Deep Learning - Theory and Practice

IE 643
Lecture 9

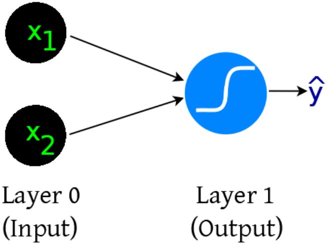
August 30, 2022.

1. Recap
   - MLP-Data Perspective
   - Stochastic Gradient Descent
   - Mini-batch SGD

2. Sample-wise Gradient Computation
   - MLP for prediction tasks

3. MLP for multi-class classification

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

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

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

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 $j_k \in \{1, \ldots, 4\}$.
  - $w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.

Stochastic Gradient Descent for our MLP Problem

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

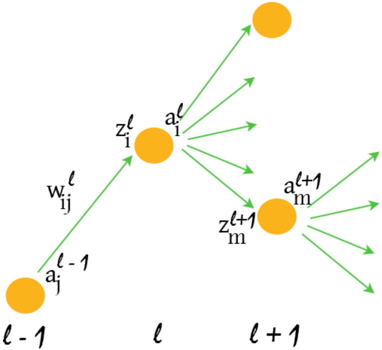
$w^{k+1} \leftarrow w^k - \gamma_k \nabla_w e^{j_k}(w^k)$.

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.

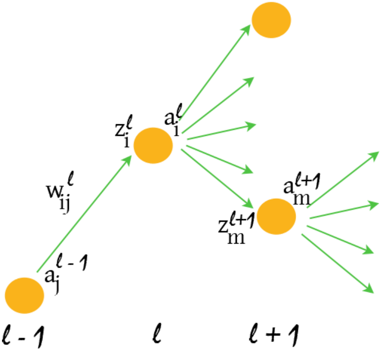
GD/SGD for MLP: Sample-wise Gradient Computation



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

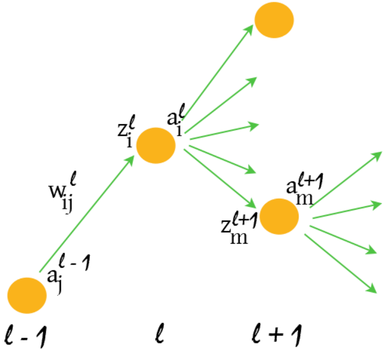


Generalized setting:

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

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

GD/SGD for MLP: Sample-wise Gradient Computation



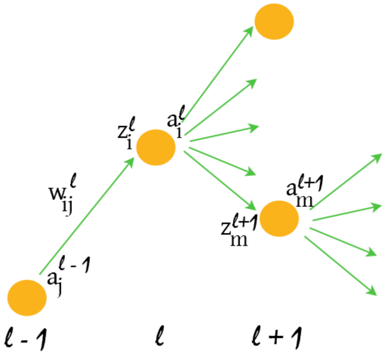
Generalized setting:

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

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

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

GD/SGD for MLP: Sample-wise Gradient Computation



Generalized setting:

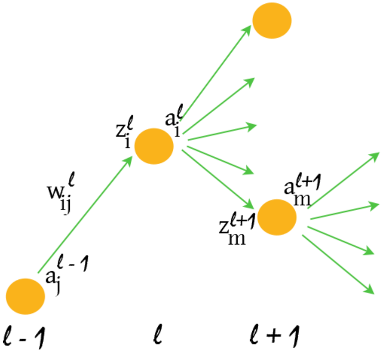
$$\frac{\partial e}{\partial w_{ij}^\ell} = \frac{\partial e}{\partial z_i^\ell} a_j^{\ell-1}$$

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

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

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

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:

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

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

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

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

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

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

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

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

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

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

$$\delta^{\ell} = (W^{\ell+1})^{\top} \mathrm{Diag}(\phi^{\ell+1'}) \delta^{\ell+1} = V^{\ell+1} \delta^{\ell+1} = V^{\ell+1} V^{\ell+2} \delta^{\ell+2}$$

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

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

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

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

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

$$\delta^{\ell} = (W^{\ell+1})^{\top} \mathrm{Diag}(\phi^{\ell+1'}) \delta^{\ell+1} = V^{\ell+1} \delta^{\ell+1} = V^{\ell+1} V^{\ell+2} \delta^{\ell+2} = V^{\ell+1} V^{\ell+2} \dots V^{L} \delta^{L}$$

Assume: The last layer in the network is L.

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell}) a_j^{\ell-1}$$

$$\implies \begin{bmatrix} \frac{\partial e}{\partial w_{1j}^{\ell}} \\ \vdots \\ \frac{\partial e}{\partial w_{N_\ell j}^{\ell}} \end{bmatrix} = \begin{bmatrix} \frac{\partial e}{\partial a_1^{\ell}} \phi'(z_1^{\ell}) a_j^{\ell-1} \\ \vdots \\ \frac{\partial e}{\partial a_{N_\ell}^{\ell}} \phi'(z_{N_\ell}^{\ell}) a_j^{\ell-1} \end{bmatrix}$$

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell}) a_j^{\ell-1}$$

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

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell}) a_j^{\ell-1}$$

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

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

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell}) a_j^{\ell-1}$$

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

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

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

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell}) a_j^{\ell-1}$$

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

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

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

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

$$\frac{\partial e}{\partial w_{ij}^{\ell}} = \frac{\partial e}{\partial z_i^{\ell}} a_j^{\ell-1} = \frac{\partial e}{\partial a_i^{\ell}} \phi'(z_i^{\ell}) a_j^{\ell-1}$$

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

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

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

Homework: 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:

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

- **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.

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:

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

- **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.
- 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:

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

- **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.

This error gradient flow back is called Backpropagation!

GD/SGD for MLP: Sample-wise Gradient Computation

Generalized setting:

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

- If $V^{\ell+1} \ldots V^L \delta^L$ 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:

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

- 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).
- 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:

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

- 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:

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

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

- 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

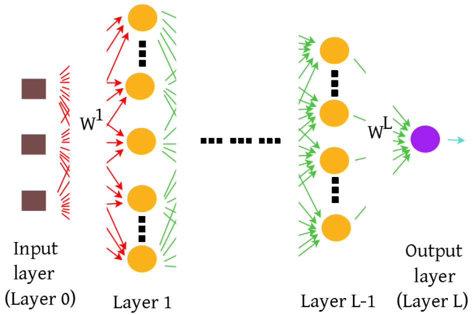
Generalized setting:

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

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

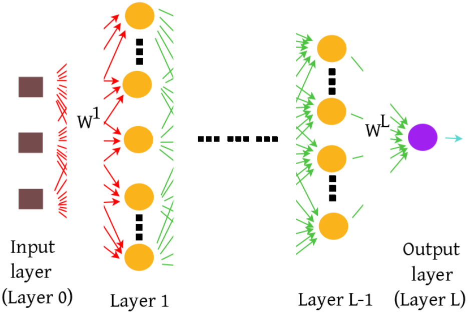
- $\frac{\partial e}{\partial a_i^L} =: \frac{\partial e}{\partial \hat{y}_i}$ denotes the gradient term with respect to a i-th neuron in the last (L-th) layer.
- So far we have considered squared error function.
- We will see more examples of constructing appropriate error functions and the corresponding 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 $e(y^i, \hat{y}^i)$, $\forall i \{1, \dots, S\}$.
- 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:

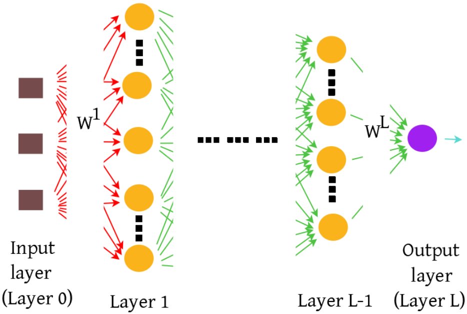
$$E = \sum_{i=1}^{S} e(\hat{y}^i, y^i; w) =: \sum_{i=1}^{S} e^i(w).$$

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:** $w^* = w^{K+1}$.

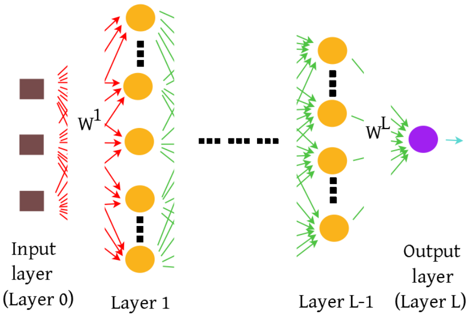
Multi Layer Perceptron for Prediction Tasks



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:

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

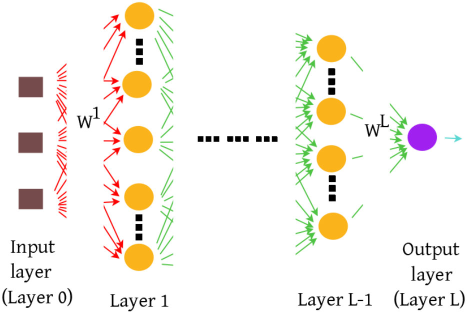
Multi Layer Perceptron for Prediction Tasks



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

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

Multi Layer Perceptron for Prediction Tasks

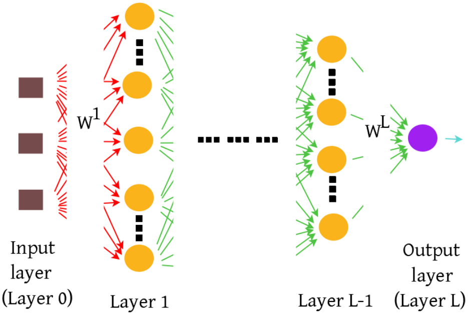


Recall backpropagation: For an arbitrary sample (x, y) from training data D, and the MLP with weights $w = (W^1, W^2 \dots, W^L)$, the error gradient with respect to weights at ℓ-th layer is computed as:

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

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

Multi Layer Perceptron for Prediction Tasks

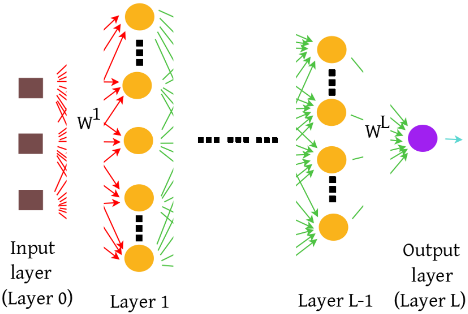


Recall backpropagation: For an arbitrary sample (x, y) from training data D, and the MLP with weights $w = (W^1, W^2 \dots, W^L)$, the error gradient with respect to weights at ℓ-th layer is computed as:

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

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

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:** $\mathrm{MLP}(\hat{x}; w) = \sigma(W^L a^{L-1})$, followed by a thresholding function.

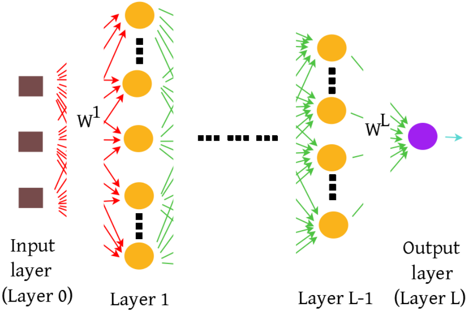
MLP for multi-class classification

- **Input:** Training Data $D = \{(x^i, y^i)\}_{i=1}^{S}$, $x^i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y^i \in \mathcal{Y}$, $\forall i \in \{1, \ldots, S\}$ and MLP architecture parametrized by weights w.
- **New Task:** $\mathcal{Y} = \{1, \ldots, C\}, C \geq 2$.
- Corresponds to multi-class classification.

Question 1: What is a suitable architecture for the MLP's last (or output) layer?

Question 2: What is a suitable loss (or error) function?

MLP for multi-class classification



Question 1: Can the same MLP architecture with single output neuron used in binary classification be used for multi-class classification?

Question 2: Can the same logistic sigmoidal activation function for the output neuron used in binary classification be used for multi-class classification?

MLP for multi-class classification

We will use the following approach for multi-class classification:

- **Input:** Training Data $D = \{(x^i, y^i)\}_{i=1}^{S}$, $x^i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y^i \in \mathcal{Y}$, $\forall i \in \{1, \dots, S\}$ and MLP architecture parametrized by weights w.
- **New Task:** $\mathcal{Y} = \{1, \dots, C\}, C \geq 2$ corresponds to multi-class classification.
- Transform $y = c$ to $y^{onehotenc} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.
- **Note:** $y^{onehotenc} \in \{0, 1\}^C$ corresponding to $y = c \in \mathcal{Y}$ has a 1 at c-th coordinate, and other entries as zeros.
- $y^{onehotenc}$ is called the one-hot encoding of y.

MLP for multi-class classification

We will use the following approach for multi-class classification:

- **Input:** Training Data $D = \{(x^i, y^i)\}_{i=1}^S$, $x^i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y^i \in \mathcal{Y}$, $\forall i \in \{1, \ldots, S\}$ and MLP architecture parametrized by weights w.

- **New Task:** $\mathcal{Y} = \{1, \ldots, C\}, C \geq 2$ corresponds to multi-class classification.

- Transform $y = c$ to $y^{onehotenc} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.

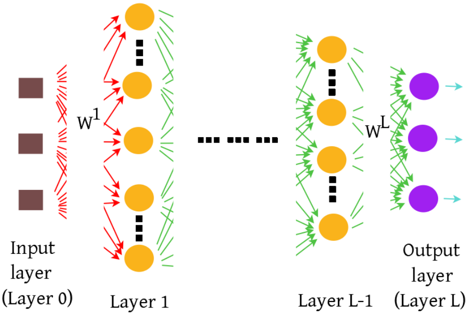
- **Note:** $y^{onehotenc} \in \{0, 1\}^C$ corresponding to $y = c \in \mathcal{Y}$ has a 1 at c-th coordinate, and other entires as zeros.

- $y^{onehotenc}$ is called the one-hot encoding of y.
- $y^{onehotenc}$ for $y = c$ corresponds to a discrete probability distribution with its entire mass concentrated at the c-th coordinate.

MLP for multi-class classification

What change can be made to the network architecture so that the MLP outputs a discrete probability distribution?

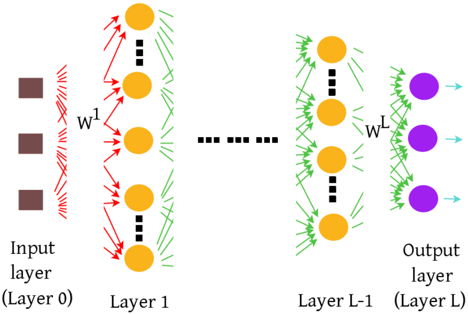
Step 1: Since $\mathcal{Y} = \{1, \ldots, C\}$, output layer of MLP to contain C neurons.



MLP for multi-class classification

What change can be made to the network architecture so that the MLP outputs a discrete probability distribution?

Step 1: Since $\mathcal{Y} = \{1, \ldots, C\}$, output layer of MLP to contain C neurons.

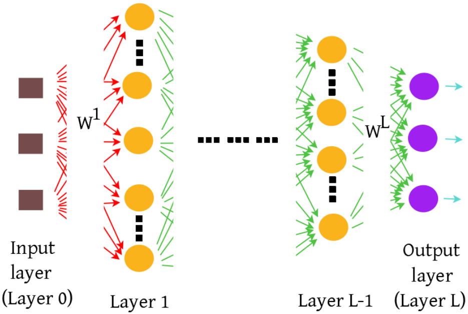


However activation functions of output neurons might be arbitrary!

MLP for multi-class classification

What change can be made to the network architecture so that the MLP outputs a discrete probability distribution?

Step 1: Since $\mathcal{Y} = \{1, \ldots, C\}$, output layer of MLP to contain C neurons.



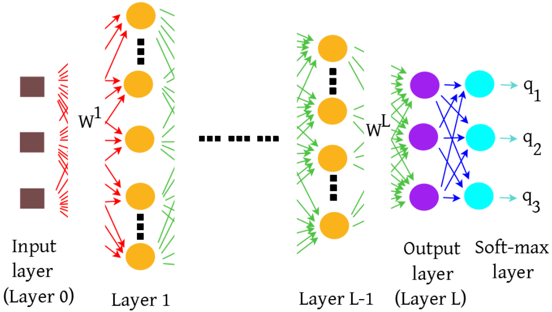
However activation functions of output neurons might be arbitrary!

How do we get probabilities as outputs?

MLP for multi-class classification

How do we get probabilities as outputs?

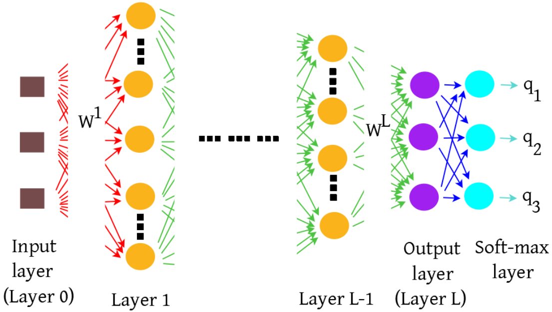
Step 2: Perform a soft-max function over the outputs from output layer so that the outputs are transformed into probabilities.



MLP for multi-class classification

How do we get probabilities as outputs?

Step 2: Perform a soft-max function over the outputs from output layer so that the outputs are transformed into probabilities.



What is a soft-max function?

MLP for multi-class classification

What is a soft-max function?

- Given arbitrary activations $a_1^L, a_2^L, \ldots, a_C^L$ from an output layer (L-th layer), how do we get probabilities?
- Perform the following transformation:

$$q_j = \frac{\exp(a_j^L)}{\sum_{r=1}^{C} \exp\left(a_r^L\right)}, \ \forall j = 1, \ldots, C.$$

- $q_1, \ldots, q_C$ form a discrete probability distribution. **(Verify this claim!)**

The transformation used to obtain the probabilities q_j is called the soft-max function.

MLP for multi-class classification

Now that the MLP outputs a discrete probability distribution, how do we compare the one-hot encoding and the output distribution?

- We will use the popular divergence measure called Kullback-Liebler divergence (or KL-divergence).
- Given two discrete probability distributions $p = (p_1, \ldots, p_C)$ and $q = (q_1, \ldots, q_C)$, where $q_j > 0 \ \forall j = 1, \ldots, C$, KL-divergence between p and q is defined as:

$$KL(p||q) = \sum_{j=1}^{C} p_j \log \frac{p_j}{q_j}.$$

- **Note:** The distribution p is usually called the true distribution and the distribution q is called the predicted distribution.
- Does the soft-max function give predictions $q_j > 0$, $j = 1, \ldots, C$?

MLP for multi-class classification: KL Divergence

Some useful properties of KL-divergence:

- KL-divergence is **not** a distance function.

MLP for multi-class classification: KL Divergence

Some useful properties of KL-divergence:

- KL-divergence is **not** a distance function.
- **Recall:** A distance function $d : X \times X \rightarrow [0, \infty)$ has the following properties:
  - $d(x, x) = 0, \ \forall x \in X$ (identity of indistinguishables)
  - $d(x, y) = d(y, x), \forall x, y \in X$ (Symmetry)
  - $d(x, z) \leq d(x, y) + d(y, z), \forall x, y, z \in X$ (triangle inequality)

MLP for multi-class classification: KL Divergence

Some useful properties of KL-divergence:

- KL-divergence is **not** a distance function.
- **Recall:** A distance function $d : X \times X \to [0, \infty)$ has the following properties:
  - $d(x, x) = 0, \ \forall x \in X$ (identity of indistinguishables)
  - $d(x, y) = d(y, x), \forall x, y \in X$ (Symmetry)
  - $d(x, z) \leq d(x, y) + d(y, z), \forall x, y, z \in X$ (triangle inequality)
- KL-divergence does not obey symmetry property.
  - Simple example: compute $KL(p||q)$ and $KL(q||p)$ for $p = (1/4, 3/4)$ and $q = (1/2, 1/2)$.

MLP for multi-class classification: KL Divergence

Some useful properties of KL-divergence:

- KL-divergence is **not** a distance function.
- **Recall:** A distance function $d : X \times X \to [0, \infty]$ has the following properties:
  - $d(x, x) = 0, \ \forall x \in X$ (identity of indistinguishables)
  - $d(x, y) = d(y, x), \forall x, y \in X$ (Symmetry)
  - $d(x, z) \leq d(x, y) + d(y, z), \forall x, y, z \in X$ (triangle inequality)
- Does KL-divergence obey triangle inequality?

MLP for multi-class classification: KL Divergence

Some useful properties of KL-divergence:

- For two discrete probability distributions $p = (p_1, p_2, \ldots, p_C)$ and $q = (q_1, q_2, \ldots, q_C)$, $q_j > 0, \ \forall j = 1, \ldots, C$, $KL(p||q) \geq 0$.

MLP for multi-class classification: KL Divergence

KL-Divergence: Equivalent Representation

- Given two discrete probability distributions $p = (p_1, \ldots, p_C)$ and $q = (q_1, \ldots, q_C)$, where $q_j > 0 \; \forall j = 1, \ldots, C$, KL-divergence between p and q is defined as:

$$KL(p||q) = \sum_{j=1}^{C} p_j \log \frac{p_j}{q_j} = \sum_{j=1}^{C} p_j \log p_j - \sum_{j=1}^{C} p_j \log q_j.$$

MLP for multi-class classification: KL Divergence

KL-Divergence: Equivalent Representation

- Given two discrete probability distributions $p = (p_1, \ldots, p_C)$ and $q = (q_1, \ldots, q_C)$, where $q_j > 0 \; \forall j = 1, \ldots, C$, KL-divergence between p and q is defined as:

$$KL(p||q) = \sum_{j=1}^{C} p_j \log \frac{p_j}{q_j} = \sum_{j=1}^{C} p_j \log p_j - \sum_{j=1}^{C} p_j \log q_j.$$

  Note: $\sum_{j=1}^{C} p_j \log p_j$ is called negative entropy associated with distribution p (denoted by $NE(p)$) and $-\sum_{j=1}^{C} p_j \log q_j$ is called cross-entropy between p and q (denoted by $CE(p, q)$).

- Hence $KL(p||q) = NE(p) + CE(p, q)$.