Deep Learning

Fundamentals and state of the art architectures

Jawad ALAOUI

Université Gustave Eiffel

NORMA

The Scope of Deeplearning module

- Introduction to Deep Learning and Neural Networks
- Standard Neural Networks
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs), LSTMs
- Transformers and Hugging Face
- Reinforcement Learning with Human Feedback (RLHF)
- Diffusion Models

The Problem

Image Classification for Autonomous Vehicles

Is this a bicycle ?

Why it matters: The classification of vehicles is essential for self-driving cars to navigate safely, as distinguishing between bicycles, motorcycles, and buses affects road decision-making.

Feature engineering vs feature learning: Classic machine learning models like linear and logistic regression struggle with complex image problems because they depend on handcrafted features, especially when processing unstructured data.
Handcrafted features cannot adequately handle the complexity of data due to variations in lighting, angles, and vehicle types in real-world situations, making manual extraction very complex.

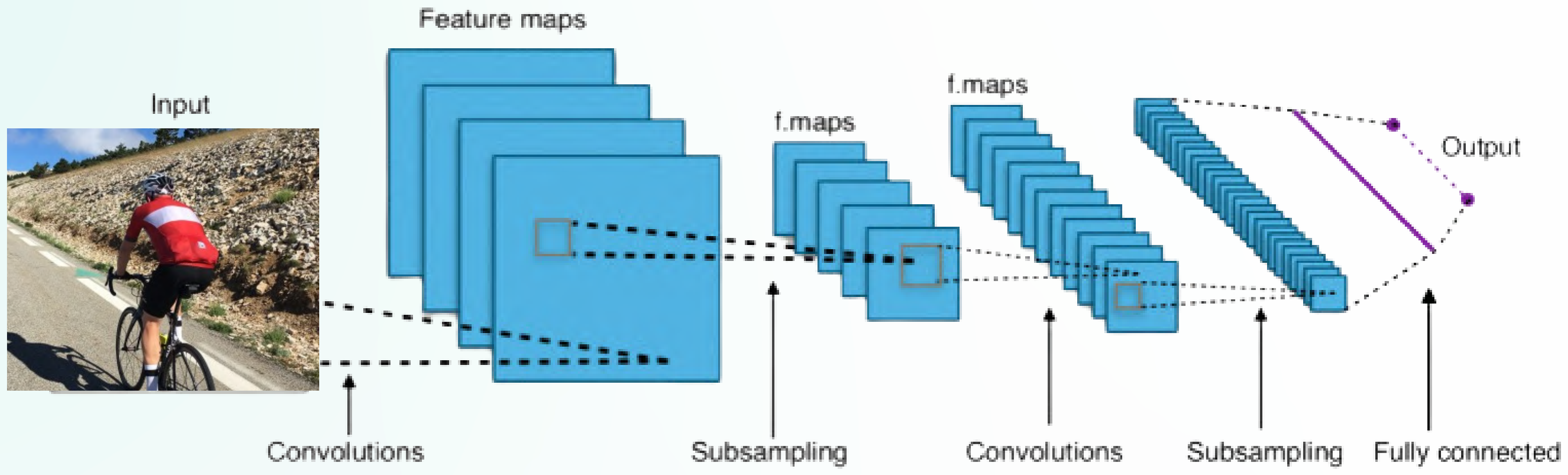
non-linearity: Moreover, many important characteristics of an image are non-linear. Recognizing the difference between a truck and a car requires complex interactions between pixel values that simple linear models can't capture.

The Solution

Introduction to Deep Learning

Deep learning demonstrates exceptional proficiency in complex tasks such as image classification by learning hierarchical representations from raw data. This capability enhances the understanding of high-dimensional data, exemplified in applications like vehicle classification in self-driving cars.

Its versatility extends across various domains, including natural language processing, speech recognition, and images segmentation, where it markedly surpasses traditional methodologies by automatically acquiring meaningful representations.

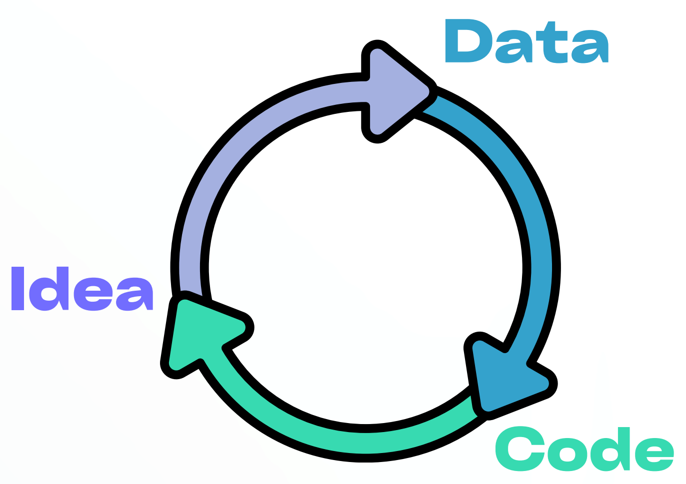


- **1943** — Warren McCulloch and Walter Pitts introduced the concept of a simplified brain mode
- **1958** — Frank Rosenblatt developed the Perceptron algorithm, one of the earliest models for neural networks capable of learning.
- **1965** — The first working deep learning algorithm Alexey Ivakhnenko and Lapa in Ukraine
- **1967** — The first deep learning multilayer perceptron trained by stochastic gradient descent by Shun'ichi Amari.
- **1989** — Yann LeCun utilized backpropagation to train convolutional neural networks (CNNs) for tasks such as handwritten digit recognition.
- **2012** — Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton won the ImageNet competition using their deep convolutional neural network, AlexNet.
- **2017** — Google researchers developed the Transformer model, transforming natural language processing and paving the way for large-scale models like BERT and GPT.

Why Deep Learning is Possible Today ?

The rise of GPUs and specialized hardware has greatly reduced the computational cost of training large neural networks, making deep learning more accessible and efficient.

The past decade has seen an explosion of data from mobile devices, sensors, and the Internet of Things (IoT), which traditional algorithms struggle to manage effectively.



Key innovations like Backpropagation, ReLU activation, and attention mechanisms have significantly enhanced the efficiency and performance of deep learning models, enabling faster training and improved results for large-scale networks.

Numerous Architectures for Various Challenges

Supervised learning

Convolutional Neural Networks (CNNs):
Used for image classification and object detection tasks.

Recurrent Neural Networks (RNNs) & LSTMs:
Applied in time series forecasting and sequential data processing.

Transformers:
Solve text classification and sequence translation tasks using labeled data.

Unsupervised learning

Autoencoders:
Used for dimensionality reduction and anomaly detection in unlabeled d

Generative Adversarial Networks (GANs):
Generate synthetic data like images or videos using a two-network system.

Self-supervised learning

Diffusion Models:
Generate high-quality images by learning to denoise noisy data.

Transformers:
Learn from masked data for tasks like language representation.

Foundations: Introducing the Perceptron

The perceptron is a fundamental algorithm for binary classification and serves as the building block for more complex neural networks.

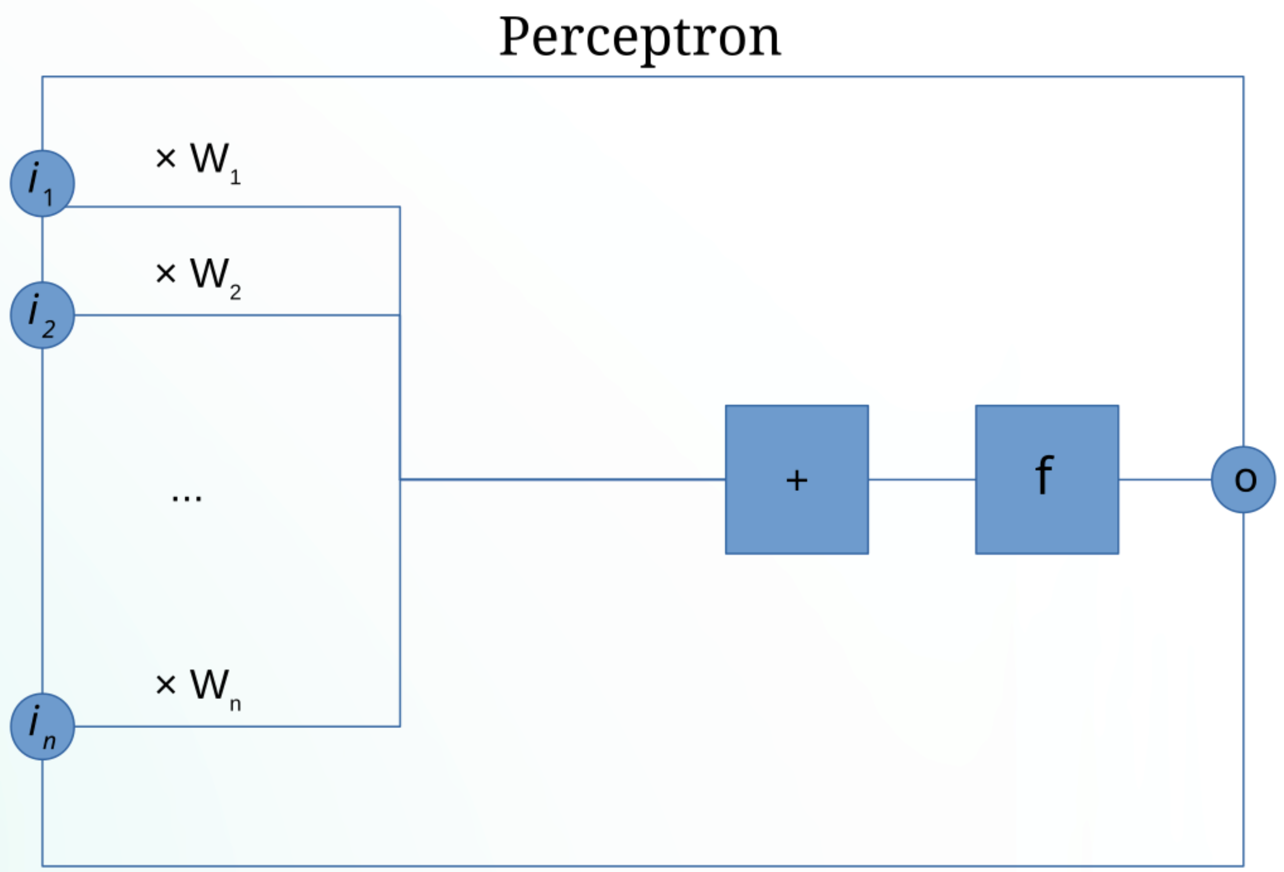
Input: $x = (x_1, x_2, \ldots, x_n)$

Weights: $w = (w_1, w_2, \ldots, w_n)$

Bias: b

Output: $o = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right)$

Activation function*: $f(x) = \dfrac{1}{1+e^{-x}}$ (Sigmoid function)



* Activation function is not always Sigmoid. many other options are available: Step function, ReLu, Thanh...

Perceptron vs logistic regression

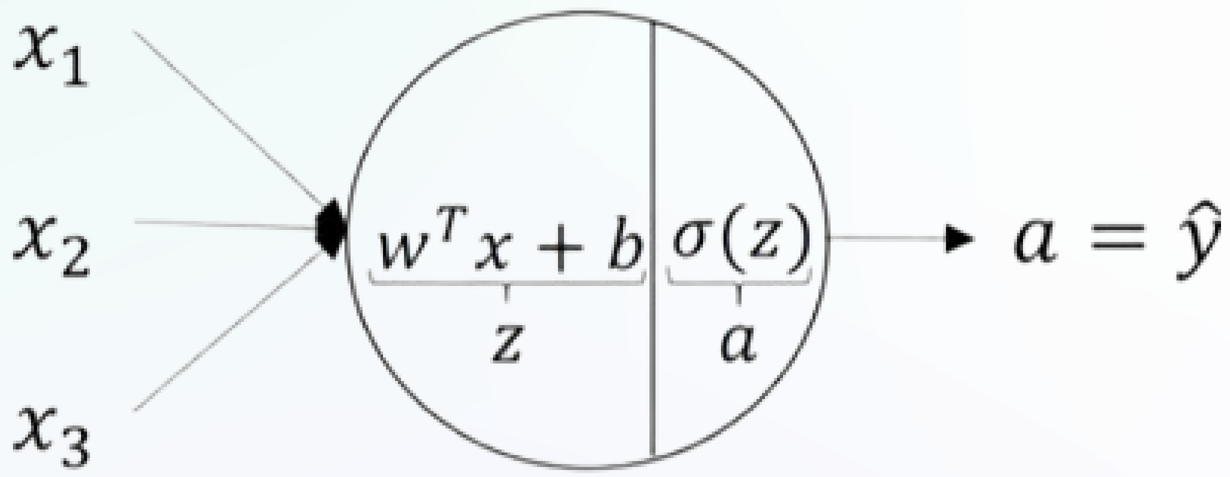
Logistic regression is A generalized linear model (GLM) for classification. It solves classification problems by outputting probabilities chosen to be the Bernoulli distribution.

The link function (logit) is: $g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$

$$P(o=1) = \mathrm{E}(o) = \mu = g^{-1}(z) = \frac{1}{1+e^{-z}}$$

$$z = w^T x + b$$

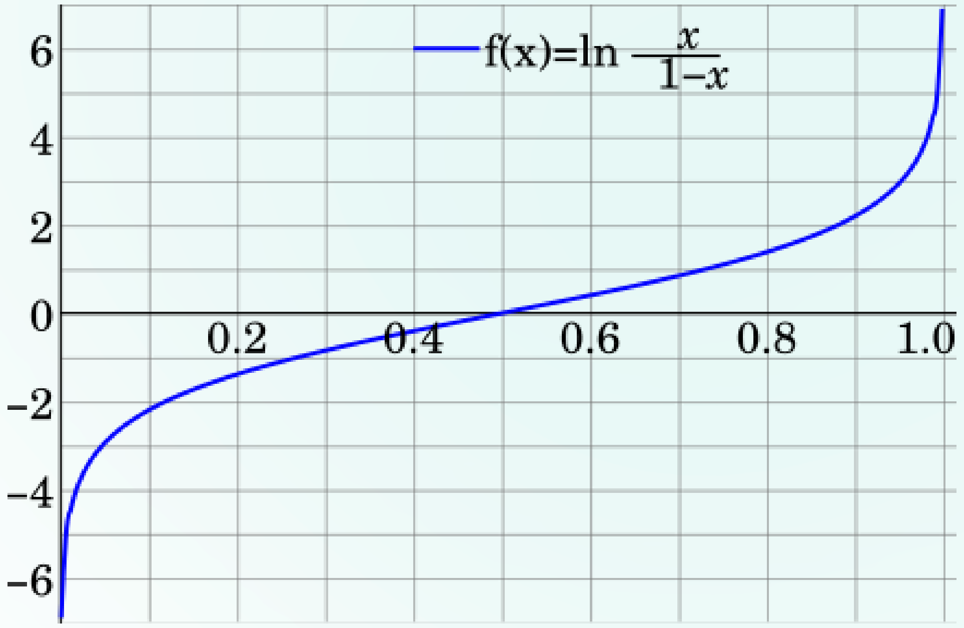
$$a = g^{-1}(z) = \sigma(z)$$



Let's begin by exploring what we can accomplish with a single neuron.

Logistic Regression – Link Function

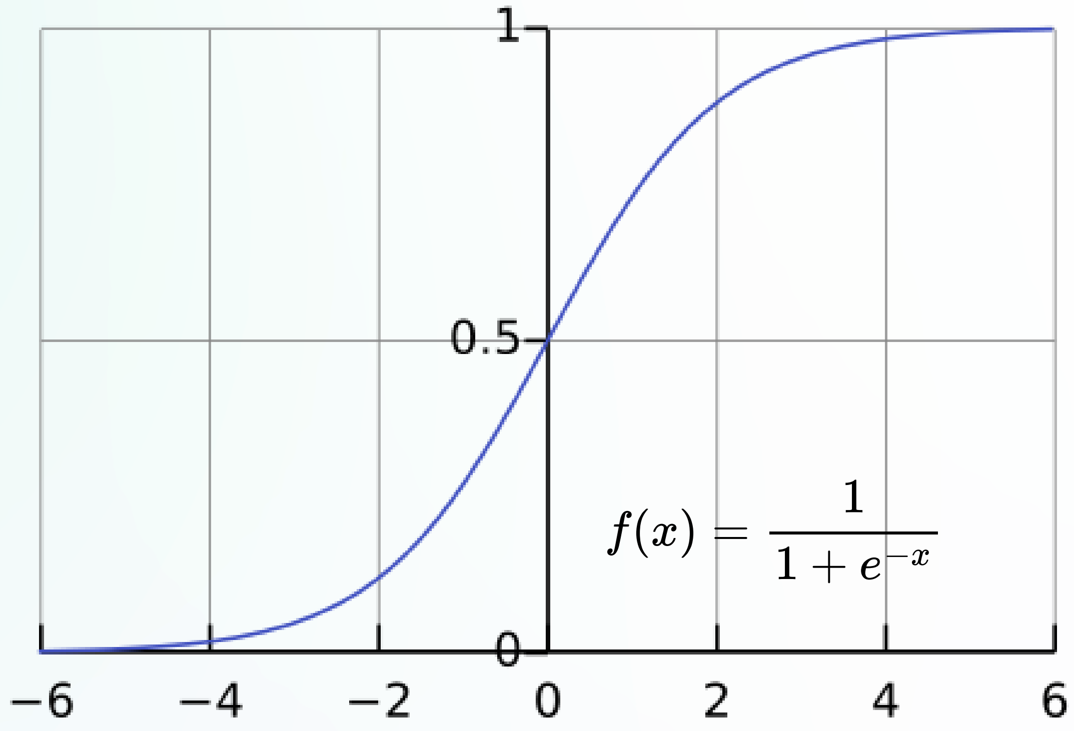
Logit Function



The logit function represents the log of the odds of the outcome, transforming probabilities from

$$(0, 1) \text{ to } (-\infty, +\infty)$$

Sigmoid Function

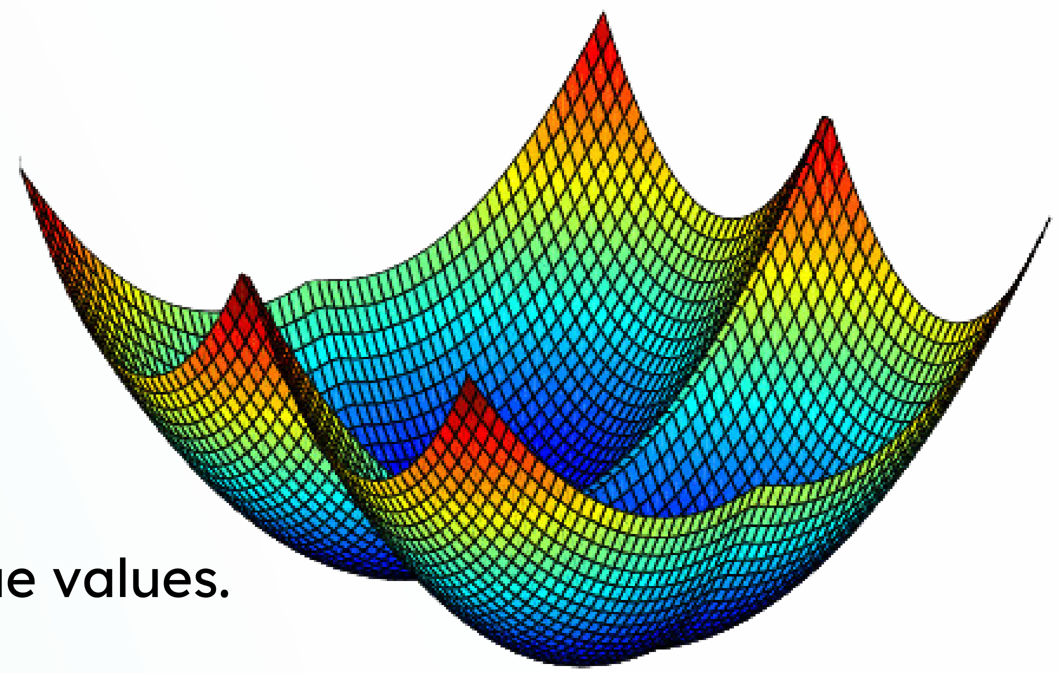


The sigmoid function is the inverse of the logit function and maps values from

$$(-\infty, +\infty) \text{ to } (0, 1)$$

What is a Cost function

A cost function measures how well a model fits the data by quantifying the error between the predicted and actual outputs. It is a key component in the optimization process.



Multiple Ways to Define a Cost Function:

There are several ways to define a cost function, depending on the model and problem

Least Squares: Used in regression, it minimizes the squared differences between predicted and true values.

$$\text{Cost} = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Cross-Entropy (Log Loss): Applied in classification tasks, it measures the error between predicted probabilities and true classes.

$$\text{Cost} = E(-log(p(X))) = -\frac{1}{m} \sum_{i=1}^{m} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right)$$

Maximum Likelihood: Maximizes the likelihood of observed data under the model, commonly used in probabilistic models.

$$\text{Cost} = -\text{Log Likelihood} = -\sum_{i=1}^{m} \log P(y_i|\theta)$$

Loss vs. Cost Function:

The loss function measures error for one training example, while the cost function aggregates this error across the entire dataset.

What is Gradient Descent?

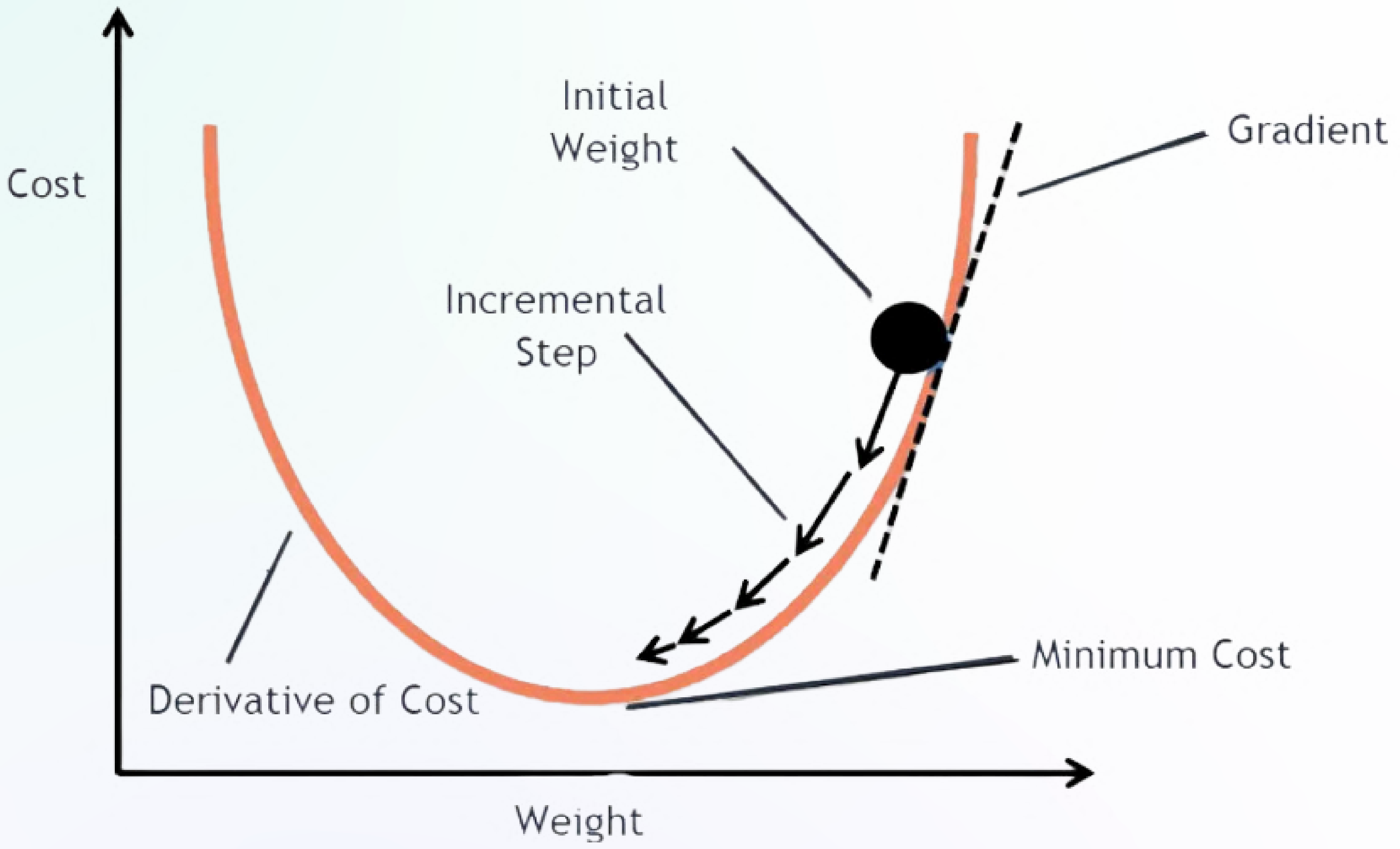
Gradient Descent, introduced by Augustin-Louis Cauchy in 1847, is an optimization method for minimizing functions. It is essential in machine learning for minimizing the cost function by iteratively adjusting parameters.

Learning rate

$$\theta := \theta - \alpha \frac{\partial}{\partial \theta} J(\theta)$$

Model parameters

Cost function



Cost function for Logistic Regression (1)

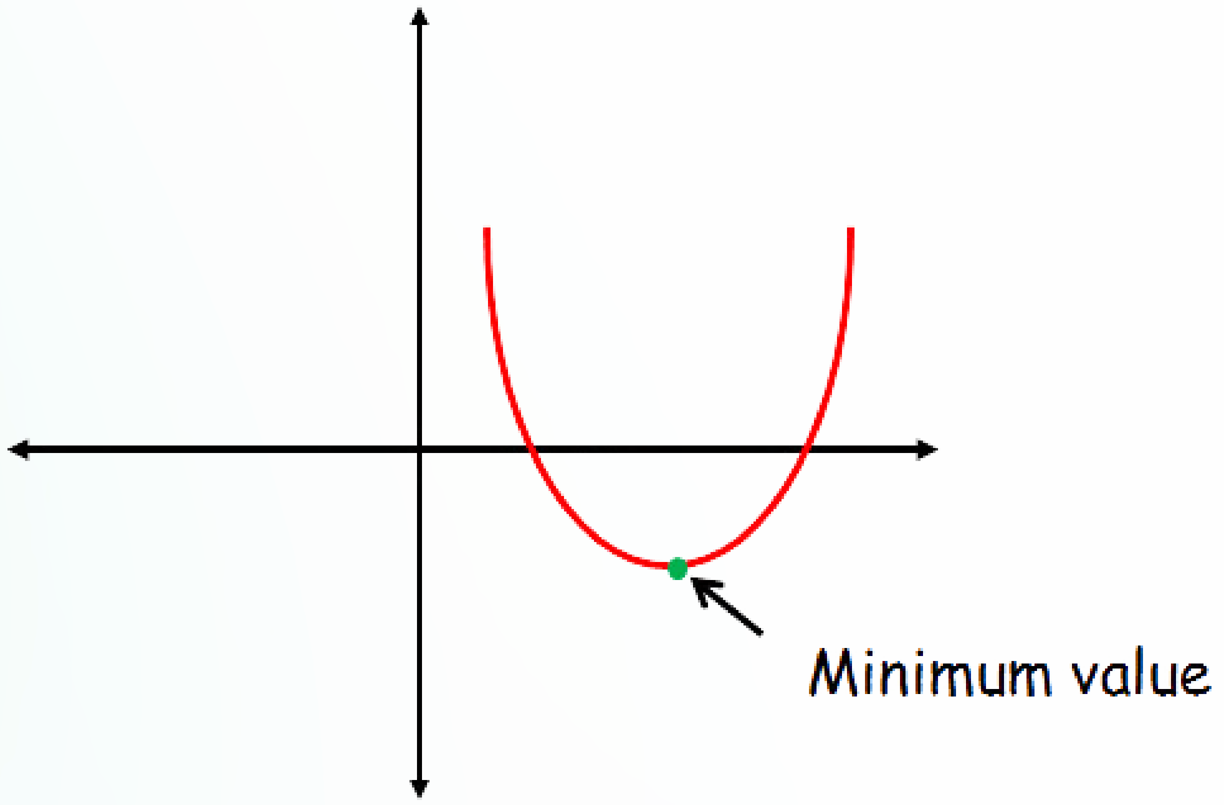
The cost function for logistic regression can be derived from both likelihood and cross-entropy, which lead to the same formulation.

The likelihood function maximizes the probability of observing the data

$$L(\beta) = \prod_{i=1}^{n} p(x_i)^{y_i} \cdot (1 - p(x_i))^{1-y_i}$$

Log-Likelihood simplifies the computation

$$\log L(\beta) = \sum_{i=1}^{n} y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))$$



Negative Log-Likelihood (Cross-Entropy): Minimizing the negative log-likelihood is equivalent to minimizing cross-entropy

$$-\log L(\beta) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

Cost function for Logistic Regression (2)

The general cost function for logistic regression is derived from the negative log-likelihood and, in this context, is divided by mmm (the number of training examples) to normalize the loss over the dataset.

$$J = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log(a_i) + (1 - y_i)\log(1 - a_i)\right)$$

Where:

$$a_i = \sigma(z_i)$$

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

$$z_i = \sum_{j=1}^{n} w_j \cdot x_{i,j} + b$$

and y are the real values of the output (taget value).

Define Partial Derivative

The general cost function for logistic regression is derived from the negative log-likelihood and, in this context, is divided by mmm (the number of training examples) to normalize the loss over the dataset.

Chain Rule to Derive the Partial Derivative:

$$\frac{\partial J}{\partial w_j} = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{\partial J_i}{\partial a_i} \cdot \frac{\partial a_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial w_j} \right)$$

$$\left\{ \begin{aligned} \frac{\partial J_i}{\partial a_i} &= -\frac{y_i}{a_i} + \frac{1-y_i}{1-a_i} = a_i - y_i \\ \frac{\partial a_i}{\partial z_i} &= a_i(1-a_i) \\ \frac{\partial z_i}{\partial w_j} &= x_{ij} \end{aligned} \right. \longrightarrow \left\{ \begin{aligned} \frac{\partial J}{\partial w_j} &= \frac{1}{m} \sum_{i=1}^{m} (a_i - y_i) x_{ij} \\ \frac{\partial J}{\partial b} &= \frac{1}{m} \sum_{i=1}^{m} (a_i - y_i) \end{aligned} \right.$$

Forward propagation algorithm

In forward propagation, we initialize the parameters W and b, then compute the linear combination of the input features:

Linear Combination vectorize for W:

$$z_i := w^T x_i + b$$

Activation (Sigmoid):

$$a_i := \frac{1}{1 + e^{-z_i}}$$

Cost Function (Logistic Regression):

$$J := -\frac{1}{m} \sum_{i=1}^{m} (y_i \log(a_i) + (1 - y_i) \log(1 - a_i))$$

Finally, the cost function measures the error between the predicted and actual values:

Back propagation

After the forward propagation step, we use backpropagation to update the model's parameters.
The update rules are based on the gradient of the cost function with respect to each parameter.

Weights Update:

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (a_i - y_i) x_{ij}$$

Bias Update:

$$b := b - \alpha \frac{1}{m} \sum_{i=1}^{m} (a_i - y_i)$$

Once these updates are made, forward propagation is repeated to improve model predictions.

Stopping Condition:
We stop when the cost function change is minimal (convergence) or when it reaches a set threshold, indicating the model has learned sufficiently for accurate predictions.