

# Présentation

TRAN-THUONG Tien-Thinh

2021-2022

# Problématique

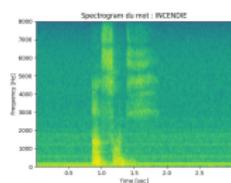
D'après le ministère de la Santé : Il y a eu plus de **31 millions** d'appels d'urgence en 2018. Seuls **69%** des appels étaient décrochés dans la minute.

## Objectif

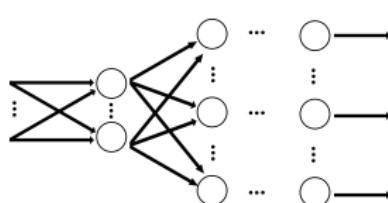
Utiliser la reconnaissance vocale par réseau de neurones pour aider à classifier rapidement l'objet d'un appel.

# La reconnaissance automatique de la parole

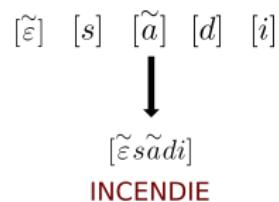
- 1 Le traitement acoustique
- 2 L'apprentissage automatique
- 3 Le décodage



(a) Spectrogramme



(b) Réseau de neurones



(c) Correspondance phonétique

## Présentation du modèle du Perceptron

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

# I - Introduction

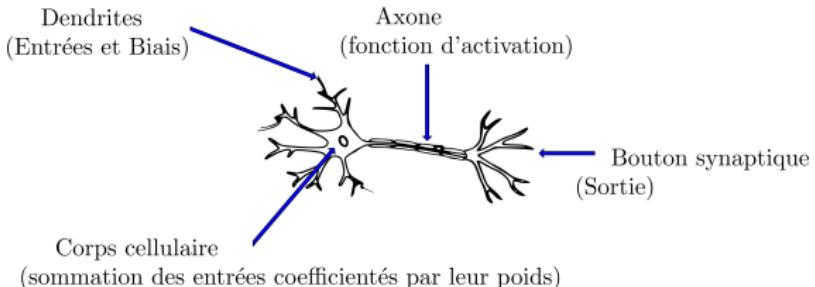


Figure – Schéma d'un neurone

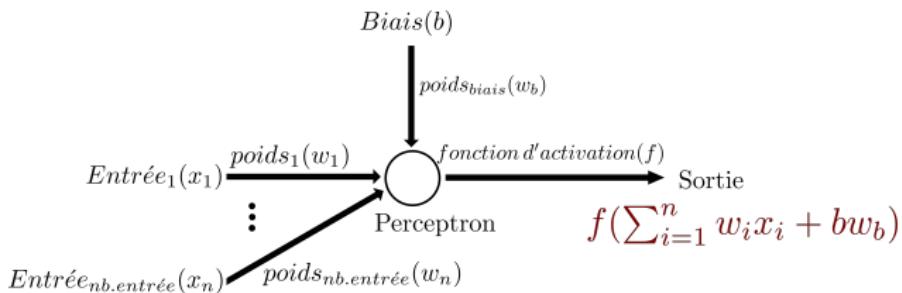


Figure – Schéma d'un perceptron

## 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

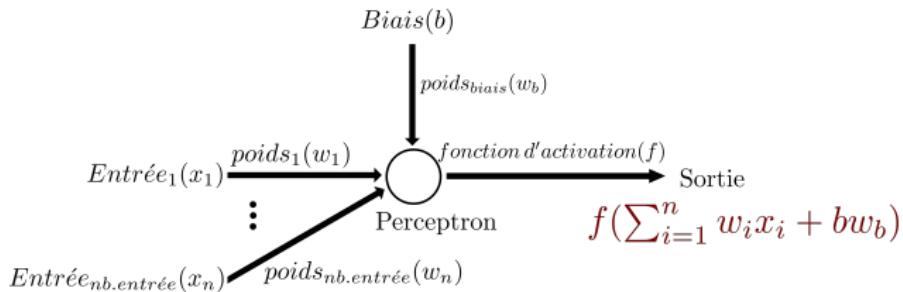


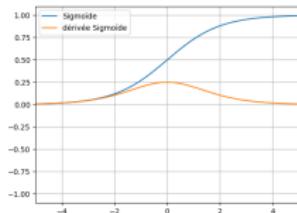
Figure – Schéma d'un perceptron

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

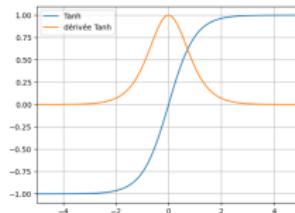
La complexité est en  $O(n)$

```
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
```

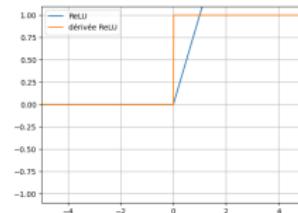
## II - Fonction d'activation



(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  $x_{k+1} = x_k - t \nabla f(x_k)$

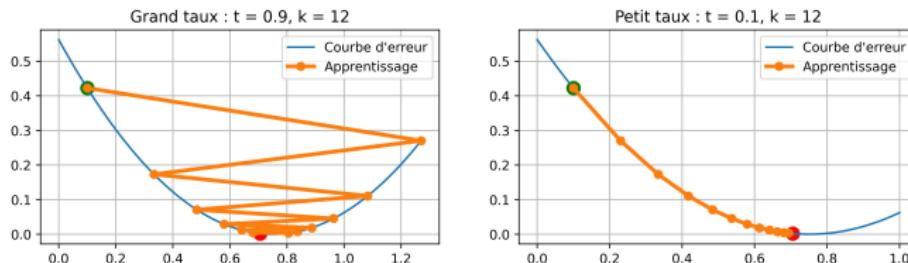


Figure – Descente de Gradient pour  $f(x) = (x - 0.75)^2$ ;  $x_0 = 0.1$  et  $\varepsilon = 0.1$

### 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.

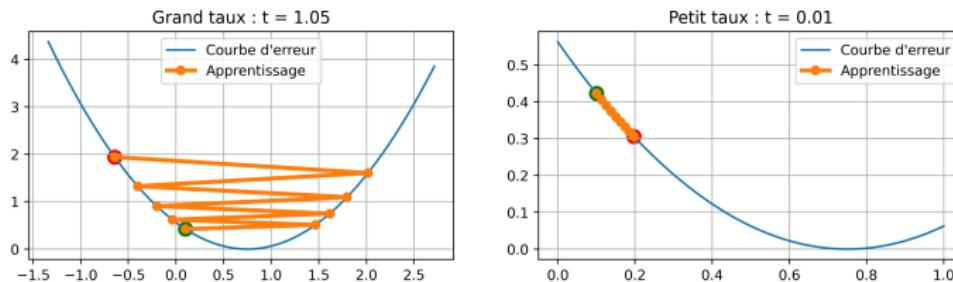


Figure – Descente de Gradient on force l'arrêt à  $k = 8$

### 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  $x_{k+1} = x_k - \omega_{k+1}$

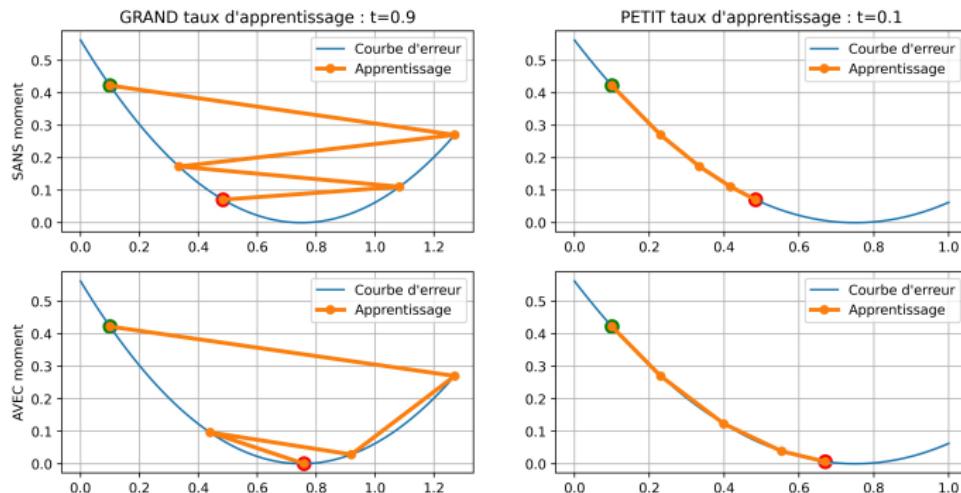


Figure – Comparaison sans puis avec dépendance au moment avec  $\gamma = 0.5$ , arrêt à  $k = 4$

### III - Utilisation du Moment

Voici, une simulation pour des taux d'apprentissage "trop grand" ou "trop petit".

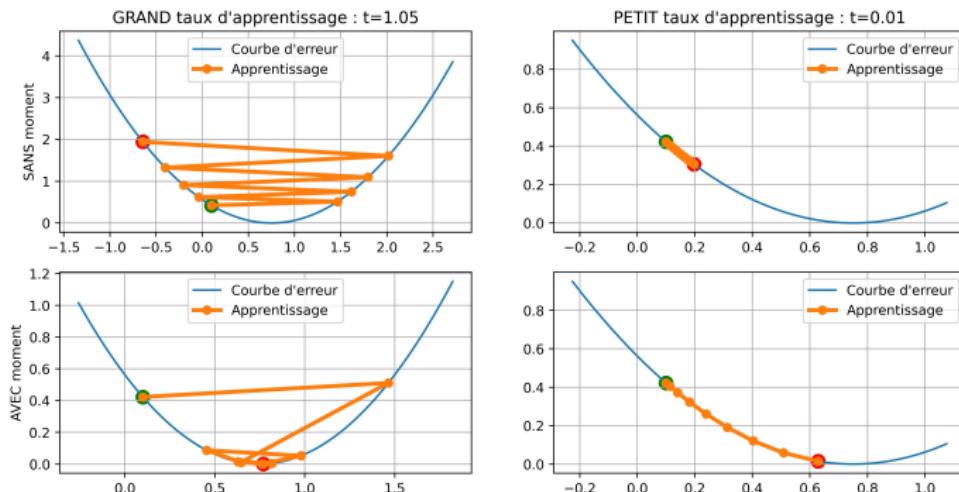


Figure – Comparaison sans puis avec dépendance au moment avec  $\gamma = 0.5$ , arrêt à  $k = 8$

### III - Utilisation du Moment

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

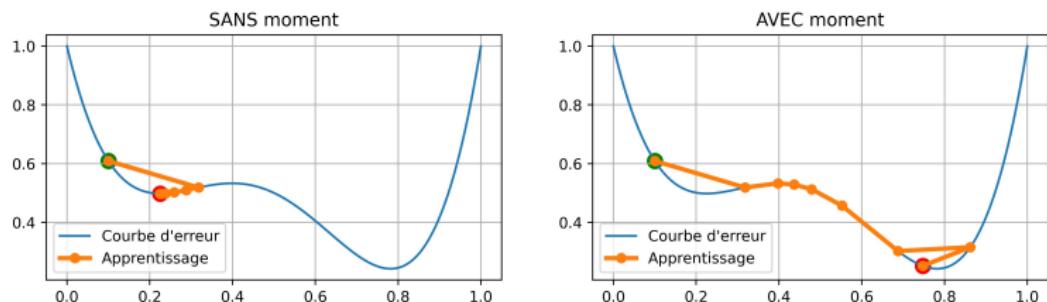
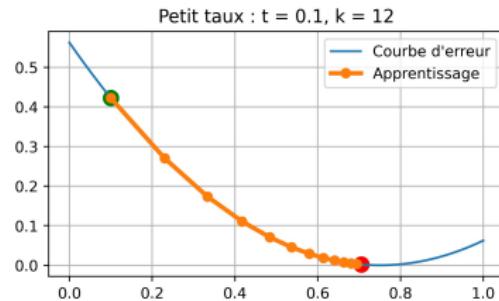
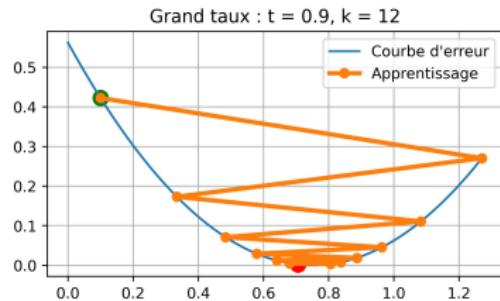


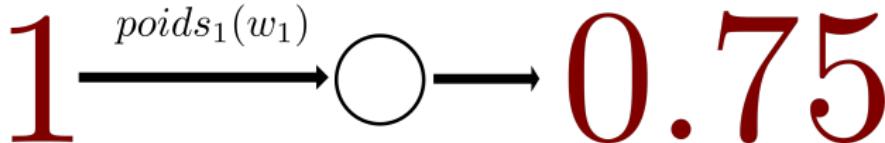
Figure – Comparaison sans puis avec dépendance au moment avec  $\gamma = 0.5$ , arrêt à  $k = 8$

## III - Apprentissage stochastique ou par paquet (Batch)

Les paramètres du réseau de neurone sont les poids qui pondèrent l'entrée. C'est sur eux que l'on opère la descente de gradient.



## Entrée



## Figure – Schéma du perceptron linéaire

### III - Apprentissage stochastique ou par paquet (Batch)

Il faut prendre en compte le fait que les D données d'apprentissage ne sont pas toujours exactes, elles sont forcément inscrites dans une marge d'erreur.

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

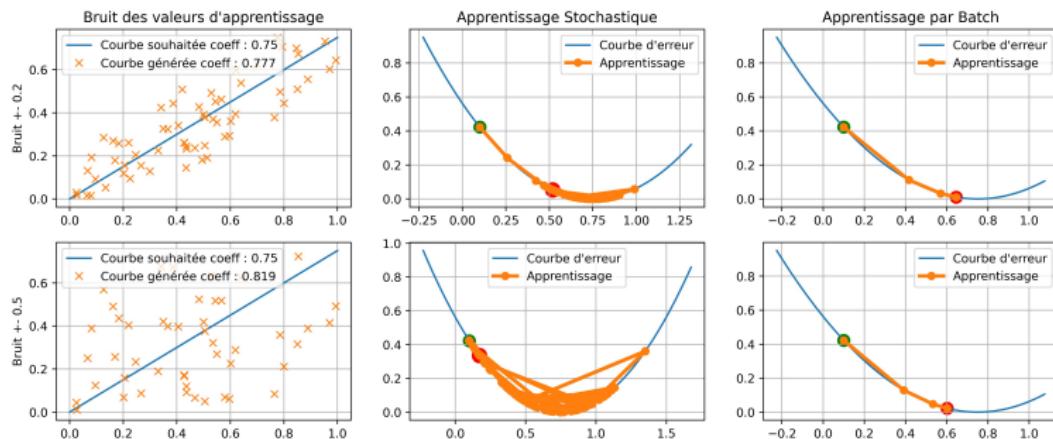


Figure – Comparaison apprentissage stochastique et par paquet,  $f(x) = (x - 0.75)^2$

## 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, pour former des couches cachées, pour reproduire ces opérateurs.

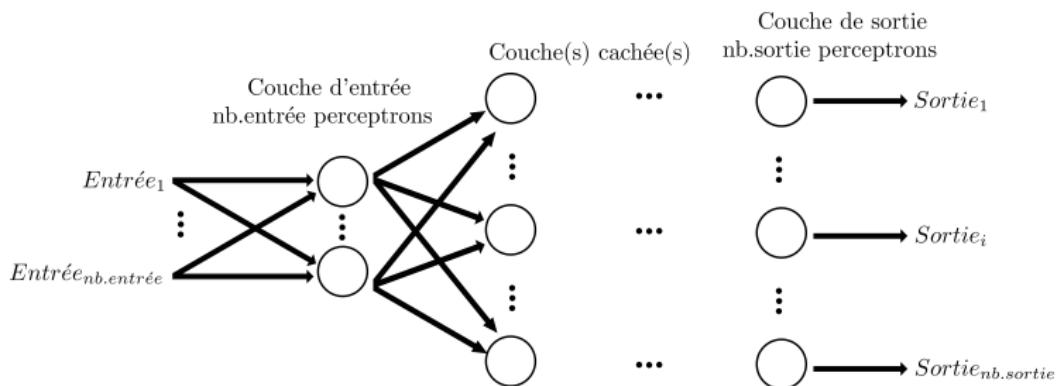


Figure – Schéma d'un réseau de neurones

## 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.

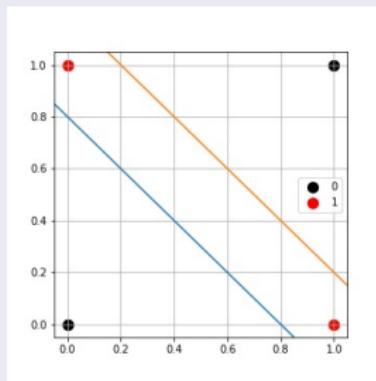


Figure – Schéma de l'opérateur XOR

## IV - Problème de reproduction de l'opérateur XOR

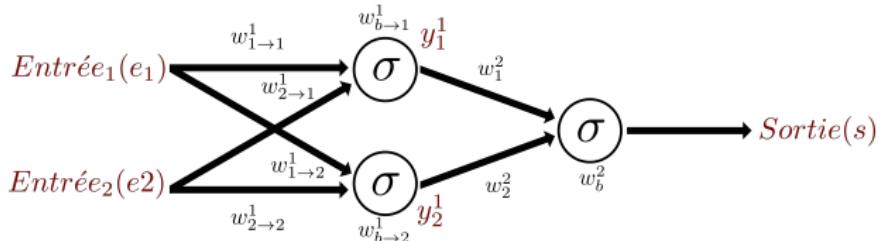


Figure – Schéma du réseau de neurone reproduisant le XOR

### Descente de gradient

$w \leftarrow w - t \frac{\partial f}{\partial w}$  où  $t$  est le taux d'apprentissage et  $f$  la fonction de coût

### Exemple

- $\frac{\partial f}{\partial w_1^2} = 2(s - s_{attendue})\sigma'_2 y_1^1$
- $\frac{\partial f}{\partial w_{1 \rightarrow 1}^1} = 2(s - s_{attendue})\sigma'_2 w_1^2 \sigma'_1 e_1$

## XOR : Erreur au cours des générations

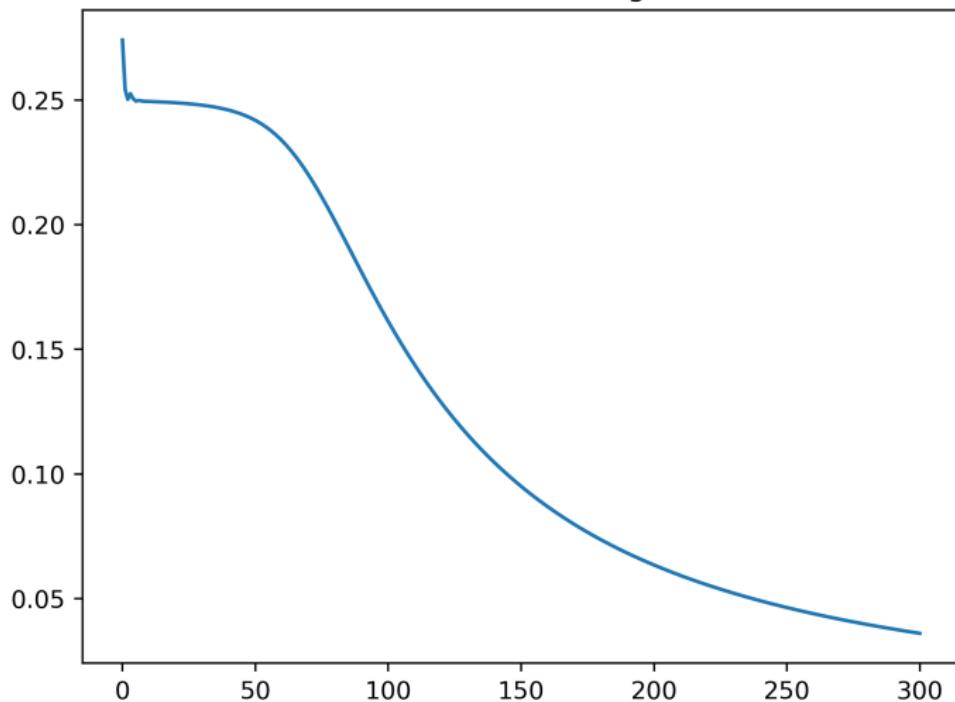


Figure – Courbe de décroissance de l'erreur

## IV - Résultat obtenu

### Données

- 4 données
- 300 générations
- Erreur minimale atteinte 0.036

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \rightarrow \sigma_{couche1} \left( \cdot \times \begin{pmatrix} 0.85 & 5.42 \\ 0.85 & 5.40 \\ 0.14 & 0.44 \end{pmatrix} \right)$$
$$\rightarrow \sigma_{couche2} \left( \cdot \times \begin{pmatrix} -18.39 \\ 14.42 \\ 0.02 \end{pmatrix} \right)$$
$$\rightarrow \begin{pmatrix} 0.12 \\ 0.81 \\ 0.81 \\ 0.24 \end{pmatrix}$$

# V - Reconnaissance d'image

## Problématique

On possède une base de données, d'image de chiffres écrit à la main. Ces images sont toutes de taille  $28 \times 28$  pixels en noir et blanc.

La base de données est divisées en 60000 images pour l'entraînement, et 10000 autres pour la vérification.

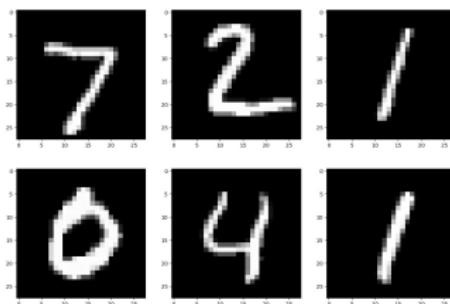


Figure – Exemple d'images

# V - Apprentissage d'un problème de classification

On prend un réseau de neurones avec  $28 \times 28 = 784$  entrées, et 10 sorties.

## Softmax et Cross-entropy

Lorsque l'on fait face à un problème de classification, il faut adapter le réseau de neurone.

La fonction d'activation softmax est utilisé pour la dernière couche de neurones, elle permet de normaliser les probabilités de sorties.

- $p_i = \frac{\exp(a_i)}{\sum_{k=1}^n \exp(e_k)}$  la probabilité de la sortie  $a_i$ ;
- $\frac{\partial p_i}{\partial a_j} = p_i(\delta_{ij} - p_j)$

La fonction de coût associée est *Cross – entropy*.

- $L = -\sum_{k=1}^n y_i \log(p_i)$  avec  $y_i$  la sortie attendue
- $\frac{\partial L}{\partial a_i} = p_i - y_i$

# V - Softmax

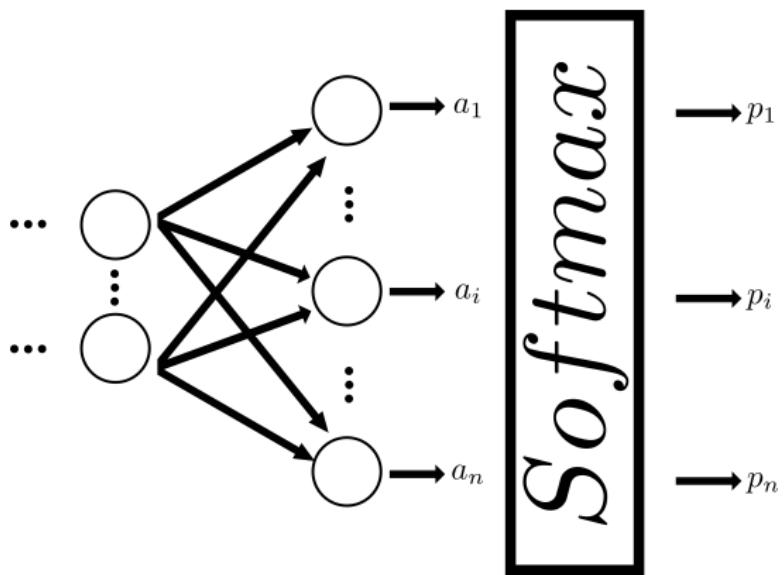


Figure – Schéma d'utilisation du Softmax

## V - Résultats

Au terme de 100 générations d'entraînement, on tend à avoir 91.5% de bonnes réponses sur les données d'entraînement et 91.3% sur les données de validation. On peut remarquer que dès le début, on a un taux de précision d'environ 10%, c'est dû au fait que c'est un problème de classification à 10 sorties.

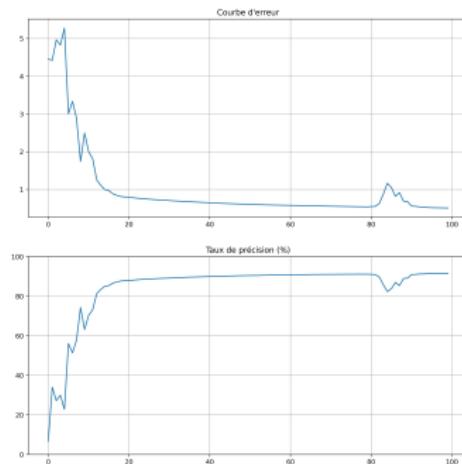


Figure – Courbes d'apprentissage

# V - Résultats



Figure – Exemple sur un échantillon de 40 images de validation

# VI - Reconnaissance de mot

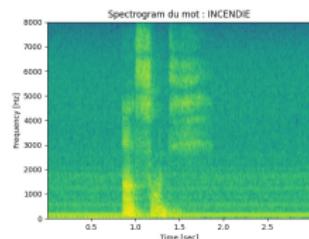
Un moyen simple de reconnaître un mot est d'en faire le spectrogramme puis de résoudre le problème de reconnaissance comme si cela était une image.

Le site du gouvernement recense ces mots clés pour les appels d'urgences :

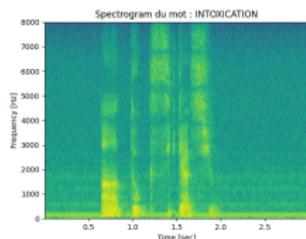
15 malaise, hémorragie, brûlure, intoxication

17 violences, agression, vol, cambriolage

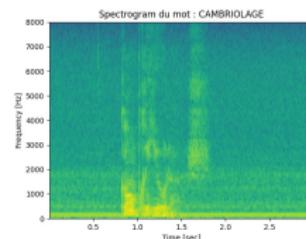
18 incendie, gaz, effondrement, électrocution



(a) INCENDIE



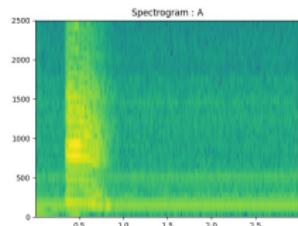
(b) INTOXICATION



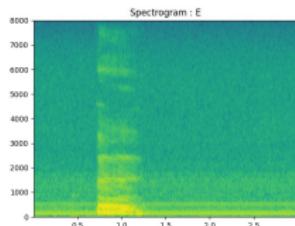
(c) CAMBRIOLAGE

# VII - Reconnaissance vocale

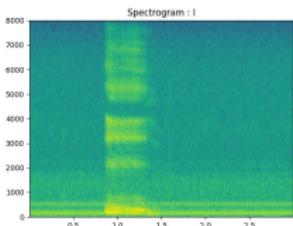
Découpage en élément lexicaux, puis décodage du mot.



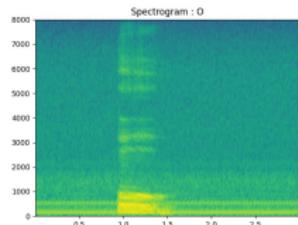
(a) A



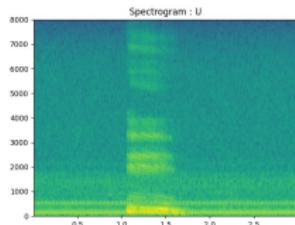
(b) E



(c) I



(a) O



(b) U

[ $\tilde{\varepsilon}$ ]   [s]   [ $\tilde{a}$ ]   [ $\tilde{d}$ ]   [i]  
↓  
[ $\tilde{\varepsilon} \tilde{s} \tilde{a} \tilde{d} i$ ]

INCENDIE