AIL721: Deep Learning

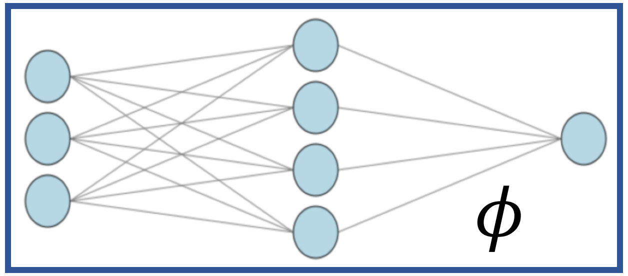
Instructor: James Arambam

ScAI | Yardi School of Artificial Intelligence
Indian Institute of Technology Delhi

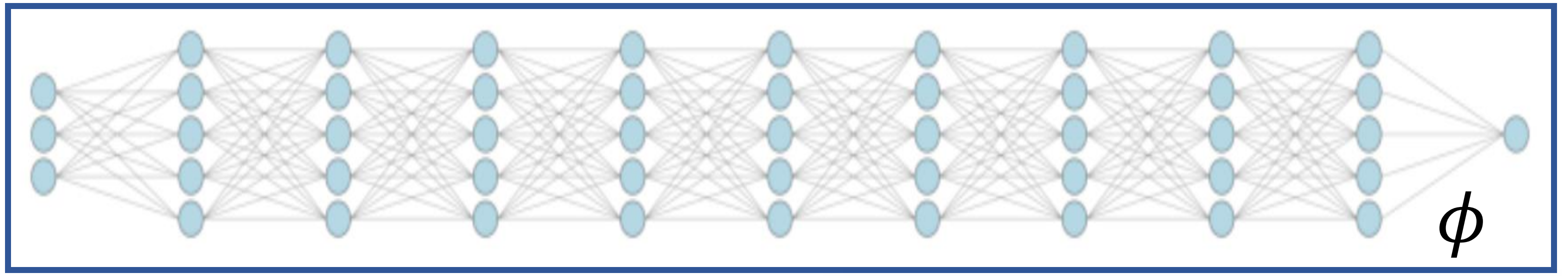
Summary

ScAI

❑ Shallow Neural Network



❑ Deep Neural Network



❑ Loss Function

Least Square Error Loss:

$$L[\phi] \quad = \quad \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2$$

Negative Loglikelihood Loss:

$$L[\boldsymbol{\phi}] = \left[-\sum_{i=1}^{I} \log\left[Pr(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]) \right] \right]$$

Cross Entropy Loss:

Next Step: Optimization

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[L(\phi) \right]$$

Optimization

ScAI

- **Optimization**

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[L(\phi) \right]$$

- Find the parameter that **minimizes** the loss function.
- This is known as the **learning or training**.

Training data: $(x_1, y_1), (x_2, y_2), \ldots, (x_I, y_I)$

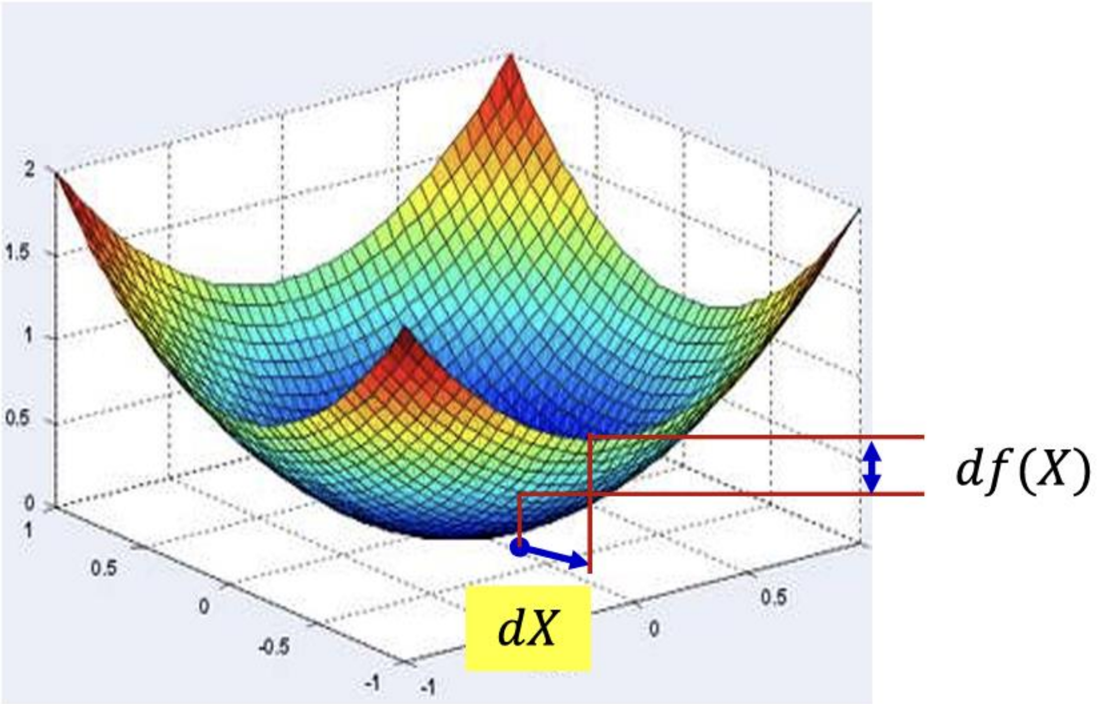
$$\begin{aligned} L[\phi] &= \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2 \\ &= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2 \end{aligned}$$

ScAI

Gradient

ScAI

Gradient



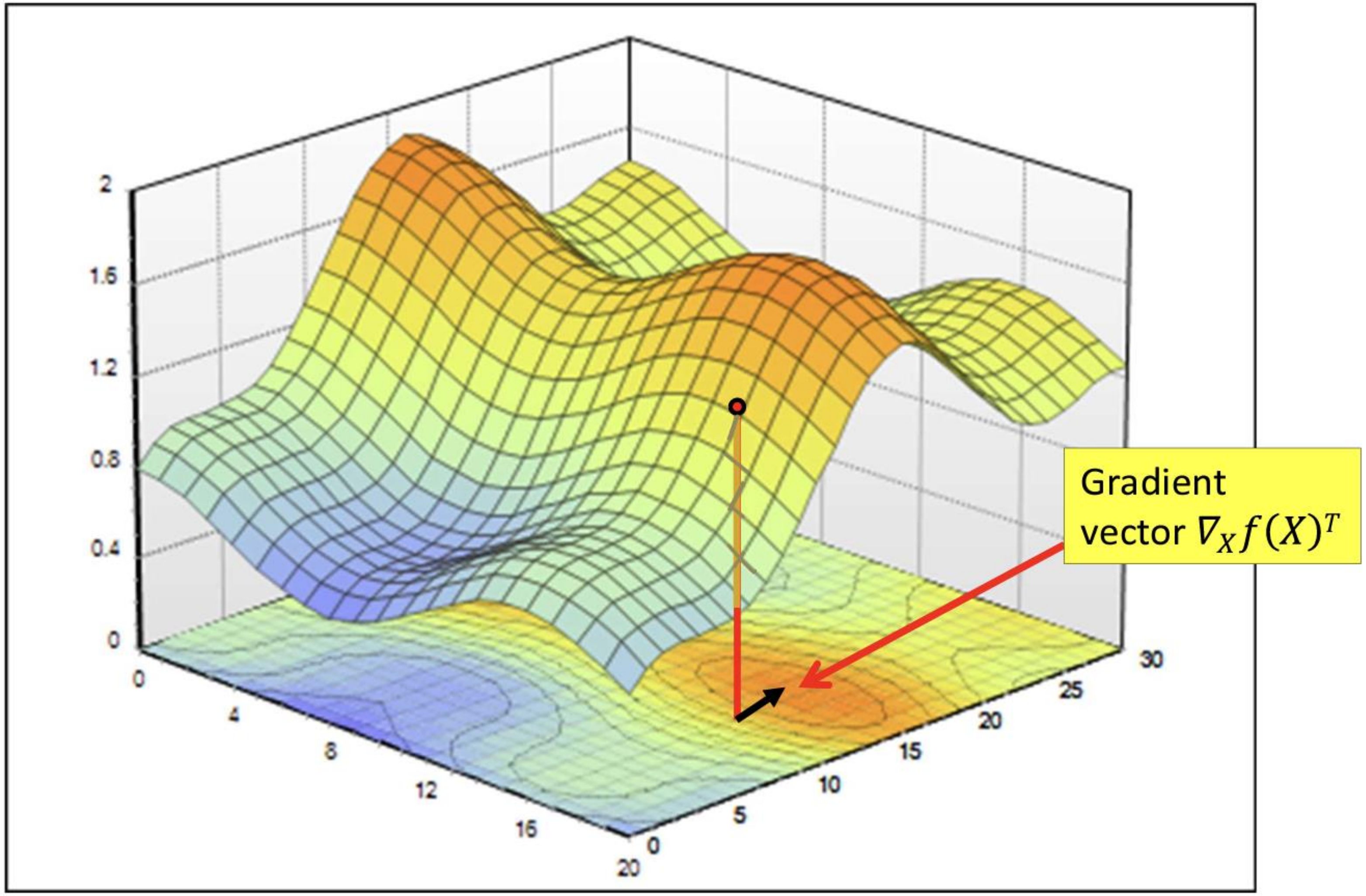
- $\nabla_X f(X) = \left[\frac{\partial f(X)}{\partial x_1} \quad \frac{\partial f(X)}{\partial x_2} \quad \cdots \quad \frac{\partial f(X)}{\partial x_n}\right]$

The ***gradient*** is the transpose of the derivative $\nabla_X f(X)^T$

- A column vector of the same dimensionality as X

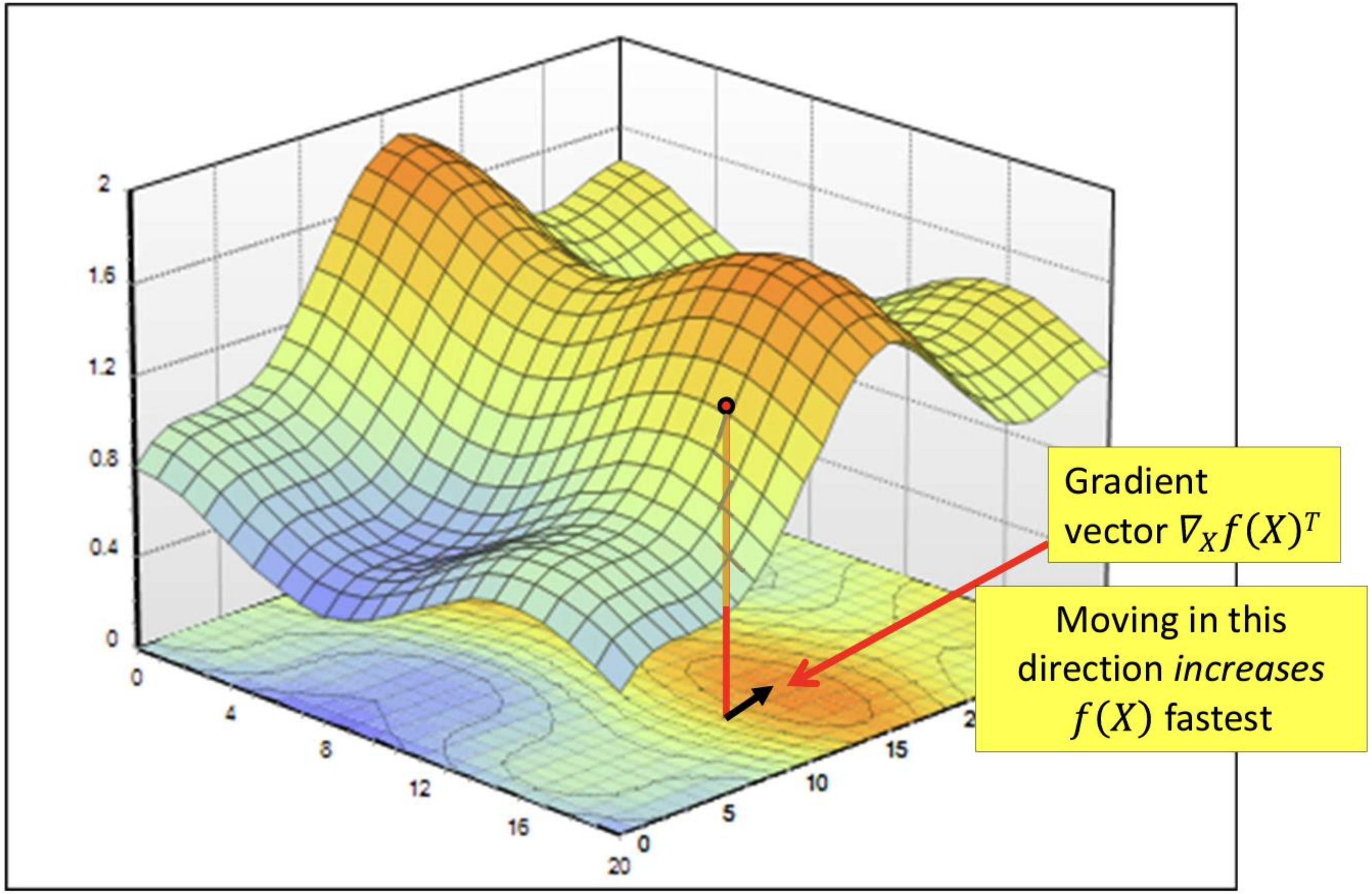
[Image from Bhiksha et al. CMU DL Course]

Optimization



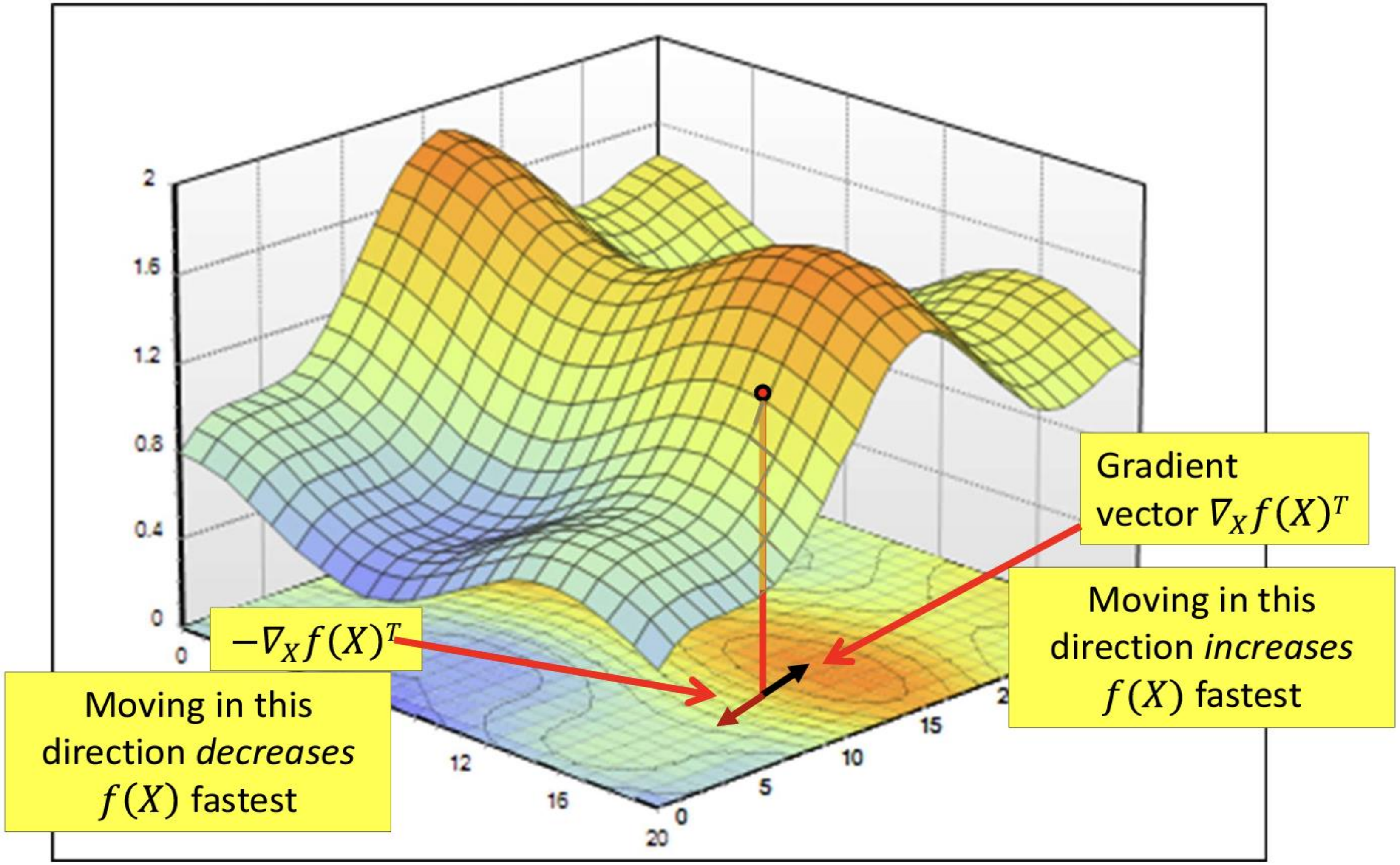
[Image from Bhiksha et al. CMU DL Course]

Optimization



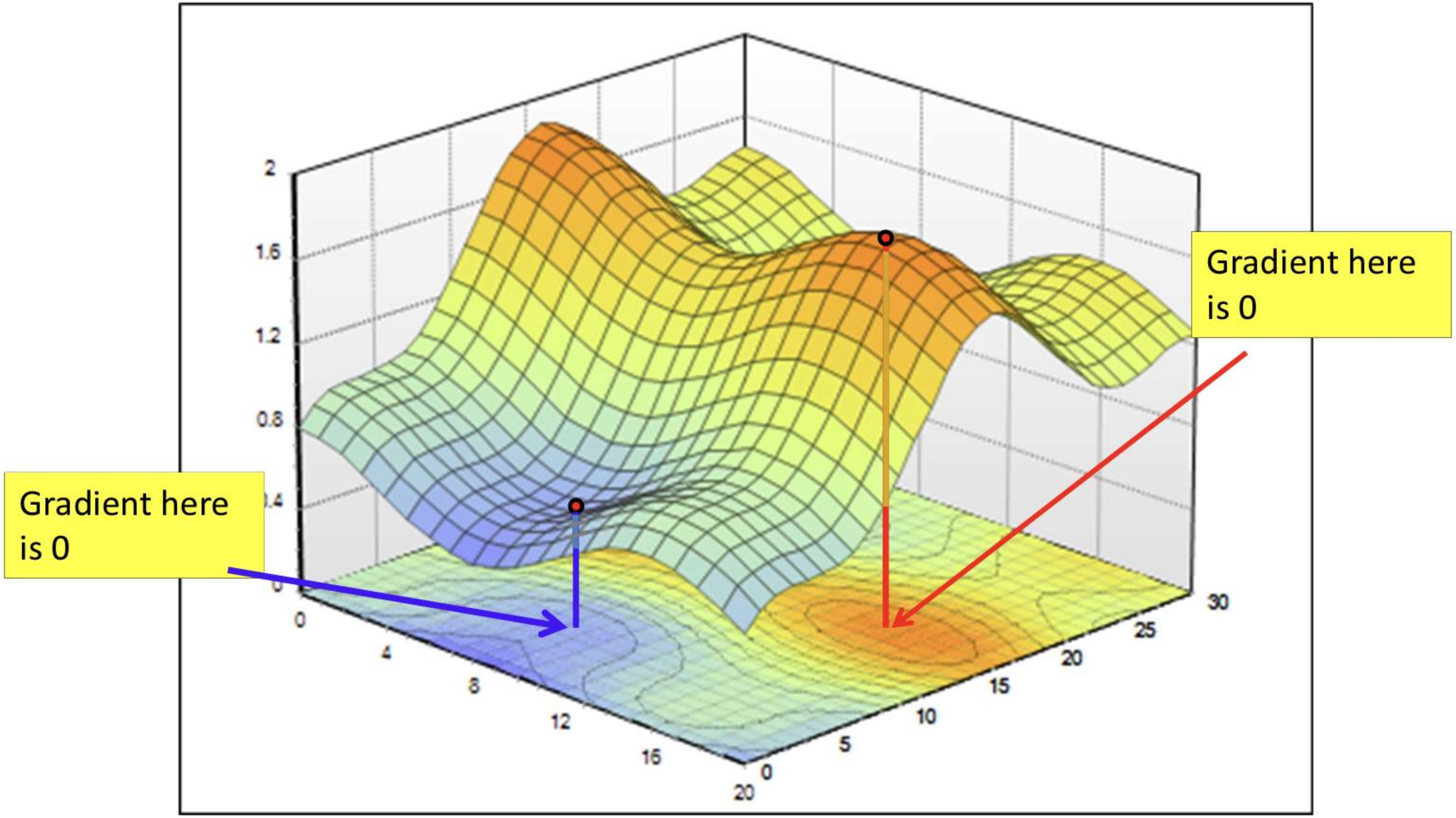
[Image from Bhiksha et al. CMU DL Course]

Optimization



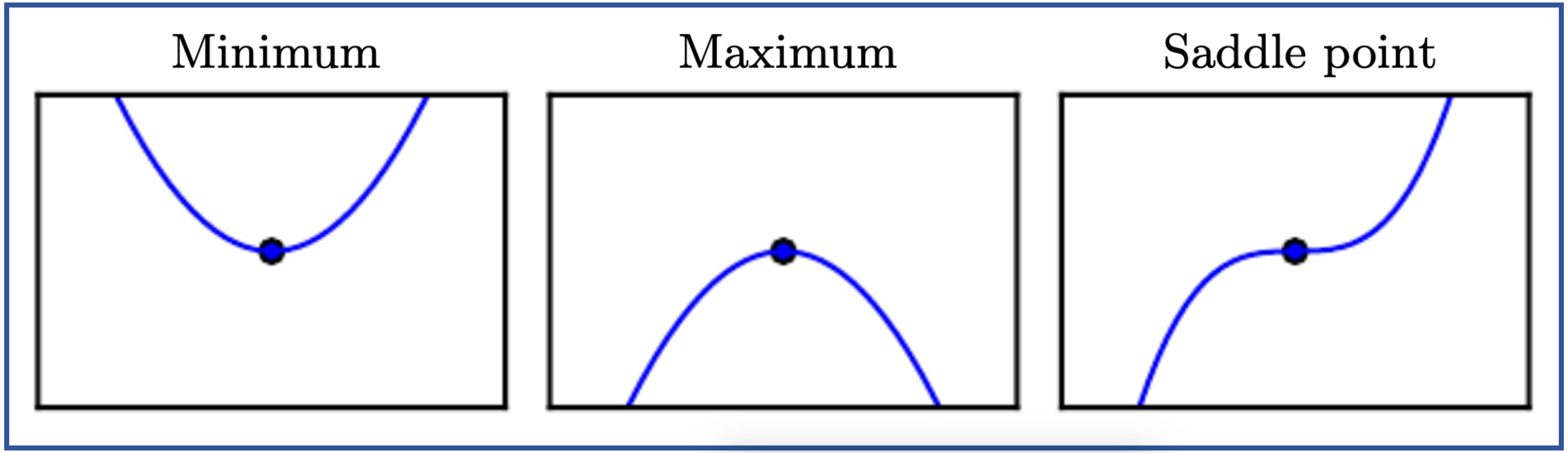
[Image from Bhiksha et al. CMU DL Course]

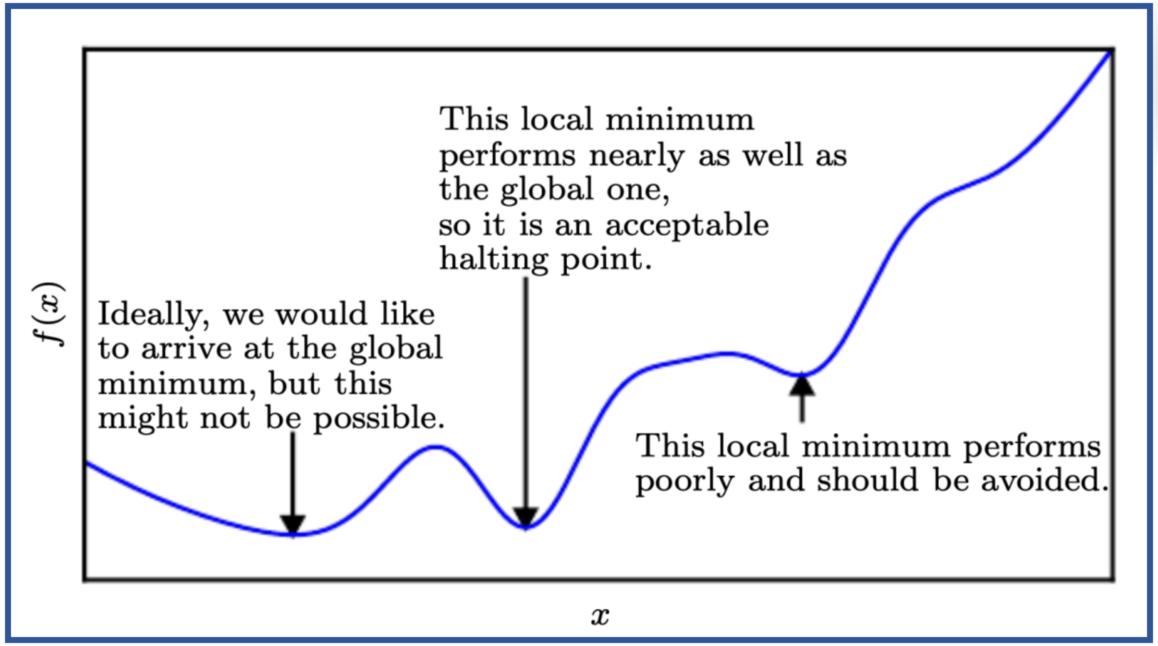
Optimization



[Image from Bhiksha et al. CMU DL Course]

Optimization



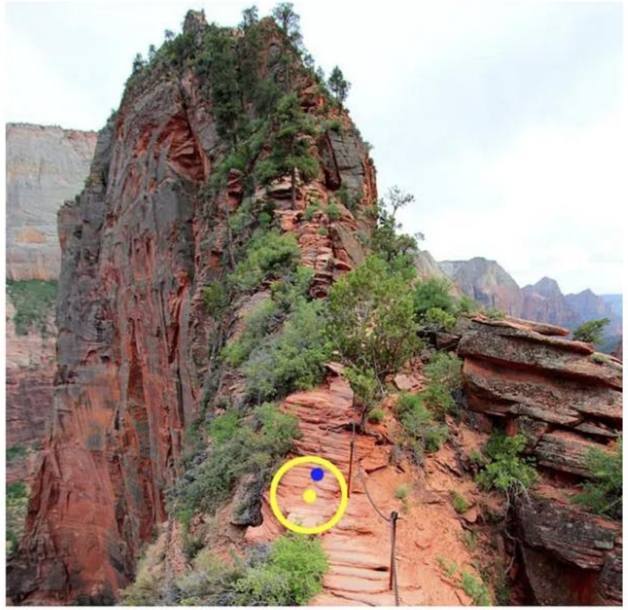


Gradient = 0

[Image from Goodfellow et al. DL Book]

Optimization

Line search
(like gradient ascent)



Trust region

$$\max_{s \in \mathbb{R}^n} \ m_k(s)$$
$$\text{s.t.} \quad \|s\| \leq \delta$$

[Image https://jonathan-hui.medium.com/]

Optimization

❑ Optimization

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \Big[L(\phi) \Big]$$

Step 1: Initialize the parameters $\boldsymbol{\phi_0}$

Step 2: Compute the derivatives of loss with respect to the parameters.

$$\frac{\partial L}{\partial \phi} = \begin{bmatrix} \frac{\partial L}{\partial \phi_0} \\ \frac{\partial L}{\partial \phi_1} \\ \vdots \\ \frac{\partial L}{\partial \phi_N} \end{bmatrix}$$

Gradient Descent

Step 3: Update the parameters.

$$\phi \longleftarrow \phi - \alpha \cdot \frac{\partial L}{\partial \phi}$$

Learning rate

Optimization

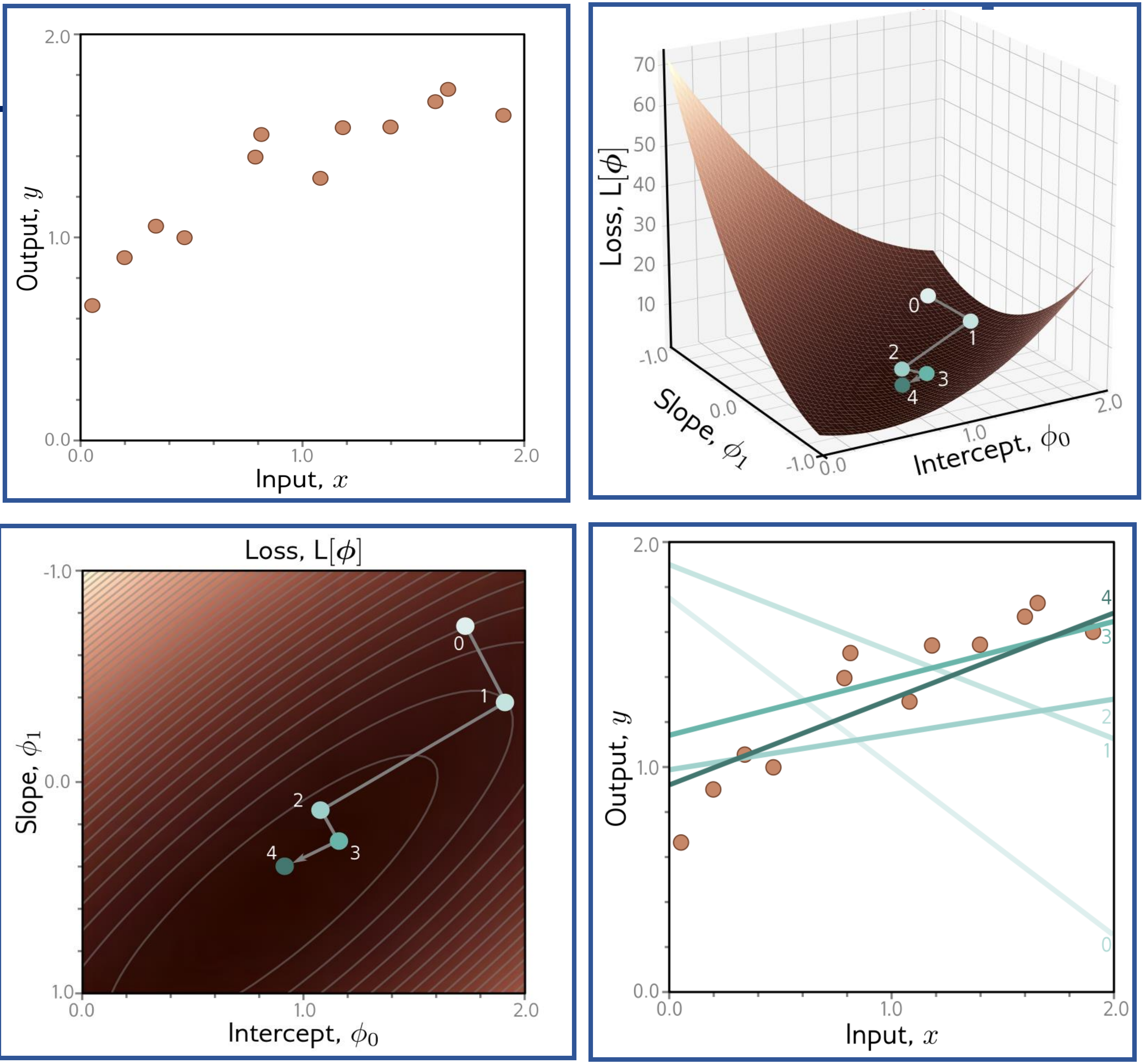
Training data:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_I, y_I)$$

$$\begin{aligned} L[\phi] &= \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2 \\ &= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2 \end{aligned}$$

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \Big[L(\phi) \Big]$$

Gradient Descent



[Image from Prince 2023 UDL Book]

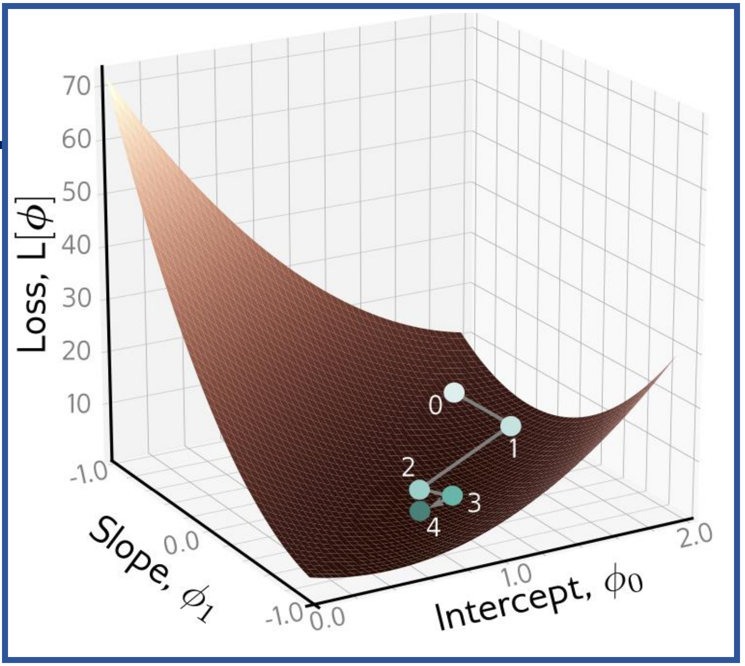
Optimization

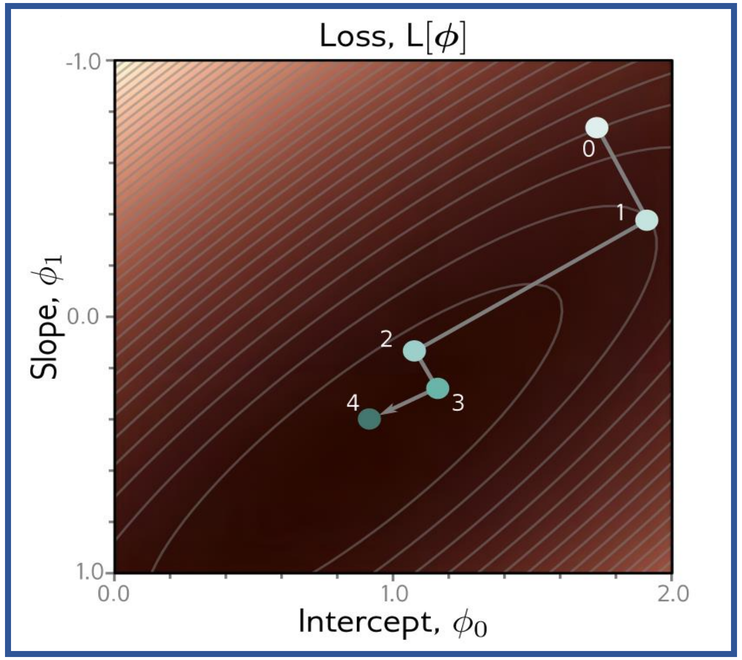
Training data:

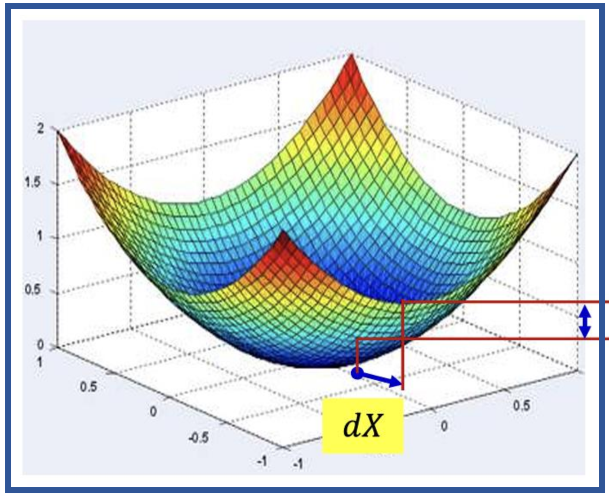
$$(x_1, y_1), (x_2, y_2), \ldots, (x_I, y_I)$$

$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$
$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \Big[L(\phi) \Big]$$



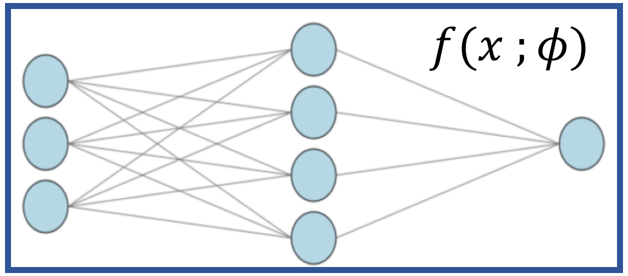




Do we reach a global minimum?

[Image from Prince 2023 UDL Book]

Optimization

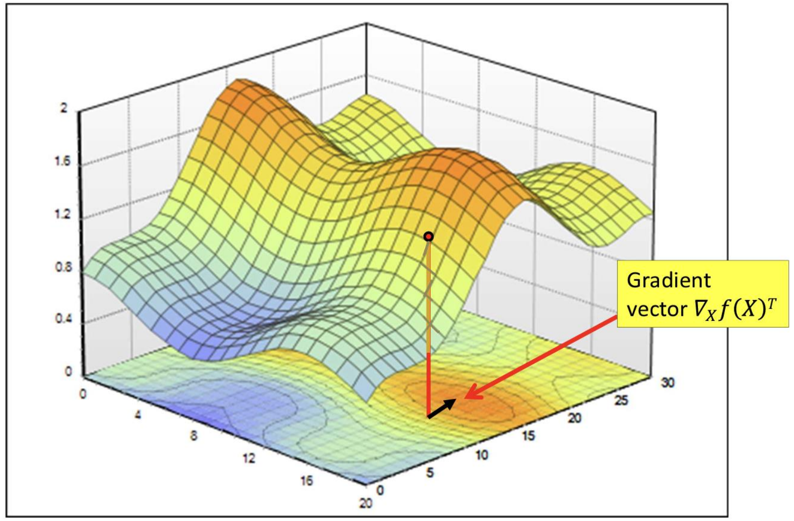


$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \Big[L(\phi) \Big]$$

Non-linear

May become a **non-convex optimization**

Do we still reach a global minimum?



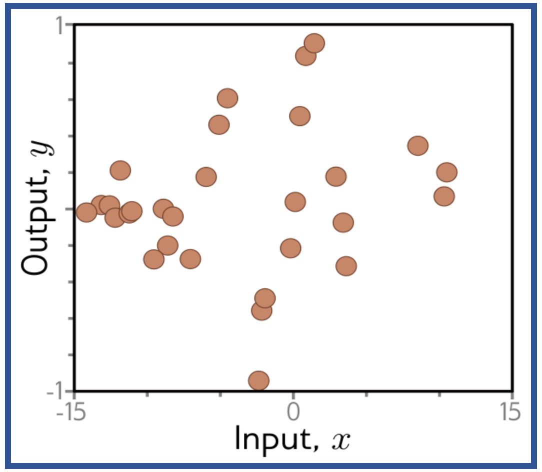
Note:

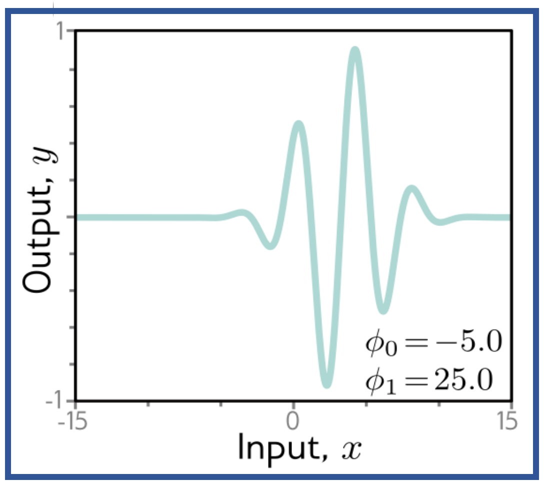
Not all non-linear functions are non-convex.

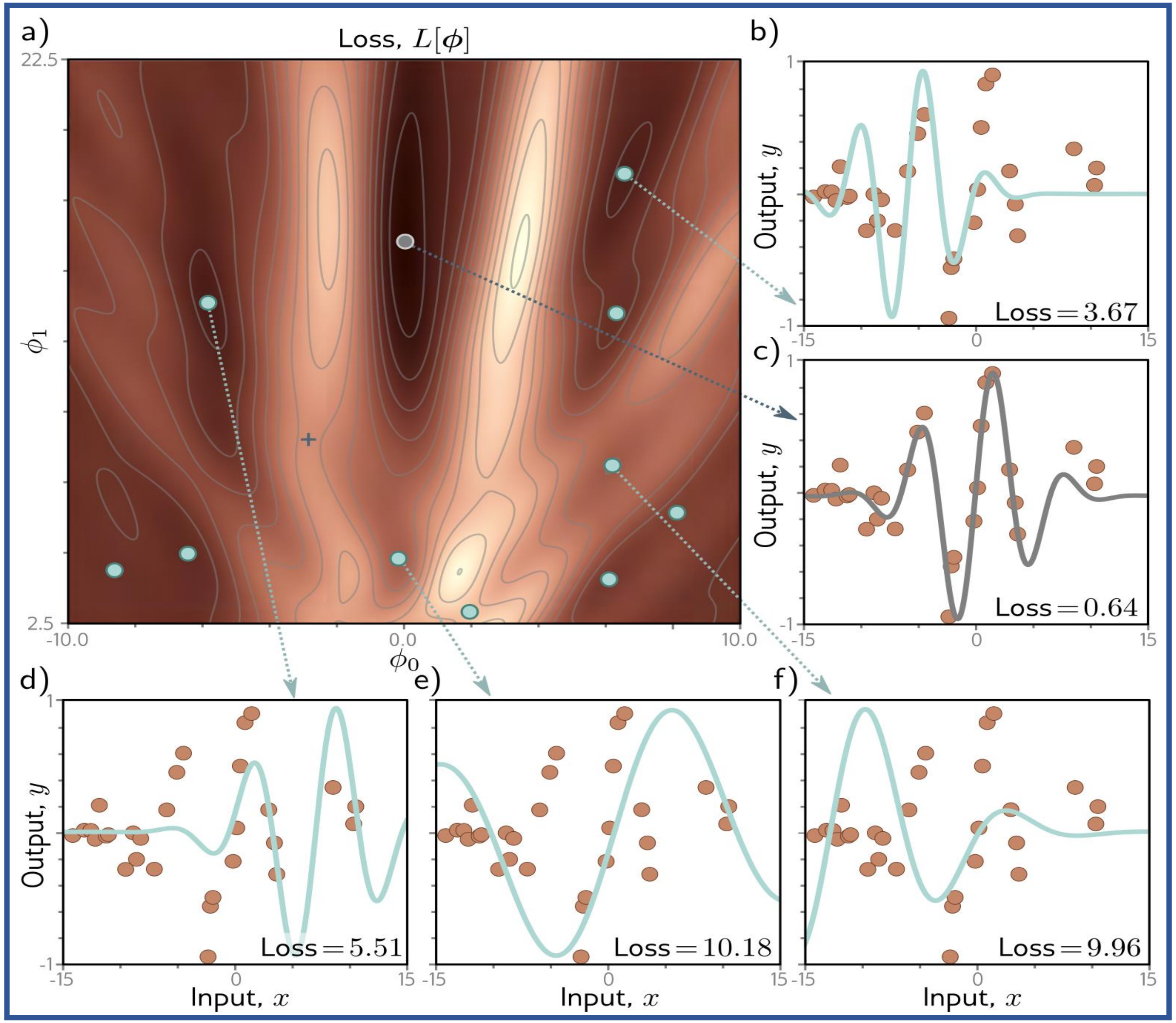
[Image from Prince 2023 UDL Book]

Optimization

❑ Non-linear function





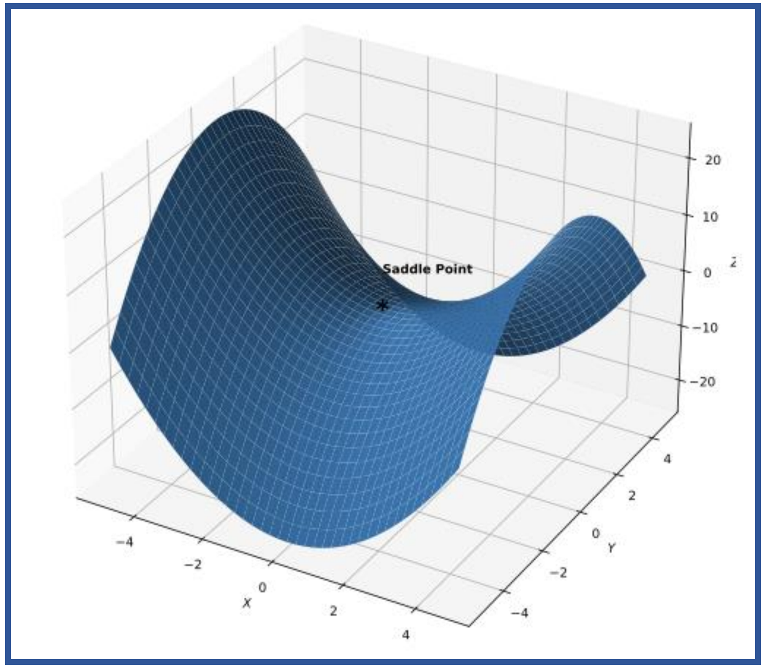


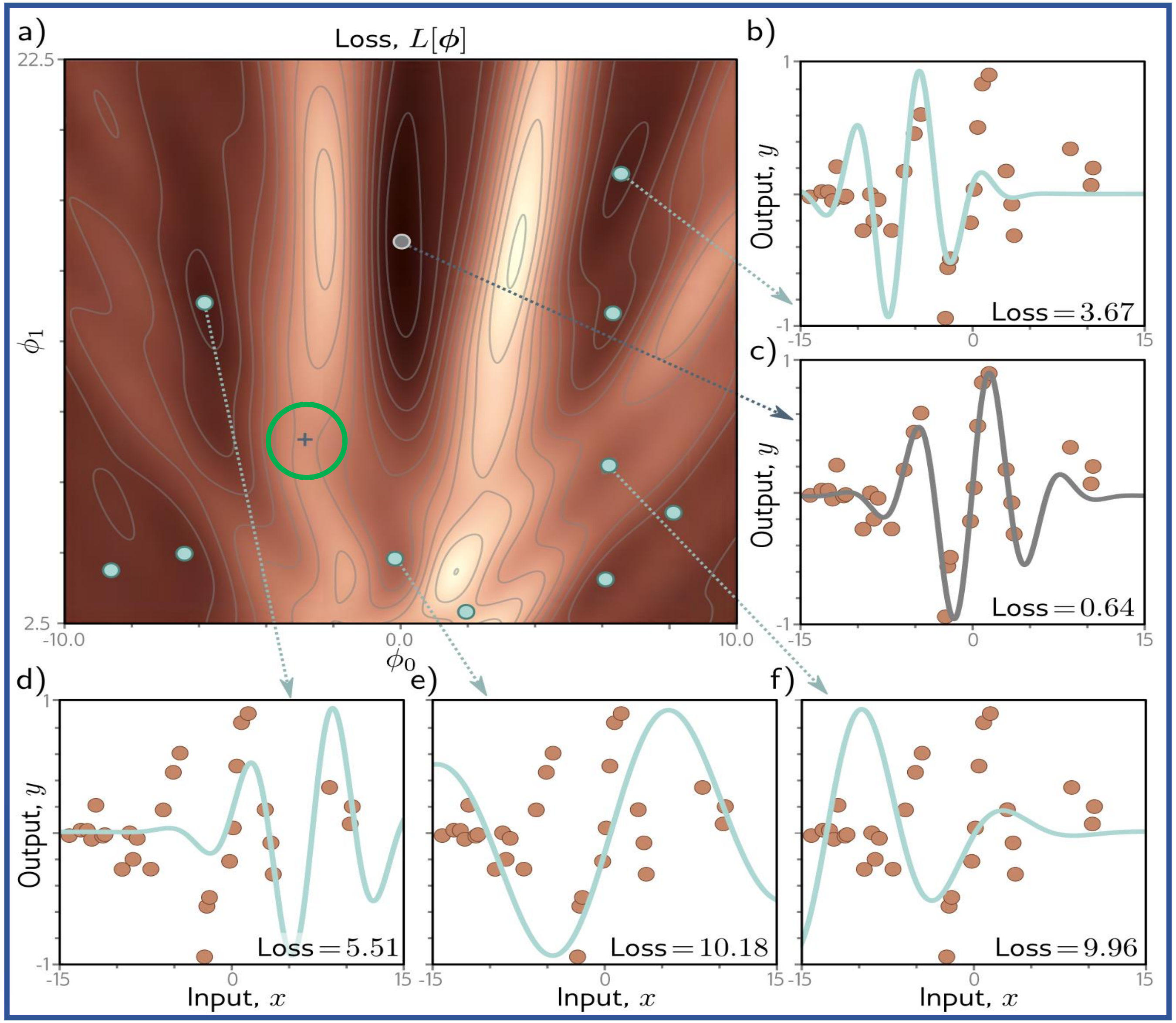
$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \Big[L(\phi) \Big]$$

[Image from Prince 2023 UDL Book]

Optimization

❑ Saddle Points





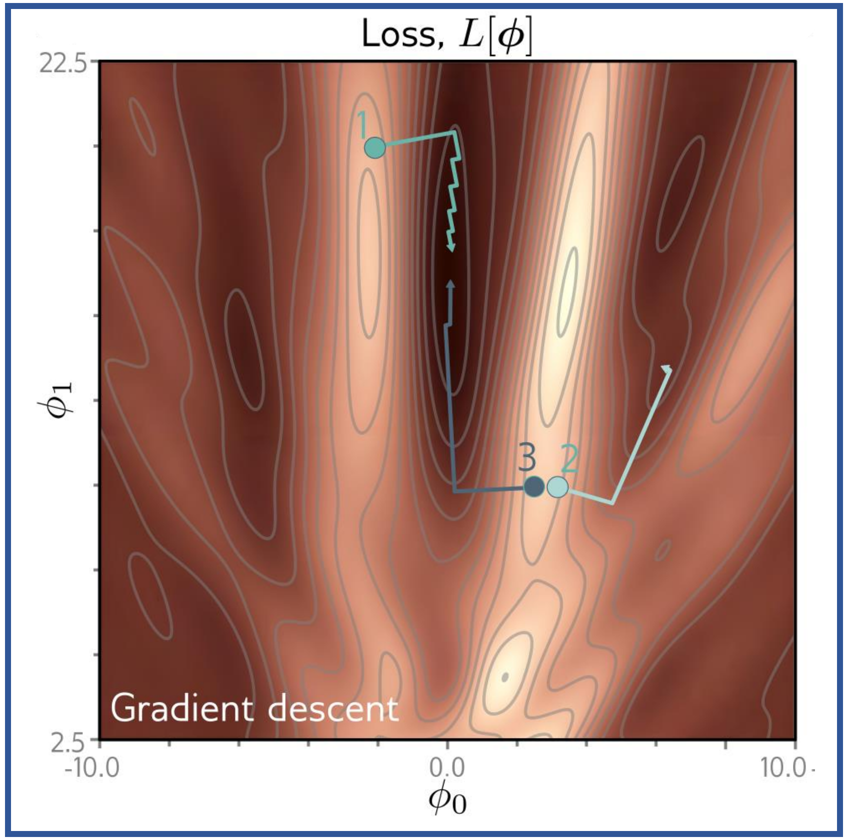
How to escape from saddle points and bad local minima?

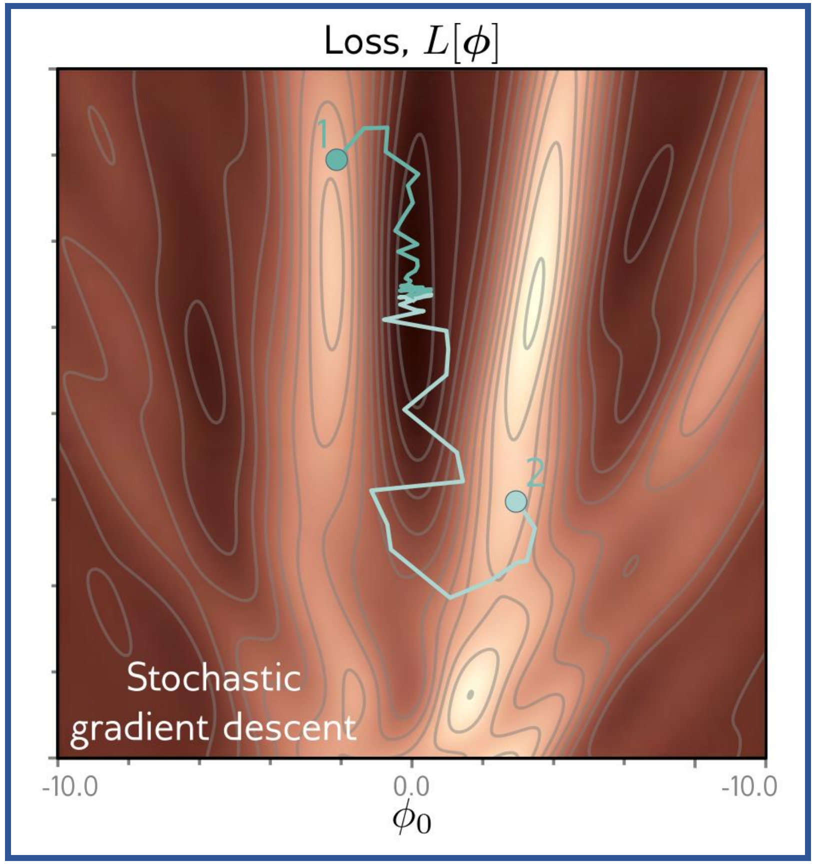
[Image from Prince 2023 UDL Book]

Optimization

❑ Stochastic Gradient Descent (SGD)

- Introduces some noise in the optimization process.





[Image from Prince 2023 UDL Book]

Optimization

❑ Stochastic Gradient Descent (SGD)

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \Big[L(\phi) \Big]$$

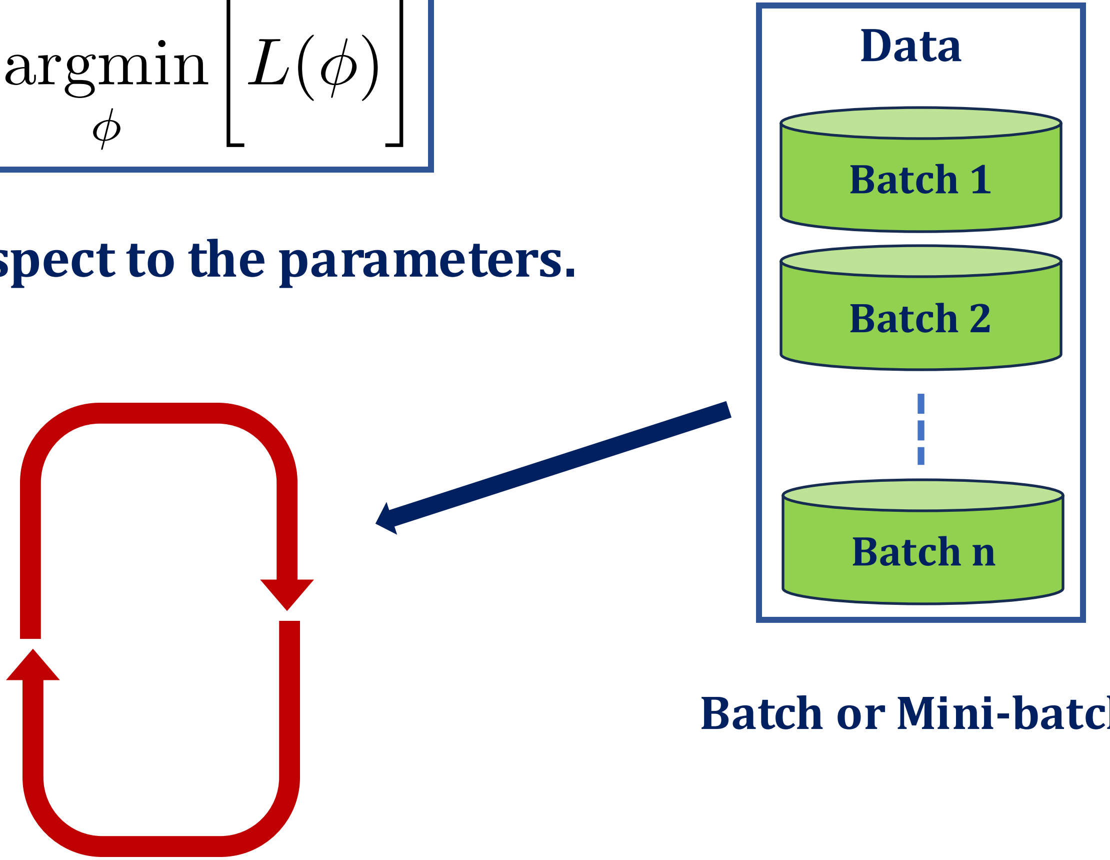
Step 1: Initialize the parameters $\boldsymbol{\phi}_0$

Step 2: Compute the derivatives of loss with respect to the parameters.

$$\sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

Step 3: Update the parameters.

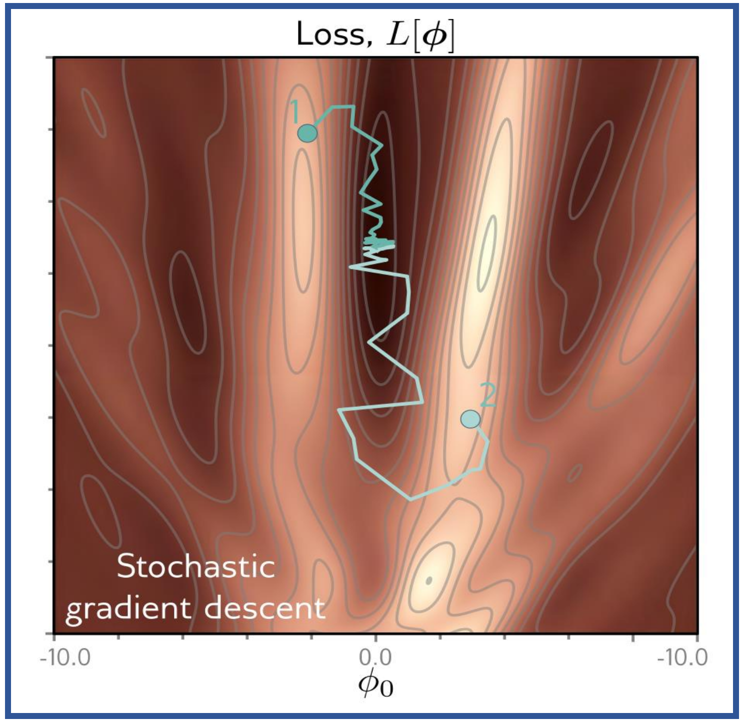
$$\boldsymbol{\phi}_{t+1} \longleftarrow \boldsymbol{\phi}_t - \alpha \cdot \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$



Batch or Mini-batch

Stochastic Gradient Descent (SGD)

- Computationally **less expensive** to compute the batch gradient.
- In principle, it can escape **local minima and saddle points**.



[Image from Prince 2023 UDL Book]

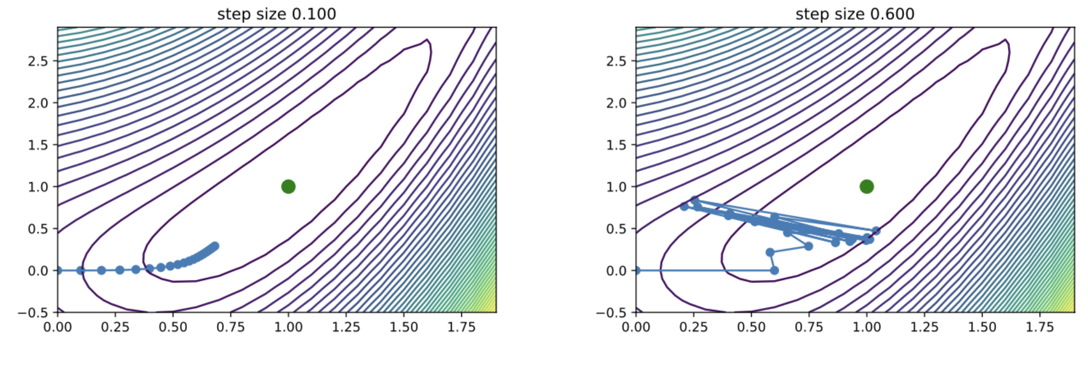
ScAI

Any Major Drawback in Gradient Descent?

Optimization

❑ Momentum

- Gradient descent move very slowly.
- Simple heuristic: **Momentum or heavy ball**
  - Move fast in the previously good direction.
  - Slow down if the gradient suddenly changes.



(a) (b)

Like a rolling ball!

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_{t-1}$$
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \boldsymbol{m}_t$$

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_{t-1} = \beta^2 \boldsymbol{m}_{t-2} + \beta \boldsymbol{g}_{t-2} + \boldsymbol{g}_{t-1} = \cdots = \sum_{\tau=0}^{t-1} \beta^\tau \boldsymbol{g}_{t-\tau-1}$$

- If gradients are same:

$$\boldsymbol{m}_t = \boldsymbol{g} \sum_{\tau=0}^{t-1} \beta^\tau$$

$$1 + \beta + \beta^2 + \cdots = \sum_{i=0}^{\infty} \beta^i = \frac{1}{1-\beta}$$

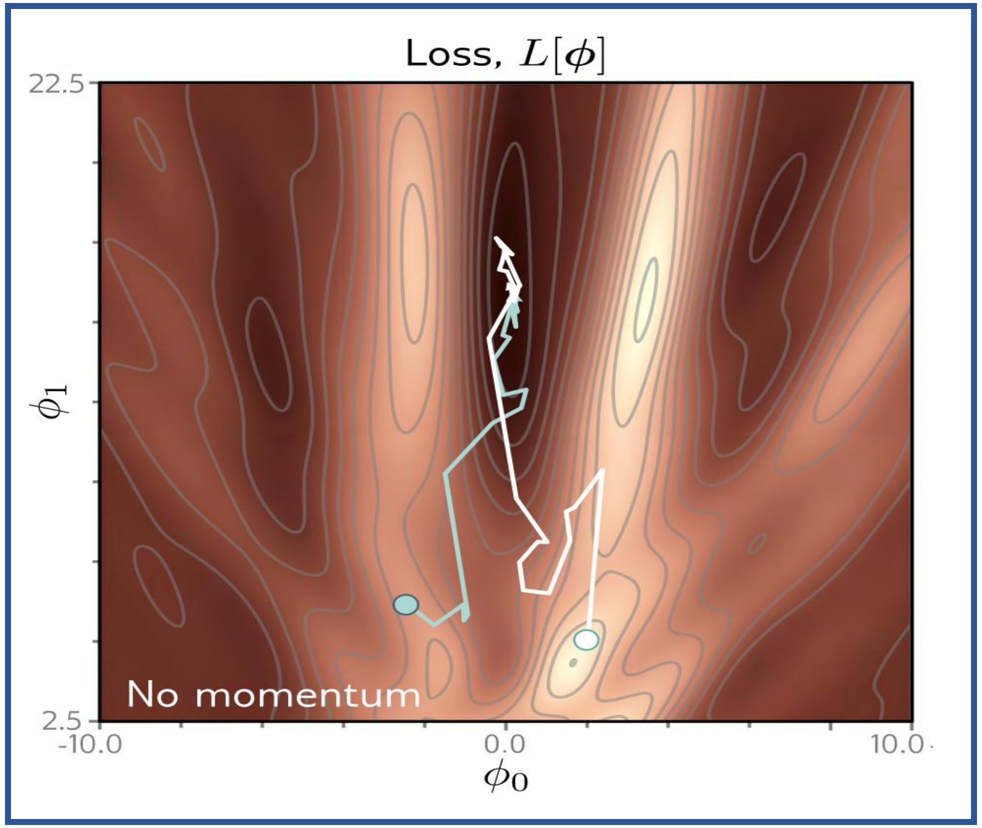
If, $\beta = 0.9$ We are **scaling** the gradient by a factor of **10**. **Move Fast!**

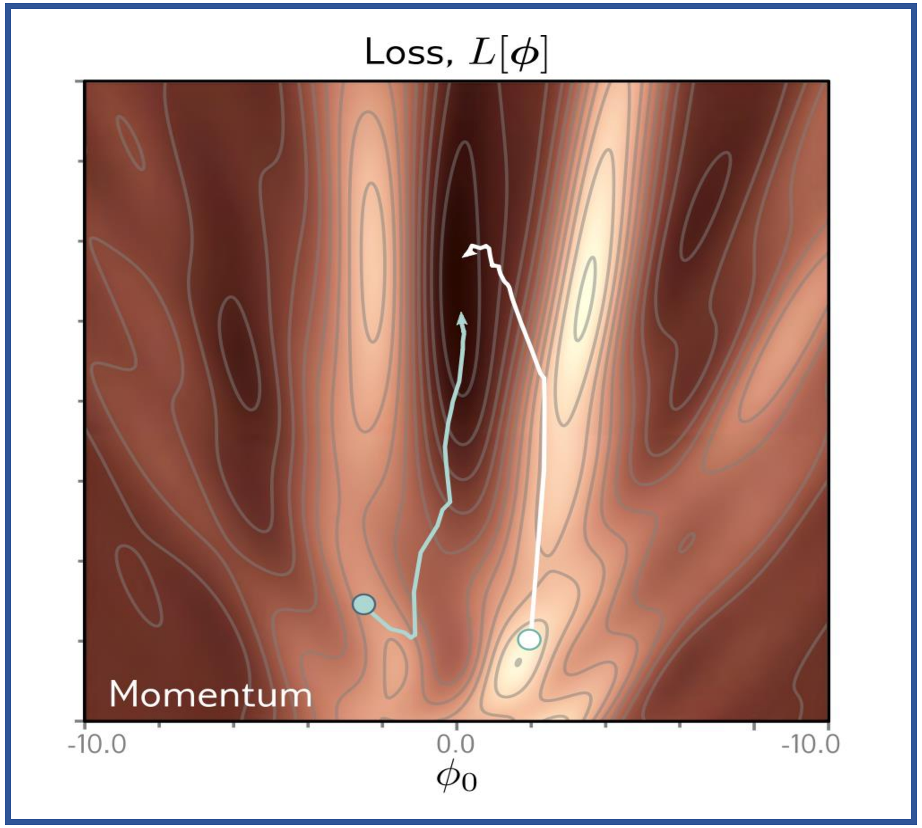
- If gradients are different:

[Image from Kevin Murphy's ML Book]

Optimization

❑ Momentum





[Image from Prince 2023 UDL Book]

Optimization

❑ Learning Rate

- **Parameter Update:**

$$\phi \longleftarrow \phi - \alpha \cdot \frac{\partial L}{\partial \phi}$$

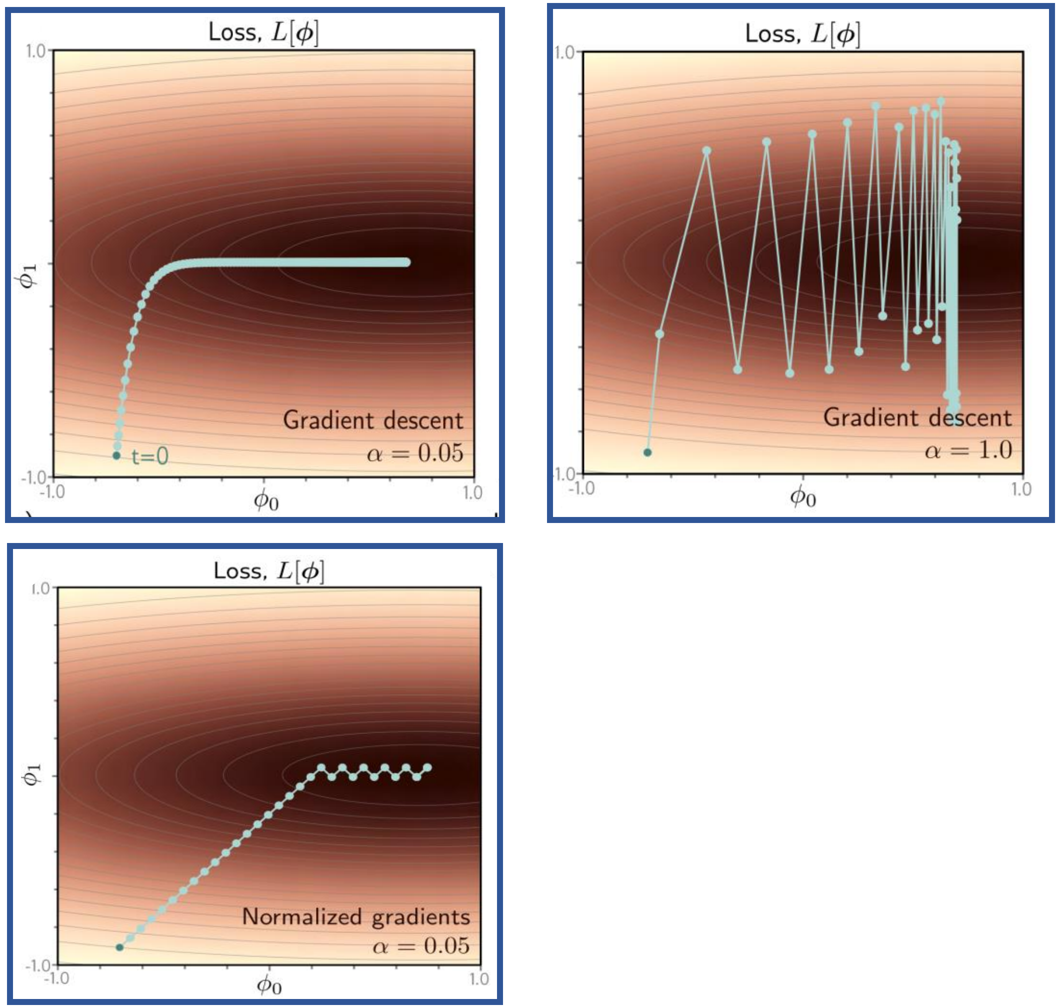
Learning rate (α)

- **Normalize the gradient:**

$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\phi_t]}{\partial \phi}$$

$$\mathbf{v}_{t+1} \leftarrow \left(\frac{\partial L[\phi_t]}{\partial \phi}\right)^2$$



[Image from Prince 2023 UDL Book]

Optimization

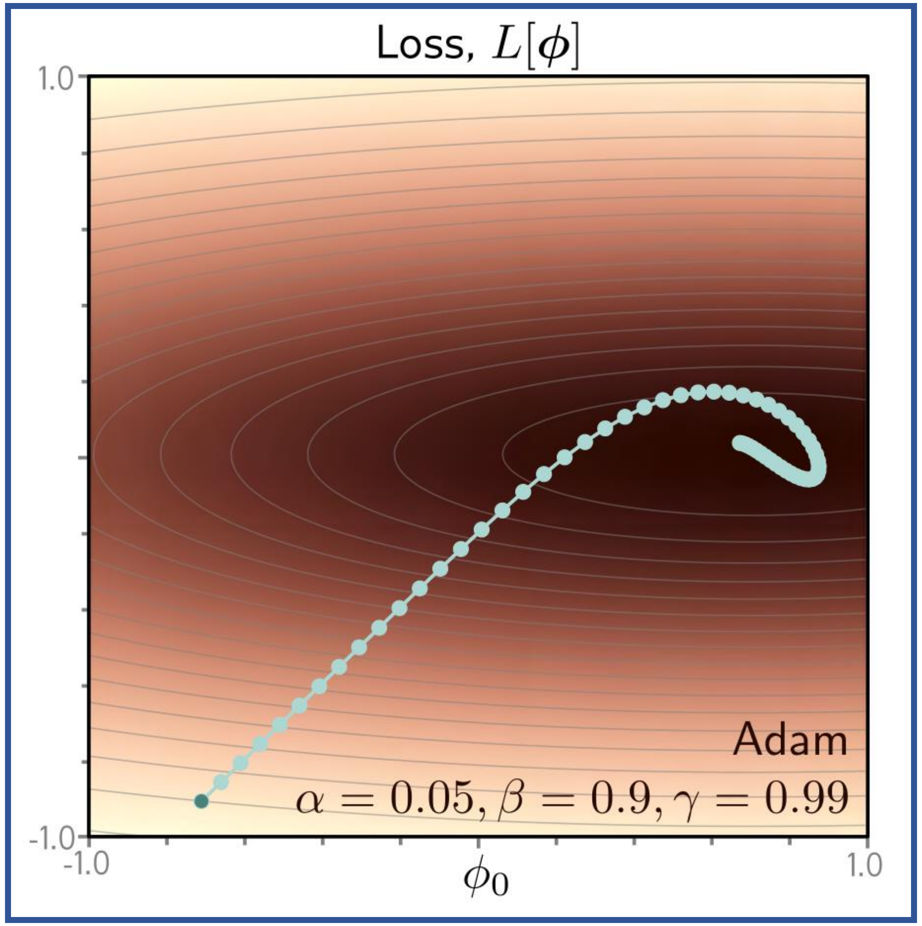
❑ Adaptive Momentum Estimation (Adam) Optimizer

- **Include Momentum:**

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1-\beta)\frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \gamma \cdot \mathbf{v}_t + (1-\gamma)\left(\frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}\right)^2$$

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}}+\epsilon}$$



[Image from Prince 2023 UDL Book]

Optimization

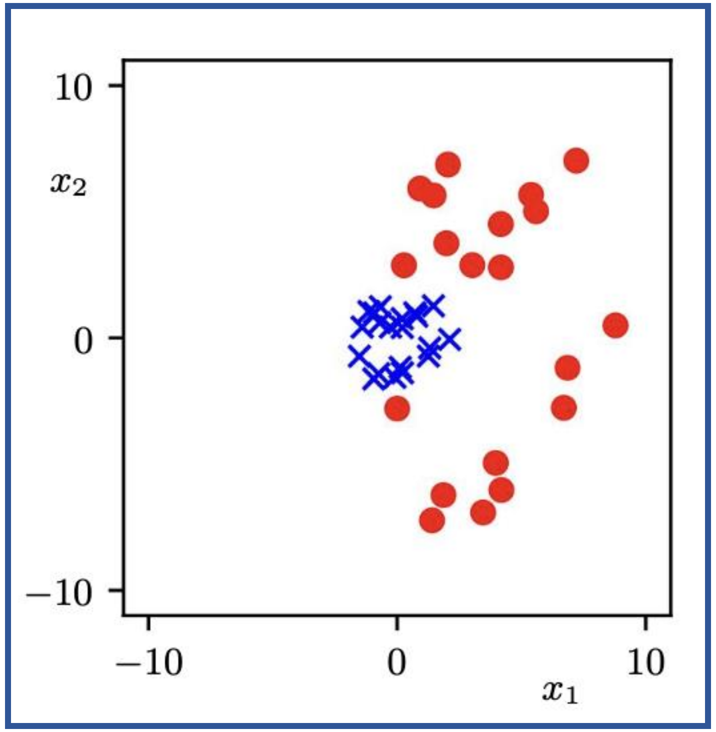
Input Data Normalization

- Input features may have a wide range of values.
- Done prior to training.

$$\mu_i = \frac{1}{N}\sum_{n=1}^{N} x_{ni}$$

$$\sigma_i^2 = \frac{1}{N}\sum_{n=1}^{N}(x_{ni}-\mu_i)^2,$$

$$\widetilde{x}_{ni} = \frac{x_{ni}-\mu_i}{\sigma_i}$$



- Zero mean and unit variance.

[Image from Bishop DL Book]

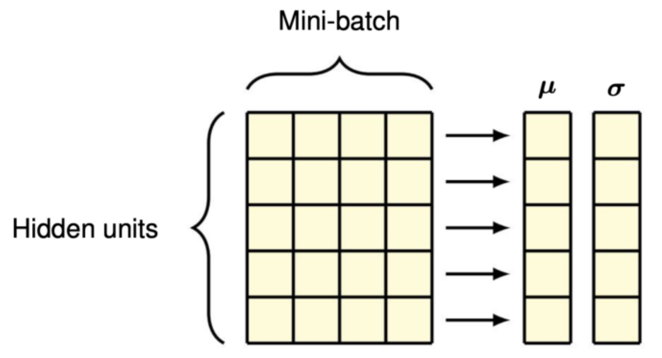
Optimization

Batch Normalization

- Normalizing the hidden layers.
- Vanishing gradients and exploding gradients.

$$\frac{\partial E}{\partial w_i} = \sum_m \cdots \sum_l \sum_j \frac{\partial z_m^{(1)}}{\partial w_i} \cdots \frac{\partial z_j^{(K)}}{\partial z_l^{(K-1)}} \frac{\partial E}{\partial z_j^{(K)}}$$

$$z_i = h(a_i)$$



$$\mu_i = \frac{1}{K}\sum_{n=1}^{K} a_{ni}$$

$$\sigma_i^2 = \frac{1}{K}\sum_{n=1}^{K} (a_{ni} - \mu_i)^2$$

$$\widehat{a}_{ni} = \frac{a_{ni} - \mu_i}{\sqrt{\sigma_i^2 + \delta}}$$

- Any problem with the normalization?
  - We reduce the degree of freedom/ representation capacity of that hidden layer.

Batch Normalization Layer

$$\widetilde{a}_{ni} = \gamma_i \widehat{a}_{ni} + \beta_i$$

Stdv Mean

- β_i and γ_i are adaptive learnable parameters.
- Trained using gradient descent, same as other parameters.

[Image from Bishop DL Book]

ScAI

END