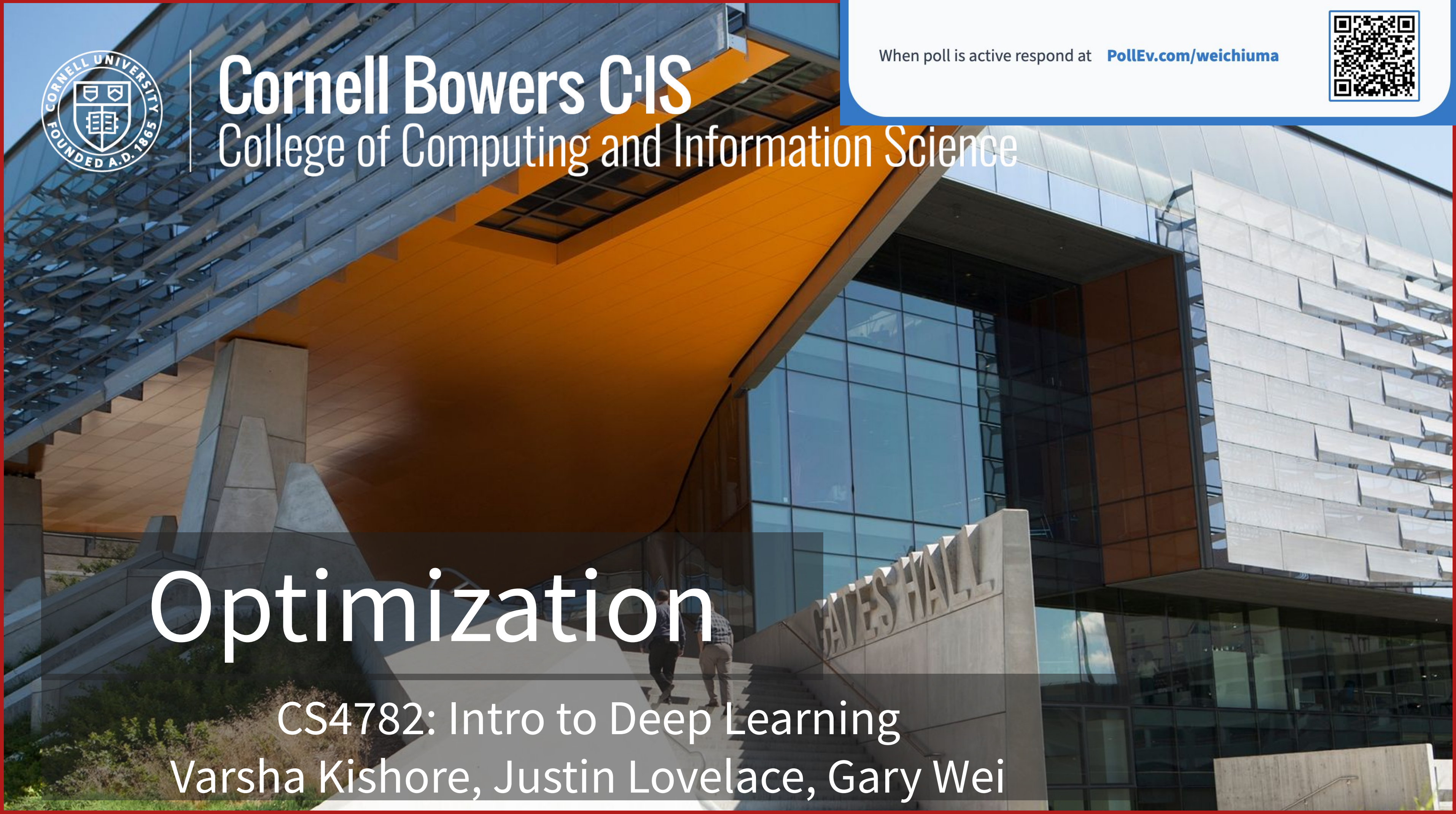
When poll is active respond at **PollEv.com/weichiuma**

Cornell Bowers C·IS
College of Computing and Information Science



Optimization

CS4782: Intro to Deep Learning

Varsha Kishore, Justin Lovelace, Gary Wei

Before we start:

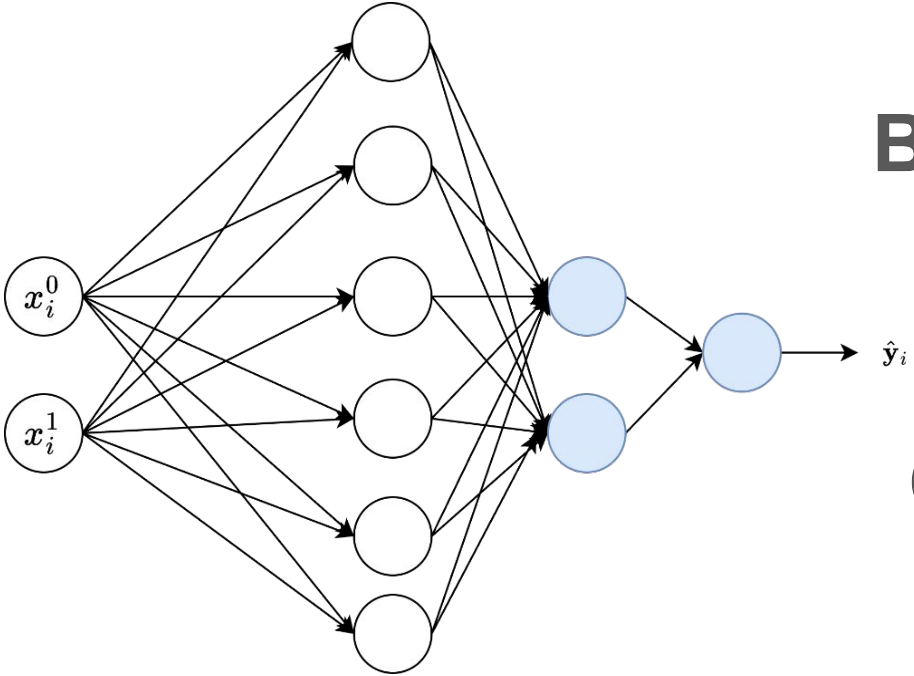
Look at recap quizzes 1,2

Discuss with your neighbors

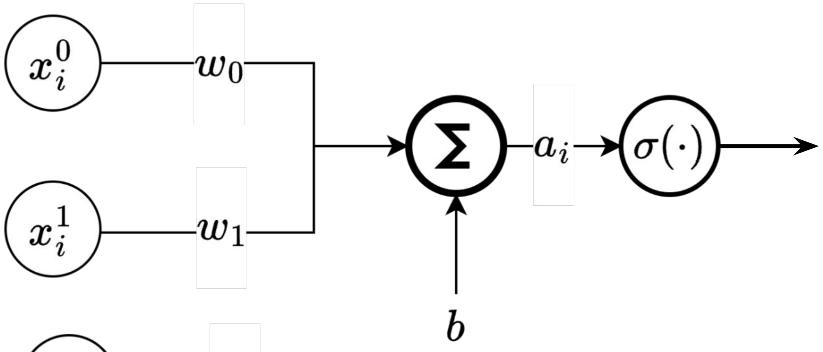
Choose option A, B, C

When poll is active respond at **PollEv.com/weichiuma**

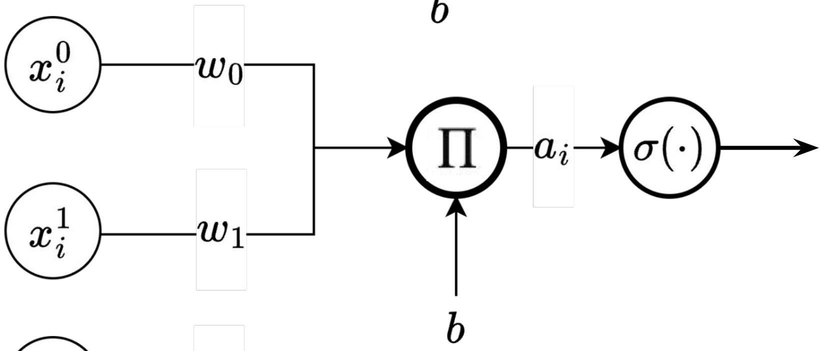
What's inside one of the circles?



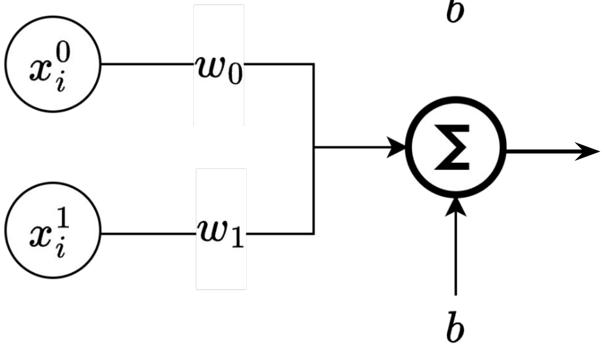
A.



B.

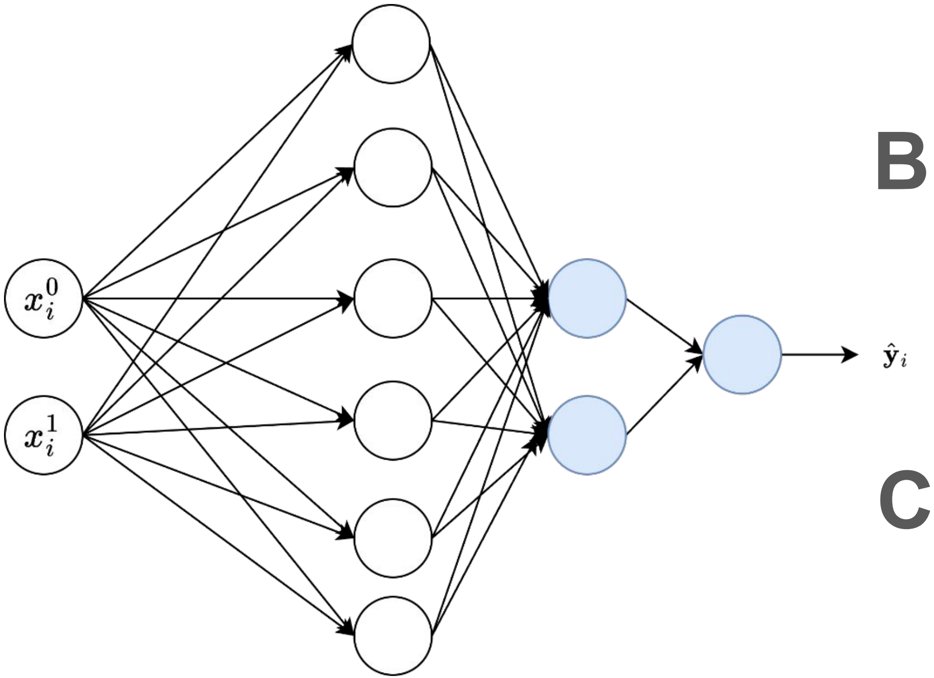


C.

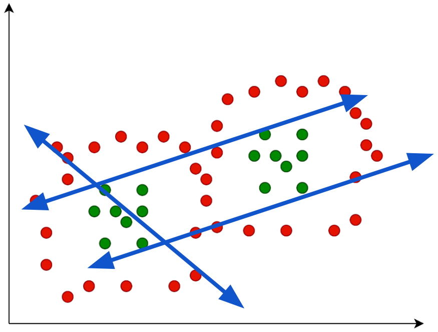


When poll is active respond at PollEv.com/weichiuma

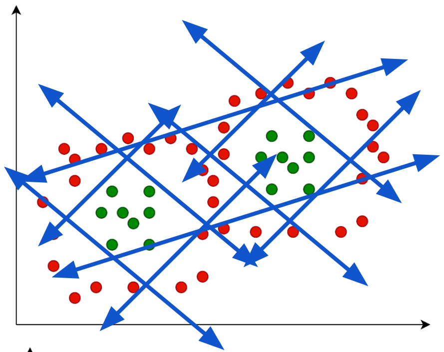
Which decision boundary CANNOT be learned by this network?



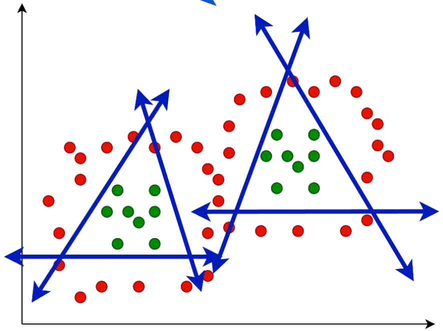
A.



B.



C.



Cornell Bowers C·IS

Course Announcement

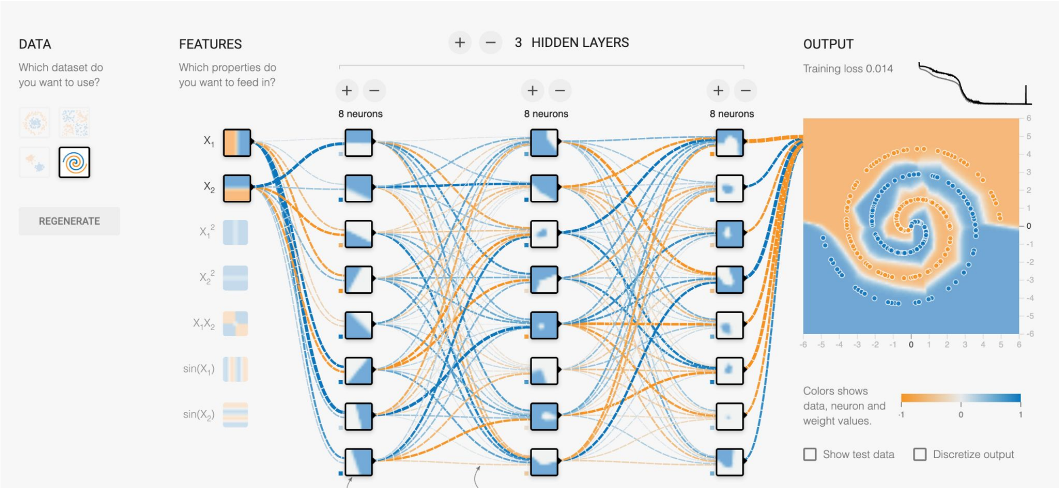
- If you are in **5782**
  - Paper quizzes are mandatory (10%)

- If you are in **4782**
  - Paper quizzes are optional
  - If you do them, we will use the better grade with or without quizzes

Cornell Bowers CIS

Agenda

- Backpropagation
- Optimizers
    - Gradient Descent
    - Stochastic Gradient Descent
    - SGD w. Momentum
    - AdaGrad
    - RMSProp
    - Adam
- Learning rate scheduling



Cornell Bowers C·IS

Calculus Review: The Chain Rule

Lagrange's Notation: If $h(x) = f(g(x))$, then $h' = f'(g(x))g'(x)$

Leibniz's Notation: If $z = h(y), y = g(x)$, then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

Calculus Review: The Chain Rule

Lagrange's Notation: If $h(x) = f(g(x))$, then $h' = f'(g(x))g'(x)$

Leibniz's Notation: If $z = h(y), y = g(x)$, then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

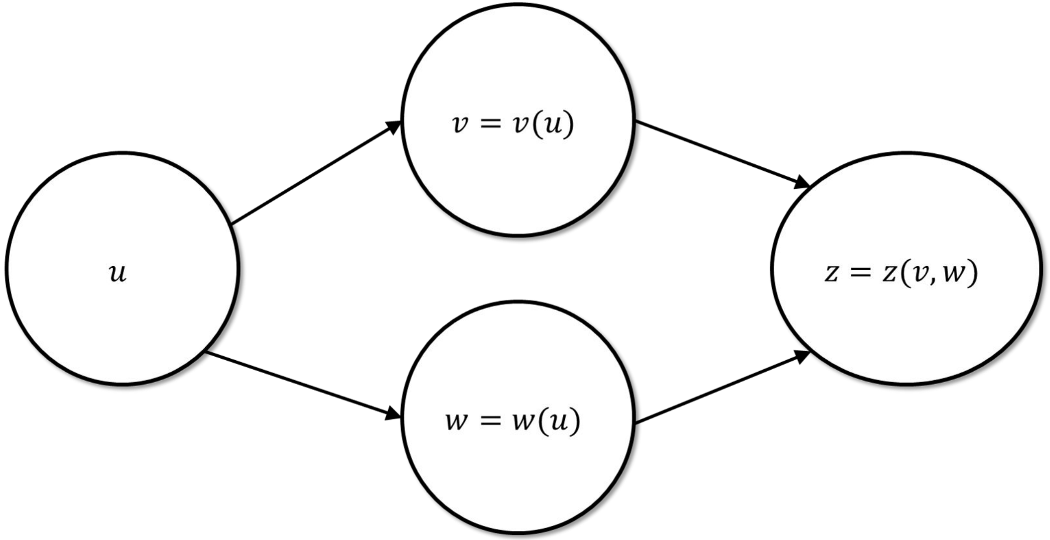
Example: If $z = \ln(y), y = x^2$, then

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

$$= (\frac{1}{y})(2x) = (\frac{1}{x^2})(2x) = \frac{2}{x}$$

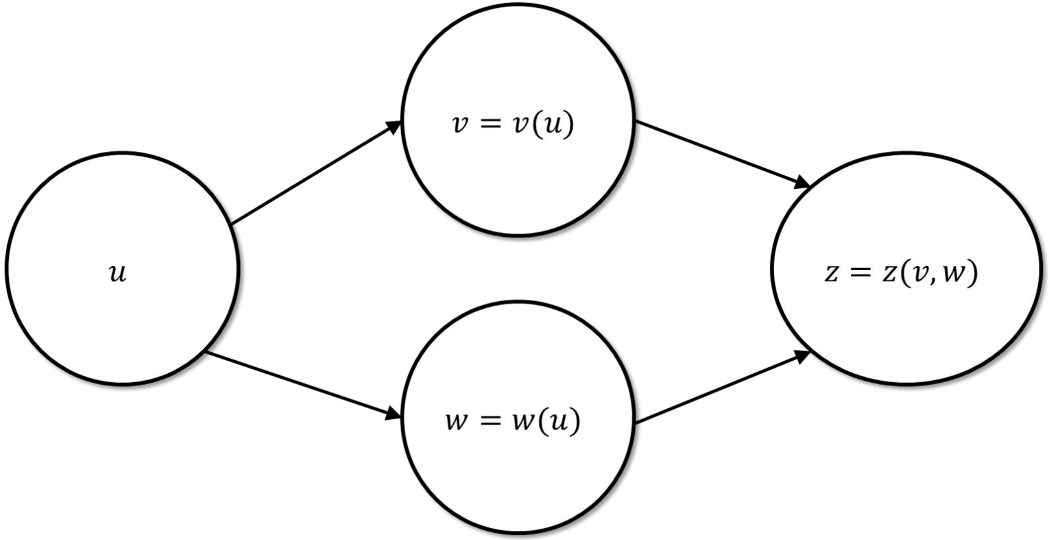
Cornell Bowers CIS

Multivariate Chain Rule



https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

Multivariate Chain Rule

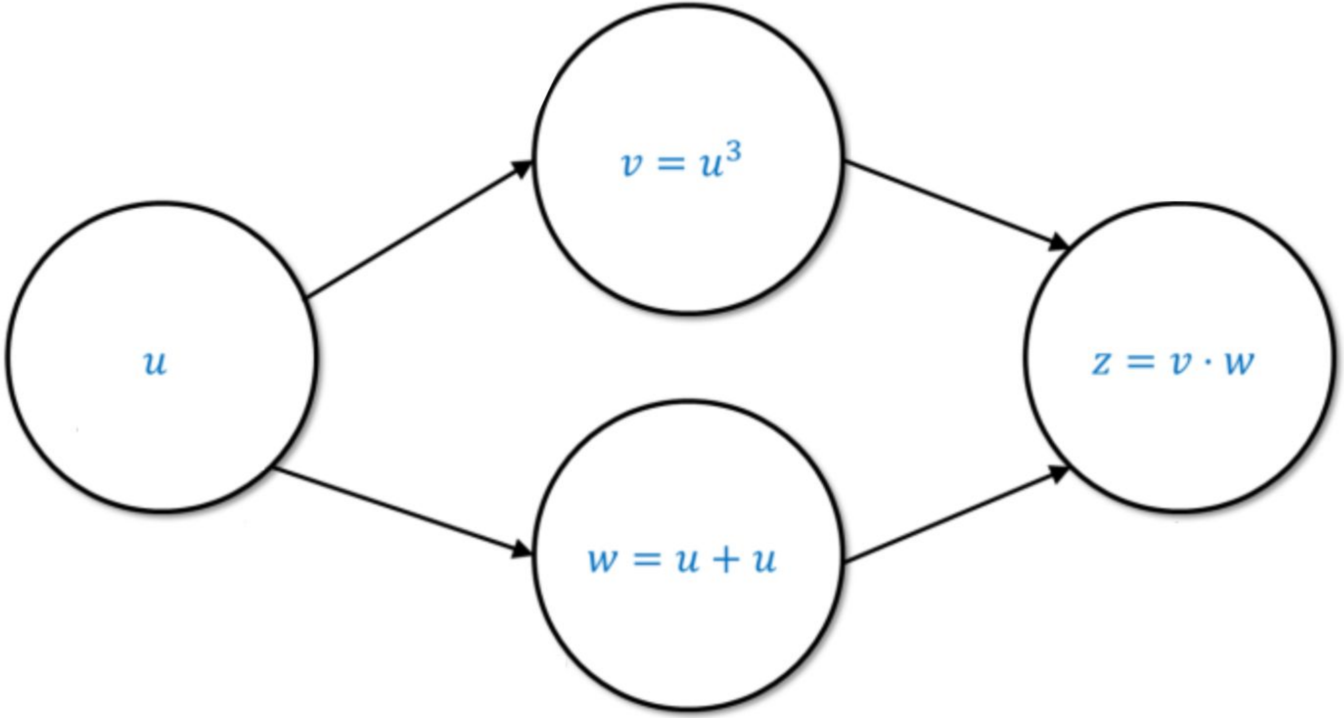


If $f(u)$ is $z = f(v(u), w(u))$, then

$$\frac{\partial f}{\partial u} = (\frac{\partial v}{\partial u}\frac{\partial z}{\partial v} + \frac{\partial w}{\partial u}\frac{\partial z}{\partial w})$$

https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

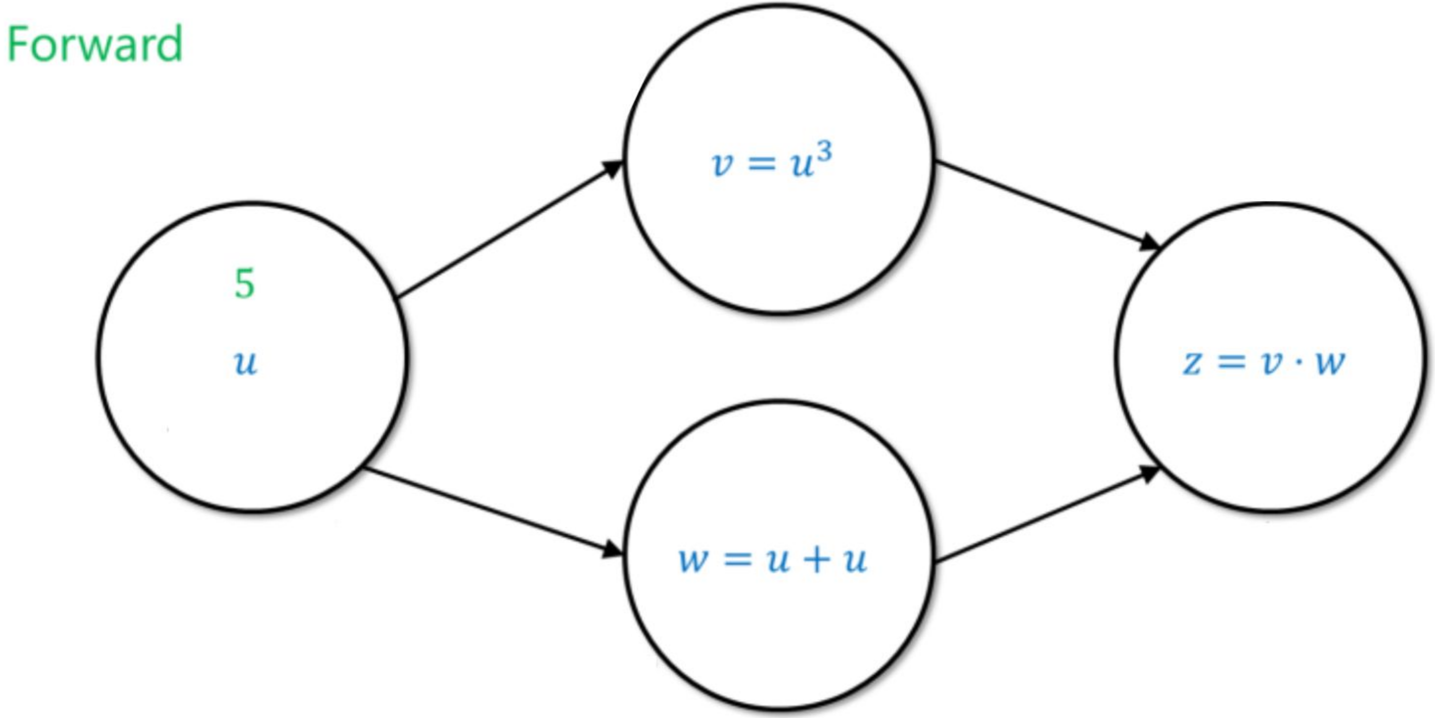
Backpropagation- An Example



https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

Cornell Bowers C·IS

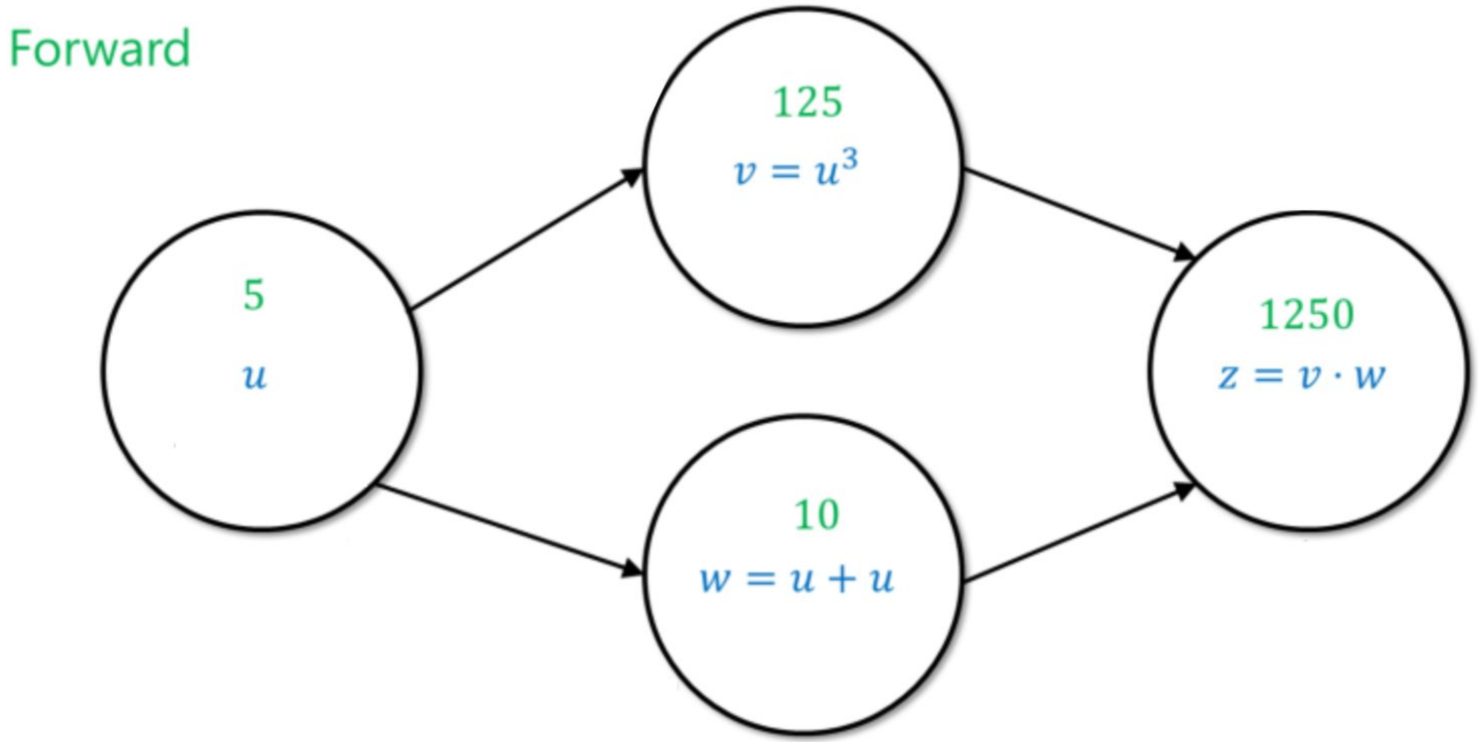
Backpropagation- An Example



https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

Cornell Bowers C·IS

Backpropagation- An Example



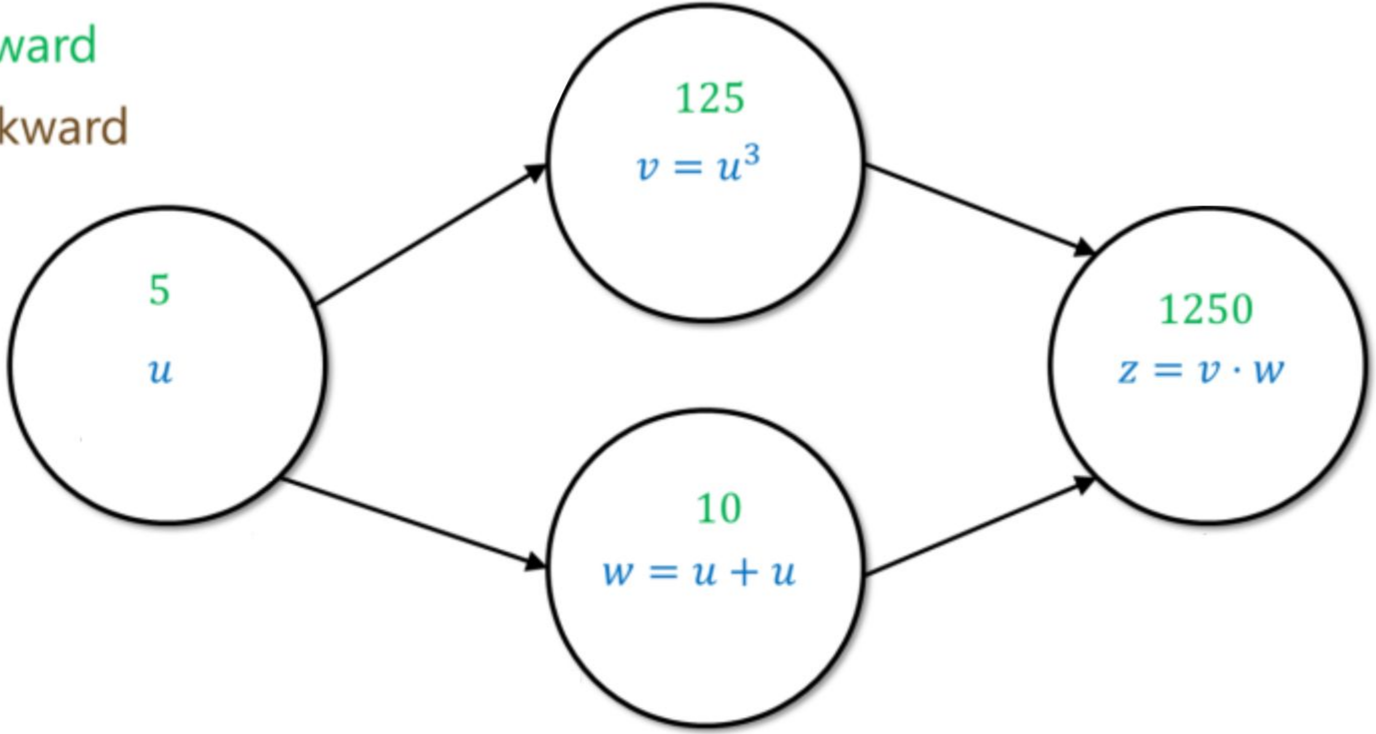
https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

Backpropagation- An Example

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v}{\partial u} \cdot \frac{\partial z}{\partial v} + \frac{\partial w}{\partial u} \cdot \frac{\partial z}{\partial w}\right)$$

Forward

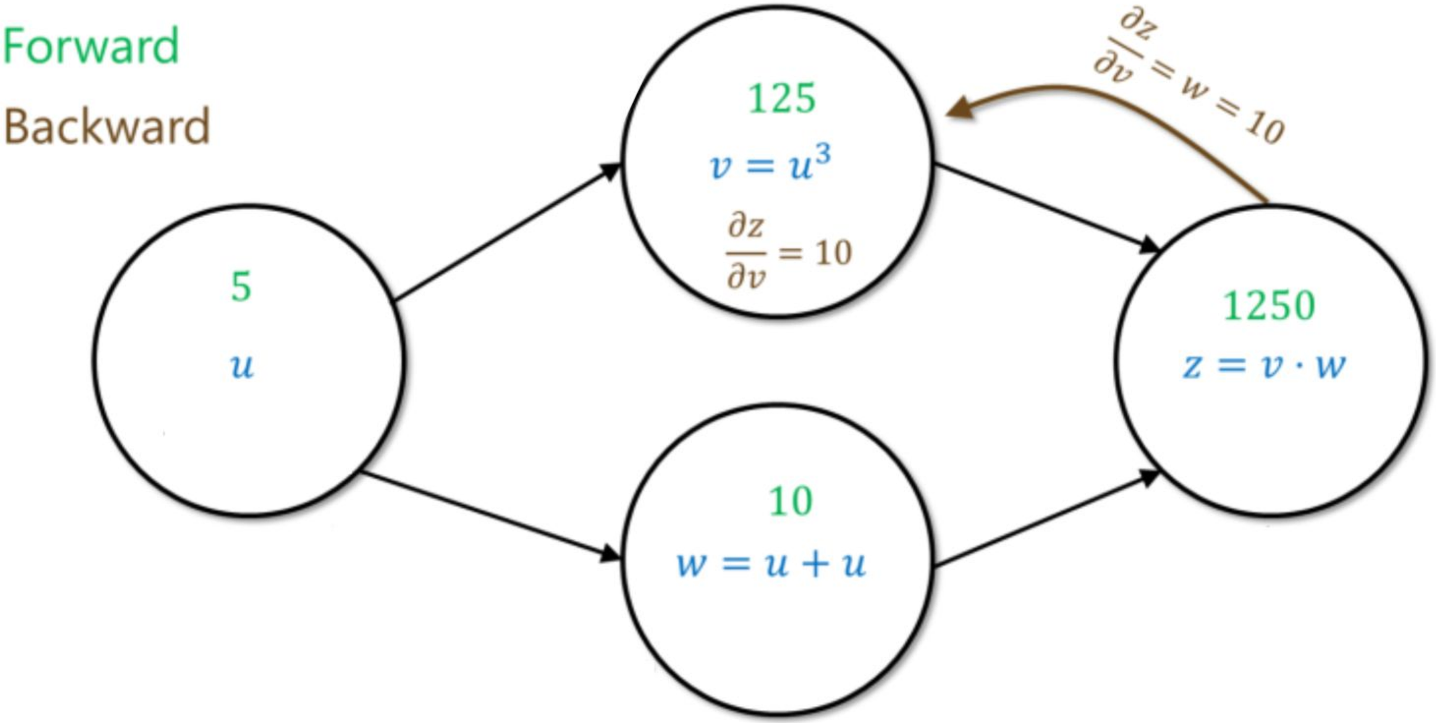
Backward



https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

Backpropagation- An Example

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v}{\partial u}\cdot\frac{\partial z}{\partial v} + \frac{\partial w}{\partial u}\cdot\frac{\partial z}{\partial w}\right)$$

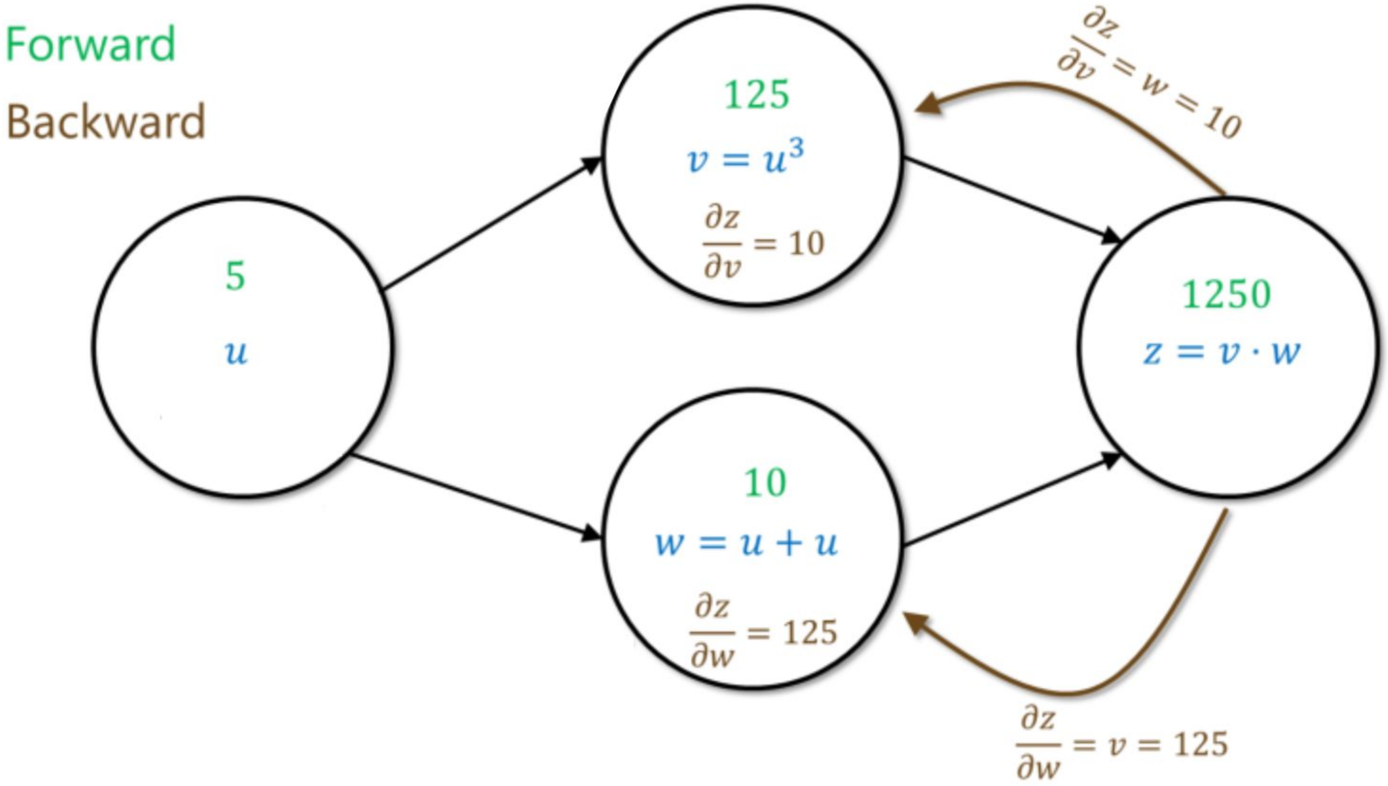


https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

Cornell Bowers C·IS

Backpropagation- An Example

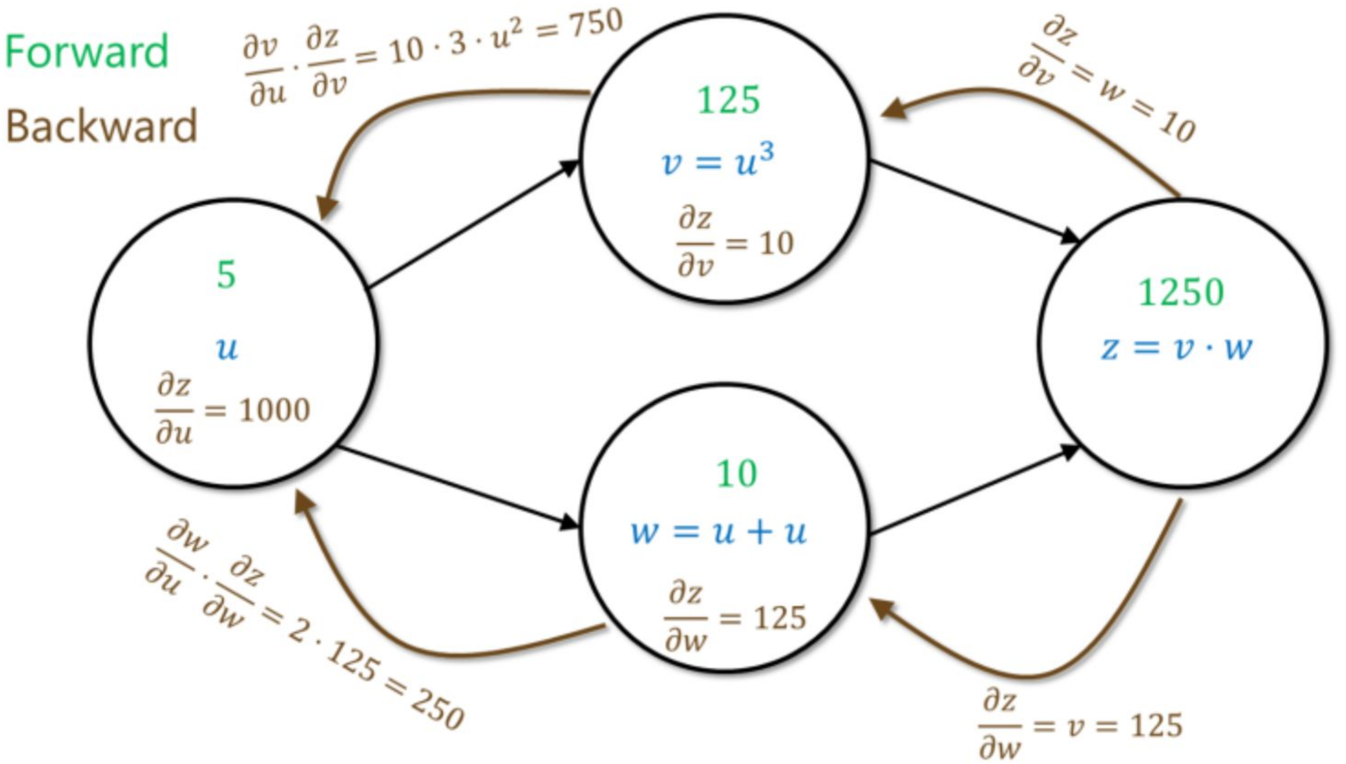
$$\frac{\partial z}{\partial u} = \left(\frac{\partial v}{\partial u} \cdot \frac{\partial z}{\partial v} + \frac{\partial w}{\partial u} \cdot \frac{\partial z}{\partial w}\right)$$



https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

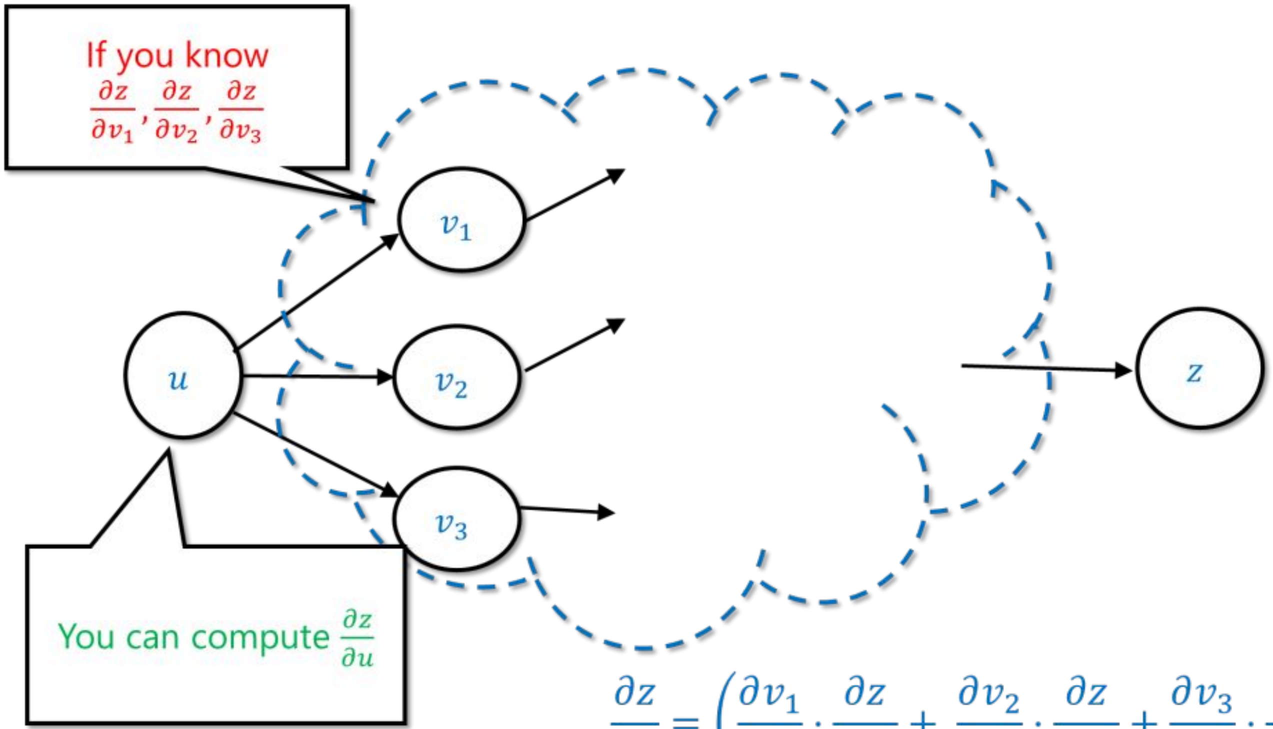
Backpropagation- An Example

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v}{\partial u} \cdot \frac{\partial z}{\partial v} + \frac{\partial w}{\partial u} \cdot \frac{\partial z}{\partial w}\right)$$



https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

Backpropagation- Key Idea

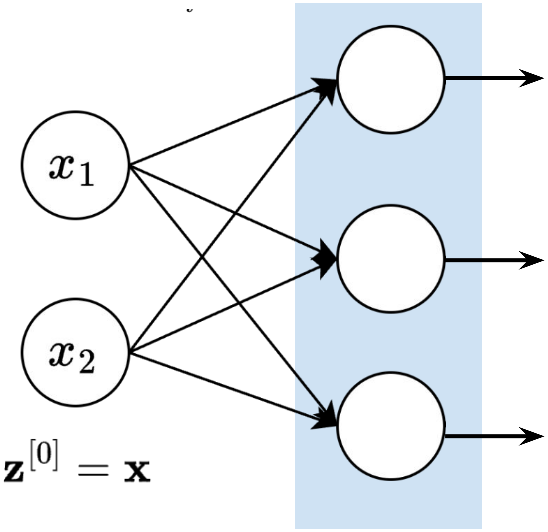


$$\frac{\partial z}{\partial u} = \left(\frac{\partial v_1}{\partial u} \cdot \frac{\partial z}{\partial v_1} + \frac{\partial v_2}{\partial u} \cdot \frac{\partial z}{\partial v_2} + \frac{\partial v_3}{\partial u} \cdot \frac{\partial z}{\partial v_3} \right)$$

https://windowsontheory.org/2020/11/03/yet-another-backpropagation-tutorial/

Forward Pass - MLP

$$\mathbf{a}^{[1]} = \mathbf{W}^{[1]}\mathbf{z}^{[0]}$$



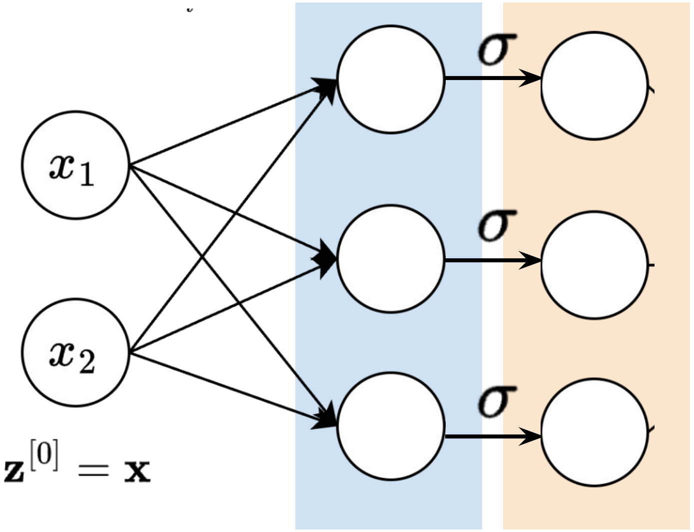
$$\mathbf{z}^{[0]} = \mathbf{x}$$

Algorithm Forward Pass through MLP

Input: input $\mathbf{x}$, weight matrices $\mathbf{W}^{[1]}, \ldots, \mathbf{W}^{[L]}$, bias vectors $\mathbf{b}^{[1]}, \ldots, \mathbf{b}^{[L]}$
$\mathbf{z}^{[0]} = \mathbf{x}$ ▷ Initialize input
for $l = 1$ **to** L **do**
  $\mathbf{a}^{[l]} = \mathbf{W}^{[l]}\mathbf{z}^{[l-1]} + \mathbf{b}^{[l]}$ ▷ Linear transformation
  $\mathbf{z}^{[l]} = \sigma^{[l]}(\mathbf{a}^{[l]})$ ▷ Nonlinear activation
end for
Output: $\mathbf{z}^{[L]}$

Forward Pass - MLP

$$\mathbf{a}^{[1]} = \mathbf{W}^{[1]}\mathbf{z}^{[0]}$$

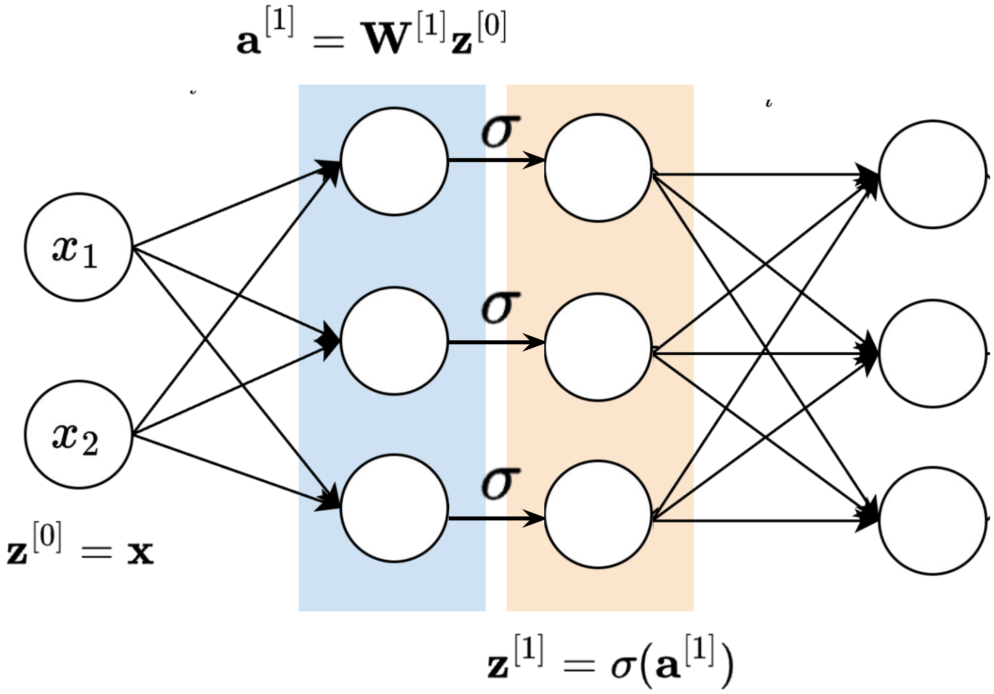


$$\mathbf{z}^{[0]} = \mathbf{x}$$

$$\mathbf{z}^{[1]} = \sigma(\mathbf{a}^{[1]})$$

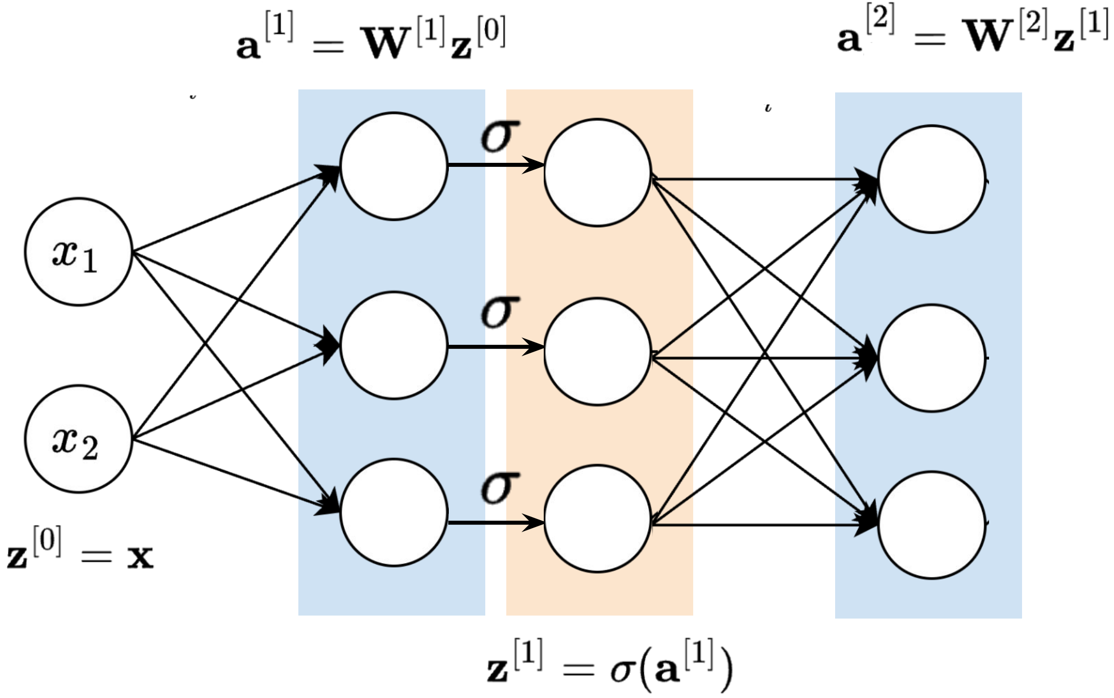
Cornell Bowers C·IS

Forward Pass - MLP

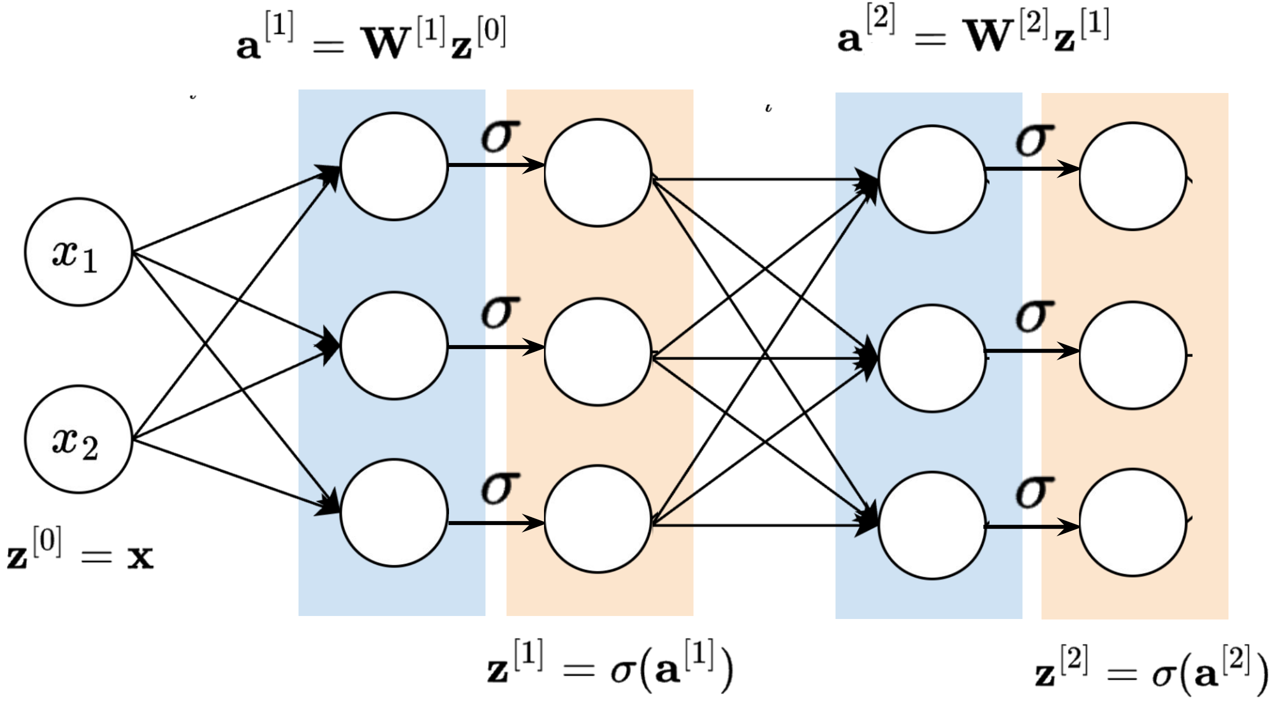


Cornell Bowers C·IS

Forward Pass - MLP

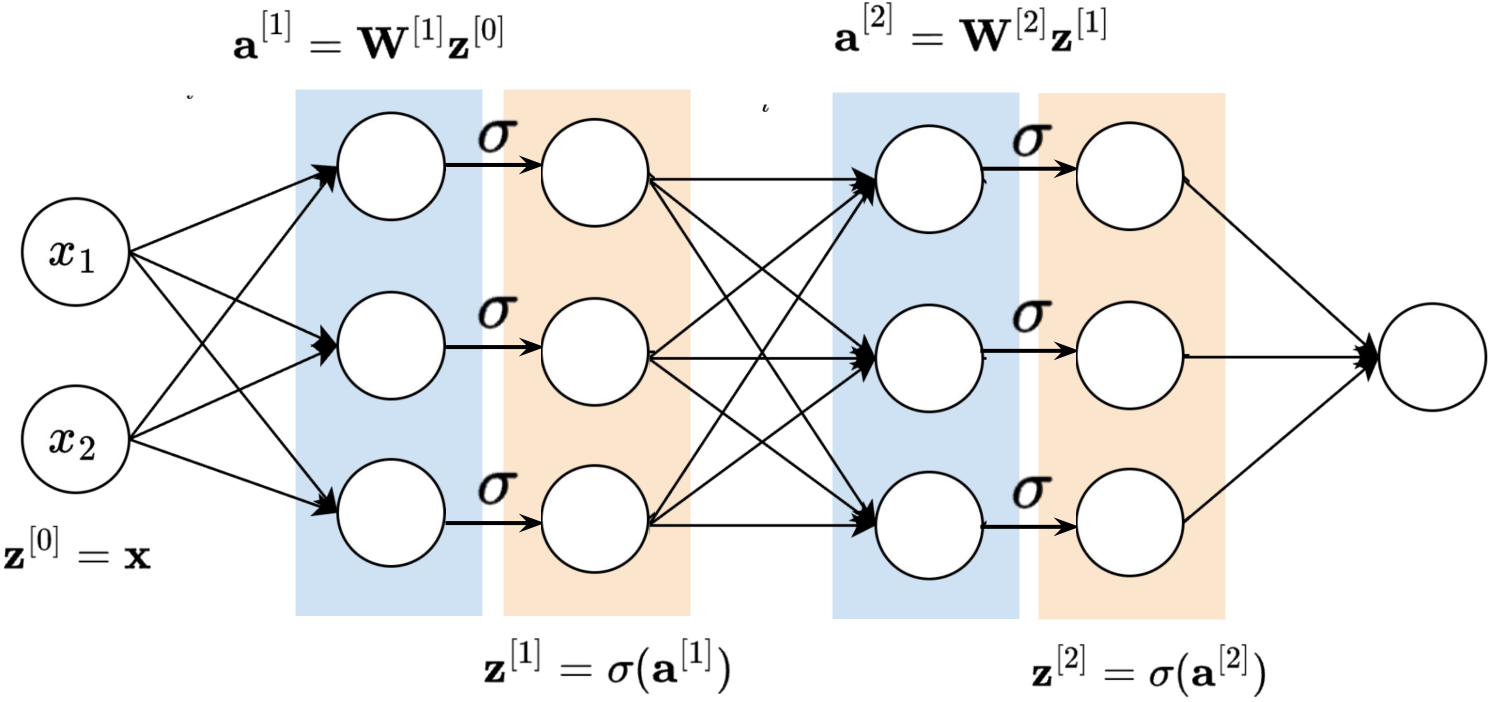


Forward Pass - MLP

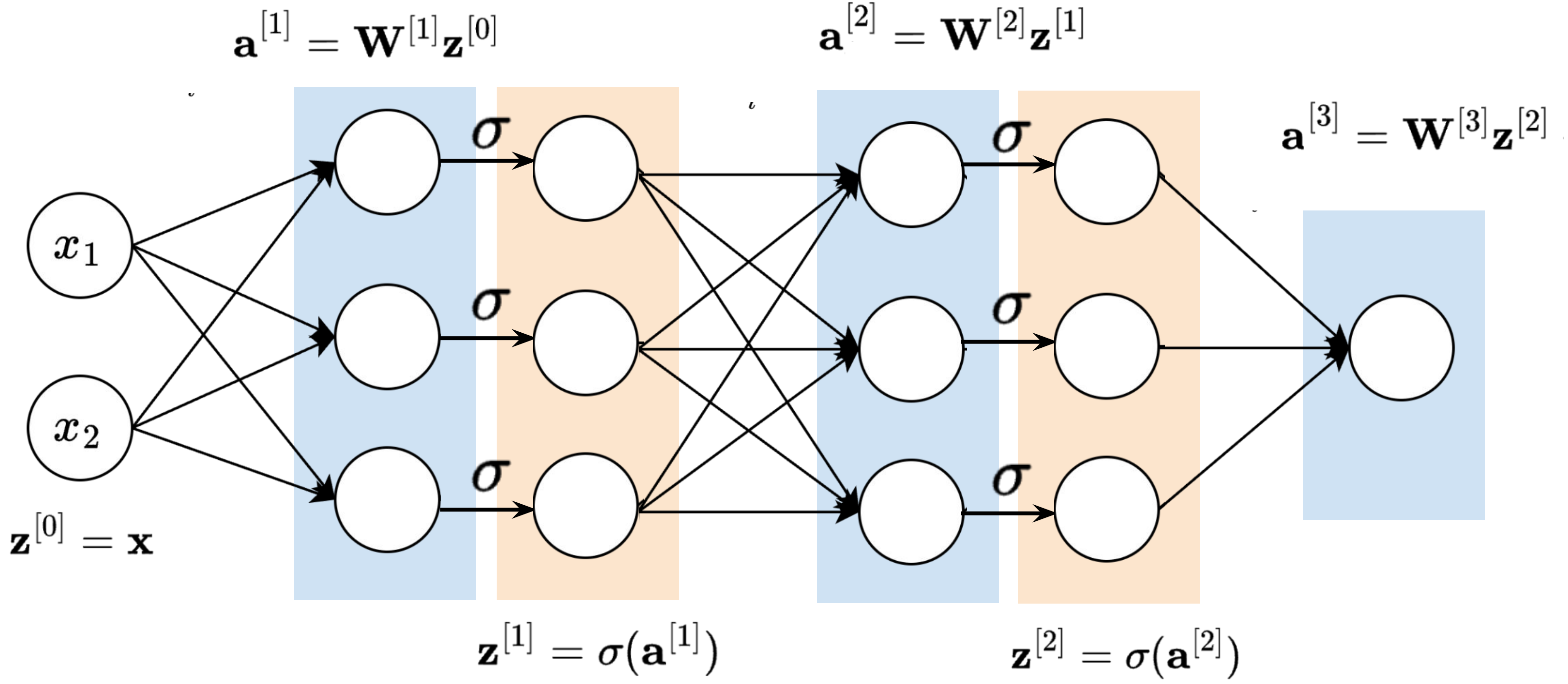


Cornell Bowers C·IS

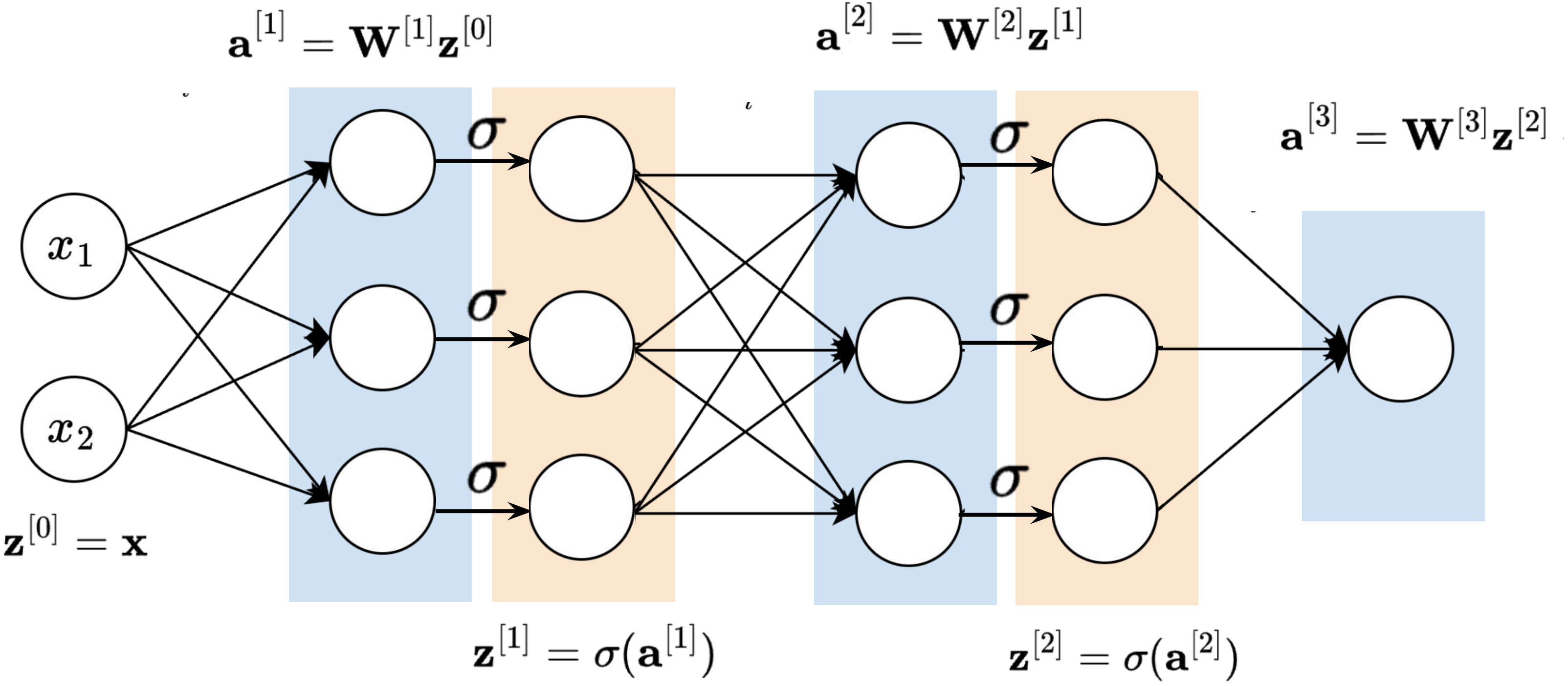
Forward Pass - MLP



Forward Pass - MLP

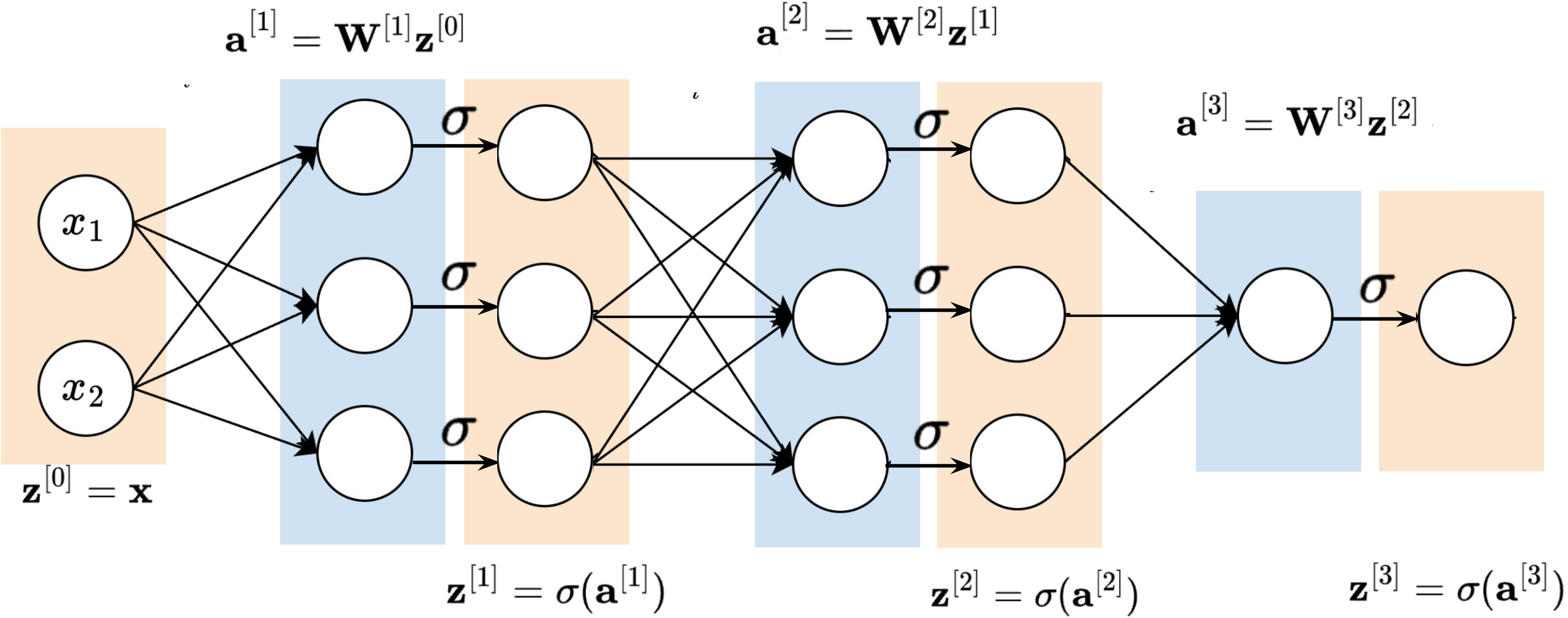


Forward Pass - MLP



Cornell Bowers C·IS

Forward Pass - MLP

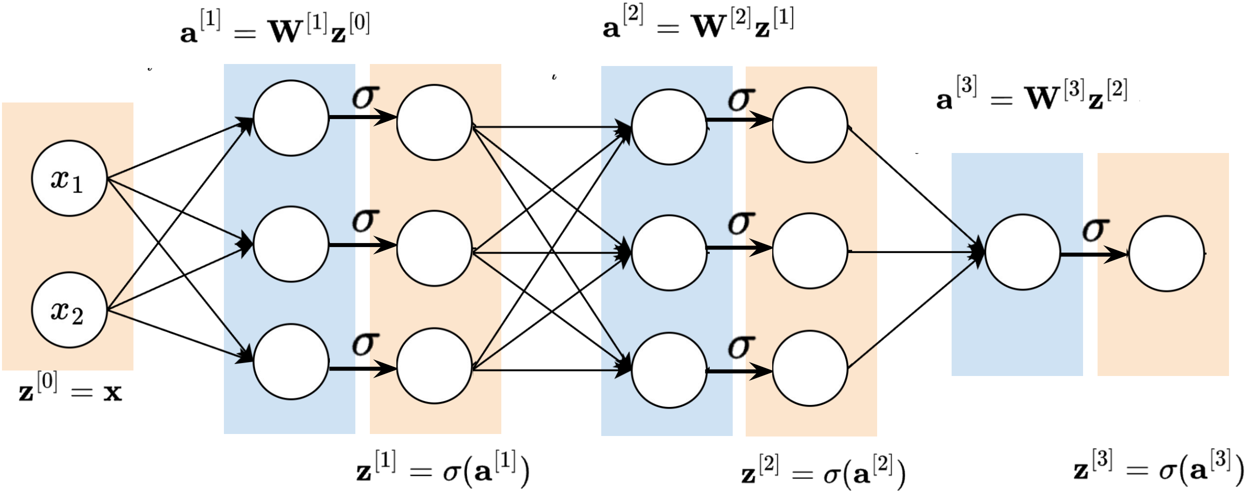


Cornell Bowers C·IS

Forward Pass - MLP

Algorithm Forward Pass through MLP

1: **Input:** input $\mathbf{x}$, weight matrices $\mathbf{W}^{[1]}, \ldots, \mathbf{W}^{[L]}$, bias vectors $\mathbf{b}^{[1]}, \ldots, \mathbf{b}^{[L]}$
2: $\mathbf{z}^{[0]} = \mathbf{x}$ ▷ Initialize input
3: **for** $l = 1$ **to** L **do**
4: $\quad \mathbf{a}^{[l]} = \mathbf{W}^{[l]}\mathbf{z}^{[l-1]} + \mathbf{b}^{[l]}$ ▷ Linear transformation
5: $\quad \mathbf{z}^{[l]} = \sigma^{[l]}(\mathbf{a}^{[l]})$ ▷ Nonlinear activation
6: **end for**
7: **Output:** $\mathbf{z}^{[L]}$

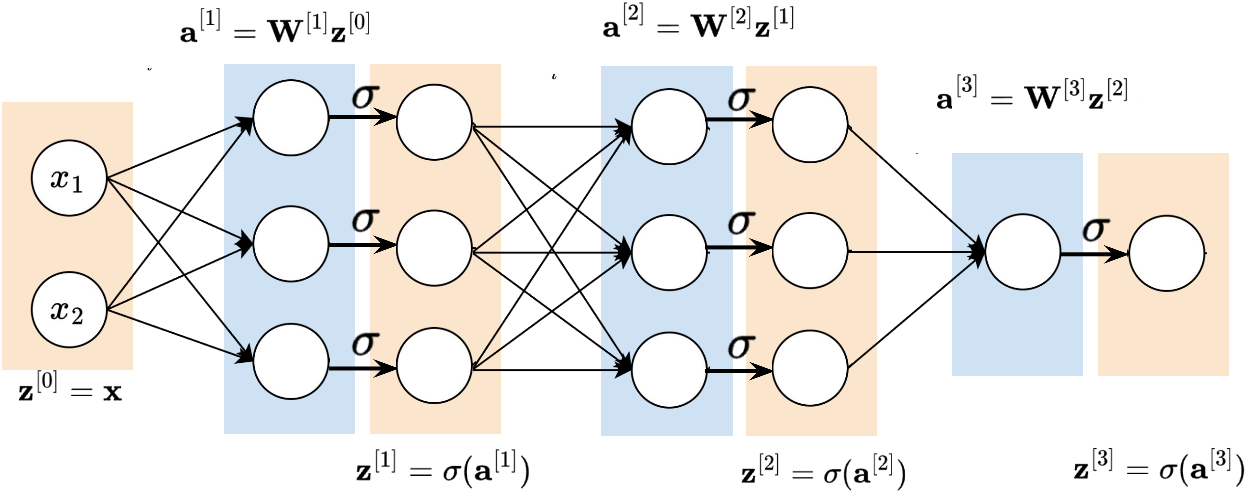


Cornell Bowers CIS

Forward Pass - MLP

Algorithm Forward Pass through MLP

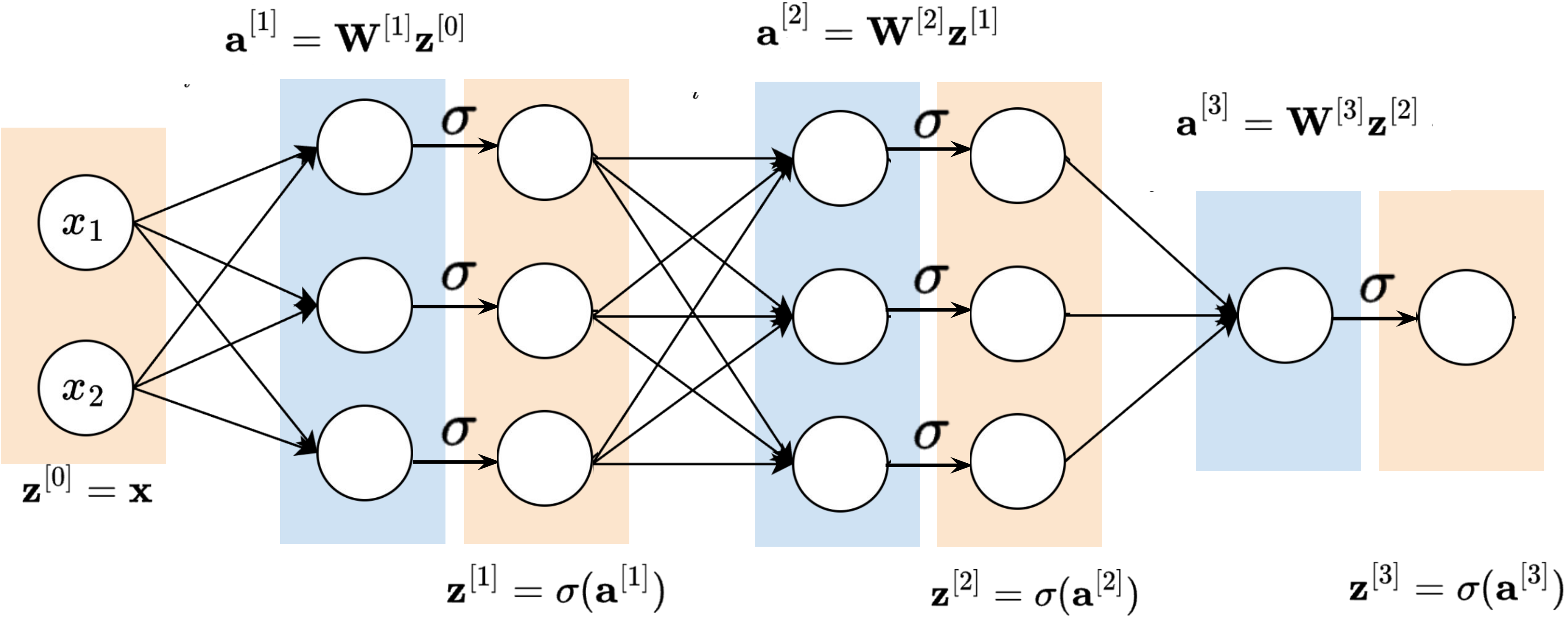
```
Input: input x, weight matrices W^[1], ..., W^[L], bias vectors b^[1], ..., b^[L]
z^[0] = x                                      ▷ Initialize input
for l = 1 to L do
    a^[l] = W^[l] z^[l-1] + b^[l]              ▷ Linear transformation
    z^[l] = σ^[l](a^[l])                       ▷ Nonlinear activation
end for
Output: z^[L]
```



Cornell Bowers C·IS

Backprop

$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$

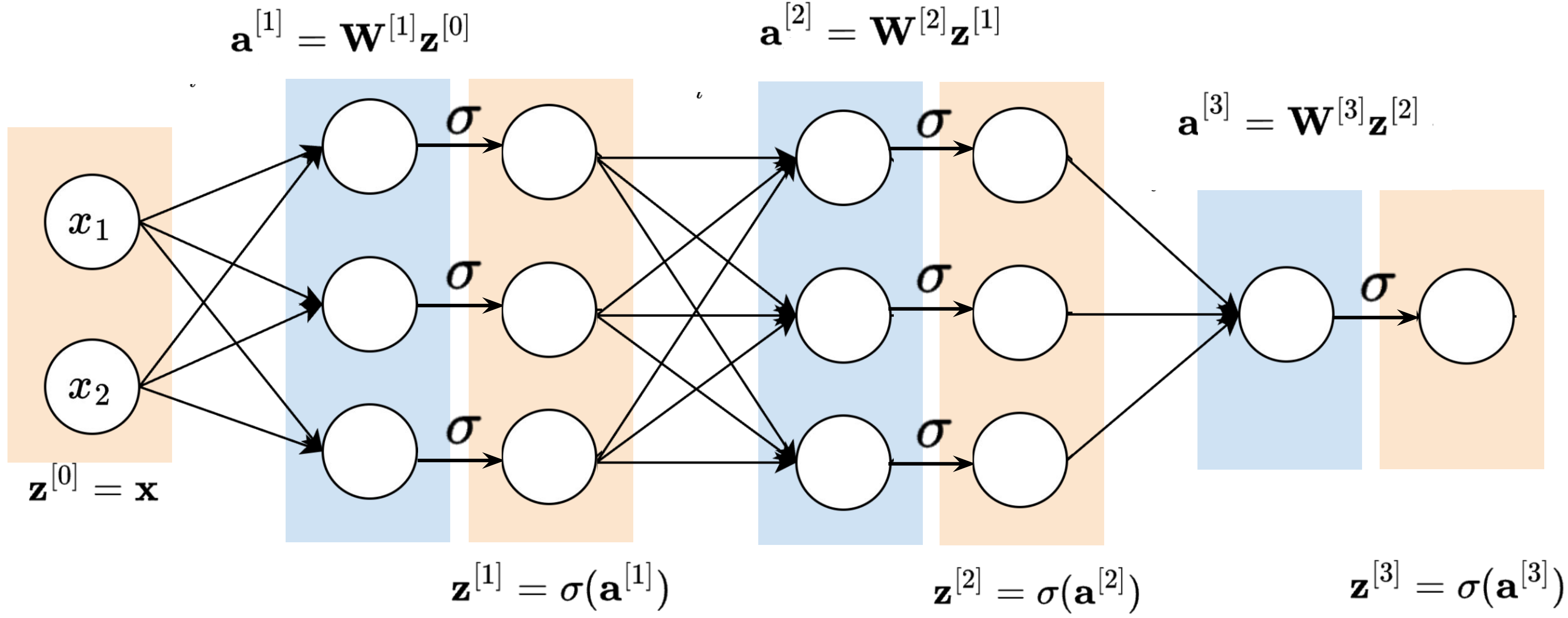


Cornell Bowers C·IS

Backprop

$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$

We can directly compute $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$!



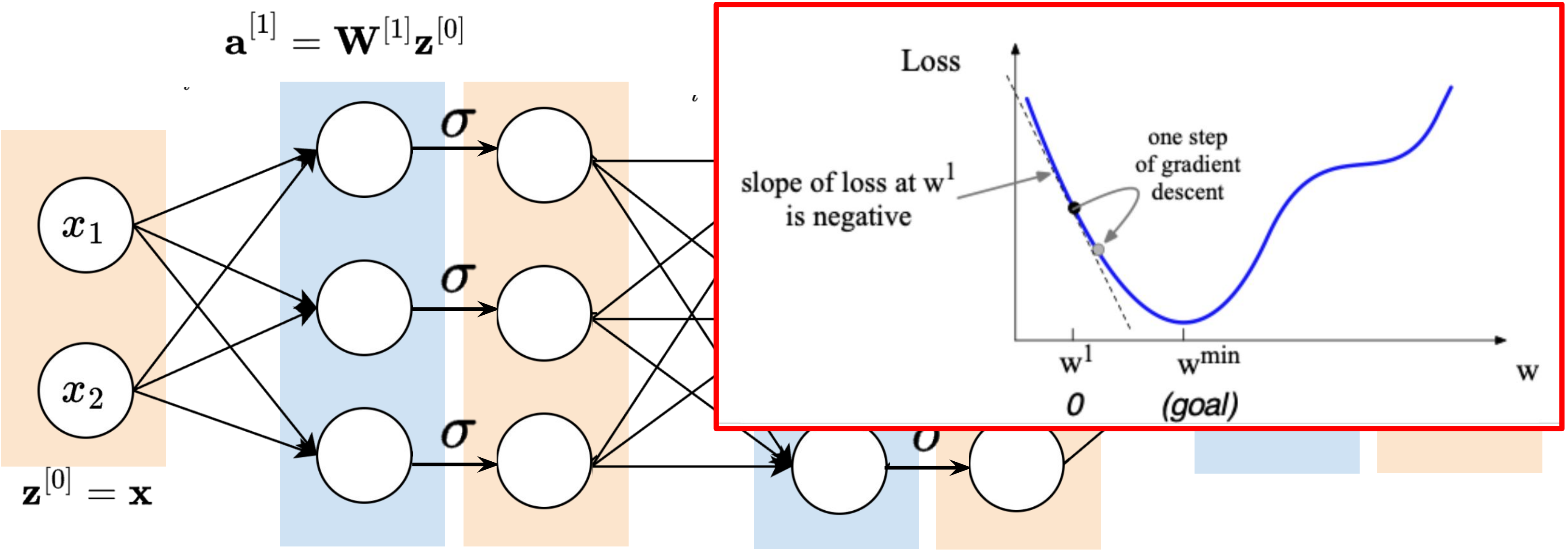
Cornell Bowers CIS

Backprop

$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$

We can directly compute $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$!

$\mathbf{a}^{[1]} = \mathbf{W}^{[1]}\mathbf{z}^{[0]}$



$\mathbf{z}^{[0]} = \mathbf{x}$

$\mathbf{z}^{[1]} = \sigma(\mathbf{a}^{[1]})$

$\mathbf{z}^{[2]} = \sigma(\mathbf{a}^{[2]})$

$\mathbf{z}^{[3]} = \sigma(\mathbf{a}^{[3]})$

Cornell Bowers C·IS

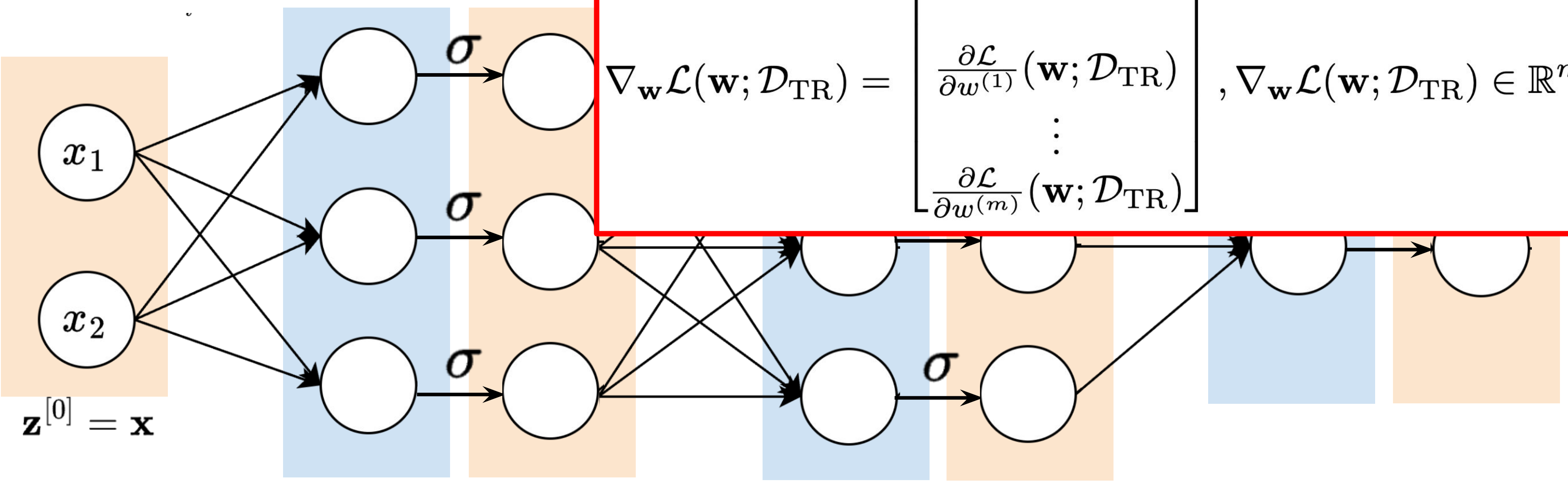
Backprop

Loss $= \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$

We can directly compute $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$!

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathcal{D}_{\text{TR}}) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w^{(0)}}(\mathbf{w}; \mathcal{D}_{\text{TR}}) \\ \frac{\partial \mathcal{L}}{\partial w^{(1)}}(\mathbf{w}; \mathcal{D}_{\text{TR}}) \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w^{(m)}}(\mathbf{w}; \mathcal{D}_{\text{TR}}) \end{bmatrix}, \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathcal{D}_{\text{TR}}) \in \mathbb{R}^m$$

$\mathbf{a}^{[1]} = \mathbf{W}^{[1]}\mathbf{z}^{[0]}$



$\mathbf{z}^{[0]} = \mathbf{x}$

$\mathbf{z}^{[1]} = \sigma(\mathbf{a}^{[1]})$

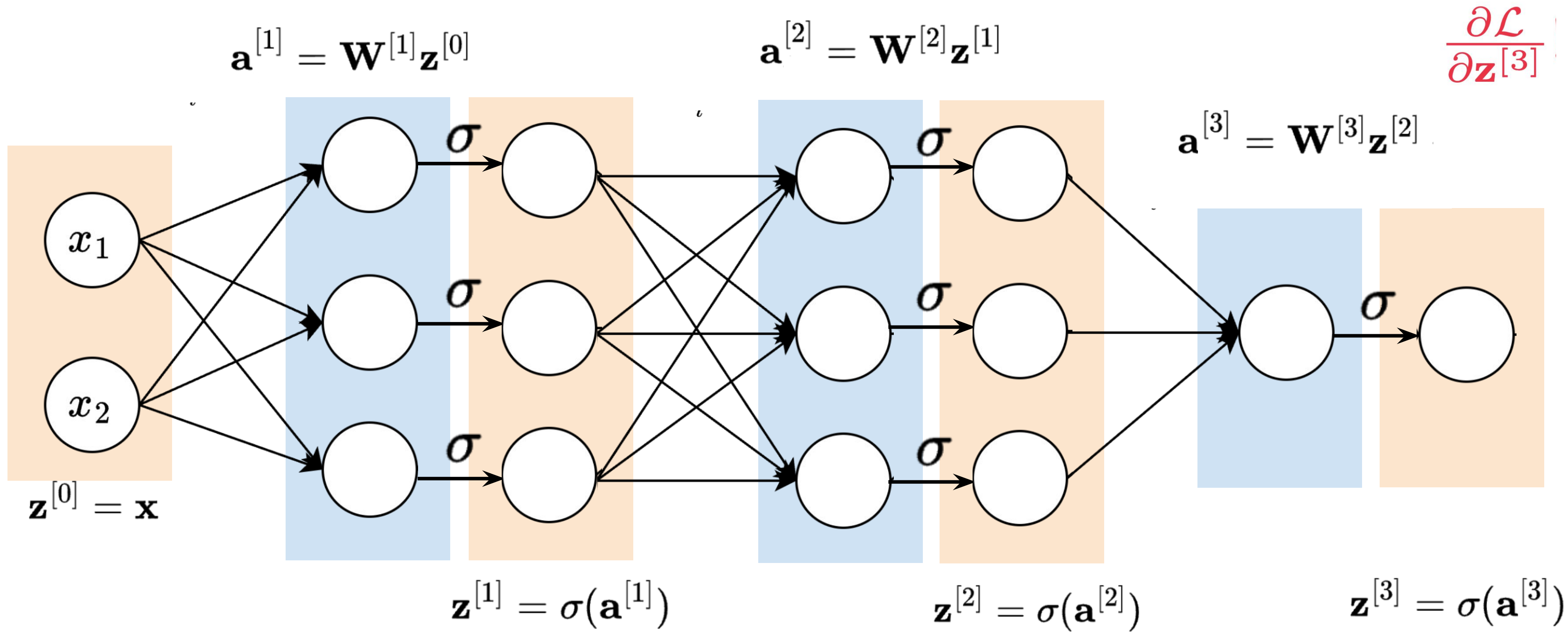
$\mathbf{z}^{[2]} = \sigma(\mathbf{a}^{[2]})$

$\mathbf{z}^{[3]} = \sigma(\mathbf{a}^{[3]})$

Cornell Bowers C·IS

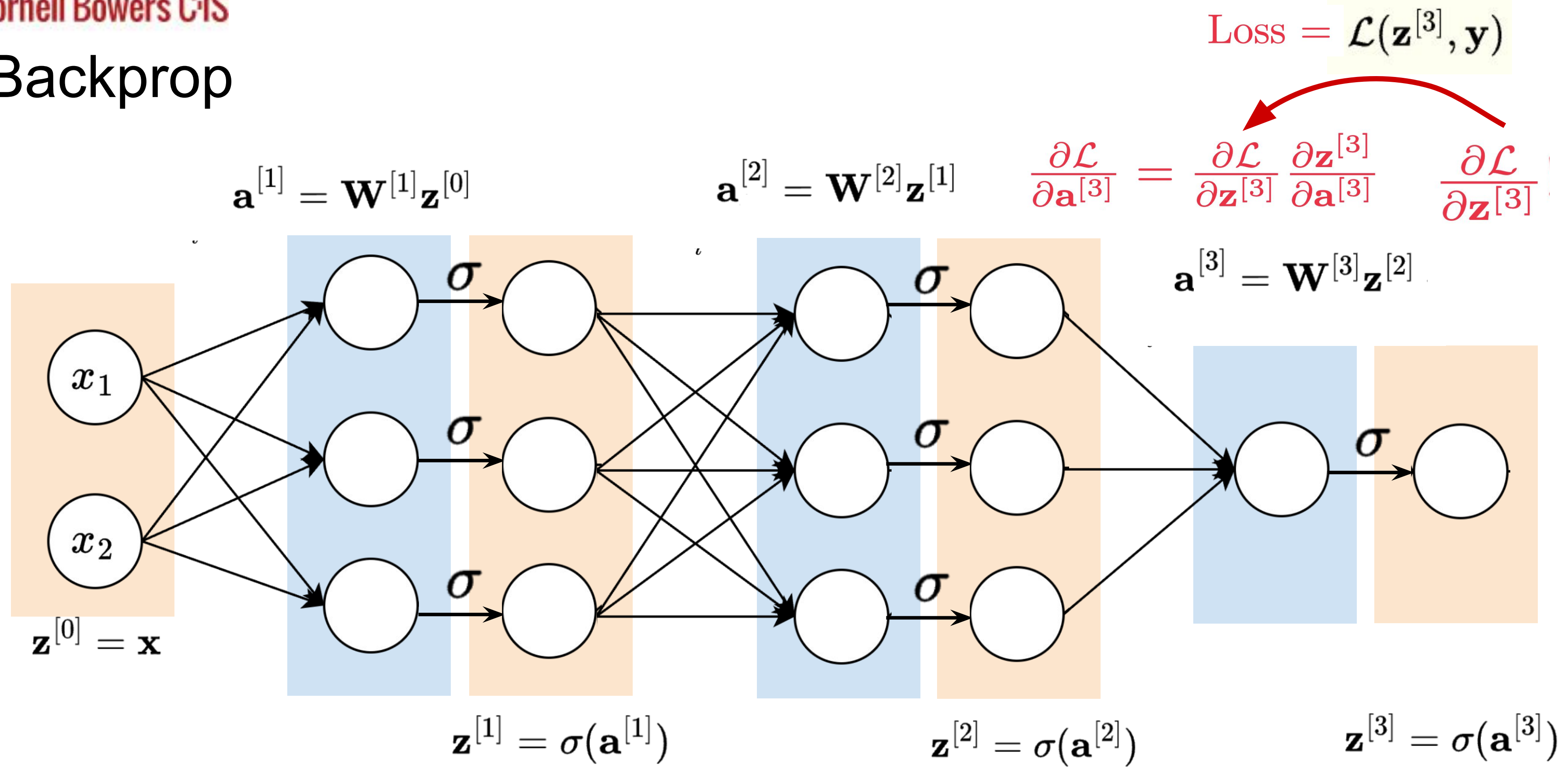
Backprop

Loss $= \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$



Cornell Bowers C·IS

Backprop



Cornell Bowers C·IS

Backprop

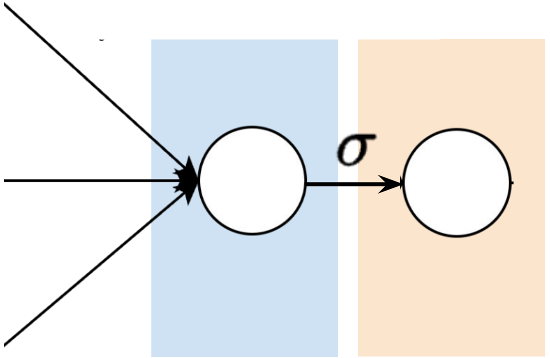
$$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{a}^{[3]}}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \odot \sigma^{[3]\prime}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$$

$$\mathbf{a}^{[3]} = \mathbf{W}^{[3]}\mathbf{z}^{[2]}$$



$$\mathbf{z}^{[3]} = \sigma(\mathbf{a}^{[3]})$$

Cornell Bowers CIS

Backprop

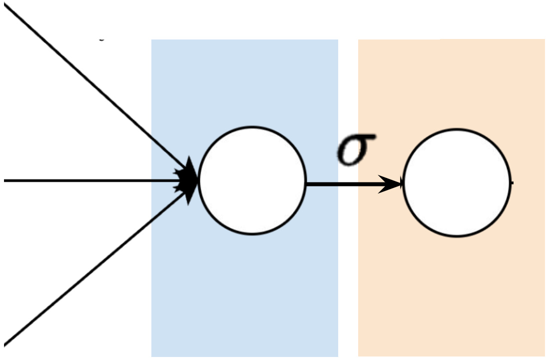
$$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$$

$$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{a}^{[3]}}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \odot \sigma^{[3]'}$$

$$\mathbf{a}^{[3]} = \mathbf{W}^{[3]}\mathbf{z}^{[2]}$$



$$\mathbf{z}^{[3]} = \sigma(\mathbf{a}^{[3]})$$

Cornell Bowers C·IS

Backprop

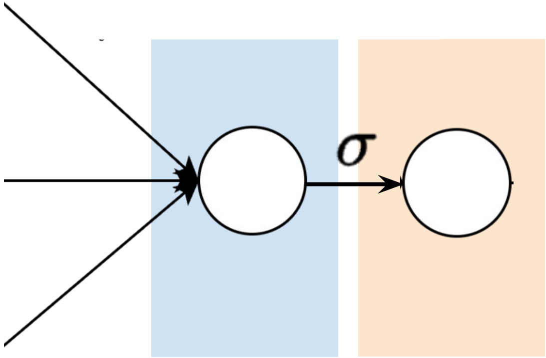
$$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$$

$$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{a}^{[3]}}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \odot \sigma^{[3]\prime}$$

$$\mathbf{a}^{[3]} = \mathbf{W}^{[3]}\mathbf{z}^{[2]}$$



$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{W}^{[3]}}$$

$$= \delta^{[3]} (\mathbf{z}^{[2]})^T$$

$$\mathbf{z}^{[3]} = \sigma(\mathbf{a}^{[3]})$$

Cornell Bowers CIS

Backprop

$$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

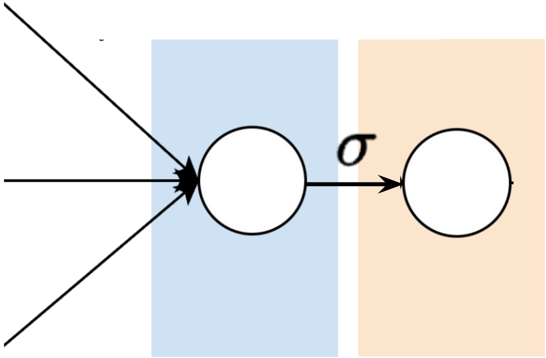
$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$$

For propagation to next layer:

$$\boxed{\delta^{[3]} =} \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{a}^{[3]}}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \odot \sigma^{[3]\prime}$$

$$\mathbf{a}^{[3]} = \mathbf{W}^{[3]}\mathbf{z}^{[2]}$$



For weight updates:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{W}^{[3]}}$$

$$= \delta^{[3]} (\mathbf{z}^{[2]})^T$$

$$\mathbf{z}^{[3]} = \sigma(\mathbf{a}^{[3]})$$

Cornell Bowers CIS

Backprop

$$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$$

$$\mathbf{a}^{[3]} = \mathbf{W}^{[3]}\mathbf{z}^{[2]}$$

For propagation to next layer:

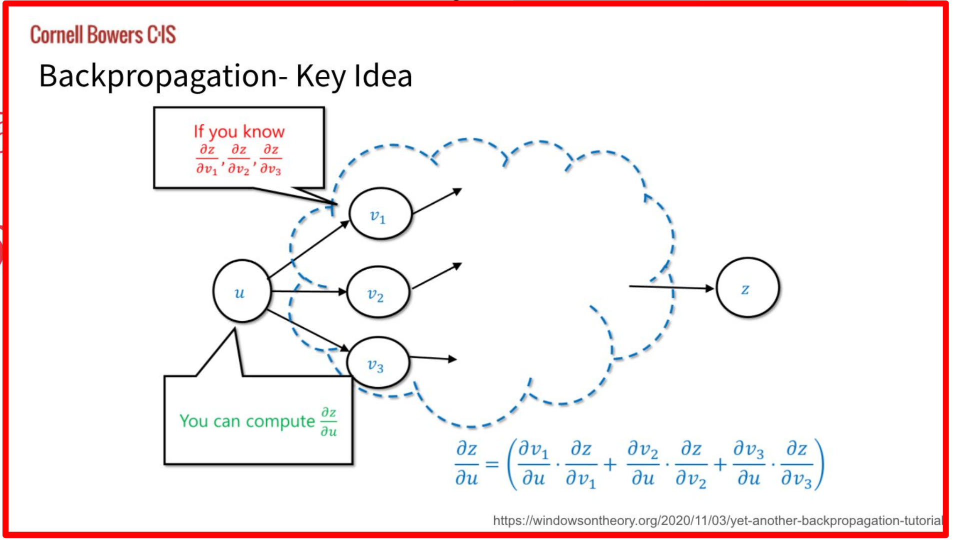
$$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{a}^{[3]}}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}} \odot \sigma^{[3]'}$$

For weight updates:

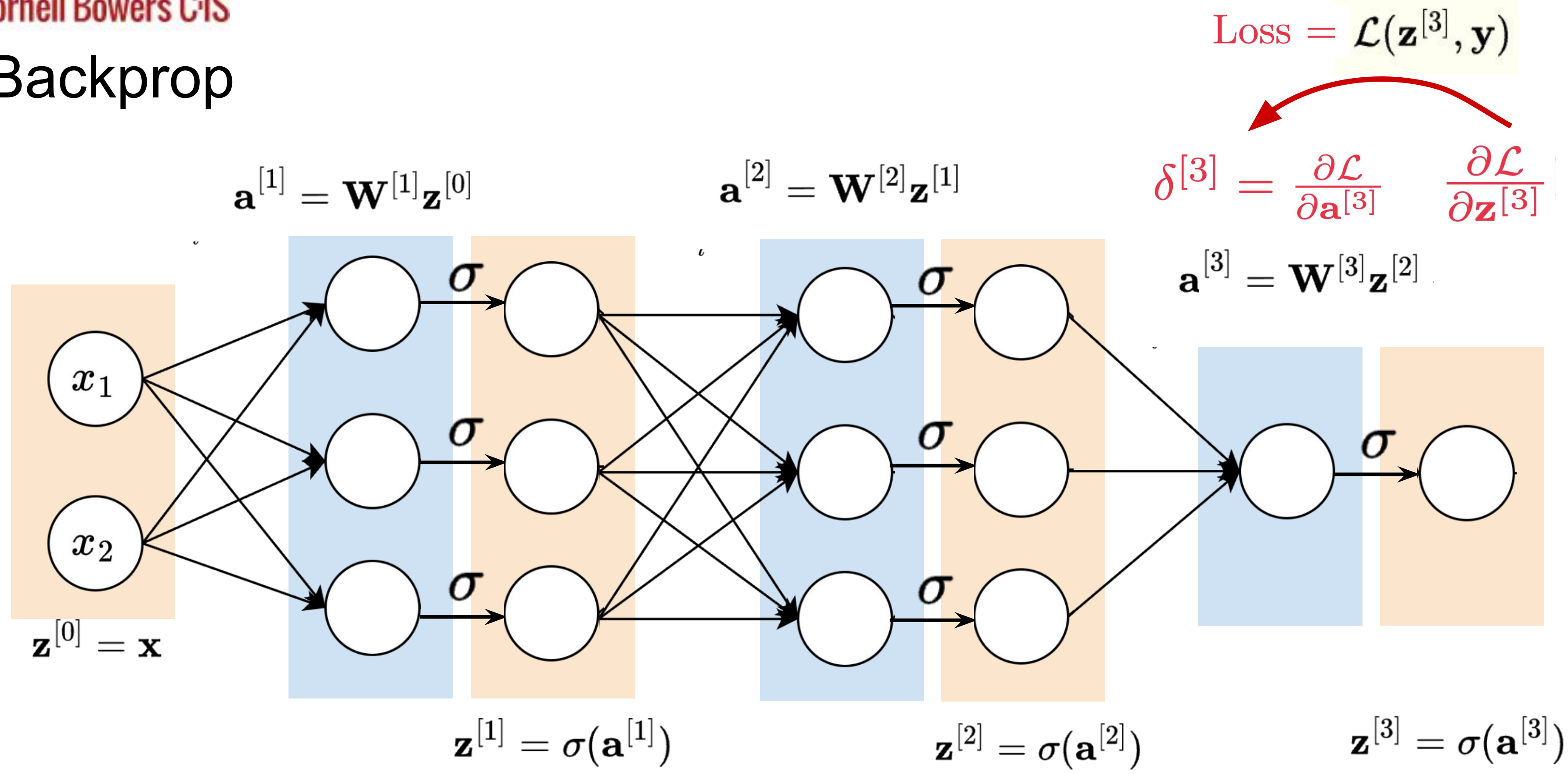
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[3]}} = \partial \ldots$$

$$= \delta \ldots$$



Cornell Bowers C·IS

Backprop

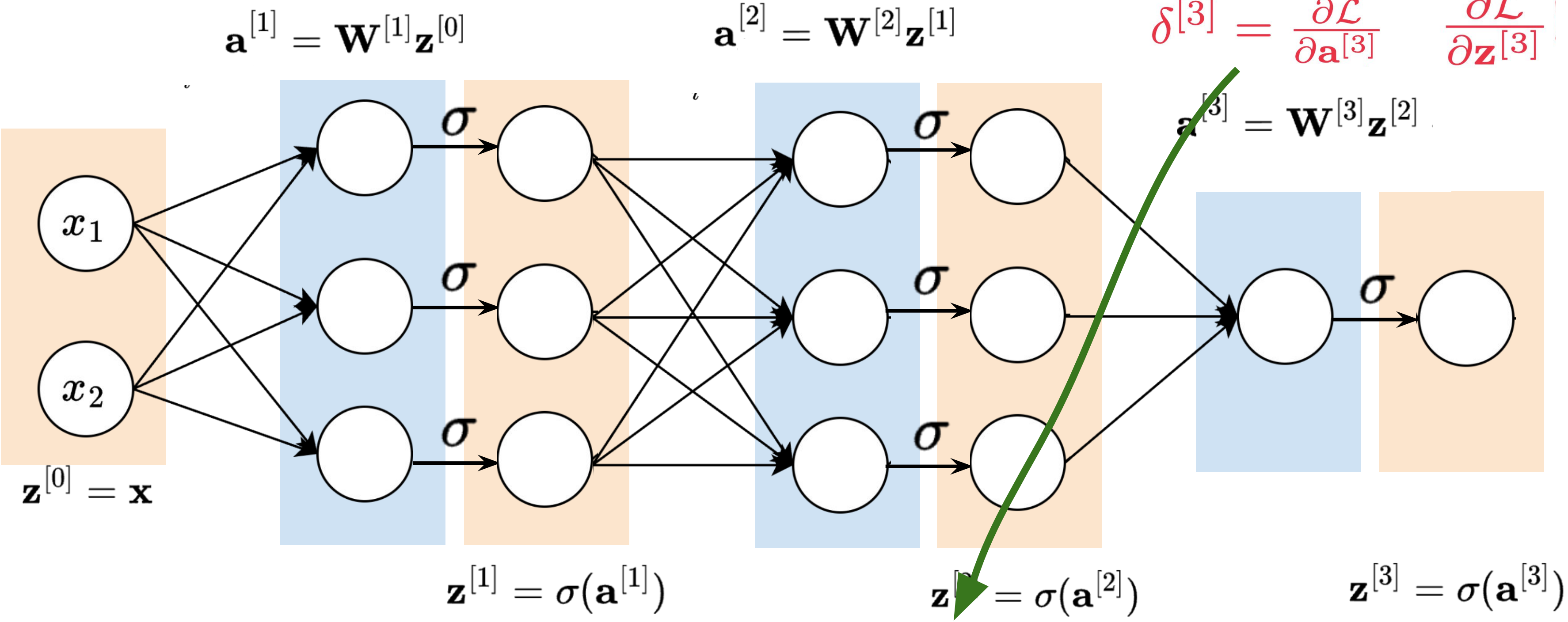


Cornell Bowers CIS

Backprop

$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$

$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$



$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{z}^{[2]}} = (W^{[3]})^T \delta^{[3]}$$

Cornell Bowers C·IS

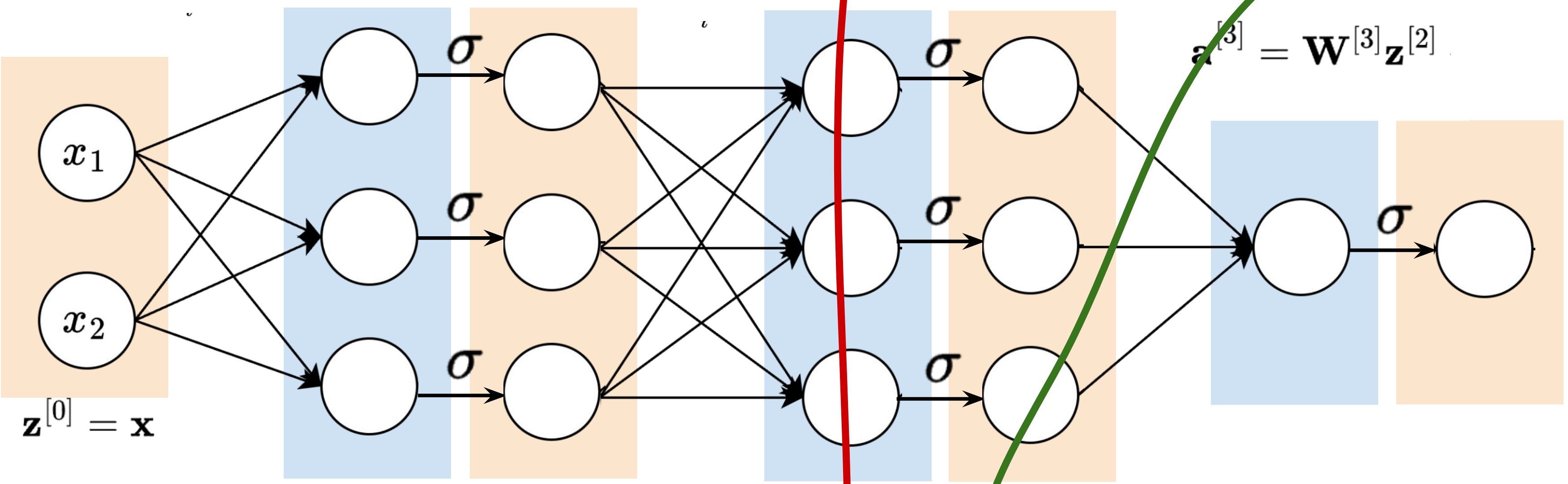
Backprop

$$\delta^{[2]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[2]}}$$

$$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

$$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$$

$$\mathbf{a}^{[1]} = \mathbf{W}^{[1]}\mathbf{z}^{[0]} \qquad \mathbf{a}^{[2]} = \mathbf{W}^{[2]}\mathbf{z}^{[1]} \qquad \mathbf{a}^{[3]} = \mathbf{W}^{[3]}\mathbf{z}^{[2]}$$



$$\mathbf{z}^{[0]} = \mathbf{x} \qquad \mathbf{z}^{[1]} = \sigma(\mathbf{a}^{[1]}) \qquad \mathbf{z}^{[2]} = \sigma(\mathbf{a}^{[2]}) \qquad \mathbf{z}^{[3]} = \sigma(\mathbf{a}^{[3]})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{z}^{[2]}} = (W^{[3]})^T \delta^{[3]}$$

Cornell Bowers C·IS

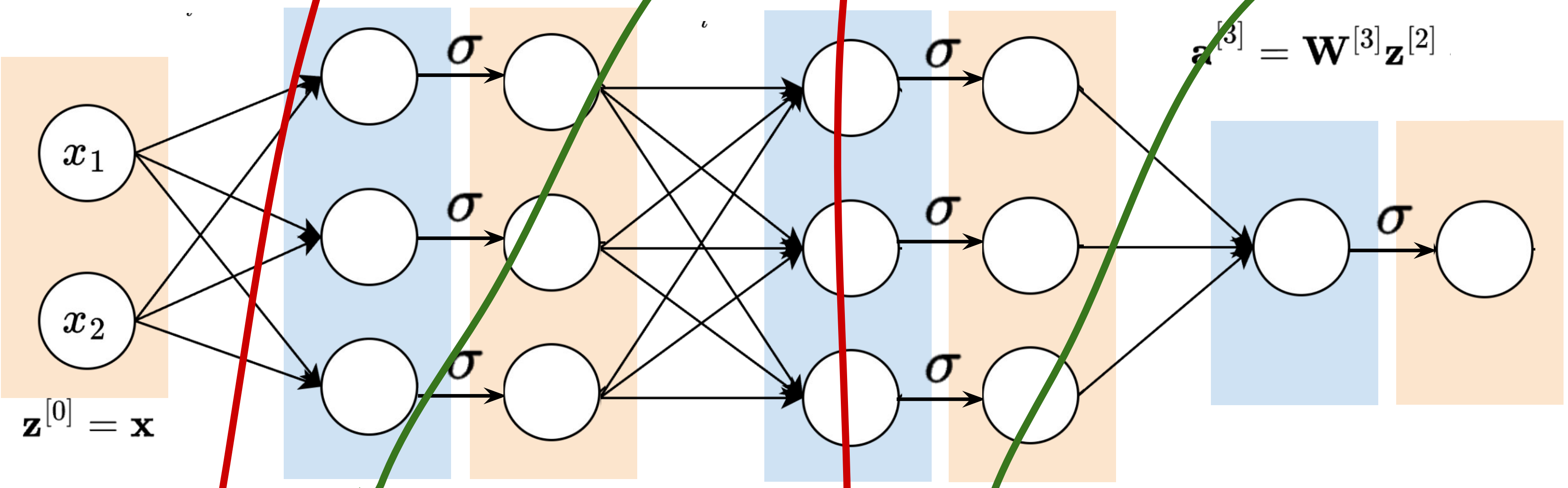
$$\text{Loss} = \mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

$$\delta^{[1]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{a}^{[1]}}$$

$$\delta^{[2]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[2]}}$$

$$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$$

$$\mathbf{a}^{[1]} = \mathbf{W}^{[1]}\mathbf{z}^{[0]} \qquad \mathbf{a}^{[2]} = \mathbf{W}^{[2]}\mathbf{z}^{[1]} \qquad \mathbf{a}^{[3]} = \mathbf{W}^{[3]}\mathbf{z}^{[2]}$$



$$\mathbf{z}^{[0]} = \mathbf{x} \qquad \mathbf{z}^{[1]} = \sigma(\mathbf{a}^{[1]}) \qquad \mathbf{z}^{[2]} = \sigma(\mathbf{a}^{[2]}) \qquad \mathbf{z}^{[3]} = \sigma(\mathbf{a}^{[3]})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[2]}} \frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[1]}} = (W^{[2]})^T \delta^{[2]}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{z}^{[2]}} = (W^{[3]})^T \delta^{[3]}$$

Cornell Bowers C·IS

Backpropagation

Algorithm Backward Pass through MLP (Detailed)

Input: $\{\mathbf{z}^{[1]}, \ldots, \mathbf{z}^{[L]}\}$, $\{\mathbf{a}^{[1]}, \ldots, \mathbf{a}^{[L]}\}$, loss gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}}$

$\delta^{[L]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \frac{\partial \mathbf{z}^{[L]}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \odot \sigma^{[L]\prime}(\mathbf{a}^{[L]})$ ▷ Error term

for $l = L$ **to** 1 **do**

$\quad \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} (\mathbf{z}^{[l-1]})^T$ ▷ Gradient of weights

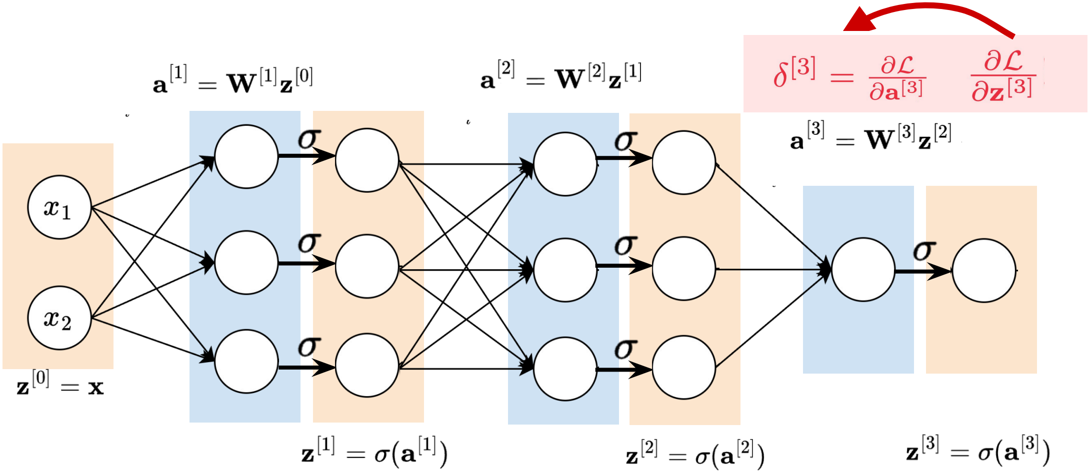
$\quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{b}^{[l]}} = \delta^{[l]}$ ▷ Gradient of biases

$\quad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{z}^{[l-1]}} = (\mathbf{W}^{[l]})^T \delta^{[l]}$

$\quad \delta^{[l-1]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} \frac{\partial \mathbf{z}^{[l-1]}}{\partial \mathbf{a}^{[l-1]}} = ((\mathbf{W}^{[l]})^T \delta^{[l]}) \odot \sigma^{[l-1]\prime}(\mathbf{a}^{[l-1]})$

end for

Output: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1:L]}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1:L]}}$



$\mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$

We can directly compute $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$!

Cornell Bowers C·IS

Backpropagation

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{W}^{[3]}} = \delta^{[3]} (\mathbf{z}^{[2]})^T$$

Algorithm Backward Pass through MLP (Detailed)

Input: $\{\mathbf{z}^{[1]}, \ldots, \mathbf{z}^{[L]}\}$, $\{\mathbf{a}^{[1]}, \ldots, \mathbf{a}^{[L]}\}$, loss gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}}$

$\delta^{[L]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \frac{\partial \mathbf{z}^{[L]}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \odot \sigma^{[L]\prime}(\mathbf{a}^{[L]})$ ▷ Error term

for $l = L$ **to** 1 **do**

  $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} (\mathbf{z}^{[l-1]})^T$ ▷ Gradient of weights

  $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{b}^{[l]}} = \delta^{[l]}$ ▷ Gradient of biases

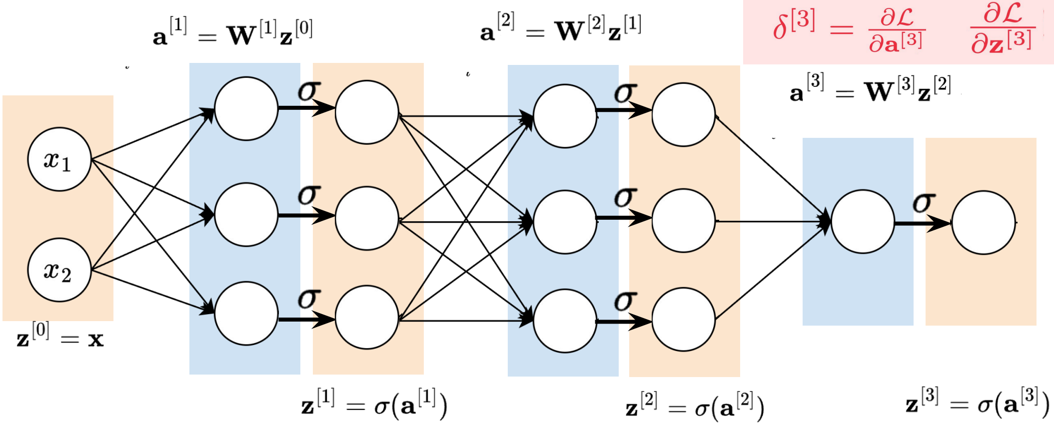
  $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{z}^{[l-1]}} = (\mathbf{W}^{[l]})^T \delta^{[l]}$

  $\delta^{[l-1]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} \frac{\partial \mathbf{z}^{[l-1]}}{\partial \mathbf{a}^{[l-1]}} = ((\mathbf{W}^{[l]})^T \delta^{[l]}) \odot \sigma^{[l-1]\prime}(\mathbf{a}^{[l-1]})$

end for

Output: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1:L]}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1:L]}}$

$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$



$$\mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

We can directly compute $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$!

Cornell Bowers CIS

Backpropagation

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{W}^{[3]}} = \delta^{[3]} (\mathbf{z}^{[2]})^T$$

Algorithm Backward Pass through MLP (Detailed)

Input: $\{\mathbf{z}^{[1]}, \ldots, \mathbf{z}^{[L]}\}$, $\{\mathbf{a}^{[1]}, \ldots, \mathbf{a}^{[L]}\}$, loss gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}}$

$\delta^{[L]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \frac{\partial \mathbf{z}^{[L]}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \odot \sigma^{[L]\prime}(\mathbf{a}^{[L]})$ ▷ Error term

for $l = L$ **to** 1 **do**

  $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} (\mathbf{z}^{[l-1]})^T$ ▷ Gradient of weights

  $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{b}^{[l]}} = \delta^{[l]}$ ▷ Gradient of biases

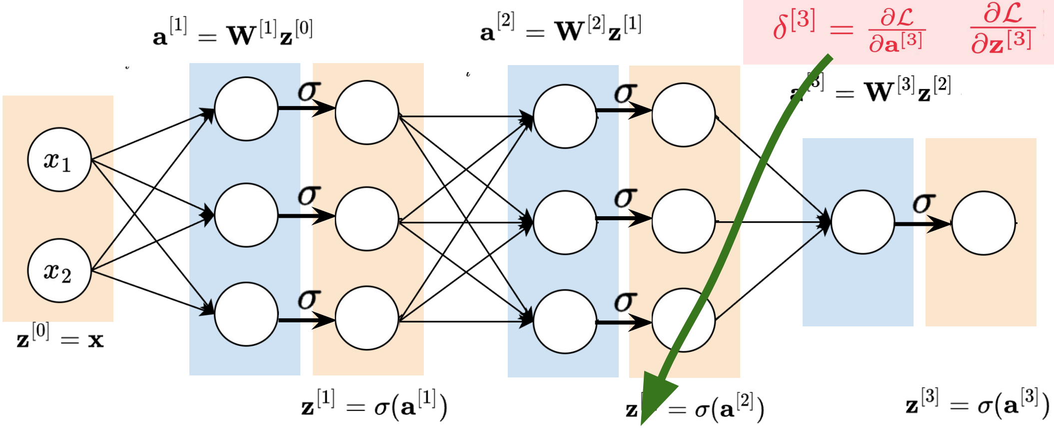
  $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{z}^{[l-1]}} = (\mathbf{W}^{[l]})^T \delta^{[l]}$

  $\delta^{[l-1]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} \frac{\partial \mathbf{z}^{[l-1]}}{\partial \mathbf{a}^{[l-1]}} = ((\mathbf{W}^{[l]})^T \delta^{[l]}) \odot \sigma^{[l-1]\prime}(\mathbf{a}^{[l-1]})$

end for

Output: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1:L]}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1:L]}}$

$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$



$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[2]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{z}^{[2]}} = (W^{[3]})^T \delta^{[3]}$$

$$\mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

We can directly compute $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$!

Cornell Bowers C·IS

Backpropagation

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[3]}} \frac{\partial \mathbf{a}^{[3]}}{\partial \mathbf{W}^{[3]}} = \delta^{[3]} (\mathbf{z}^{[2]})^T$$

Algorithm Backward Pass through MLP (Detailed)

Input: $\{\mathbf{z}^{[1]}, \ldots, \mathbf{z}^{[L]}\}$, $\{\mathbf{a}^{[1]}, \ldots, \mathbf{a}^{[L]}\}$, loss gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}}$

$\delta^{[L]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \frac{\partial \mathbf{z}^{[L]}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \odot \sigma^{[L]\prime}(\mathbf{a}^{[L]})$ ▷ Error term

for $l = L$ **to** 1 **do**

$\quad \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} (\mathbf{z}^{[l-1]})^T$ ▷ Gradient of weights

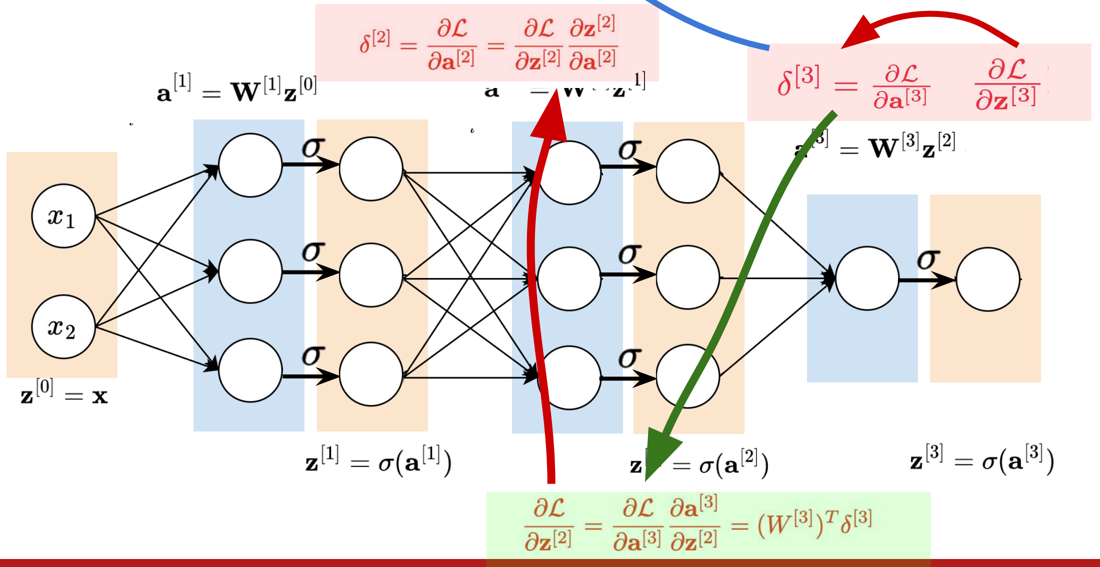
$\quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{b}^{[l]}} = \delta^{[l]}$ ▷ Gradient of biases

$\quad \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{z}^{[l-1]}} = (\mathbf{W}^{[l]})^T \delta^{[l]}$

$\quad \delta^{[l-1]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} \frac{\partial \mathbf{z}^{[l-1]}}{\partial \mathbf{a}^{[l-1]}} = ((\mathbf{W}^{[l]})^T \delta^{[l]}) \odot \sigma^{[l-1]\prime}(\mathbf{a}^{[l-1]})$

end for

Output: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1:L]}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1:L]}}$

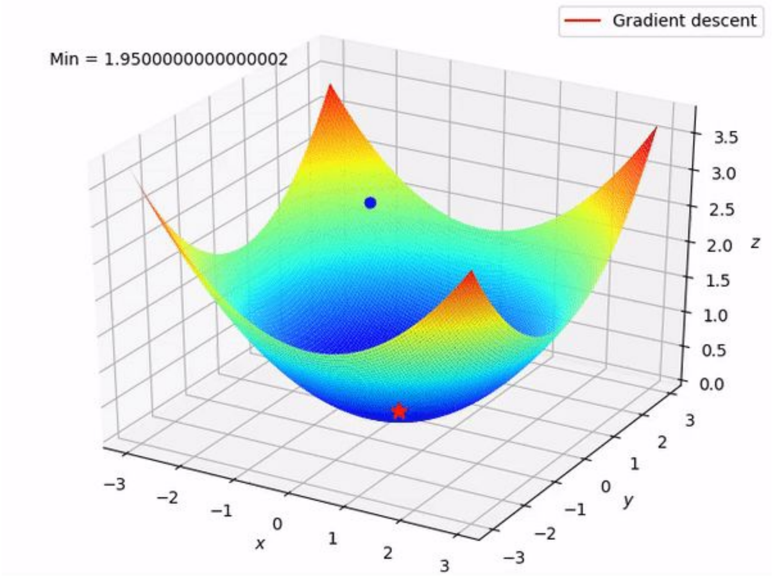


$$\mathcal{L}(\mathbf{z}^{[3]}, \mathbf{y})$$

We can directly compute $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[3]}}$!

Cornell Bowers C·IS

What is Optimization?



In deep learning, optimization methods attempt to find model weights that **minimize the loss function**.

Cornell Bowers CIS

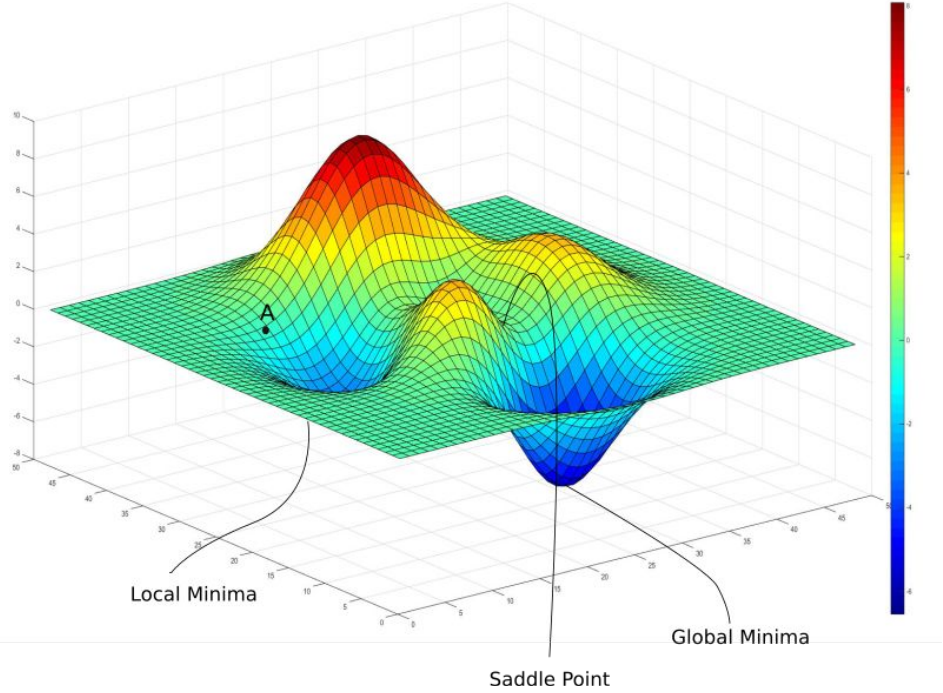
Loss function

Empirical Risk:

$$\mathcal{L}(\mathbf{w}_t) = \frac{1}{n} \sum_{i=1,\ldots,n} \ell(\mathbf{w}_t, \mathbf{x}_i)$$

t : at time step t
$\mathbf{w}_t$: Model weights (parameters) at time t
$\mathbf{x}_i$: The i-th input training data

$\mathcal{L}$: the Loss function (optimization target)
ℓ : per-sample loss



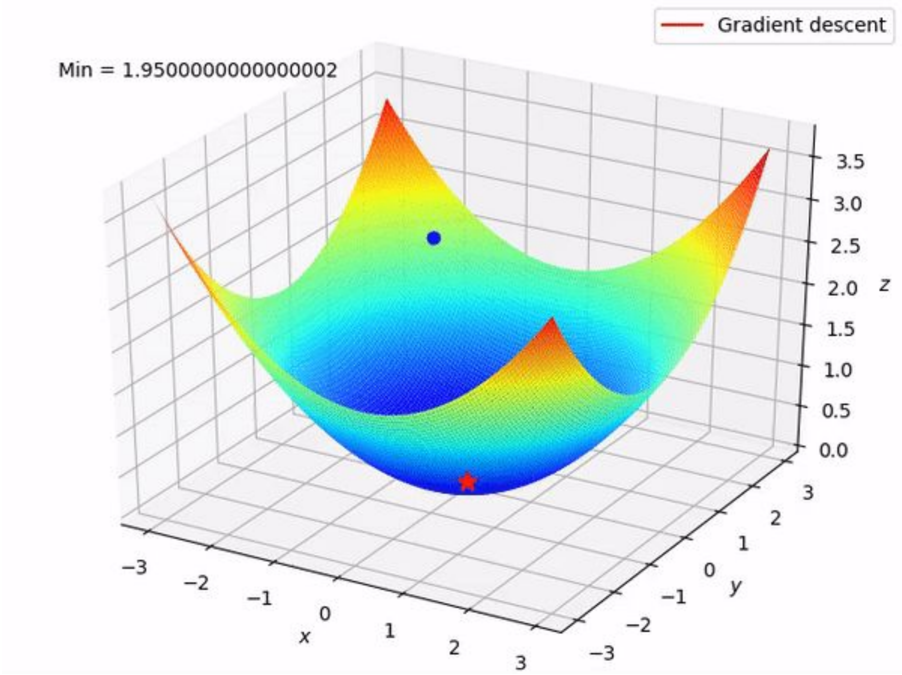
Cornell Bowers C·IS

Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

α: the learning rate

$\nabla \mathcal{L}(\mathbf{w}_t)$: the gradient of Loss w.r.t. $\mathbf{w}_t$



Cornell Bowers C·IS

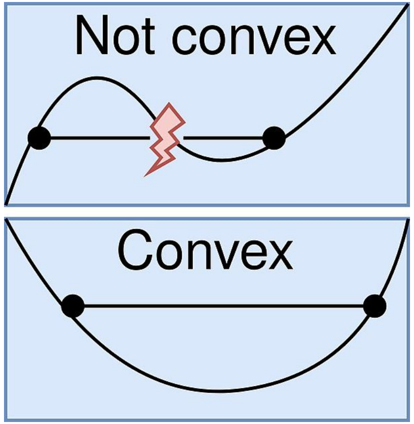
Demo

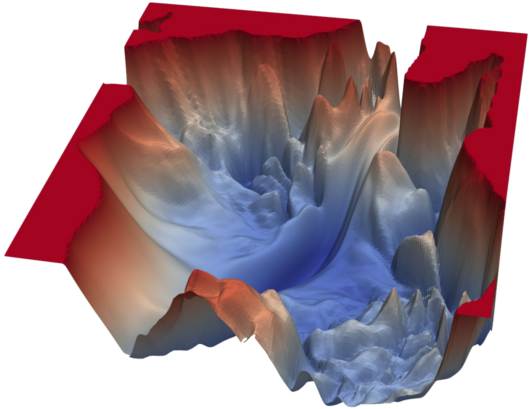
Gradient descent with global minimum

What are some potential problems with gradient descent?

Convexity

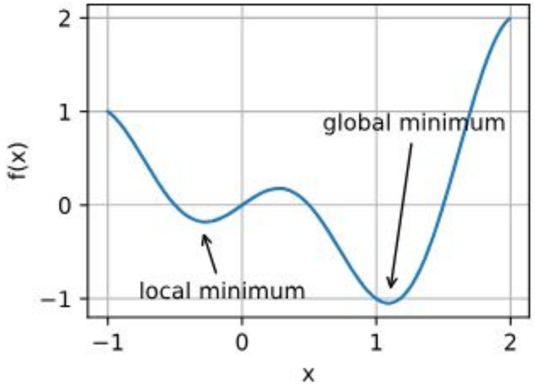
- A function on a graph is **convex** if a line segment drawn through any two points on the line of the function, then it never lies below the curved line segment
- Convexity implies that every local minimum is **global minimum**.
- Neural networks are **not** convex!



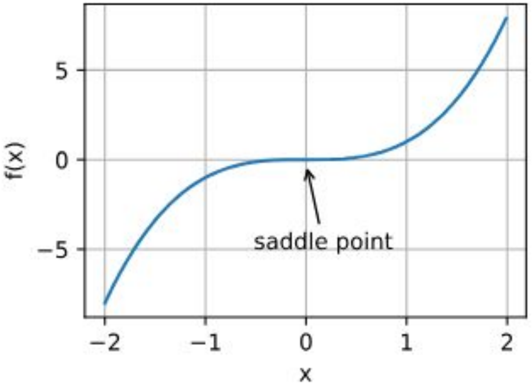


Cornell Bowers C·IS

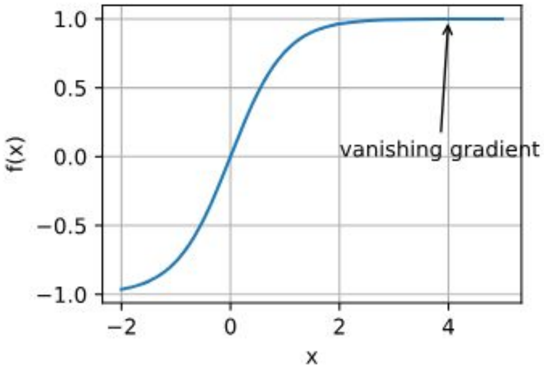
Challenges in Non-Convex Optimization



Local Minima vs. Global Minima



Saddle Points



Vanishing gradient

Demo

Gradient descent with local minimum

Cornell Bowers C·IS

Gradient Descent (GD)

$$\mathcal{L}(\mathbf{w}_t) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}_t, \mathbf{x}_i)$$

$$\nabla\mathcal{L}(\mathbf{w}_t) = \frac{1}{n}\sum_{i=1}^{n} \nabla\ell(\mathbf{w}_t, \mathbf{x}_i)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\nabla\mathcal{L}(\mathbf{w}_t)$$

Full gradient: $\mathcal{O}(n)$ time => ***Too expensive!***

- *Statistically, why don't we use 1 or a few samples from the training dataset to approximate the full gradient?*

Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

Cornell Bowers C·IS

Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

Select **1** example randomly each time

Cornell Bowers C·IS

Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

Select **1** example randomly each time

Per-sample gradient is equivalent to full gradient in expectation!

$$\mathbb{E}[\nabla \ell(\mathbf{w}_t, \mathbf{x}_i)] = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i) = \nabla \mathcal{L}(\mathbf{w}_t)$$

Cornell Bowers CIS

Stochastic Gradient Descent (SGD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

Select **1** example randomly each time

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

Per-sample gradient is equivalent to full gradient in expectation!

$$\mathbb{E}[\nabla \ell(\mathbf{w}_t, \mathbf{x}_i)] = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i) = \nabla \mathcal{L}(\mathbf{w}_t)$$

Cornell Bowers C·IS

Stochastic Gradient Descent (SGD)

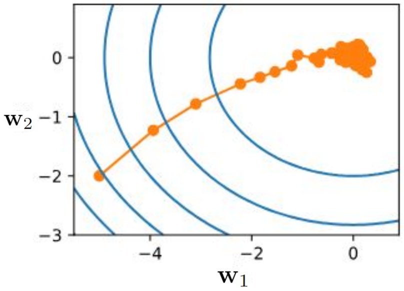
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

Select **1** example randomly each time

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

Trade off convergence*!*

Per-sample gradients not necessarily points to the local minimum, introducing a ***noise ball****…*



Stochastic Gradient Descent (SGD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

↓ Select **1** example randomly each time

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

↓ Select a batch $\mathcal{B}_t$ of examples randomly each time, with *batch size* b

Cornell Bowers C·IS

Minibatch SGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

Select **1** example randomly each time

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

Select a batch $\mathcal{B}_t$ of examples randomly each time, with *batch size* b

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \frac{1}{b} \sum_{i \in \mathcal{B}_t} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

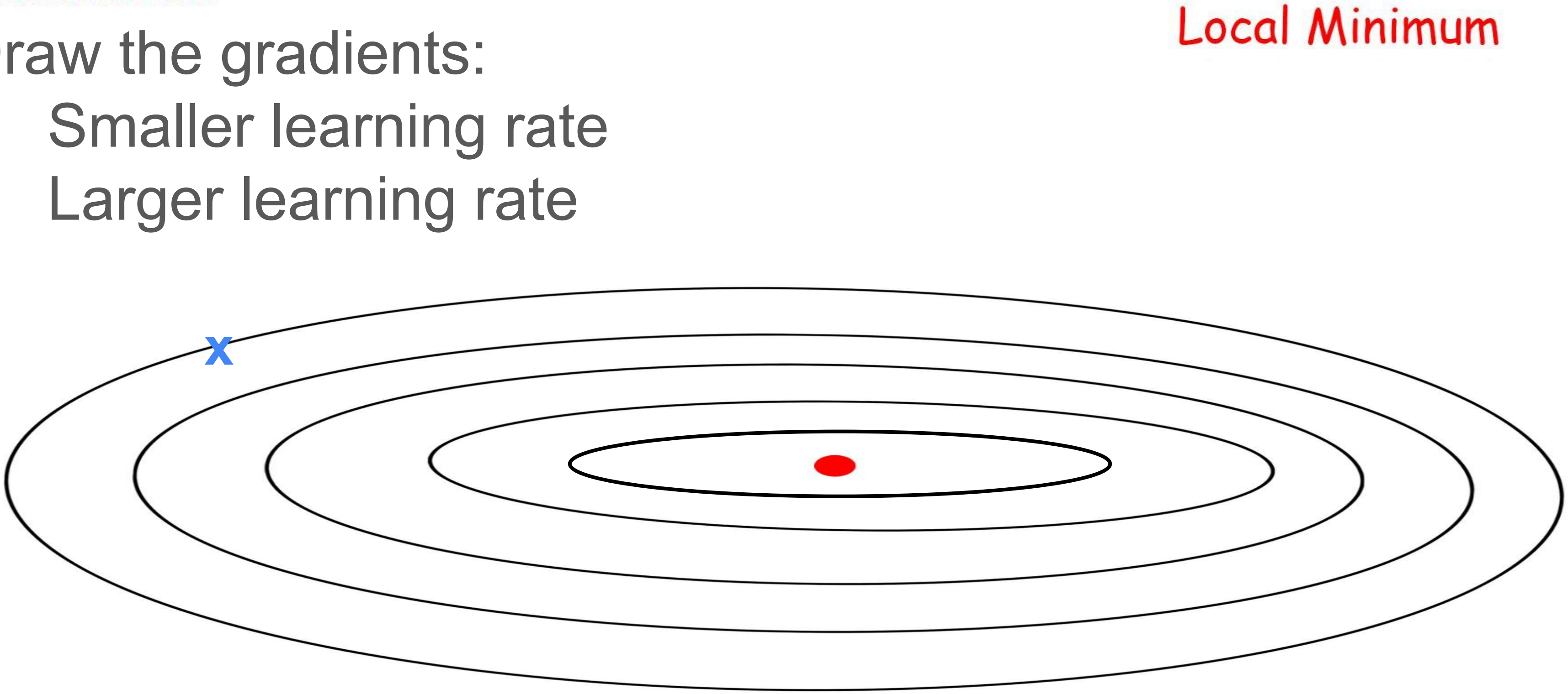
Cornell Bowers C·IS

Let's look at an example!

Cornell Bowers C·IS

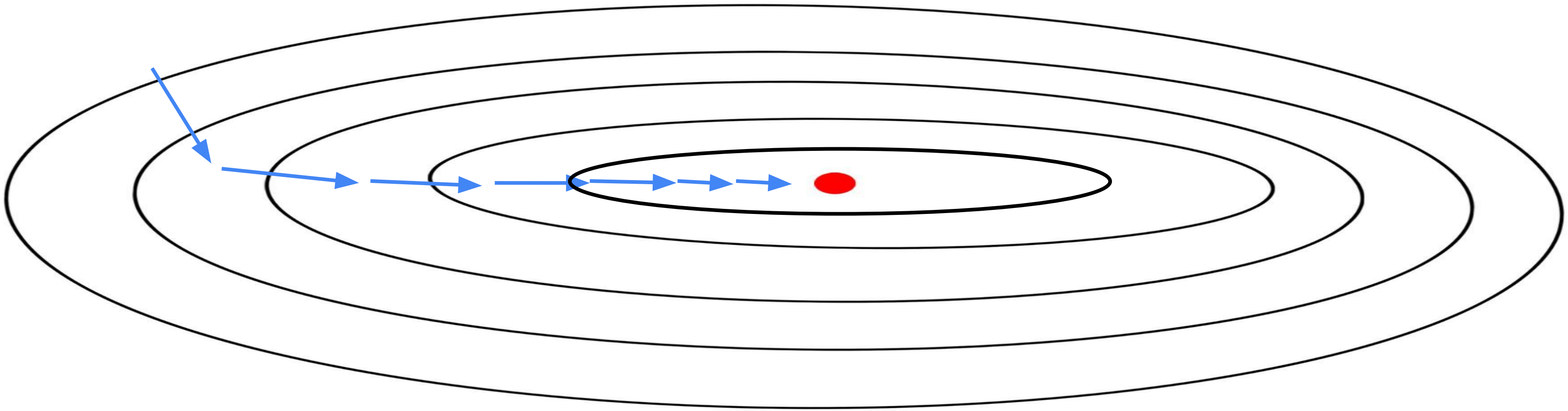
Draw the gradients:

- Smaller learning rate
- Larger learning rate



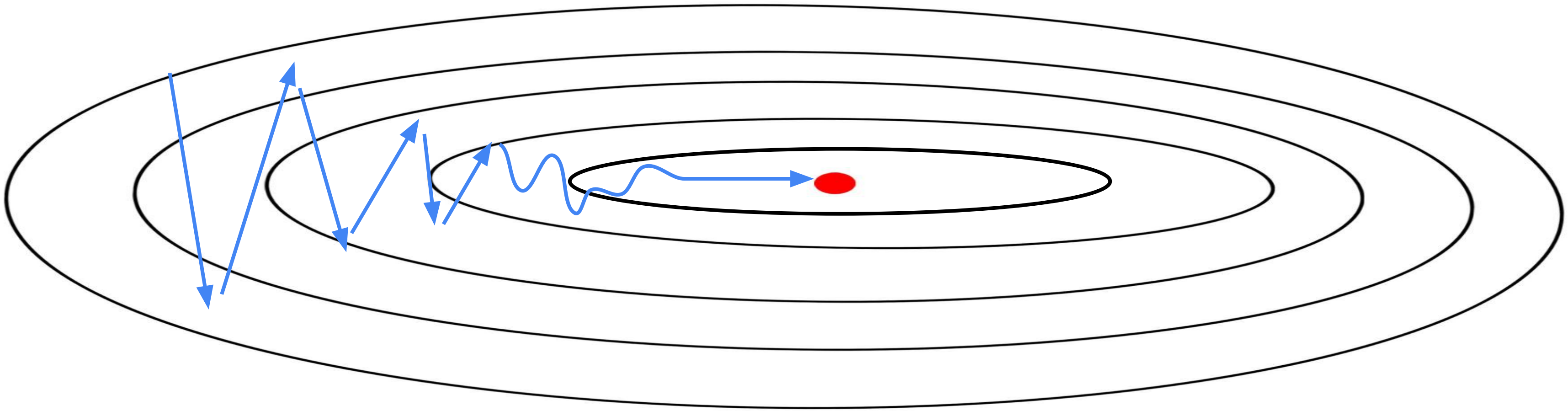
Cornell Bowers CIS

Local Minimum



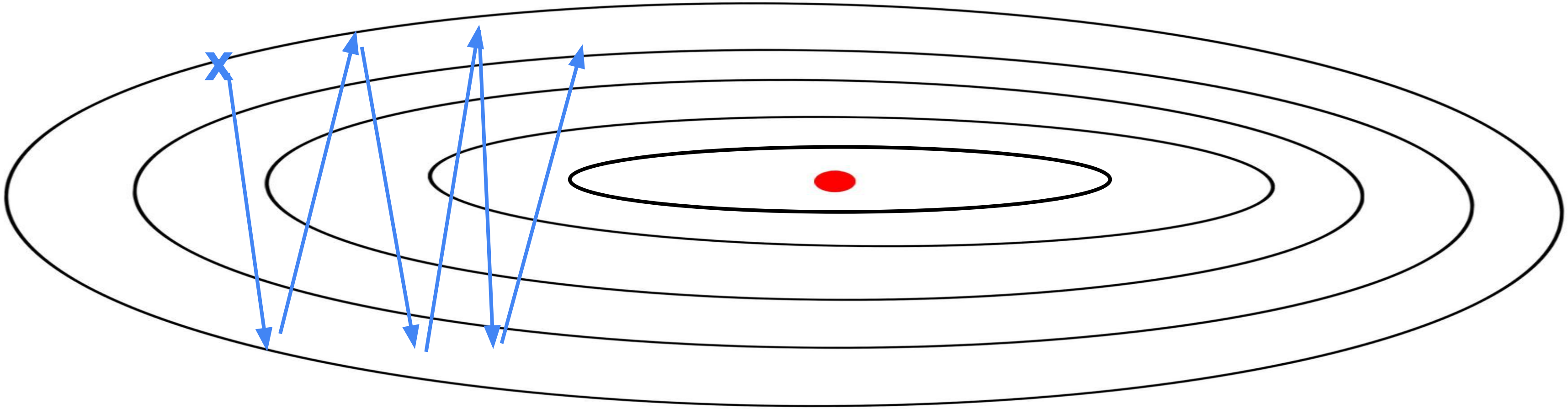
Cornell Bowers C·IS

Local Minimum



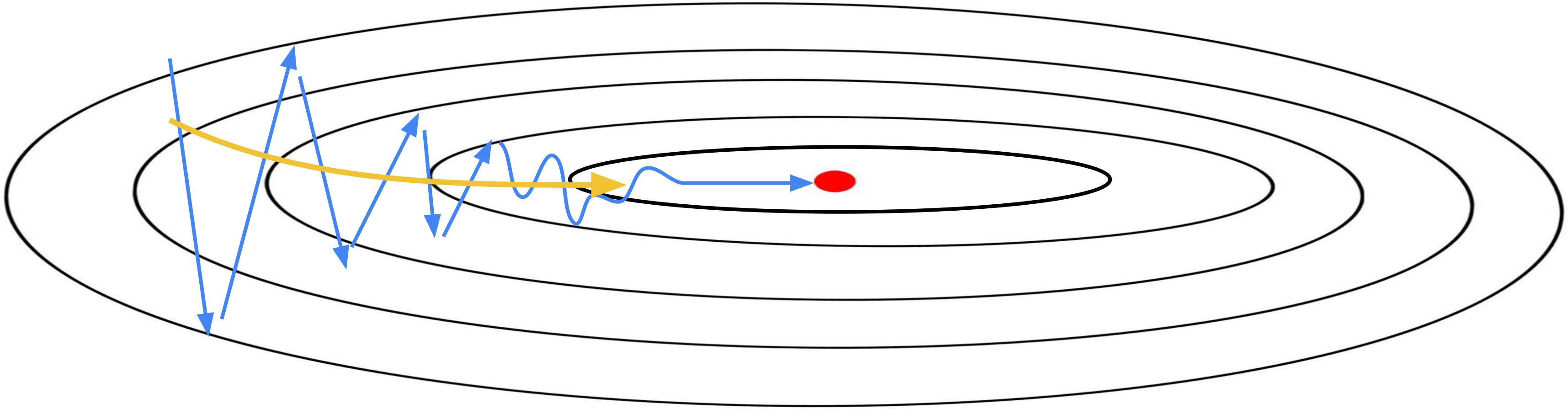
Cornell Bowers C·IS

Local Minimum



Cornell Bowers C·IS

Local Minimum
Minibatch SGD
Momentum



Cornell Bowers C·IS

SGD with Momentum (Polyak, 1964)

*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*

Cornell Bowers C·IS

SGD with Momentum (Polyak, 1964)

*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*

SGD with Momentum (Polyak, 1964)

Compute an ***Exponentially Weighted Moving Average (EWMA)*** *of the gradients as* ***momentum*** *and use that to update the weight instead.*

SGD Update Rule

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

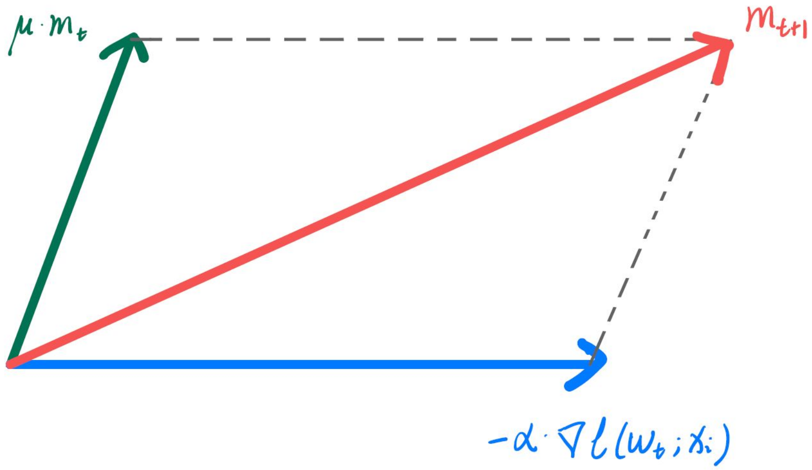
→

$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

where $\mu \in [0, 1]$ is the momentum coefficient.

SGD with Momentum (Polyak, 1964)

*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*



$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

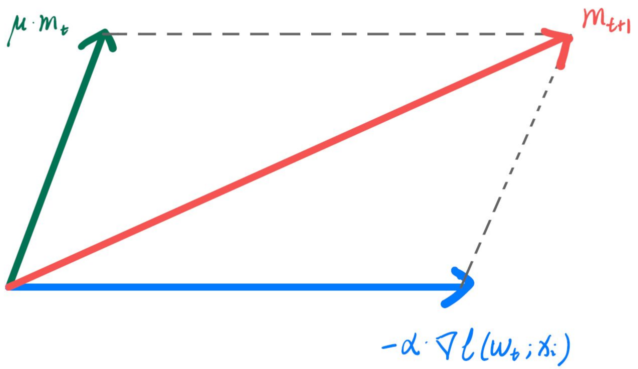
where $\mu \in [0, 1]$ is the momentum coefficient.

Cornell Bowers C·IS

SGD with Momentum

*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*

$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

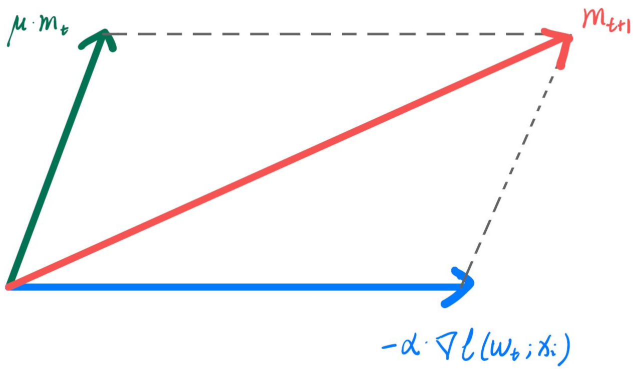


Cornell Bowers C·IS

SGD with Momentum

*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*

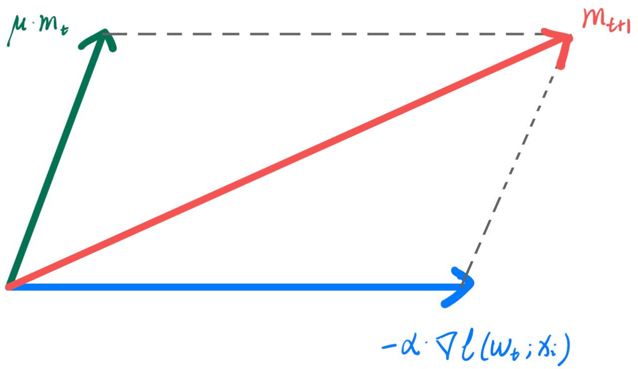
$$\mathbf{g}_t = \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$
$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \mathbf{g}_t$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$



SGD with Momentum

*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*

$$\mathbf{g}_t = \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$
$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \mathbf{g}_t$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu \mathbf{m}_t - \alpha \mathbf{g}_t$$

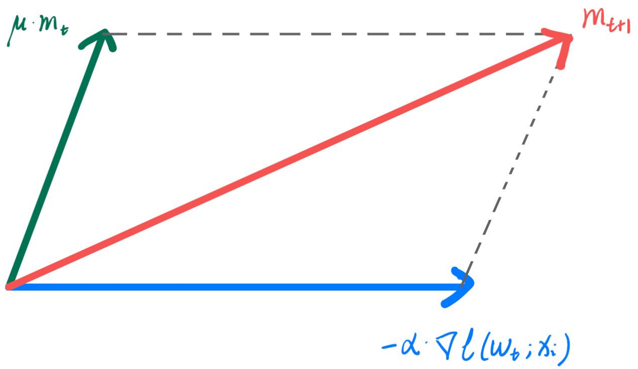


Cornell Bowers C·IS

SGD with Momentum

*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*

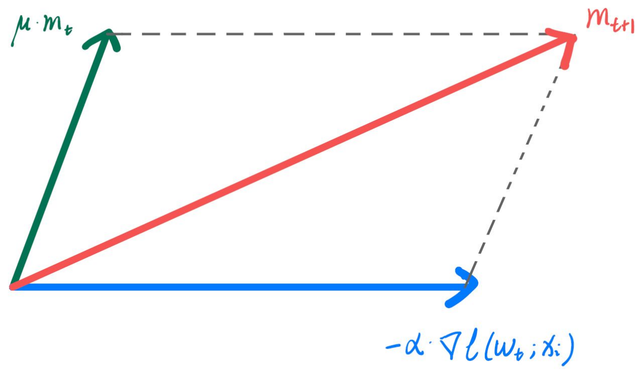
$$
\begin{aligned}
\mathbf{g}_t &= \nabla l(\mathbf{w}_t; \mathbf{x}_i) \\
\mathbf{m}_{t+1} &= \mu \mathbf{m}_t - \alpha \mathbf{g}_t \\
\mathbf{w}_{t+1} &= \mathbf{w}_t + \mu \mathbf{m}_t - \alpha \mathbf{g}_t \\
&= \mathbf{w}_t + \mu(\mu \mathbf{m}_{t-1} - \alpha \mathbf{g}_{t-1}) - \alpha \mathbf{g}_t
\end{aligned}
$$



SGD with Momentum

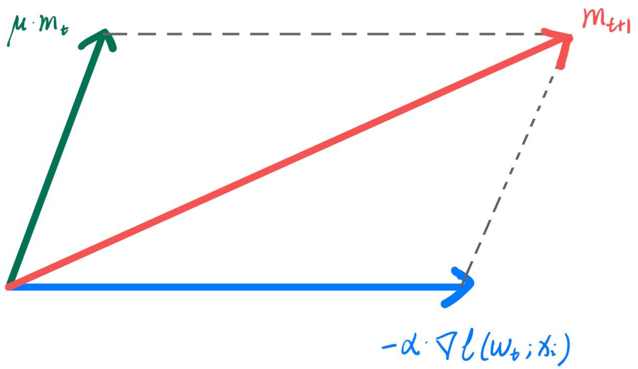
*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*

$$\begin{aligned}
\mathbf{g}_t &= \nabla l(\mathbf{w}_t; \mathbf{x}_i) \\
\mathbf{m}_{t+1} &= \mu \mathbf{m}_t - \alpha \mathbf{g}_t \\
\mathbf{w}_{t+1} &= \mathbf{w}_t + \mu \mathbf{m}_t - \alpha \mathbf{g}_t \\
&= \mathbf{w}_t + \mu(\mu \mathbf{m}_{t-1} - \alpha \mathbf{g}_{t-1}) - \alpha \mathbf{g}_t \\
&= \mathbf{w}_t + \mu(\mu(\mu \mathbf{m}_{t-2} - \alpha \mathbf{g}_{t-2}) - \alpha \mathbf{g}_{t-1}) - \alpha \mathbf{g}_t \\
&= \mathbf{w}_t - \alpha \mathbf{g}_t - \mu \alpha \mathbf{g}_{t-1} - \mu^2 \alpha \mathbf{g}_{t-2} - \mu^3 \alpha \mathbf{g}_{t-3} - \ldots
\end{aligned}$$



SGD with Momentum

*Compute an **Exponentially Weighted Moving Average (EWMA)** of the gradients as **momentum** and use that to update the weight instead.*

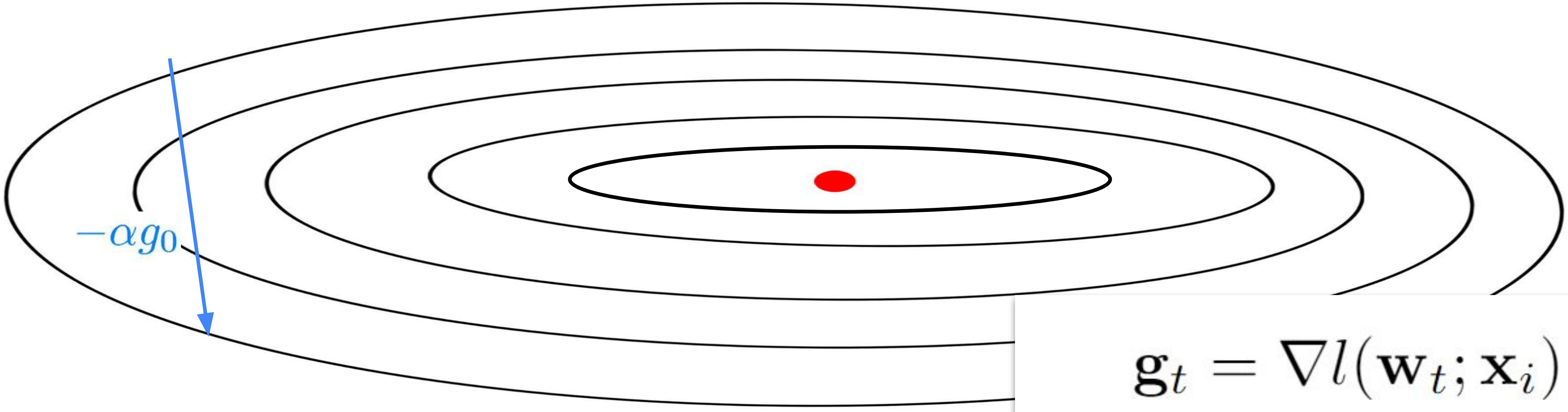


$$
\begin{aligned}
\mathbf{g}_t &= \nabla l(\mathbf{w}_t; \mathbf{x}_i) \\
\mathbf{m}_{t+1} &= \mu \mathbf{m}_t - \alpha \mathbf{g}_t \\
\mathbf{w}_{t+1} &= \mathbf{w}_t + \mu \mathbf{m}_t - \alpha \mathbf{g}_t \\
&= \mathbf{w}_t + \mu(\mu \mathbf{m}_{t-1} - \alpha \mathbf{g}_{t-1}) - \alpha \mathbf{g}_t \\
&= \mathbf{w}_t + \mu(\mu(\mu \mathbf{m}_{t-2} - \alpha \mathbf{g}_{t-2}) - \alpha \mathbf{g}_{t-1}) - \alpha \mathbf{g}_t \\
&= \mathbf{w}_t - \alpha \mathbf{g}_t - \mu \alpha \mathbf{g}_{t-1} - \mu^2 \alpha \mathbf{g}_{t-2} - \mu^3 \alpha \mathbf{g}_{t-3} - \dots \\
&= \mathbf{w}_t - \alpha \sum_{i=0}^{t} \mu^i \mathbf{g}_{t-i}
\end{aligned}
$$

Cornell Bowers C·IS

Local Minimum

Minibatch SGD



$$\mathbf{g}_t = \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$

$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \mathbf{g}_t$$

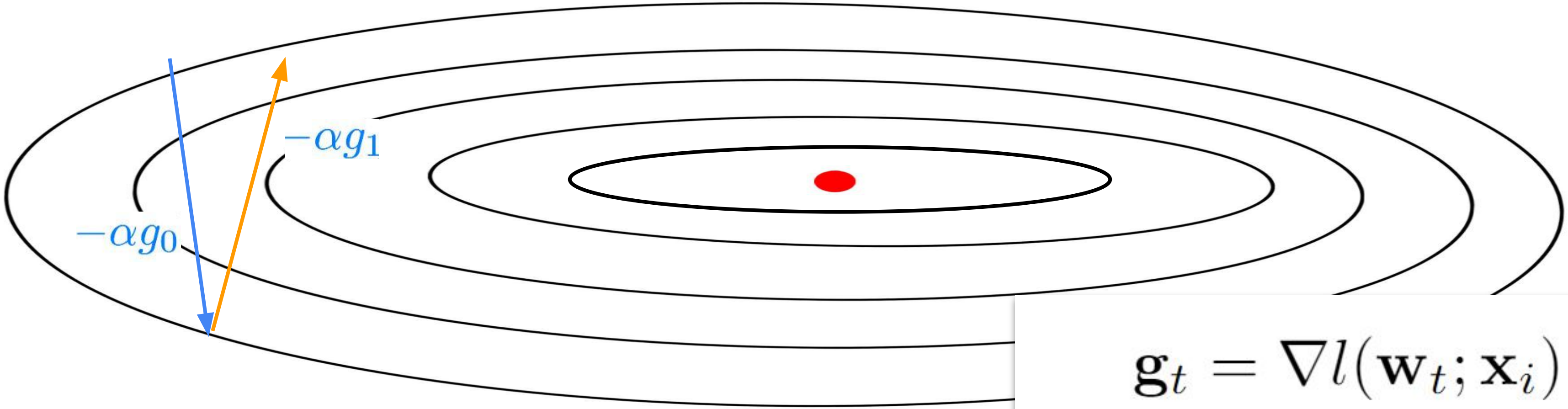
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

Cornell Bowers C·IS

Local Minimum

Minibatch SGD

$$\mathbf{m}_2 = \mu \mathbf{m}_1 - \alpha \mathbf{g}_1$$



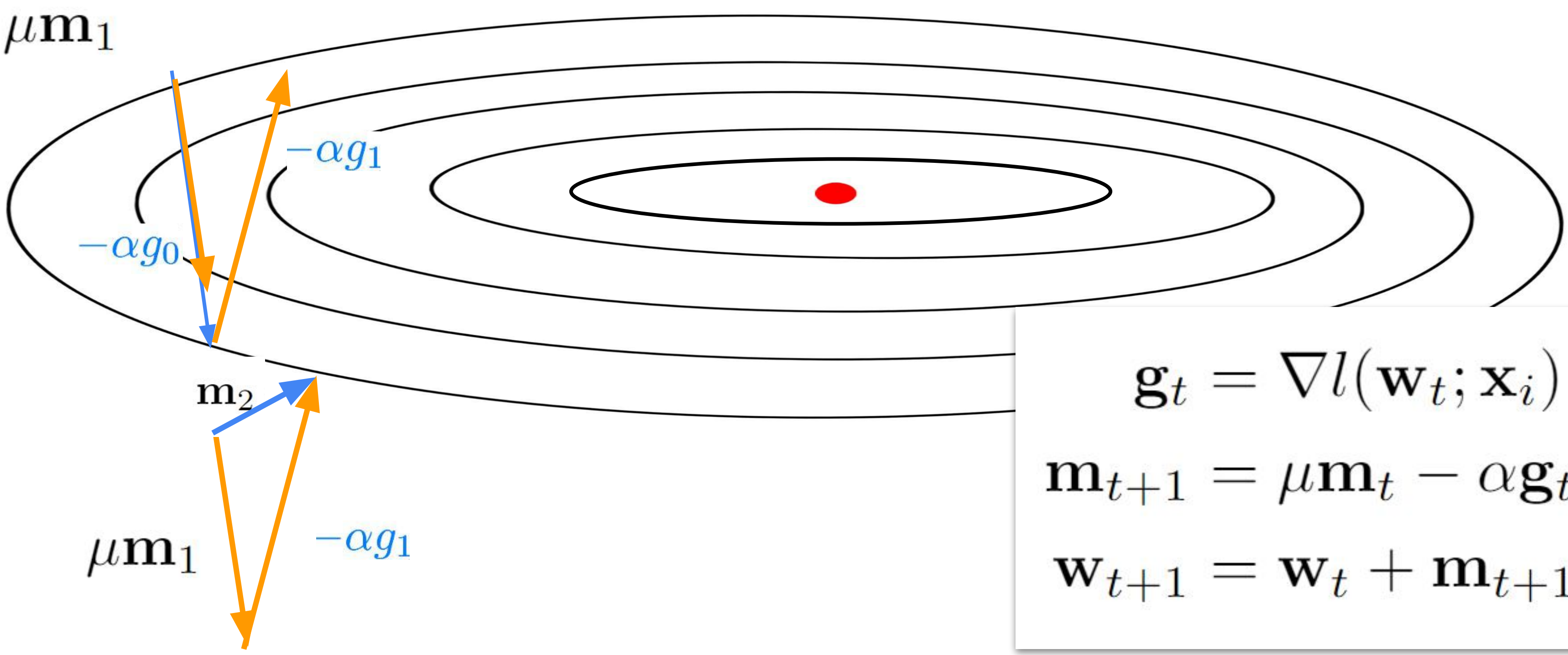
$$\mathbf{g}_t = \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$

$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \mathbf{g}_t$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

Local Minimum
Minibatch SGD

$$\mathbf{m}_2 = \mu\mathbf{m}_1 - \alpha\mathbf{g}_1$$



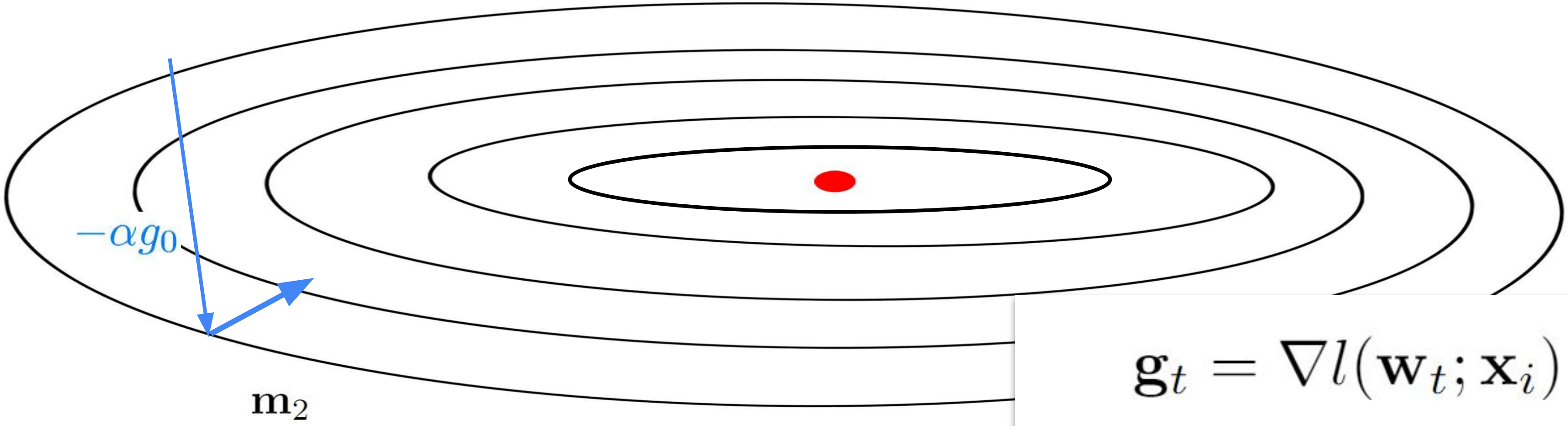
$$\mathbf{g}_t = \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$

$$\mathbf{m}_{t+1} = \mu\mathbf{m}_t - \alpha\mathbf{g}_t$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

Local Minimum
Minibatch SGD

$$\mathbf{m}_2 = \mu \mathbf{m}_1 - \alpha \mathbf{g}_1$$



$$\mathbf{g}_t = \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$

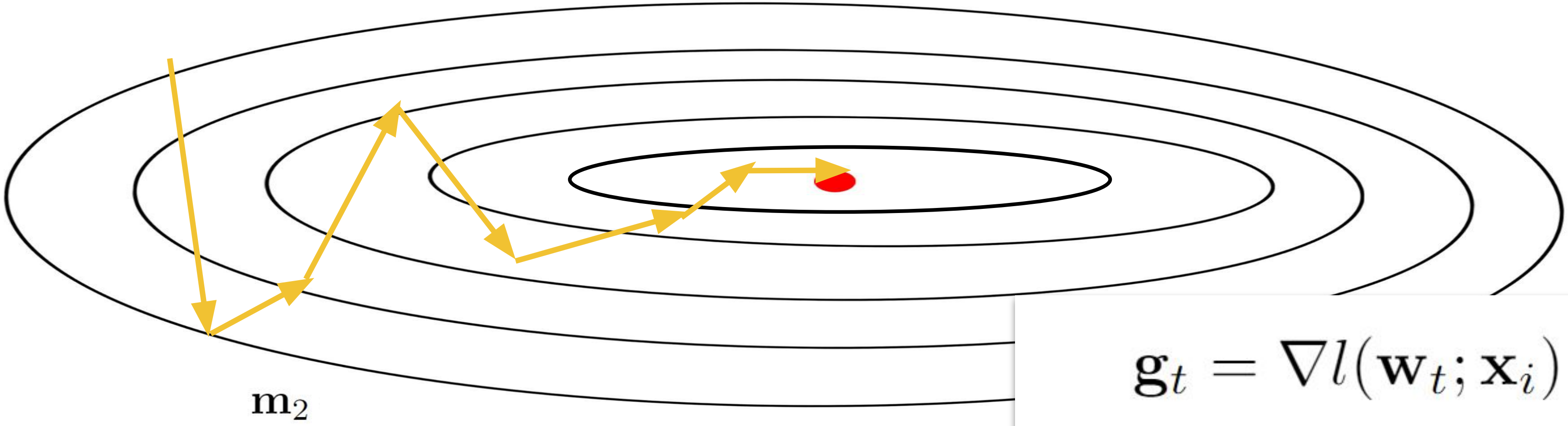
$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \mathbf{g}_t$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

Cornell Bowers CIS

$$\mathbf{m}_2 = \mu \mathbf{m}_1 - \alpha \mathbf{g}_1$$

Local Minimum
Minibatch SGD
Momentum



$$\mathbf{g}_t = \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$

$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \mathbf{g}_t$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

Cornell Bowers C·IS

Local Minimum
Minibatch SGD
Momentum

$$\mathbf{m}_2 = \mu \mathbf{m}_1 - \alpha \mathbf{g}_1$$

Momentum converges almost always faster than standard SGD!

$$\mathbf{g}_t = \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$

$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \mathbf{g}_t$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

Cornell Bowers C·IS

Quick Recap

Gradient Descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

Stochastic Gradient Descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

Minibatch SGD

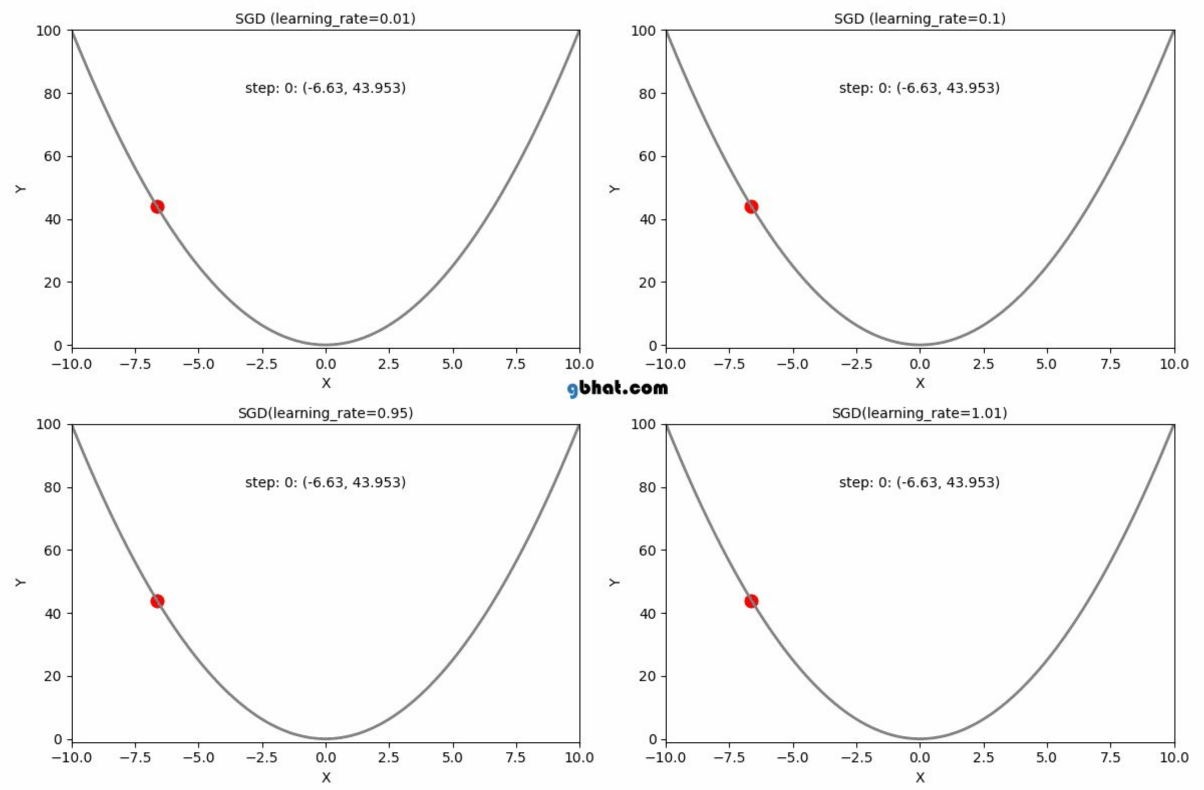
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \frac{1}{b} \sum_{i \in \mathcal{B}_t} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i)$$

SGD w. Momentum

$$m_{t+1} = \mu m_t - \alpha \nabla l(w_t; x_i)$$

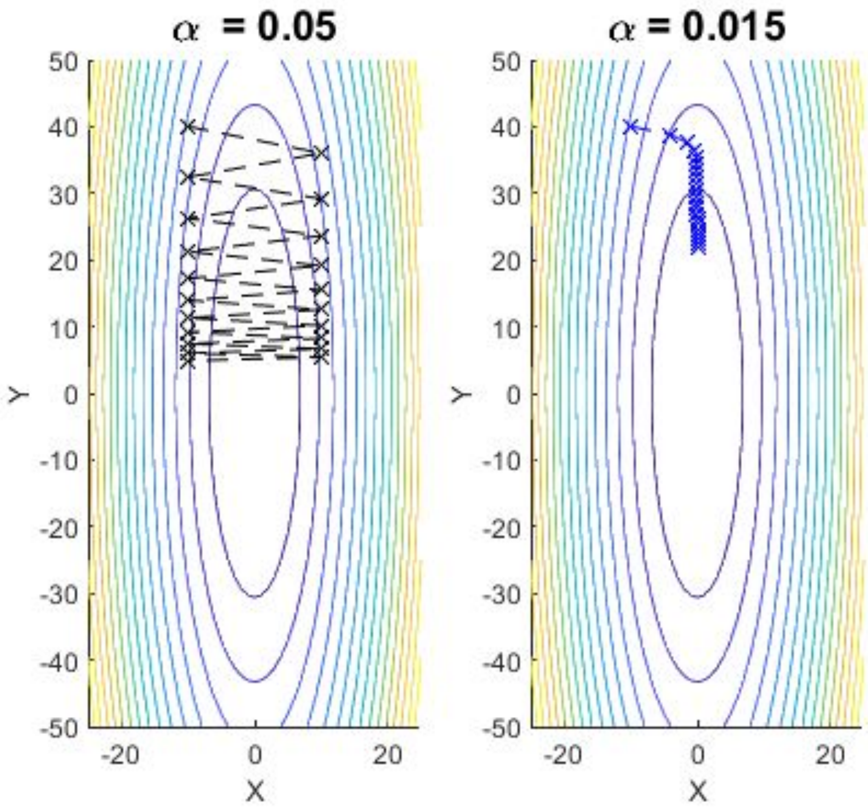
$$w_{t+1} = w_t + m_{t+1}$$

Importance of Learning Rate

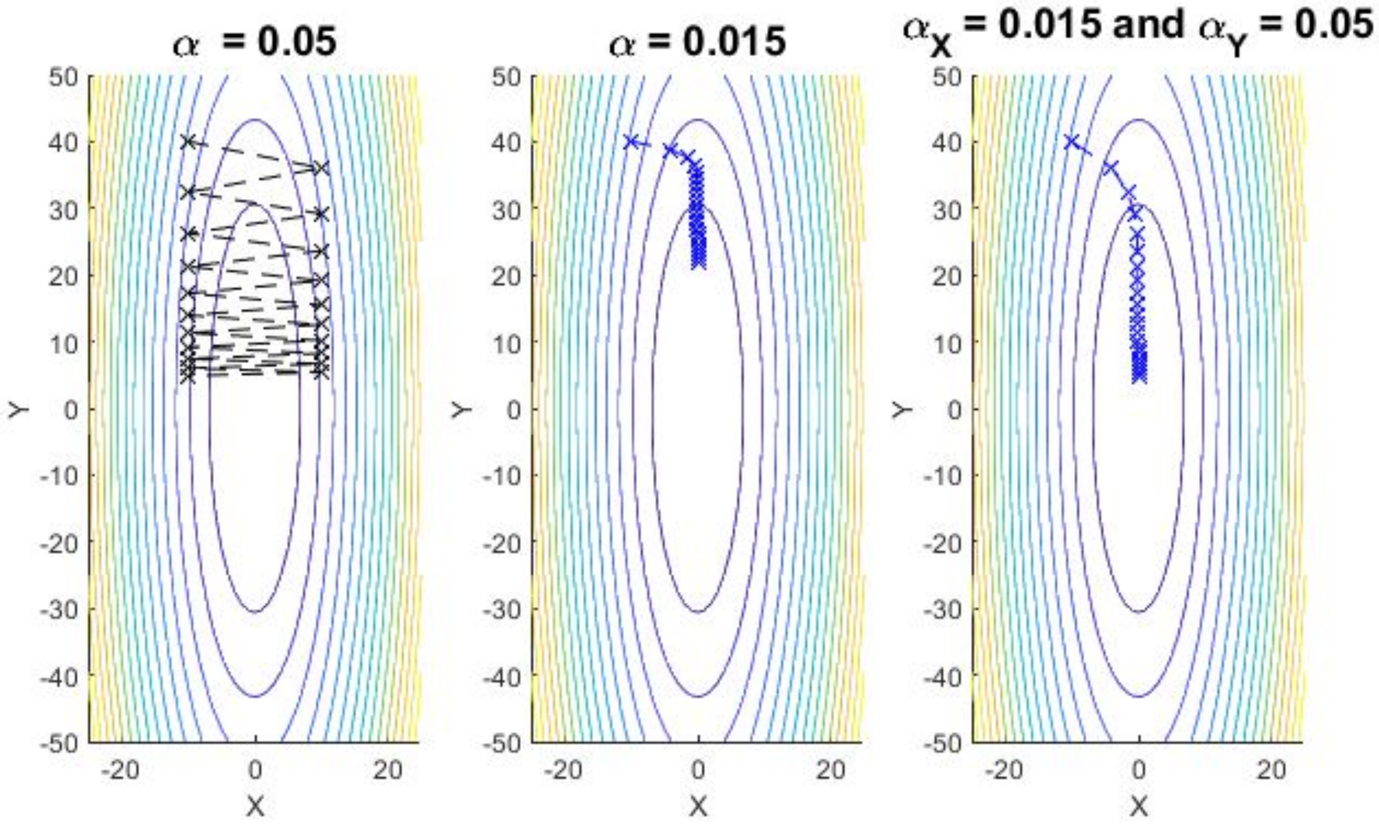


Cornell Bowers C·IS

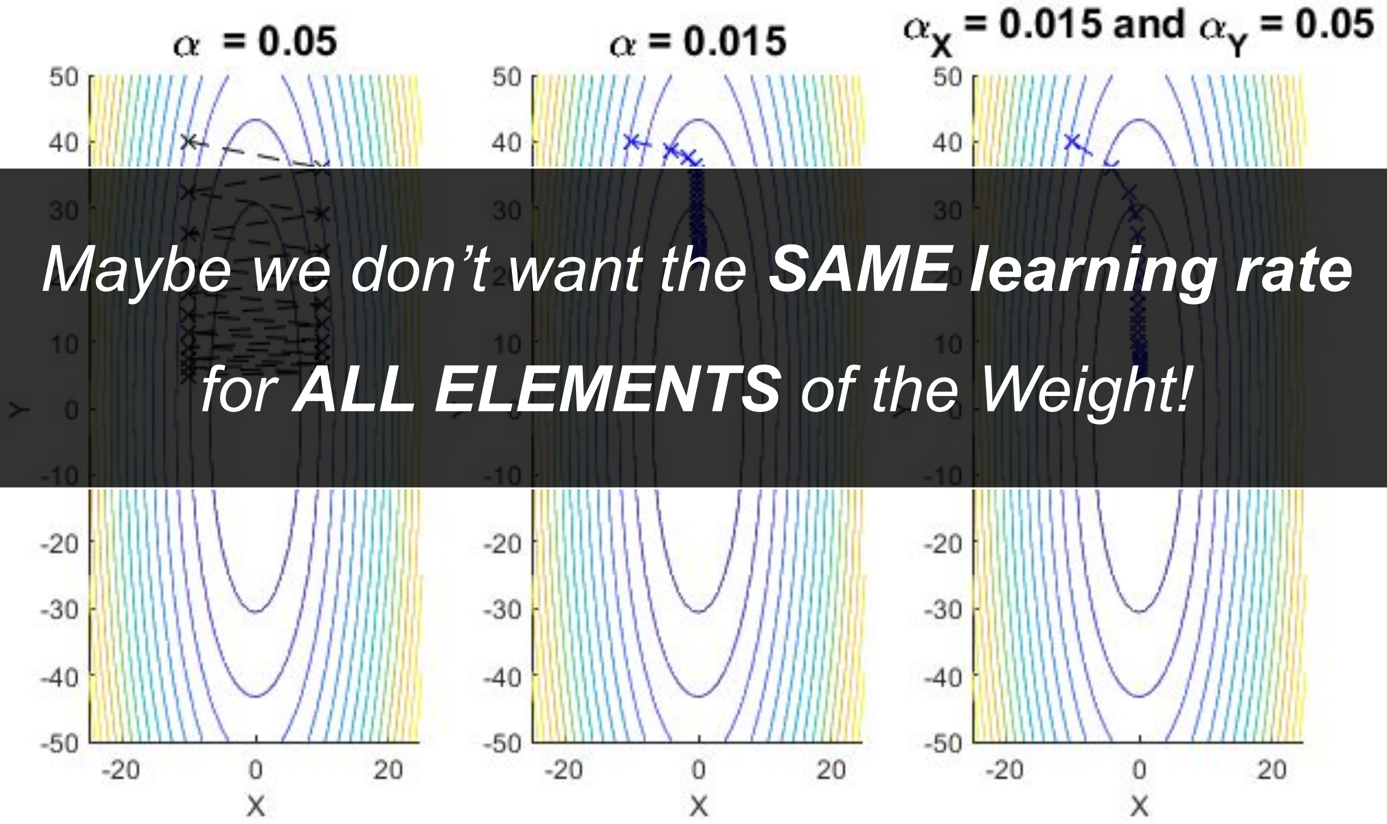
Another example



Another example



Adaptive Learning Rate



Maybe we don't want the ***SAME learning rate***

for ***ALL ELEMENTS*** *of the Weight!*

Cornell Bowers C·IS

Adaptive Optimizers

Different Learning Rate *for* ***each element*** *of the* ***Model Weights!***

AdaGrad (Duchi et al. 2011)

More updates -> more decay

- Handle sparse gradients well
  - *Sparse: The vector has 0 in most of the entries*

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

SGD

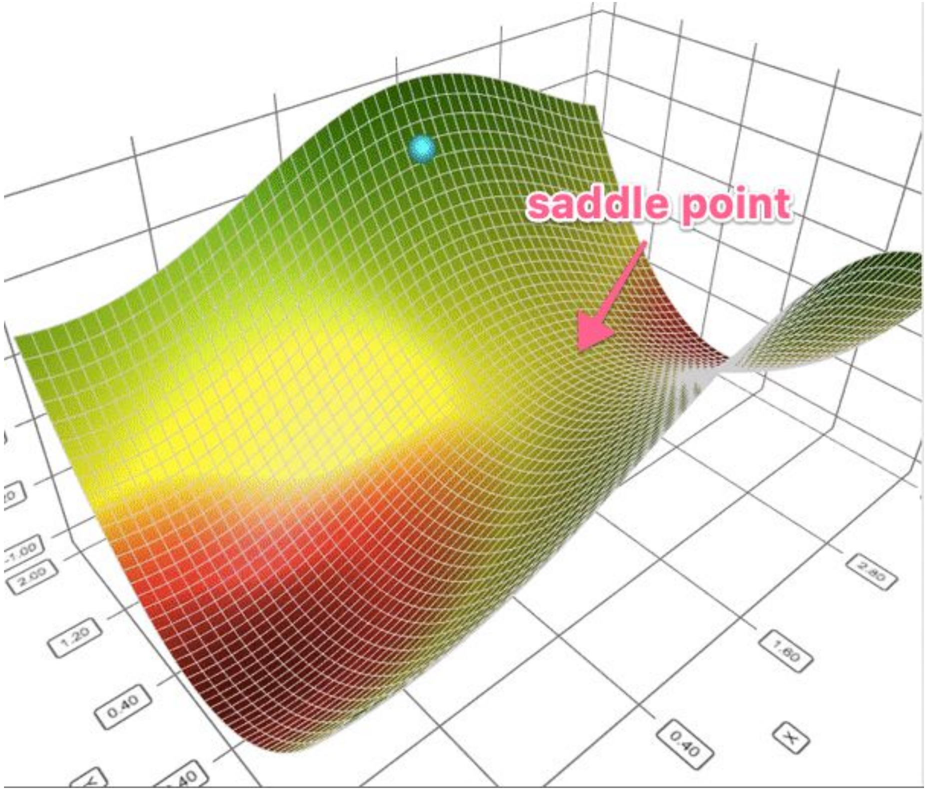
Element-wise product

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{g}_t^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

Adagrad

Cornell Bowers C·IS



Gradient Descent
AdaGrad

AdaGrad (Duchi et al. 2011)

More updates → more decay

- Handle sparse gradients well
  - *Sparse: The vector has 0 in most of the entries*

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

SGD

Element-wise product

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{g}_t^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

Adagrad

Exercise:
What's could be wrong with this optimizator?
(What would happen to the denominator.)

Cornell Bowers C·IS

AdaGrad (Duchi et al. 2011)

More updates → more decay

- Handle sparse gradients well
  - *Sparse: The vector has 0 in most of the entries*

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \mathcal{L}(\mathbf{w}_t)$$

SGD

Element-wise product

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{g}_t^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

Adagrad

***Issue**: decays too aggressively!*

RMSProp (Graves, 2013)

Keep an **exponential moving average** of the squared gradient for each element

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{g}_t^2$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

Adagrad

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1-\beta)\mathbf{g}_t^2$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

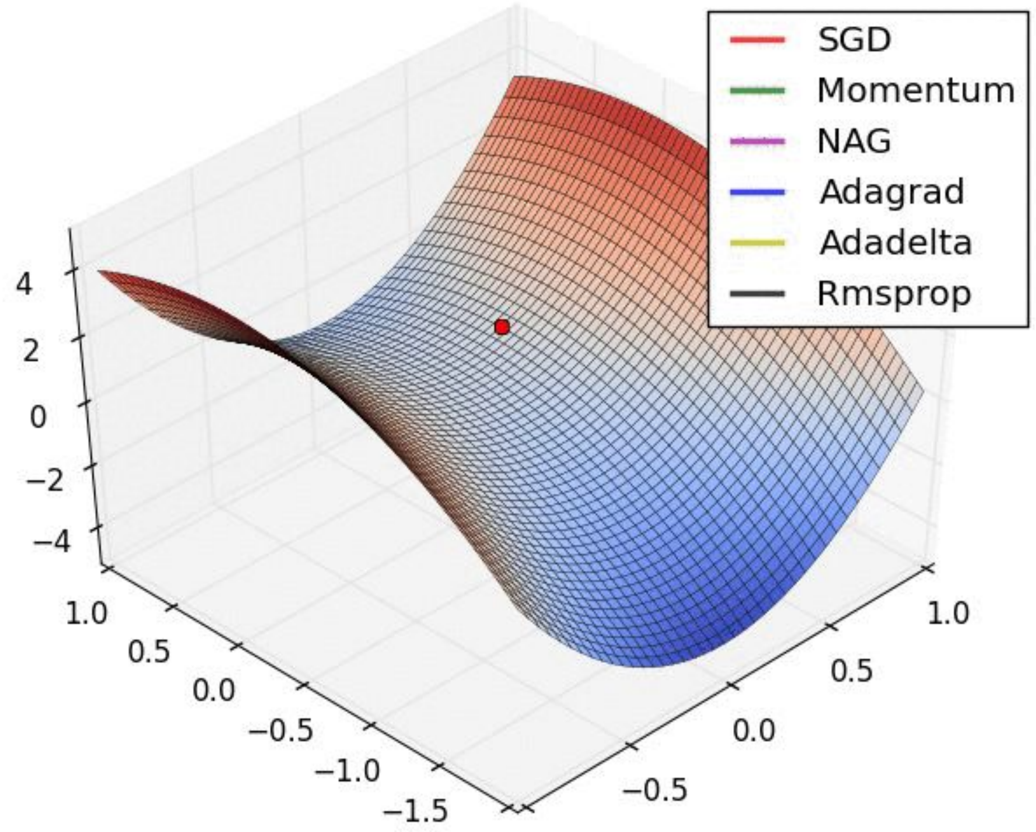
RmsProp

where $\beta \in [0,1]$ the exponential moving average constant.

Cornell Bowers C·IS

Demo

Adagrad & RMSprop



$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

Momentum

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1-\beta)\mathbf{g}_t^2$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

RMSProp

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1-\beta)\mathbf{g}_t^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

RMSProp

ADAM
(**Ada**ptive **M**oment Estimate)

Cornell Bowers C·IS

$$\mathbf{v}_{t+1} = \beta_2 \mathbf{v}_t + (1 - \beta_2)\mathbf{g}_t^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\widehat{\mathbf{v}}_{t+1} + \epsilon}} \odot \widehat{\mathbf{m}}_{t+1}$$

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1 - \beta)\mathbf{g}_t^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

RMSProp

ADAM
(**Ada**ptive **M**oment Estimate)

$$\mathbf{m}_{t+1} = \mu \mathbf{m}_t - \alpha \nabla l(\mathbf{w}_t; \mathbf{x}_i)$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{m}_{t+1}$$

Momentum

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1-\beta)\mathbf{g}_t^2$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\mathbf{v}_{t+1} + \epsilon}} \odot \mathbf{g}_t$$

RMSProp

$$\mathbf{m}_{t+1} = \beta_1 \mathbf{m}_t + (1-\beta_1)\mathbf{g}_t$$
$$\mathbf{v}_{t+1} = \beta_2 \mathbf{v}_t + (1-\beta_2)\mathbf{g}_t^2$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\widehat{\mathbf{v}}_{t+1} + \epsilon}} \odot \widehat{\mathbf{m}}_{t+1}$$

ADAM
(**Ada**ptive **M**oment Estimate)

$\mathbb{E}[\widehat{\mathbf{m}}_{t+1}]$

$\mathbb{E}[\widehat{\mathbf{v}}_{t+1}]$

$$\mathbf{m}_{t+1} = \beta_1 \mathbf{m}_t + (1-\beta_1)\mathbf{g}_t$$

$$\mathbf{v}_{t+1} = \beta_2 \mathbf{v}_t + (1-\beta_2)\mathbf{g}_t^2$$

$$\widehat{\mathbf{m}}_{t+1} = \frac{\mathbf{m}_{t+1}}{1-\beta_1^{t+1}}$$

$$\widehat{\mathbf{v}}_{t+1} = \frac{\mathbf{v}_{t+1}}{1-\beta_2^{t+1}}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\alpha}{\sqrt{\widehat{\mathbf{v}}_{t+1}+\epsilon}} \odot \widehat{\mathbf{m}}_{t+1}$$

ADAM

(**Ada**ptive **M**oment Estimate)

Cornell Bowers C·IS

Optimizers Recap

- Gradient Descent
  - *Vanilla, costly, but for best convergence rate*
- Stochastic Gradient Descent
  - *Simple, lightweight*
- **Mini-batch SGD**
  - *balanced between SGD and GD*
  - ***1st choice for small, simple models***

- SGD w. Momentum
  - *Faster, capable to jump out local minimum*

- AdaGrad
- RMSProp
- **ADAM**
  - **JUST USE ADAM IF YOU DON'T KNOW WHAT TO USE IN DEEP LEARNING**