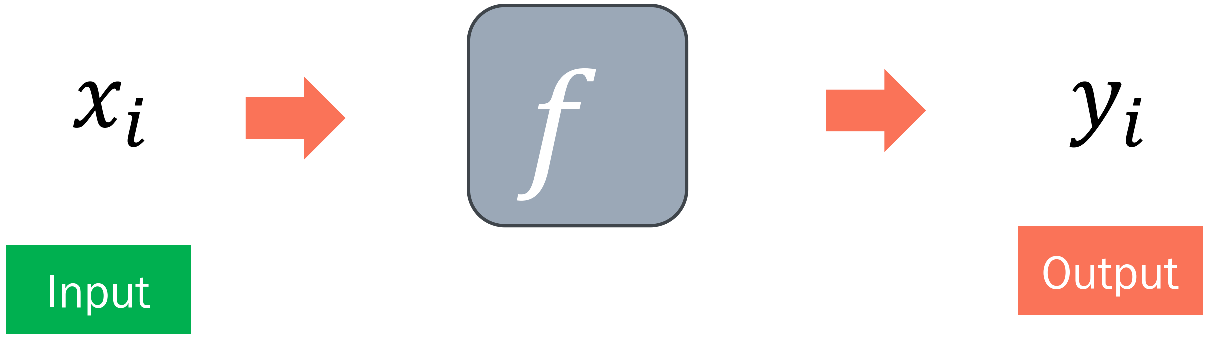
Naturally, we want the value of the loss function to be as small as possible.

Loss Function

Assume we have a training data

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$$

It means that when input is x_i, the correct answer should be y_i .



Loss Function

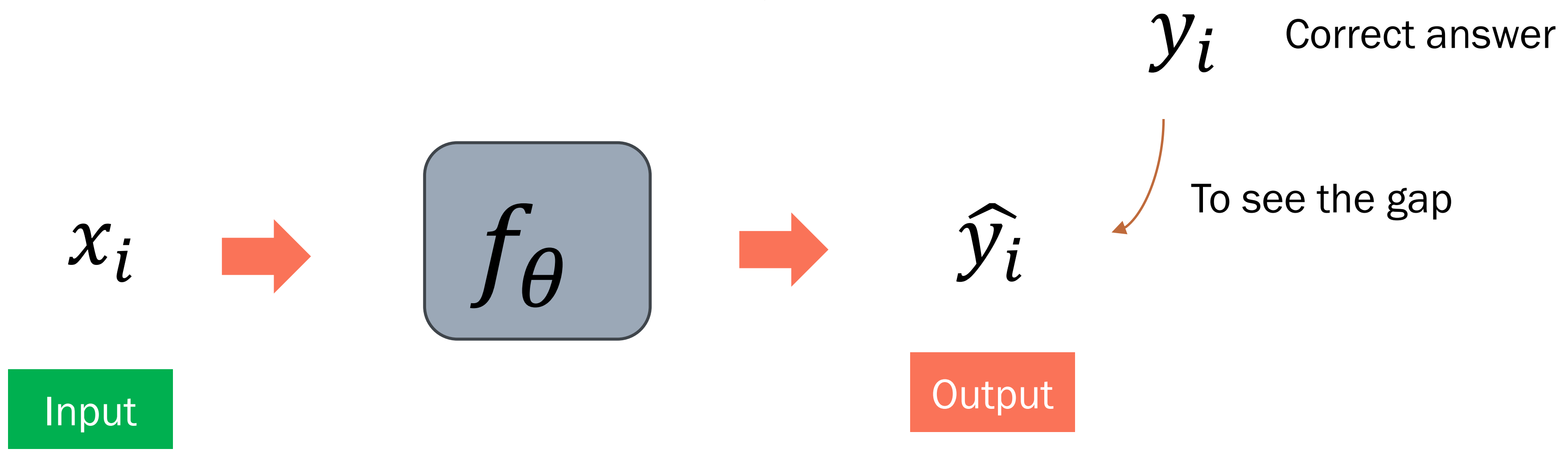
Any set of parameters θ will define a function f_θ. This function f_θ will give a value $\hat{y}_i$ for any given input x_i.

$$\begin{tikzcd}
x_i \arrow[r] & \boxed{f_\theta} \arrow[r] & \hat{y}_i \\
\text{Input} & & \text{Output}
\end{tikzcd}$$

$$\boxed{f_\theta(x_i) = \hat{y}_i}$$

Loss Function

Of course, we hope that the $\hat{y}_i$ given by the neural network is as close as possible to the correct answer y_i !



$$f_\theta(x_i) = \hat{y}_i$$

Loss Function

The loss function is a function used to calculate **how far** of the neural network's answer is from the correct answer.
For example, here is a common loss function:

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{k} \|y_i - f_\theta(x_i)\|_2$$

Loss Function

It is simply calculating the difference from the correct answer!

The answer given by the function learning machine

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{k} \| y_i - f_\theta(x_i) \|_2$$

Correct answer

Another commonly used one is Cross-Entropy Loss: **Binary:** $L = -[y log(\hat{y}) + (1-y)\log(1-\hat{y})]$

Parameter Adjustment

How is it adjusted? For a given parameter, it's actually done using this formula:

$$-\eta \frac{\partial L}{\partial w}$$

Parameter Adjustment

Remember, the loss function is a function of all weights, biases such like $\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{b_1}$ and other parameters.

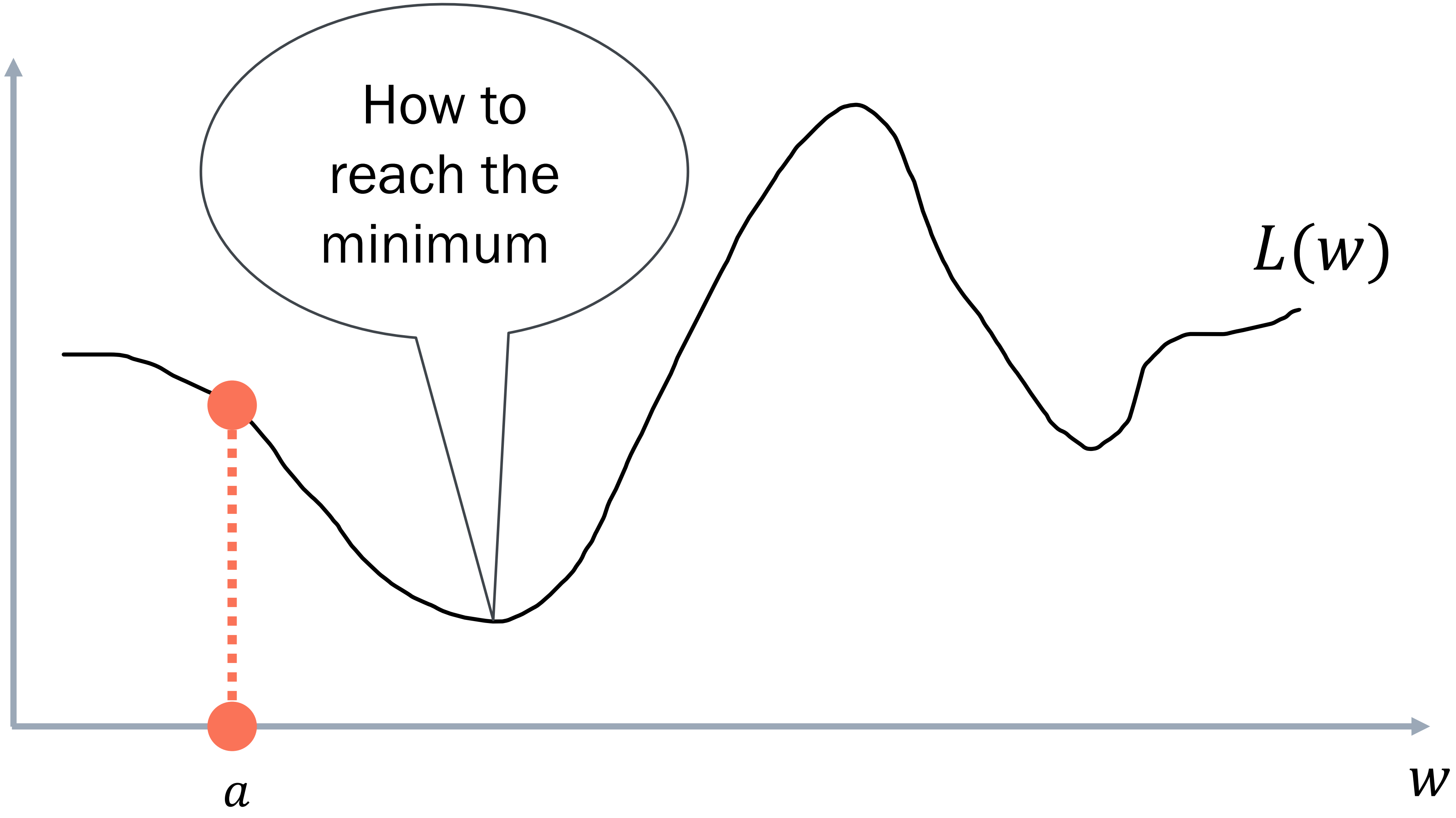
Assume there's only one parameter!

Mathematicians always simplify the problem first.

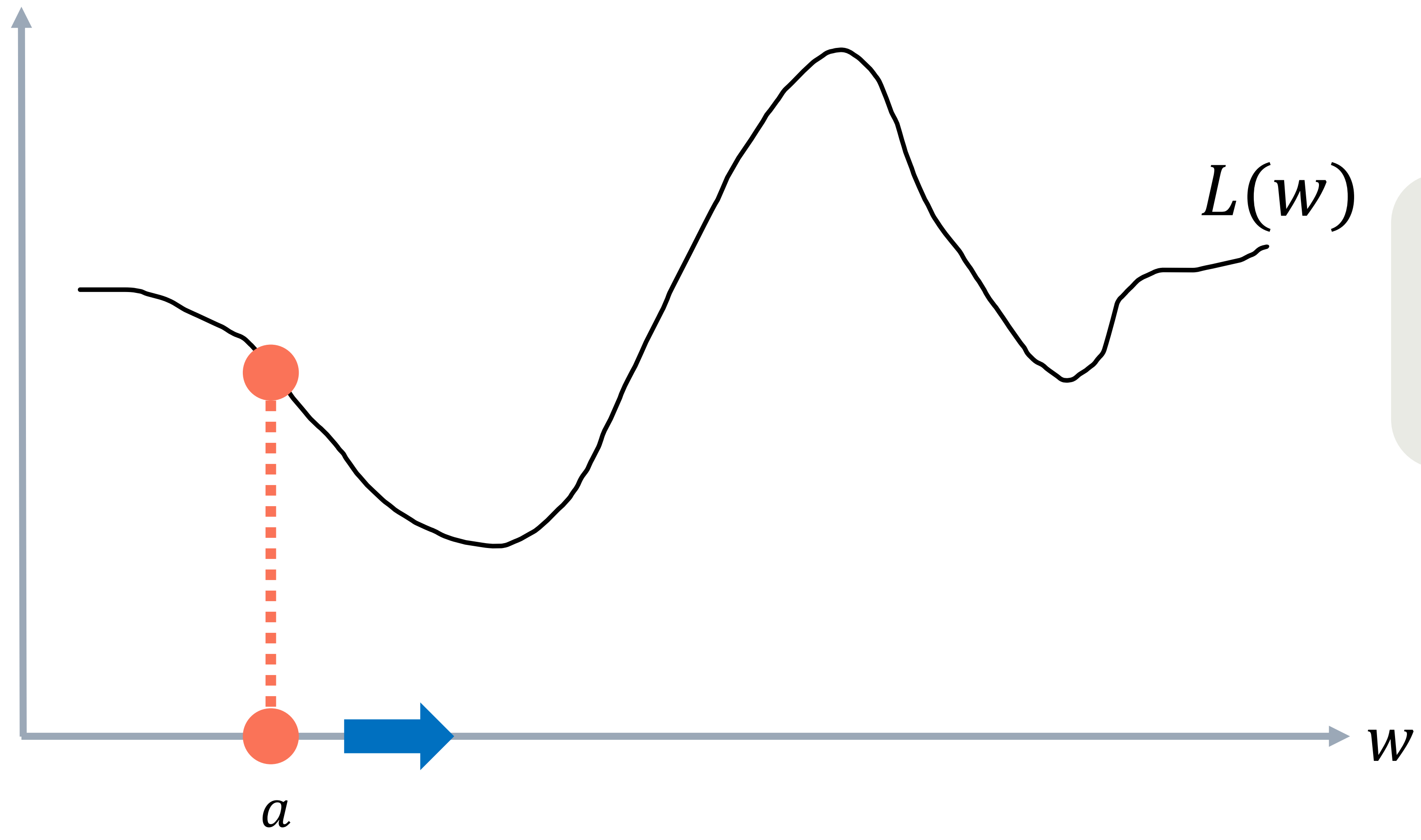
Let's assume the function learning machine we built with deep learning has only one parameter w

Is it okay?

Assume there's only one parameter!



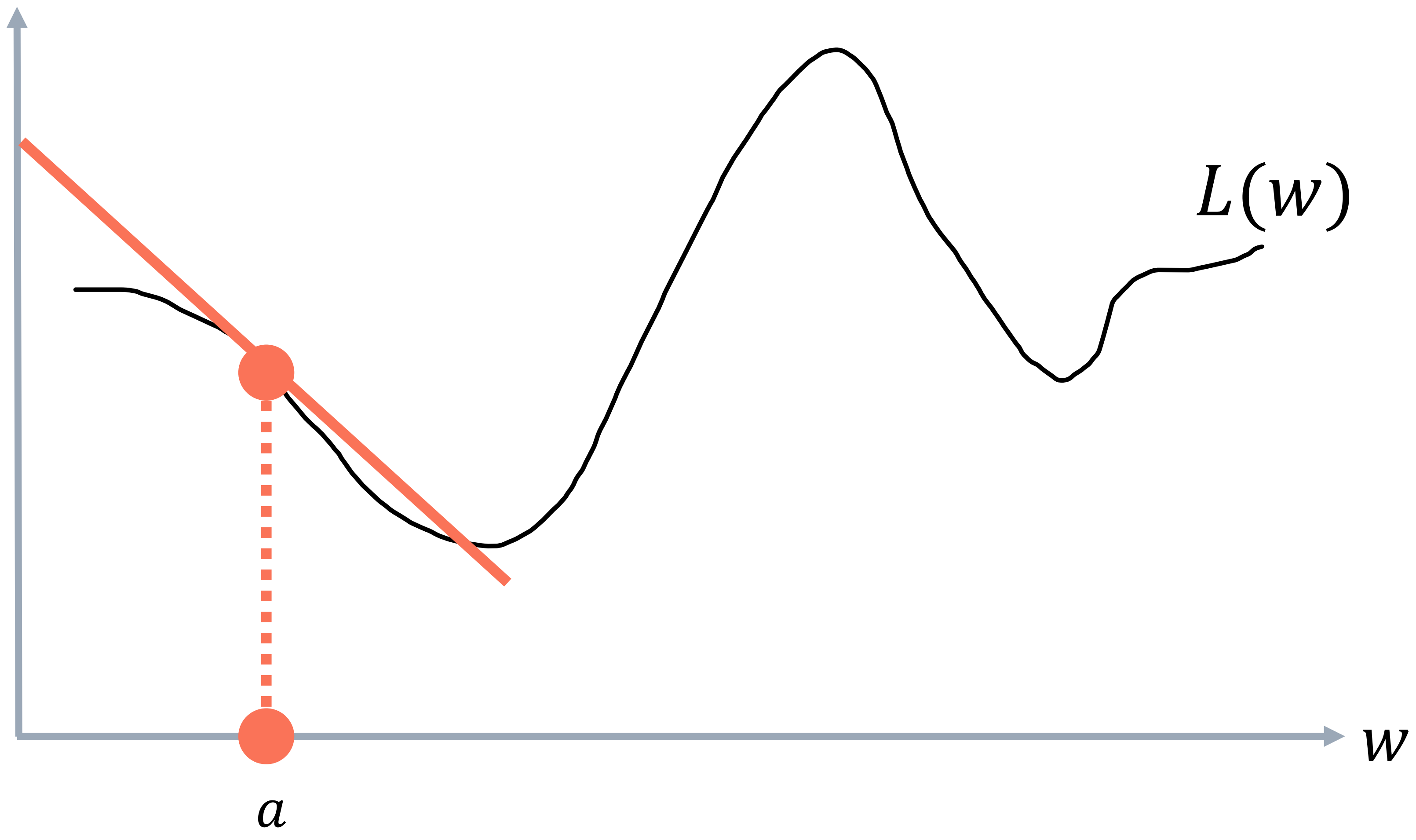
Assume there's only one parameter!



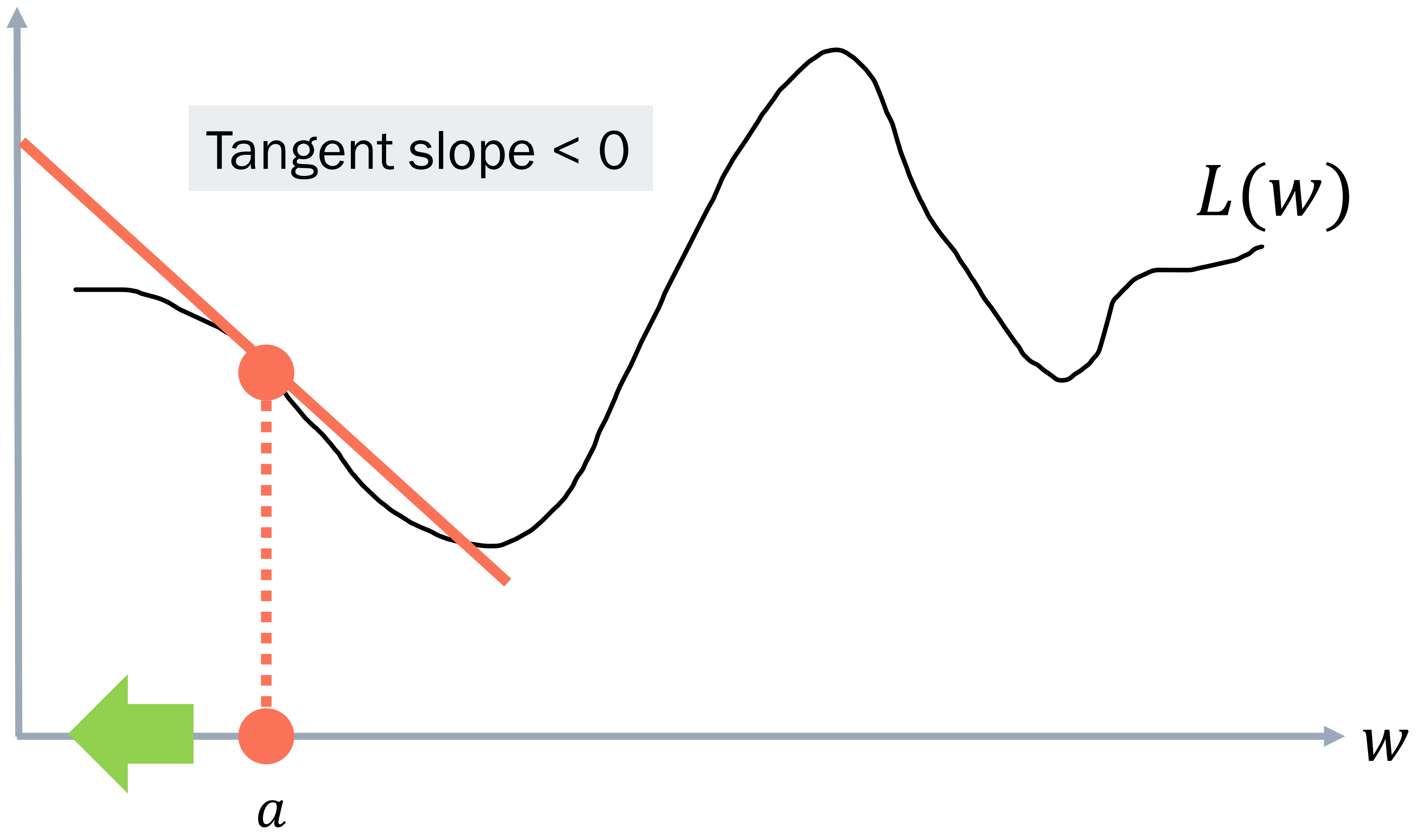
I know, the point $w = a$ needs to move to the right!

How does the computer know which direction to go?

How does the computer know which direction to go?

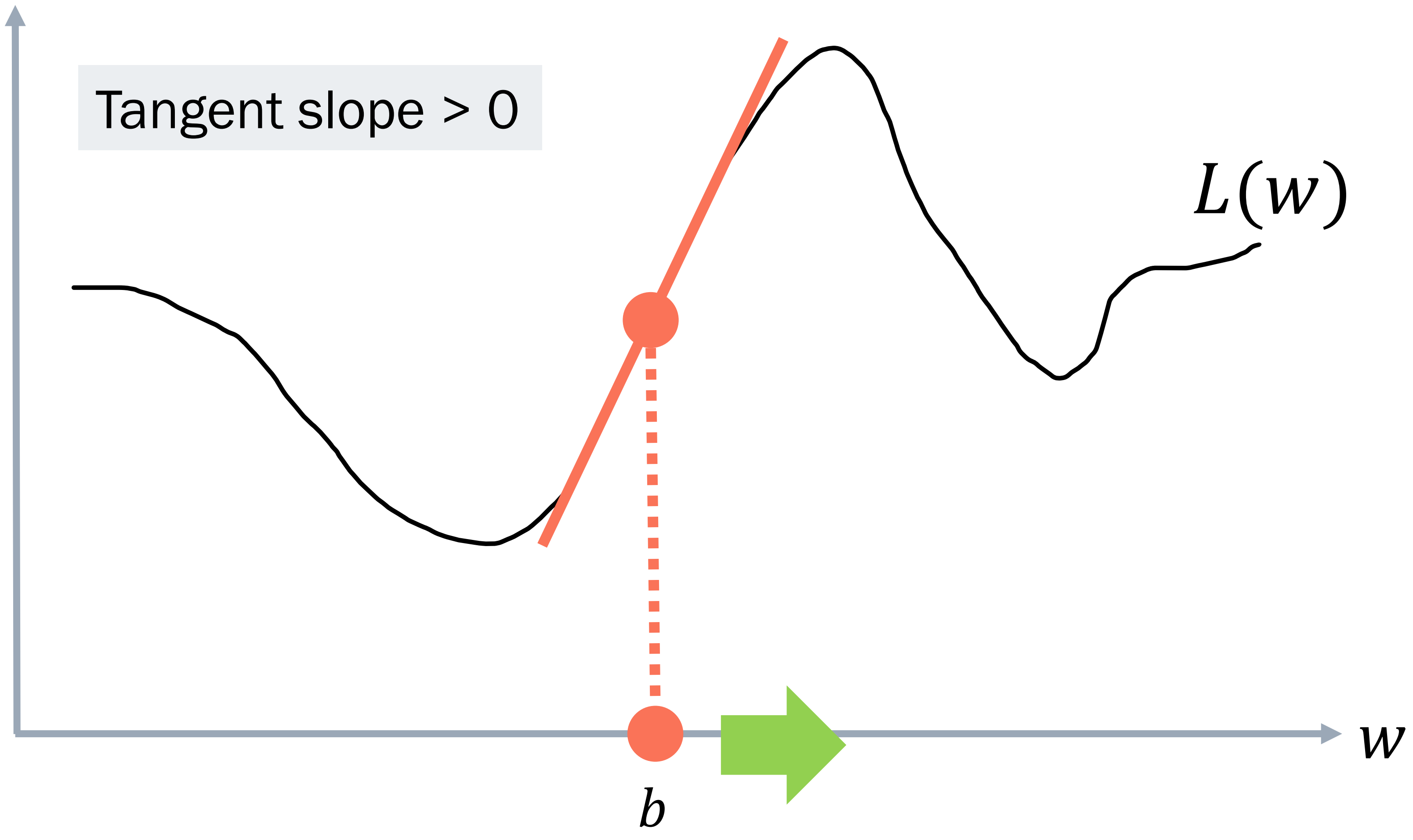


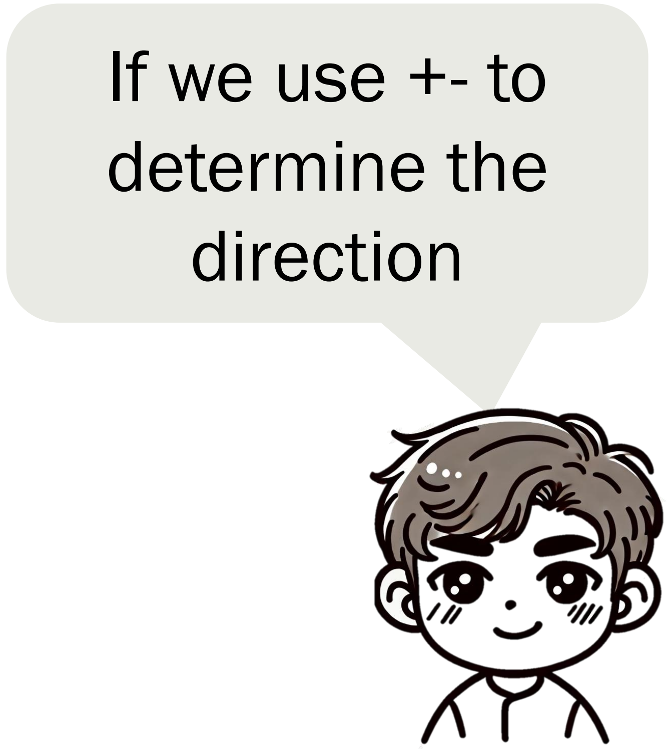
How does the computer know which direction to go?



If we use +- to determine the direction

How does the computer know which direction to go?





How does the computer know which direction to go?

The tangent slope points in the opposite direction of the minimum!

In fact, it points towards the (local) maximum!

The fancy symbol for the tangent slope

The symbol for the tangent slope at the point $w = a$ is like this:

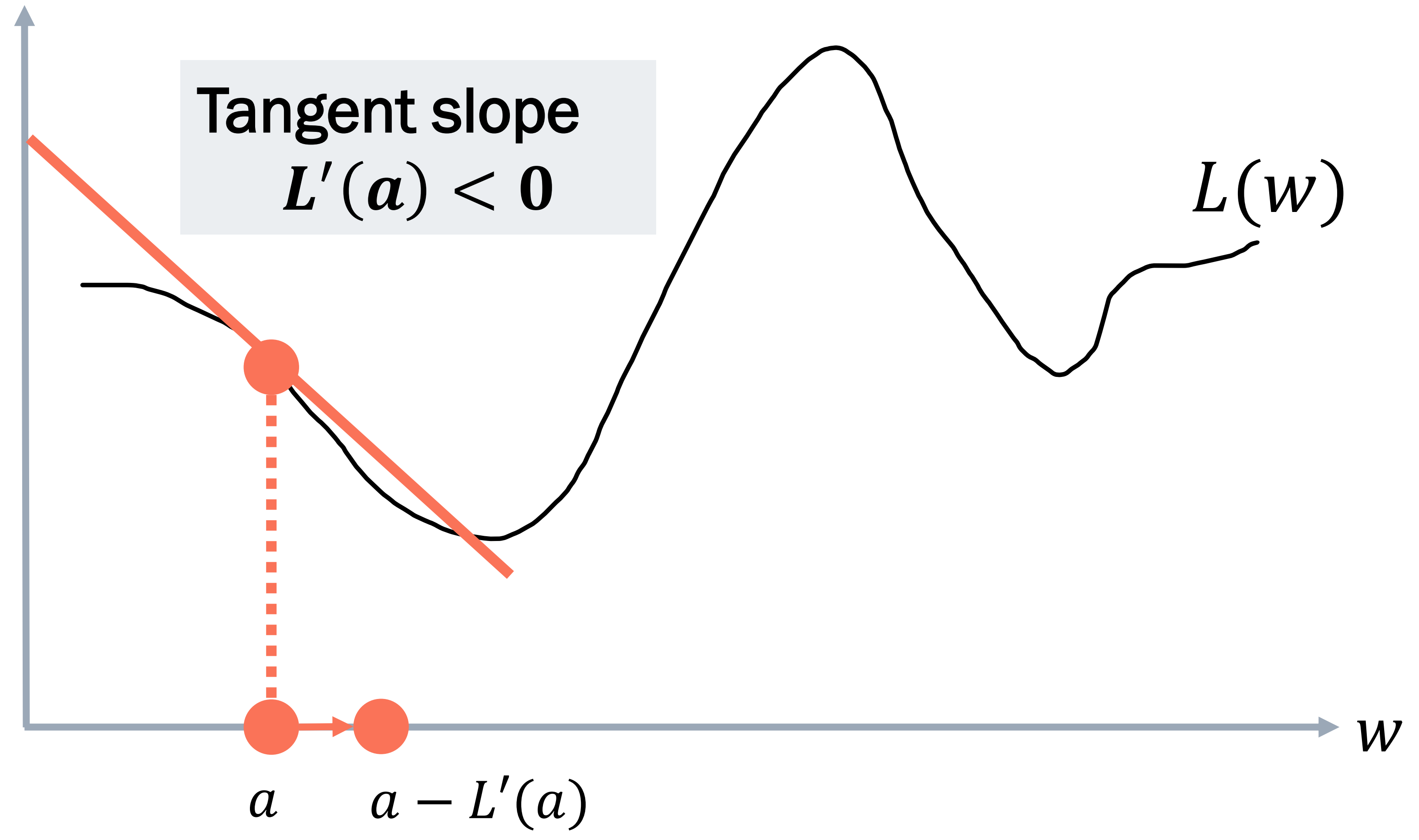
$$L'(a)$$

For any point, we write it in function form like this:

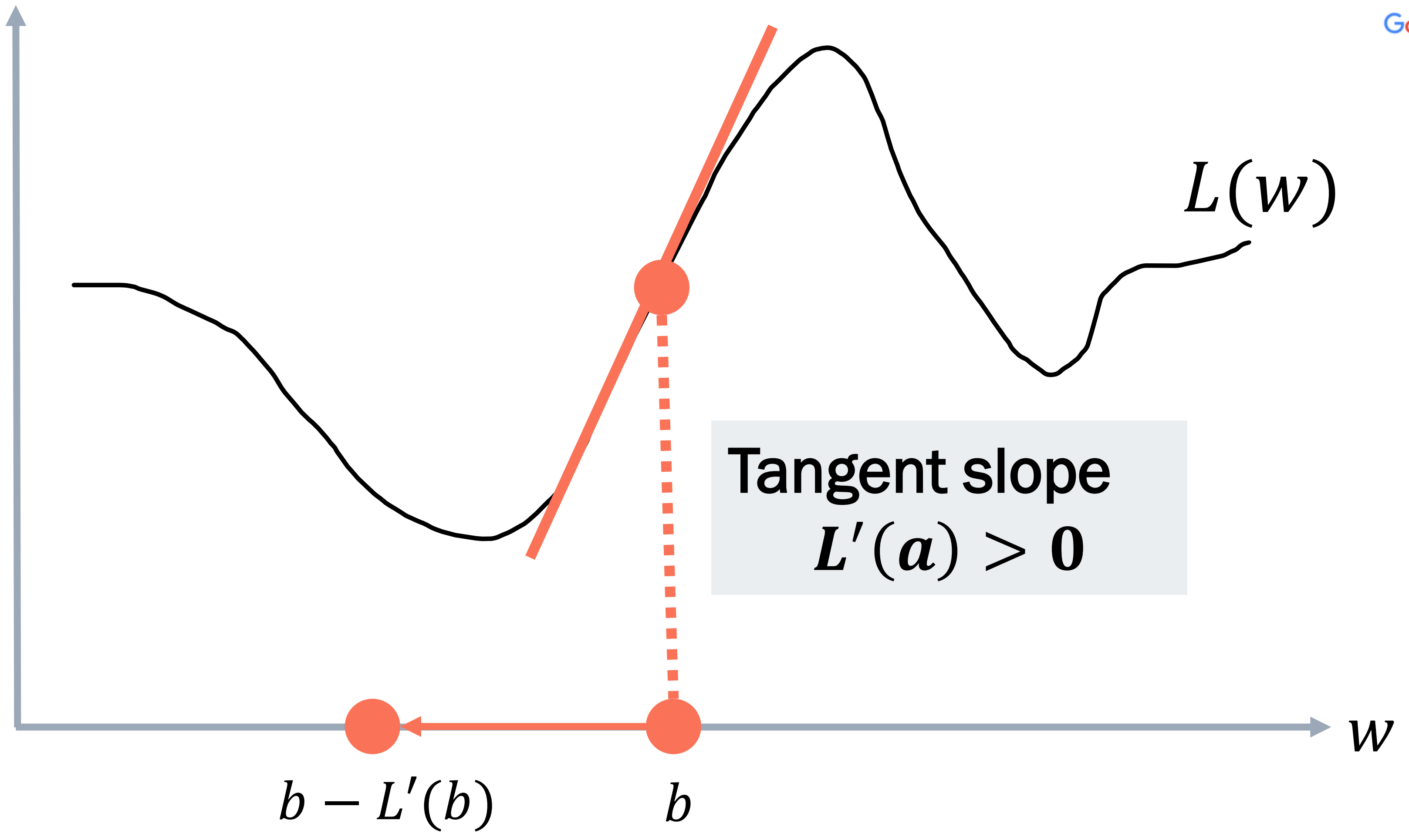
$$L'(w) = \frac{\mathrm{d}L}{\mathrm{d}w}$$

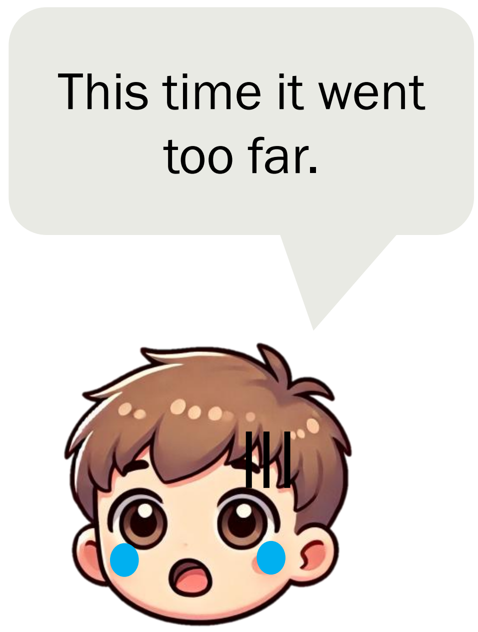
We're really good at calculating these in calculus

Move towards the (local) minimum!



Move towards the (local) minimum!





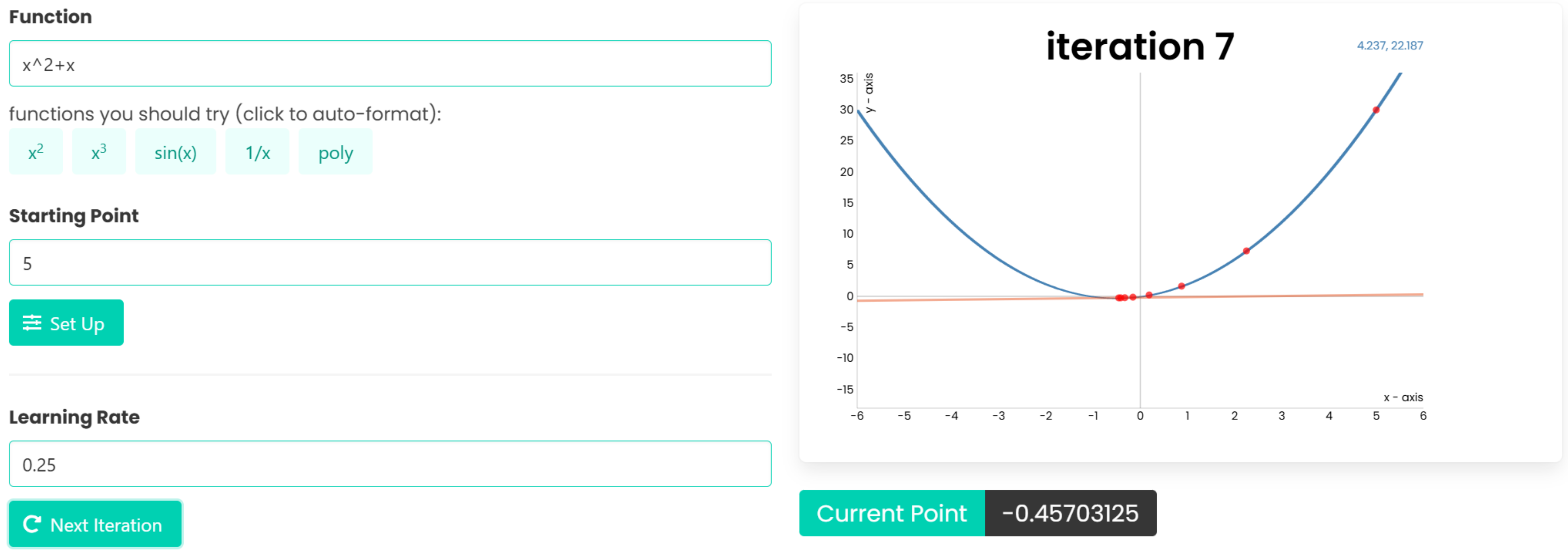
Learning Rate

$$W - \eta \frac{\mathrm{d}L}{\mathrm{d}w}$$

$$0 < \eta < 1$$

To avoid going too far, we won't make big adjustments at once. We'll multiply by a small number called the Learning Rate.

Let’s play in this playground (5 mins)



Function

x^2+x

functions you should try (click to auto-format):

Starting Point

5

Learning Rate

0.25

Current Point -0.45703125

https://uclaacm.github.io/gradient-descent-visualiser/#playground

Optimizer is a total solution for gradient decent process.

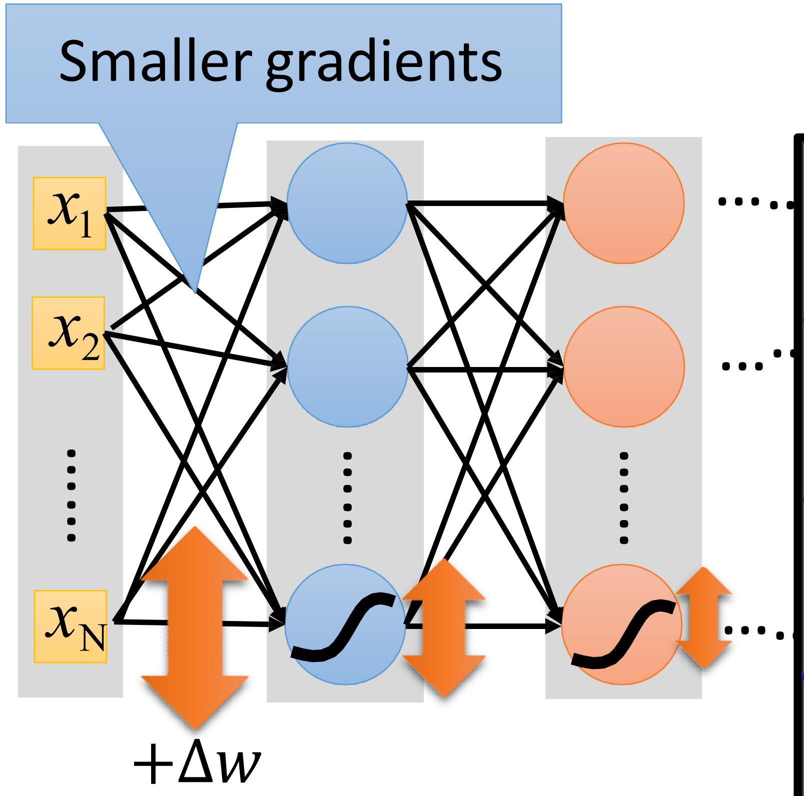
Adjusting learning rate is one of the important parts in an optimizer

```
optimizer = torch.optim.AdamW(model.parameters(), lr=3e-4, weight_decay=0.01)
```

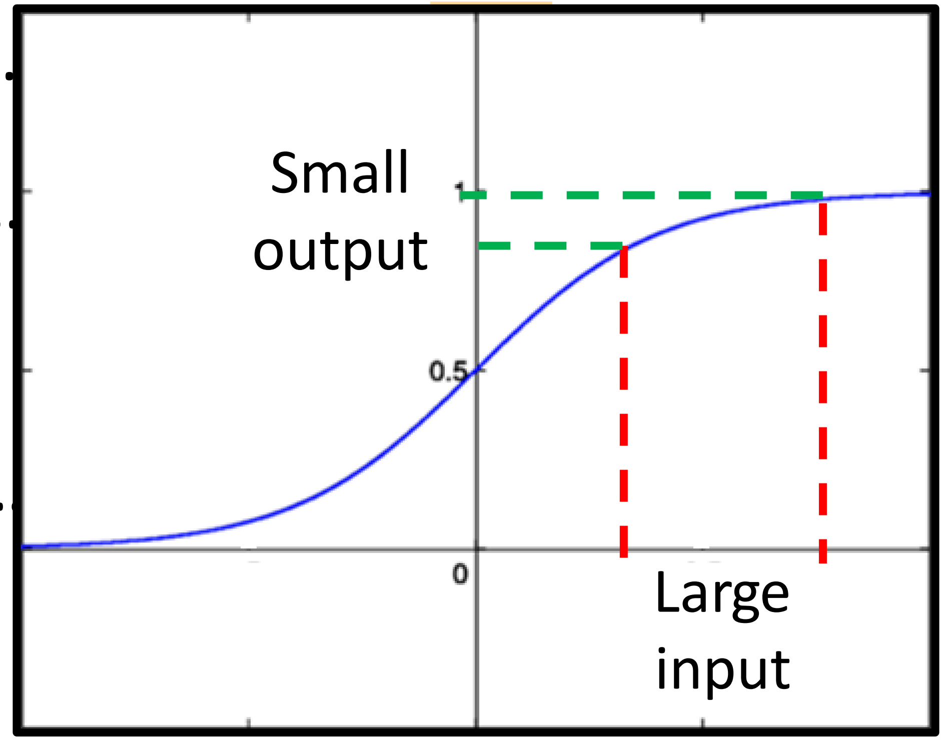
```
optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
```

Optimizer	Full Name	Key Characteristics
SGD, 1974	Stochastic Gradient Descent	Updates model parameters using one (or a small batch of) sample(s) at a time. Simple, stable, and offers good generalization but may converge slowly.
Adam, 2014	Adaptive Moment Estimation	Combines momentum and adaptive learning rates; faster convergence and widely used in deep learning models such as CNNs and Transformers.
Lion, 2023		
Sofia, 2023		

Back to activation function - Gradient Vanishing



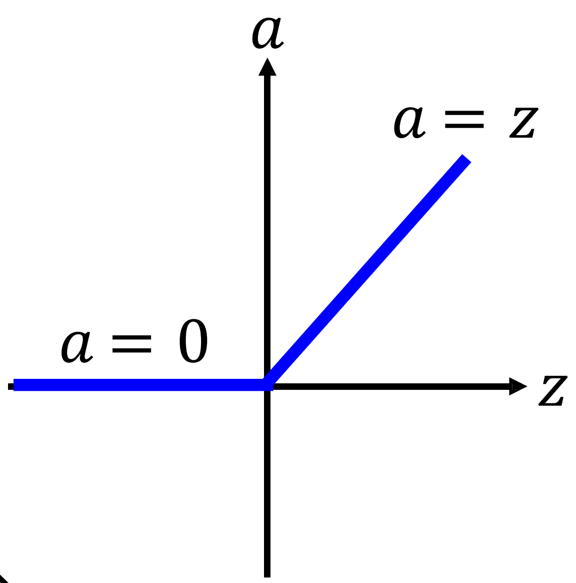
More intuitive view…

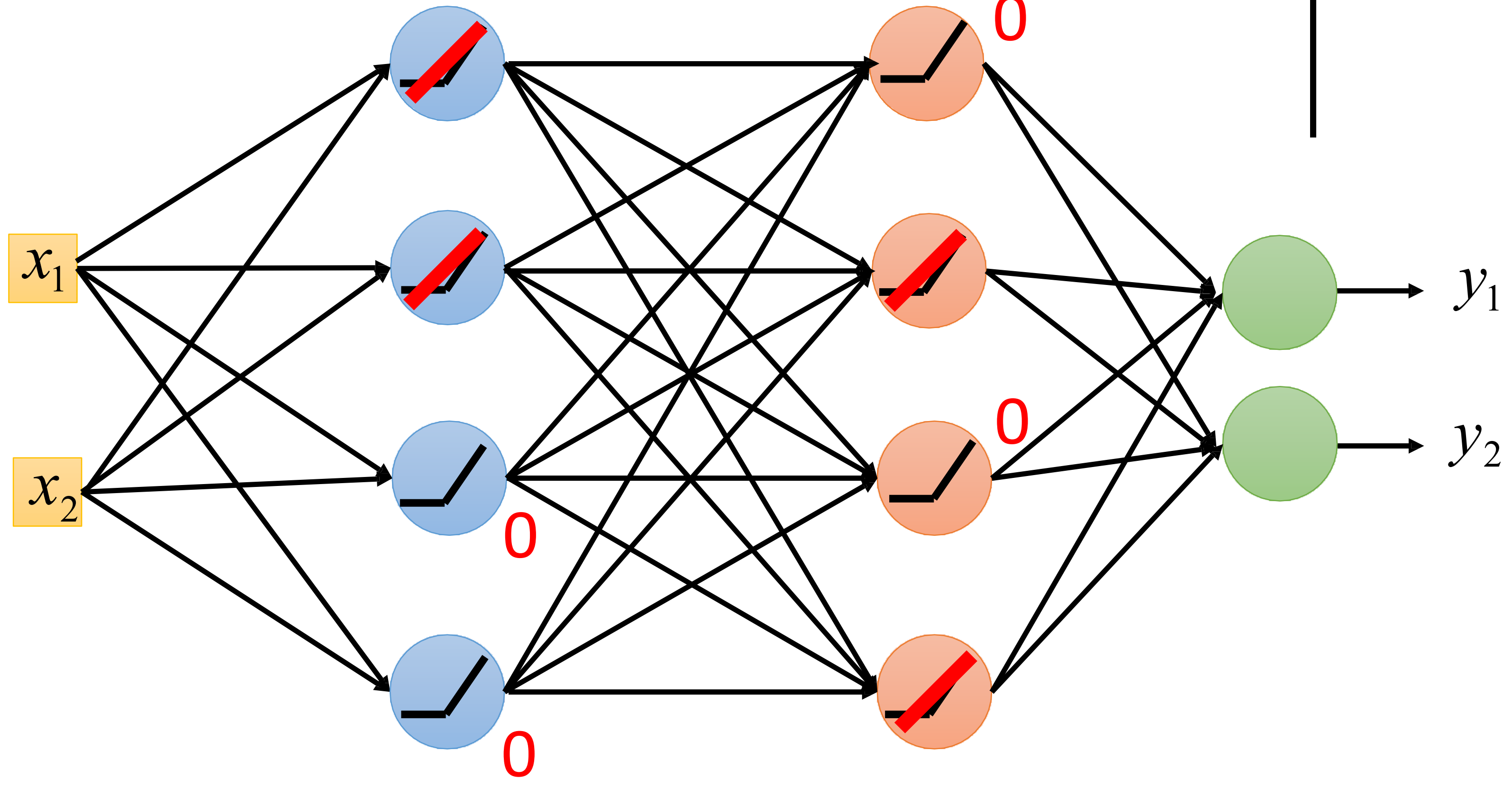


$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

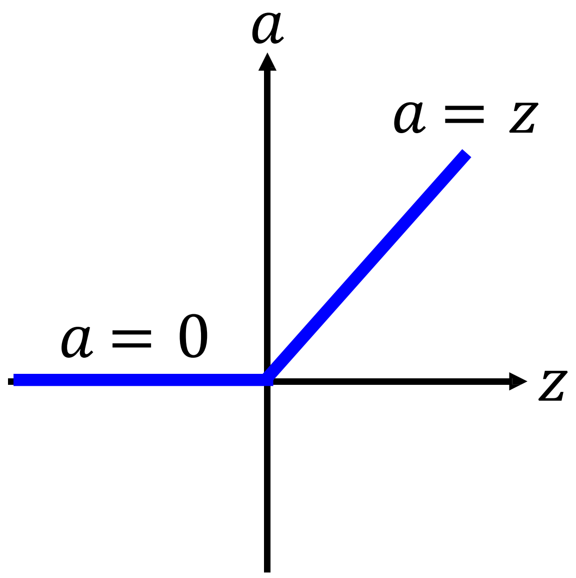
Back to activation function

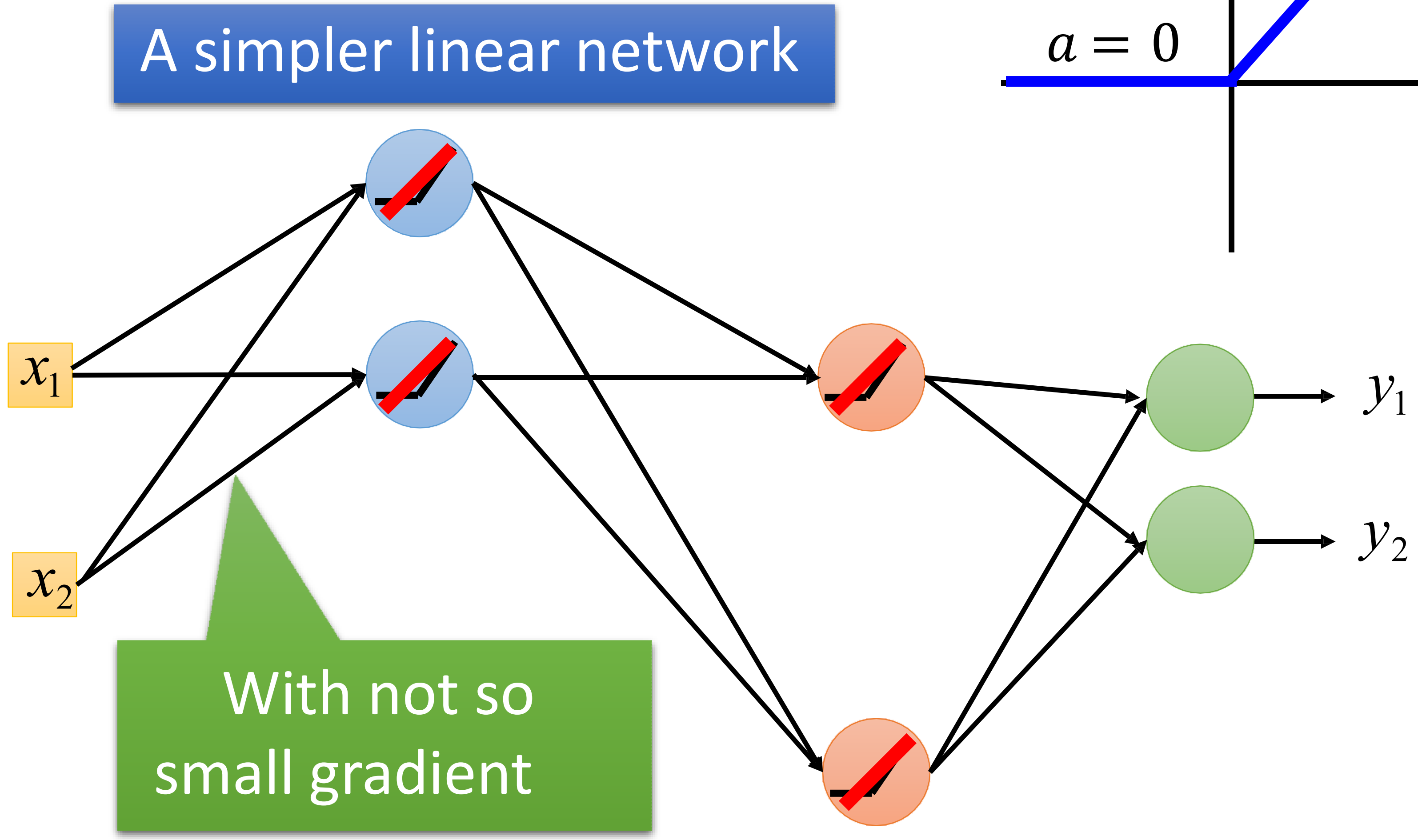
Fast!!!!





Back to activation function





What if there is more than one parameter?

Mathematicians are……..

Let's just pretend there's only one parameter w_1

Do not ask me why…

Assume there's only one parameter!

E.g.

$$L(w_1, w_2, b_1) = (b + 2w_1 - w_2 - 3)^2$$

$$w_1 = 1, w_2 = -1, b_1 = 2$$

$$L_{w_1}(\omega_1) = L(w_1, -1, 2) = 4w_1^2$$

In L, we only keep the variable w_1 , and directly substitute all the other values!

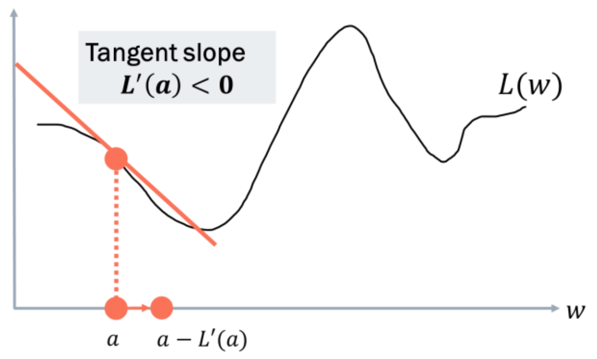
So, at that moment, we're left with just one parameter!

Assume there's only one parameter!

New

Previous

$$w_1 \leftarrow w_1 - \eta \frac{dLw_1}{dw_1}$$



Then we can adjust our weight using a single-variable adjustment method!

Assume there's only one parameter!

$$\frac{\partial L}{\partial w_1} = \frac{dL\omega_1}{dw_1}$$

It means that we will adjust w_1 as:

$$w_1 - \eta \frac{\partial L}{\partial w_1}$$

In multivariable functions, pretending there's only one variable for differentiation is called partial differentiation.

It's really just adjusting one by one

Sensitivity

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

$$w_2 \leftarrow w_2 - \eta \frac{\partial L}{\partial w_2}$$

$$b_1 \leftarrow b_1 - \eta \frac{\partial L}{\partial b_1}$$

It is the same method as with a single variable.

It just with more difficult-looking notation

We write the three equations together

$$\begin{array}{ccc} \text{New} & & \text{Previous} \\ \begin{bmatrix} w_1 \\ w_2 \\ b_1 \end{bmatrix} & \longleftarrow & \begin{bmatrix} w_1 \\ w_2 \\ b_1 \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \frac{\partial L}{\partial b_1} \end{bmatrix} \end{array}$$

Gradient

$$\begin{bmatrix} w_1 \\ w_2 \\ b_1 \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \frac{\partial L}{\partial b_1} \end{bmatrix}$$

This key vector is called the **gradient**, and it is denoted as:

$$\nabla L$$

Gradient descent

$$\begin{bmatrix} w_1 \\ w_2 \\ b_1 \end{bmatrix} - \eta \nabla L$$

The equation for adjusting the parameters can be written like this, and we call this method:

$Gradient\ Descent$

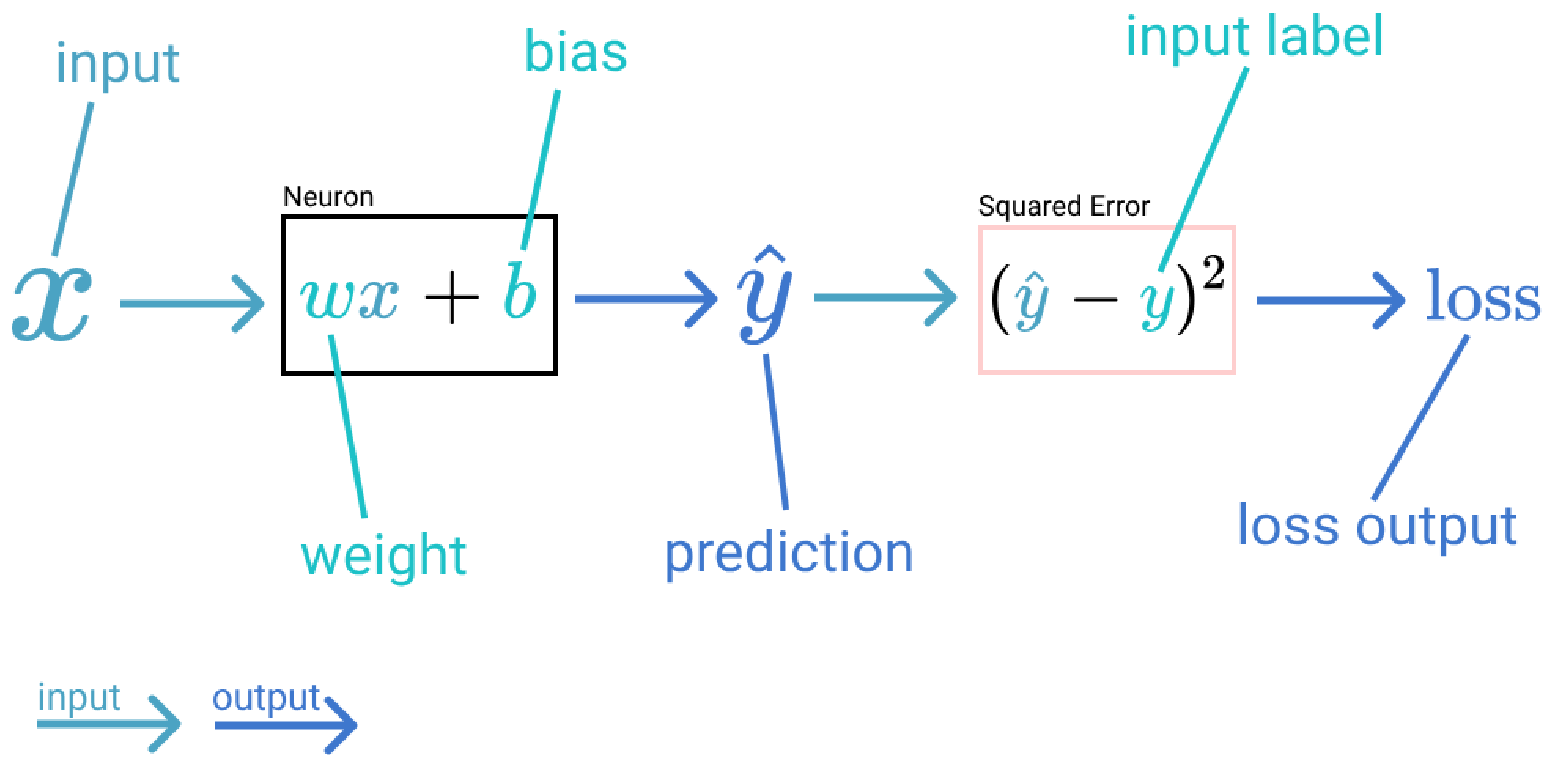
Gradient descent

No matter how your deep learning function learning machine is structured, and no matter what loss function you choose, you can use gradient descent to train your neural network!

https://www.youtube.com/watch?v=Ilg3gGewQ5U

E.g.

Let's go through a concrete example of forward propagation, then an emphasis on backward propagation. The training example will be $(x = 2.1, y = 4)$, the weight will be $w = 1$, and the bias will be $b = 0$. $\hat{y}$ represents the neuron output and predicted value, and loss represents squared error loss.



Exercise 5.

E.g.

Now we can go backward and compute partial derivatives with the chain rule to get the ∇*loss*

$$\begin{tikzcd} \boxed{\begin{matrix}\dfrac{\partial \hat{y}}{\partial b}\dfrac{\partial \text{loss}}{\partial \hat{y}} \\ \dfrac{\partial \hat{y}}{\partial w}\dfrac{\partial \text{loss}}{\partial \hat{y}}\end{matrix}} & \dfrac{\partial \text{loss}}{\partial \hat{y}} \arrow[l] & \boxed{2(\hat{y}-y)} \arrow[l] \end{tikzcd}$$

E.g.

Now we can go backward and compute partial derivatives with the chain rule to get the $\nabla loss$

$$\begin{tikzcd} \boxed{\begin{matrix}(2.1)(-3.8)\\ (1)(-3.8)\end{matrix}} & -3.8 \arrow[l] & \boxed{2(2.1-4)} \arrow[l] \end{tikzcd}$$

$$\frac{\partial loss}{\partial w} = -7.98$$

$$\frac{\partial loss}{\partial b} = -3.8$$

E.g.

Then, we update the parameters with opposite gradient to descend loss, in this case learning rate is lr $= 0.01$

$$w := w - lr \cdot \frac{\partial loss}{\partial w} = (1) - (0.01) \cdot (-7.98) = 1.0798$$

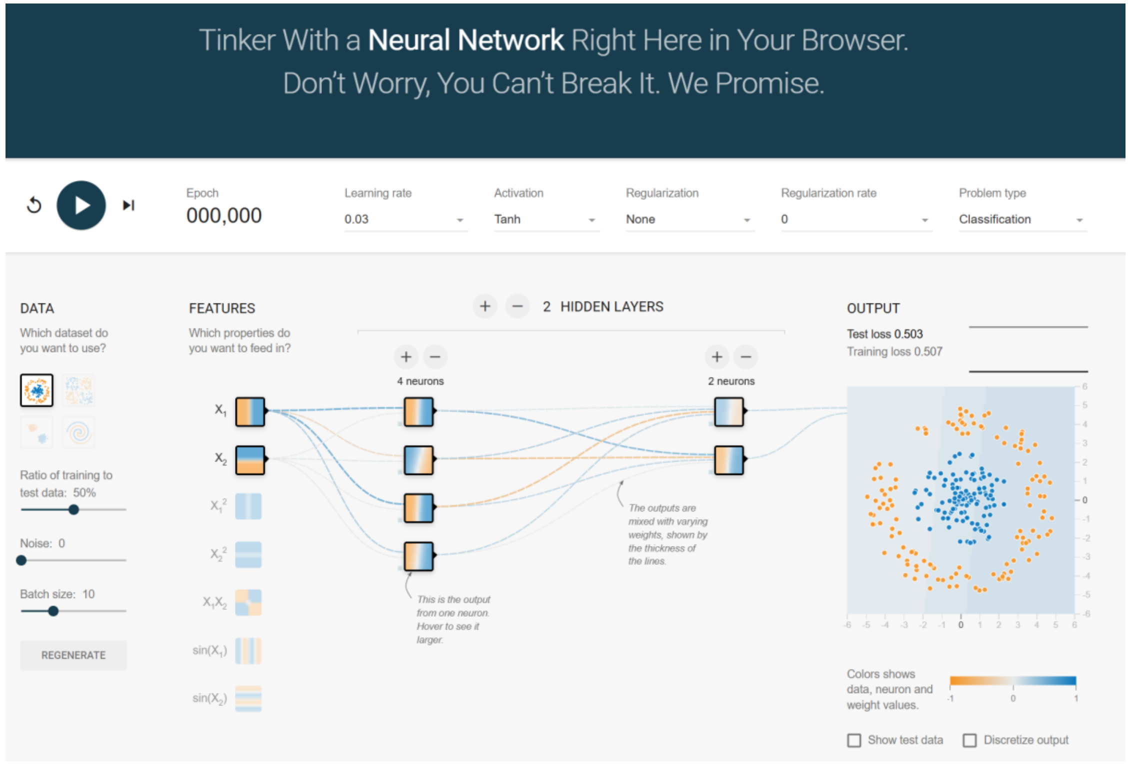
$$b := b - lr \cdot \frac{\partial loss}{\partial b} = (0) - (0.01) \cdot (-3.8) = 0.038$$

To see how well our tuned parameters do, let's do one more forward pass

$$\text{loss} = \big((1.0798)(2.1) + 0.038\big) - 4)^2 = 2.87$$

Total loss decrease of $3.61 - 2.87 = 0.74$. Loss went down!

Let's play in this playground (5 mins)



Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.

Epoch 000,000

Learning rate 0.03

Activation Tanh

Regularization None

Regularization rate 0

Problem type Classification

DATA
Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES
Which properties do you want to feed in?

2 HIDDEN LAYERS

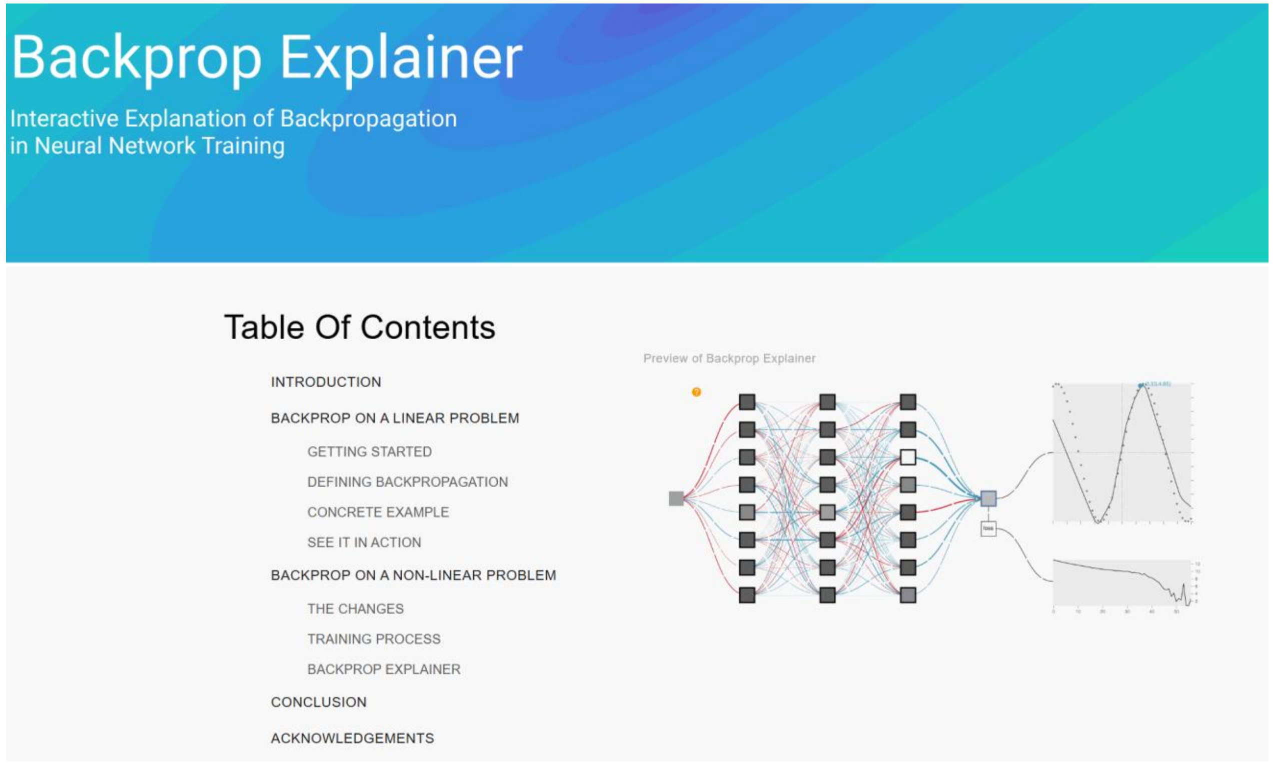
OUTPUT
Test loss 0.503
Training loss 0.507

The outputs are mixed with varying weights, shown by the thickness of the lines.

This is the output from one neuron. Hover to see it larger.

Colors shows data, neuron and weight values.

https://playground.tensorflow.org/



Backprop Explainer

Interactive Explanation of Backpropagation
in Neural Network Training

Table Of Contents

https://xnought.github.io/backprop-explainer/

Softmax

We have several numbers, and we want to apply a transformation so that they add up to 1. However, the larger numbers should remain larger, and the smaller numbers should remain smaller. How can we achieve this?

$$\begin{array}{ccc} a & & \alpha \\ b & \longrightarrow & \beta \\ c & & \gamma \end{array}$$

$$\boxed{\alpha + \beta + \gamma = 1}$$

Softmax

$$T = e^{a} + e^{b} + e^{c}$$

We will do it like this:

$$\begin{tikzcd} a & e^{a} & e^{a}/T \\ b \arrow[r] & e^{b} \arrow[r] & e^{b}/T \\ c & e^{c} & e^{c}/T \end{tikzcd}$$

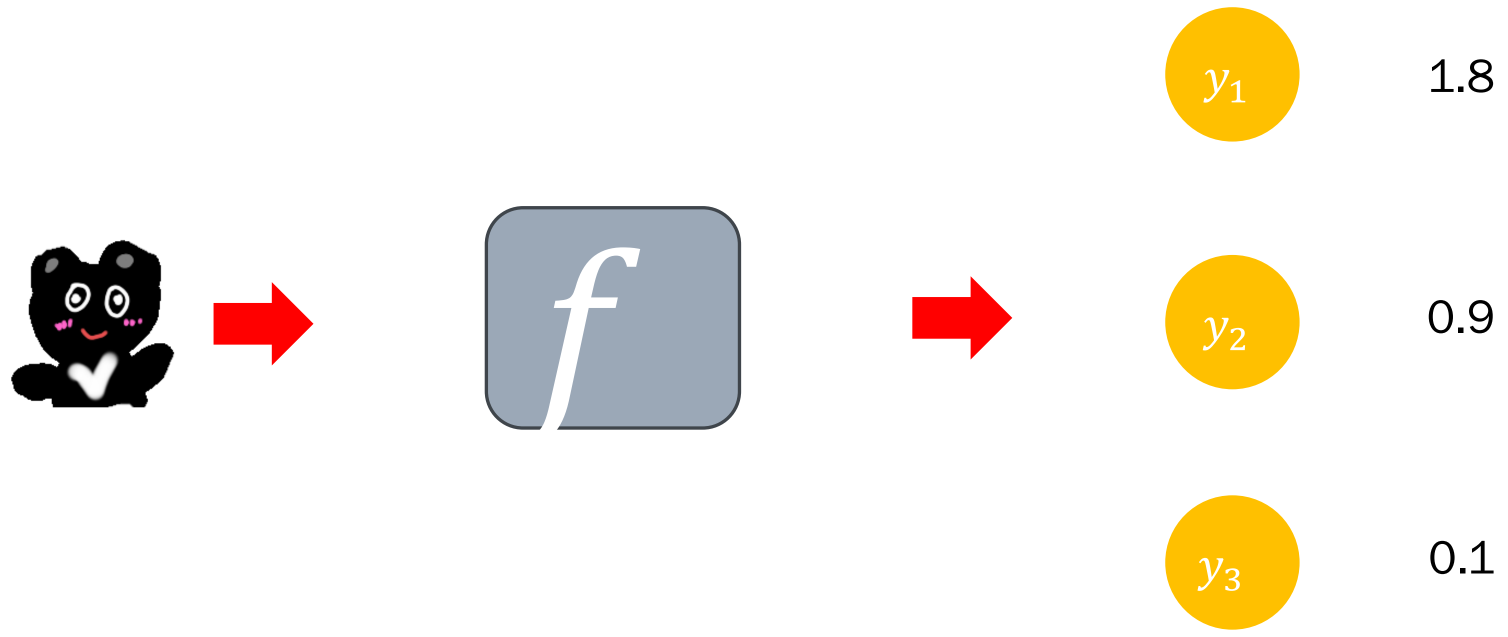
Each number is transformed by an exponential function.

In proportion

This is called **softmax**

Softmax

Quiz: Calculate the value using softmax function



Exercise 6.

What are the outputs calculated by softmax function?

https://opendatacam.github.io/demo-object-detection-browser/

The probability of an event A is defined as:

$$P(A) = \frac{Number\ of\ outcomes\ for\ event\ A}{Total\ number\ of\ possible\ outcomes}$$

Hyperparameter

Hyperparameter is to tune these parameters instead of w and b

Uncontrollable	Controllable
Parameters/bias	η Learning rate
	Activation function
	Optimizer
	Batch size
	Epochs
	Dropout rate

Overfitting and Underfitting

Overfitting happens when a model fits the training data too well, resulting in poor performance on new data, while

Underfitting happens when a model is too simple and fails to capture the underlying patterns, leading to poor performance on both training and new data.

Overfitting

Batch normalization, L2 regularization, dropout

Underfitting

Add data, add layer, check features

Job interview for the position of data scientist.

Website: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-interview-questions

Top Deep Learning Interview Questions and Answers for 2024

Lesson 17 of 26

By Kartik Menon

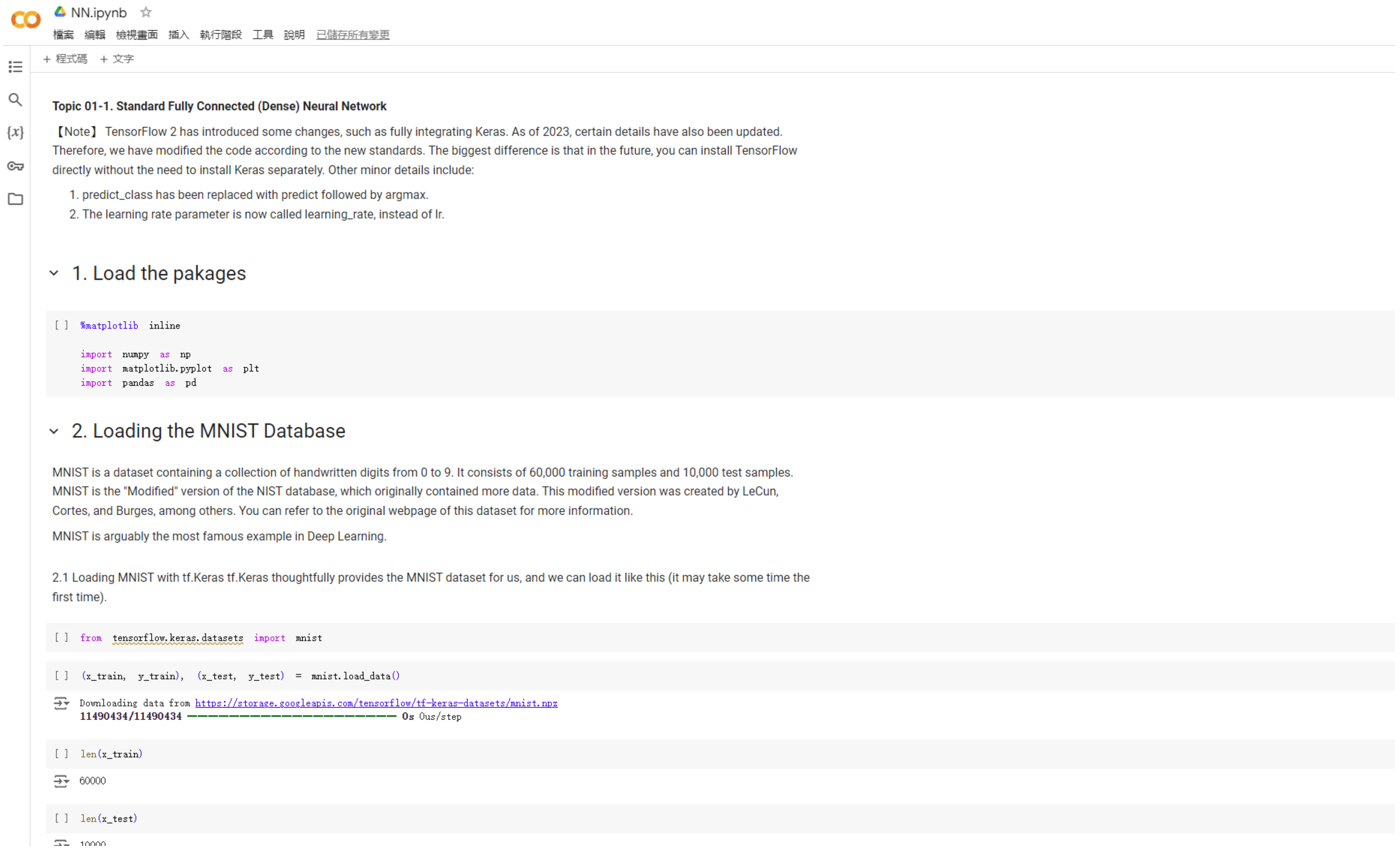
Last updated on Sep 17, 2024

316780

1. What is a Neural Network?
2. What Is a Multi-layer Perceptron(MLP)?
3. What is the Boltzmann Machine?
4. What Is the Role of Activation Functions in a Neural Network?
5. What Is the Cost(Lost) Function?
6. What Is Gradient Descent?
7. What Do You Understand by Backpropagation?
8. What Are the Softmax and ReLU Functions?
9. What Will Happen If the Learning Rate Is Set Too Low or Too High?
10. What is Overfitting and Underfitting, and How to Combat Them?

Now, we are going to see how to implement a neural network

Source code



NN.ipynb

Topic 01-1. Standard Fully Connected (Dense) Neural Network

【Note】 TensorFlow 2 has introduced some changes, such as fully integrating Keras. As of 2023, certain details have also been updated. Therefore, we have modified the code according to the new standards. The biggest difference is that in the future, you can install TensorFlow directly without the need to install Keras separately. Other minor details include:

1. predict_class has been replaced with predict followed by argmax.
2. The learning rate parameter is now called learning_rate, instead of lr.

1. Load the pakages

```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

2. Loading the MNIST Database

MNIST is a dataset containing a collection of handwritten digits from 0 to 9. It consists of 60,000 training samples and 10,000 test samples. MNIST is the "Modified" version of the NIST database, which originally contained more data. This modified version was created by LeCun, Cortes, and Burges, among others. You can refer to the original webpage of this dataset for more information.

MNIST is arguably the most famous example in Deep Learning.

2.1 Loading MNIST with tf.Keras tf.Keras thoughtfully provides the MNIST dataset for us, and we can load it like this (it may take some time the first time).

```
from tensorflow.keras.datasets import mnist
```

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
len(x_train)
```

```
60000
```

```
len(x_test)
```

```
10000
```

Exercise today

1. Use the provided source code and dataset, and then implement in **colab environment**
2. Try to adjust the hidden layer and amouts of neurals, feel free to change **batch size**, and **epochs**
3. Try to change the different **activation function**, **loss function**, **optimizer**, also you can add **dropout** as well