

Deep Learning - Theory and Practice

IE 643
Lecture 5

August 11, 2025.

- 1 Recap
 - Perceptron Convergence
- 2 Moving on from Perceptron
- 3 Multi Layer Perceptron
 - MLP-Data Perspective

Recap: Convergence of Perceptron Training

Perceptron Convergence - Separability Assumption

Linear Separability Assumption

Let $D = \{(x^t, y^t)\}_{t=1}^{\infty}$ denote the training data where $x^t \in \mathbb{R}^d$, $y^t \in \{+1, -1\}$, $\forall t = 1, 2, \dots$. Then there exist $\mathbb{R}^d \ni w^* \neq 0$, $\gamma > 0$, such that:

$$\begin{aligned}\langle w^*, x^t \rangle &> \gamma \text{ where } y^t = 1, \\ \langle w^*, x^t \rangle &< -\gamma \text{ where } y^t = -1.\end{aligned}$$

Perceptron Convergence - Separability Assumption

Linear Separability Assumption

Let $D = \{(x^t, y^t)\}_{t=1}^{\infty}$ denote the training data where $x^t \in \mathbb{R}^d$, $y^t \in \{+1, -1\}$, $\forall t = 1, 2, \dots$. Then there exist $\mathbb{R}^d \ni w^* \neq 0$, $\gamma > 0$, such that:

$$y^t \langle w^*, x^t \rangle > \gamma.$$

Perceptron Convergence - Mistake Bound

- We will try to derive useful bounds on the number of mistakes that a perceptron can commit during its training.
- **Assumption on data:** Linear Separability
- Assume that the T rounds of training have been completed in perceptron training. Assume T to be some large number.
- Assume that M mistakes are made by the perceptron in these T rounds. (Obviously, $M \leq T$.)
- We ask if the number of mistakes M can be bounded by some suitable quantity.

Perceptron Convergence - Mistake Bound

Recall: Initially we got the lower bound for the inner product $\langle w^*, w^{T+1} \rangle$ as:

$$\langle w^*, w^{T+1} \rangle > M\gamma$$

Perceptron Convergence - Mistake Bound

Recall: We wanted to handle the inner product term:

$$\langle w^*, w^{T+1} \rangle > M\gamma$$

Perceptron Convergence - Mistake Bound

Recall: We wanted to handle the inner product term:

$$\langle w^*, w^{T+1} \rangle > M\gamma$$

Then using Cauchy-Schwarz inequality we had

$$M\gamma < \langle w^*, w^{T+1} \rangle \leq \|w^*\|_2 \|w^{T+1}\|_2$$

Perceptron Convergence - Mistake Bound

Recall: We wanted to handle the inner product term:

$$\langle w^*, w^{T+1} \rangle > M\gamma$$

Then using Cauchy-Schwarz inequality we had

$$M\gamma < \langle w^*, w^{T+1} \rangle \leq \|w^*\|_2 \|w^{T+1}\|_2$$

Using the bound $\|w^{T+1}\|_2^2 \leq MR^2$ we obtain:

$$\begin{aligned} M^2\gamma^2 &< \|w^*\|_2^2 MR^2 \\ \implies M &< \frac{\|w^*\|_2^2 R^2}{\gamma^2} \end{aligned}$$

Perceptron Convergence - Mistake Bound

Recall: We wanted to handle the inner product term:

$$\langle w^*, w^{T+1} \rangle > M\gamma$$

Then using Cauchy-Schwarz inequality we had

$$M\gamma < \langle w^*, w^{T+1} \rangle \leq \|w^*\|_2 \|w^{T+1}\|_2$$

Using the bound $\|w^{T+1}\|_2^2 \leq MR^2$ we obtain:

$$\begin{aligned} M^2\gamma^2 &< \|w^*\|_2^2 MR^2 \\ \implies M &< \frac{\|w^*\|_2^2 R^2}{\gamma^2} \end{aligned}$$

Thus, assuming that $\|w^*\|_2$ and R can be controlled, the number of mistakes M is inversely proportional to γ^2 , which determines the closeness of the data points to the separating hyperplane.

Perceptron - Caveat

- Not suitable when **linear separability assumption** fails
- Example: Classical XOR problem



Perceptron - Caveat

- Not suitable when **linear separability assumption** fails
- Example: Classical XOR problem



Heavily criticized by **M. Minsky** and **S. Papert** in their book: **Perceptrons**, *MIT Press*, 1969.

Perceptron - Caveat

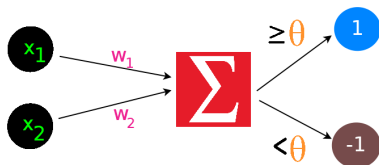
- Not suitable when **linear separability assumption** fails
- Example: Classical XOR problem



x_1	x_2	$y = x_1 \oplus x_2$
0	0	-1
0	1	1
1	0	1
1	1	-1

Perceptron - Caveat

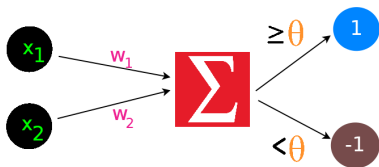
- Not suitable when **linear separability assumption** fails
- Example: Classical XOR problem



x_1	x_2	$y = x_1 \oplus x_2$	$\hat{y} = \text{sign}(w_1x_1 + w_2x_2 - \theta)$
0	0	-1	$\text{sign}(-\theta)$
0	1	1	$\text{sign}(w_2 - \theta)$
1	0	1	$\text{sign}(w_1 - \theta)$
1	1	-1	$\text{sign}(w_1 + w_2 - \theta)$

Perceptron - Caveat

- Not suitable when **linear separability assumption** fails
- Example: Classical XOR problem



$$\text{sign}(-\theta) = -1 \implies \theta > 0$$

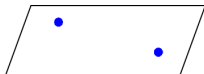
$$\text{sign}(w_2 - \theta) = 1 \implies w_2 - \theta \geq 0$$

$$\text{sign}(w_1 - \theta) = 1 \implies w_1 - \theta \geq 0$$

$$\text{sign}(w_1 + w_2 - \theta) = -1 \implies -w_1 - w_2 + \theta > 0$$

Note: This system is inconsistent. (Homework!)

Moving away from perceptron - Dealing with XOR problem



Moving away from perceptron - Dealing with XOR problem



- Assume that the sample features $x \in \mathbb{R}^d$.

Moving away from perceptron - Dealing with XOR problem



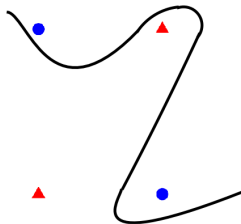
- Assume that the sample features $x \in \mathbb{R}^d$.
- **Idea:** Use a transformation $\psi : \mathbb{R}^d \rightarrow \mathbb{R}^q$, where $q \gg d$, to lift the data samples $x \in \mathbb{R}^d$ into $\psi(x) \in \mathbb{R}^q$ hoping to see a separating hyperplane in the transformed space.

Moving away from perceptron - Dealing with XOR problem

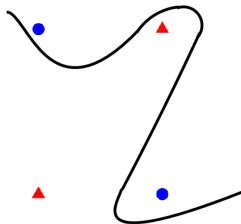


- Assume that the sample features $x \in \mathbb{R}^d$.
- **Idea:** Use a transformation $\psi : \mathbb{R}^d \rightarrow \mathbb{R}^q$, where $q \gg d$, to lift the data samples $x \in \mathbb{R}^d$ into $\psi(x) \in \mathbb{R}^q$ hoping to see a separating hyperplane in the transformed space.
- Forms the core idea behind kernel methods. (Will not be pursued in this course!)

Moving away from perceptron - Dealing with XOR problem

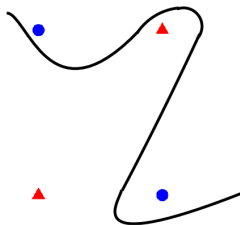


Moving away from perceptron - Dealing with XOR problem



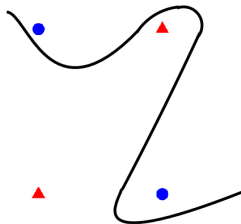
- **Idea:** The separating surface need not be linear and can be assumed to take some non-linear form.

Moving away from perceptron - Dealing with XOR problem



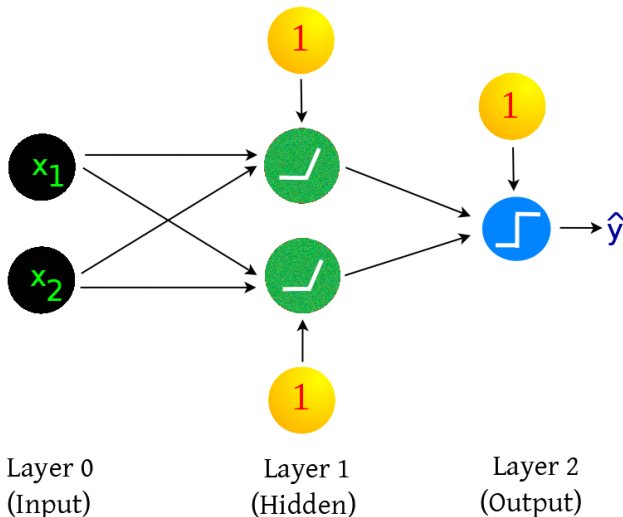
- **Idea:** The separating surface need not be linear and can be assumed to take some non-linear form.
- Hence for an input space \mathcal{X} and output space \mathcal{Y} , the learned map $h : \mathcal{X} \rightarrow \mathcal{Y}$ can take some non-linear form.

Moving away from perceptron - Dealing with XOR problem

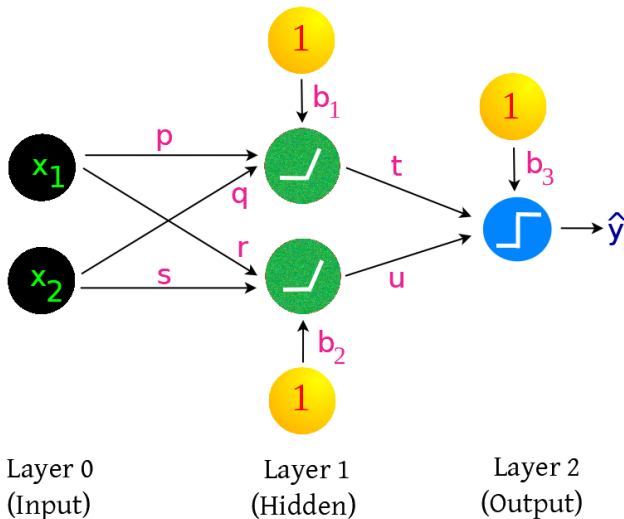


- **Idea:** The separating surface need not be linear and can be assumed to take some non-linear form.
- Hence for an input space \mathcal{X} and output space \mathcal{Y} , the learned map $h : \mathcal{X} \rightarrow \mathcal{Y}$ can take some non-linear form.
- Forms the idea behind multi-layer perceptrons!

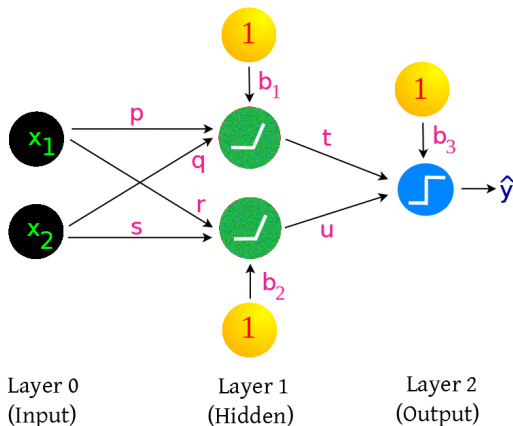
Moving away from perceptron - Dealing with XOR problem



Moving away from perceptron - Dealing with XOR problem



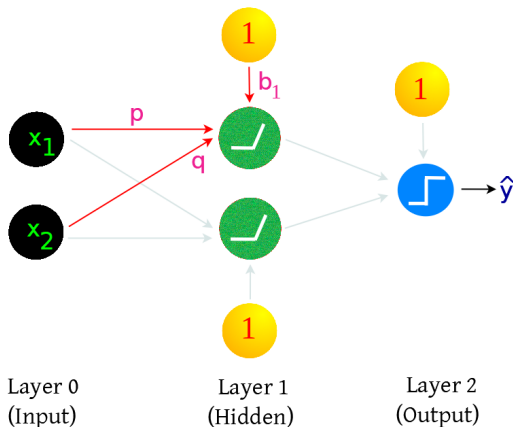
Moving away from perceptron - Dealing with XOR problem



Some notations

- n_k^ℓ denotes k -th neuron at layer ℓ .
- a_k^ℓ denotes the activation of the neuron n_k^ℓ .

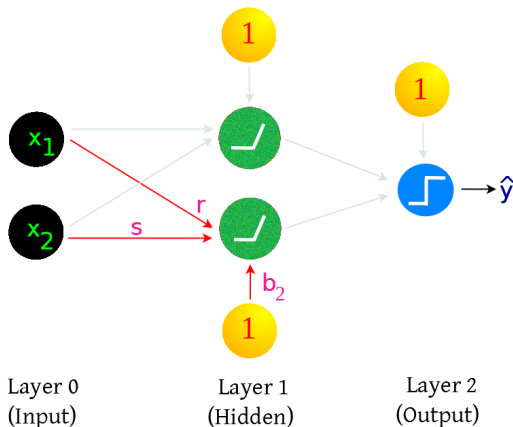
Moving away from perceptron - Dealing with XOR problem



- Activation at neuron n_1^1 :

$$a_1^1 = \max\{px_1 + qx_2 + b_1, 0\}.$$

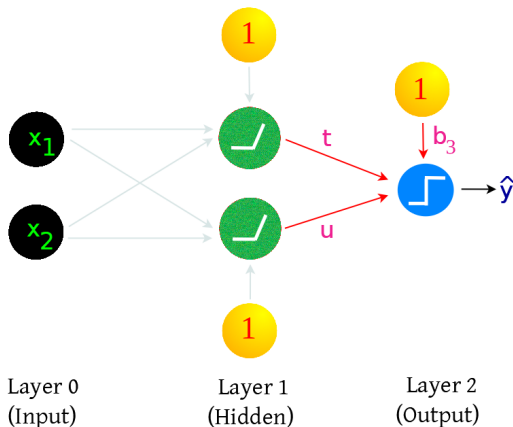
Moving away from perceptron - Dealing with XOR problem



- Activation at neuron n_2^1 :

$$a_2^1 = \max\{rx_1 + sx_2 + b_2, 0\}.$$

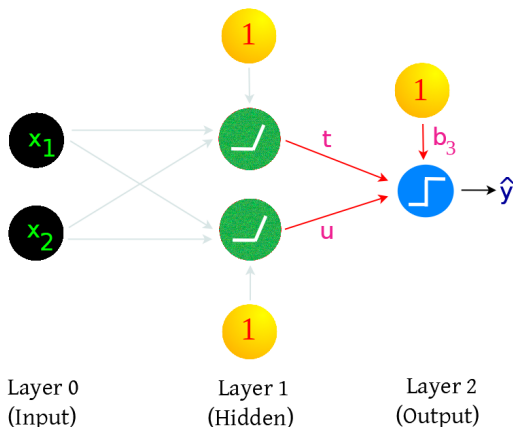
Moving away from perceptron - Dealing with XOR problem



- Activation at neuron n_1^2 :

$$a_1^2 = \text{sign}(ta_1^1 + ua_2^1 + b_3).$$

Moving away from perceptron - Dealing with XOR problem

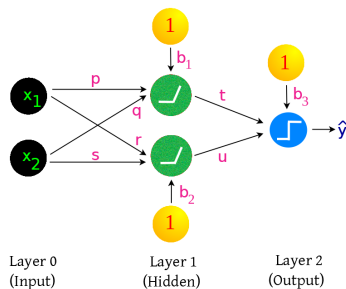


- Activation at neuron n_1^2 :

$$a_1^2 = \text{sign}(ta_1^1 + ua_2^1 + b_3).$$

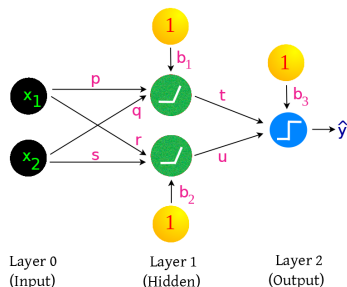
- Note:** The activation a_1^2 is the output of the network denoted by \hat{y} .

Moving away from perceptron - Dealing with XOR problem



x_1	x_2	a_1^1	a_2^1	\hat{y}	y
0	0	$\max\{b_1, 0\}$	$\max\{b_2, 0\}$	$\text{sign}(ta_1^1 + ua_2^1 + b_3)$	-1
0	1	$\max\{q + b_1, 0\}$	$\max\{s + b_2, 0\}$	$\text{sign}(ta_1^1 + ua_2^1 + b_3)$	+1
1	0	$\max\{p + b_1, 0\}$	$\max\{r + b_2, 0\}$	$\text{sign}(ta_1^1 + ua_2^1 + b_3)$	+1
1	1	$\max\{p + q + b_1, 0\}$	$\max\{r + s + b_2, 0\}$	$\text{sign}(ta_1^1 + ua_2^1 + b_3)$	-1

Moving away from perceptron - Dealing with XOR problem



x_1	x_2	a_1^1	a_2^1	\hat{y}	y
0	0	$\max\{b_1, 0\}$	$\max\{b_2, 0\}$	$\text{sign}(ta_1^1 + ua_2^1 + b_3)$	-1
0	1	$\max\{q + b_1, 0\}$	$\max\{s + b_2, 0\}$	$\text{sign}(ta_1^1 + ua_2^1 + b_3)$	+1
1	0	$\max\{p + b_1, 0\}$	$\max\{r + b_2, 0\}$	$\text{sign}(ta_1^1 + ua_2^1 + b_3)$	+1
1	1	$\max\{p + q + b_1, 0\}$	$\max\{r + s + b_2, 0\}$	$\text{sign}(ta_1^1 + ua_2^1 + b_3)$	-1

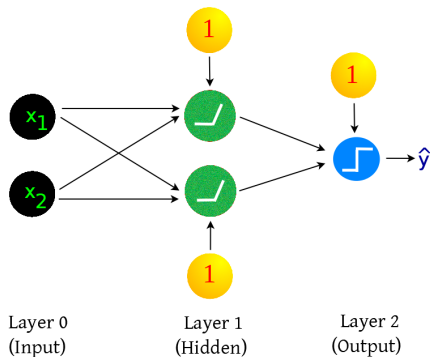
Homework: Find weights $p, q, r, s, t, u, b_1, b_2, b_3$ such that the MLP solves the XOR problem.

Moving away from perceptron - Dealing with XOR problem

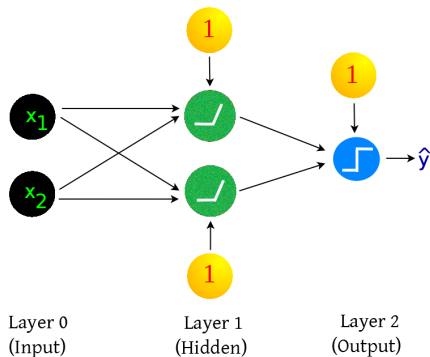
A different Multi Layer Perceptron (MLP) architecture is given for XOR problem in:

- David. E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams.
Learning Internal Representations by Error Propagation,
Technical Report, UCSD, 1985.

Multi Layer Perceptron

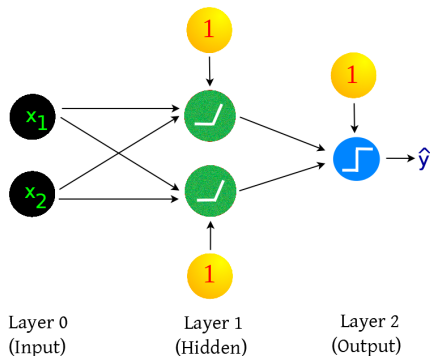


Multi Layer Perceptron



Notable features:

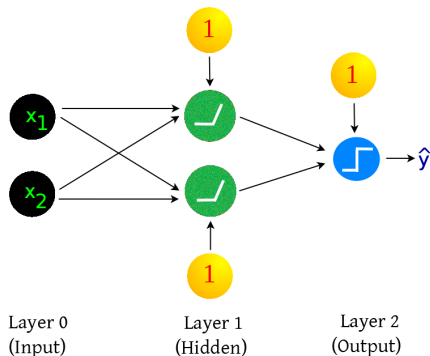
Multi Layer Perceptron



Notable features:

- Multiple layers stacked together.

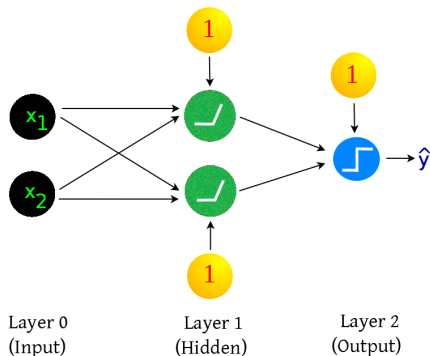
Multi Layer Perceptron



Notable features:

- Multiple layers stacked together.
- Zero-th layer usually called input layer.

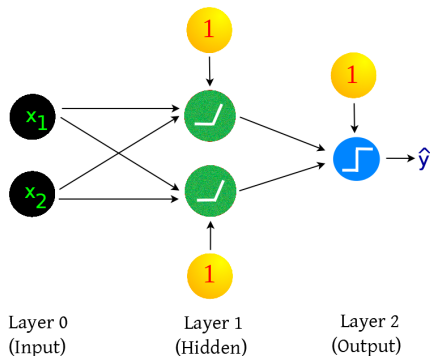
Multi Layer Perceptron



Notable features:

- Multiple layers stacked together.
- Zero-th layer usually called input layer.
- Final layer usually called output layer.

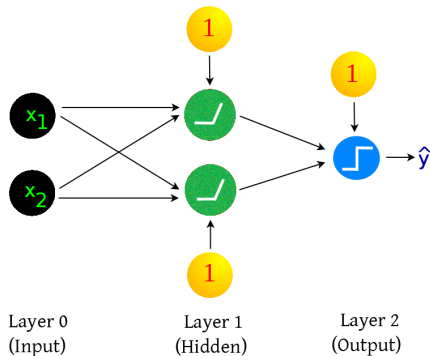
Multi Layer Perceptron



Notable features:

- Multiple layers stacked together.
- Zero-th layer usually called input layer.
- Final layer usually called output layer.
- Intermediate layers are called hidden layers.

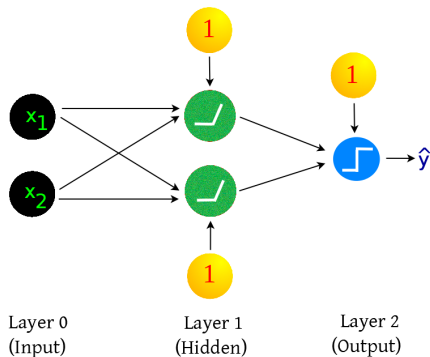
Multi Layer Perceptron



Notable features:

- Each neuron in the hidden and output layer is like a perceptron.

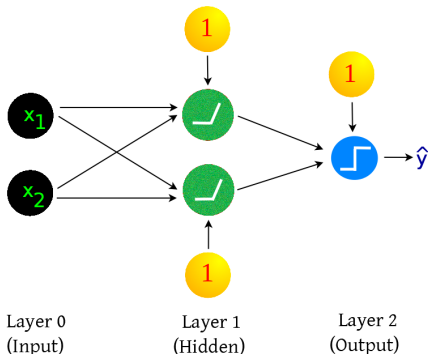
Multi Layer Perceptron



Notable features:

- Each neuron in the hidden and output layer is like a perceptron.
- However, unlike perceptron, different activation functions are used.

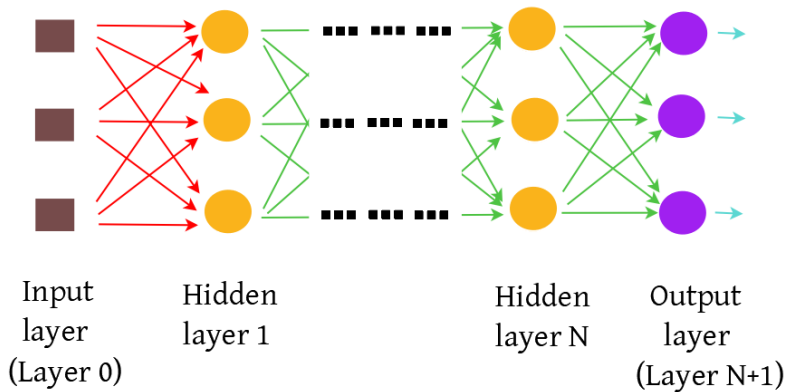
Multi Layer Perceptron



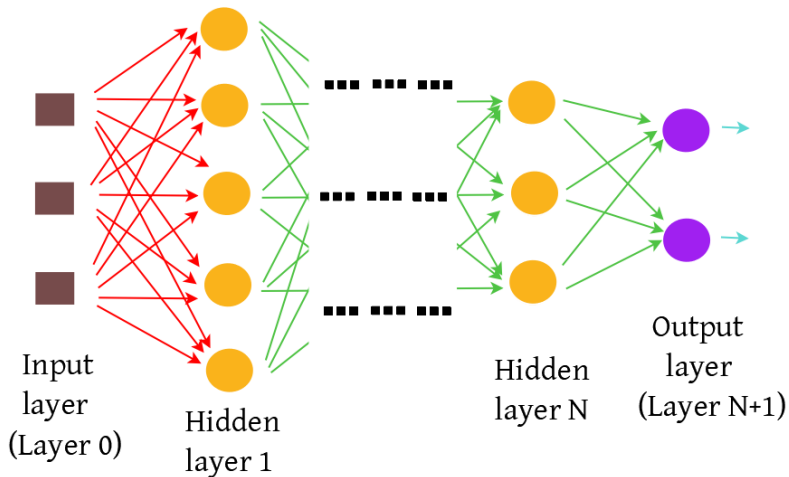
Notable features:

- Each neuron in the hidden and output layer is like a perceptron.
- However, unlike perceptron, different activation functions are used.
- $\max\{x, 0\}$ has a special name called **ReLU (Rectified Linear Unit)**.

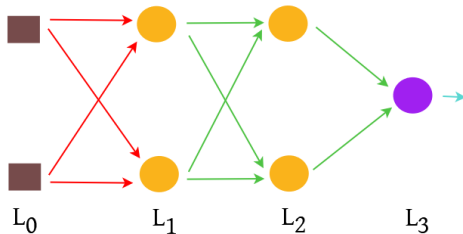
Multi Layer Perceptron



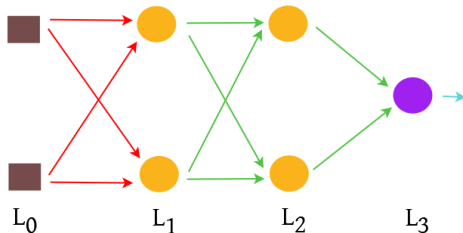
Multi Layer Perceptron



Multi Layer Perceptron - More notations

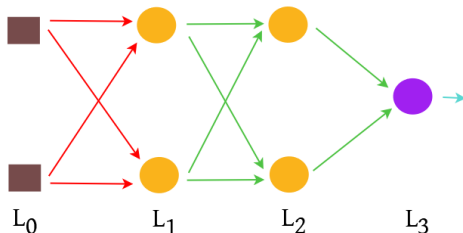


Multi Layer Perceptron - More notations



- This MLP contains an input layer L_0 , 2 hidden layers denoted by L_1, L_2 , and output layer L_3 .

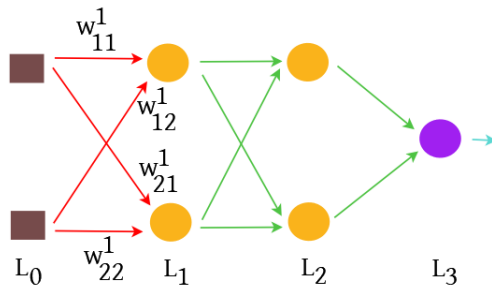
Multi Layer Perceptron - More notations



Recall:

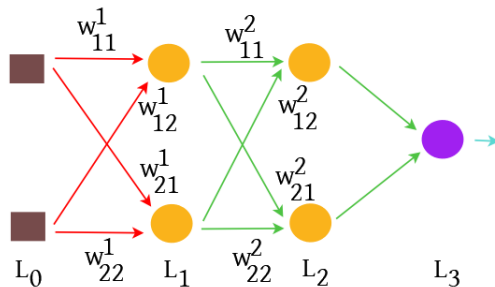
- n_k^ℓ denotes k -th neuron at ℓ -th layer.
- a_k^ℓ denotes activation of neuron n_k^ℓ .

Multi Layer Perceptron - More notations



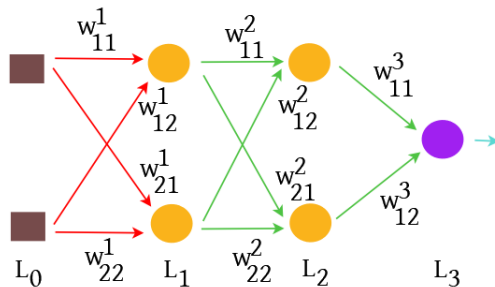
- w_{ij}^ℓ denotes weight of connection connecting n_i^ℓ from $n_j^{\ell-1}$.

Multi Layer Perceptron - More notations



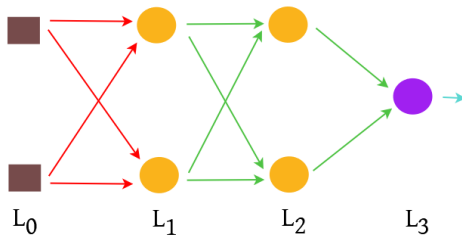
- w_{ij}^ℓ denotes weight of connection connecting n_i^ℓ from $n_j^{\ell-1}$.

Multi Layer Perceptron - More notations



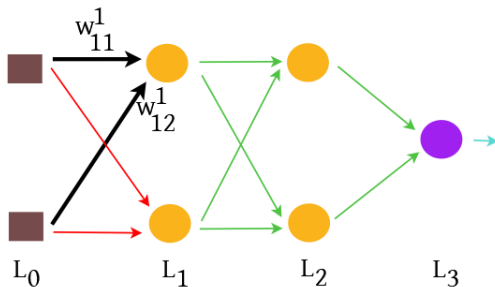
- w_{ij}^ℓ denotes weight of connection connecting n_i^ℓ from $n_j^{\ell-1}$.

Multi Layer Perceptron - More notations



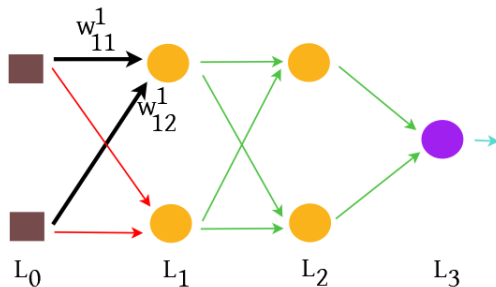
- In this particular case, the inputs are x_1 and x_2 at input layer L_0 .

Multi Layer Perceptron - More notations



- At layer L_1 :
 - ▶ At neuron n_1^1 :
 - ★ $a_1^1 = \phi(w_{11}^1 x_1 + w_{12}^1 x_2)$.

Multi Layer Perceptron - More notations

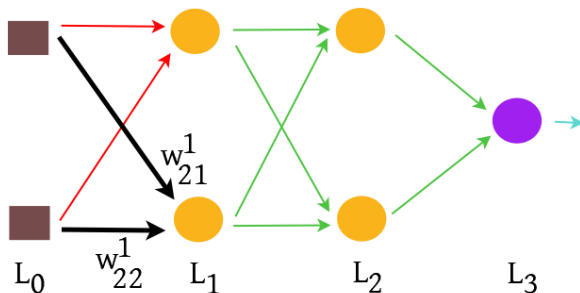


- At layer L_1 :

- At neuron n_1^1 :

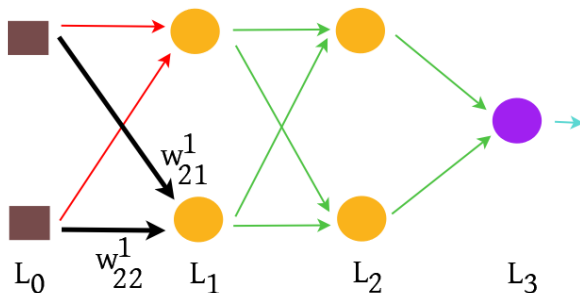
- ★ $a_1^1 = \phi(w_{11}^1 x_1 + w_{12}^1 x_2) =: \phi(z_1^1)$.

Multi Layer Perceptron - More notations



- At layer L_1 :
 - At neuron n_2^1 :
 - ★ $a_2^1 = \phi(w_{21}^1 x_1 + w_{22}^1 x_2)$.

Multi Layer Perceptron - More notations

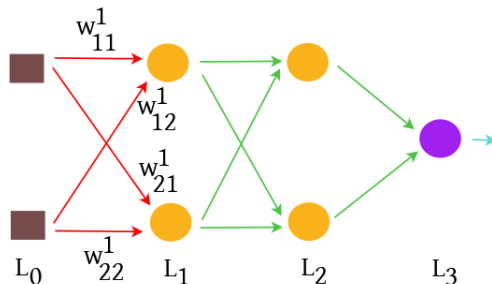


- At layer L_1 :

- At neuron n_2^1 :

- ★ $a_2^1 = \phi(w_{21}^1 x_1 + w_{22}^1 x_2) =: \phi(z_2^1)$.

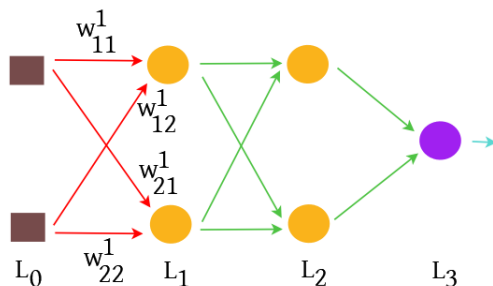
Multi Layer Perceptron - More notations



- At layer L_1 :

$$\begin{bmatrix} a_1^1 \\ a_2^1 \end{bmatrix} = \begin{bmatrix} \phi(z_1^1) \\ \phi(z_2^1) \end{bmatrix} = \begin{bmatrix} \phi(w_{11}^1 x_1 + w_{12}^1 x_2) \\ \phi(w_{21}^1 x_1 + w_{22}^1 x_2) \end{bmatrix}$$

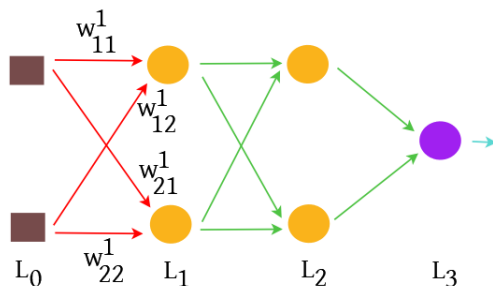
Multi Layer Perceptron - More notations



- Letting $W^1 = \begin{bmatrix} w^1_{11} & w^1_{12} \\ w^1_{21} & w^1_{22} \end{bmatrix}$ and $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, we have at layer L_1 :

$$\begin{bmatrix} a_1^1 \\ a_2^1 \end{bmatrix} = \phi \left(\begin{bmatrix} z_1^1 \\ z_2^1 \end{bmatrix} \right) = \phi \left(\begin{bmatrix} w^1_{11}x_1 + w^1_{12}x_2 \\ w^1_{21}x_1 + w^1_{22}x_2 \end{bmatrix} \right) = \phi(W^1x)$$

Multi Layer Perceptron - More notations



- Letting $a^1 = \begin{bmatrix} a_1^1 \\ a_2^1 \end{bmatrix}$, we have at layer L_1 :

$$a^1 = \begin{bmatrix} a_1^1 \\ a_2^1 \end{bmatrix} = \phi(W^1 x)$$