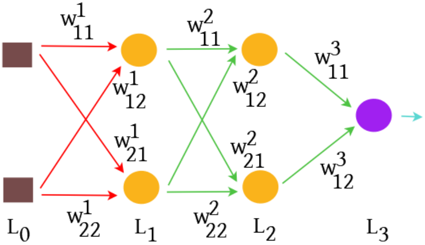
Deep Learning - Theory and Practice

IE 643
Lectures 6 & 7

Aug 14 & 18, 2025.

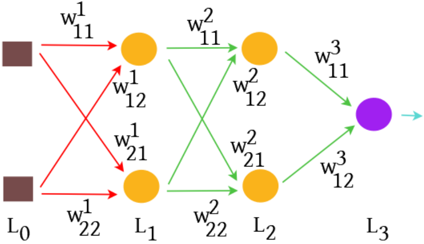
1. Recap
   - MLP-Data Perspective
2. Optimization Concepts
   - Gradient Descent
   - Stochastic Gradient Descent
   - Mini-batch SGD
3. Sample-wise Gradient Computation
   - MLP for prediction tasks

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

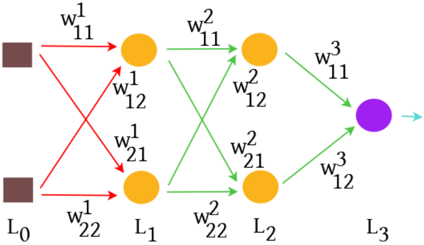


Optimization perspective

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

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

Multi Layer Perceptron - Data Perspective

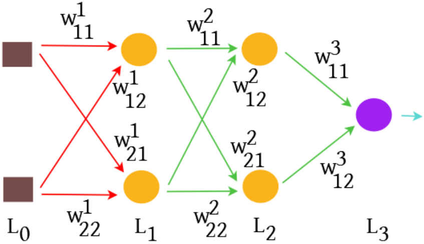


Optimization perspective

- 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

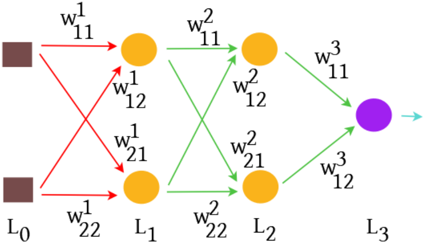


Optimization perspective

- 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



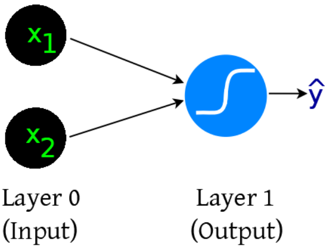
Optimization perspective

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

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

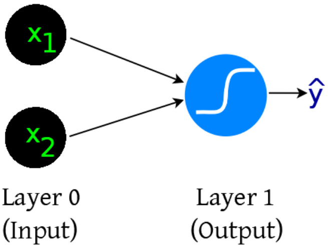
- Note: The minimization is over the weights of the MLP $W^1, \ldots, W^L$, where L denotes number of layers in MLP.

MLP - Data Perspective: A Simple Example



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

MLP - Data Perspective: A Simple Example

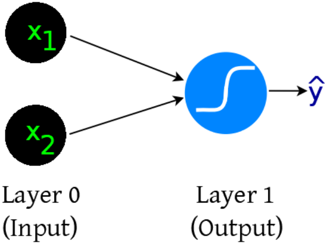


$$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2) = \frac{1}{1 + \exp\left(-[w_{11}^1 x_1 + w_{12}^1 x_2]\right)}$$

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

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

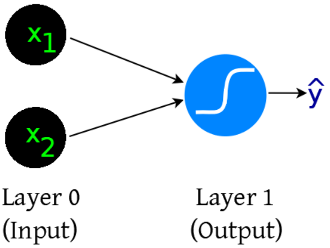
MLP - Data Perspective: A Simple Example



- Let

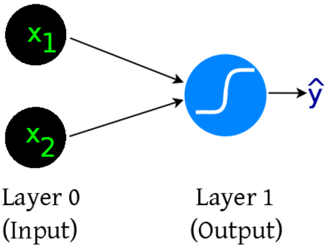
$$
\begin{aligned}
D = \{ & (x^1 = (-3, -3), y^1 = 1), \\
& (x^2 = (-2, -2), y^2 = 1), \\
& (x^3 = (4, 4), y^3 = 0), \\
& (x^4 = (2, -5), y^4 = 0)\}.
\end{aligned}
$$

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	$\sigma(2w_{11}^1 - 5w_{12}^1)$

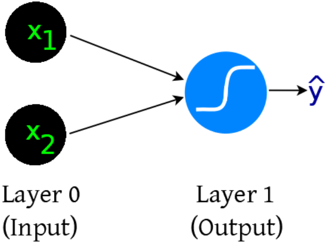
MLP - Data Perspective: A Simple Example



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

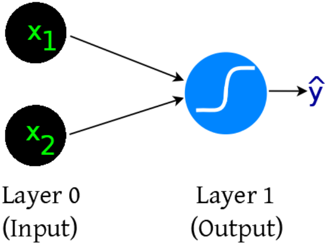
MLP - Data Perspective: A Simple Example



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

MLP - Data Perspective: A Simple Example

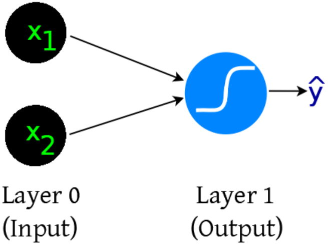


x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$

- 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

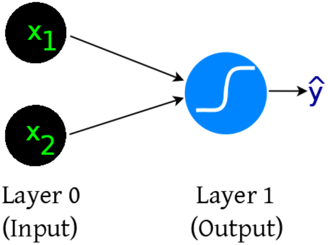


x_1	x_2	y	$\hat{y} = \sigma(w_{11}^1 x_1 + w_{12}^1 x_2)$
-3	-3	1	$\sigma(-3w_{11}^1 - 3w_{12}^1)$
-2	-2	1	$\sigma(-2w_{11}^1 - 2w_{12}^1)$
4	4	0	$\sigma(4w_{11}^1 + 4w_{12}^1)$
2	-5	0	$\sigma(2w_{11}^1 - 5w_{12}^1)$

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

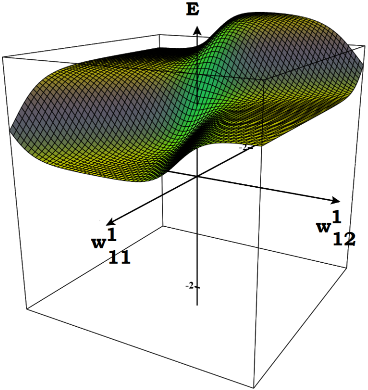
- Aim: To minimize the total error (or loss), which is

$$\min_{w^1_{11}, w^1_{12}} E = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w^1_{11}x^i_1 + w^1_{12}x^i_2\right]\right)} \right)^2$$

MLP - Data Perspective: A Simple Example

Visualizing the loss surface:

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



$$E = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w^1_{11}x^i_1 + w^1_{12}x^i_2\right]\right)} \right)^2$$

Optimization Concepts

General Optimization Problem

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

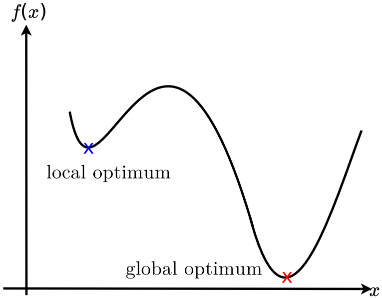
General Optimization Problem

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

- 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.

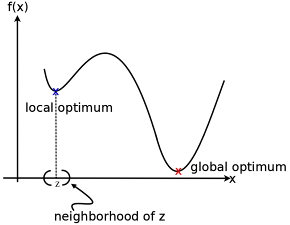
General Optimization Problem

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

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 - 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 $\mathcal{C}$

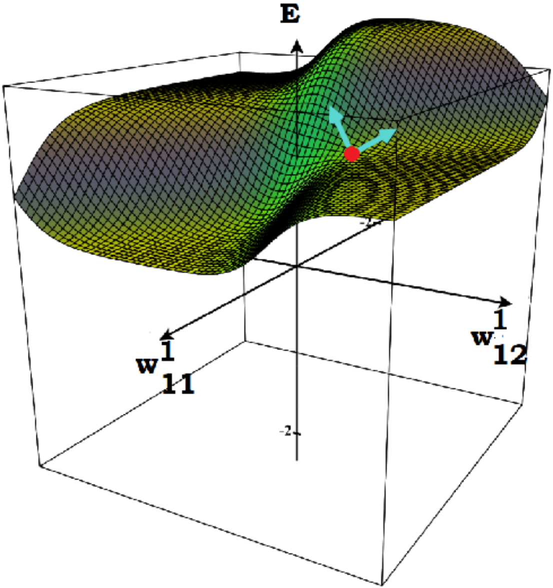
Let $\mathcal{C} \subseteq \mathbb{R}^d$. Then $int(\mathcal{C})$ is defined by:

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

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 $\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1} \ \dots \ \frac{\partial f(x)}{\partial x_d} \right]^\top$.

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 $d = -\nabla f(x)$.

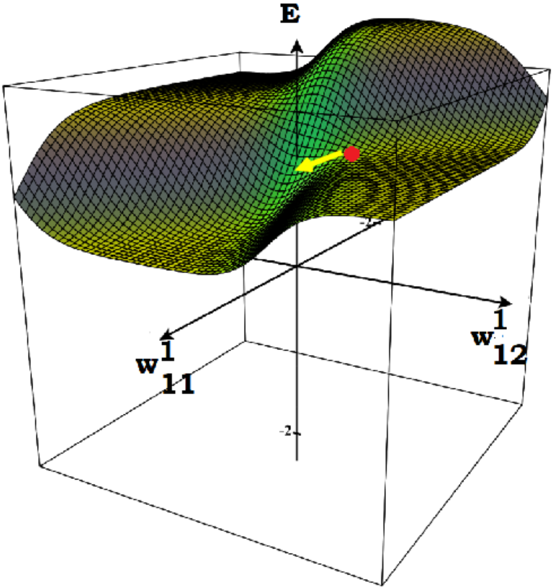
Descent Direction

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:

Algorithm Development using Descent Direction

Consider the general optimization problem:

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

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 $\nabla f(x^\star) = \mathbf{0}$.

Proof idea:

Algorithm Development using Descent Direction

Consider the general optimization problem:

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

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

Algorithm to solve (GEN-OPT)

- Start with $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \dots$
  - 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:

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

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

Gradient Descent Algorithm to solve (GEN-OPT)

- Start with $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \ldots$
  - $d^k = -\nabla f(x^k)$.
  - $\alpha^k = \text{argmin}_{\alpha > 0} f(x^k + \alpha d^k)$.
  - $x^{k+1} = x^k + \alpha^k d^k$.
  - 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:

$$\min_{w=(w^1_{11}, w^1_{12})} E(w) = \sum_{i=1}^{4} e^i(w) = \sum_{i=1}^{4} \left(y^i - \frac{1}{1 + \exp\left(-\left[w^1_{11} x^i_1 + w^1_{12} x^i_2\right]\right)} \right)^2$$

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 $k = 0, 1, 2, \ldots$
  - $d^k = -\nabla E(w^k)$.
  - $\alpha^k = \text{argmin}_{\alpha > 0} E(w^k + \alpha d^k)$.
  - $w^{k+1} = w^k + \alpha^k d^k$.
  - 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 $k = 0, 1, 2, \ldots$
  - $d^k = -\nabla E(w^k)$.
  - $\alpha^k = \text{argmin}_{\alpha > 0} E(w^k + \alpha d^k)$.
  - $w^{k+1} = w^k + \alpha^k d^k$.
  - 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 $k = 0, 1, 2, \ldots$
  - $d^k = -\sum_{i=1}^{4} \nabla e^i(w^k)$.
  - $\alpha^k = \text{argmin}_{\alpha > 0} E(w^k + \alpha d^k)$.
  - $w^{k+1} = w^k + \alpha^k d^k$.
  - 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 $k = 0, 1, 2, \ldots$
  - Choose a sample $j_k \in \{1, \ldots, 4\}$.
  - $w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.

$\nabla_w e^{j_k}(w^k)$: 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

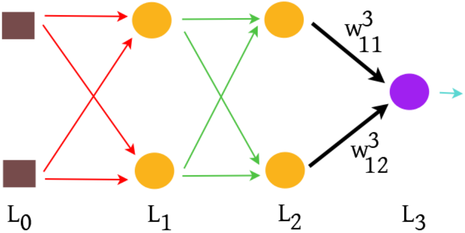
$w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.

Crucial step in Mini-batch SGD Algorithm

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

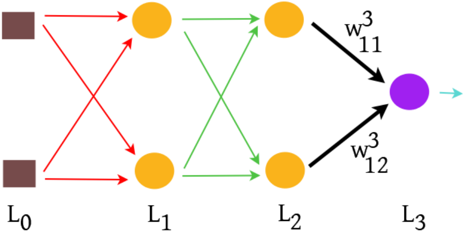
Note: $\nabla e^i(w^k)$, $\nabla_w e^{j_k}(w^k)$, $\nabla e^j(w^k)$ denote sample-wise gradient computation.

GD/SGD for MLP: Sample-wise Gradient Computation



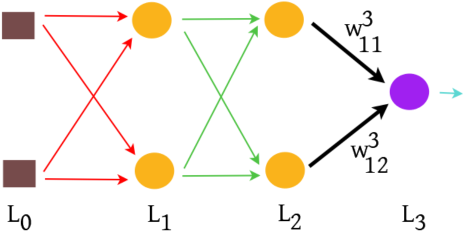
- Consider an arbitrary training sample $(x, y) \in D$.

GD/SGD for MLP: Sample-wise Gradient Computation



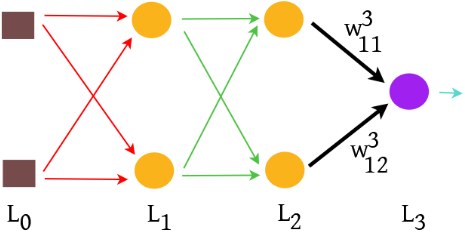
- Consider an arbitrary training sample $(x, y) \in D$.
- At layer L_3, $\hat{y} = a^3_1 = \phi\left(z^3_1\right) = \phi\left(w^3_{11}a^2_1 + w^3_{12}a^2_2\right)$.

GD/SGD for MLP: Sample-wise Gradient Computation



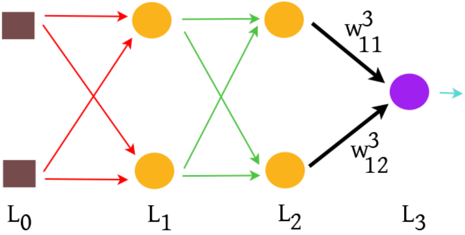
- Consider an arbitrary training sample $(x, y) \in D$.
- At layer L_3, $\hat{y} = a_1^3 = \phi\left(z_1^3\right) = \phi\left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Sample-wise error: $e = (\hat{y} - y)^2$.

GD/SGD for MLP: Sample-wise Gradient Computation



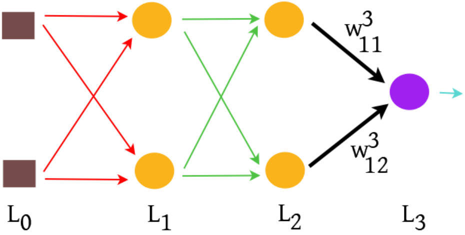
- Consider an arbitrary training sample $(x, y) \in D$.
- At layer L_3, $\hat{y} = a_1^3 = \phi\left(z_1^3\right) = \phi\left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Sample-wise error: $e = (\hat{y} - y)^2$.
- **Aim:** To find $\nabla_w e = [\nabla_{w_{11}^1} e \;\; \nabla_{w_{12}^1} e \;\; \ldots \;\; \nabla_{w_{12}^3} e]^\top$.

GD/SGD for MLP: Sample-wise Gradient Computation



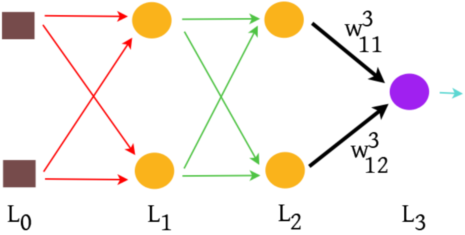
- Consider an arbitrary training sample $(x, y) \in D$.
- At layer L_3, $\hat{y} = a_1^3 = \phi\left(z_1^3\right) = \phi\left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Sample-wise error: $e = (\hat{y} - y)^2$.
- **Note:** $\nabla_{w_{11}^3} e = \frac{\partial e}{\partial z_1^3} \frac{\partial z_1^3}{\partial w_{11}^3}$.

GD/SGD for MLP: Sample-wise Gradient Computation



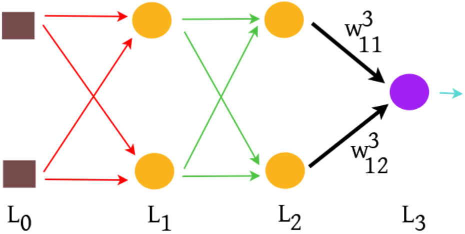
- Consider an arbitrary training sample $(x, y) \in D$.
- At layer L_3, $\hat{y} = a_1^3 = \phi\left(z_1^3\right) = \phi\left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Sample-wise error: $e = (\hat{y} - y)^2$.
- **Note:** $\nabla_{w_{11}^3} e = \frac{\partial e}{\partial z_1^3} \frac{\partial z_1^3}{\partial w_{11}^3} = \frac{\partial e}{\partial z_1^3} a_1^2$.

GD/SGD for MLP: Sample-wise Gradient Computation



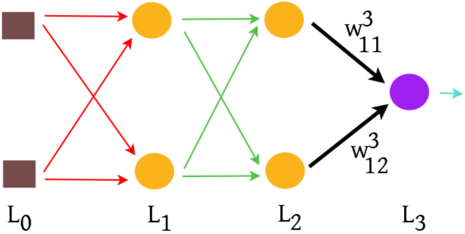
- Consider an arbitrary training sample $(x, y) \in D$.
- At layer L_3, $\hat{y} = a^3_1 = \phi\left(z^3_1\right) = \phi\left(w^3_{11}a^2_1 + w^3_{12}a^2_2\right)$.
- Sample-wise error: $e = (\hat{y} - y)^2$.
- **Note:** $\nabla_{w^3_{11}} e = \frac{\partial e}{\partial z^3_1} \frac{\partial z^3_1}{\partial w^3_{11}} = \frac{\partial e}{\partial a^3_1} \frac{\partial a^3_1}{\partial z^3_1} a^2_1$.

GD/SGD for MLP: Sample-wise Gradient Computation



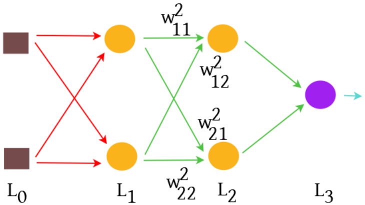
- Consider an arbitrary training sample $(x, y) \in D$.
- At layer L_3, $\hat{y} = a_1^3 = \phi\left(z_1^3\right) = \phi\left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Sample-wise error: $e = (\hat{y} - y)^2$.
- **Note:** $\nabla_{w_{11}^3} e = \frac{\partial e}{\partial z_1^3} \frac{\partial z_1^3}{\partial w_{11}^3} = \frac{\partial e}{\partial a_1^3} \frac{\partial a_1^3}{\partial z_1^3} a_1^2 = \frac{\partial e}{\partial \hat{y}} \phi'(z_1^3) a_1^2$.

GD/SGD for MLP: Sample-wise Gradient Computation



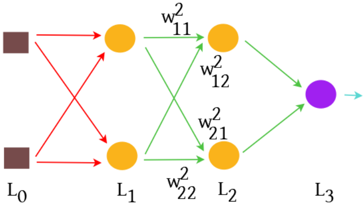
- Consider an arbitrary training sample $(x, y) \in D$.
- At layer L_3, $\hat{y} = a^3_1 = \phi\left(z^3_1\right) = \phi\left(w^3_{11} a^2_1 + w^3_{12} a^2_2\right)$.
- Sample-wise error: $e = (\hat{y} - y)^2$.
- **Note:** $\nabla_{w^3_{11}} e = \frac{\partial e}{\partial z^3_1} \frac{\partial z^3_1}{\partial w^3_{11}} = \frac{\partial e}{\partial a^3_1} \frac{\partial a^3_1}{\partial z^3_1} a^2_1 = \frac{\partial e}{\partial \hat{y}} \phi'(z^3_1) a^2_1$.
- Similarly, $\nabla_{w^3_{12}} e = \frac{\partial e}{\partial \hat{y}} \phi'(z^3_1) a^2_2$.

GD/SGD for MLP: Sample-wise Gradient Computation



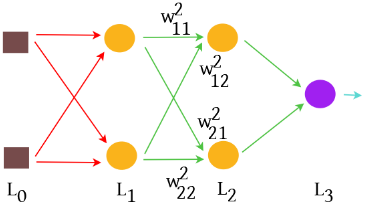
- We have at layer L_2: $a^2_1 = \phi\left(z^2_1\right) = \phi\left(w^2_{11}a^1_1 + w^2_{12}a^1_2\right)$.

GD/SGD for MLP: Sample-wise Gradient Computation



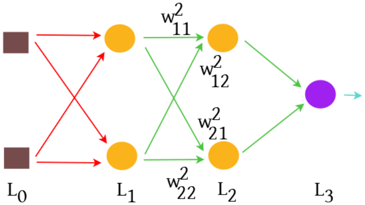
- We have at layer L_2: $a_1^2 = \phi\left(z_1^2\right) = \phi\left(w_{11}^2 a_1^1 + w_{12}^2 a_2^1\right)$.
- Hence, $\nabla_{w_{11}^2} e = \frac{\partial e}{\partial z_1^2} \frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial e}{\partial z_1^2} a_1^1$.

GD/SGD for MLP: Sample-wise Gradient Computation



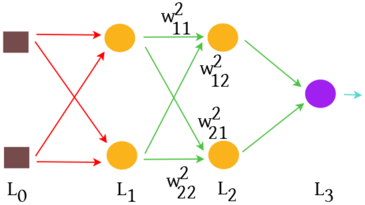
- We have at layer L_2: $a^2_1 = \phi\left(z^2_1\right) = \phi\left(w^2_{11}a^1_1 + w^2_{12}a^1_2\right)$.
- Hence, $\nabla_{w^2_{11}} e = \frac{\partial e}{\partial z^2_1}\frac{\partial z^2_1}{\partial w^2_{11}} = \frac{\partial e}{\partial z^2_1} a^1_1$.

GD/SGD for MLP: Sample-wise Gradient Computation



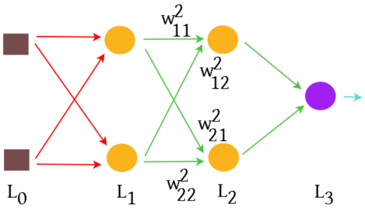
- We have at layer L_2: $a^2_1 = \phi\left(z^2_1\right) = \phi\left(w^2_{11}a^1_1 + w^2_{12}a^1_2\right)$.
- Hence, $\nabla_{w^2_{11}} e = \frac{\partial e}{\partial z^2_1}\frac{\partial z^2_1}{\partial w^2_{11}} = \frac{\partial e}{\partial z^2_1} a^1_1 = \frac{\partial e}{\partial a^2_1}\frac{\partial a^2_1}{\partial z^2_1} a^1_1 = \frac{\partial e}{\partial a^2_1}\phi'(z^2_1)a^1_1$.

GD/SGD for MLP: Sample-wise Gradient Computation



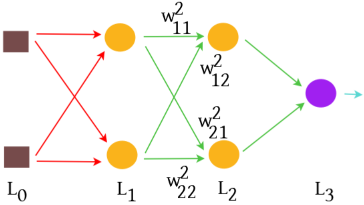
- We have at layer L_2: $a^2_1 = \phi\left(z^2_1\right) = \phi\left(w^2_{11}a^1_1 + w^2_{12}a^1_2\right)$.
- Hence, $\nabla_{w^2_{11}} e = \frac{\partial e}{\partial z^2_1}\frac{\partial z^2_1}{\partial w^2_{11}} = \frac{\partial e}{\partial z^2_1} a^1_1 = \frac{\partial e}{\partial a^2_1}\frac{\partial a^2_1}{\partial z^2_1} a^1_1 = \frac{\partial e}{\partial a^2_1}\phi'(z^2_1) a^1_1$.
- Now recall that $z^3_1 = \left(w^3_{11}a^2_1 + w^3_{12}a^2_2\right)$.

GD/SGD for MLP: Sample-wise Gradient Computation



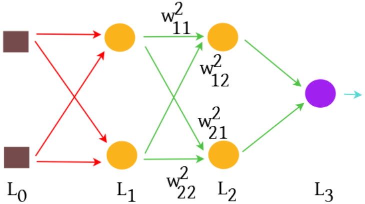
- We have at layer L_2: $a_1^2 = \phi\left(z_1^2\right) = \phi\left(w_{11}^2 a_1^1 + w_{12}^2 a_2^1\right)$.
- Hence, $\nabla_{w_{11}^2} e = \frac{\partial e}{\partial z_1^2}\frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial e}{\partial z_1^2} a_1^1 = \frac{\partial e}{\partial a_1^2}\frac{\partial a_1^2}{\partial z_1^2} a_1^1 = \frac{\partial e}{\partial a_1^2}\phi'(z_1^2) a_1^1$.
- Now recall that $z_1^3 = \left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Hence $\frac{\partial e}{\partial a_1^2} = \frac{\partial e}{\partial z_1^3}\frac{\partial z_1^3}{\partial a_1^2} = \frac{\partial e}{\partial z_1^3} w_{11}^3$.

GD/SGD for MLP: Sample-wise Gradient Computation



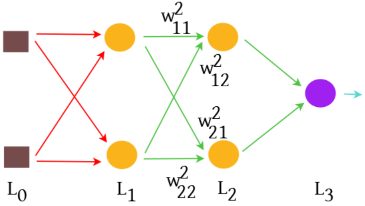
- We have at layer L_2: $a_1^2 = \phi\left(z_1^2\right) = \phi\left(w_{11}^2 a_1^1 + w_{12}^2 a_2^1\right)$.
- Hence, $\nabla_{w_{11}^2} e = \frac{\partial e}{\partial z_1^2}\frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial e}{\partial z_1^2} a_1^1 = \frac{\partial e}{\partial a_1^2}\frac{\partial a_1^2}{\partial z_1^2} a_1^1 = \frac{\partial e}{\partial a_1^2}\phi'(z_1^2)a_1^1$.
- Now recall that $z_1^3 = \left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Hence $\frac{\partial e}{\partial a_1^2} = \frac{\partial e}{\partial z_1^3}\frac{\partial z_1^3}{\partial a_1^2} = \frac{\partial e}{\partial z_1^3} w_{11}^3$.
- Recall: We have already computed $\frac{\partial e}{\partial z_1^3} = \frac{\partial e}{\partial \hat{y}}\phi'(z_1^3)$.

GD/SGD for MLP: Sample-wise Gradient Computation



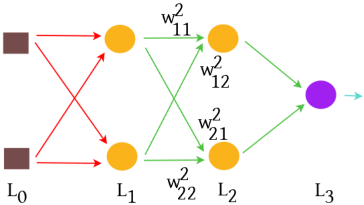
- We have at layer L_2: $a_1^2 = \phi\left(z_1^2\right) = \phi\left(w_{11}^2 a_1^1 + w_{12}^2 a_2^1\right)$.
- Hence, $\nabla_{w_{11}^2} e = \frac{\partial e}{\partial z_1^2}\frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial e}{\partial z_1^2} a_1^1 = \frac{\partial e}{\partial a_1^2}\frac{\partial a_1^2}{\partial z_1^2} a_1^1 = \frac{\partial e}{\partial a_1^2}\phi'(z_1^2) a_1^1$.
- Now recall that $z_1^3 = \left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Hence $\frac{\partial e}{\partial a_1^2} = \frac{\partial e}{\partial z_1^3}\frac{\partial z_1^3}{\partial a_1^2} = \frac{\partial e}{\partial z_1^3} w_{11}^3 = \frac{\partial e}{\partial \hat{y}}\phi'\left(z_1^3\right) w_{11}^3$.

GD/SGD for MLP: Sample-wise Gradient Computation



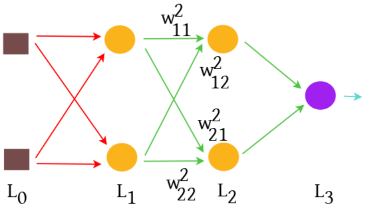
- We have at layer L_2: $a_1^2 = \phi\left(z_1^2\right) = \phi\left(w_{11}^2 a_1^1 + w_{12}^2 a_2^1\right)$.
- Hence, $\nabla_{w_{11}^2} e = \frac{\partial e}{\partial z_1^2}\frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial e}{\partial z_1^2} a_1^1 = \frac{\partial e}{\partial a_1^2}\frac{\partial a_1^2}{\partial z_1^2} a_1^1 = \frac{\partial e}{\partial a_1^2}\phi'(z_1^2) a_1^1$.
- Now recall that $z_1^3 = \left(w_{11}^3 a_1^2 + w_{12}^3 a_2^2\right)$.
- Hence $\frac{\partial e}{\partial a_1^2} = \frac{\partial e}{\partial z_1^3}\frac{\partial z_1^3}{\partial a_1^2} = \frac{\partial e}{\partial z_1^3} w_{11}^3 = \frac{\partial e}{\partial \hat{y}}\phi'(z_1^3) w_{11}^3$.
- Combining, we have $\nabla_{w_{11}^2} e = \frac{\partial e}{\partial \hat{y}}\phi'(z_1^3) w_{11}^3 \phi'(z_1^2) a_1^1$.

GD/SGD for MLP: Sample-wise Gradient Computation



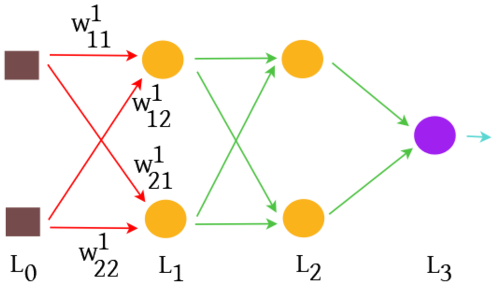
- Thus, $\nabla_{w^2_{11}} e = \frac{\partial e}{\partial \hat{y}} \phi'(z^3_1) w^3_{11} \phi'(z^2_1) a^1_1$.
- Similarly, $\nabla_{w^2_{12}} e = \frac{\partial e}{\partial \hat{y}} \phi'(z^3_1) w^3_{11} \phi'(z^2_1) a^1_2$.

GD/SGD for MLP: Sample-wise Gradient Computation



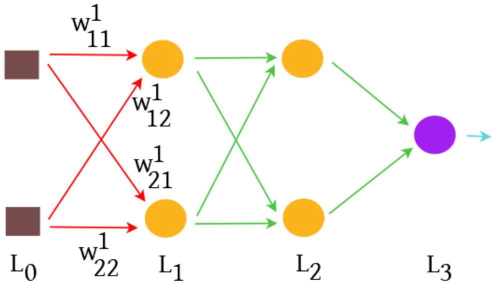
- Also, we have at layer L_2: $a_2^2 = \phi\left(z_2^2\right) = \phi\left(w_{21}^2 a_1^1 + w_{22}^2 a_2^1\right)$.
- Hence, $\nabla_{w_{21}^2} e = ?$, $\nabla_{w_{22}^2} e = ?$

GD/SGD for MLP: Sample-wise Gradient Computation



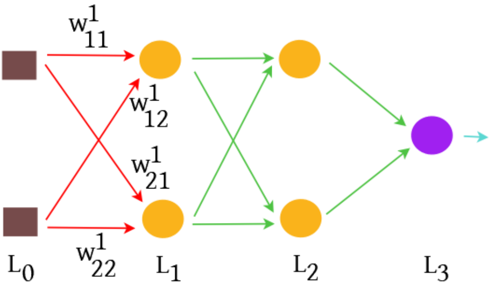
- We have at layer L_1: $a^1_1 = \phi\left(z^1_1\right) = \phi\left(w^1_{11}x_1 + w^1_{12}x_2\right)$.

GD/SGD for MLP: Sample-wise Gradient Computation



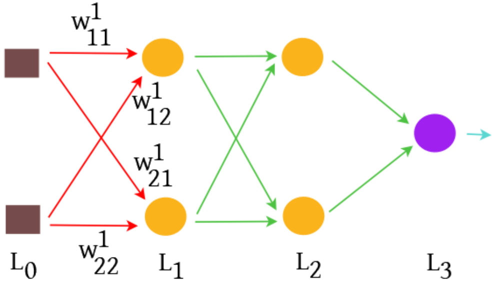
- We have at layer L_1: $a^1_1 = \phi\left(z^1_1\right) = \phi\left(w^1_{11}x_1 + w^1_{12}x_2\right)$.
- **Note:** $\nabla_{w^1_{11}} e = \frac{\partial e}{\partial z^1_1}\frac{\partial z^1_1}{\partial w^1_{11}} = \frac{\partial e}{\partial z^1_1}x_1$.

GD/SGD for MLP: Sample-wise Gradient Computation



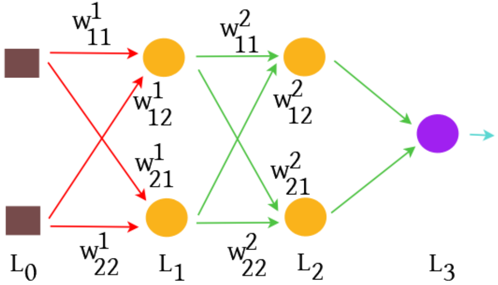
- We have at layer L_1: $a_1^1 = \phi\left(z_1^1\right) = \phi\left(w_{11}^1 x_1 + w_{12}^1 x_2\right)$.
- **Note:** $\nabla_{w_{11}^1} e = \frac{\partial e}{\partial z_1^1} x_1 = \frac{\partial e}{\partial a_1^1} \phi'(z_1^1) x_1$.

GD/SGD for MLP: Sample-wise Gradient Computation



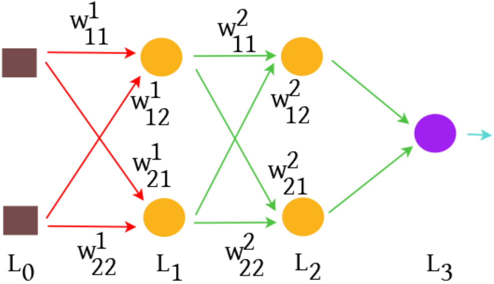
- We have at layer L_1: $a^1_1 = \phi\left(z^1_1\right) = \phi\left(w^1_{11}x_1 + w^1_{12}x_2\right)$.
- **Note:** $\nabla_{w^1_{11}} e = \frac{\partial e}{\partial z^1_1} x_1 = \frac{\partial e}{\partial a^1_1} \phi'(z^1_1) x_1$.
- Now we see that a^1_1 contributes to both z^2_1 and z^2_2.

GD/SGD for MLP: Sample-wise Gradient Computation



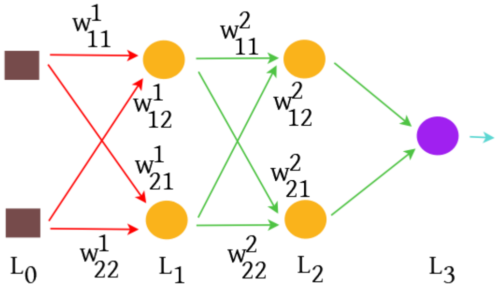
- We have at layer L_1: $a^1_1 = \phi\left(z^1_1\right) = \phi\left(w^1_{11}x_1 + w^1_{12}x_2\right)$.
- **Note:** $\nabla_{w^1_{11}} e = \frac{\partial e}{\partial z^1_1} x_1 = \frac{\partial e}{\partial a^1_1} \phi'(z^1_1) x_1$.
- Now we see that a^1_1 contributes to both z^2_1 and z^2_2.
- **Recall:** $z^2_1 = w^2_{11}a^1_1 + w^2_{12}a^1_2$ and $z^2_2 = w^2_{21}a^1_1 + w^2_{22}a^1_2$.

GD/SGD for MLP: Sample-wise Gradient Computation



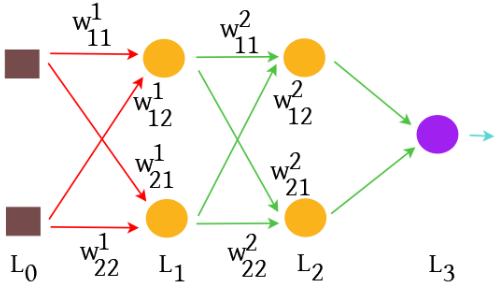
- We have at layer L_1: $a^1_1 = \phi\left(z^1_1\right) = \phi\left(w^1_{11}x_1 + w^1_{12}x_2\right)$.
- **Note:** $\nabla_{w^1_{11}} e = \frac{\partial e}{\partial z^1_1} x_1 = \frac{\partial e}{\partial a^1_1} \phi'(z^1_1) x_1$.
- Now we see that a^1_1 contributes to both z^2_1 and z^2_2.
- **Recall:** $z^2_1 = w^2_{11}a^1_1 + w^2_{12}a^1_2$ and $z^2_2 = w^2_{21}a^1_1 + w^2_{22}a^1_2$.
- Hence $\frac{\partial e}{\partial a^1_1} = \sum_{i=1}^{2} \frac{\partial e}{\partial z^2_i} \frac{\partial z^2_i}{\partial a^1_1}$.

GD/SGD for MLP: Sample-wise Gradient Computation



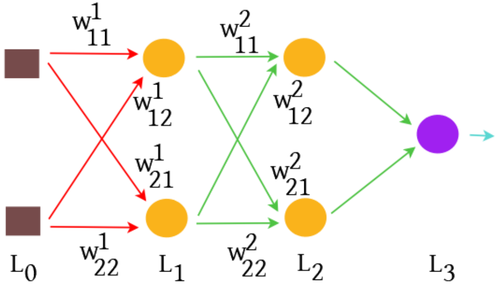
- We have at layer L_1: $a_1^1 = \phi\left(z_1^1\right) = \phi\left(w_{11}^1 x_1 + w_{12}^1 x_2\right)$.
- **Note:** $\nabla_{w_{11}^1} e = \frac{\partial e}{\partial z_1^1} x_1 = \frac{\partial e}{\partial a_1^1} \phi'(z_1^1) x_1$.
- Now we see that a_1^1 contributes to both z_1^2 and z_2^2.
- **Recall:** $z_1^2 = w_{11}^2 a_1^1 + w_{12}^2 a_2^1$ and $z_2^2 = w_{21}^2 a_1^1 + w_{22}^2 a_2^1$.
- Hence $\frac{\partial e}{\partial a_1^1} = \sum_{i=1}^2 \frac{\partial e}{\partial z_i^2} \frac{\partial z_i^2}{\partial a_1^1} = \sum_{i=1}^2 \frac{\partial e}{\partial z_i^2} w_{i1}^2$.

GD/SGD for MLP: Sample-wise Gradient Computation



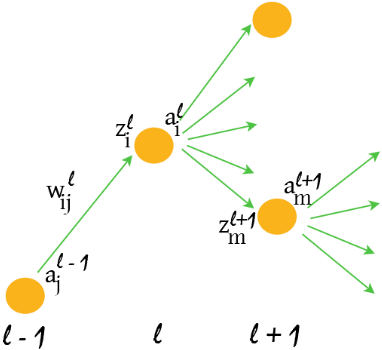
- We have at layer L_1: $a_1^1 = \phi\left(z_1^1\right) = \phi\left(w_{11}^1 x_1 + w_{12}^1 x_2\right)$.
- **Note:** $\nabla_{w_{11}^1} e = \frac{\partial e}{\partial z_1^1} x_1 = \frac{\partial e}{\partial a_1^1} \phi'(z_1^1) x_1$.
- Now we see that a_1^1 contributes to both z_1^2 and z_2^2.
- **Recall:** $z_1^2 = w_{11}^2 a_1^1 + w_{12}^2 a_2^1$ and $z_2^2 = w_{21}^2 a_1^1 + w_{22}^2 a_2^1$.
- Hence $\frac{\partial e}{\partial a_1^1} = \sum_{i=1}^2 \frac{\partial e}{\partial z_i^2} \frac{\partial z_i^2}{\partial a_1^1} = \sum_{i=1}^2 \frac{\partial e}{\partial z_i^2} w_{i1}^2$.
- Recall: We have already computed $\frac{\partial e}{\partial z_i^2}, i = 1, 2$.

GD/SGD for MLP: Sample-wise Gradient Computation



- We have at layer L_1: $a_1^1 = \phi\left(z_1^1\right) = \phi\left(w_{11}^1 x_1 + w_{12}^1 x_2\right)$.
- **Note:** $\nabla_{w_{11}^1} e = \frac{\partial e}{\partial z_1^1} x_1 = \frac{\partial e}{\partial a_1^1} \phi'(z_1^1) x_1$.
- Now we see that a_1^1 contributes to both z_1^2 and z_2^2.
- **Recall:** $z_1^2 = w_{11}^2 a_1^1 + w_{12}^2 a_2^1$ and $z_2^2 = w_{21}^2 a_1^1 + w_{22}^2 a_2^1$.
- Hence $\frac{\partial e}{\partial a_1^1} = \sum_{i=1}^2 \frac{\partial e}{\partial z_i^2} \frac{\partial z_i^2}{\partial a_1^1} = \sum_{i=1}^2 \frac{\partial e}{\partial z_i^2} w_{i1}^2$.
- Recall: We have already computed $\frac{\partial e}{\partial z_i^2} = \frac{\partial e}{\partial a_i^2} \phi'(z_i^2), i = 1, 2$.

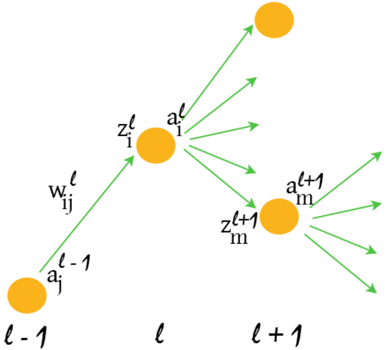
GD/SGD for MLP: Sample-wise Gradient Computation



Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1}$$

GD/SGD for MLP: Sample-wise Gradient Computation

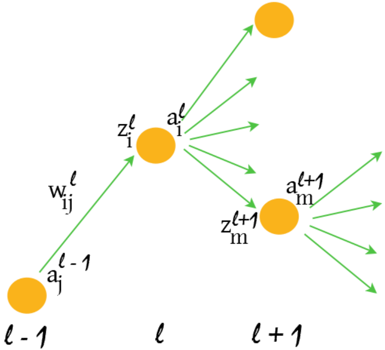


Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^\ell} = \frac{\partial e}{\partial z_i^\ell} a_j^{\ell-1}$$

$$\frac{\partial e}{\partial z_i^\ell} = \frac{\partial e}{\partial a_i^\ell} \phi'(z_i^\ell)$$

GD/SGD for MLP: Sample-wise Gradient Computation



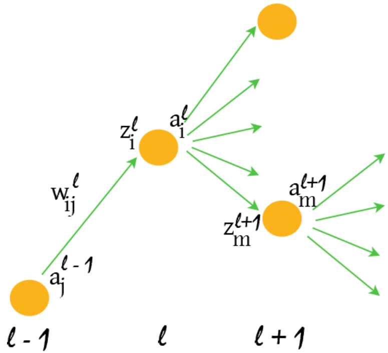
Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^\ell} = \frac{\partial e}{\partial z_i^\ell} a_j^{\ell-1}$$

$$\frac{\partial e}{\partial z_i^\ell} = \frac{\partial e}{\partial a_i^\ell} \phi'(z_i^\ell)$$

$$\frac{\partial e}{\partial a_i^\ell} = \sum_{m=1}^{N_{\ell+1}} \frac{\partial e}{\partial z_m^{\ell+1}} w_{mi}^{\ell+1}$$

GD/SGD for MLP: Sample-wise Gradient Computation



Generalized setting:

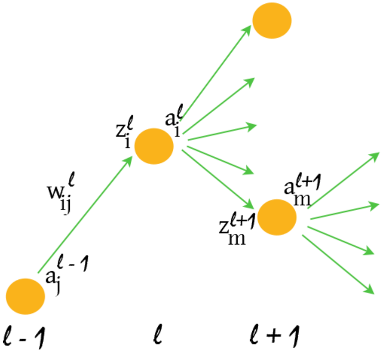
$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1}$$

$$\frac{\partial e}{\partial z_i^{\ell}} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell})$$

$$\frac{\partial e}{\partial a_i^{\ell}} = \sum_{m=1}^{N_{\ell+1}} \frac{\partial e}{\partial z_m^{\ell+1}} w_{mi}^{\ell+1}$$

$$= \sum_{m=1}^{N_{\ell+1}} \frac{\partial e}{\partial a_m^{\ell+1}} \phi'(z_m^{\ell+1}) w_{mi}^{\ell+1}$$

GD/SGD for MLP: Sample-wise Gradient Computation



Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1}$$

$$\frac{\partial e}{\partial z_i^{\ell}} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell})$$

$$\begin{aligned}
\frac{\partial e}{\partial a_i^{\ell}} &= \sum_{m=1}^{N_{\ell+1}} \frac{\partial e}{\partial z_m^{\ell+1}} w_{mi}^{\ell+1} \\
&= \sum_{m=1}^{N_{\ell+1}} \frac{\partial e}{\partial a_m^{\ell+1}} \phi'(z_m^{\ell+1}) w_{mi}^{\ell+1} \\
&= \left[\phi'(z_1^{\ell+1}) w_{11}^{\ell+1} \dots \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}1}^{\ell+1}\right] \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell+1}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_{\ell+1}}^{\ell+1}} \end{bmatrix}
\end{aligned}$$

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1}$$

$$\frac{\partial e}{\partial z_i^{\ell}} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell})$$

$$\begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^{\ell}} \end{bmatrix} = \begin{bmatrix} \phi'(z_1^{\ell+1}) w_{11}^{\ell+1} & \dots & \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}1}^{\ell+1} \\ \vdots & \dots & \vdots \\ \phi'(z_1^{\ell+1}) w_{1N_\ell}^{\ell+1} & \dots & \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}N_\ell}^{\ell+1} \end{bmatrix} \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell+1}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_{\ell+1}}^{\ell+1}} \end{bmatrix}$$

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1}$$

$$\frac{\partial e}{\partial z_i^{\ell}} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell})$$

$$\begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^{\ell}} \end{bmatrix} = \begin{bmatrix} \phi'(z_1^{\ell+1}) w_{11}^{\ell+1} & \dots & \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}1}^{\ell+1} \\ \vdots & \dots & \vdots \\ \phi'(z_1^{\ell+1}) w_{1N_\ell}^{\ell+1} & \dots & \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}N_\ell}^{\ell+1} \end{bmatrix} \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell+1}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_{\ell+1}}^{\ell+1}} \end{bmatrix}$$

$$\implies \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^{\ell}} \end{bmatrix} = \begin{bmatrix} w_{11}^{\ell+1} & \dots & w_{N_{\ell+1}1}^{\ell+1} \\ \vdots & \dots & \vdots \\ w_{1N_\ell}^{\ell+1} & \dots & w_{N_{\ell+1}N_\ell}^{\ell+1} \end{bmatrix} \begin{bmatrix} \phi'(z_1^{\ell+1}) & & \\ & \ddots & \\ & & \phi'(z_{N_{\ell+1}}^{\ell+1}) \end{bmatrix} \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell+1}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_{\ell+1}}^{\ell+1}} \end{bmatrix}$$

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1}$$

$$\frac{\partial e}{\partial z_i^{\ell}} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell})$$

$$\begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^{\ell}} \end{bmatrix} = \begin{bmatrix} \phi'(z_1^{\ell+1}) w_{11}^{\ell+1} & \dots & \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}1}^{\ell+1} \\ \vdots & \dots & \vdots \\ \phi'(z_1^{\ell+1}) w_{1N_\ell}^{\ell+1} & \dots & \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}N_\ell}^{\ell+1} \end{bmatrix} \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell+1}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_{\ell+1}}^{\ell+1}} \end{bmatrix}$$

$$\implies \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^{\ell}} \end{bmatrix} = \begin{bmatrix} w_{11}^{\ell+1} & \dots & w_{N_{\ell+1}1}^{\ell+1} \\ \vdots & \dots & \vdots \\ w_{1N_\ell}^{\ell+1} & \dots & w_{N_{\ell+1}N_\ell}^{\ell+1} \end{bmatrix} \begin{bmatrix} \phi'(z_1^{\ell+1}) & & \\ & \ddots & \\ & & \phi'(z_{N_{\ell+1}}^{\ell+1}) \end{bmatrix} \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell+1}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_{\ell+1}}^{\ell+1}} \end{bmatrix}$$

$$\delta^{\ell} = (W^{\ell+1})^{\top} \mathrm{Diag}(\phi^{\ell+1'}) \delta^{\ell+1}$$

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1}$$

$$\frac{\partial e}{\partial z_i^{\ell}} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell})$$

$$\begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^{\ell}} \end{bmatrix} = \begin{bmatrix} \phi'(z_1^{\ell+1}) w_{11}^{\ell+1} & \dots & \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}1}^{\ell+1} \\ \vdots & \dots & \vdots \\ \phi'(z_1^{\ell+1}) w_{1N_\ell}^{\ell+1} & \dots & \phi'(z_{N_{\ell+1}}^{\ell+1}) w_{N_{\ell+1}N_\ell}^{\ell+1} \end{bmatrix} \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell+1}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_{\ell+1}}^{\ell+1}} \end{bmatrix}$$

$$\implies \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^{\ell}} \end{bmatrix} = \begin{bmatrix} w_{11}^{\ell+1} & \dots & w_{N_{\ell+1}1}^{\ell+1} \\ \vdots & \dots & \vdots \\ w_{1N_\ell}^{\ell+1} & \dots & w_{N_{\ell+1}N_\ell}^{\ell+1} \end{bmatrix} \begin{bmatrix} \phi'(z_1^{\ell+1}) & & \\ & \ddots & \\ & & \phi'(z_{N_{\ell+1}}^{\ell+1}) \end{bmatrix} \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell+1}} \\ \vdots \\ \frac{\partial e}{\partial a_{N_{\ell+1}}^{\ell+1}} \end{bmatrix}$$

$$\delta^{\ell} = (W^{\ell+1})^{\top} \text{Diag}(\phi^{\ell+1'}) \delta^{\ell+1} = V^{\ell+1} \delta^{\ell+1}$$

GD/SGD for MLP: Sample-wise Gradient Computation

Now the error gradient with respect to W^ℓ can be written as:

Generalized setting:

$$\nabla_{W^\ell} e = \mathrm{Diag}(\phi^{\ell\prime})\delta^\ell(a^{\ell-1})^\top = \mathrm{Diag}(\phi^{\ell\prime})V^{\ell+1}\dots V^L\delta^L(a^{\ell-1})^\top$$

Homework: Derive this expression from the previous discussions.

GD/SGD for MLP: Sample-wise Gradient Computation

Now the error gradient with respect to W^ℓ can be written as:

Generalized setting:

$$\nabla_{W^\ell} e = \mathrm{Diag}(\phi^{\ell'})\delta^\ell (a^{\ell-1})^\top = \mathrm{Diag}(\phi^{\ell'}) V^{\ell+1} \dots V^L \delta^L (a^{\ell-1})^\top$$

Homework: Derive this expression from the previous discussions.

Homework: Assume each neuron with a bias term and compute the gradients of loss with respect to bias terms.

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\nabla_{W^\ell} e = \text{Diag}(\phi^{\ell'})\delta^\ell(a^{\ell-1})^\top = \text{Diag}(\phi^{\ell'})V^{\ell+1}\dots V^L\delta^L(a^{\ell-1})^\top$$

- **Recall:** W^ℓ represents the matrix of weights connecting layer $\ell - 1$ to layer ℓ.
- **Recall:** δ^L represents the error gradients with respect to the activations at the last layer.

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\nabla_{W^\ell} e = \mathrm{Diag}(\phi^{\ell\prime})\delta^\ell (a^{\ell-1})^\top = \mathrm{Diag}(\phi^{\ell\prime}) V^\ell \dots V^L \delta^L (a^{\ell-1})^\top$$

- **Recall:** W^ℓ represents the matrix of weights connecting layer $\ell - 1$ to layer ℓ.
- **Recall:** δ^L represents the error gradients with respect to the activations at the last layer.
- Hence, the error gradients with respect to weights W^ℓ depend on the error gradients δ^L at the last layer.

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\nabla_{W^\ell} e = \text{Diag}(\phi^{\ell\prime})\delta^\ell(a^{\ell-1})^\top = \text{Diag}(\phi^{\ell\prime})V^{\ell+1}\dots V^L\delta^L(a^{\ell-1})^\top$$

- **Recall:** W^ℓ represents the matrix of weights connecting layer $\ell - 1$ to layer ℓ.
- **Recall:** δ^L represents the error gradients with respect to the activations at the last layer.
- Hence, the error gradients with respect to weights W^ℓ depend on the error gradients δ^L at the last layer.
- **Or** the error gradients at the last layer flow back into the previous layers.

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\nabla_{W^\ell} e = \mathrm{Diag}(\phi^{\ell\prime})\delta^\ell(a^{\ell-1})^\top = \mathrm{Diag}(\phi^{\ell\prime})V^{\ell+1}\ldots V^L\delta^L(a^{\ell-1})^\top$$

- **Recall:** W^ℓ represents the matrix of weights connecting layer $\ell - 1$ to layer ℓ.
- **Recall:** δ^L represents the error gradients with respect to the activations at the last layer.
- Hence, the error gradients with respect to weights W^ℓ depend on the error gradients δ^L at the last layer.
- **Or** the error gradients at the last layer flow back into the previous layers.

This error gradient flow back is called Backpropagation!

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\nabla_{W^\ell} e = \text{Diag}(\phi^{\ell\prime})\delta^\ell(a^{\ell-1})^\top = \text{Diag}(\phi^{\ell\prime}) V^{\ell+1} \dots V^L \delta^L (a^{\ell-1})^\top$$

- If $V^{\ell+1} \dots V^L \delta^L$ leads to large values (in magnitude), then $\nabla_{W^\ell} e$ gradients can also become large (in magnitude).

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\nabla_{W^\ell} e = \mathrm{Diag}(\phi^{\ell\prime})\delta^\ell(a^{\ell-1})^\top = \mathrm{Diag}(\phi^{\ell\prime}) V^{\ell+1} \dots V^L \delta^L (a^{\ell-1})^\top$$

- If $V^{\ell+1} \dots V^L \delta^L$ leads to large values (in magnitude), then $\nabla_{W^\ell} e$ gradients can also become large (in magnitude).
- Similarly, if $V^{\ell+1} \dots V^L \delta^L$ leads to small values (in magnitude), then $\nabla_{W^\ell} e$ gradients can also approach zero (in magnitude).

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\nabla_{W^\ell} e = \text{Diag}(\phi^{\ell\prime})\delta^\ell (a^{\ell-1})^\top = \text{Diag}(\phi^{\ell\prime}) V^{\ell+1} \dots V^L \delta^L (a^{\ell-1})^\top$$

- If $V^{\ell+1} \dots V^L \delta^L$ leads to large values (in magnitude), then $\nabla_{W^\ell} e$ gradients can also become large (in magnitude). This problem is called exploding gradient problem.
- Similarly, if $V^{\ell+1} \dots V^L \delta^L$ leads to small values (in magnitude), then $\nabla_{W^\ell} e$ gradients can also approach zero (in magnitude). This problem is called vanishing gradient problem.

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\nabla_{W^\ell} e = \text{Diag}(\phi^{\ell\prime})\delta^\ell(a^{\ell-1})^\top = \text{Diag}(\phi^{\ell\prime})V^{\ell+1}\dots V^L\delta^L(a^{\ell-1})^\top$$

$$\implies \|\nabla_{W^\ell} e\|_2 \leq \|\text{Diag}(\phi^{\ell\prime})\|_2\|V^{\ell+1}\dots V^L\delta^L\|_2\|(a^{\ell-1})^\top\|_2$$

- If $V^\ell + 1 \dots V^L\delta^L$ leads to large values (in magnitude), then $\nabla_{W^\ell} e$ gradients can also become large (in magnitude). This problem is called exploding gradient problem.
- Similarly, if $V^{\ell+1} \dots V^L\delta^L$ leads to small values (in magnitude), then $\nabla_{W^\ell} e$ gradients can also approach zero (in magnitude). This problem is called vanishing gradient problem.

GD/SGD for MLP: Sample-wise Gradient Computation

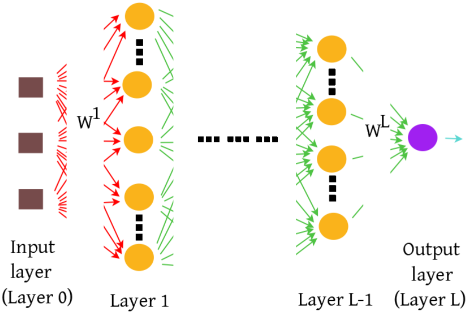
Generalized setting:

$$\nabla_{W^\ell} e = \text{Diag}(\phi^{\ell\prime})\delta^\ell (a^{\ell-1})^\top = \text{Diag}(\phi^{\ell\prime}) V^{\ell+1} \dots V^L \delta^L (a^{\ell-1})^\top$$

$$\text{recall:} \delta^L = \begin{bmatrix} \frac{\partial e}{\partial a_1^L} \\ \vdots \\ \frac{\partial e}{\partial a_{N_L}^L} \end{bmatrix}$$

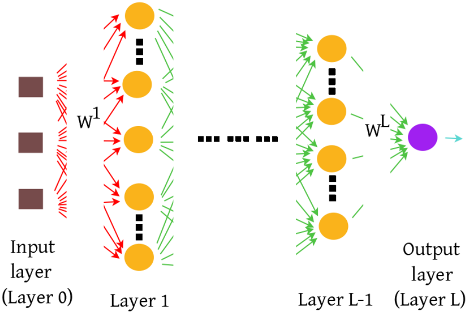
- $\frac{\partial e}{\partial a_i^L} =: \frac{\partial e}{\partial \hat{y}_i}$ denotes the gradient term with respect to a i-th neuron in the last (L-th) layer.
- So far we have considered squared error function.
- We will see more examples of constructing appropriate error functions and the corresponding gradient computation.

Multi Layer Perceptron for Prediction Tasks



- **Input:** Training Data $D = \{(x^i, y^i)\}_{i=1}^S$, $x^i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y \in \mathcal{Y}$, $\forall i \in \{1, \ldots, S\}$ and MLP architecture parametrized by weights w.
- **Aim of training MLP:** To learn a parametrized map $h_w : \mathcal{X} \to \mathcal{Y}$ such that for the training data D, we have $y^i = h_w(x^i)$, $\forall i \in \{1, \ldots, S\}$.
- **Aim of using the trained MLP model:** For an unseen sample $\hat{x} \in \mathcal{X}$, predict $\hat{y} = h_w(\hat{x}) = MLP(\hat{x}; w)$.

Multi Layer Perceptron for Prediction Tasks



Methodology for training MLP

- Design a suitable loss (or error) function $e : \mathcal{Y} \times \mathcal{Y} \to [0, +\infty)$ to compare the actual label y^i and the prediction $\hat{y}^i$ made by MLP using $e(y^i, \hat{y}^i)$, $\forall i \{1, \ldots, S\}$.
- Usually the error is parametrized by the weights w of the MLP and is denoted by $e(\hat{y}^i, y^i; w)$.
- Use Gradient descent/SGD/mini-batch SGD to minimize the total error:

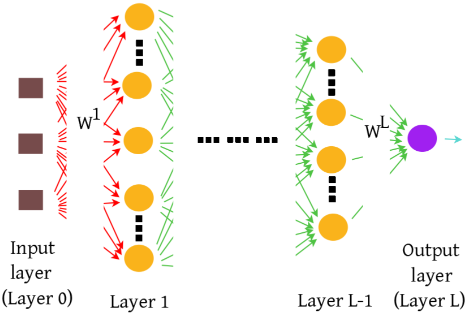
$$E = \sum_{i=1}^{S} e(\hat{y}^i, y^i; w) =: \sum_{i=1}^{S} e^i(w).$$

Stochastic Gradient Descent for training MLP

SGD Algorithm to train MLP

- **Input:** Training Data $D = \{(x^i, y^i)\}_{i=1}^S$, $x^i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y^i \in \mathcal{Y}$, $\forall i$; MLP architecture, max epochs K, learning rates γ_k, $\forall k \in \{1, \dots, K\}$.
- Start with $w^0 \in \mathbb{R}^d$.
- For $k = 0, 1, 2, \dots, K$
  - Choose a sample $j_k \in \{1, \dots, S\}$.
  - Find $\hat{y}^{j_k} = \text{MLP}(x^{j_k}; w^k)$. (forward pass)
  - Compute error $e^{j_k}(w^k)$.
  - Compute error gradient $\nabla_w e^{j_k}(w^k)$ using backpropagation.
  - Update: $w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.
- **Output:** $w^* = w^{K+1}$.

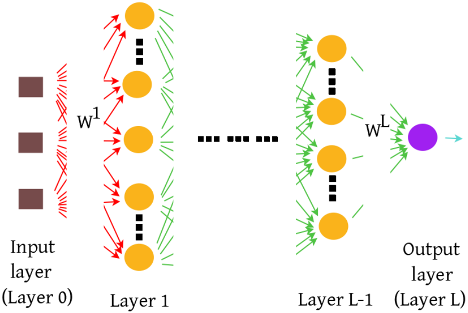
Multi Layer Perceptron for Prediction Tasks



Recall forward pass: For an arbitrary sample (x, y) from training data D, and the MLP with weights $w = (W^1, W^2 \ldots, W^L)$, the prediction $\hat{y}$ is computed using forward pass as:

$$\hat{y} = \text{MLP}(x; w) = \phi(W^L \phi(W^{L-1} \ldots \phi(W^1 x) \ldots)).$$

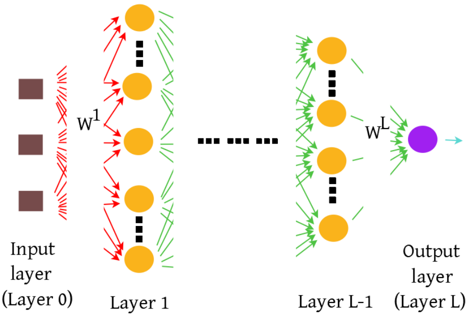
Multi Layer Perceptron for Prediction Tasks



Recall backpropagation: For an arbitrary sample (x, y) from training data D, and the MLP with weights $w = (W^1, W^2 \dots, W^L)$, the error gradient with respect to weights at ℓ-th layer is computed as:

$$\nabla_{W^\ell} e = \mathrm{Diag}(\phi^{\ell'})\delta^\ell(a^{\ell-1})^\top$$

Multi Layer Perceptron for Prediction Tasks

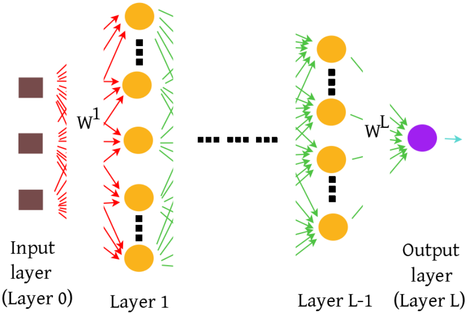


Recall backpropagation: For an arbitrary sample (x, y) from training data D, and the MLP with weights $w = (W^1, W^2 \ldots, W^L)$, the error gradient with respect to weights at ℓ-th layer is computed as:

$$\nabla_{W^\ell} e = \mathrm{Diag}(\phi^{\ell\prime})\delta^\ell (a^{\ell-1})^\top$$

$$\text{where } \mathrm{Diag}(\phi^{\ell\prime}) = \begin{bmatrix} \phi'(z_1^\ell) & & \\ & \ddots & \\ & & \phi'(z_{N_\ell}^\ell) \end{bmatrix}, \ \delta^\ell = \begin{bmatrix} \frac{\partial e}{\partial a_1^\ell} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^\ell} \end{bmatrix} \text{ and } a^{\ell-1} = \begin{bmatrix} a_1^{\ell-1} \\ \vdots \\ a_{N_{\ell-1}}^{\ell-1} \end{bmatrix}.$$

Multi Layer Perceptron for Prediction Tasks

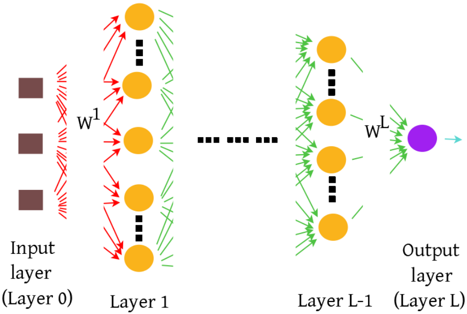


Recall backpropagation: For an arbitrary sample (x, y) from training data D, and the MLP with weights $w = (W^1, W^2 \dots, W^L)$, the error gradient with respect to weights at ℓ-th layer is computed as:

$$\nabla_{W^\ell} e = \mathrm{Diag}(\phi^{\ell\prime})\delta^\ell(a^{\ell-1})^\top = \mathrm{Diag}(\phi^{\ell\prime})V^{\ell+1}V^{\ell+2}\dots V^L\delta^L(a^{\ell-1})^\top$$

$$\text{where } V^{\ell+1} = (W^{\ell+1})^\top \mathrm{Diag}(\phi^{\ell+1\prime}).$$

Multi Layer Perceptron for Prediction Tasks



- **Task considered so far:** $\mathcal{Y} = \{+1, -1\}$.
- Corresponds to two-class (or binary) classification.
- Usually a single neuron at the last (L-th) layer of MLP, with logistic sigmoid function $\sigma : \mathbb{R} \to (0, 1)$ with $\sigma(z) = \frac{1}{1+e^{-z}}$, for some $z \in \mathbb{R}$.
- **Prediction:** $\text{MLP}(\hat{x}; w) = \sigma(W^L a^{L-1})$, followed by a thresholding function.

References

Gradient Descent introduced in:

- 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.

Idea of SGD introduced in:

- H. Robbins, and S. Monro: A stochastic approximation method. Annals of Mathematical Statistics. Vol. 22(3), pp. 400-407, 1951.

Backpropagation introduced in:

- P. J. Werbos: Beyond regression: new tools for prediction and analysis in the behavioral sciences. PhD Thesis. Harvard University, 1974.
- D. E. Rumelhart, G. E. Hinton, R. J. Williams: Learning internal representations by error propagation. Chapter in Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol I: Foundation, MIT Press, 1986.

Acknowledgments:

- CalcPlot3D website for plotting.