

CS 1674: Neural Networks

PhD. Nils Murrugarra-Llerena
nem177@pitt.edu



Outline

- Motivation Neural Networks
- Perceptron
- Activation Functions
- Training a Neural Network
 - Weight Initialization
 - Forward Propagation
 - Loss Function
 - Regularization
 - Gradient Descent
 - Backpropagation

To join, go to: ahaslides.com/51QST 



What AI courses are you taking this semester?

 Get Feedback



 14  3  2  7  0/100 

Many inventions were inspired by Nature ...

Birds inspired us to fly

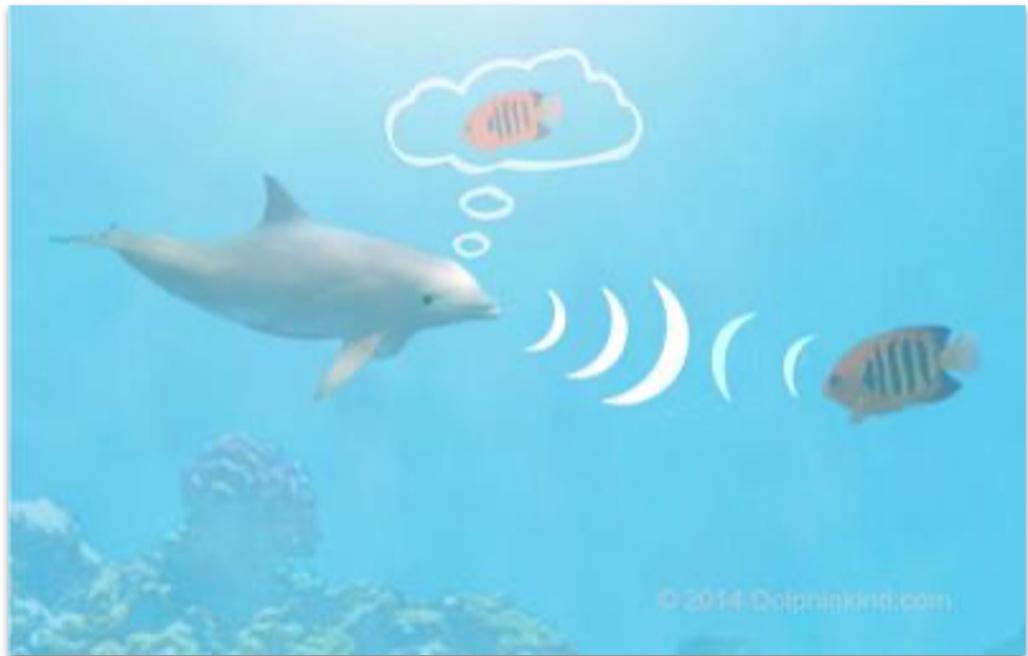
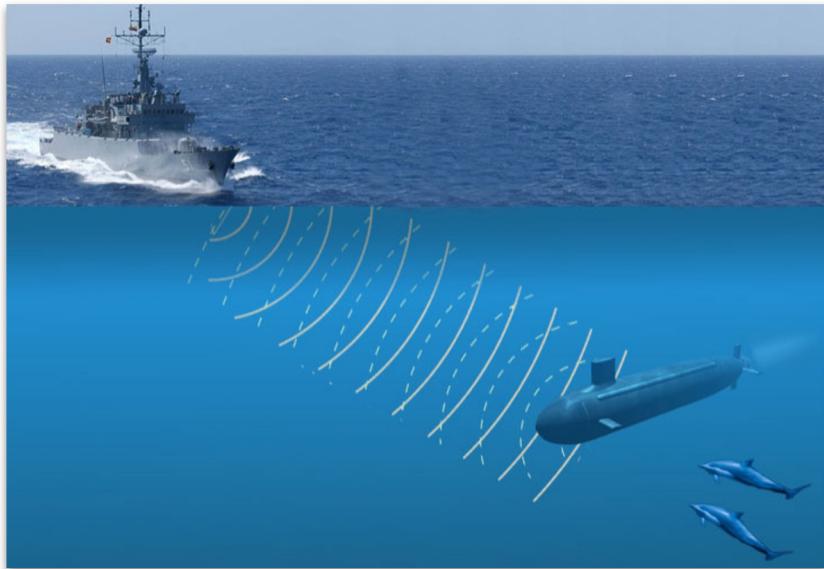


<https://i.pinimg.com/736x/18/4a/cd/184accff6b4d7980d22e8090ff295cd7--animal-tracks-friends.jpg>

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Many inventions were inspired by Nature ...

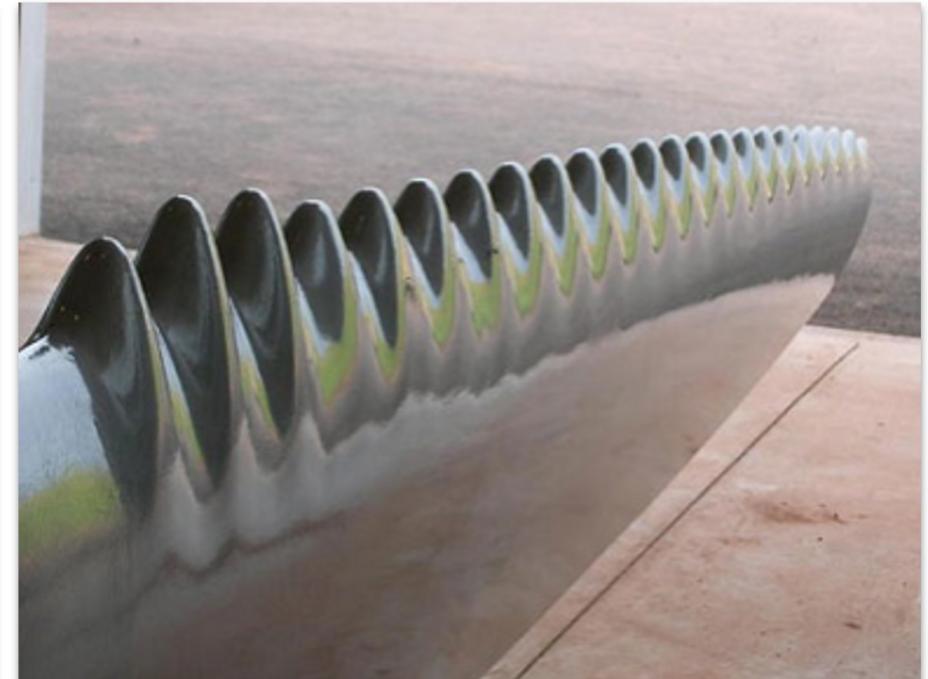
Dolphins inspired sonar development



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Many inventions were inspired by Nature ...

Humpback whales inspired wind turbines



<https://images.app.goo.gl/7muFjRTQLLRD1dh9>

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

It seems logical to look at the
brain's architecture for inspiration on
how to build an intelligent machine.



ILSVRC 2012 – Image Classification task

Rank	Name	Error Rate (%)	Description
1	University of Toronto		Deep Learning
2	University of Tokyo	26.2	Hand-crafted features and learning models
3	University of Oxford	26.9	
4	Xerox/INRIA	27.0	

Object recognition over 1,000,000 images and 1,000 categories (2 GPU)

“ImageNet classification with deep convolutional neural networks”. NIPS, 2012.

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP



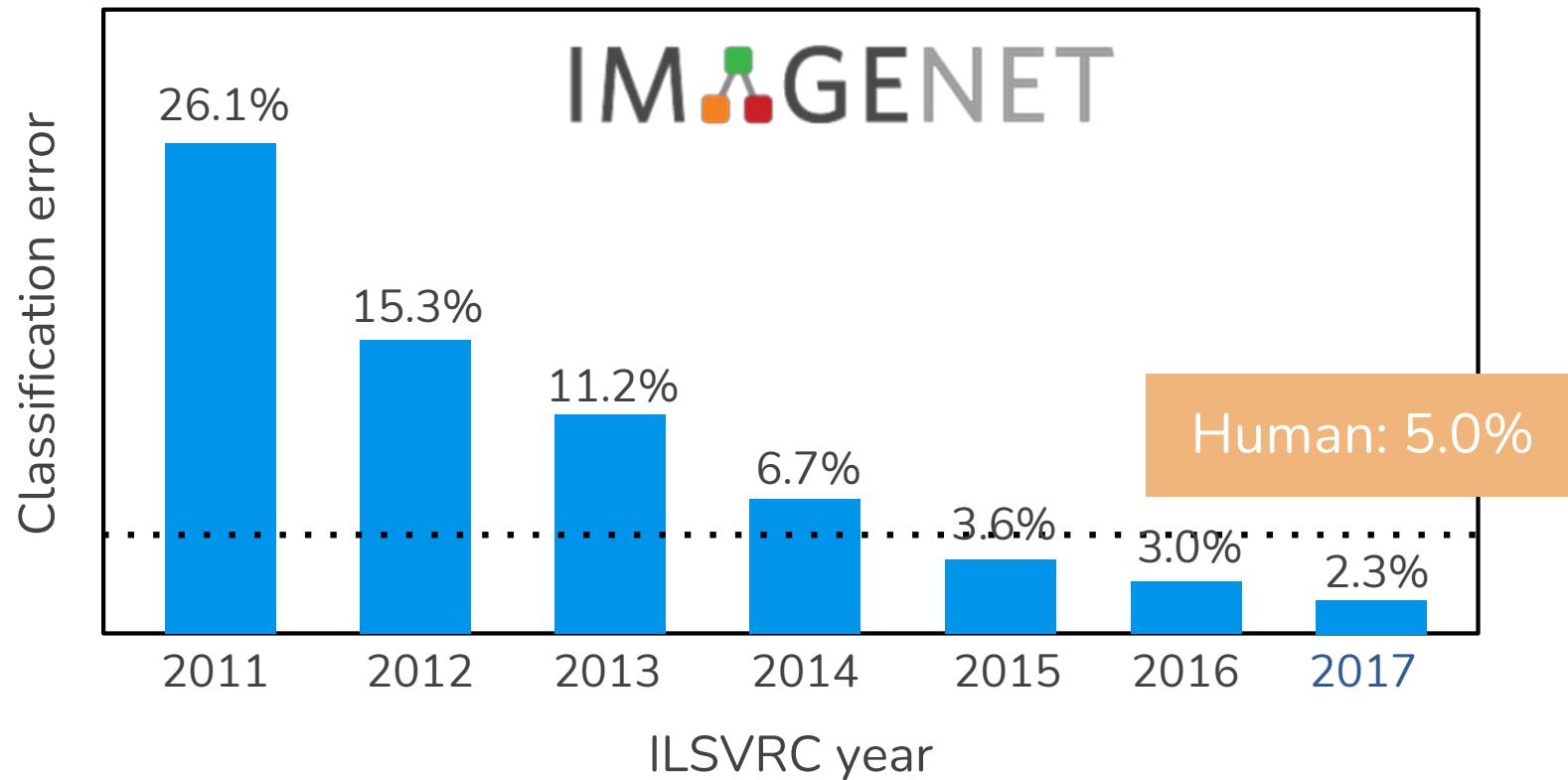
ILSVRC 2012 – Image Classification task

Rank	Name	Error Rate (%)	Description
1	University of Toronto	15.3	Deep Learning
2	University of Tokyo	26.2	Hand-crafted features and learning models
3	University of Oxford	26.9	
4	Xerox/INRIA	27.0	

Object recognition over 1,000,000 images and 1,000 categories (2 GPU)

“ImageNet classification with deep convolutional neural networks”. NIPS, 2012.

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP



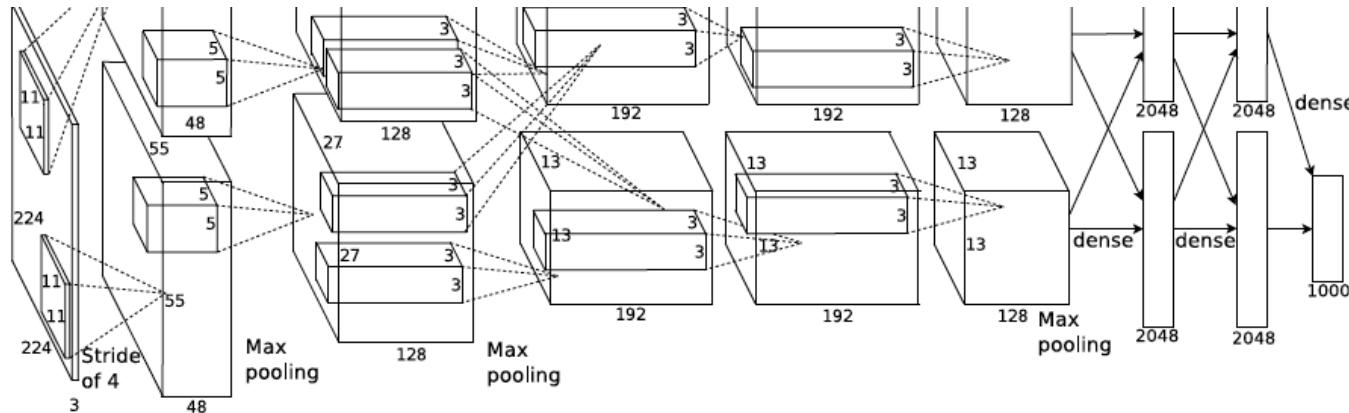
“ImageNet classification with deep convolutional neural networks”. NIPS, 2012.

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

ImageNet Challenge 2012



- AlexNet: Similar framework to LeCun'98 but:
 - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
 - More data (10^6 vs. 10^3 images)
 - GPU implementation (50x speedup over CPU)
 - Trained on two GPUs for a week
 - Better regularization for training (DropOut)



A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](#), NIPS 2012

Adapted from Lana Lazebnik

Will this wave die out like the previous ones did?

From Biological to Artificial Neurons

1. There is now a **huge quantity of data** available to train neural networks.



IM^AGENET

www.image-net.org

22K categories and **14M** images

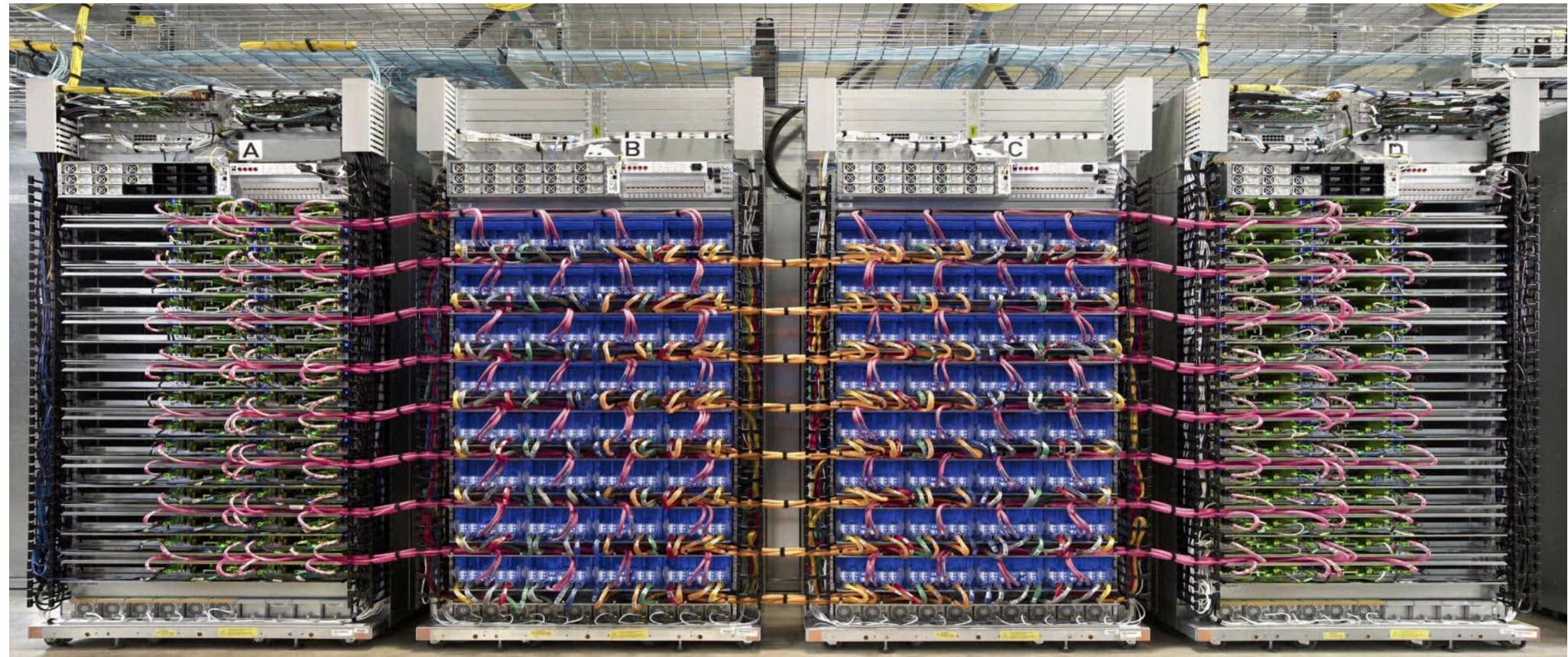
- Animals
 - Bird
 - Fish
 - Mammal
 - Invertebrate
- Plants
 - Tree
 - Flower
- Food
- Materials
- Structures
 - Artifact
 - Tools
 - Appliances
 - Structures
- Person
- Scenes
 - Indoor
 - Geological Formations
- Sport Activities



Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009

From Biological to Artificial Neurons

1. There is now a **huge quantity of data** available to train neural networks.
 2. **Computing power** now makes it possible to train large neural networks in a reasonable amount of time.
-



<http://www.tomshardware.com/news/google-automl-artifical-intelligence-ai,34533.html>

Lab 6: Pytorch

Duration: 20 min



To join, go to: ahaslides.com/JAD9A



Please, from Lab 6: Pytorch [Coding Exercise 1.2], submit the top left number of the output A matrix.

12	24	[12,14] [22,22]	
12	12	12	
12	12	12	
12	12	12	
12	20	12	
20	12	12	
12	12	12	

Get Feedback

Assignment 6: Pytorch



From Biological to Artificial Neurons

1. There is now a **huge quantity of data** available to train neural networks.
2. **Computing power** now makes it possible to train large neural networks in a reasonable amount of time.
3. The **training algorithms** have been **improved**.

Science AAAS

f t y G+

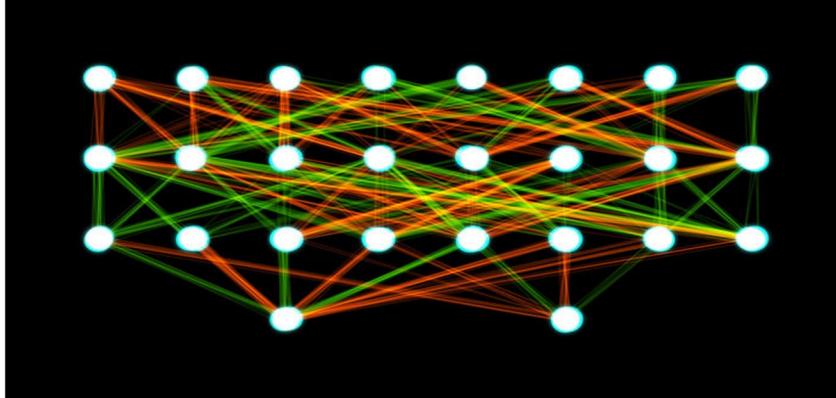
Authors | Members | Librarians | Advertisers

Home News Journals Topics Careers

Search

Latest News ScienceInsider ScienceShots Sifter From the Magazine About News Quizzes

SHARE



A representation of a neural network.

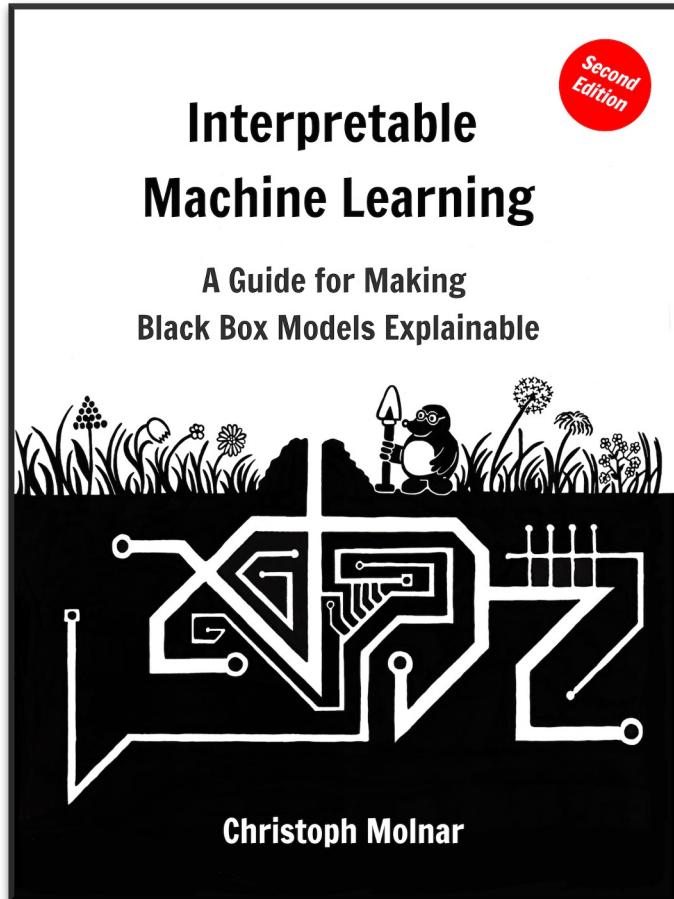
Akritasa/Wikimedia Commons

Brainlike computers are a black box. Scientists are finally peering inside

By Jackie Snow | Mar. 7, 2017, 3:15 PM

Last month, Facebook announced software that could simply look at a photo and tell, for example, whether it was a picture of a cat or a dog. A related program identifies cancerous

<http://www.sciencemag.org/news/2017/03/brainlike-computers-are-black-box-scientists-are-finally-peering-inside>



"Interpretable Machine Learning", 2022

<https://christophm.github.io/interpretable-ml-book>

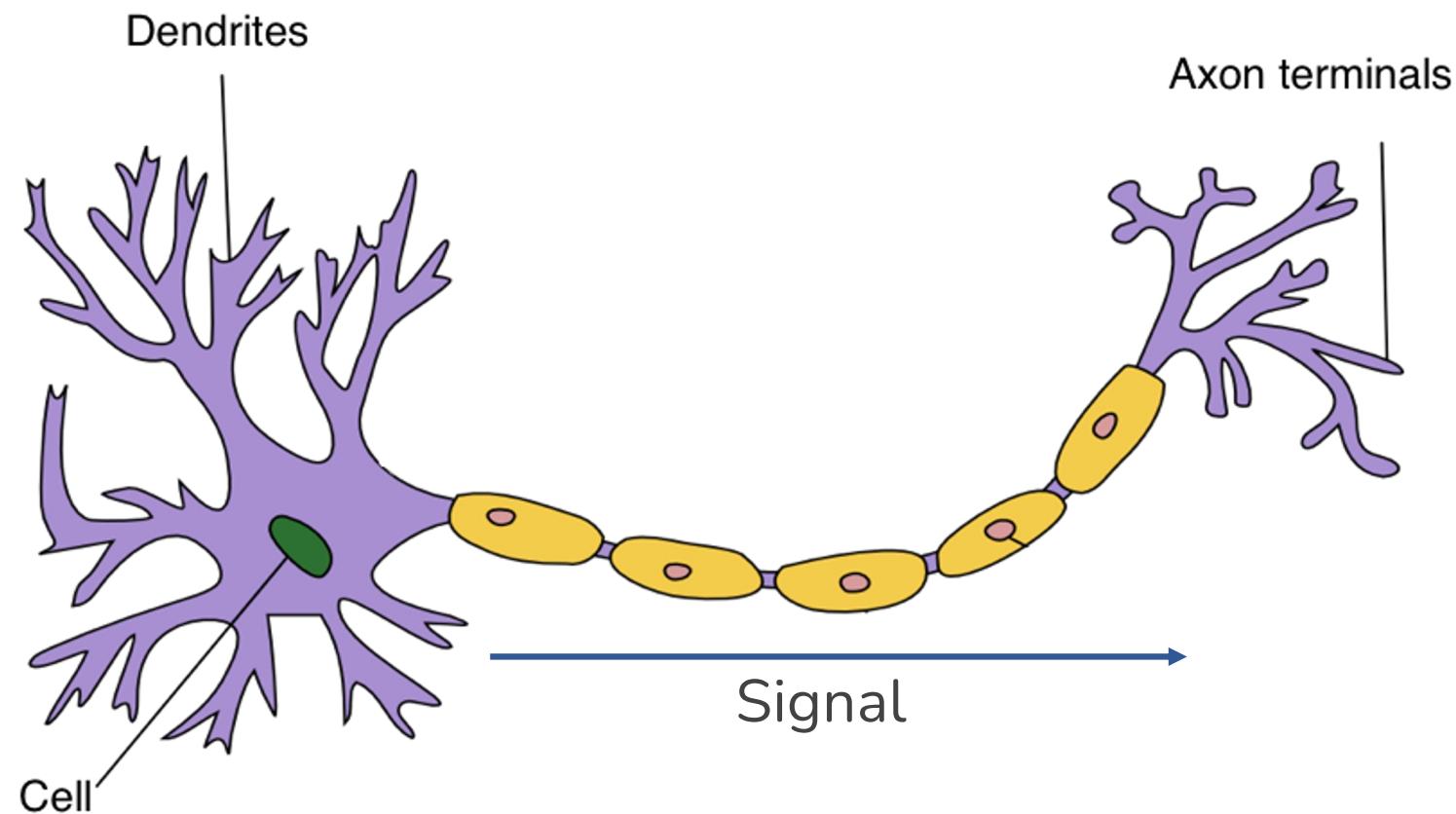
Chap. 10 Neural Network Interpretation

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

From Biological to Artificial Neurons

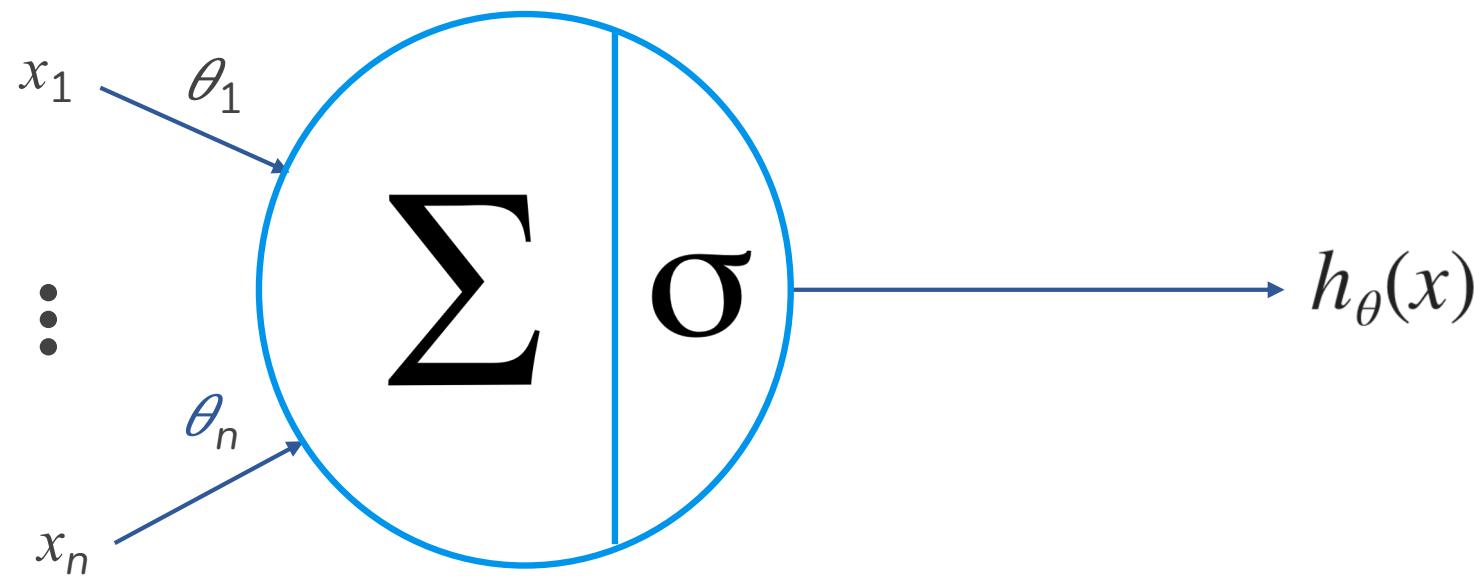
1. There is now a **huge quantity of data** available to train neural networks.
2. **Computing power** now makes it possible to train large neural networks in a reasonable amount of time.
3. The **training algorithms** have been **improved**.
4. ANNs seem to have entered a virtuous circle of funding and progress.

Biological Neurons



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Biological Neurons - Analogy

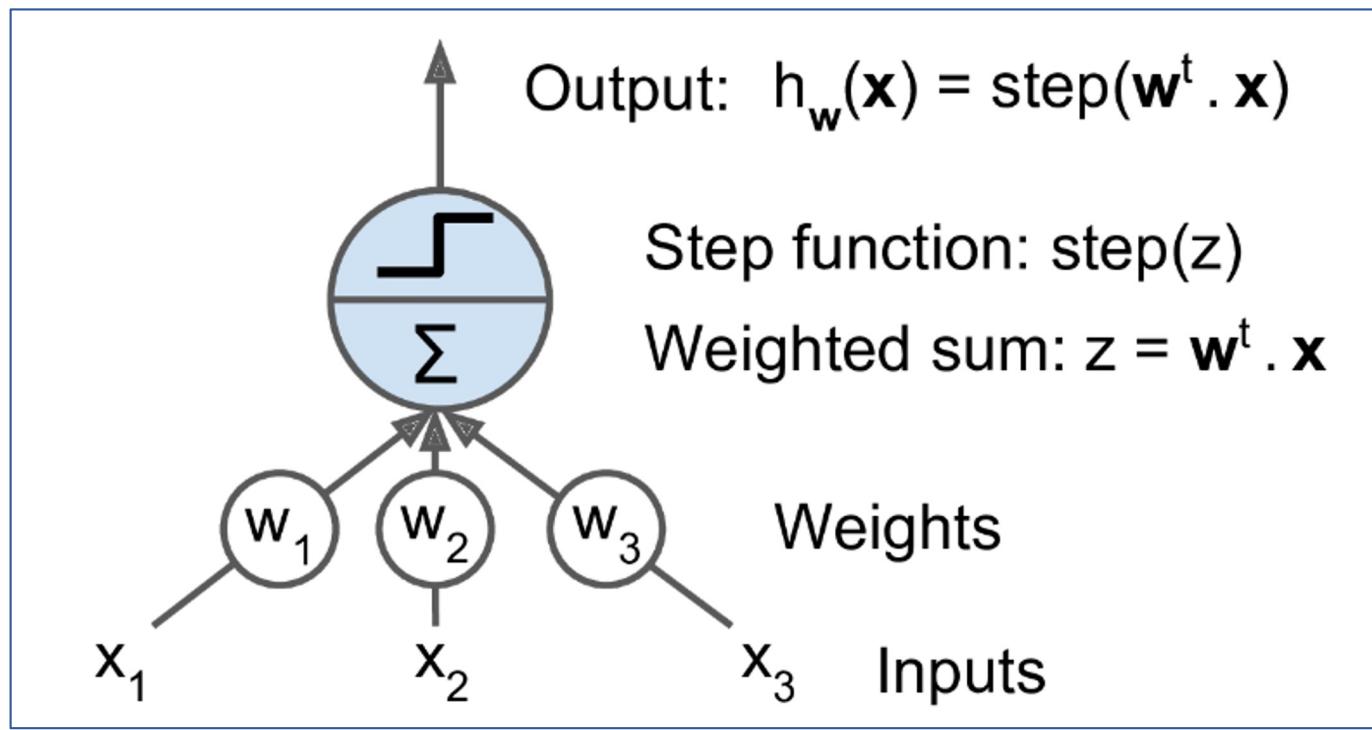


The Perceptron

Invented in 1957 by Frank Rosenblatt.

- It is based on a Linear Threshold Unit (LTU):
 - The inputs and output are now **numbers** and each input connection is associated with a **weight**.
- The LTU computes a weighted sum of its inputs then it applies a step function to that sum and outputs the result.

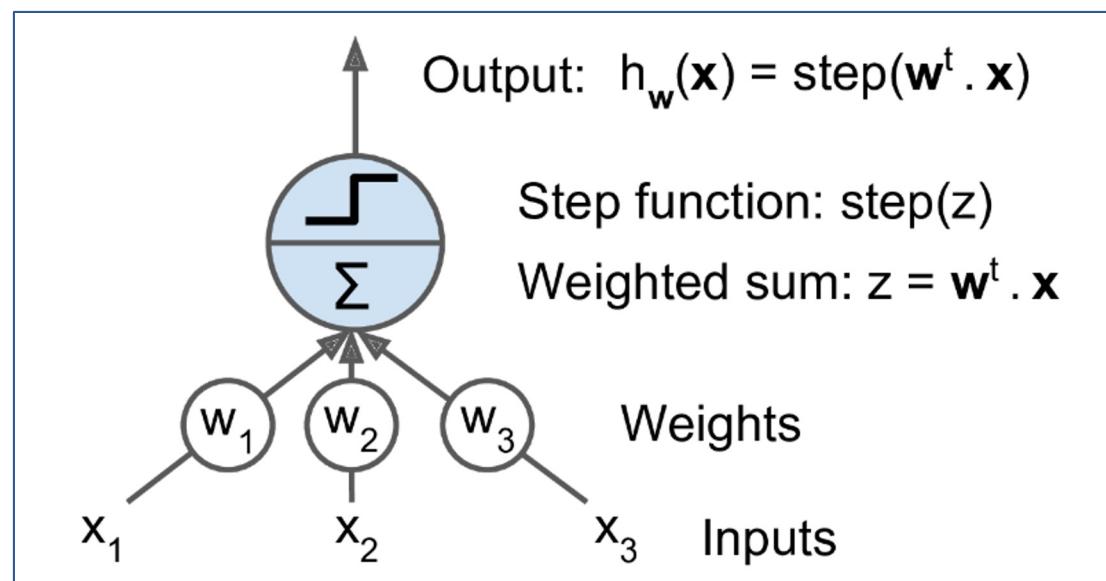
The Perceptron



Linear Threshold Unit

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

The Perceptron

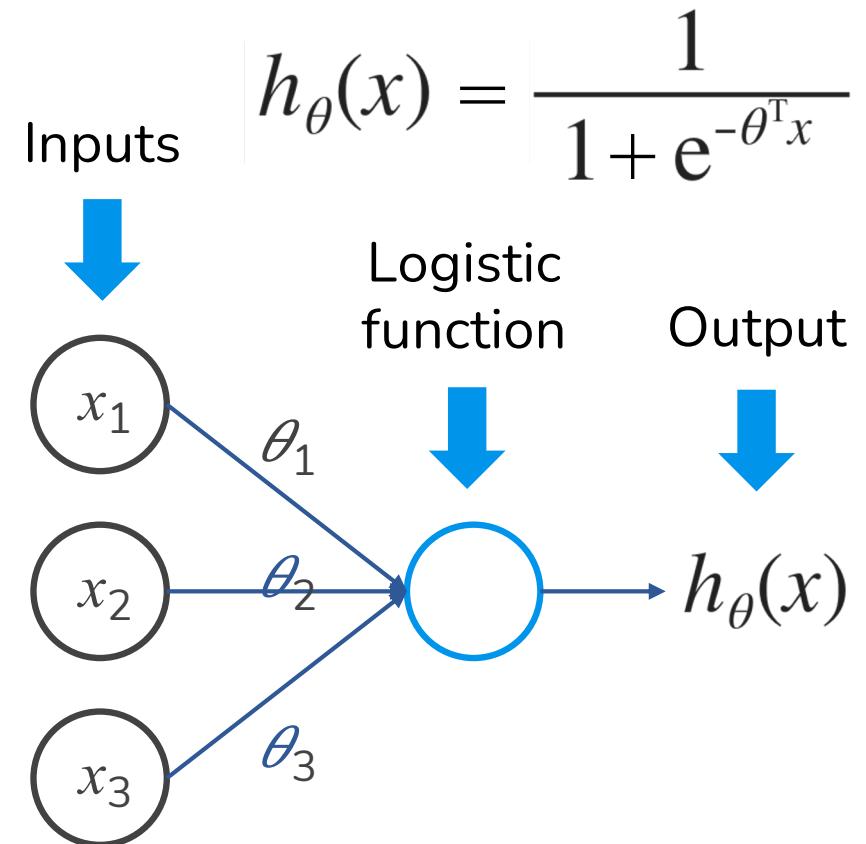
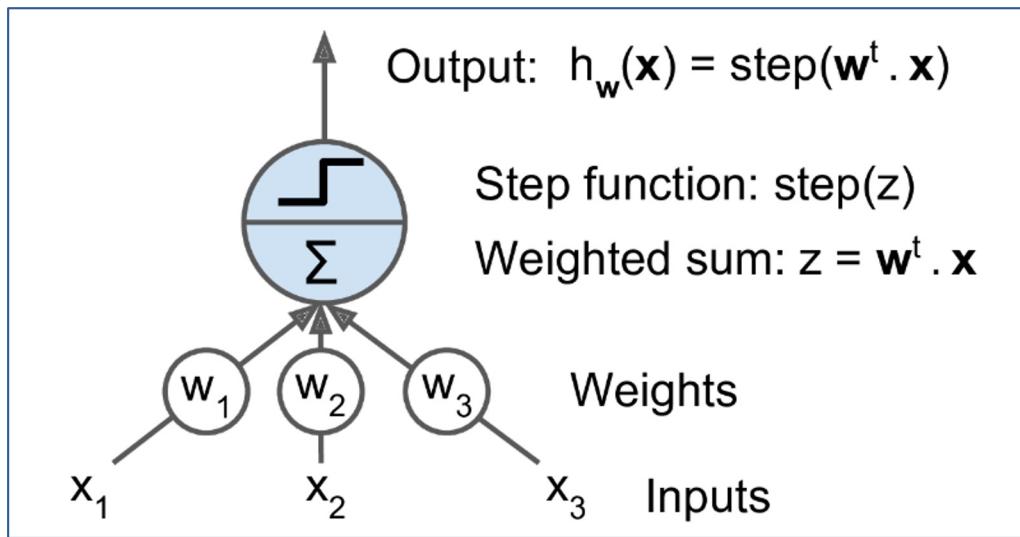


Linear Threshold Unit

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

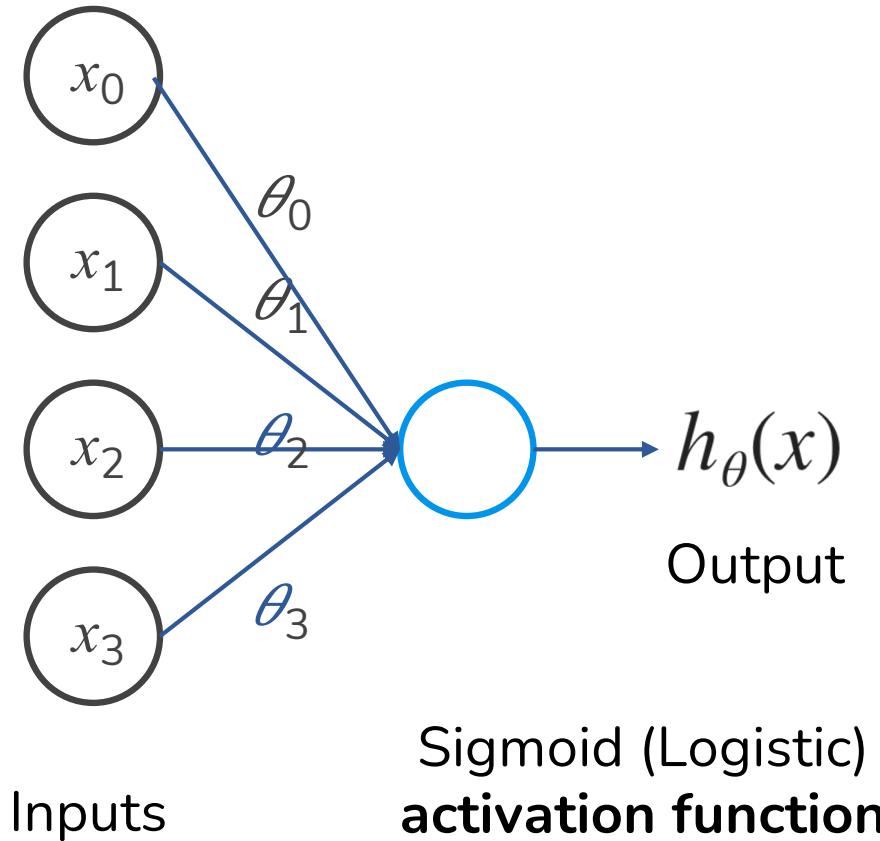
$$\text{sign}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

Neuron Model: Logistic Unit



30
weights

Neuron Model: Logistic Unit



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

TensorFlow PlayGround

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.

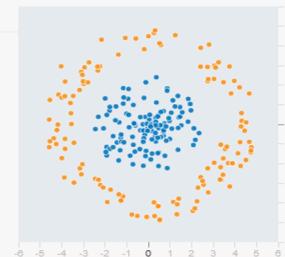
Epoch 000,000 Learning rate 0.03 Activation Sigmoid Regularization None Regularization rate 0 Problem type Classification

DATA
Which dataset do you want to use?

Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE

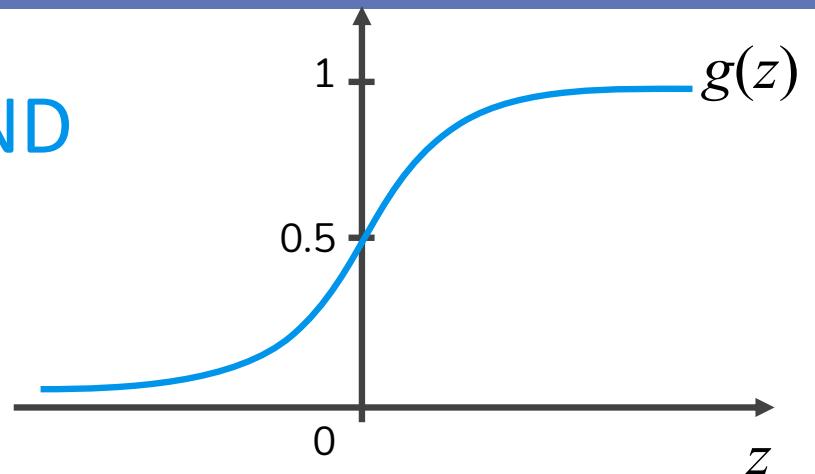
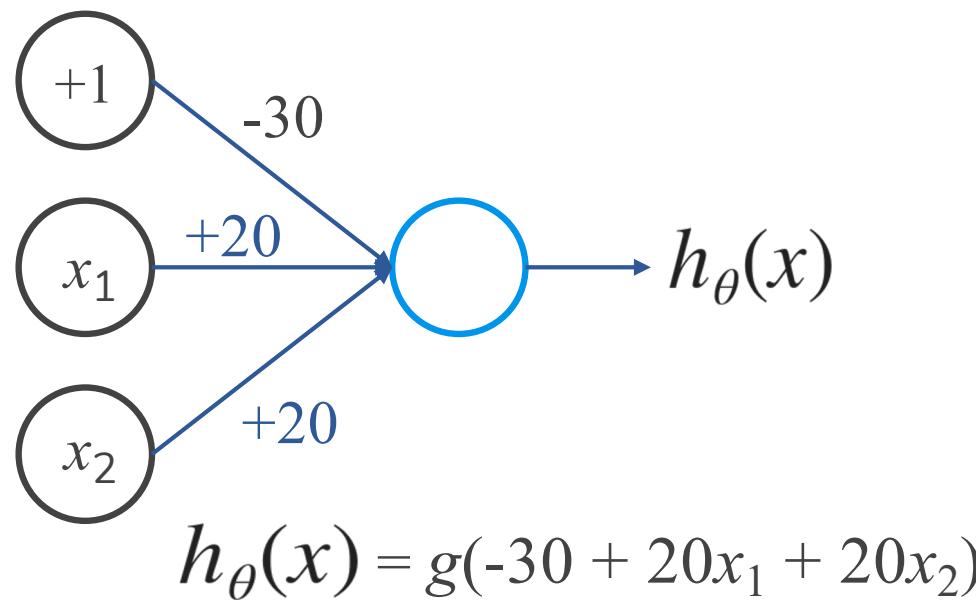
FEATURES
Which properties do you want to feed in?
X1
X2
X12
X22
X1X2
 $\sin(X_1)$
 $\sin(X_2)$

1 HIDDEN LAYER
+ - 1 neuron
This is the output from one neuron. Hover to see it larger.

OUTPUT
Test loss 0.504
Training loss 0.502

Colors shows data, neuron and weight values.
 Show test data Discretize output

[Example] Simple Example: AND

$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ AND } x_2$$

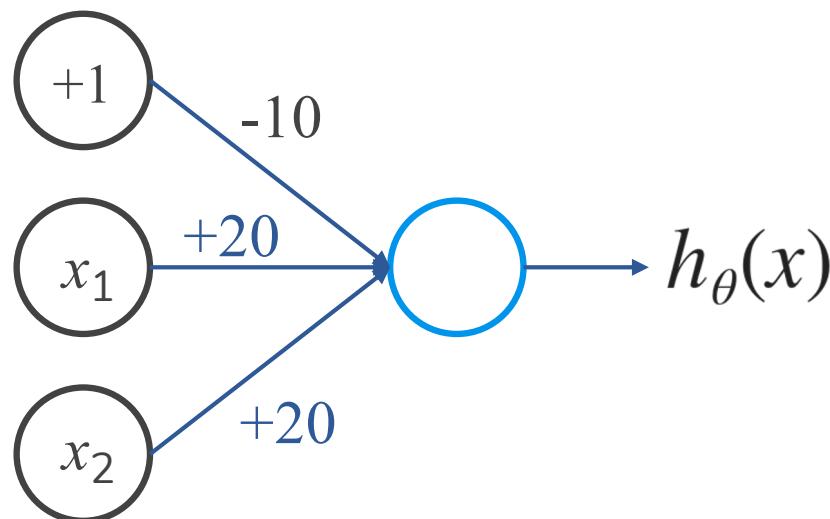


x_1	x_2	$h_\theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

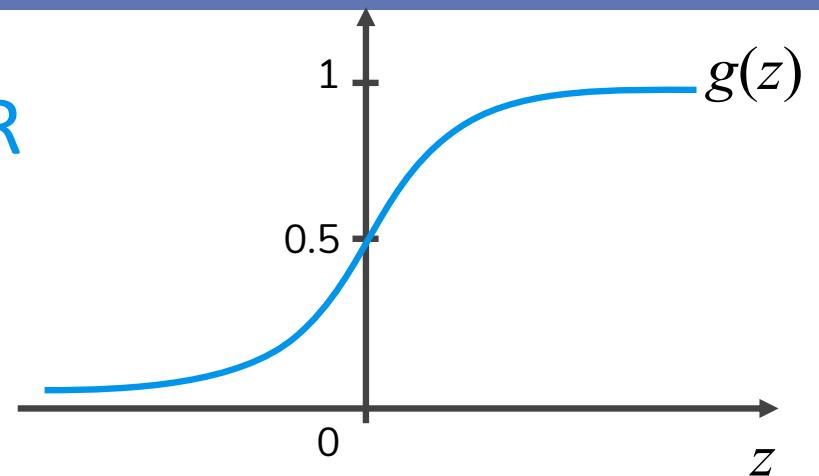
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

[Example] Simple Example: OR

$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ OR } x_2$$



$$h_{\theta}(x) = g(-10 + 20x_1 + 20x_2)$$



x_1	x_2	$h_{\theta}(x)$
0	0	$g(-10) \approx 0$
0	1	$g(10) \approx 1$
1	0	$g(10) \approx 1$
1	1	$g(30) \approx 1$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Multi-class Classification?

Multi-class Classification



Cat



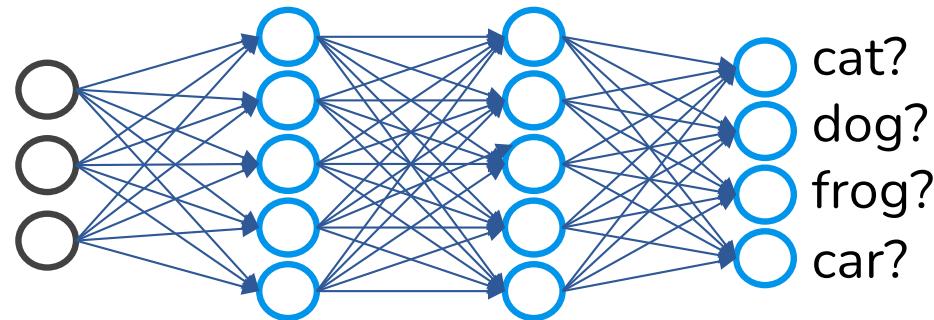
Dog



Frog



Car



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

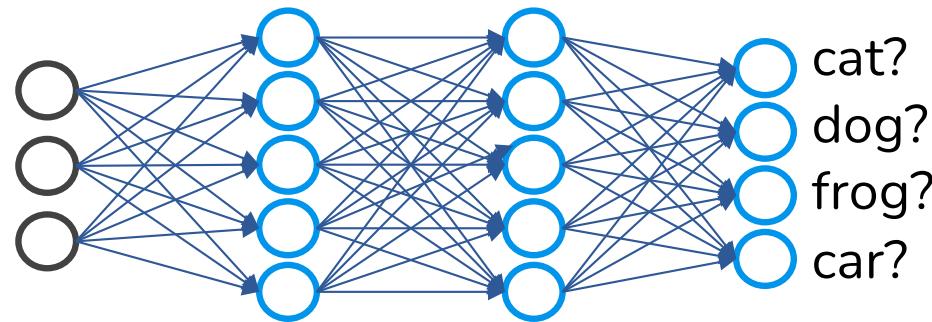


Cat

Dog

Frog

Car

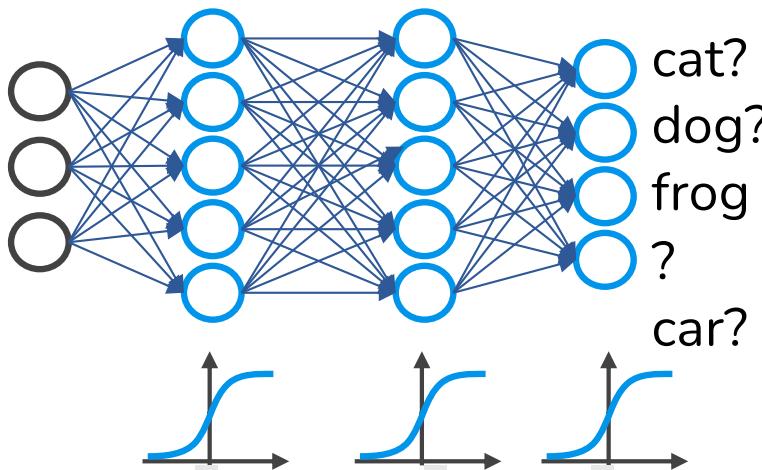


$$\text{Want } h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

when cat when dog when frog when car

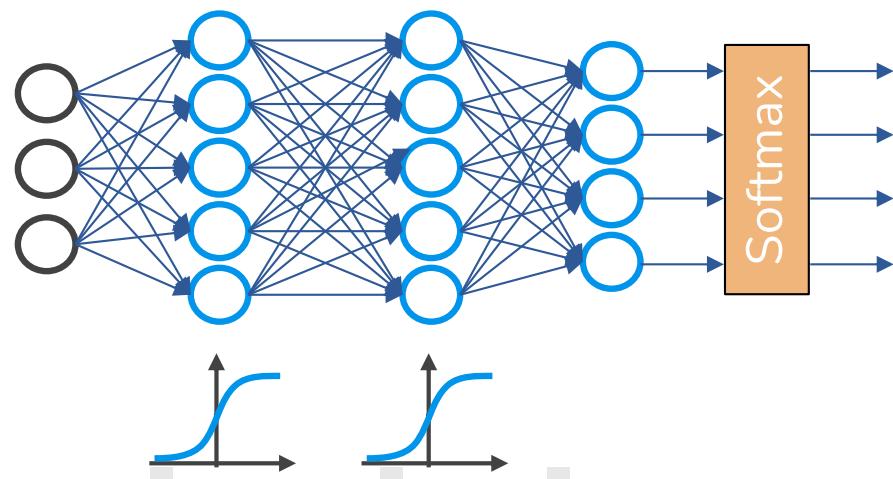
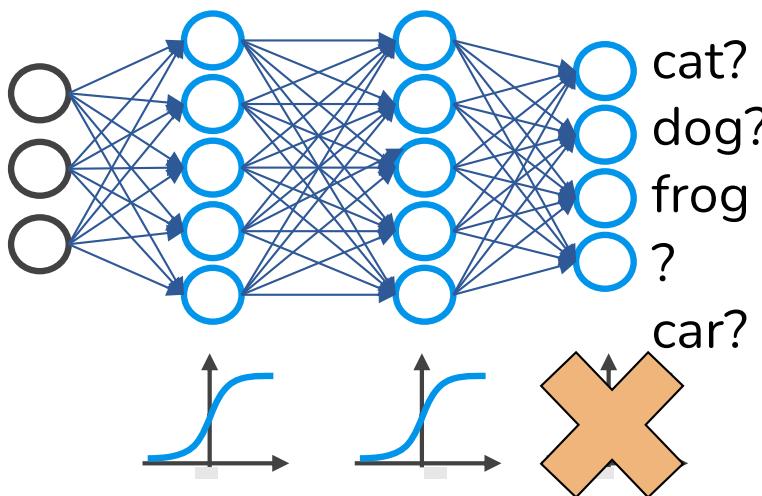
Softmax Classification

The **output layer** is typically modified by replacing the individual activation functions **by a shared softmax** function.



Softmax Classification

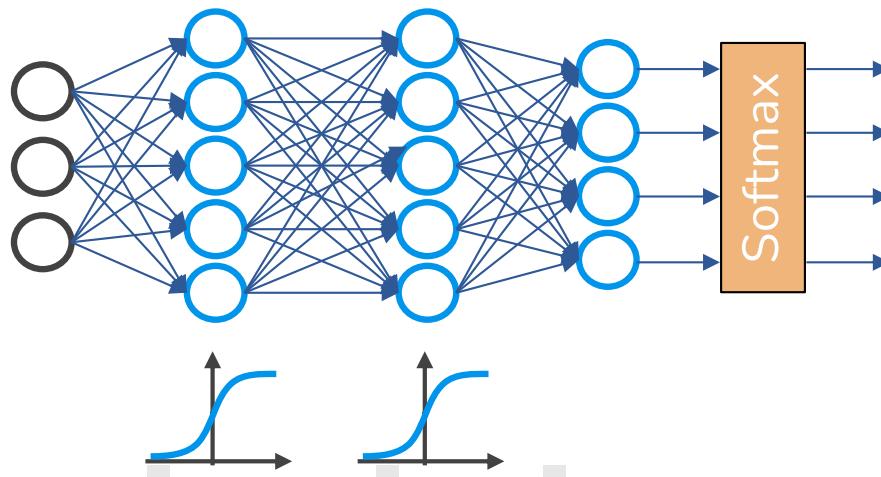
The **output layer** is typically modified by replacing the individual activation functions **by a shared softmax** function.



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Softmax Classification

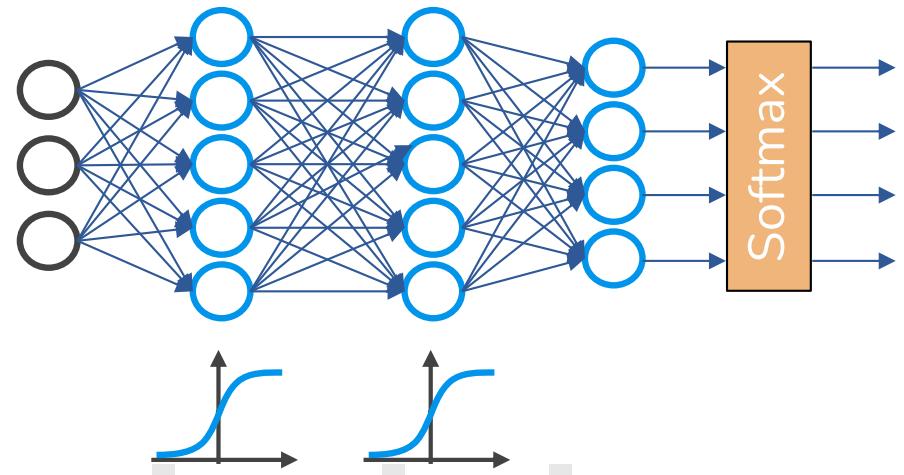
The **output layer** is typically modified by replacing the individual activation functions **by a shared softmax** function.



$$f(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Softmax Classification



Cat	5.1	164.0	0.87
Dog	3.2	24.5	0.13
Frog	-1.7	0.18	0.00
Car	-2.0	0.13	0.00

$$f(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP



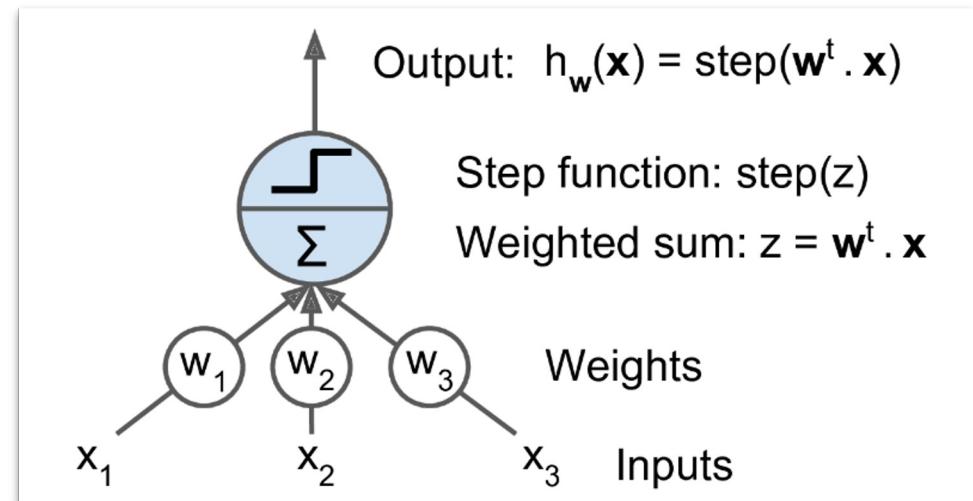
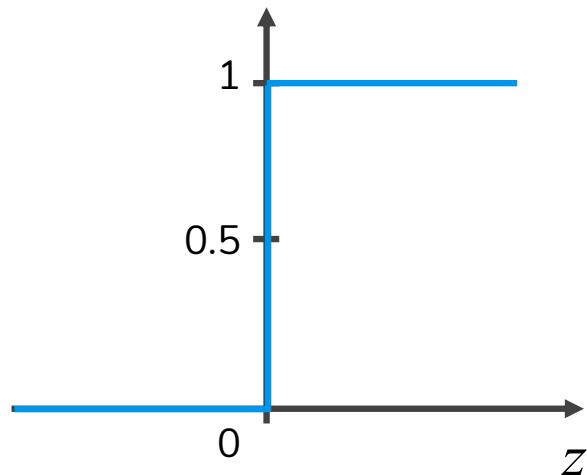
How do we decide whether the neuron should fire or not?

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

We decided to add “activation functions” for this purpose.

Step Function

Its output is **1 (activated)** when value > 0 (threshold) and outputs a **0 (not activated)** otherwise.



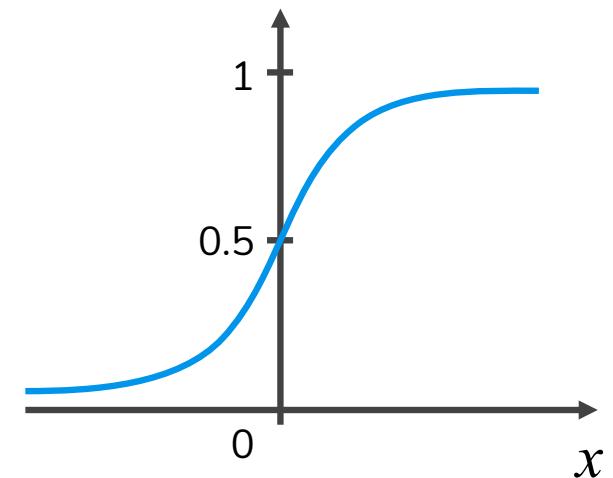
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Step Function: Problem?

- Binary classifier (“yes” or “no”, activate or not activate). A Step function could do that for you!
- Multi classifier (class1, class2, class3, etc). What will happen if more than 1 neuron is “activated”?

Sigmoid Function

- The output of the activation function is always going to be in range **(0,1)**.
- It is nonlinear in nature.

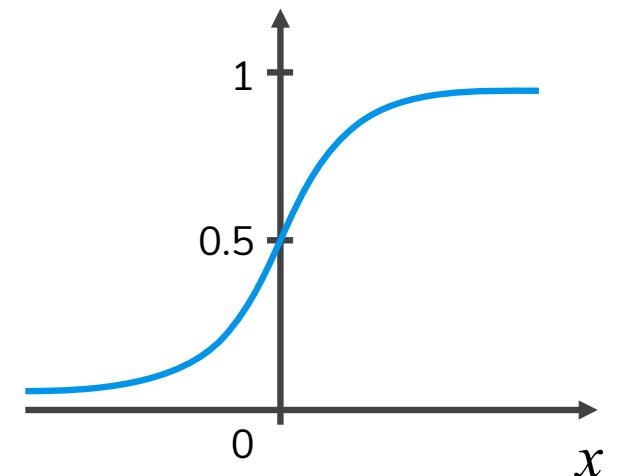


$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Sigmoid Function

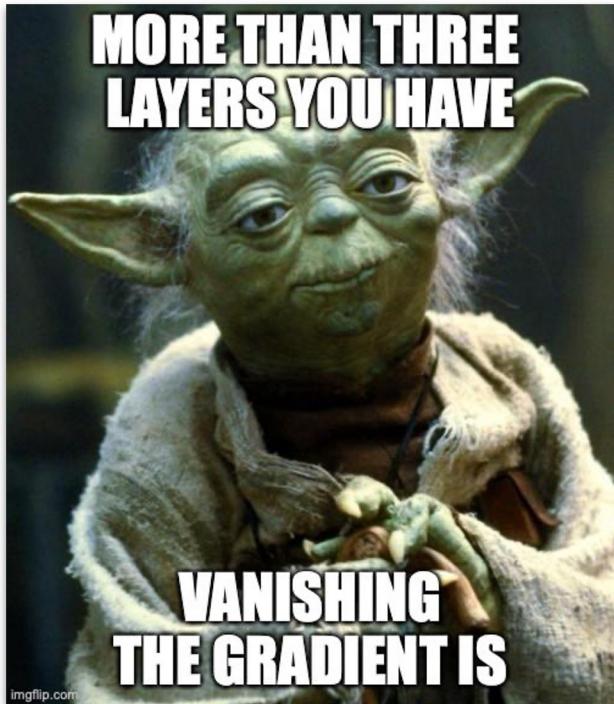
- The output of the activation function is always going to be in range **(0,1)**.
- It is nonlinear in nature.
- Combinations of this function are also nonlinear! Great!!



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Sigmoid Function: Problem?

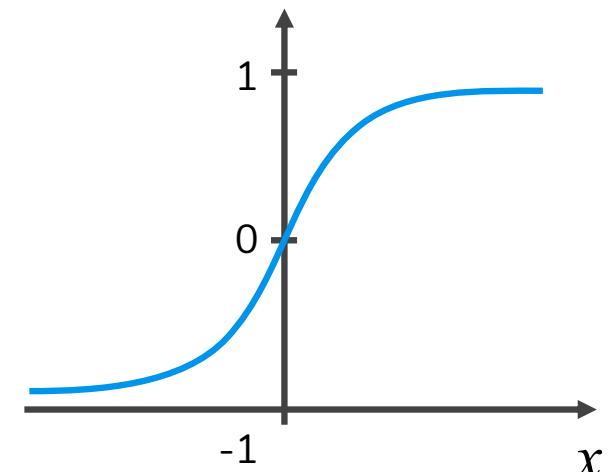


- Towards either end of the sigmoid function, the $\sigma(x)$ values tend to respond very less to changes in x .
- The problem of “**vanishing gradients**”.
 - Cannot make significant change because of the extremely small value.

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Tanh Function

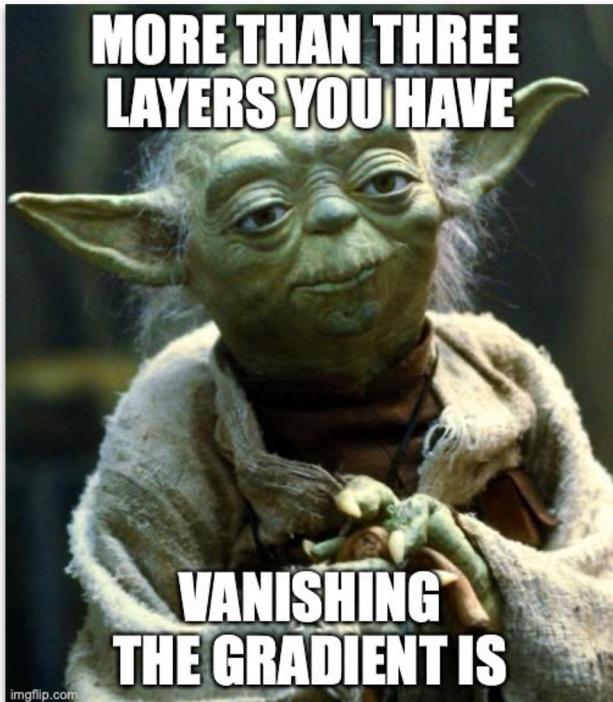
- The output of the activation function is always going to be in range **(-1,1)**.
- It is nonlinear in nature.
- Combinations of this function are also nonlinear! Great!!



$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

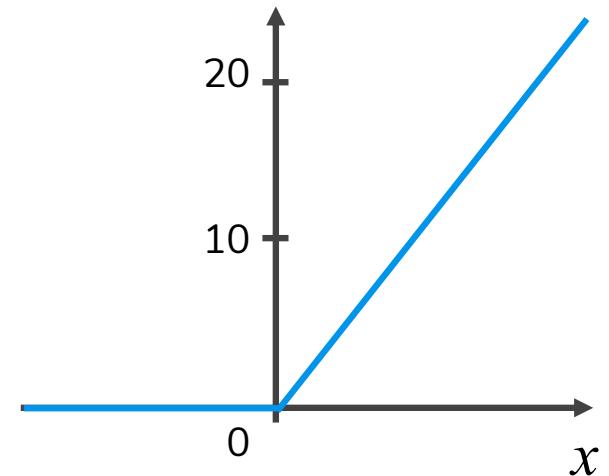
Tanh Function: Problem?



- Like sigmoid, tanh also has the vanishing gradient problem.

ReLU (Rectified Linear Unit) Function

- It gives an output x if x is positive and 0 otherwise. The range is $[0, \infty)$.
- It is nonlinear in nature. Combinations of this function are also nonlinear!
- Sparsity of the activation!



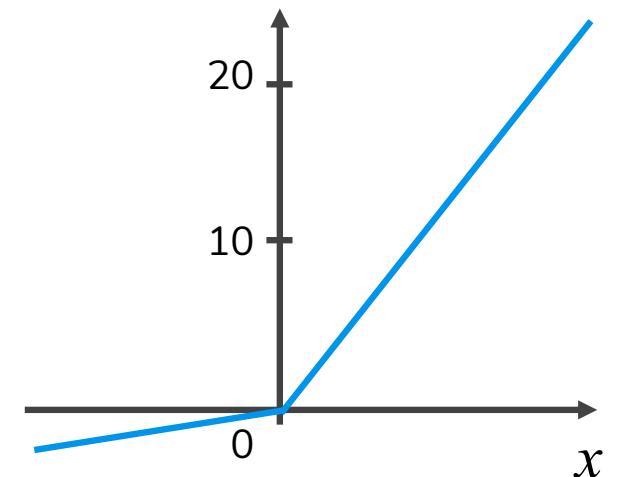
$$\text{ReLU}(x) = \max(0, x)$$

ReLU Function: Problem?

- Because of the horizontal line in ReLU (for negative x), the gradient can go towards 0.
- “Dying ReLU problem”: several neurons can just die and not respond making a substantial part of the network passive.

Leaky ReLU Function

- It gives an output x if x is positive and 0 otherwise. The range is $[0, \infty)$.
- (Leaky) ReLU is less computationally expensive than *tanh* and *sigmoid* because it involves simpler mathematical operations.



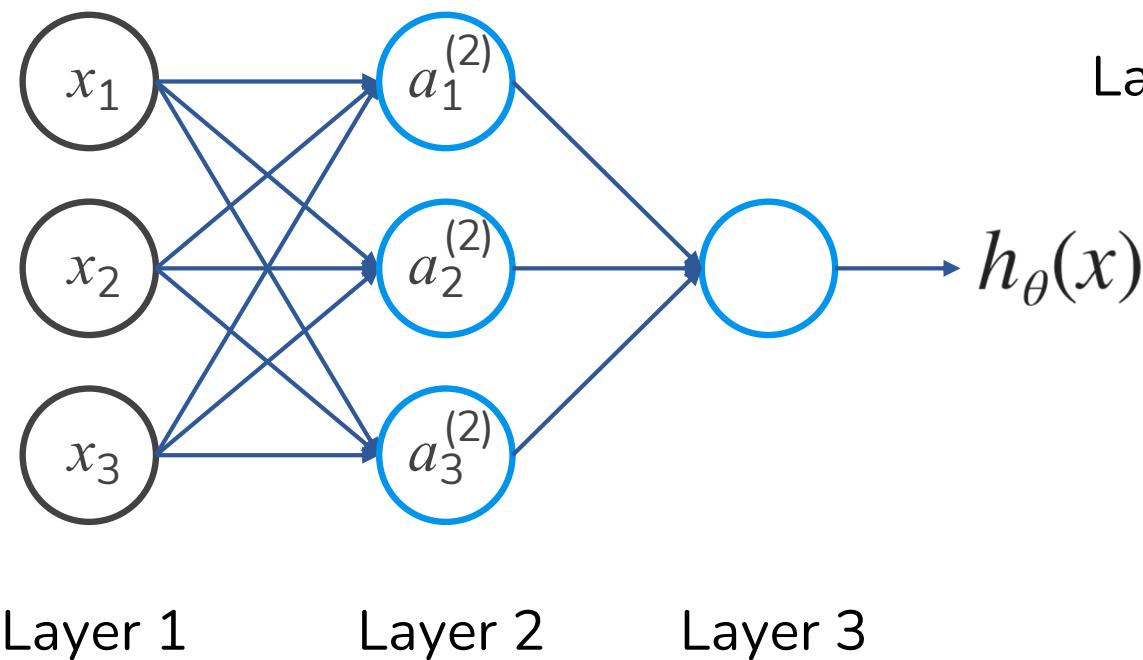
$$\begin{aligned}\text{Leaky ReLU}(x) &= \\ &= \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}\end{aligned}$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Ok! Which One Do We Use?

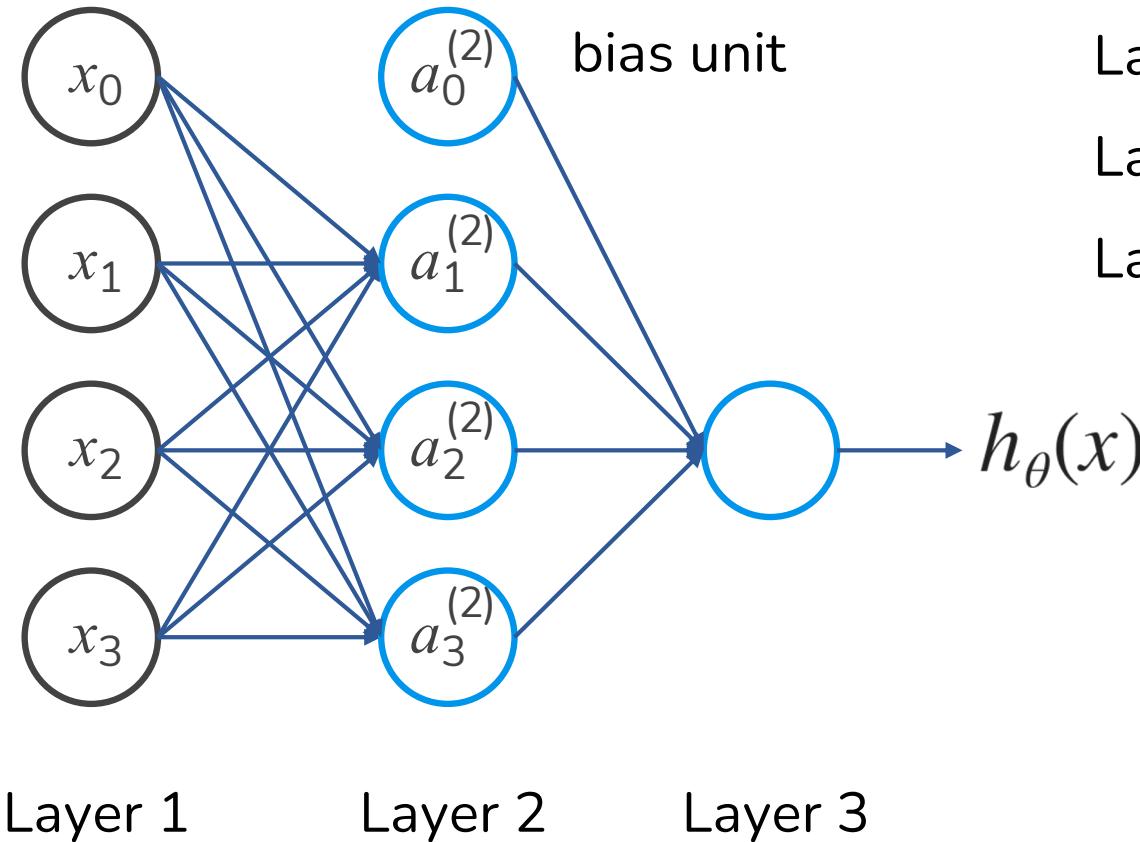
- If you don't know the nature of the function you are trying to learn, start with ReLU.
- You can use your own custom functions too!

Neural Network



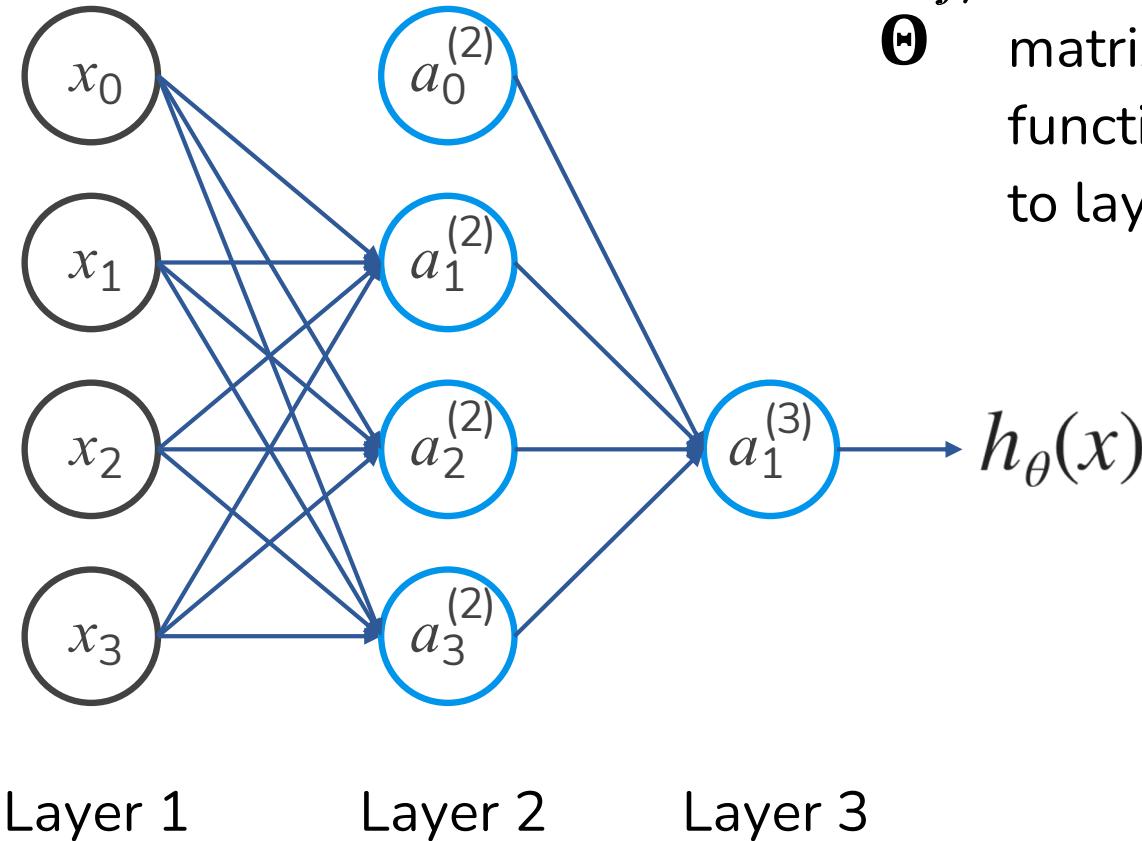
Layer 1 = Input layer
Layer 2 = Hidden layer
Layer 3 = Output layer

Neural Network



Layer 1 = Input layer
Layer 2 = Hidden layer
Layer 3 = Output layer

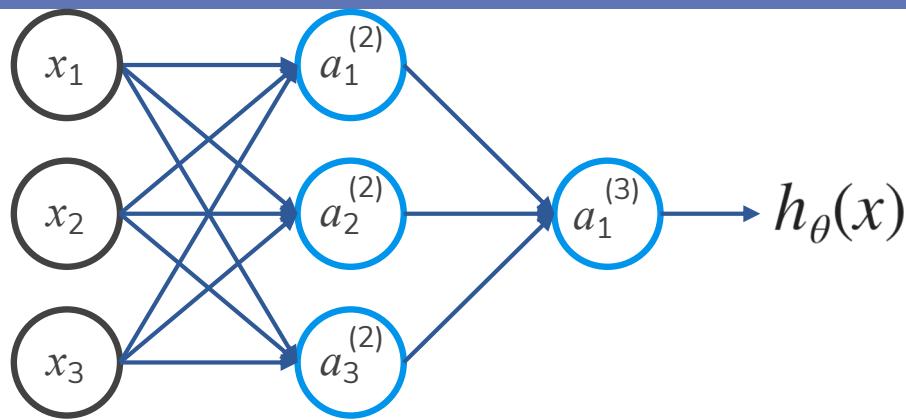
Neural Network


$$(j)$$
$$a_i^{(j)}$$
$$\Theta$$

"activation" of unit i in layer j

matrix of weights controlling
function mapping from layer j
to layer $j + 1$

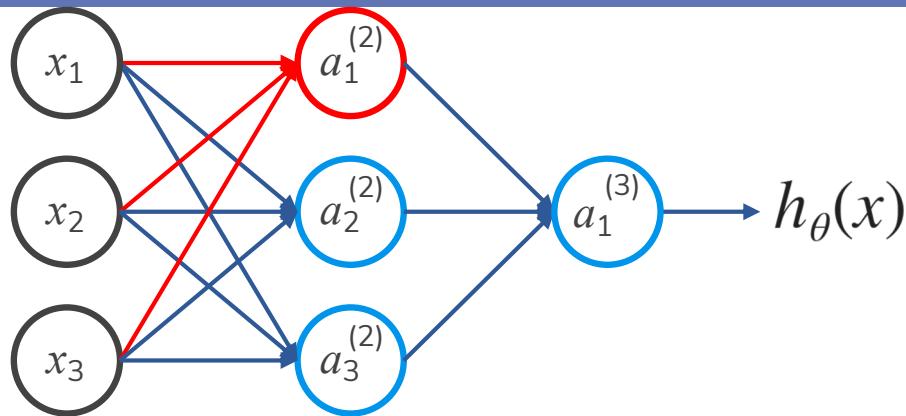
$$h_{\theta}(x)$$

 $a_i^{(j)}$

“activation” of unit i in layer j

 Θ

matrix of weights controlling
function mapping from layer j
to layer $j + 1$

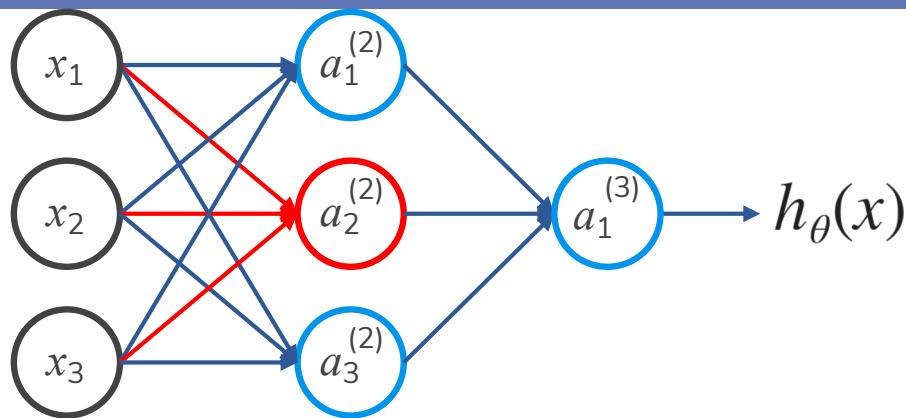

 $a_i^{(j)}$

“activation” of unit i in layer j

 Θ

matrix of weights controlling
function mapping from layer j
to layer $j + 1$

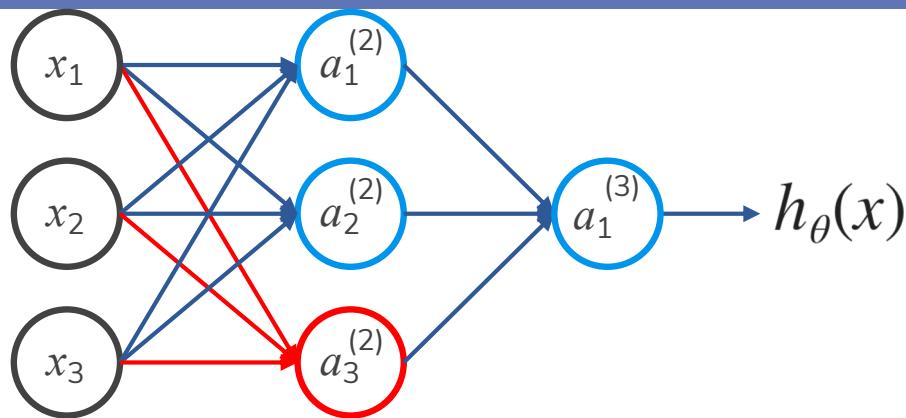
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$



$a_i^{(j)}$ “activation” of unit i in layer j
 Θ matrix of weights controlling
 function mapping from layer j
 to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

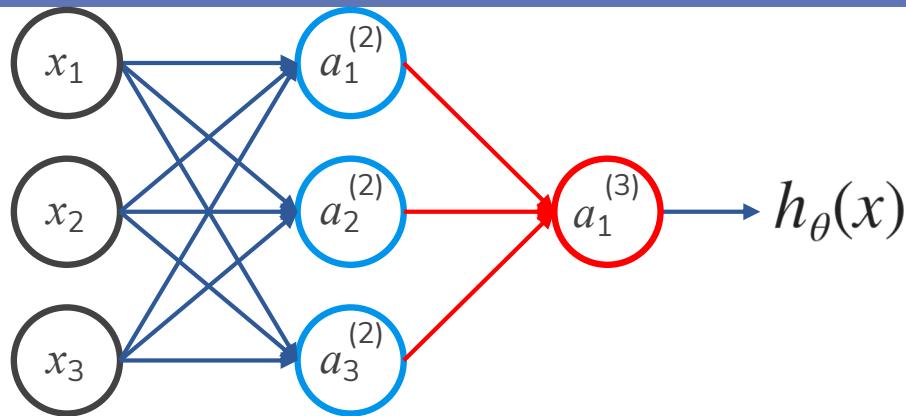


$a_i^{(j)}$ “activation” of unit i in layer j
 Θ matrix of weights controlling
 function mapping from layer j
 to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$



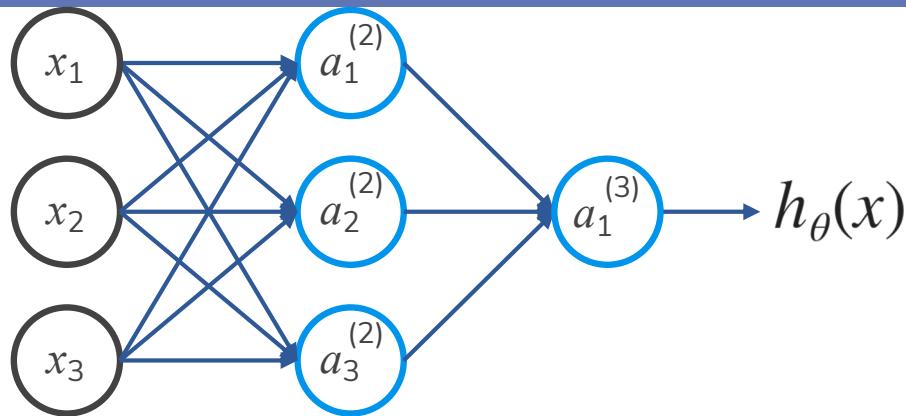
$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling function mapping from layer j to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

 $a_i^{(j)}$

“activation” of unit i in layer j

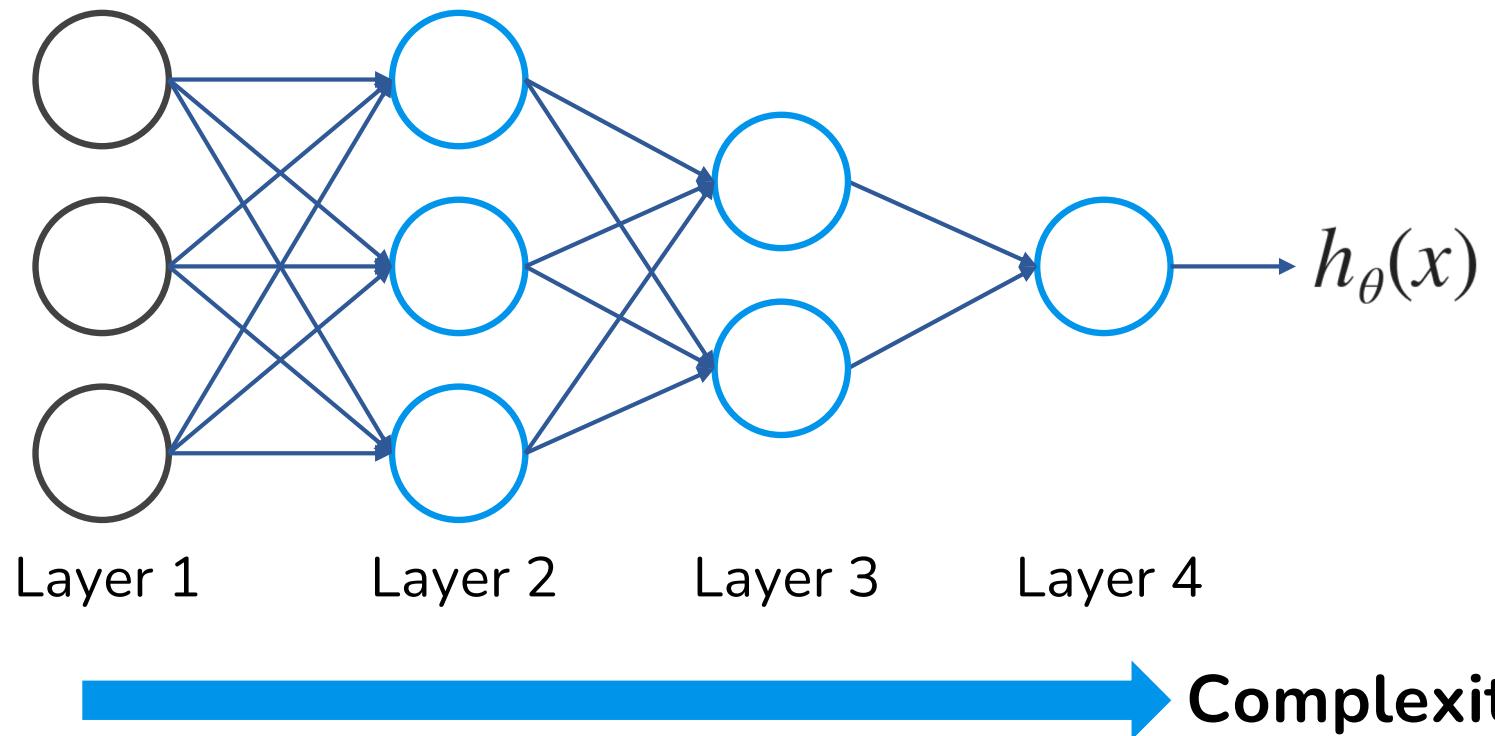
 Θ

matrix of weights controlling
function mapping from layer j
to layer $j + 1$

Feedforward Neural Network (forward propagating)

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

Neural Network Intuition

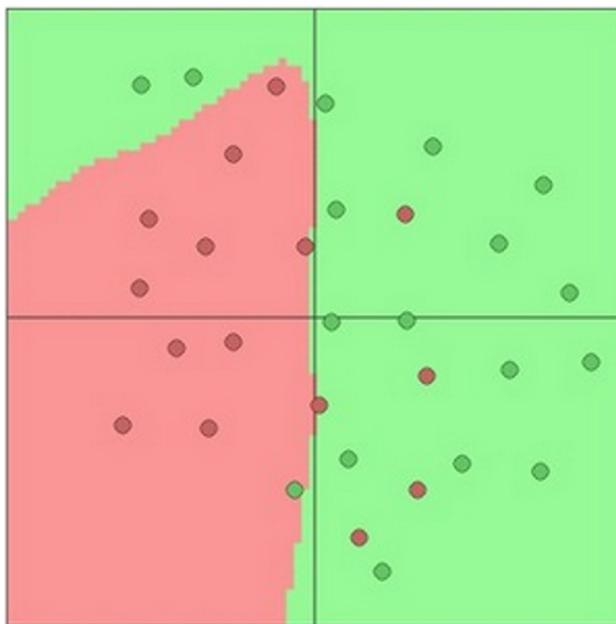


Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

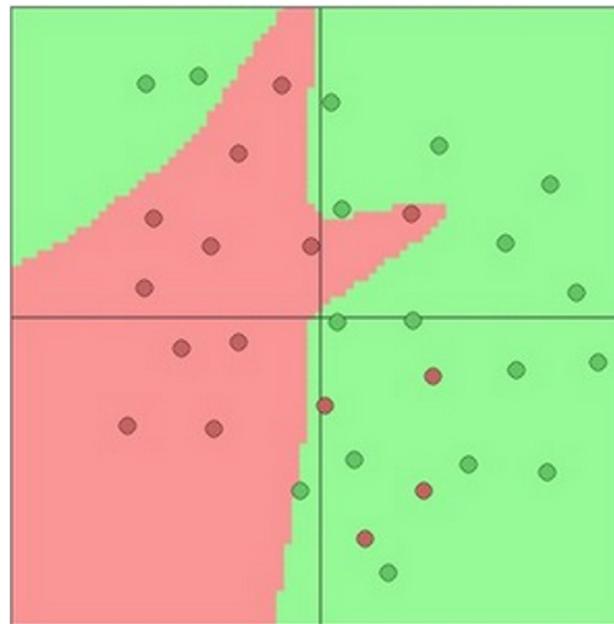
Neural Network Intuition

Toy 2d classification with 2-layer neural network

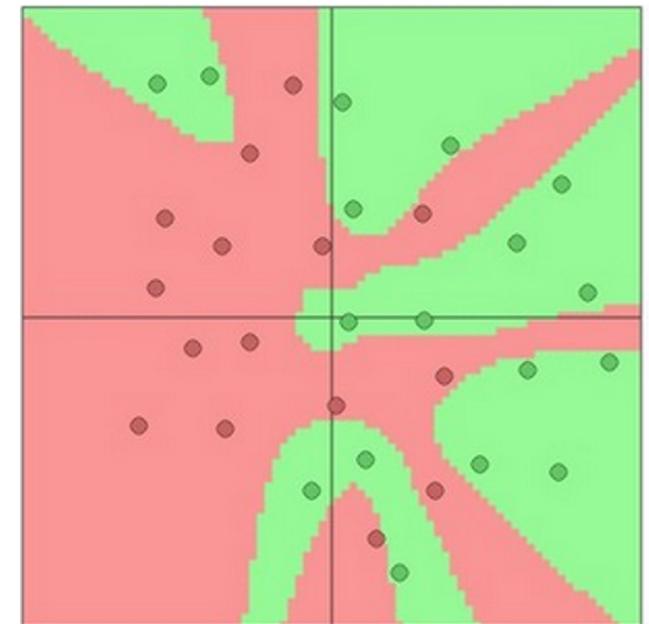
<http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>



3 hidden neurons



6 hidden neurons



20 hidden neurons

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

https://adamharley.com/nv_vis/mlp/2d.html

Draw your number here



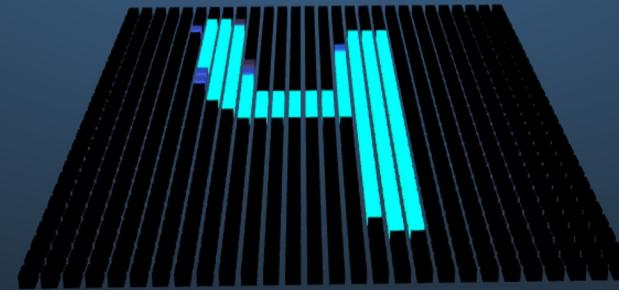
0 1 2 3 4 5 6 7 8 9



Downsampled drawing: 

First guess: 

Second guess: 



Training a Neural Network

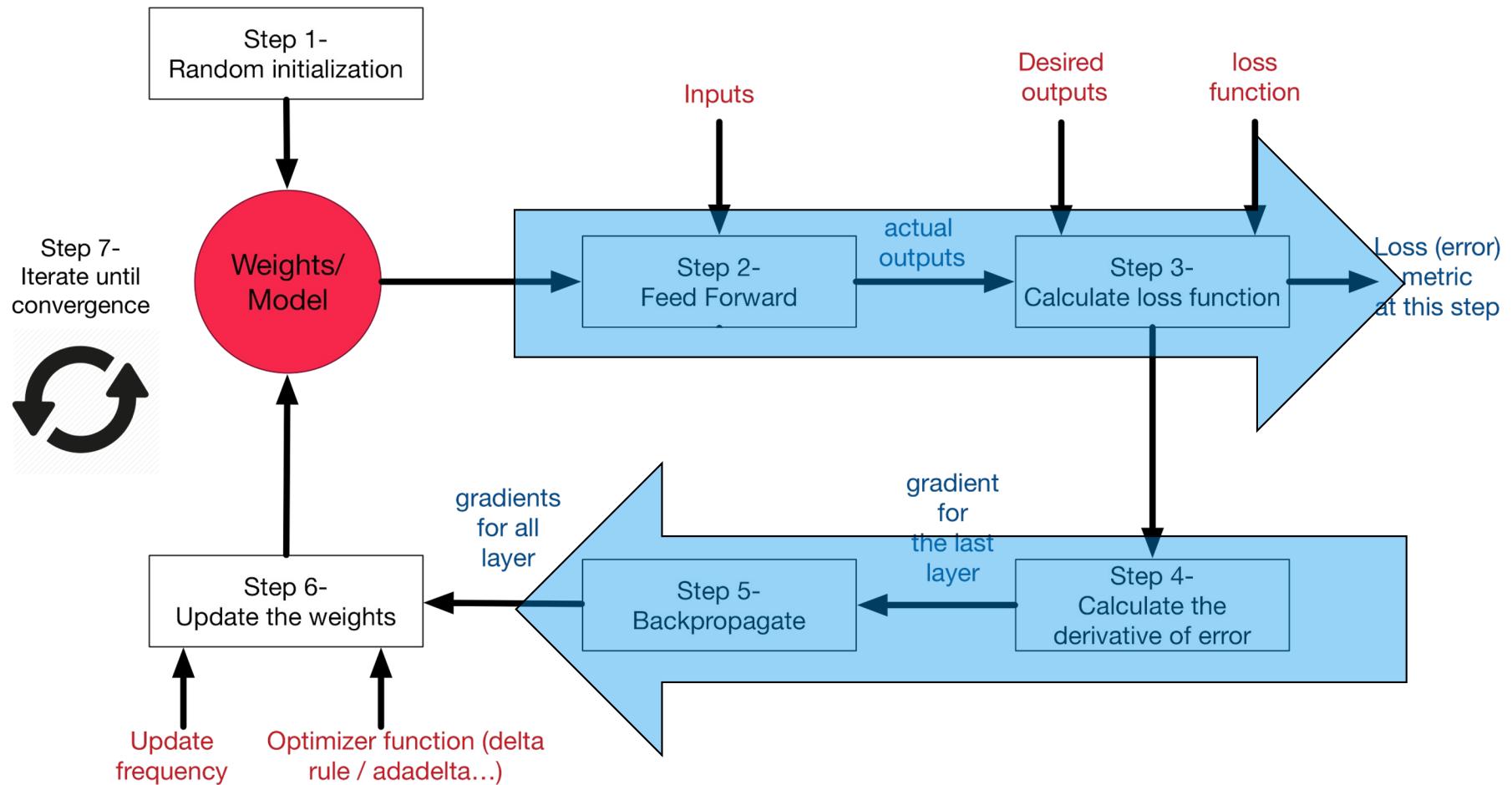
Training a Neural Network

- The first thing we need to do is to “**select**” an architecture.
- **Input units:** dimensionality of the problem (features x)
- **Output units:** Number of classes
- **Hidden units** (per layer)

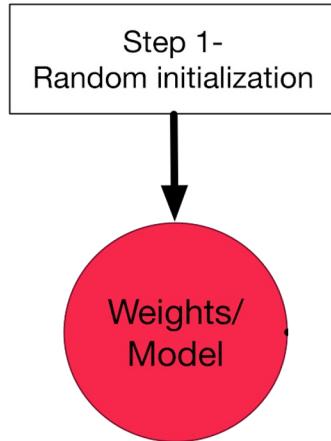
Training a Neural Network

- **Hidden units (per layer):**
 - Usually, the more, the better
 - Good start: a number close to the number of input
 - Default: 1 hidden layer. If you have >1 hidden layer, then it is interesting that you have the **same number of units in every hidden layer.**

Training a Neural Network



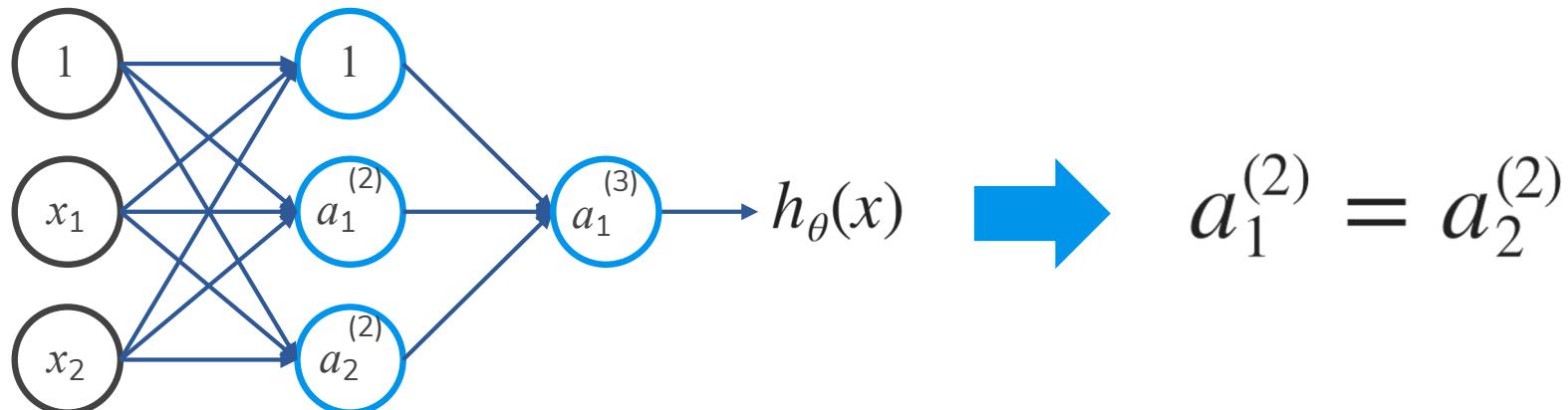
Training a Neural Network



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Zero Initialization

Symmetric Weights



After each update, parameters corresponding to inputs going into each of two hidden units are identical.

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Symmetric Breaking

- We must initialize Θ to a **random value** in $[-\varepsilon, \varepsilon]$
(i.e. $[-\varepsilon \leq \Theta \leq \varepsilon]$)

Today's Initialization

- Xavier initialization [Glorot & Bengio, 2010]:
“Understanding the difficulty of training deep feedforward neural networks”, <http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf>

```
 $\Theta$  = np.random.randn(n) * sqrt(2.0/n)
```

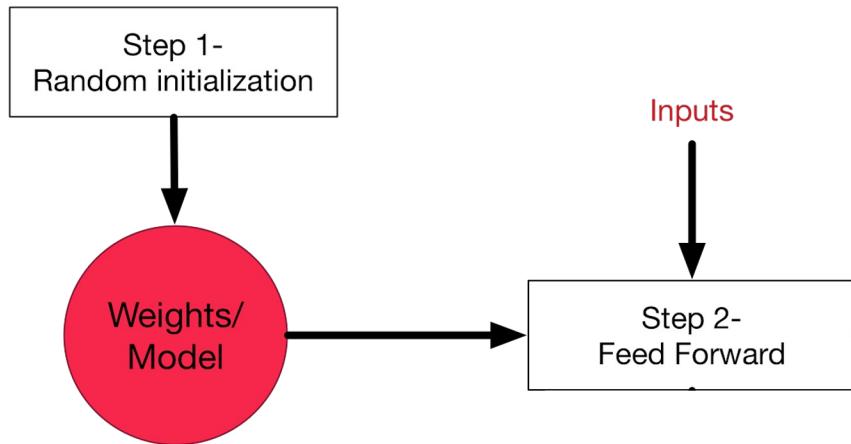
- He initialization [He et al., 2015]: “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification” <https://arxiv.org/pdf/1502.01852>

Today's Initialization

- Xavier initialization [Glorot & Bengio, 2010]:
 $n = \text{input} + \text{output}$
- He initialization [He et al., 2015]:
 $n = \text{input}$

```
 $\Theta$  = np.random.randn(n) * sqrt(2.0/n)
```

Training a Neural Network



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Forward Propagation

Given one training example (x, y) :

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)}a^{(1)}$$

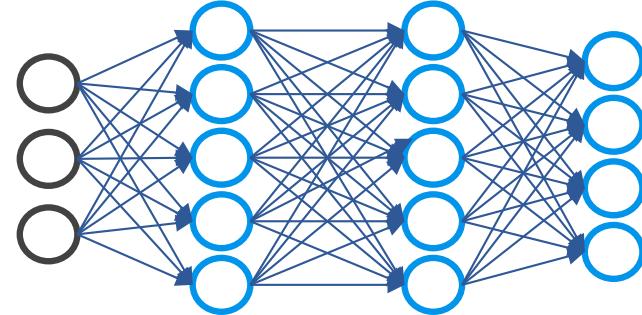
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$

$$z^{(3)} = \Theta^{(2)}a^{(2)}$$

$$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$$

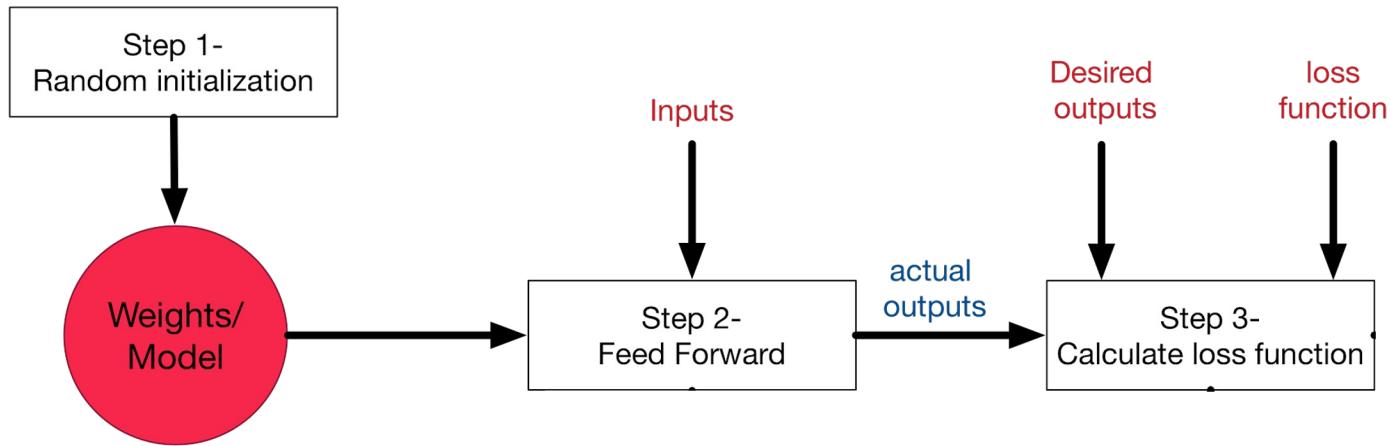
$$z^{(4)} = \Theta^{(3)}a^{(3)}$$

$$a^{(4)} = h_{\Theta}(x) = g(z^{(4)})$$



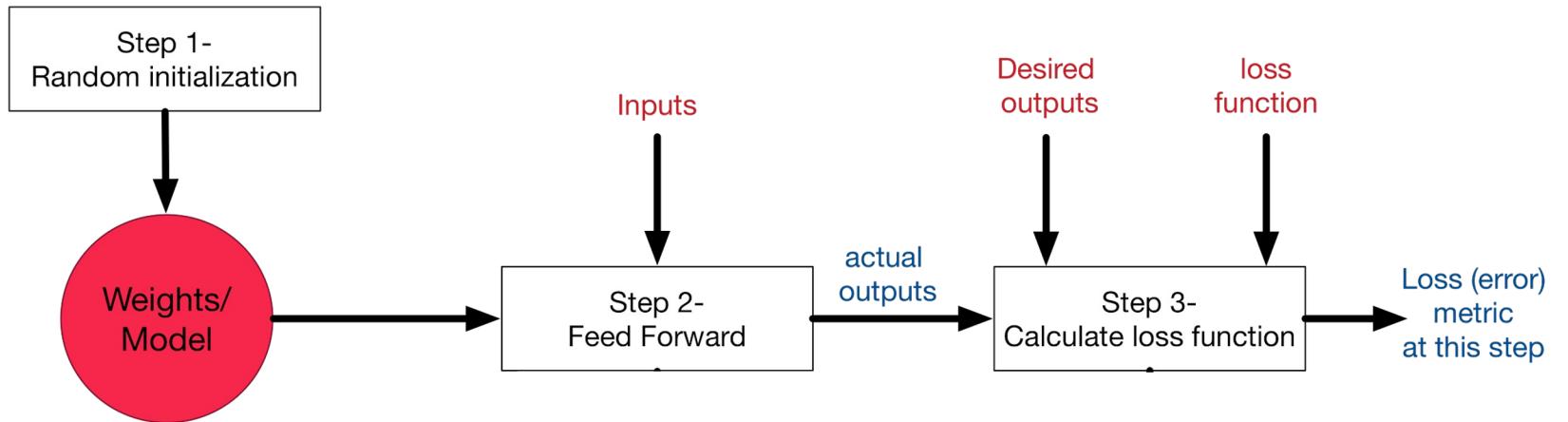
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Training a Neural Network



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Training a Neural Network

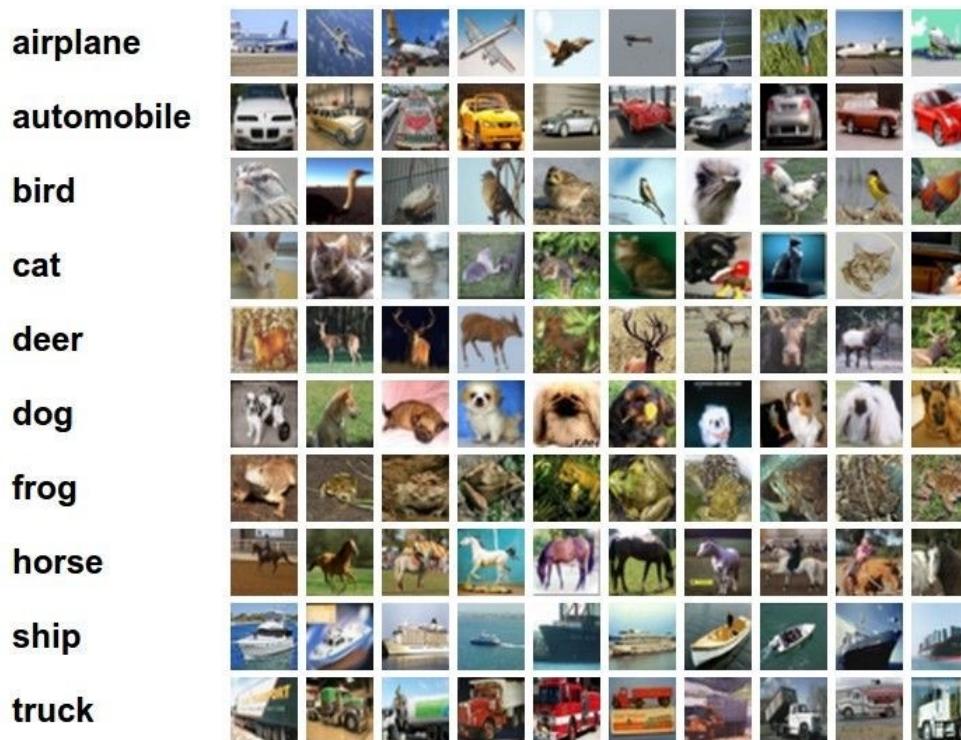


Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

[Loss Function] How do we train deep neural networks?

- The goal is to find such a set of weights that allow the activations/outputs to match the desired output: $f(\Theta, \mathbf{x}_i) \sim y_i$
- Unfortunately, no closed-form solution for weights Θ , but we can express our objective.
- We want to *minimize* a **loss function** (a function of the weights in the network), we'll do so iteratively.
- For now, let's simplify and assume there's a single layer of weights in the network.

Classification goal



Example dataset: **CIFAR-10**

10 labels

50,000 training images
each image is **32x32x3**

10,000 test images

Classification scores

$$f(x, \Theta) = \Theta x$$

$$f(\mathbf{x}, \Theta)$$



10 numbers,
indicating class
scores

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Linear classifier



[32x32x3]
array of numbers 0...1

$$f(x, \Theta) = \Theta x$$

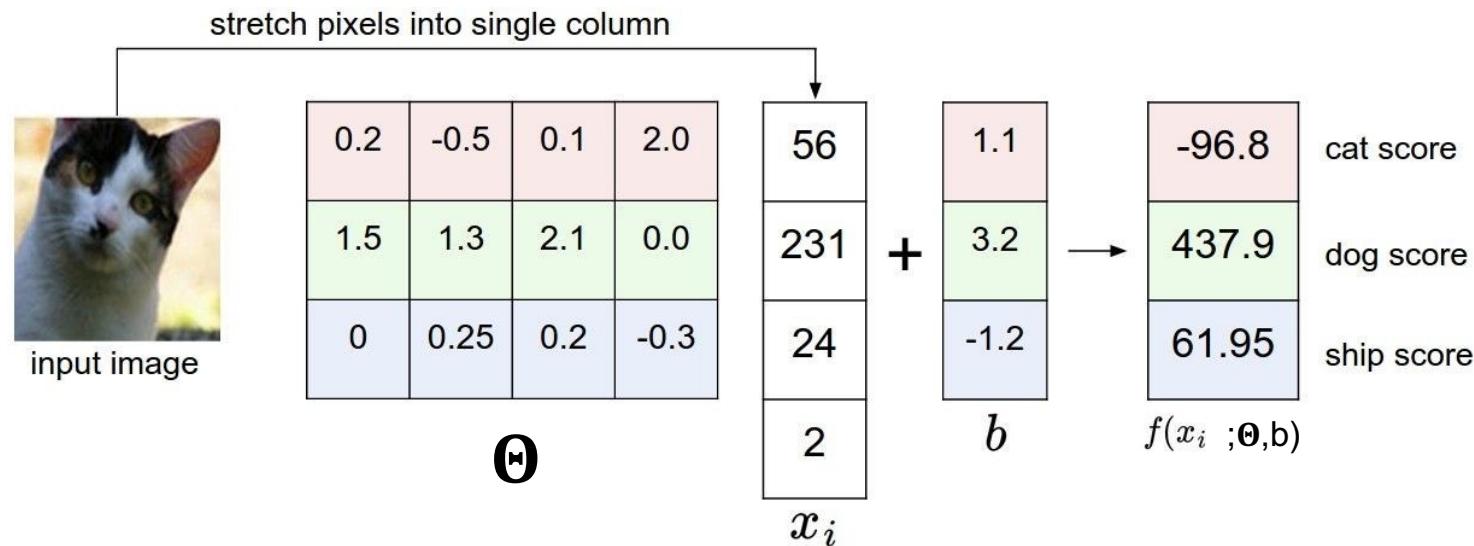
10x1 **10x3072** **3072x1** **(+b)** **10x1**

10 numbers,
indicating class
scores

parameters, or “weights”

Linear classifier

Example with an image with 4 pixels, and 3 classes (**cat/dog/ship**)



Linear classifier

Going forward: Loss function/Optimization



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

TODO:

1. Define a **loss function** that **quantifies** our **unhappiness** with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function (**optimization**)

Linear classifier

Suppose: 3 training examples, 3 classes. With some Θ . The scores

$f(x, \Theta) = \Theta x$: are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes. With some W the scores $f(x, \Theta) = \Theta x$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

Hinge loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x, \Theta)$

the loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Want: $s_{y_i} \geq s_j + 1$, for $j \neq y_i$
 i.e. $s_j - s_{y_i} + 1 \leq 0$

If **true**, loss is 0

If **false**, loss is magnitude of violation

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes. With some Θ the scores $f(x, \Theta) = \Theta x$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
<hr/>			
Losses:	2.9		

Hinge loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x, \Theta)$

the loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 5.1 - 3.2 + 1) \\
 &\quad + \max(0, -1.7 - 3.2 + 1) \\
 &= \max(0, 2.9) + \max(0, -3.9) \\
 &= 2.9 + 0 \\
 &= 2.9
 \end{aligned}$$

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes. With some W the scores $f(x, \Theta) = \Theta x$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	

Hinge loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x, \Theta)$

the loss has the form:

$$\begin{aligned}
 L_i &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\
 &= \max(0, 1.3 - 4.9 + 1) \\
 &\quad + \max(0, 2.0 - 4.9 + 1) \\
 &= \max(0, -2.6) + \max(0, -1.9) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes. With some W the scores $f(x, \Theta) = \Theta x$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Hinge loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x, \Theta)$

the loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 2.2 - (-3.1) + 1) \\
 &\quad + \max(0, 2.5 - (-3.1) + 1) \\
 &= \max(0, 5.3 + 1) + \max(0, 5.6 + 1) \\
 &= 6.3 + 6.6 \\
 &= 12.9
 \end{aligned}$$

Linear classifier: Hinge loss

Suppose: 3 training examples, 3 classes. With some W the scores $f(x, \Theta) = \Theta x$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Hinge loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x, \Theta)$

the loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

and the full training loss is the **mean over all examples in the training data**:

$$L = \frac{1}{N} \sum_{i=1}^N L_i$$

$$\begin{aligned} L &= (2.9 + 0 + 12.9)/3 \\ &= 15.8 / 3 = 5.3 \end{aligned}$$

Linear classifier: Hinge loss

$$f(x, \Theta) = \Theta x$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i ; \Theta)_j - f(x_i ; \Theta)_{y_i} + 1)$$

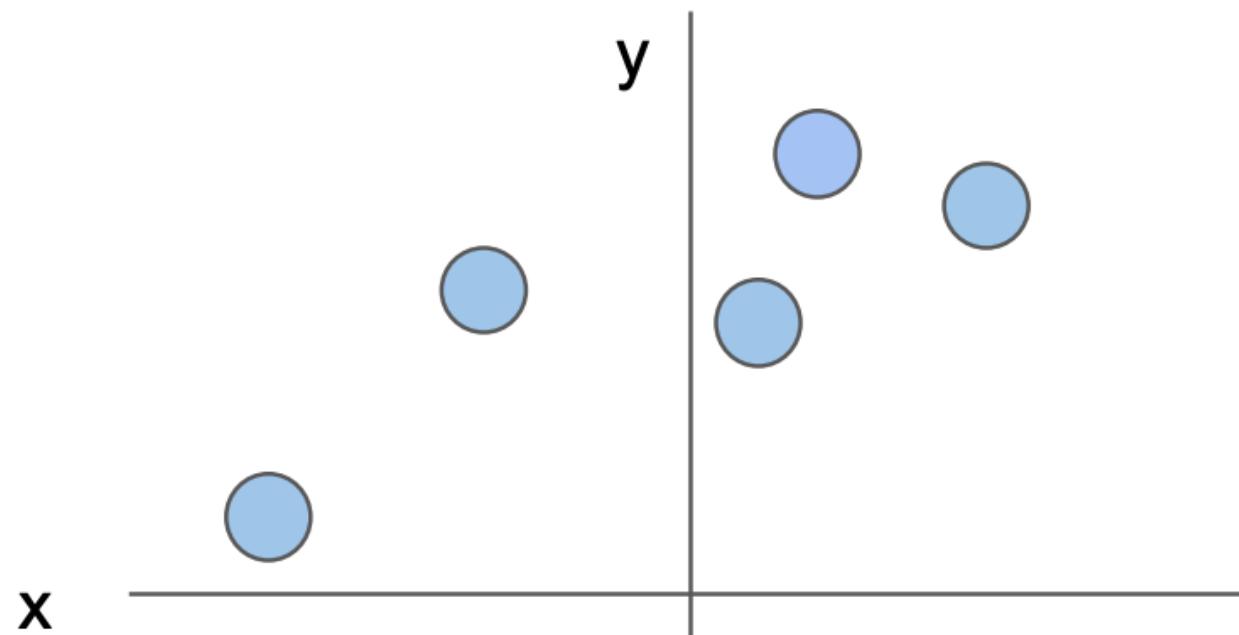
$$L(\Theta) = \frac{1}{N} \sum_{i=1}^N L_i(f(x_i, \Theta), y_i)$$

Linear classifier: Regularization

$$L(\boldsymbol{\theta}) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, \boldsymbol{\theta}), y_i)}_{\text{Data loss}}$$

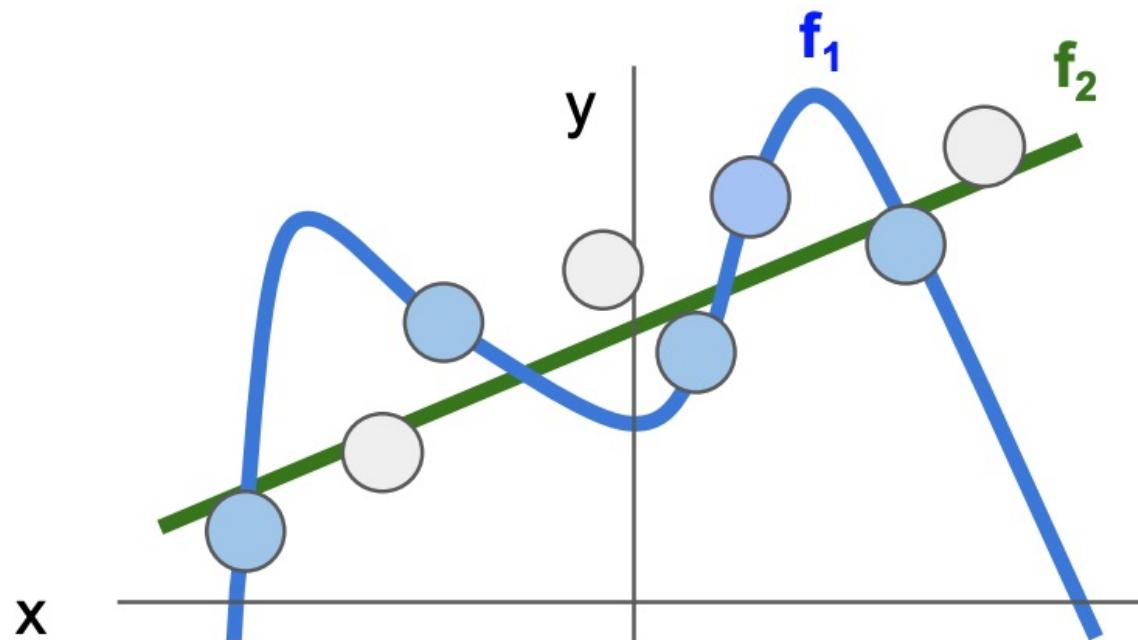
Data loss: Model predictions
should match training data

Linear classifier: Regularization Intuition



Slide Credit: <https://cs231n.stanford.edu/>

Linear classifier: Regularization – Prefer simpler models



Slide Credit: <https://cs231n.stanford.edu/>

Linear classifier: Regularization – Overfitting



Slide Credit: Prof. Sandra Avila - UNICAMP

Linear classifier: Regularization

$$L(\boldsymbol{\theta}) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, \boldsymbol{\theta}), y_i)}_{\text{Data loss: Model predictions}} + \lambda \underbrace{R(\boldsymbol{\theta})}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too well* on training data

Linear classifier: Regularization

$$L(\boldsymbol{\theta}) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, \boldsymbol{\theta}), y_i)}_{\text{Data loss: Model predictions}} + \lambda \underbrace{R(\boldsymbol{\theta})}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too well* on training data

Why Regularization?

- Express preferences over weights
- Make the model simple so it works on test data
- Improve optimization by adding curvature

Linear classifier: Regularization – Express Preferences

$$x = [1, 1, 1, 1]$$

$$\Theta_1 = [1, 0, 0, 0]$$

$$\Theta_2 = [0.25, 0.25, 0.25, 0.25]$$

L2 Regularization

$$R(\Theta) = \sum_k \sum_l \Theta_{k,l}^2$$

Which of Θ_1 or Θ_2 will the L2 regularizer prefer?

$$\Theta_1^T x = \Theta_2^T x = 1$$

Linear classifier: Hinge loss

Weight Regularization

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; \Theta)_j - f(x_i; \Theta)_{y_i} + 1) + \lambda R(\Theta)$$

λ = regularization strength
(hyperparameter)

In common use:

L2 regularization

L1 regularization

Dropout (will see later)

$$R(\Theta) = \sum_k \sum_l \Theta_{k,l}^2$$

$$R(\Theta) = \sum_k \sum_l |\Theta_{k,l}|$$

[Extra] Lab 8: Regularization

Section 1



Another loss: Softmax (cross-entropy)



scores = unnormalized log probabilities of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

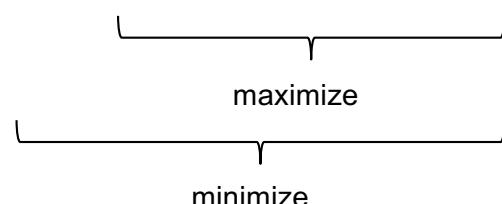
cat	3.2
car	5.1
frog	-1.7

where

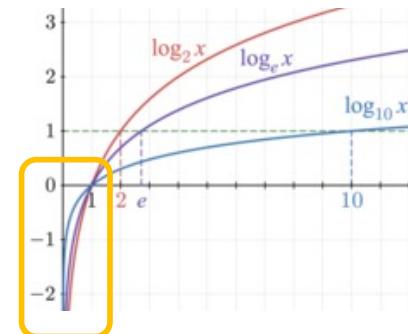
$$s = f(x_i; \Theta)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

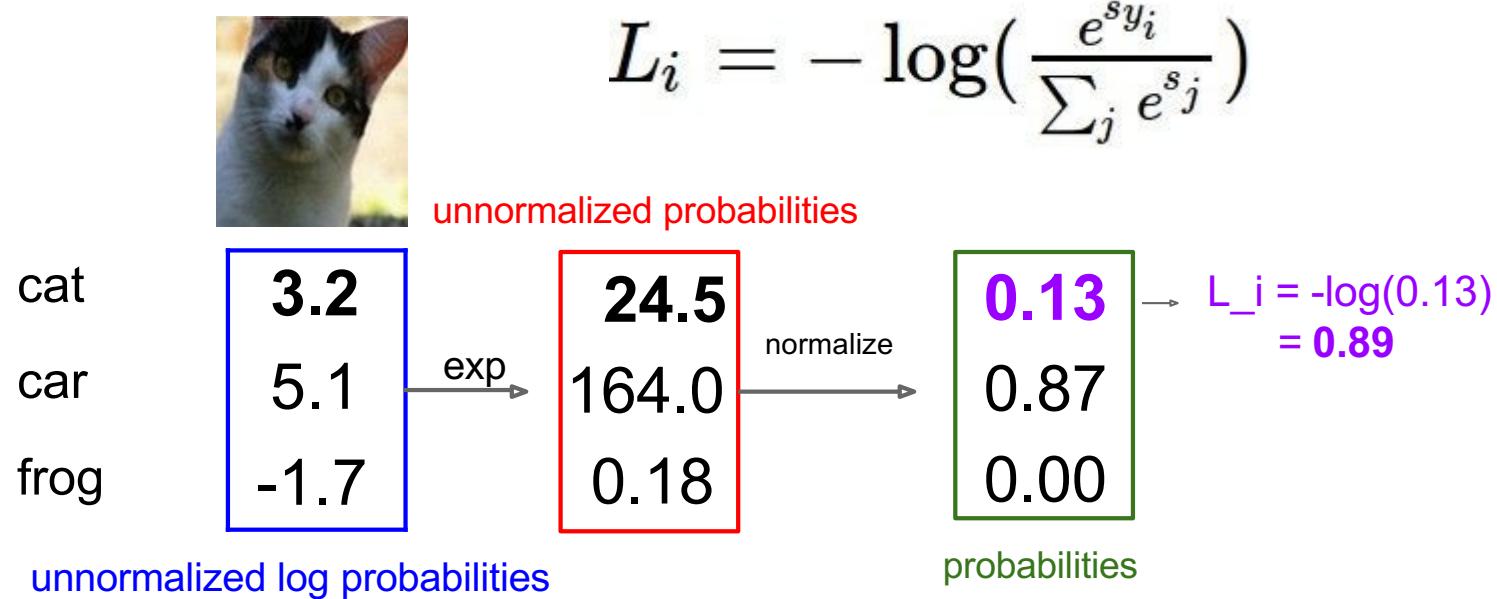
$$L_i = -\log P(Y = y_i|X = x_i)$$



Adapted from Andrej Karpathy

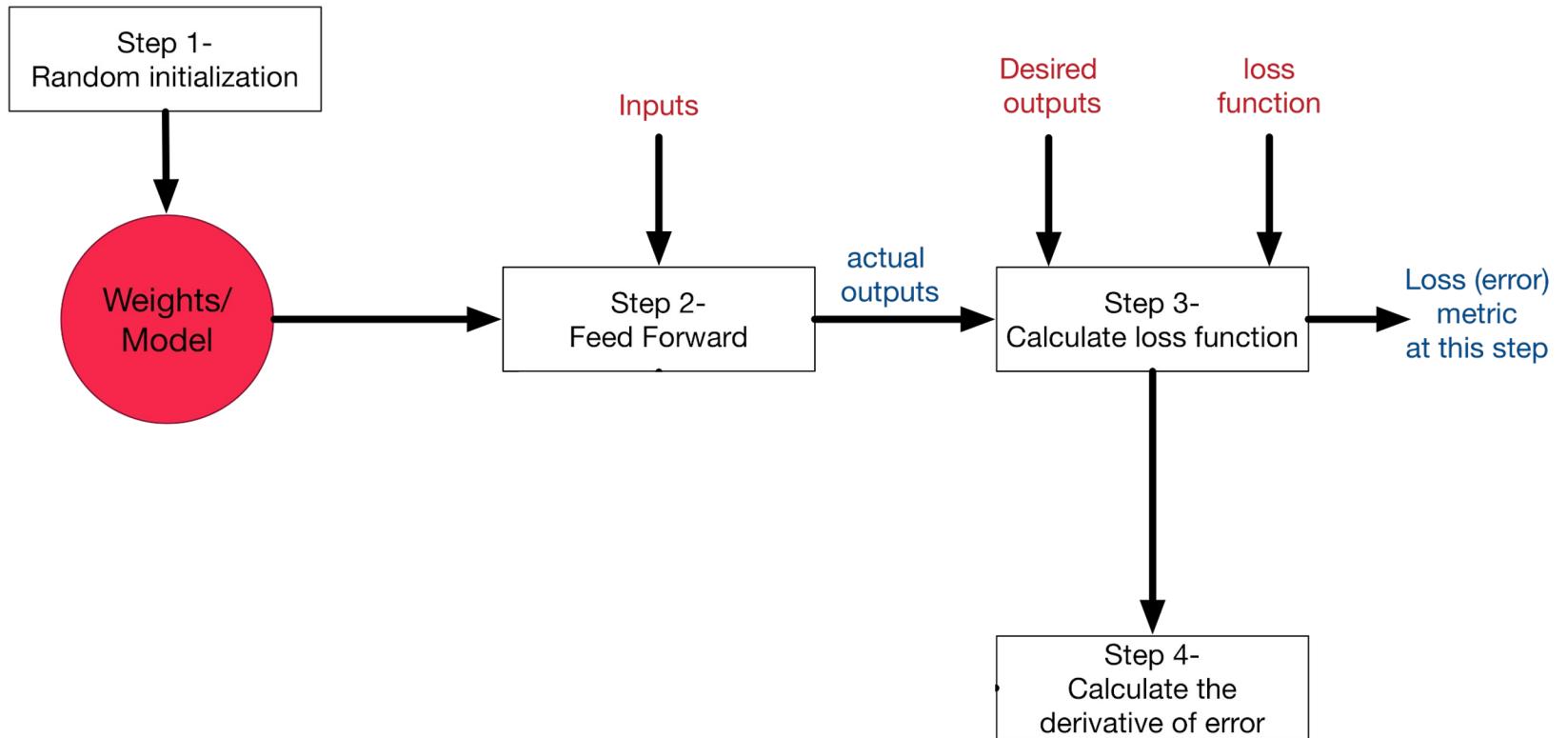


Another loss: Softmax (cross-entropy)



Adapted from Andrej Karpathy

Training a Neural Network



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Gradient Computation: Backpropagation Algorithm

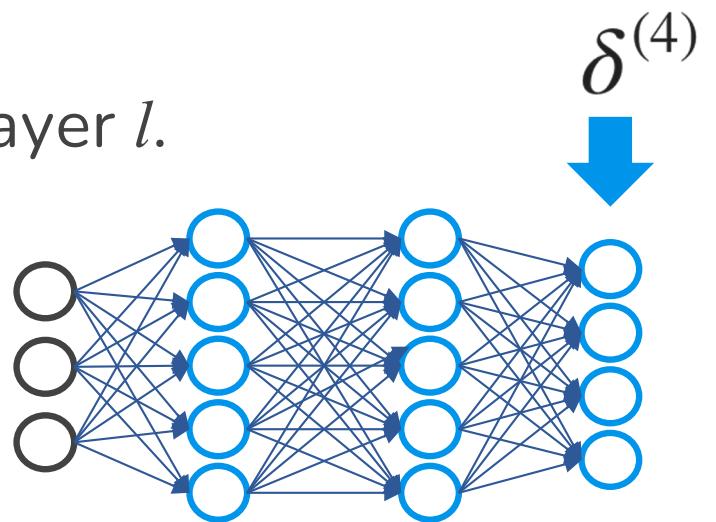
Intuition: $\delta_j^{(l)}$ = “error” of node j in layer l .

For each output unit (layer $L = 4$)

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$



$$(h_{\Theta}(x))_j$$



Gradient Computation: Backpropagation Algorithm

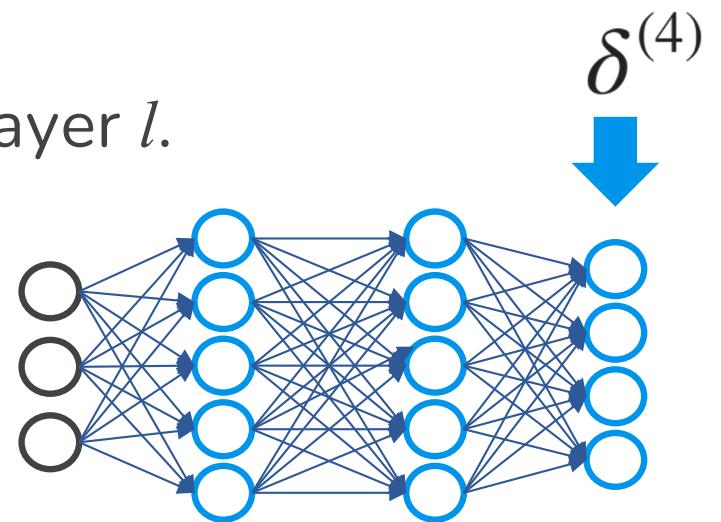
Intuition: $\delta_j^{(l)}$ = “error” of node j in layer l .

For each output unit (layer $L = 4$)

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

Vectorizing it, we have:

$$\delta^{(4)} = a^{(4)} - y$$





NM

Join at menti.com | use code 7783 3151

Mentimeter

Menti

Min_loss

**Choose a slide to present**

Given the mountain, select the minimum altitude



◀ ▶

How to minimize the loss function?



Andrej Karpathy

How to minimize the loss function?

In 1-dimension, the derivative of a function is:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

f: loss function
x: weights Θ
h: small value

In multiple dimensions, the **gradient** is the vector of (partial derivatives):

That is, for $f: \mathbf{R}^n \rightarrow \mathbf{R}$, its gradient $\nabla f: \mathbf{R}^n \rightarrow \mathbf{R}^n$ is defined at the point $p = (x_1, \dots, x_n)$ in n -dimensional space as the vector:^[b]

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}.$$

The **nabla symbol** ∇ , written as an upside-down triangle and pronounced "del", denotes the **vector differential operator**.

Computing the gradient numerically

current Θ :

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

gradient $d\Theta$:

[?,
?,
?,
?,
?,
?,
?,
?,
?,
?,...]

Computing the gradient numerically

current Θ :

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

$\Theta + h$ (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25322

gradient $d\Theta$:

[?,
?,
?,
?,
?,
?,
?,
?,
?,
?,...]

Computing the gradient numerically

current Θ :

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]
loss 1.25347

$\Theta + h$ (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]
loss 1.25322

gradient $d\Theta$:

[-2.5,
?,
?,

$$\frac{(1.25322 - 1.25347)}{0.0001} = -2.5$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

?,
?,...]

Computing the gradient numerically

current Θ :

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]
loss 1.25347

$\Theta + h$ (second dim):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]
loss 1.25353

gradient $d\Theta$:

[-2.5,
?,
?,
?,
?,
?,
?,
?,
?,
?,...]

Computing the gradient numerically

current Θ :

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]
loss 1.25347

$\Theta + h$ (second dim):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]
loss 1.25353

gradient $d\Theta$:

**[-2.5,
0.6,
?,
?,
?,
?,
?,
?,
?]**

$$\frac{(1.25353 - 1.25347)}{0.0001} = 0.6$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

?,...]

Computing the gradient numerically

current Θ :

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]
loss 1.25347

$\Theta + h$ (third dim):

[0.34,
-1.11,
0.78 + **0.0001**,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]
loss 1.25347

gradient $d\Theta$:

[-2.5,
0.6,
?,
?,
?,
?,
?,
?,
?,
?,...]

Computing the gradient analytically

The loss is just a function of Θ :

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \sum_k \|\Theta_k\|^2$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; \Theta) = \Theta x$$

want $\nabla_{\Theta} L$

Computing the gradient analytically

The loss is just a function of Θ :

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \sum_k \Theta_k^2$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; \Theta) = \Theta x$$

want $\nabla_{\Theta} L$

Calculus

$$\nabla_{\Theta} L = \dots$$



Computing the gradient analytically

current Θ :

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

gradient $d\Theta$:

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,...]

$d\Theta = \dots$
(some function of
data and Θ)

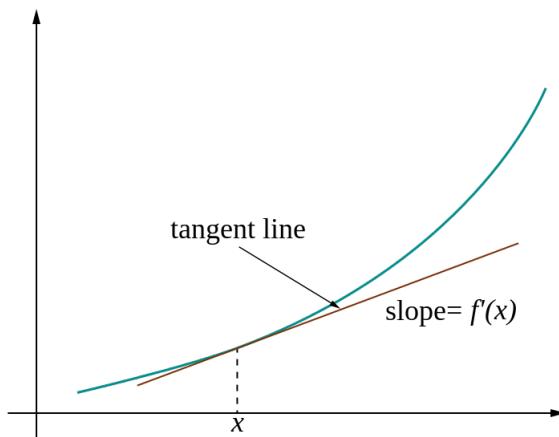


Computing the gradient analytically

- $f(\mathbf{x}, \boldsymbol{\theta}) = \text{dot}(\boldsymbol{\theta}, \mathbf{x}) = \theta_1 * x_1 + \theta_2 * x_2 + \dots + \theta_D * x_D$
- $d f(\mathbf{x}, \boldsymbol{\theta}) / d \theta_i = ?$
- $d f(\mathbf{x}, \boldsymbol{\theta}) / d \theta_1 = x_1$
- $d f(\mathbf{x}, \boldsymbol{\theta}) / d \theta_2 = x_2$
- ...
- Gradient of $f(\mathbf{x}, \boldsymbol{\theta})$ wrt $\boldsymbol{\theta}$ is $[x_1 \ x_2 \ \dots \ x_D]$ i.e. \mathbf{x}

Loss gradients

- Different notations: $\frac{\partial E_n}{\partial \Theta_{ji}^{(1)}}$ $\nabla_{\Theta} L$
- i.e. how does the loss change as a function of the weights
- We want to change weights in a way that makes the loss decrease as fast as possible



Gradient descent

- We'll update weights
- Move in direction opposite to gradient:

$$\Theta^{(\tau+1)} = \Theta^{(\tau)} - \eta \nabla E(\Theta^{(\tau)})$$

↑
Time ↑
Learning rate

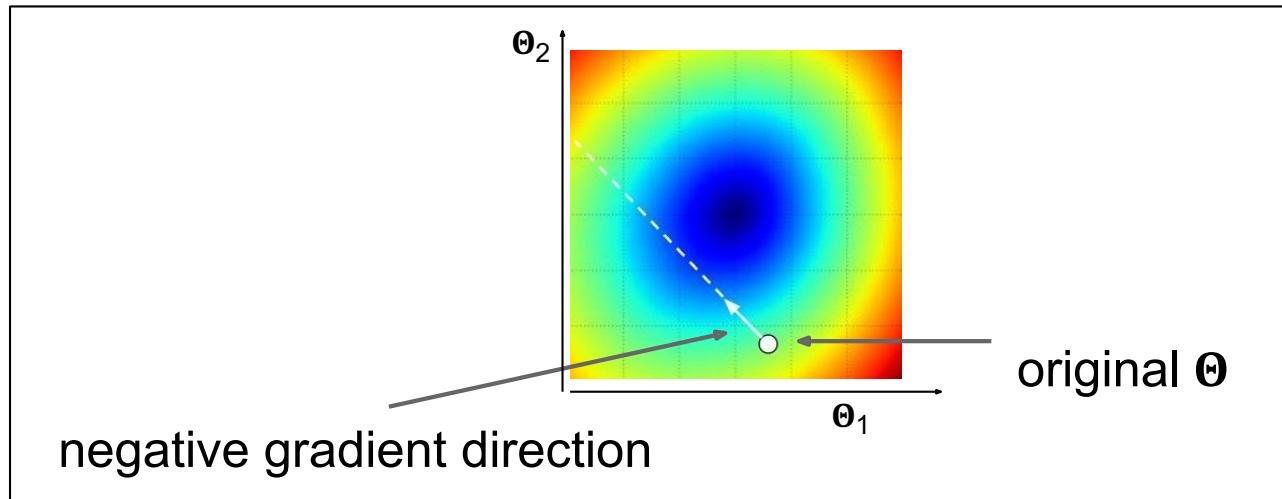
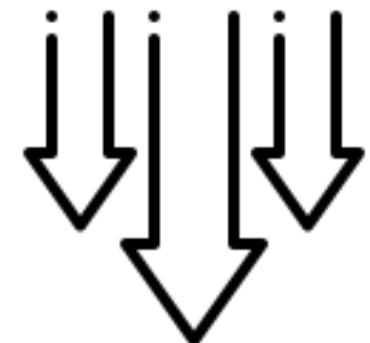


Figure from Andrej Karpathy

Gradient descent

- Iteratively *subtract* the gradient with respect to the model parameters (Θ)
- I.e. we're moving in a direction opposite to the gradient of the loss
- I.e. we're moving towards *smaller* loss



Lab 7: Gradient Descent

Duration: 10 min



To join, go to: ahaslides.com/KN212

AhaSlides



Please, from Lab 7: Gradient Descent [Coding Exercise 2.1], submit the generated regression plot.

Join at:
[ahaslides.com/
KN212](https://ahaslides.com/KN212)

Get Feedback

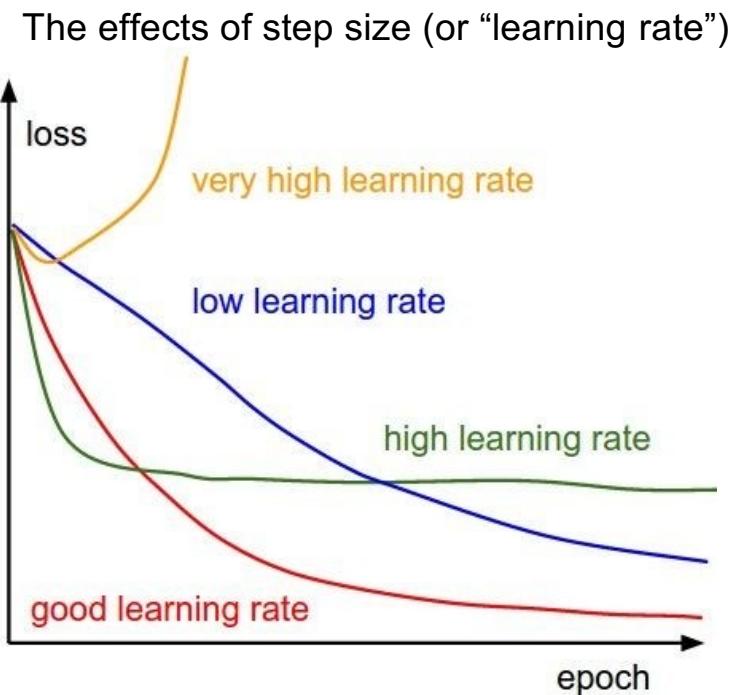
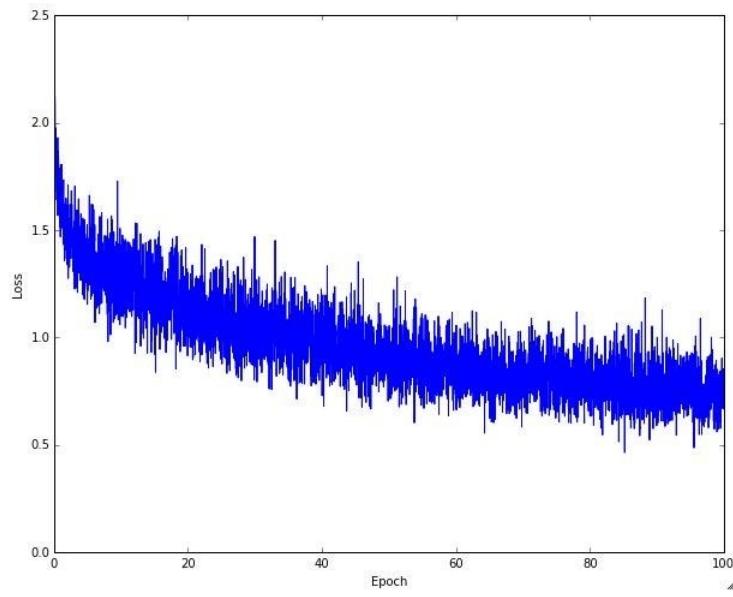
Submissions closed

0 / 0 / 100

How to compute the loss/gradient?

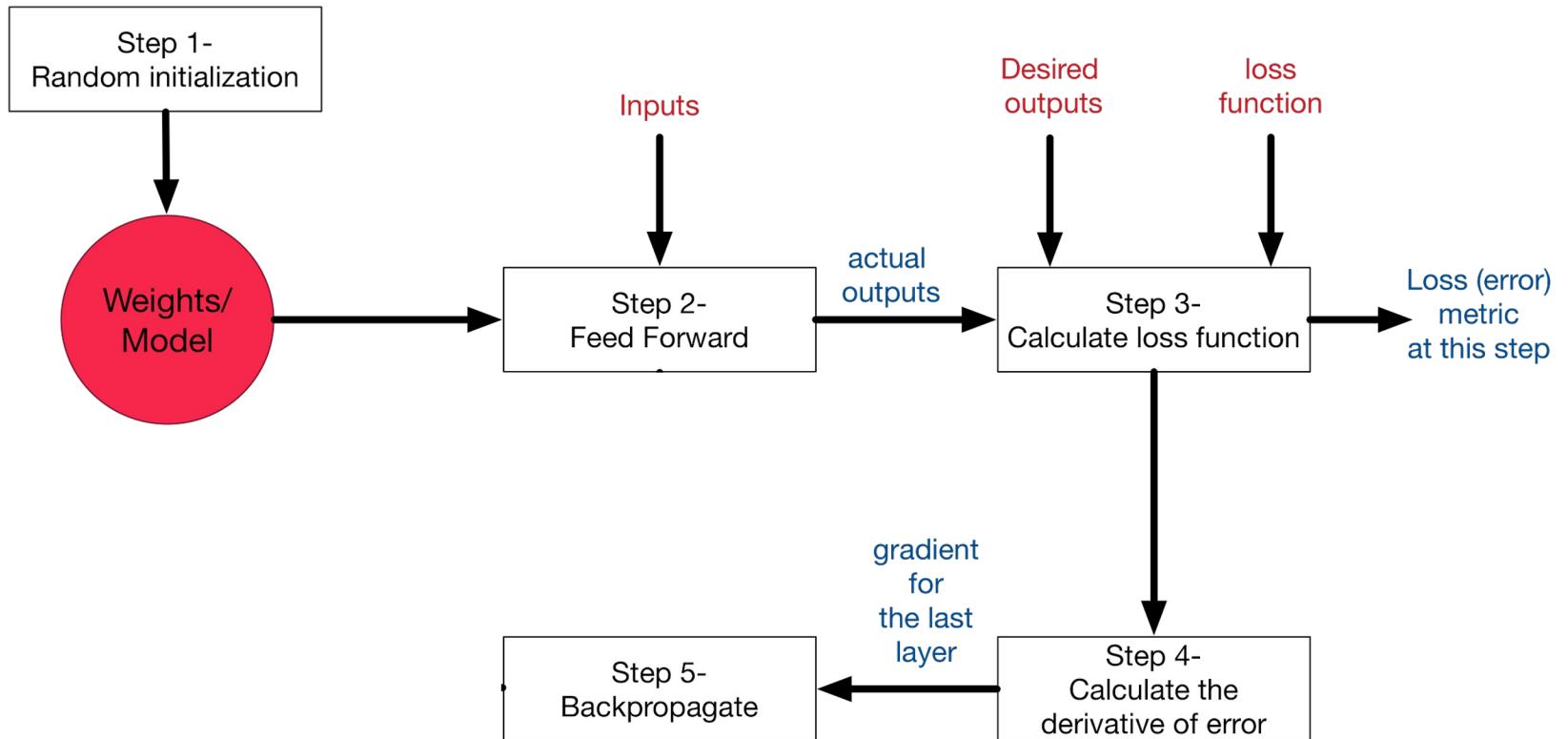
- In **classic gradient descent**, we compute the gradient from the loss for all training examples
- **Mini-batch gradient descent**: Only use *some* of the data for each gradient update, cycle through training examples multiple times
 - Each time we've cycled through all of them once is called an ‘epoch’
 - Allows **faster training** (e.g. on GPUs), **parallelization**
 - Some benefits for learning due to randomness

Learning rate selection



<https://www.deeplearning.ai/ai-notes/optimization/>

Training a Neural Network



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Gradient descent in multi-layer nets

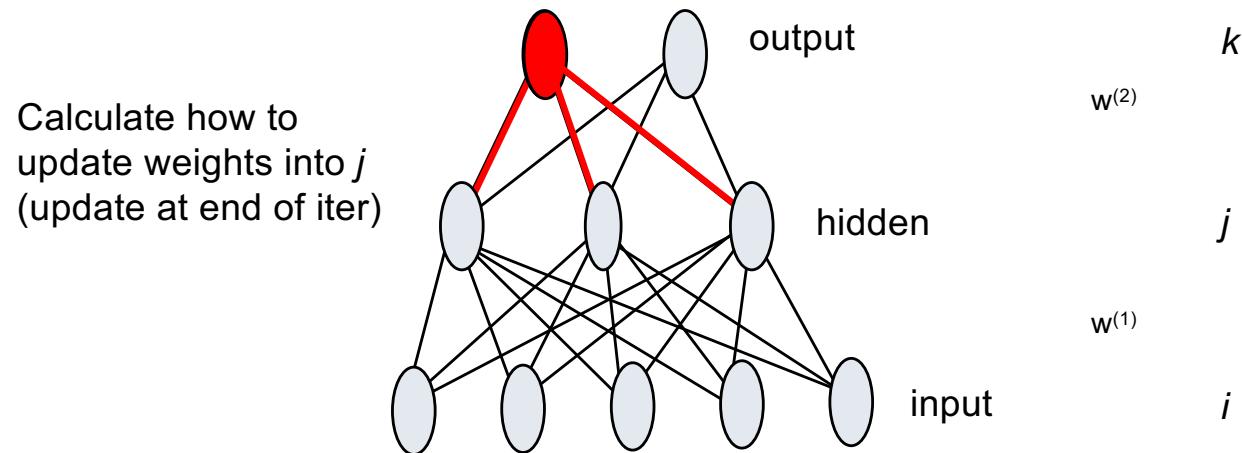
- We'll update weights
- Move in direction opposite to gradient:

$$\Theta^{(\tau+1)} = \Theta^{(\tau)} - \eta \nabla E(\Theta^{(\tau)})$$

- How to update the weights at all layers?
 - **Answer:** backpropagation of error from higher layers to lower layers

Backpropagation: Graphic example

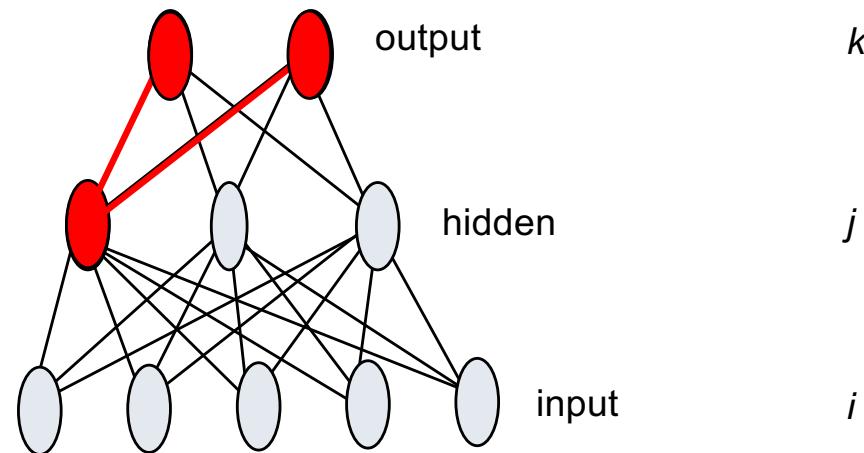
- First calculate error of output units and use this to change the top layer of weights.



Adapted from Ray Mooney, equations from Chris Bishop

Backpropagation: Graphic example

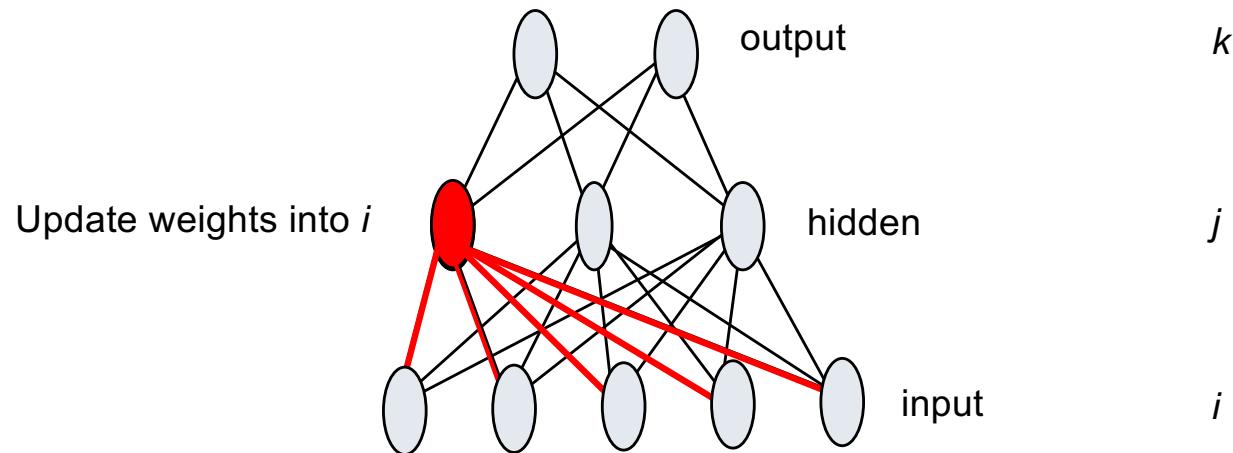
- Next calculate error for hidden units based on errors on the output units it feeds into.



Adapted from Ray Mooney, equations from Chris Bishop

Backpropagation: Graphic example

- Finally update bottom layer of weights based on errors calculated for hidden units.



Adapted from Ray Mooney, equations from Chris Bishop

Backpropagation

A Simple Example

Backpropagation: A Simple Example

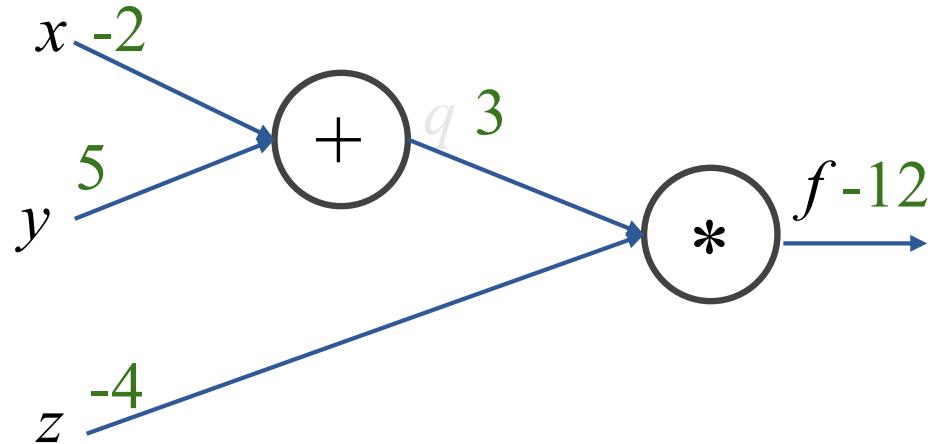
$$f(x, y, z) = (x + y)z$$

e.g., $x = -2, y = 5, z = -4$

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

e.g., $x = -2, y = 5, z = -4$



Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

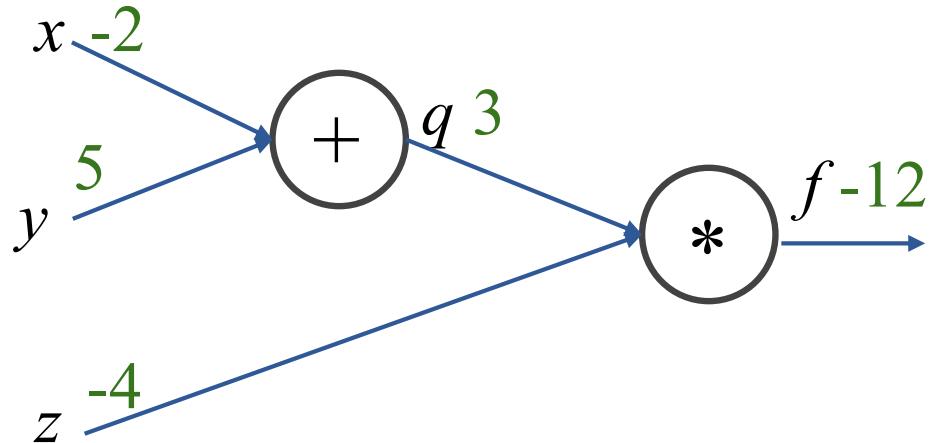
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

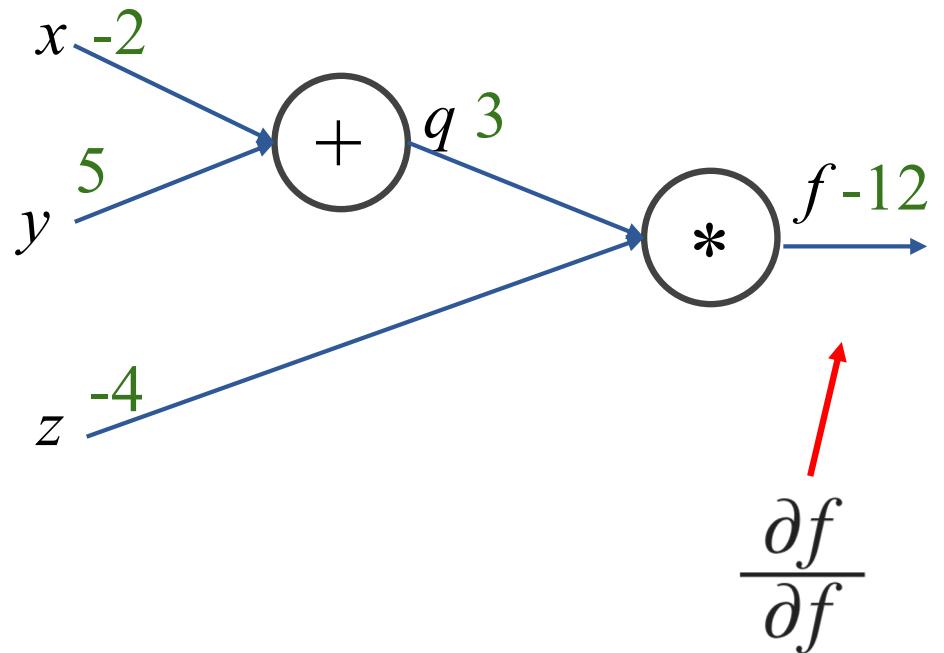
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

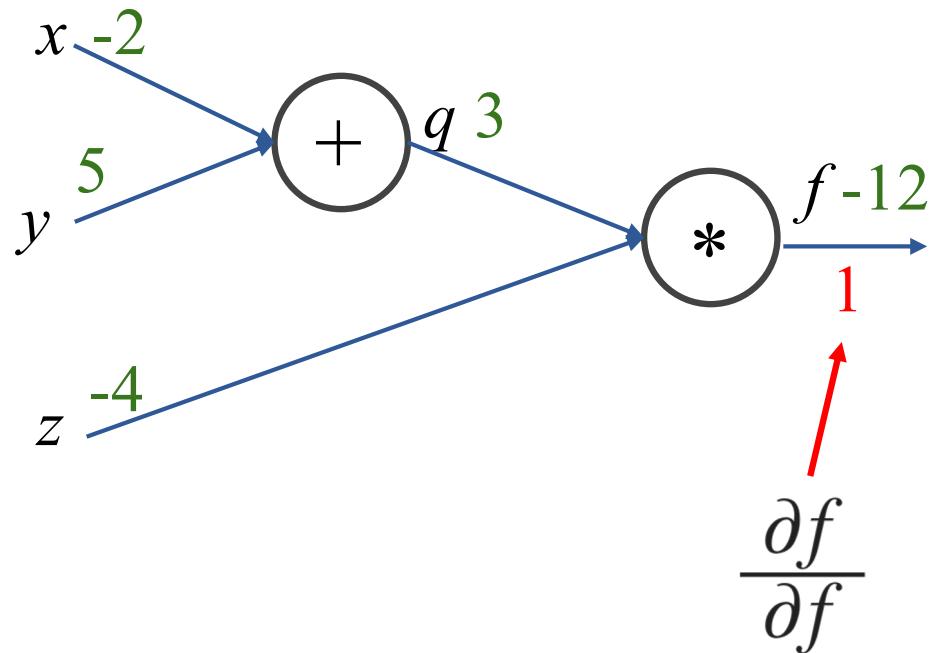
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

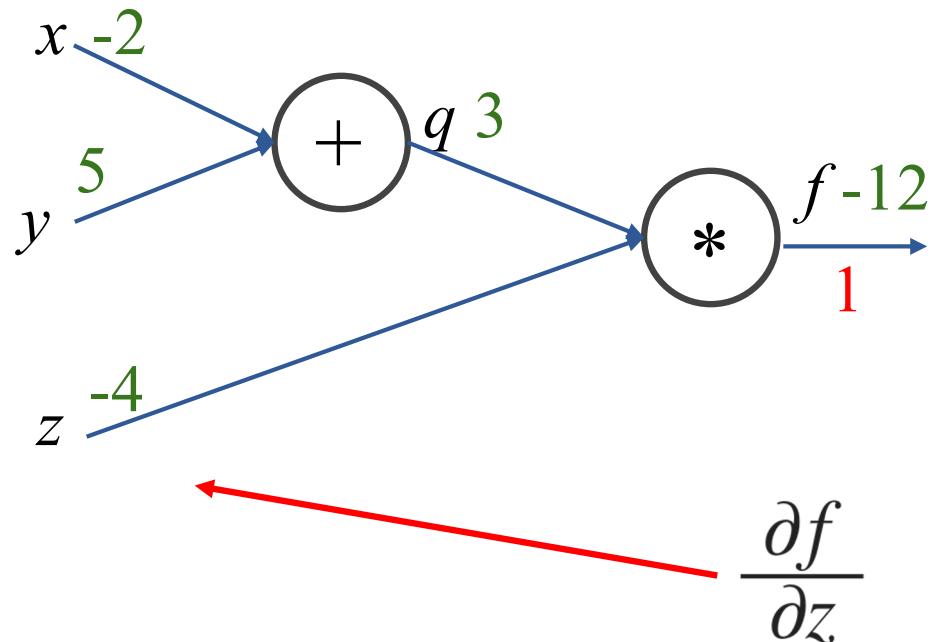
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

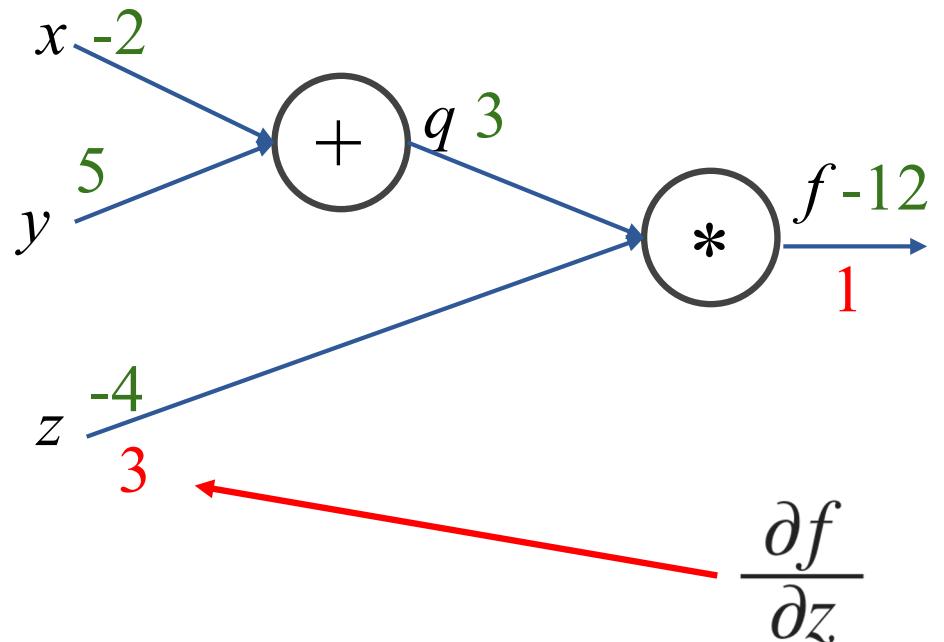
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

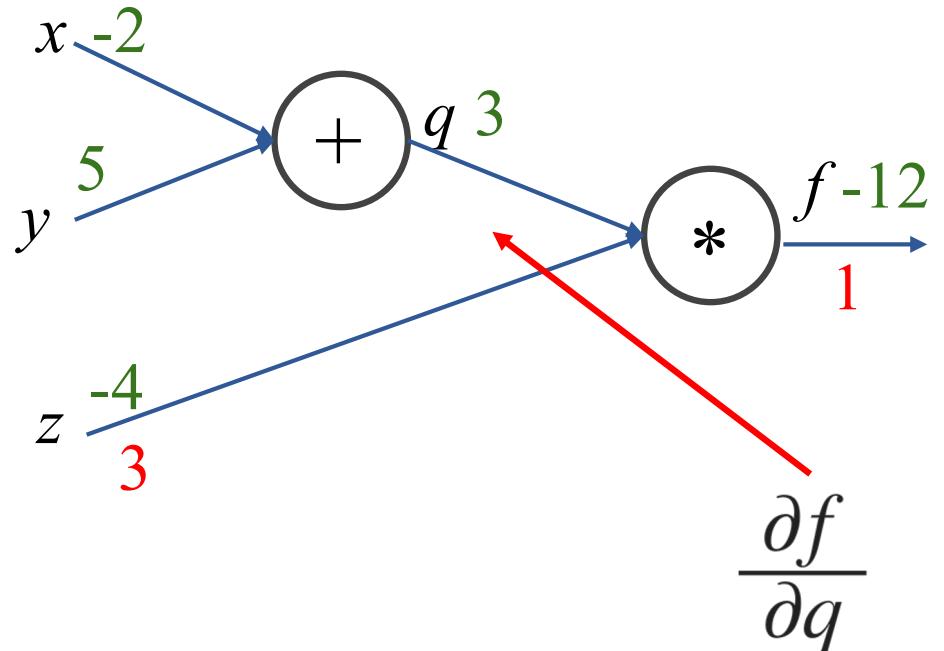
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

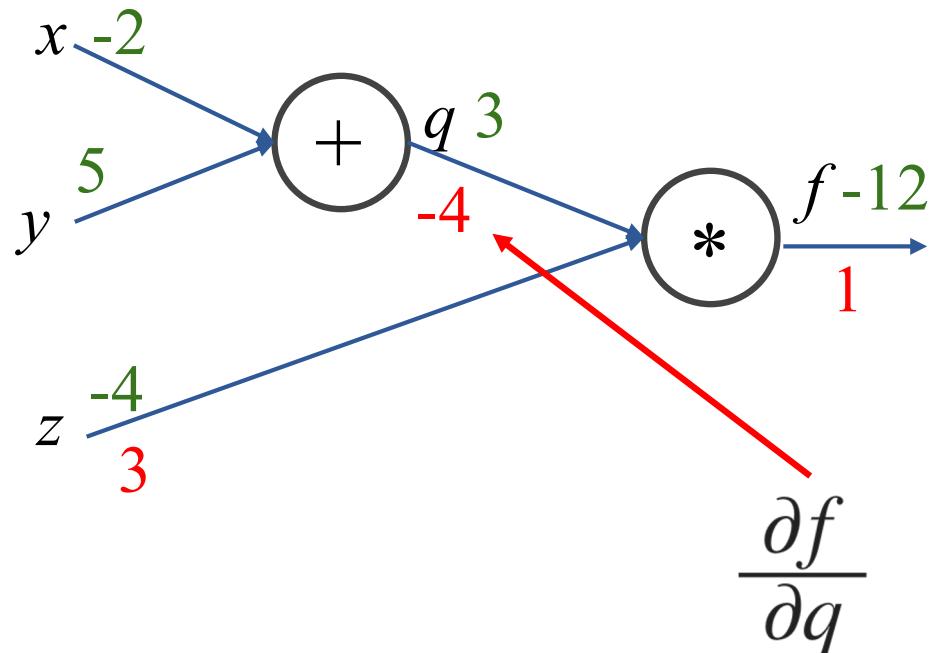
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

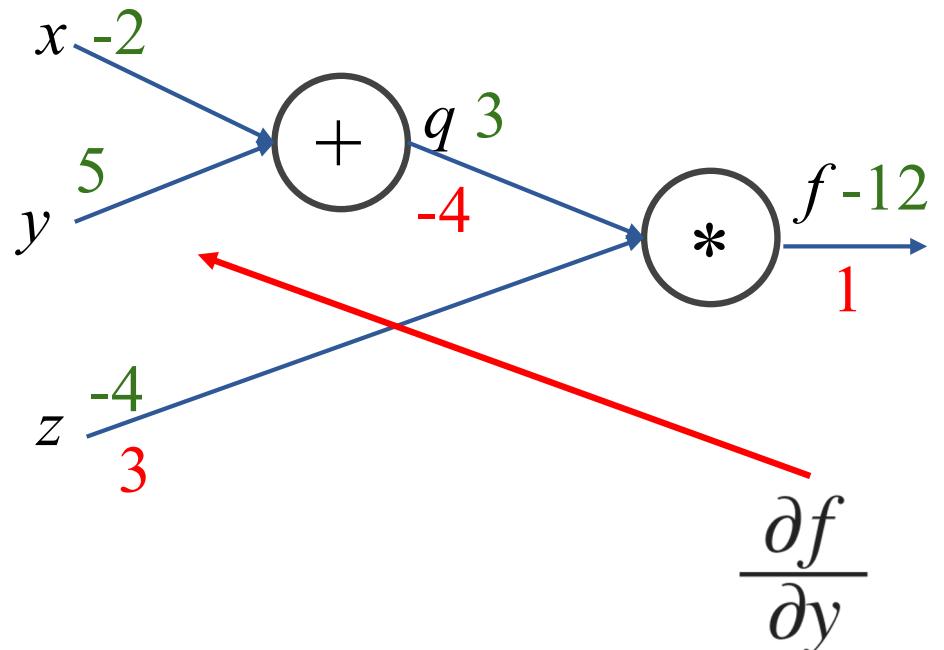
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

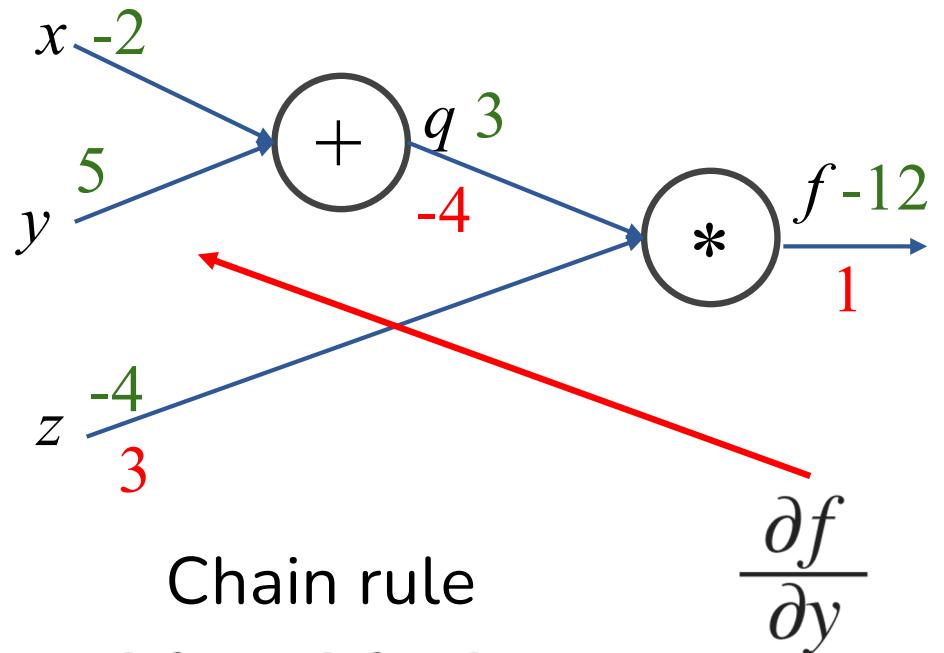
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Chain rule

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

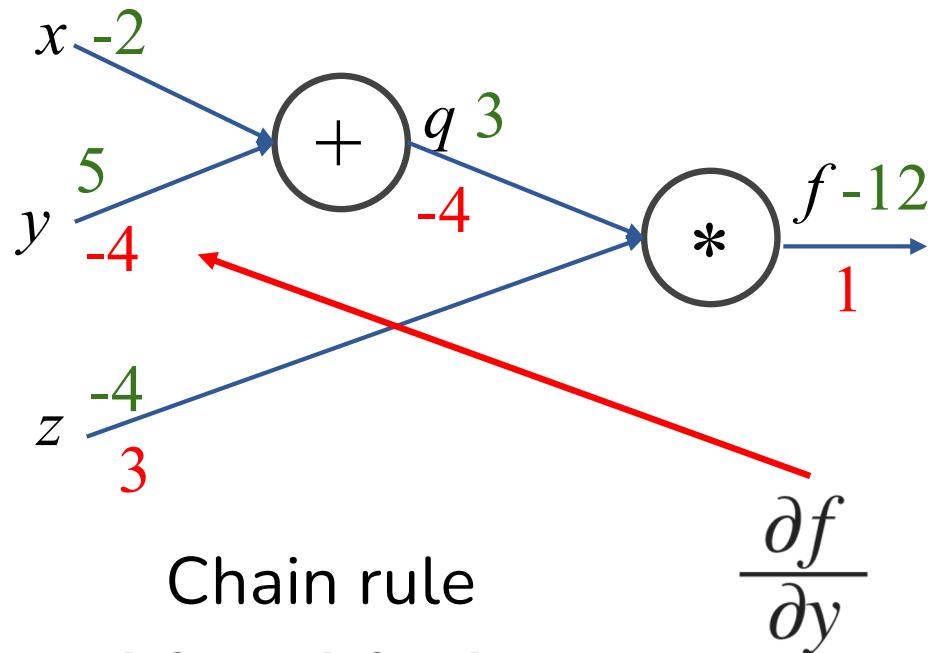
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



$$\frac{\partial f}{\partial y}$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

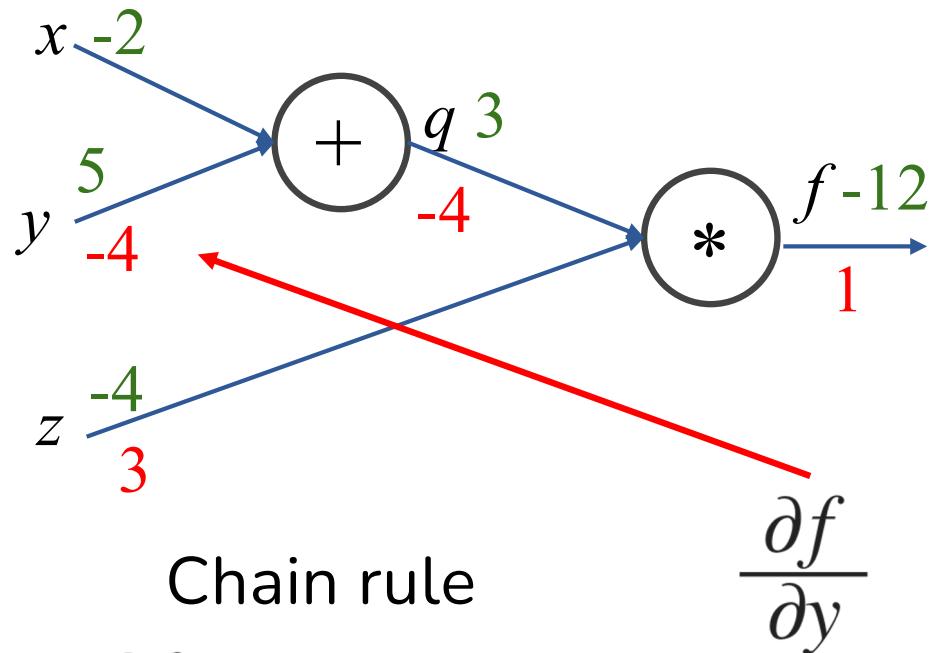
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



$$\frac{\partial f}{\partial y} = -4 \times 1 = -4$$

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

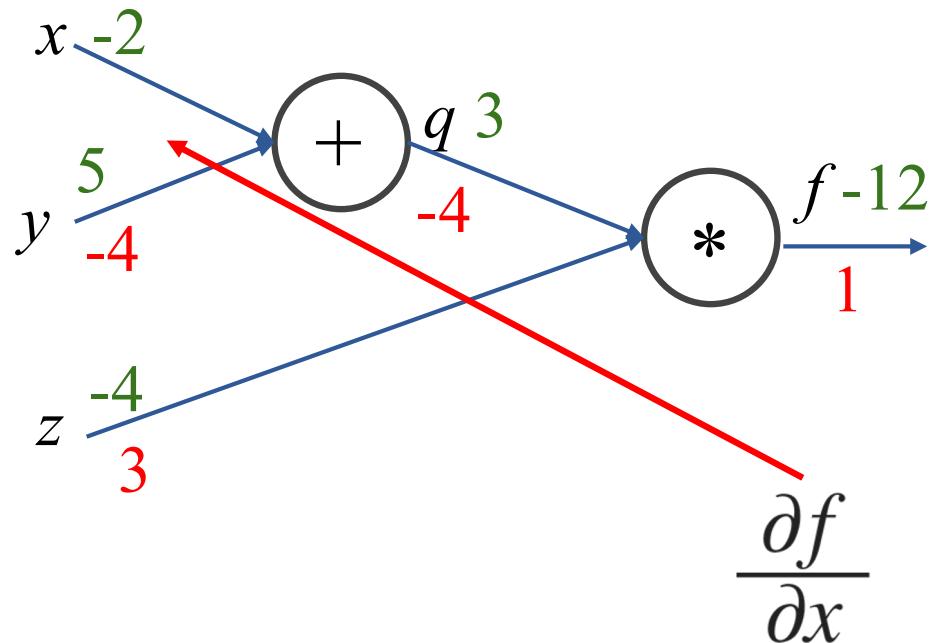
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Andrej Karpathy



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Backpropagation: A Simple Example

$$f(x, y, z) = (x + y)z$$

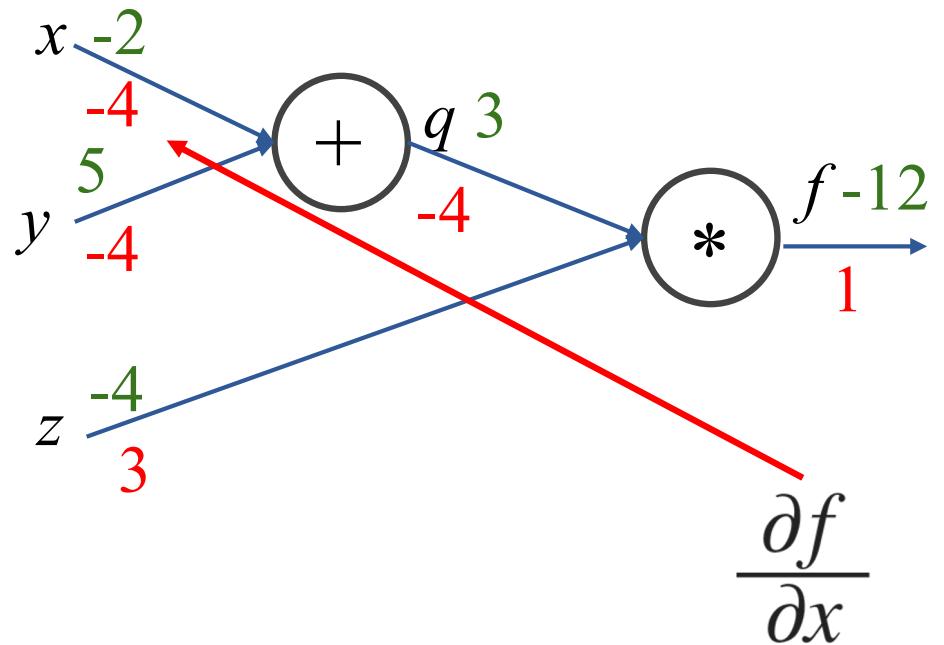
e.g., $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

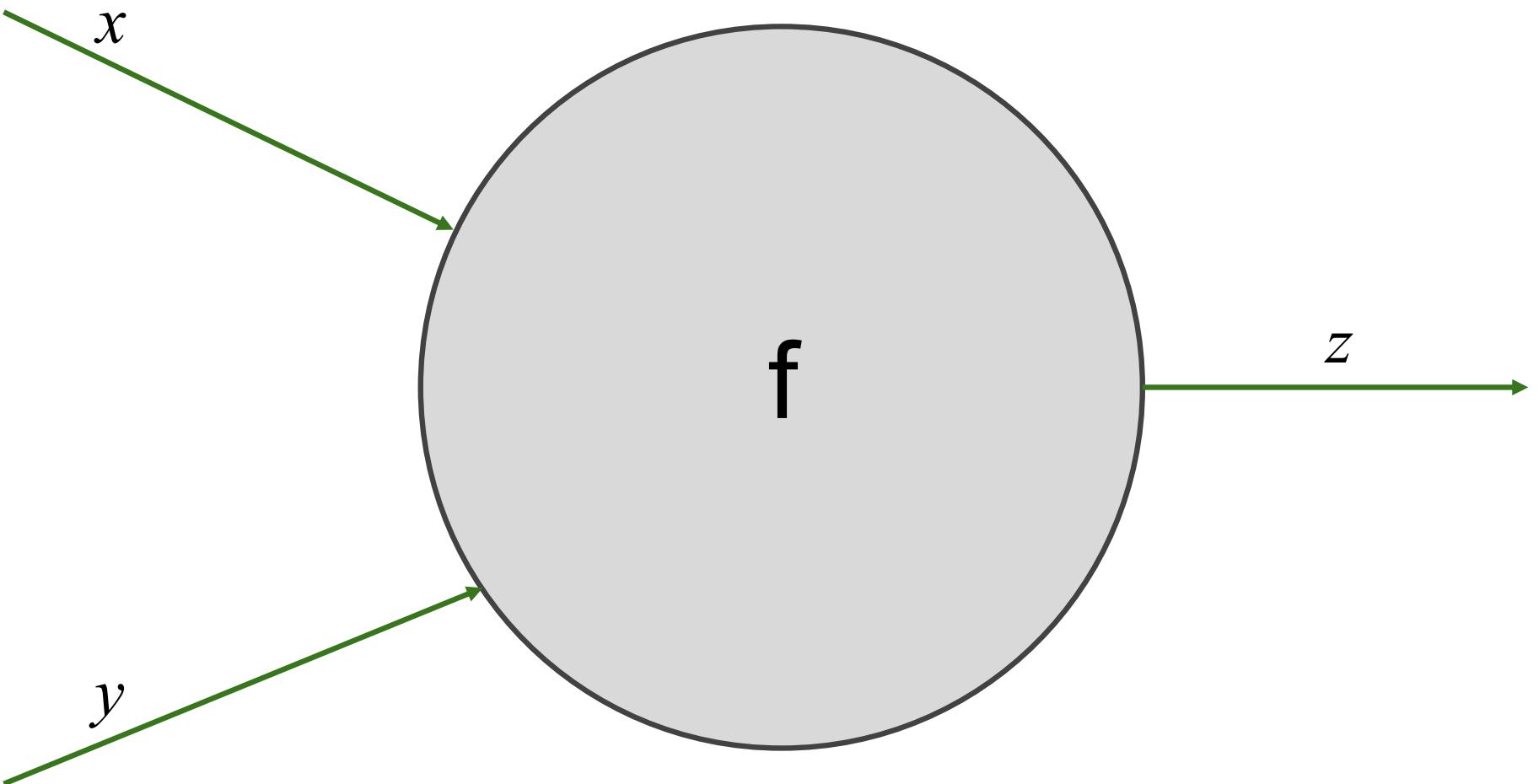
$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

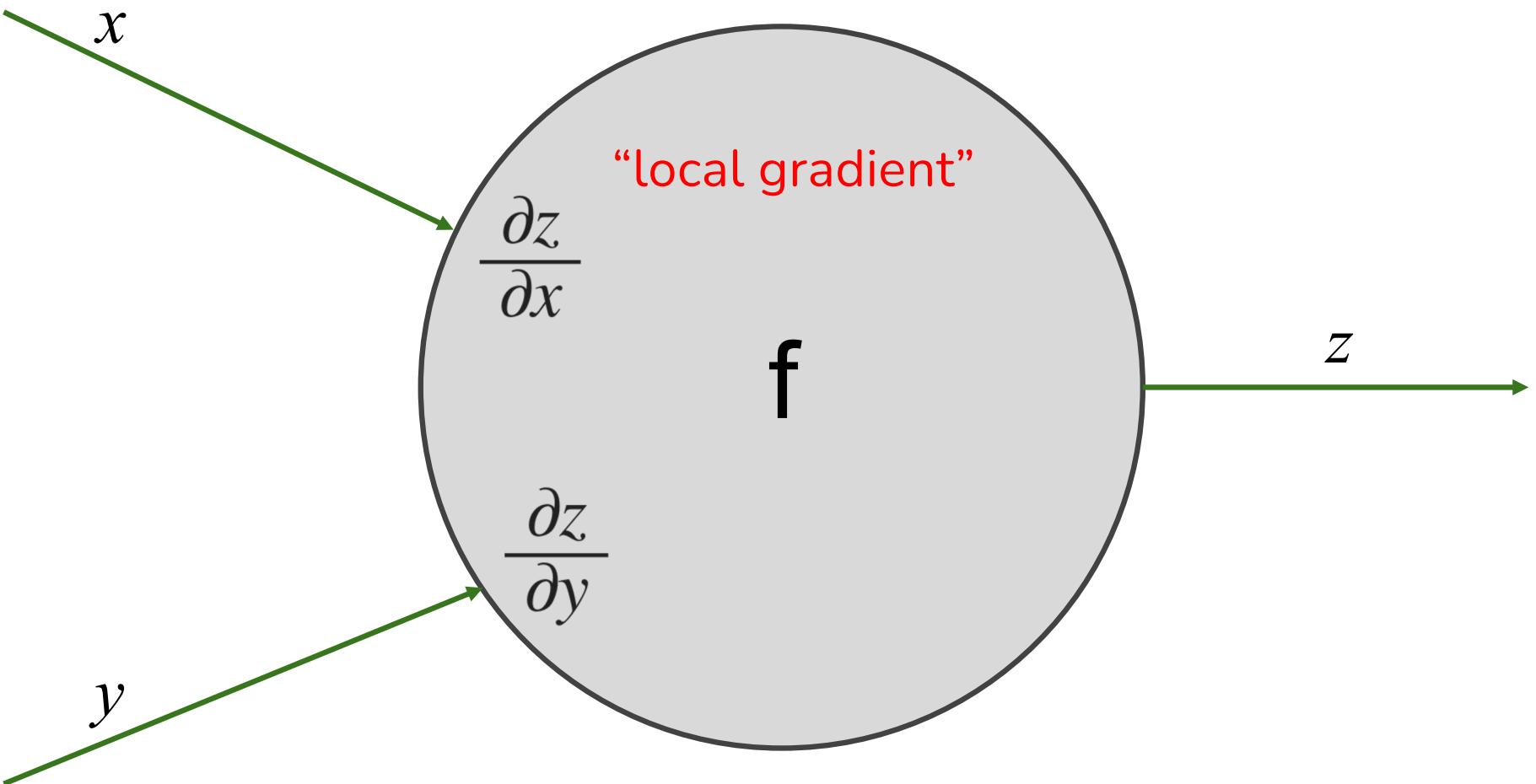
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

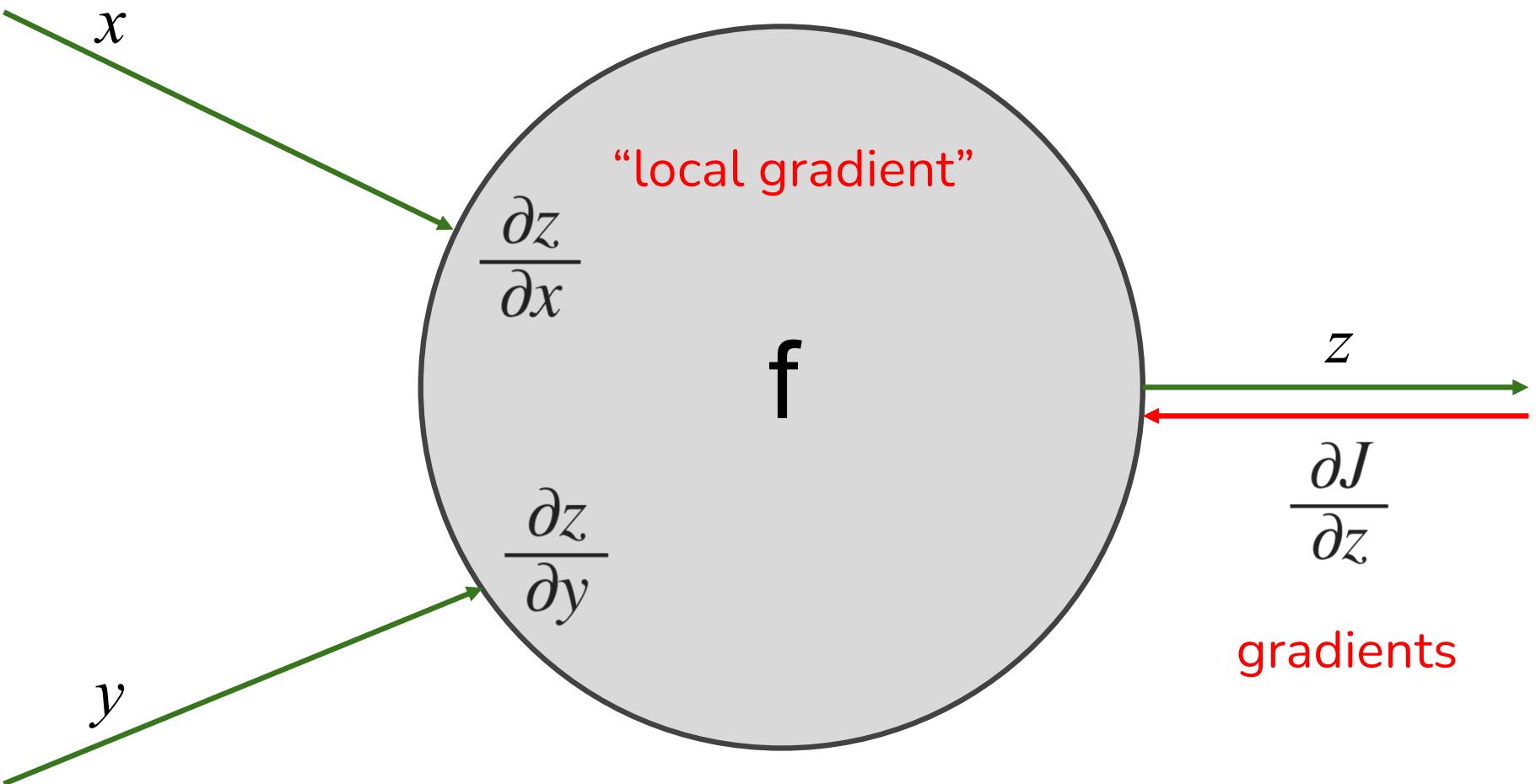
Andrej Karpathy

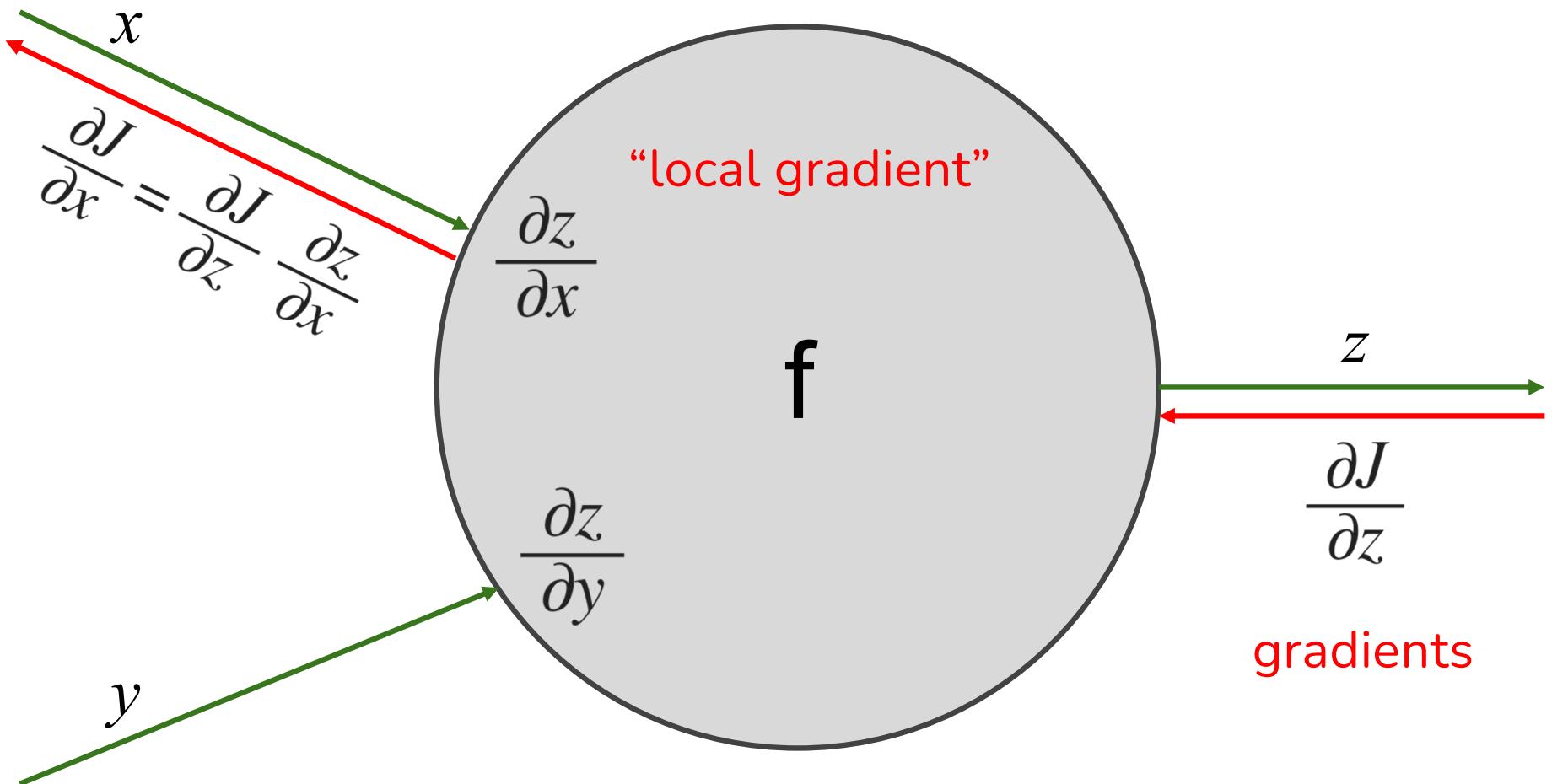


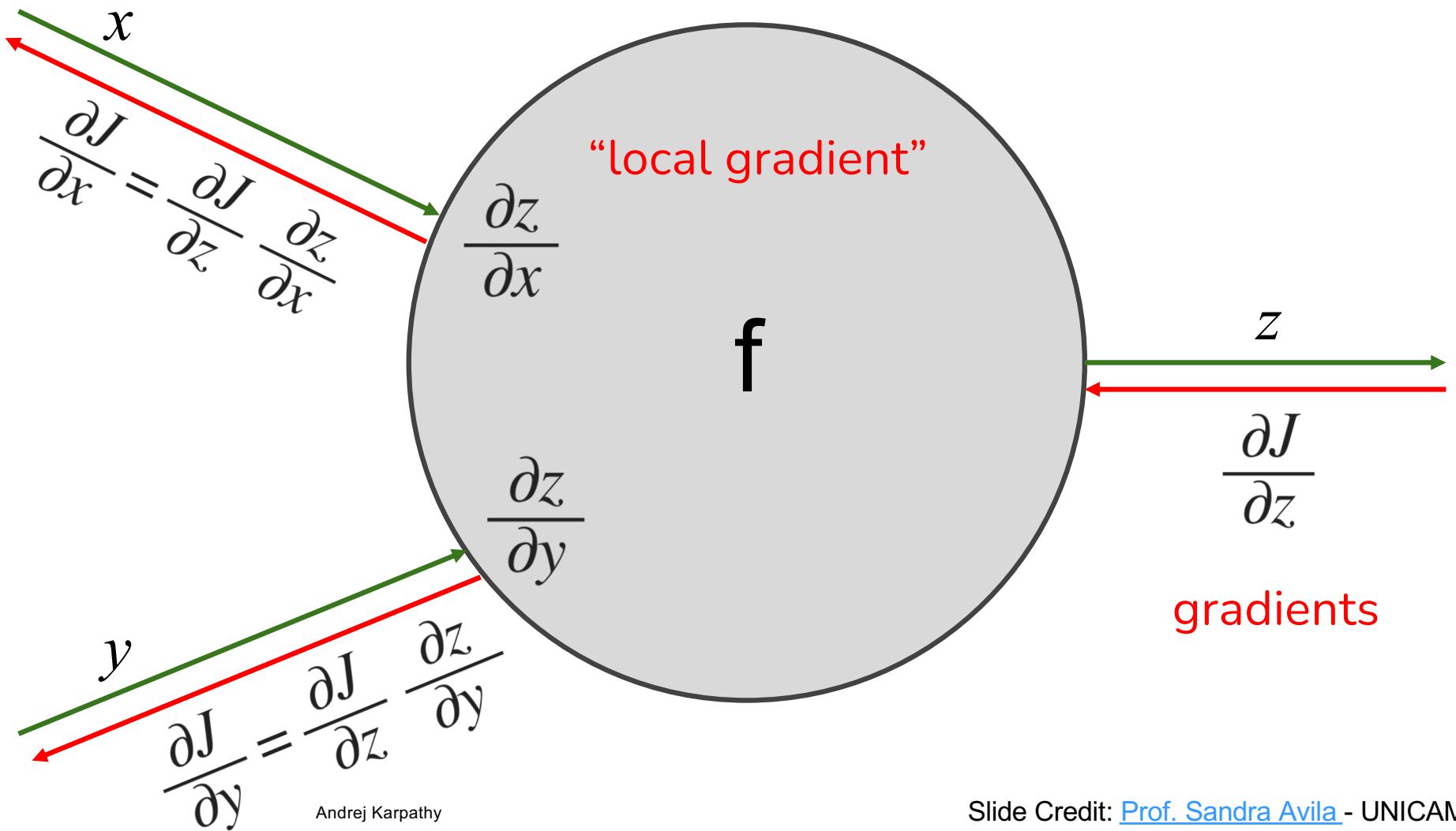
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP











Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial z} \frac{\partial z}{\partial x}$$

$$\frac{\partial z}{\partial x}$$

“local gradient”

f

$$\frac{\partial z}{\partial y}$$

$$\frac{\partial J}{\partial y} = \frac{\partial J}{\partial z} \frac{\partial z}{\partial y}$$

Andrej Karpathy

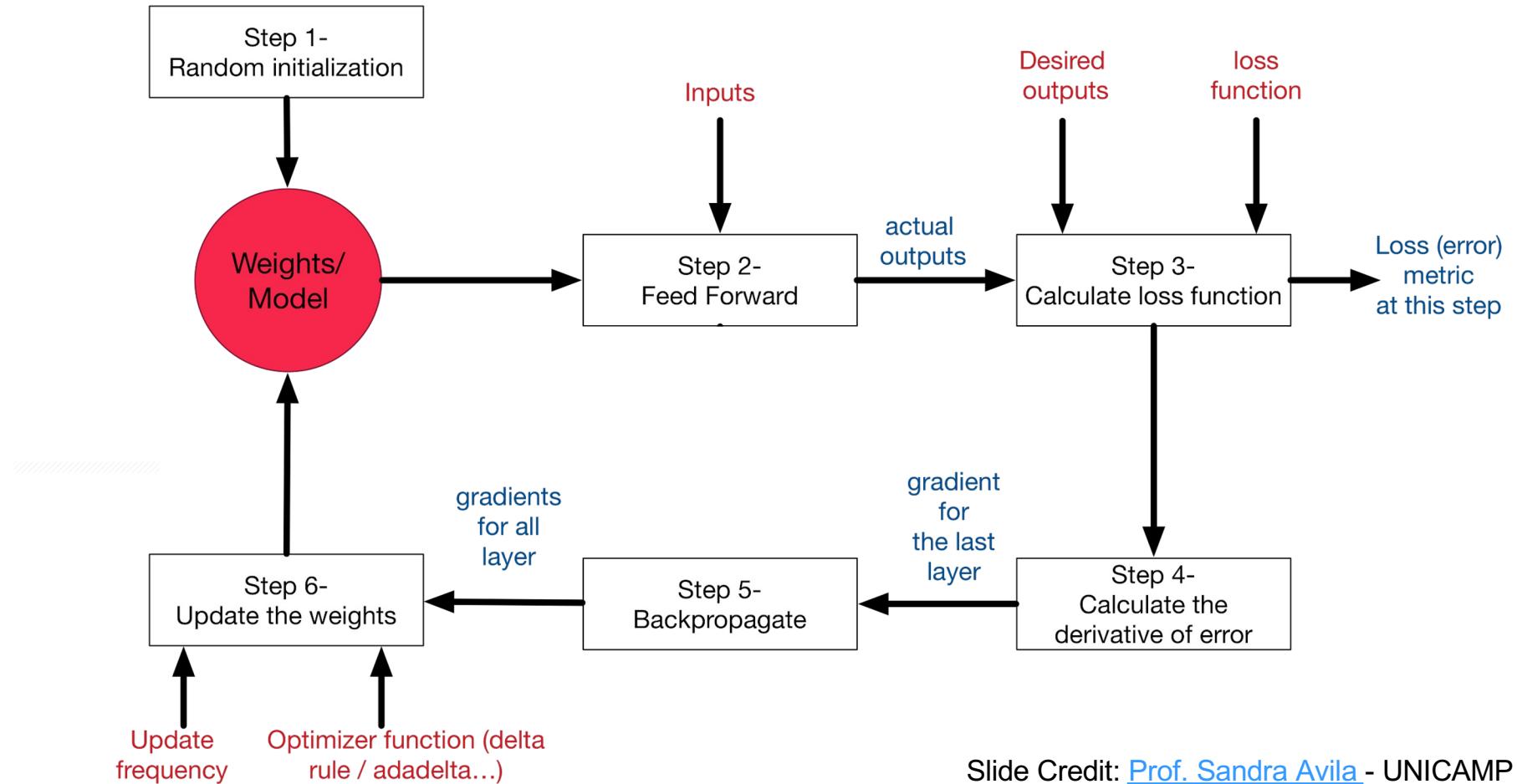
$$z$$

$$\frac{\partial J}{\partial z}$$

gradients

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Training a Neural Network



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Gradient Descent

$$E(\Theta) = -\frac{1}{m} \sum_{i=1}^m L_i$$

Want $\min_{\Theta} E(\Theta)$:

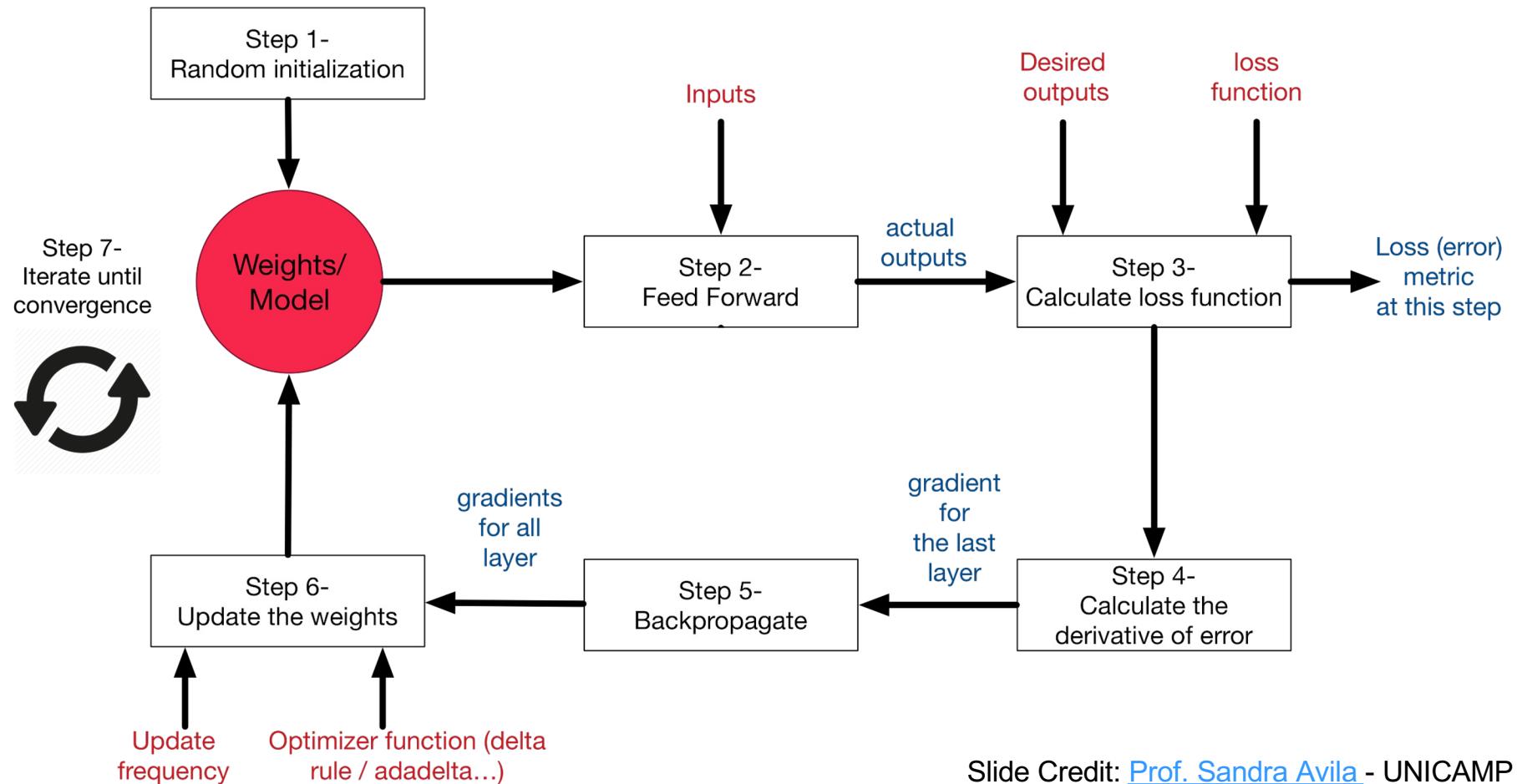
repeat {

$$\Theta_{ij}^{(l)} := \Theta_{ij}^{(l)} - \alpha \frac{\partial}{\partial \Theta_{ij}^{(l)}} E(\Theta)$$

}

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Training a Neural Network



Assignment 6: Data Loaders



Additional Resources

Neural Networks ([3Blue1Brown](#))

YouTube BR

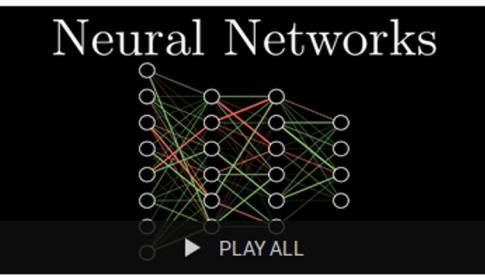
Search

Home Trending Subscriptions

LIBRARY History Watch later Liked videos Chansons Franca... Show more

3Blue1Brown SUBSCRIBED 1.1M

Neural Networks



Neural networks

4 videos • 320,262 views • Last updated on Aug 1, 2018

SEASON 3 ▾

1 Neural Networks 3BLUE1BROWN SERIES S3 • E1
But what *is* a Neural Network? | Deep learning, chapter 1
3Blue1Brown 19:13

2 How machines learn 3BLUE1BROWN SERIES S3 • E2
Gradient descent, how neural networks learn | Deep learning, chapter 2
3Blue1Brown 21:01

3 Backpropagation 3BLUE1BROWN SERIES S3 • E3
What is backpropagation really doing? | Deep learning, chapter 3
3Blue1Brown 13:54

4 Backpropagation calculus 3BLUE1BROWN SERIES S3 • E4
Backpropagation calculus | Deep learning, chapter 4
3Blue1Brown 10:18

Neural Networks Demystified (in Python)

The image shows a screenshot of a YouTube channel page titled "Neural Networks Demystified". The channel has 197,878 views and was last updated on Oct 2, 2015. It features a "PLAY ALL" button and a thumbnail illustrating a neural network architecture with layers labeled "HOURS SLEEP", "HOURS STUDY", "HIDDEN LAYER(S) (MYSTERY)", and "DEEP". Below the video player, there's a "SUBSCRIBE 101K" button. The channel is associated with "Welch Labs".

1 Neural Networks Demystified [Part 1: Data and Architecture] 3:08 Welch Labs

2 Neural Networks Demystified [Part 2: Forward Propagation] 4:28 Welch Labs

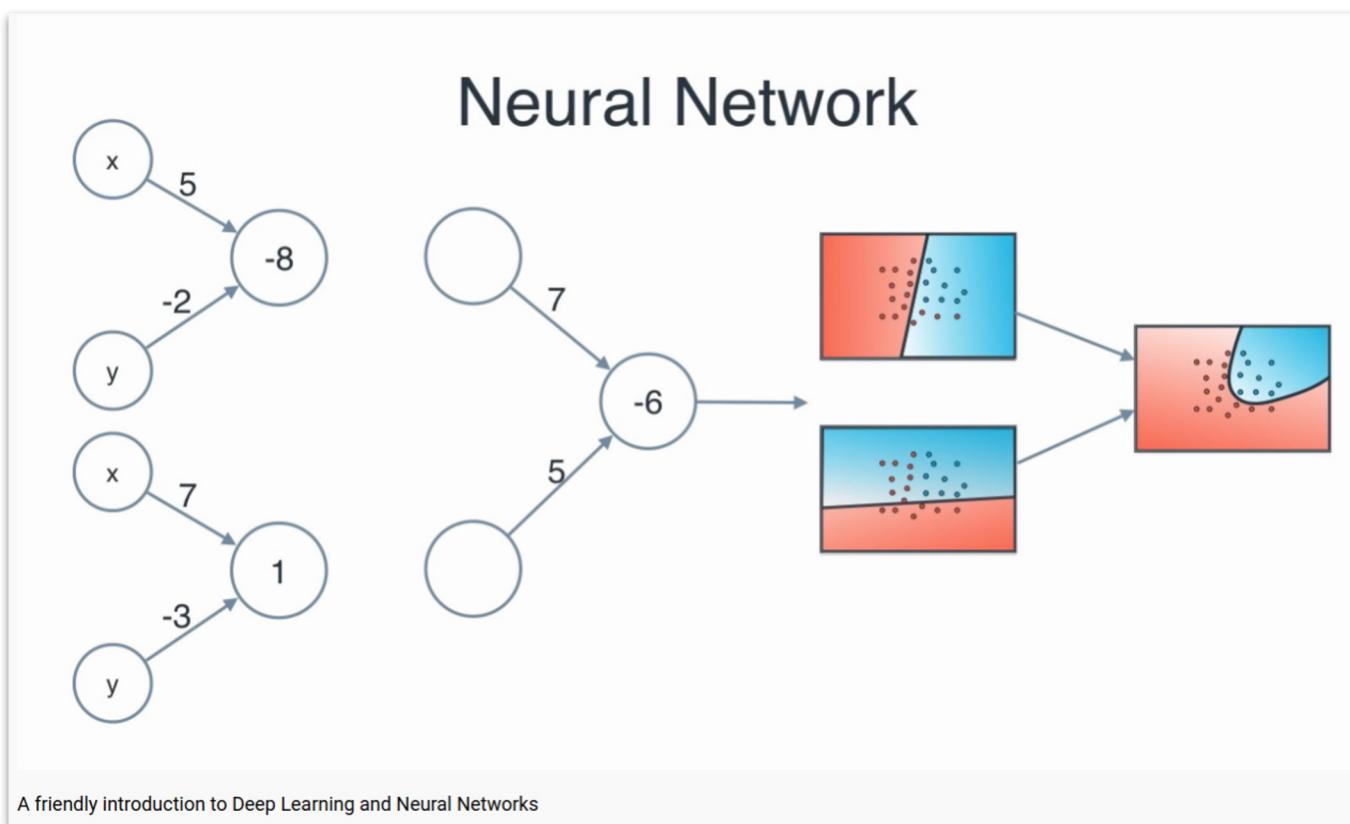
3 Neural Networks Demystified [Part 3: Gradient Descent] 6:56 Welch Labs

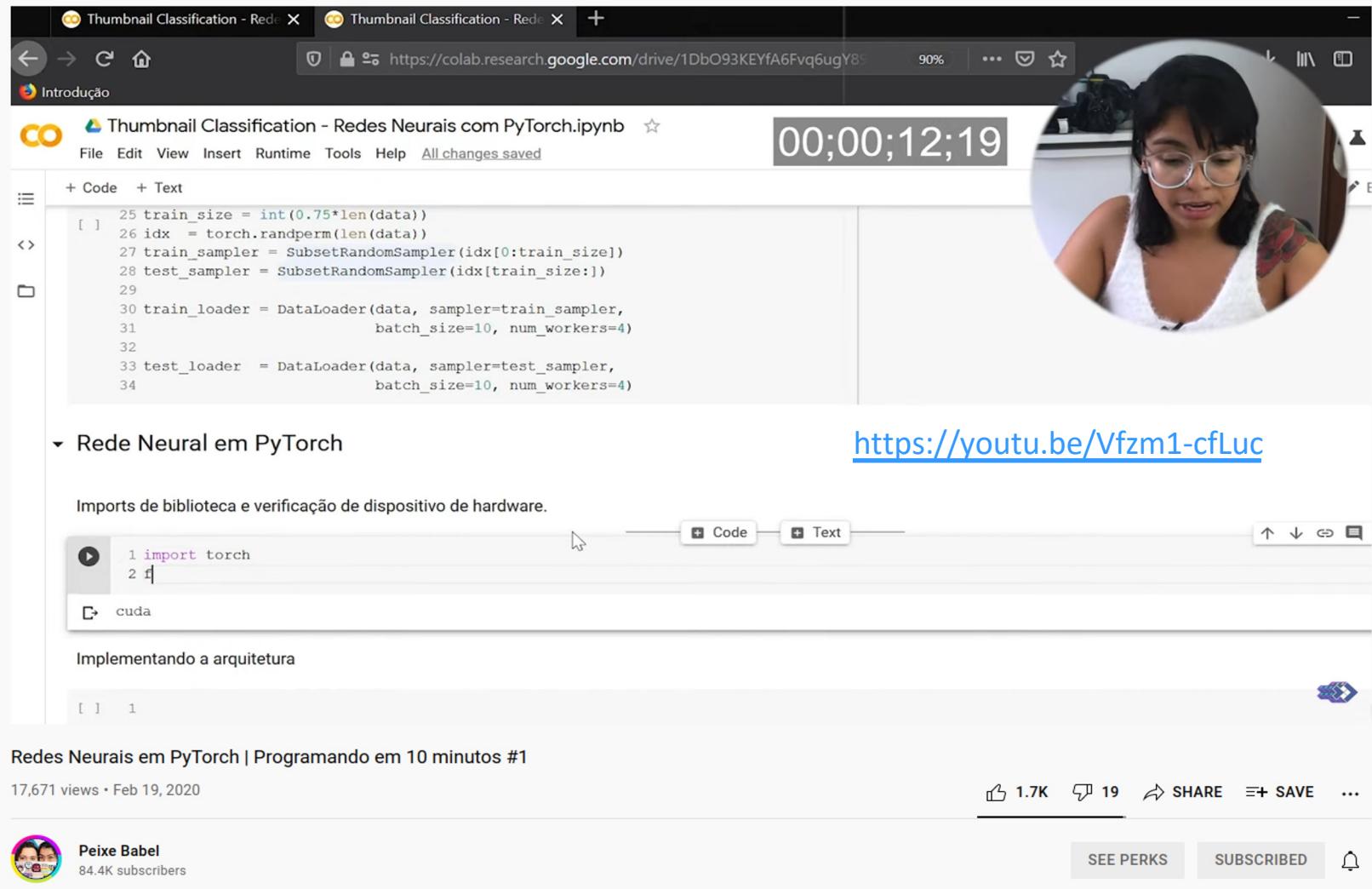
4 Neural Networks Demystified [Part 4: Backpropagation] 7:56 Welch Labs

5 Neural Networks Demystified [Part 5: Numerical Gradient Checking] 4:14 Welch Labs

“A friendly introduction to Neural Networks”

<https://youtu.be/BR9h47Jtqyw>





Thumbnail Classification - Redes Neurais com PyTorch.ipynb

File Edit View Insert Runtime Tools Help All changes saved

00:00:12:19

Rede Neural em PyTorch

<https://youtu.be/Vfzm1-cfLuc>

Imports de biblioteca e verificação de dispositivo de hardware.

```
1 import torch
2 if
```

Implementando a arquitetura

```
[ ] 1
```

Redes Neurais em PyTorch | Programando em 10 minutos #1

17,671 views • Feb 19, 2020

Peixe Babel 84.4K subscribers

SEE PERKS SUBSCRIBED

Summary

- We use deep neural networks because of their strong performance in practice
- Training deep neural nets
 - We need an **objective function** that measures and guides us towards good performance
 - We need a way **to minimize the loss function**: stochastic gradient descent
 - We need **backpropagation** to propagate error from end of net towards all layers and change weights at those layers
- Practices for preventing overfitting
 - Regularization