LECTURE 15

WINTER 2021
APPLIED MACHINE LEARNING
CIHANG XIE

SLIDE CREDIT:
NARGES NOROUZI
HUNG-YI LEE

THREE STEPS FOR DEEP LEARNING

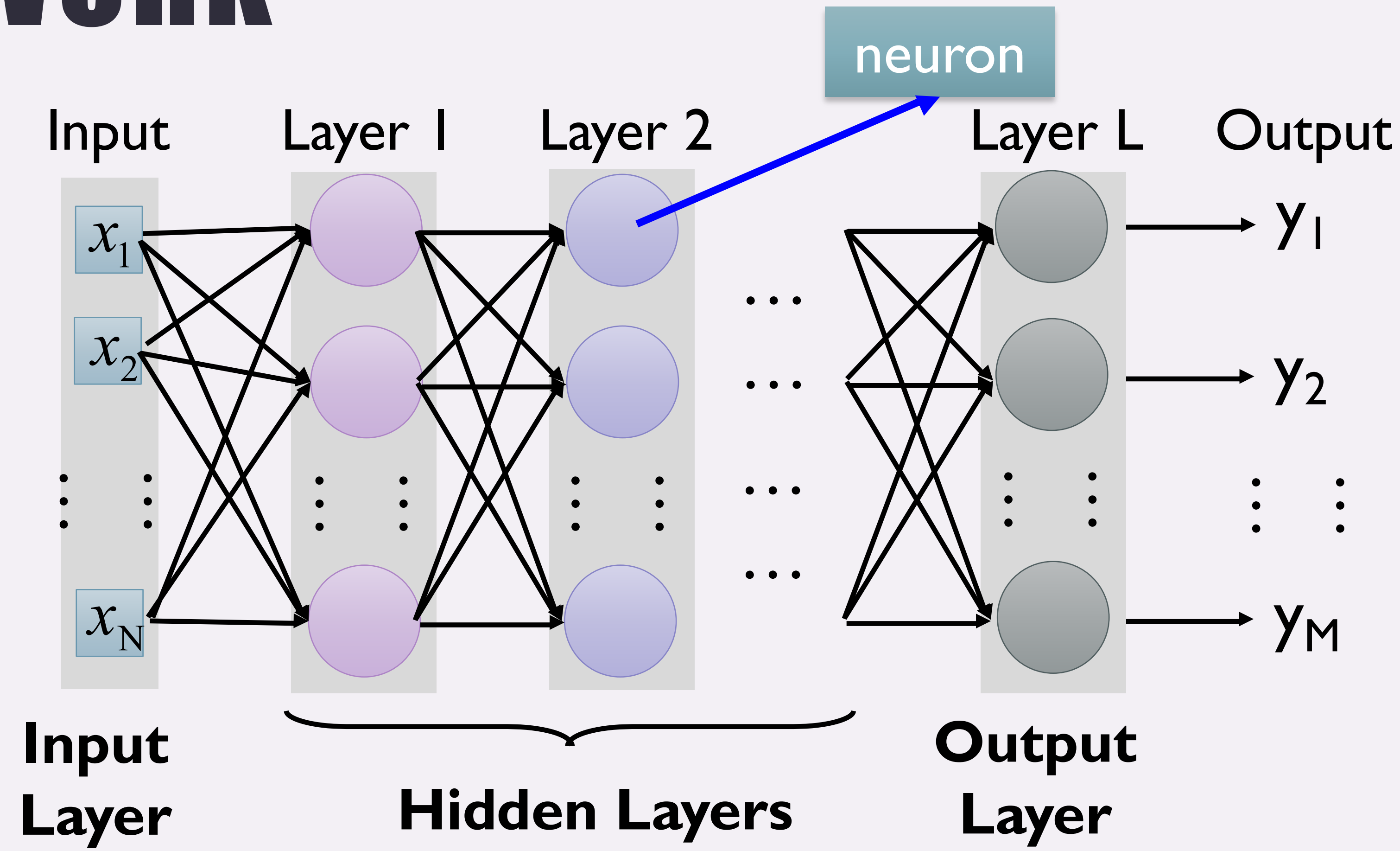
Step 1: define a set of function

Neural Network

Step 2: goodness of function

Step 3: pick the best function

FULLY CONNECT FEEDFORWARD NETWORK



Deep means many hidden layers

THREE STEPS FOR DEEP LEARNING

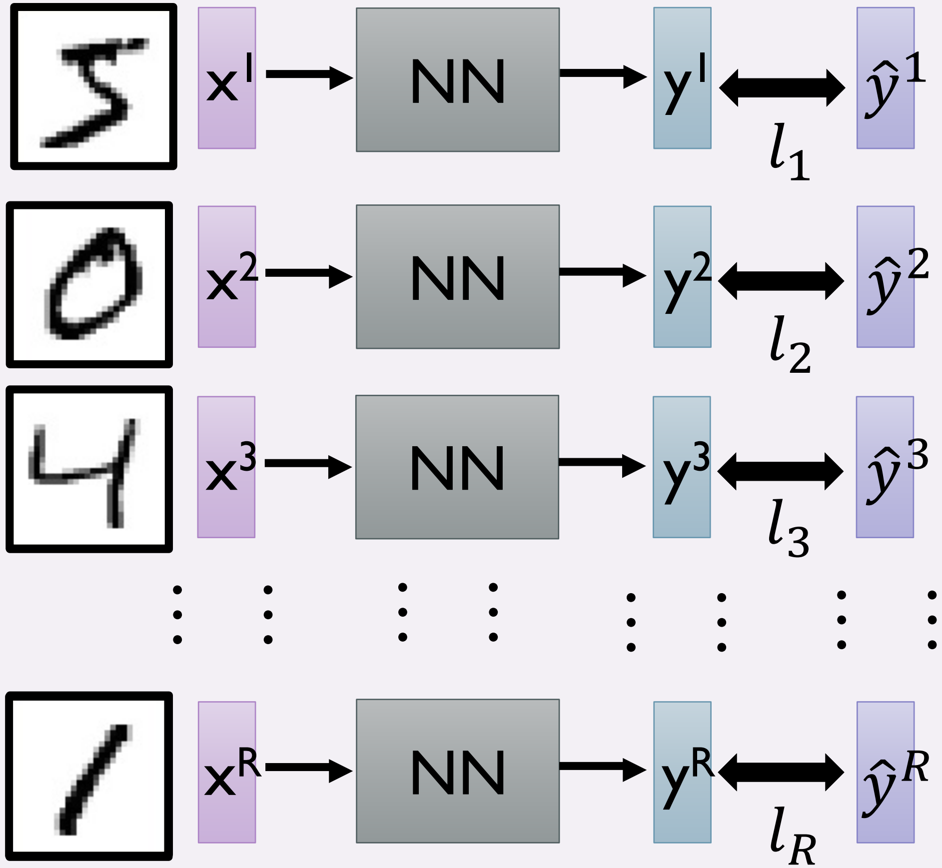
Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

TOTAL LOSS

For all training data …



Total Loss:

$$\boxed{L = \sum_{r=1}^{R} l_r}$$

As small as possible

Find ***a function in function set*** that minimizes total loss L

Find ***the network parameters θ^**** that minimize total loss L

THREE STEPS FOR DEEP LEARNING

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

GRADIENT DESCENT

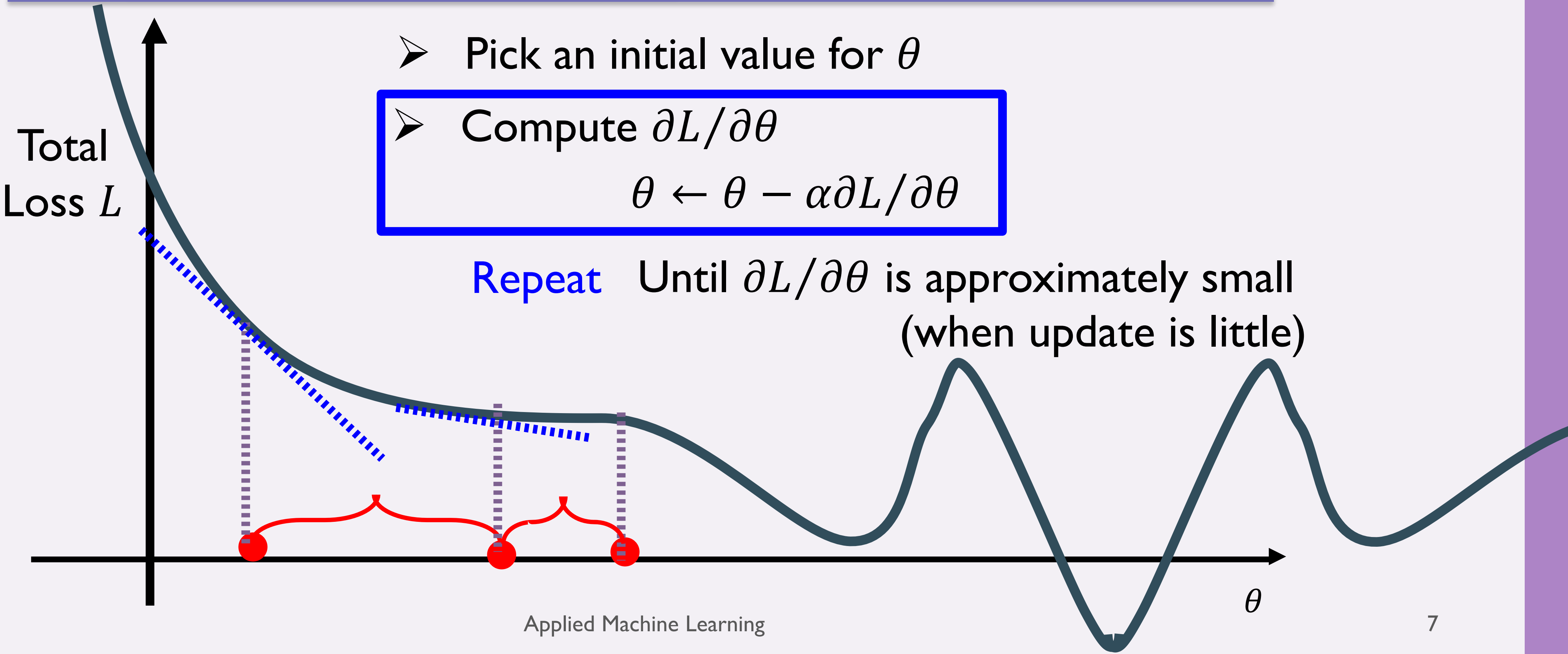
Network parameters $\theta = \{\theta_1, \theta_2, \cdots\}$

Find ***network parameters*** $\boldsymbol{\theta^*}$ that minimize total loss L

- Pick an initial value for θ
- Compute $\partial L / \partial \theta$

$$\boxed{\theta \leftarrow \theta - \alpha \partial L / \partial \theta}$$

Repeat Until $\partial L / \partial \theta$ is approximately small (when update is little)



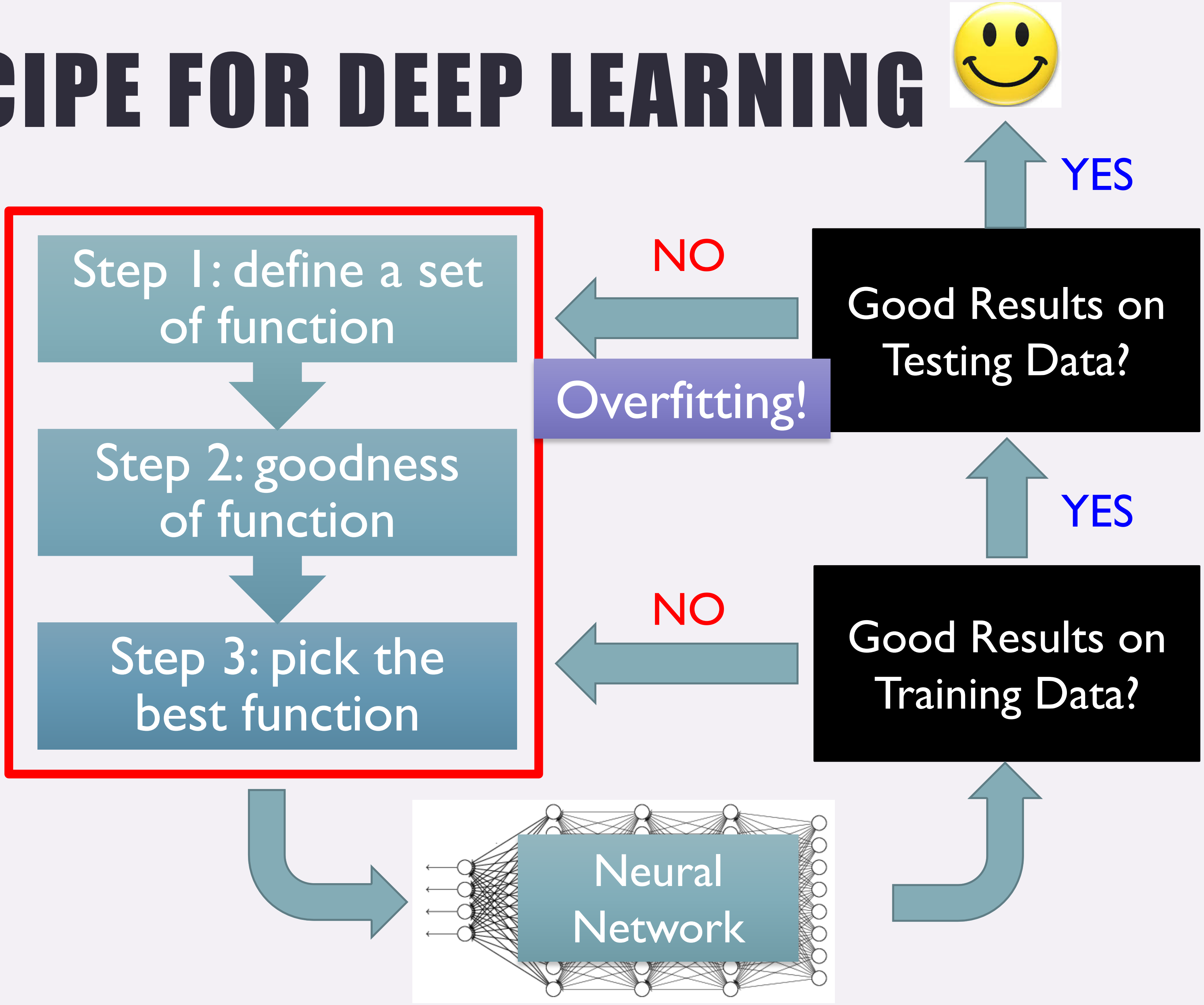
OUTLINE

Introduction of Deep Learning

“Hello World” for Deep Learning

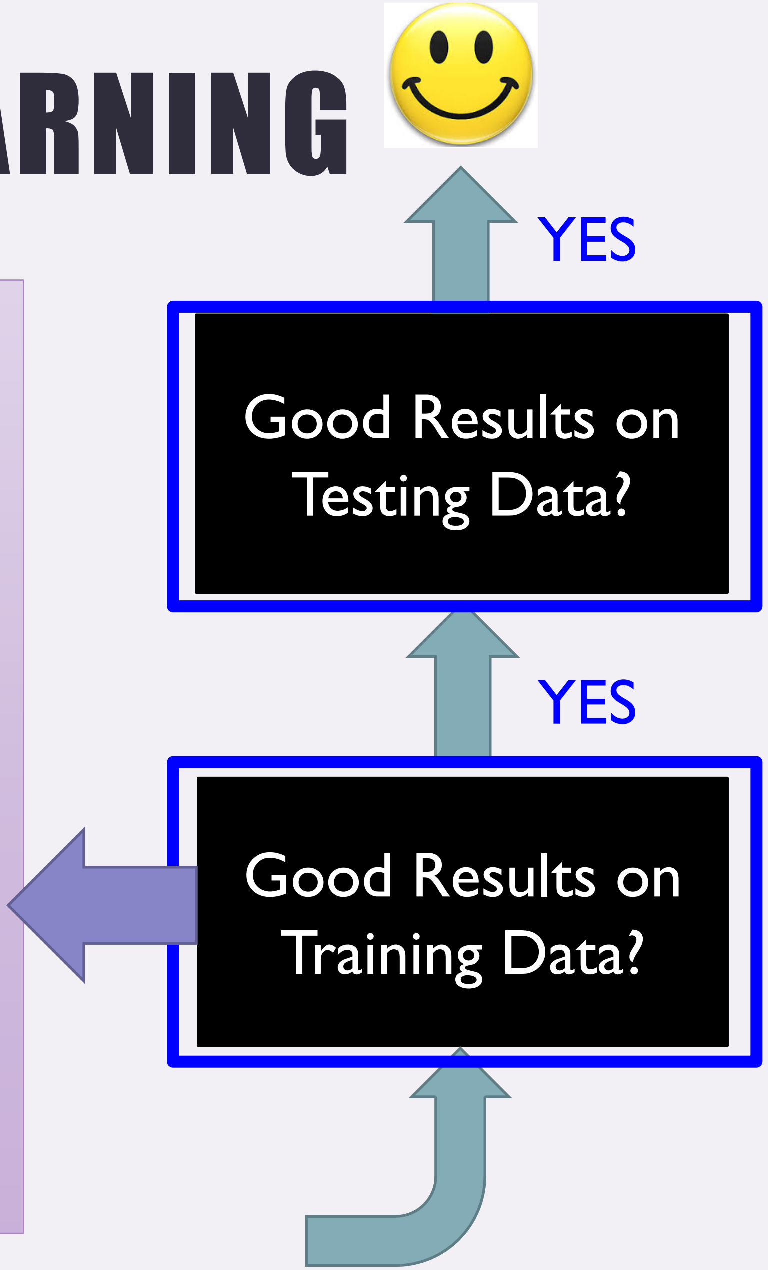
Tips for Deep Learning

RECIPE FOR DEEP LEARNING



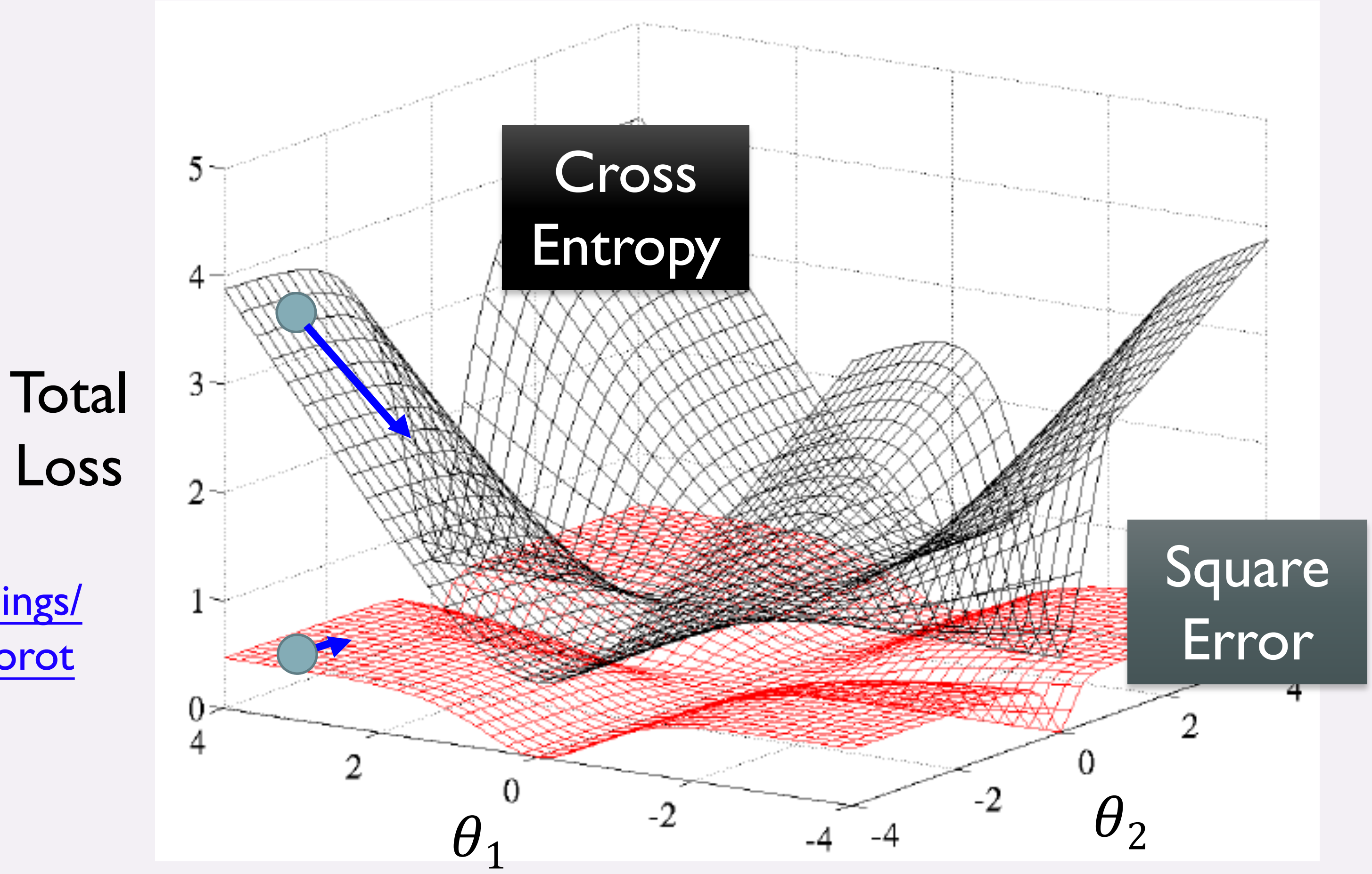
RECIPE FOR DEEP LEARNING

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum



CHOOSING PROPER LOSS

When using softmax output layer, choose cross entropy



http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

RECIPE FOR DEEP LEARNING

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

Good Results on Testing Data?

YES

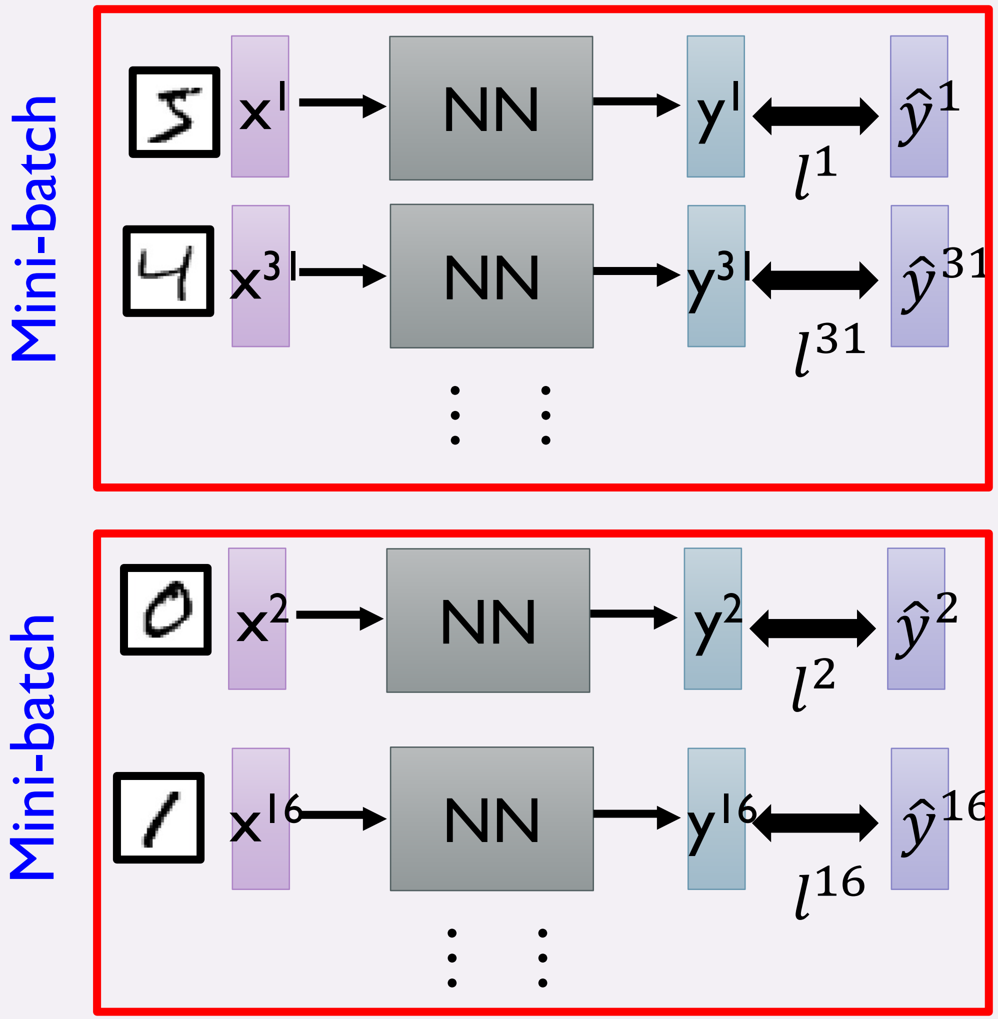
Good Results on Training Data?

YES

```
model.fit(x_train, y_train, epochs = 200, batch_size = 100)
```

MINI-BATCH

We do not really minimize total loss!



- Randomly initialize network parameters
- Pick the 1st batch
  $L' = l^1 + l^{31} + \cdots$
  Update parameters once
- Pick the 2nd batch
  $L'' = l^2 + l^{16} + \cdots$
  Update parameters once

  ⋮
- Until all mini-batches have been picked

one epoch

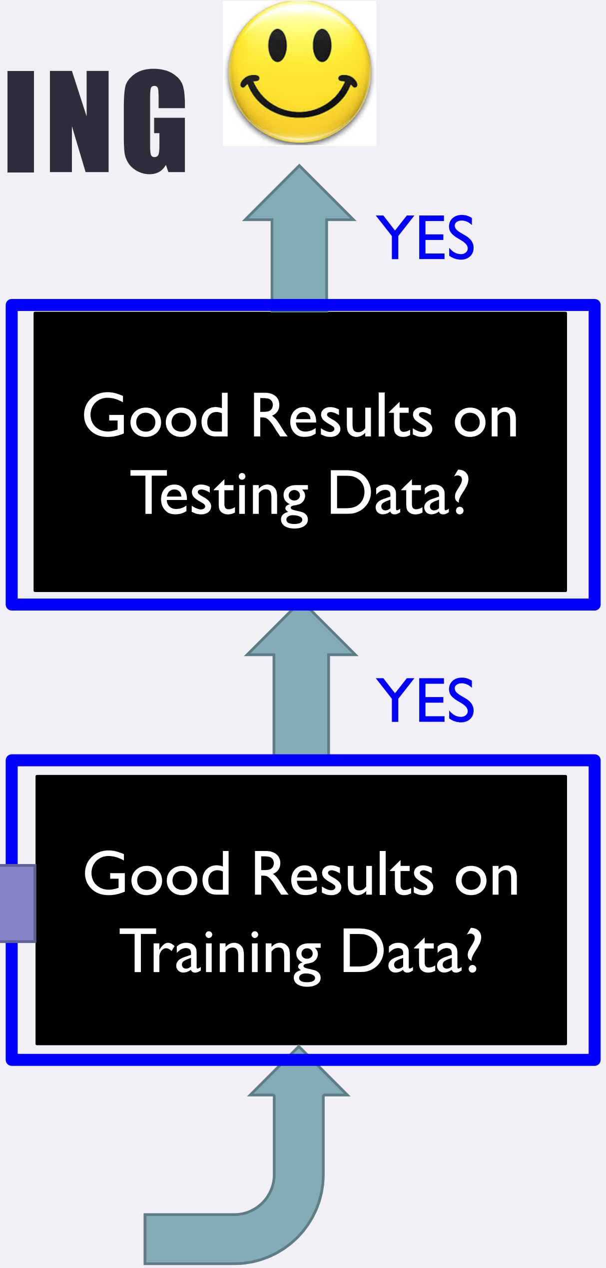
Repeat the above process

TODAY

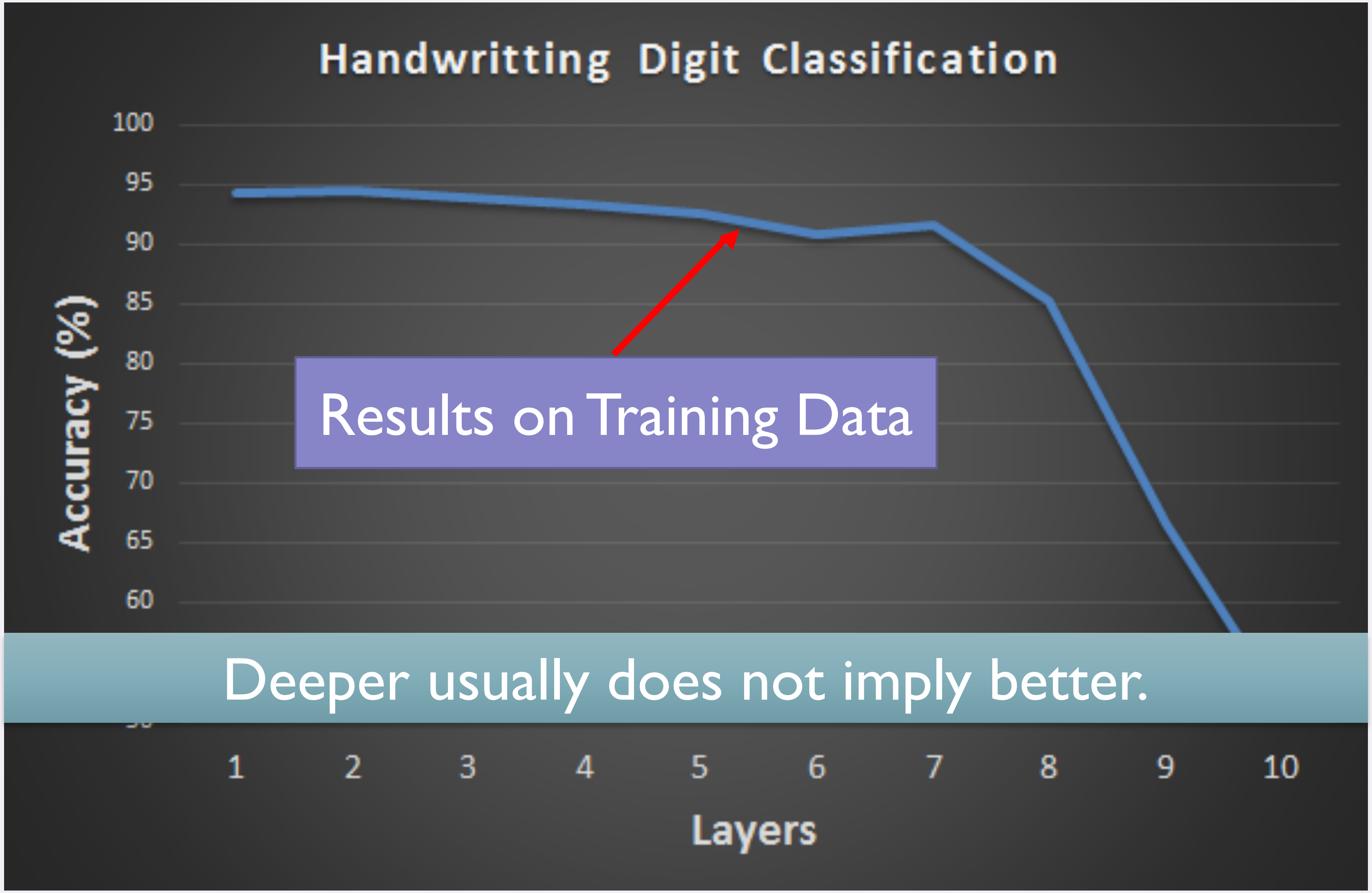
- More Deep Learning Topics
  - Another activation function – ReLu
  - Adaptive learning rate
  - Momentum

RECIPE FOR DEEP LEARNING

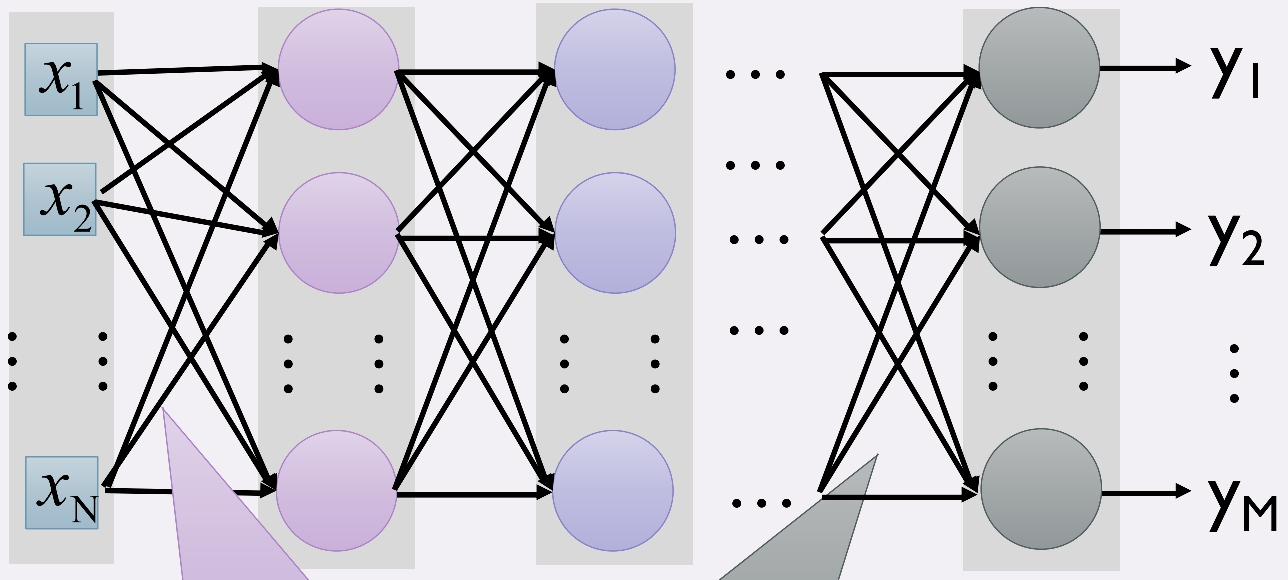
- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum



HARD TO GET THE POWER OF DEEP …



VANISHING GRADIENT PROBLEM



Smaller gradients

Learn very slow

Almost random

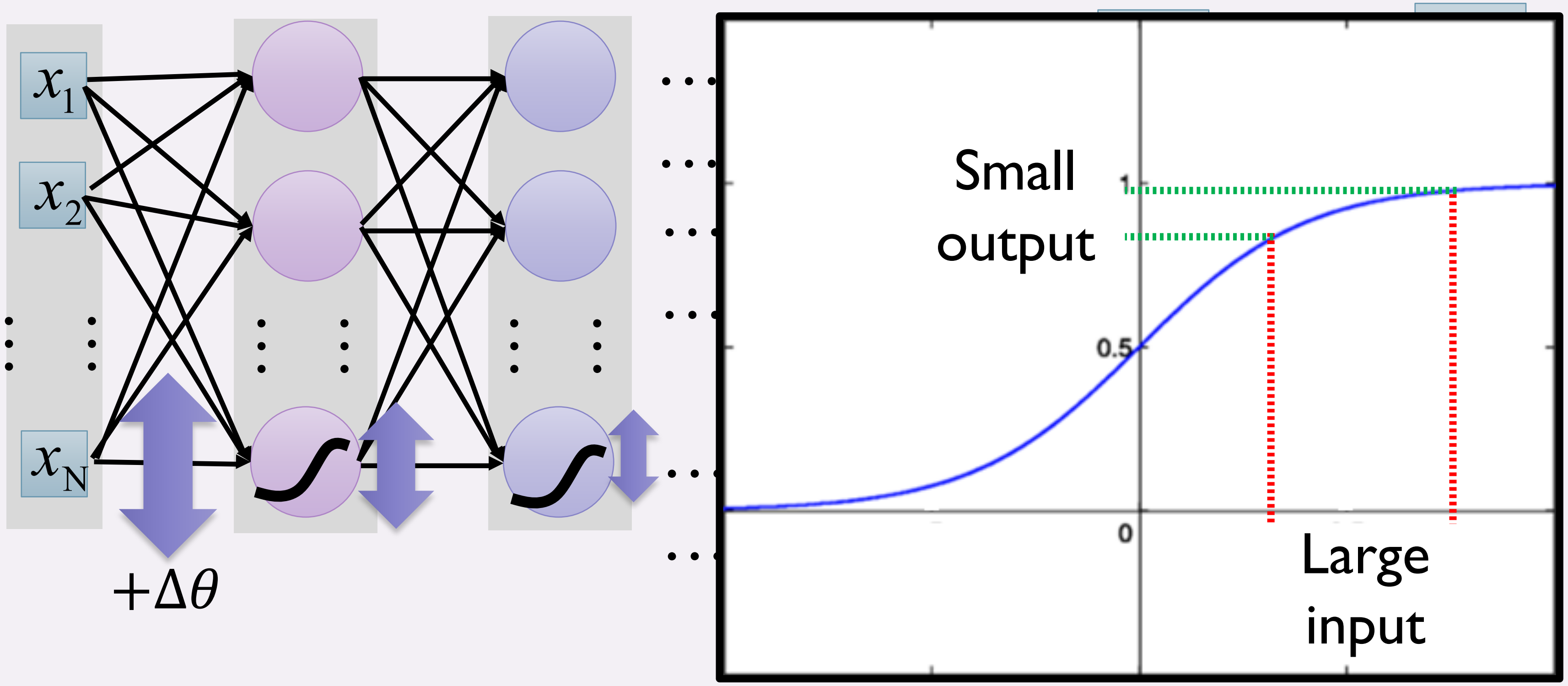
Larger gradients

Learn very fast

Already converge

based on random!?

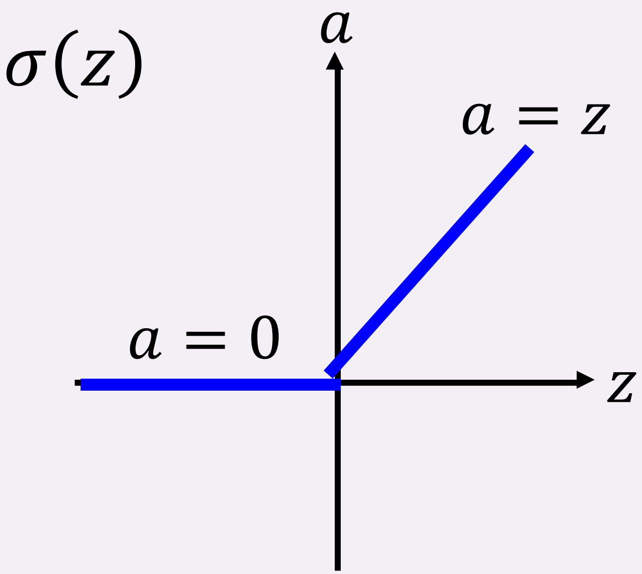
VANISHING GRADIENT PROBLEM



Intuitive way to compute the derivatives … $\frac{\partial l}{\partial \theta} =? \frac{\Delta l}{\Delta \theta}$

RELU

- Rectified Linear Unit (ReLU)

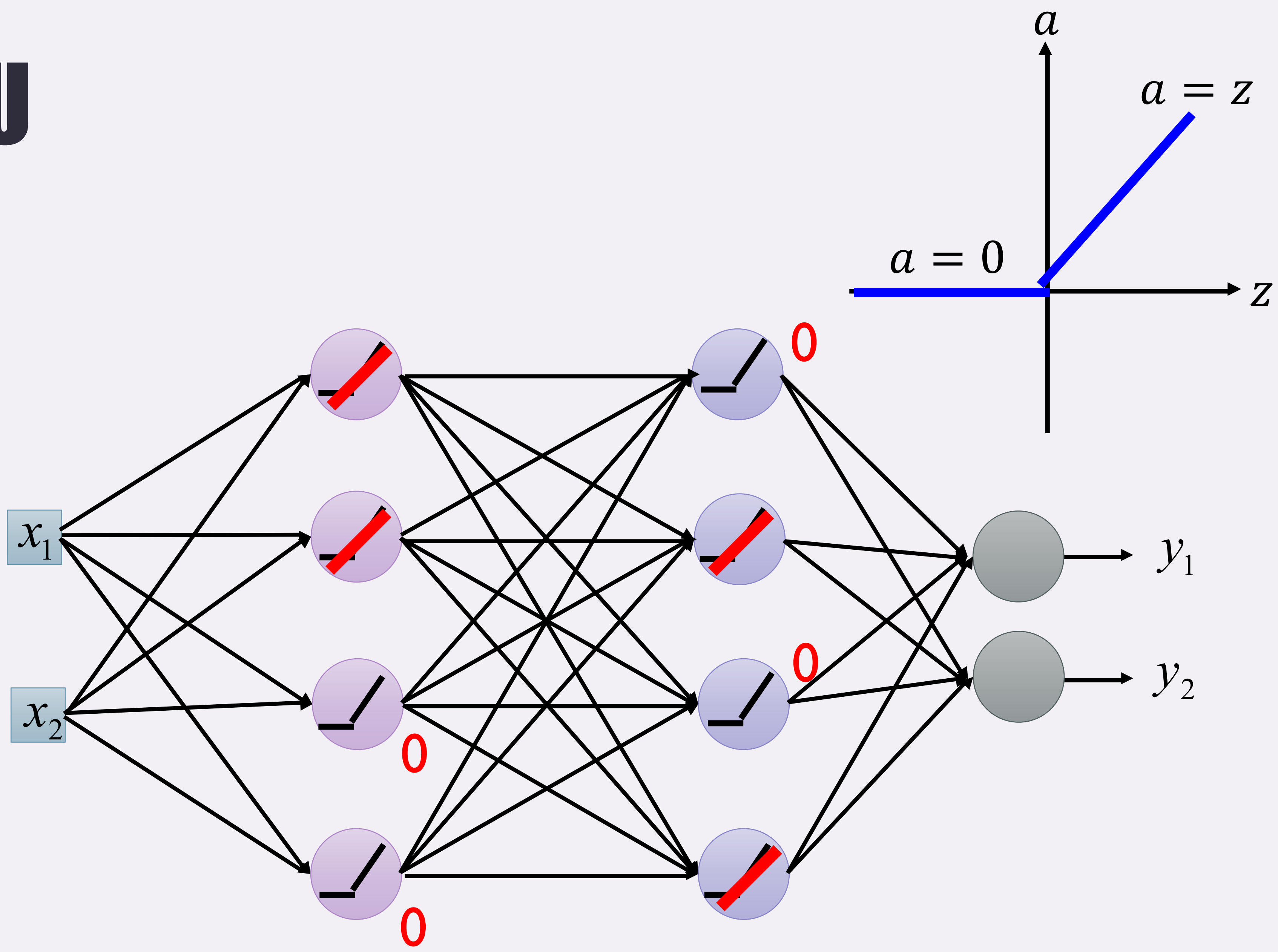


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

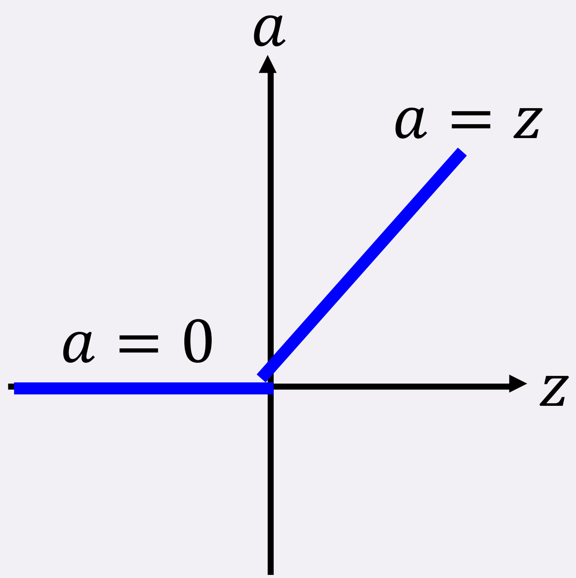
1. Fast to compute
2. Biological reason
3. $\boxed{\text{Vanishing gradient problem}}$

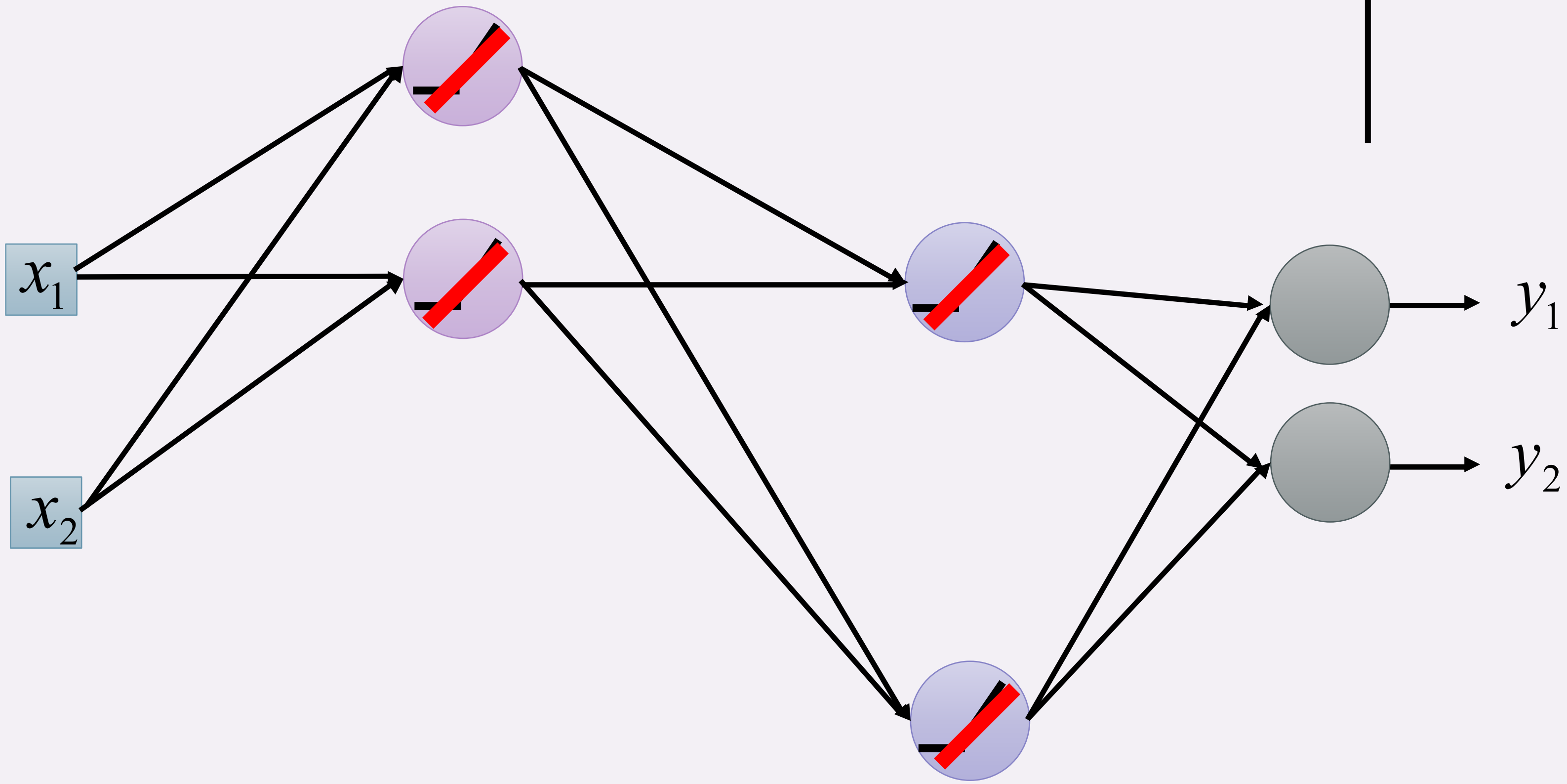
RELU



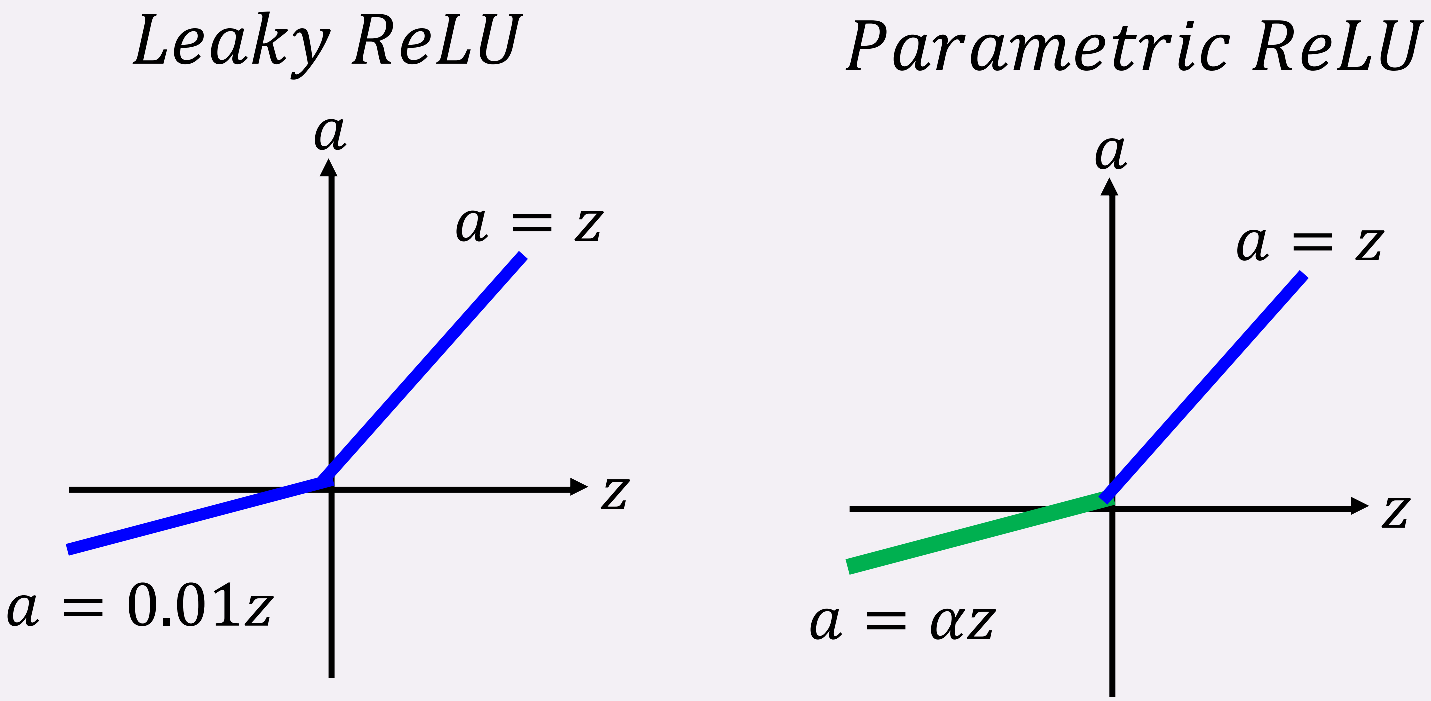
RELU

A Thinner linear network





RELU - VARIANT



α also learned by gradient descent

RELU – "SMOOTH" VARIANT

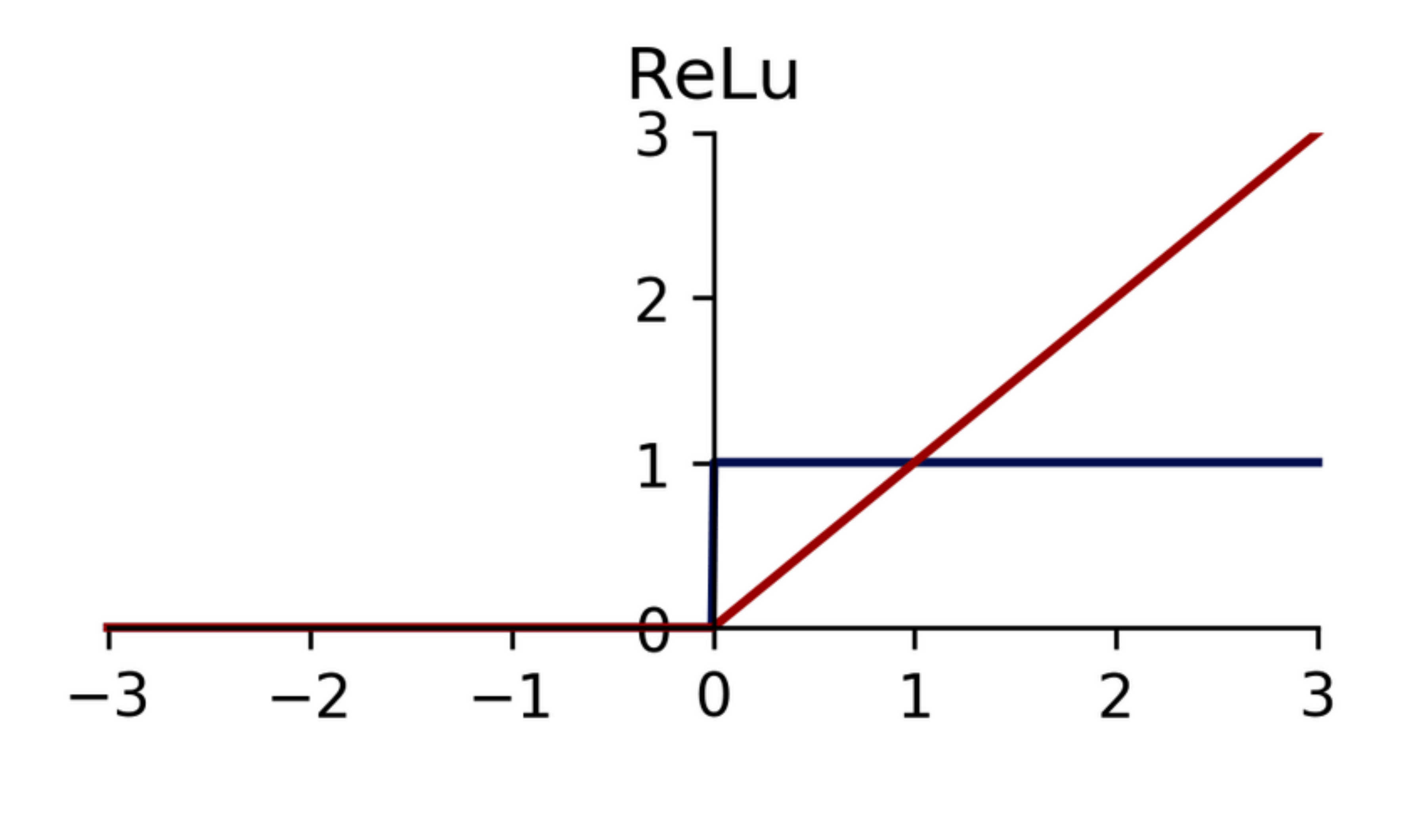


Image credit: https://arxiv.org/pdf/2006.13311.pdf

RELU – "SMOOTH" VARIANT

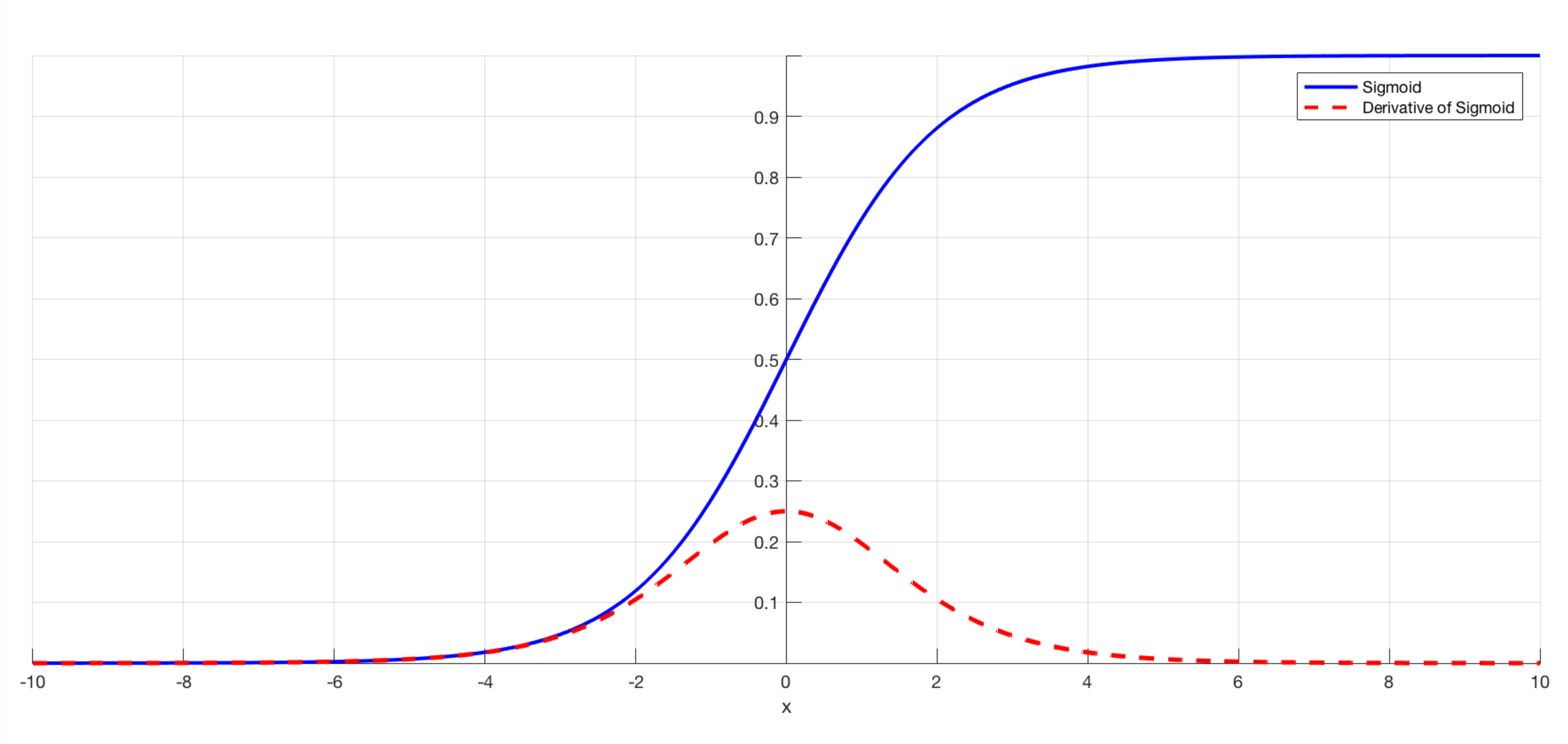


Image credit: https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e

RELU – “SMOOTH” VARIANT

- SILU [Elfwing et al 2018; Hendrycks et al 2017; Ramachandran et al 2017]

$$\text{SILU}(x) = x\ sigmoid(x)$$

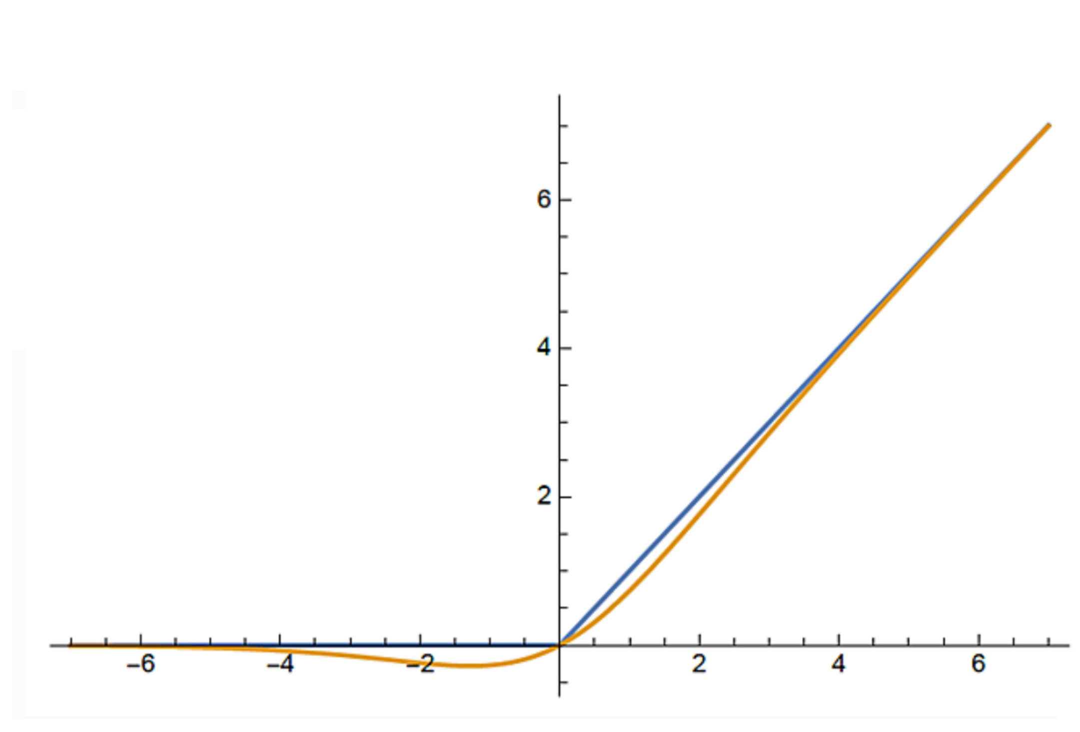


Image credit: https://arxiv.org/pdf/1805.08878.pdf

RELU – "SMOOTH" VARIANT

- SmoothReLU

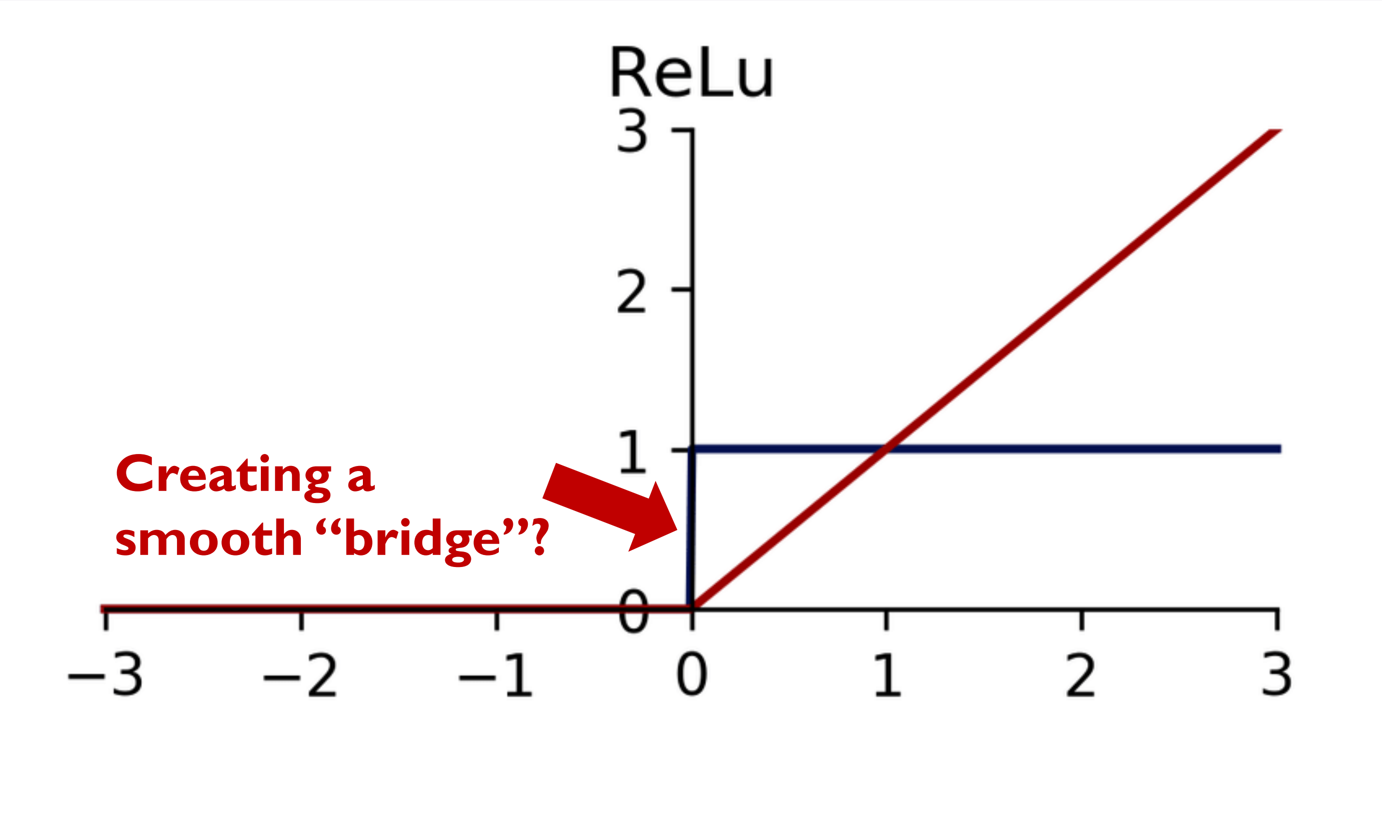


Image credit: https://arxiv.org/pdf/2006.13311.pdf

RECIPE FOR DEEP LEARNING

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

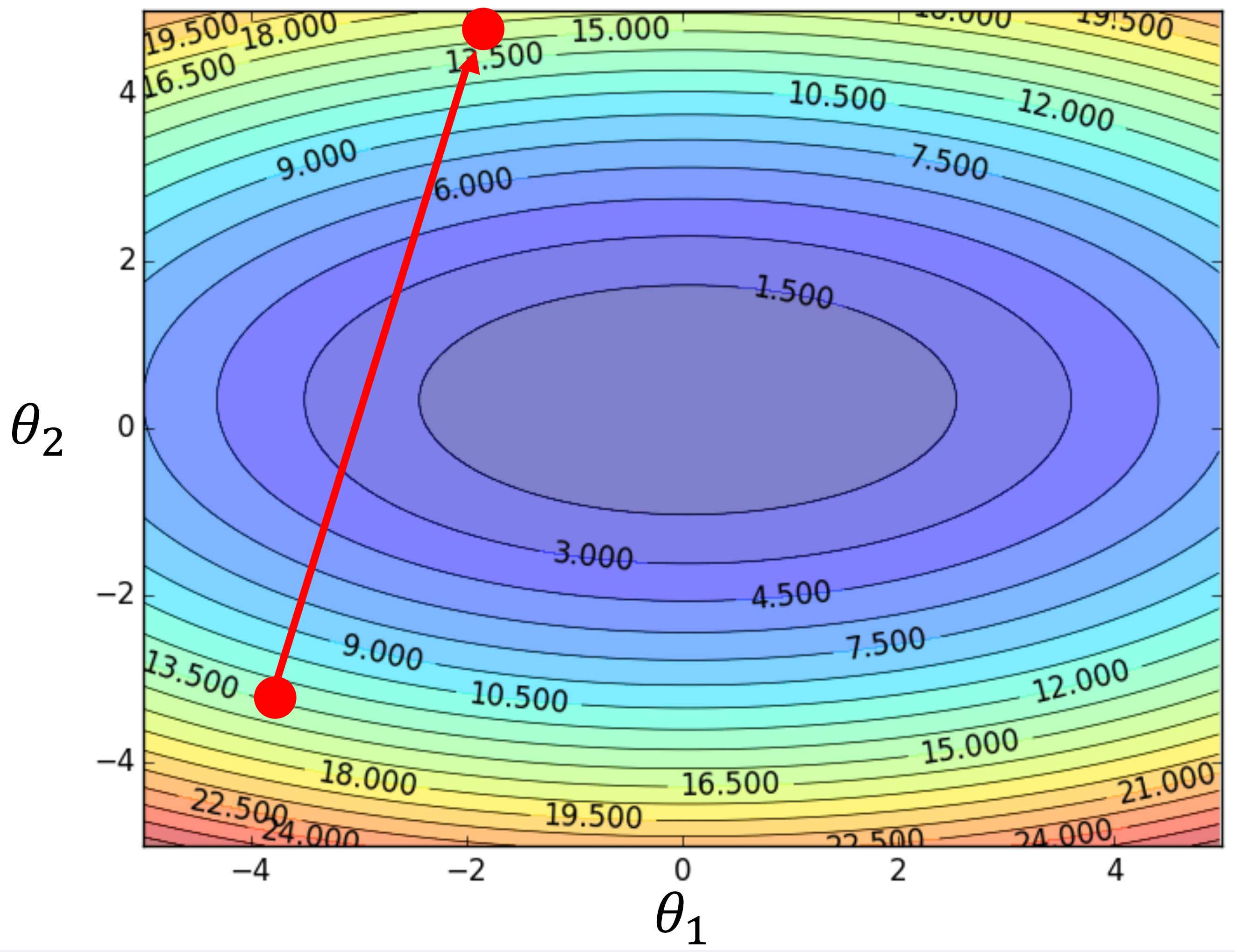
YES

Good Results on Testing Data?

YES

Good Results on Training Data?

LEARNING RATES

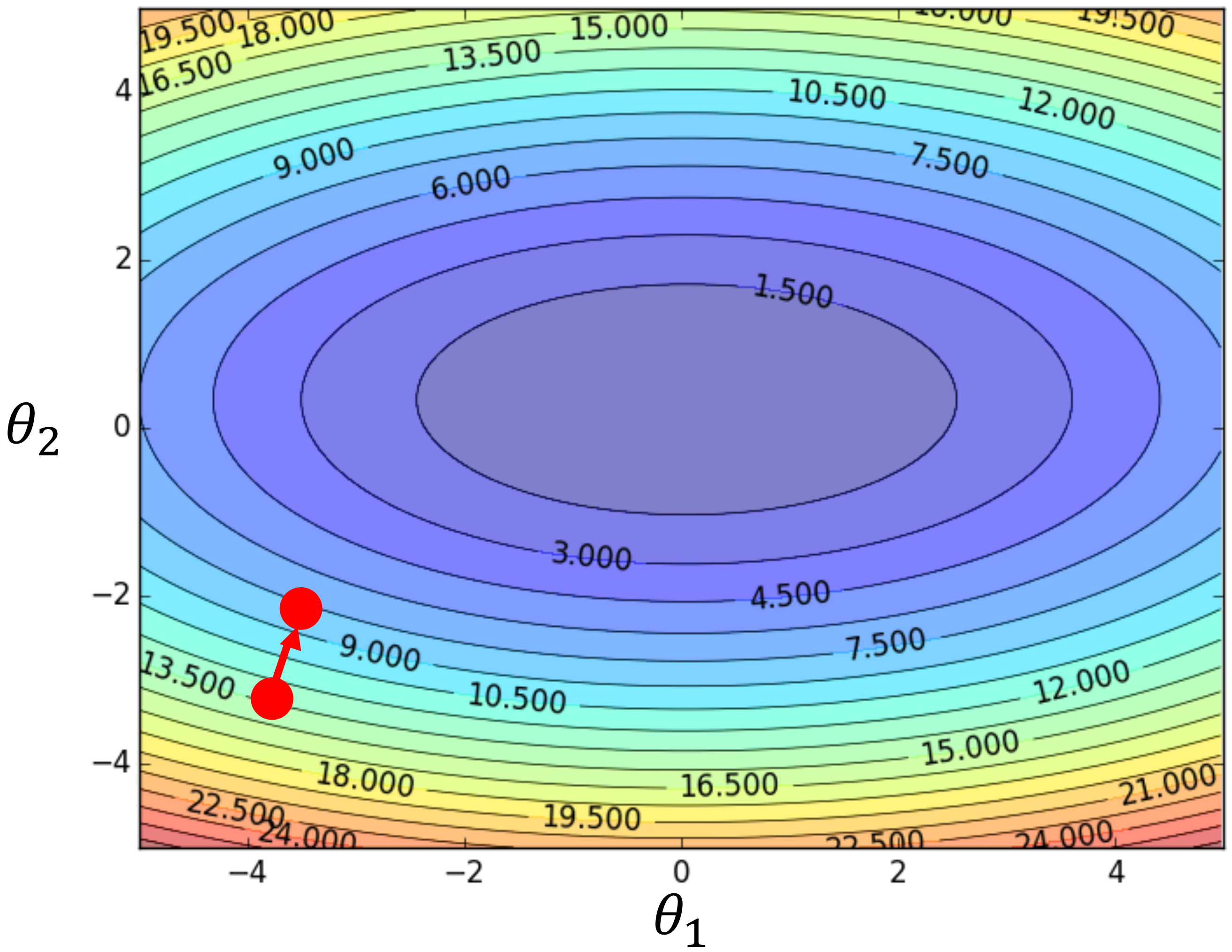


Set the learning rate α carefully

If learning rate is too large

Total loss may not decrease after each update

LEARNING RATES



Set the learning rate α carefully

If learning rate is too large

Total loss may not decrease after each update

If learning rate is too small

Training would be too slow

LEARNING RATES

- Popular & Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate
  - E.g. $\frac{1}{t}$ decay: $\alpha^{(t)} = \frac{\alpha}{\sqrt{t+1}}$
- Learning rate cannot be one-size-fits-all
  - Giving different parameters different learning rates

ADAGRAD

Original: $\theta \leftarrow \theta - \alpha \partial L \,/\, \partial \theta$

Adagrad: $\theta \leftarrow \theta - \boxed{\alpha_\theta} \partial L \,/\, \partial \theta$

Parameter-dependent learning rate

$$\alpha_\theta = \frac{\alpha}{\sqrt{\boxed{\sum_{i=0}^{t} (g^i)^2}}}$$

constant

g^i is $\partial L \,/\, \partial \theta$ obtained at the i^{th} update

Summation of the square of the previous derivatives

ADAGRAD

$$\alpha_\theta = \boxed{\frac{\alpha}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

θ_1	g^0
	0.1

θ_2	g^0
	20.0

Learning rate:

$$\frac{\alpha}{\sqrt{0.1^2}} = \frac{\alpha}{0.1}$$

$$\frac{\alpha}{\sqrt{0.1^2 + 0.2^2}} = \frac{\alpha}{0.22}$$

Learning rate:

$$\frac{\alpha}{\sqrt{20^2}} = \frac{\alpha}{20}$$

$$\frac{\alpha}{\sqrt{20^2 + 10^2}} = \frac{\alpha}{22}$$

Observation: 1. Learning rate is smaller and smaller for all parameters

2. Smaller derivatives, larger learning rate, and vice versa

Very useful tutorial on an overview of gradient descent optimization algorithms

https://ruder.io/optimizing-gradient-descent/

QUIZ 4

Quiz 4

Not available until May 18 at 3:00pm | **Due** May 18 at 11:59pm | 8 pts | 8 Questions

QUESTIONS?