

Introduction to Deep Learning

General Information on the Course

Course Plan

Session 1 – **Generalities on Deep Learning**

Session 2 – **Neural Network Estimation**

Session 3 – **Hyperparameters Selection**

Session 4 – **Convolutional Networks**

Session 5 – **Generative Networks**

Session 6 – **Recurrent Networks**

Session 7 – **AI & Ethics**

Exam - 20/03

Content

- Theory and General Knowledge
- Applied Coding Questions

Duration

- 3 + 3 hours

Contact

yvenn.amara-ouali@universite-paris-saclay.fr

Reading List

- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. **Deep learning**. MIT press, 2016.
<https://www.deeplearningbook.org/>
- LeCun, Yann & Bengio, Y. & Hinton, Geoffrey. (2015). **Deep Learning**. Nature. 521. 436-44.
10.1038/nature14539.

Getting to know each other!

1. Your name
2. Your curriculum
3. What you'd like to get from the course

What is Deep Learning?

Common Definition

Deep Learning is a subset of **machine learning**, which is essentially a **neural network** with three or more **layers**. These neural networks attempt to **simulate the behavior of the human brain** — albeit far from matching its ability — allowing it to “**learn**” from large amounts of **data**. While a neural network with a single **layer** can still make approximate predictions, additional hidden **layers** can help to optimize and refine for accuracy.

IBM

General Definition

Deep learning allows computational **models** that are composed of multiple processing **layers** to **learn** representations of **data** with multiple levels of **abstraction**.

LeCun, Bengio and Hinton – Deep Learning, Nature (2015)

Different types of “Learning”

Supervised

The model is trained on labeled data in order to make predictions on new, unseen examples



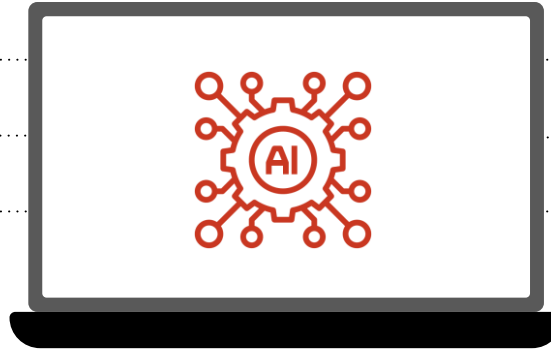
Unsupervised

The model is not given any labels and must discover the data structure



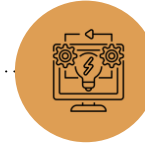
Self-supervised

The model uses the input data to generate its own supervision signals



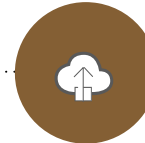
Generative

The model learns the data's underlying distribution to generate new samples



Reinforcement

The model learns a policy by maximising a reward signal in order to interact with an environment



Transfer

A model trained on one task is re-purposed on a second related task

A few Deep Learning applications

Image Recognition

Autonomous Vehicles, Healthcare (e.g., cancer detection), Object recognition, Facial Recognition, Augmented Reality, Robotics

Natural Language Processing

Text Classification, Language Modeling, Speech Recognition, Caption Generation, Machine Translation, Document Summarization

Times Series Modelling

Financial Markets, Predictive Maintenance, Energy Consumption, Traffic forecasting

Game Automation

AlphaGo, Leela, AlphaZero, AlphaStar

Generative Models

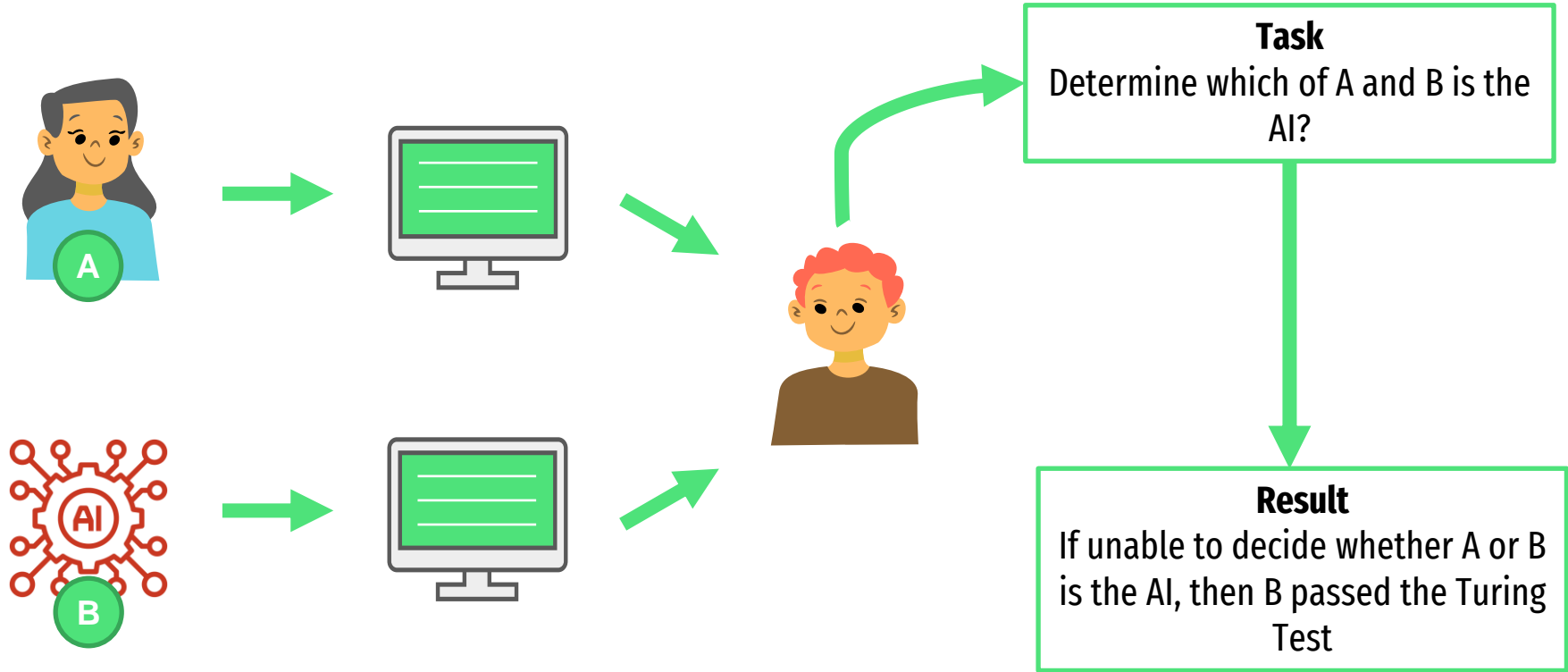
Dall-E, GPT-3

Deep Learning

A brief history



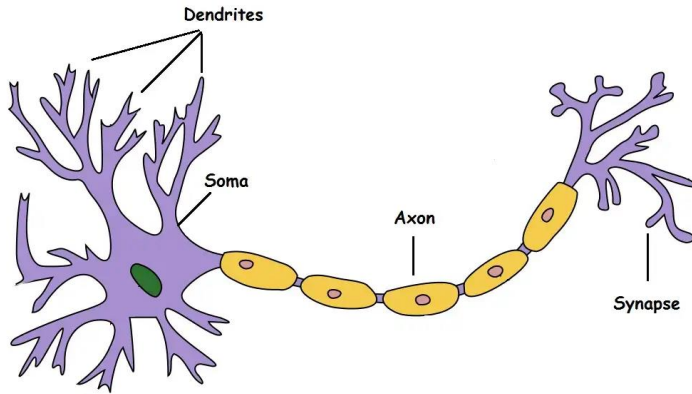
Turing Test - 1950



McCulloch and Pitts Neuron - 1943

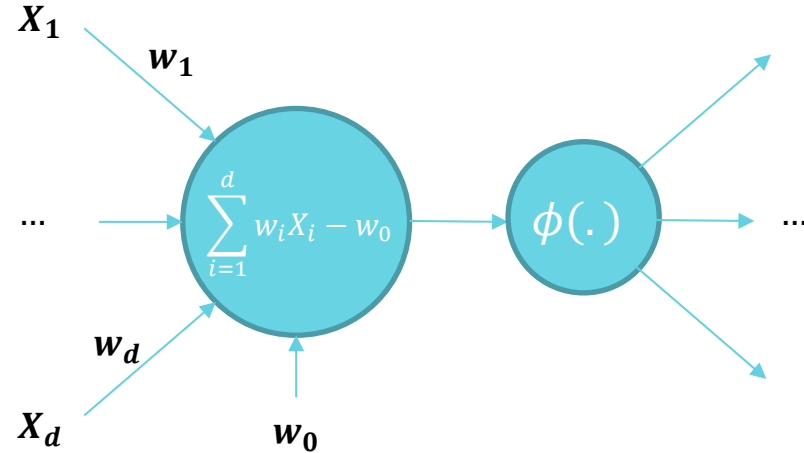
A visual representation

Biological Neuron



- Dendrite:** Receives signals from other neurons
- Soma:** Processes the information
- Axon:** Transmits the output of this neuron
- Synapse:** Point of connection to other neurons

Formal Neuron

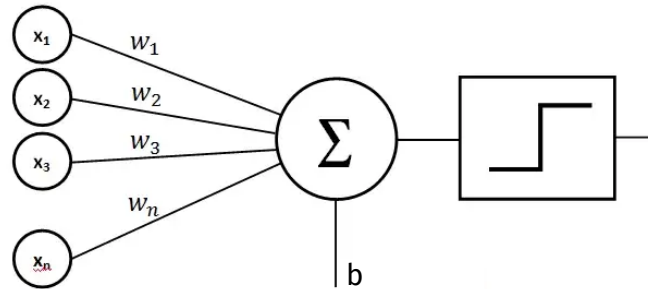
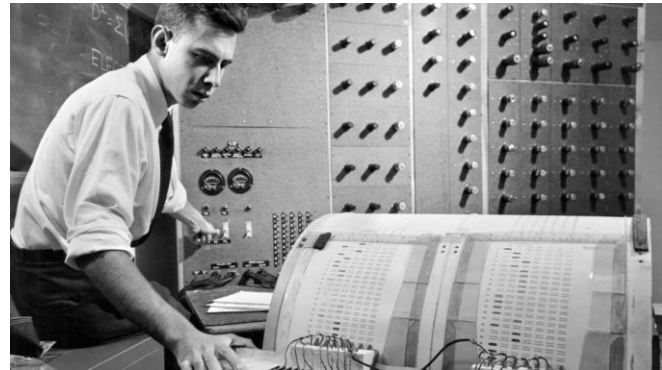


- X_i neuron input (e.g., covariate)
- w_i weight associated to an input (w_0 is a threshold)
- ϕ activation function

Rosenblatt's Perceptron - 1958

A practical implementation

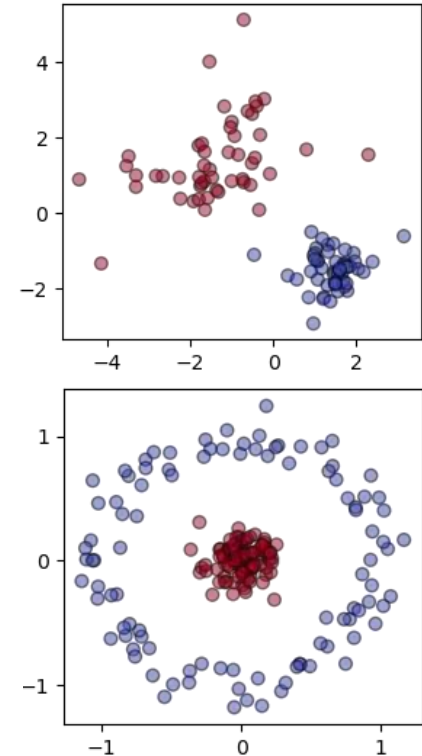
- Frank Rosenblatt was an American psychologist who developed the perceptron inspired by the works of McCulloch and Pitts and the Hebbian theory of synaptic plasticity
- The perceptron allowed artificial neurons to learn from data and figure out the correct weights directly from training data through a supervised learning algorithm
- He proposed a practical implementation of a neuron for image recognition



Minsky and Papert - 1969

- The perceptron algorithm became popular due to its simplicity and efficiency for solving linearly separable problems.
- However, there were unrealistic expectations about the capabilities of the perceptron, which were fueled by media reports and overhyped claims.
- In 1969, Minsky and Papert published a book showing the significant limitations of the perceptron, including its inability to learn the XOR function.
- This book is believed to have contributed to the "AI winter" of the 1970s, during which funding and interest in artificial intelligence research declined.

The first AI Winter



Backpropagation – 1960s to 1980s

Estimating weights

- Backpropagation, or the method of efficiently training neural networks by iteratively adjusting the weights in the network through **gradient descent**, has been discussed since the 1960s but was not applied in practice to neural networks until 1985
- The first explicit, efficient implementation of backpropagation was described in a 1970 master's thesis by Linnainmaa accompanied with a FORTRAN code, though it was not explicitly linked to neural networks at the time.
- Backpropagation was later used to minimize cost functions by adjusting control parameters in 1973 and was discussed in relation to neural networks for the first time in a 1974 dissertation by Werbos, with a computer program for implementing it in differentiable systems developed in 1980.
- In 1985, Rumelhart et al. conducted an experimental analysis of backpropagation and demonstrated its effectiveness in producing useful internal representations in hidden layers of neural networks.

The first CNNs – 1979 to 1989

- Kunihiro Fukushima developed the first "convolutional neural networks" in 1979 with the creation of Neocognitron a hierarchical, multilayered design to recognize visual patterns. It was trained with a reinforcement strategy of recurring activation in multiple layers.
- A decade later, in 1989, after Rumelhart et al. had shown that backpropagation could be applied to neural networks successfully, Yann LeCun demonstrated the practical use of backpropagation for reading handwritten digits using convolutional neural networks at Bell Labs.
- Interestingly, this last discovery happened during the second AI winter, that occurred between the late 1980s and early 1990s which again impacted research on neural networks and deep learning. The origins of this second AI winter were again the failure of AI systems to live up to high expectations.
- Other contributing factors included the high cost of developing AI systems and limited computing power at the time.

Universal Approximation Theorem -1990s

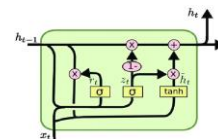
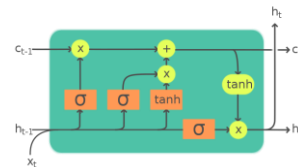
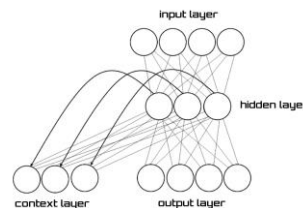
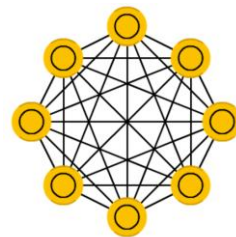
UAP

- Universal approximation refers to the ability of a class of functions to approximate with an arbitrary precision, any continuous function (or Lebesgue measurable).
- In 1989, Cybenko proved that arbitrary width neural networks with sigmoid activation functions can achieve universal approximation.
- In the same year, Hornik, Stinchcombe, and White demonstrated that multi-layer feedforward networks with at least one hidden layer are universal approximators.
- Hornik also showed in 1991 that the architecture of the network, rather than the specific choice of activation function, enables neural networks to perform universal approximation.
- Leshno et al. in 1993 and Pinkus in 1999 showed that universal approximation is equivalent to having a non-polynomial activation function.

The birth of RNNs – 1982 to 1997

- The first RNN is credited to the Hopfield Network introduced in 1982 by John Hopfield, a physicist at Caltech. They had a significant impact helping renew interest in neural networks in the early 1980s. Hopfield aimed to explore the question of emergence in cognitive systems, specifically asking if stable cognitive phenomena like memories could arise from large numbers of simple neurons
- The Elman Network, introduced by Jeffrey Elman in 1990, was the first successful example of a recurrent network trained with backpropagation. However, when attempting to train deep networks like RNNs, the gradients were vanishing or exploding, making the task challenging
- The Long Short-Term Memory (LSTM) Network, introduced in 1997 by Sepp Hochreiter and Jurgen Schmidhuber, was a significant advancement in the field. LSTMs are characterized by the inclusion of units with both short-term and long-term memory capabilities to counter vanishing gradient issues
- Another similar structure, although more condensed, was proposed in 2014 by Cho et al. with the Gated Recurrent Units (GRUs)

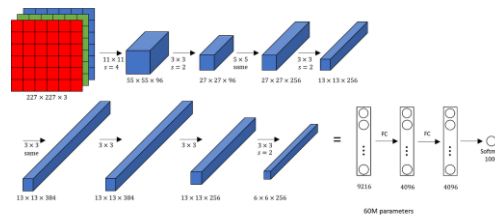
Recurrent Neural Networks



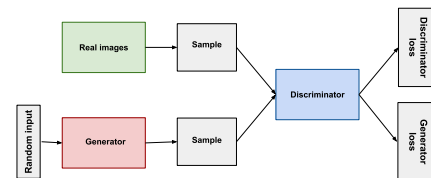
Major Breakthroughs – 2010s

The AI Spring

2012 – AlexNet, a deep convolutional neural network (CNN), wins the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a large margin, demonstrating the effectiveness of deep learning for image classification.



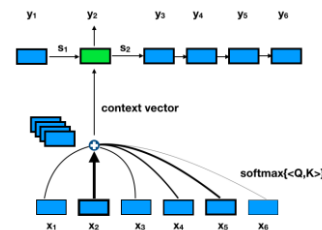
2014 – Generative Adversarial Networks (GANs) are introduced by Ian Goodfellow et al. GANs are able to generate new, previously unseen data samples that are indistinguishable from real data.



2015 – Google DeepMind's AlphaGo defeats the world champion at the board game Go, marking a major milestone in the capabilities of artificial intelligence.



2016 – Attention mechanisms are introduced, allowing neural networks to selectively focus on certain parts of their input when processing it. The following year, deep learning approaches achieve state-of-the-art results on a wide range of natural language processing tasks, including language translation, summarization, and question answering.



Some of the best recent models

Today

Computer Vision state-of-the-art models

- Residual Networks (ResNets)
- Densely Connected Convolutional Networks (DenseNets)
- Inception Networks (InceptionNets)
- Transformative Feedback Networks (TFNets)

Time series modelling

- Long Short-Term Memory (LSTM) networks
- Gated Recurrent Units (GRUs)
- Temporal Convolutional Networks (TCNs)

Generative models

- Generative Adversarial Networks (GANs)
- Variational Autoencoders (VAEs)
- Diffusion Models (DMs)
- Transformative Feedback Networks (TFNets)

Have fun with deep neural networks

Today

ChatGPT

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

GPT-5

lichess.org JOUER PROBLEMES APPRENDRE REGARDER COMMUNAUTE OUTILS

BULLET 2601? 57 838 parties

BLITZ 2658? 7 12 447 parties

RAPIDE 2493? 22 4074 parties

CLASSIQUE 2240? 49 28 parties

CORRESPONDANCE 70 parties

BOT LeelaChess

4 Etudes Message Post

Leela Chess
Leela Chess Zero playing full strength. I play UltraBullet, Bullet, Blitz and Rapid. Send me a challenge and wait for your turn! If you are titled and want a longer matchup, please DM me. Donations: paypal.me/leelacss2

Canada
Membre depuis 14 avr. 2018
Actif il y a 1 an
Temps total à jouer : 185 jours, 3 heures et 14 minutes

Chess Insights
Analytics from LeelaChess's games

Leela Zero

DALL-E History Collections

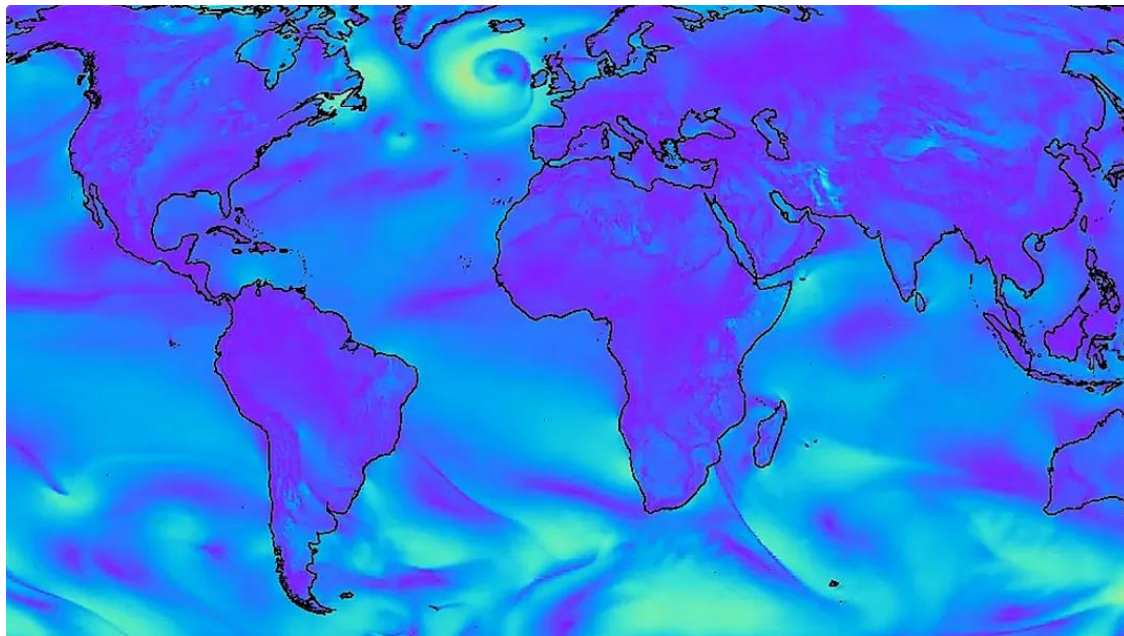
Impressionist oil painting of sunflowers in a purple vase. Generate

Grid of generated images: a colorful fish in a glass bowl, an avocado, a person in a dynamic pose, a purple furry creature, a robot playing chess, a person in a dark landscape, and an orange slice.

DALL-E 3

Challenging our world

Today



GraphCast

The Formal Neuron

At the beginning



McCulloch and Pitts Formal Neuron

A mathematical formulation

For a set of inputs $(X_i)_{i \in \{1, \dots, d\}}$, weights $(w_i)_{i \in \{1, \dots, d\}}$, an activation function ϕ , and a threshold w_0 (which we later call bias) we can define the output of one neuron, written y , as follows:

Neuron output

$$y = \phi(z), \quad z = \sum_{i=1}^d w_i X_i - w_0$$

Historically, the activation function chosen by McCulloch and Pitts was the Heaviside function which can be defined as follows:

Heaviside function

$$\forall x \in \mathbb{R}, \phi(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \quad \Rightarrow \quad \phi(z) = \begin{cases} 1, & \text{if } \sum_{i=1}^d w_i X_i > w_0 \\ 0, & \text{if } \sum_{i=1}^d w_i X_i \leq w_0 \end{cases}$$

Perceptron weights estimation

Algorithm

Initialisation

1. Input $\mathbf{X} = (X_i)_{i \in \{1, \dots, d\}} \in \mathbb{R}^d$
2. Weights $(w_i)_{i \in \{1, \dots, d\}} = (0, \dots, 0)$
3. Biases $(b_i)_{i \in \{1, \dots, d\}} = (0, \dots, 0)$
4. Target $Y \in \{0, 1\}$

Perceptron activation

$$f(\mathbf{X}) = \phi \left(\sum_{i=1}^d w_i X_i + b_i \right)$$

Estimation

While convergence criteria is not met:

Update step k: For $i \in \{1, \dots, d\}$ $w_i^{(k+1)} = w_i^{(k)} + r(Y - f(\mathbf{X}))X_i = \begin{cases} w_i^{(k)}, & \text{if } Y = f(\mathbf{X}) \\ w_i^{(k)} + rYX_i, & \text{if } Y \neq f(\mathbf{X}) \end{cases}$

Convergence criteria

Implementation

Can the perceptron algorithm converge regardless of the input and target data ?

Convergence criteria

Implementation

NO ! The algorithm presented does not converge regardless of the data.

The solution space is the set of hyperplanes of \mathbb{R}^p so it is not possible to separate non-linear data

Some solutions to this caveat involve setting a limited amount of iterations (or epochs) to approximate a plausible margin and slightly modify the assumptions to allow for misclassification of a small set of targets

Convergence criteria

Theorem

Convergence of the Perceptron Learning Algorithm

For any finite set of linearly separable labeled examples, the Perceptron Learning Algorithm will stop after a finite number of iterations. In other words, after a finite number of iterations, the algorithm yields a vector w that classifies perfectly all the examples.

Remark

Although the number of iterations is finite, it is usually larger than the size of the training set, because each example needs to be processed more than once.

Convergence criteria

Theorem

Proof

Neural Networks

Universality
Approximation Theorem



Activation functions

Examples

- Heaviside
- Linear
- Sigmoid
- Tanh
- ReLU
- eLU
- Leaky ReLU
- Parameterised ReLU
- Swish
- Softmax

Universal Approximation

Theorem

G. Cybenko (1989)

Let σ be a sigmoidal function: $\sigma(x) = \begin{cases} 1 & \text{for } x \rightarrow +\infty \\ 0 & \text{for } x \rightarrow -\infty \end{cases}$

Then finite sums of the form $g(x) = \sum_{j=1}^N w_j^2 \sigma((w_j^1)^T x + b_j^1)$ are dense in $C(I_n)$ with respect to the supremum norm. Where $x \in \mathbb{R}^n$, $w_j^2 \in \mathbb{R}$, $w_j^1 \in \mathbb{R}^n$, $b_j^1 \in \mathbb{R}$. I_n is the n-dimensional unit cube $[0, 1]^n$ and $C(I_n)$ is noted for the space of continuous functions on I_n . In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum, $g(x)$, of the above form, for which $|f(x) - g(x)| < \epsilon, \forall x \in I_n$

Interpretation

Any continuous function on a compact subset of $[0, 1]^n$ can be approximated by a feed forward neural network with only one single hidden layer and a finite number of neurons.