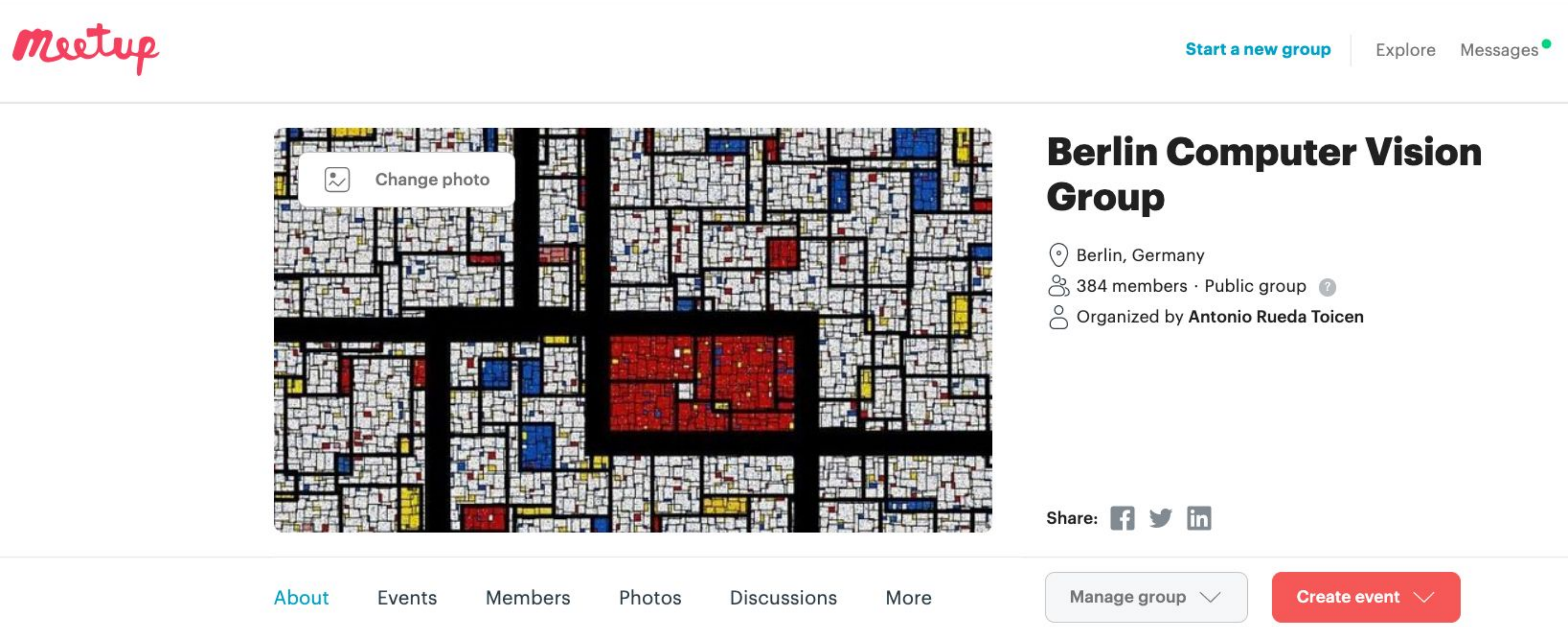
Backpropagation and Gradient Descent in Neural Networks

Antonio Rueda-Toicen
Data Science Retreat
April 2023

About me

- Senior Data Scientist at Vinted
- Background in academia (computer science and bioengineering)
  - Organizer of the Berlin Computer Vision Group
  - Arepa-lover (try them, they’re *awesome*)

Fig 1: Arepas



https://www.meetup.com/Berlin-Computer-Vision-Group/

Agenda

- Intro to deep learning
  - DL in the machine learning project-cycle
  - What is DL and why should we use it?
    - Neural networks
    - Perceptrons
    - The feedforward pass
    - Backpropagation
    - Activation functions
    - Stochastic gradient descent
  - Our goal: build a image classifier for digits, first in Numpy, then in PyTorch

Knowing the jargon

Model aka network aka architecture (although a fine-grained distinction exists here with respect to training)

Weight aka parameter aka connection strength

Hyperparameter

Capacity

Loss aka cost aka error function

Error rate aka 1 - accuracy

Sensitivity aka recall

Activation function

Transfer learning

Epoch vs Batch

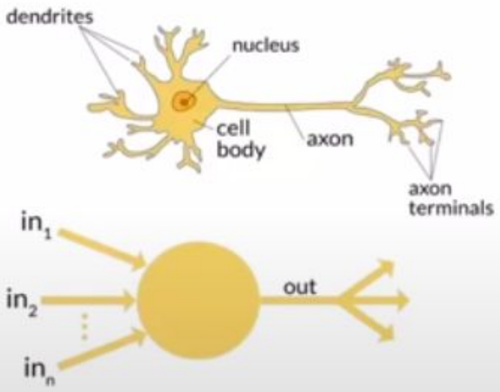
Convolution, receptive field

Linear aka “Dense” layer

What is an artificial neural network?

In 1943 Warren McCulloch, a neurophysiologist, and Walter Pitts, a logician, teamed up to develop a mathematical model of an artificial neuron. They declared that:

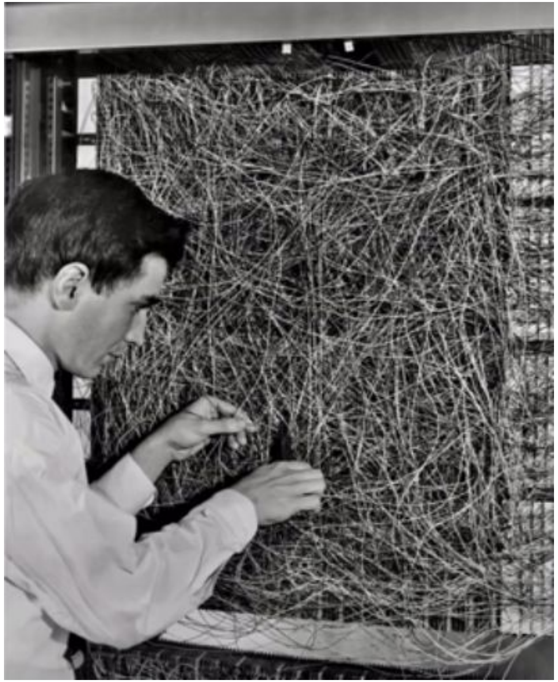
> *Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms.* (Pitts and McCulloch; A Logical Calculus of the Ideas Immanent in Nervous Activity)



Artificial "neurons" are also called "processing units"

Recommended read: "[The Man Who Tried to Redeem the World with Logic]"

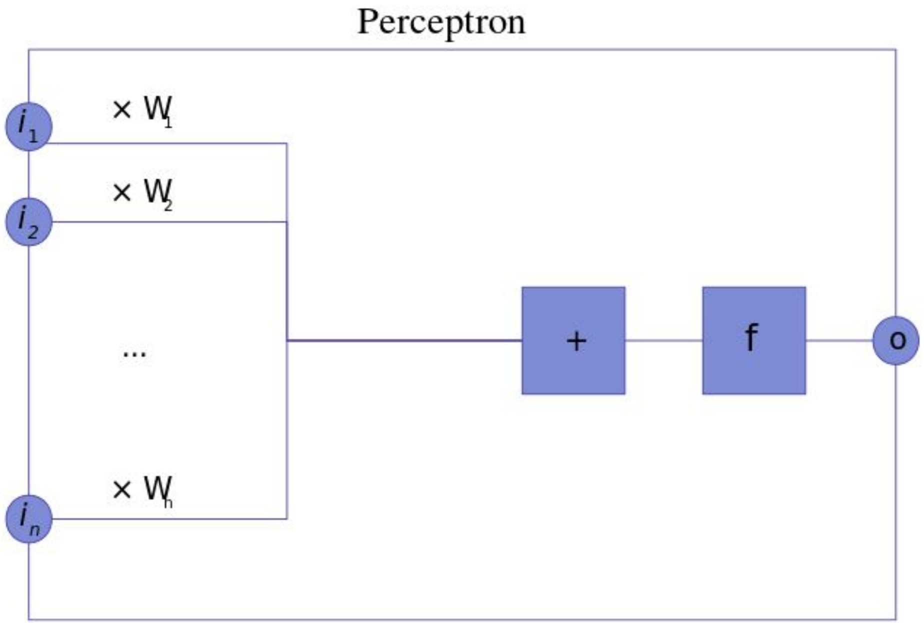
What is an artificial neural network?



"we are about to witness the birth of such a machine – a machine capable of perceiving, recognizing and identifying its surroundings without any human training or control".
Frank Rosenblatt

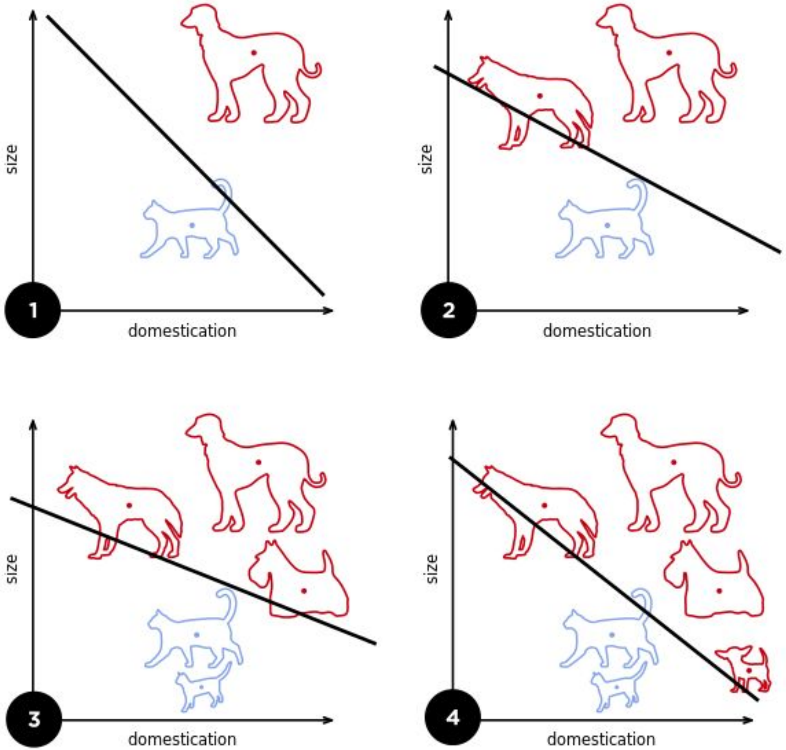
Image: Frank Rosenblatt with the Mark I Perceptron at Cornell University (1961)

What is a perceptron?



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

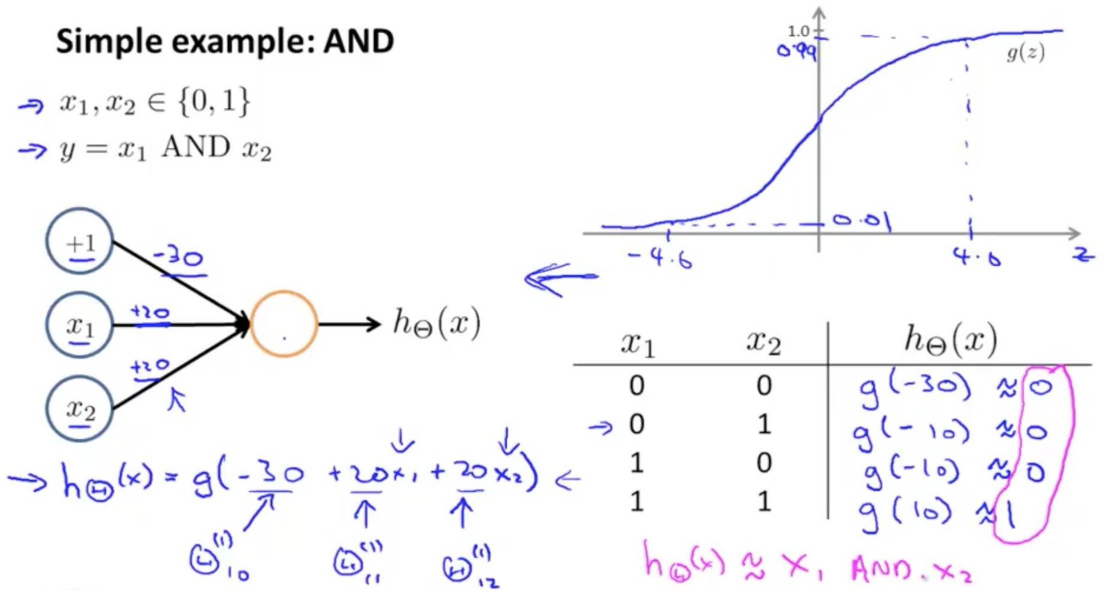
What is a perceptron?



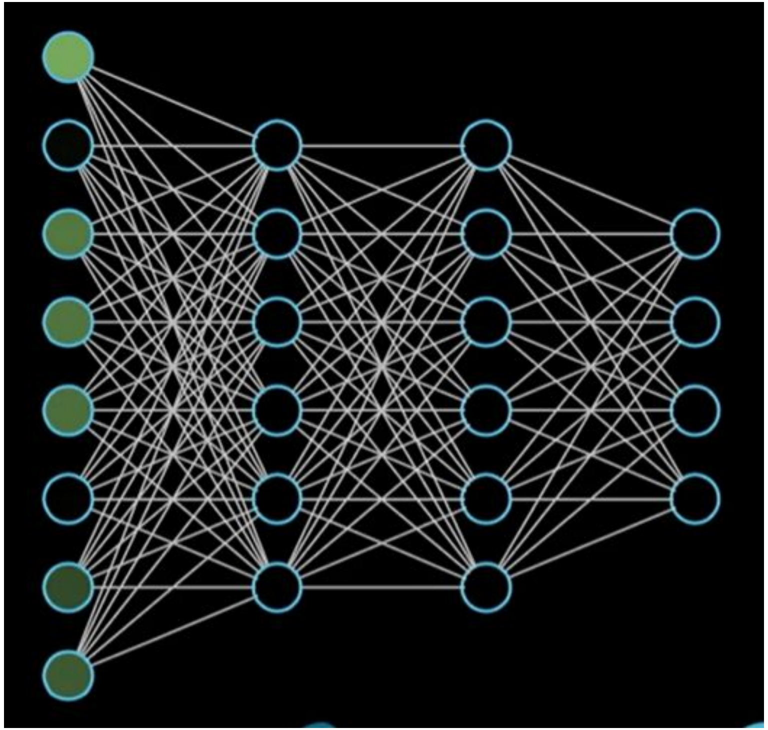
A fine-grained discussion about the differences in logistic regression and the perceptron algorithm

What does a neural network ‘learn’?

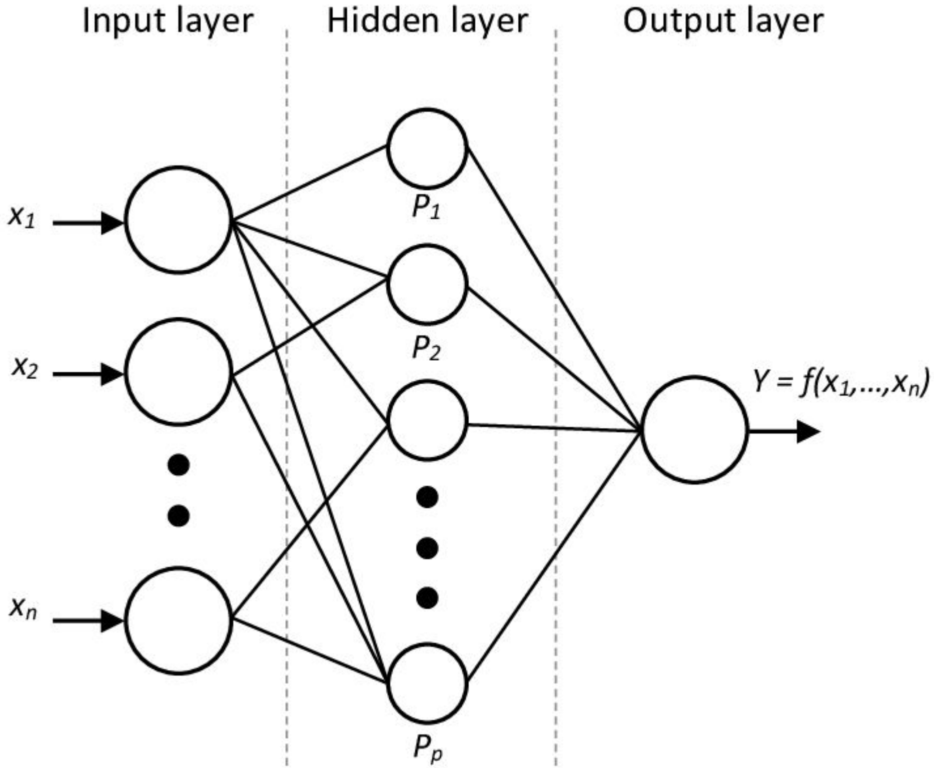
It learns “*weights*” aka “*parameters*” aka “*connection strengths*” that minimize a measure of error (aka “loss” aka “cost”) function given a set of training inputs x (aka “a dataset”)



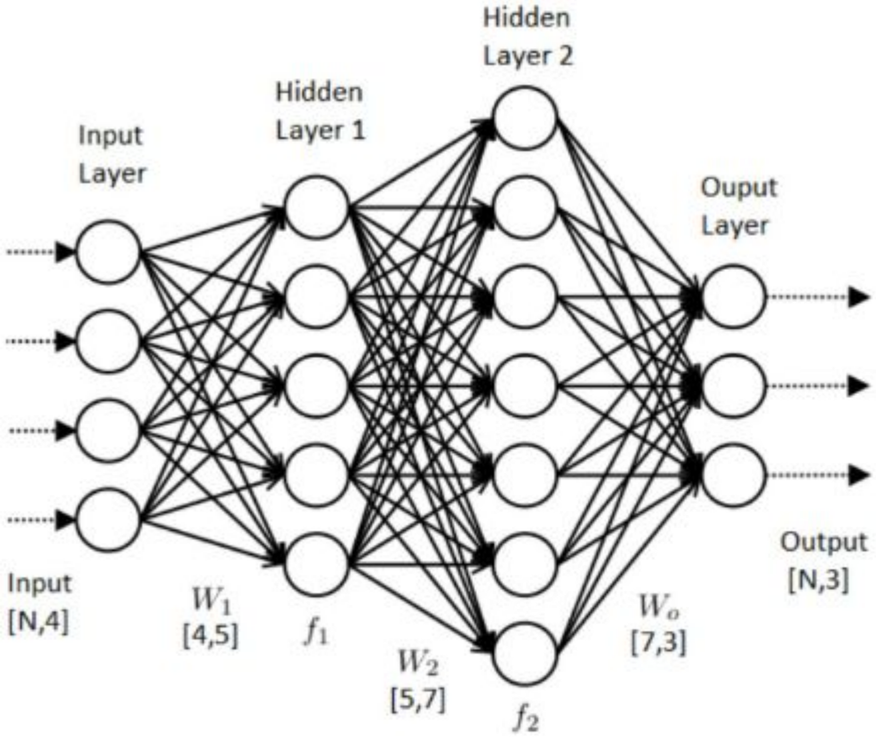
Multilayer perceptrons



Hidden layers

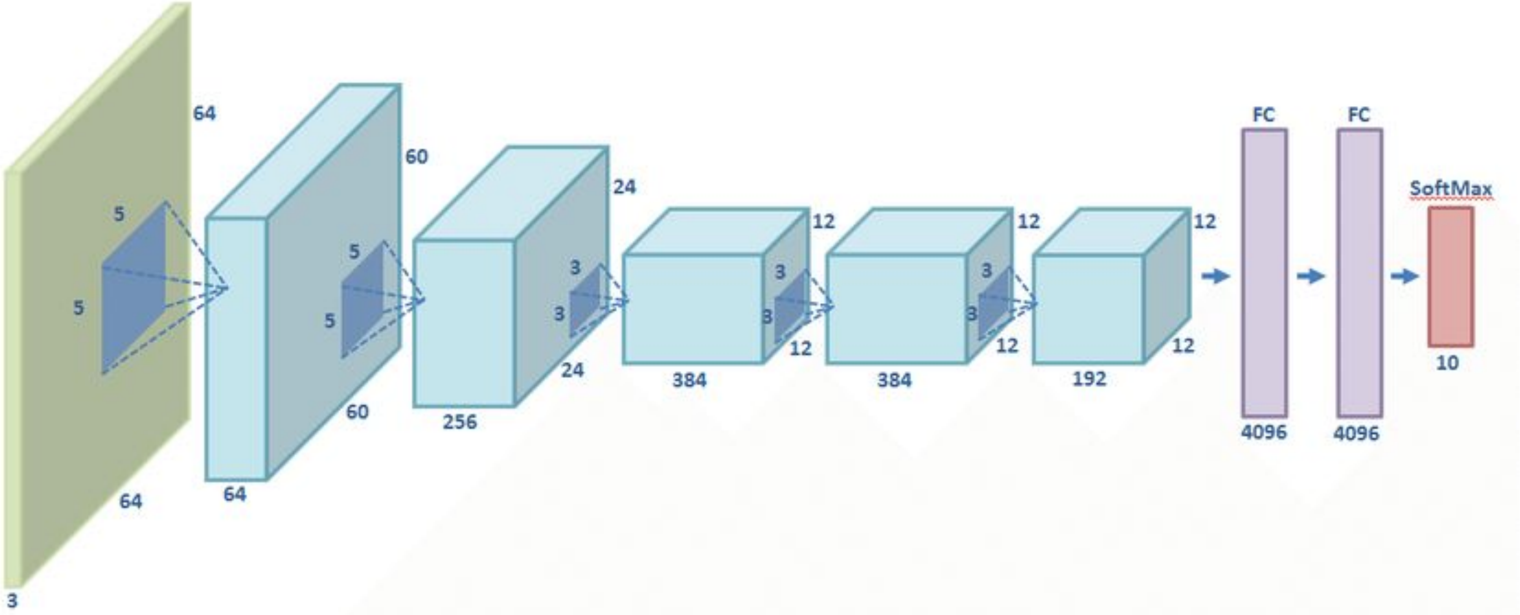


Hidden layers



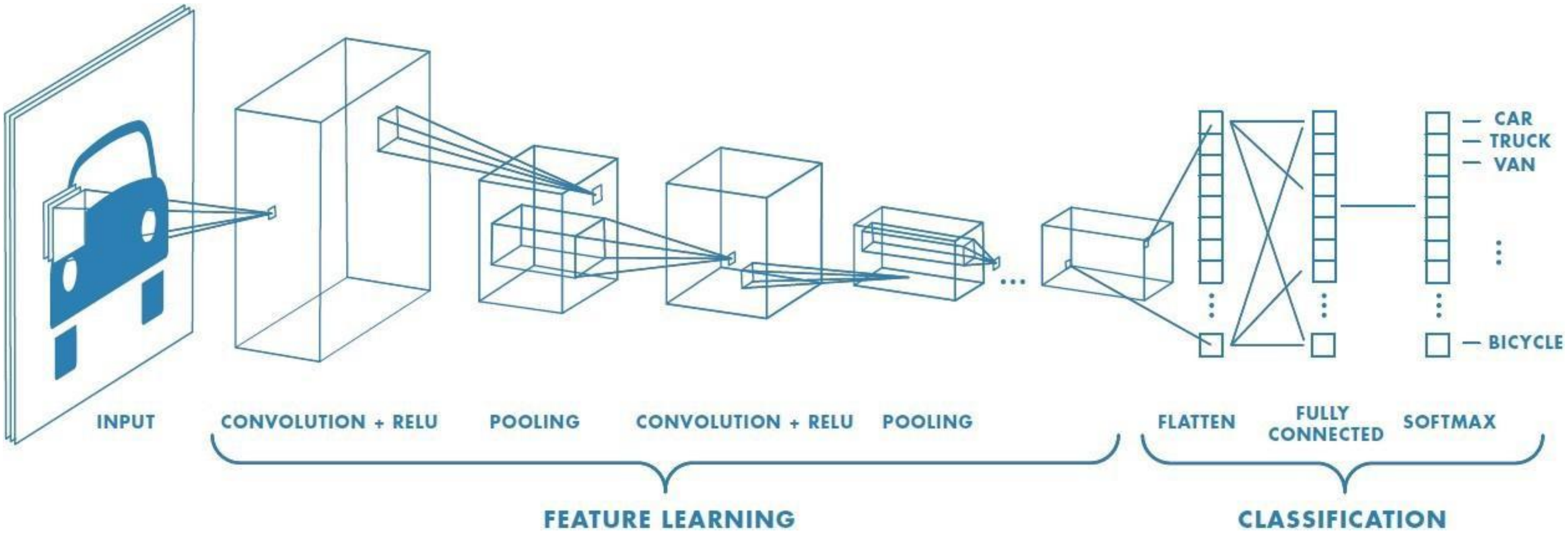
Stir the pile

Different network architectures represent different ways to 'stir the pile'



Architecture of the Alexnet network

What is deep learning?



Visualizing a CNN

Convolutional Neural Network Visualization by Otavio Good

What is a GPU?

Most deep learning libraries require the use of NVIDIA graphical processing units with CUDA capabilities

Why use a GPU for deep learning?

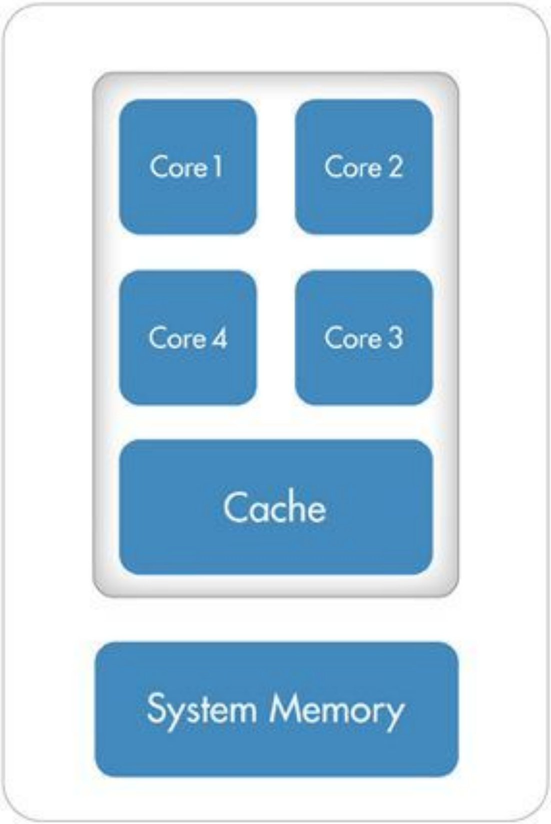
How is a CPU different from a GPU?

- More cores!

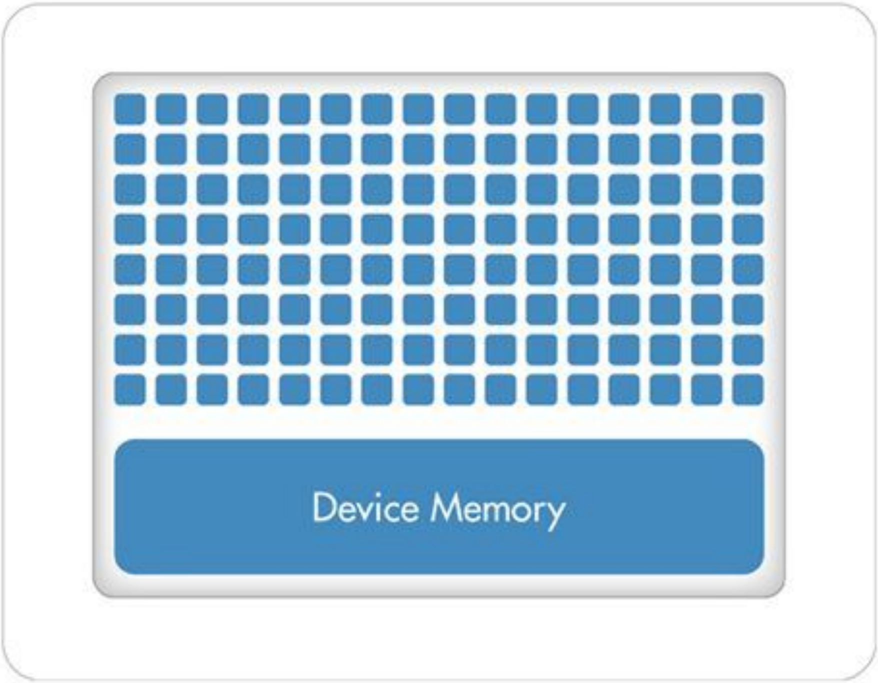
Why are CPUs less effective for deep learning?

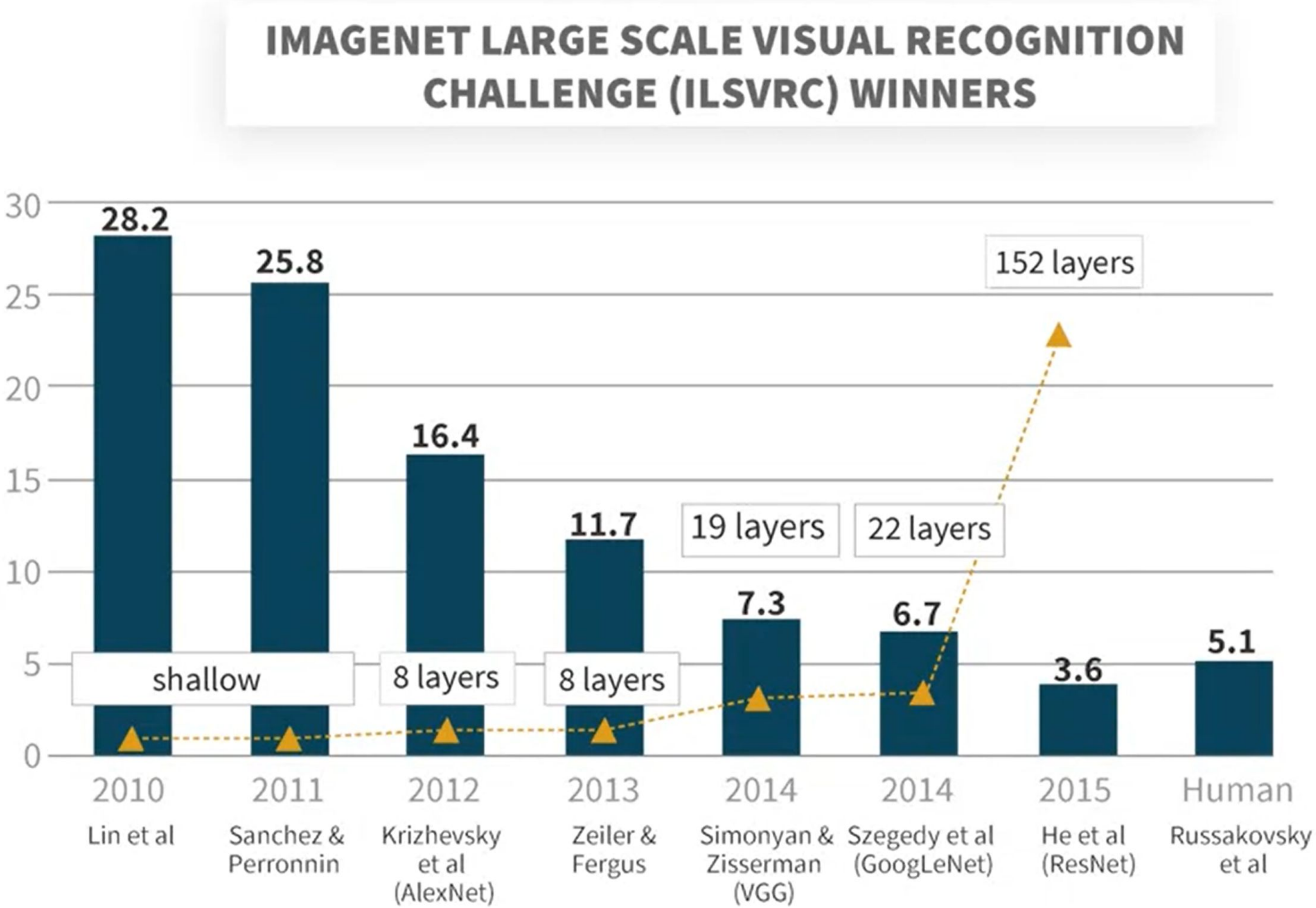
- Less cores :(

CPU (Multiple Cores)



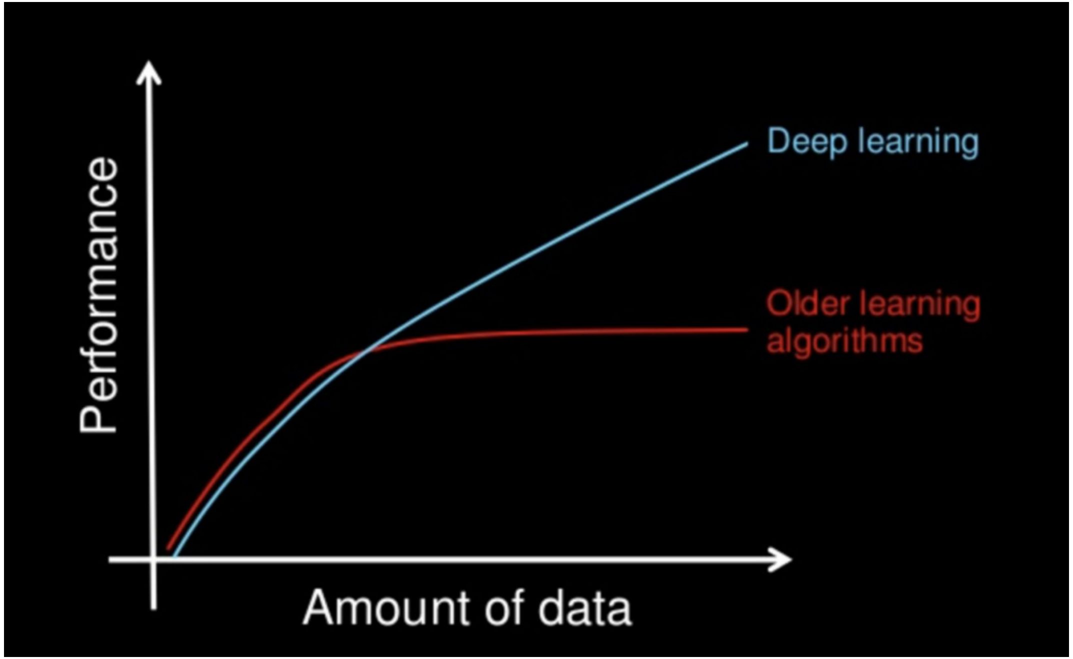
GPU (Hundreds of Cores)





IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC) WINNERS
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
Human
Russakovsky et al

Why do we use deep learning?



- Disclaimer: the [No Free Lunch Theorems] explain why this graph might be hype, although things like the [current GPT experiments] show promise

Applications of deep learning

- Computer vision: image classification, object detection, segmentation, image description, image generation, satellite and drone image analysis

- Natural Language Processing: translation, text and DNA sequence prediction, text generation

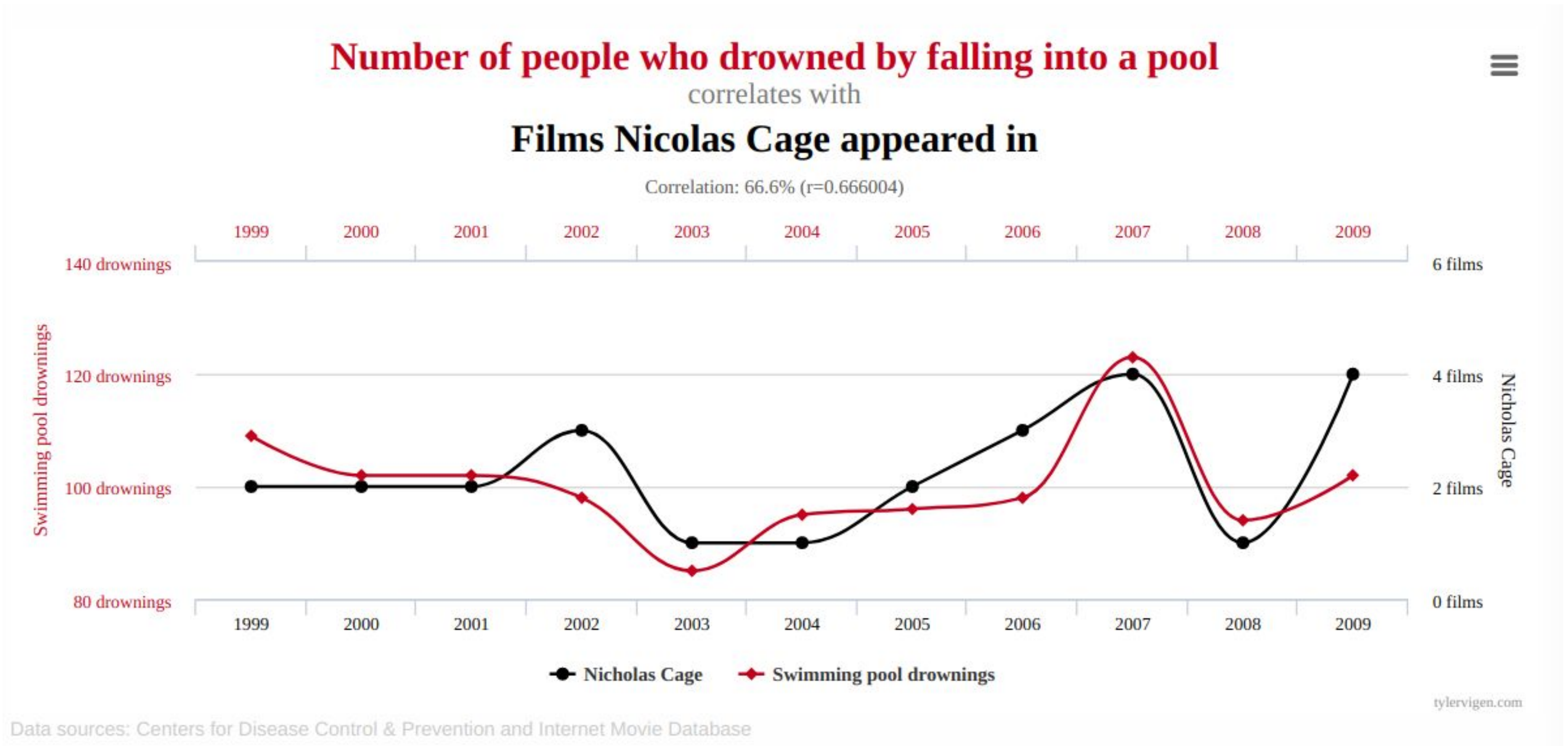
- Audio understanding, audio generation, lip reading

- Playing games, finding optimal strategies (when paired with reinforcement learning)

- Any system that requires the finding of **significant correlations** or patterns/”intuitions” in the data, i.e. the ‘thinking fast’ systems referred by Daniel Kahneman in his famous book.

Why do we use deep learning?

- Deep neural networks are ***universal function approximators***
    - *This means that given the right architecture and training, they can **compute everything that's computable**, just like a Universal Turing Machine*
    - *In practice this means that deep networks are **very good** at finding very sophisticated decision boundaries (aka "curve fitting")*
    - *Finding the function means the finding **architecture and or training** of the neural network*
    - *It's **fair** to call deep neural networks "**linear regression on steroids**"*
    - *Deep neural networks pick hidden **correlations** in the data, but most architectures don't pick on **causality** (current research topic)*



Number of people who drowned by falling into a pool
correlates with
Films Nicolas Cage appeared in
Correlation: 66.6% (r=0.666004)
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
140 drownings
120 drownings
100 drownings
80 drownings
Swimming pool drownings
6 films
4 films
2 films
0 films
Nicholas Cage
Nicholas Cage
Swimming pool drownings
tylervigen.com
Data sources: Centers for Disease Control & Prevention and Internet Movie Database

Choose the right loss function for the task

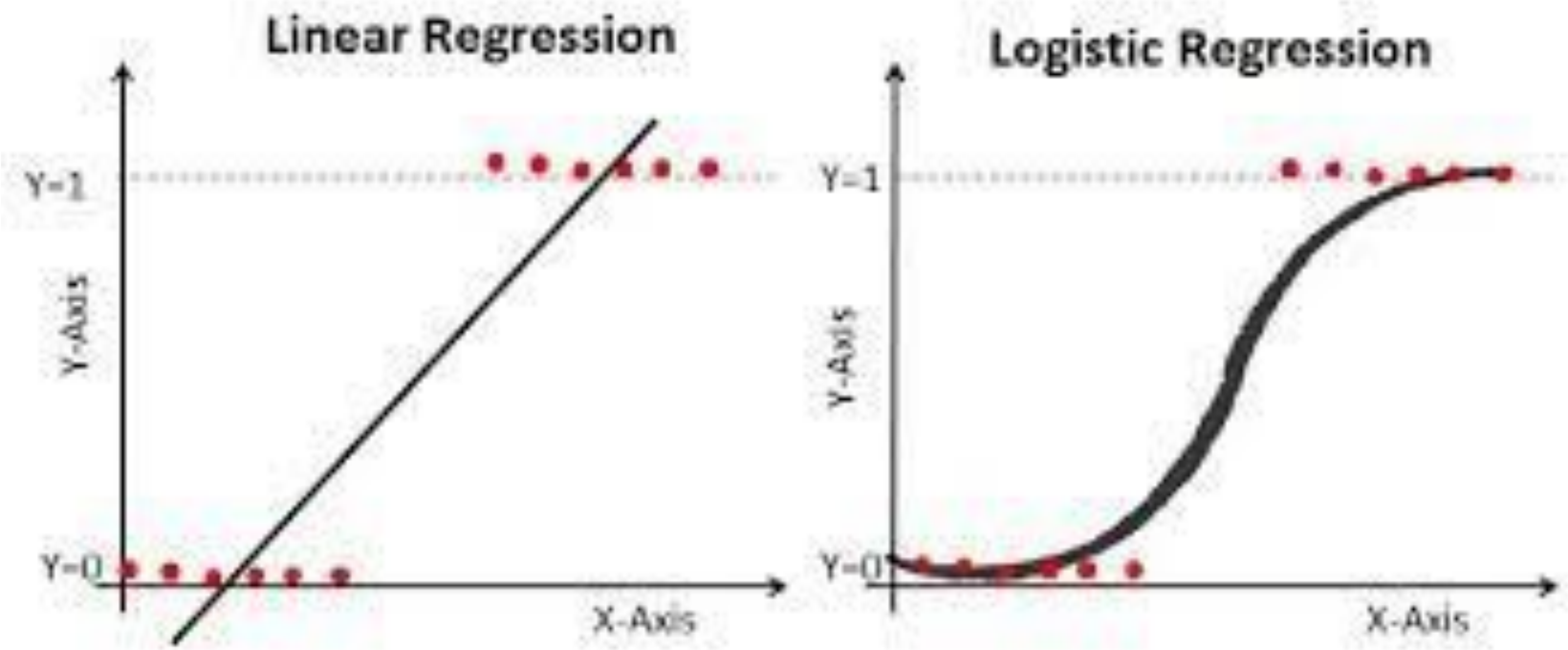
$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Mean Squared Error is a loss function for regression

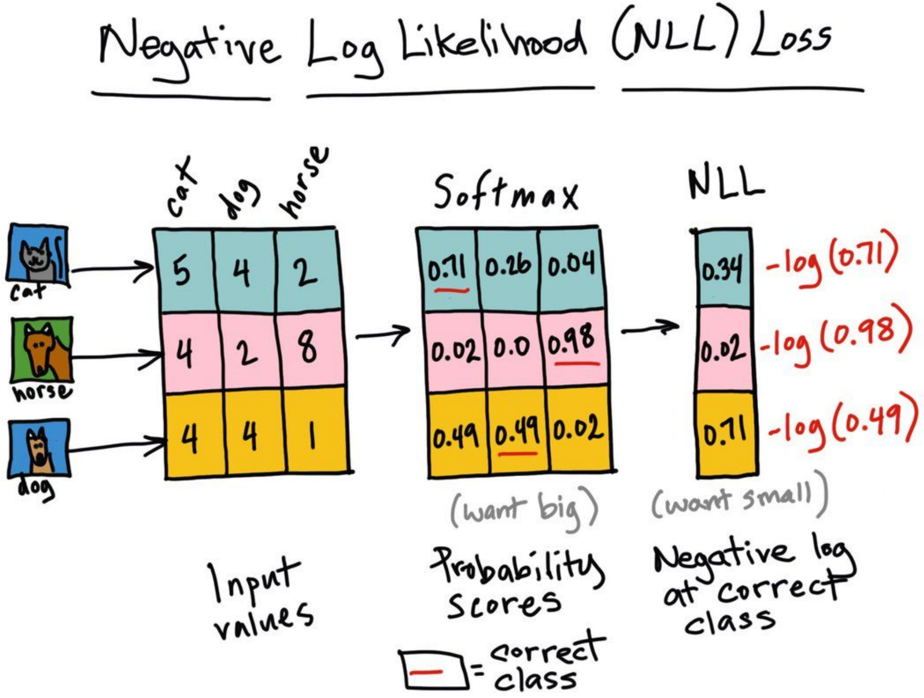
Choose the right loss function for the task

$$\text{Loss} = -\frac{1}{\substack{\text{output}\\\text{size}}} \sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Cross entropy is a function for classification, used interchangeably with negative log-likelihood, here we see it in its binary case


Linear Regression
Y=1
Y-Axis
Y=0
X-Axis
Logistic Regression
Y=1
Y-Axis
Y=0
X-Axis

Can we solve this task with logistic regression?



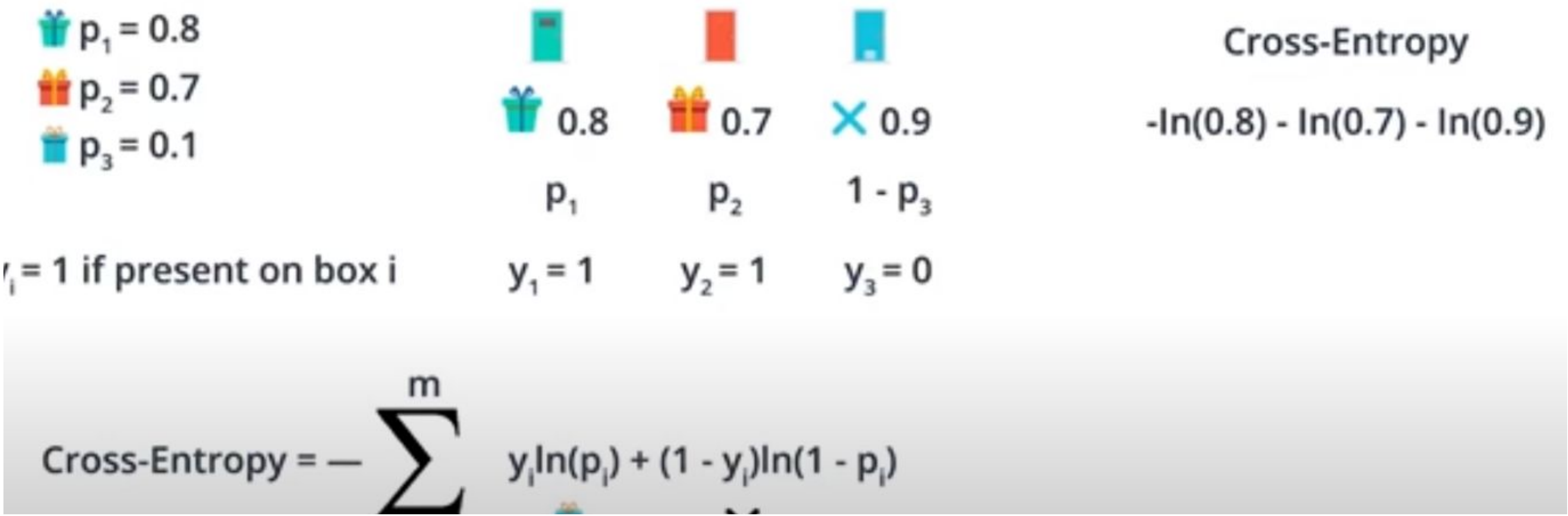
Q: Which loss function should we use for binary logistic regression?

A. Mean Squared Error
B. Cross Entropy
C. Negative log-likelihood (huh? Maybe read this too later.)
D. Precision & Recall
E. None of the above

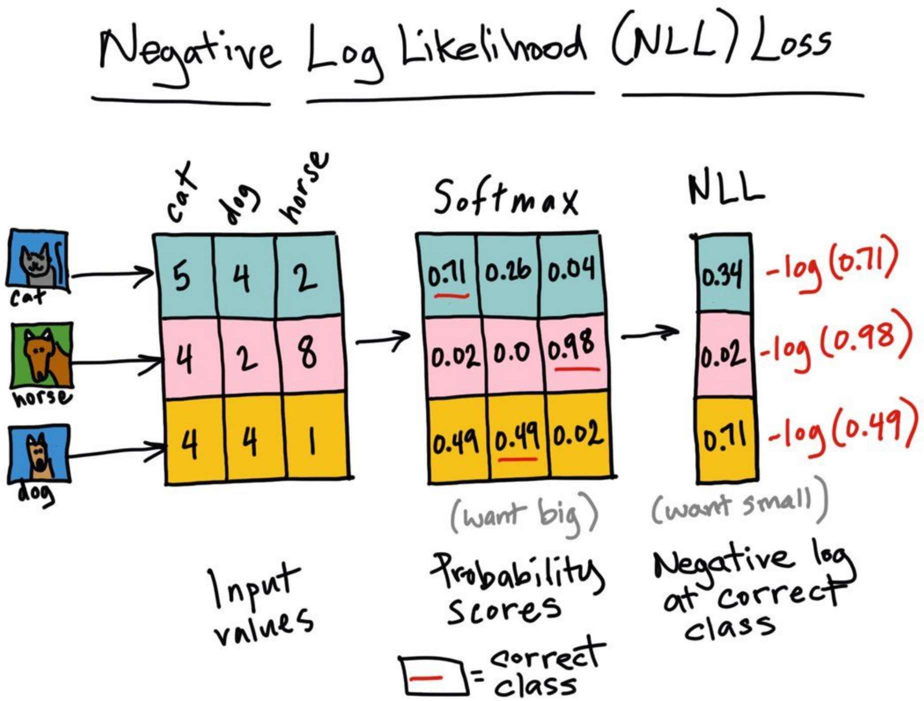
Cross Entropy

$$H(p,q) = -\sum_{x\in\mathcal{X}} p(x)\, \log q(x)$$

Cross entropy

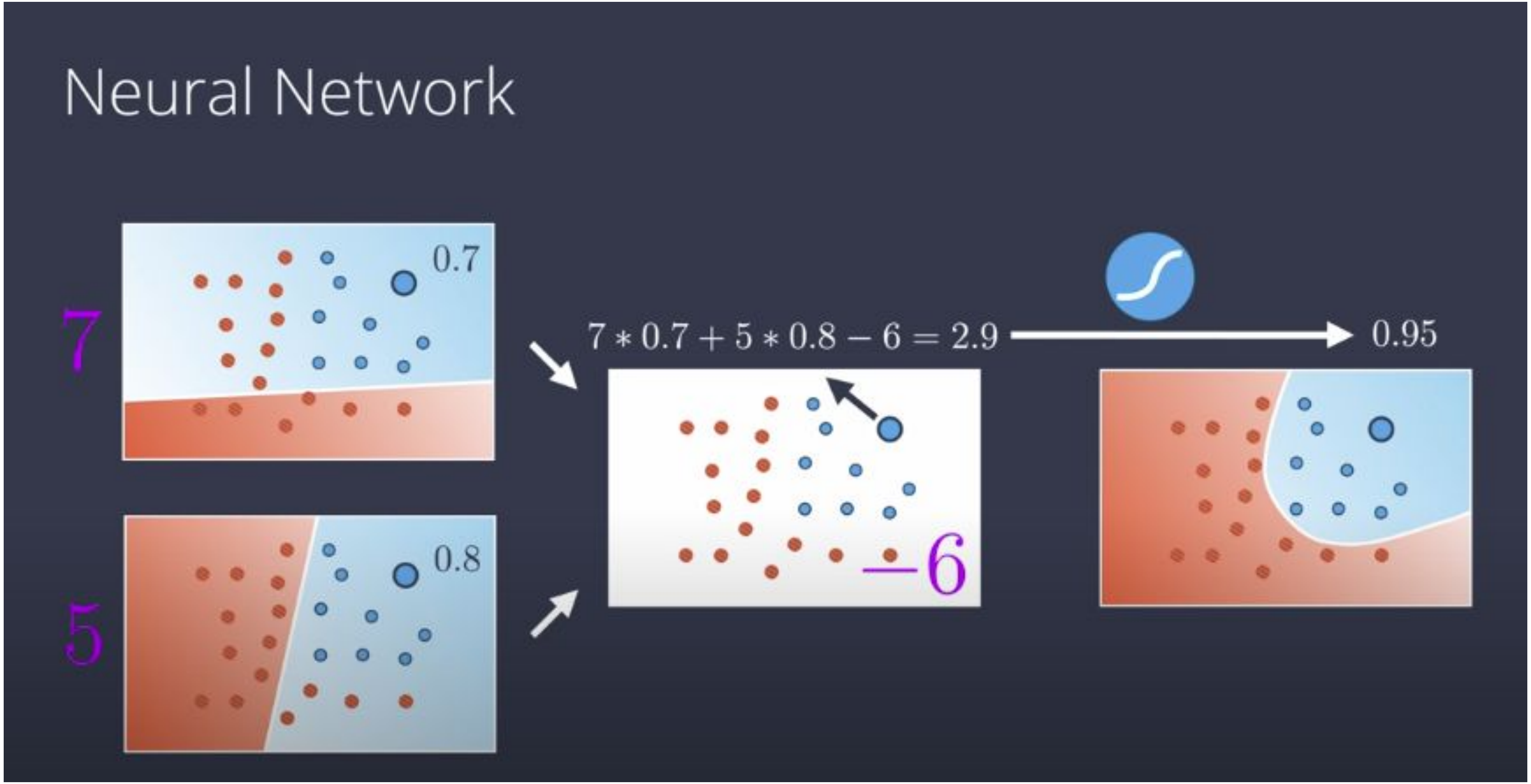


[Explore the Colab notebook]



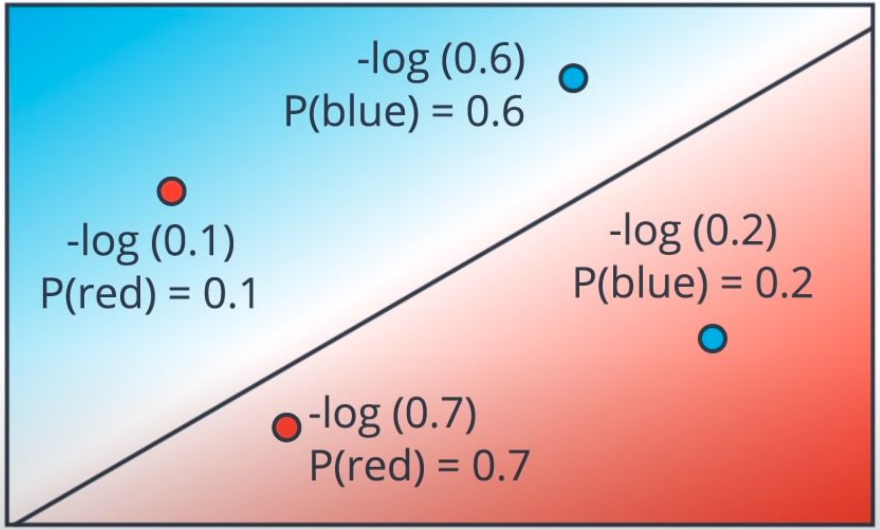
[Exercise](#)
[Solution](#)

Refer to [this tutorial](#) and [this discussion](#)



Neural Network
7
0.7
5
0.8
7 * 0.7 + 5 * 0.8 − 6 = 2.9
0.95
−6

Error Function



$$-\log(0.6) - \log(0.2) - \log(0.1) - \log(0.7) = 4.8$$

0.51 1.61 2.3 0.36

If $y = 1$

$P(\text{blue}) = \hat{y}$

$\text{Error} = -\ln(\hat{y})$

If $y = 0$

$P(\text{red}) = 1 - P(\text{blue}) = 1 - \hat{y}$

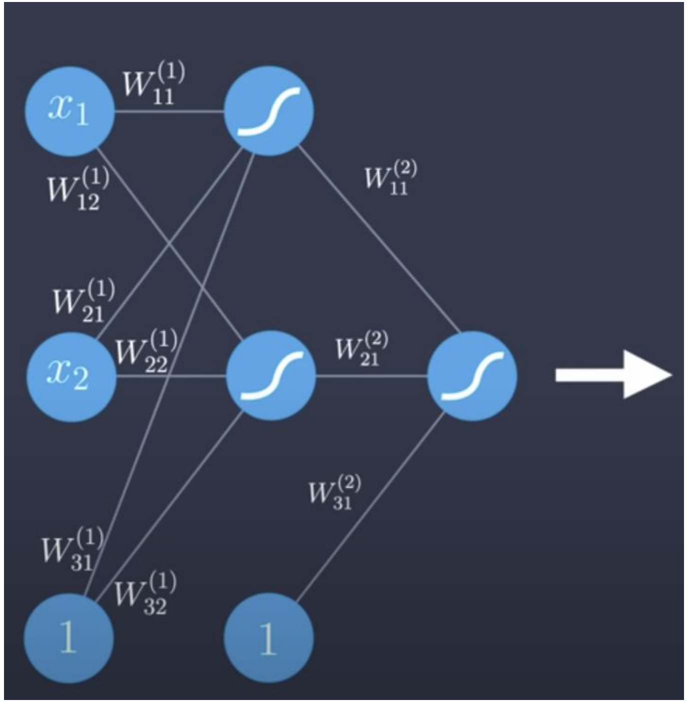
$\text{Error} = -\ln(1 - \hat{y})$

$$\text{Error} = -(1-y)(\ln(1-\hat{y})) - y\ln(\hat{y})$$

$$\text{Error Function} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)(\ln(1-\hat{y}_i)) + y_i\ln(\hat{y}_i)$$

Q: Which of these points have higher cross entropy?

The feedforward pass

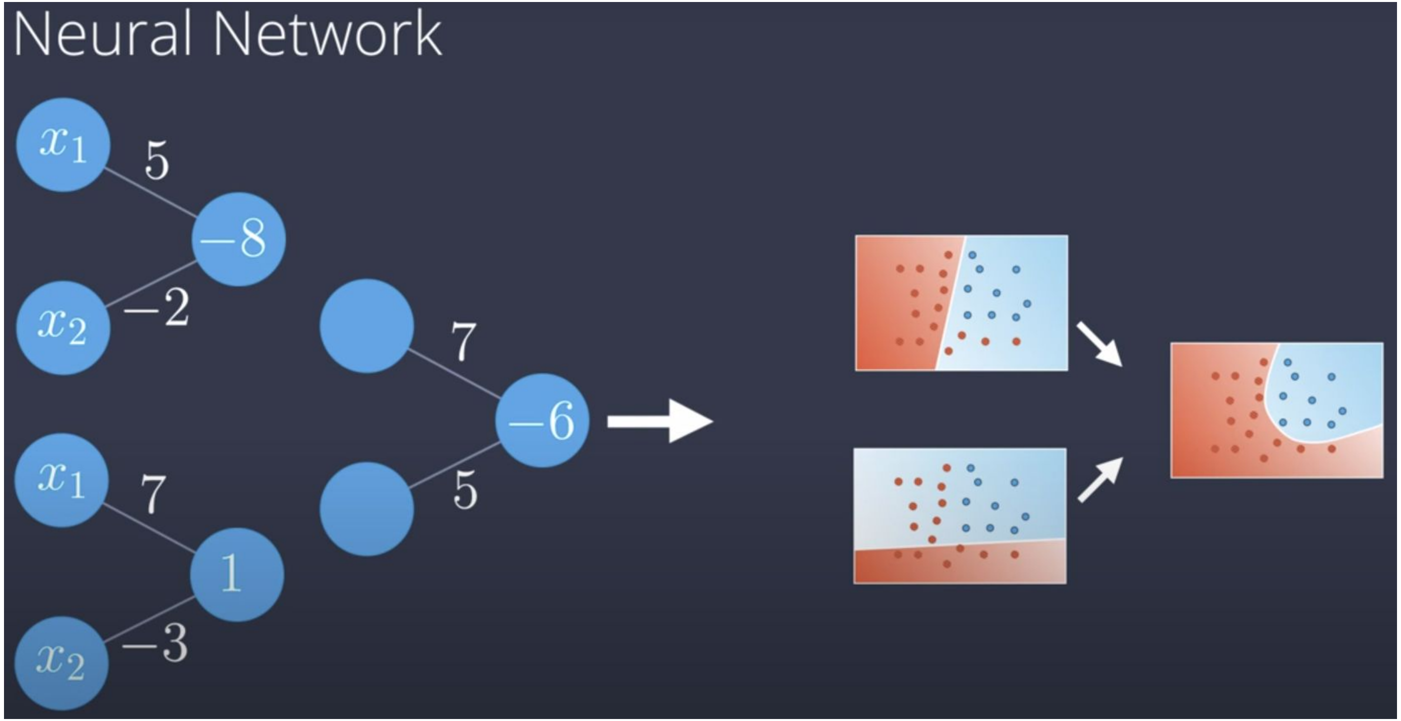


$$\hat{y} = \sigma \begin{pmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{pmatrix} \sigma \begin{pmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

$$\hat{y} = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

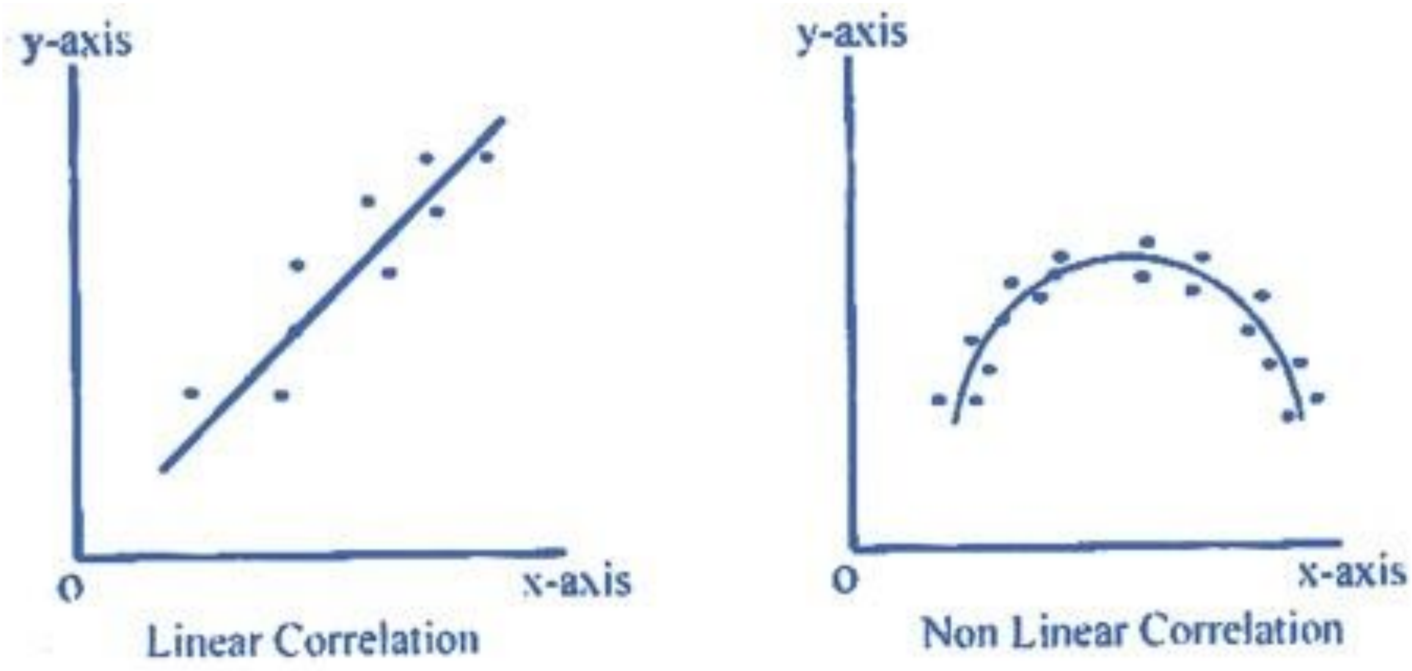
[Understanding the feedforward pass]

Learning non-linearities



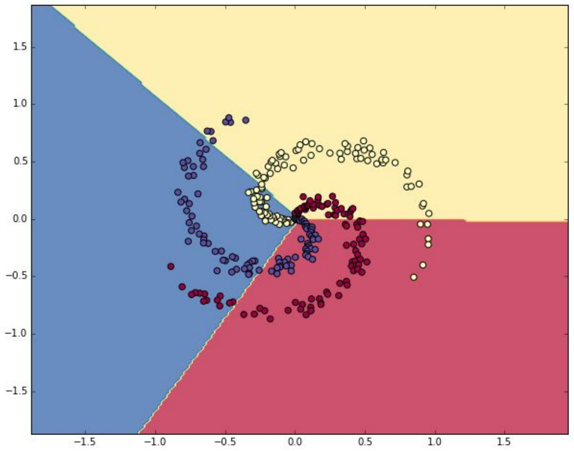
[How neural networks learn non-linearities]

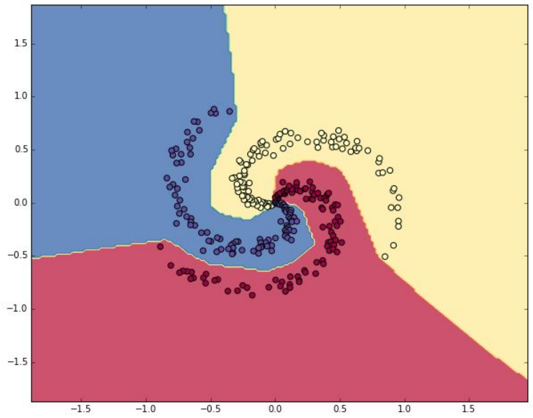
Learning non-linearities



[How neural networks learn non-linearities]

Learning non-linearities

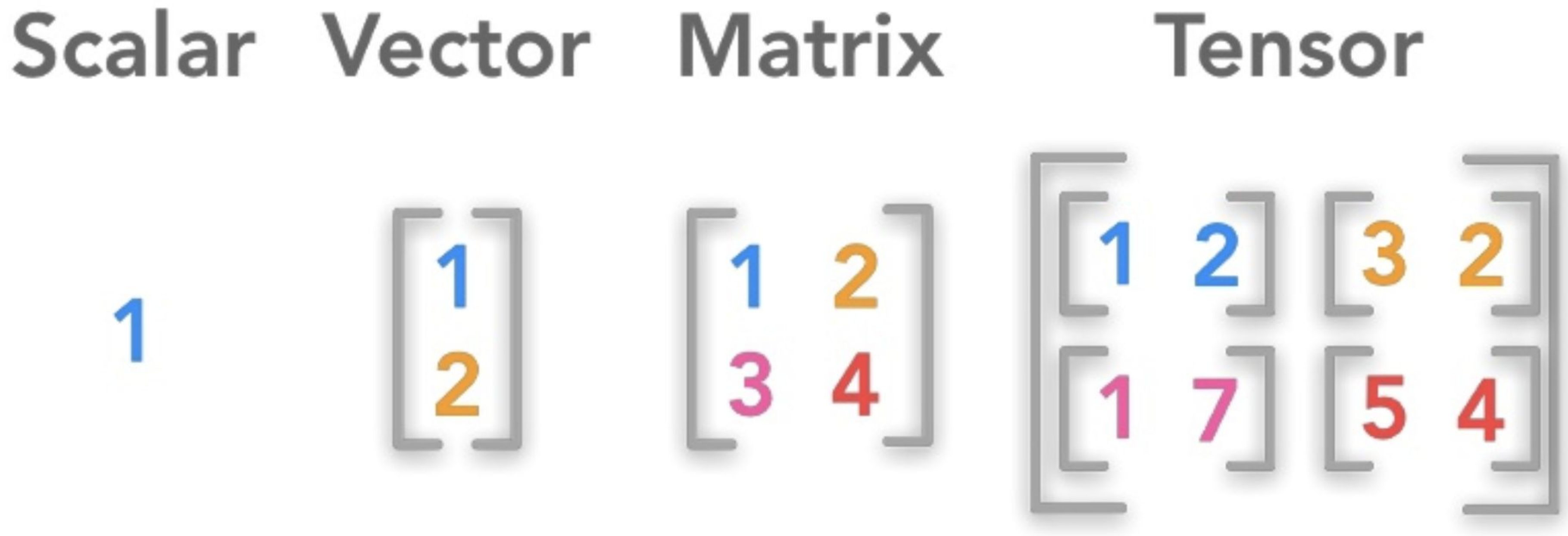




https://cs231n.github.io/neural-networks-case-study

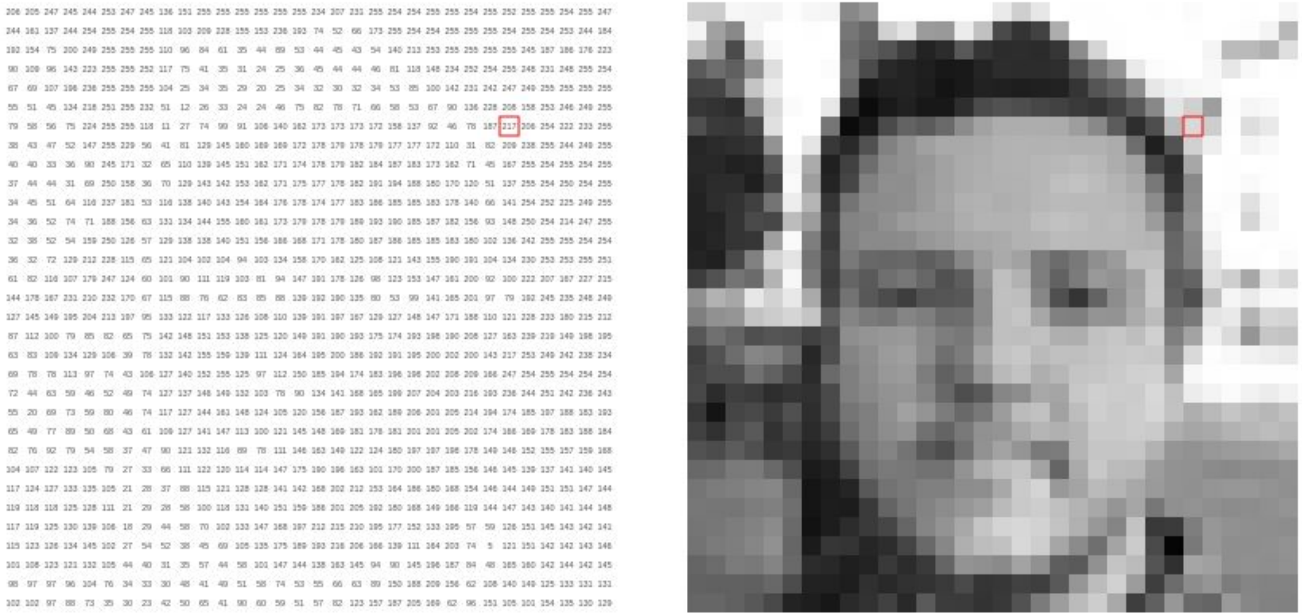
Experiment notebook

Q: What happens with large learning rates? What happens with large coefficients for l2 regularization?



- [Read: scalars, vectors, matrices, and tensors]()
  - [Understanding rank terminology]()

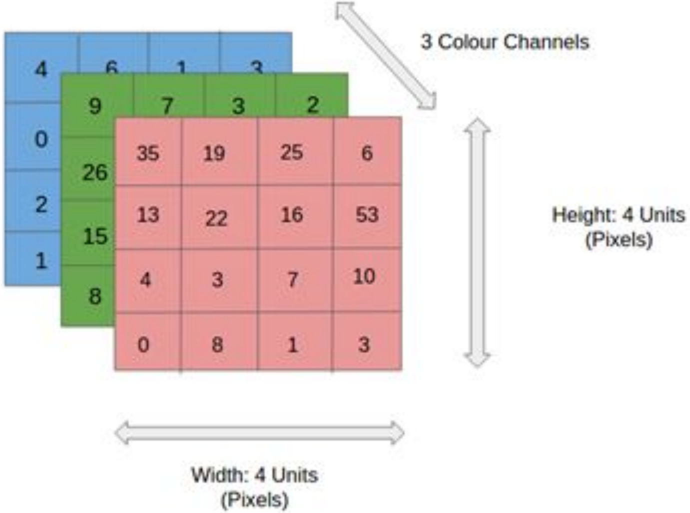
How do computers represent images?



https://setosa.io/ev/image-kernels/

Images as tensors

What a human sees



What the computer 'sees'

Matrix multiplication

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 2 & 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 5 \\ 6 & 7 \\ 1 & 8 \end{bmatrix}$$

Multiply

http://matrixmultiplication.xyz/

Matrix multiplication

$$\begin{bmatrix} 2 & -2 \\ 5 & 3 \end{bmatrix} \times \begin{bmatrix} -1 & 4 \\ 7 & -6 \end{bmatrix} = \begin{bmatrix} (2)(-1)+(-2)(7) & 2\cdot 4+(-2)(-6) \\ 5(-1)+3(7) & 5\cdot 4+3(-6) \end{bmatrix}$$

[Khan Academy Tutorial]

Pay attention to the size of the input

Backpropagation

Finding the derivative of the error [edit]

Calculating the partial derivative of the error with respect to a weight w_{ij} is done using the chain rule twice:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net}_j}\frac{\partial \mathrm{net}_j}{\partial w_{ij}} \qquad \textbf{(Eq. 1)}$$

In the last factor of the right-hand side of the above, only one term in the sum net_j depends on w_{ij}, so that

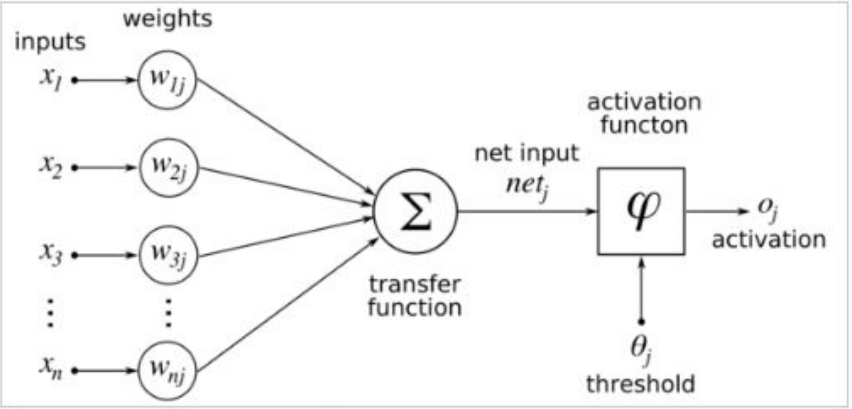
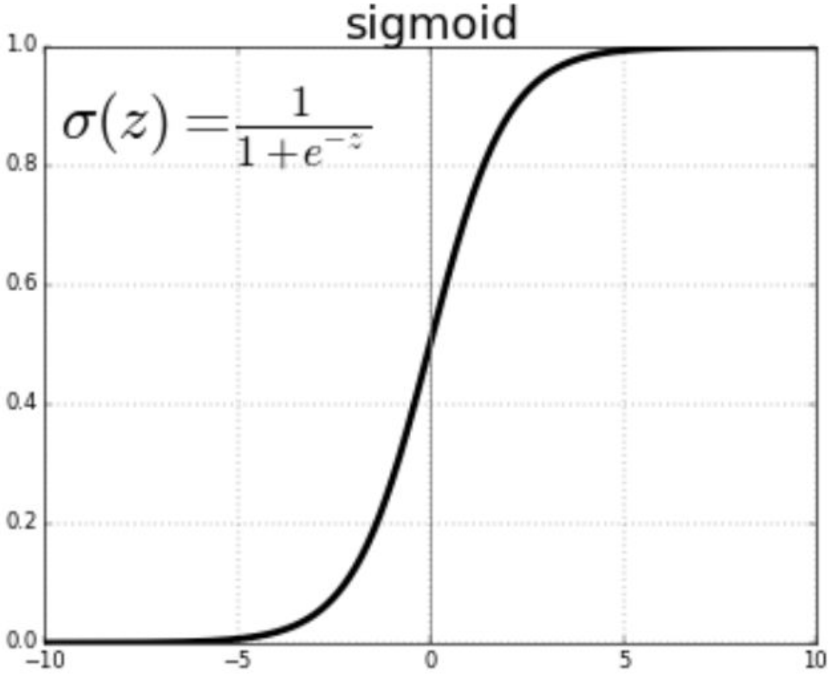


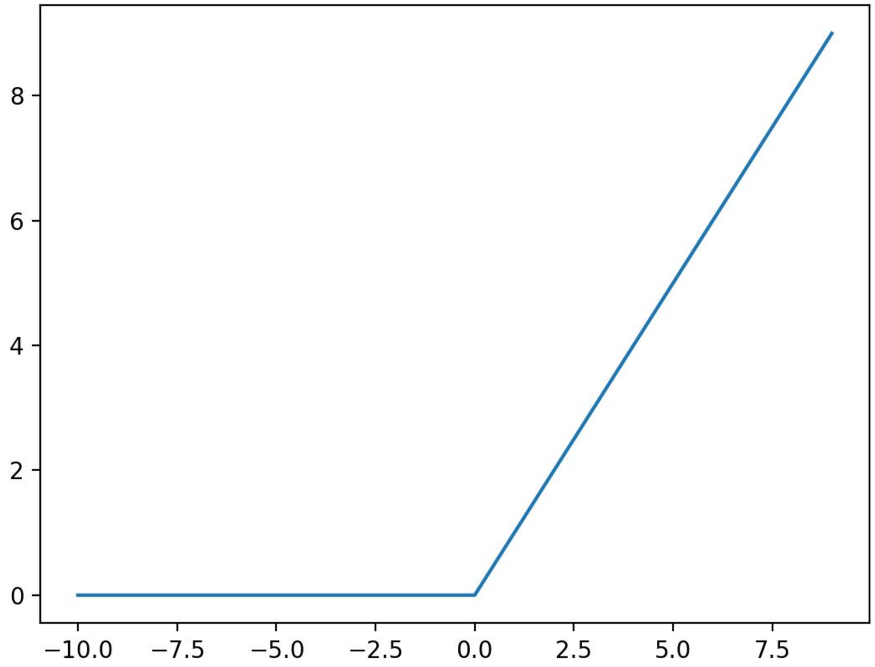
Diagram of an artificial neural network to illustrate the notation used here.

https://en.wikipedia.org/wiki/Backpropagation

The sigmoid activation function



The rectifier linear unit (ReLU) activation function



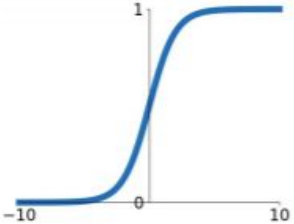
$$f(x) = x^+ = \max(0, x)$$

Activation functions

ReLU is a good default choice for most problems

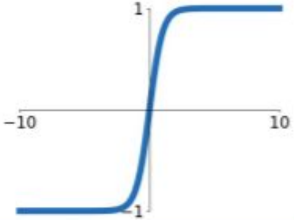
Sigmoid

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



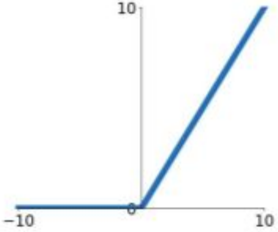
tanh

$\tanh(x)$



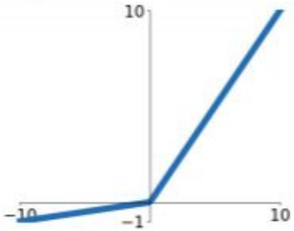
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

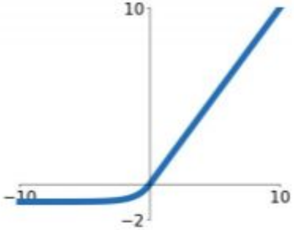


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

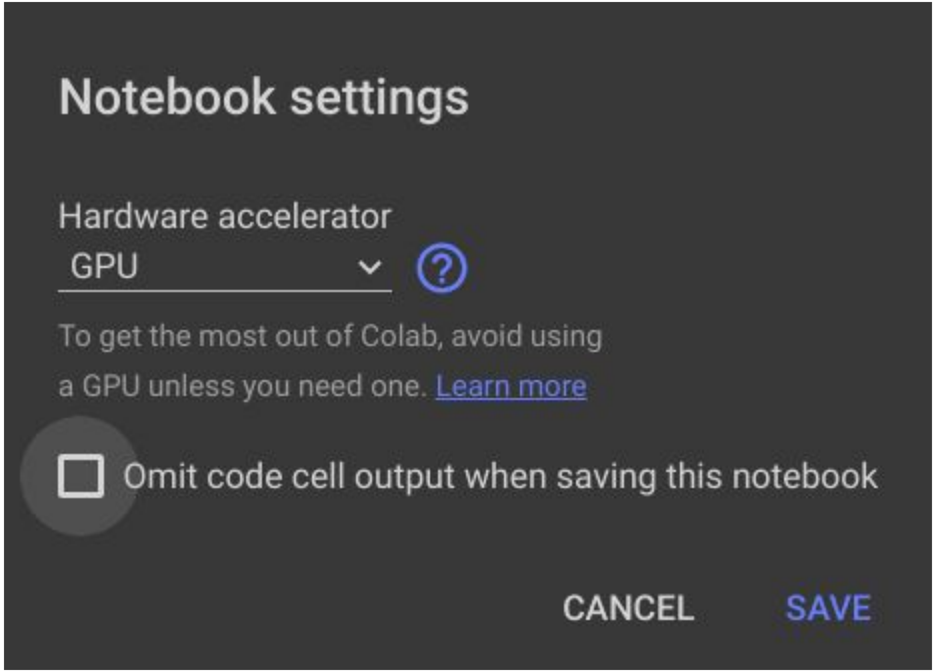
ELU

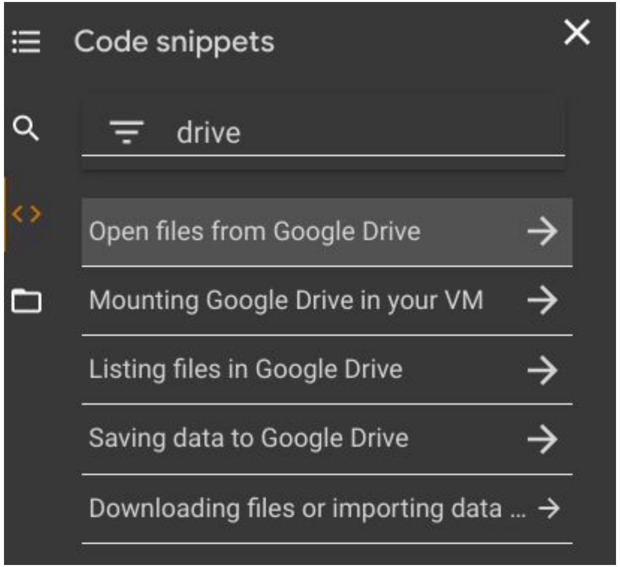
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Setting up Google Colab





Installing PyTorch

PyTorch Build	Stable (1.7.1)	Preview (Nightly)			
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python	C++ / Java			
CUDA	9.2	10.1	10.2	11.0	None

Run this Command:

```
pip install torch==1.7.1+cpu torchvision==0.8.2+cpu torchaudio==0.7.2 -f h
ttps://download.pytorch.org/whl/torch_stable.html
```

https://pytorch.org/

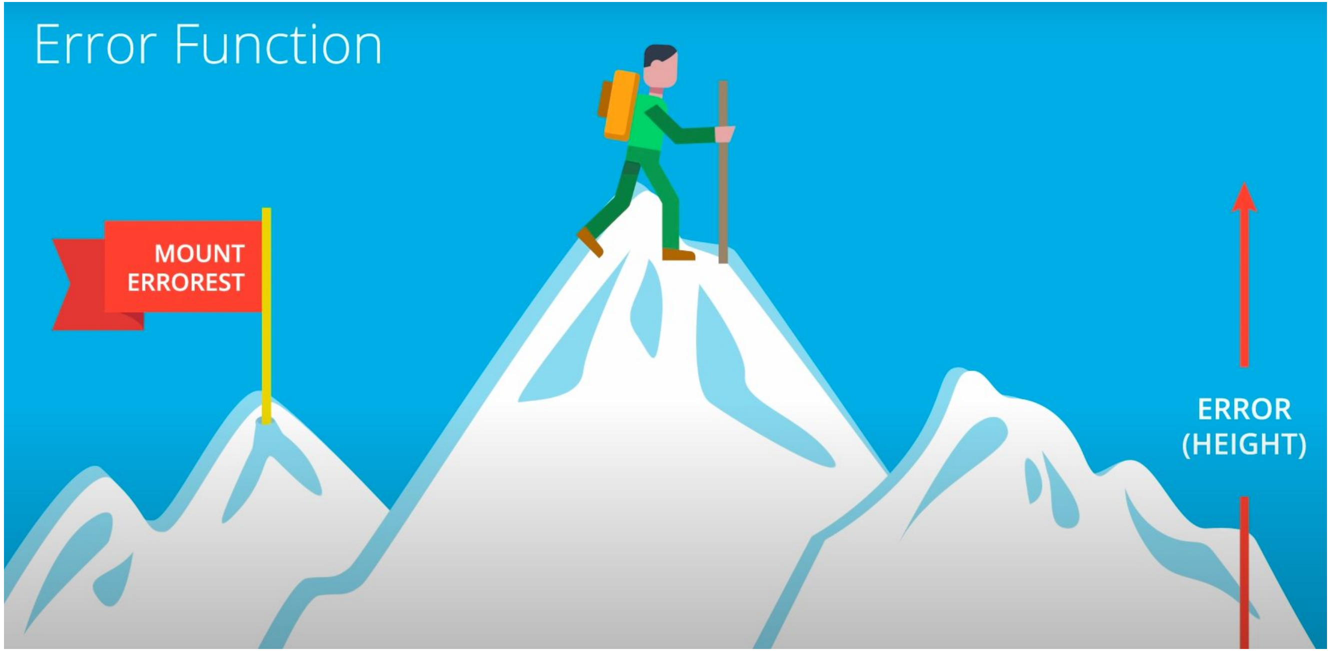
Practice: tensors in Pytorch

- Notebook
- Solutions

Questions:

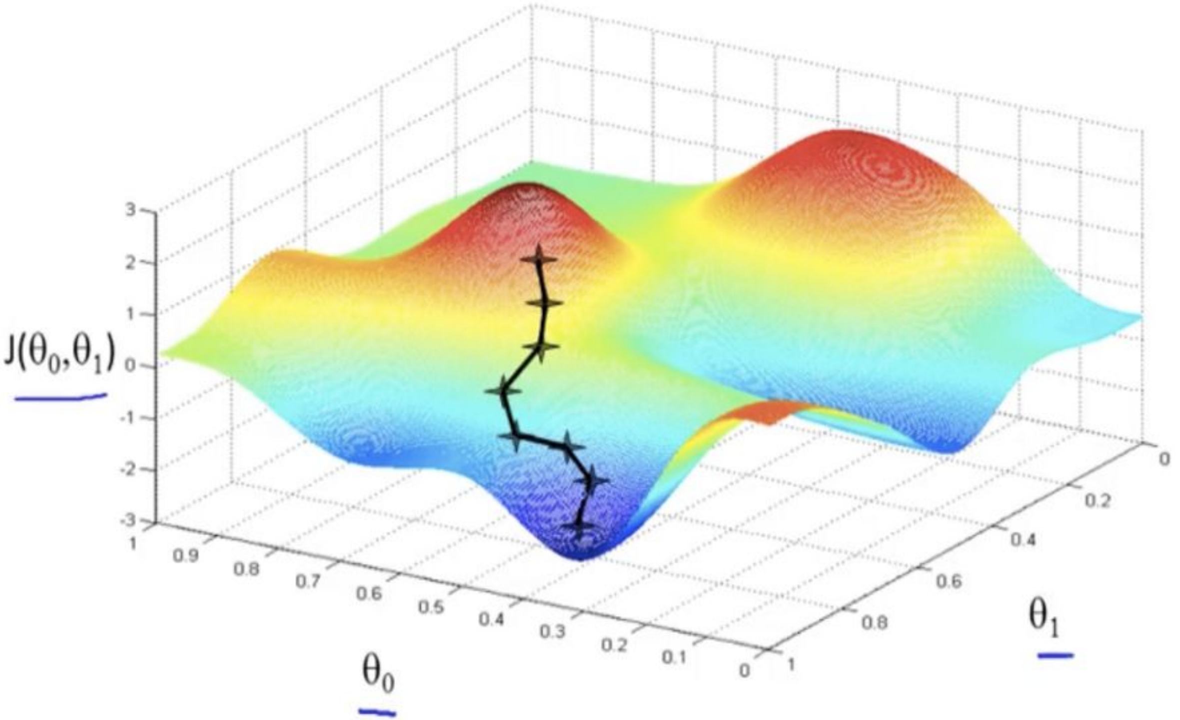
- Why do we use bias terms?
- How is a torch tensor different from a numpy array?
- Is a torch tensor always run on a GPU?
- What is an in-place operation? How are in-place methods named in Pytorch?
- Is the rectifier linear unit a linear or a non-linear function?

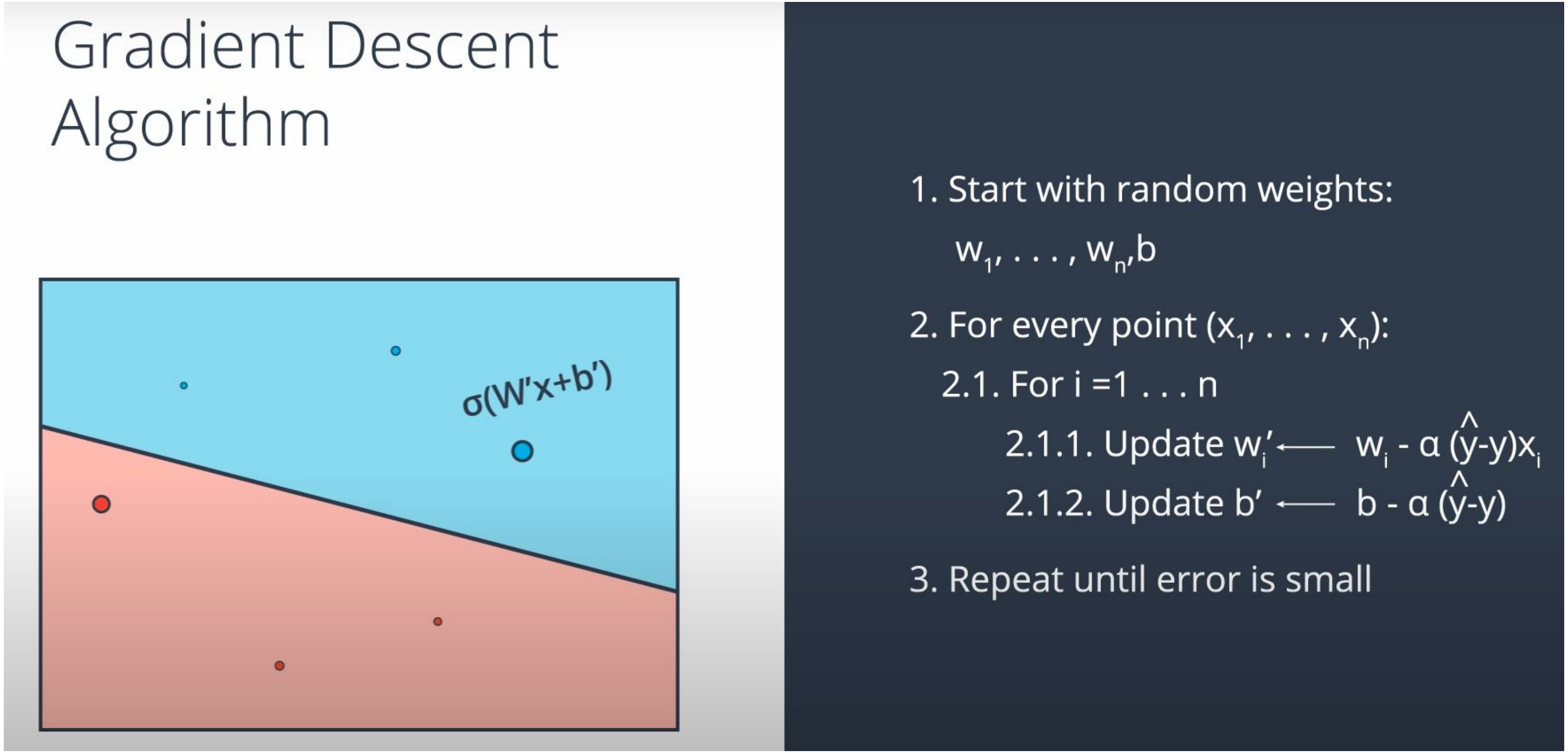
Gradient descent



https://github.com/fastai/fastbook/blob/master/04_mnist_basics.ipynb

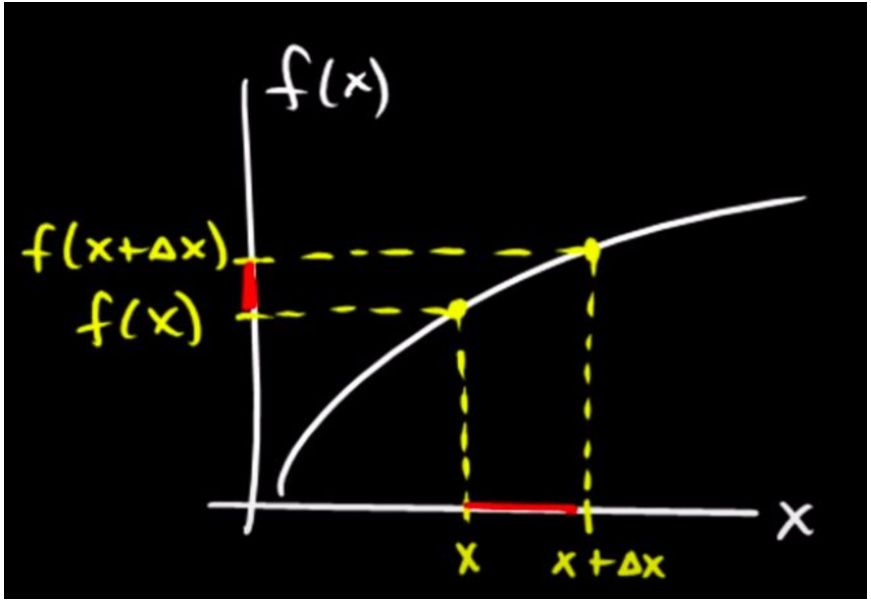
Stochastic gradient descent





[Implementing gradient descent]

Numerical calculation of derivatives



[Numerical derivatives]

[Numerical and analytic derivatives in Python]

Gradient computation graphs

Backpropagation: a simple example

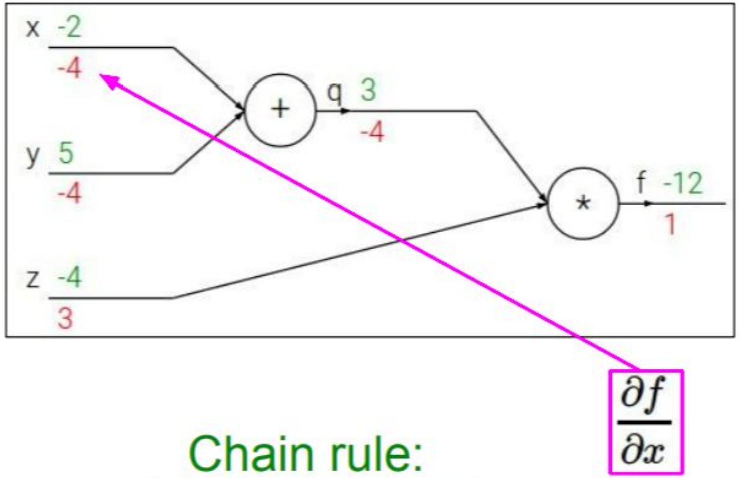
$$f(x, y, z) = (x + y)z$$

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

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

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



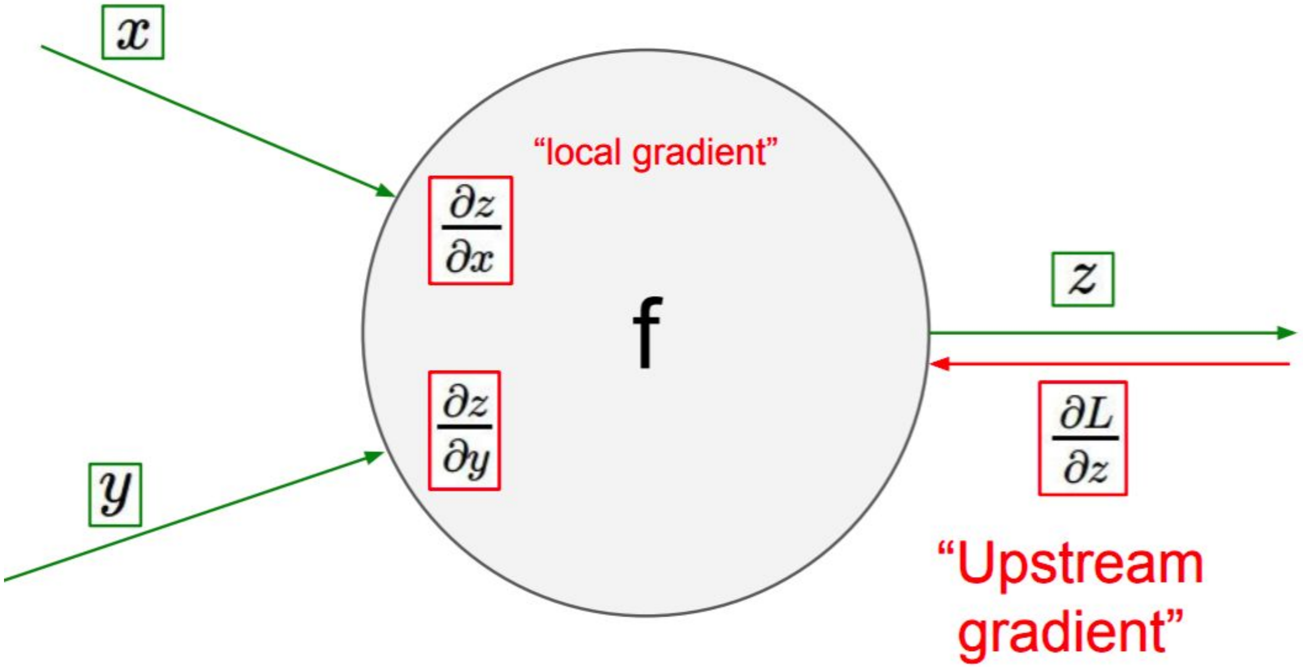
Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

Upstream gradient   Local gradient

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Gradient computation graphs



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Scalar to Scalar

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Regular derivative:

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

If x changes by a small amount, how much will y change?

Vector to Scalar

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

Derivative is **Gradient**:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left(\frac{\partial y}{\partial x}\right)_n = \frac{\partial y}{\partial x_n}$$

For each element of x, if it changes by a small amount then how much will y change?

Vector to Vector

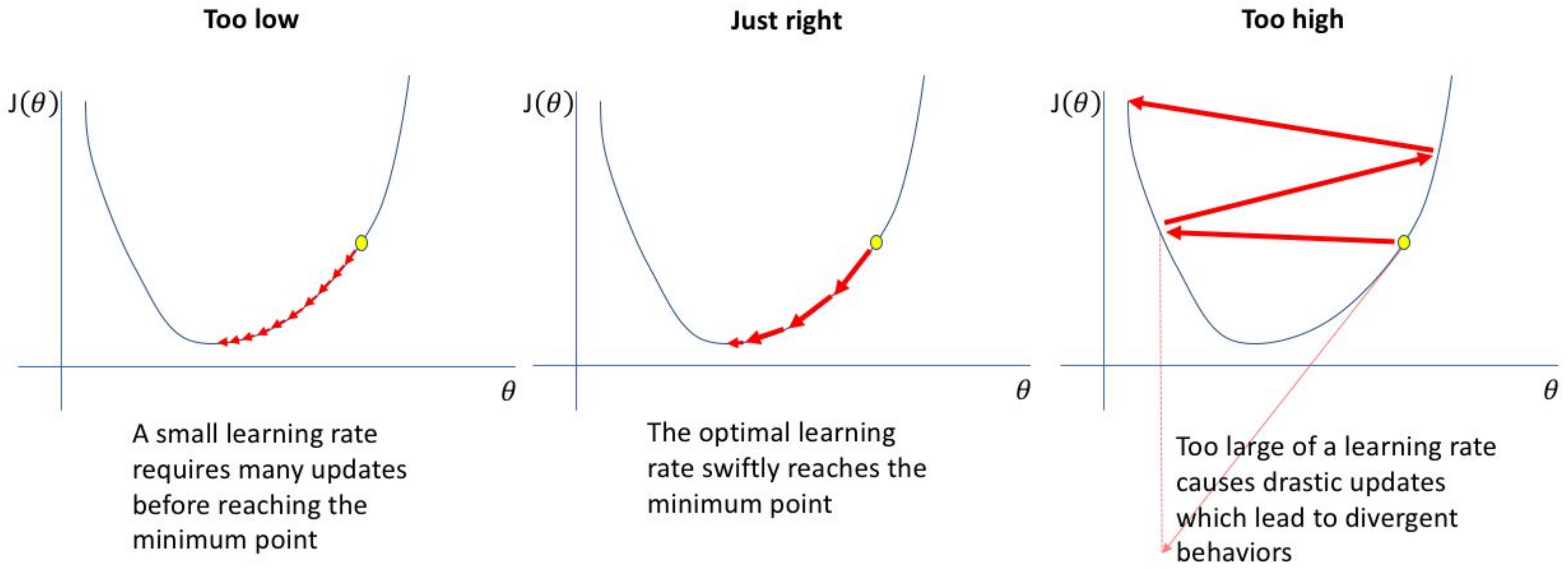
$$x \in \mathbb{R}^N, y \in \mathbb{R}^M$$

Derivative is **Jacobian**:

$$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M} \quad \left(\frac{\partial y}{\partial x}\right)_{n,m} = \frac{\partial y_m}{\partial x_n}$$

For each element of x, if it changes by a small amount then how much will each element of y change?

The importance of the learning rate



The importance of the learning rate

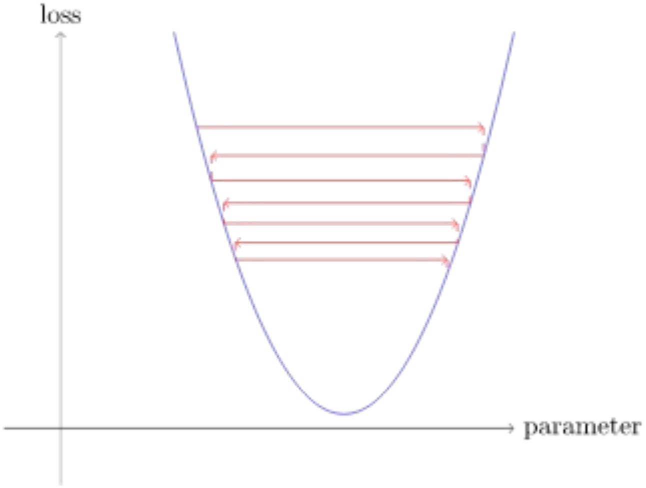
$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment

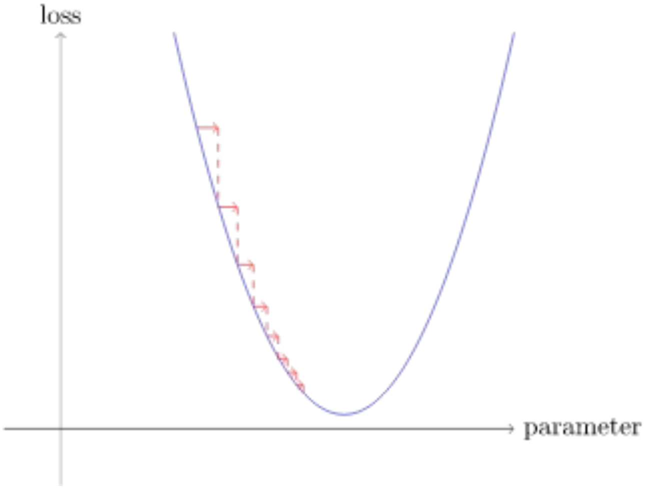
learning rate

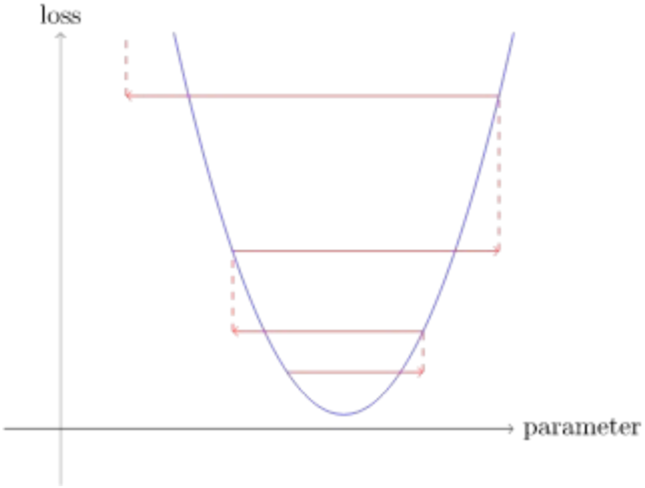
weight gradient

Choosing a learning rate



Choosing a learning rate





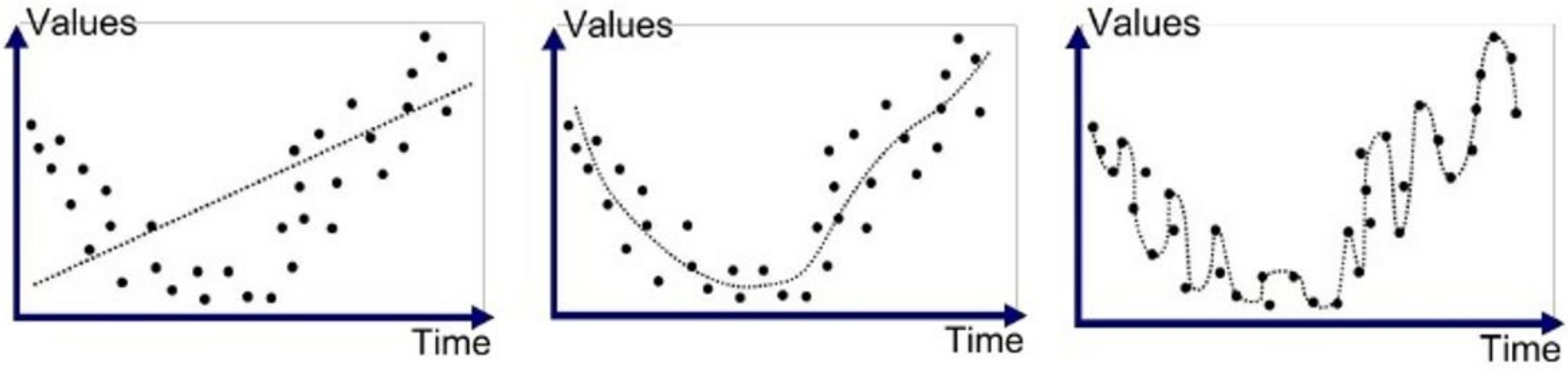
Weight decay aka lx-regularization

$$\text{L1 ERROR FUNCTION} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i)+y_i ln(\hat{y}_i)+\boxed{\lambda(|w_1|+...+|w_n|)}$$

$$\text{L2 ERROR FUNCTION} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i)+y_i ln(\hat{y}_i)+\boxed{\lambda(w_1{}^2+...+w_n{}^2)}$$

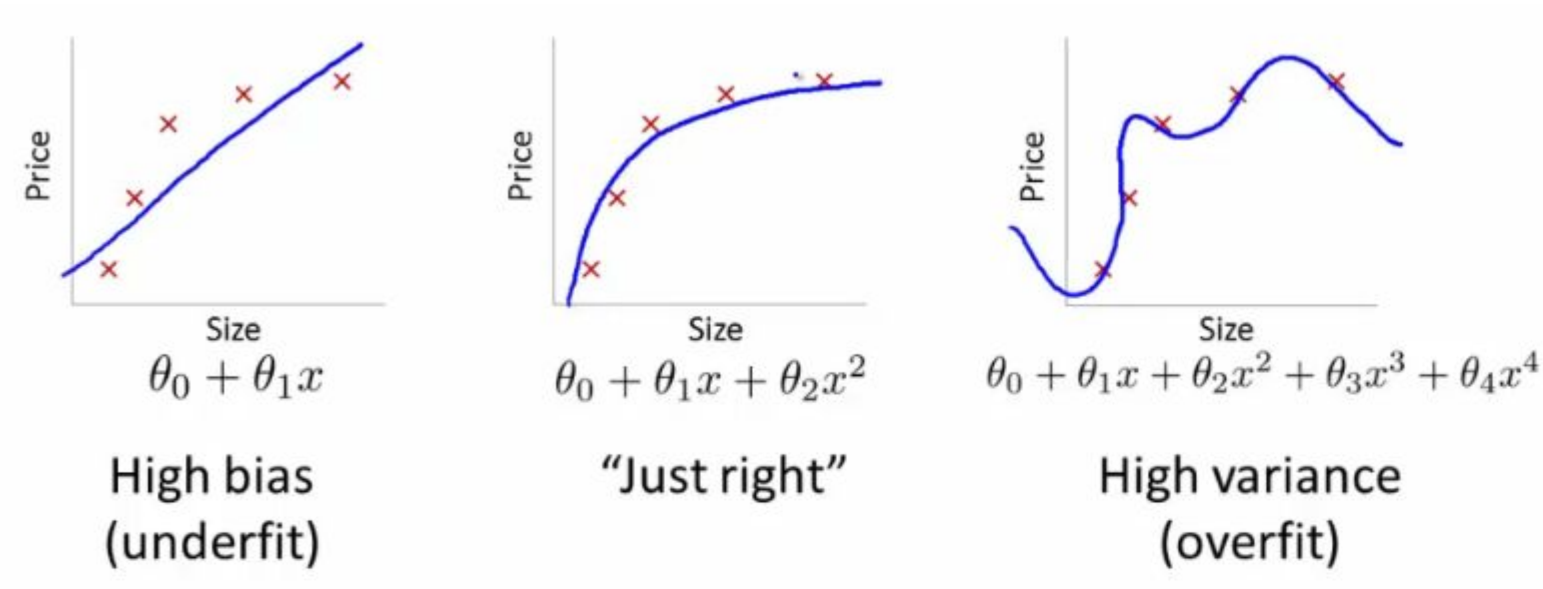
[L1 and L2 regularization]

The bias vs variance tradeoff

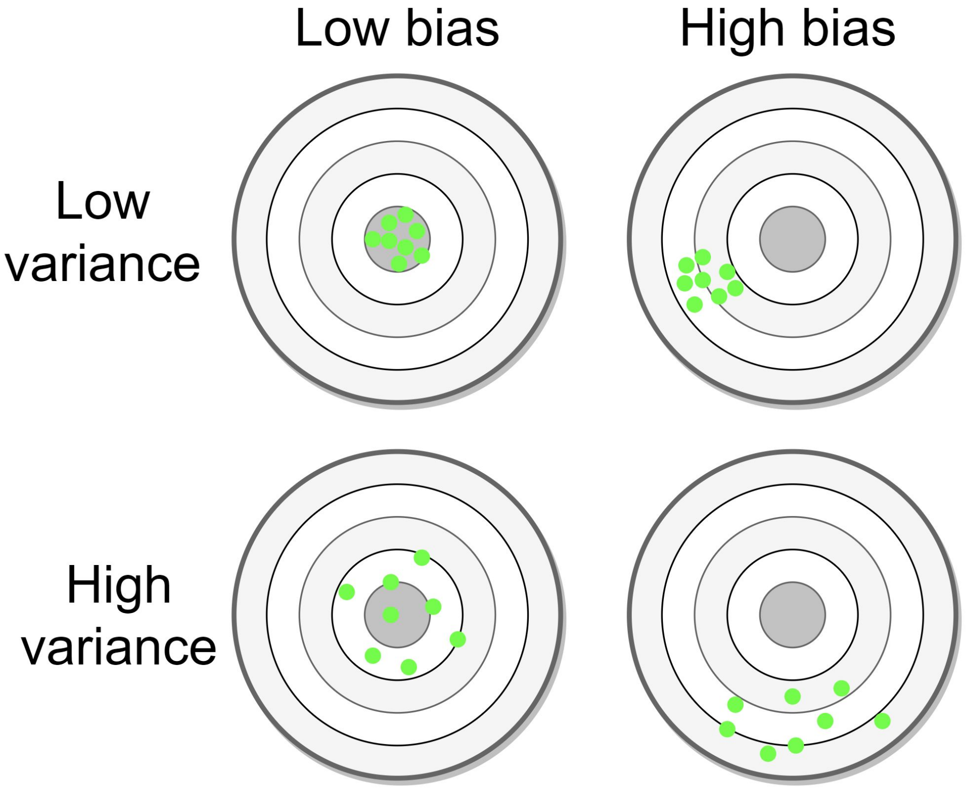


Underfitted | Good Fit/Robust | Overfitted

The bias vs variance tradeoff



The bias vs variance tradeoff



Practice: neural networks in PyTorch

- Notebook
- Solutions

Questions

- How do we subclass nn.Module?
  - What is a subclass?
- How many layers do we need to learn non-linear mappings?
- What is the purpose of a dataloader?
- What is nn.Linear?
- What is the purpose of nn.Sequential?
- Why would we use OrderedDict to define an architecture?

The Softmax function

The softmax function takes as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the interval $(0,1)$, and the components will add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities.

The standard (unit) softmax function $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ is defined by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

https://en.wikipedia.org/wiki/Softmax_function

The softmax activation function in Python

Practice: training in PyTorch

- Notebook
- Solutions

Questions:

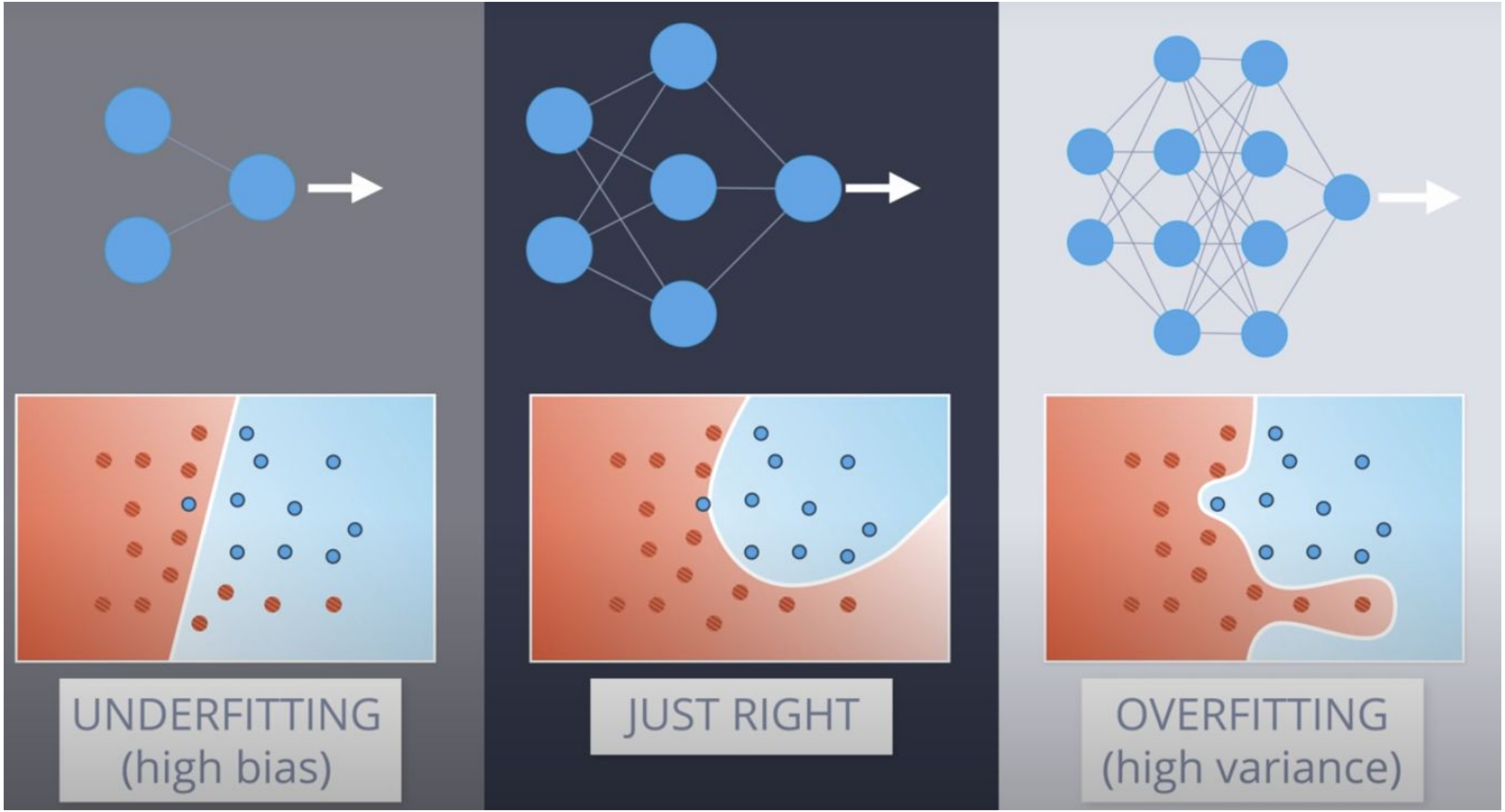
- What is a logit (in the context of neural networks)?
- How do we turn logits into probabilities?
- What criterion should we use if we change nn.LogSoftmax to Softmax? (Read this discussion)
- Where do we define the size of the batch?

Performance metrics vs loss functions

A model performance metric measures the quality of predictions in the validation or test sets. Accuracy, precision, recall, specificity, fbeta, f1, and ROC are performance metrics, not loss functions

The default classification error metric in fastai is error_rate = 1 - accuracy

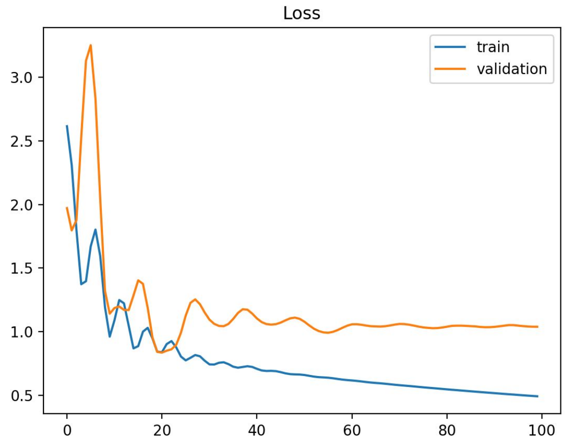
The loss function is used to adjust the weights of the model (through its gradient). The error metric is a report of performance and it can be opposite to the loss function

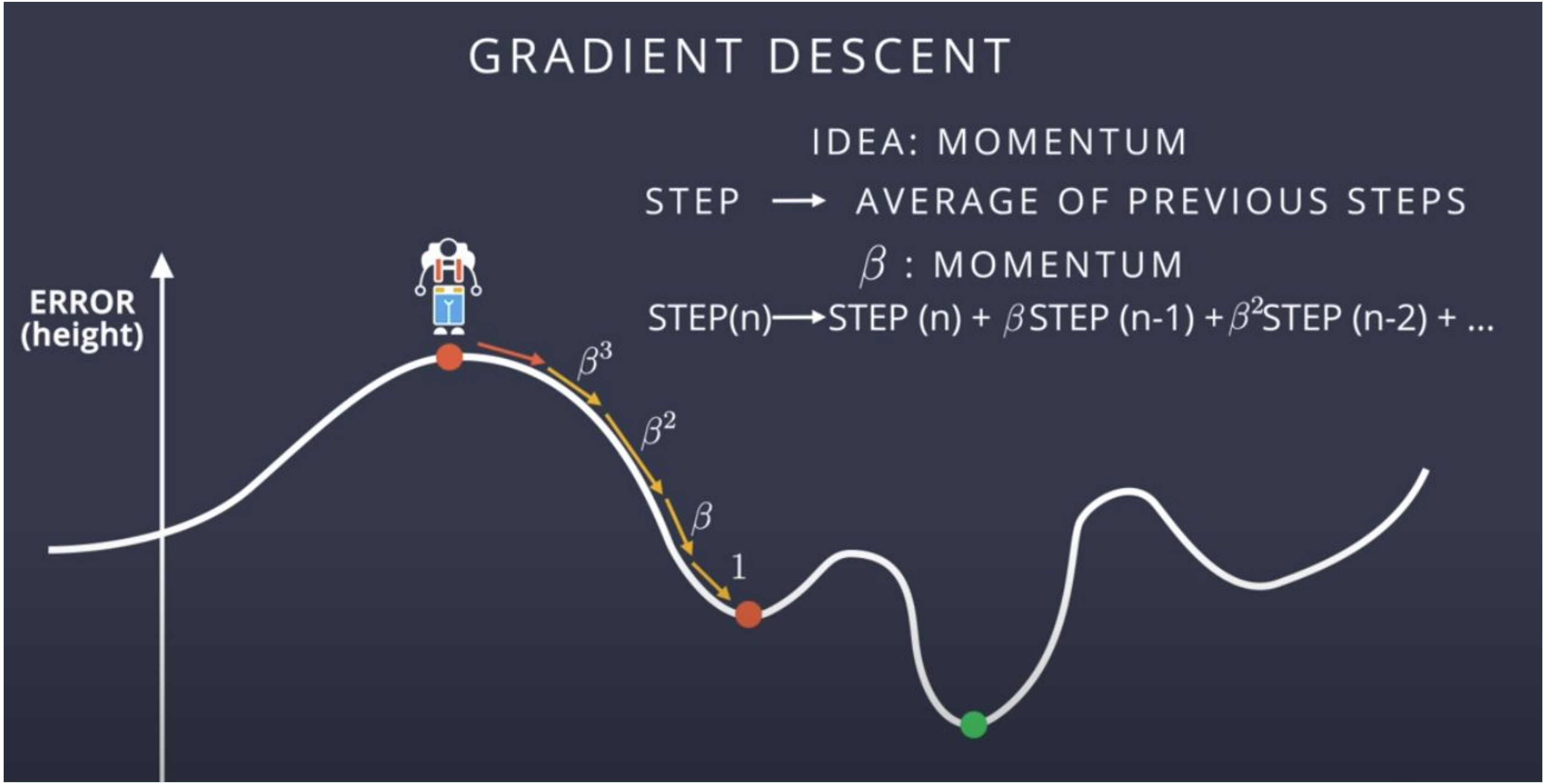


[Understanding the bias vs variance tradeoff in neural network architectures]

How long should we train the network?

- Until it overfits! (Training loss goes down while validation loss goes up)
- This general principle applies to **all** hyperparameters related to model complexity





[The intuition of momentum]

[Exploring momentum's beta (notebook)]

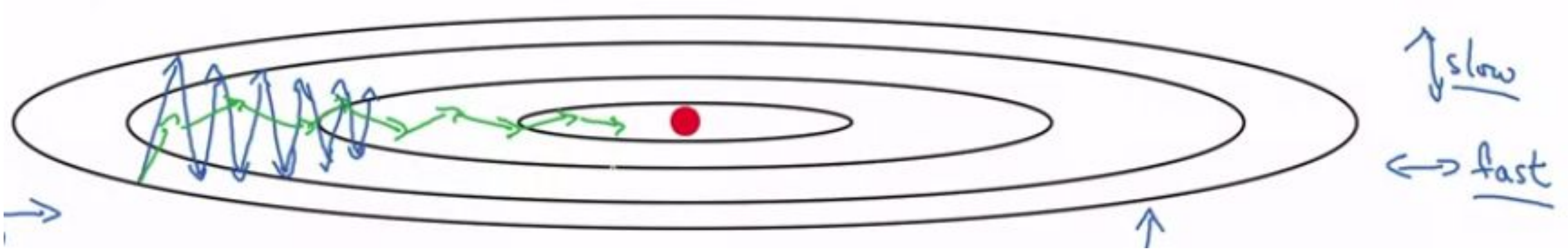
Understanding momentum

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor

weight increment, previous iteration

The ADAM optimizer



Exponentially moving averages and used to smoothen the trajectory of the gradient descent. Averages are computed across training batches. This optimization algorithm usually trains faster than SGD

Practice: classifying images with fully connected networks

- Notebook
- Solutions

Questions:

- How is the image fed into the network?
- Why shouldn't we use the network's accuracy as a Pytorch **criterion**?
- What is the optimizer?
- How does ADAM differ from Stochastic Gradient Descent? How are they similar?
- What is the **lr** parameter that we pass to the optimizer?
- Why do we pass model.parameters() to the optimizer?

Review questions

- What is a loss function? Is the f1 measure a loss function or a performance metric?
- Explain what the ReLU activation function does in one sentence.
- Can we use accuracy as a loss function? Can we use precision or recall as loss functions?
- Is the accuracy guaranteed to increase as the value of the loss function goes down?
- Which parts of the model need to be differentiable for an architecture to be trainable?
- What are we trying to minimize when using gradient descent? What is the learning rate?
- What is a GPU? Why are GPUs useful for deep learning?

Review questions

- What is an epoch? What is a batch? What is the maximum size of a batch? How does the batch size influence training?
- What is the difference between stochastic gradient descent (SGD) and 'vanilla' AKA 'regular' gradient descent?
- What is momentum? What is the difference between ADAM and SGD?
- What is a Jacobian? What is a Hessian? Do we use Hessians in deep learning?
- Suppose that we have a ground truth for a 'cat' label that is encoded as [0, 1]. We can assume that the ground truth for a 'dog' label is [1, 0] (**cat** is the training point).
    - The network first outputs [0.75, 0.25] as prediction
    - In the next epoch it outputs [0.99, 0.01] as prediction
    - Has the cross entropy loss increased or decreased?

Review questions

- What does it mean for a dataset to be ‘linearly separable’?
- How many layers does a neural network need in order to learn non-linear correlations?
- What’s the difference between a performance metric and a loss function?
- Why should we shuffle the data in every training epoch?
- Explain the bias vs variance tradeoff.
  - Effect of increasing the number of training epochs
  - Increasing the number of layers
  - Increasing the amount of dropped out connections
- How would you define ‘regularization’?
- What is the purpose of dropout? How does it work? Should dropout be active during inference time?

Review questions

- What is the vanishing gradient problem?
- What does it mean to “transfer a tensor” to the GPU?
- What is CUDA?
- How does the amount of RAM available in the GPU affect the types of models and data that we can use to train?
- What should we try when torch reports ‘CUDA out of memory’ errors?

Review questions

- Why do we need to call `optimizer.zero_grad()` in the training loop?
- What’s the difference between a parameter and a hyperparameter? What’s another common name for ‘parameter’ in a neural network?
- How can we know what’s the “optimal” number of units in a hidden layer?
- How can we know the “optimal” number of layers in a network?
- Is data augmentation a regularization technique? Why should we apply data augmentations probabilistically?
- Why should we shuffle the data inside a training batch?
- If a model is increasing its loss through several epochs should we increase or decrease the learning rate?
- How do we evaluate training and validation loss to do early stopping?

Review questions

- What is weight decay? What is learning rate decay?
- Suppose that after going through certain number of epochs the training loss is higher than the validation loss. Is this an example of underfitting or overfitting?
- What is the vanishing gradient problem? Why does it appear?
- Are vanishing gradients more likely to appear when representing Jacobians and weight vectors with 8 bits, 16 bits, or 32 bits of precision?
- Why are ReLU activation functions less likely to display vanishing gradients than sigmoid activation functions?

Review questions

- Why do we resize images before training?
- If we retrain or do inference on a network pre-trained on Imagenet, why do we need to normalize its image channels using the mean and std of the intensities of the original dataset?
- What is a receptive field?
- What is a convolution?
- Why do we apply padding to images before doing convolutions?
- When calling conv2d(...,..., 64) how many output channels are we producing? How are they different from each other?
- Why is the vanishing gradient a ‘numerical’ problem? What does that mean?
- Are we more likely to get vanishing gradients when using fp16 or fp32 data types for our weights?

Resources

- Intro to Deep Learning with PyTorch at Udacity
- Convolutional Neural Networks at Coursera
- Fast.ai - Deep Learning for Coders
- PyImageSearch Blog
- Stanford's CS231: Convolutional Neural Networks for Visual Recognition
- 3blue1brown's series on neural networks
- Goodfellow's Deep Learning Book (great for theory)
- Visualizing and understanding neural networks (Stanford lecture)
- Review content on cross entropy
    - On ML Mastery
    - Brief video by Udacity
    - Aurelien Geron's explanation

Regression with a neural network

https://utstat.toronto.edu/reid/sta2212s/2021/Galton85.pdf

https://www.researchgate.net/publication/280970132_Galton's_Family_Heights_Data_Revisited