Introduction to Neural Networks

Inspired from Raschka et al. [2022], Chollet [2021], Géron [2022]

Benoit Gaüzère

INSA Rouen Normandie - Laboratoire LITIS

November 20, 2023

Outline

Introduction to Representation Learning

Historical of Neural Networks

The Perceptron Model

The Multi Layer Perceptron

CNN, RNN etc

Limitations of others methods

Analysis of classic ML

1. Choose a dataset and a task
2. Compute features from the data
3. Learn the model on features
4. Predict

Problems

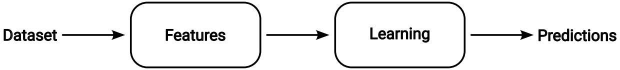
How can we be sure that our features are optimal ?

- They define the latent space
- Model ability to learn is limited by these representations

Neural Networks and Deep Learning

Motivation

- End to end learning
- Let's the model to find the optimal representation according to a task



Neural Networks and Deep Learning

Motivation

- End to end learning
- Let's the model to find the optimal representation according to a task

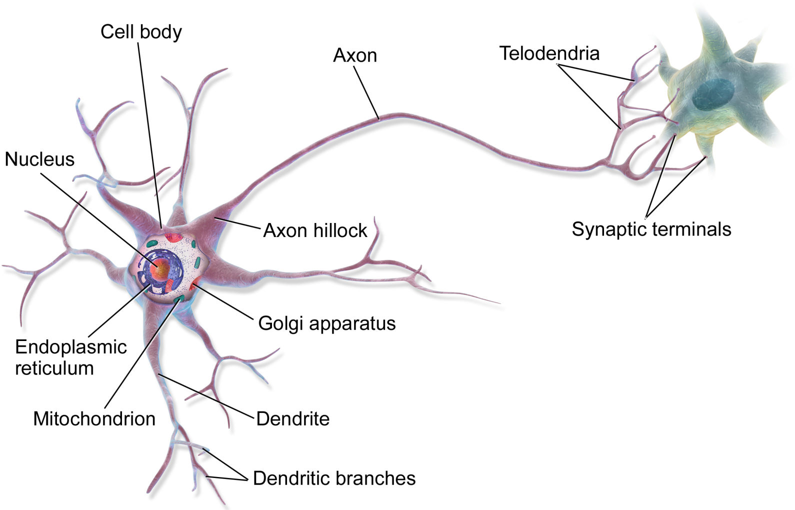
History of NN

How to simulate the human intelligence ?

The human brain

- Able to learn and adapt
- Simple neurons connected together
- Good connections are boosted

Brain Neuron



McCulloch Pitts (MCP) Neuron in 1943

BULLETIN OF
MATHEMATICAL BIOPHYSICS
VOLUME 5, 1943

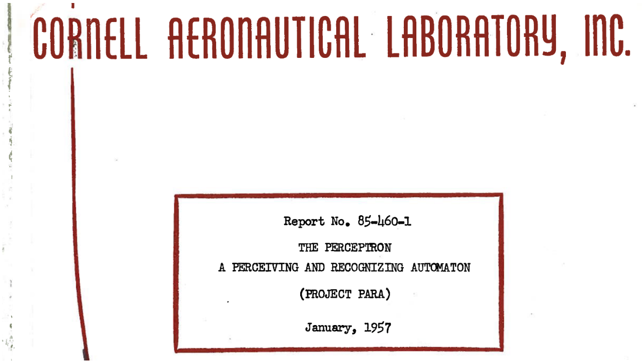
A LOGICAL CALCULUS OF THE
IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH AND WALTER PITTS

FROM THE UNIVERSITY OF ILLINOIS, COLLEGE OF MEDICINE,
DEPARTMENT OF PSYCHIATRY AT THE ILLINOIS NEUROPSYCHIATRIC INSTITUTE,
AND THE UNIVERSITY OF CHICAGO

- A first simple approach

The perceptron learning rule


CORNELL AERONAUTICAL LABORATORY, INC.

Report No. 85-460-1

THE PERCEPTRON

A PERCEIVING AND RECOGNIZING AUTOMATON

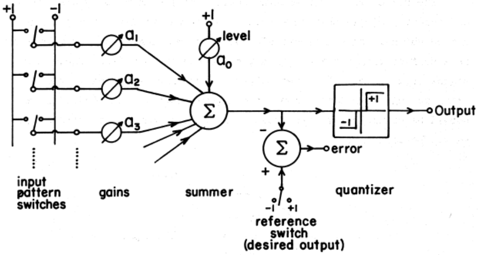
(PROJECT PARA)

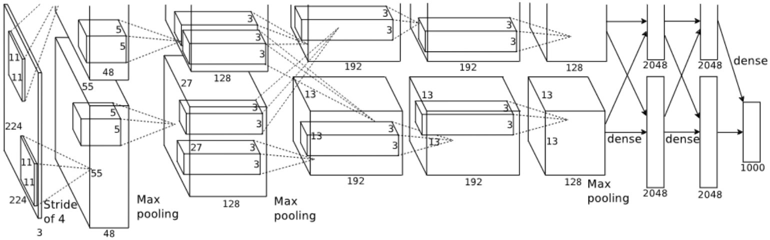
January, 1957

- The learning rule
- Basics of NN

And then

- Adaline
- Multilayer Perceptron
- LeNet
- AlexNet
- ...





The Perceptron I

Artifical neurons

- inputs x : vector components
- weights w: how inputs are used
- output $z = w^\top x + b$: net input
- Neuron fires if $z > 0$ i.e.

$$\sigma(z) = \begin{cases} 1 \text{ if } z > 0 \\ 0 \text{ otherwise} \end{cases}$$

Perceptron Learning Rule

The Perceptron II

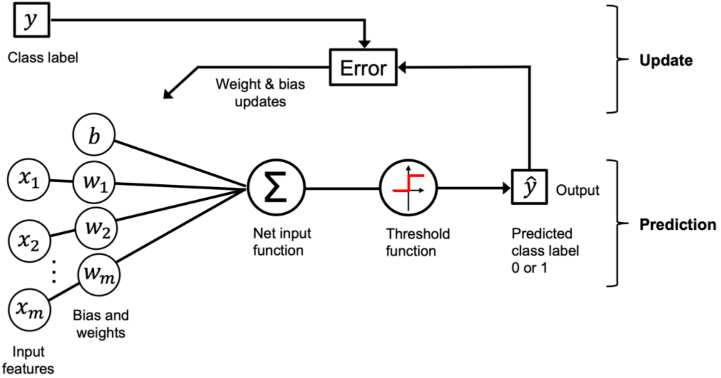
1. Initialize weights to small random values
2. For each training sample $\mathbf{x}_i$:
    2.1 Compute the output value $\hat{y}$
    2.2 Update the weights
    2.3 Repeat until convergence

How to update

- Update the weights according to the error
- $w_j = w_j + \Delta w_j$
- $\Delta w_j = \eta(y_i - \hat{y}_i)x_i(j)$
- η is the learning rate
- $b = b + \Delta b, \Delta b = \eta(y_i - \hat{y}_i)$

Let's try it !

The Perceptron III



Adding some complexity

Linearity

The model is linear by design

- Add some linearities !
- $z = g(w^\top x + b)$
- g is a differentiable function (ReLU, tanh, sigmoid, ...)

Layers

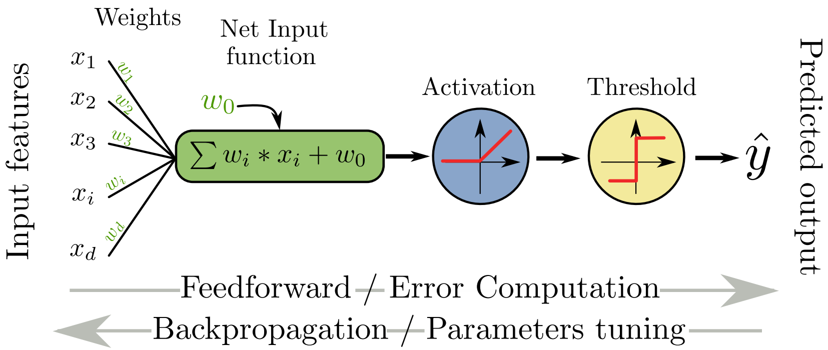
Add more layers to complexify the interactions between the components of x

- Lead to Multi Layer Perceptron
- And **Deep** Learning

Multi Layer Perceptron

Principle

Learn the best representation of data



- Weights $\boldsymbol{w}$ are optimized by gradient descent
- Sequence of layers

MLP Hyperparameters

Hidden layers

Define the architecture of your MLP

- Number of layers : a high number tends to deep networks
- Number of neurons per layer : a high number tends to wide networks

The model will be more complex if more neurons are used

Activation function

Determine how the non linearity is brought to the model

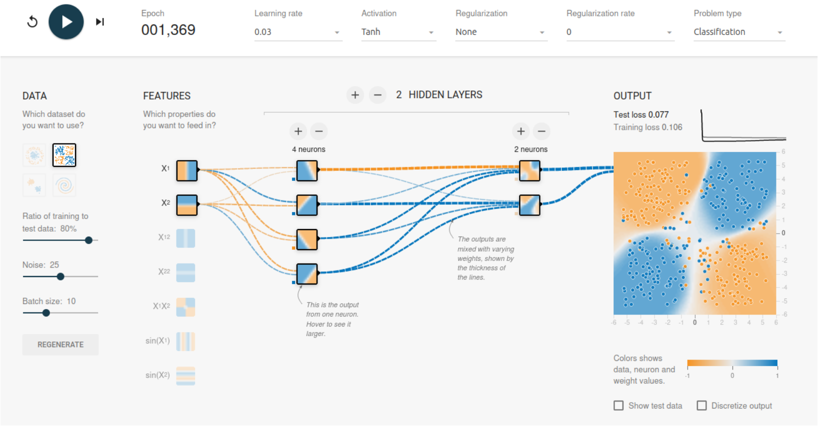
- identity : linear model
- tanh, relu, logistic : non linears. ReLU is a very popular choice

MLP : the code !

```
from sklearn.neural_network import MLPClassifier
activation = 'relu' # default
layers = [32,64,128,64,32] #5 layers avec différentes tailles
clf = MLPClassifier(hidden_layer_sizes=layers,max_iter=500)
clf.fit(X,y)
ypred = clf.predict(X)
```

- User guide for tips and help : link
- $\Rightarrow$ Notebook
- the documentation

Tensorflow playground



https://playground.tensorflow.org/

Exemple with MNIST

→ Notebook

```python

import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split

X,y = load_digits(return_X_y = True)
plt.imshow(X[124,:].reshape(8,8),cmap="gray")

mlp = MLPClassifier(hidden_layer_sizes=(64,32,16),
    activation='relu',verbose=True)
mlp.fit(X_train,y_train)
X_train, X_test, y_train, y_test = train_test_split(X,y)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,mlp.predict(X_test)))
0.97777777...
```

Extension of MLP

How to learn on non tabular data ?

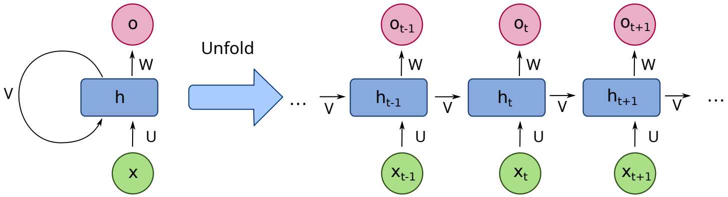
Constraints

- Data must be of fixed size dimensions
- Continuous
- All parts must be differentiable

Nature of the data

- How to take into account data topology ?
- Is MLP sufficient ?

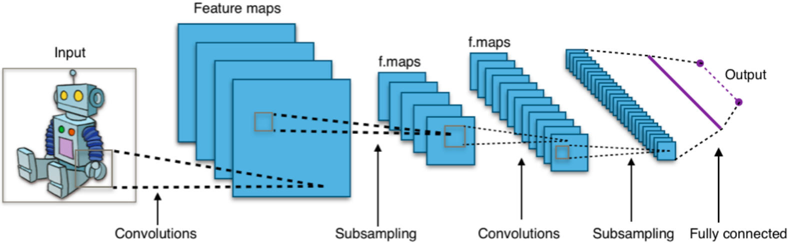
RNN : Adaptation to sequences



[wikipedia]

$\Rightarrow$ LSTM, GRU, ...

CNN : Adaptation to images



And other things . . .

Transformers

- SOTA for NLP and Images
- GPT is for Generative Pretrained Transformer
- Embed the context to derive a better decision

Generative models

- GAN
- Diffusion models
- dots

Conclusion

- Neural Networks is a powerful ML method
- Paradigm shift : representation is learnt
- Strong results since 10 years

What's next ?

How to adapt NN and CNN to molecules ?

References

References

Francois Chollet. Deep learning with Python. Simon and Schuster, 2021.

Aurélien Géron. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. " O'Reilly Media, Inc.", 2022.

Sebastian Raschka, Yuxi Hayden Liu, Vahid Mirjalili, and Dmytro Dzhulgakov. Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. Packt Publishing Ltd, 2022.