Deep Learning - Theory and Practice

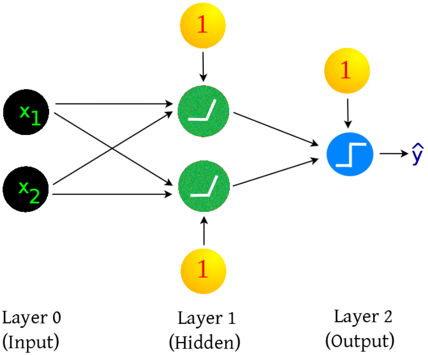
IE 643
Lectures 6, 7

August 23 & 26, 2022.

1. Recap
   - Multi Layer Perceptron
   - MLP-Data Perspective

2. Optimization Concepts
   - Gradient Descent
   - Stochastic Gradient Descent
   - Mini-batch SGD

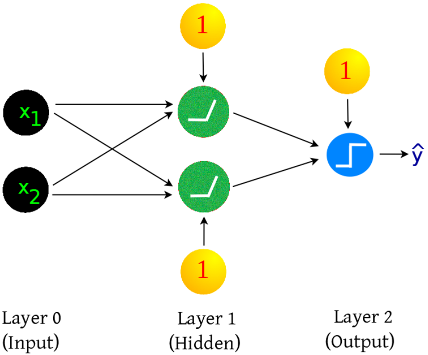
Multi Layer Perceptron



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.

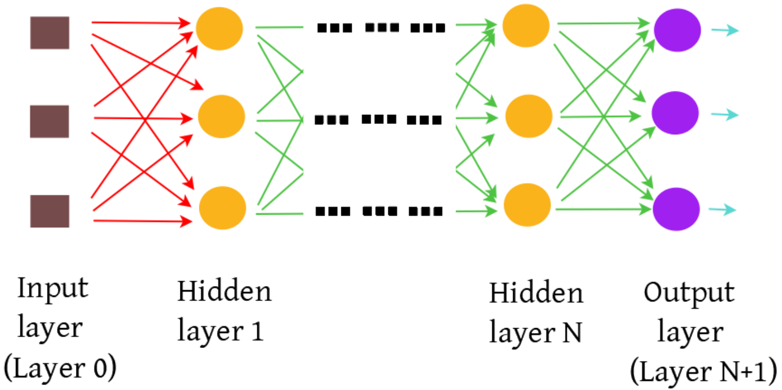
Multi Layer Perceptron



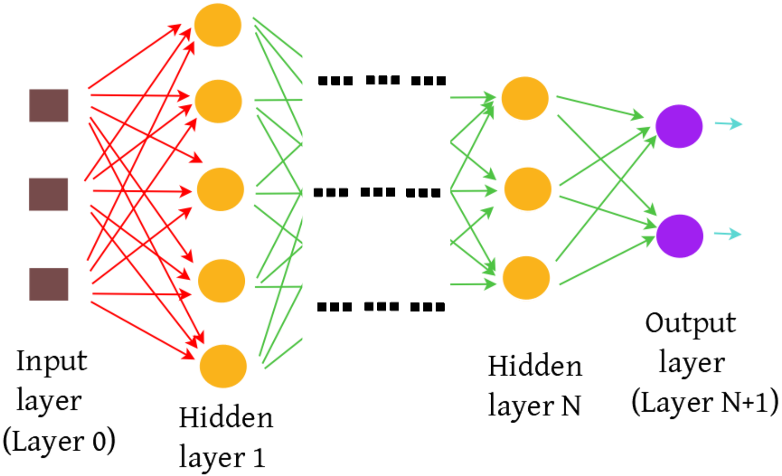
Notable features:

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

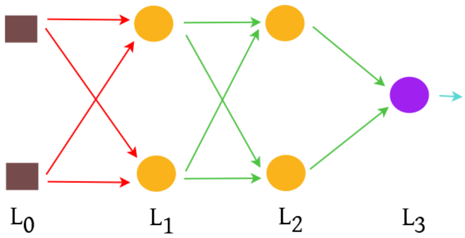
Multi Layer Perceptron



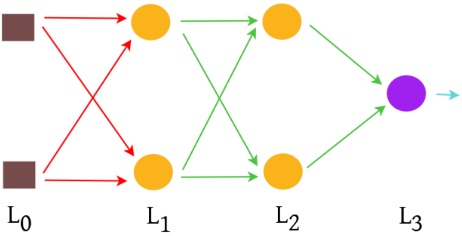
Multi Layer Perceptron



Multi Layer Perceptron - More notations

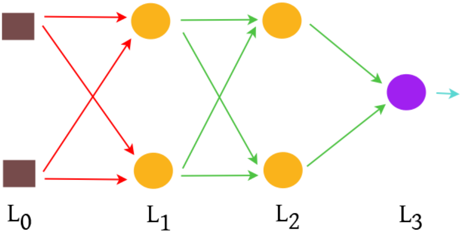


Multi Layer Perceptron - More notations



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

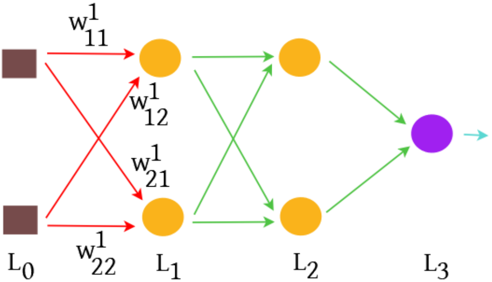
Multi Layer Perceptron - More notations



Recall:

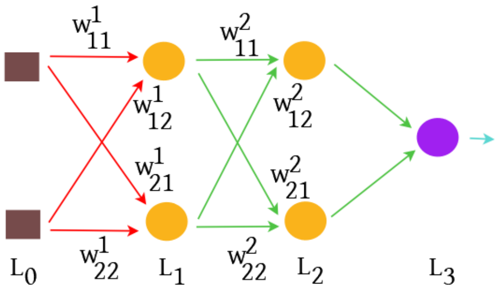
- n_k^ℓ denotes k-th neuron at ℓ-th layer.
- a_k^ℓ denotes activation of neuron n_k^ℓ.

Multi Layer Perceptron - More notations



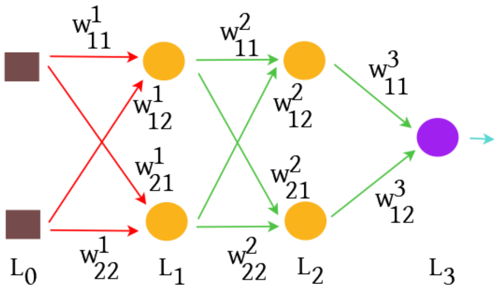
- w_{ij}^{ℓ} denotes weight of connection connecting n_i^{ℓ} from $n_j^{\ell-1}$.

Multi Layer Perceptron - More notations



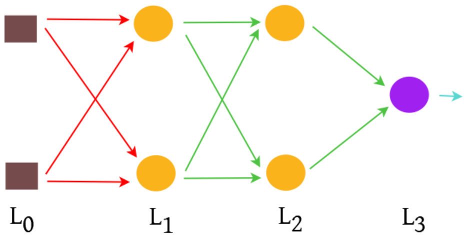
- w_{ij}^{ℓ} denotes weight of connection connecting n_i^{ℓ} from $n_j^{\ell-1}$.

Multi Layer Perceptron - More notations



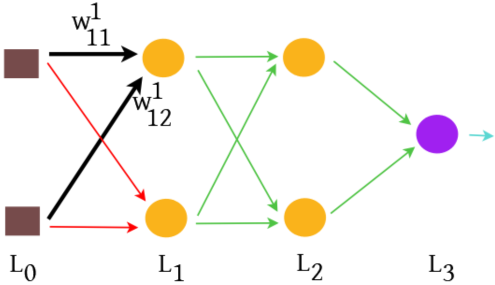
- w^{ℓ}_{ij} denotes weight of connection connecting n^{ℓ}_i from $n^{\ell-1}_j$.

Multi Layer Perceptron - More notations



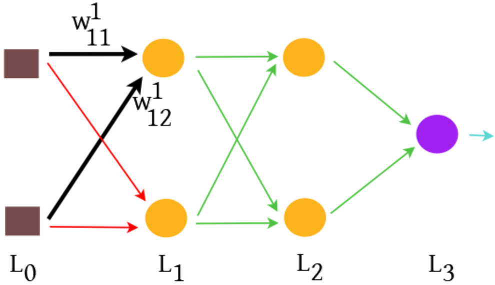
- In this particular case, the inputs are x_1 and x_2 at input layer L_0.

Multi Layer Perceptron - More notations



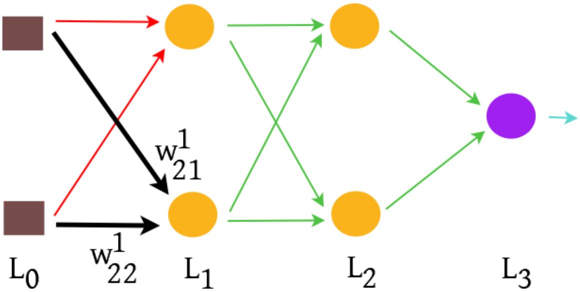
- At layer L_1:
  - At neuron n^1_1:
    - ★ $a^1_1 = \phi(w^1_{11}x_1 + w^1_{12}x_2)$.

Multi Layer Perceptron - More notations



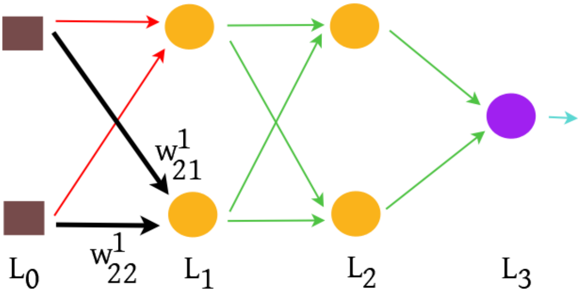
- At layer L_1:
  - At neuron n^1_1:
    - ★ $a^1_1 = \phi(w^1_{11}x_1 + w^1_{12}x_2) =: \phi(z^1_1)$.

Multi Layer Perceptron - More notations



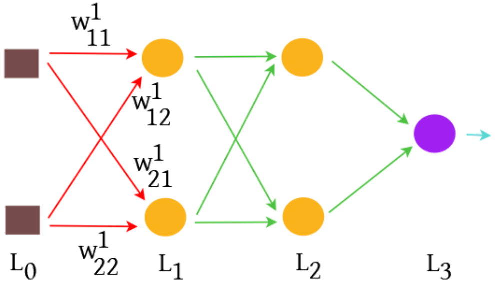
- At layer L_1:
  - At neuron n_2^1:
    - ★ $a_2^1 = \phi(w_{21}^1 x_1 + w_{22}^1 x_2)$.

Multi Layer Perceptron - More notations



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

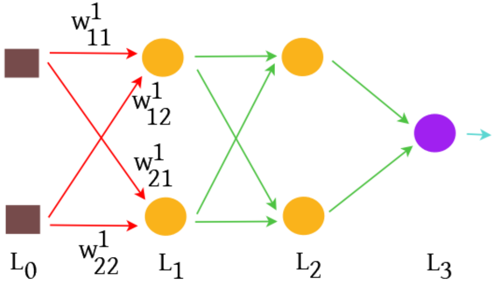
Multi Layer Perceptron - More notations



- At layer L_1:

$$\begin{bmatrix} a^1_1 \\ a^1_2 \end{bmatrix} = \begin{bmatrix} \phi(z^1_1) \\ \phi(z^1_2) \end{bmatrix} = \begin{bmatrix} \phi(w^1_{11}x_1 + w^1_{12}x_2) \\ \phi(w^1_{21}x_1 + w^1_{22}x_2) \end{bmatrix}$$

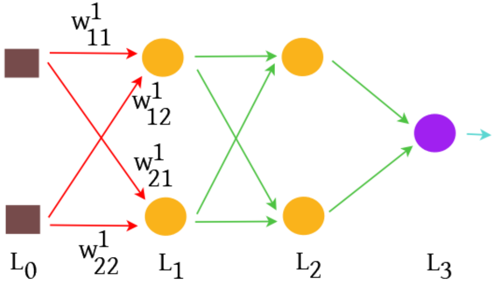
Multi Layer Perceptron - More notations



- 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:

$$\begin{bmatrix} a^1_1 \\ a^1_2 \end{bmatrix} = \phi\left(\begin{bmatrix} z^1_1 \\ z^1_2 \end{bmatrix}\right) = \phi\left(\begin{bmatrix} w^1_{11}x_1 + w^1_{12}x_2 \\ w^1_{21}x_1 + w^1_{22}x_2 \end{bmatrix}\right) = \phi(W^1 x)$$

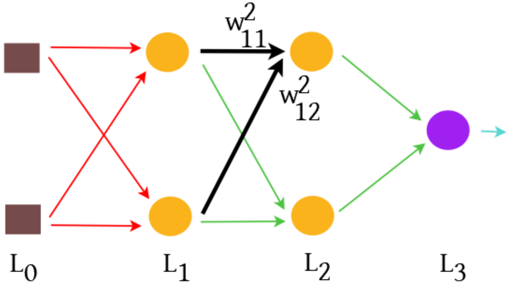
Multi Layer Perceptron - More notations



- Letting $a^1 = \begin{bmatrix} a^1_1 \\ a^1_2 \end{bmatrix}$, we have at layer L_1:

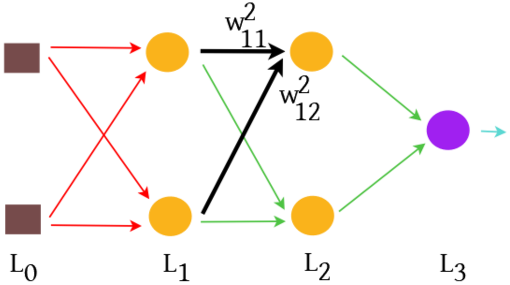
$$a^1 = \begin{bmatrix} a^1_1 \\ a^1_2 \end{bmatrix} = \phi(W^1 x)$$

Multi Layer Perceptron - More notations



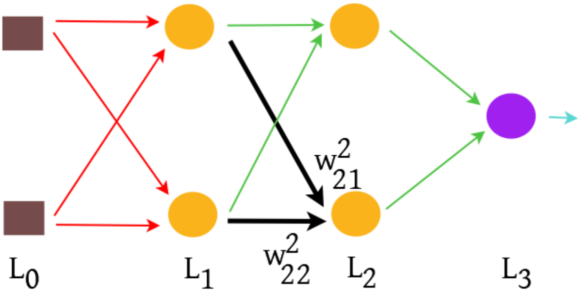
- At layer L_2:
  - At neuron n_1^2:
    - $a_1^2 = \phi(w_{11}^2 a_1^1 + w_{12}^2 a_2^1)$.

Multi Layer Perceptron - More notations



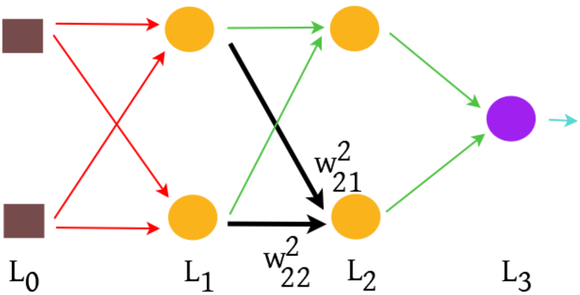
- At layer L_2:
  - At neuron n^2_1:
    - ★ $a^2_1 = \phi(w^2_{11} a^1_1 + w^2_{12} a^1_2) =: \phi(z^2_1)$.

Multi Layer Perceptron - More notations



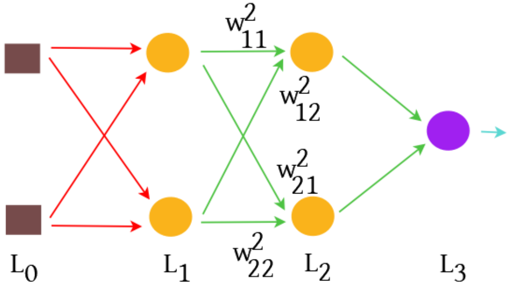
- At layer L_2:
  - At neuron n_2^2:
    - ★ $a_2^2 = \phi(w_{21}^2 a_1^1 + w_{22}^2 a_2^1)$.

Multi Layer Perceptron - More notations



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

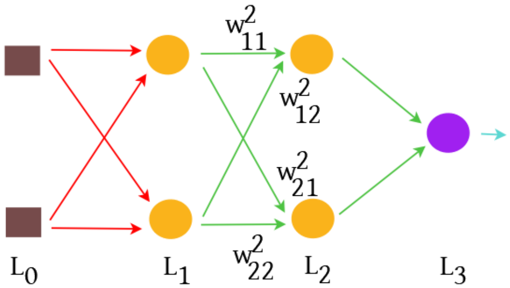
Multi Layer Perceptron - More notations



- At layer L_2:

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

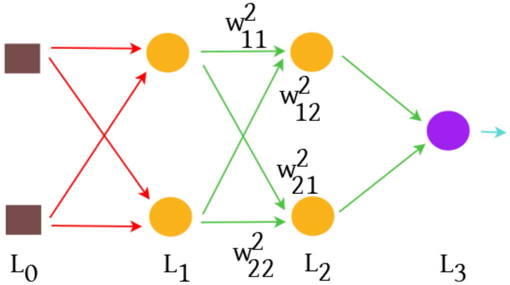
Multi Layer Perceptron - More notations



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

$$a^2 = \begin{bmatrix} a^2_1 \\ a^2_2 \end{bmatrix} = \phi\left(\begin{bmatrix} z^2_1 \\ z^2_2 \end{bmatrix}\right) = \phi\left(\begin{bmatrix} w^2_{11}a^1_1 + w^2_{12}a^1_2 \\ w^2_{21}a^1_1 + w^2_{22}a^1_2 \end{bmatrix}\right) = \phi\left(W^2 \begin{bmatrix} a^1_1 \\ a^1_2 \end{bmatrix}\right)$$

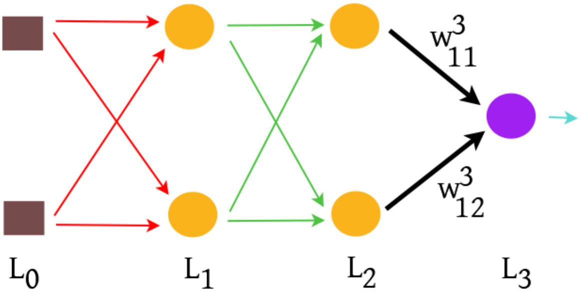
Multi Layer Perceptron - More notations



- We have at layer L_2:

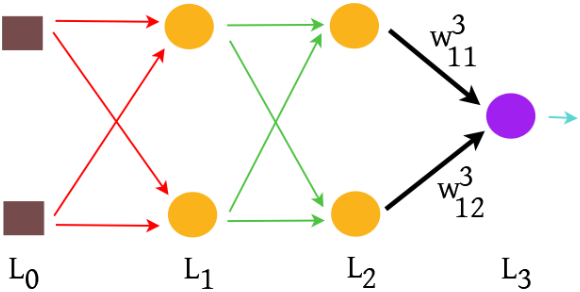
$$a^2 = \begin{bmatrix} a^2_1 \\ a^2_2 \end{bmatrix} = \phi\left(\begin{bmatrix} z^2_1 \\ z^2_2 \end{bmatrix}\right) = \phi\left(W^2 \begin{bmatrix} a^1_1 \\ a^1_2 \end{bmatrix}\right) = \phi(W^2 a^1)$$

Multi Layer Perceptron - More notations



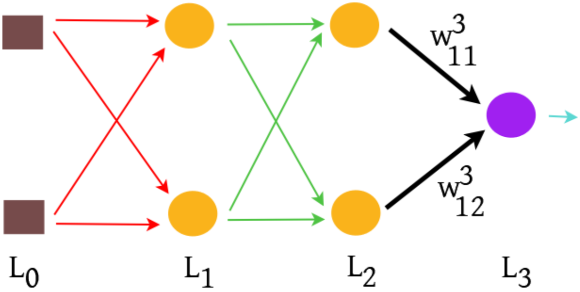
- At layer L_3:
  - At neuron n^3_1:
    - ★ $a^3_1 = \phi(w^3_{11} a^2_1 + w^3_{12} a^2_2)$.

Multi Layer Perceptron - More notations



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

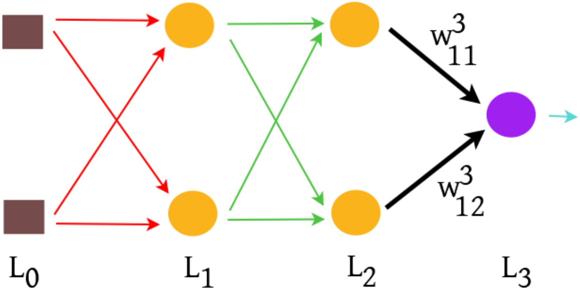
Multi Layer Perceptron - More notations



- At layer L_3:

$$a^3 = \left[a_1^3\right] = \left[\phi(z_1^3)\right] = \left[\phi(w_{11}^3 a_1^2 + w_{12}^3 a_2^2)\right]$$

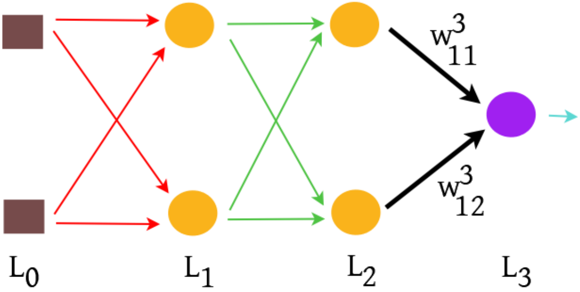
Multi Layer Perceptron - More notations



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

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

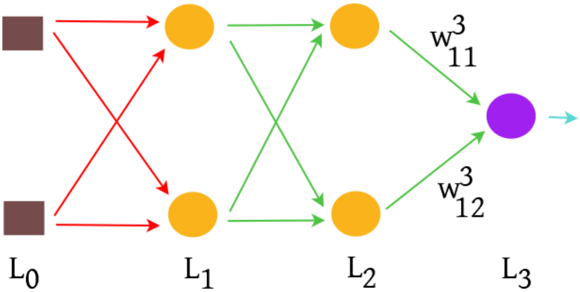
Multi Layer Perceptron - More notations



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

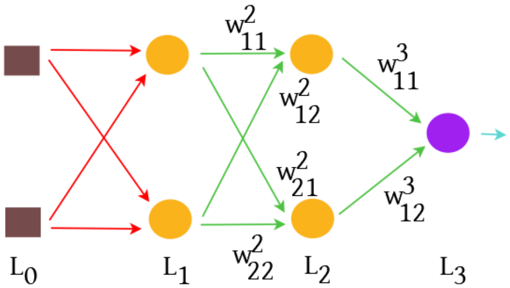
$$a^3 = \begin{bmatrix} a_1^3 \end{bmatrix} = \phi\left(\begin{bmatrix} z_1^3 \end{bmatrix}\right) = \phi\left(W^3 \begin{bmatrix} a_1^2 \\ a_2^2 \end{bmatrix}\right) = \phi(W^3 a^2)$$

Multi Layer Perceptron - More notations



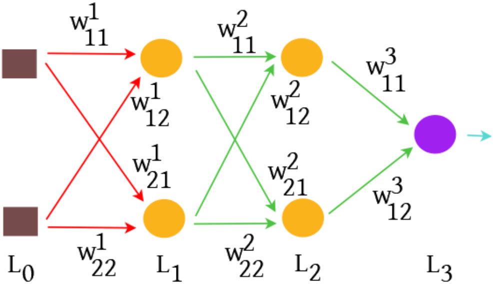
$$a^3 = \phi(W^3 a^2)$$

Multi Layer Perceptron - More notations



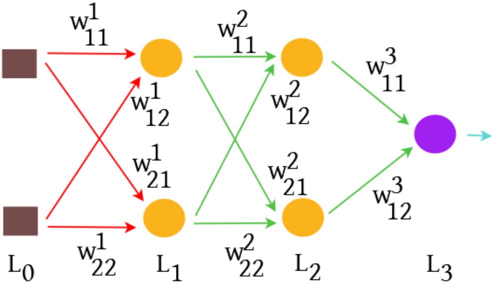
$$a^3 = \phi(W^3 a^2) = \phi(W^3 \phi(W^2 a^1))$$

Multi Layer Perceptron - More notations



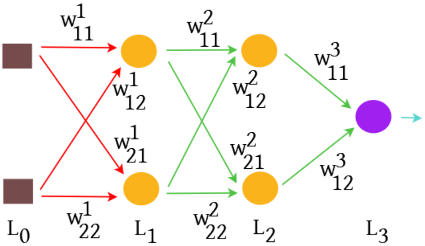
$$a^3 = \phi(W^3 a^2) = \phi(W^3 \phi(W^2 a^1)) = \phi(W^3 \phi(W^2 \phi(W^1 x)))$$

Multi Layer Perceptron - More notations



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

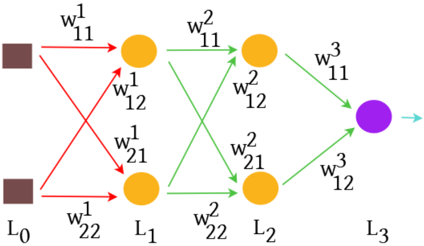
Multi Layer Perceptron - Data Perspective



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

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

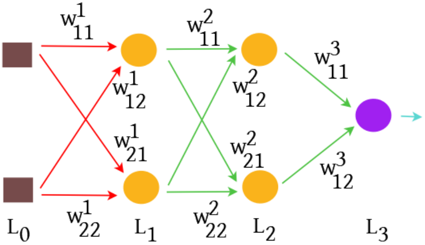
Multi Layer Perceptron - Data Perspective



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:

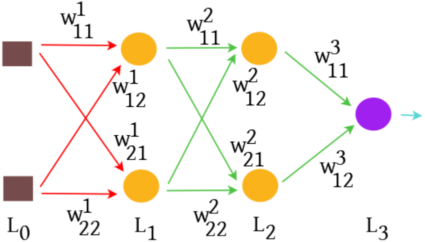
$$\hat{y} = \phi(W^3\phi(W^2\phi(W^1x))) =: \mathrm{MLP}(x)$$

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

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

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

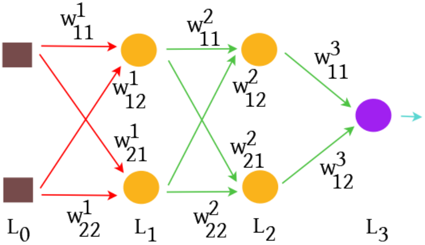


Given data (x, y), multi layer perceptron predicts:

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

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

Multi Layer Perceptron - Data Perspective



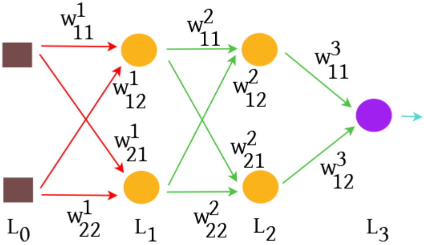
Given data (x, y), multi layer perceptron predicts:

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

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:

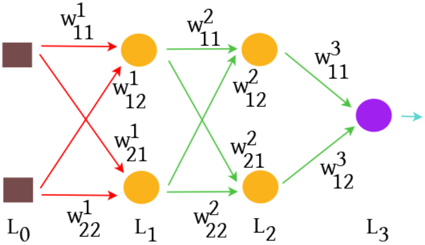
$$\hat{y} = \phi(W^3\phi(W^2\phi(W^1x))) =: \text{MLP}(x)$$

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.

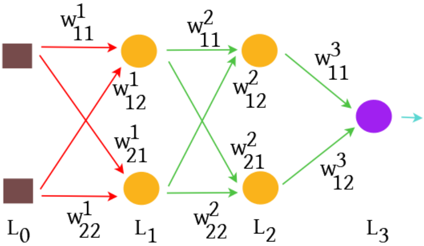
Leads to an error minimization problem.

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

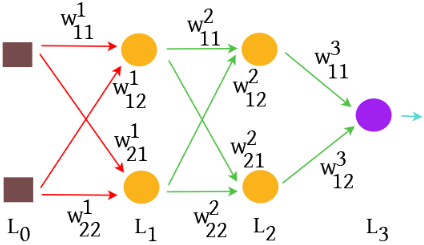


Optimization perspective

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

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

Multi Layer Perceptron - Data Perspective

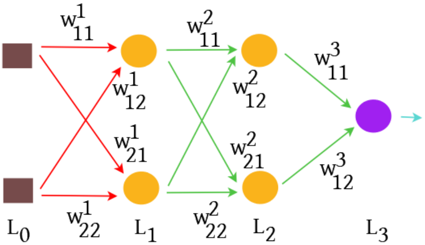


Optimization perspective

- 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

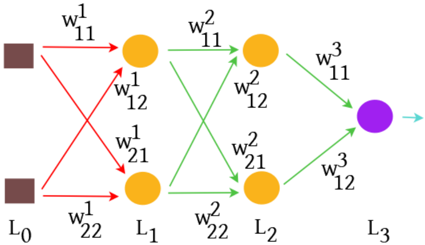


Optimization perspective

- 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



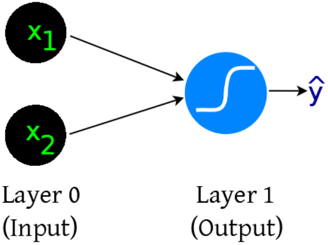
Optimization perspective

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

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

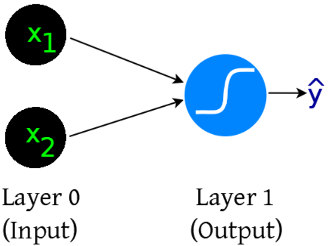
- Note: The minimization is over the weights of the MLP $W^1, \ldots, W^L$, where L denotes number of layers in MLP.

MLP - Data Perspective: A Simple Example



$$\hat{y} = \sigma\big(w_{11}^1 x_1 + w_{12}^1 x_2\big) = \frac{1}{1 + \exp\big(-[w_{11}^1 x_1 + w_{12}^1 x_2]\big)}$$

MLP - Data Perspective: A Simple Example

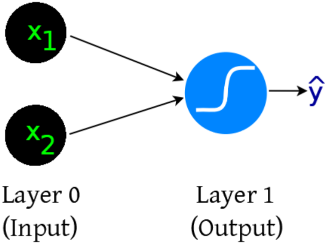


$$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2) = \frac{1}{1 + \exp\left(-[w_{11}^1 x_1 + w_{12}^1 x_2]\right)}$$

Property of 0-1 sigmoid $\sigma : \mathbb{R} \to [0, 1]$

- σ is continuous
- σ is monotonic
- $\sigma(z) \to \begin{cases} 0 & \text{if } z \to -\infty \\ 1 & \text{if } z \to +\infty \end{cases}$

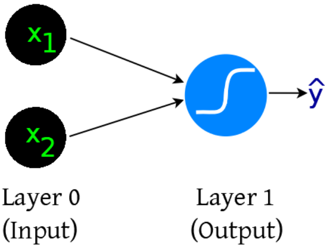
MLP - Data Perspective: A Simple Example



- Let

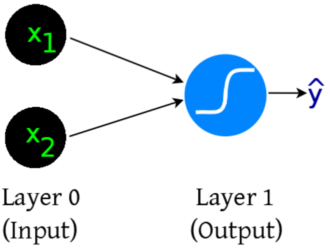
$$
\begin{aligned}
D = \{&(x^1 = (-3,-3), y^1 = 1),\\
&(x^2 = (-2,-2), y^2 = 1),\\
&(x^3 = (4,4), y^3 = 0),\\
&(x^4 = (2,-5), y^4 = 0)\}.
\end{aligned}
$$

MLP - Data Perspective: A Simple Example



x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$

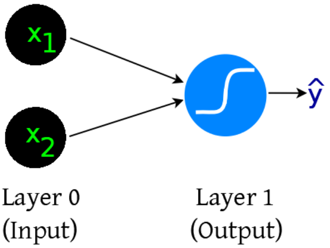
MLP - Data Perspective: A Simple Example



x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$

- **Assume:** $\text{Err}(y, \hat{y}) = (y - \hat{y})^2$.

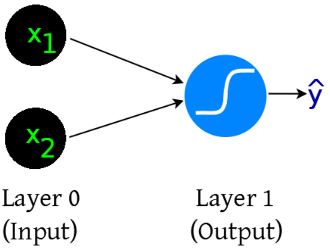
MLP - Data Perspective: A Simple Example



x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$

- **Assume:** $\mathrm{Err}(y, \hat{y}) = (y - \hat{y})^2$.
- Popularly called the squared error.

MLP - Data Perspective: A Simple Example

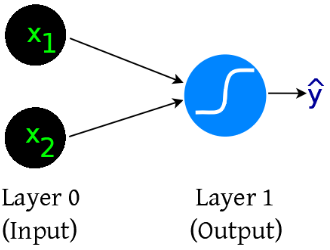


x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$

- Total error (or loss):

$$E = \sum_{i=1}^{4} e^i = \sum_{i=1}^{4} \mathrm{Err}(y^i, \hat{y}^i)$$

MLP - Data Perspective: A Simple Example

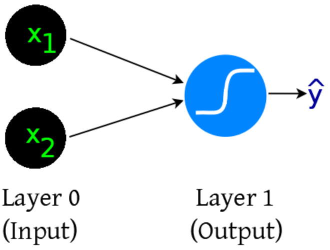


x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$

- Total error (or loss):

$$E = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

MLP - Data Perspective: A Simple Example



x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$

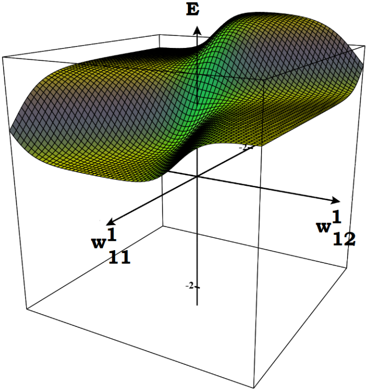
- Aim: To minimize the total error (or loss), which is

$$\min_{w_{11}^1, w_{12}^1} E = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

MLP - Data Perspective: A Simple Example

Visualizing the loss surface:

x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$



$$E = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

Optimization Concepts

General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x)$$

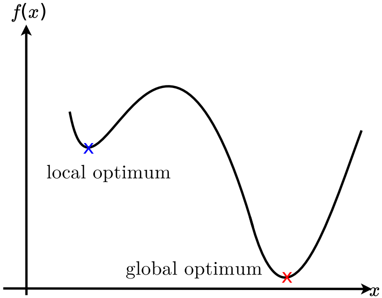
General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x)$$

- f is called objective function and $\mathcal{C}$ is called feasible set.
- Let $f^* = \min_{x \in \mathcal{C}} f(x)$ denote the **optimal objective function value**.
- **Optimal Solution Set** $S^* = \{x \in \mathcal{C} : f(x) = f^*\}$.
- Let us denote by x^* an optimal solution in S^*.

General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x) \tag{OP}$$



General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x) \tag{OP}$$

Local Optimal Solution

A solution z to (OP) is called local optimal solution if $f(z) \leq f(\hat{z})$, $\forall \hat{z} \in \mathcal{N}(z, \epsilon)$ for some $\epsilon > 0$.

Note: $\mathcal{N}(z, \epsilon)$ denotes suitable $\epsilon-$neighborhood of z.

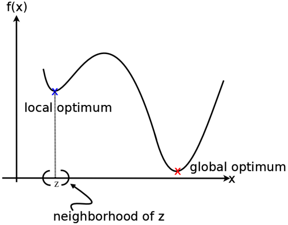
General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x) \tag{OP}$$

Local Optimal Solution

A solution z to (OP) is called local optimal solution if $f(z) \leq f(\hat{z})$, $\forall \hat{z} \in \mathcal{N}(z, \epsilon)$ for some $\epsilon > 0$.

Note: $\mathcal{N}(z, \epsilon)$ denotes suitable $\epsilon-$neighborhood of z.



General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x) \tag{OP}$$

Local Optimal Solution

A solution z to (OP) is called local optimal solution if $f(z) \leq f(\hat{z})$, $\forall \hat{z} \in \mathcal{N}(z, \epsilon)$ for some $\epsilon > 0$.

Note: $\mathcal{N}(z, \epsilon)$ denotes suitable $\epsilon-$neighborhood of z.

$\epsilon-$ Neighborhood of $z \in \mathcal{C}$

$$\mathcal{N}(z, \epsilon) = \{u \in \mathcal{C} : \text{dist}(z, u) \leq \epsilon\}.$$

General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x) \tag{OP}$$

Local Optimal Solution

A solution z to (OP) is called local optimal solution if $f(z) \leq f(\hat{z})$, $\forall \hat{z} \in \mathcal{N}(z, \epsilon)$ for some $\epsilon > 0$.

Global Optimal Solution

A solution z to (OP) is called global optimal solution if $f(z) \leq f(\hat{z})$, $\forall \hat{z} \in \mathcal{C}$.

General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x)$$

- **General Assumption:** $\mathcal{C} \subseteq \mathbb{R}^d$.

High Dimensional Representation

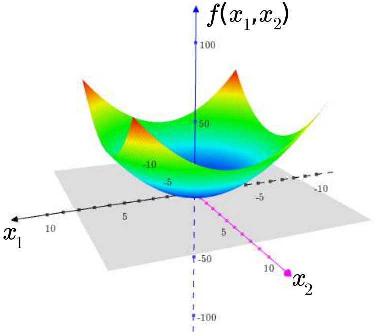


Image courtesy: www.intmath.com

- Points $x \in \mathbb{R}^d$, an Euclidean Space of dimension d.

High Dimensional Representation - Notations

- Vector Representation of $x \in \mathbb{R}^d$

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}$$

- Transpose of vector x

$$x^\top = \begin{pmatrix} x_1 & x_2 & \dots x_d \end{pmatrix}.$$

- ℓ_2 Norm of vector x

$$\|x\|_2 = \sqrt{x^\top x} = (|x_1|^2 + |x_2|^2 + \ldots + |x_d|^2)^{1/2}.$$

- ℓ_1 Norm of vector x

$$\|x\|_1 = |x_1| + |x_2| + \ldots + |x_d|.$$

Note: $|x_1|$ denotes the absolute value of x_1.

High Dimensional Representation - Notations

- **Zero vector** has all its components to be zero:

$$\begin{pmatrix}0 & 0 & \dots 0\end{pmatrix}^\top.$$

- **Non-zero vector** $(x \neq 0)$ has at least one non-zero component.
- **One vector** has all its components to be one:

$$\begin{pmatrix}1 & 1 & \dots 1\end{pmatrix}^\top.$$

High Dimensional Representation - Notations

- Gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ at a point x

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \vdots \\ \vdots \\ \frac{\partial f(x)}{\partial x_d} \end{pmatrix}$$

General Optimization Problem

$$\min_{x \in \mathcal{C}} f(x)$$

- $\mathcal{C} \subseteq \mathbb{R}^d$.
- $f : \mathcal{C} \longrightarrow \mathbb{R}$.

Directional derivative

Let $f : \mathcal{C} \longrightarrow \mathbb{R}$ be a function defined over $\mathcal{C} \subseteq \mathbb{R}^d$. Let $x \in int(\mathcal{C})$. Let $\mathbf{0} \neq d \in \mathbb{R}^d$. If the limit

$$\lim_{\alpha \downarrow 0} \frac{f(x + \alpha d) - f(x)}{\alpha}$$

exists, then it is called the directional derivative of f at x along the direction d, and is denoted by $f'(x; d)$.

Directional derivative

Interior of a set $\mathcal{C}$

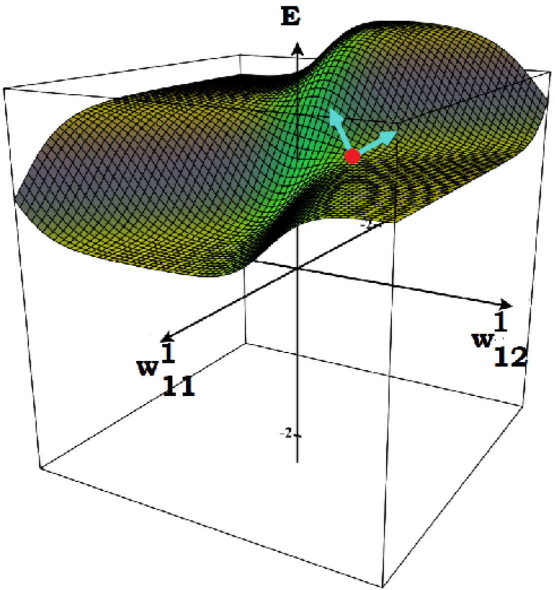
Let $\mathcal{C} \subseteq \mathbb{R}^d$. Then $int(\mathcal{C})$ is defined by:

$$int(\mathcal{C}) = \{x \in \mathcal{C} : B(x, \epsilon) \subseteq \mathcal{C}, \text{ for some } \epsilon > 0\},$$

where $B(x, \epsilon)$ is the open ball centered at x with radius ϵ given by

$$B(x, \epsilon) = \{y \in \mathcal{C} : \|x - y\| < \epsilon\}.$$

Directional derivative



Directional derivative

Let $f : \mathcal{C} \longrightarrow \mathbb{R}$ be a function defined over $\mathcal{C} \subseteq \mathbb{R}^d$. Let $x \in int(\mathcal{C})$. Let $d \neq \mathbf{0} \in \mathbb{R}^d$. If the limit

$$\lim_{\alpha \downarrow 0} \frac{f(x + \alpha d) - f(x)}{\alpha}$$

exists, then it is called the directional derivative of f at x along the direction d, and is denoted by $f'(x; d)$.

Note: If all partial derivatives of f exist at x, then $f'(x; d) = \langle \nabla f(x), d \rangle$, where $\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1} \ \ldots \ \frac{\partial f(x)}{\partial x_d} \right]^\top$.

Descent Direction

Let $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ be a continuously differentiable function over $\mathbb{R}^d$. Then a vector $\mathbf{0} \neq d \in \mathbb{R}^d$ is called a descent direction of f at x if the directional derivative of f at x is negative; that is,

$$f'(x; d) = \langle \nabla f(x), d \rangle < 0.$$

Descent Direction

Let $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ be a continuously differentiable function over $\mathbb{R}^d$. Then a vector $\mathbf{0} \neq d \in \mathbb{R}^d$ is called a descent direction of f at x if the directional derivative derivative of f at x is negative; that is,

$$f'(x; d) = \langle \nabla f(x), d \rangle < 0.$$

Note: A natural candidate for a descent direction is $d = -\nabla f(x)$.

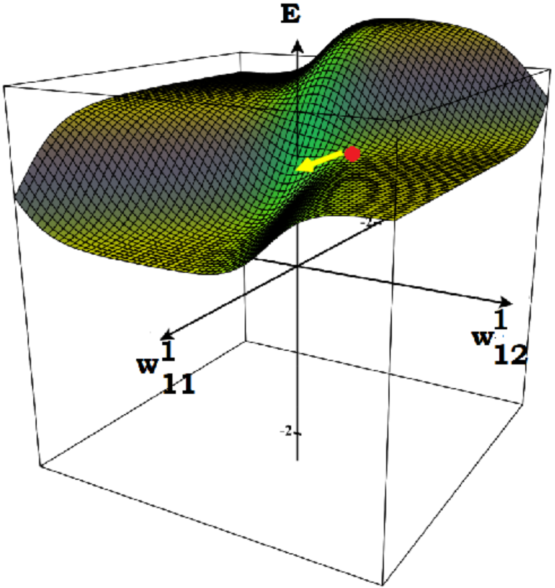
Descent Direction

Proposition

Let $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ be a continuously differentiable function over $\mathbb{R}^d$. Let $\mathbf{0} \neq d \in \mathbb{R}^d$ be a descent direction of f at x. Then there exists $\epsilon > 0$ such that $\forall \alpha \in (0, \epsilon]$ we have

$$f(x + \alpha d) < f(x).$$

Descent Direction



Descent Direction

Proposition

Let $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ be a continuously differentiable function over $\mathbb{R}^d$. Let $\mathbf{0} \neq d \in \mathbb{R}^d$ be a descent direction of f at x. Then there exists $\epsilon > 0$ such that $\forall \alpha \in (0, \epsilon]$ we have

$$f(x + \alpha d) < f(x).$$

Proof idea:

Descent Direction

Proposition

Let $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ be a continuously differentiable function over $\mathbb{R}^d$. Let $\mathbf{0} \neq d \in \mathbb{R}^d$ be a descent direction of f at x. Then there exists $\epsilon > 0$ such that $\forall \alpha \in (0, \epsilon]$ we have

$$f(x + \alpha d) < f(x).$$

Proof idea: Since $\mathbf{0} \neq d \in \mathbb{R}^d$ is a descent direction, by definition of the directional derivative we have

$$f'(x; d) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha d) - f(x)}{\alpha} < 0$$

$\implies \exists \epsilon > 0$ such that $\forall \alpha \in (0, \epsilon]$, $f(x + \alpha d) < f(x)$.

Descent Direction

Proposition

Let $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ be a continuously differentiable function over $\mathbb{R}^d$. Let $\mathbf{0} \neq d \in \mathbb{R}^d$ be a descent direction of f at x. Then there exists $\epsilon > 0$ such that $\forall \alpha \in (0, \epsilon]$ we have

$$f(x + \alpha d) < f(x).$$

Proof idea: Since $\mathbf{0} \neq d \in \mathbb{R}^d$ is a descent direction, by definition of the directional derivative we have

$$f'(x; d) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha d) - f(x)}{\alpha} < 0$$

$\implies \exists \epsilon > 0$ such that $\forall \alpha \in (0, \epsilon]$, $f(x + \alpha d) < f(x)$.

Note: If we cannot find such ϵ, d is no longer a descent direction. **Why?**

Algorithm Development using Descent Direction

Consider the general optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) \qquad \text{(GEN-OPT)}$$

where $f : \mathbb{R}^d \longrightarrow \mathbb{R}$

Algorithm to solve (GEN-OPT)

- Start with $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - Find a descent direction d^k of f at x^k and $\alpha^k > 0$ such that $f(x^k + \alpha^k d^k) < f(x^k)$.
  - $x^{k+1} = x^k + \alpha^k d^k$.
  - Check for some stopping criterion and break from loop.

Characterization Of Local Optimum

Proposition

Let $f : \mathcal{C} \longrightarrow \mathbb{R}$ be a function over the set $\mathcal{C} \subseteq \mathbb{R}^d$. Let $x^\star \in int(\mathcal{C})$ be a local optimum point of f. Let all partial derivatives of f exist at $x^\star$. Then $\nabla f(x^\star) = \mathbf{0}$.

Characterization Of Local Optimum

Proposition

Let $f : \mathcal{C} \longrightarrow \mathbb{R}$ be a function over the set $\mathcal{C} \subseteq \mathbb{R}^d$. Let $x^\star \in int(\mathcal{C})$ be a local optimum point of f. Let all partial derivatives of f exist at $x^\star$. Then $\nabla f(x^\star) = \mathbf{0}$.

Exercise: Prove this result!

Algorithm Development using Descent Direction

Consider the general optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) \qquad \text{(GEN-OPT)}$$

where $f : \mathbb{R}^d \longrightarrow \mathbb{R}$.

Algorithm to solve (GEN-OPT)

- Start with $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - Find a descent direction d^k of f at x^k and $\alpha^k > 0$ such that $f(x^k + \alpha^k d^k) < f(x^k)$.
  - $x^{k+1} = x^k + \alpha^k d^k$.
  - If $\|\nabla f(x^{k+1})\|_2 = 0$, set $x^* = x^{k+1}$, break from loop.
- Output x^*.

Algorithm Development using Descent Direction

Algorithm to solve (GEN-OPT)

- Start with $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - Find a descent direction d^k of f at x^k and $\alpha^k > 0$ such that $f(x^k + \alpha^k d^k) < f(x^k)$.
  - $x^{k+1} = x^k + \alpha^k d^k$.
  - If $\|\nabla f(x^{k+1})\|_2 = 0$, set $x^* = x^{k+1}$, break from loop.
- Output x^*.

Homework: Compare the structure of this algorithm with the Perceptron training algorithm and try to understand the perceptron update rule from an optimization perspective.

Algorithm Development using Descent Direction

Consider the general optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) \qquad \text{(GEN-OPT)}$$

where $f : \mathbb{R}^d \longrightarrow \mathbb{R}$.

Gradient Descent Algorithm to solve (GEN-OPT)

- Start with $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \dots$
  - $d^k = -\nabla f(x^k)$.
  - $\alpha^k = \text{argmin}_{\alpha > 0} f(x^k + \alpha d^k)$.
  - $x^{k+1} = x^k + \alpha^k d^k$.
  - If $\|\nabla f(x^{k+1})\|_2 = 0$, set $x^* = x^{k+1}$, break from loop.
- Output x^*.

Gradient Descent for our MLP Problem

Recall: For MLP, the loss minimization problem is:

$$\min_{w=(w_{11}^1, w_{12}^1)} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

where $E : \mathbb{R}^2 \longrightarrow \mathbb{R}$.

Gradient Descent Algorithm to solve MLP Loss Minimization Problem

- Start with $w^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - $d^k = -\nabla E(w^k)$.
  - $\alpha^k = \operatorname{argmin}_{\alpha > 0} E(w^k + \alpha d^k)$.
  - $w^{k+1} = w^k + \alpha^k d^k$.
  - If $\|\nabla E(w^{k+1})\|_2 = 0$, set $w^* = w^{k+1}$, break from loop.
- Output w^*.

Gradient Descent for our MLP Problem

Recall: For MLP, the loss minimization problem is:

$$\min_{w=(w_{11}^1, w_{12}^1)} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

Gradient Descent Algorithm to solve MLP Loss Minimization Problem

- Start with $w^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - $d^k = -\nabla E(w^k)$.
  - $\alpha^k = \text{argmin}_{\alpha>0} E(w^k + \alpha d^k)$.
  - $w^{k+1} = w^k + \alpha^k d^k$.
  - If $\|\nabla E(w^{k+1})\|_2 = 0$, set $w^* = w^{k+1}$, break from loop.
- Output w^*.

Gradient Descent for our MLP Problem

Recall: For MLP, the loss minimization problem is:

$$\min_{w=(w_{11}^1, w_{12}^1)} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

Gradient Descent Algorithm to solve MLP Loss Minimization Problem

- Start with $w^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \dots$
  - $d^k = -\sum_{i=1}^{4} \nabla e^i(w^k)$.
  - $\alpha^k = \text{argmin}_{\alpha>0} E(w^k + \alpha d^k)$.
  - $w^{k+1} = w^k + \alpha^k d^k$.
  - If $\|\nabla E(w^{k+1})\|_2 = 0$, set $w^* = w^{k+1}$, break from loop.
- Output w^*.

Gradient Descent for our MLP Problem

Recall: For MLP, the loss minimization problem is:

$$\min_{w=(w_{11}^1, w_{12}^1)} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

Gradient Descent:

- Function values $E(w^t)$ exhibit $O(1/\sqrt{k})$ convergence under minor assumptions and the assumption of existence of a local optimum.
- $O(1/k^2)$ convergence possible.
- Linear convergence also possible for strongly convex and smooth function $E(w)$.
- Arbitrary accuracy possible $|W(w^{gd}) - E(w^*)| \approx O(10^{-15})$.

Gradient Descent for our MLP Problem

Recall: For MLP, the loss minimization problem is:

$$\min_{w=(w_{11}^1, w_{12}^1)} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(- \left[w_{11}^1 x_1^i + w_{12}^1 x_2^i \right] \right)} \right)^2$$

Gradient Descent:

- Blind to structure of $E(w)$.
- Finding proper α^k at each k is computationally intensive - takes at least $O(Sd)$ time.
- Storage complexity: $O(d)$

Stochastic Gradient Descent for our MLP Problem

Recall: For MLP, the loss minimization problem is:

$$\min_{w=(w_{11}^1,w_{12}^1)} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1+\exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

Stochastic Gradient Descent Algorithm to solve MLP Loss Minimization Problem

- Start with $w^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - Choose a sample $j_k \in \{1, \ldots, 4\}$.
  - $w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.

Regularized Empirical Loss Minimization - Optimization Methods

Stochastic Gradient Descent Algorithm to solve MLP Loss Minimization Problem

- Start with $w^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \dots$
  - Choose a sample $j_k \in \{1, \dots, 4\}$.
  - $w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.

$\nabla_w e^{j_k}(w^k)$: Gradient at point w^k, of e^{i_k} with respect to w. Takes only $O(d)$ time.

Under suitable conditions on γ_k ($\sum_k \gamma_k^2 < \infty$, $\sum_k \gamma_k \to \infty$), this procedure converges **asymptotically**.
For smooth functions, $O(1/k)$ convergence possible (in theory!).
Typical choice: $\gamma_k = \frac{1}{k+1}$.

Mini-Batch Stochastic Gradient Descent for our MLP Problem

Mini-batch SGD Algorithm to solve MLP Loss Minimization Problem

- Start with $w^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - Choose a block of samples $B_k \subseteq \{1, \ldots, 4\}$.
  - $w^{k+1} \leftarrow w^k - \gamma_k \sum_{j \in B_k} \nabla_w e^j(w^k)$.

Mini-batch Stochastic Gradient Descent for our MLP Problem

Mini-batch SGD Algorithm to solve MLP Loss Minimization Problem

- Start with $w^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - Choose a block of samples $B_k \subseteq \{1, \ldots, 4\}$.
  - $w^{k+1} \leftarrow w^k - \gamma_k \sum_{j \in B_k} \nabla_w e^j(w^k)$.

- Restrictions on γ_k similar to that in SGD.
- **Asymptotic convergence** !

GD/SGD: Crucial Step

Recall: For MLP, the loss minimization problem is:

$$\min_{w=(w_{11}^1, w_{12}^1)} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

Crucial step in Gradient Descent Algorithm

$w^{k+1} = w^k - \alpha^k \sum_{i=1}^{4} \nabla e^i(w^k)$

Crucial step in Stochastic Gradient Descent Algorithm

$w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.

Crucial step in Mini-batch SGD Algorithm

$w^{k+1} \leftarrow w^k - \gamma_k \sum_{j \in B_k} \nabla_w e^j(w^k)$.

GD/SGD for MLP: Crucial Step

Recall: For MLP, the loss minimization problem is:

$$\min_{w=(w_{11}^1, w_{12}^1)} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w_{11}^1 x_1^i + w_{12}^1 x_2^i\right]\right)} \right)^2$$

Crucial step in Gradient Descent Algorithm

$w^{k+1} = w^k - \alpha^k \sum_{i=1}^4 \nabla e^i(w^k)$

Crucial step in Stochastic Gradient Descent Algorithm

$w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.

Crucial step in Mini-batch SGD Algorithm

$w^{k+1} \leftarrow w^k - \gamma_k \sum_{j \in B_k} \nabla_w e^j(w^k)$.

Note: $\nabla e^i(w^k)$, $\nabla_w e^{j_k}(w^k)$, $\nabla e^j(w^k)$ denote sample-wise gradient computation.

References:

- Cauchy, A.: Méthode générale pour la résolution des systèmes d'équations simultanées. Comptes rendus des séances de l'Académie des sciences de Paris 25, 536–538, 1847.
- H. Robbins, and D.Siegmund: A convergence theorem for non-negative almost supermartingales and some applications. Optimizing Methods in Statistics. Academic Press, 1971.

Acknowledgments:

- CalcPlot3D website for plotting.