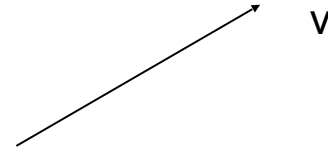


DEEP LEARNING
part II: neural networks

marco milanesio

About tensors

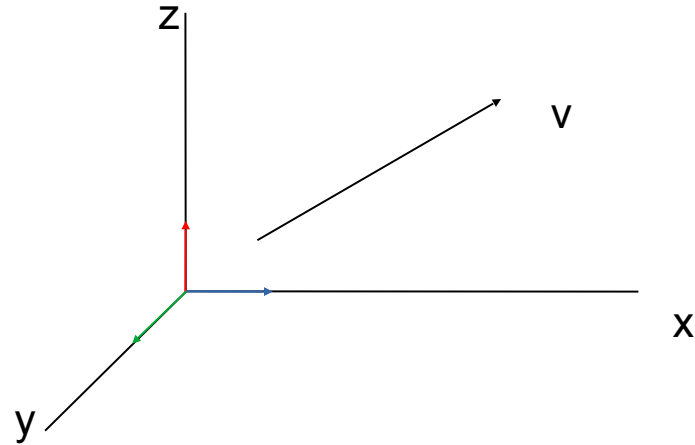
- Vector



- Usually, an **arrow**
- Something with a **magnitude** and a **direction**
- Representing “stuff”
 - Velocity, Force, Area, ...

About tensors

- Vector
- Unit-vector



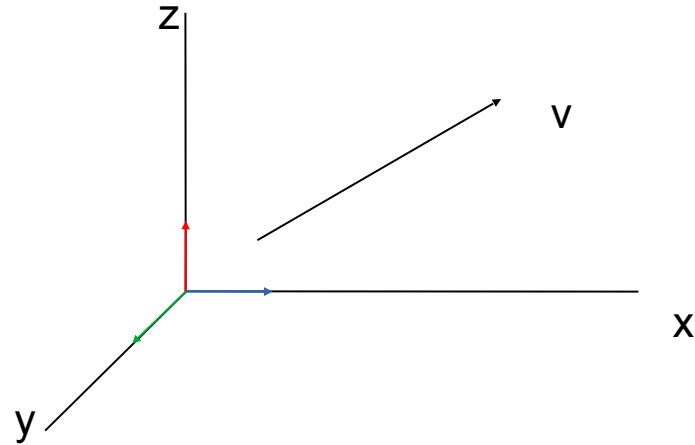
– i, j, k

– length = 1

– $v = a*i + b*j + c*k$

About tensors

- Vector
- Unit-vector
- Component



$$v = (a, b, c)$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$



This is a **rank-1 tensor**

tensors

- Generalization of vectors
- Rank is related to the number of "simultaneous" directions
- In a **N**-dimensional space:

0	scalar
1	vector
2	NxN matrix
≥ 3	tensor

tensors in pytorch

- `np.ndarray`
- on steroids
- GPUs love tensors
- can convert to/from numpy

about activation functions

- step
- sigmoid
- tanh
- ReLU
- → depends on the data!
- see notebook

about neural networks

- perceptron
- Feed-forward NN
- Multilayer perceptron
- CNN
- RNN
- ...

why layers?

- Layer == collection of neurons
- Each layer has its purpose
- Learning is done with the layers
- **ALL NEURONS IN ALL LAYERS WORK IN THE EXACT SAME WAY**
 - Calculate sum of weighted inputs + bias
 - Calculate the result of the activation function

how many neurons?

- Input layer
 - Number of features + 1 (for bias)
- Output layer
 - 1
 - 1
 - N
- Hidden layer
 - # samples / factor * (input + output)
 - Empirical
 - Factor in (1,10) to avoid overfitting

how many layers?

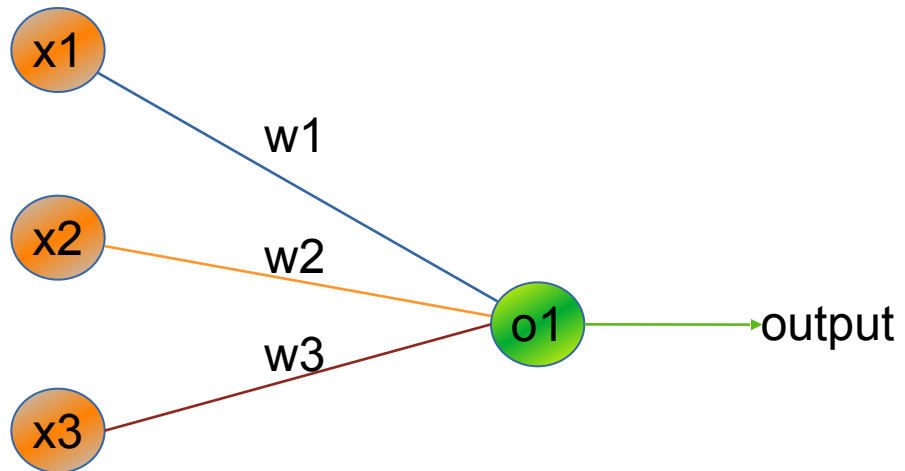
- Input layer
 - 1 (of course)
- Output layer
 - 1 (of course)
- Hidden layer
 - 1 (universal approximation theorem)
- Not so deep...
 - 0 layer -> linearly separable functions
 - 1 layer -> any continuous function
 - 2 layers -> arbitrary decision boundaries
 - >2 layers -> complex representations, automatic feature engineering

perceptron

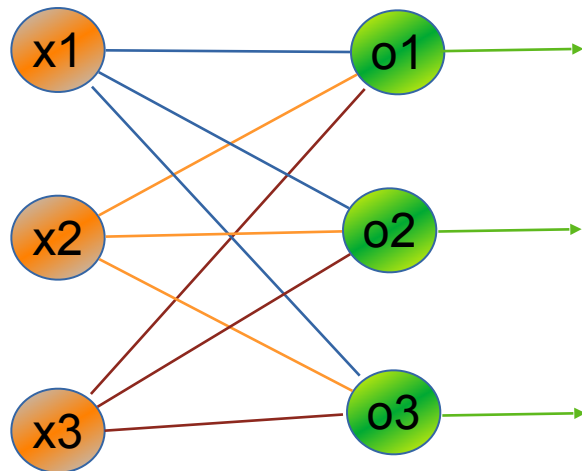
Perceptron

- . binary classifier

- + can implement SOME logic gates



Feed-Forward NN



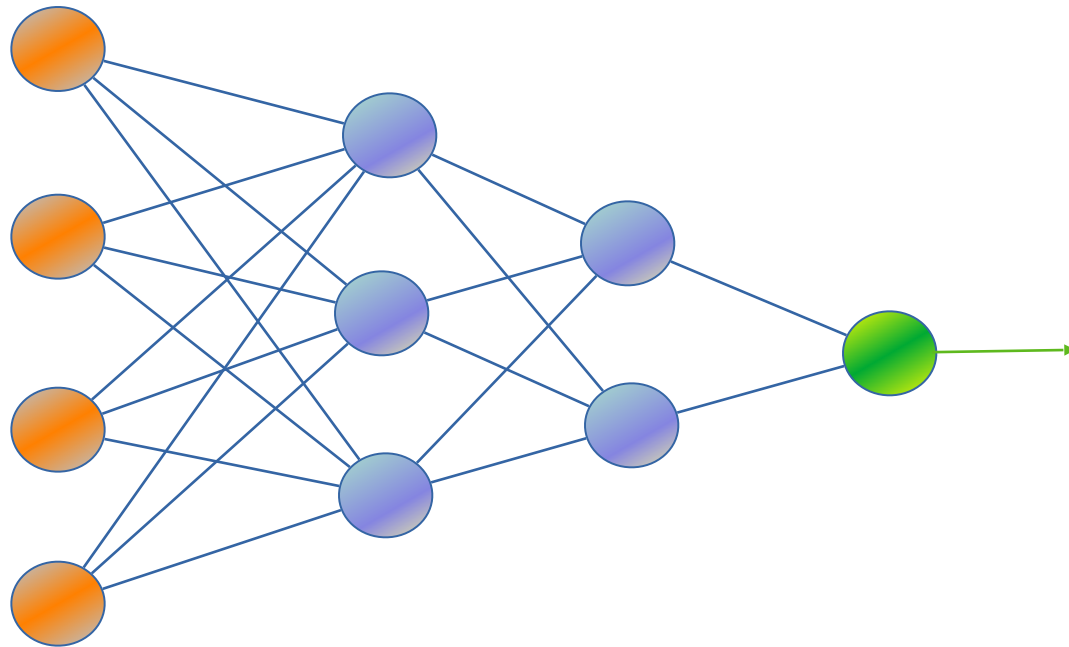
Basic Neural Network

- . classification
- . computer vision

- + w/wout hidden layers
- + no backpropagation
- + easy to design
- + highly responsive to noisy data
- + number of layers ~ complexity of func

- static weights
- no deep learning

Multi-layer perceptron

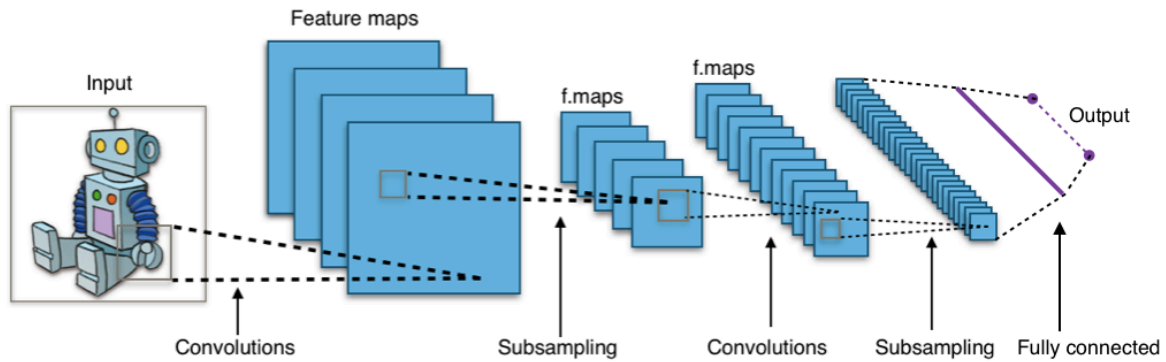


Standard Neural Network
· speech recognition
· complex classification

+ hidden layers
+ backpropagation
+ deep learning

- difficult to design
- dynamic weights

Convolutional NN



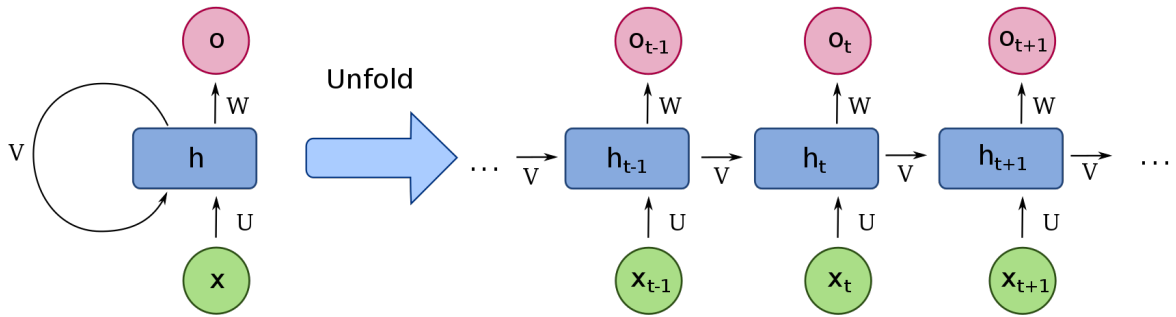
source: wikipedia

CNN

- . image classification

- + 3D arrangements of neurons
- + learn image by part
- + FF-only if more convolutional layers
- + FF + BP if outputs to fully connected
- + fewer parameters than fully connected

Recurrent NN



source: wikipedia

RNN

- . speech recognition
- . text to speech
- . sentiment analysis

+ save the output of a layer
+ model sequential data

- gradient vanishing

about frameworks

- pytorch
 - Low level API
 - Fine tuning
 - Focus: Broader machine learning
- Tensorflow
 - Low/High level API
 - Focus: Machine learning
- Keras
 - High level API
 - Works on Tensorflow, Theano, etc..
 - Focus: Deep Neural Networks

about frameworks

- `pip install torch torchvision`
- `pip install -upgrade tensorflow`
- `pip install keras`

- Use a virtual-environment if you do not want to mess up too badly. Your choice.
 - `Python3-venv`
 - `Conda`
 - ...

- Or google colab