Tips for training

Tips for Training

1. Batch size

Optimization with Batch

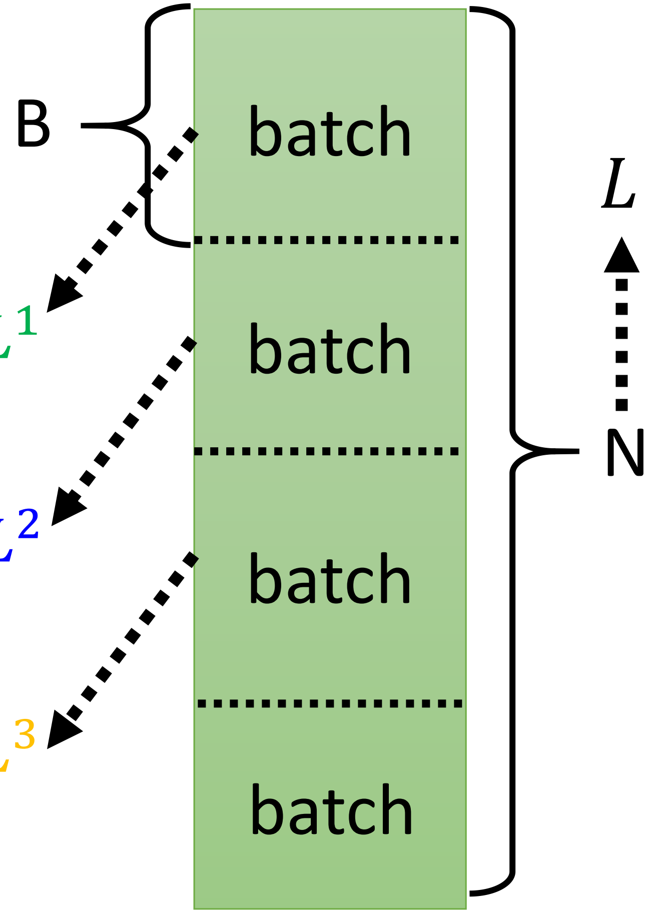
$$\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$$

- ➢ (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- ➢ Compute gradient $\boldsymbol{g}^{\mathbf{0}} = \nabla L^1(\boldsymbol{\theta}^0)$ L^1

  update $\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}^{\mathbf{0}}$
- ➢ Compute gradient $\boldsymbol{g}^{\mathbf{1}} = \nabla L^2(\boldsymbol{\theta}^1)$ L^2

  update $\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}^{\mathbf{1}}$
- ➢ Compute gradient $\boldsymbol{g}^{\mathbf{3}} = \nabla L^3(\boldsymbol{\theta}^2)$ L^3

  update $\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}^{\mathbf{3}}$



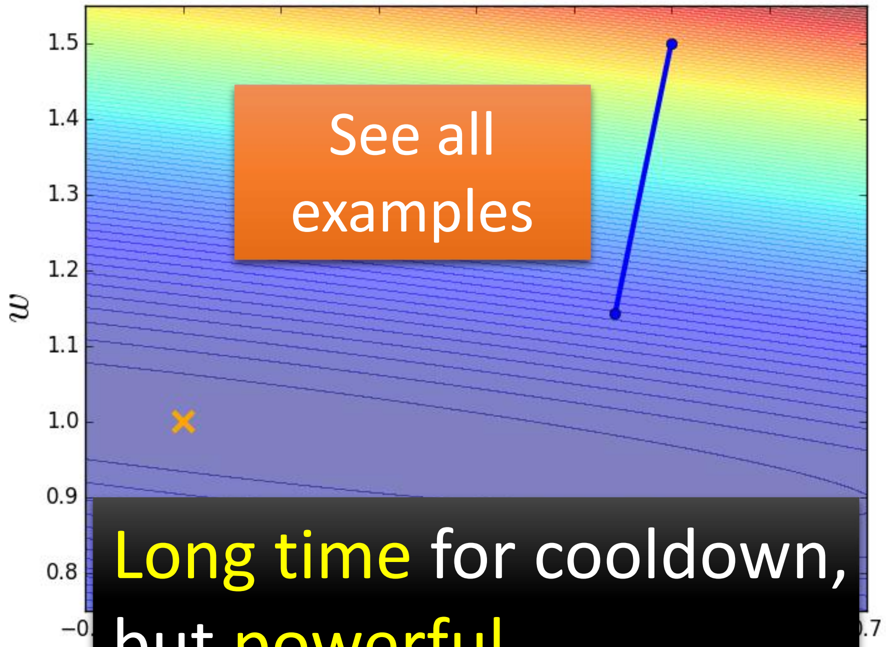
1 **epoch** = see all the batches once ➡ **Shuffle** after each epoch

Small Batch v.s. Large Batch

Consider 20 examples (N=20)

Batch size = N (Full batch)

Update after seeing all the 20 examples

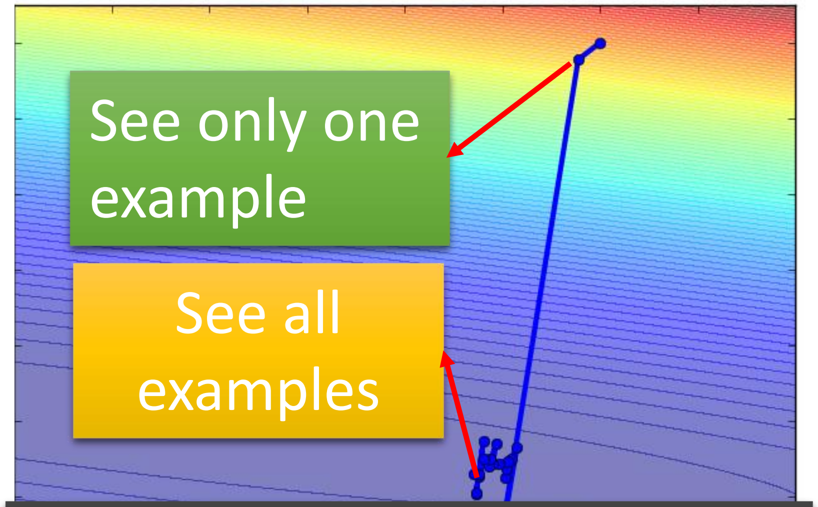


Long time for cooldown, but powerful

Batch size = 1

Update for each example

Update 20 times in an epoch



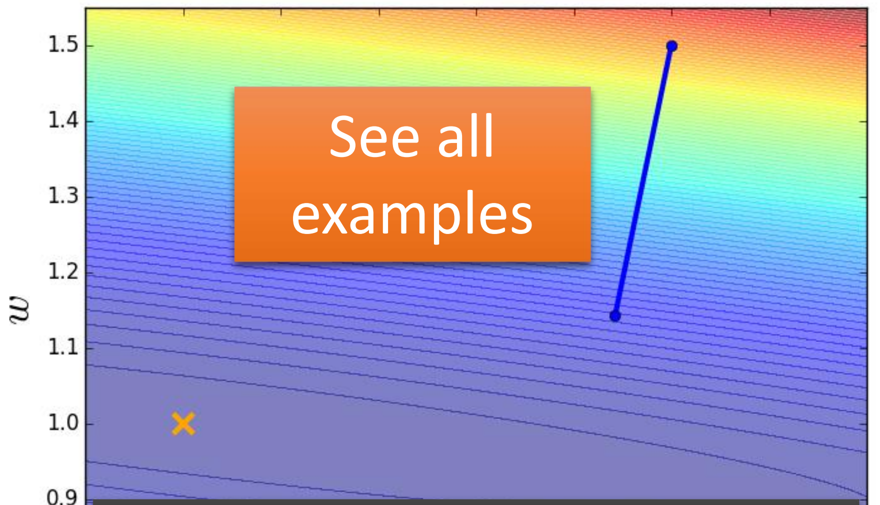
Short time for cooldown, but noisy

Small Batch v.s. Large Batch

Consider 20 examples (N=20)

Batch size = N (Full Batch)

Update after seeing all the 20 examples

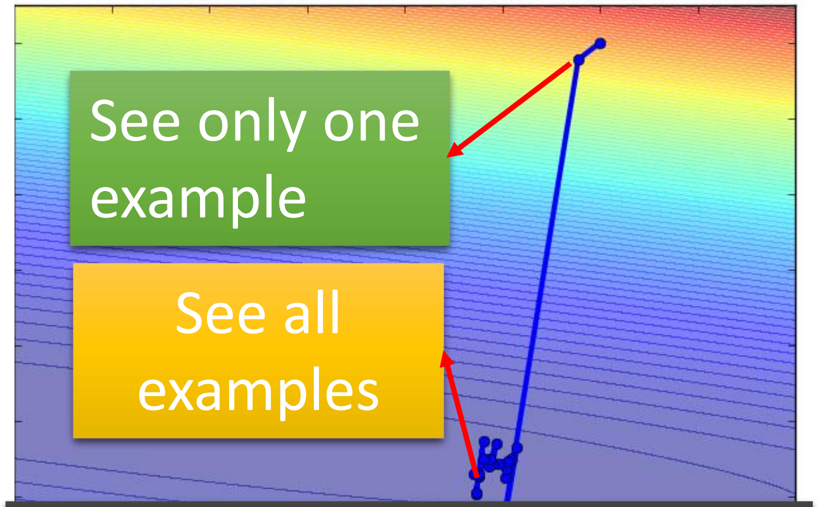


~~Long time for cooldown,~~ but powerful

Batch size = 1

Update for each example

Update 20 times in an epoch

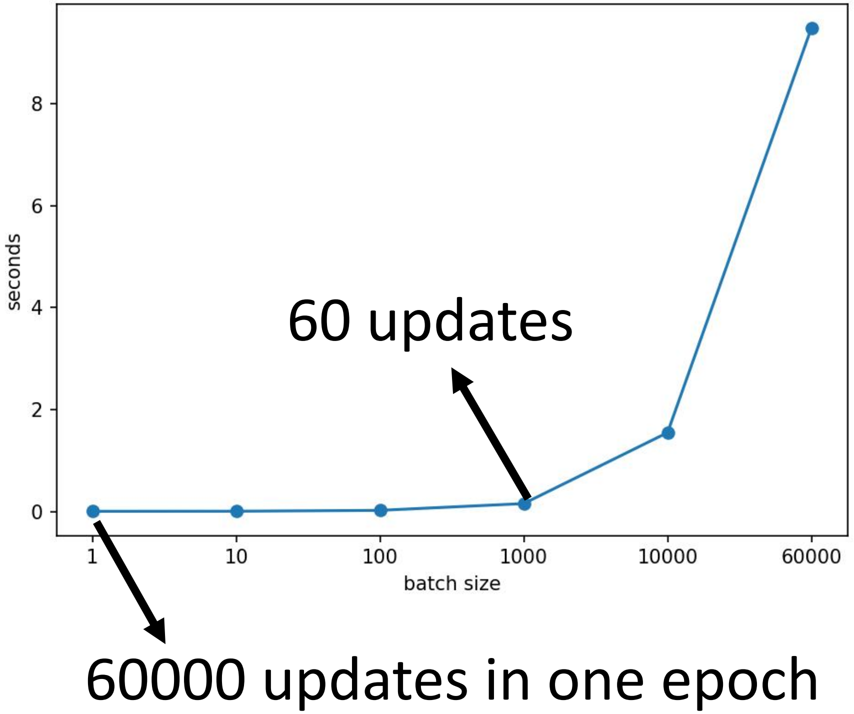


~~Short time for cooldown,~~ but noisy

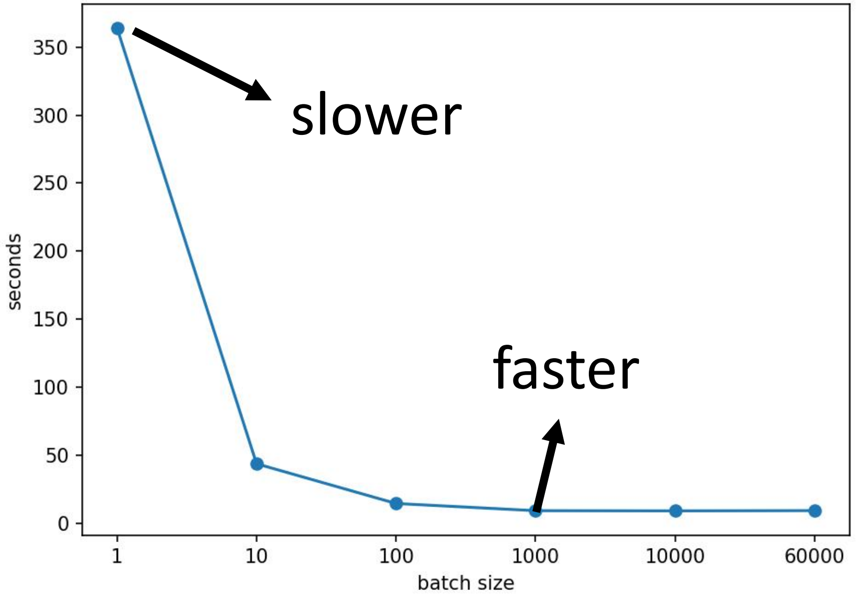
Small Batch v.s. Large Batch

- Smaller batch requires longer time for one epoch (longer time for seeing all data once)

Time for one **update**



Time for one **epoch**



MNIST: digit classification

Tesla V100 GPU

Small Batch v.s. Large Batch

- Small batch is better on testing data?

SB = 256

LB = 0.1 x data set

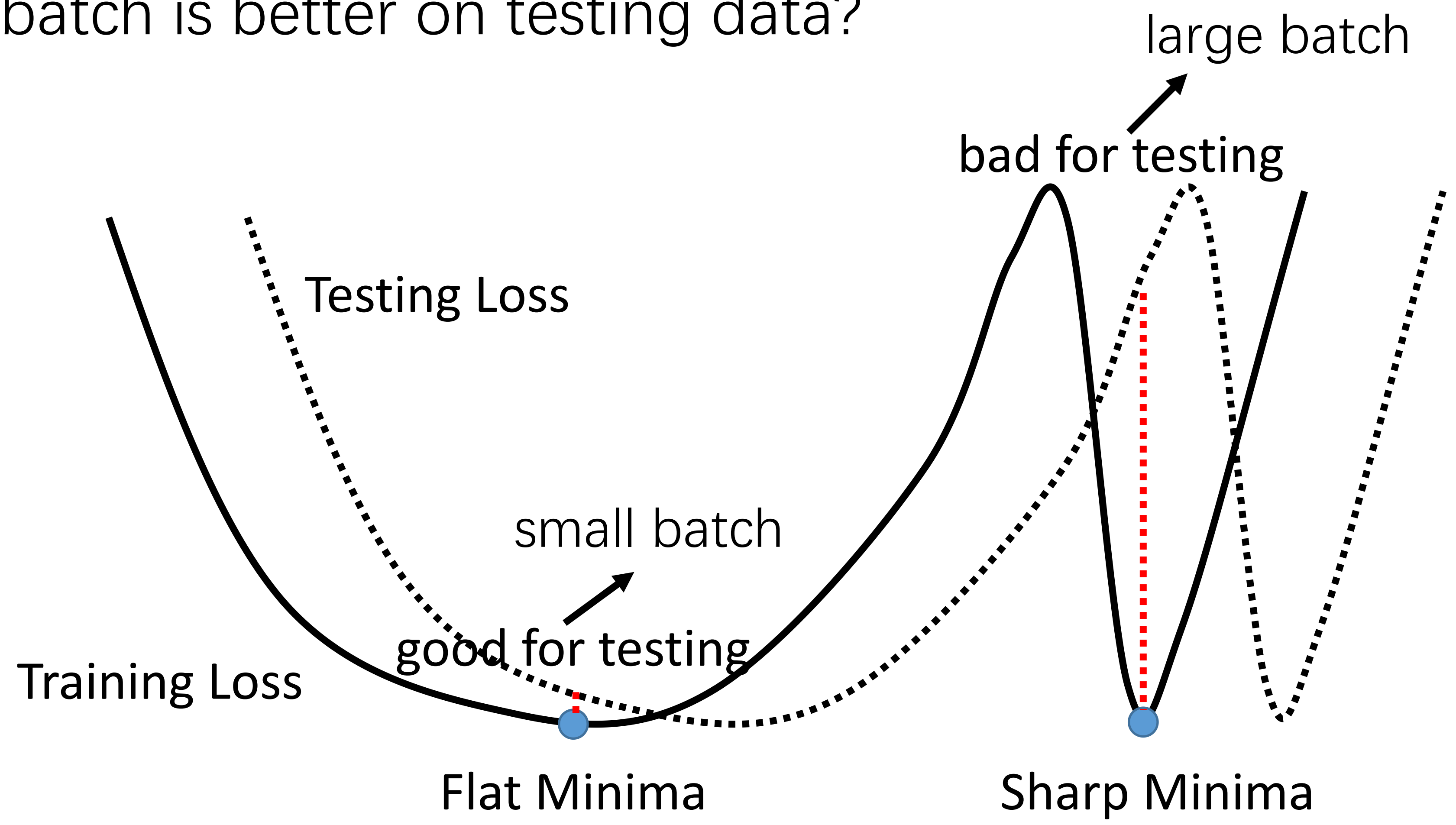
Name	Network Type	Data set
F_1	Fully Connected	MNIST (LeCun et al., 1998a)
F_2	Fully Connected	TIMIT (Garofolo et al., 1993)
C_1	(Shallow) Convolutional	CIFAR-10 (Krizhevsky & Hinton, 2009)
C_2	(Deep) Convolutional	CIFAR-10
C_3	(Shallow) Convolutional	CIFAR-100 (Krizhevsky & Hinton, 2009)
C_4	(Deep) Convolutional	CIFAR-100

	Training Accuracy		Testing Accuracy	
Name	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	$99.99\% \pm 0.04\%$	$99.99\% \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima
https://arxiv.org/abs/1609.04836

Small Batch v.s. Large Batch

- Small batch is better on testing data?



Small Batch v.s. Large Batch

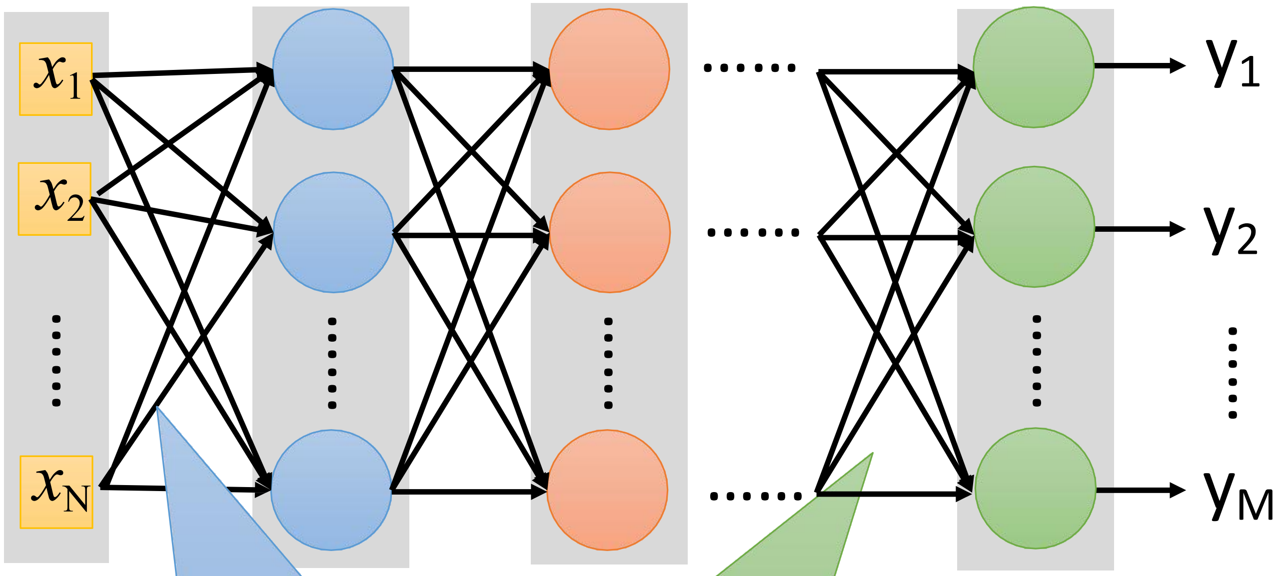
	Small	Large
Speed for one update (no parallel)	Faster	Slower
Speed for one update (with parallel)	Same	Same (not too large)
Time for one epoch	Slower	Faster
Gradient	Noisy	Stable
Optimization	Better	Worse
Generalization	Better	Worse

Batch size is a hyperparameter you have to decide.

Tips for Training

2. Activation Function

Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

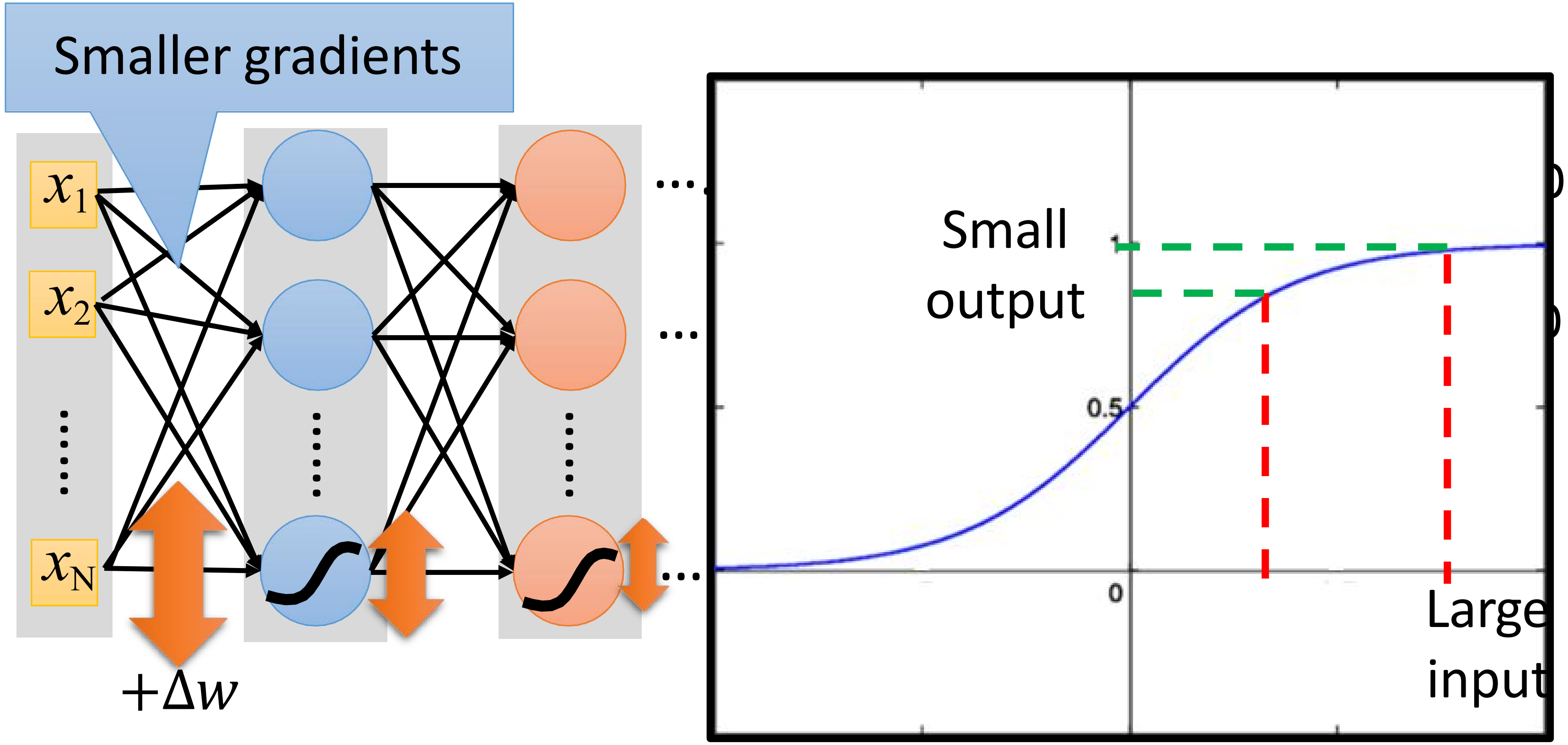
Larger gradients

Learn very fast

Already converge

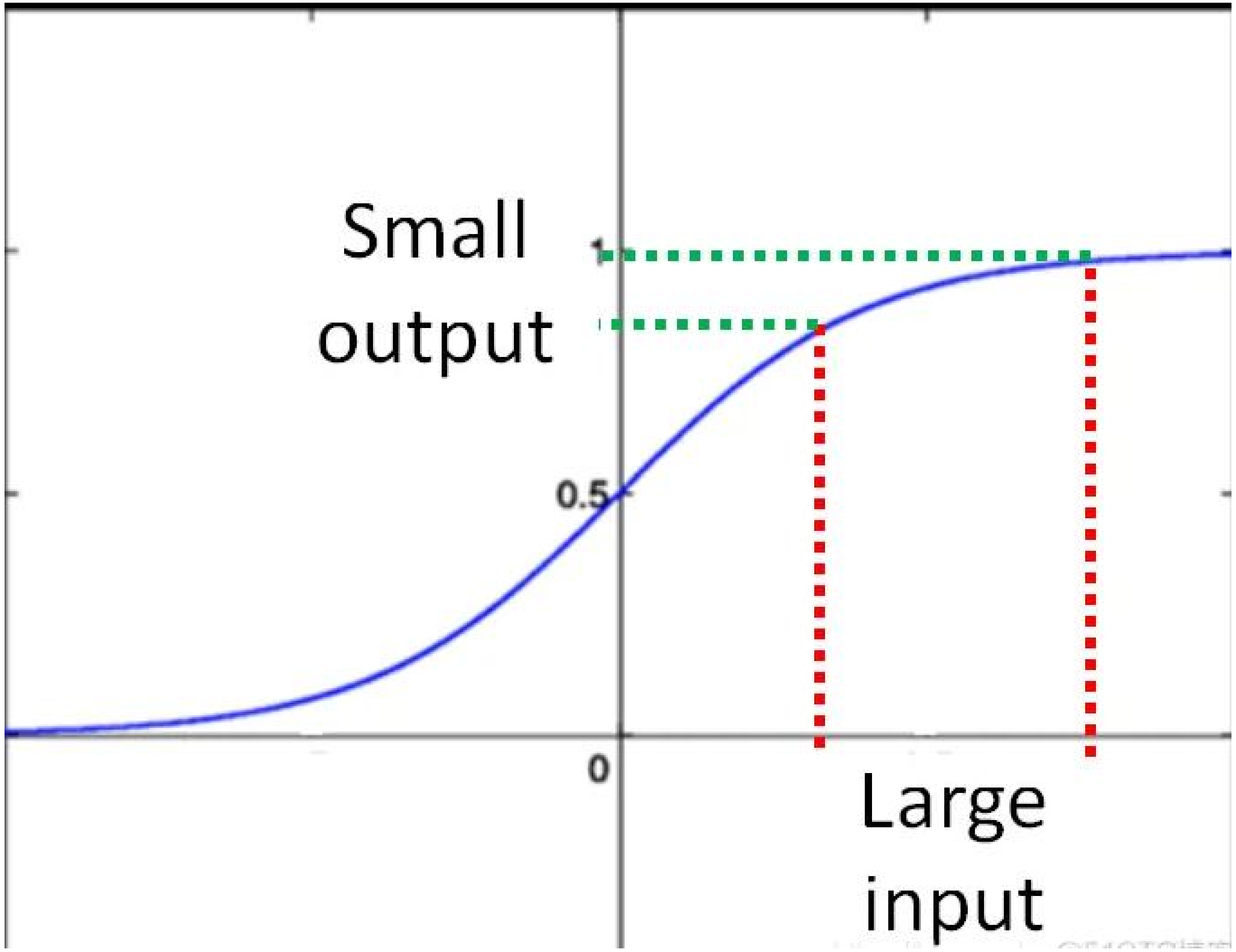
based on random!?

Vanishing Gradient Problem



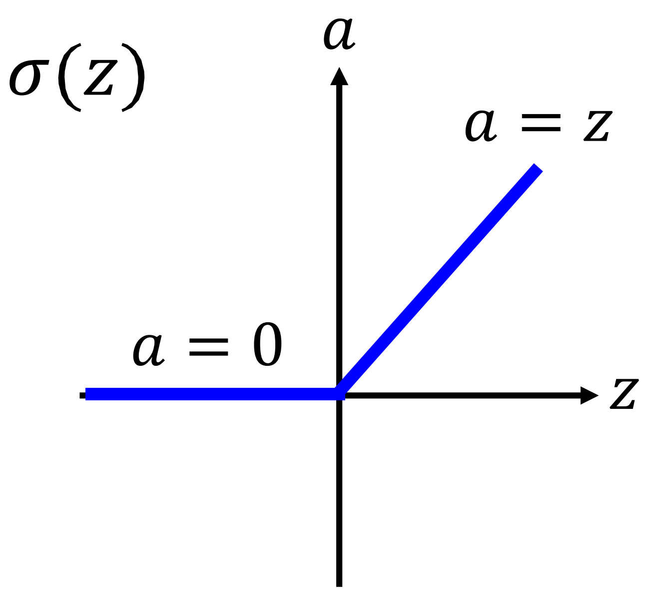
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$


Small
output
1
0.5
0
Large
input

ReLU

- Rectified Linear Unit (ReLU)

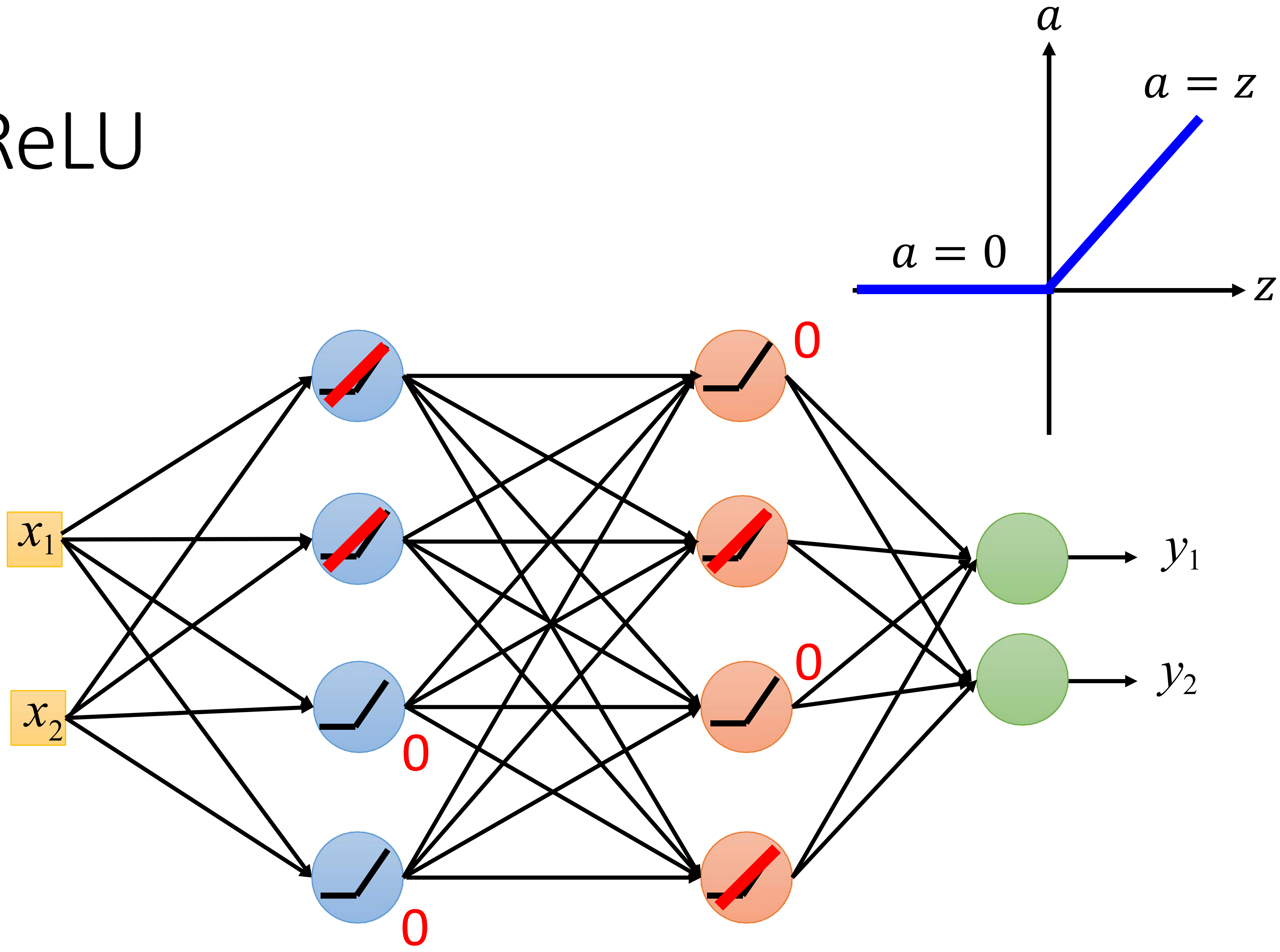


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

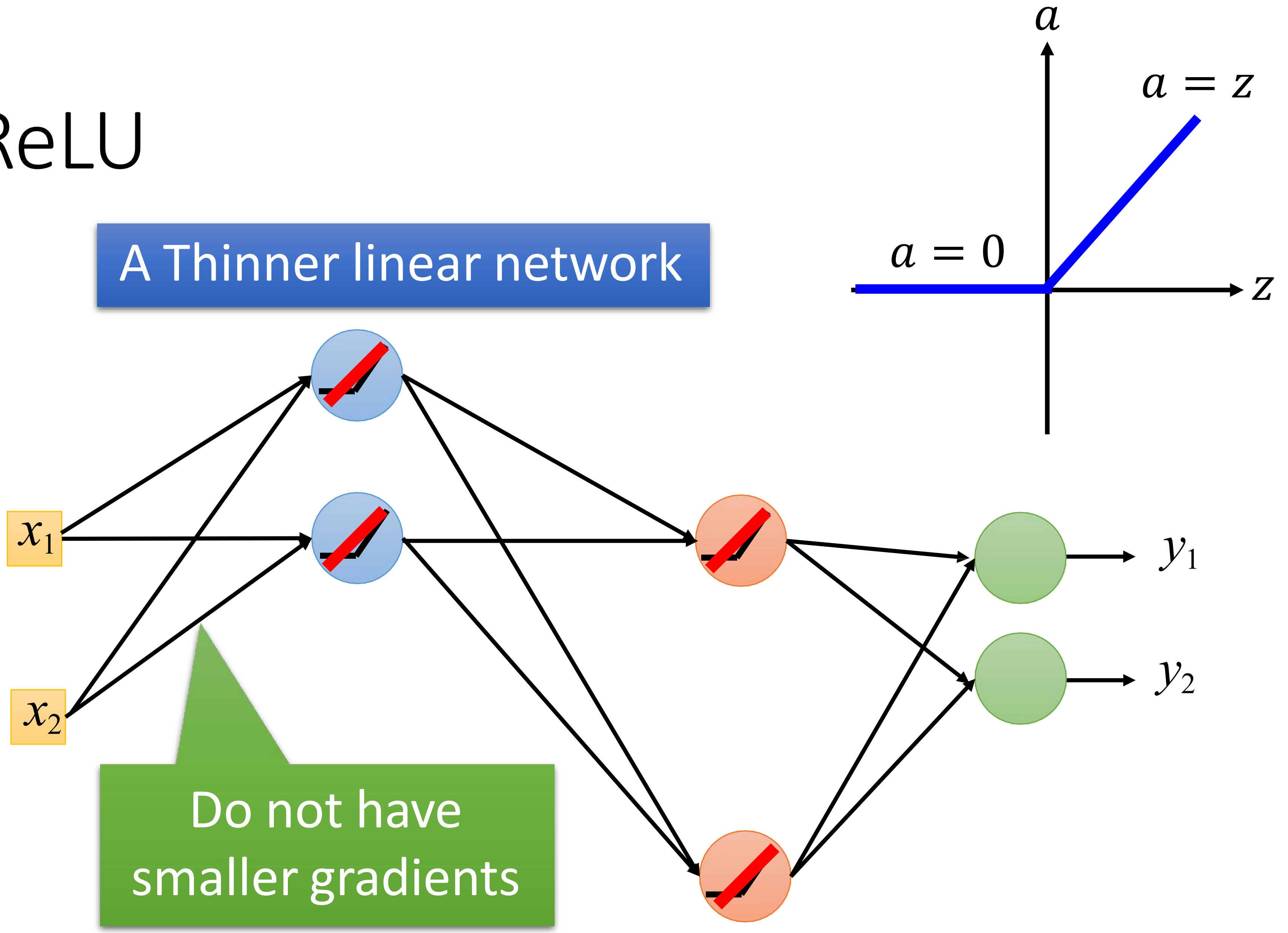
Reason:

1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

ReLU



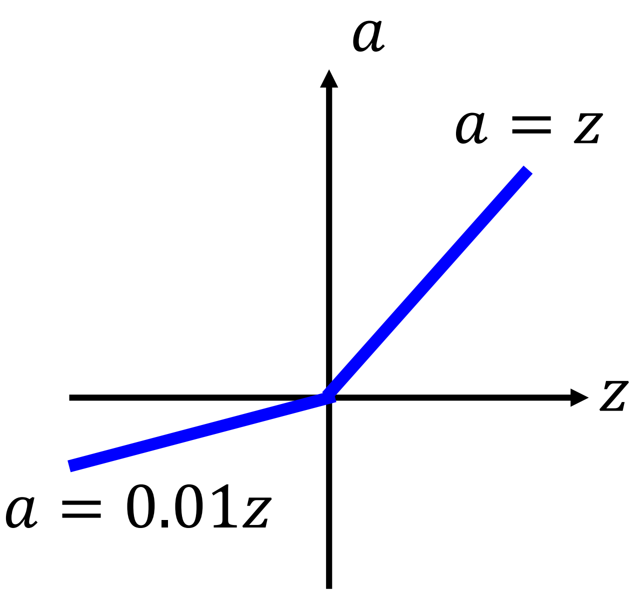
ReLU



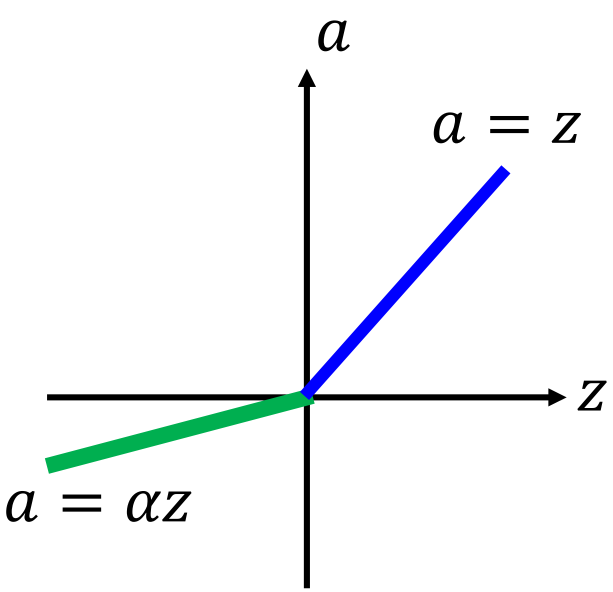
A Thinner linear network
Do not have
smaller gradients
x_1
x_2
y_1
y_2
a
$a = z$
$a = 0$
z

ReLU - variant

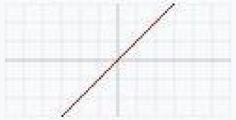
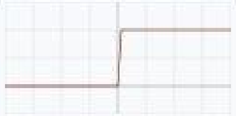
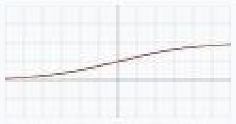
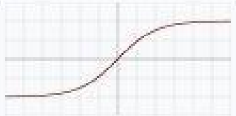
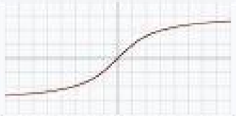
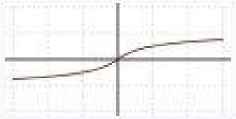
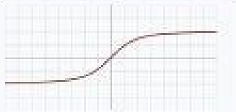
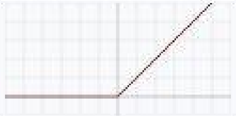
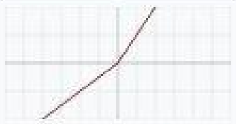
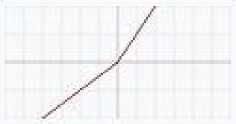
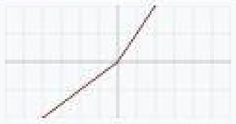
$Leaky\ ReLU$

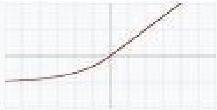
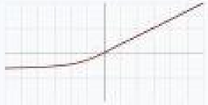
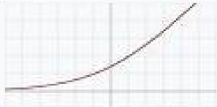
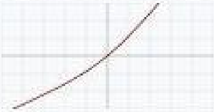
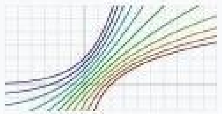
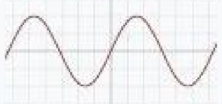
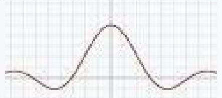


$Parametric\ ReLU$



α also learned by gradient descent

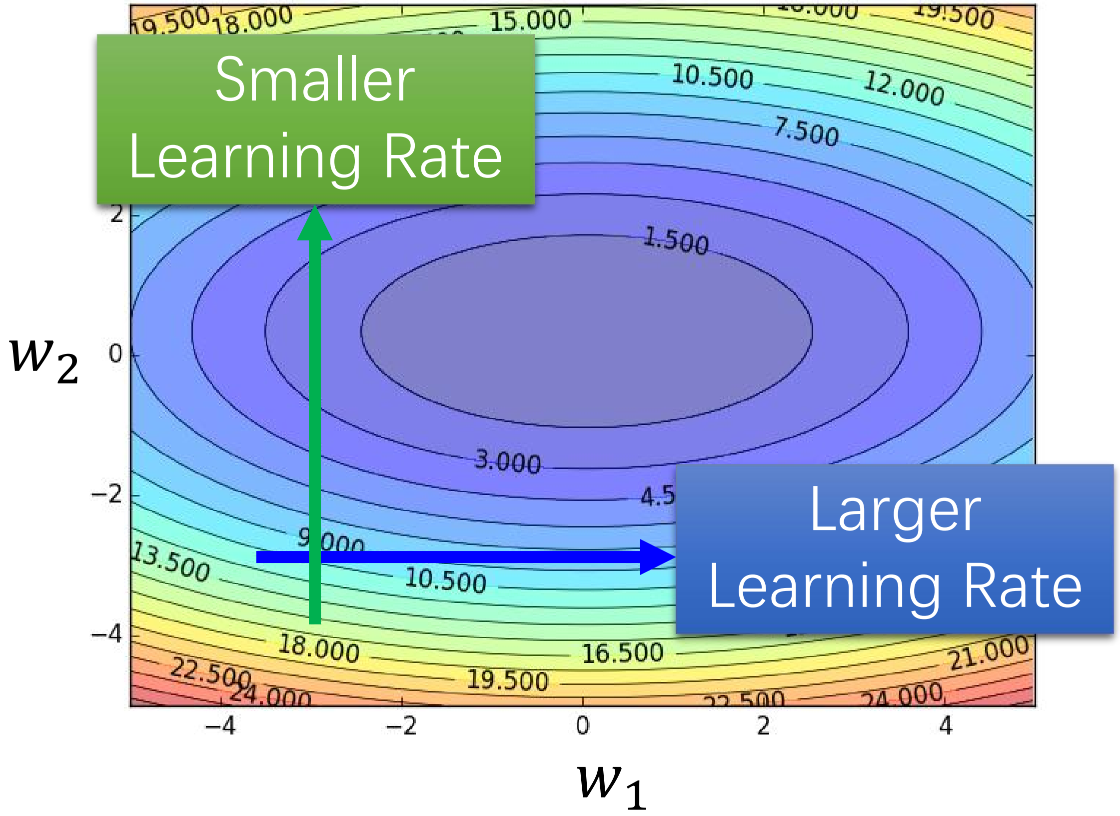
Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]
TanH		$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign[9][10]		$f(x) = \frac{x}{1+\lvert x \rvert}$
Inverse square root unit (ISRU)[11]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$
Rectified linear unit (ReLU)[12]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU)[13]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric rectified linear unit (PReLU)[14]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Randomized leaky rectified linear unit (RReLU)[15]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ [3]

Exponential linear unit (ELU)[16]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
Scaled exponential linear unit (SELU)[17]		$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ with $\lambda = 1.0507$ and $\alpha = 1.67326$
S-shaped rectified linear activation unit (SReLU)[18]		$f_{t_l, a_l, t_r, a_r}(x) = \begin{cases} t_l + a_l(x - t_l) & \text{for } x \leq t_l \\ x & \text{for } t_l < x < t_r \\ t_r + a_r(x - t_r) & \text{for } x \geq t_r \end{cases}$ t_l, a_l, t_r, a_r are parameters.
Inverse square root linear unit (ISRLU)[11]		$f(x) = \begin{cases} \frac{x}{\sqrt{1 + \alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Adaptive piecewise linear (APL)[19]		$f(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$
SoftPlus[20]		$f(x) = \ln(1 + e^x)$
Bent identity		$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x$
Sigmoid-weighted linear unit (SiLU)[21] (a.k.a. Swish[22])		$f(x) = x \cdot \sigma(x)$ [5]
SoftExponential[23]		$f(\alpha, x) = \begin{cases} -\frac{\ln(1 - \alpha(x + \alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$
Sinusoid[24]		$f(x) = \sin(x)$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$

Tips for Training

3. Optimizer

Different parameters needs different learning rate



Formulation for **one** parameter:

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\eta} \boldsymbol{g}_i^{t}$$

$$\boldsymbol{g}_i^{t} = \frac{\partial L}{\partial \boldsymbol{\theta}_i} |_{\boldsymbol{\theta}=\boldsymbol{\theta}^t}$$

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^{t}$$

Parameter dependent

Root Mean Square

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^t$$

$$\boldsymbol{\theta}_i^{1} \leftarrow \boldsymbol{\theta}_i^{0} - \frac{\eta}{\sigma_i^0} \boldsymbol{g}_i^0 \qquad \sigma_i^0 = \sqrt{\left(\boldsymbol{g}_i^0\right)^2} = \left|\boldsymbol{g}_i^0\right|$$

$$\boldsymbol{\theta}_i^{2} \leftarrow \boldsymbol{\theta}_i^{1} - \frac{\eta}{\sigma_i^1} \boldsymbol{g}_i^1 \qquad \sigma_i^1 = \sqrt{\frac{1}{2}\left[\left(\boldsymbol{g}_i^0\right)^2 + \left(\boldsymbol{g}_i^1\right)^2\right]}$$

$$\boldsymbol{\theta}_i^{3} \leftarrow \boldsymbol{\theta}_i^{2} - \frac{\eta}{\sigma_i^2} \boldsymbol{g}_i^2 \qquad \sigma_i^2 = \sqrt{\frac{1}{3}\left[\left(\boldsymbol{g}_i^0\right)^2 + \left(\boldsymbol{g}_i^1\right)^2 + \left(\boldsymbol{g}_i^2\right)^2\right]}$$

$$\vdots$$

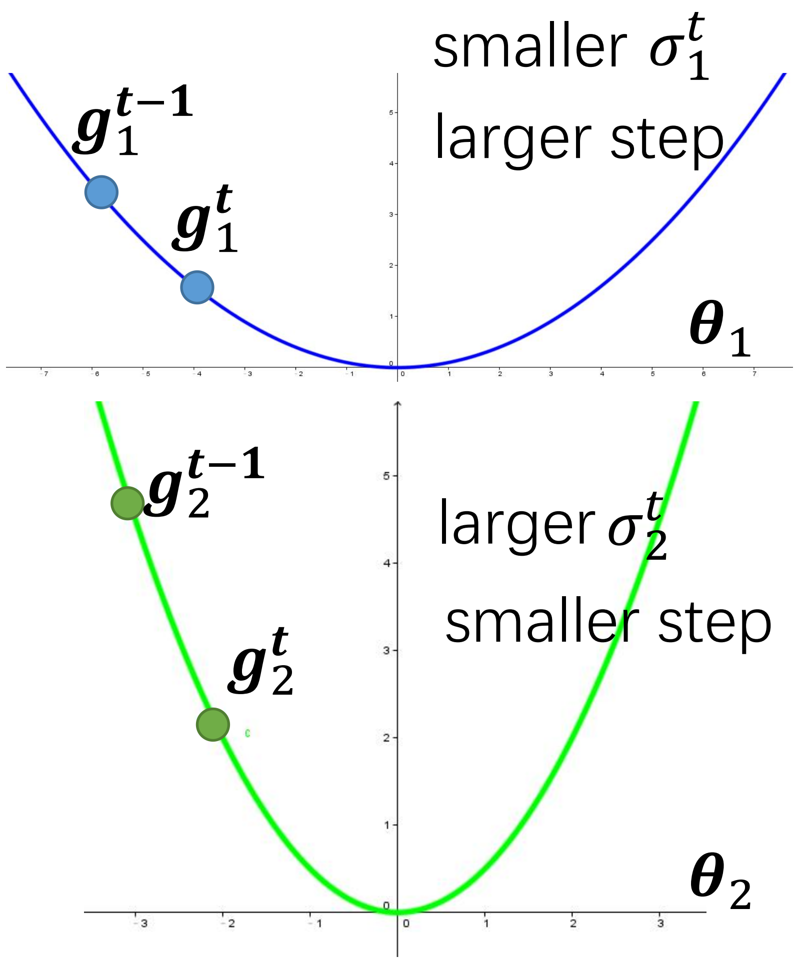
$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \frac{\eta}{\sigma_i^t} \boldsymbol{g}_i^t \qquad \sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{j=0}^{t}\left(\boldsymbol{g}_i^j\right)^2}$$

Root Mean Square

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^t$$

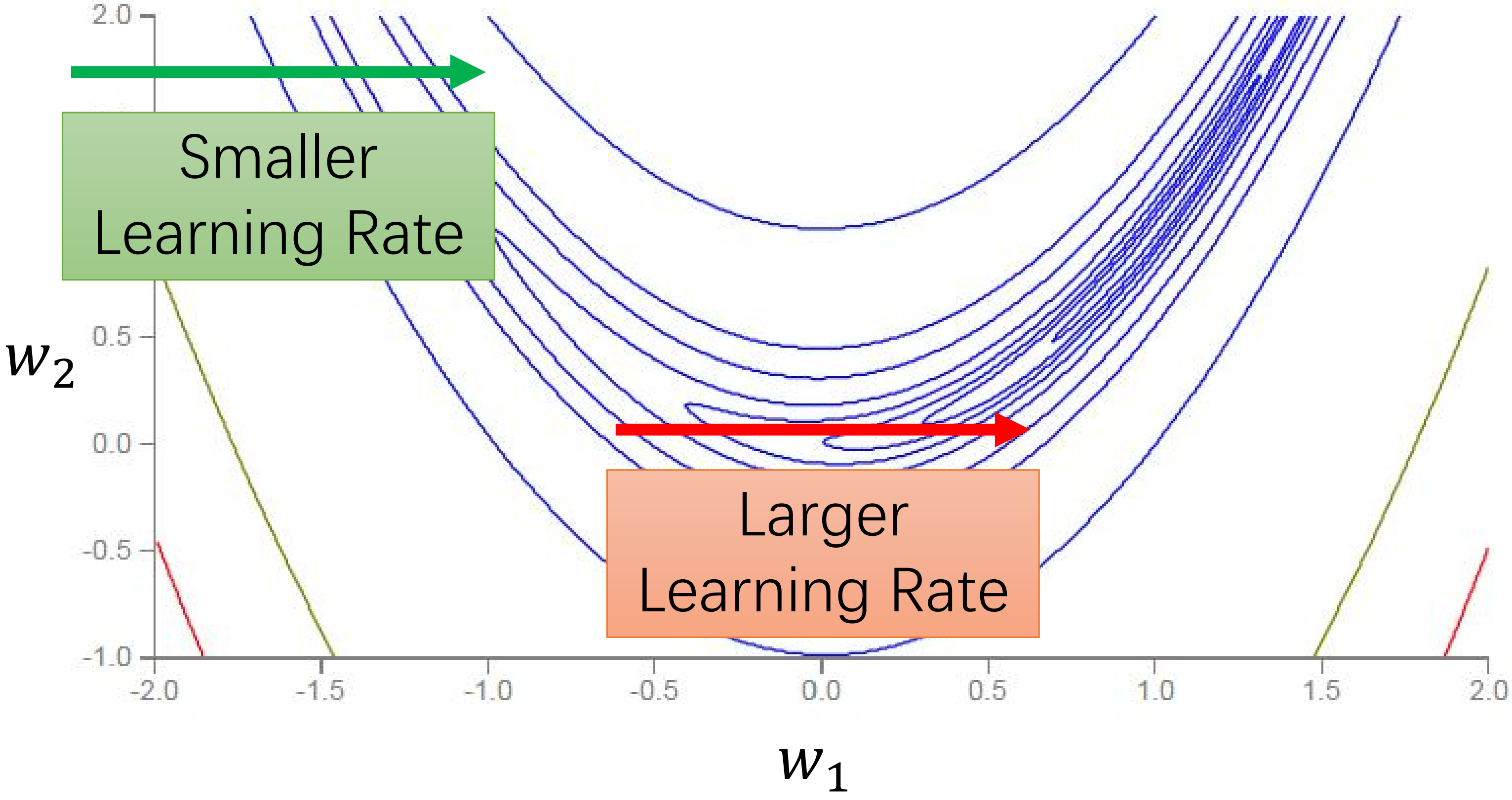
$$\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{j=0}^{t}\left(\boldsymbol{g}_i^j\right)^2}$$

Used in **Adagrad**



Learning rate adapts dynamically

Error Surface can be very complex.



RMSProp

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^t$$

$$\boldsymbol{\theta}_i^{1} \leftarrow \boldsymbol{\theta}_i^{0} - \frac{\eta}{\sigma_i^0} \boldsymbol{g}_i^0 \qquad \sigma_i^0 = \sqrt{\left(\boldsymbol{g}_i^0\right)^2}$$

$$0 < \alpha < 1$$

$$\boldsymbol{\theta}_i^{2} \leftarrow \boldsymbol{\theta}_i^{1} - \frac{\eta}{\sigma_i^1} \boldsymbol{g}_i^1 \qquad \sigma_i^1 = \sqrt{\alpha\left(\sigma_i^0\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^1\right)^2}$$

$$\boldsymbol{\theta}_i^{3} \leftarrow \boldsymbol{\theta}_i^{2} - \frac{\eta}{\sigma_i^2} \boldsymbol{g}_i^2 \qquad \sigma_i^2 = \sqrt{\alpha\left(\sigma_i^1\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^2\right)^2}$$

$$\vdots$$

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \frac{\eta}{\sigma_i^t} \boldsymbol{g}_i^t \qquad \sigma_i^t = \sqrt{\alpha\left(\sigma_i^{t-1}\right)^2 + (1-\alpha)(\boldsymbol{g}_i^t)^2}$$

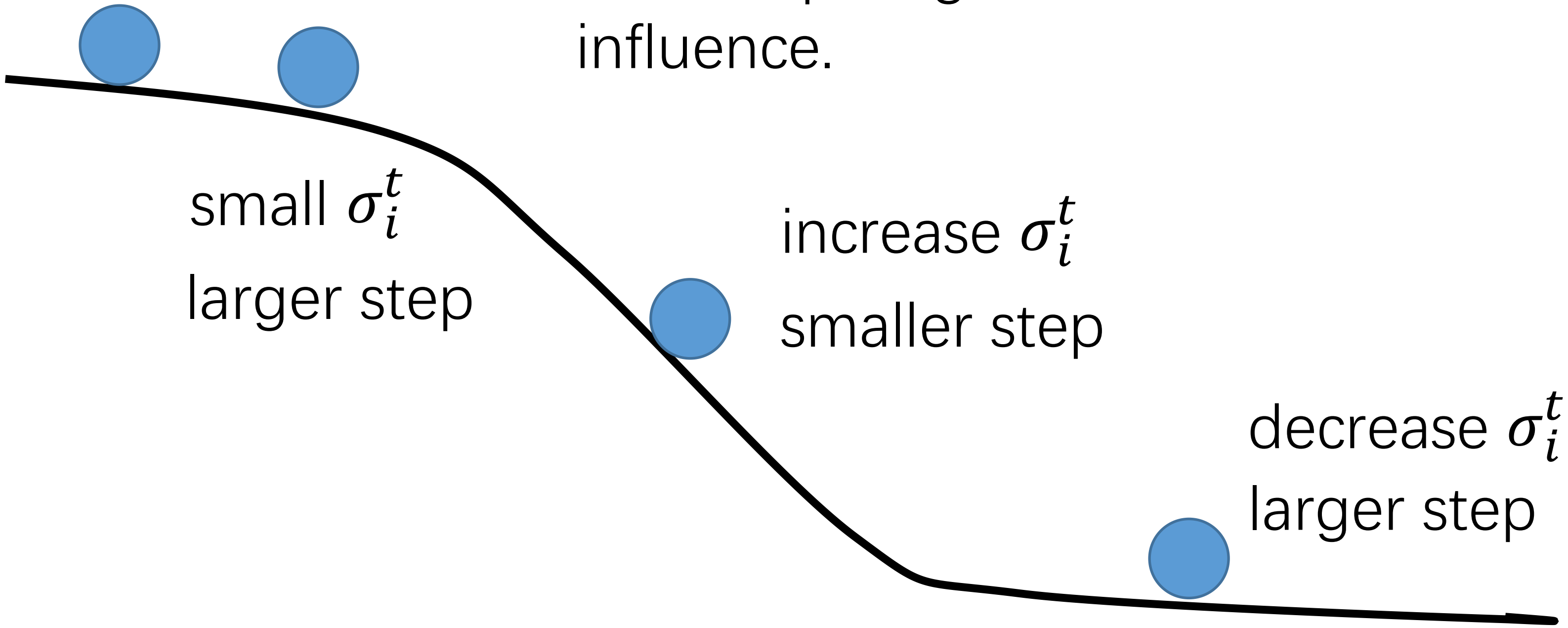
RMSProp

$$\boldsymbol{g}_i^{\mathbf{1}} \quad \boldsymbol{g}_i^{\mathbf{2}} \cdots\cdots \quad \boldsymbol{g}_i^{\boldsymbol{t}-\mathbf{1}}$$

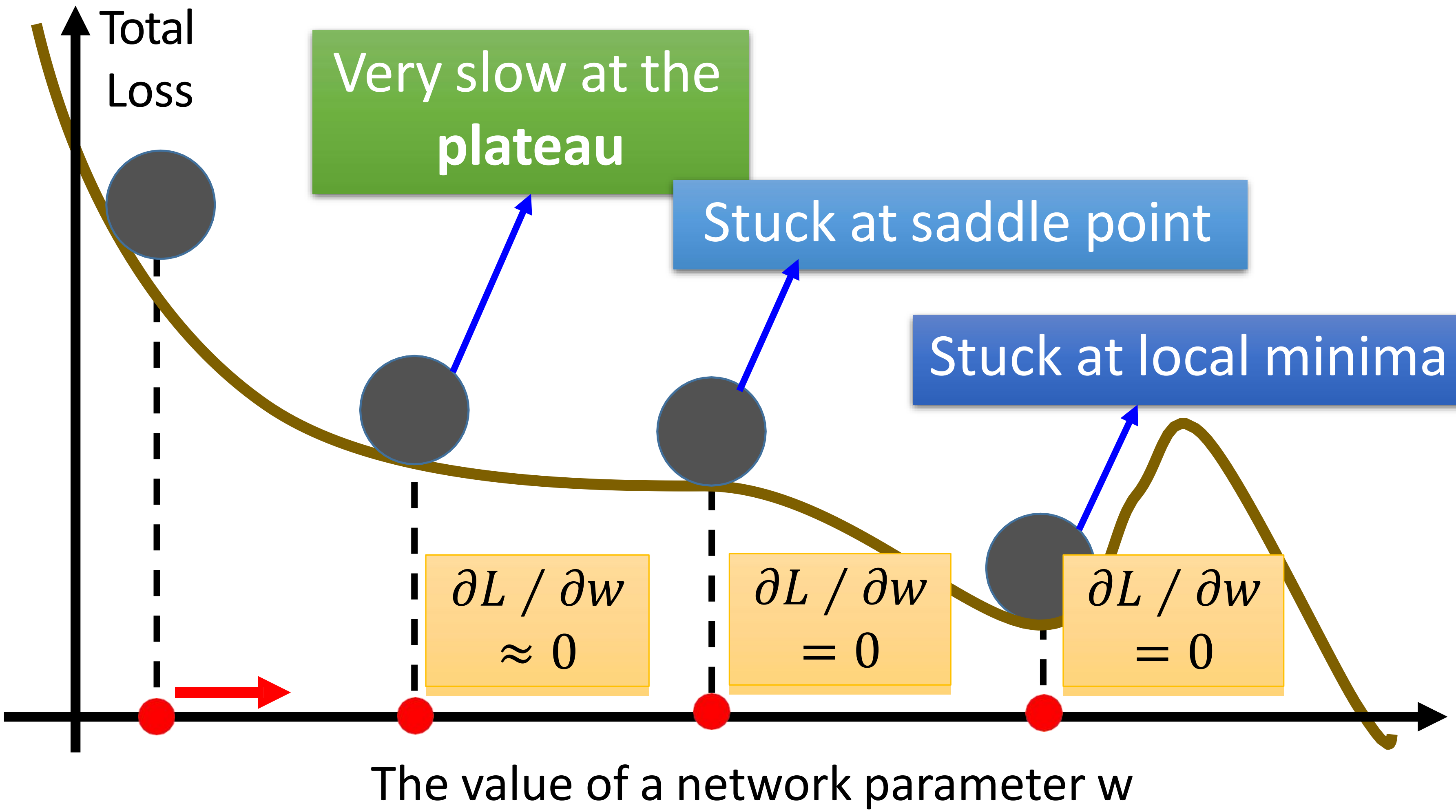
$$0 < \alpha < 1$$

$$\boldsymbol{\theta}_i^{\boldsymbol{t}+\mathbf{1}} \leftarrow \boldsymbol{\theta}_i^{\boldsymbol{t}} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^{\boldsymbol{t}} \qquad \sigma_i^t = \sqrt{\alpha\left(\sigma_i^{t-1}\right)^2 + (1-\alpha)(\boldsymbol{g}_i^{\boldsymbol{t}})^2}$$

The recent gradient has larger influence, and the past gradients have less influence.

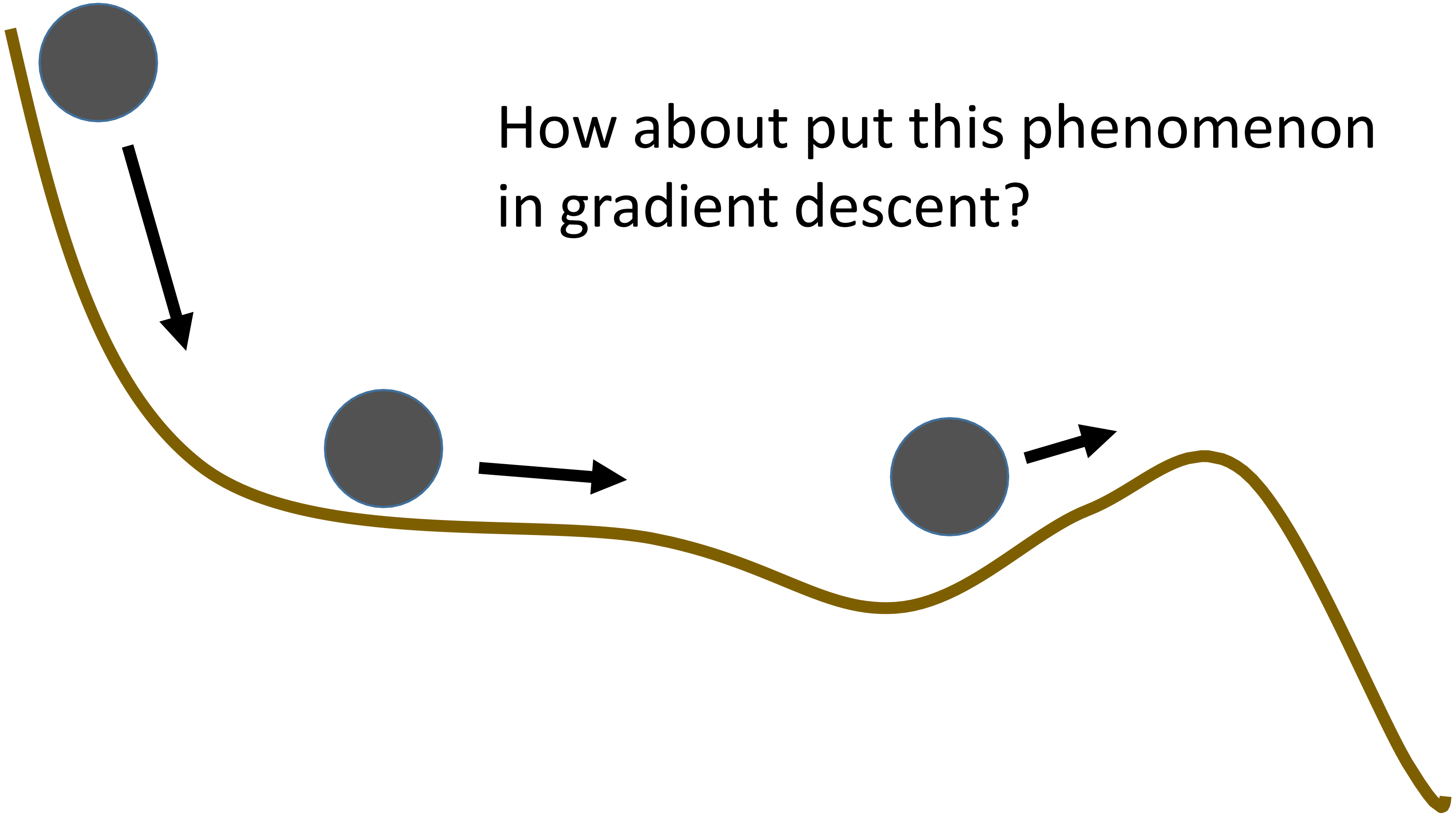


Hard to find optimal network parameters



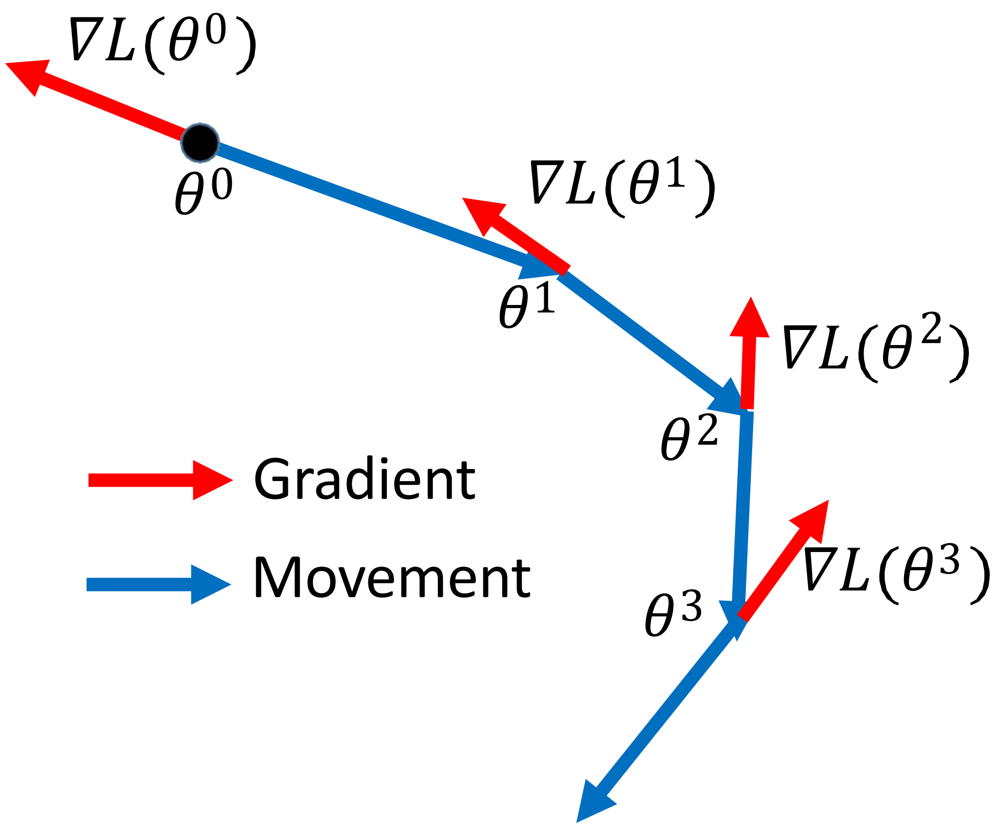
In physical world ……

- Momentum



How about put this phenomenon in gradient descent?

Review: Vanilla Gradient Descent



Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta\nabla L(\theta^0)$

Compute gradient at θ^1

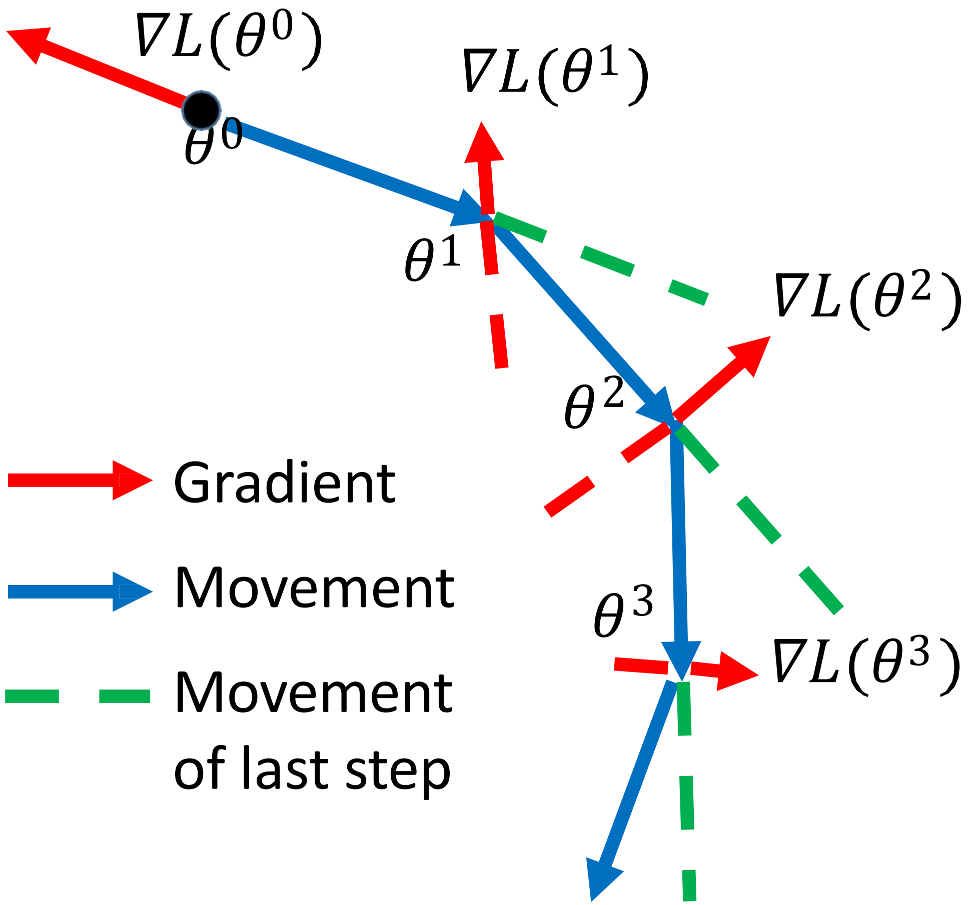
Move to $\theta^2 = \theta^1 - \eta\nabla L(\theta^1)$

⋮

Stop until $\nabla L(\theta^t) \approx 0$

Momentum

Movement: movement of last step minus gradient at present



- Start at point θ^0 Movement $v^0=0$
- Compute gradient at θ^0

  Movement $v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$

  Move to $\theta^1 = \theta^0 + v^1$
- Compute gradient at θ^1

  Movement $v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$
- Move to $\theta^2 = \theta^1 + v^2$

Movement not just based on gradient, but previous movement.

Momentum

Movement: movement of last step minus gradient at present

v^i is actually the weighted sum of all the previous gradient:
$\nabla L(\theta^0), \nabla L(\theta^1), \ldots \nabla L(\theta^{i-1})$

$v^0 = 0$

$v^1 = -\eta \nabla L(\theta^0)$

$v^2 = -\lambda \eta \nabla L(\theta^0) - \eta \nabla L(\theta^1)$

$\vdots$

- Start at point θ^0 Movement $v^0=0$
- Compute gradient at θ^0

Movement $v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$

Move to $\theta^1 = \theta^0 + v^1$

- Compute gradient at θ^1

Movement $v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$

- Move to $\theta^2 = \theta^1 + v^2$

Movement not just based on gradient, but previous movement

Original paper: https://arxiv.org/pdf/1412.6980.pdf

Adam: RMSProp + Momentum

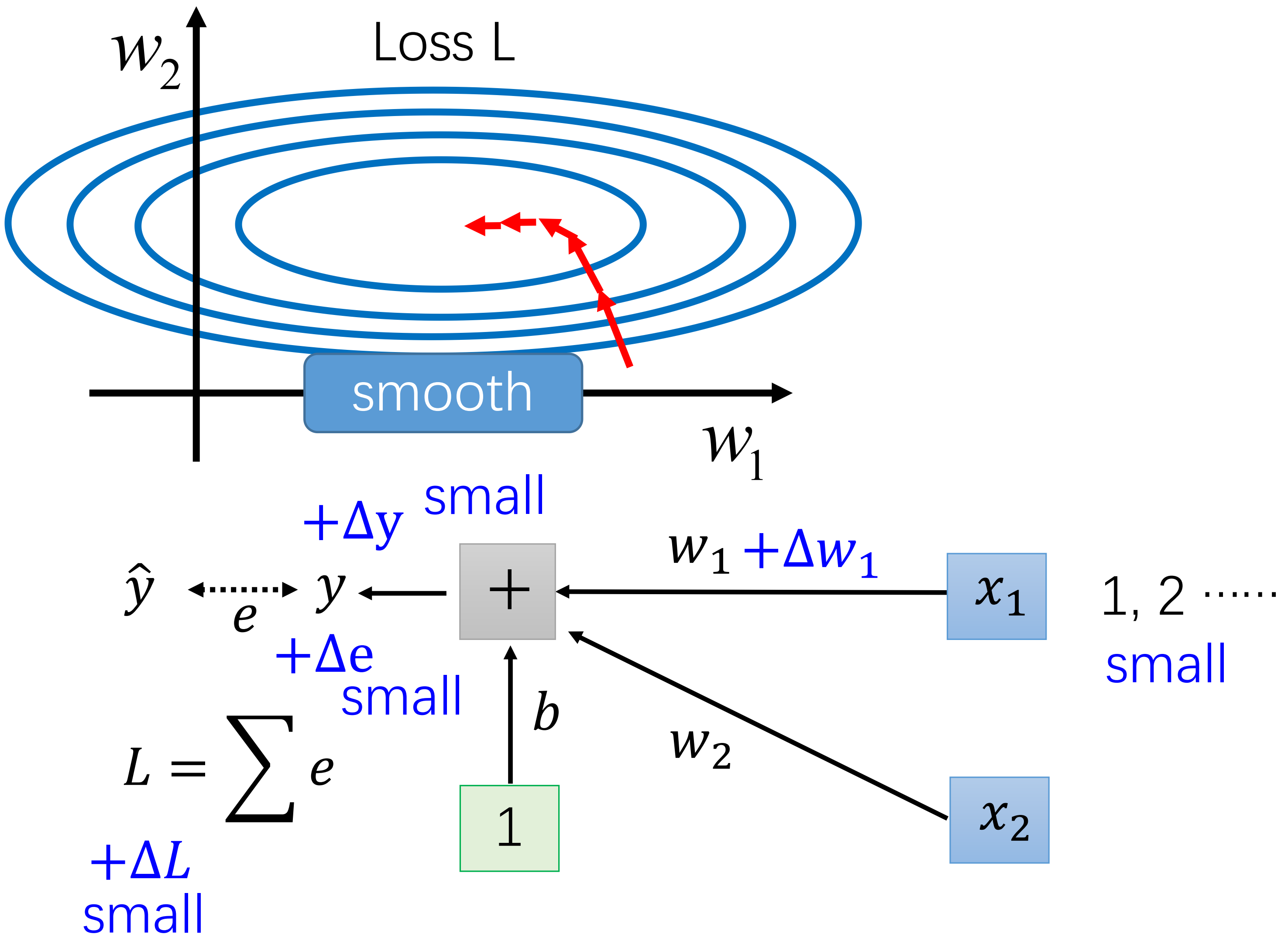
Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector) → for RMSprop
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

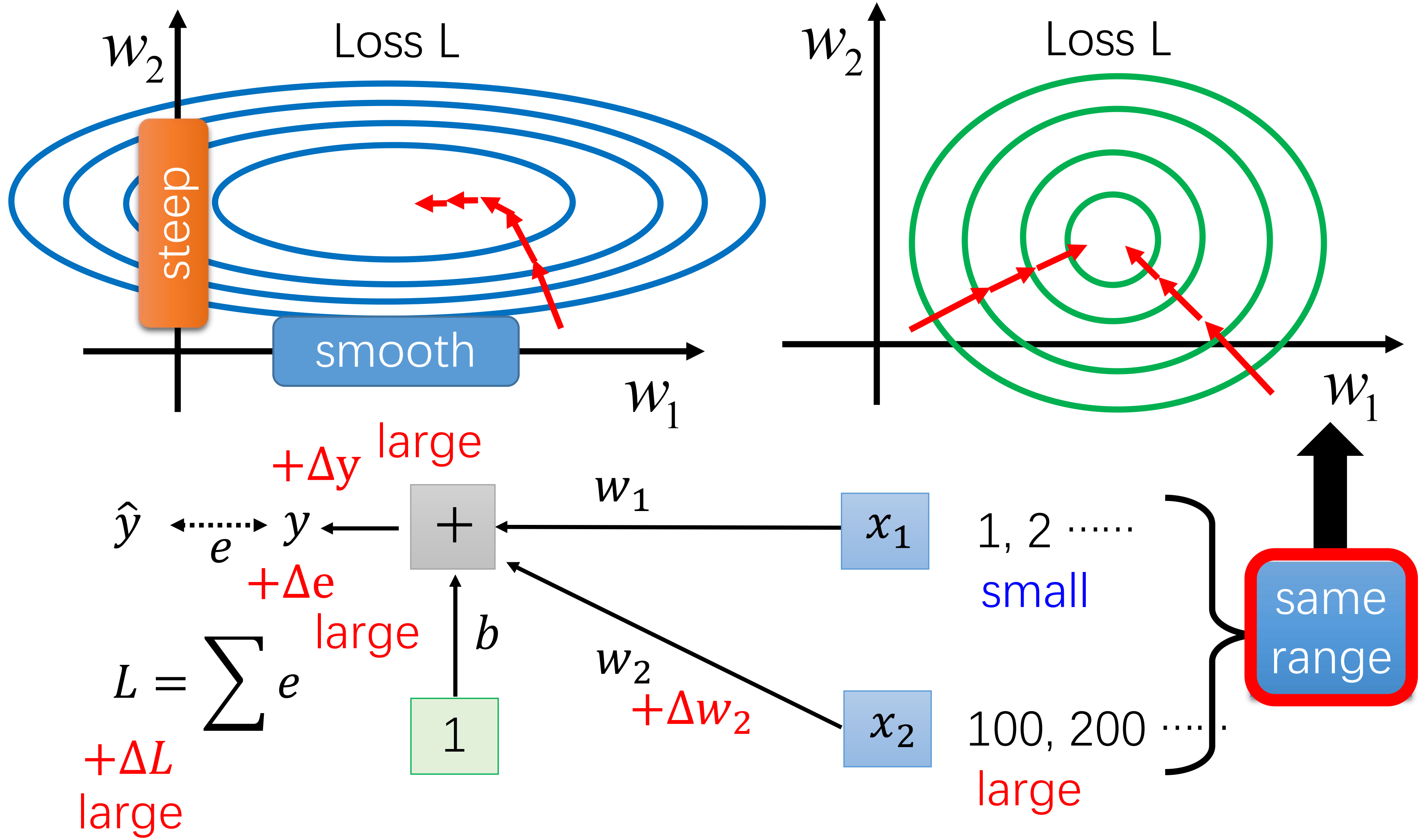
Tips for Training

4. Normalization

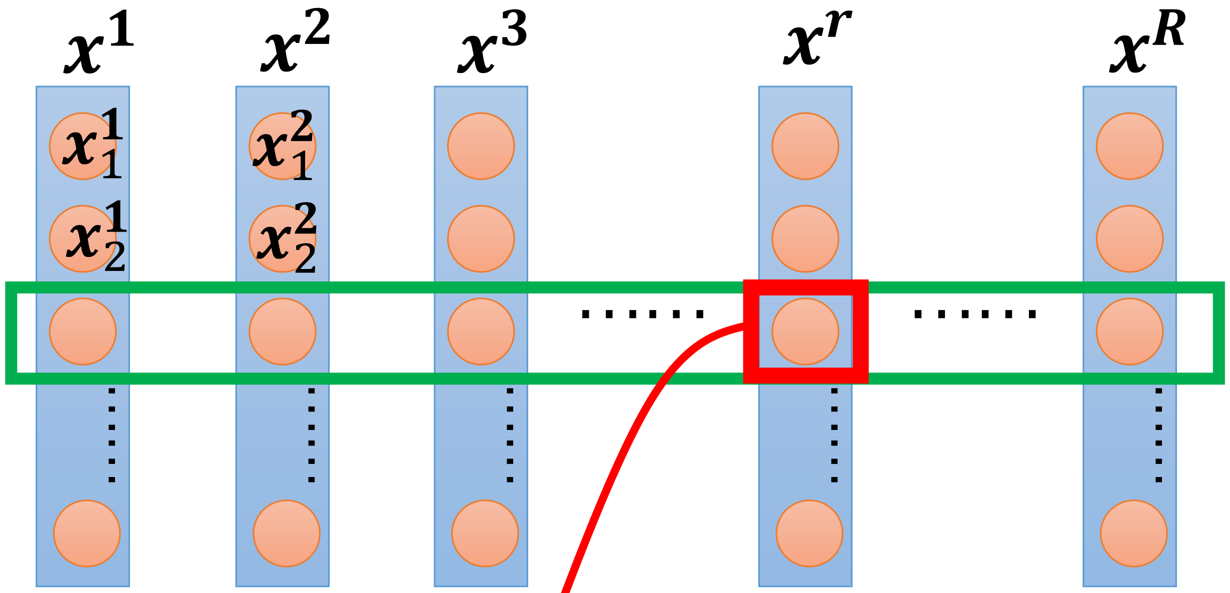
Changing Landscape



Changing Landscape



Feature Normalization



For each dimension i:

mean: m_i

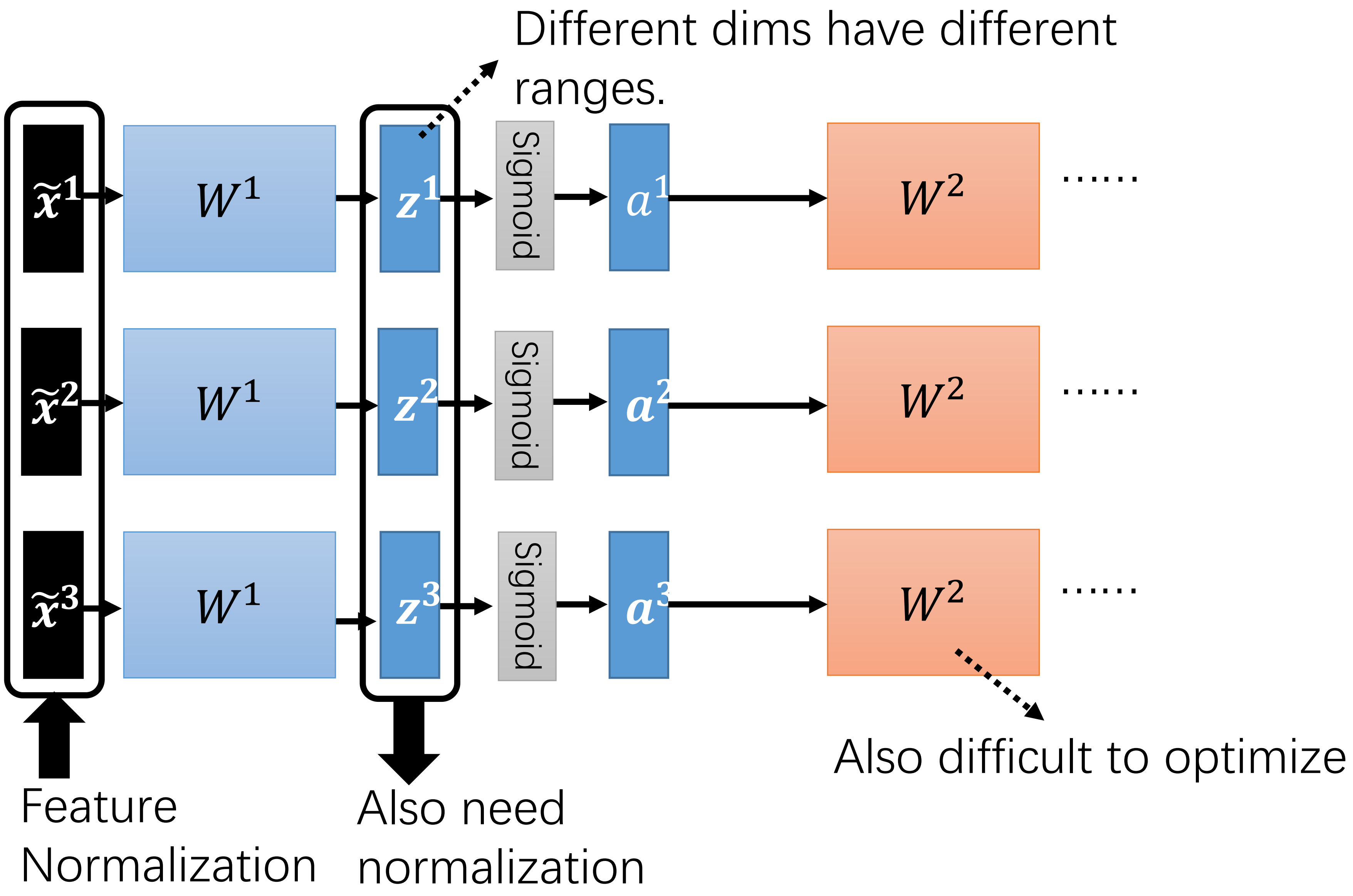
standard deviation: σ_i

$$\tilde{x}_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

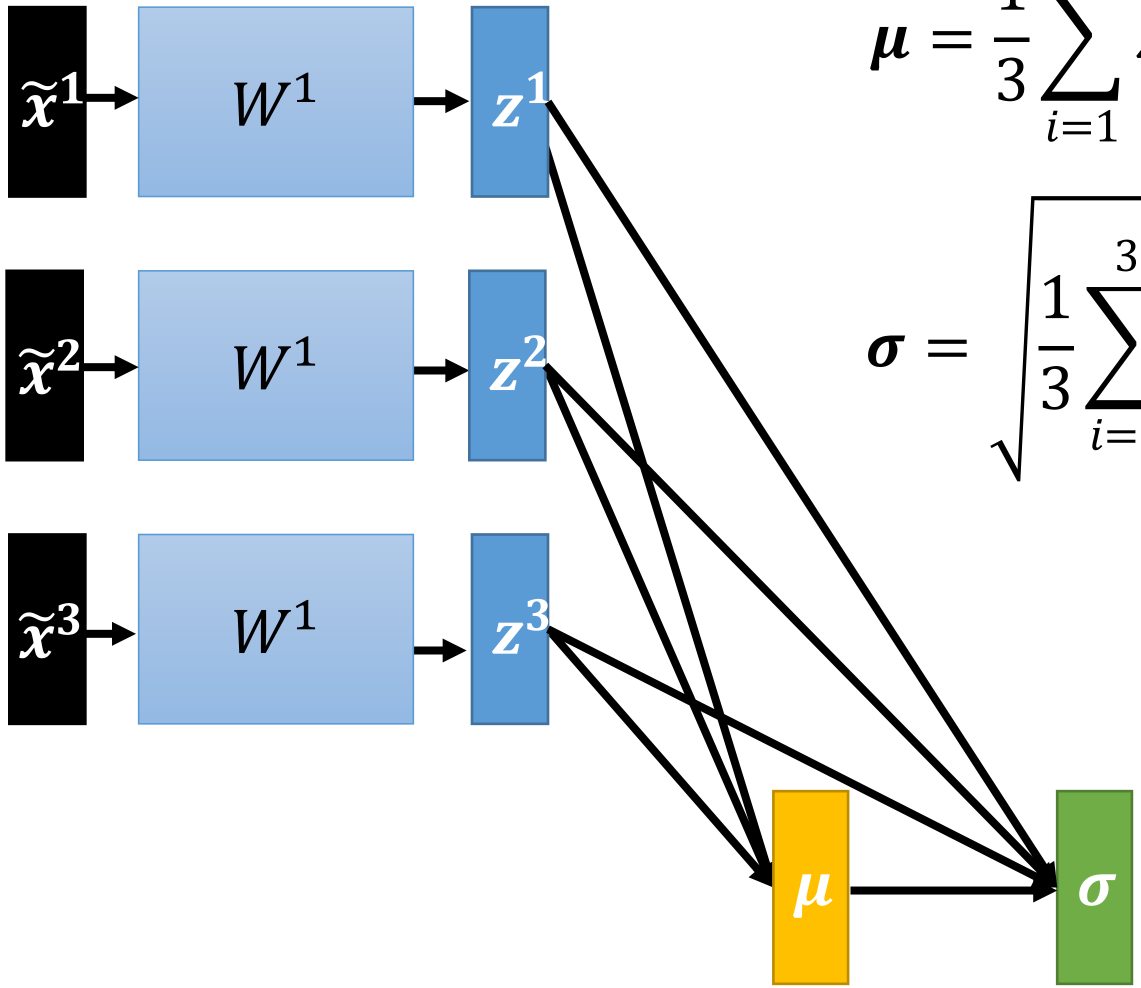
The means of all dims are 0, and the variances are all 1

In general, feature normalization makes gradient descent converge faster.

Considering Deep Learning



Considering Deep Learning



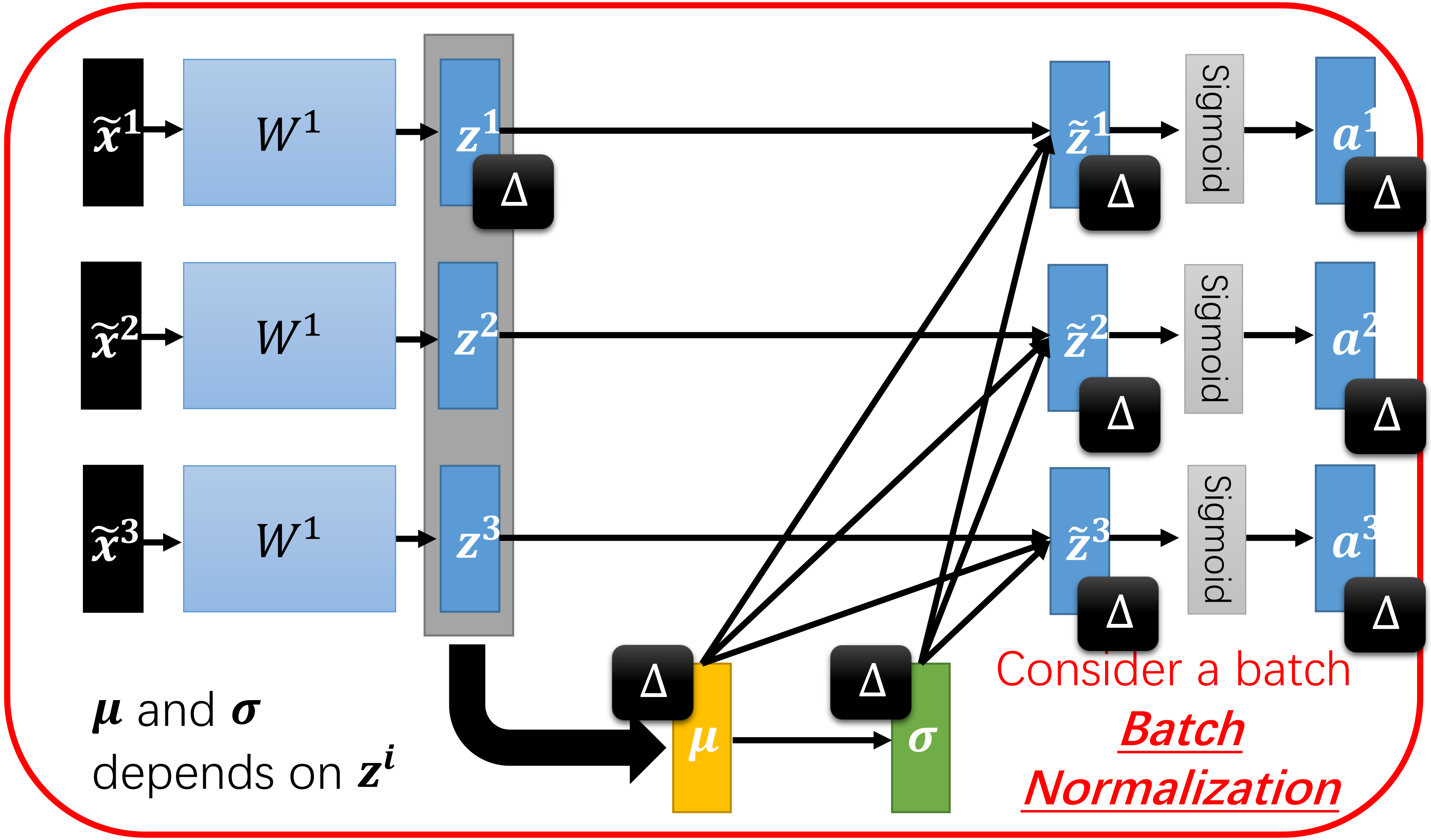
$$\mu = \frac{1}{3}\sum_{i=1}^{3} z^i$$

$$\sigma = \sqrt{\frac{1}{3}\sum_{i=1}^{3}(z^i - \mu)^2}$$

Considering Deep Learning

$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$

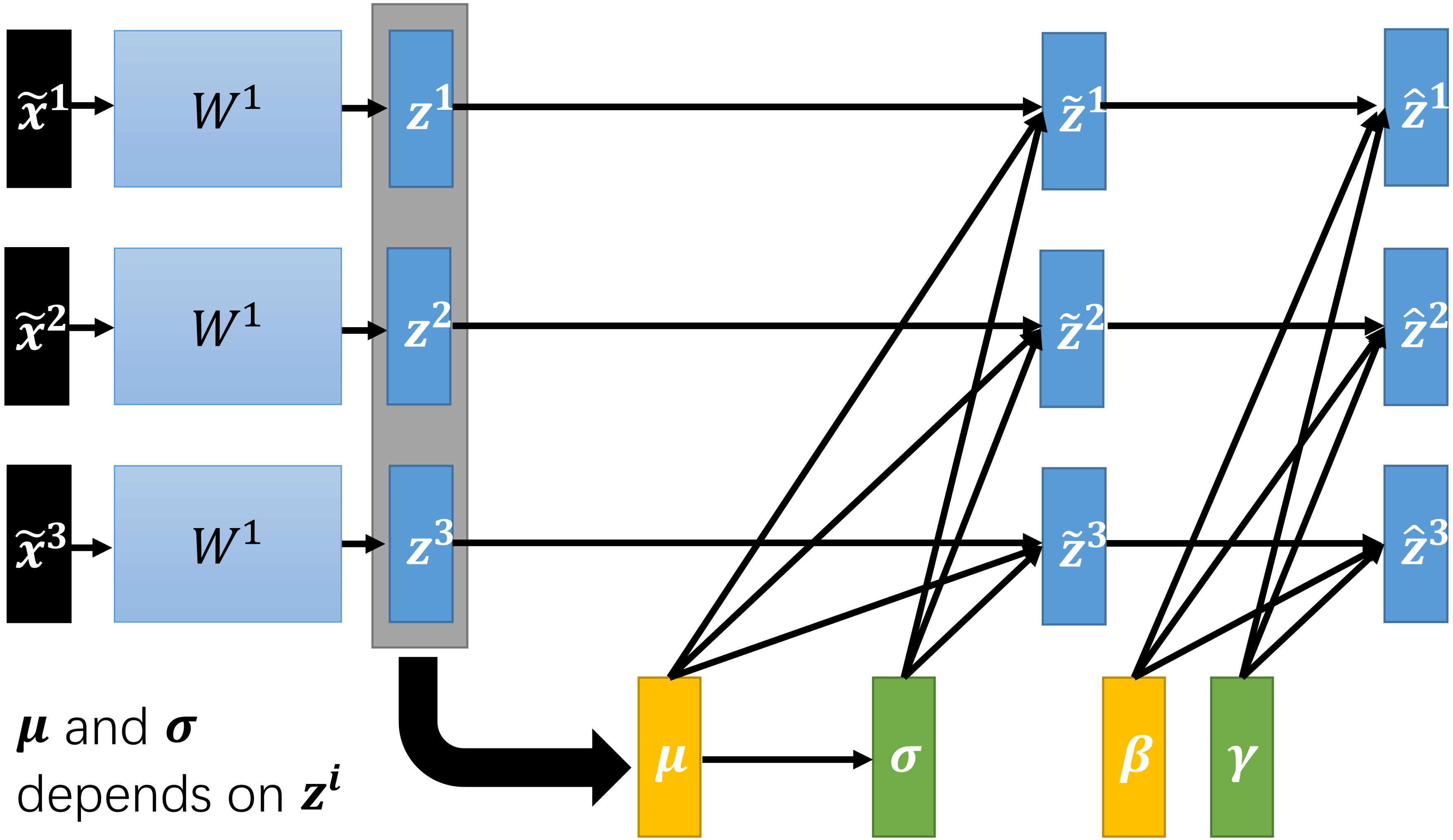
This is a large network!



Batch normalization

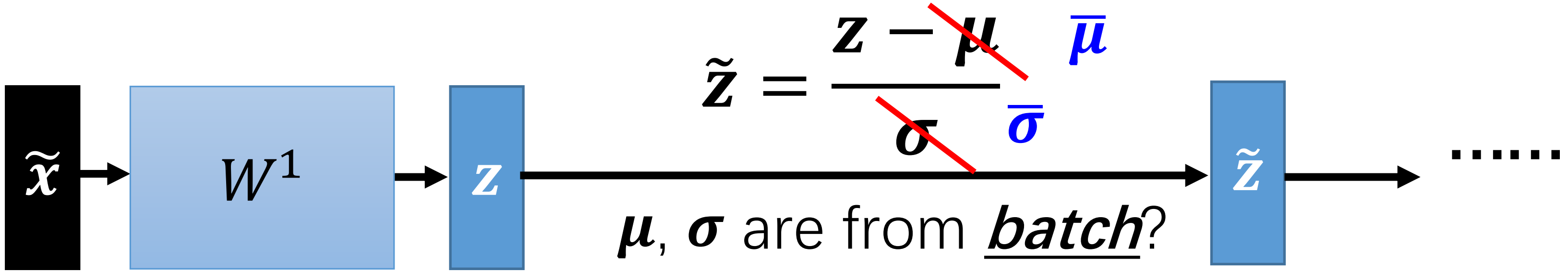
$$\tilde{\mathbf{z}}^i = \frac{\boldsymbol{z}^i - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

$$\hat{\mathbf{z}}^i = \boldsymbol{\gamma} \odot \tilde{\mathbf{z}}^i + \boldsymbol{\beta}$$



$\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ depends on $\boldsymbol{z}^i$

Batch normalization – Testing



We do not always have ***batch*** at testing stage.

Computing the moving average of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ of the batches during training.

$$\boldsymbol{\mu}^1 \quad \boldsymbol{\mu}^2 \quad \boldsymbol{\mu}^3 \quad \cdots\cdots \quad \boldsymbol{\mu}^t$$

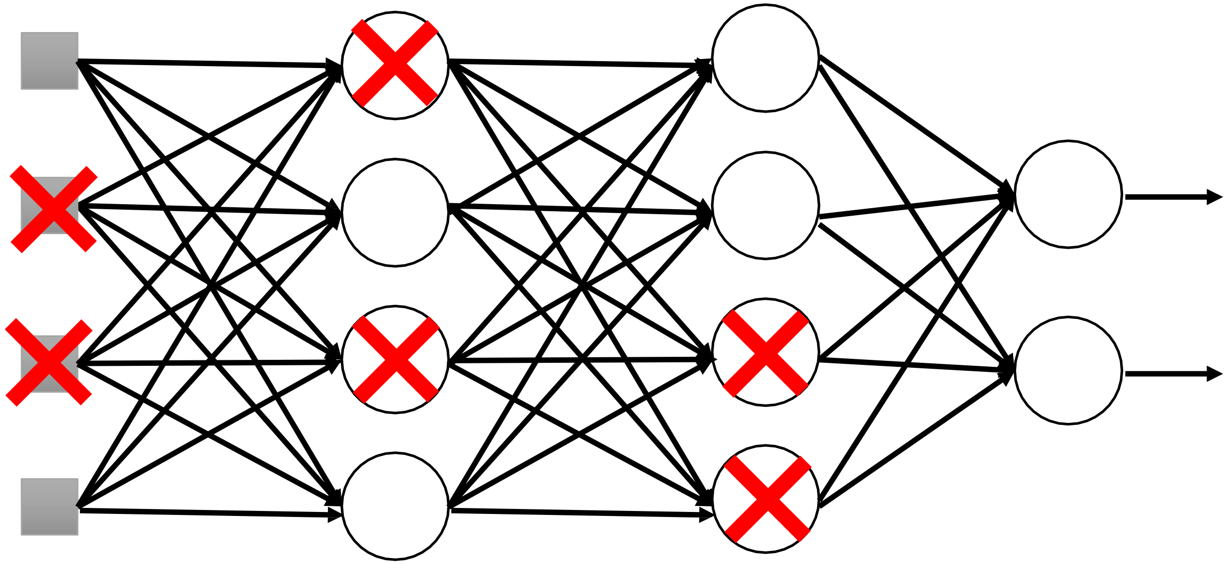
$$\overline{\boldsymbol{\mu}} \leftarrow p\overline{\boldsymbol{\mu}} + (1-p)\boldsymbol{\mu}^t$$

Tips for Training

5. Regularization

Dropout

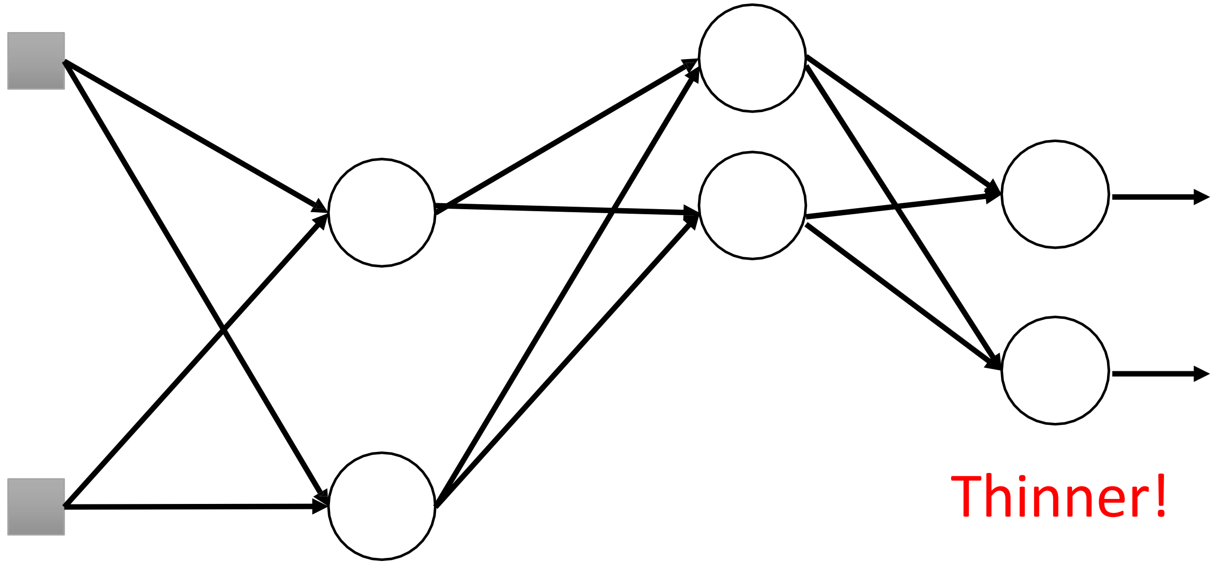
Training:



- **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout

Training:



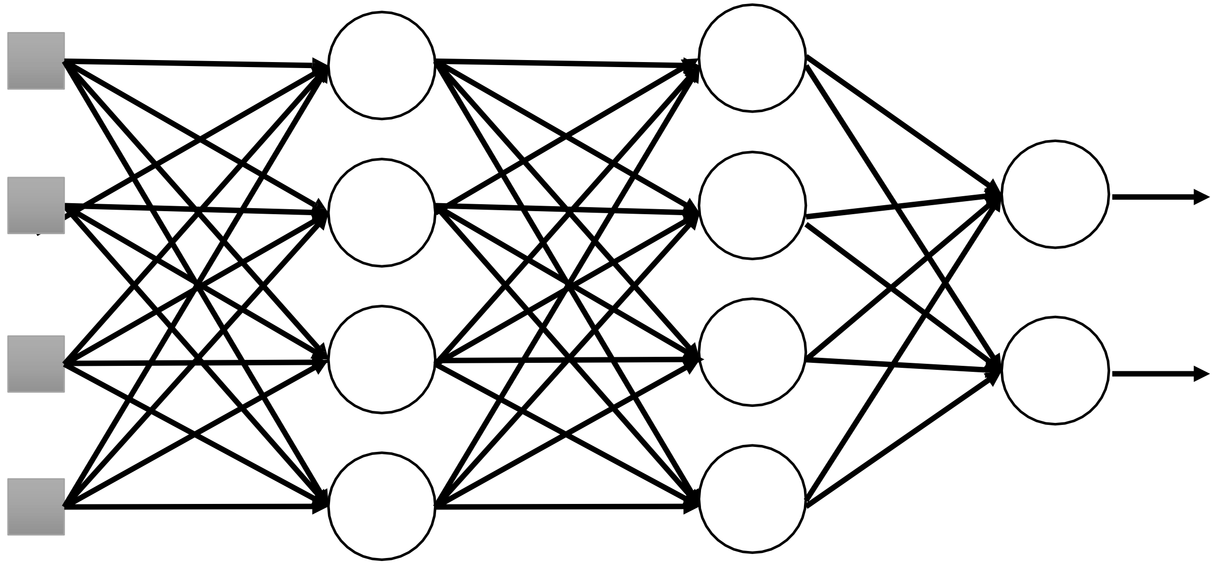
- **Each time before updating the parameters**
  - Each neuron has p% to dropout

    The structure of the network is changed.
  - Using the new network for training

For each batch, we resample the dropout neurons

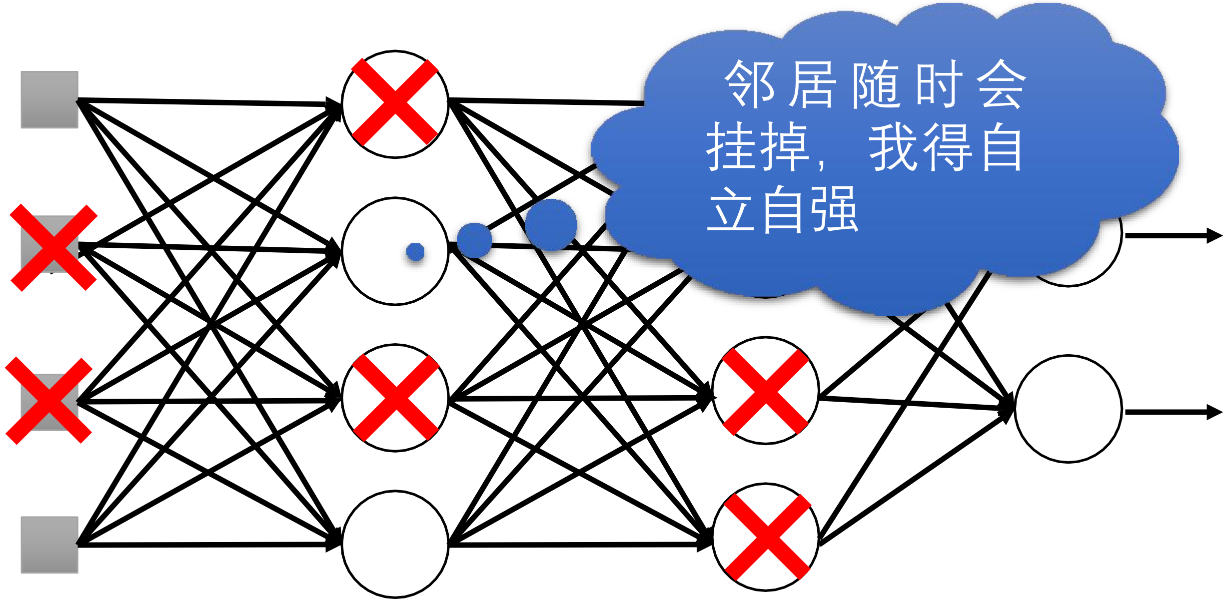
Dropout

Testing:



- **No dropout**
  - If the dropout rate at training is p%, all the weights times 1-p%
  - Assume that the dropout rate is 50%. If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Dropout - Intuitive Reason



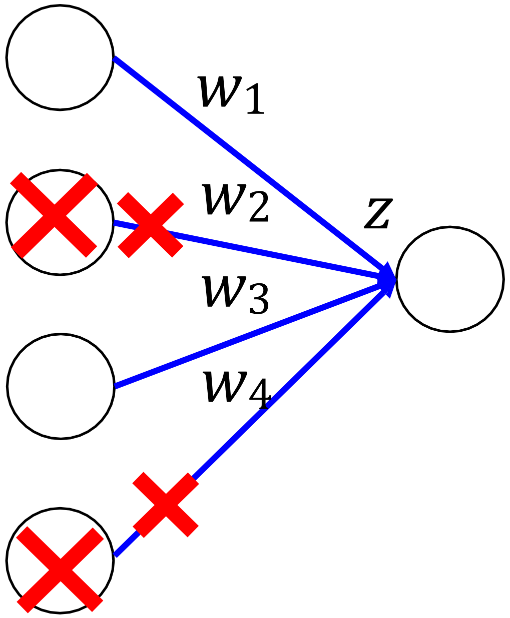
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

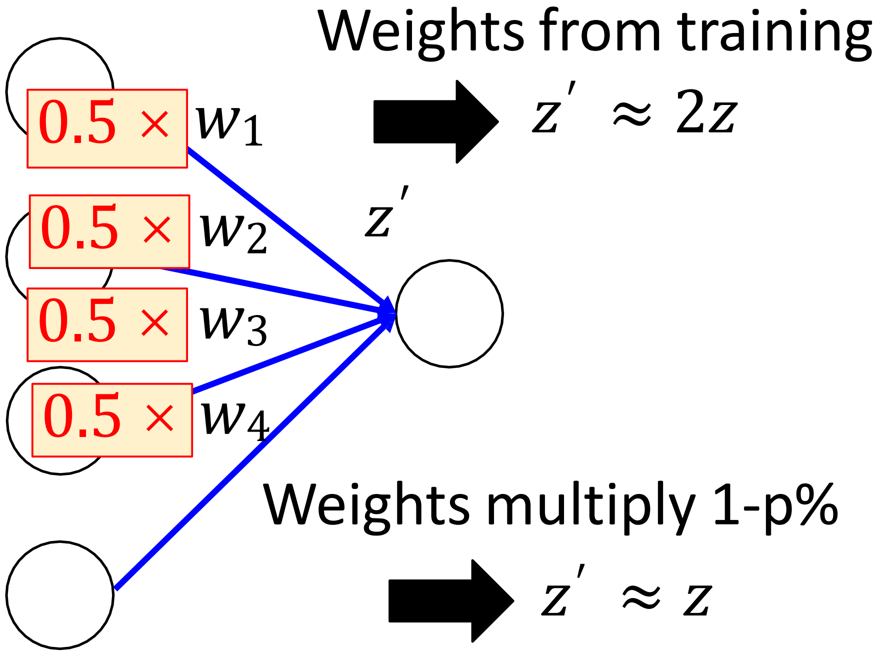
Training of Dropout

Assume dropout rate is 50%



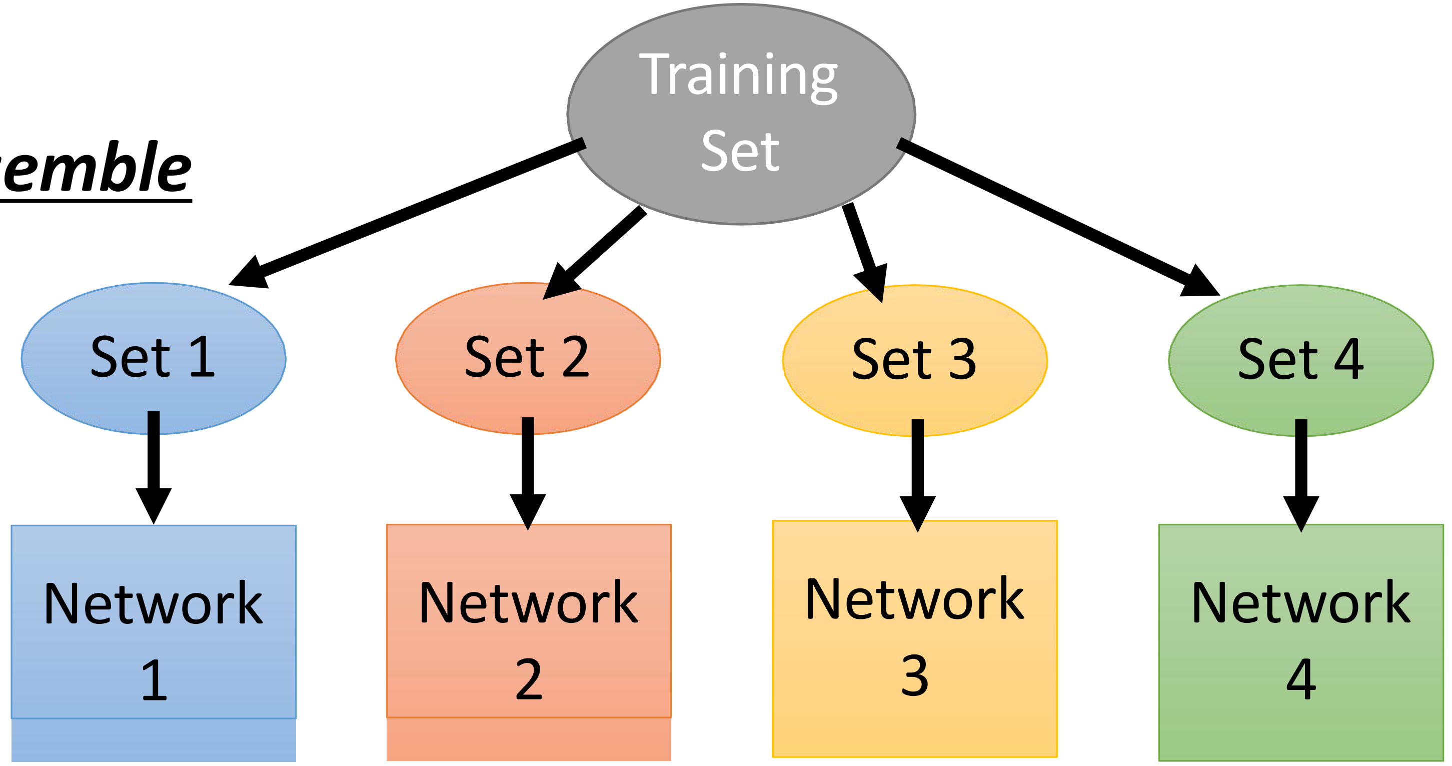
Testing of Dropout

No dropout



Dropout is a kind of ensemble.

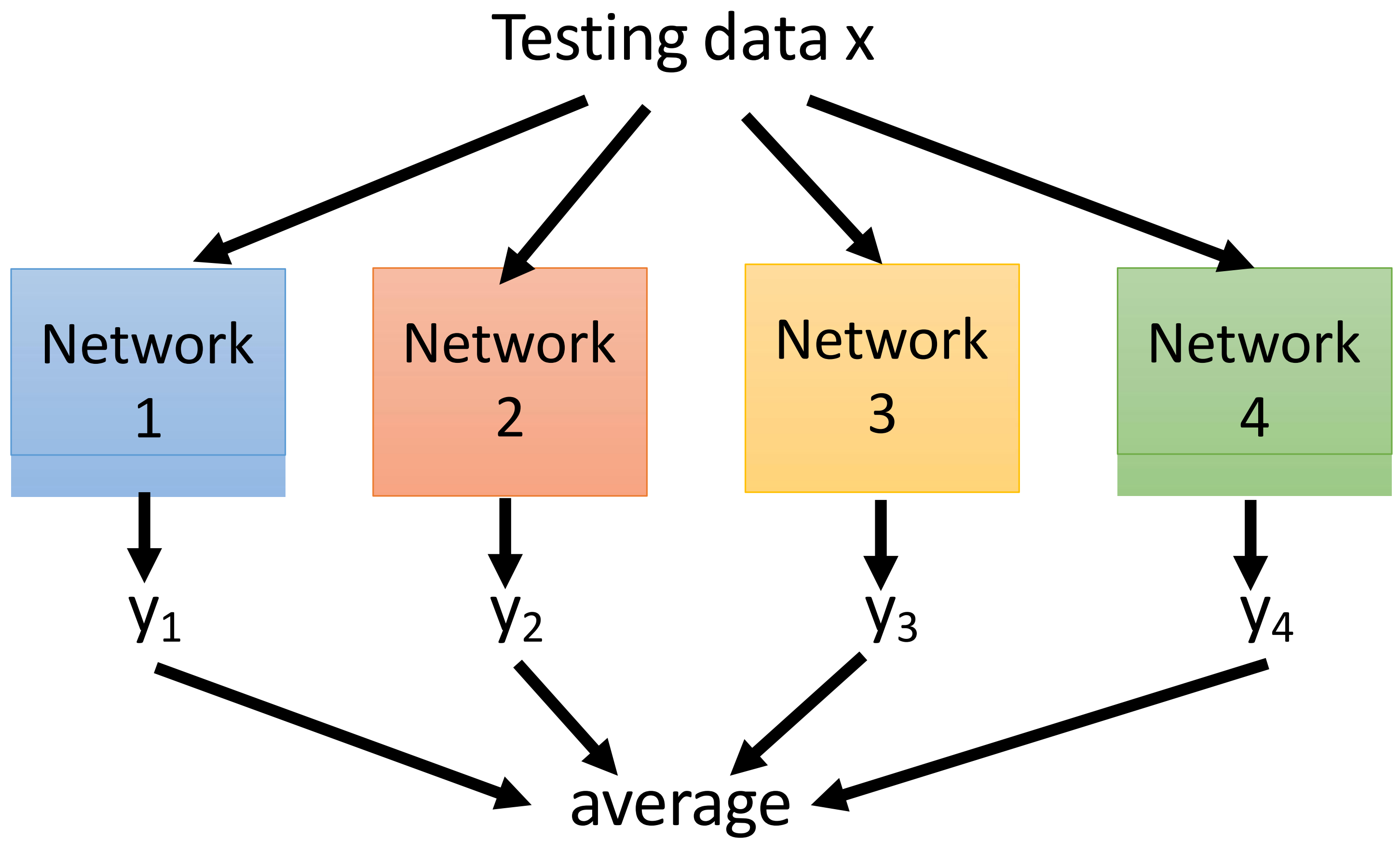
Ensemble



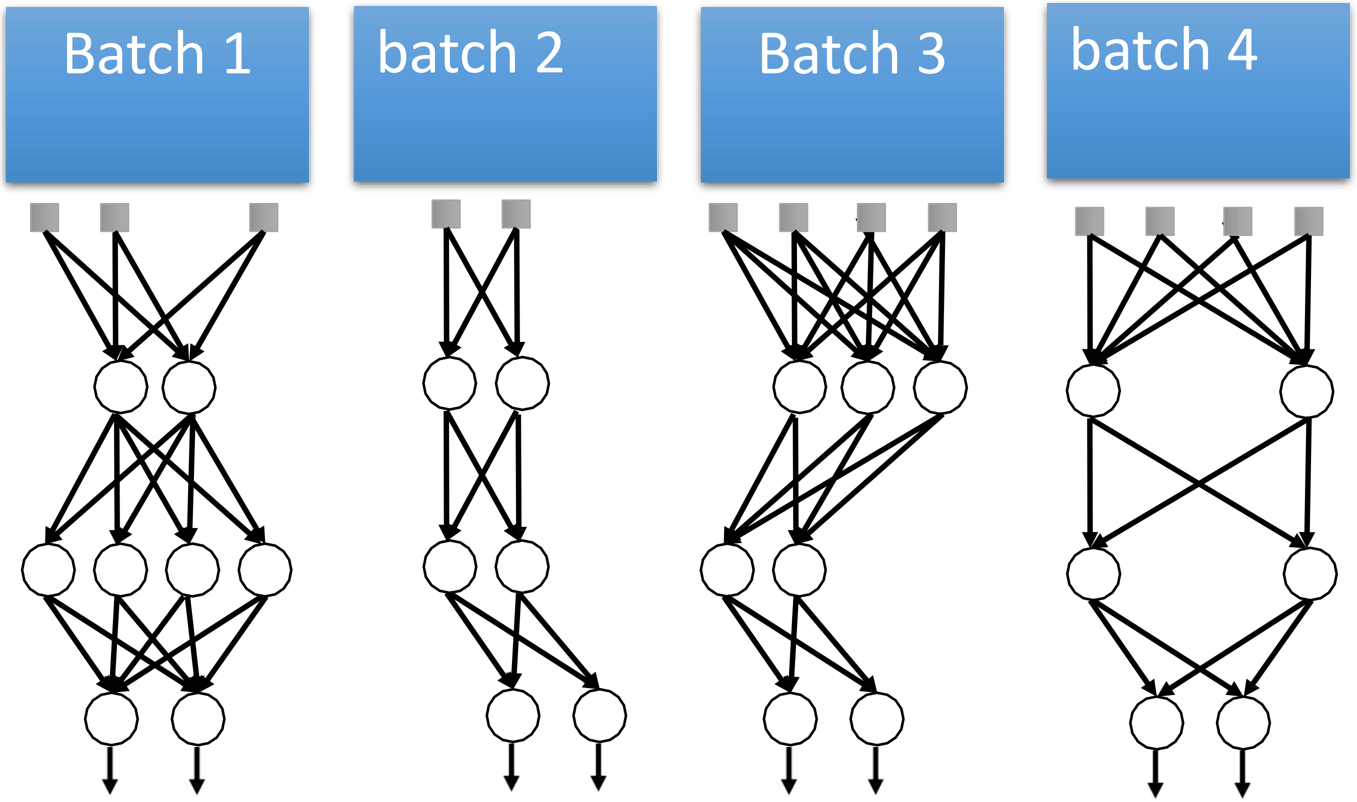
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

<u>Ensemble</u>



Dropout is a kind of ensemble.



Training of Dropout

- M neurons

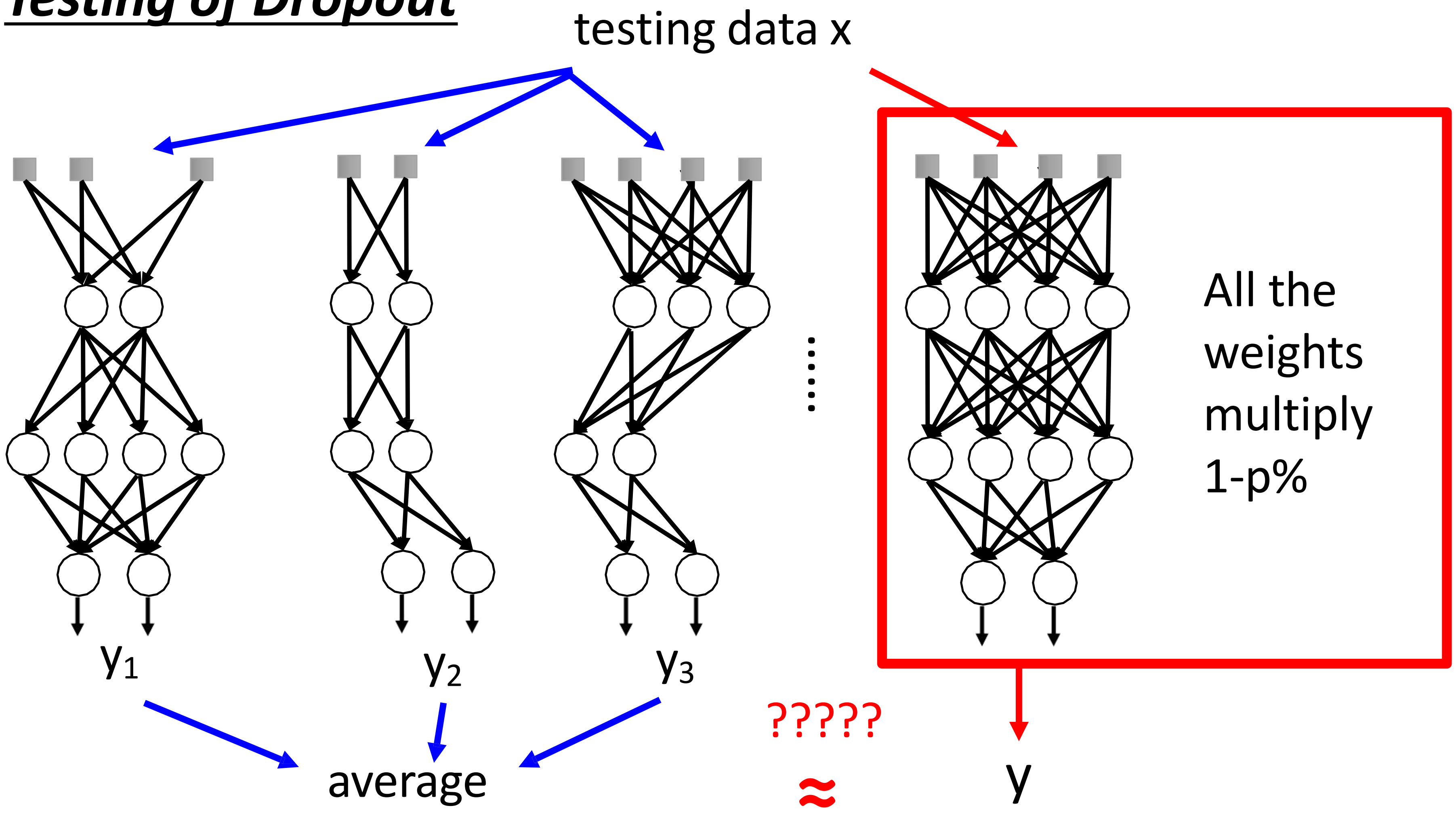
- 2^M possible networks

➢ Using one mini-batch to train one network

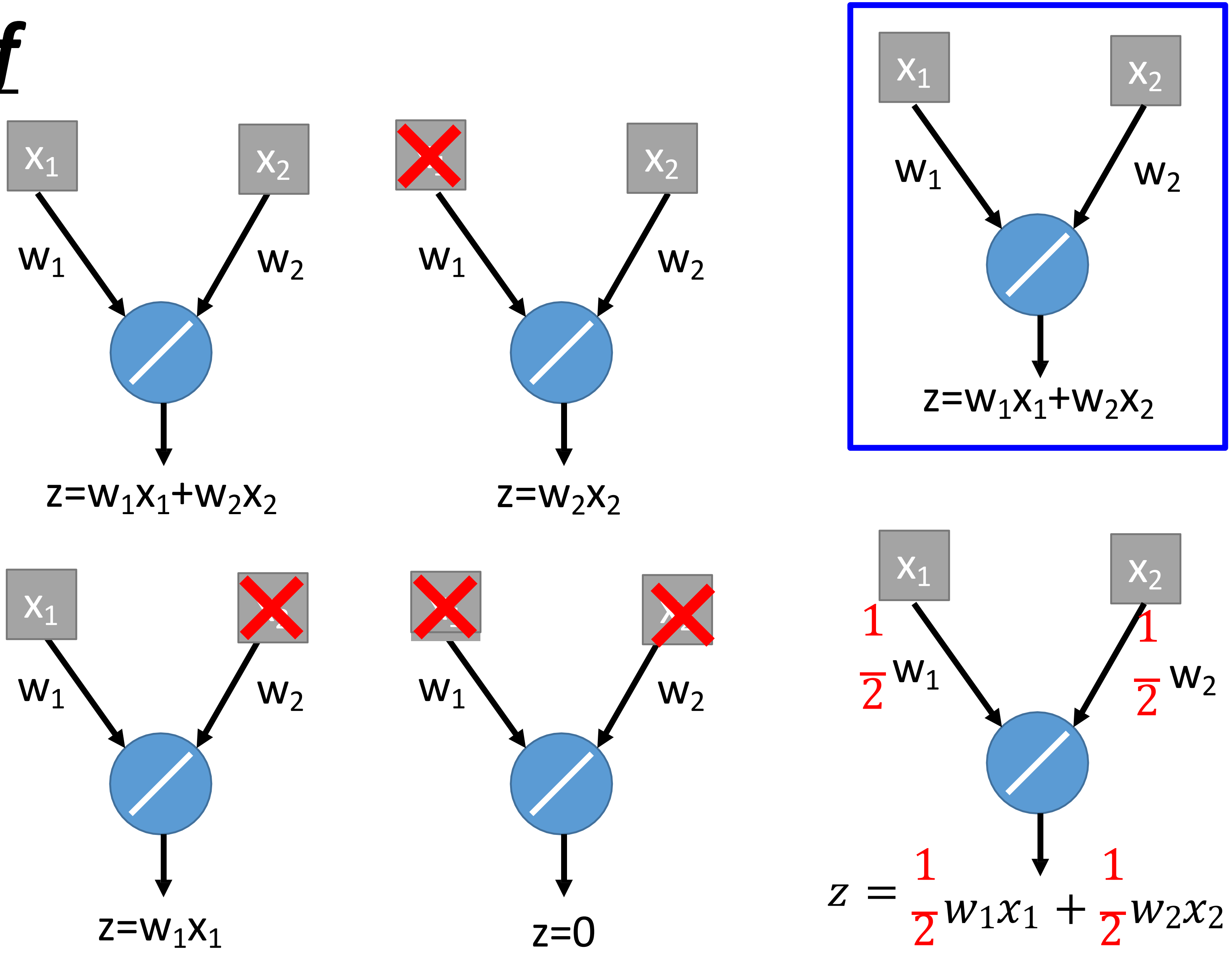
➢ Some parameters in the network are shared

Dropout is a kind of ensemble.

Testing of Dropout

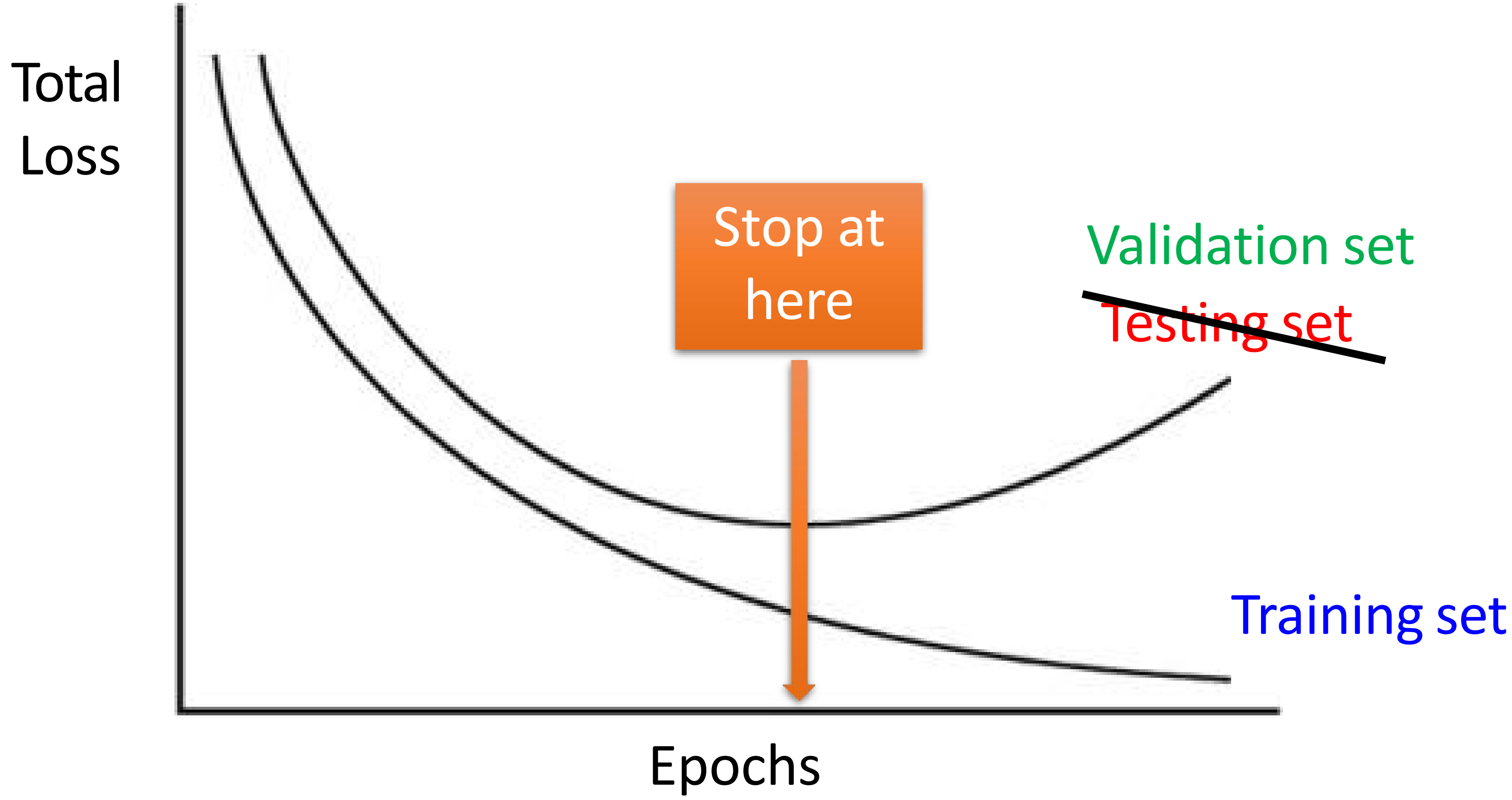


Testing of Dropout



$$z = \frac{1}{2} w_1 x_1 + \frac{1}{2} w_2 x_2$$

Early Stopping



Keras: http://keras.io/getting-started/faq/#how-can-i-interrupt-training-when-the-validation-loss-isnt-decreasing-anymore

Why Deep?

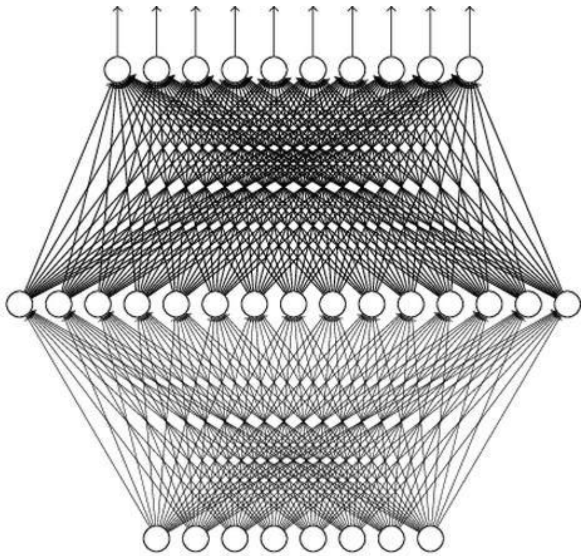
Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^M$$

Can be realized by a network with one hidden layer

(given **enough** hidden neurons)



Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

Why "Deep" neural network not "Fat" neural network?

Deeper is Better?

Layer X Size	Word Error Rate (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

Not surprised, more parameters, better performance

1 X 3772	22.5
1 X 4634	22.6
1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.