

Perceptron and Logistic Regression

Lê Anh Cường

2020

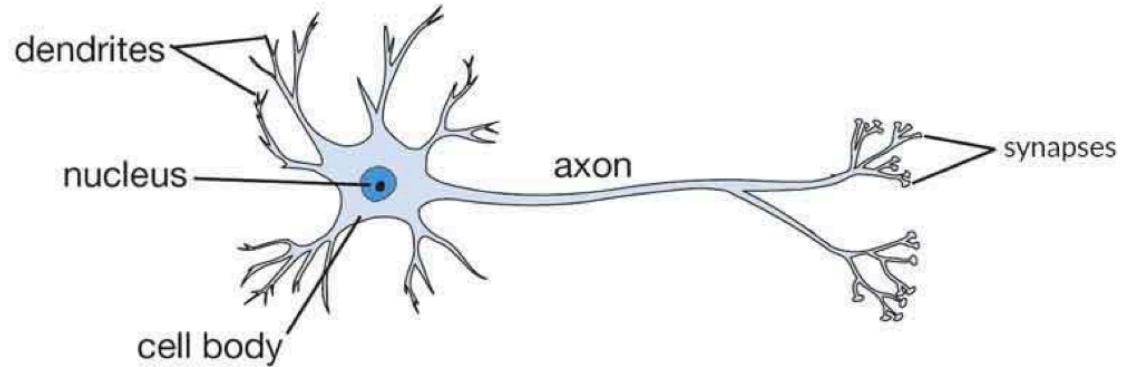
Outline

1. Biological Neuron and Artificial Neural Network
2. Perceptron and Learning Perceptron
3. Information Theory
4. Logistic function
5. Logistic classification with Cross Entropy Loss

References

- <https://www.xenonstack.com/blog/artificial-neural-network-applications/>
- <https://peterroelants.github.io/posts/cross-entropy-logistic/>
- <https://peterroelants.github.io/posts/cross-entropy-softmax/>

Biological Neuron



- **Function of Dendrite (đuôi gai)**

It receives signals from other neurons.

- **Soma (cell body)**

It sums all the incoming signals to generate input.

- **Axon Structure (cấu trúc trực)**

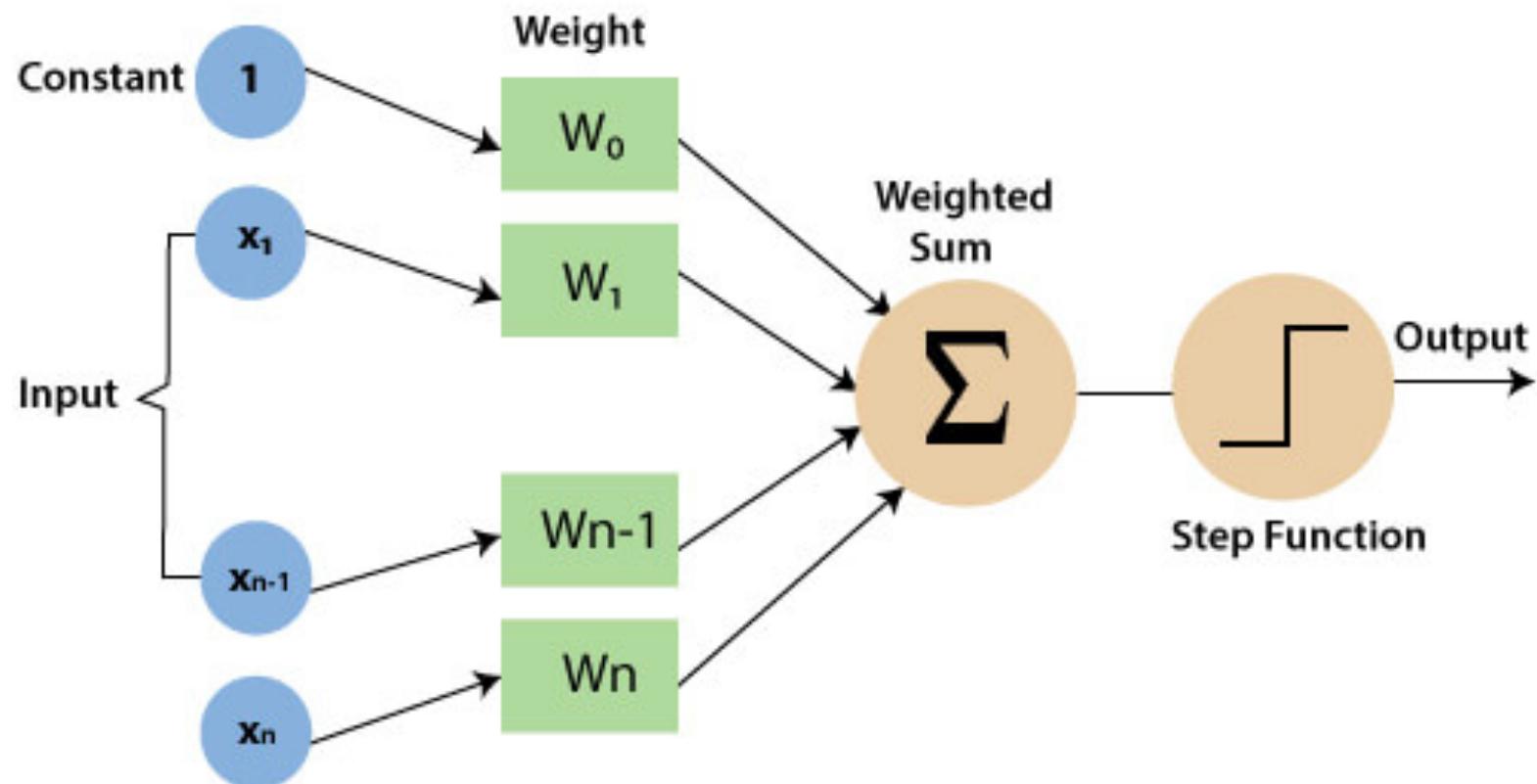
When the sum reaches a threshold value, neuron fires and the signal travels down the axon to the other neurons.

- **Synapses Working (khớp thần kinh)**

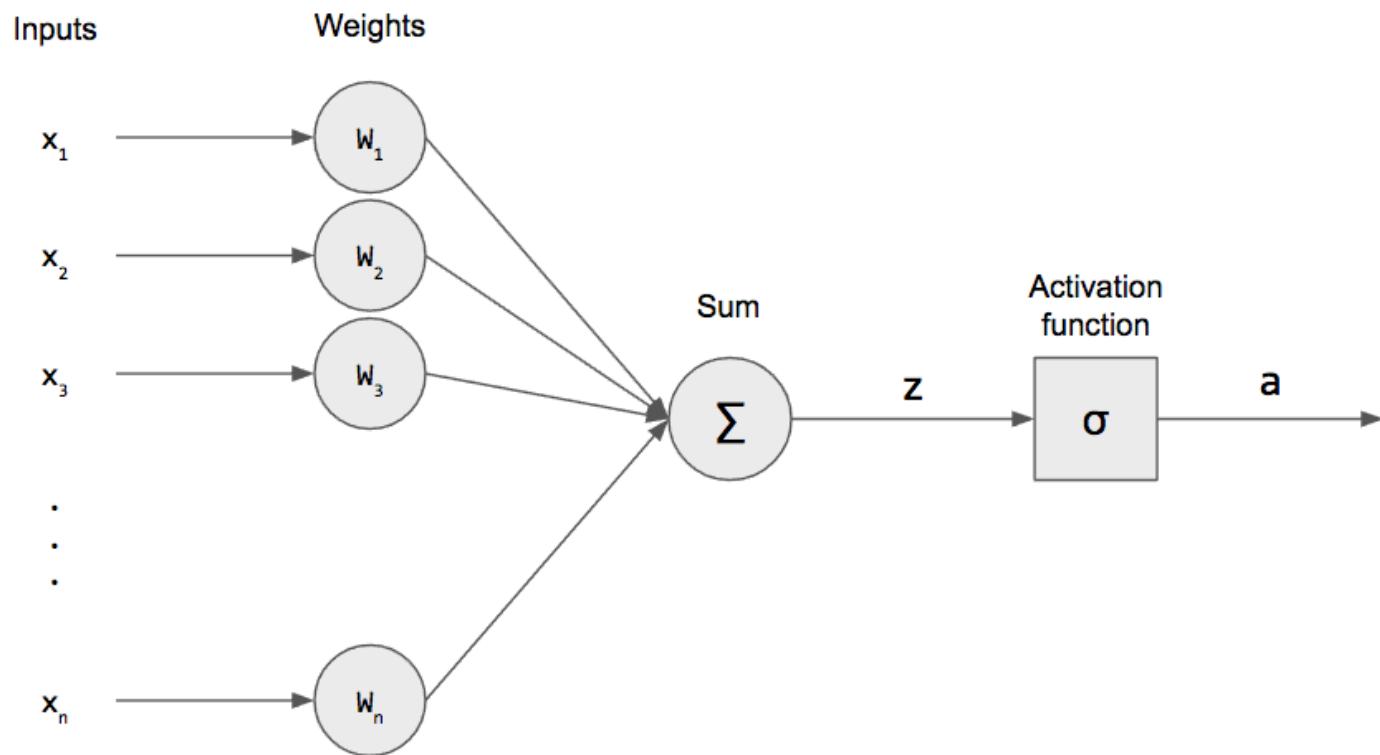
The point of interconnection of one neuron with other neurons. The amount of signal transmitted depend upon the strength (synaptic weights) of the connections.

The connections can be inhibitory (decreasing strength) or excitatory (increasing strength) in nature.

Artificial Neural or Perceptron

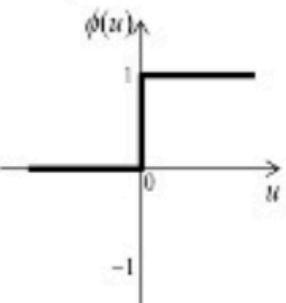


Perceptron



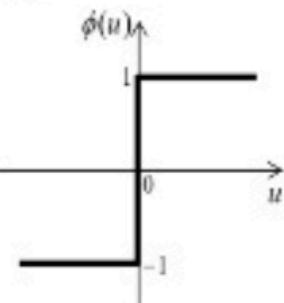
Activation functions

step function



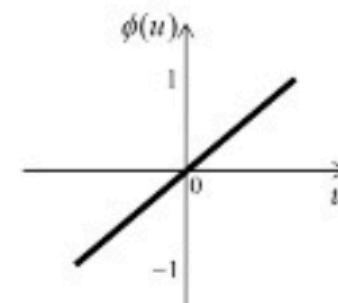
$$\phi_{\text{step}}(u) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

sign function

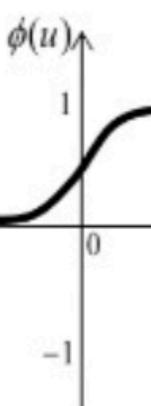


$$\phi_{\text{sign}}(u) = \begin{cases} 1 & \text{if } u \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

identity function



$$\phi_{\text{id}}(u) = u$$



sigmoid function

$$\phi_{\text{sig}}(u) = \frac{1}{1 + e^{-u}}$$



hyper tangent function

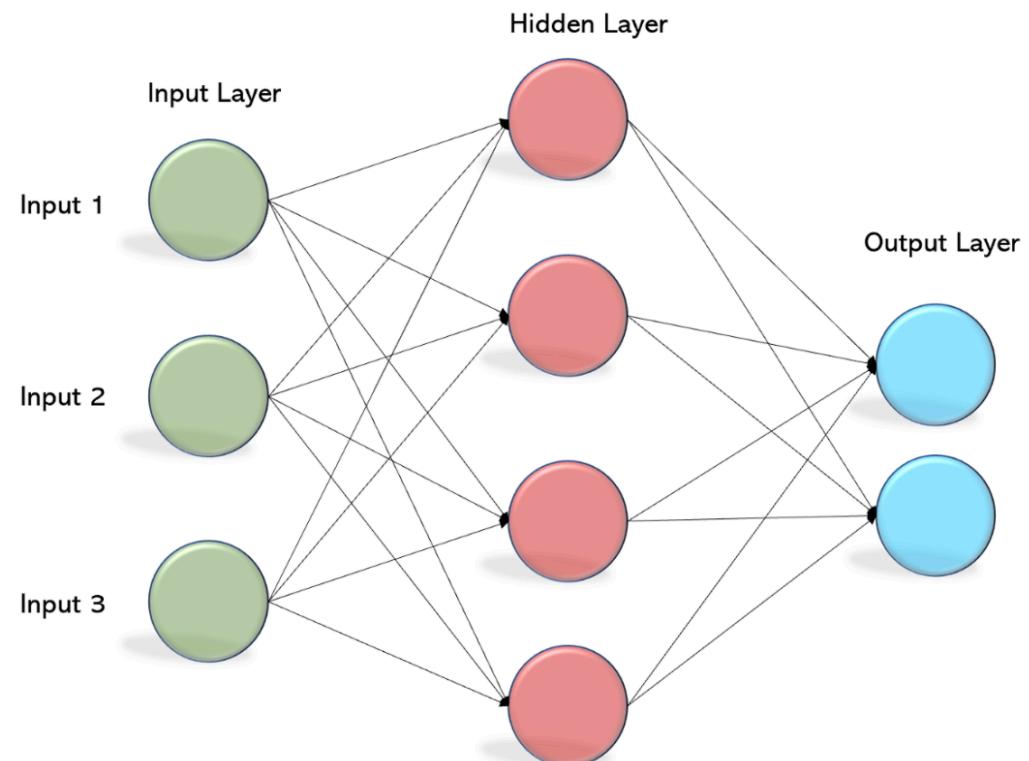
$$\phi_h = \frac{e^u - 1}{e^u + 1}$$

Activation functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU) [2]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Multi Layer Perceptron

- A multi layer perceptron (MLP) is a class of feed forward artificial neural network. MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called back propagation for training.



Learning Perceptron

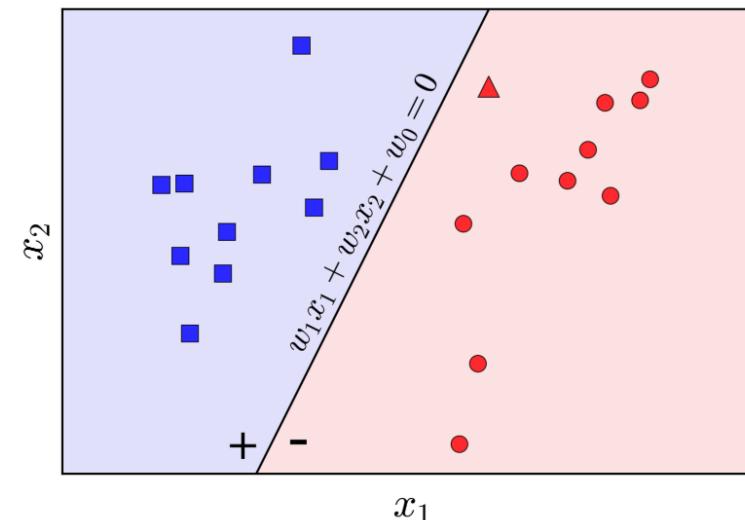
$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{d \times N}$$

$$\mathbf{y} = [y_1, y_2, \dots, y_N] \in \mathbb{R}^{1 \times N}$$

$$f_{\mathbf{w}}(\mathbf{x}) = w_1x_1 + \dots + w_dx_d \quad \text{label}(\mathbf{x}) = 1 \text{ if } \mathbf{w}^T \mathbf{x} \geq 0, \text{ otherwise } -1$$
$$= \mathbf{w}^T \bar{\mathbf{x}} = 0$$

$$\text{label}(\mathbf{x}) = 1 \text{ if } \mathbf{w}^T \mathbf{x} \geq 0, \text{ otherwise } -1$$

$$\text{label}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x})$$



Learning Perceptron: Loss function

$$J_1(\mathbf{w}) = \sum_{\mathbf{x}_i \in \mathcal{M}} (-y_i \text{sgn}(\mathbf{w}^T \mathbf{x}_i))$$

Loss function counts the missed classified, in that case y_i and $\text{sgn}(\mathbf{w}^T \mathbf{x})$ are opposite signs, therefore: $-y_i \text{sgn}(\mathbf{w}^T \mathbf{x}_i) = 1$.

However, it is discrete then difficult to calculate derivative



$$J(\mathbf{w}) = \sum_{\mathbf{x}_i \in \mathcal{M}} (-y_i \mathbf{w}^T \mathbf{x}_i)$$

Learning Perceptron: update weight

$$J(\mathbf{w}) = \sum_{\mathbf{x}_i \in \mathcal{M}} (-y_i \mathbf{w}^T \mathbf{x}_i)$$

Loss function

$$\nabla_{\mathbf{w}} J(\mathbf{w}; \mathbf{x}_i; y_i) = -y_i \mathbf{x}_i$$

derivative

$$\mathbf{w} = \mathbf{w} + \eta y_i \mathbf{x}_i$$

Update weight

Logistic Regression

Recall

Đầu ra dự đoán của:

- Linear Regression:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

- PLA:

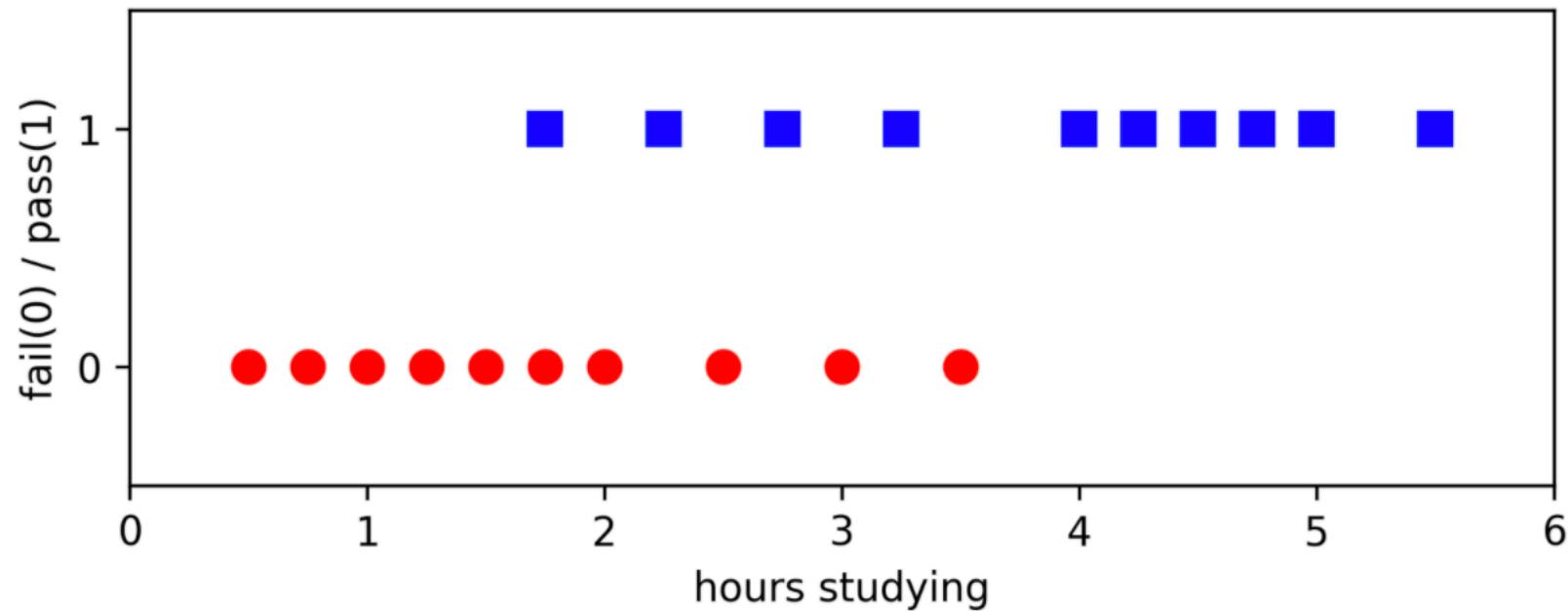
$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x})$$

Example: house studying

Một nhóm 20 sinh viên dành thời gian trong khoảng từ 0 đến 6 giờ cho việc ôn thi. Thời gian ôn thi này ảnh hưởng đến xác suất sinh viên vượt qua kỳ thi như thế nào?

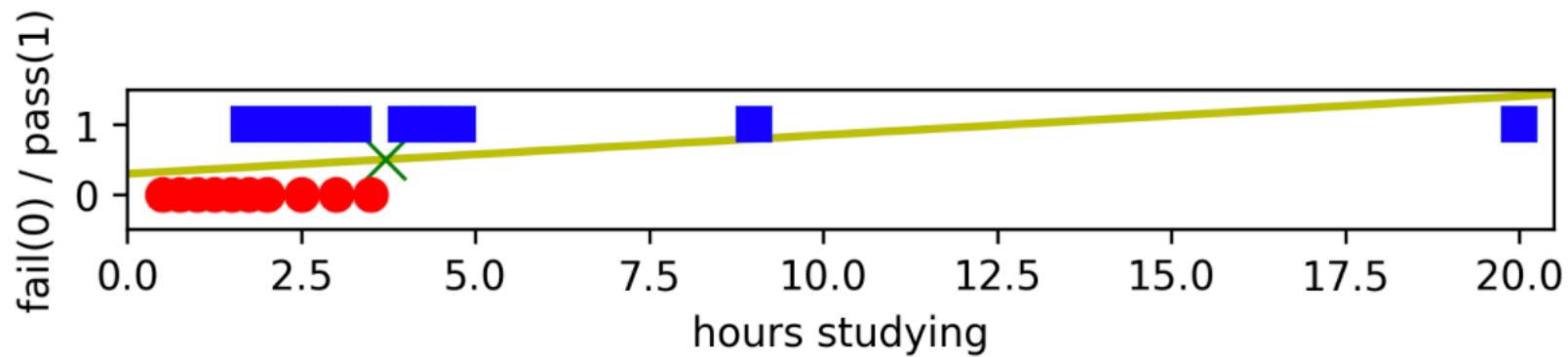
Hours	Pass	Hours	Pass
.5	0	2.75	1
.75	0	3	0
1	0	3.25	1
1.25	0	3.5	0
1.5	0	4	1
1.75	0	4.25	1
1.75	1	4.5	1
2	0	4.75	1
2.25	1	5	1
2.5	0	5.5	1

Example: house studying



Hình 1: Ví dụ về kết quả thi dựa trên số giờ ôn tập.

Example: house studying



Hình 3: Tại sao Linear Regression không phù hợp?

Logistic Regression

Đầu ra dự đoán của:

- Linear Regression:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

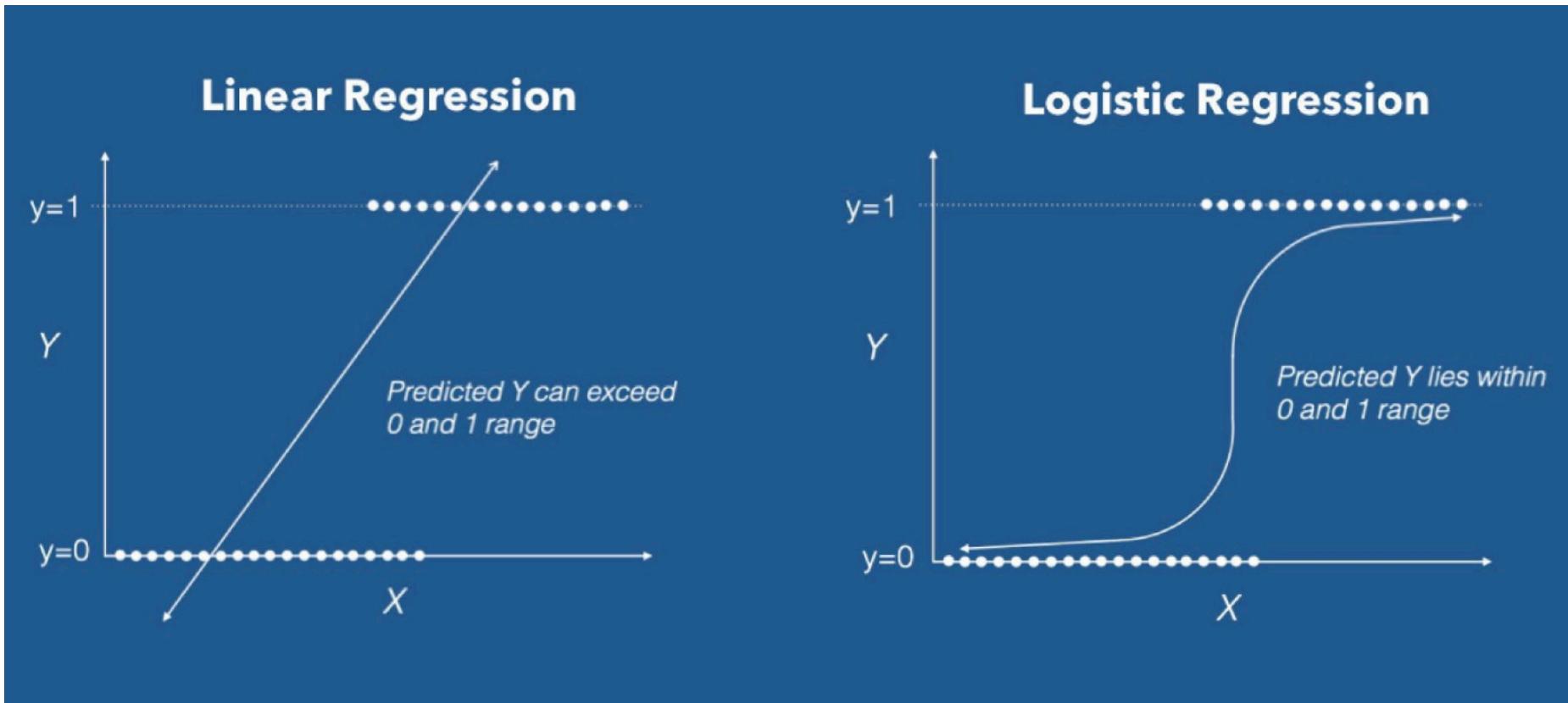
- PLA:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x})$$

Đầu ra dự đoán của logistic regression thường được viết chung dưới dạng:

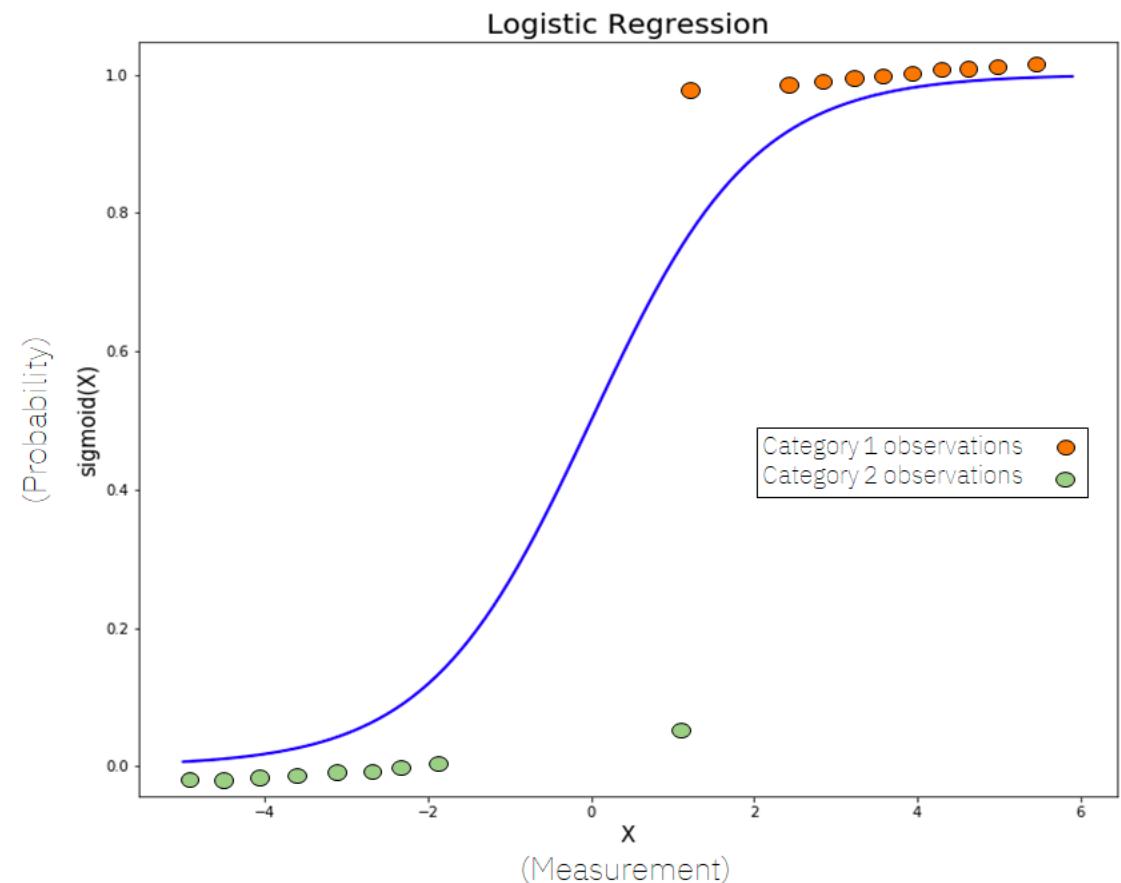
$$f(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x})$$

Logistic Regression



Logistic Regression

- Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable(or output), y , can take only discrete values for given set of features(or inputs), X .
- Contrary to popular belief, logistic regression a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”.



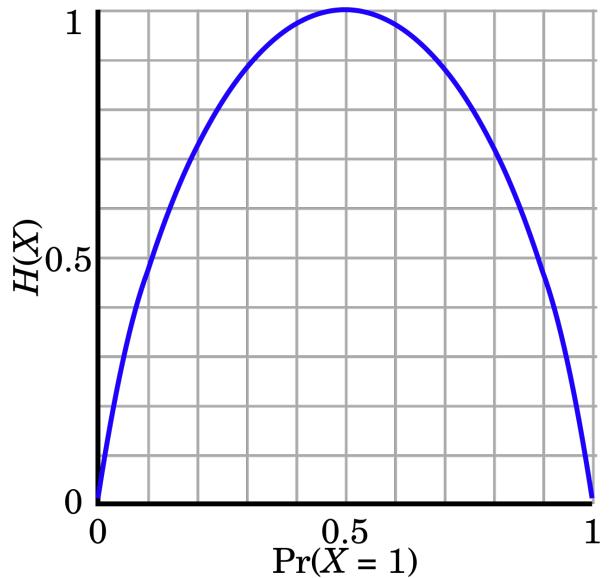
Entropy

Entropy is a **measure** of the unpredictability of the state, or equivalently, of its average **information** content.

$$H(X) = - \sum_{i=1}^n P(x_i) \log_b P(x_i)$$

$p=0.7$, then

$$\begin{aligned} H(X) &= -p \log_2(p) - q \log_2(q) \\ &= -0.7 \log_2(0.7) - 0.3 \log_2(0.3) \\ &\approx -0.7 \cdot (-0.515) - 0.3 \cdot (-1.737) \\ &= 0.8816 < 1 \end{aligned}$$



$$\begin{aligned} H(X) &= - \sum_{i=1}^n P(x_i) \log_b P(x_i) \\ &= - \sum_{i=1}^2 \frac{1}{2} \log_2 \frac{1}{2} \\ &= - \sum_{i=1}^2 \frac{1}{2} \cdot (-1) = 1 \end{aligned}$$

Kullback–Leibler divergence

In [mathematical statistics](#), the **Kullback–Leibler divergence** (also called **relative entropy**) is a measure of how one [probability distribution](#) is different from a second, reference probability distribution.^[4]

For [discrete probability distributions](#) P and Q defined on the same [probability space](#), \mathcal{X} , the Kullback–Leibler divergence from Q to P is defined^[4] to be

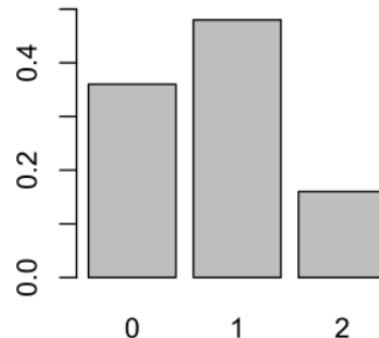
$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

which is equivalent to

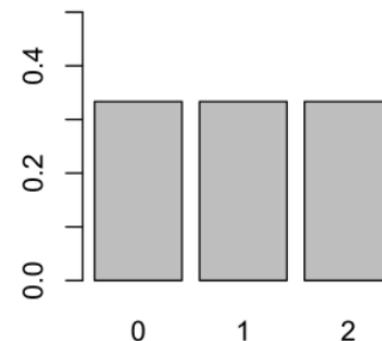
$$D_{\text{KL}}(P \parallel Q) = - \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{Q(x)}{P(x)}\right)$$

Kullback–Leibler divergence

Distribution P
Binomial with $p = 0.4$, $N = 2$



Distribution Q
Uniform with $p = 1/3$



x	0	1	2
Distribution $P(x)$	0.36	0.48	0.16
Distribution $Q(x)$	0.333	0.333	0.333

$$\begin{aligned} D_{\text{KL}}(P \parallel Q) &= \sum_{x \in \mathcal{X}} P(x) \ln \left(\frac{P(x)}{Q(x)} \right) \\ &= 0.36 \ln \left(\frac{0.36}{0.333} \right) + 0.48 \ln \left(\frac{0.48}{0.333} \right) + 0.16 \ln \left(\frac{0.16}{0.333} \right) \\ &= 0.0852996 \end{aligned}$$

Kullback–Leibler divergence

x	0	1	2
Distribution $P(x)$	0.36	0.48	0.16
Distribution $Q(x)$	0.333	0.333	0.333

$$\begin{aligned} D_{\text{KL}}(P \parallel Q) &= \sum_{x \in \mathcal{X}} P(x) \ln \left(\frac{P(x)}{Q(x)} \right) \\ &= 0.36 \ln \left(\frac{0.36}{0.333} \right) + 0.48 \ln \left(\frac{0.48}{0.333} \right) + 0.16 \ln \left(\frac{0.16}{0.333} \right) \\ &= 0.0852996 \end{aligned}$$

Interpretations [edit]

The Kullback–Leibler divergence from Q to P is often denoted $D_{\text{KL}}(P \parallel Q)$.

In the context of [machine learning](#), $D_{\text{KL}}(P \parallel Q)$ is often called the [information gain](#) achieved if Q is used instead of P . By analogy with information theory, it is also called the **relative entropy** of P with respect to Q . In the context of [coding theory](#), $D_{\text{KL}}(P \parallel Q)$ can be constructed by measuring the expected number of extra [bits](#) required to [code](#) samples from P using a code optimized for Q rather than the code optimized for P .

Cross Entropy

In [information theory](#), the **cross entropy** between two [probability distributions](#) p and q over the same underlying set of events measures the average number of [bits](#) needed to identify an event drawn from the set if a coding scheme used for the set is optimized for an estimated probability distribution q , rather than the true distribution p .

$$H(p, q) = H(p) + D_{\text{KL}}(p\|q) ,$$

where $H(p)$ is the [entropy](#) of p .

Cross Entropy

The cross entropy formula takes in two distributions, $p(x)$, the true distribution, and $q(x)$, the estimated distribution, defined over the discrete variable x and is given by

$$H(p, q) = - \sum_{\forall x} p(x) \log(q(x))$$

$$H(p, q) = H(p) + D_{\text{KL}}(p \| q)$$

$$D_{\text{KL}}(P \| Q) = \sum_{x \in \mathcal{X}} P(x) \ln\left(\frac{P(x)}{Q(x)}\right)$$

$$\begin{aligned} H(p) &= - \sum_x P(x) \cdot \log P(x) \\ &= \sum_x P(x) \cdot \log\left(\frac{1}{P(x)}\right) \end{aligned}$$

Cross Entropy: especial case

Having set up our notation, $p \in \{y, 1 - y\}$ and $q \in \{\hat{y}, 1 - \hat{y}\}$, we can use cross entropy to get a measure of dissimilarity between p and q :

$$H(p, q) = - \sum_i p_i \log q_i = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

Cross Entropy: machine learning

The cross entropy formula takes in two distributions, $p(x)$, the true distribution, and $q(x)$, the estimated distribution, defined over the discrete variable x and is given by

$$H(p, q) = - \sum_{\forall x} p(x) \log(q(x))$$

$$H(p, q) = H(p) + D_{\text{KL}}(p\|q)$$

Cross Entropy: machine learning

$$H(p, q) = - \sum_{\forall x} p(x) \log(q(x)) \rightarrow \text{try to minimize}$$

$$H(p, q) = \underbrace{H(p)}_{\text{constant}} + \underbrace{D_{\text{KL}}(p||q)}_{\rightarrow \text{try to minimize}}$$

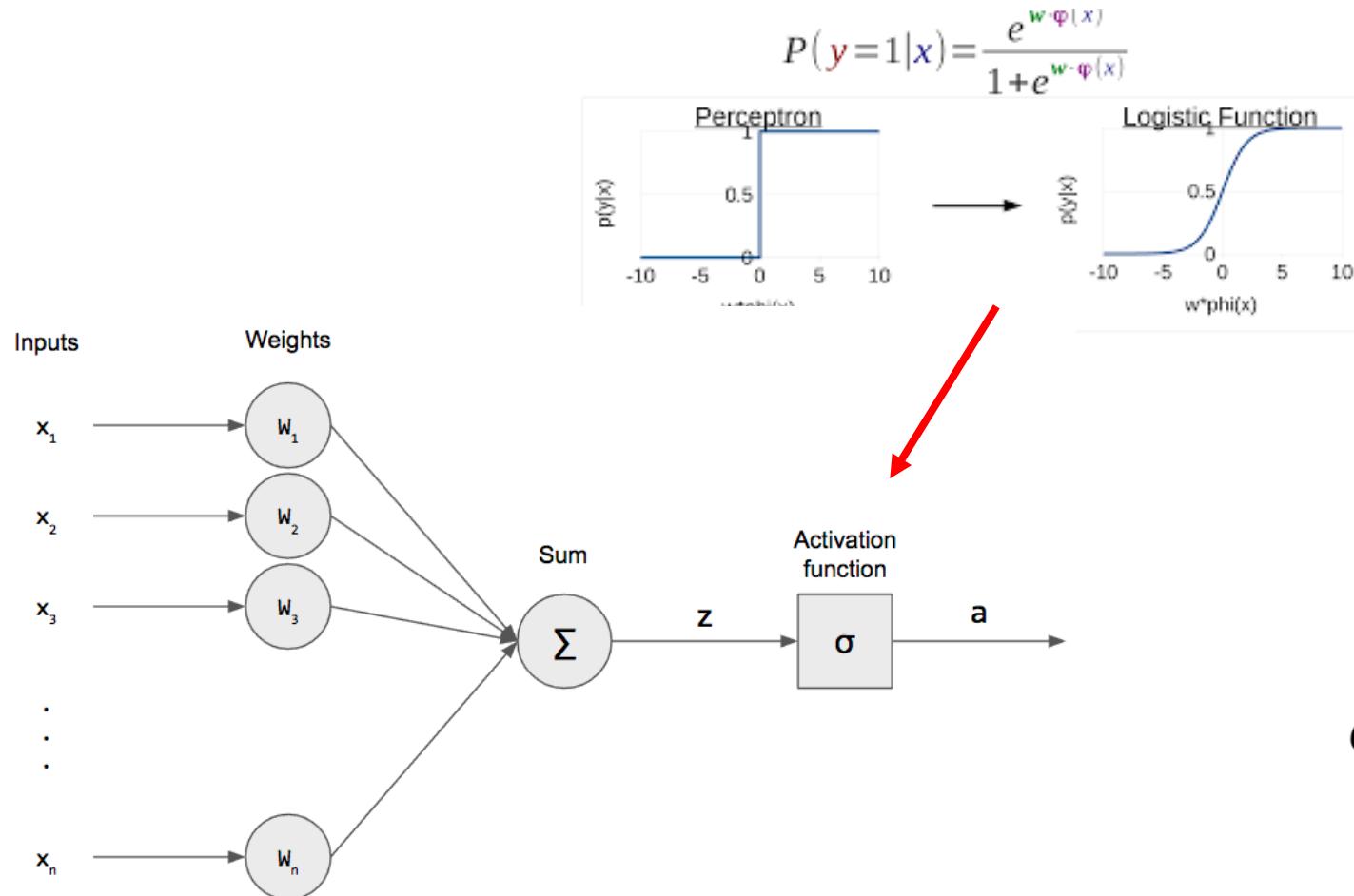
Cross Entropy: especial case and machine learning

Having set up our notation, $p \in \{y, 1 - y\}$ and $q \in \{\hat{y}, 1 - \hat{y}\}$, we can use cross entropy to get a measure of dissimilarity between p and q :

$$H(p, q) = - \sum_i p_i \log q_i = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

learning = minimize

Logistic Regression



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

Logistic Function

The goal is to predict the target class t from an input z . The probability $P(t = 1|z)$ that input z is classified as class $t = 1$ is represented by the output y of the logistic function computed as $y = \sigma(z)$. The [logistic function](#) σ is defined as:

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

We can write the probabilities that the class is $t = 1$ or $t = 0$ given input z as:

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

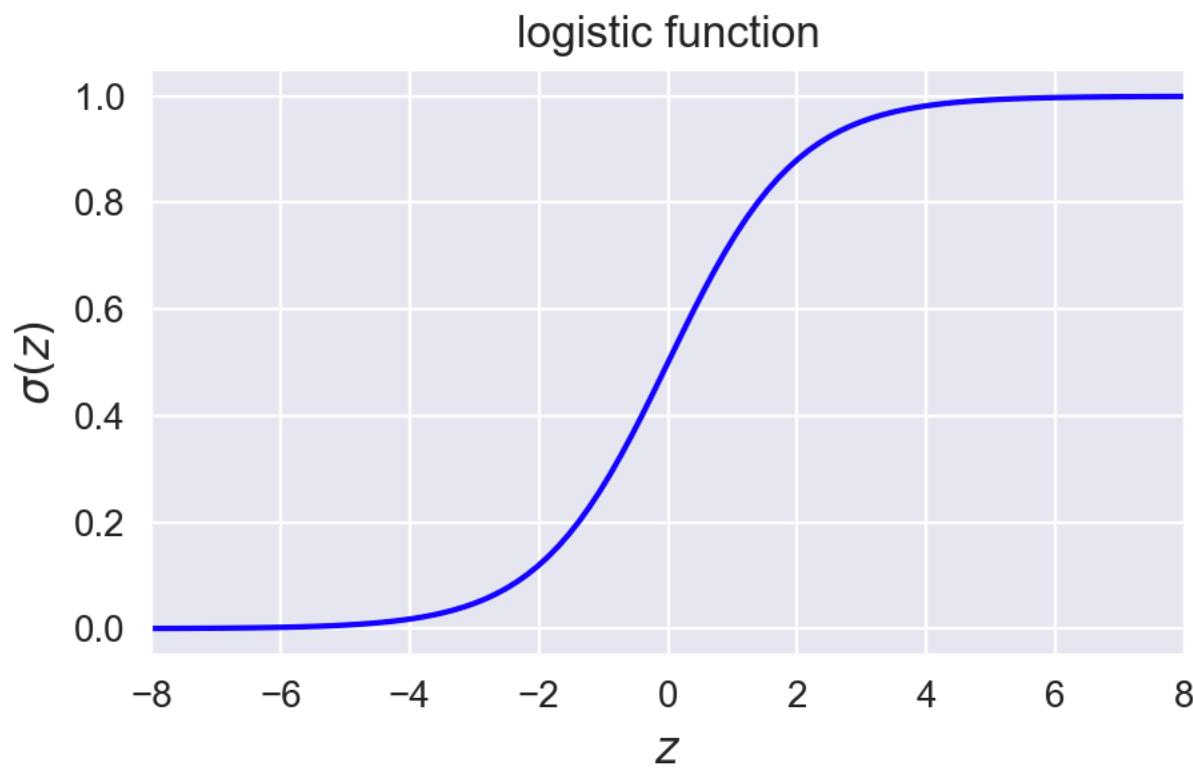
$$P(t = 0|z) = 1 - \sigma(z) = \frac{e^{-z}}{1 + e^{-z}}$$

Note that input z to the logistic function corresponds to the log **odds ratio** of $P(t = 1|z)$ over $P(t = 0|z)$.

$$\begin{aligned}\log \frac{P(t = 1|z)}{P(t = 0|z)} &= \log \frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} = \log \frac{1}{e^{-z}} \\ &= \log(1) - \log(e^{-z}) = z\end{aligned}$$

This means that the logg odds ratio $\log(P(t = 1|z)/P(t = 0|z))$ changes linearly with z . And if $z = x \cdot w$ as in neural networks, this means that the logg odds ratio changes linearly with the parameters w and input samples x .

```
def logistic(z):
    """Logistic function."""
    return 1 / (1 + np.exp(-z))
```



Derivative of the logistic function

Since neural networks typically use [gradient](#) based optimization techniques such as [gradient descent](#) it is important to define the [derivative](#) of the output y of the logistic function with respect to its input z . $\partial y / \partial z$ can be calculated as:

$$\frac{\partial y}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \frac{\partial \frac{1}{1+e^{-z}}}{\partial z} = \frac{-1}{(1+e^{-z})^2} \cdot e^{-z} \cdot -1 = \frac{1}{1+e^{-z}} \frac{e^{-z}}{1+e^{-z}}$$

And since $1 - \sigma(z) = 1 - 1/(1 + e^{-z}) = e^{-z}/(1 + e^{-z})$ this can be rewritten as:

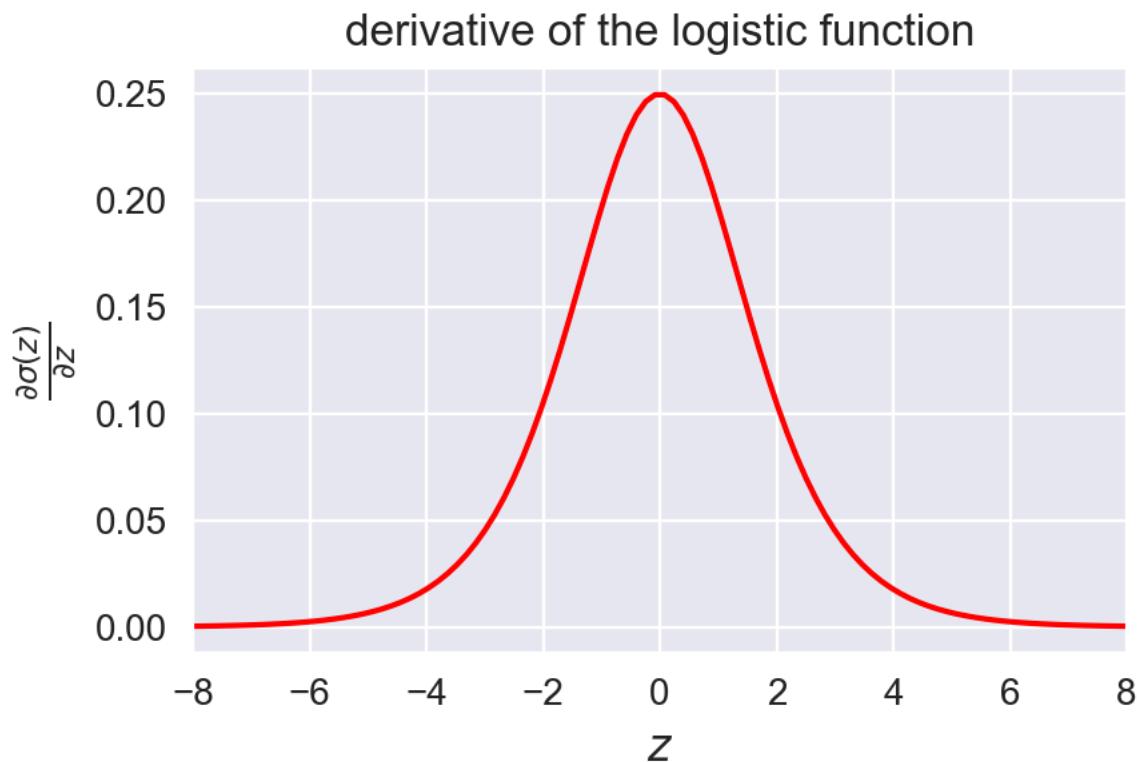
$$\frac{\partial y}{\partial z} = \frac{1}{1+e^{-z}} \frac{e^{-z}}{1+e^{-z}} = \sigma(z) \cdot (1 - \sigma(z)) = y(1 - y)$$

This derivative is implemented as [logistic_derivative\(z\)](#) and is plotted below.

```
def logistic_derivative(z):
    """Derivative of the logistic function."""
    return logistic(z) * (1 - logistic(z))
```

```
def logistic_derivative(z):
    """Derivative of the logistic function."""
    return logistic(z) * (1 - logistic(z))
```

```
# Plot the derivative of the logistic function
```



Cross-entropy loss function for the logistic function

Having set up our notation, $p \in \{y, 1 - y\}$ and $q \in \{\hat{y}, 1 - \hat{y}\}$, we can use cross entropy to get a measure of dissimilarity between p and q :

$$H(p, q) = - \sum_i p_i \log q_i = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

learning \Rightarrow minimize

Cross-entropy ~ Log likelihood

The neural network model will be optimized by maximizing the [likelihood](#) that a given set of parameters θ of the model can result in a prediction of the correct class of each input sample. The parameters θ transform each input sample i into an input to the logistic function z_i . The likelihood maximization can be written as:

$$\operatorname{argmax}_{\theta} \mathcal{L}(\theta|t, z) = \operatorname{argmax}_{\theta} \prod_{i=1}^n \mathcal{L}(\theta|t_i, z_i)$$

$$\operatorname{argmax}_{\theta} \mathcal{L}(\theta|t, z) = \operatorname{argmax}_{\theta} \prod_{i=1}^n \mathcal{L}(\theta|t_i, z_i)$$

The likelihood $\mathcal{L}(\theta|t, z)$ can be rewritten as the **joint probability** of generating t and z given the parameters θ : $P(t, z|\theta)$. Since $P(A, B) = P(A|B)P(B)$ this can be written as:

$$P(t, z|\theta) = P(t|z, \theta)P(z|\theta)$$

Since we are not interested in the probability of z we can reduce this to: $\mathcal{L}(\theta|t, z) = P(t|z, \theta) = \prod_{i=1}^n P(t_i|z_i, \theta)$. Since t_i is a **Bernoulli variable**, and the probability $P(t|z) = y$ is fixed for a given θ we can rewrite this as:

$$\begin{aligned} P(t|z) &= \prod_{i=1}^n P(t_i = 1|z_i)^{t_i} \cdot (1 - P(t_i = 1|z_i))^{1-t_i} \\ &= \prod_{i=1}^n y_i^{t_i} \cdot (1 - y_i)^{1-t_i} \end{aligned}$$

Since the logarithmic function is a monotone increasing function we can optimize the log-likelihood function $\underset{\theta}{\operatorname{argmax}} \log \mathcal{L}(\theta|t, z)$. This maximum will be the same as the maximum from the regular likelihood function. The benefit of using the log-likelihood is that it can prevent numerical **underflow** when the probabilities are low. The log-likelihood function can be written as:

$$\begin{aligned}\log \mathcal{L}(\theta|t, z) &= \log \prod_{i=1}^n y_i^{t_i} \cdot (1 - y_i)^{1-t_i} \\ &= \sum_{i=1}^n t_i \log(y_i) + (1 - t_i) \log(1 - y_i)\end{aligned}$$

→ maximize

Having set up our notation, $p \in \{y, 1 - y\}$ and $q \in \{\hat{y}, 1 - \hat{y}\}$, we can use cross entropy to get a measure of dissimilarity between p and q :

$$H(p, q) = - \sum_i p_i \log q_i = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

learning → minimize

Gradient Descent

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right]$$

Want $\min_{\theta} J(\theta)$:

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

$$\begin{aligned}
\frac{\partial}{\partial \theta} J(\theta) &= -\frac{\partial}{\partial \theta} \frac{1}{m} \sum_{i=1}^m \left(y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right) \\
&= -\frac{1}{m} \sum_{i=1}^m \left(y^{(i)} \frac{\partial}{\partial \theta} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \frac{\partial}{\partial \theta} \log(1 - h_\theta(x^{(i)})) \right)
\end{aligned}$$

$$\frac{d}{dz} \log(z) = \frac{1}{z}$$

$$\frac{d}{dz} h(z) = h(z)(1 - h(z))$$

$$\begin{aligned}
\frac{\partial \log(h(z))}{\partial \theta} &= \frac{\partial \log h(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta} \\
&= \frac{\partial \log h(z)}{\partial h(z)} \cdot \frac{\partial h(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta}
\end{aligned}$$

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

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

$$\begin{aligned}
\frac{\partial}{\partial \theta} J(\theta) &= -\frac{\partial}{\partial \theta} \frac{1}{m} \sum_{i=1}^m \left(y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right) \\
&= -\frac{1}{m} \sum_{i=1}^m \left(y^{(i)} \frac{\partial}{\partial \theta} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \frac{\partial}{\partial \theta} \log(1 - h_\theta(x^{(i)})) \right)
\end{aligned}$$

$$\frac{d}{dz} \log(z) = \frac{1}{z}$$

$$\frac{d}{dz} h(z) = h(z)(1 - h(z))$$

$$\begin{aligned}
\frac{\partial \log(h(z))}{\partial \theta} &= \frac{\partial \log h(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta} \\
&= \frac{\partial \log h(z)}{\partial h(z)} \cdot \frac{\partial h(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= -\frac{1}{m} \sum_{i=1}^m y^{(i)} \frac{x_j^{(i)}}{h(x^{(i)})} h(x^{(i)})(1 - h(x^{(i)})) + (1 - y^{(i)}) \frac{x_j^{(i)}}{1 - h(x^{(i)})} (-h(x^{(i)})(1 - h(x^{(i)}))) \\
&= -\frac{1}{m} \sum_{i=1}^m (y^{(i)} (1 - h(x^{(i)})) - (1 - y^{(i)}) h(x^{(i)})) x_j^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^m \left(y^{(i)} - y^{(i)} h(x^{(i)}) - h(x^{(i)}) + y^{(i)} h(x^{(i)}) \right) x_j^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^m \left(h(x^{(i)}) - y^{(i)} \right) x_j^{(i)}
\end{aligned}$$

Gradient Descent

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right]$$

Want $\min_{\theta} J(\theta)$:

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

(simultaneously update all θ_j)