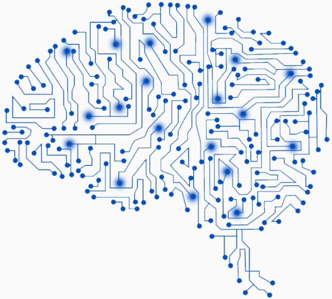
Apprentissage pour l'image
Machine learning for image processing

Multivariate logistic regression

Emile Pierret

Vendredi 28 mars 2025



Objectifs

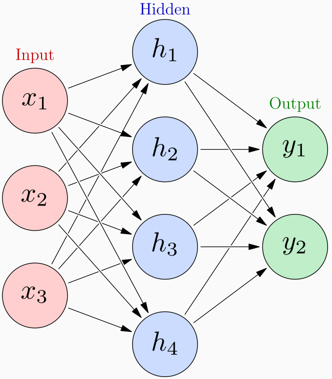
À la fin du cours :

- Comprendre ce qu'est un CNN (Réseau de Neurones Convolutifs).
- Implémenter l'entraînement d'un CNN pour la classification avec Pytorch.

Pour cette session :

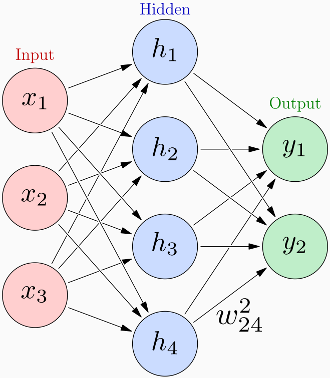
- Entraîner la dernière couche d'un réseau multicouche pour la classification avec numpy.
- Appliquer un gradient stochastique.
- Pour le moment, pas d'images, pas de convolutions.

Reminder - What is a multilayer network ?



- Inter-connection of several artificial neurons (also called nodes or units).
- Each level in the graph is called a layer:
  - Input layer,
  - Hidden layer(s),
  - Output layer.
- Each neuron in the hidden layers acts as a classifier / feature detector.
- Feedforward NN (no cycle)
  - first and simplest type of NN,
  - information moves in one direction.
- Recurrent NN (with cycle)
  - used for time sequences,
  - such as speech-recognition.

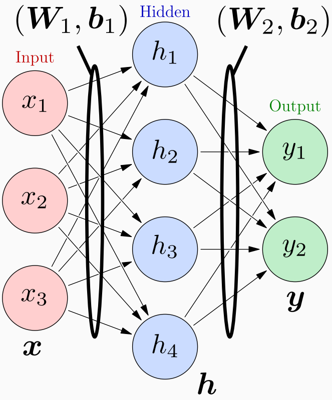
Reminder - What is a multilayer network ?



$$h_1 = g_1\left(w^1_{11}x_1 + w^1_{12}x_2 + w^1_{13}x_3 + b^1_1\right)$$
$$h_2 = g_1\left(w^1_{21}x_1 + w^1_{22}x_2 + w^1_{23}x_3 + b^1_2\right)$$
$$h_3 = g_1\left(w^1_{31}x_1 + w^1_{32}x_2 + w^1_{33}x_3 + b^1_3\right)$$
$$h_4 = g_1\left(w^1_{41}x_1 + w^1_{42}x_2 + w^1_{43}x_3 + b^1_4\right)$$

$$y_1 = g_2\left(w^2_{11}h_1 + w^2_{12}h_2 + w^2_{13}h_3 + w^2_{14}h_4 + b^2_1\right)$$
$$y_2 = g_2\left(w^2_{21}h_1 + w^2_{22}h_2 + w^2_{23}h_3 + w^2_{24}h_4 + b^2_2\right)$$

Reminder - What is a multilayer network ?



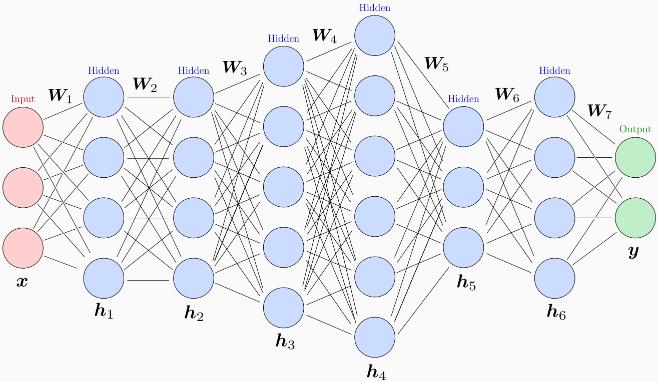
$$h_1 = g_1\left(w_{11}^1 x_1 + w_{12}^1 x_2 + w_{13}^1 x_3 + b_1^1\right)$$
$$h_2 = g_1\left(w_{21}^1 x_1 + w_{22}^1 x_2 + w_{23}^1 x_3 + b_2^1\right)$$
$$h_3 = g_1\left(w_{31}^1 x_1 + w_{32}^1 x_2 + w_{33}^1 x_3 + b_3^1\right)$$
$$h_4 = g_1\left(w_{41}^1 x_1 + w_{42}^1 x_2 + w_{43}^1 x_3 + b_4^1\right)$$

$$\boldsymbol{h} = g_1\left(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1\right)$$

$$y_1 = g_2\left(w_{11}^2 h_1 + w_{12}^2 h_2 + w_{13}^2 h_3 + w_{14}^2 h_4 + b_1^2\right)$$
$$y_2 = g_2\left(w_{21}^2 h_1 + w_{22}^2 h_2 + w_{23}^2 h_3 + w_{24}^2 h_4 + b_2^2\right)$$

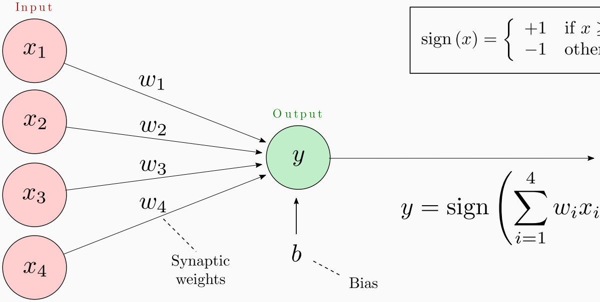
$$\boldsymbol{y} = g_2\left(\boldsymbol{W}_2 \boldsymbol{h} + \boldsymbol{b}_2\right)$$

Artificial neural network / Multilayer perceptron



Machine learning – Perceptron – Representation

Representation of the Perceptron



$$\operatorname{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$y = \operatorname{sign}\left(\sum_{i=1}^{4} w_i x_i + b\right)$$

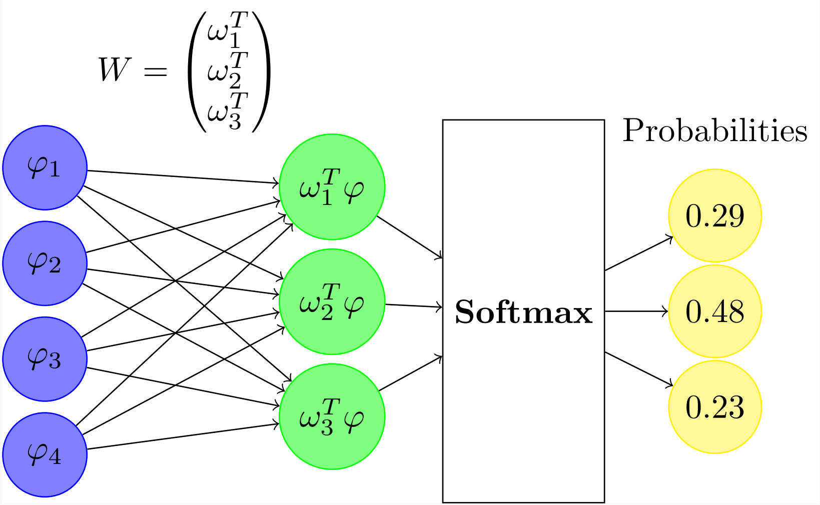
Parameters of the perceptron

- $\boldsymbol{w}_k$: synaptic weights
- b: bias

$\Big\}$ $\longleftarrow$ real parameters to be estimated.

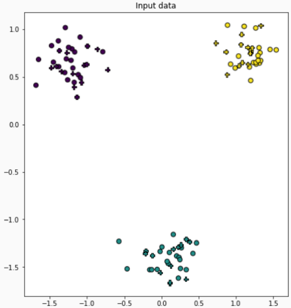
Training = adjusting the weights and biases

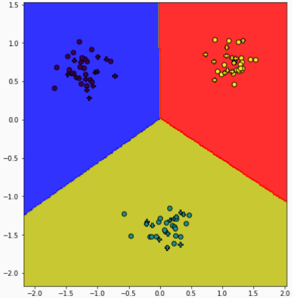
Multivariate logistic regression



Output: $y_k(\varphi) = \frac{e^{\omega_k^T \varphi}}{\sum_{j=1}^{K} e^{\omega_j^T \varphi}}$

Adapted for linearly separable data





Comme observé en TP la semaine dernière, on sait optimiser les poids $\boldsymbol{W}$ pour classifier des données linéairement séparables en maximisant une certain logvraissemblance.

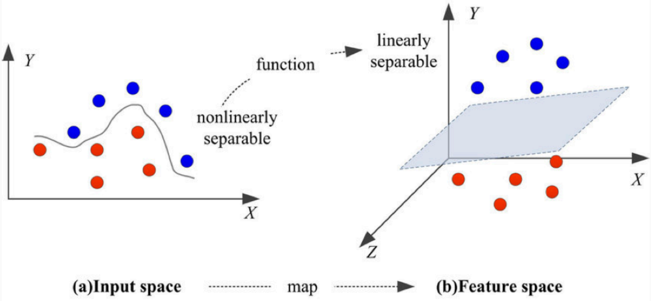
En réalité, maximiser cette logvraisemblance revient à minimiser un coût appelé "Cros-entropy".

Feature transform

- Nous appliquons une "feature transform" $\varphi : \mathbb{R}^d \to \mathbb{R}^D$ à chaque $\boldsymbol{x_n}$:

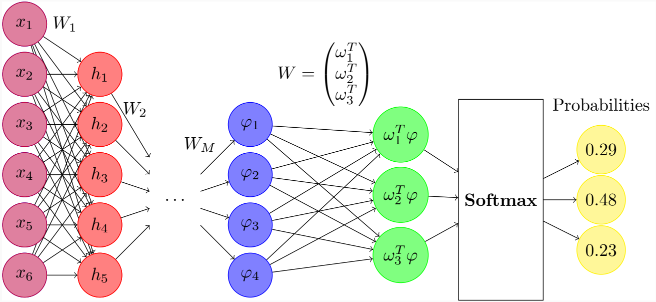
$$\varphi_n = \varphi(\boldsymbol{x_n}), \quad n = 1, \ldots, N.$$

- Selon le contexte, cela permet d'augmenter $(D > p)$ ou de diminuer $(D < p)$ la dimension de manière à favoriser la discrimination des classes .
- Il s'agit d'une application non linéaire qui devrait rendre les classes séparables linéairement.



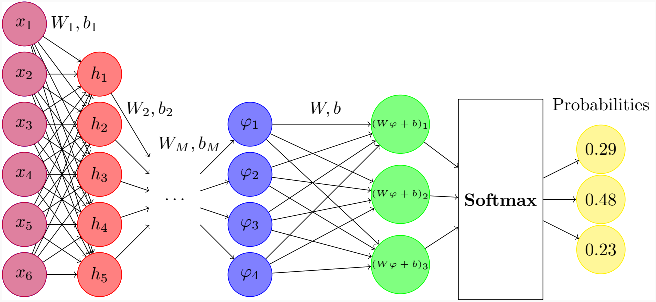
(Source: url)

Plus tard,



Paramètres : $W_1, W_2, \dots, W_M, W$
On entraînera un tel réseau à faire une 'feature transform' et classifier les données linéaires en même temps toujours en minimisant le coût 'Cross-entropy' avec une descente de gradient.

Un autre détail



Paramètres : $W_1, , b_1, W_2, b_2, \dots, W_M, b_M, W, b$

En TP aujourd'hui

- Introduction aux matrices de confusion
- Test d'un réseau de neurones non entraînés.

Questions?

Next class: Backpropagation

Slides from Charles Deledalle

Sources, images courtesy and acknowledgment

K. Chatfield	A. Horodniceanu	L. Masuch
P. Gallinari,	Y. LeCun	A. Ng
C. Hazırbaş	V. Lepetit	M. Ranzato

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ?

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

Consequently,

$$\frac{\partial z}{\partial x}$$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

Consequently,

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial w}\frac{\partial w}{\partial x}$$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

Consequently,

$$\begin{aligned} \frac{\partial z}{\partial x} &= \frac{\partial z}{\partial w} \frac{\partial w}{\partial x} \\ &= \frac{\partial z}{\partial w} \frac{\partial w}{\partial v} \frac{\partial v}{\partial x} \end{aligned}$$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

Consequently,

$$\begin{aligned}
\frac{\partial z}{\partial x} &= \frac{\partial z}{\partial w}\frac{\partial w}{\partial x} \\
&= \frac{\partial z}{\partial w}\frac{\partial w}{\partial v}\frac{\partial v}{\partial x} \\
&= 1 \times
\end{aligned}$$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

Consequently,

$$\begin{aligned} \frac{\partial z}{\partial x} &= \frac{\partial z}{\partial w}\frac{\partial w}{\partial x} \\ &= \frac{\partial z}{\partial w}\frac{\partial w}{\partial v}\frac{\partial v}{\partial x} \\ &= 1 \times \frac{1}{v} \times \end{aligned}$$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

Consequently,

$$\begin{aligned}
\frac{\partial z}{\partial x} &= \frac{\partial z}{\partial w}\frac{\partial w}{\partial x} \\
&= \frac{\partial z}{\partial w}\frac{\partial w}{\partial v}\frac{\partial v}{\partial x} \\
&= 1 \times \frac{1}{v} \times y
\end{aligned}$$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

Consequently,

$$\begin{aligned}
\frac{\partial z}{\partial x} &= \frac{\partial z}{\partial w} \frac{\partial w}{\partial x} \\
&= \frac{\partial z}{\partial w} \frac{\partial w}{\partial v} \frac{\partial v}{\partial x} \\
&= 1 \times \frac{1}{v} \times y \\
&= \frac{y}{v}
\end{aligned}$$

A simple example

Let consider $f : (x, y) \in \mathbb{R}^2 \mapsto \log(xy)$. How to differentiate f step by step with respect to x ? We can split f such that :

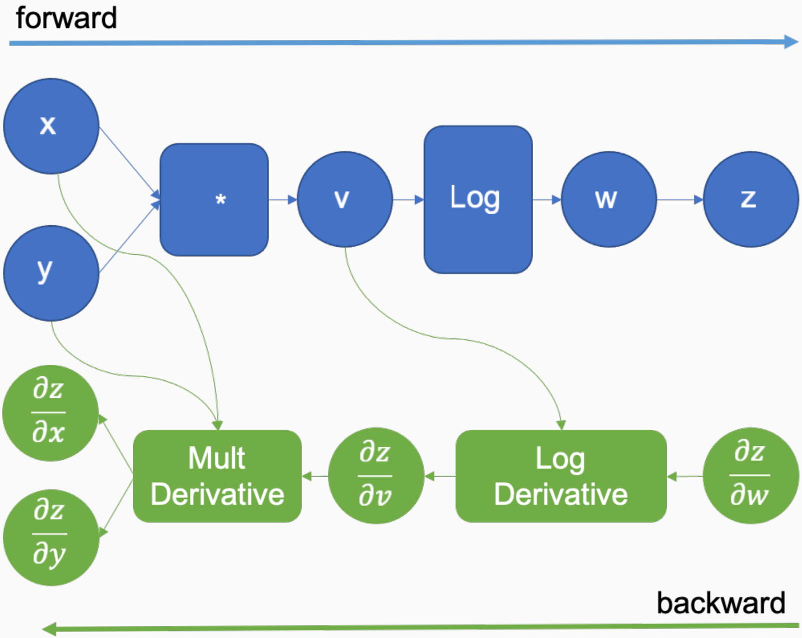
1. $v = xy$
2. $w = \log(v) = \log(xy)$
3. $z = w$

As a reminder, for $g, h : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

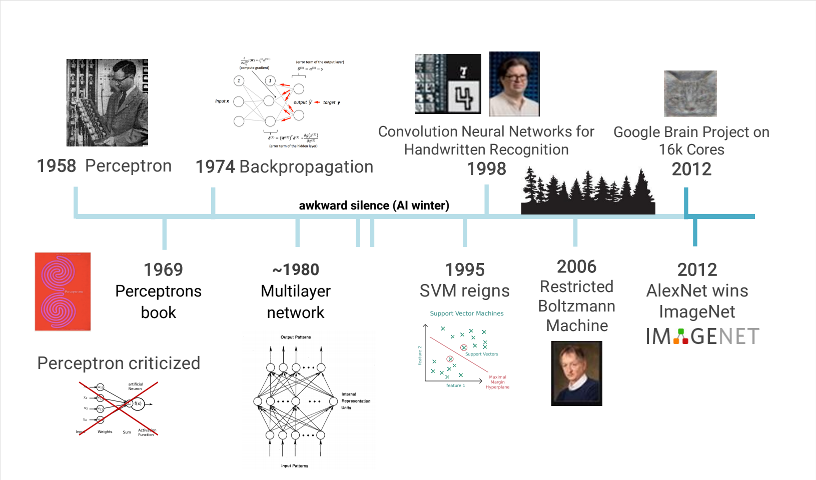
Consequently,

$$\begin{aligned}
\frac{\partial z}{\partial x} &= \frac{\partial z}{\partial w}\frac{\partial w}{\partial x} \\
&= \frac{\partial z}{\partial w}\frac{\partial w}{\partial v}\frac{\partial v}{\partial x} \\
&= 1 \times \frac{1}{v} \times y \\
&= \frac{y}{v} \\
&= \frac{1}{x}
\end{aligned}$$


forward
x
y
*
v
Log
w
z
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
Mult Derivative
$\frac{\partial z}{\partial v}$
Log Derivative
$\frac{\partial z}{\partial w}$
backward

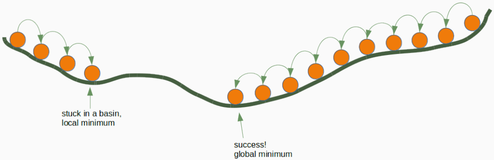
Machine learning – Timeline

Timeline of (deep) learning

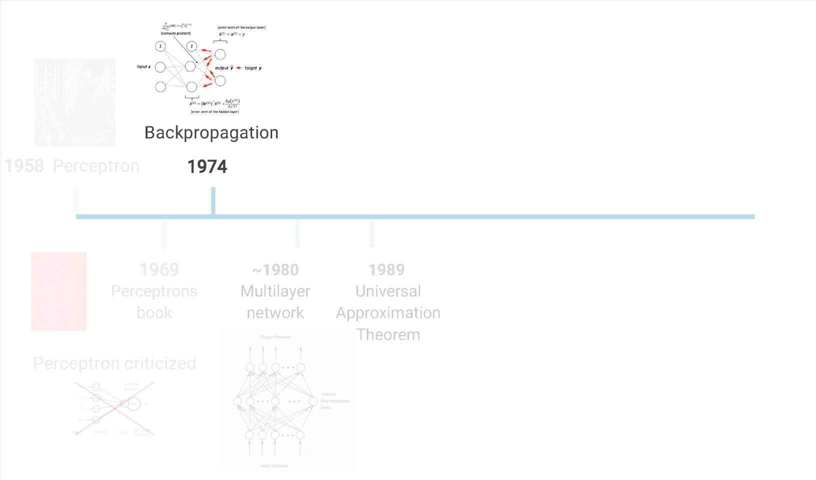


(Source: Lucas Masuch & Vincent Lepetit)

Backpropagation

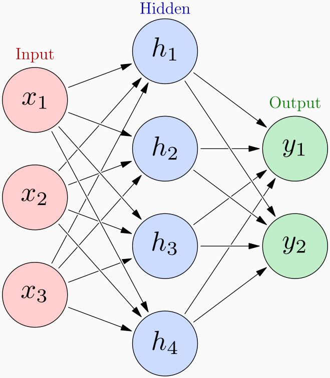


Learning with backpropagation



(Source: Lucas Masuch & Vincent Lepetit)

Artificial neural network / Multilayer perceptron / NeuralNet



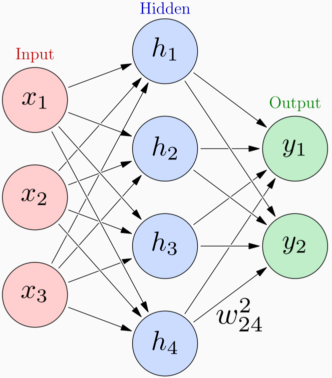
$$h_1 = g_1\left(w_{11}^1 x_1 + w_{12}^1 x_2 + w_{13}^1 x_3 + b_1^1\right)$$
$$h_2 = g_1\left(w_{21}^1 x_1 + w_{22}^1 x_2 + w_{23}^1 x_3 + b_2^1\right)$$
$$h_3 = g_1\left(w_{31}^1 x_1 + w_{32}^1 x_2 + w_{33}^1 x_3 + b_3^1\right)$$
$$h_4 = g_1\left(w_{41}^1 x_1 + w_{42}^1 x_2 + w_{43}^1 x_3 + b_4^1\right)$$

$$y_1 = g_2\left(w_{11}^2 h_1 + w_{12}^2 h_2 + w_{13}^2 h_3 + w_{14}^2 h_4 + b_1^2\right)$$
$$y_2 = g_2\left(w_{21}^2 h_1 + w_{22}^2 h_2 + w_{23}^2 h_3 + w_{24}^2 h_4 + b_2^2\right)$$

w_{ij}^k synaptic weight between previous node j and next node i at layer k.

g_k are any activation function applied to each coefficient of its input vector.

Artificial neural network / Multilayer perceptron / NeuralNet



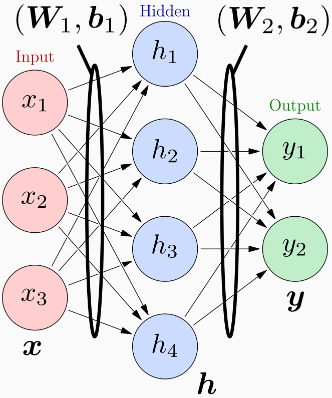
$$h_1 = g_1\left(w^1_{11}x_1 + w^1_{12}x_2 + w^1_{13}x_3 + b^1_1\right)$$
$$h_2 = g_1\left(w^1_{21}x_1 + w^1_{22}x_2 + w^1_{23}x_3 + b^1_2\right)$$
$$h_3 = g_1\left(w^1_{31}x_1 + w^1_{32}x_2 + w^1_{33}x_3 + b^1_3\right)$$
$$h_4 = g_1\left(w^1_{41}x_1 + w^1_{42}x_2 + w^1_{43}x_3 + b^1_4\right)$$

$$y_1 = g_2\left(w^2_{11}h_1 + w^2_{12}h_2 + w^2_{13}h_3 + w^2_{14}h_4 + b^2_1\right)$$
$$y_2 = g_2\left(w^2_{21}h_1 + w^2_{22}h_2 + w^2_{23}h_3 + w^2_{24}h_4 + b^2_2\right)$$

w^k_{ij} synaptic weight between previous node j and next node i at layer k.

g_k are any activation function applied to each coefficient of its input vector.

Artificial neural network / Multilayer perceptron / NeuralNet



$$h_1 = g_1\left(w^1_{11}x_1 + w^1_{12}x_2 + w^1_{13}x_3 + b^1_1\right)$$
$$h_2 = g_1\left(w^1_{21}x_1 + w^1_{22}x_2 + w^1_{23}x_3 + b^1_2\right)$$
$$h_3 = g_1\left(w^1_{31}x_1 + w^1_{32}x_2 + w^1_{33}x_3 + b^1_3\right)$$
$$h_4 = g_1\left(w^1_{41}x_1 + w^1_{42}x_2 + w^1_{43}x_3 + b^1_4\right)$$

$$\boldsymbol{h} = g_1\left(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1\right)$$

$$y_1 = g_2\left(w^2_{11}h_1 + w^2_{12}h_2 + w^2_{13}h_3 + w^2_{14}h_4 + b^2_1\right)$$
$$y_2 = g_2\left(w^2_{21}h_1 + w^2_{22}h_2 + w^2_{23}h_3 + w^2_{24}h_4 + b^2_2\right)$$

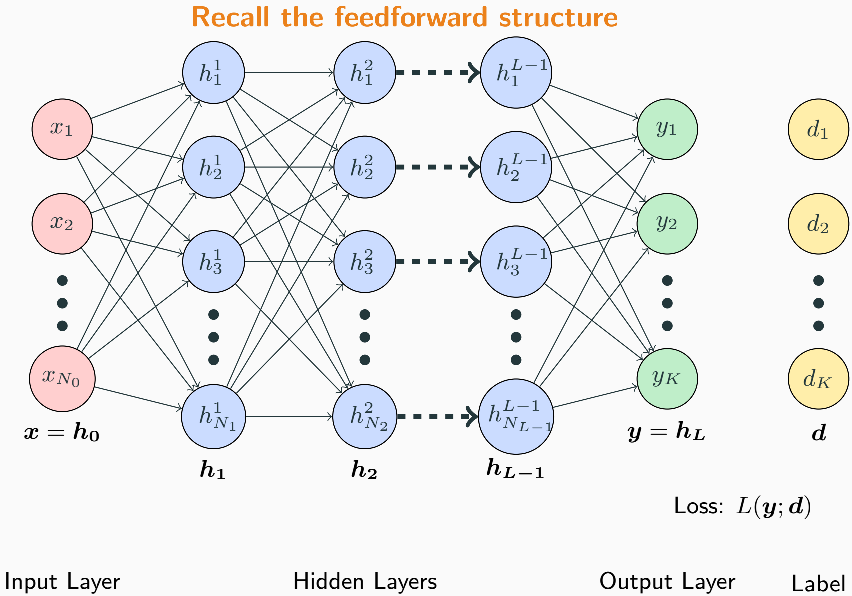
$$\boldsymbol{y} = g_2\left(\boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2\right)$$

w^k_{ij} synaptic weight between previous node j and next node i at layer k.

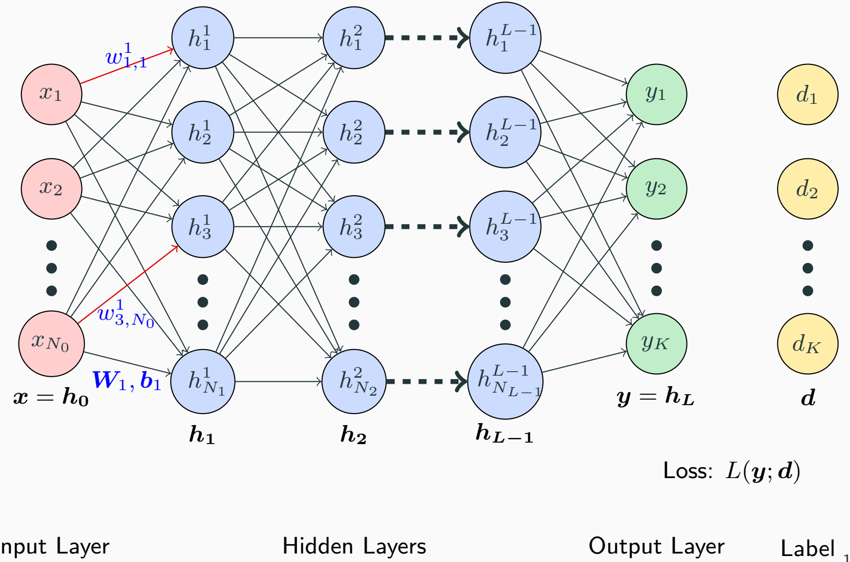
g_k are any activation function applied to each coefficient of its input vector.

The matrices $\boldsymbol{W_k}$ and biases $\boldsymbol{b}_k$ are learned from labeled training data.

Feedforward Artificial Neural Network

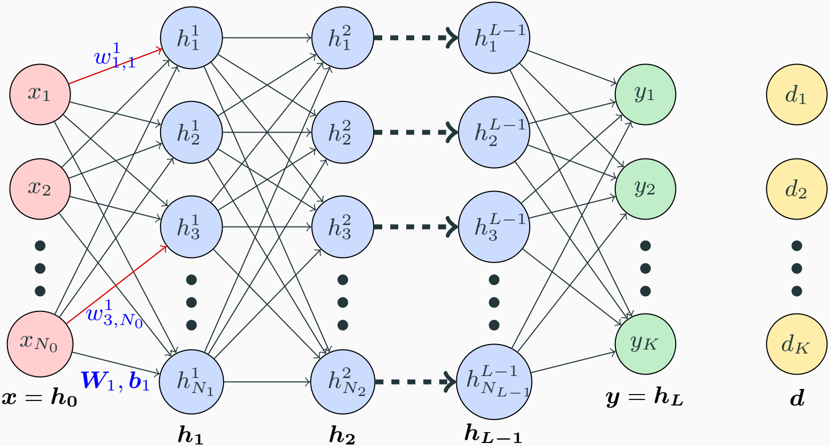


Recall the feedforward structure



Feedforward Artificial Neural Network

Recall the feedforward structure

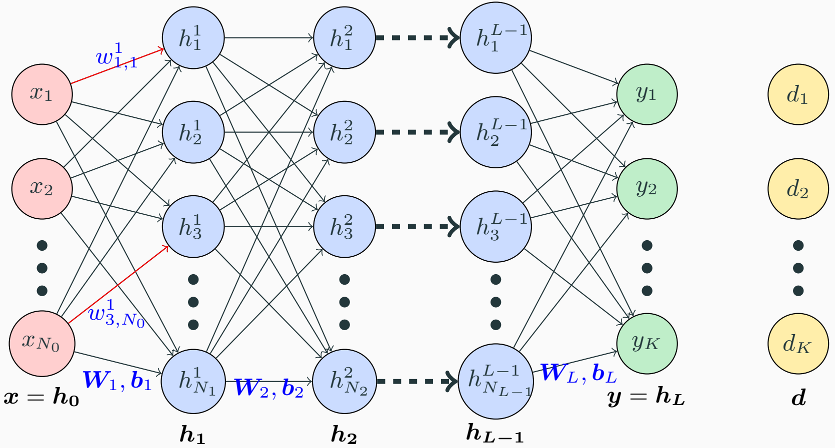


$= g_1\left(\boldsymbol{W}_1 \boldsymbol{h_0} + \boldsymbol{b}_1\right)$

Loss: $L(\boldsymbol{y}; \boldsymbol{d})$

Input Layer　　Hidden Layers　　Output Layer　　Label

Recall the feedforward structure



$= g_1\left(\boldsymbol{W}_1 \boldsymbol{h_0} + \boldsymbol{b}_1\right)$

Loss: $L(\boldsymbol{y}; \boldsymbol{d})$

Input Layer | Hidden Layers | Output Layer | Label