

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



What AI courses did you take before this semester?

 Submission is closed 

Open 

0

cs1571: Intro to AI

0

cs1675: Intro to
Machine Learning

0

cs1678: Intro to
Deep Learning

0

cs2731: Intro to
Natural Language
Processing

0

cs1503: Math
Foundations of
Machine Learning

0

cs 1671: Human
Language
Technologies

 Presentation selected for PowerPoint



1



 0  0 / 100 

Many inventions were inspired by Nature ...

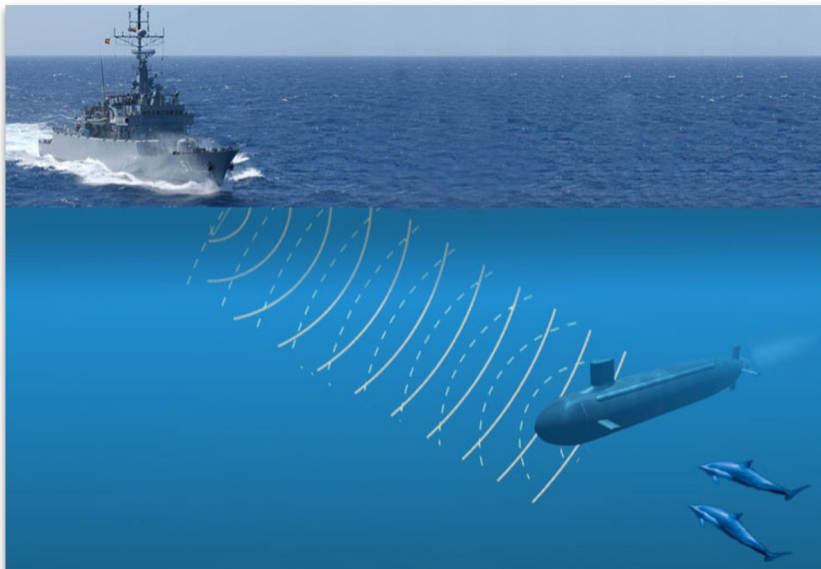
Birds inspired us to fly



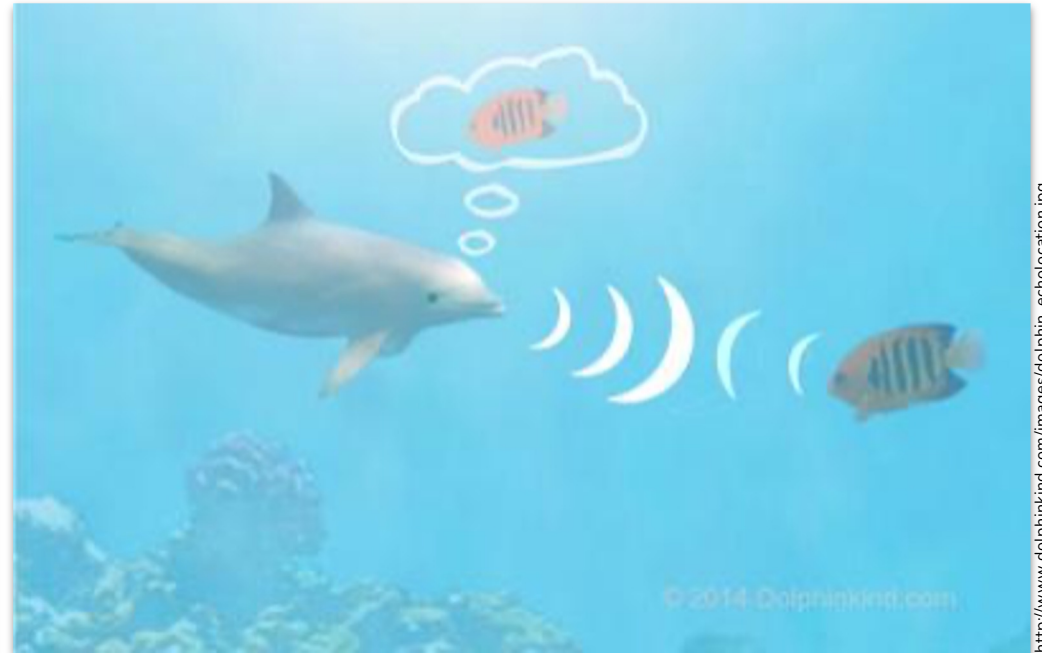
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Many inventions were inspired by Nature ...

Dolphins inspired sonar development



https://sites.google.com/site/echolocationawproject/_/rsrc/1459209762464/sonars/image.jpeg



http://www.dolphinkind.com/images/dolphin_echolocation.jpg

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

Many inventions were inspired by Nature ...

Humpback whales inspired wind turbines



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



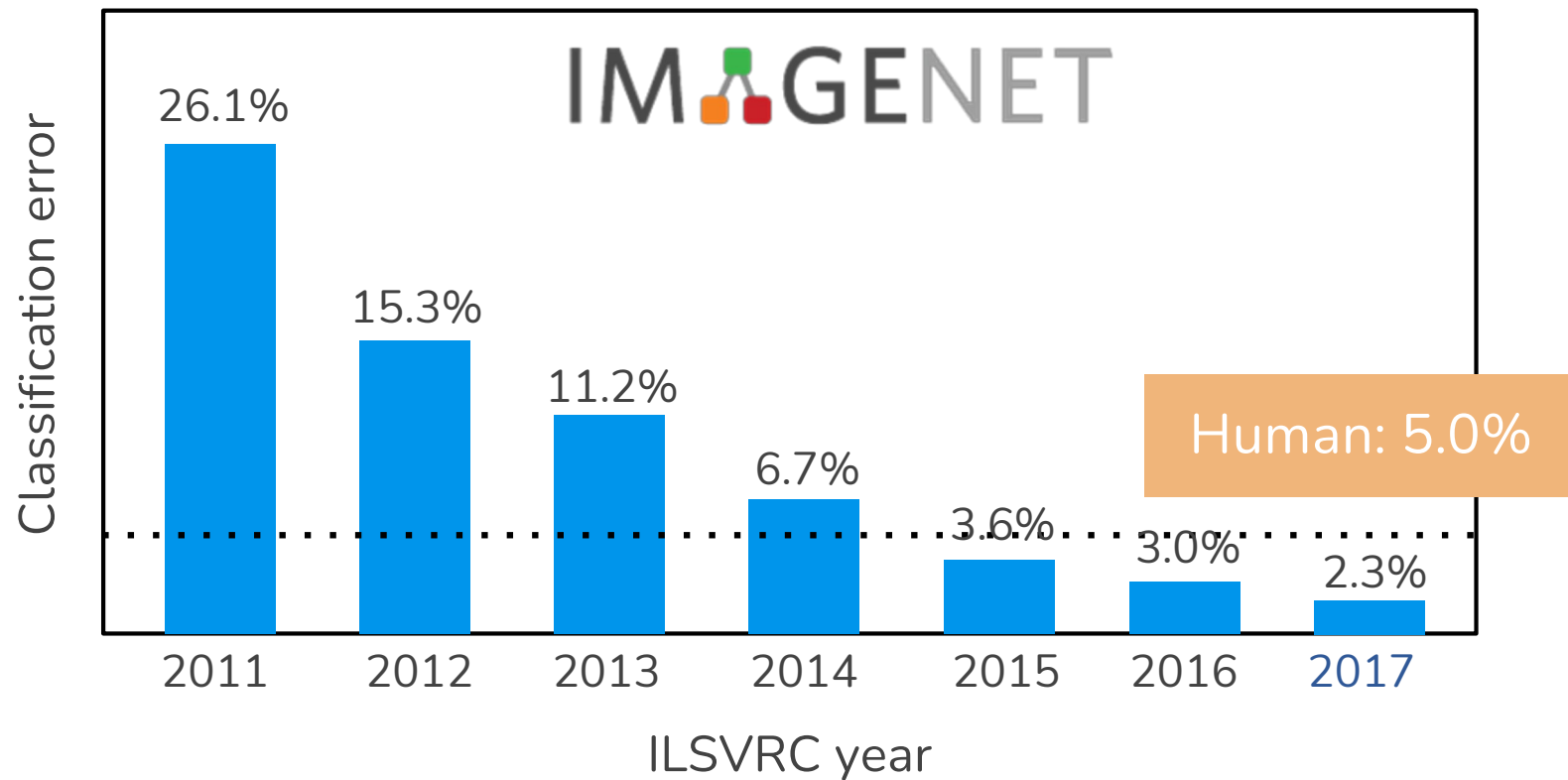
ILSVRC 2012 — Image Classification task

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

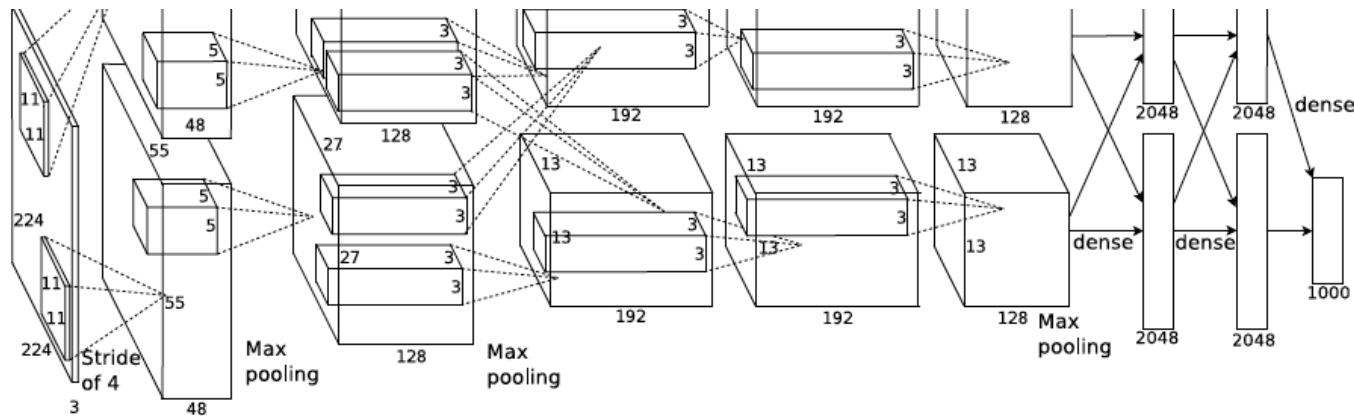


“ImageNet classification with deep convolutional neural networks”. NIPS, 2012.

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.



IMGENET

www.image-net.org

22K categories and **14M** images

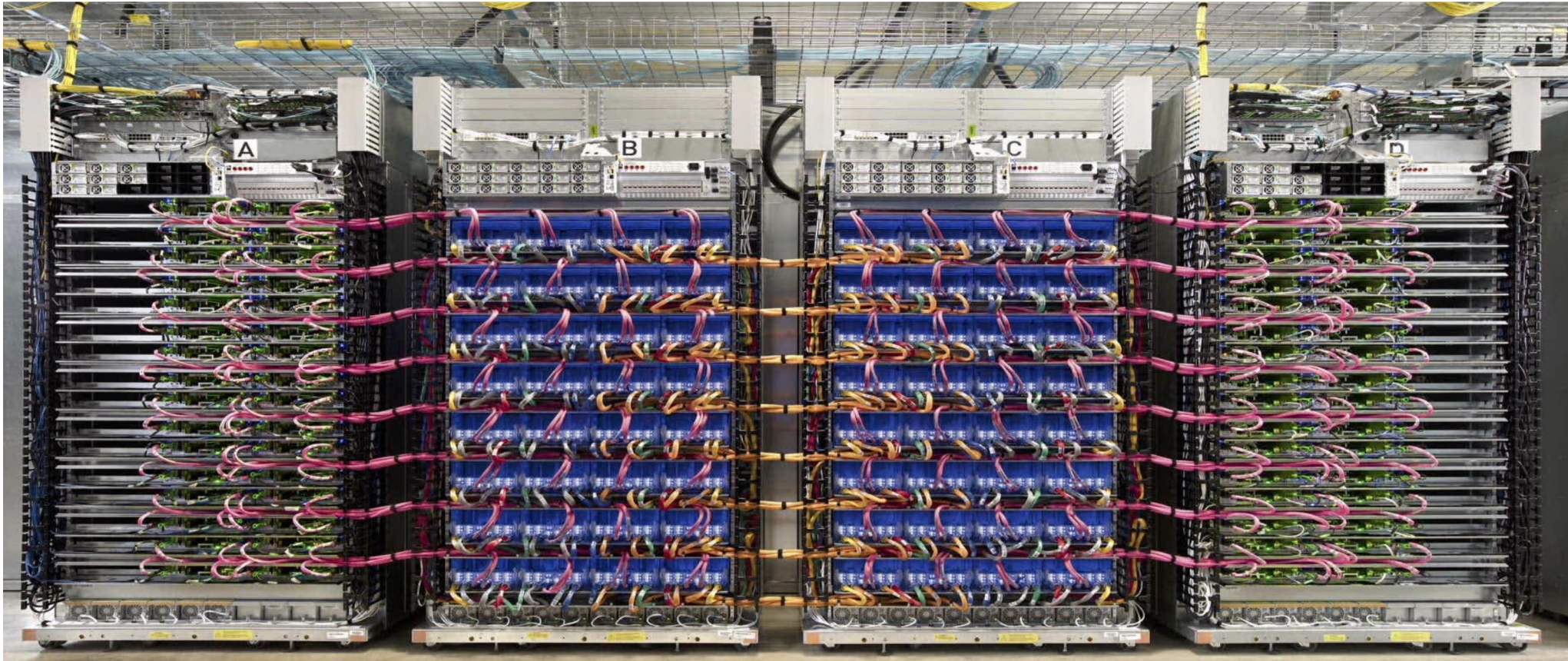
- Animals
 - Bird
 - Fish
 - Mammal
 - Invertebrate
- Plants
 - Tree
 - Flower
 - Food
 - Materials
- Structures
 - Artifact
 - Tools
 - Appliances
 - Structures
- Person
 - Scenes
 - Indoor
 - Geological Formations
 - Sport Activities



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.



Lab 3: Pytorch

Duration: 20 min



To join, go to: ahaslides.com/ZKTDM 



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

Get Feedback

✓ Slide 1 selected for PowerPoint [Switch slide](#)



 0  0 / 100 

Assignment 3: 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**.

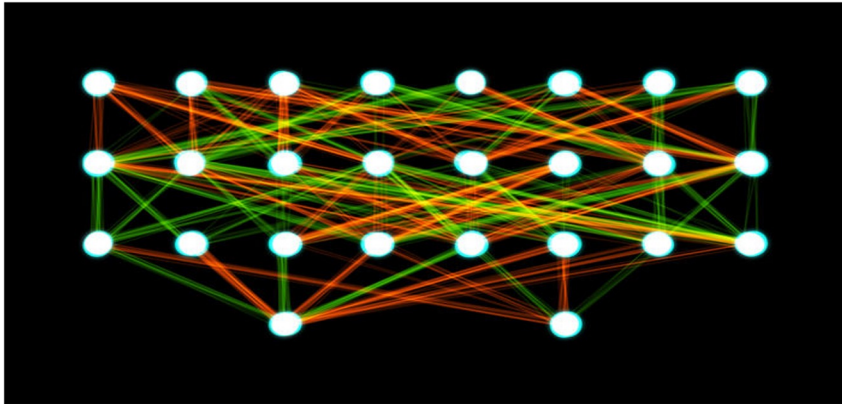
Science
AAAS

Authors | Members | Librarians | Advertisers

Home News Journals Topics Careers

Latest News ScienceInsider ScienceShots Sifter From the Magazine About News Quizzes

SHARE

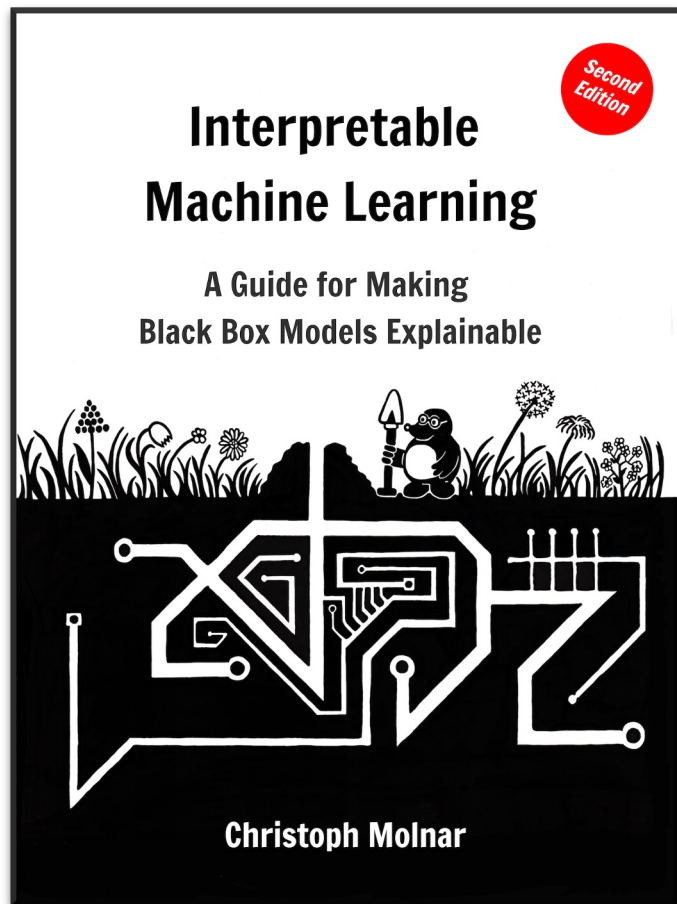


A representation of a neural network. Akritasa/Wikimedia Commons

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



“Interpretable Machine Learning”, 2022

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

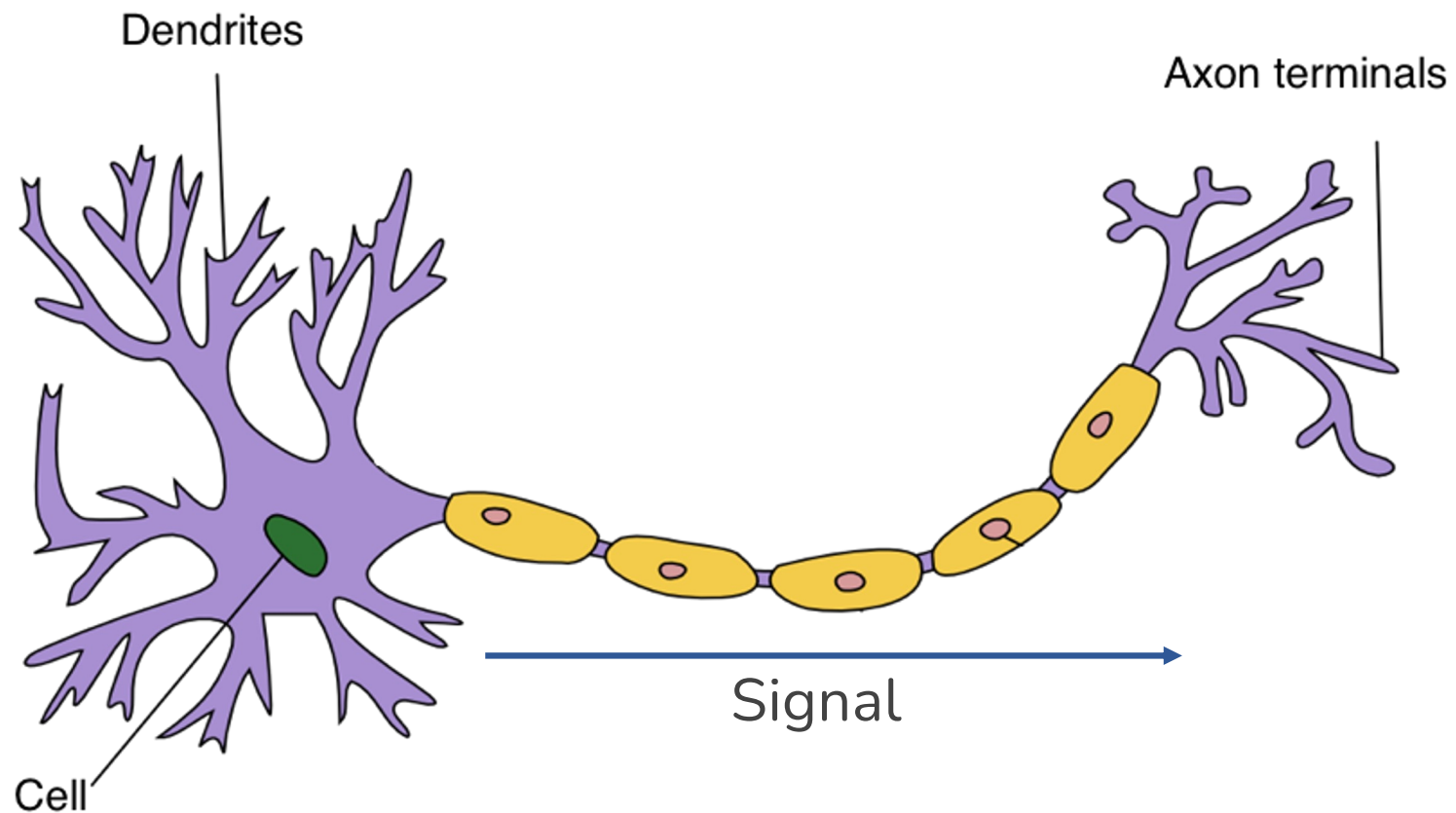
Chap. 10 Neural Network Interpretation

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

From Biological to Artificial Neurons

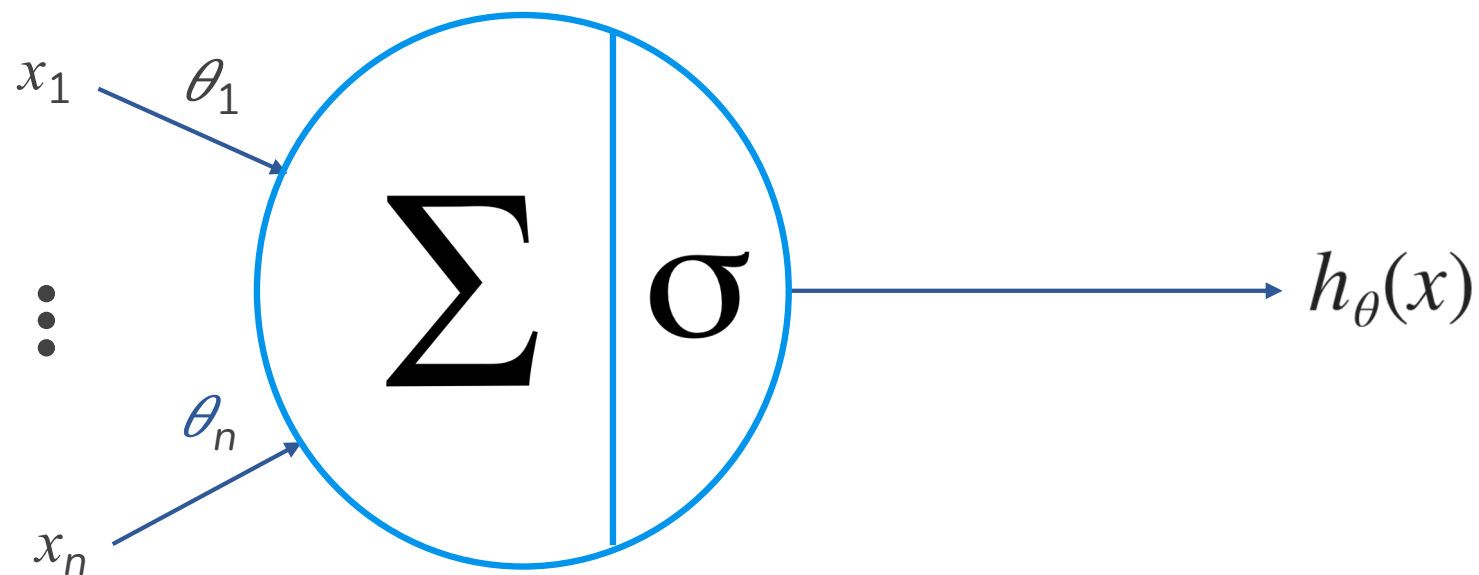
1. There is now a **huge quantity of data** available to train neural networks.
2. **Computing power** now makes it possible to train large neural networks in a reasonable amount of time.
3. The **training algorithms** have been **improved**.
4. ANNs seem to have entered a virtuous circle of funding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

Biological Neurons



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

Biological Neurons - Analogy

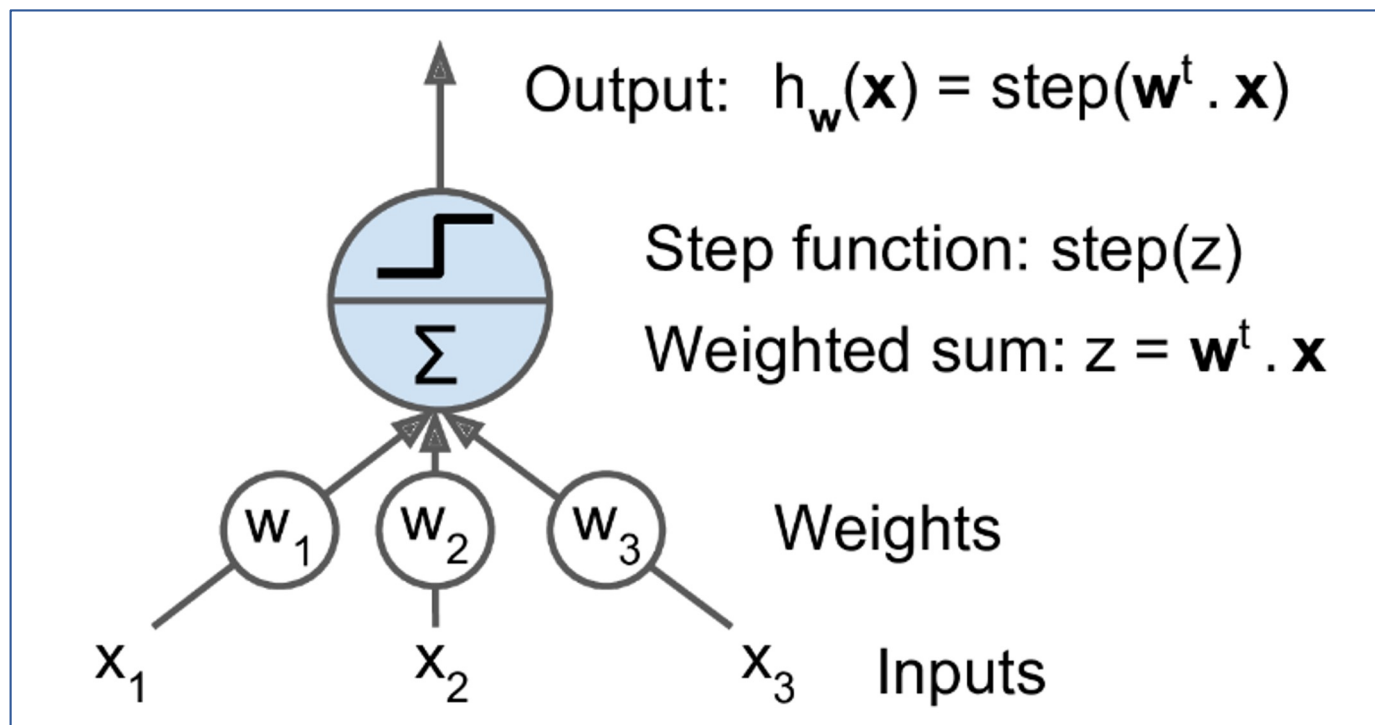


The Perceptron

Invented in 1957 by Frank Rosenblatt.

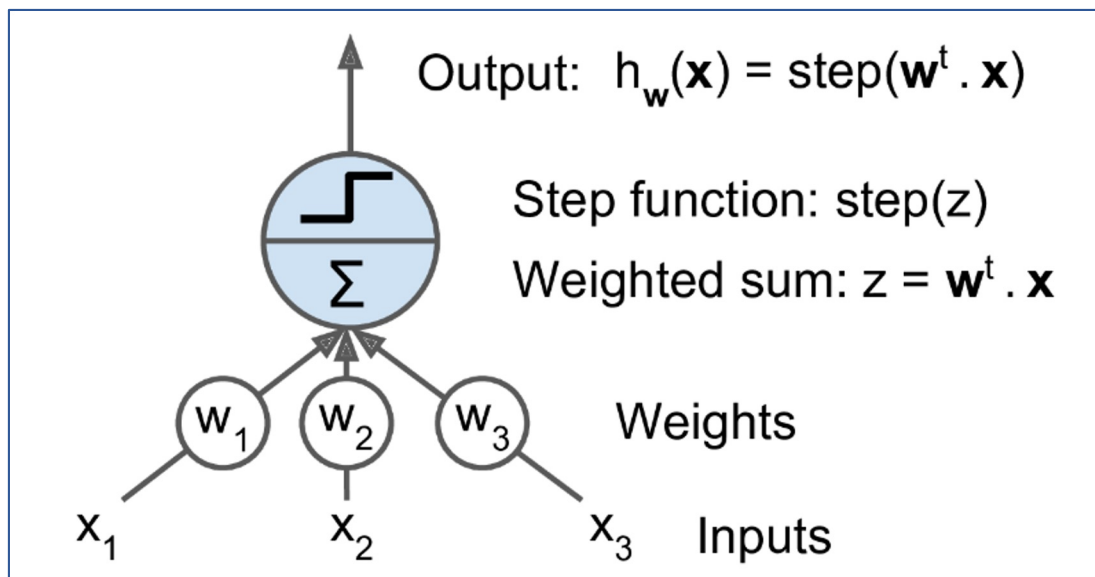
- It is based on a Linear Threshold Unit (LTU):
 - The inputs and output are now **numbers** and each input connection is associated with a **weight**.
- The LTU computes a weighted sum of its inputs then it applies a step function to that sum and outputs the result.

The Perceptron



Linear Threshold Unit

The Perceptron

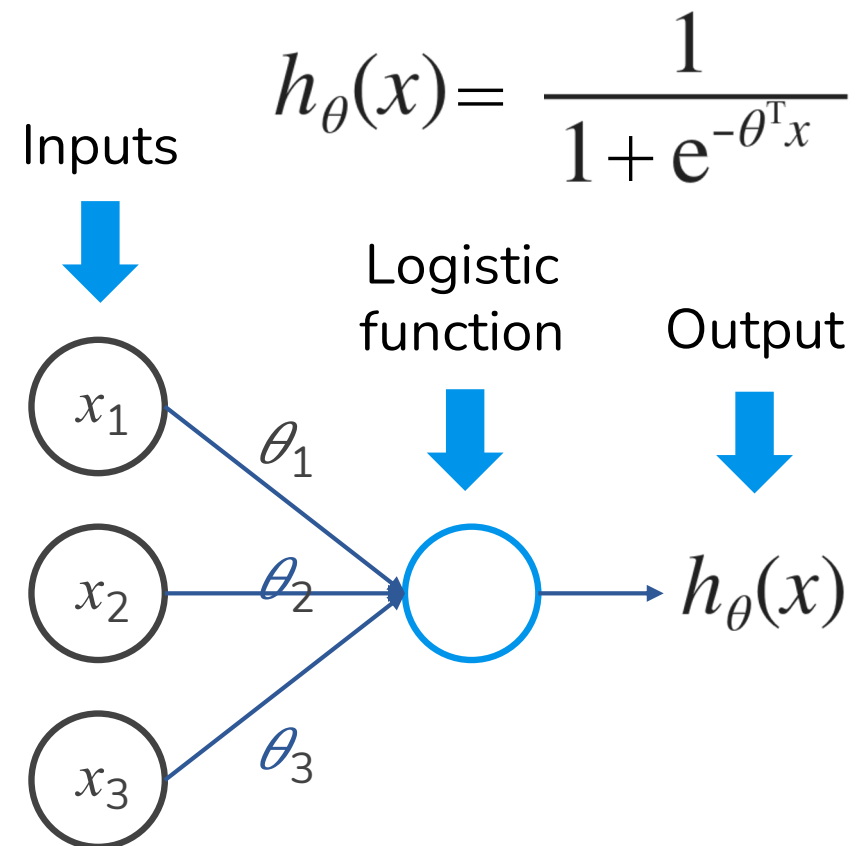
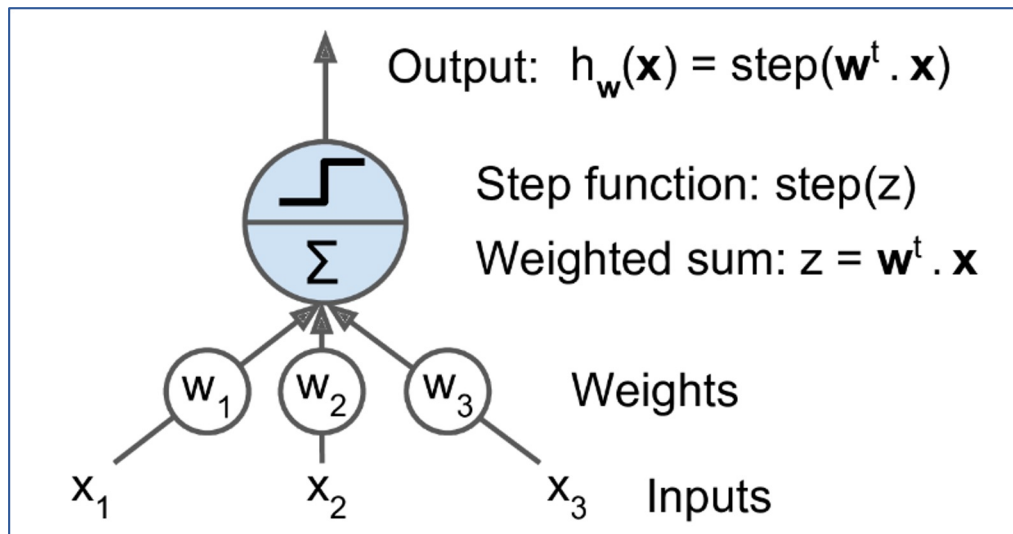


Linear Threshold Unit

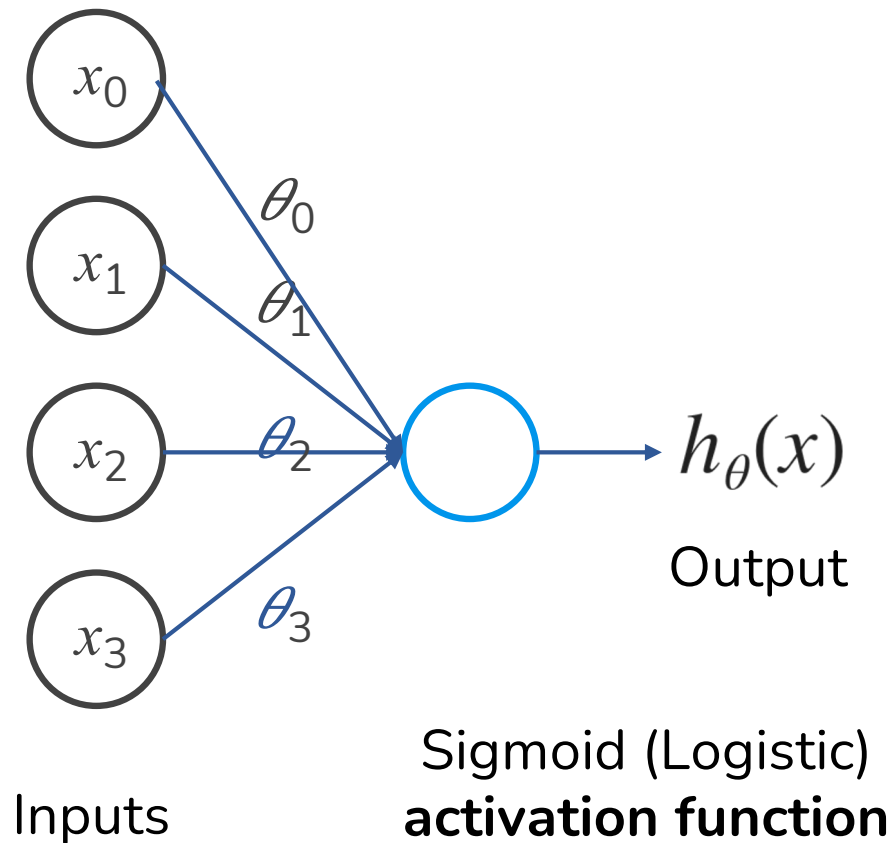
$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

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

Neuron Model: Logistic Unit



Neuron Model: Logistic Unit



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

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

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(X¹)

sin(X²)

+ - 1 HIDDEN LAYER

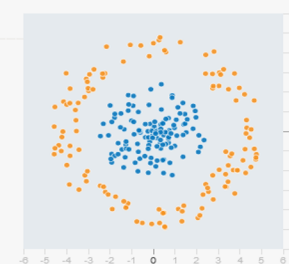
+ -

1 neuron

This is the output from one neuron. Hover to see it larger.

OUTPUT

Test loss 0.504
Training loss 0.502

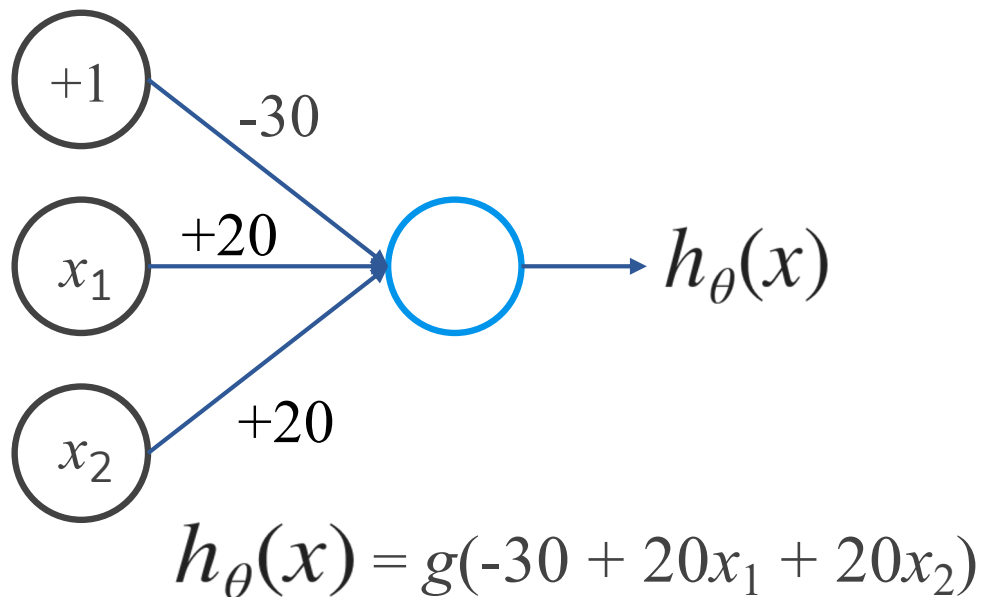
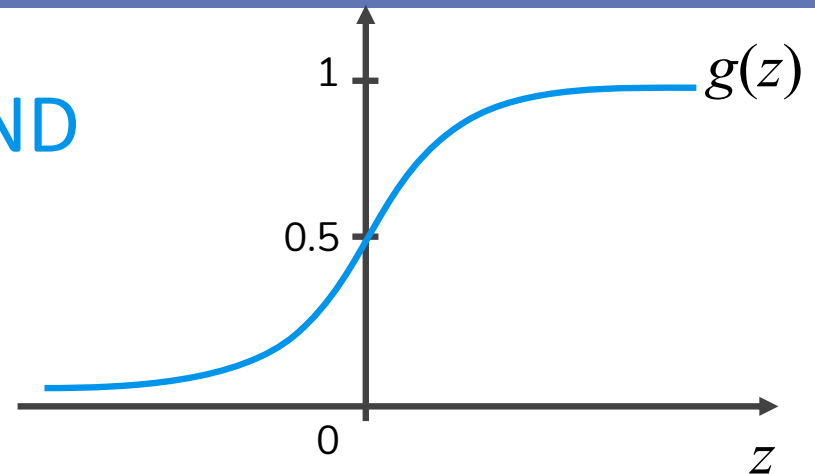


Colors shows data, neuron and weight values.

☐ Show test data ☐ Discretize output

[Example] Simple Example: AND

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

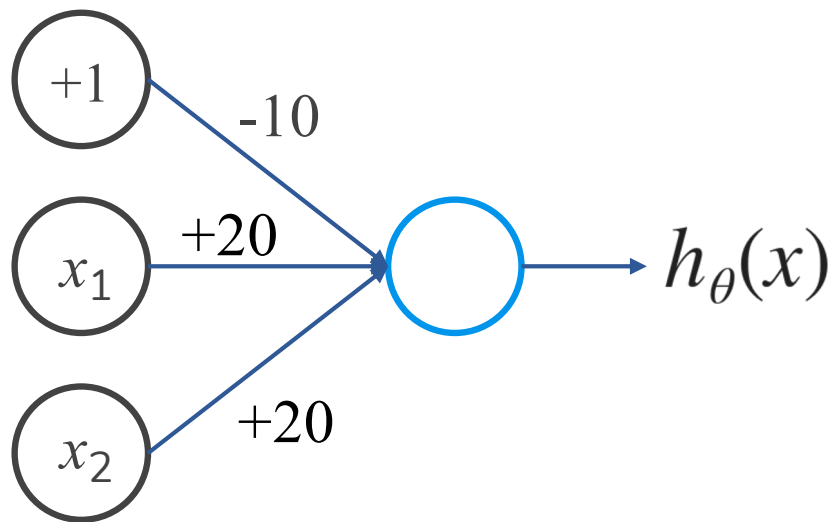


x_1	x_2	$h_{\theta}(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

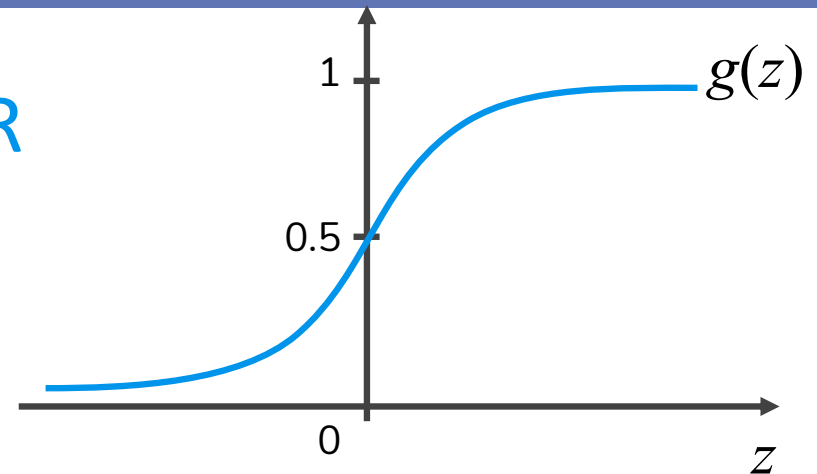
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

[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	$g(30) \approx 1$

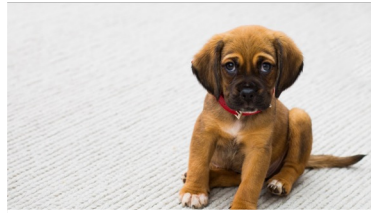
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Multi-class Classification?

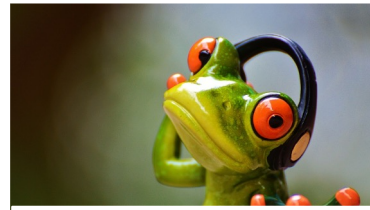
Multi-class Classification



Cat



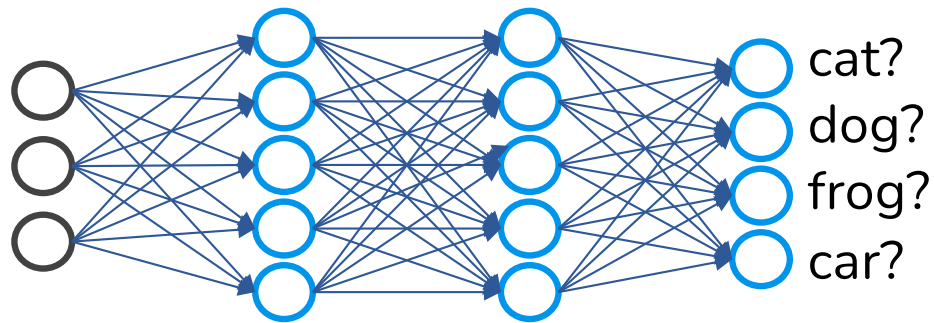
Dog



Frog



Car





Cat



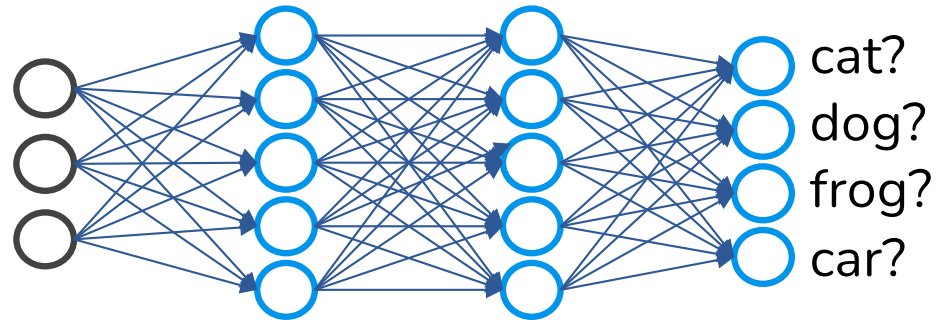
Dog



Frog



Car



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

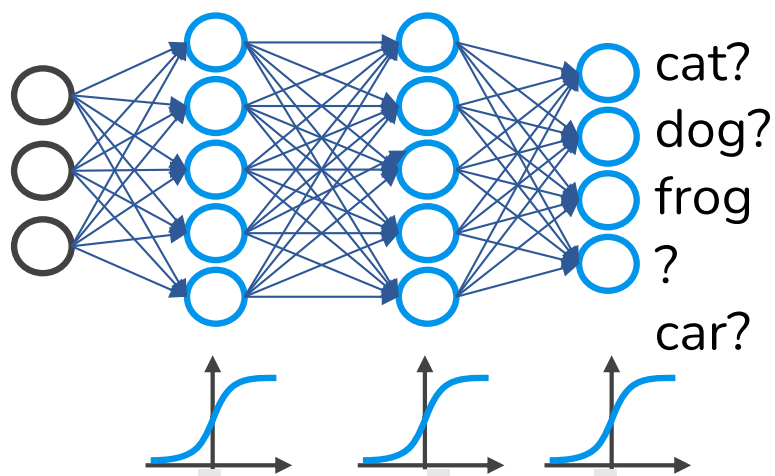
$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$,
when dog

$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$,
when frog

$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$
when car

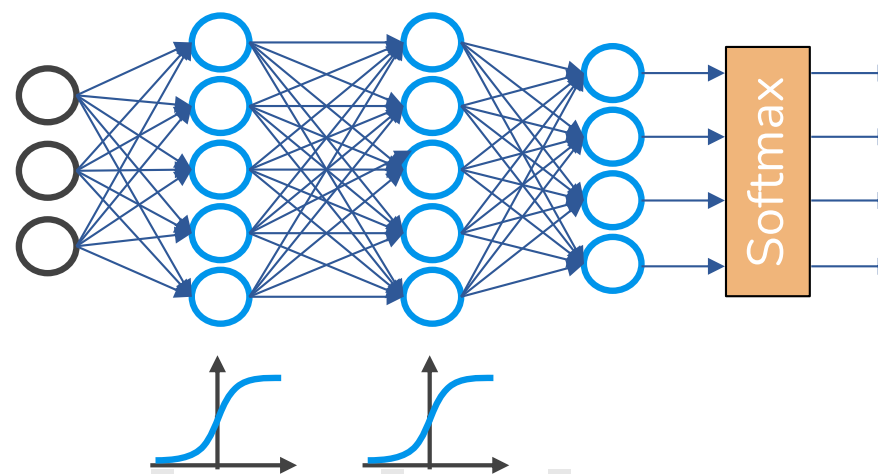
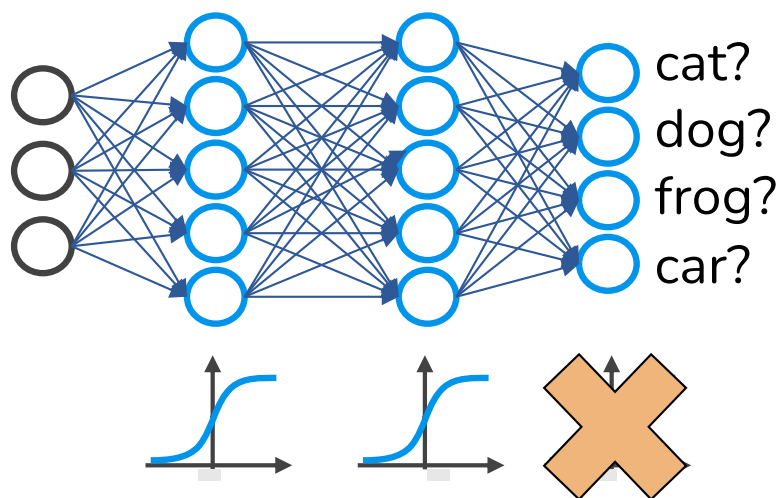
Softmax Classification

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



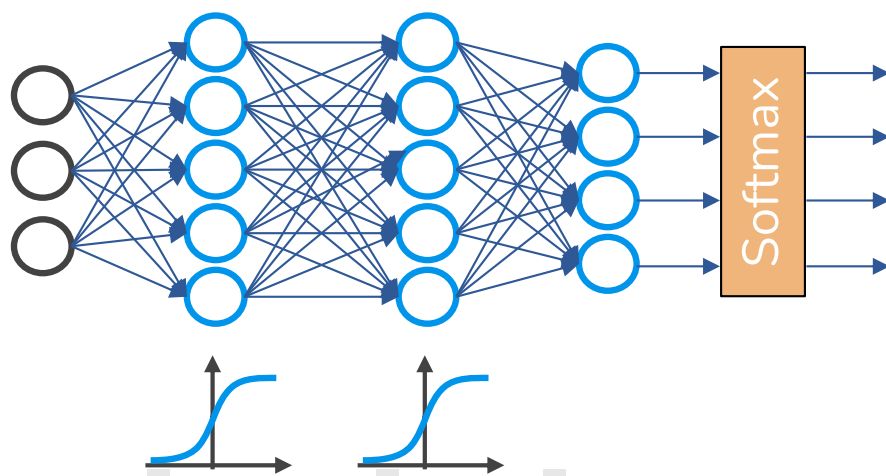
Softmax Classification

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



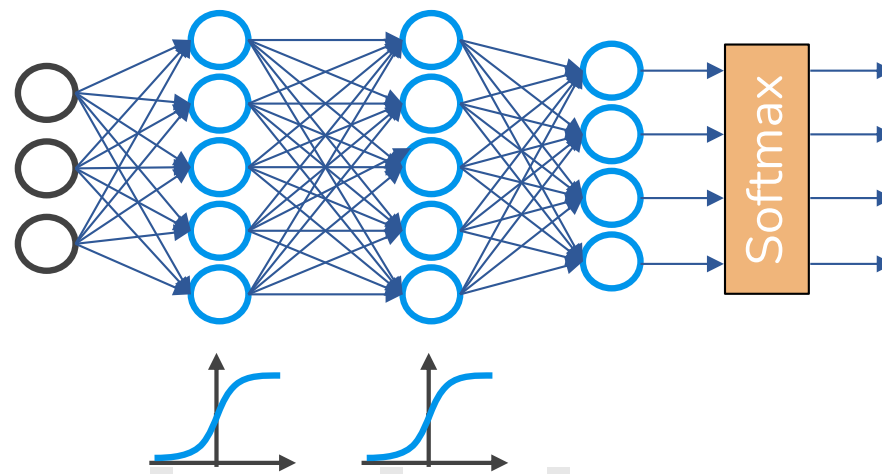
Softmax Classification

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



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

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

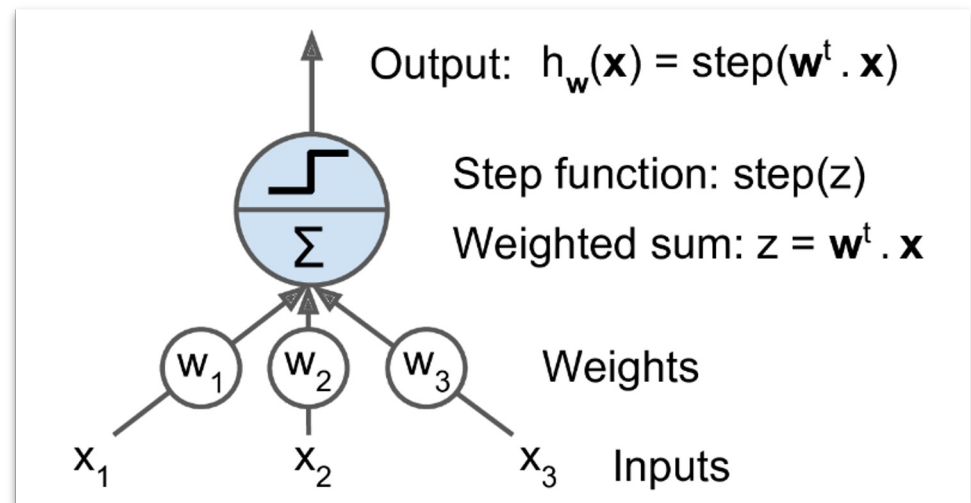
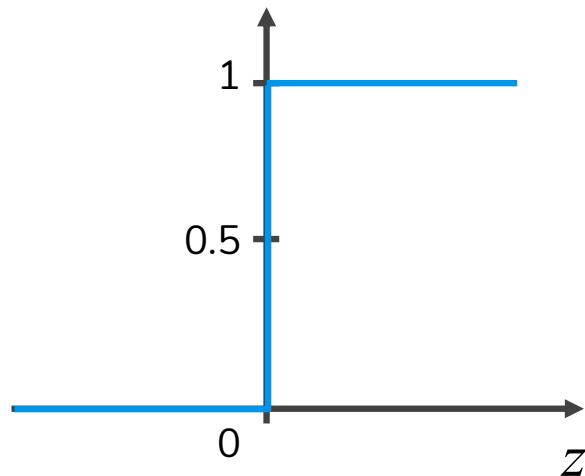


We decided to add “activation functions”
for this purpose.

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

Step Function

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



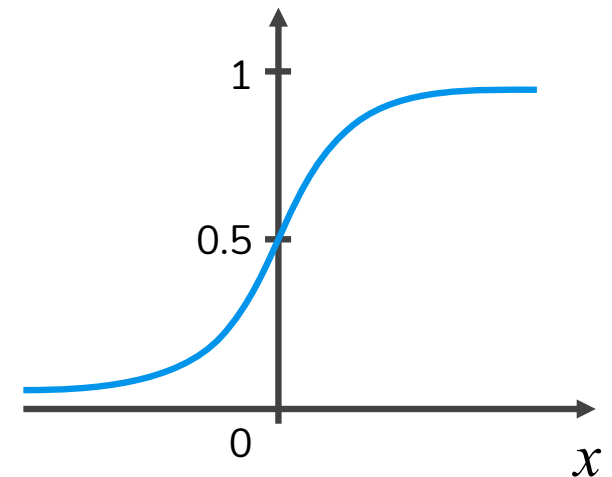
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

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

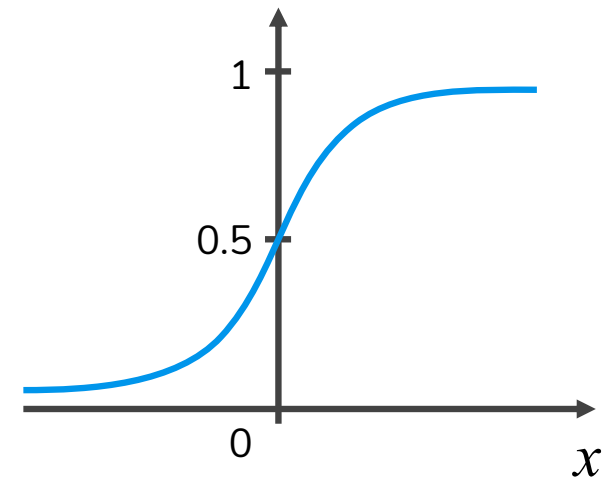
- The output of the activation function is always going to be in range **(0,1)**.
- It is nonlinear in nature.



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

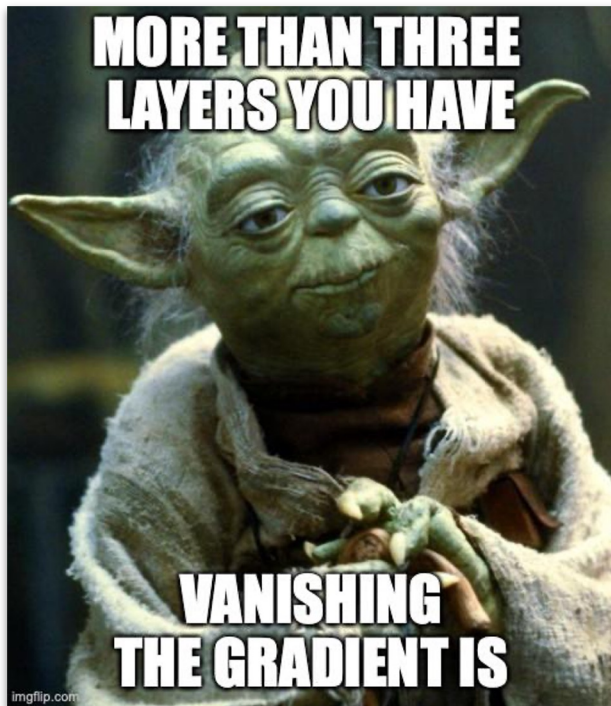
Sigmoid Function

- 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!!



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

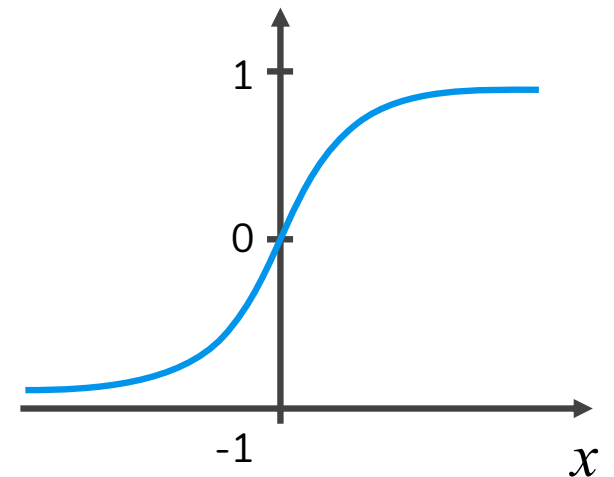
Sigmoid Function: Problem?



- 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.

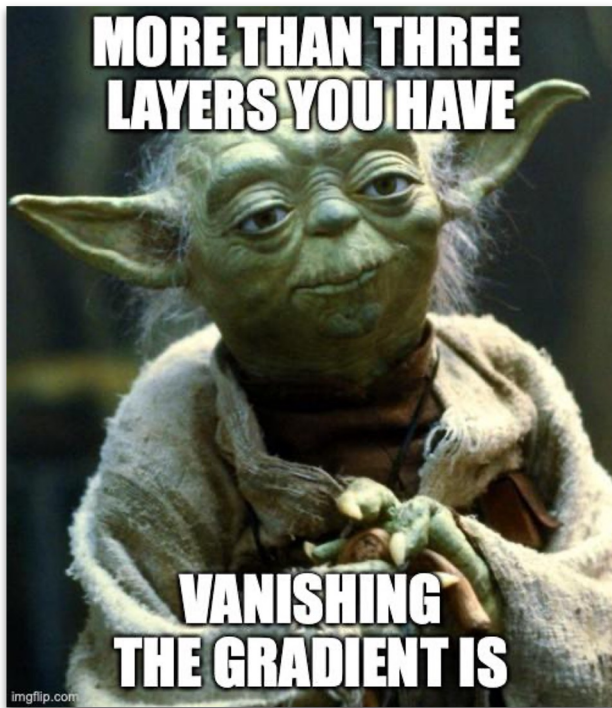
Tanh Function

- 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!!



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

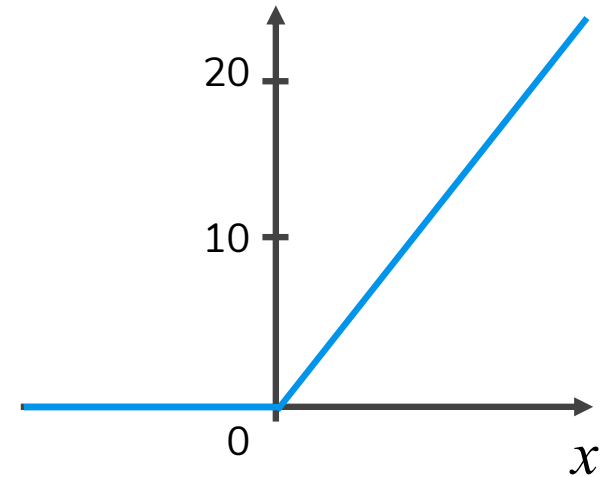
Tanh Function: Problem?



- 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, inf)**.
- It is nonlinear in nature.
Combinations of this function are also nonlinear!
- Sparsity of the activation!



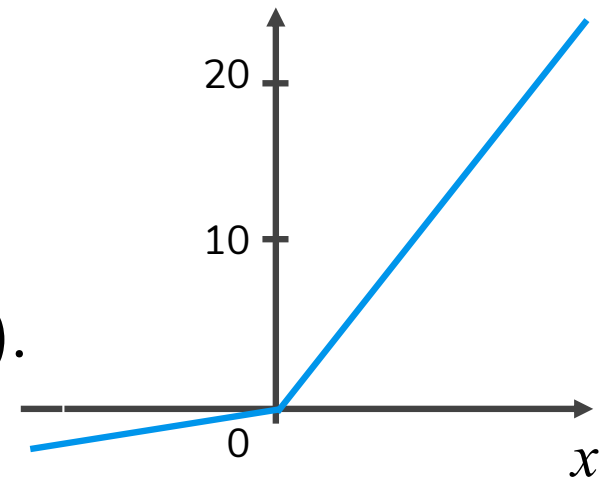
$$\text{ReLU}(x) = \max(0, x)$$

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 **(-inf, inf)**.
- (Leaky) ReLU is less computationally expensive than *tanh* and *sigmoid* because it involves simpler mathematical operations.

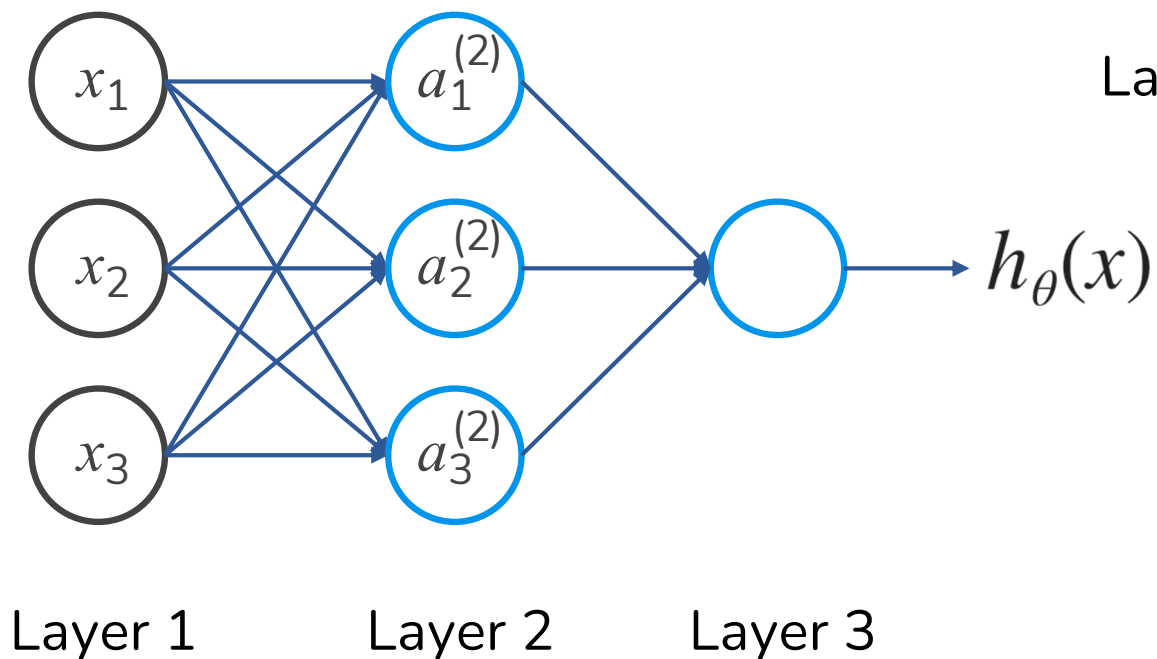


$$\begin{aligned}\text{Leaky ReLU}(x) &= \\ &= \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}\end{aligned}$$

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

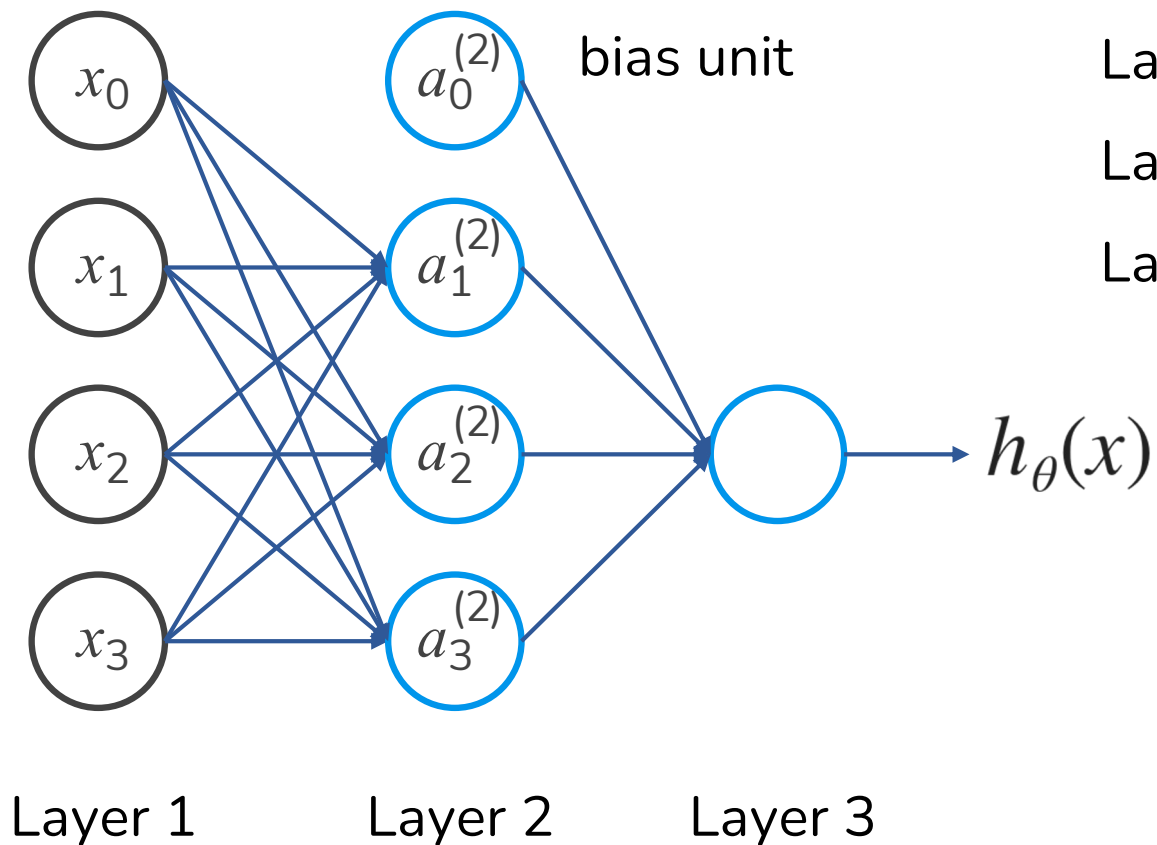


Layer 1 = Input layer

Layer 2 = Hidden layer

Layer 3 = Output layer

Neural Network

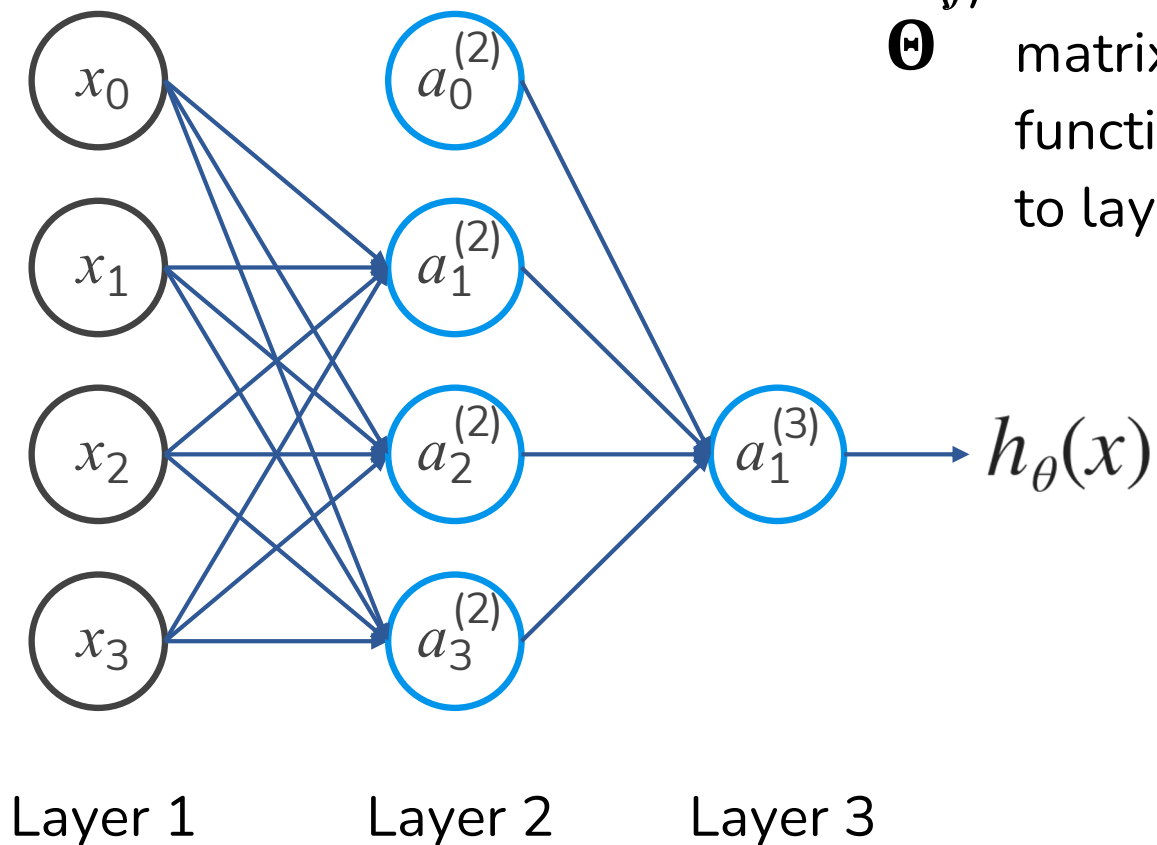


Layer 1 = Input layer

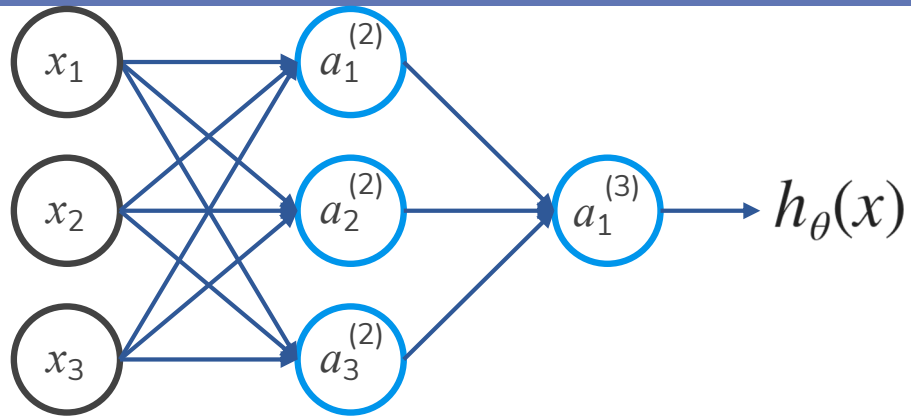
Layer 2 = Hidden layer

Layer 3 = Output layer

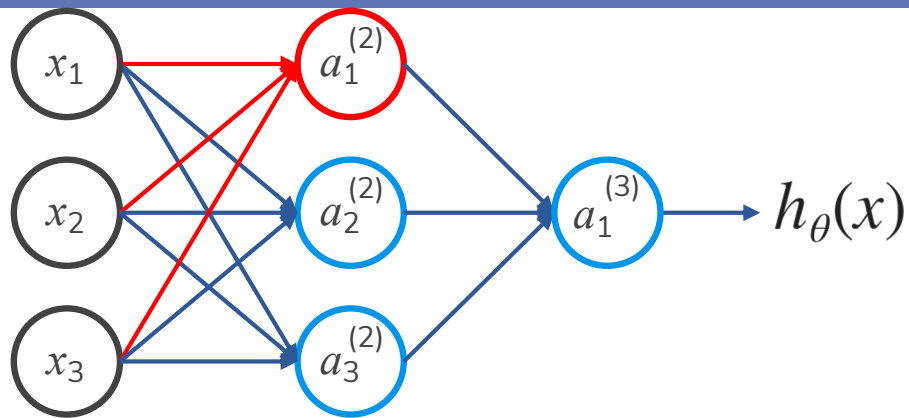
Neural Network



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

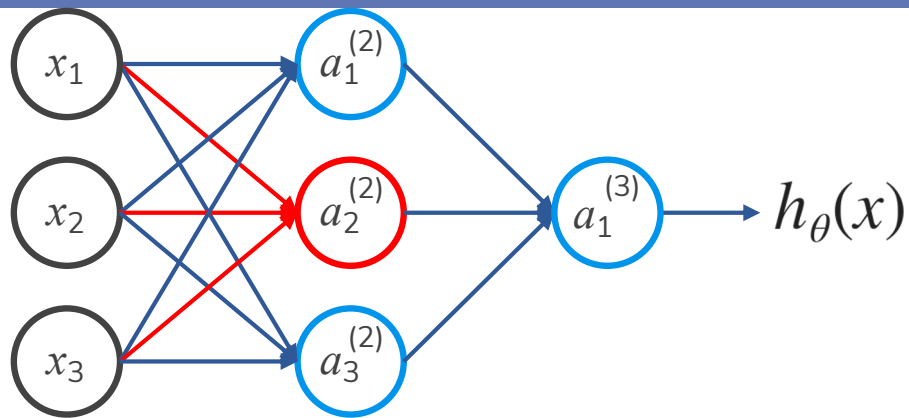


$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling function mapping from layer j to layer $j + 1$



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

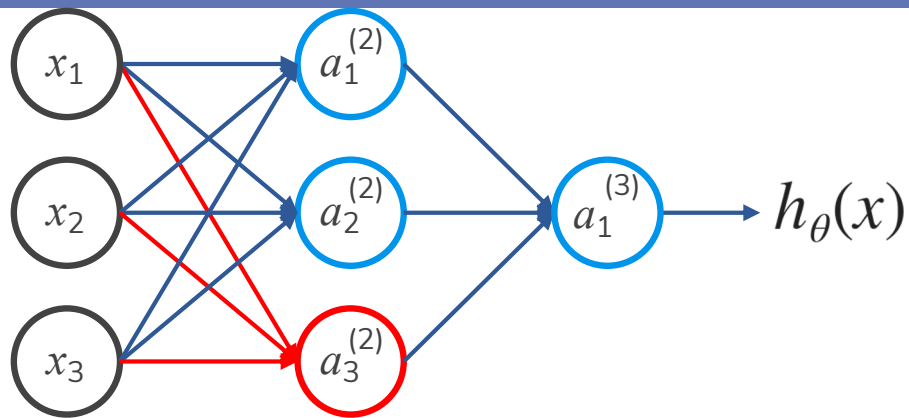
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$



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

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

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

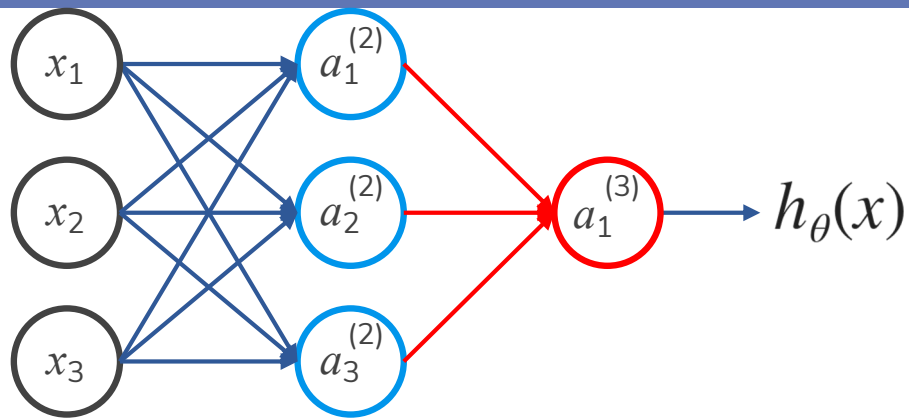


$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling function mapping from layer j to layer $j + 1$

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

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

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



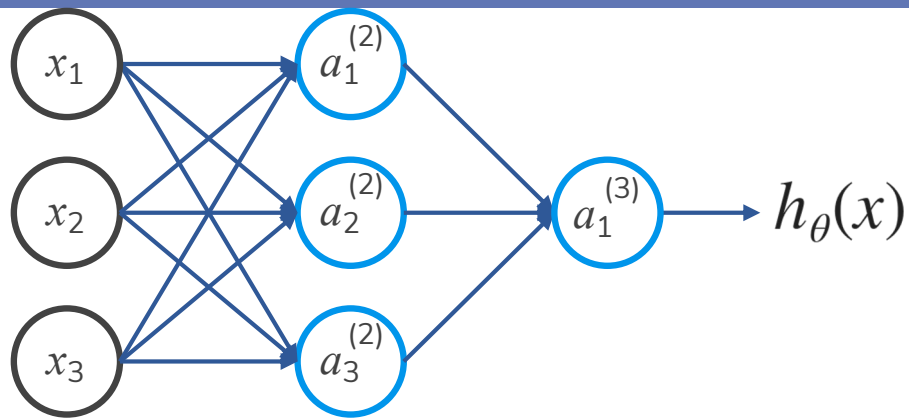
$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling
 function mapping from layer j
 to layer $j + 1$

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

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

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

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

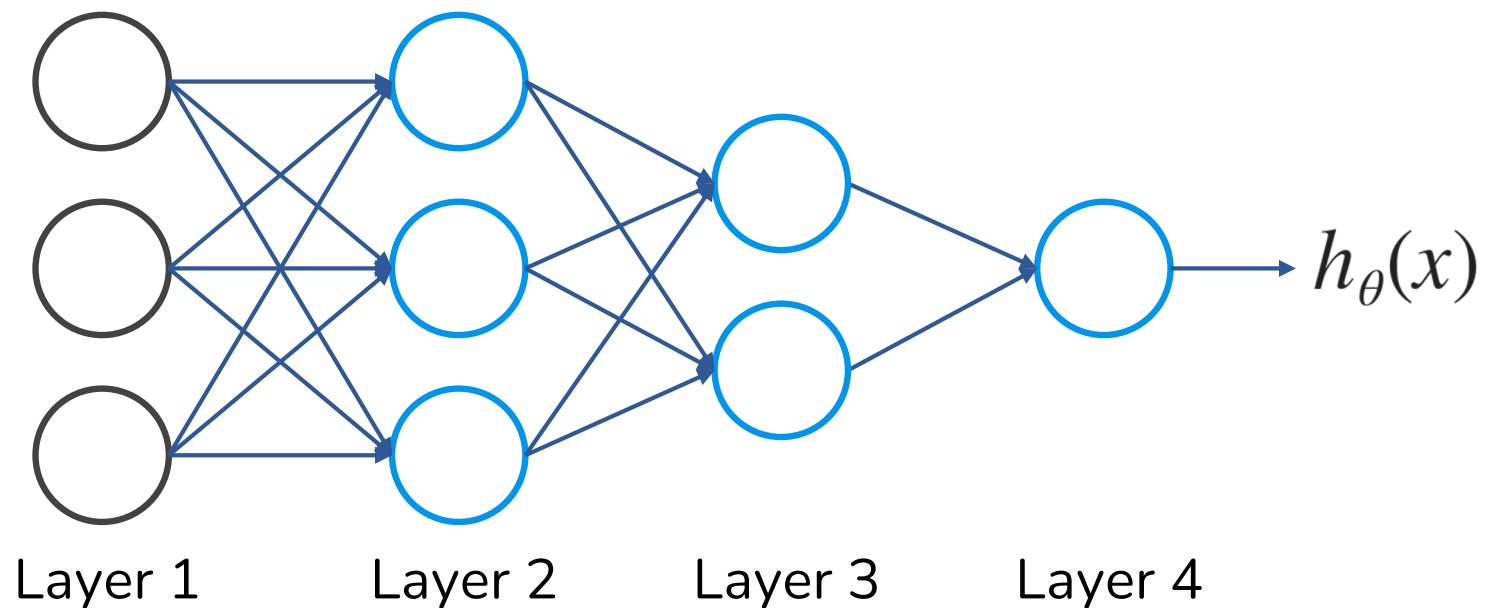


$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling function mapping from layer j to layer $j + 1$

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



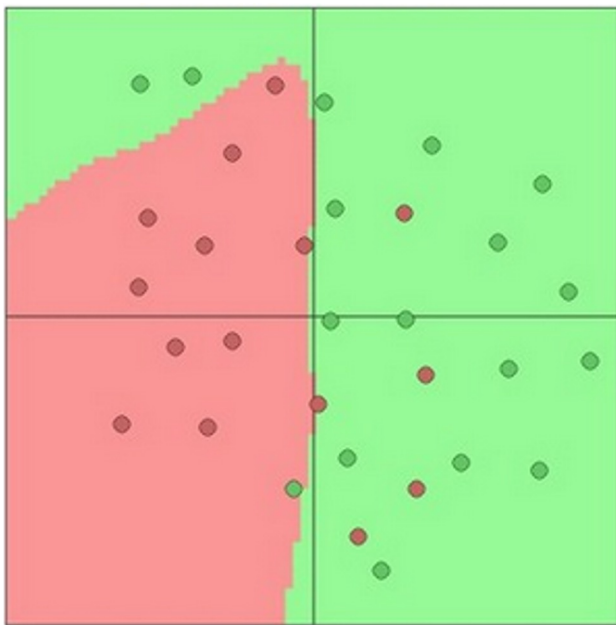
 **Complexity**

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

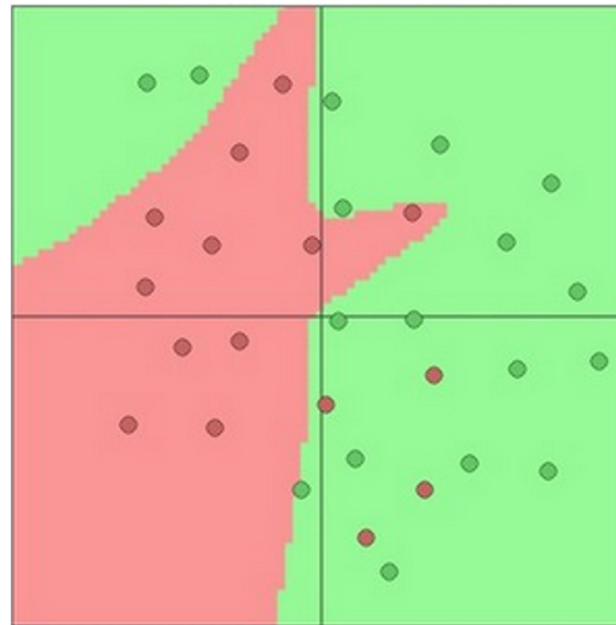
Neural Network Intuition

Toy 2d classification with 2-layer neural network

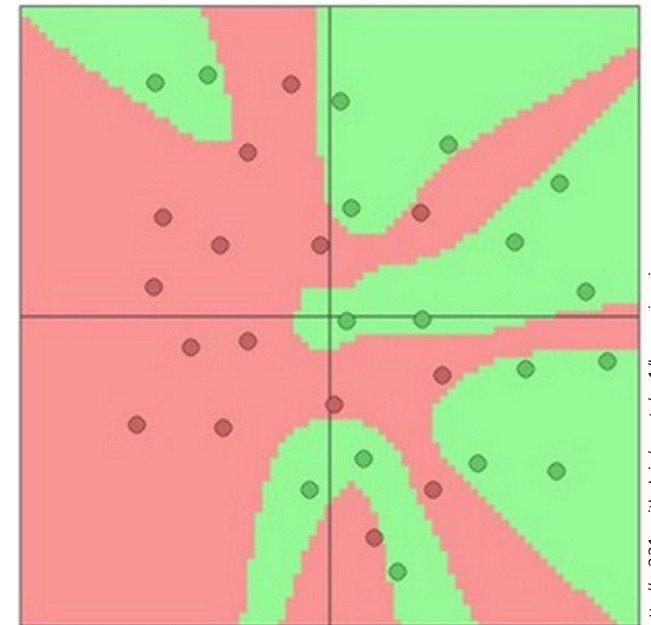
<http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>



3 hidden neurons



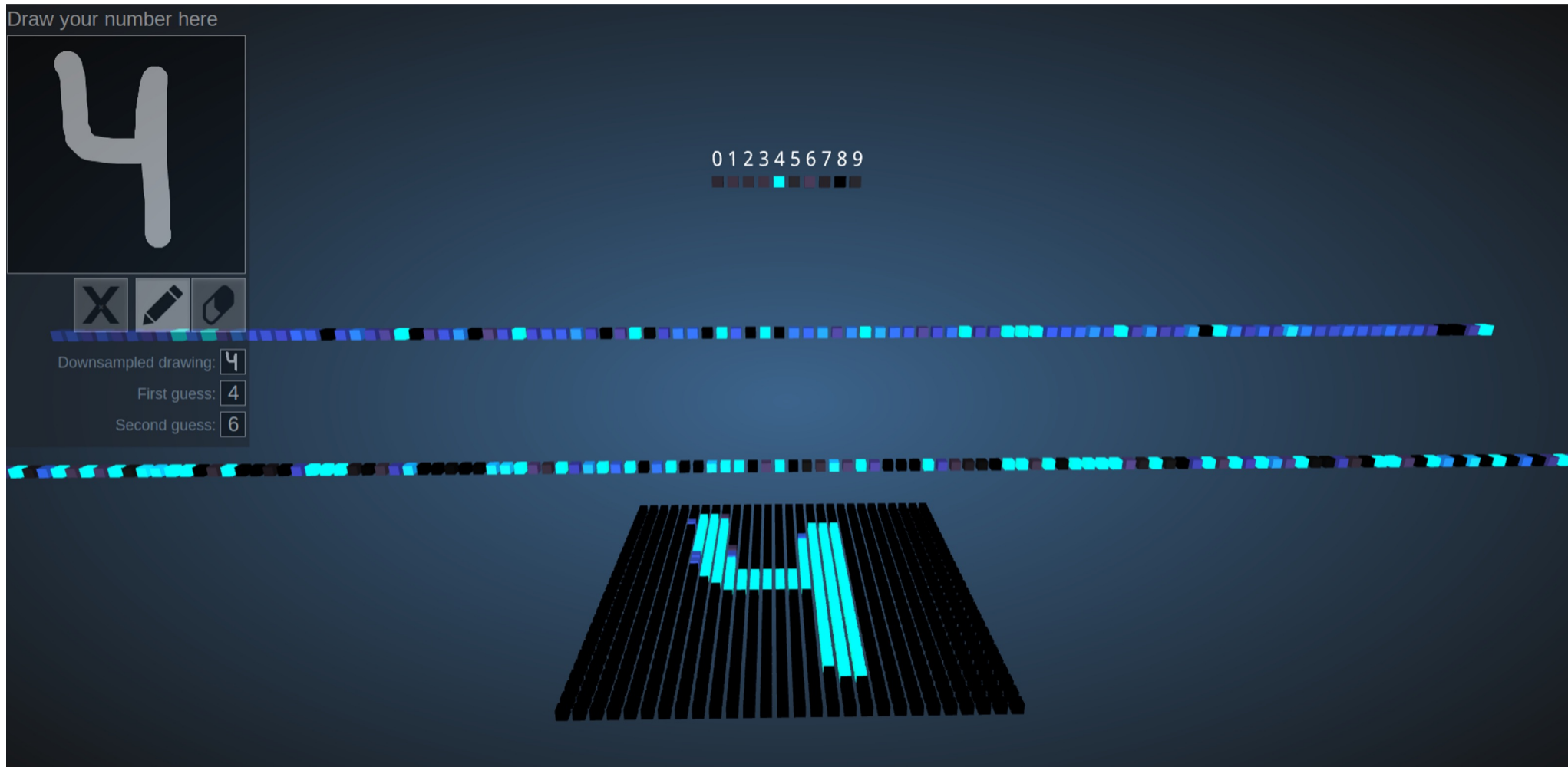
6 hidden neurons



20 hidden neurons

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

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



To join, go to: ahaslides.com/E7KI5 



Upload an images where you fool the neural network.

Get Feedback

✓ Slide 1 selected for PowerPoint [Switch slide](#)



 0  0 / 100 

Training a Neural Network

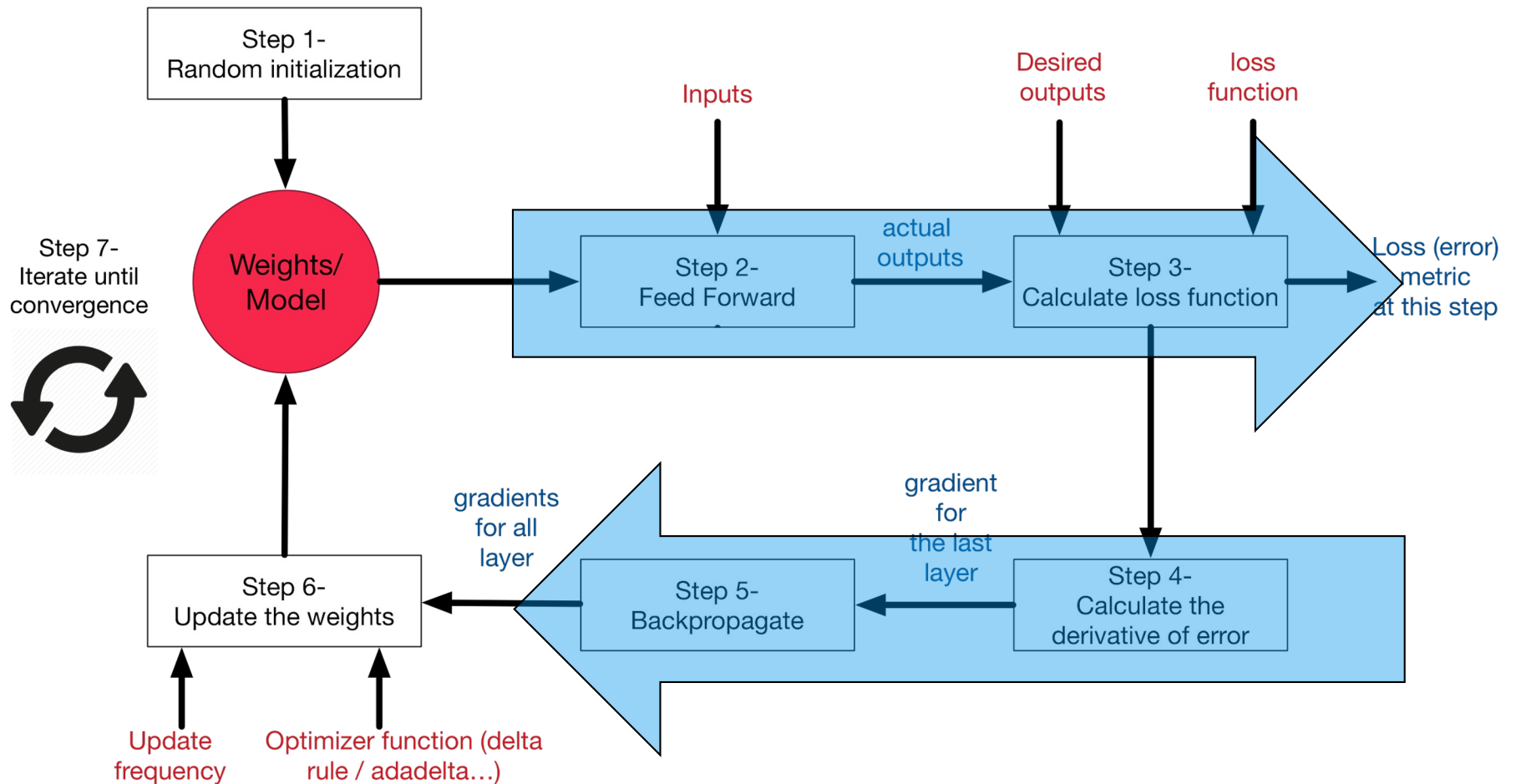
Training a Neural Network

- 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)

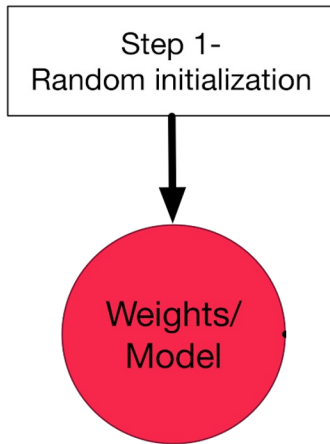
Training a Neural Network

- **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



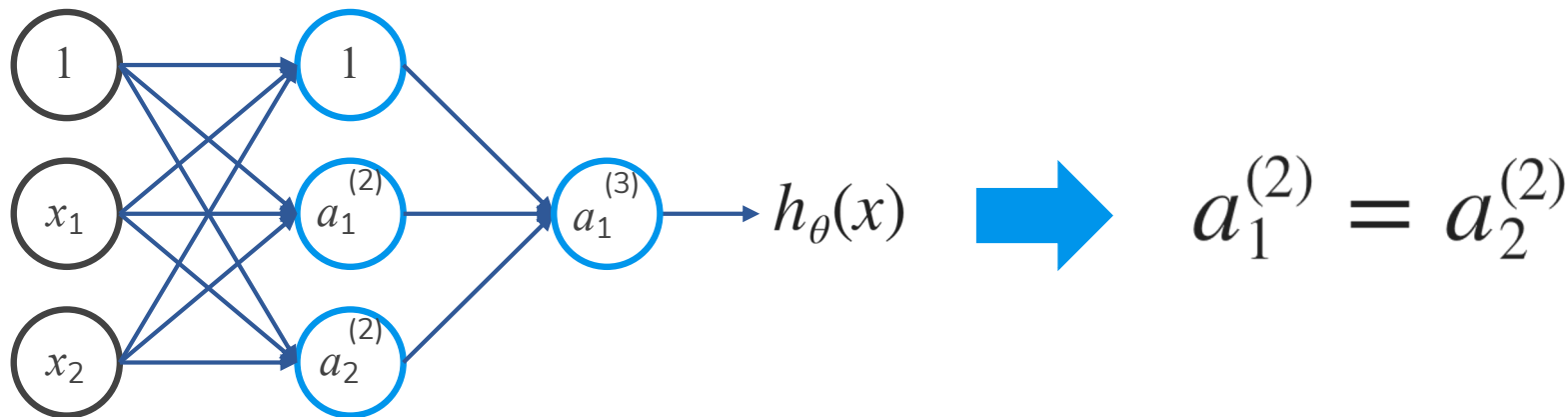
Training a Neural Network



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.

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>

```
W = np.random.randn(n) * sqrt(2.0/n)
```

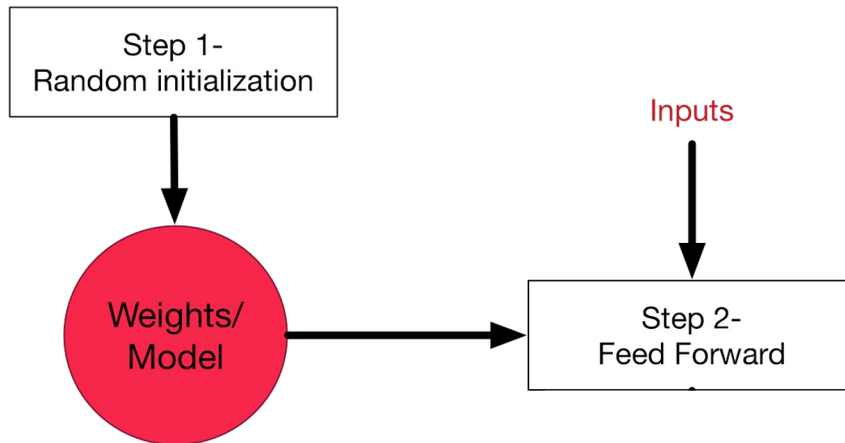
- He initialization [He et al., 2015]: “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification” <https://arxiv.org/pdf/1502.01852>

Today's Initialization

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

```
W = 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) :

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

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

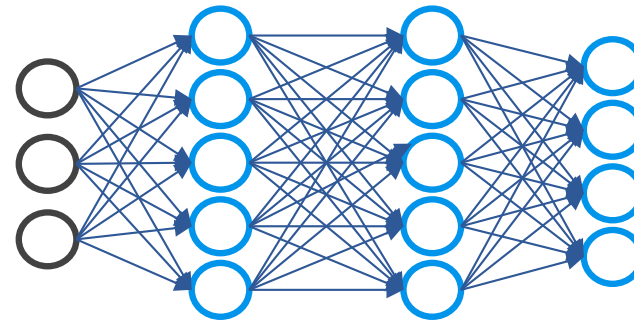
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$

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

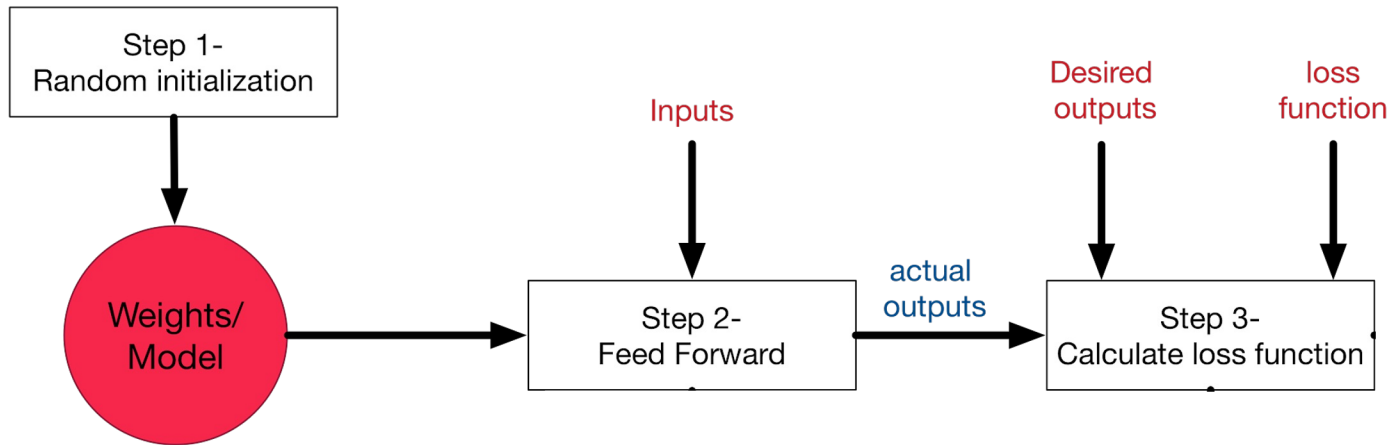
$$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$$

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

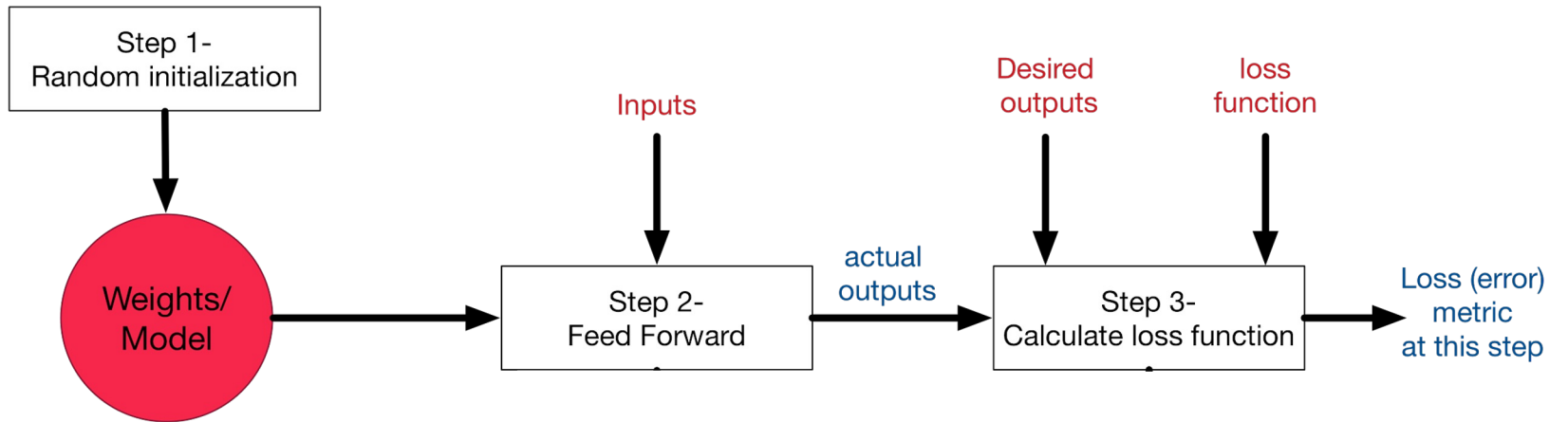
$$a^{(4)} = h_{\Theta}(x) = g(z^{(4)})$$



Training a Neural Network



Training a Neural Network



[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

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck



Example dataset: **CIFAR-10**

10 labels

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

10,000 test images

Classification scores



[32x32x3]

array of numbers 0...1
(3072 numbers total)

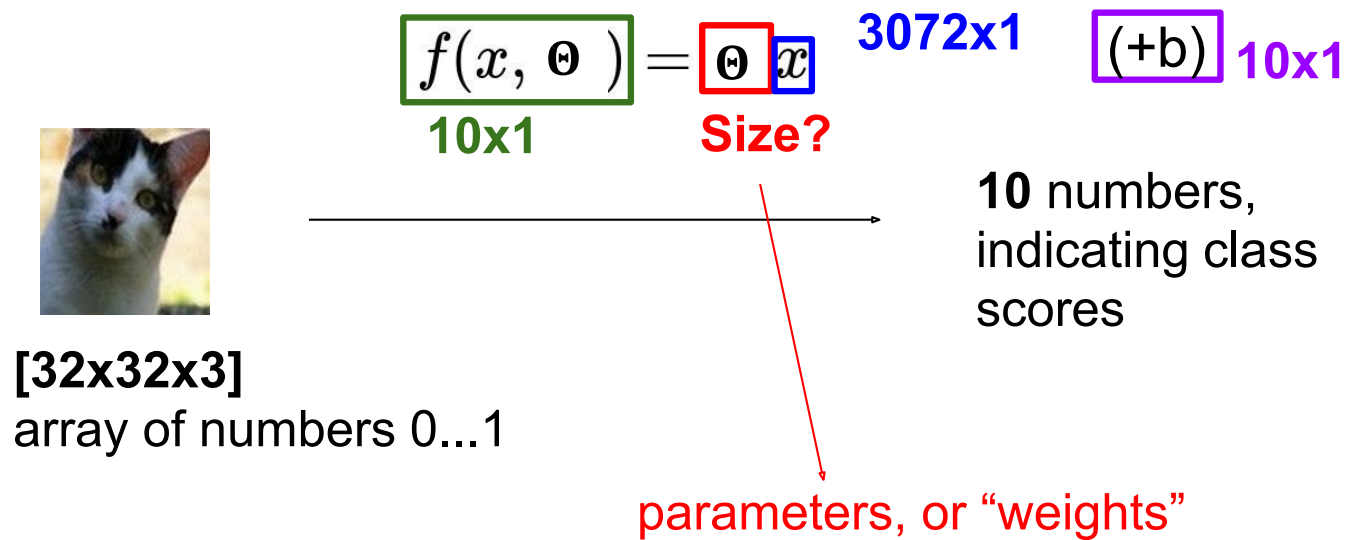
$$f(x, \Theta) = \Theta x$$

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



10 numbers,
indicating class
scores

Linear classifier



To join, go to: ahaslides.com/KTISE 



Quiz question 1 of 1

0 players ready

Waiting for players to join...

Get Feedback

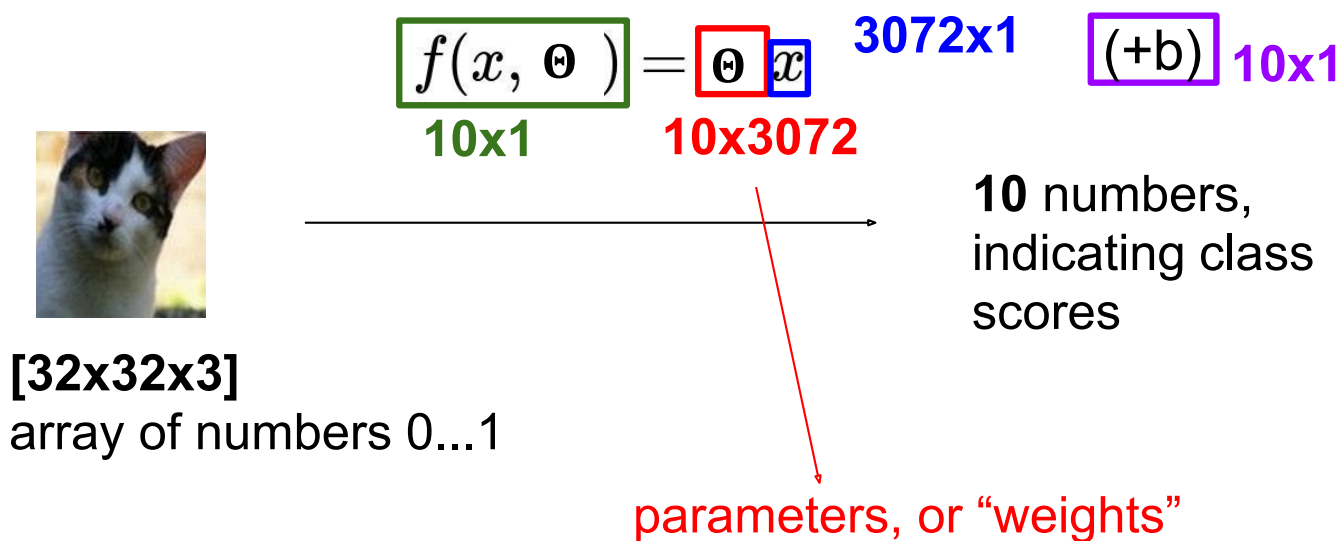
Start 

✓ Slide 1 selected for PowerPoint [Switch slide](#)



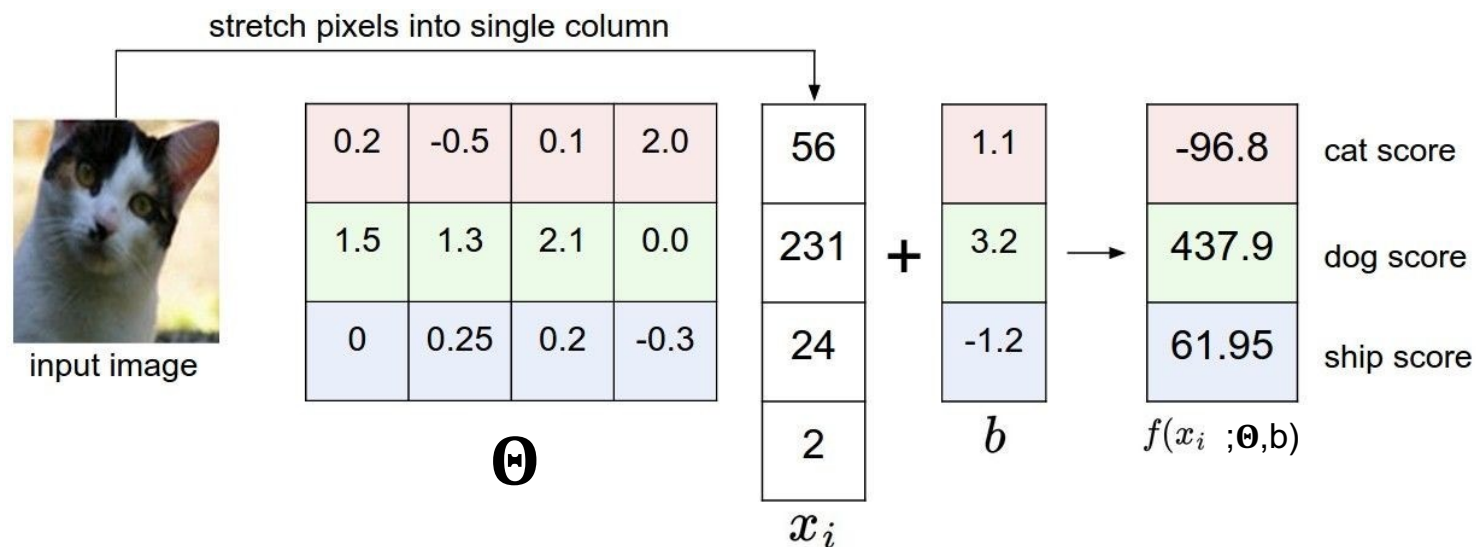
 0  0 / 100 

Linear classifier



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

$f(x, \theta) = \theta^T x$ 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 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		

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:

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

$$L = (2.9 + 0 + 12.9)/3 \\ = 15.8 / 3 = 5.3$$

Linear classifier: Hinge loss

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

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

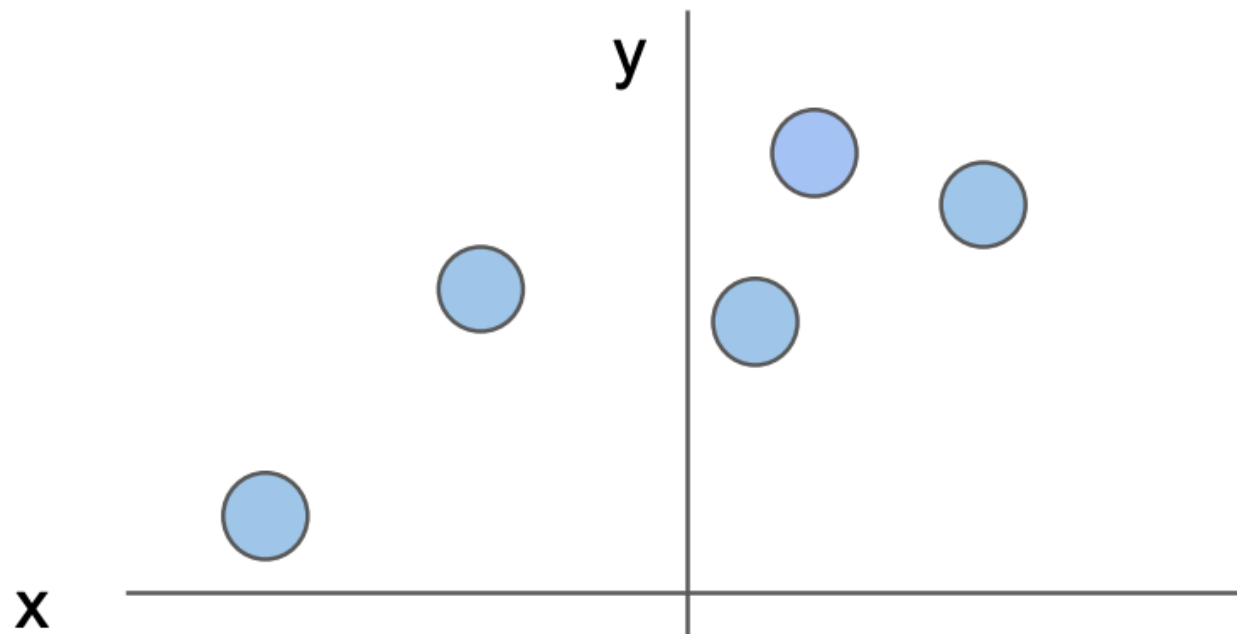
$$L(\Theta) = \frac{1}{N} \sum_{i=1}^N \underline{L_i(f(x_i, \Theta), y_i)}$$

Linear classifier: Regularization

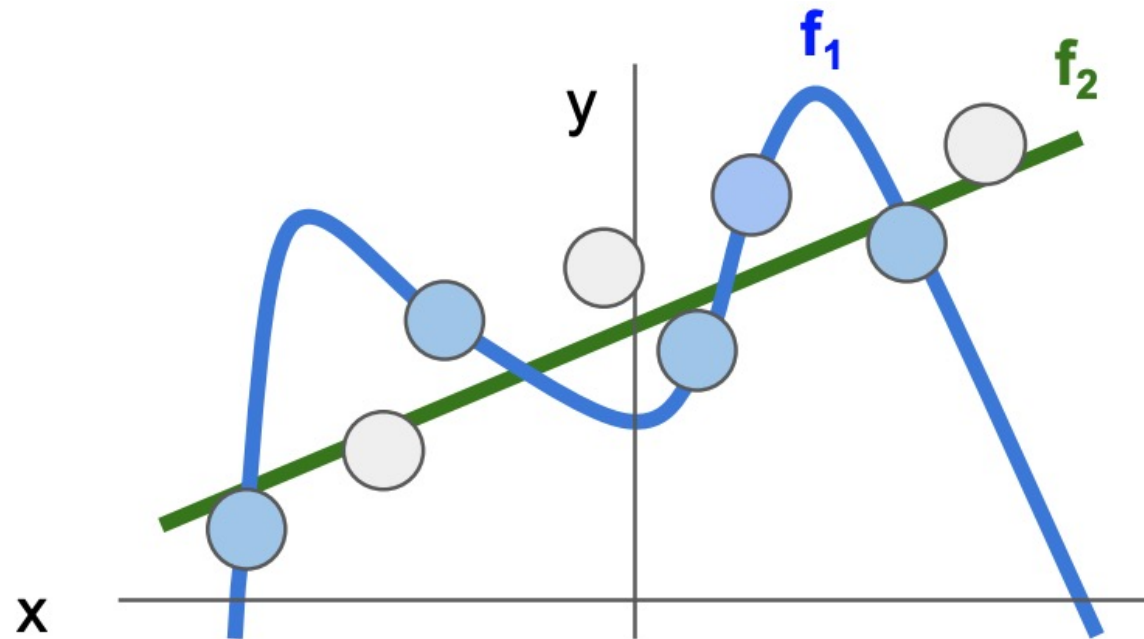
$$L(\Theta) = \frac{1}{N} \sum_{i=1}^N L_i(f(x_i, \Theta), y_i)$$


Data loss: Model predictions
should match training data

Linear classifier: Regularization Intuition



Linear classifier: Regularization – Prefer simpler models



Linear classifier: Regularization – Overfitting



Slide Credit: Prof. Sandra Avila - UNICAMP

Linear classifier: Regularization

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

Data loss: Model predictions should match training data

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

Linear classifier: Regularization

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

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

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

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

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

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

L2 Regularization

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

Which of Θ_1 or Θ_2 will the L2 regularizer prefer?

Linear classifier: Hinge loss

Weight Regularization

λ = regularization strength
(hyperparameter)

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

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 5](#): 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

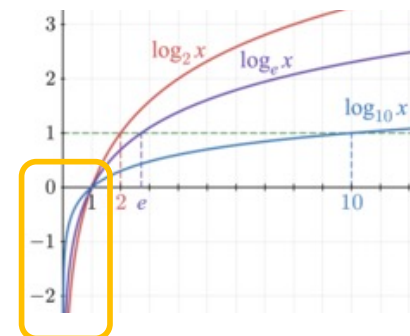
$$s = f(x_i; \Theta)$$

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

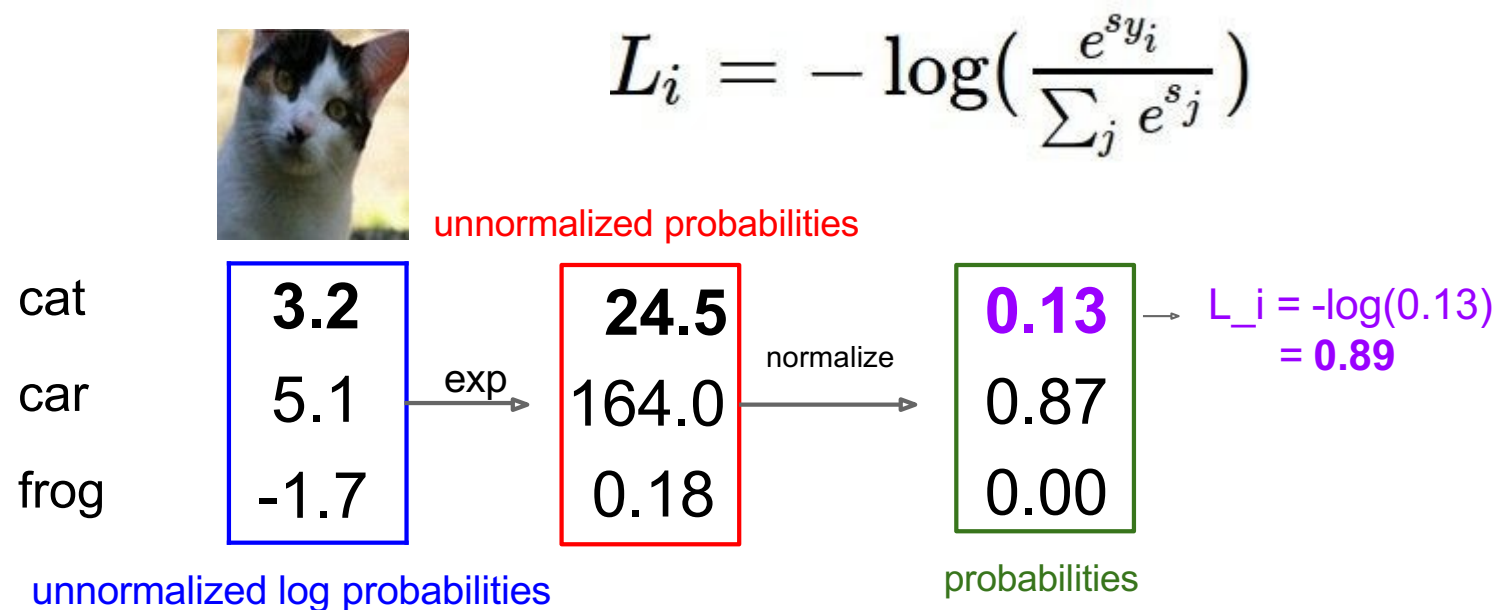
$$L_i = -\log P(Y = y_i|X = x_i)$$

maximize

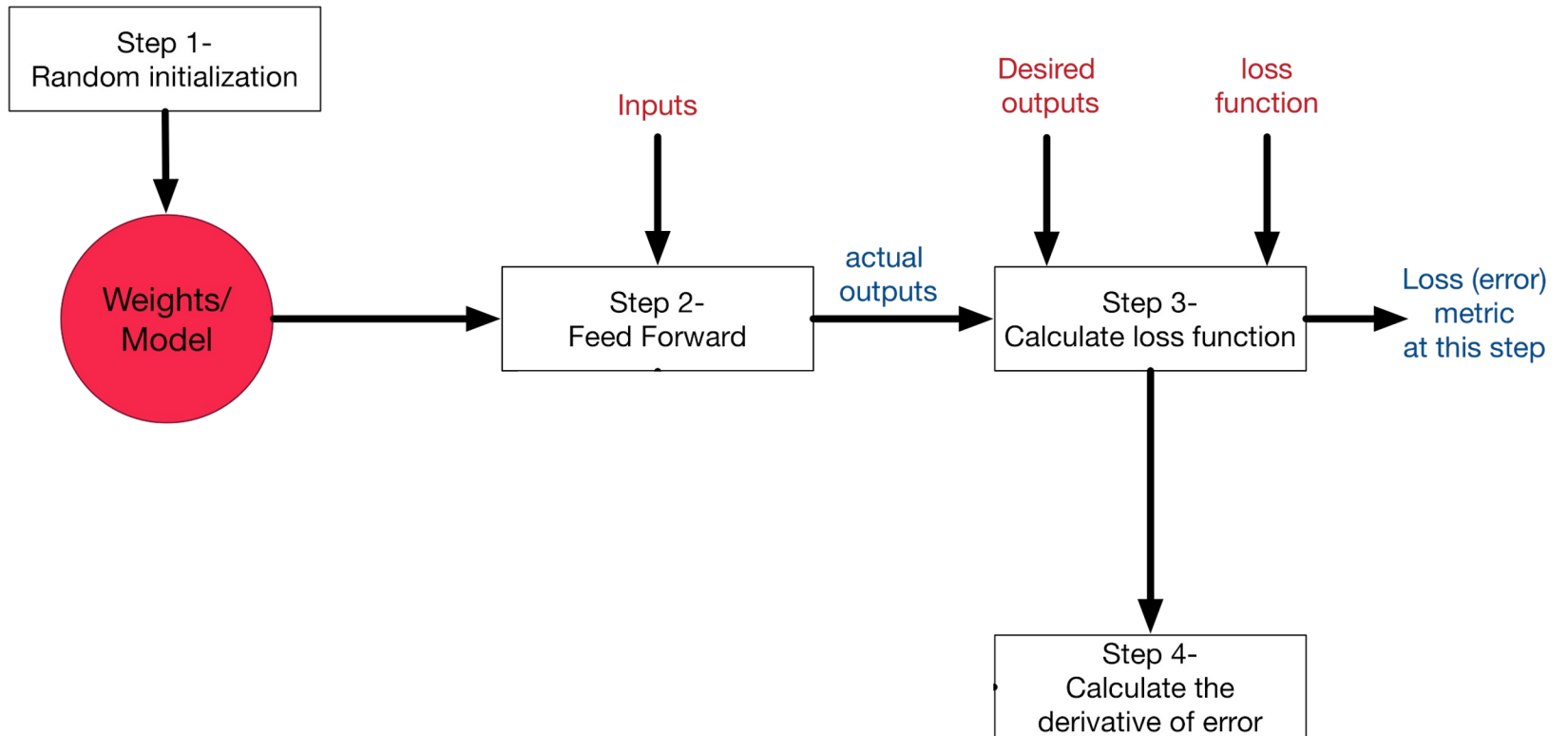
minimize



Another loss: Softmax (cross-entropy)



Training a Neural Network



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

Gradient Computation: Backpropagation Algorithm

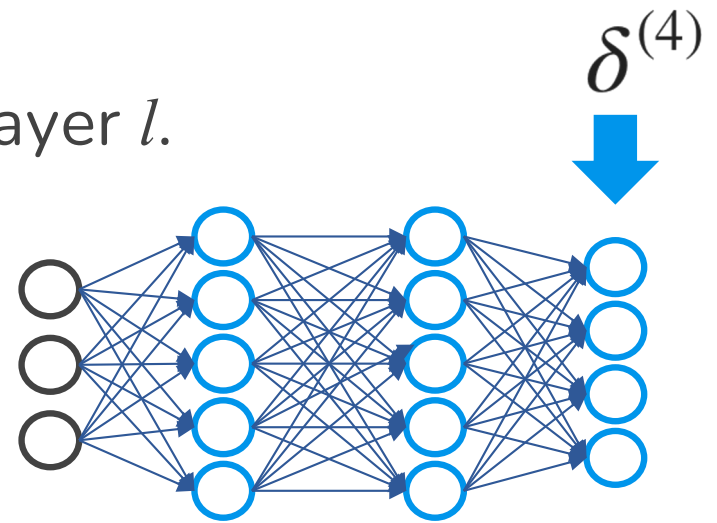
Intuition: $\delta_j^{(l)}$ = “error” of node j in layer l .

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

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



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



Gradient Computation: Backpropagation Algorithm

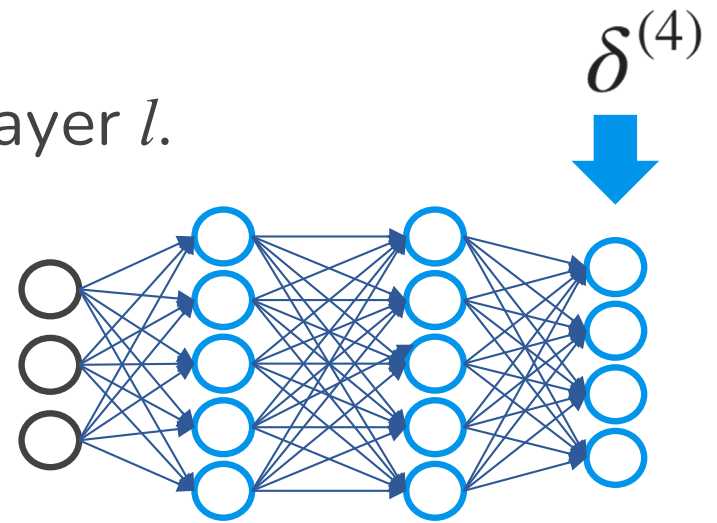
Intuition: $\delta_j^{(l)}$ = “error” of node j in layer l .

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

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

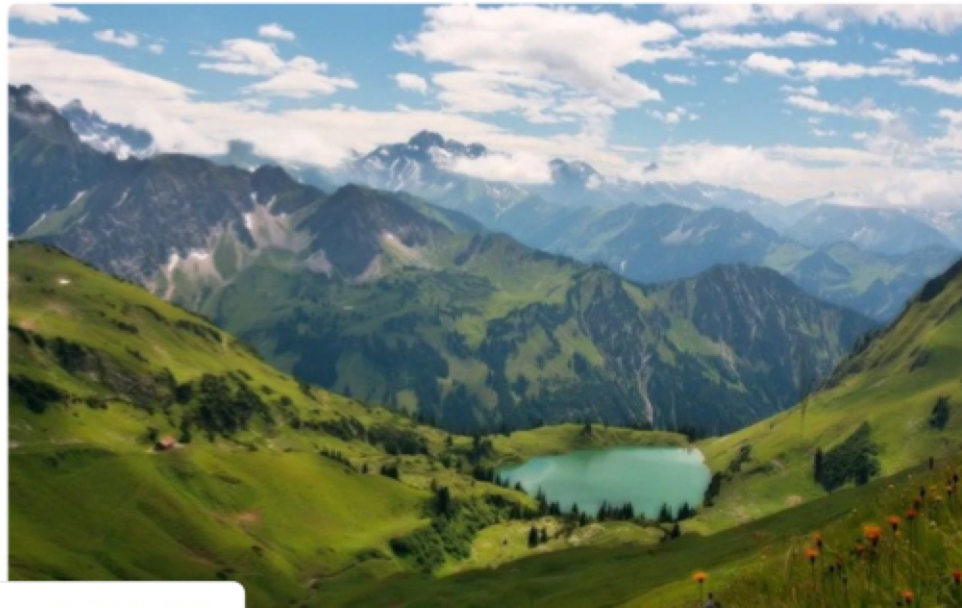
Vectorizing it, we have:

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





Given the light green mountain, select the minimum altitude.



✓ Slide 1 selected for PowerPoint [Switch slide](#)



How to minimize the loss function?



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 $\mathbf{p} = (x_1, \dots, x_n)$ in n -dimensional space as the vector:^[b]

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

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,
?,
?,
?,...]

$$(1.25322 - 1.25347)/0.0001 = -2.5$$

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

?,
?,...]

[Negative Gradient]

Decrement of loss

Min loss --> Increase weights

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,
?,
?,

$$(1.25353 - 1.25347)/0.0001 = 0.6$$

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

?,...]

[Positive Gradient]

Increment loss

Min loss --> Decrease weights

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):

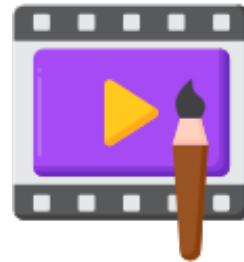
[0.34,
-1.11,
0.78 + **0.0001**,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

gradient $d\Theta$:

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

Derivative Animation



Positive Gradient --> Decrease Weights
Negative Gradient --> Increase Weights

<https://web-apps.thecoatlessprofessor.com/calculus/derivative-visualizer.html>

Moving from left to right

Loss function --> $f(x) = x^2$
Weights --> x
Gradient (Derivative) --> $f'(x) = 2x$

Patterns

- **Positive gradient $f'(x)$** means **increment loss $f(x)$**
In order to **minimize loss $f(x)$** , we should **decrease weights x** .
- **Negative gradient $f'(x)$** means **decrement loss $f(x)$**
In order to **minimize loss $f(x)$** , we should **increase weights x** .

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

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



gradient $d\Theta$:

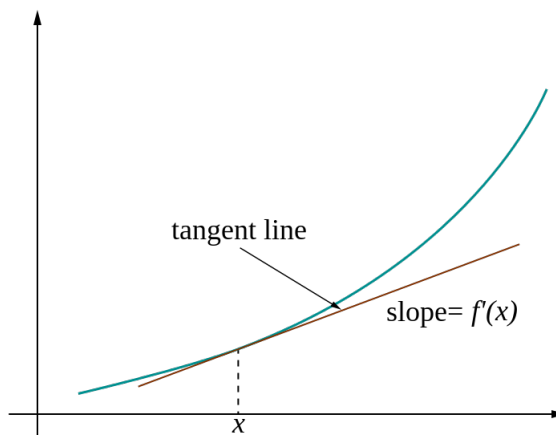
[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,...]

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

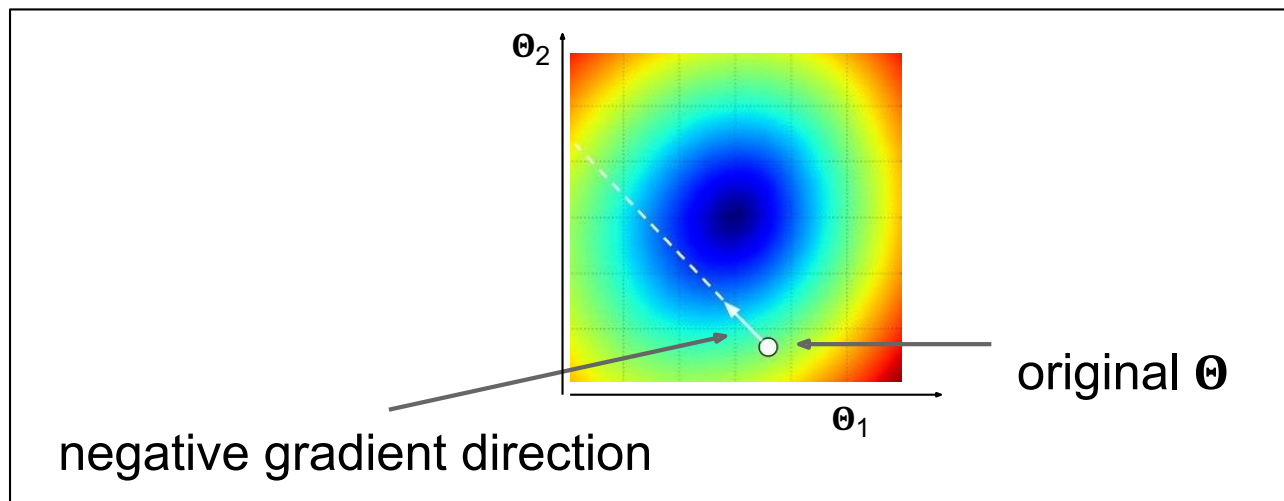
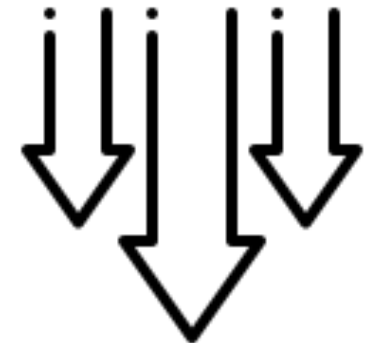


Figure from Andrej Karpathy

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 4: Gradient Descent

Duration: 10 min



To join, go to: ahaslides.com/HNII9 



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

Get Feedback

✓ Slide 1 selected for PowerPoint [Switch slide](#)



Group into themes

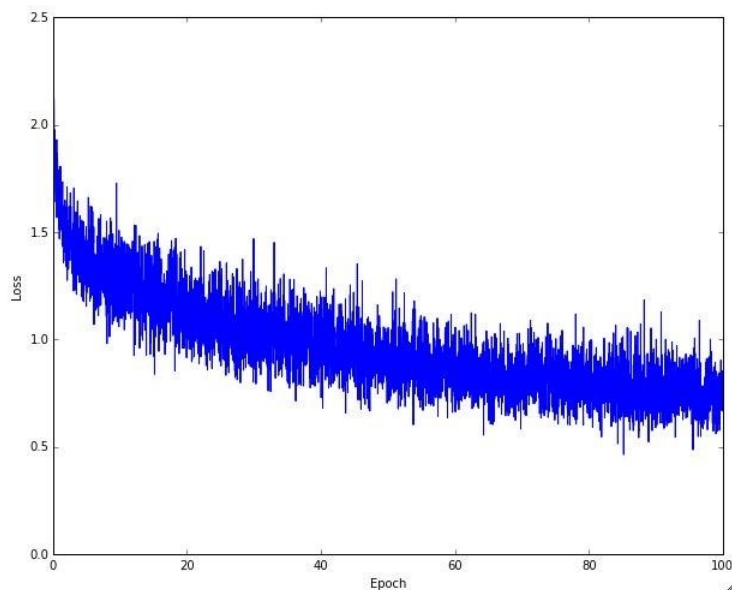


 0  0 / 100 

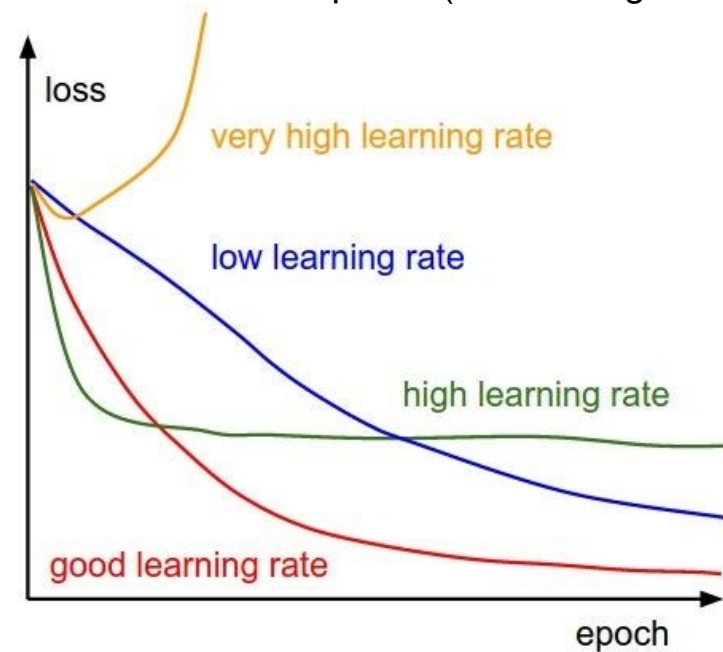
How to compute the loss/gradient?

- 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

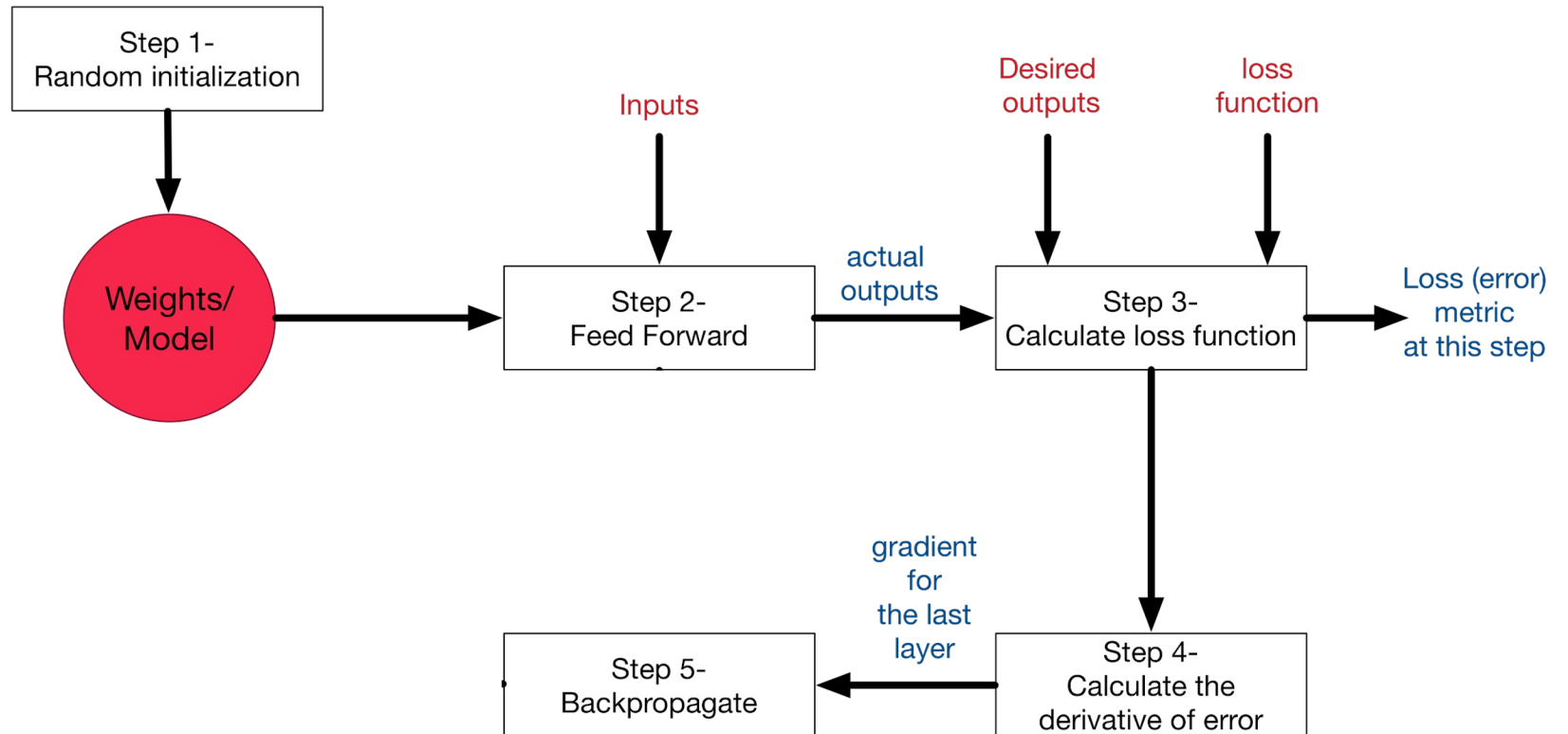


The effects of step size (or “learning rate”)



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

Training a Neural Network



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

Gradient descent in multi-layer nets

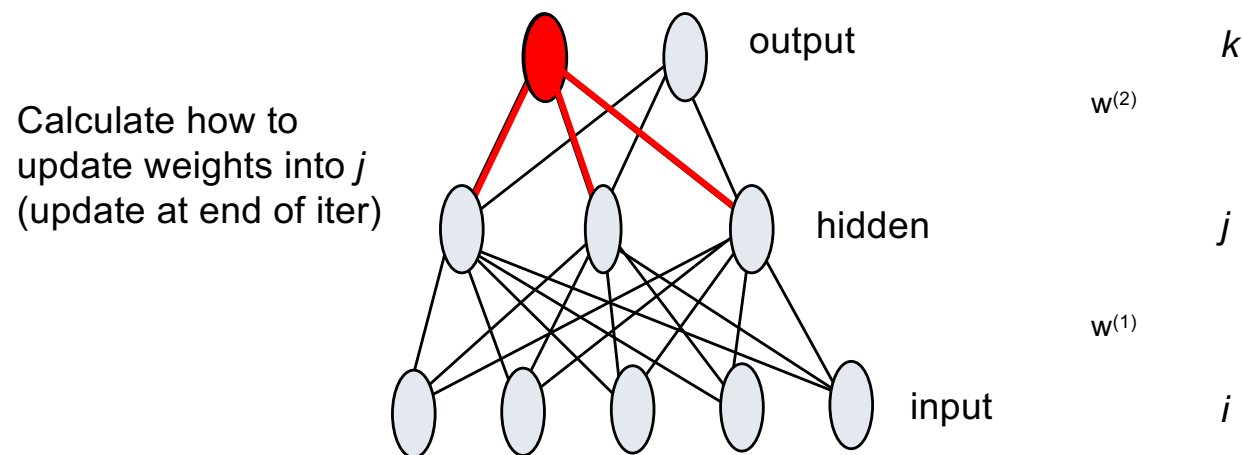
- We'll update weights
- Move in direction opposite to gradient:

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

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

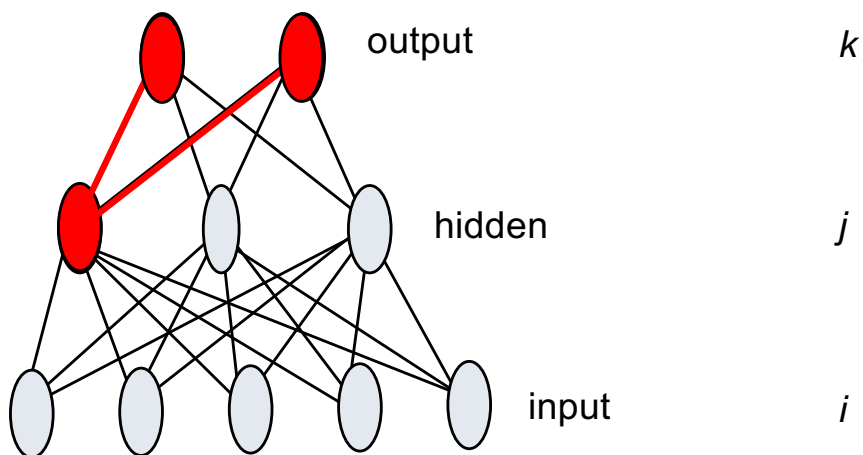
Backpropagation: Graphic example

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



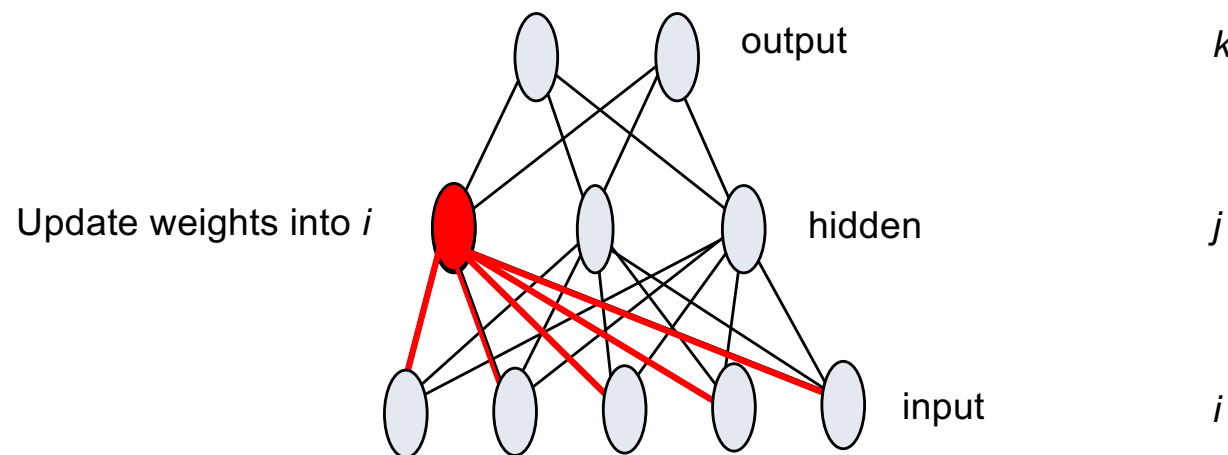
Backpropagation: Graphic example

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



Backpropagation: Graphic example

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



Backpropagation

A Simple Example

Backpropagation: A Simple Example

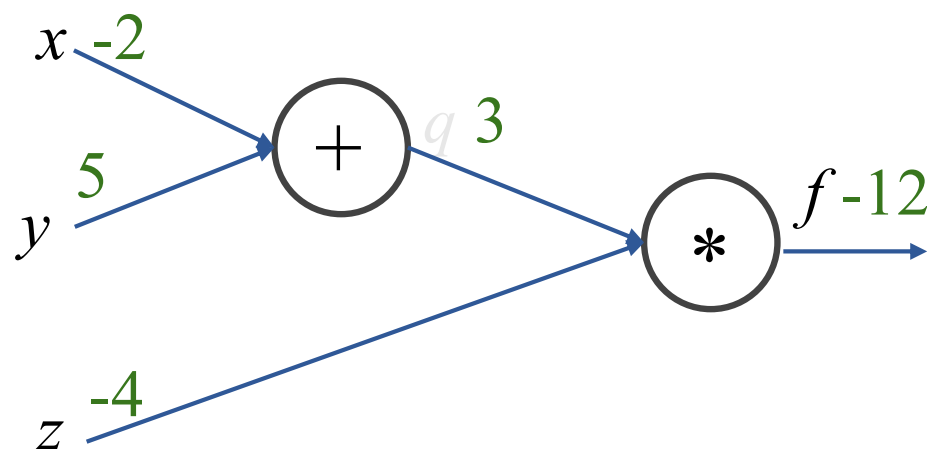
$$f(x, y, z) = (x + y)z$$

$$\text{e.g., } x = -2, y = 5, z = -4$$

Backpropagation: A Simple Example

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

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



Backpropagation: A Simple Example

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

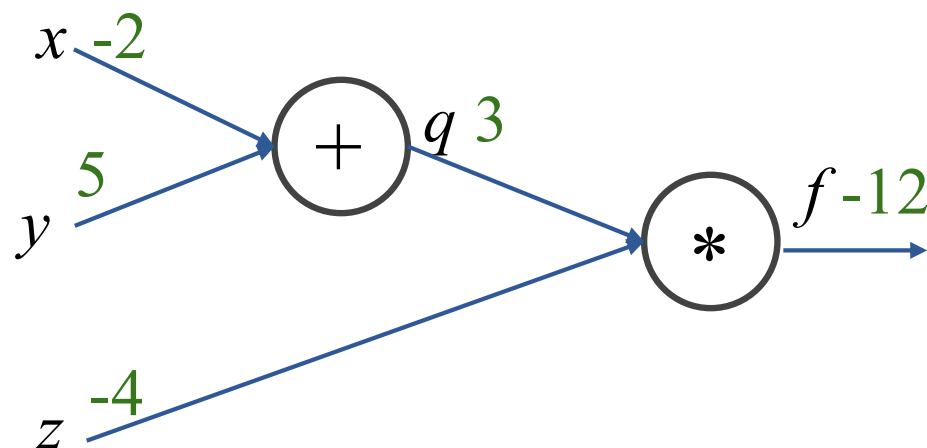
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

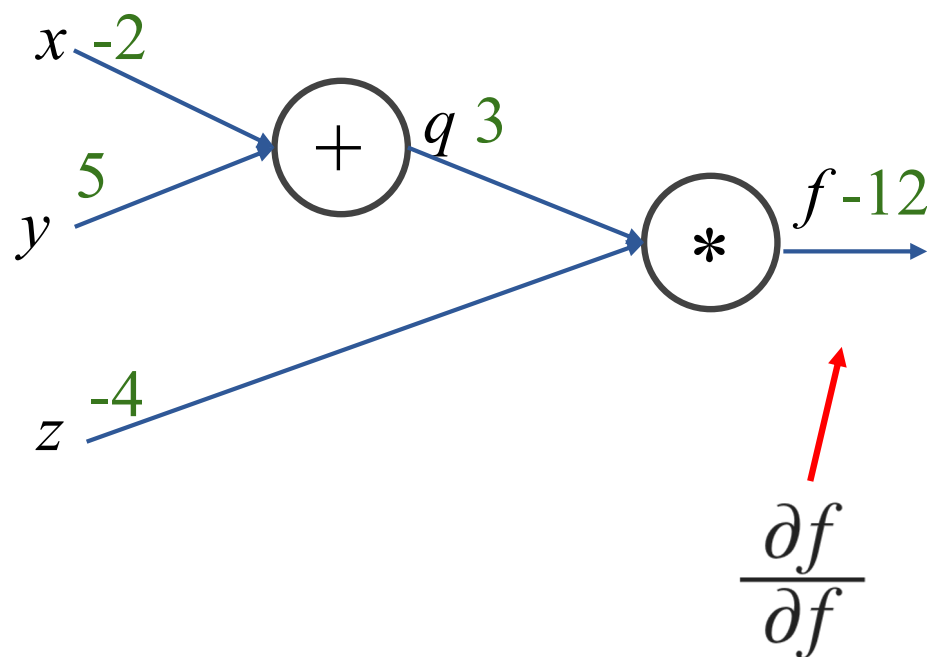
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

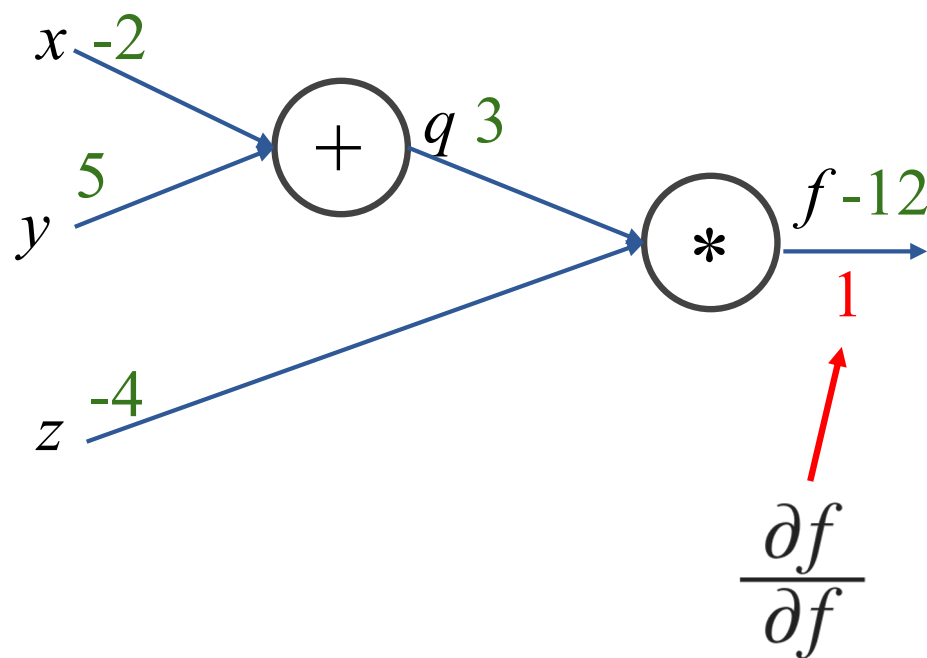
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

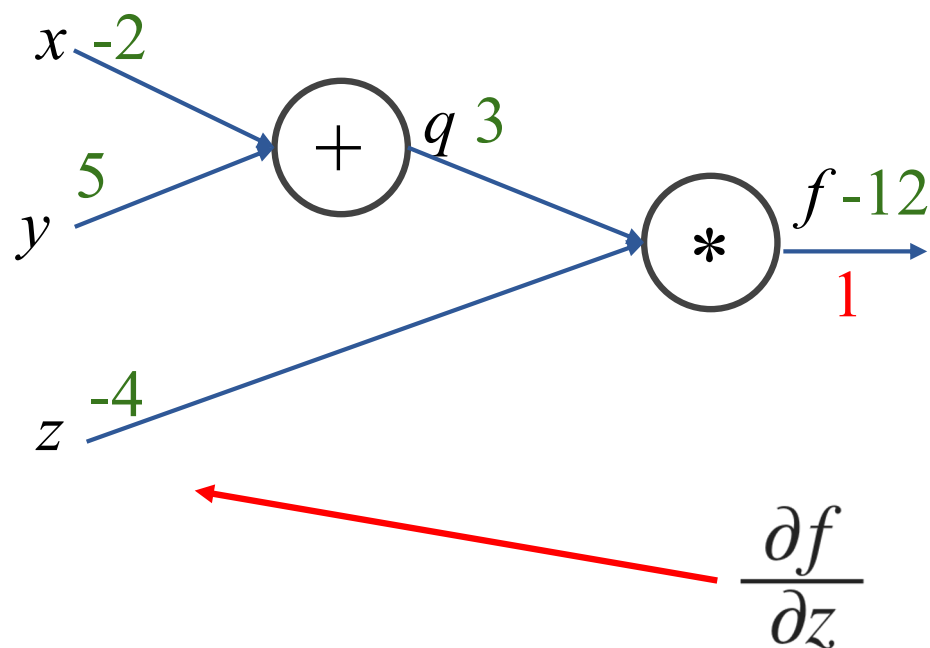
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

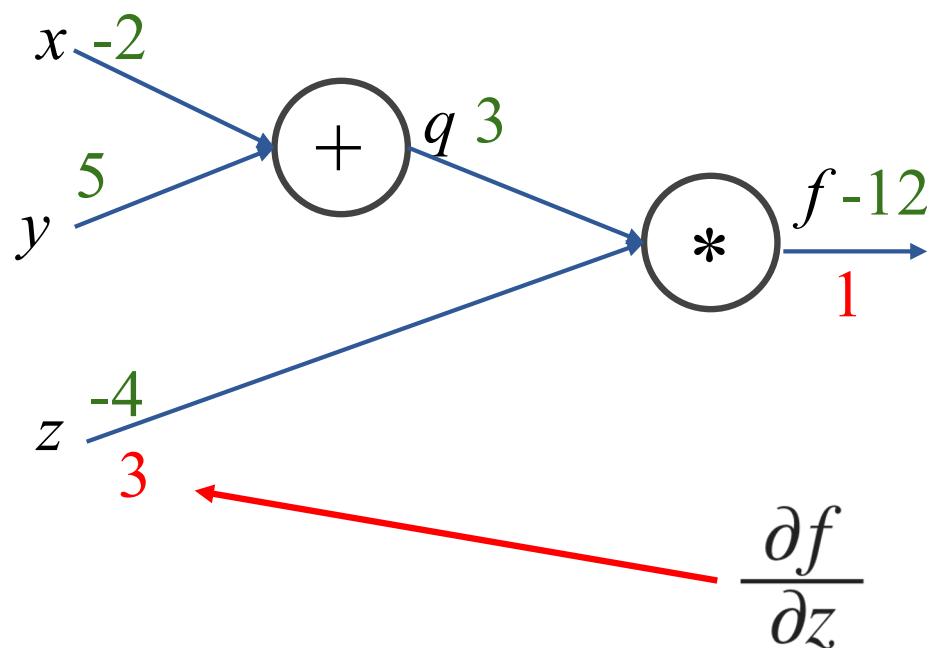
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

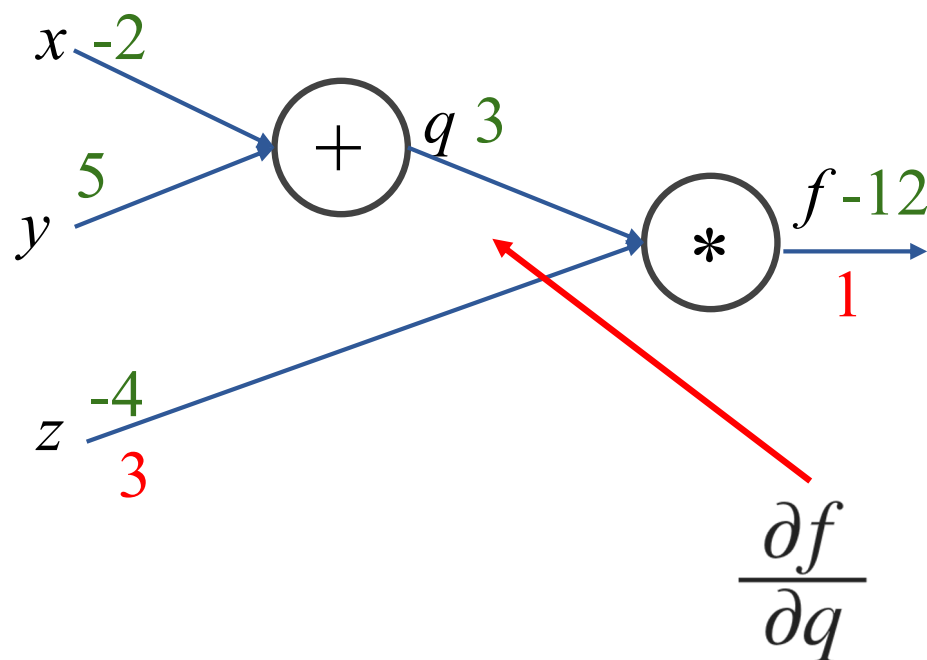
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

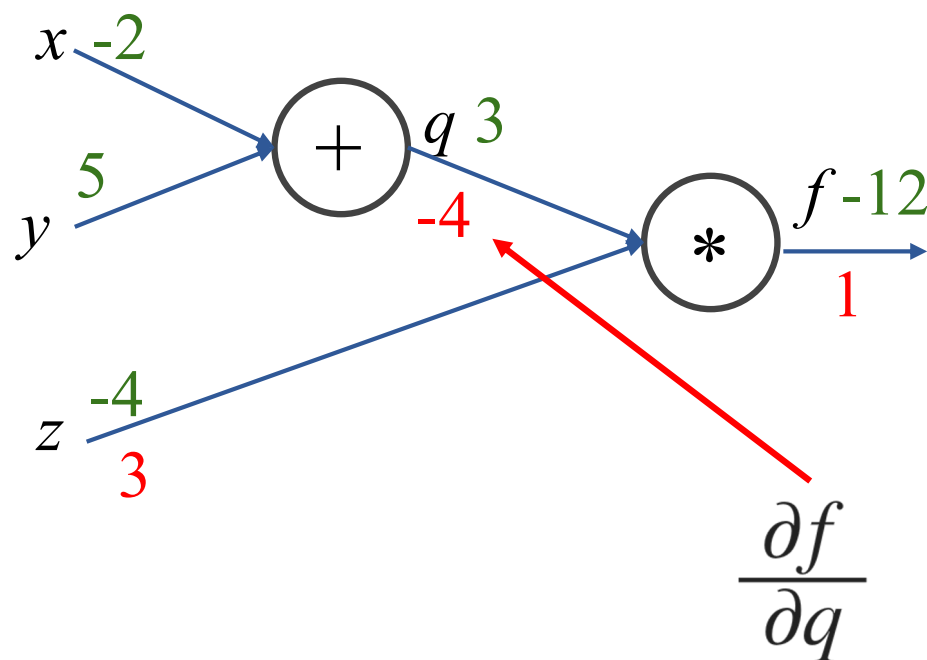
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

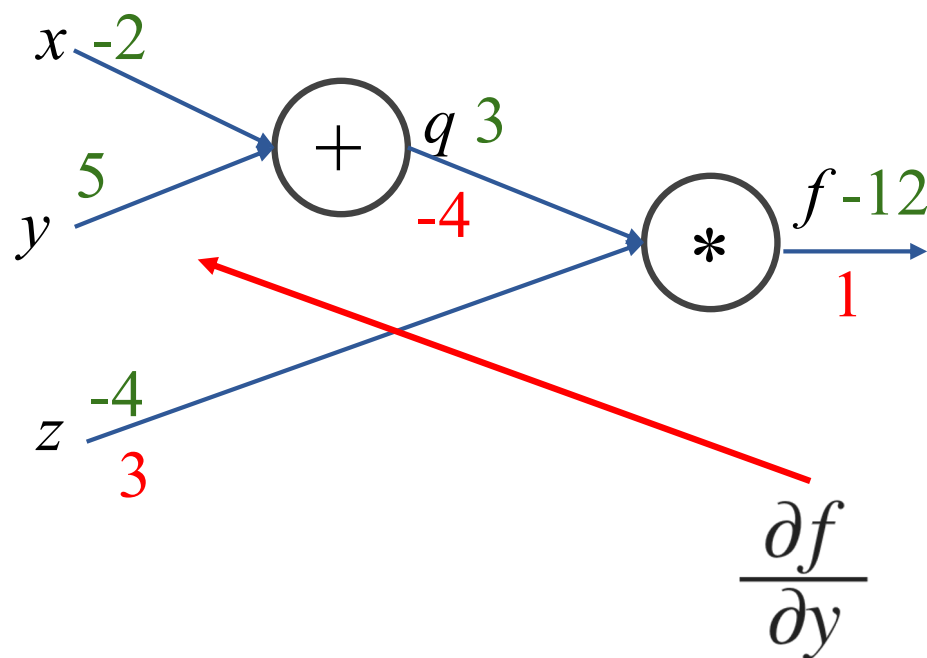
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

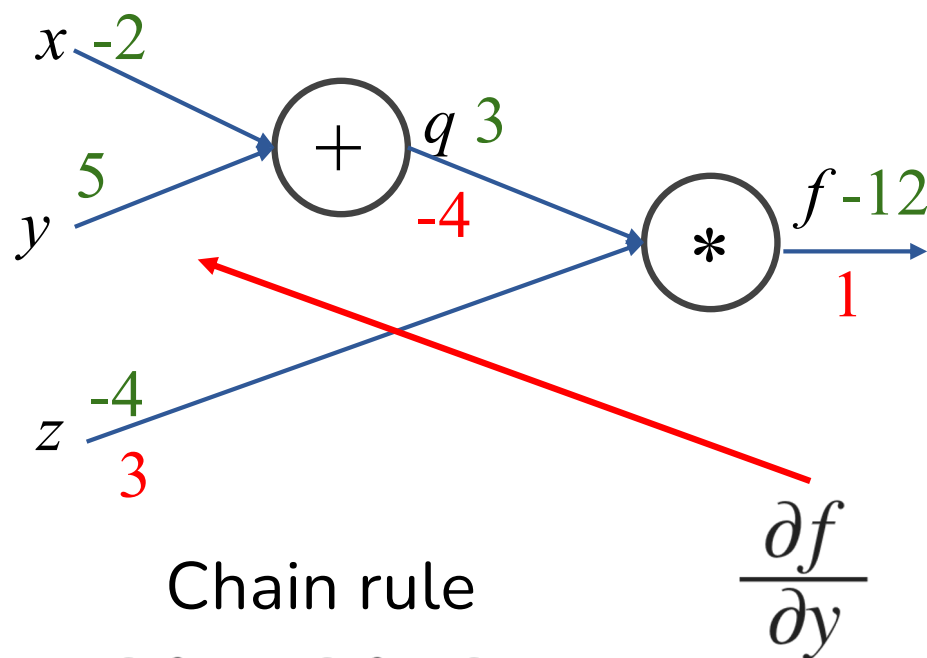
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

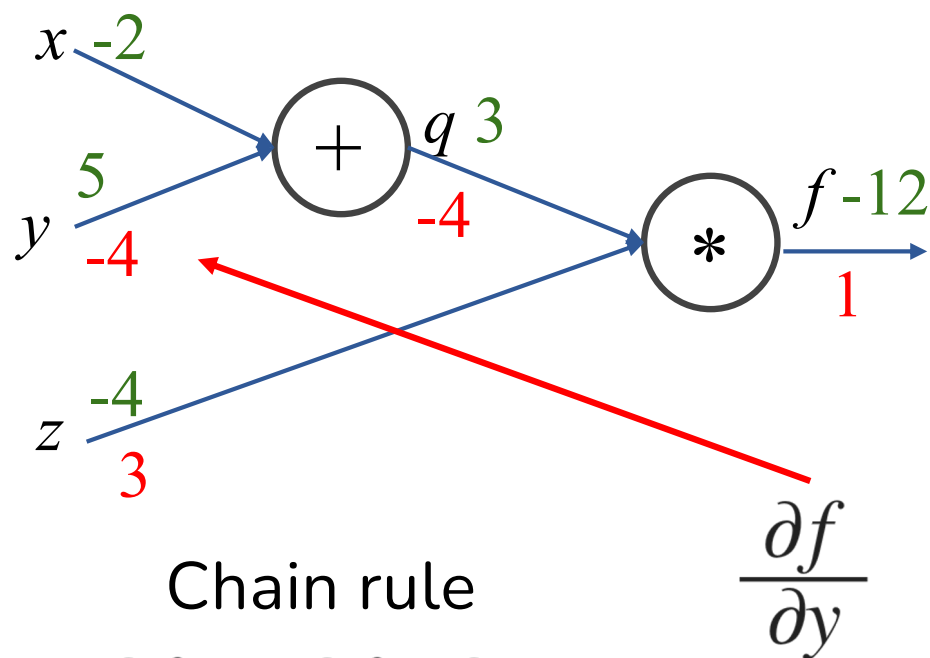
$$\text{e.g., } x = -2, y = 5, z = -4$$

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

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

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

Andrej Karpathy



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

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

Backpropagation: A Simple Example

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

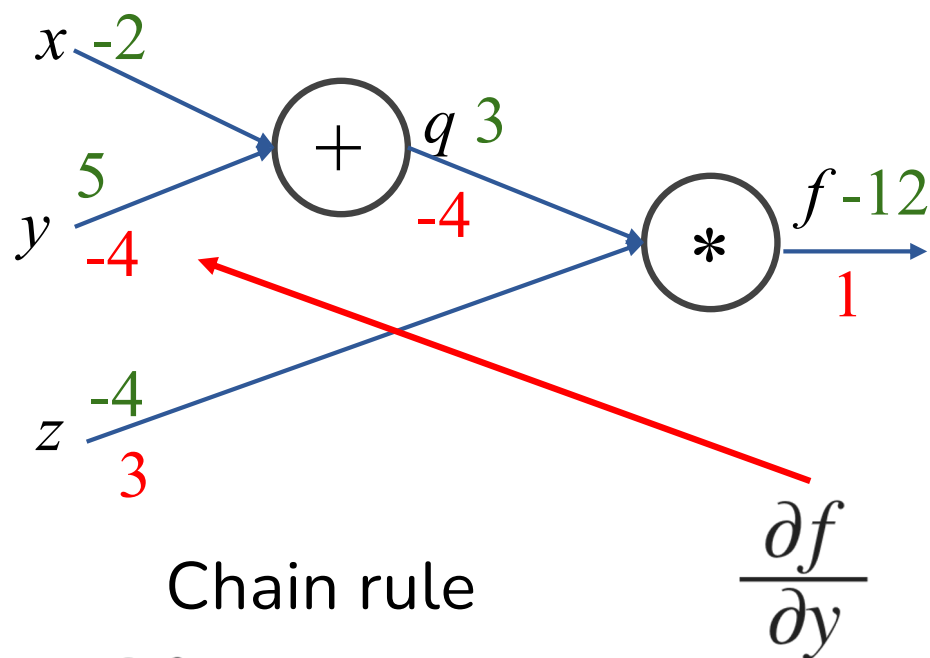
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

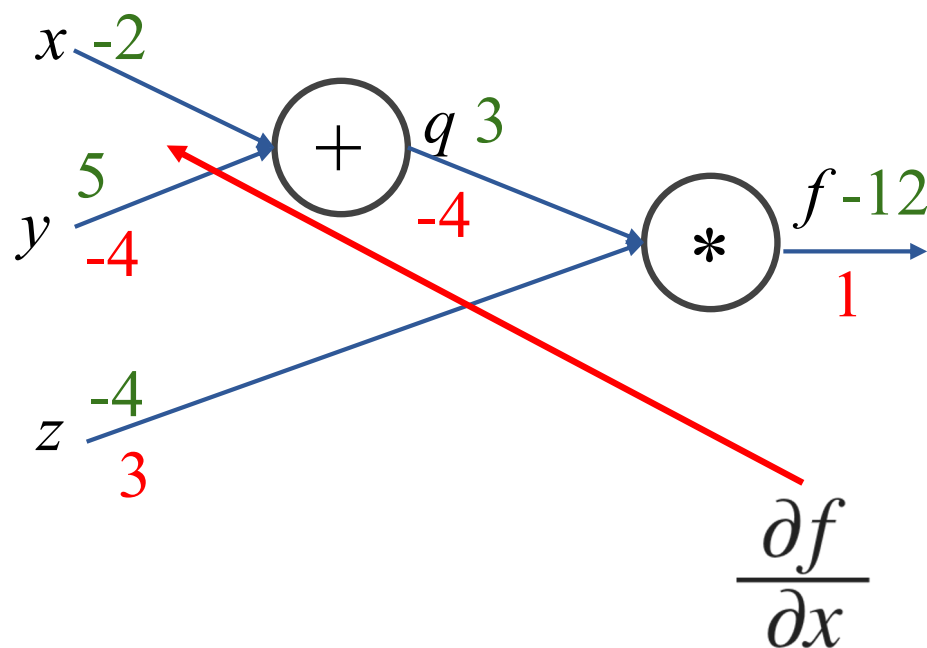
e.g., $x = -2$, $y = 5$, $z = -4$

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

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

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

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

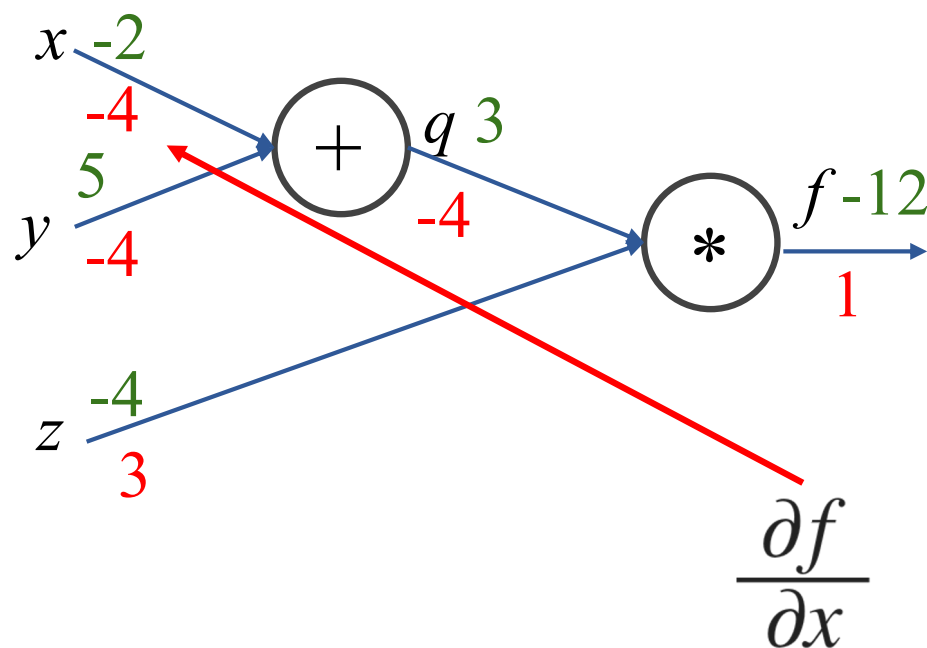
e.g., $x = -2$, $y = 5$, $z = -4$

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

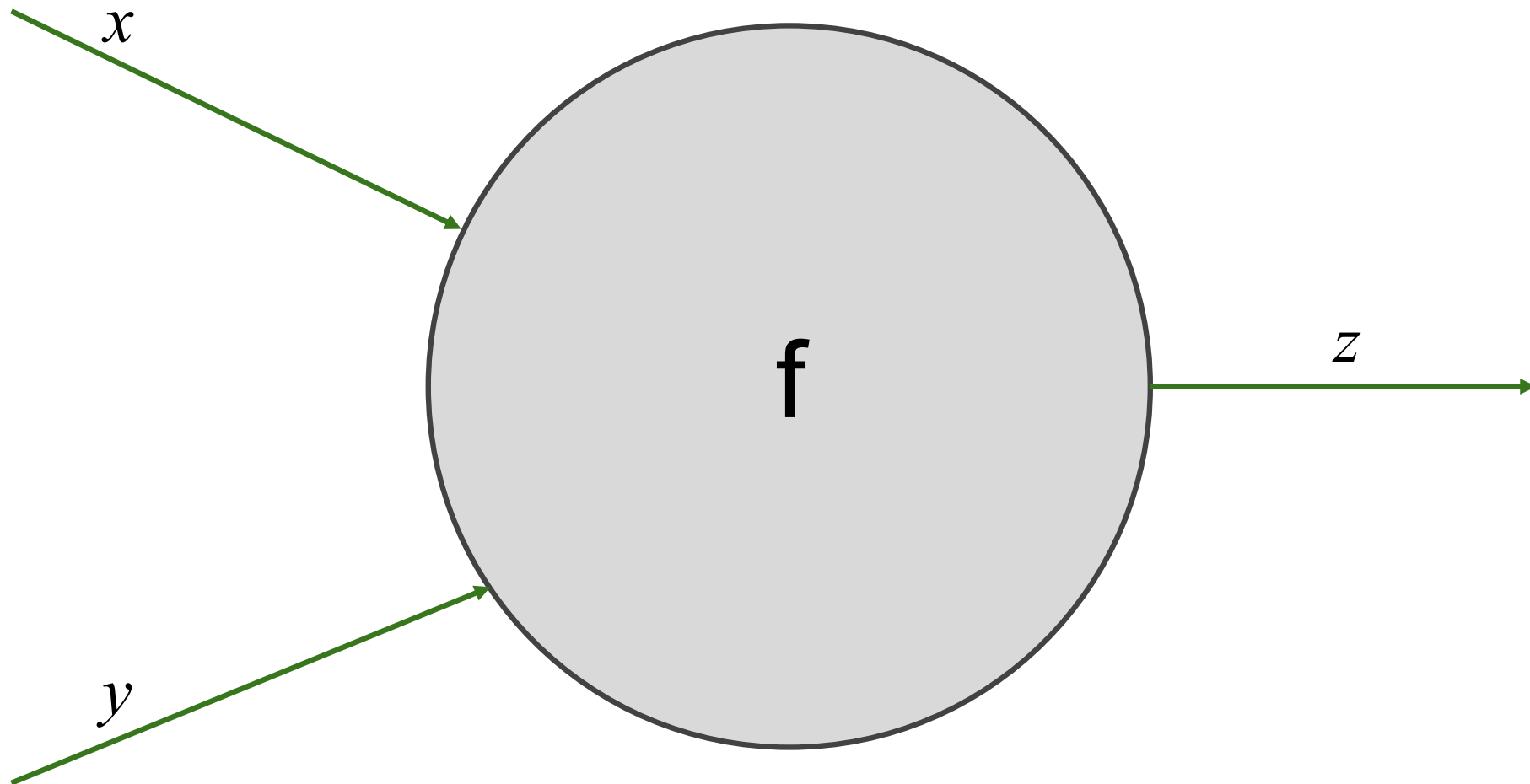
$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

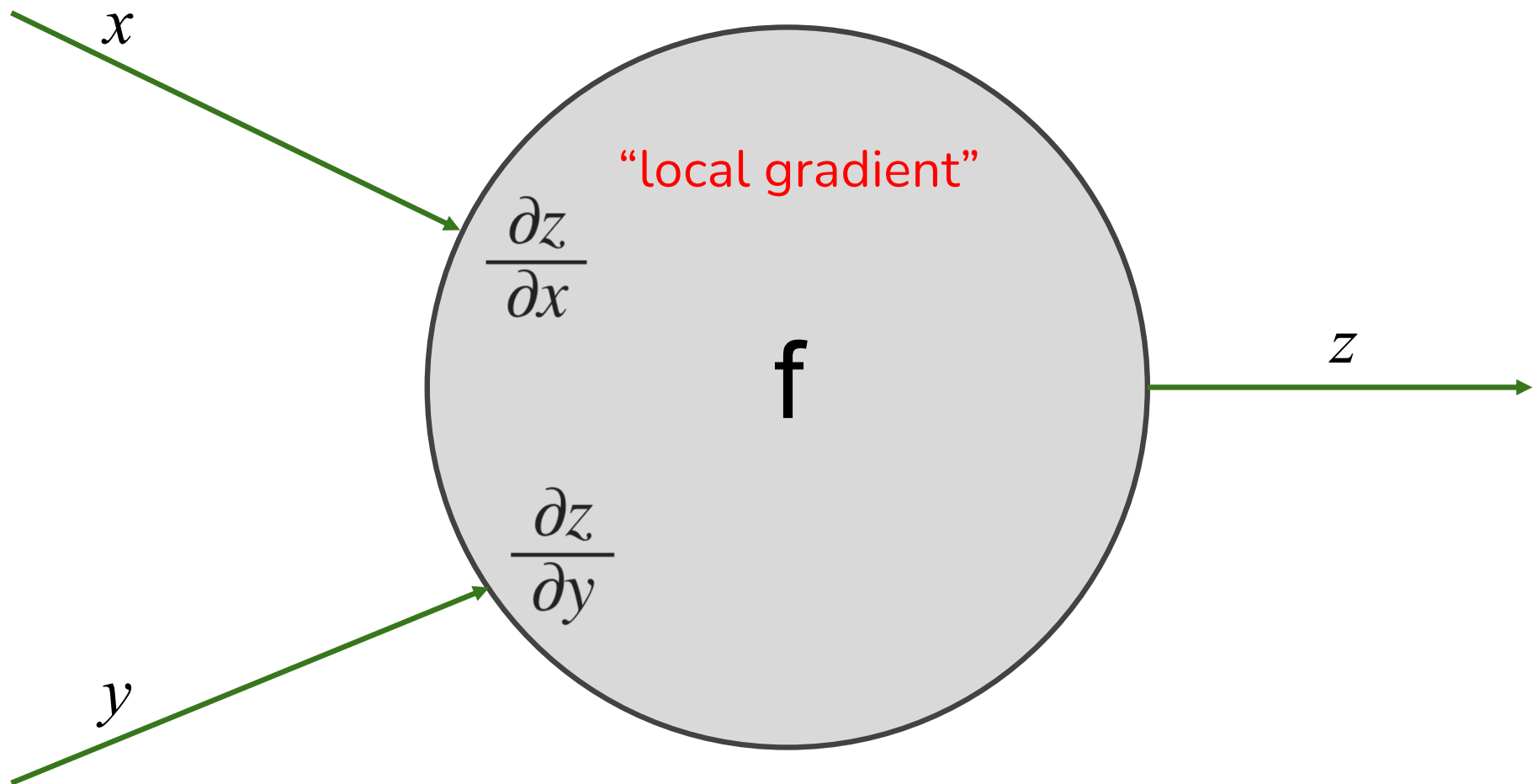
Want: $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial f}{\partial z}$

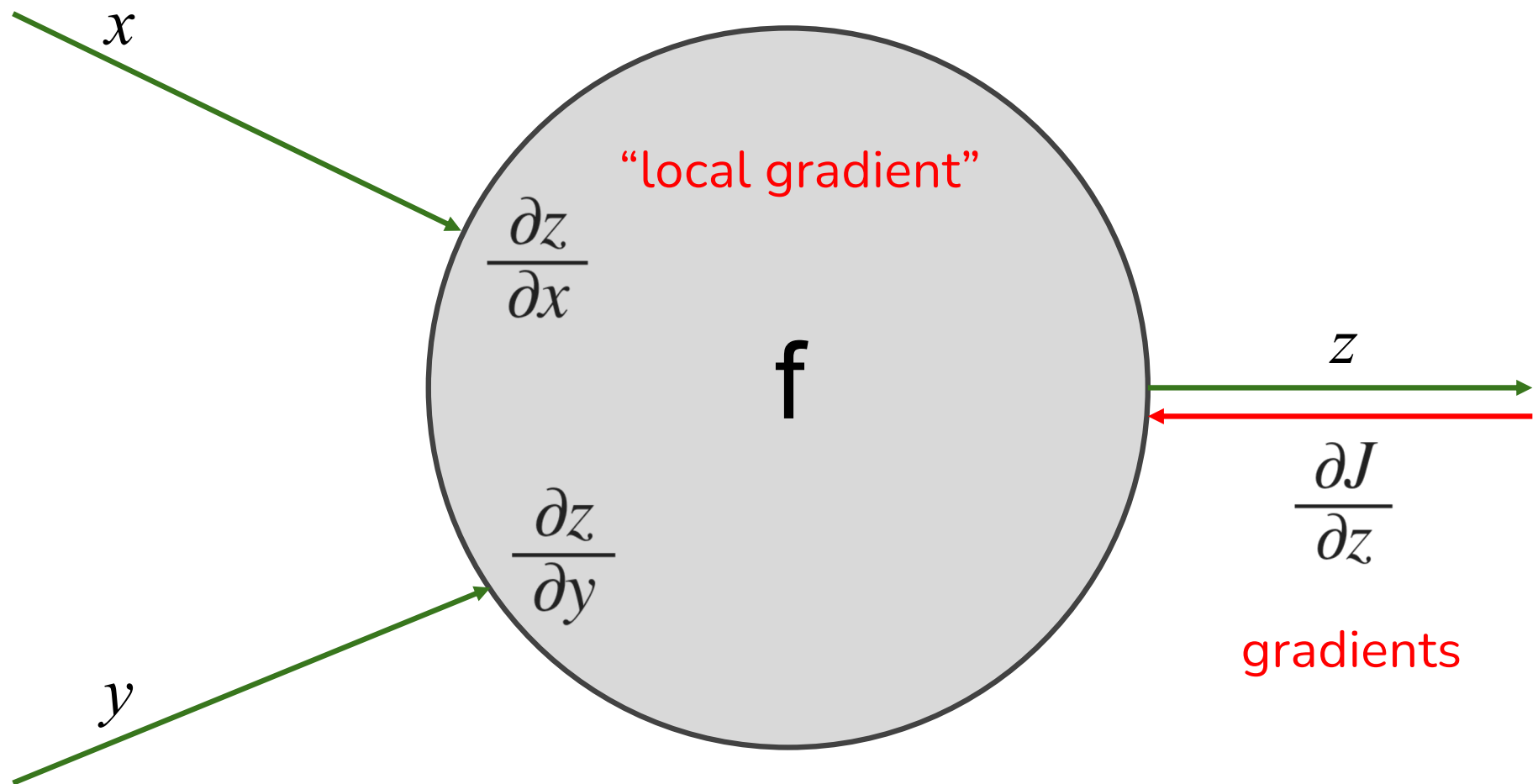
Andrej Karpathy

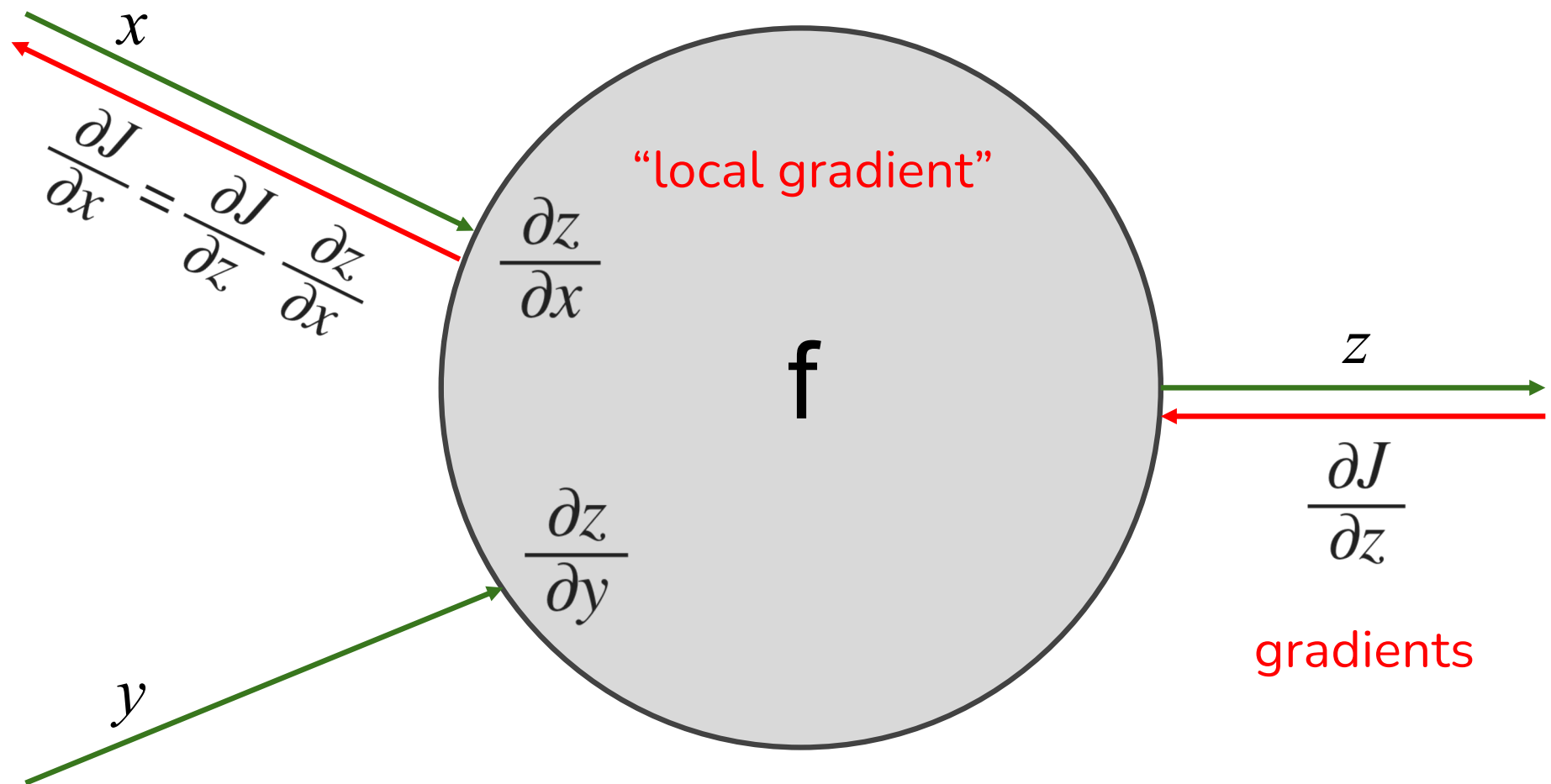


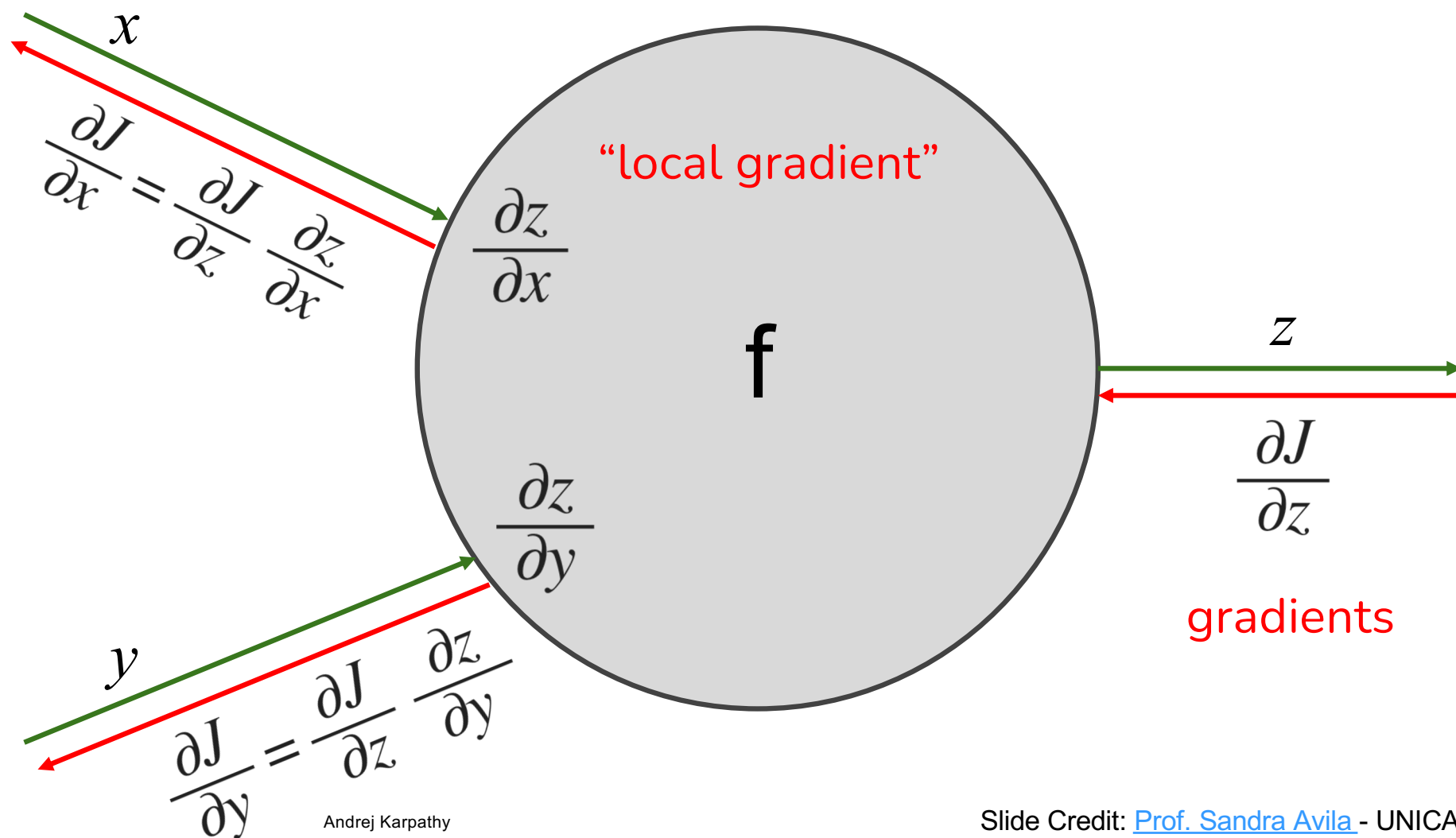
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

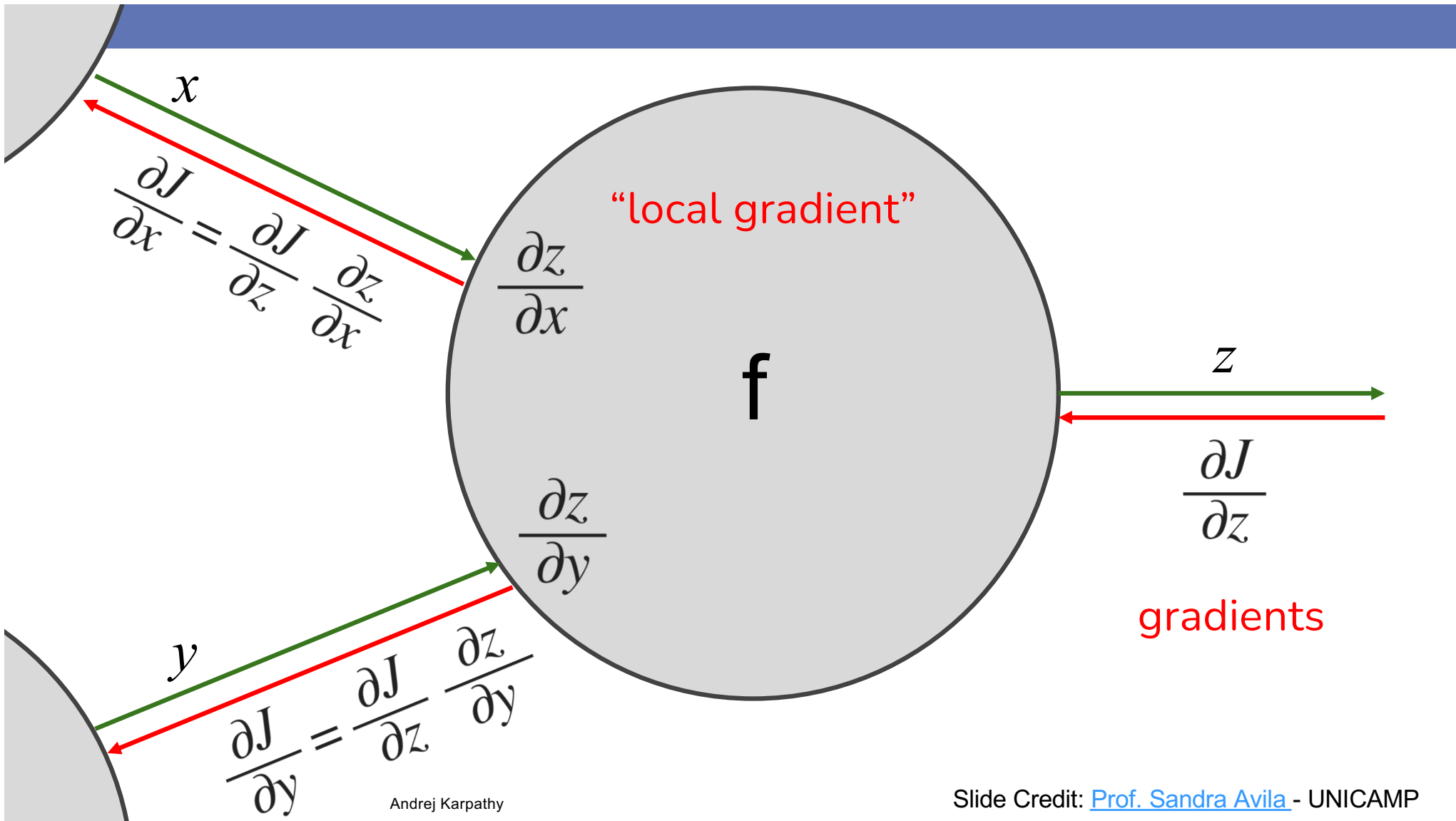






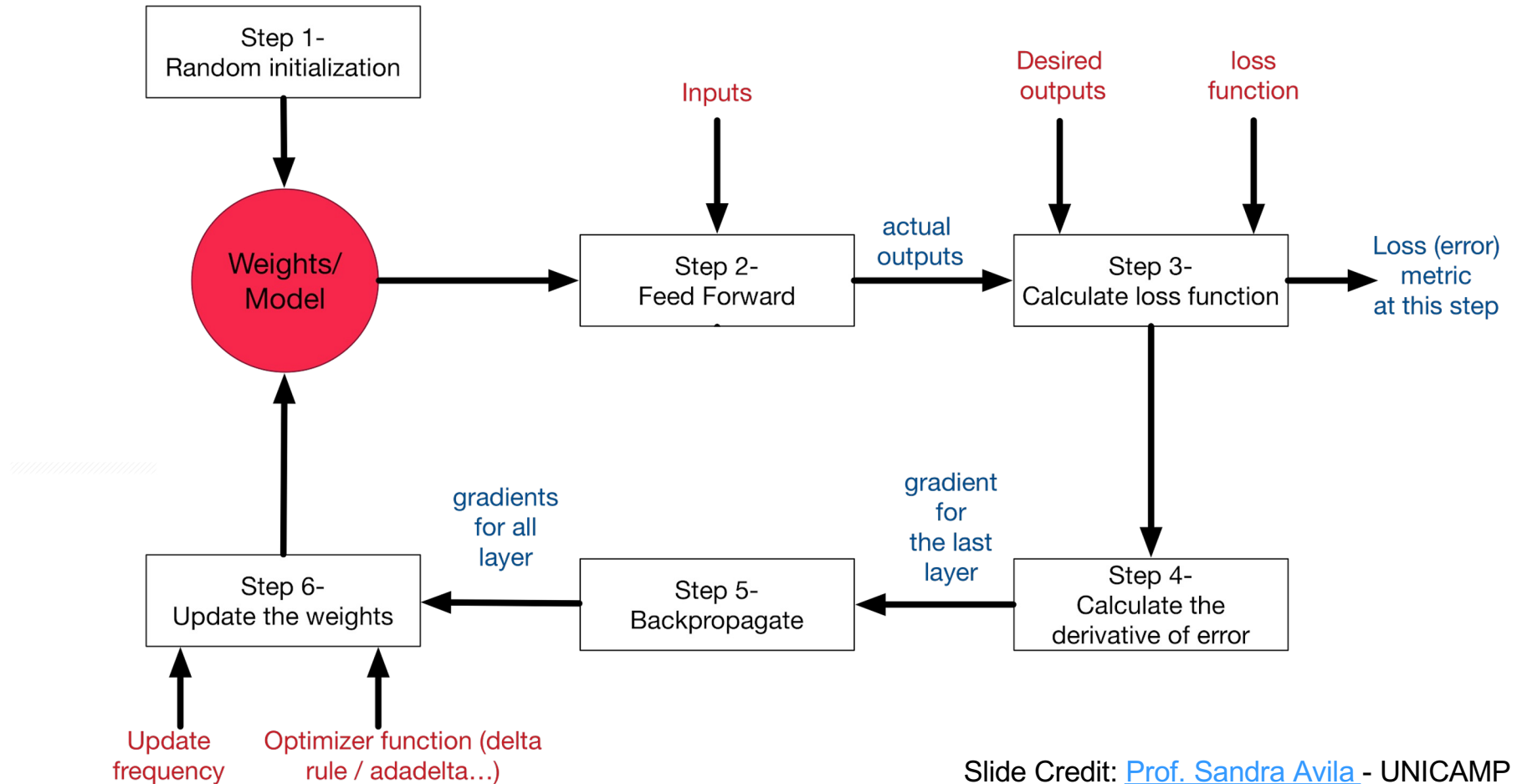






Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Training a Neural Network



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

Gradient Descent

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

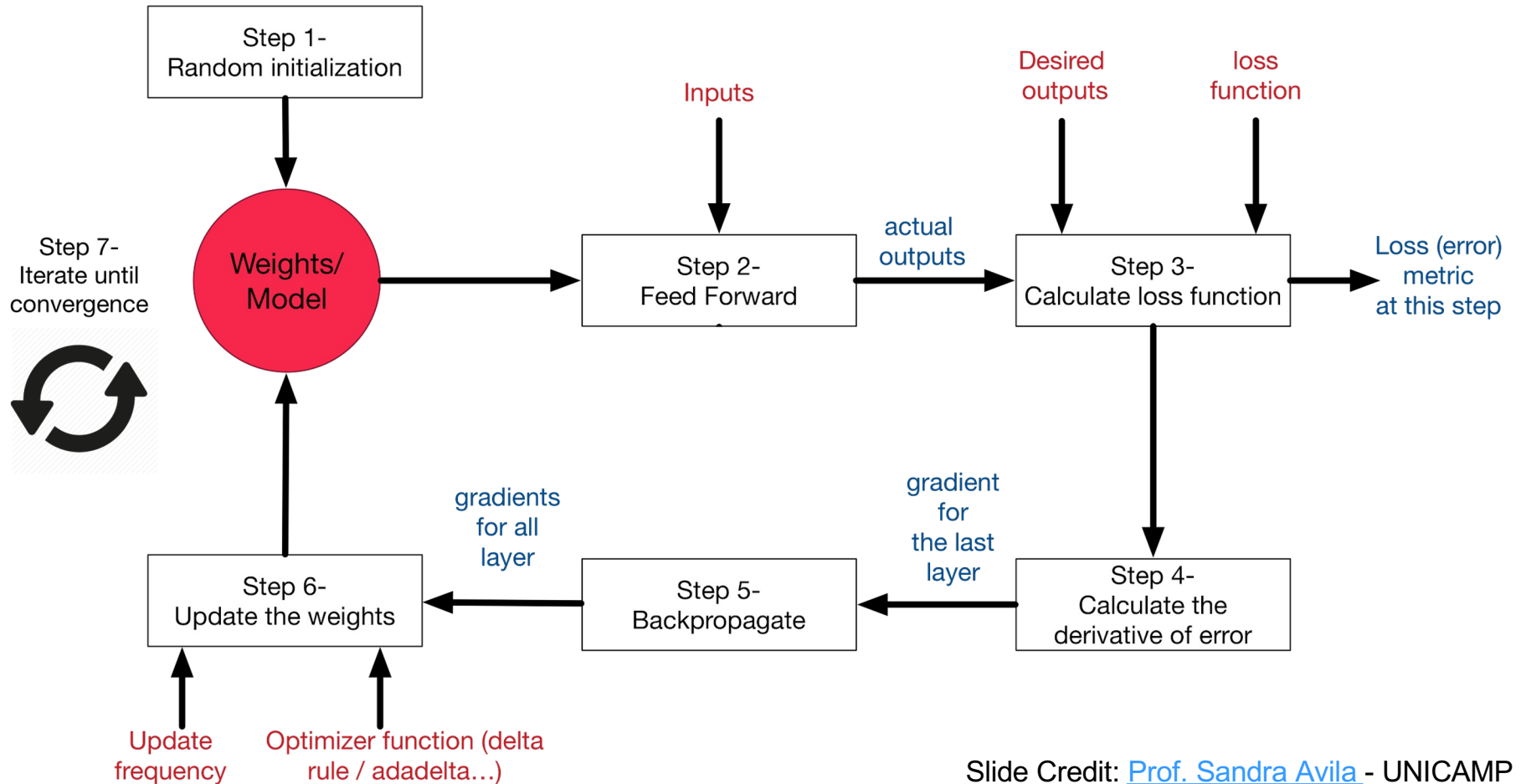
Want $\min_{\Theta} E(\Theta)$:

repeat {

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

}

Training a Neural Network



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

Assignment 3: Data Loaders



Additional Resources

Neural Networks ([3Blue1Brown](#))



Search





Home



Trending



Subscriptions

LIBRARY



History



Watch later



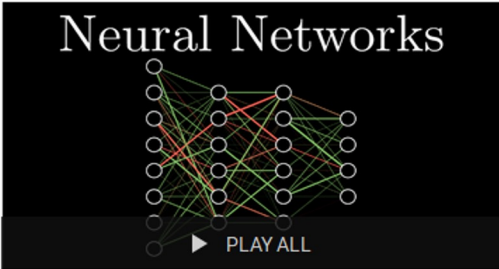
Liked videos



Chansons Franca...



Show more



Neural Networks

PLAY ALL

Neural networks

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



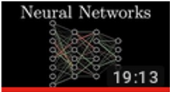
3Blue1Brown

SUBSCRIBED 1.1M



SEASON 3 ▾

1



Neural Networks

19:13

3BLUE1BROWN SERIES S3 • E1

But what *is* a Neural Network? | Deep learning, chapter 1

3Blue1Brown

2



How machines learn

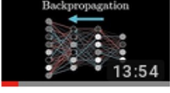
21:01

3BLUE1BROWN SERIES S3 • E2

Gradient descent, how neural networks learn | Deep learning, chapter 2

3Blue1Brown

3



Backpropagation

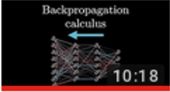
13:54

3BLUE1BROWN SERIES S3 • E3

What is backpropagation really doing? | Deep learning, chapter 3

3Blue1Brown

4



Backpropagation calculus

10:18

3BLUE1BROWN SERIES S3 • E4

Backpropagation calculus | Deep learning, chapter 4

3Blue1Brown

Neural Networks Demystified (in Python)

⋮

🏠 📺 🔔

Home

Trending

Subscriptions

LIBRARY

History

Watch later

Liked videos

SUBSCRIPTIONS

▶ PLAY ALL

Neural Networks Demystified

7 videos • 197,878 views • Last updated on Oct 2, 2015

Welch Labs

SUBSCRIBE 101K

- 1

Neural Networks Demystified [Part 1: Data and Architecture]

Welch Labs

3:08
- 2

Neural Networks Demystified [Part 2: Forward Propagation]

Welch Labs

4:28
- 3

Neural Networks Demystified [Part 3: Gradient Descent]

Welch Labs

6:56
- 4

Neural Networks Demystified [Part 4: Backpropagation]

Welch Labs

7:56
- 5

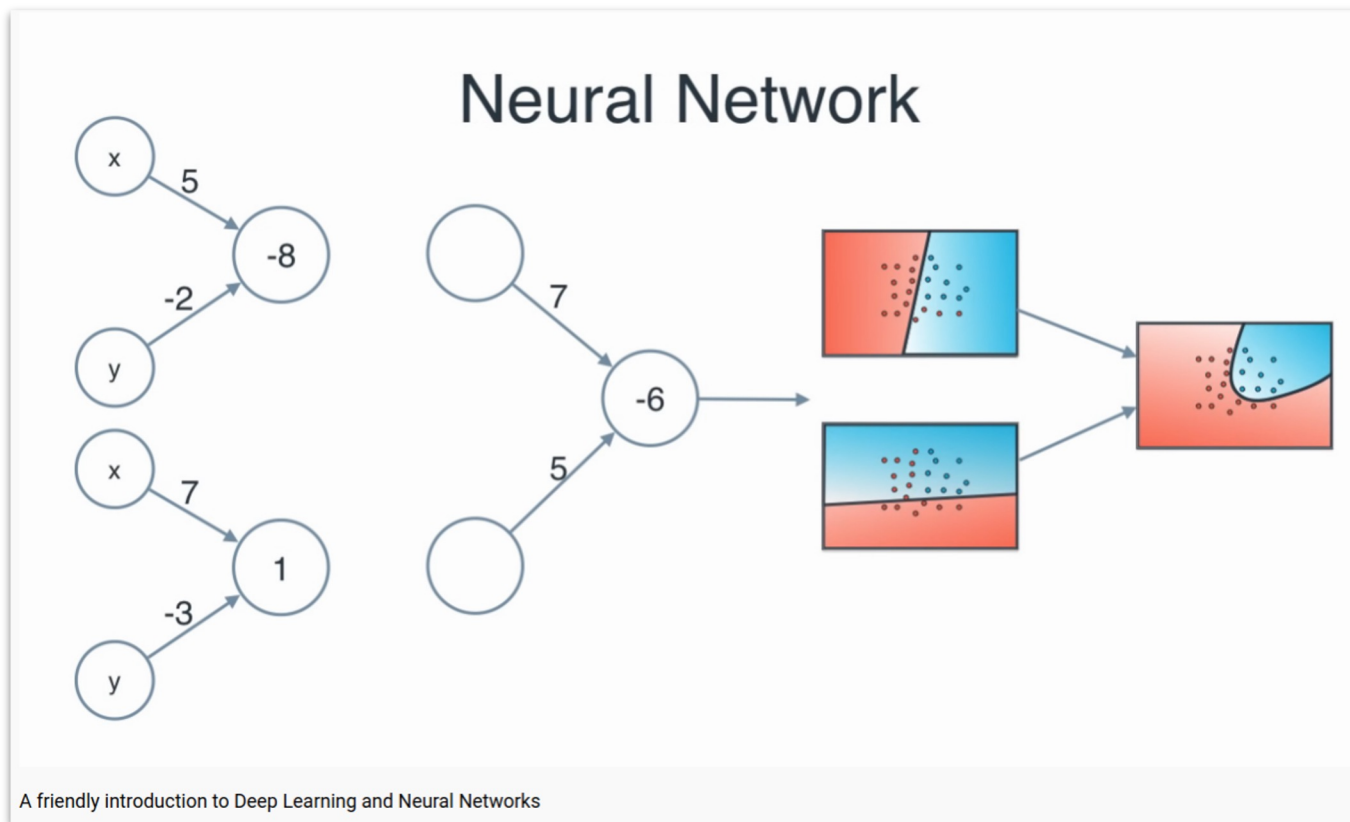
Neural Networks Demystified [Part 5: Numerical Gradient Checking]

Welch Labs

4:14

“A friendly introduction to Neural Networks”

<https://youtu.be/BR9h47Jtqyw>



Thumbnail Classification - Rede X

Thumbnail Classification - Rede X

+

← → ↺ 🏠

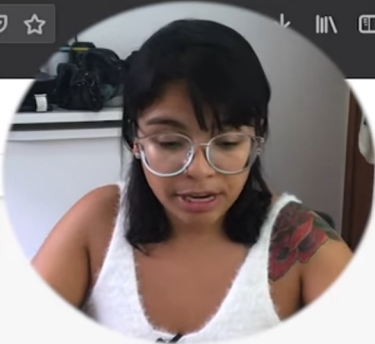
🔒 <https://colab.research.google.com/drive/1DbO93KEYfA6Fvq6ugY8>

90% ... 📄 ⌵

Introdução

 Thumbnail Classification - Redes Neurais com PyTorch.ipynb ☆

00;00;12;19



+ Code + Text

```
[ ]
25 train_size = int(0.75*len(data))
26 idx = torch.randperm(len(data))
27 train_sampler = SubsetRandomSampler(idx[0:train_size])
28 test_sampler = SubsetRandomSampler(idx[train_size:])
29
30 train_loader = DataLoader(data, sampler=train_sampler,
31                             batch_size=10, num_workers=4)
32
33 test_loader = DataLoader(data, sampler=test_sampler,
34                             batch_size=10, num_workers=4)
```

▼ Rede Neural em PyTorch

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

Imports de biblioteca e verificação de dispositivo de hardware.



```
1 import torch
2
```

Code

Text

 cuda

Implementando a arquitetura

```
[ ] 1
```

Redes Neurais em PyTorch | Programando em 10 minutos #1

17,671 views • Feb 19, 2020

 1.7K  19  SHARE  SAVE ...

 **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