

Présentation

TRAN-THUONG Tien-Thinh

2021-2022

Problématique

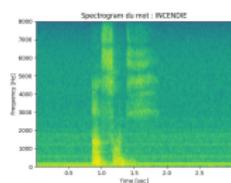
D'après le ministère de la Santé : Il y a eu plus de **31 millions** d'appels d'urgence en 2018. Seuls **69%** des appels étaient décrochés dans la minute.

Objectif

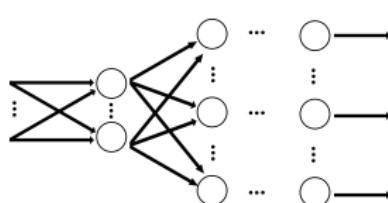
Utiliser la reconnaissance vocale par réseau de neurones pour aider à classifier rapidement l'objet d'un appel.

La reconnaissance automatique de la parole

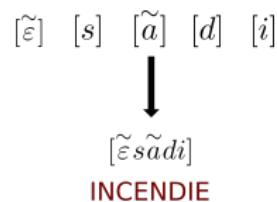
- 1 Le traitement acoustique
- 2 L'apprentissage automatique
- 3 Le décodage



(a) Spectrogramme



(b) Réseau de neurones



(c) Correspondance phonétique

I - Introduction

Présentation du modèle du Perceptron

McCulloh et Pitts introduisent le modèle du Perceptron en 1943, basé sur le fonctionnement du neurone humain.

I - Introduction

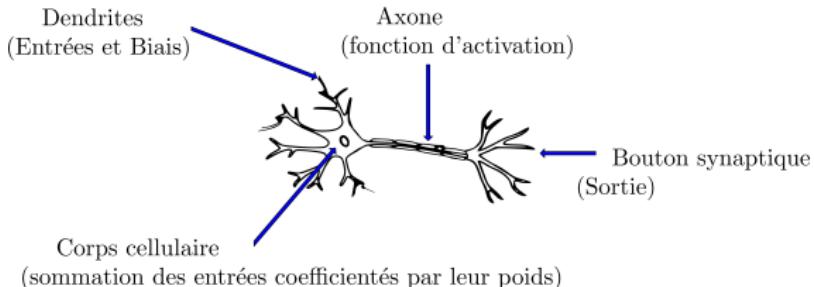


Figure – Schéma d'un neurone

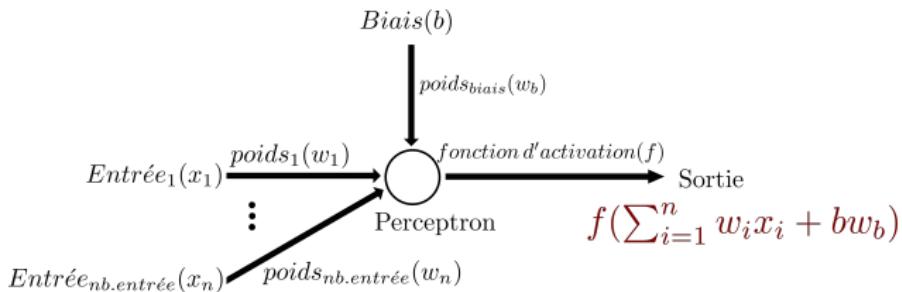


Figure – Schéma d'un perceptron

II - Fonction d'activation

Fonction d'activation

Sans l'utilisation de la fonction d'activation, le neurone est multilinéaire par rapport à ses entrées, il n'est donc capable que de faire des régressions linéaires sur les données d'entrées.

Les fonctions d'activation permettent donc une classification non linéaire.

II - Représentation informatique

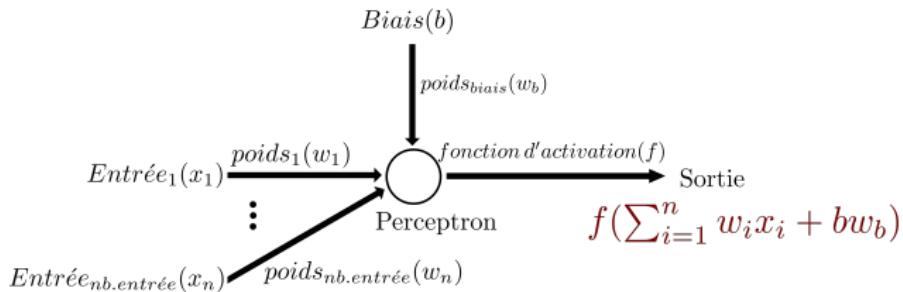


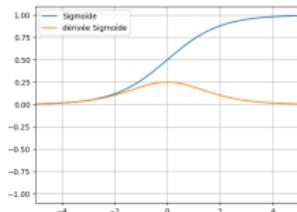
Figure – Schéma d'un perceptron

$$f \left(\begin{pmatrix} x_1 & \dots & x_n & b \end{pmatrix} \times \begin{pmatrix} w_1 \\ \vdots \\ w_n \\ w_b \end{pmatrix} \right)$$

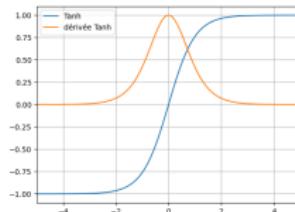
La complexité est en $O(n)$

```
1 import numpy as np
2
3 def calcul(activation, X, W):
4     # Ajout du biais
5     X = np.concatenate((X, np.
6         ones((len(X), 1))), axis=1)
7     # Calcul de la sortie
8     z = activation(np.dot(X, W))
9
10    return z
```

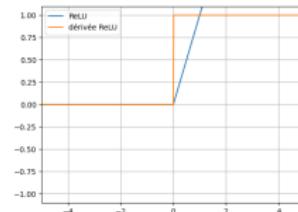
II - Fonction d'activation



(a) Sigmoïde



(b) Tanh



(c) ReLU

Fonction	Formule	Dérivée
Sigmoïde (a)	$\frac{1}{1 + e^{-x}}$	$f(x) \times (1 - f(x))$
Tangente Hyperbolique (Tanh) (b)	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f(x)^2$
Unité Linéaire Rectifiée (ReLU) (c)	$\max(0, x)$	$\begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{sinon} \end{cases}$

II - Fonction d'activation, les spécificités

Fonction	Avantage	Inconvénient
Sigmoïde	A valeur dans $]0, 1[$ ce qui facilite les classifications binaires	Dérivée petite vers $\pm\infty$, il y a peu d'apprentissage pour ces valeurs
Tanh	Utilisé dans les couches cachées car fonction impaire	Même problème que la Sigmoïde
ReLU	Plus simple à calculer, prend en compte le gradient pour toute valeur positive	Dérivée nulle en x négatif ce qui peut rendre des neurones inutiles

Descente de gradient

Descente de gradient

La Descente de Gradient est un algorithme d'optimisation qui permet de trouver un minimum local d'une fonction en convergeant progressivement.

Dans l'apprentissage des réseaux de neurones, la descente de gradient est utilisée pour trouver le minimum d'une fonction coût, évaluant l'erreur entre la valeur de sortie du réseau et celle attendu.

En effet, trouver des paramètres (poids, architecture du réseau, fonction d'activation) permettant d'avoir une erreur nulle revient à résoudre le problème qu'évalue cette fonction coût par rapport aux entrées données.

III - Descente de gradient

Algorithme du gradient

Soit $n \in \mathbb{N}, \varepsilon > 0$. On munit \mathbb{R}^n de son produit scalaire canonique.

Soit f une fonction différentiable de $\mathbb{R}^n \rightarrow \mathbb{R}$.

Soit x_0 une valeur initiale aléatoire, t le taux d'apprentissage.

Supposons x_0, \dots, x_k construits.

- Si $\|\nabla f(x_k)\| \leq \varepsilon$, on s'arrête.
- Sinon on pose $x_{k+1} = x_k - t \nabla f(x_k)$

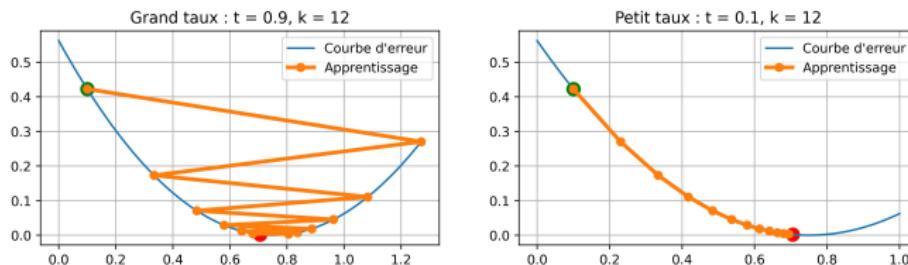


Figure – Descente de Gradient pour $f(x) = (x - 0.75)^2$; $x_0 = 0.1$ et $\varepsilon = 0.1$

III - Importance du choix du taux d'apprentissage

Pour la suite on continuera avec la fonction $f(x) = (x - 0.75)^2$ et $x_0 = 0.1$.
On montre qu'en choisissant un taux d'apprentissage trop petit ou trop grand, il est possible que la descente de gradient diverge, ou ne converge pas assez vite.

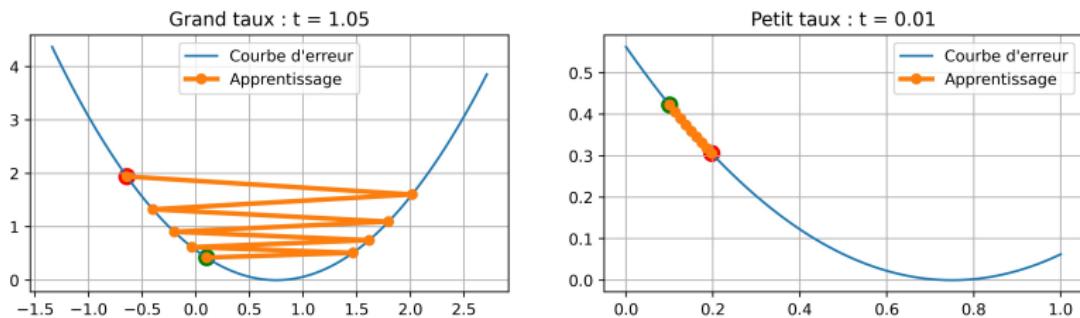


Figure – Descente de Gradient on force l'arrêt à $k = 8$

III - Utilisation du Moment

Descente de gradient avec moment

x_0 aléatoire et le moment $\omega_0 = 0$. Supposons x_0, \dots, x_k et $\omega_0, \dots, \omega_k$ construits.

- On pose $\omega_{k+1} = \gamma\omega_k + t\nabla f(x_k)$
- On pose $x_{k+1} = x_k - \omega_{k+1}$

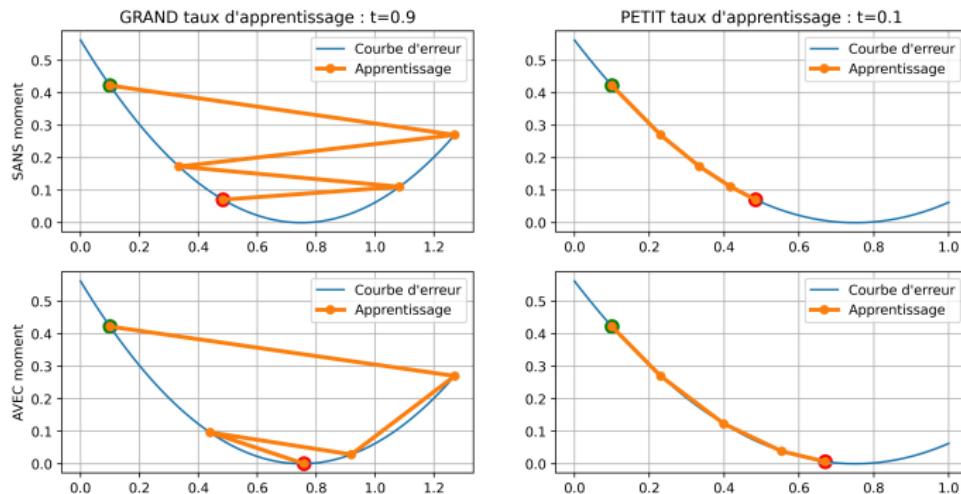


Figure – Comparaison sans puis avec dépendance au moment avec $\gamma = 0.5$, arrêt à $k = 4$

III - Utilisation du Moment

Le moment permet également de s'échapper de certains minimum locaux.

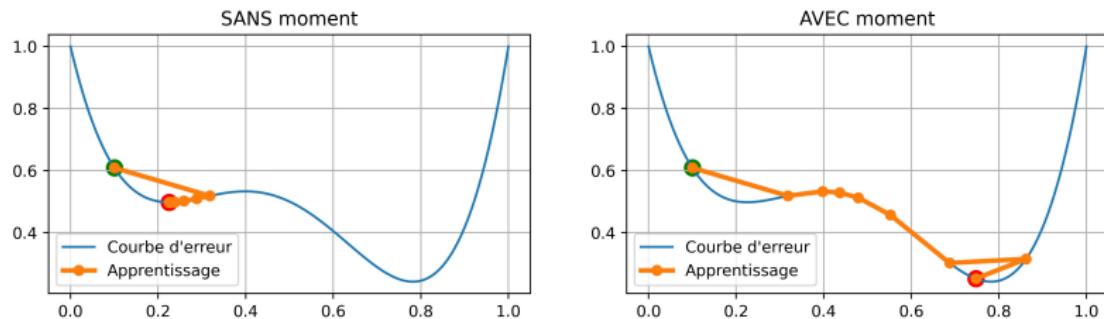


Figure – Comparaison sans puis avec dépendance au moment avec $\gamma = 0.5$, arrêt à $k = 8$

III - Apprentissage stochastique ou par paquet (Batch)

Il faut prendre en compte le fait que les D données d'apprentissage ne sont pas toujours juste, elles peuvent contenir des erreurs.

$$\left\langle f \begin{pmatrix} x_1^1 & \dots & x_n^1 & b \\ \vdots & \ddots & \vdots & \vdots \\ x_1^D & \dots & x_n^D & b \end{pmatrix} \times \begin{pmatrix} w_1 \\ \vdots \\ w_n \\ w_b \end{pmatrix} \right\rangle$$

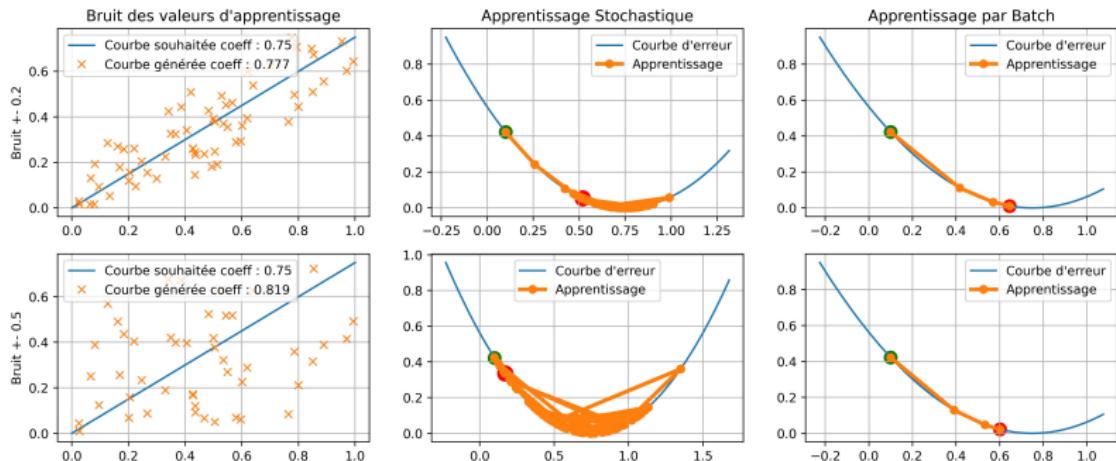


Figure – Comparaison apprentissage stochastique et par paquet

IV - Problème de reproduction de l'opérateur XOR

Problème non linéairement séparables

Un perceptron ou une couche de perceptron est incapable de reproduire des opérateurs non linéairement séparables.

Il faut alors mettre des couches de perceptrons en série, des couches cachées, pour reproduire ces opérateurs.

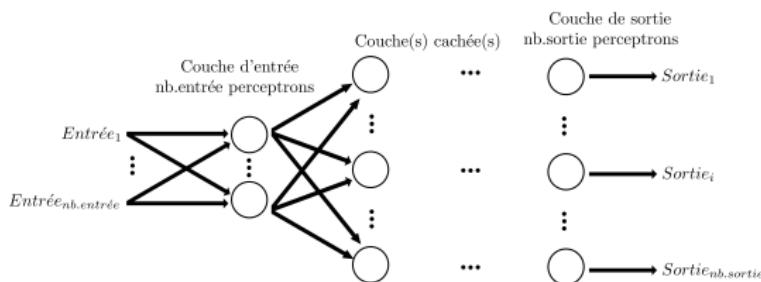


Figure – Schéma d'un réseau de neurones

IV - Problème de reproduction de l'opérateur XOR

Le XOR nécessite un réseau

Le XOR, ou exclusif, est un opérateur non linéairement séparable.

On peut par exemple démontrer que l'ajout d'une couche cachée de 2 perceptrons suffit à reproduire l'opérateur XOR.