Università degli
Studi di Milano-Bicocca

Neural Networks

Prof. Flavio Piccoli

a.a. 2022-2023

Welcome to the course!

What will you learn in Advanced Computational Techniques for Big Imaging and Signal Data?

1. **Base technologies**
   a. Neural Networks
   b. Convolutional Neural Networks

2. **The R&D process**
   a. from data collection to production
   b. you will become a full-stack data scientist!

3. **Task and data specific techniques**
   a. signal data
   b. temporal data
   c. images
   d. audio

Welcome to the course!

References:

- the page of the course: https://elearning.unimib.it/course/view.php?id=45585
- my email: flavio.piccoli@unimib.it

Final exam:

- It will be a project on the topics learned during the course

Lesson time:

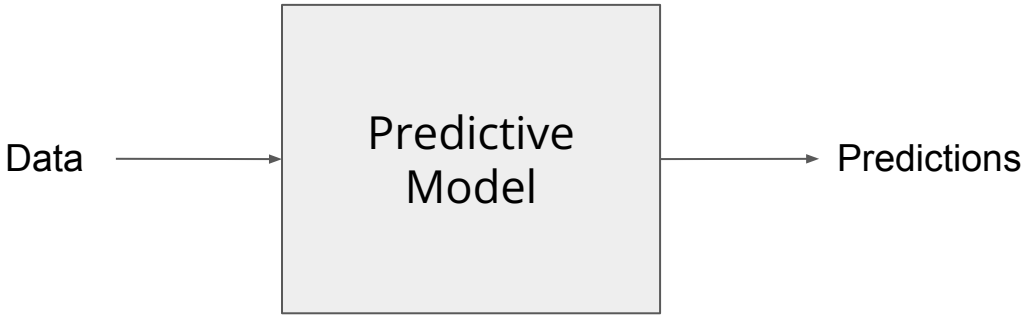
- Tuesday 09:45 - 12:15 (3h)
- Friday 10:45 - 12:15 (2h)

Introduction to machine learning

Definition of Machine learning from Oxford Dictionary:

"*The use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data.*"

Data driven process

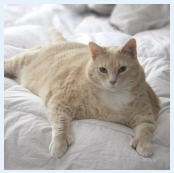


Types of algorithms

Supervised

Every instance of the training set is composed of:

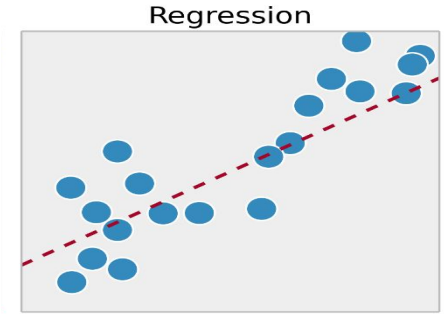
- a set of inputs X
- the expected output Y (ground-truth)

TASK	X (a set of inputs)	Y (the expected output)
Regression		45 years old
Classification		Cat

Unsupervised

The samples do not contain the ground truth Y

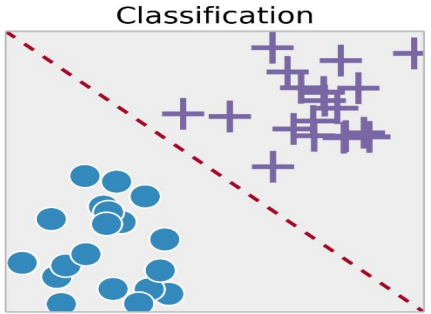
Examples



Output is expected to be a **real number**

Example

Estimate the house price from:
- the squared meters m^2
- the number of rooms



Output is expected to be a **class label**

Example

Tell whether a person is sportif or not from:
- body mass index (BMI)
- age

Gradient Descent

Gradient Descent

FINDING THE MINIMUM OF A FUNCTION

- Suppose you want to find the mininum of a function, what would you do?

$$f(x) = x^2$$

Gradient Descent

FINDING THE MINIMUM OF A FUNCTION

- Suppose you want to find the mininum of a function, what would you do?

$$f(x) = x^2$$

1. Calculate the derivative of f

$$\frac{df}{dx} = 2x$$

2. Set it to 0

$$2x \doteq 0 \rightarrow x = 0$$

How would you build a system that given a function and its derivative, finds a minimum?

Gradient Descent

FINDING THE MINIMUM OF A FUNCTION

IDEA: use the gradient to minimize the function

$$f(x) = x^2 \qquad \frac{df}{dx} = 2x$$

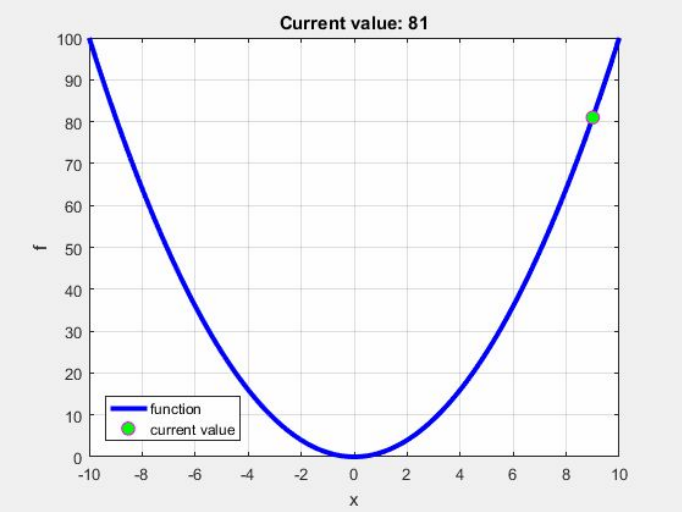
1. Initialize x at a random value
2. while $f'(x) \neq 0$
   3. update x using gradient

$$x_{t+1} = x_t - lr \cdot \frac{df}{dx}(x_t)$$

current value

learning rate

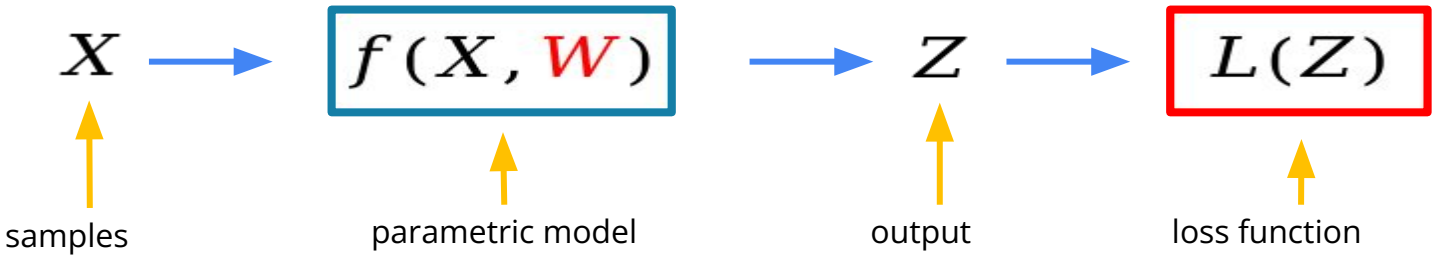
gradient



Gradient Descent

FINDING PARAMETERS OF A MODEL

GOAL: find the parameters W of the model **f** that minimizes the loss function **L**



$$L(Z) = L(F(X, W))$$

The loss function **L**:

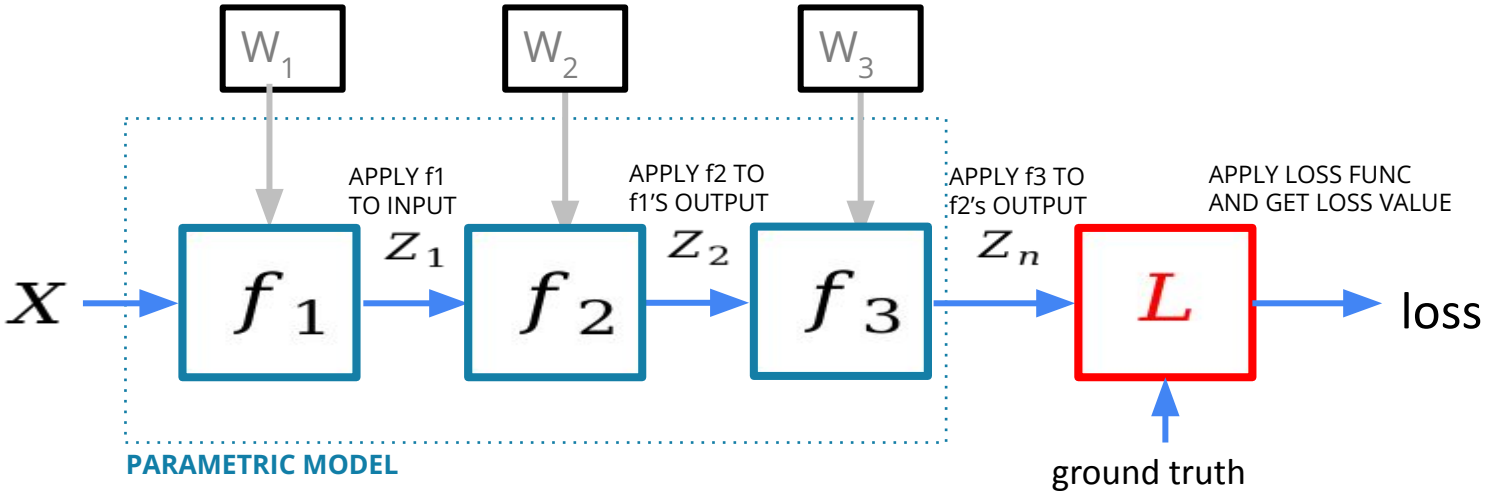
- measures the goodness of the model in describing the input data
- drives the backpropagation
- must NOT have multiple minimums in the z-domain

The parametric model **f**:

- is assumed to be capable to describe the input data
- has weights that have to be learned

Gradient Descent

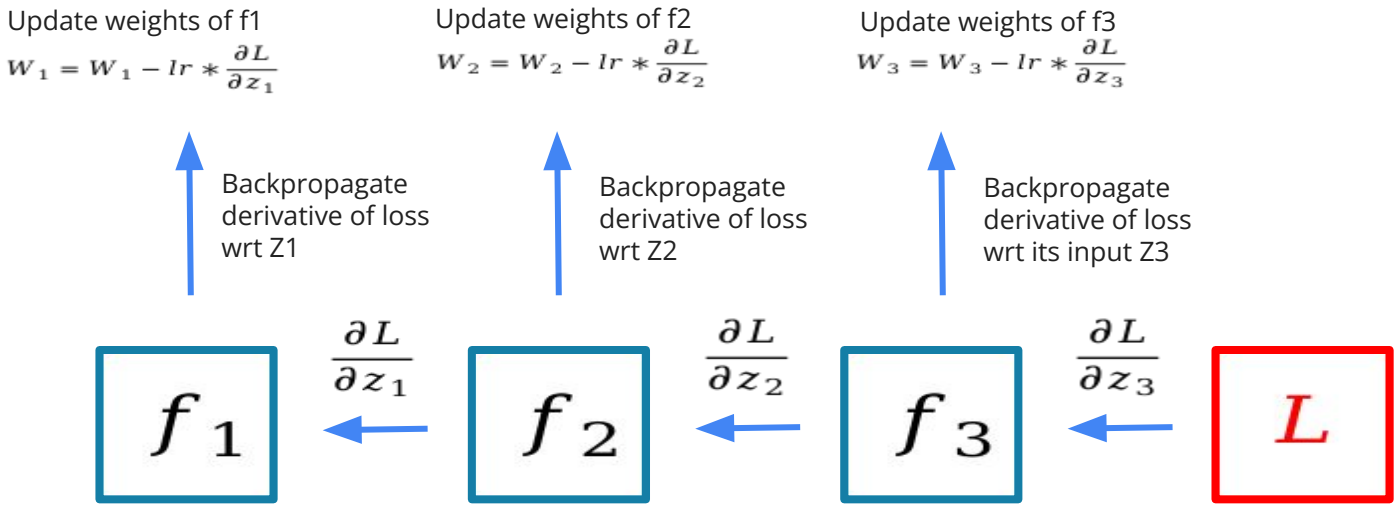
FORWARD PROPAGATION



Gradient Descent

BACKWARD PROPAGATION

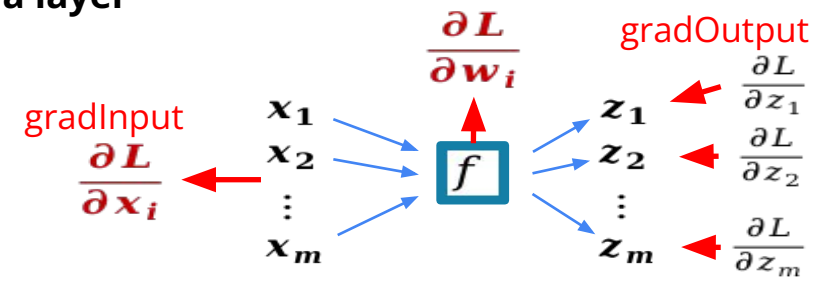
2 - Backward Propagation

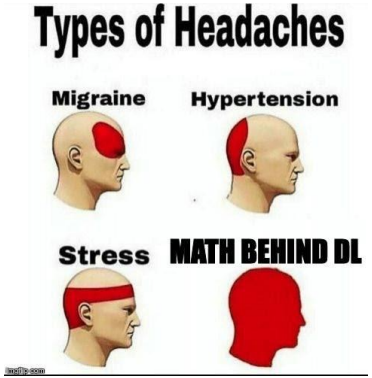


Gradient Descent

CHAIN RULE

Backpropagation of a layer





Calculate derivative of L wrt the weights W

for each weight wi: $\frac{\partial L}{\partial w_i} = \sum_{j=1}^{|Z|} \frac{\partial z_j}{\partial w_i} \frac{\partial L}{\partial z_j}$

for all weights: $\frac{\partial L}{\partial W} = \left(\frac{\partial Z}{\partial W}\right)^T \frac{\partial L}{\partial Z}$

Calculate derivative of L wrt the inputs X

for each input xi: $\frac{\partial L}{\partial x_i} = \sum_{j=1}^{|Z|} \frac{\partial z_j}{\partial x_i} \frac{\partial L}{\partial z_j}$

for all inputs: $\frac{\partial L}{\partial X} = \left(\frac{\partial Z}{\partial X}\right)^T \frac{\partial L}{\partial Z}$

gradInput

gradOutput

jacobian

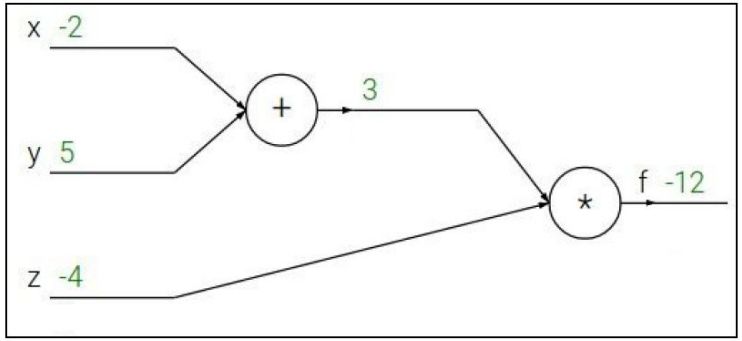
Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

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

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



Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

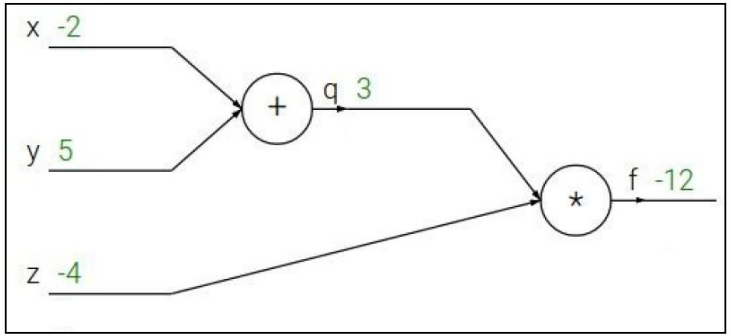
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

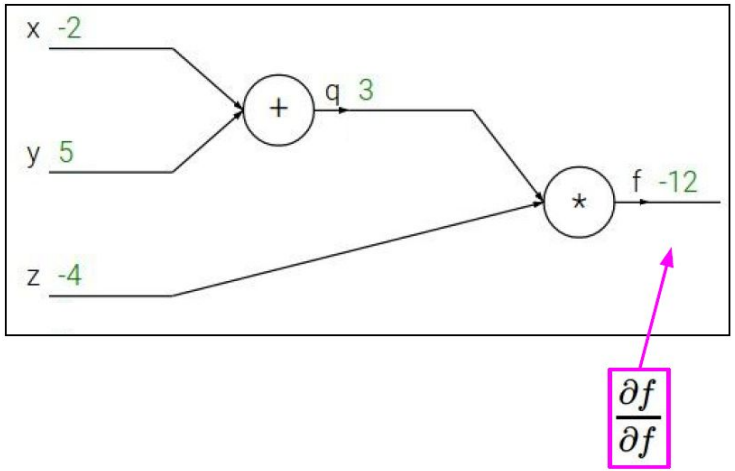
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

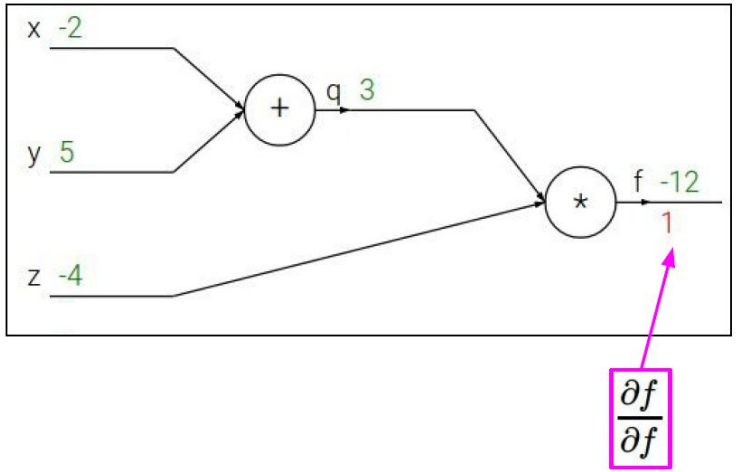
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

●BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

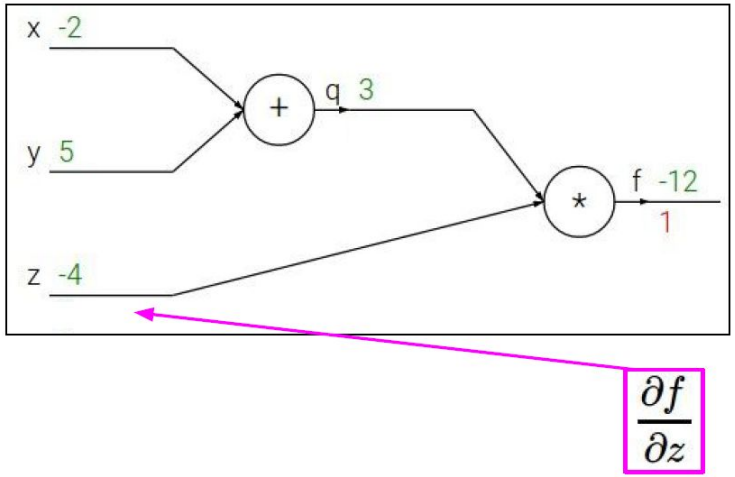
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

●BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

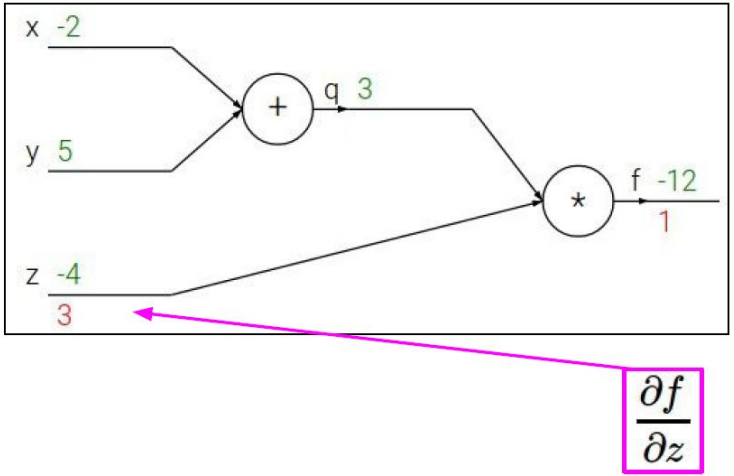
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

●BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

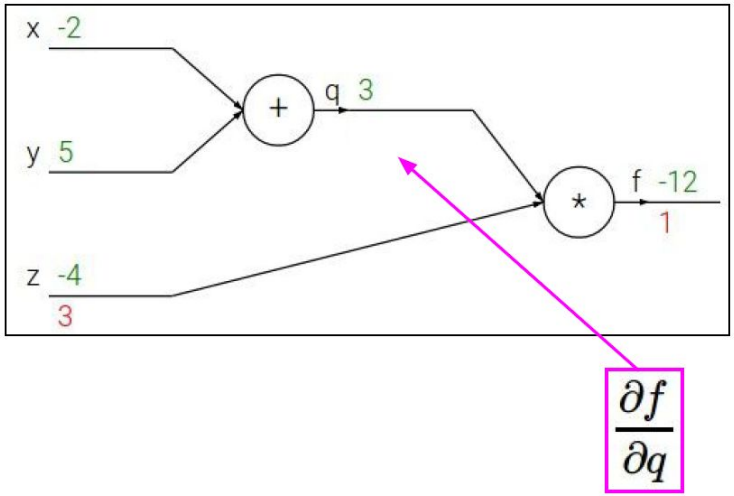
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

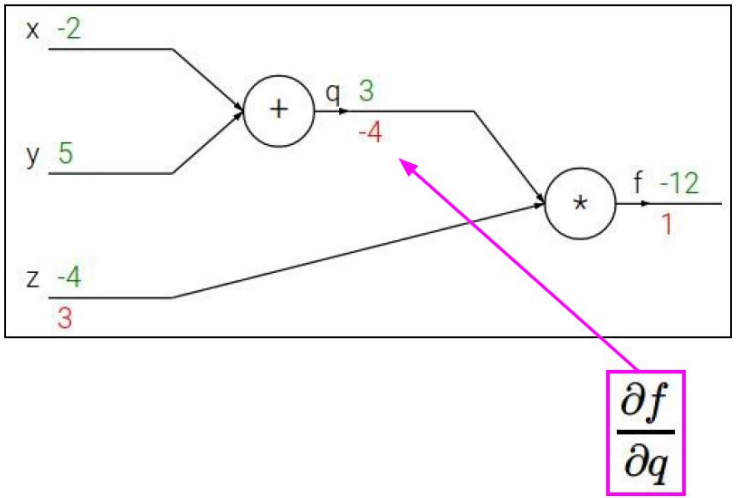
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

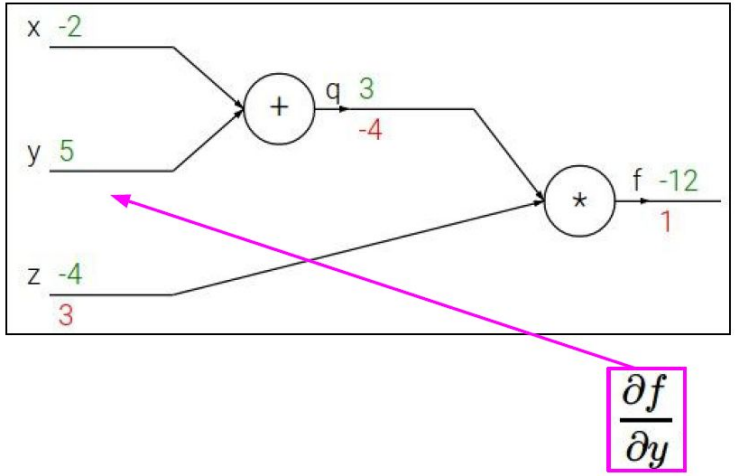
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

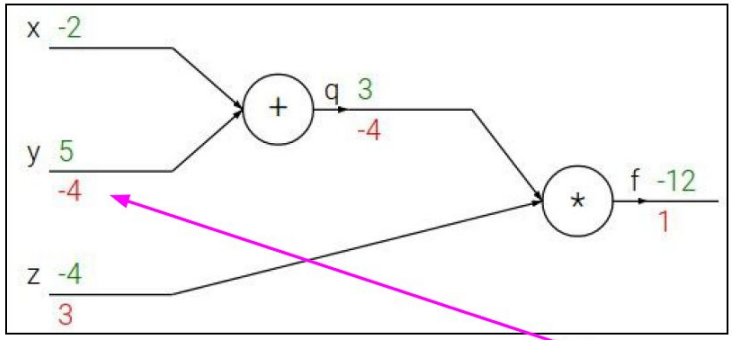
$$f(x, y, z) = (x + y)z$$

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



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

Chain rule:

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

Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

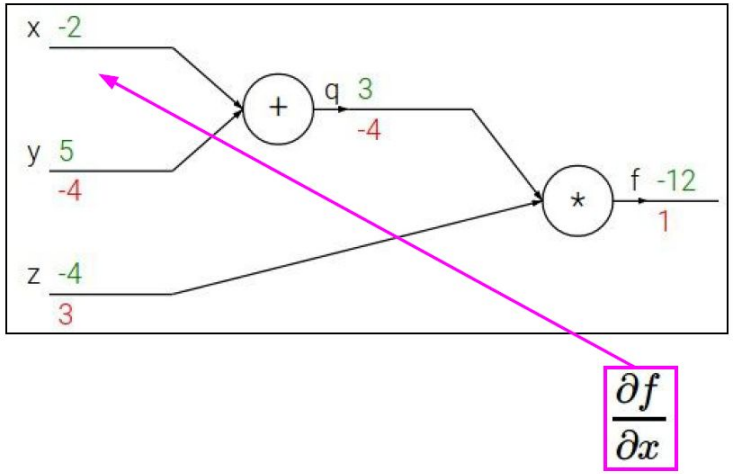
$$f(x, y, z) = (x + y)z$$

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



Gradient Descent

BACKPROPAGATION EXAMPLE

Backpropagation: a simple example

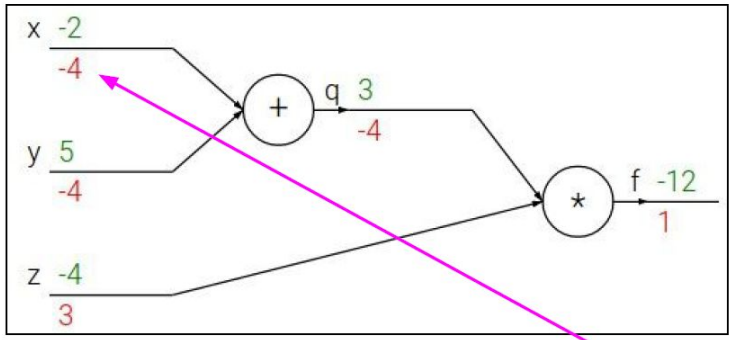
$$f(x, y, z) = (x + y)z$$

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

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

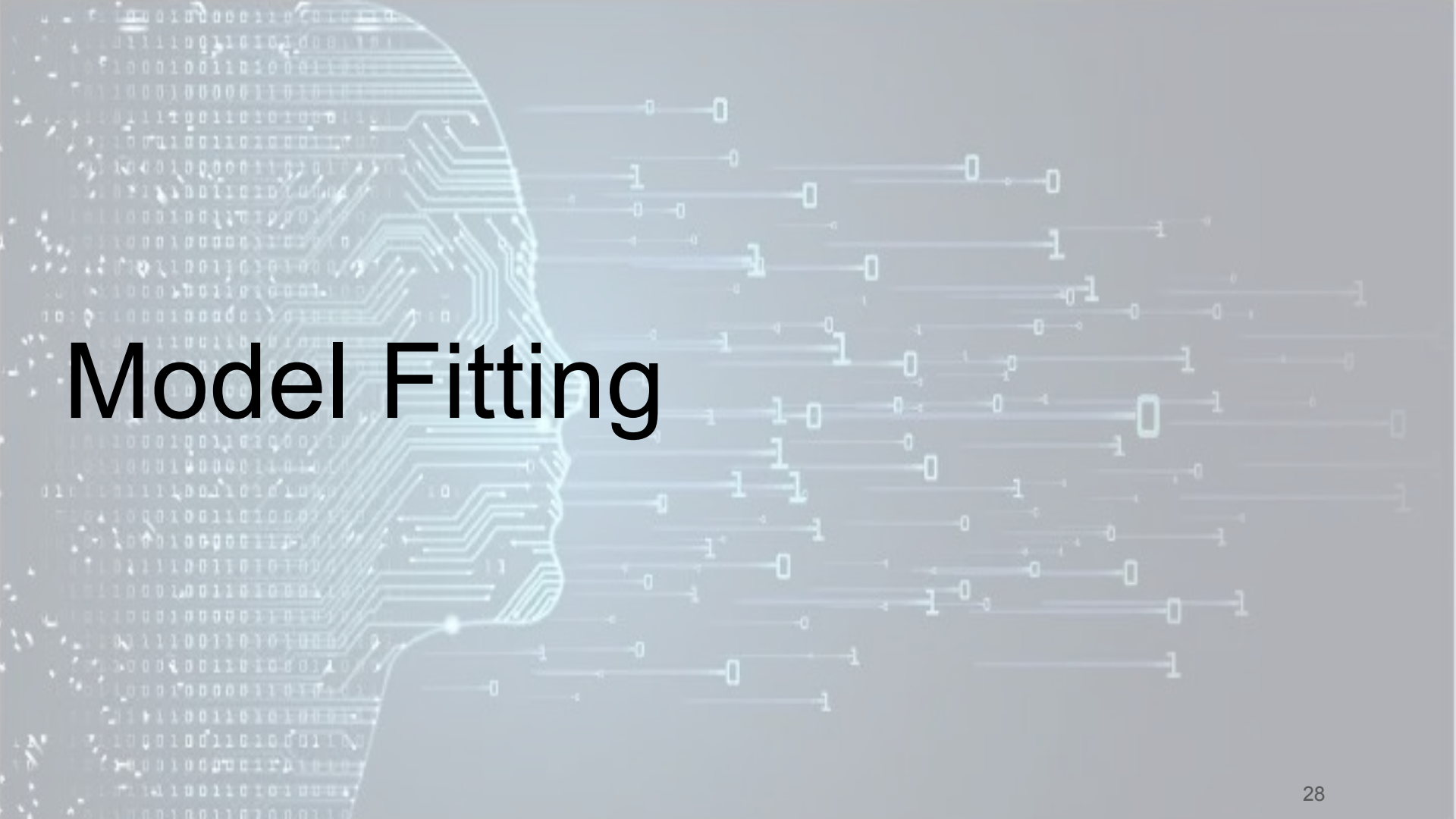
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



Chain rule:

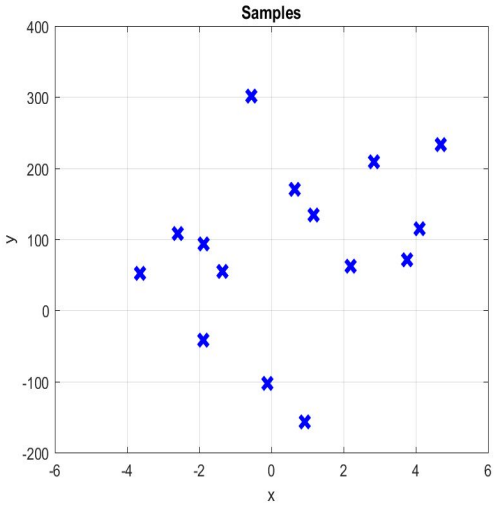
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

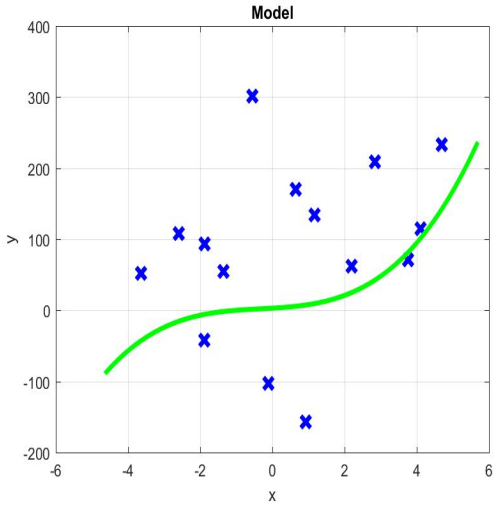


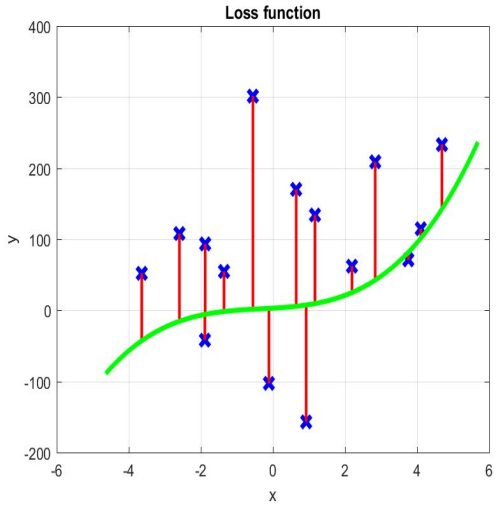
Model Fitting

Model Fitting

INTRODUCTION







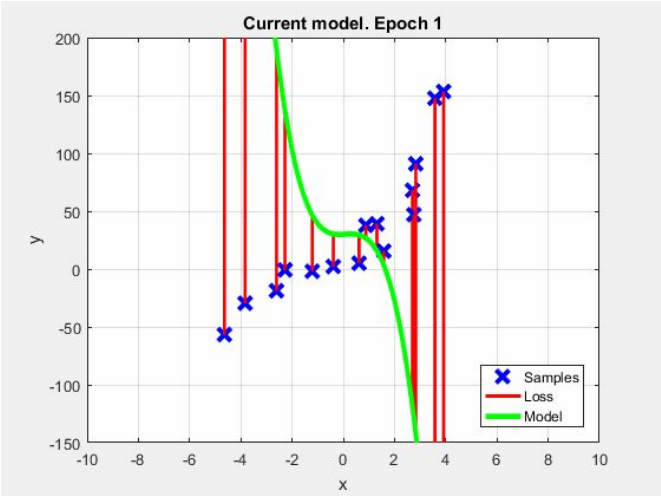
$$[(x_1, y_1),\ (x_i, y_i),\ \ldots, (x_N, y_N)]$$

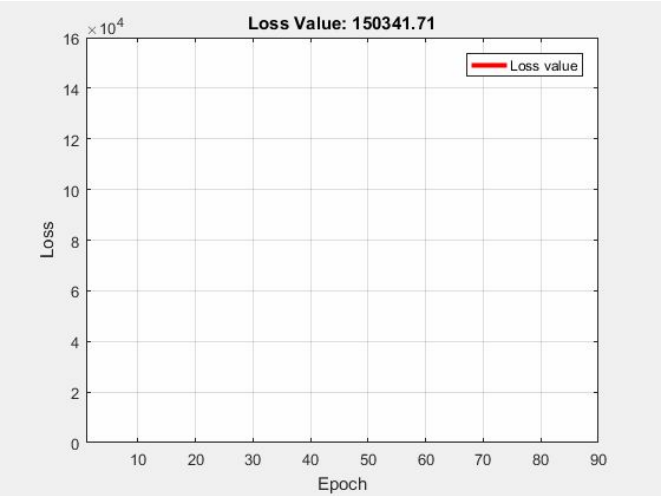
$$f = w_1 x^3 + w_2 x^2 + w_3 x + w_4$$

$$l = \sum_{i=1}^{N} (z_i - y_i)^2$$

Model Fitting

TRAINING



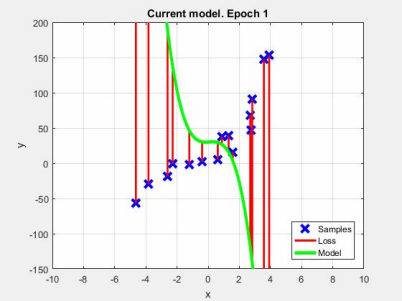


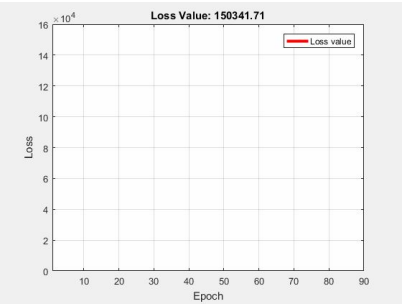
$$y = w_1x^3 + w_2x^2 + w_3x^1 + w_4$$

Model Fitting

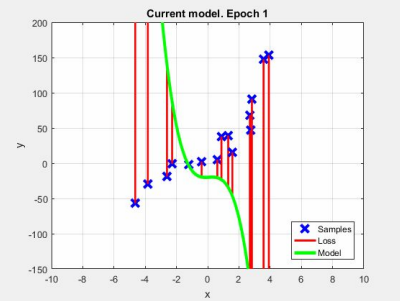
DIFFERENT INITIALIZATIONS

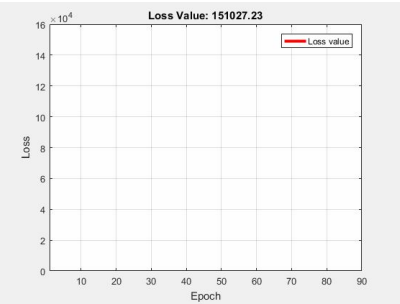
w = [-9, 2, 1, 30]



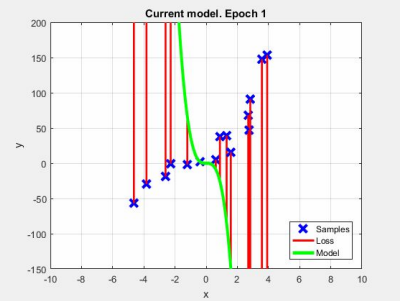


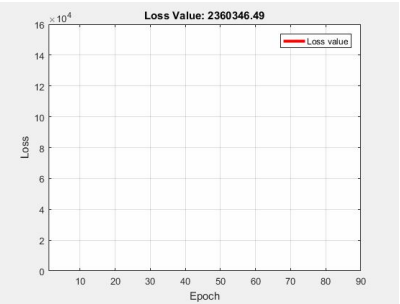
w = [-9, 2, 1, -20]



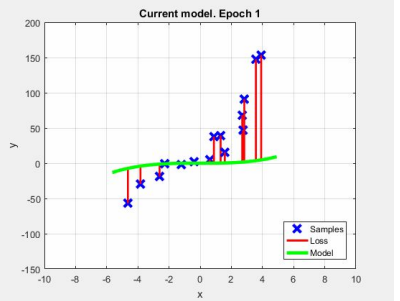


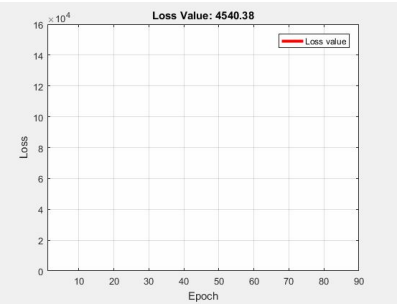
w = [-40, 0, 0, 0]





w = [-0, 0, 0, 0]





$$y = w_1 x^3 + w_2 x^2 + w_3 x^1 + w_4$$

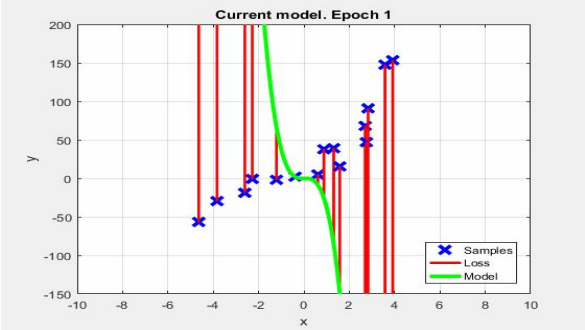
Model Fitting

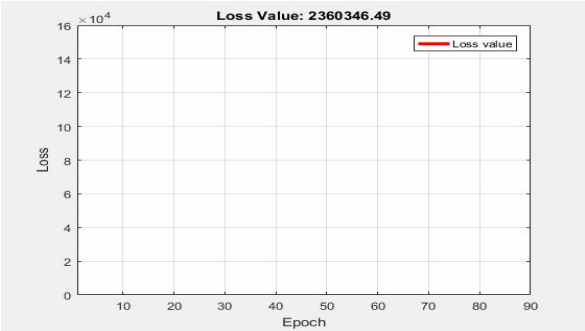
DIFFERENT LEARNING RATE

$y = w_1x^3 + w_2x^2 + w_3x^1 + w_4$

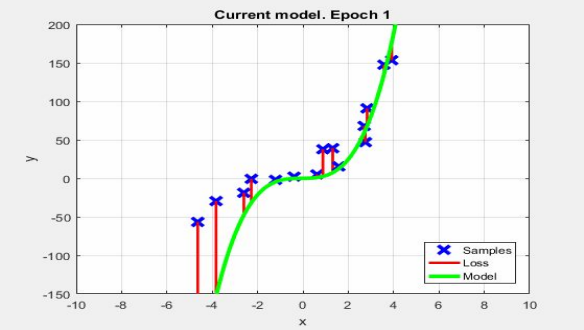
$W = [0,0,0,0]$

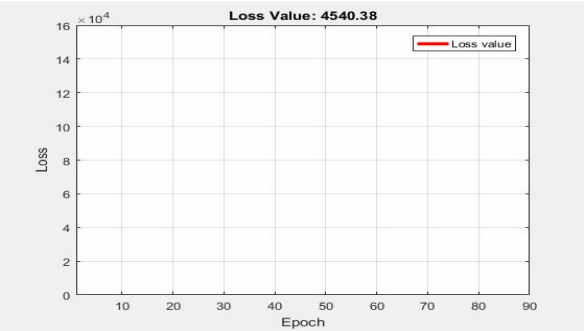
$lr = 2*10^{-5}$





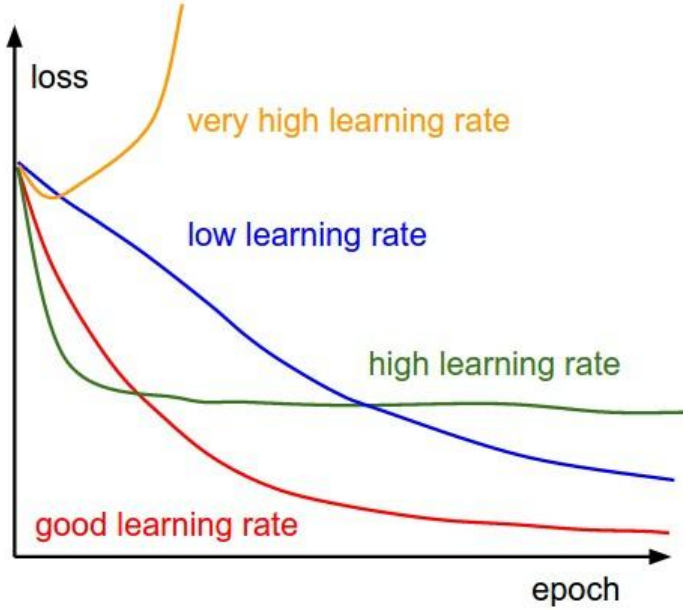
$lr = 7.5*10^{-4}$





Model Fitting

MONITORING THE LOSS

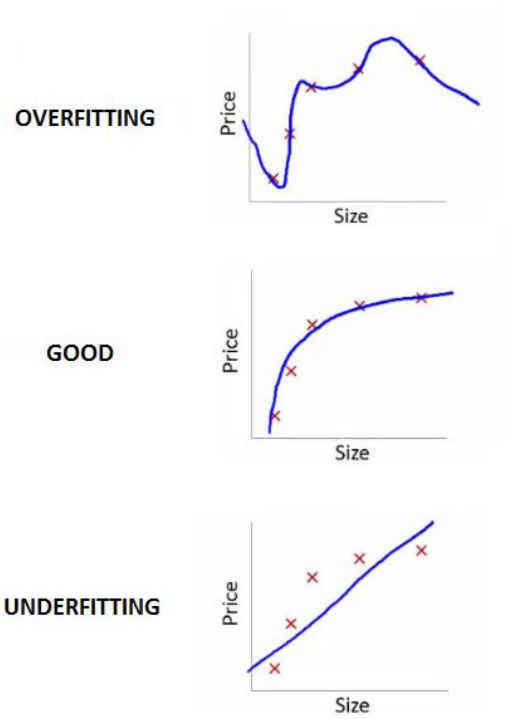


Common Problems

OVERFITTING AND UNDERFITTING

- **Overfitting:** The learned hypothesis fit the training set very well, but fails to generalize to new samples
  - **Solution**
    - Regularization
    - “Simpler” hypothesis
    - Less prone to overfitting
- **Underfitting:** The learned hypothesis don’t fit the training set, so it will also fail to generalize to new samples

Always use a validation set !!!



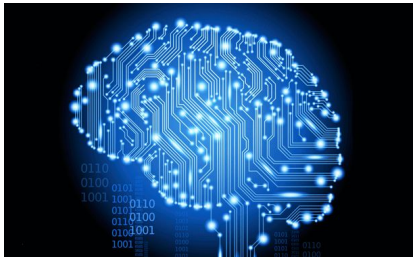
History Of Deep Learning

History of Deep Learning

●THE BIRTH OF NEURAL NETWORKS

W. McCulloch & W. Pitts

- First paper describing brain functions in abstract terms



History of Deep Learning

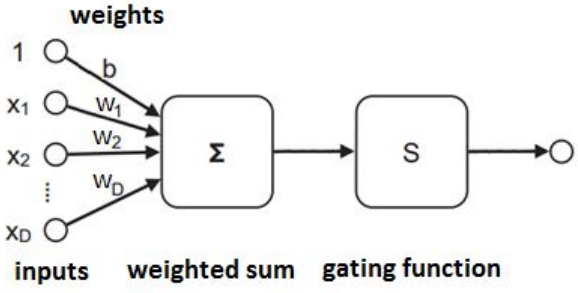
●THE BIRTH OF NEURAL NETWORKS

W. McCulloch & W. Pitts

- First paper describing brain functions in abstract terms

F. Rosenblatt

- First artificial neural network model: the Perceptron

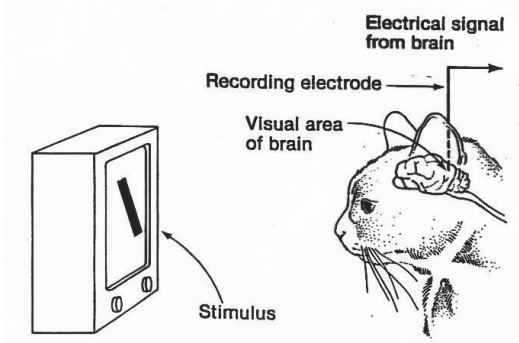


History of Deep Learning

- **DEEP UNDERSTANDING OF VISUAL CORTEX**

D. H. Hubel & T. Wiesel

- Discovered a hierarchy of feature detectors in cat's visual cortex
  - Higher level features respond to patters of activation in lower level cells, and propagate activation upwards to still higher level cells

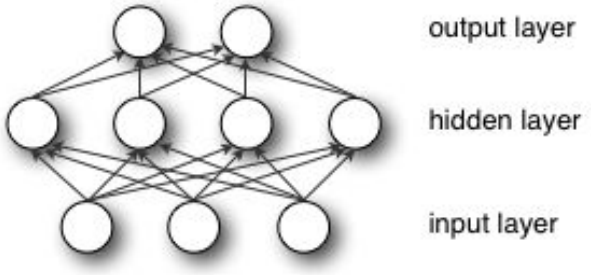


History of Deep Learning

- **MOVING CLOSER TO DEEP LEARNING**

J. Hinton – Y. LeCun

- Multi-layer networks
- Back-propatation



History of Deep Learning

●MOVING CLOSER TO DEEP LEARNING

J. Hinton – Y. LeCun

- Multi-layer networks
- Back-propatation

But it’s still too early

- Hard to train models
- Insufficient computational resources
- Small training sets

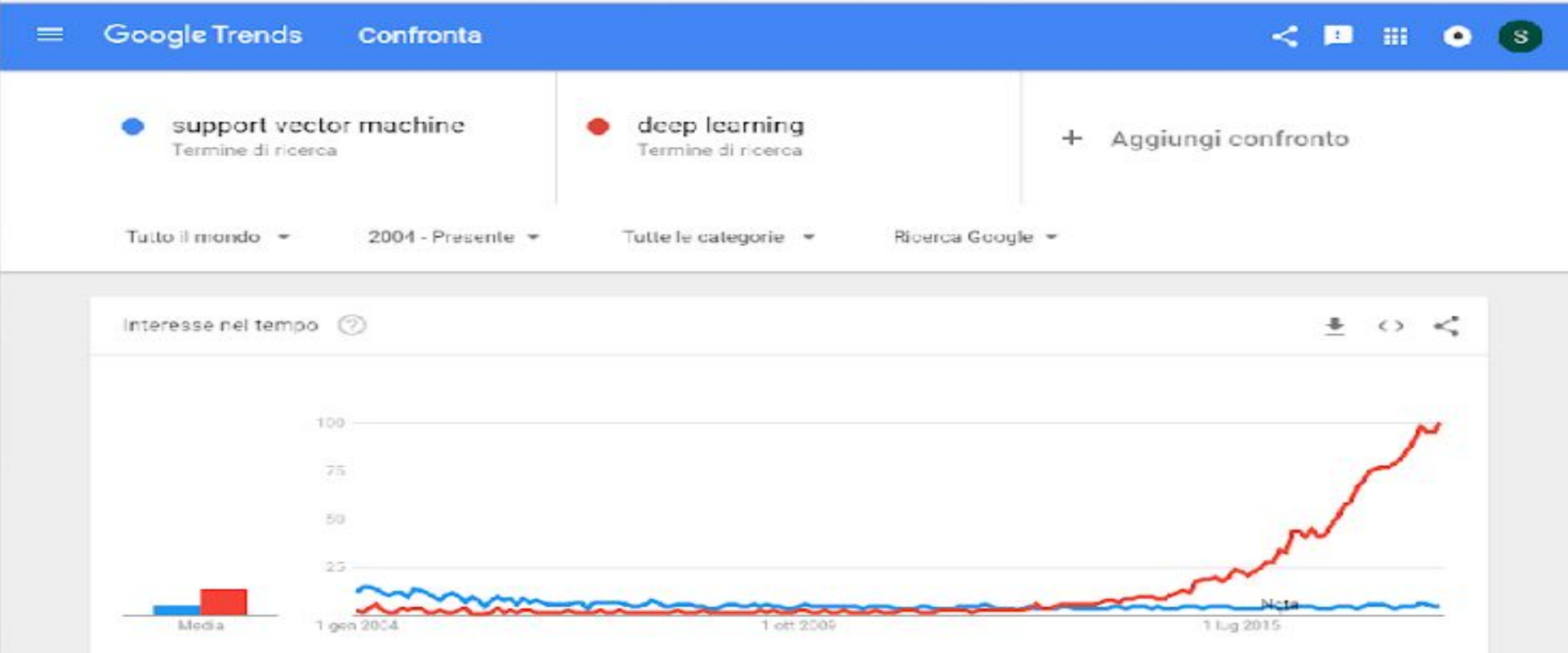
History of Deep Learning

- **DEEP LEARNING BACK ON STAGE!**

- Better designs for modeling and training (normalization, non-linearity, dropout)
- New development of computer architectures
  - Graphical Processing Units (GPU)
  - Multi-core computer systems
- Large scale databases

History of Deep Learning

- **DEEP LEARNING BACK ON STAGE!**



History of Deep Learning

- **DEEP LEARNING IS NOW EVERYWHERE..**

History of Deep Learning

- **.. AND YOU USE IT EVERYDAY**

Google

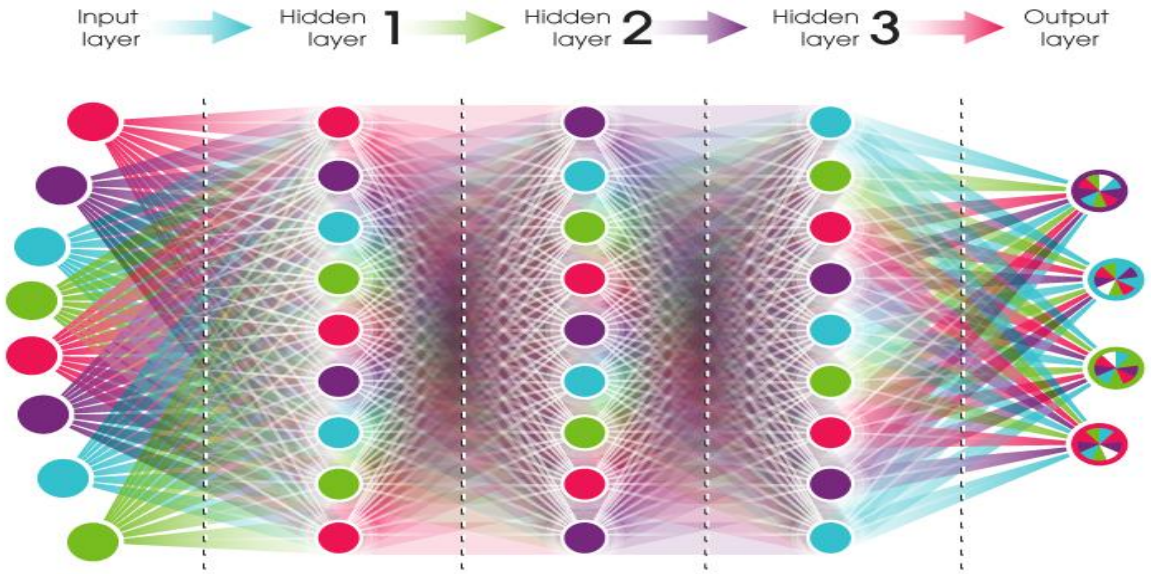
LinkedIn

Quora

History of Deep Learning

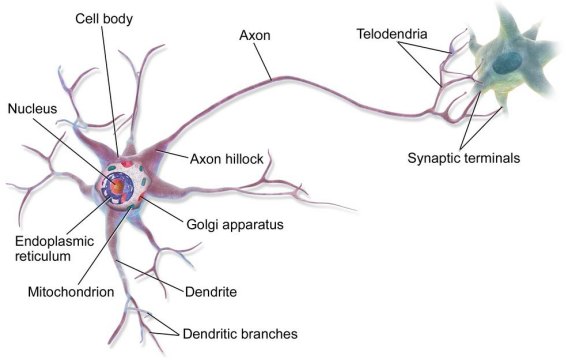
●BUT.. WHAT IS EXACTLY DEEP LEARNING?

- Is a new area of Machine Learning research (ML), with the objective of moving ML closer to Artificial Intelligence
- It is about learning multiple levels (a hierarchy) of representations and abstractions

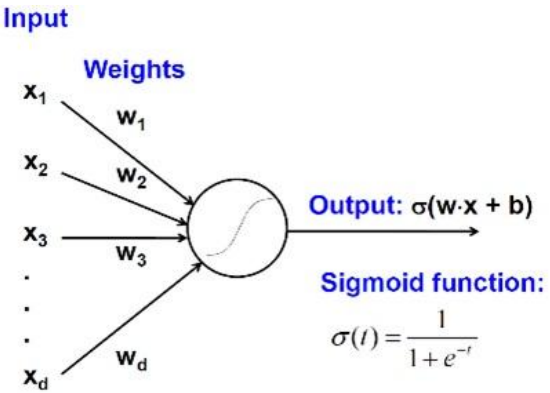


Neural Networks

Introduction

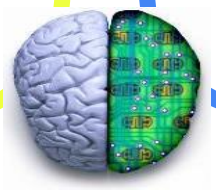


A biological neuron



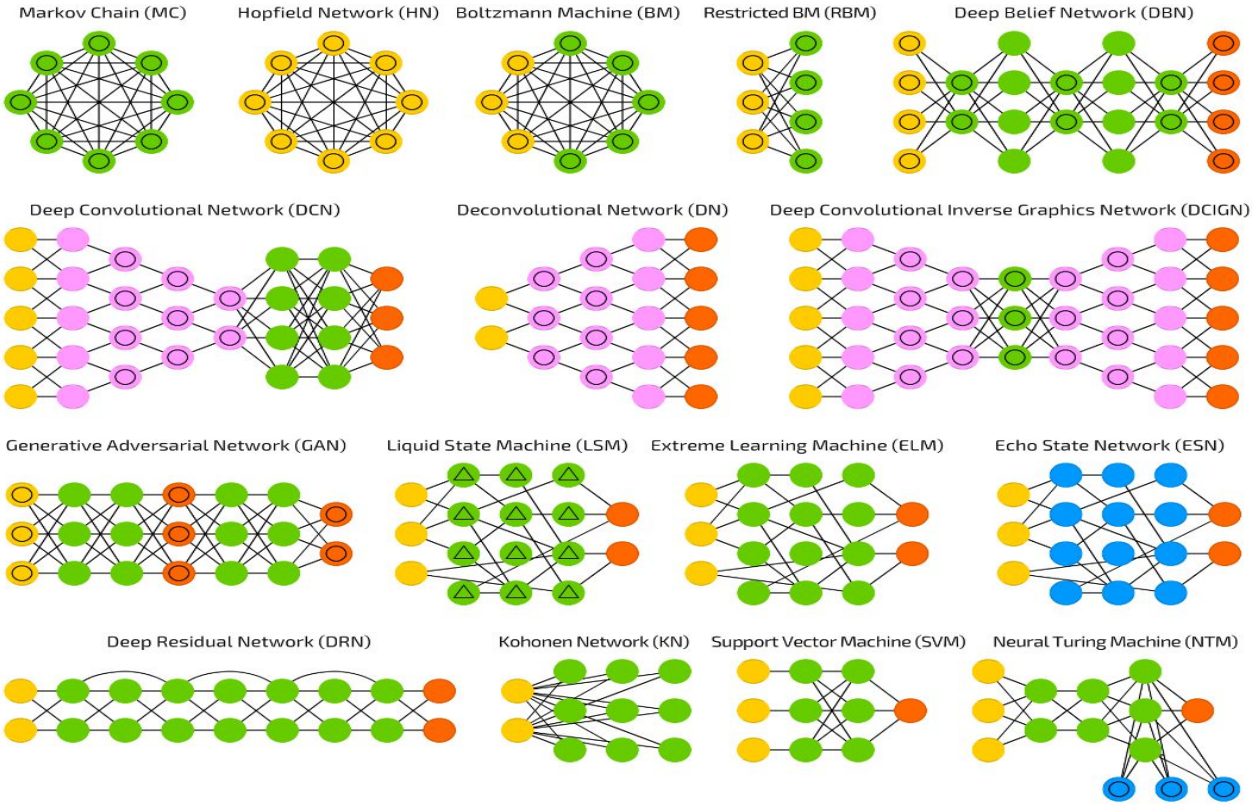
An artificial neuron (Perceptron)

➢ A *linear* classifier

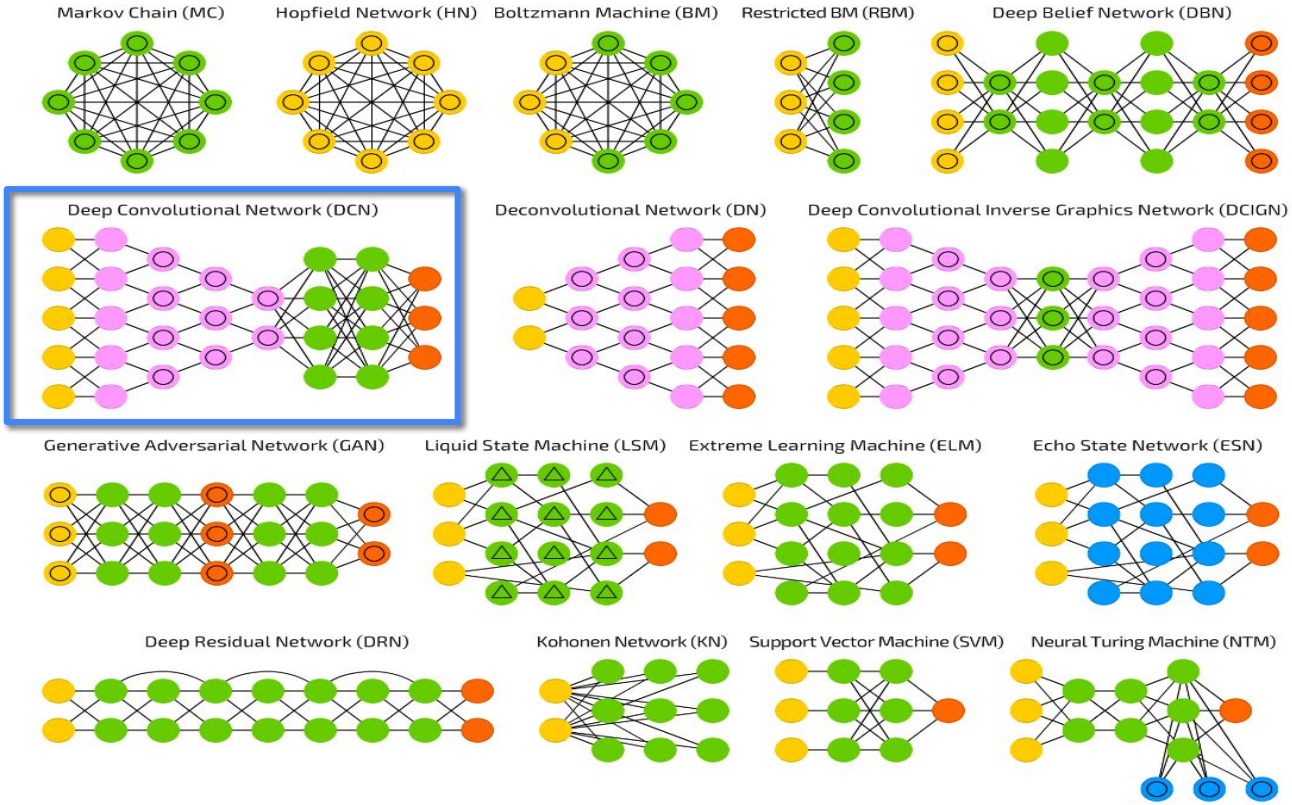


Slide credit: J. Huang

Types of Neural Networks



Types of Neural Networks

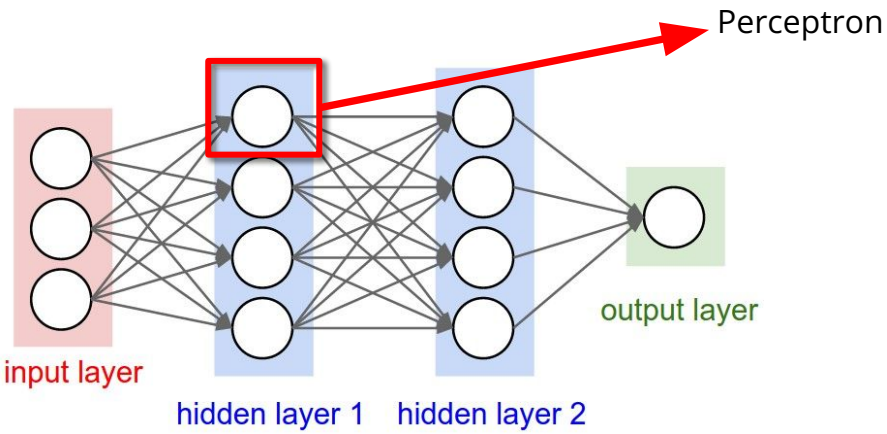


Characteristics

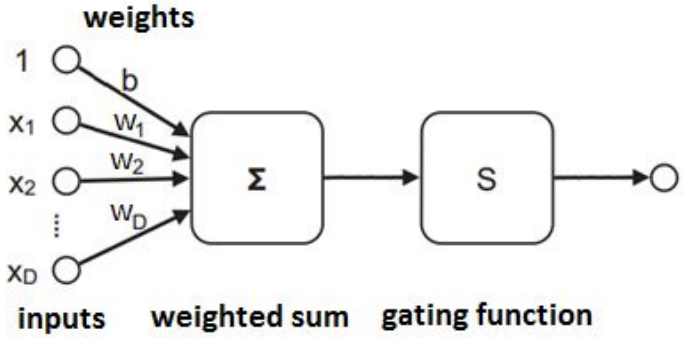
Neural networks can be seen as directed acyclic graph (DAG):

- set of vertices and edges
- each edge connects two vertices
- following any direction it will never form a closed loop

Non-linear classifiers out of perceptrons created with multi-layer neural network



Closeup of a neuron

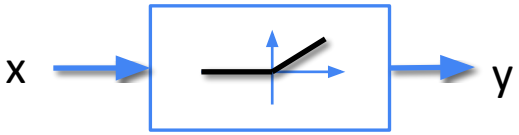


$$y = gating\_function(x_1 w_1 + x_2 w_2 + \cdots + x_N w_N + b)$$

weighted sum of inputs

bias

Activation functions (a.k.a. gating functions)

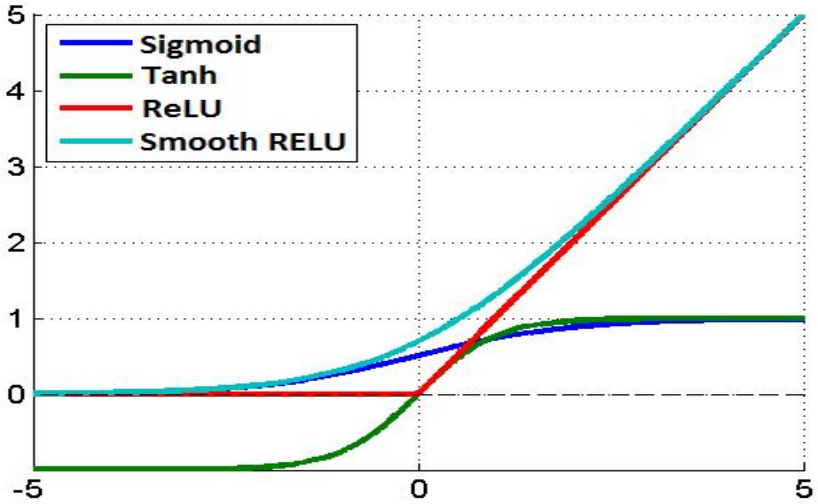


$$y = \frac{1}{1 + e^{-x}}$$ Sigmoid

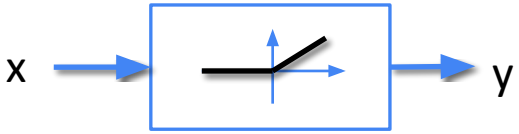
$$y = \tanh x$$ Hyperbolic tangent

$$y = \max\{0, x\}$$ Rectified Linear Unit (ReLU)

$$y = \log \quad (1 + e^{x})$$ Smooth ReLU



Rectified Linear Unit (ReLU)



$$y = \max\{0, x\}$$

Rectified Linear Unit (ReLU)

Problem: this function is not derivable in 0!

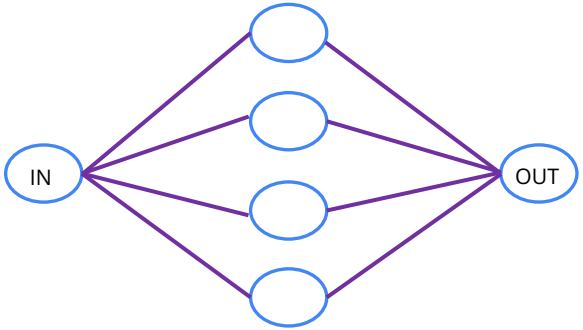
- What can we do?
  - we arbitrarily assign left or right derivative

Why activation functions are needed?

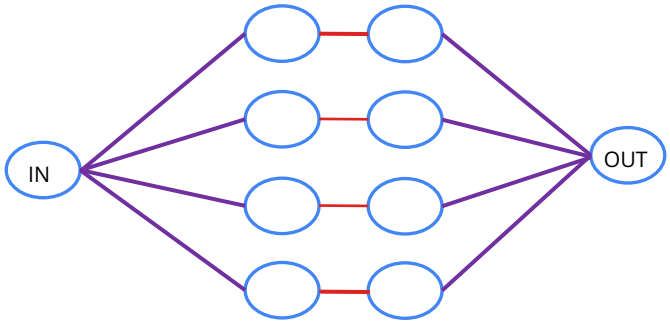
Let's compare two architectures

- both composed of two linear layers
- the second interleaved with Rectified Linear Unit

WITHOUT RELU

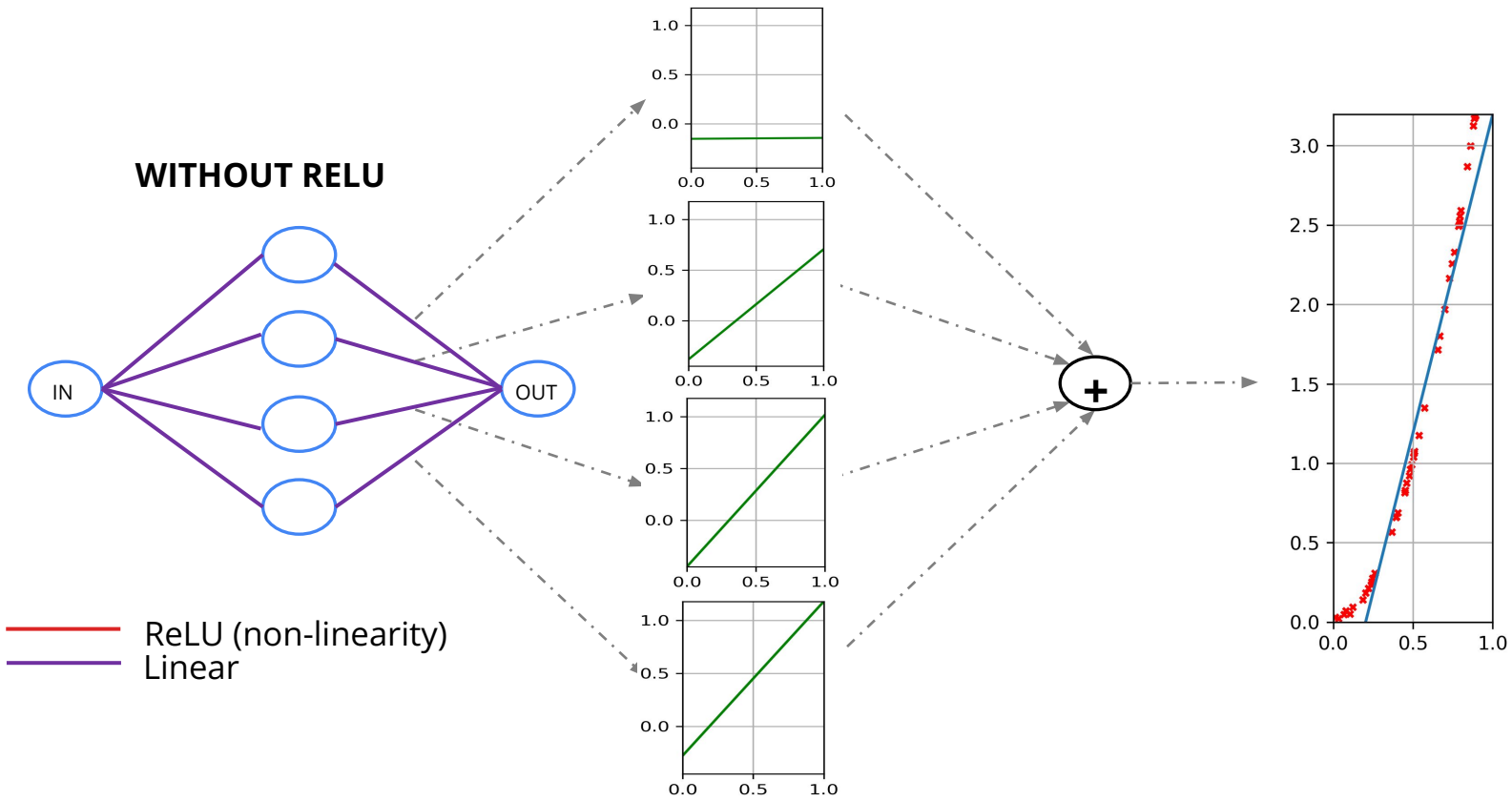


WITH RELU

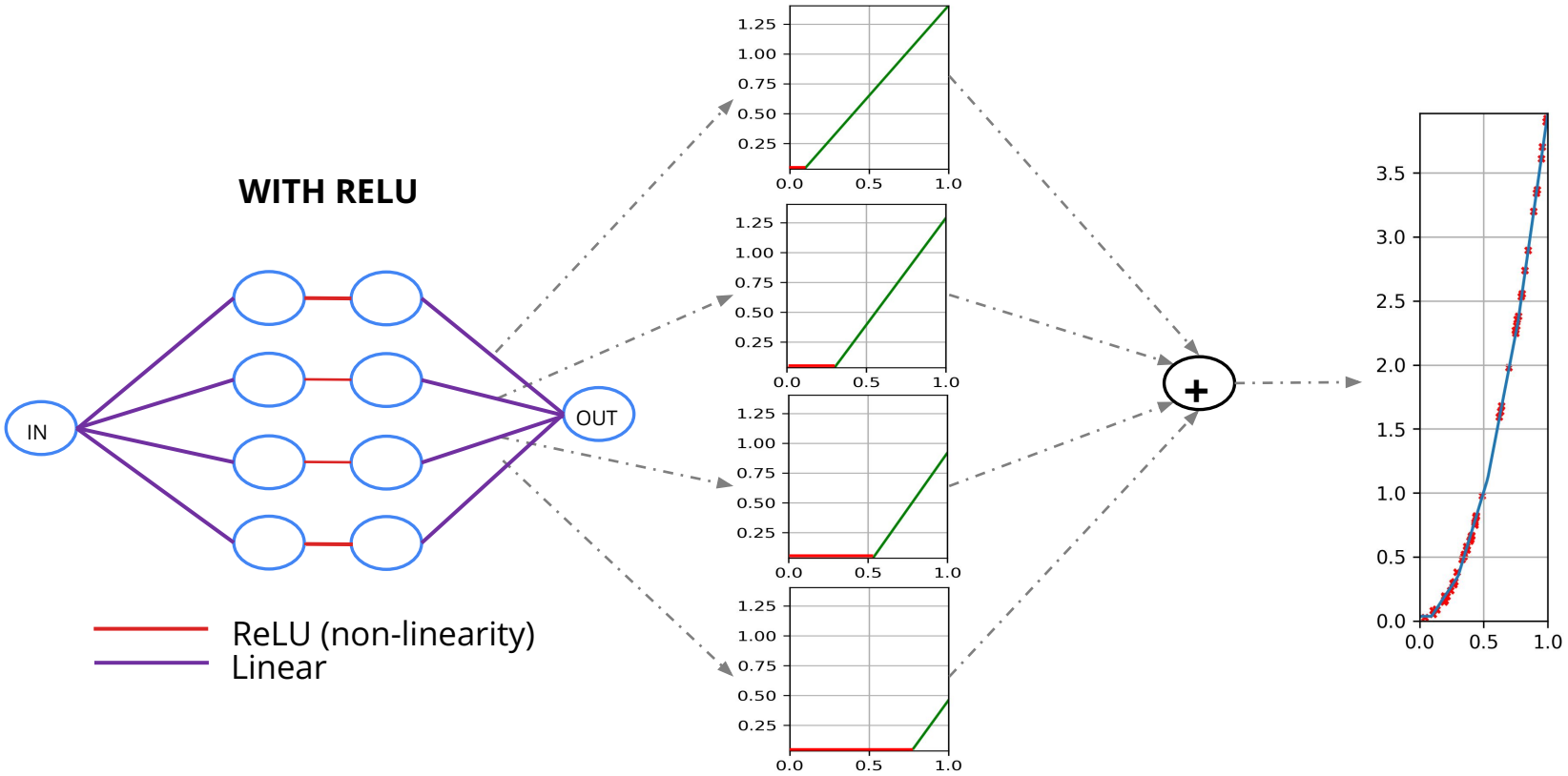


ReLU (non-linearity)
Linear

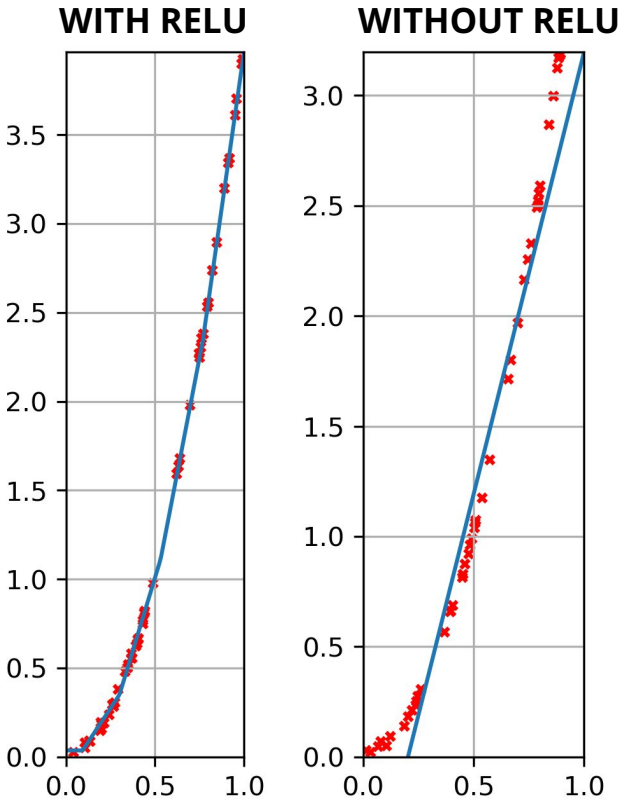
Example without ReLU



Example with ReLU

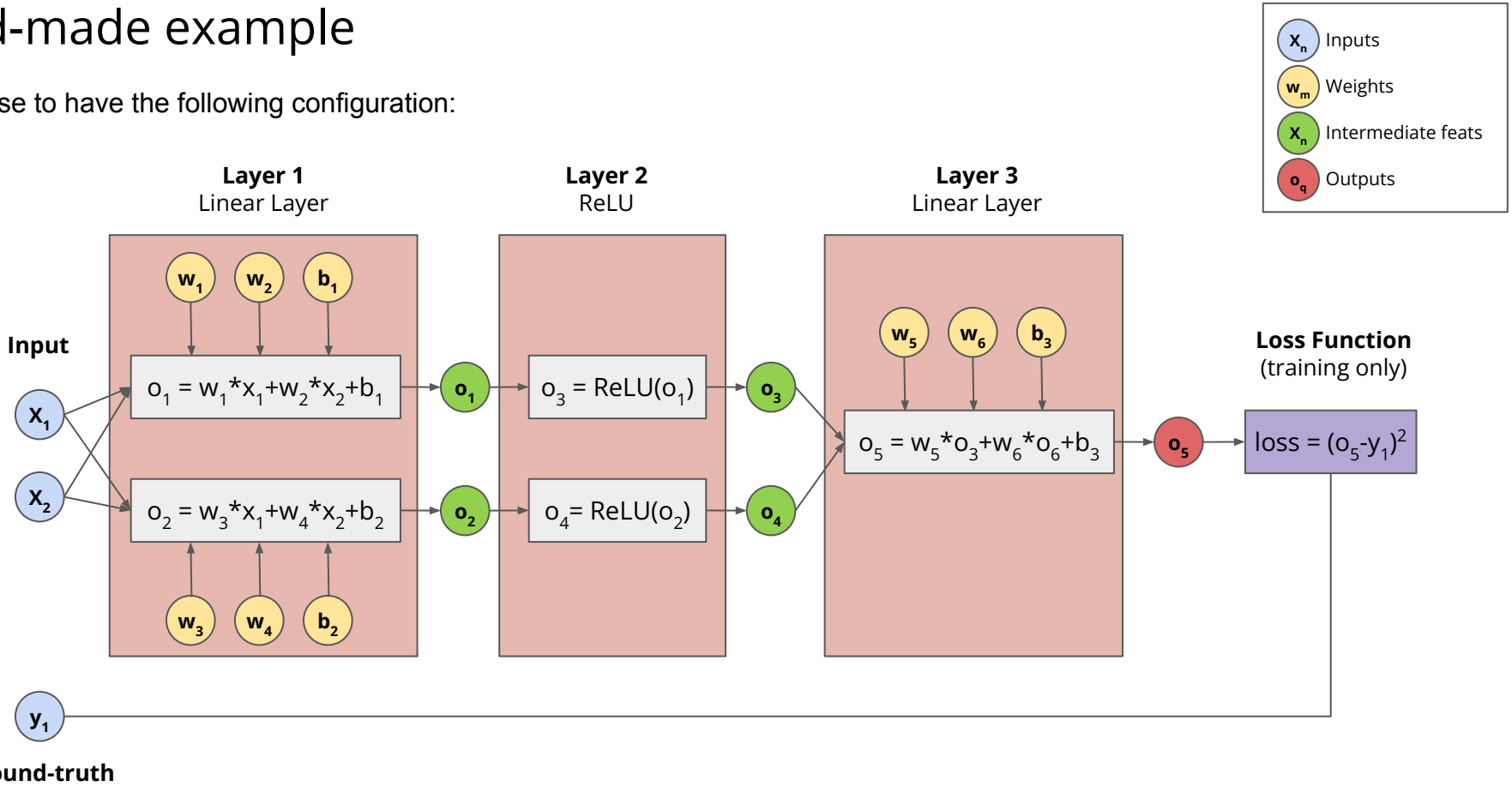


Comparison among the two examples



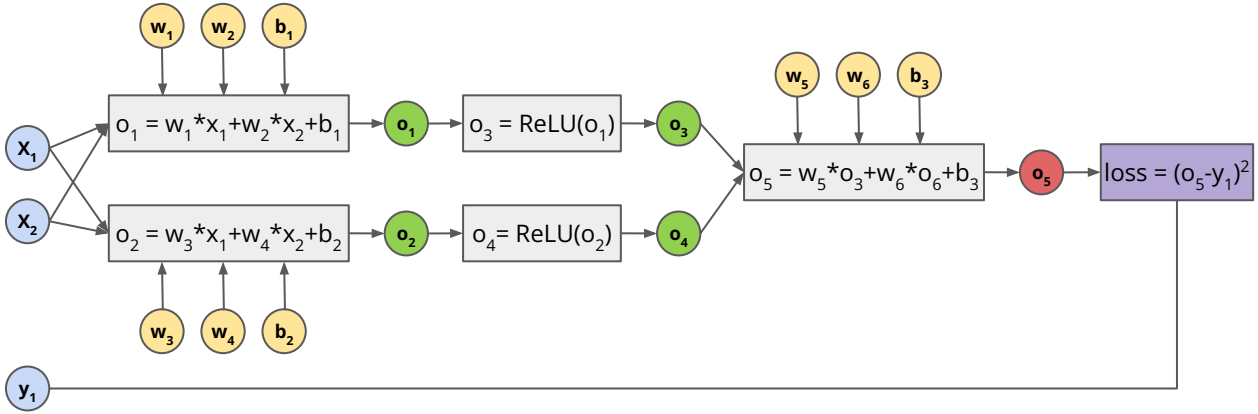
Hand-made example

Suppose to have the following configuration:



Compute manually the update of w_1 and w_5

Hand-made example



Given the following inputs / weights:

x_1	x_2	y_1
2	1	5

w_1	w_2	b_1
0	2	3

w_3	w_4	b_2
-4	-1	1

w_5	w_6	b_3
-2	0	6

Supposing a learning rate of 10^{-3}, compute:

- the final output o_5 and the loss value through forward propagation
- the new values of w_5 and w_1 using back propagation

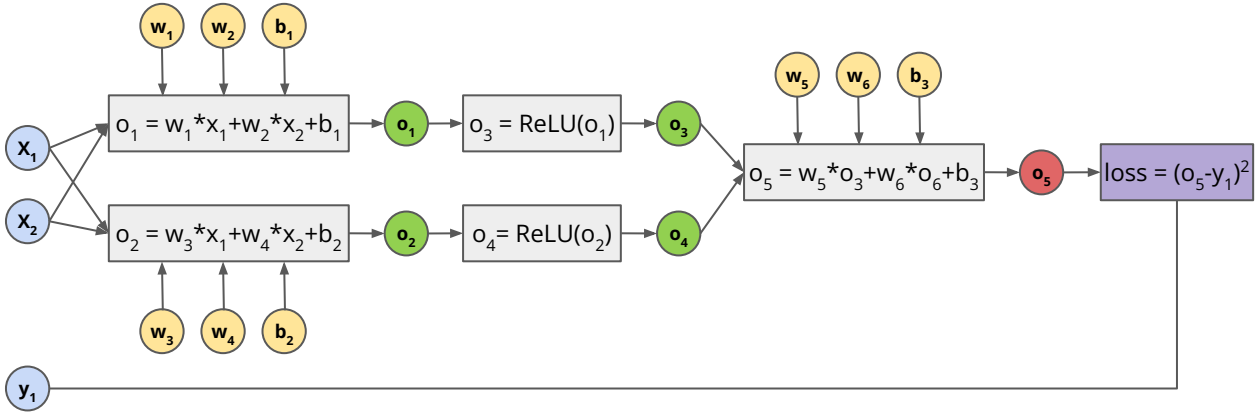
Hints:

- perform fwd propagation
- compute $\frac{\partial Loss}{\partial o_5}$
- compute $\frac{\partial Loss}{\partial w_5} = \frac{\partial Loss}{\partial o_5} \cdot \frac{\partial o_5}{\partial w_5}$
- compute $\frac{\partial Loss}{\partial o_3} = \frac{\partial Loss}{\partial o_5} \cdot \frac{\partial o_5}{\partial o_3}$
- compute $\frac{\partial Loss}{\partial o_1} = \frac{\partial Loss}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_1}$
- compute $\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_1}$

- use $\frac{\partial Loss}{\partial w_5}$ to update w_5
- use $\frac{\partial Loss}{\partial w_1}$ to update w_1
- remember to use the GD update rule:

$$w_n = w_n - lr \cdot \frac{\partial Loss}{\partial w_n}$$

Hand-made example (solution)



Given the following inputs / weights:

x_1	x_2	y_1
2	1	5

w_1	w_2	b_1
0	2	3

w_3	w_4	b_2
-4	-1	1

w_5	w_6	b_3
-2	0	6

Supposing a learning rate of 10^{-3}, compute:

- the final output o_5 and the loss value through forward propagation
- the new values of w_5 and w_1 using back propagation

Hints:

- perform fwd propagation $o_1 = 5, o_2 = -8, o_3 = 5, o_4 = 0, o_5 = -4, loss = 81$
- compute $\frac{\partial Loss}{\partial o_5} = 2(o_5 - y_1) = -18$
- compute $\frac{\partial Loss}{\partial w_5} = \frac{\partial Loss}{\partial o_5} \cdot \frac{\partial o_5}{\partial w_5} = -18 \cdot o_3 = -90$
- compute $\frac{\partial Loss}{\partial o_3} = \frac{\partial Loss}{\partial o_5} \cdot \frac{\partial o_5}{\partial o_3} = -18 \cdot w_5 = 36$
- compute $\frac{\partial Loss}{\partial o_1} = \frac{\partial Loss}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_1} = 36 \cdot 1 = 36$
- compute $\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_1} = 36 \cdot x_1 = 72$

- 1,91

$$w_5 = w_5 - lr \cdot \frac{\partial Loss}{\partial w_5} = -2 + 90 \cdot 10^{-3} = -2.09$$

$$w_1 = w_1 - lr \cdot \frac{\partial Loss}{\partial w_1} = 0 - 72 \cdot 10^{-3} = -0.072$$

Losses

- Depend from the task
- guide the training process
- must be:
  - continuous
  - have only one minimum / maximum

- Examples of losses:

Mean Squared Error (MSE)

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2.$$

Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

Quiz time!

Quiz time!

Go to the website:

PollEv.com/flaviopiccoli014

Which of these tasks is not a regression?

Estimate bus arrival based on angle of arrival (AoA) and delay of the radio signal emitted by the bus

Estimate age through face analysis

Estimate timezone from RGB picture

Estimate house price from audio recording (supposing there is a correlation among house price and noise)

Which properties must have a loss function?

It must be derivable in the entire domain

It must be continuous in the entire domain

It must be derivable in the entire domain and have only one minimum/maximum

It must be continuous in the entire domain and have at least one minimum/maximum

Why is learning rate needed?

Helps to avoid over fitting/underfitting

It is useful to mitigate the magnitude of the gradient

It is required to speed up the training

It helps to find the direction of the minimum

What is the definition of overfitting?

The learned hypotesis does not fit the training set and also fails to generalize on new data

The learned hypotesis explains very well the test set but not the training set

The learned hypotesis fits the training set very well but fails to generalize on new data

The learned hypotesis does not fit neither training and test set

What is the definition of underfitting?

The learned hypotesis does not fit the training set and also fails to generalize on new data

The learned hypotesis explains very well the test set but not the training set

The learned hypotesis fits the training set very well but fails to generalize on new data

The learned hypotesis does not fit neither training and test set

Which of these functions can be used as a loss? Suppose that e is the error ($e = prediction - groundtruth$)

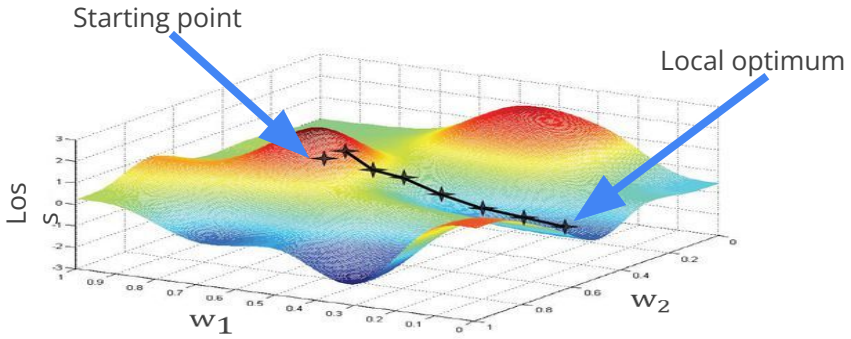
$$l = e^3$$
$$l = log(e)$$
$$l = sin(e)$$
$$l = 1/e$$
$$l = 5e^2 - 2$$

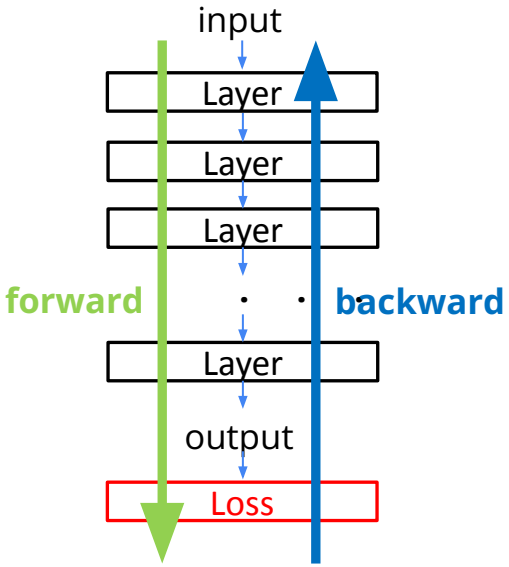
Training Procedure

Training procedure

Pseudocode:

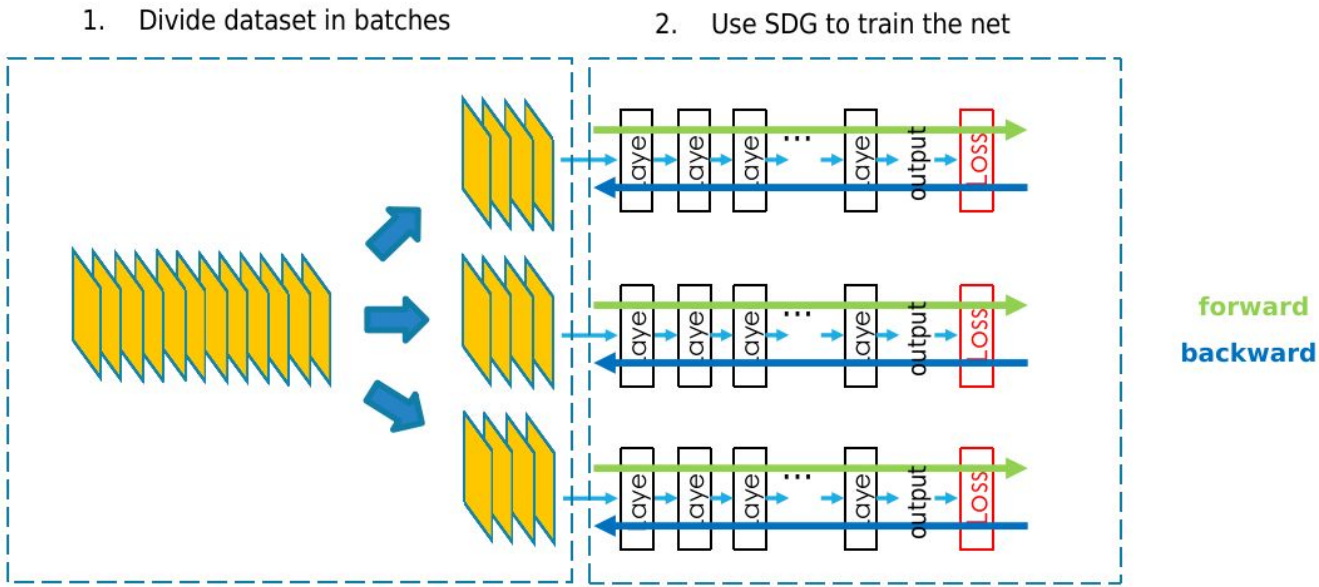
- Initialize weights
- while not convergency (or max attemps reached)
  1. Do forward propagation
  2. Do backward propagation
  3. Update weights





Division in minibatches

All images should be used at the same time. This is often not possible.



Training procedure

The pseudocode corresponding to the full training procedure is:

1. initialize dataset (load filenames and divide in batches)
2. define predictive model
   a. initialize weights
3. define optimizer

Initial setup

4. for each epoch (or until convergence)
   a. for each batch in the dataset:
      i. compute forward propagation to get output
      ii. compute loss
      iii. compute gradients (back-propagation)
      iv. update weights of model

1-epoch training

   b. validate model
   c. if best model, save it

testing and checkpointing

Weights initialization

Weights initialization

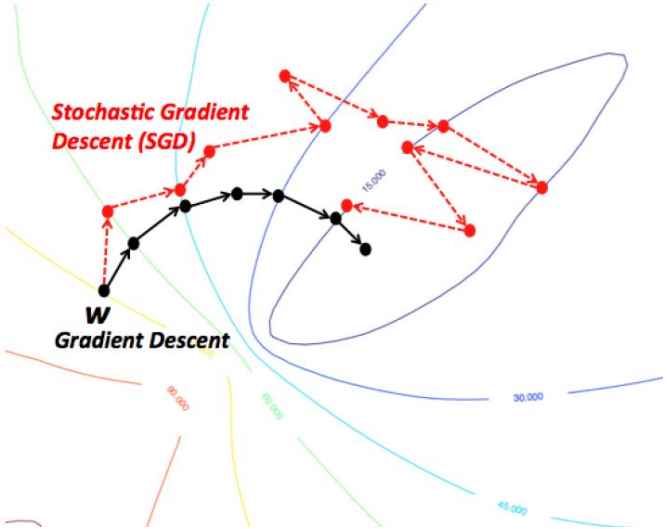
https://www.deeplearning.ai/ai-notes/initialization/index.html

Optimizers

Stochastic gradient descent (SGD)

- It is an approximation of the gradient descent (GD)
- Much more fluctuation w.r.t. gradient descent
- the smaller is the batch, the higher will be the fluctuation

$$\theta = \theta - \eta * \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

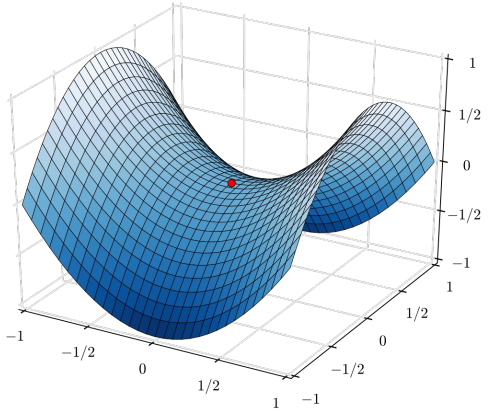


Momentum

- SGD becomes very slow:
  - in saddle points
  - near the optimum

$$\theta_{t+1} = \theta_t - \eta g$$

in these points, g tends to 0



- To overcome this problems, weight update is done through a velocity term
- v is the direction and speed at which the parameter should be tweaked

$$v_{t+1} = \alpha v_t - \eta g$$

$$\theta_{t+1} = \theta_t + v_{t+1}$$

the higher is alpha, the higher will be the momentum

Nesterov Accelerated Gradient

- The difference of Nesterov Accelerated Gradient and Momentum is the gradient computation phase
- With momentum is:

$$g = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \mathcal{L}(x^{(i)}, y^{(i)}, \theta_t)$$

- With Nesterov Accelerated Gradient, velocity is also applied to the parameters θ:

$$\tilde{\theta} = \theta_t + \alpha v_t$$ ← **ad-interim parameters**

- The gradients, the velocity and the update rule are then:

$$g_{NAG} = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \mathcal{L}(x^{(i)}, y^{(i)}, \tilde{\theta})$$

$$v_{t+1} = \alpha v_t - \eta g_{NAG}$$

$$\theta_{t+1} = \theta_t + v_{t+1}.$$

Adaptive Gradient Algorithm (AdaGrad)

- Converges faster than SGD
- Uses an **independent *learning rate* for each parameter**
- Use higher *lr* for parameters with more sparse distribution, and slower lr for parameters with denser distribution

Parameter update:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\varepsilon I + diag(G_t)}} \cdot g_t$$

η = the initial learning rate
ε = small quantity used to avoid the division of zero
I = the identity matrix
G_t is the gradient estimate in time-step t

Gradient estimate in time-step t:

$$g_t = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathcal{L}(x^{(i)}, y^{(i)}, \theta_t)$$

Gt matrix

$$G_t = \sum_{\tau=1}^{t} g_\tau g_\tau^\top$$

G_t is the sum of the outer product of the gradients until time-step *t*

Adaptive subgradient methods for online learning and stochastic optimization
Duchi, J., Hazan, E., & Singer, Y. (2011).
Journal of machine learning research

Adaptive Gradient Algorithm (AdaGrad)

- Expanding the formula we have:

$$\begin{bmatrix} \theta_{t+1}^{(1)} \\ \theta_{t+1}^{(2)} \\ \vdots \\ \theta_{t+1}^{(m)} \end{bmatrix} = \begin{bmatrix} \theta_{t}^{(1)} \\ \theta_{t}^{(2)} \\ \vdots \\ \theta_{t}^{(m)} \end{bmatrix} - \eta \left(\begin{bmatrix} \varepsilon & 0 & \cdots & 0 \\ 0 & \varepsilon & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \varepsilon \end{bmatrix} + \begin{bmatrix} G_t^{(1,1)} & 0 & \cdots & 0 \\ 0 & G_t^{(2,2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G_t^{(m,m)} \end{bmatrix} \right)^{-1/2} \cdot \begin{bmatrix} g_t^{(1)} \\ g_t^{(2)} \\ \vdots \\ g_t^{(m)}, \end{bmatrix} \tag{4}$$

The update rule of AdaGrad becomes:

$$\begin{bmatrix} \theta_{t+1}^{(1)} \\ \theta_{t+1}^{(2)} \\ \vdots \\ \theta_{t+1}^{(m)} \end{bmatrix} = \begin{bmatrix} \theta_{t}^{(1)} \\ \theta_{t}^{(2)} \\ \vdots \\ \theta_{t}^{(m)} \end{bmatrix} - \begin{bmatrix} \frac{\eta}{\sqrt{\varepsilon + G_t^{(1,1)}}} g_t^{(1)} \\ \frac{\eta}{\sqrt{\varepsilon + G_t^{(2,2)}}} g_t^{(2)} \\ \vdots \\ \frac{\eta}{\sqrt{\varepsilon + G_t^{(m,m)}}} g_t^{(m)}. \end{bmatrix}$$

For reference, this is the one of SGD:

$$\begin{bmatrix} \theta_{t+1}^{(1)} \\ \theta_{t+1}^{(2)} \\ \vdots \\ \theta_{t+1}^{(m)} \end{bmatrix} = \begin{bmatrix} \theta_{t}^{(1)} \\ \theta_{t}^{(2)} \\ \vdots \\ \theta_{t}^{(m)} \end{bmatrix} - \begin{bmatrix} \eta g_t^{(1)} \\ \eta g_t^{(2)} \\ \vdots \\ \eta g_t^{(m)}. \end{bmatrix}$$

Adaptive subgradient methods for online learning and stochastic optimization
Duchi, J., Hazan, E., & Singer, Y. (2011).
Journal of machine learning research

AdaDelta

- It's an extension of AdaGrad
- Instead of accumulating all past squared gradients, it restricts the window of accumulated past gradients to a fixed size
- This is done with a decaying average of all past squared gradients

$$E\left[g^2\right]_t = \gamma E\left[g^2\right]_{t-1} + (1-\gamma)\, g_t^2$$

- The parameter update is then:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

ADAM

- It is the most used optimizer
- it combines the advantages of previous optimizers
- The aggregate of gradients m_t at time t is:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

- The velocity v_t at time t is:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t$$

- More details about this optimizer can be found at this link

Summing up

Optimizer	Batch wise	Momentum	Independent learning rate for each param.	Details
GD	X	X	X	Impractical as it requires all samples at once
SGD	V	X	X	Very slow in saddle points and near optimum
Momentum	V	V	X	Uses velocity to speed up on low-gradient areas
Nesterov	V	V	X	Velocity is also applied to weights
AdaGrad	V	V	V	Higher lr for parameters with more sparse distribution
AdaDelta	V	V	V	It restricts the window of accumulated past gradients to a fixed size through a decay term
ADAM	V	V	V	Decay is also applied to the velocity term

Comparison

https://imgur.com/a/Hqolp#NKsFHJb