Machine Learning: Principles and Techniques

IE 506

Moving on from Perceptrons to Multi-layer Perceptrons

March 9, 2023.

1. Moving on from Perceptron
2. Multi Layer Perceptron
   - MLP-Data Perspective

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.

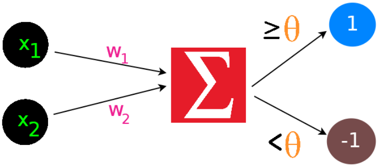
Perceptron - Caveat

- 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

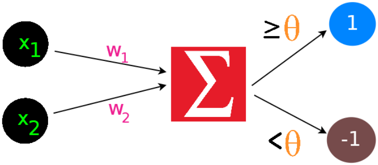
- Not suitable when **linear separability assumption** fails
- Example: Classical XOR problem



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

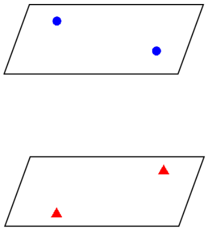
- Not suitable when **linear separability assumption** fails
- Example: Classical XOR problem



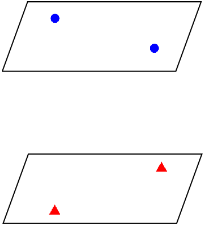
$$
\begin{aligned}
\text{sign}(-\theta) = -1 &\implies \theta > 0\\
\text{sign}(w_2 - \theta) = 1 &\implies w_2 - \theta \geq 0\\
\text{sign}(w_1 - \theta) = 1 &\implies w_1 - \theta \geq 0\\
\text{sign}(w_1 + w_2 - \theta) = -1 &\implies -w_1 - w_2 + \theta > 0
\end{aligned}
$$

Note: This system is inconsistent. (Homework!)

Moving away from perceptron - Dealing with XOR problem

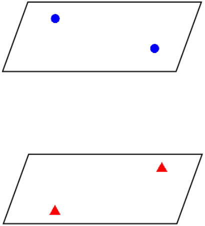


Moving away from perceptron - Dealing with XOR problem



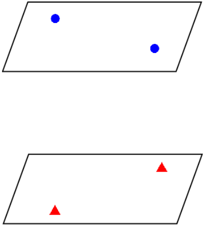
- Assume that the sample features $x \in \mathbb{R}^d$.

Moving away from perceptron - Dealing with XOR problem



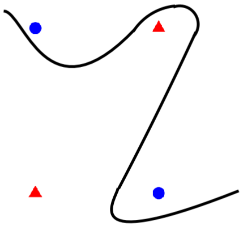
- Assume that the sample features $x \in \mathbb{R}^d$.
- **Idea:** Use a transformation $\psi : \mathbb{R}^d \to \mathbb{R}^q$, where $q \gg d$, to lift the data samples $x \in \mathbb{R}^d$ into $\psi(x) \in \mathbb{R}^q$ hoping to see a separating hyperplane in the transformed space.

Moving away from perceptron - Dealing with XOR problem

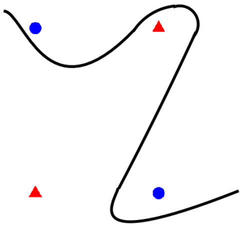


- Assume that the sample features $x \in \mathbb{R}^d$.
- **Idea:** Use a transformation $\psi : \mathbb{R}^d \to \mathbb{R}^q$, where $q \gg d$, to lift the data samples $x \in \mathbb{R}^d$ into $\psi(x) \in \mathbb{R}^q$ hoping to see a separating hyperplane in the transformed space.
- Forms the core idea behind kernel methods. (which we discussed earlier!)

Moving away from perceptron - Dealing with XOR problem

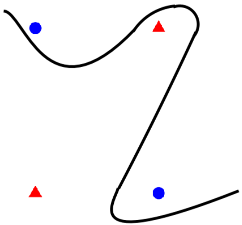


Moving away from perceptron - Dealing with XOR problem



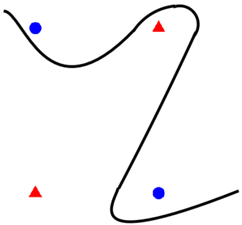
- **Idea:** The separating surface need not be linear in the ambient space and can be assumed to take some non-linear form.

Moving away from perceptron - Dealing with XOR problem



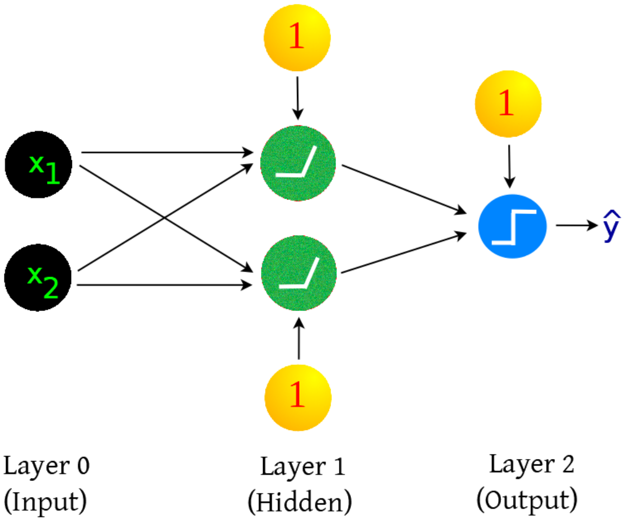
- **Idea:** The separating surface need not be linear in the ambient space and can be assumed to take some non-linear form.
- Hence for an input space $\mathcal{X}$ and output space $\mathcal{Y}$, the learned map $h : \mathcal{X} \to \mathcal{Y}$ can take some non-linear form.

Moving away from perceptron - Dealing with XOR problem

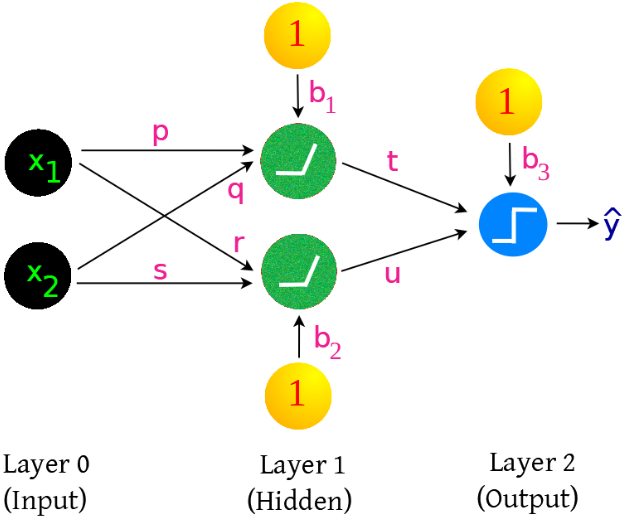


- **Idea:** The separating surface need not be linear in the ambient space and can be assumed to take some non-linear form.
- Hence for an input space $\mathcal{X}$ and output space $\mathcal{Y}$, the learned map $h : \mathcal{X} \to \mathcal{Y}$ can take some non-linear form.
- Forms the idea behind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

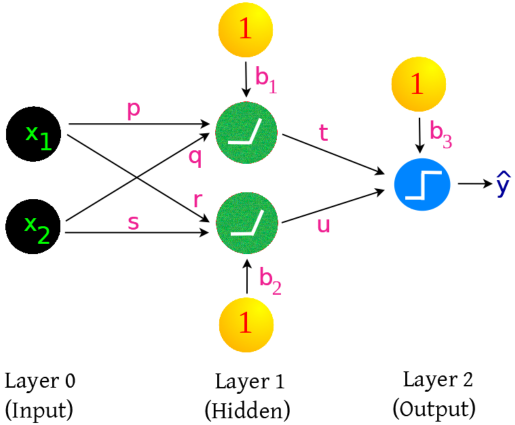
Moving away from perceptron - Dealing with XOR problem



Moving away from perceptron - Dealing with XOR problem



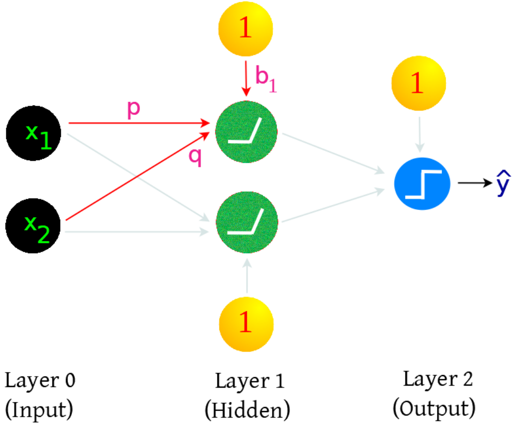
Moving away from perceptron - Dealing with XOR problem



Some notations

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

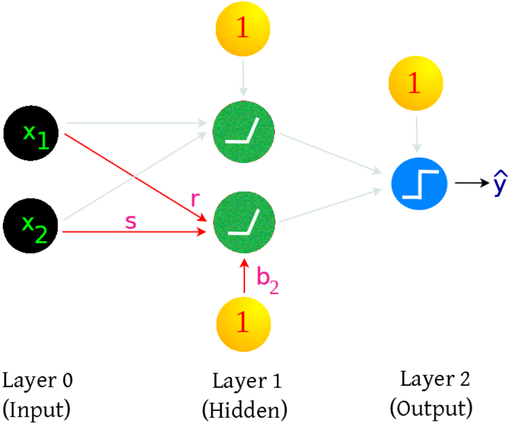
Moving away from perceptron - Dealing with XOR problem



- Activation at neuron n_1^1:

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

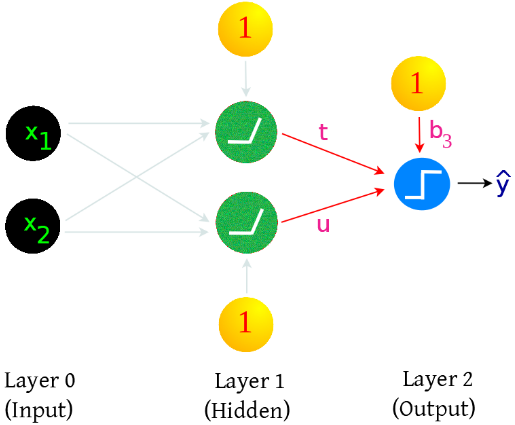
Moving away from perceptron - Dealing with XOR problem



- Activation at neuron n_2^1:

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

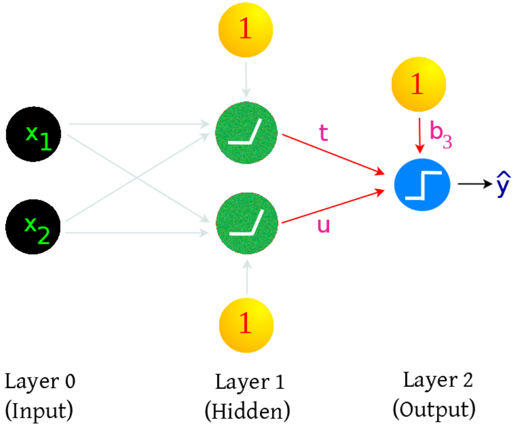
Moving away from perceptron - Dealing with XOR problem



- 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

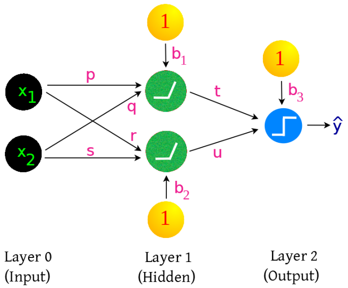


- Activation at neuron n_1^2:

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

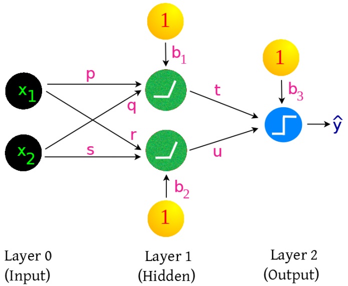
- **Note:** The activation a_1^2 is the output of the network denoted by $\hat{y}$.

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

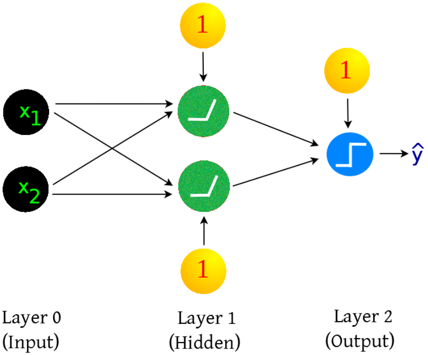
Homework: Find weights $p, q, r, s, t, u, b_1, b_2, b_3$ such that the MLP solves the XOR problem.

Moving away from perceptron - Dealing with XOR problem

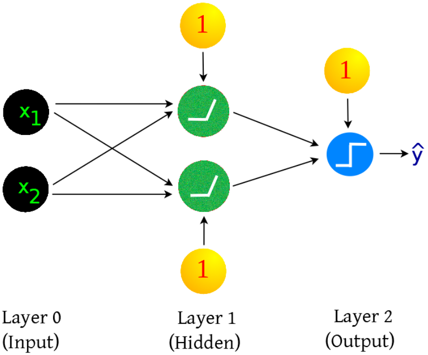
A different Multi Layer Perceptron (MLP) architecture is given for XOR problem in:

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

Multi Layer Perceptron

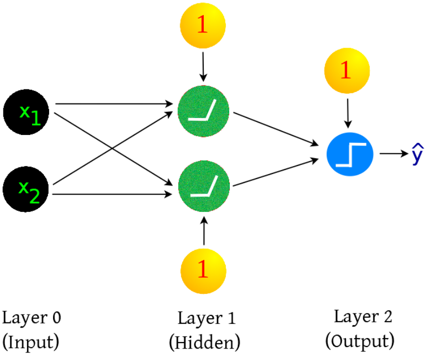


Multi Layer Perceptron



Notable features:

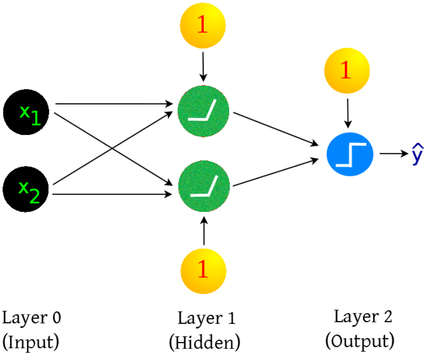
Multi Layer Perceptron



Notable features:

- Multiple layers stacked together.

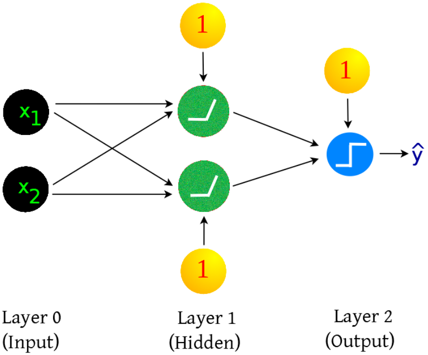
Multi Layer Perceptron



Notable features:

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

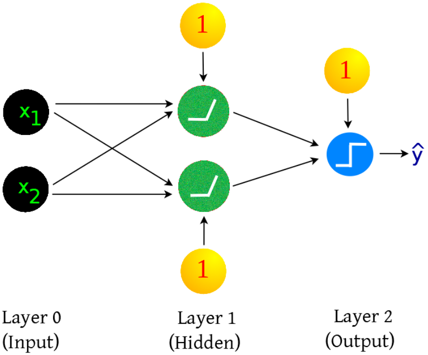
Multi Layer Perceptron



Notable features:

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

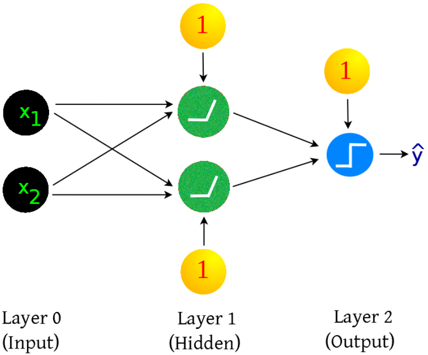
Multi Layer Perceptron



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.

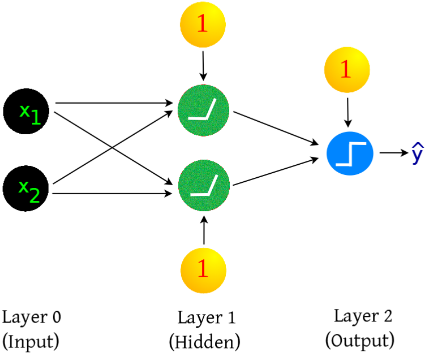
Multi Layer Perceptron



Notable features:

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

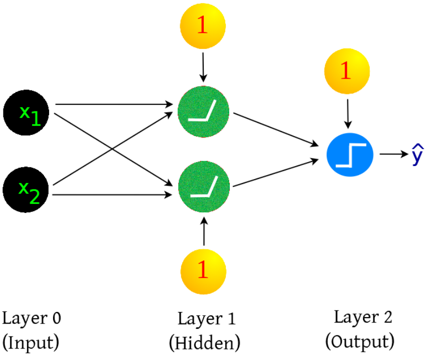
Multi Layer Perceptron



Notable features:

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

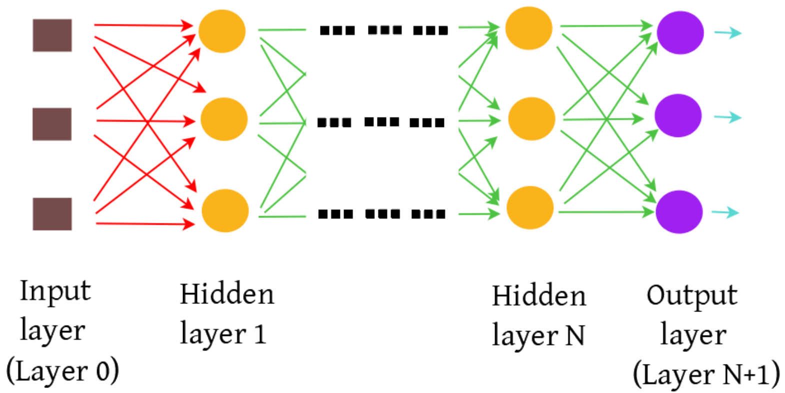
Multi Layer Perceptron



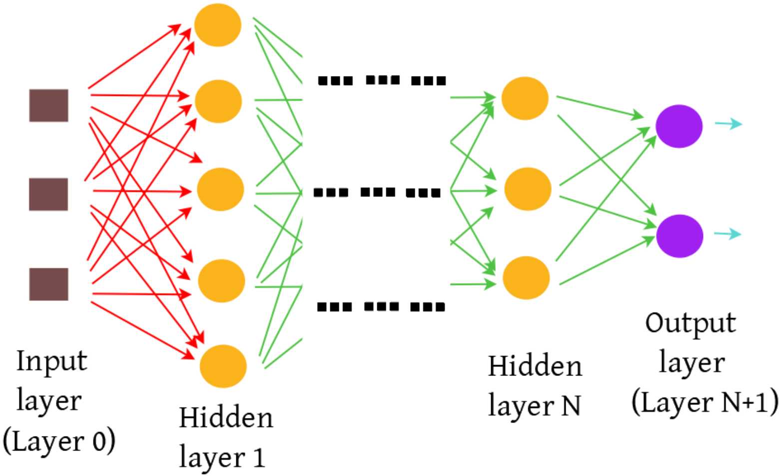
Notable features:

- 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** (**R**ectified **L**inear **U**nit).

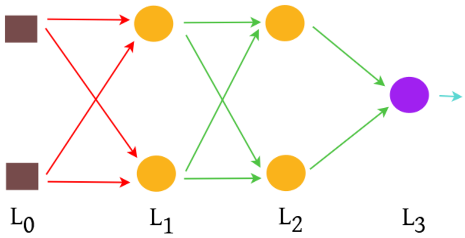
Multi Layer Perceptron



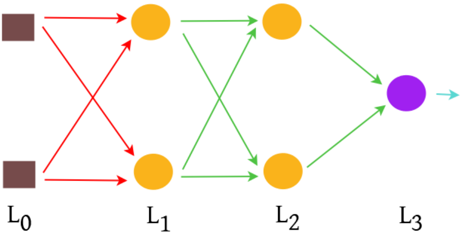
Multi Layer Perceptron



Multi Layer Perceptron - More notations

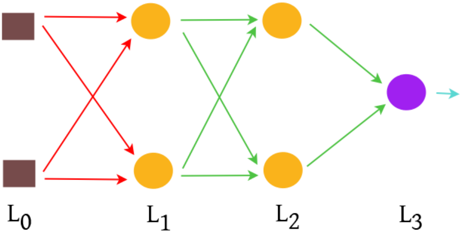


Multi Layer Perceptron - More notations



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

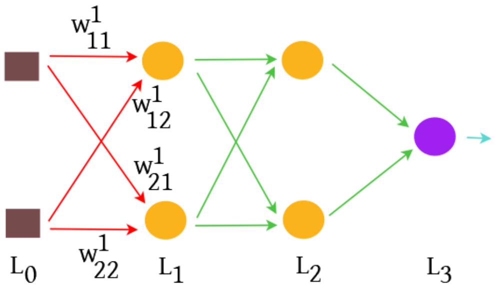
Multi Layer Perceptron - More notations



Recall:

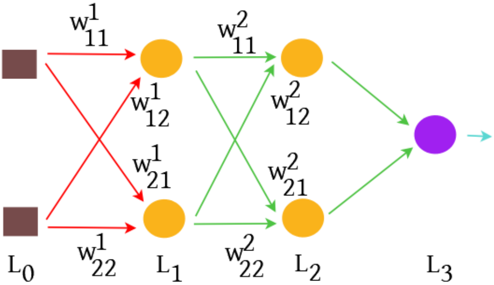
- n_k^ℓ denotes k-th neuron at ℓ-th layer.
- a_k^ℓ denotes activation of neuron n_k^ℓ.

Multi Layer Perceptron - More notations



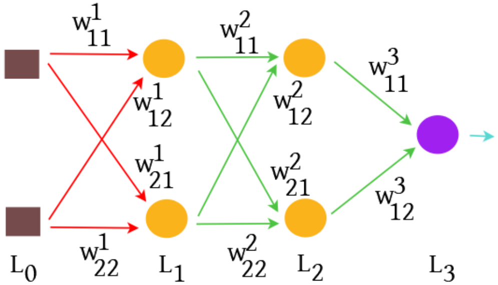
- w_{ij}^{ℓ} denotes weight of connection connecting n_i^{ℓ} from $n_j^{\ell-1}$.

Multi Layer Perceptron - More notations



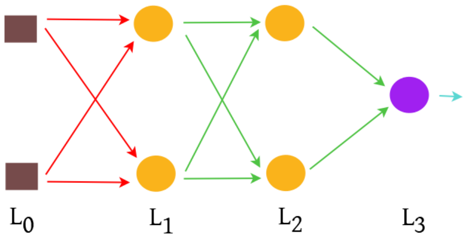
- w^{ℓ}_{ij} denotes weight of connection connecting n^{ℓ}_i from $n^{\ell-1}_j$.

Multi Layer Perceptron - More notations



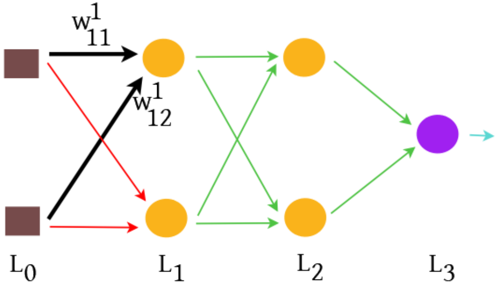
- w^{ℓ}_{ij} denotes weight of connection connecting n^{ℓ}_i from $n^{\ell-1}_j$.

Multi Layer Perceptron - More notations



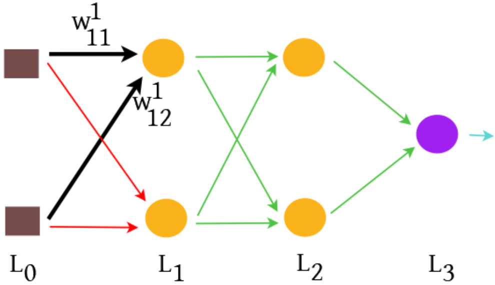
- In this particular case, the inputs are x_1 and x_2 at input layer L_0.

Multi Layer Perceptron - More notations



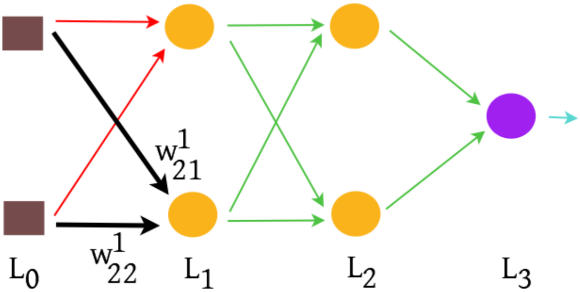
- At layer L_1:
  - At neuron n^1_1:
    - ★ $a^1_1 = \phi(w^1_{11}x_1 + w^1_{12}x_2)$.

Multi Layer Perceptron - More notations



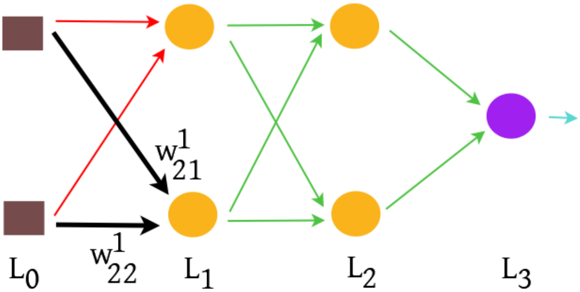
- At layer L_1:
  - At neuron n^1_1:
    - ★ $a^1_1 = \phi(w^1_{11}x_1 + w^1_{12}x_2) =: \phi(z^1_1)$.

Multi Layer Perceptron - More notations



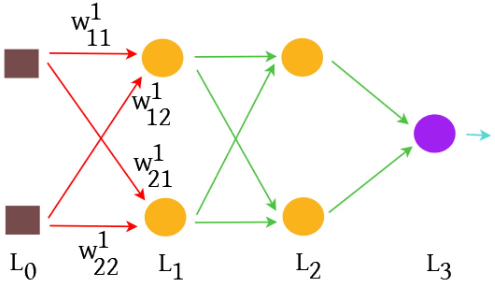
- At layer L_1:
  - At neuron n_2^1:
    - ★ $a_2^1 = \phi(w_{21}^1 x_1 + w_{22}^1 x_2)$.

Multi Layer Perceptron - More notations



- At layer L_1:
  - At neuron n^1_2:
    - ★ $a^1_2 = \phi(w^1_{21}x_1 + w^1_{22}x_2) =: \phi(z^1_2)$.

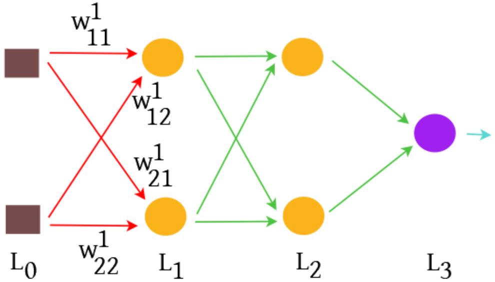
Multi Layer Perceptron - More notations



- At layer L_1:

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

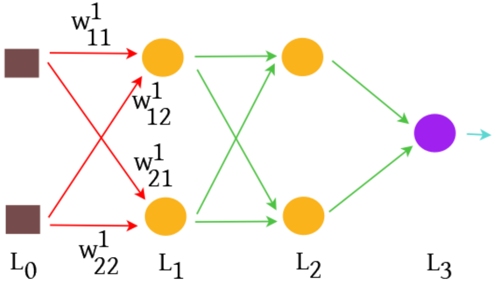
Multi Layer Perceptron - More notations



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

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

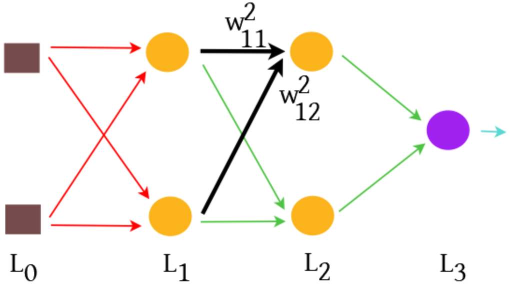
Multi Layer Perceptron - More notations



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

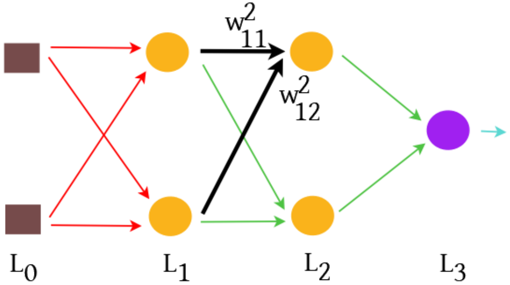
$$a^1 = \begin{bmatrix} a_1^1 \\ a_2^1 \end{bmatrix} = \phi(W^1 x)$$

Multi Layer Perceptron - More notations



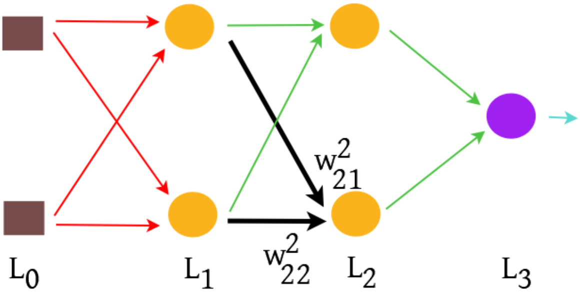
- At layer L_2:
  - At neuron n_1^2:
    - ⋆ $a_1^2 = \phi(w_{11}^2 a_1^1 + w_{12}^2 a_2^1)$.

Multi Layer Perceptron - More notations



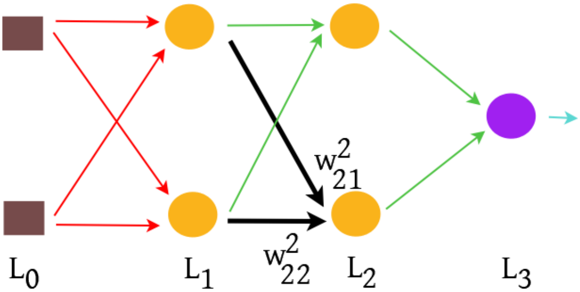
- At layer L_2:
  - At neuron n^2_1:
    - ★ $a^2_1 = \phi(w^2_{11}a^1_1 + w^2_{12}a^1_2) =: \phi(z^2_1)$.

Multi Layer Perceptron - More notations



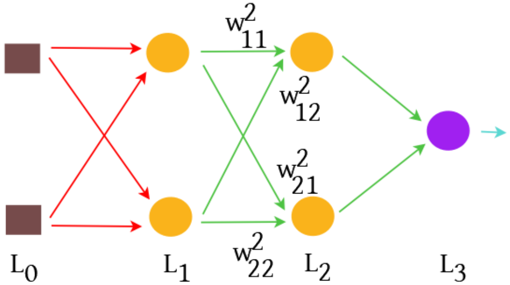
- At layer L_2:
  - At neuron n^2_2:
    - $a^2_2 = \phi(w^2_{21}a^1_1 + w^2_{22}a^1_2)$.

Multi Layer Perceptron - More notations



- At layer L_2:
  - At neuron n_2^2:
    - ★ $a_2^2 = \phi(w_{21}^2 a_1^1 + w_{22}^2 a_2^1) =: \phi(z_2^2)$.

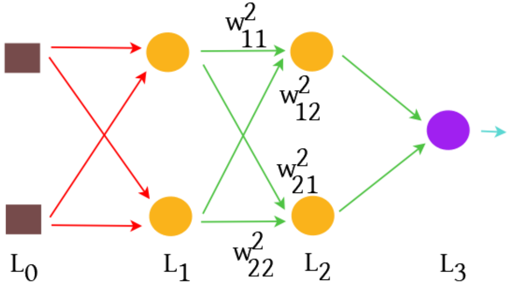
Multi Layer Perceptron - More notations



- At layer L_2:

$$a^2 = \begin{bmatrix} a_1^2 \\ a_2^2 \end{bmatrix} = \begin{bmatrix} \phi(z_1^2) \\ \phi(z_2^2) \end{bmatrix} = \begin{bmatrix} \phi(w_{11}^2 a_1^1 + w_{12}^2 a_2^1) \\ \phi(w_{21}^2 a_1^1 + w_{22}^2 a_2^1) \end{bmatrix}$$

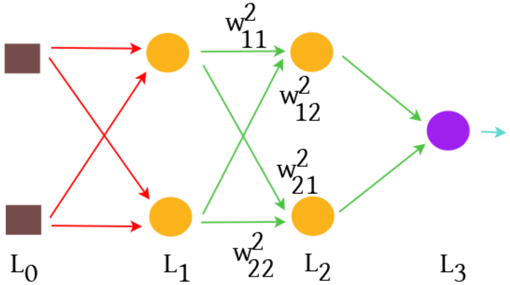
Multi Layer Perceptron - More notations



- Letting $W^2 = \begin{bmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \end{bmatrix}$, we have at layer L_2:

$$a^2 = \begin{bmatrix} a_1^2 \\ a_2^2 \end{bmatrix} = \phi\left(\begin{bmatrix} z_1^2 \\ z_2^2 \end{bmatrix}\right) = \phi\left(\begin{bmatrix} w_{11}^2 a_1^1 + w_{12}^2 a_2^1 \\ w_{21}^2 a_1^1 + w_{22}^2 a_2^1 \end{bmatrix}\right) = \phi\left(W^2 \begin{bmatrix} a_1^1 \\ a_2^1 \end{bmatrix}\right)$$

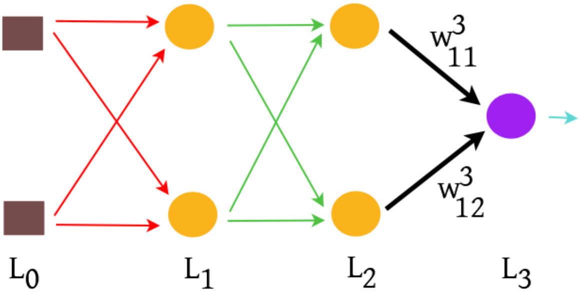
Multi Layer Perceptron - More notations



- We have at layer L_2:

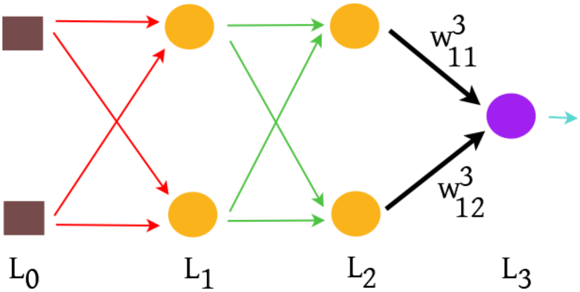
$$a^2 = \begin{bmatrix} a^2_1 \\ a^2_2 \end{bmatrix} = \phi\left(\begin{bmatrix} z^2_1 \\ z^2_2 \end{bmatrix}\right) = \phi\left(W^2 \begin{bmatrix} a^1_1 \\ a^1_2 \end{bmatrix}\right) = \phi(W^2 a^1)$$

Multi Layer Perceptron - More notations



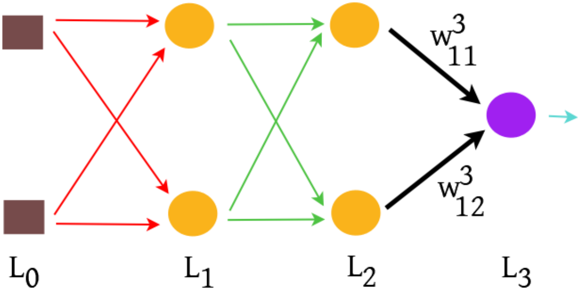
- At layer L_3:
  - At neuron n^3_1:
    - ★ $a^3_1 = \phi(w^3_{11}a^2_1 + w^3_{12}a^2_2)$.

Multi Layer Perceptron - More notations



- At layer L_3:
  - At neuron n^3_1:
    - ★ $a^3_1 = \phi(w^3_{11}a^2_1 + w^3_{12}a^2_2) =: \phi(z^3_1)$.

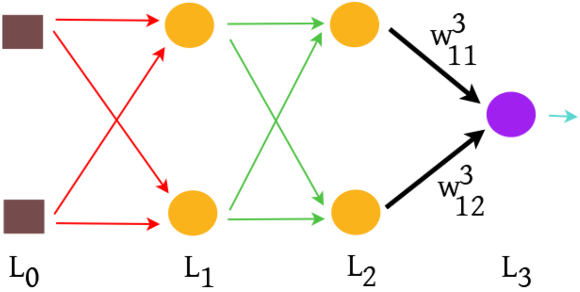
Multi Layer Perceptron - More notations



- At layer L_3:

$$a^3 = \left[a^3_1\right] = \left[\phi(z^3_1)\right] = \left[\phi\left(w^3_{11}a^2_1 + w^3_{12}a^2_2\right)\right]$$

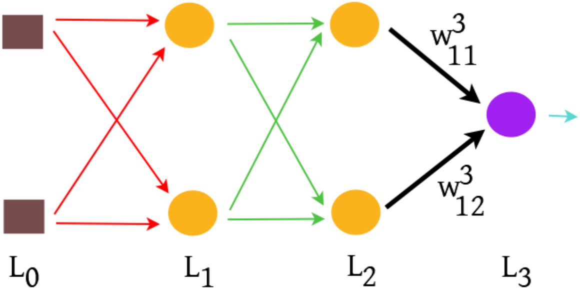
Multi Layer Perceptron - More notations



- Letting $W^3 = \begin{bmatrix} w^3_{11} & w^3_{12} \end{bmatrix}$, we have at layer L_3:

$$a^3 = \begin{bmatrix} a^3_1 \end{bmatrix} = \phi\left(\begin{bmatrix} z^3_1 \end{bmatrix}\right) = \phi\left(\begin{bmatrix} w^3_{11} a^2_1 + w^3_{12} a^2_2 \end{bmatrix}\right) = \phi\left(W^3 \begin{bmatrix} a^2_1 \\ a^2_2 \end{bmatrix}\right)$$

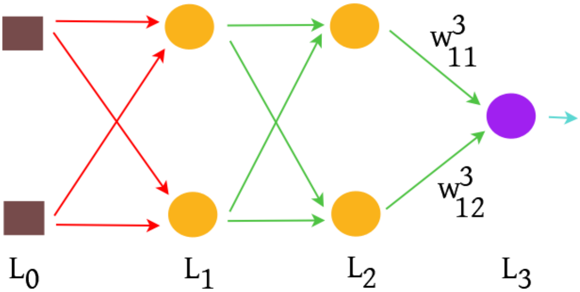
Multi Layer Perceptron - More notations



- Letting $W^3 = \begin{bmatrix} w^3_{11} & w^3_{12} \end{bmatrix}$, we have at layer L_3:

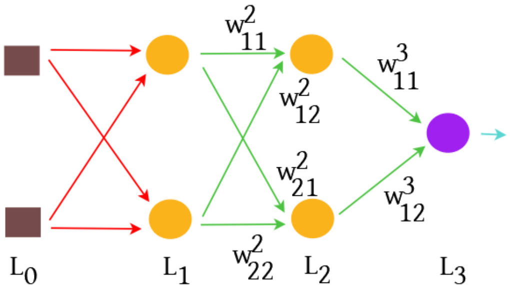
$$a^3 = \begin{bmatrix} a^3_1 \end{bmatrix} = \phi\left(\begin{bmatrix} z^3_1 \end{bmatrix}\right) = \phi\left(W^3 \begin{bmatrix} a^2_1 \\ a^2_2 \end{bmatrix}\right) = \phi(W^3 a^2)$$

Multi Layer Perceptron - More notations



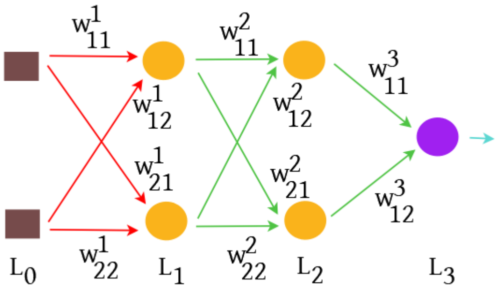
$$a^3 = \phi(W^3 a^2)$$

Multi Layer Perceptron - More notations



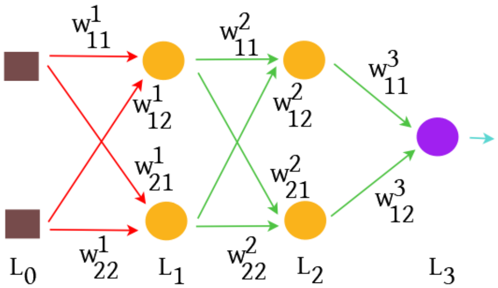
$$a^3 = \phi(W^3 a^2) = \phi(W^3 \phi(W^2 a^1))$$

Multi Layer Perceptron - More notations



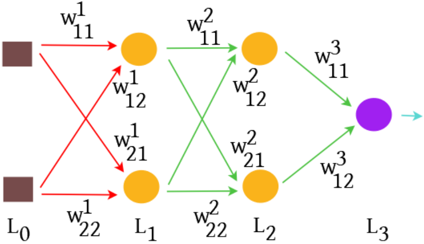
$$a^3 = \phi(W^3 a^2) = \phi(W^3 \phi(W^2 a^1)) = \phi(W^3 \phi(W^2 \phi(W^1 x)))$$

Multi Layer Perceptron - More notations



$$\hat{y} = a^3 = \phi(W^3 a^2) = \phi(W^3 \phi(W^2 a^1)) = \phi(W^3 \phi(W^2 \phi(W^1 x)))$$

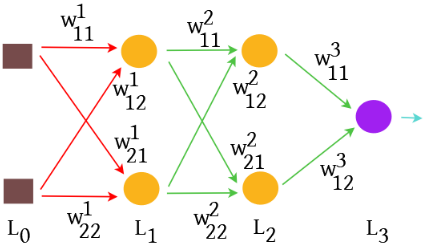
Multi Layer Perceptron - Data Perspective



Given data (x, y), multi layer perceptron predicts:

$$\hat{y} = \phi(W^3\phi(W^2\phi(W^1x)))$$

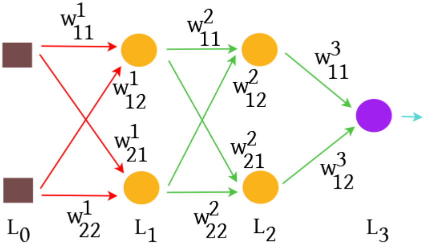
Multi Layer Perceptron - Data Perspective



Given data (x, y), multi layer perceptron predicts:

$$\hat{y} = \phi(W^3\phi(W^2\phi(W^1x))) =: \mathrm{MLP}(x)$$

Multi Layer Perceptron - Data Perspective



Given data (x, y), multi layer perceptron predicts:

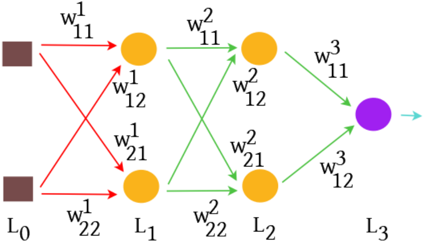
$$\hat{y} = \phi(W^3\phi(W^2\phi(W^1x))) =: \text{MLP}(x)$$

Note: The same activation function ϕ was assumed for simplicity. Typically different activations functions are used for different layers. Then we can write:

$$\hat{y} = \phi_3(W^3\phi_2(W^2\phi_1(W^1x))) =: \text{MLP}(x)$$

where ϕ_1, ϕ_2 and ϕ_3 are activation functions for layers L_1, L_2 and L_3 respectively.

Multi Layer Perceptron - Data Perspective

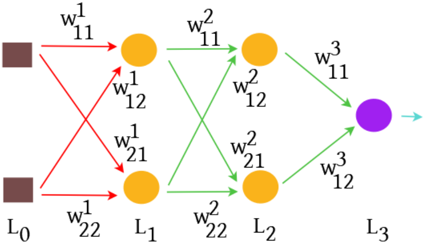


Given data (x, y), multi layer perceptron predicts:

$$\hat{y} = \phi(W^3\phi(W^2\phi(W^1x))) =: \mathrm{MLP}(x)$$

Similar to perceptron, if $y \neq \hat{y}$ an error $E(y, \hat{y})$ is incurred.

Multi Layer Perceptron - Data Perspective



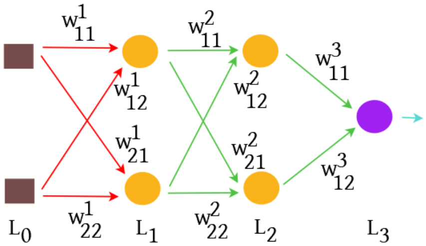
Given data (x, y), multi layer perceptron predicts:

$$\hat{y} = \phi(W^3\phi(W^2\phi(W^1x))) =: \mathrm{MLP}(x)$$

Similar to perceptron, if $y \neq \hat{y}$ an error $E(y, \hat{y})$ is incurred.

Aim: To change the weights W^1, W^2, W^3, such that the error $E(y, \hat{y})$ is minimized.

Multi Layer Perceptron - Data Perspective



Given data (x, y), multi layer perceptron predicts:

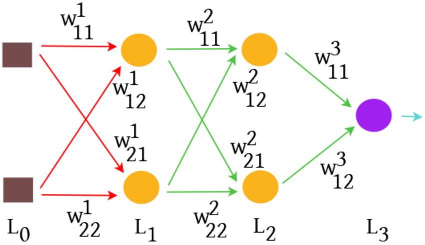
$$\hat{y} = \phi(W^3\phi(W^2\phi(W^1x))) =: \mathrm{MLP}(x)$$

Similar to perceptron, if $y \neq \hat{y}$ an error $E(y, \hat{y})$ is incurred.

Aim: To change the weights W^1, W^2, W^3, such that the error $E(y, \hat{y})$ is minimized.

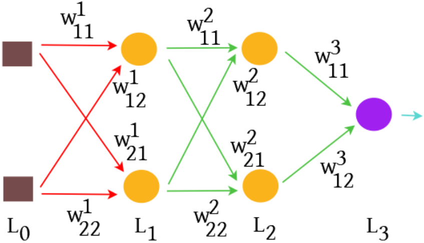
Leads to an error minimization problem.

Multi Layer Perceptron - Data Perspective



- **Input:** Training Data $D = \{(x^s, y^s)\}_{s=1}^{S}$.
- For each sample x^s the prediction $\hat{y}^s = \text{MLP}(x^s)$.
- **Error:** $e^s = E(y^s, \hat{y}^s)$.
- **Aim:** To minimize $\sum_{s=1}^{S} e^s$.

Multi Layer Perceptron - Data Perspective

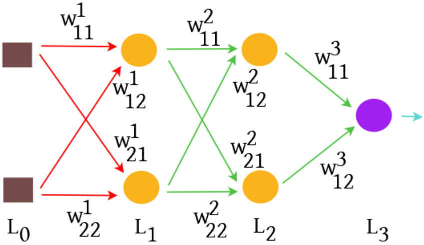


Optimization perspective

- Given training data $D = \{(x^s, y^s)\}_{s=1}^{S}$,

$$\min \sum_{s=1}^{S} e^s$$

Multi Layer Perceptron - Data Perspective

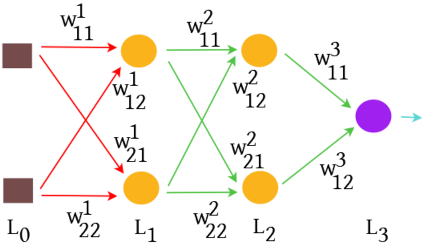


Optimization perspective

- Given training data $D = \{(x^s, y^s)\}_{s=1}^{S}$,

$$\min \sum_{s=1}^{S} e^s = \sum_{s=1}^{S} E(y^s, \hat{y}^s)$$

Multi Layer Perceptron - Data Perspective

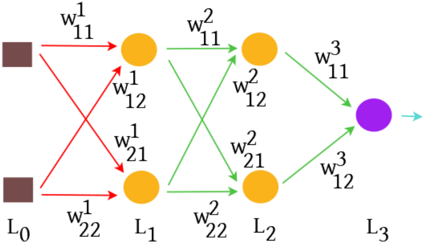


Optimization perspective

- Given training data $D = \{(x^s, y^s)\}_{s=1}^{S}$,

$$\min \sum_{s=1}^{S} e^s = \sum_{s=1}^{S} E(y^s, \hat{y}^s) = \sum_{s=1}^{S} E(y^s, \mathrm{MLP}(x^s))$$

Multi Layer Perceptron - Data Perspective



Optimization perspective

- Given training data $D = \{(x^s, y^s)\}_{s=1}^{S}$,

$$\min \sum_{s=1}^{S} e^s = \sum_{s=1}^{S} E(y^s, \hat{y}^s) = \sum_{s=1}^{S} E(y^s, \mathrm{MLP}(x^s))$$

- Note: The minimization is over the weights of the MLP $W^1, \ldots, W^L$, where L denotes number of layers in MLP.