

ISEP

Reconnaissance vocale

Réseaux de neurones

TIPE 2016 - 2017

De TOCQUEVILLE Adrien

FESSARD Grégoire

Sommaire

Introduction

**I- Traitement du signal**

**1) Réduction du bruit**

**2) Calcul des vecteurs caractéristiques**

**a. Fenêtrage et échantillonnage  
 b. Calcul des MFCCs (Mel Frequency Cepstral Coefficients)**

**c. Extraction du vecteur caractéristique**

**3) Reconnaissance de plusieurs locuteurs**

**II- Modèle acoustique**

**1) Les réseaux de neurones**

**2) Réseaux de Kohonen**

**a. Présentation**

**b. Algorithme d’apprentissage  
 c. Paramètres variables**

**d. Représentation graphique**

**III- Modèle linguistique**

**IV- Configuration et expériences**

**V- Annexes**

Conclusion

Sources :

http://www.etsi.org/deliver/etsi\_es/201100\_201199/201108/01.01.03\_60/es\_201108v010103p.pdf

http://www.cs.toronto.edu/~graves/phd.pdf

http://isl.anthropomatik.kit.edu/cmu-kit/downloads/PhD\_1995\_Tebelskis.pdf

http://www.k-netweb.net/projects/tipe/reseauxdeneuronesartificiels.pdf

http://galaxy.agh.edu.pl/~vlsi/AI/backp\_t\_en/backprop.html

http://cache.media.eduscol.education.fr/file/ecole/36/3/mots\_nature\_frequence\_124997\_292363.pdf

[www.recognize-speech.com](http://www.recognize-speech.com)

http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

<http://fr.wikipedia.org/wiki/Carte_auto_adaptative>

https://en.wikipedia.org/wiki/Self-organizing\_map

Table des figures

[Figure 1: Représentation graphique d’un bruit blanc 5](#_Toc482284494)

[Figure 2: Représentation graphique de la fenêtre 5](#_Toc482284495)

[Figure 3: Formation des fenêtres à partir d’un signal échantillonné. 6](#_Toc482284496)

[Figure 4: Étapes du calcul des MFCC 6](#_Toc482284497)

[Figure 5: Illustration de l’application des filtres 8 et 20 à un spectre donné 7](#_Toc482284498)

[Figure 6: Effet du calcul du cepstre sur la fondamentale 9](#_Toc482284499)

[Figure 7: Apprentissage d'un SOM 11](file:///C:\Users\Otarie\Desktop\Ecrit\TIPE.docx#_Toc482284500)

[Figure 8: Un SOM classant des vecteurs de. A gauche, les poids représentés en format RGB et à droite, l’U-Matrice du réseau. 13](#_Toc482284501)

Introduction

La reconnaissance vocale est un domaine sujet à la recherche depuis de nombreuses années. Initialement réalisée à l’aide de chaines de Markov et de systèmes probabilistes, les logiciels de reconnaissance vocale ont connu un essor considérable il y a quelques années grâce aux progrès réalisés dans le domaine de l’intelligence artificielle et des réseaux de neurones. En effet, ce genre de logiciel est parfaitement adapté aux techniques du deep learning (apprentissage profond).

Le but de ce TIPE sera de réaliser un logiciel de reconnaissance vocale capable de transcrire, sous forme de mots, une phrase prononcée par l’utilisateur.

# I- Traitement du signal

## 1) Réduction du bruit

Déjà, il faut distinguer le bruit produit par le micro (par exemple, un grésillement) du bruit créé par l’environnement.

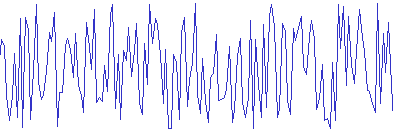


Figure : Représentation graphique d’un bruit blanc

Le premier, appelé bruit blanc, est un signal qui mélange toutes les fréquences. Il est donc impossible de l’éliminer grâce à une transformée de Fourier, car dans le domaine fréquentiel, on ne pourra distinguer le bruit du reste. Nous nous sommes donc intéressés à la décomposition en ondelettes (DWT : discret wavelet transform). Celle-ci à la particularité de dépeindre à la fois le comportement temporel et fréquentiel d’une partie du signal. En supprimant tous les éléments de cette décomposition inferieurs à un certain seuil, on peut supprimer le bruit d’un signal. L’implémentation d’une telle décomposition étant complexe, nous avons utilisé la bibliothèque GSL (GNU Science Library). Cependant, les résultats obtenus détérioraient la qualité du signal sans éliminer la totalité du bruit. Nous nous sommes donc tournés vers un logiciel déjà existant donnant de meilleurs résultats.

Nous avons choisi d’utiliser le logiciel SoX. Pour l’utiliser, il faut lui donner un échantillon sonore composé uniquement du bruit généré par le micro afin qu’il établisse un profil du bruit. On peut ensuite utiliser ce profil pour retirer le bruit de tout signal enregistré à l’aide de ce micro.

Le second type de bruit est plus difficile à enlever puisqu’il est extrêmement variable et imprédictible. Aucun effort particulier n’a été fait dans le but de le réduire et la reconnaissance faite par le logiciel sera donc plus efficace dans des environnements relativement calmes.

## 2) Calcul des vecteurs caractéristiques

### a. Fenêtrage et échantillonnage

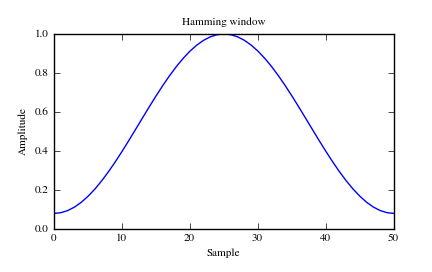
**Le signal sonore que l’on cherche à analyser est le plus souvent non stationnaire. Cependant, celui-ci est constitué d’une suite de phonèmes dont les propriétés ne varient pas pendant une durée pouvant aller de 5 à 100 millisecondes. Ainsi, on supposera que sur une fenêtre de quelques millisecondes, le signal est stationnaire. On utilisera pour cela le fenêtrage de Hamming défini par l’expression suivante :

Figure : Représentation graphique de la fenêtre

La valeur de N étant adaptée pour couvrir 25 ms du signal ( avec la fréquence d’échantillonnage).

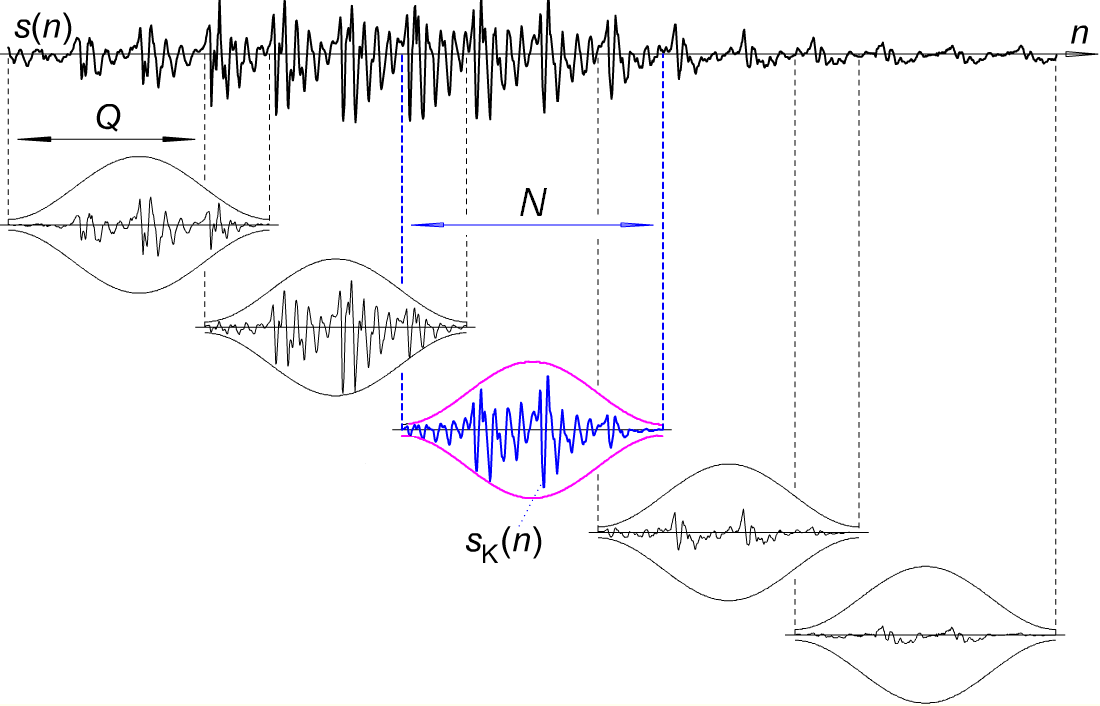
Pour ne pas perdre les informations correspondant aux amplitudes faibles de la fonction de fenêtrage, on créera des fenêtres superposées pendant un laps de temps adapté à la longueur de la fenêtre.

Figure : Formation des fenêtres à partir d’un signal échantillonné.

L’expression de la fonction correspondant à la k-ième fenêtre est donc la suivante :

Les valeurs de Q et N ainsi que la fonction de fenêtrage sont adaptables. Celles données ci-dessus sont classiques pour effectuer de la reconnaissance vocale et permettent de trouver un bon compromis entre la quantité de calculs et la précision des résultats. La fenêtre de Hamming évite d’importantes discontinuités aux bords de la fenêtre mais pourrait être remplacé par une fenêtre de Hann qui présente des propriétés similaires.

A ce stade, on peut analyser les signaux tronqués sont indépendants. Il faut maintenant extraire le vecteur caractéristique de chaque fenêtre en déterminant ses coefficients.

### b. Calcul des MFCCs (Mel Frequency Cepstral Coefficients)

### Les premiers coefficients d’un vecteur caractéristique sont appelés MFCC. Des études sur les performances d’un système de reconnaissance vocale en fonction du nombre de MFCC ont montré que 13 coefficients peuvent donner de très bons résultats. Les calculs se font en suivant les étapes présentées ici.

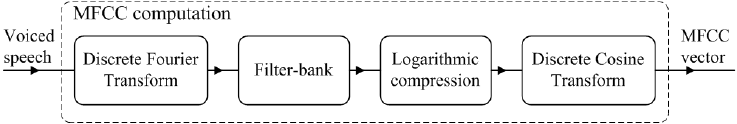


Figure : Étapes du calcul des MFCC

**Transformée de Fourier**

La première étape du calcul des MFCC consiste à calculer la transformée de Fourier du signal tronqué. Ceci est fait grâce à un algorithme de FFT pour économiser du temps de calcul. On complétera ce signal par des zéros de manière à ce qu’il comporte 512 points.

Un signal sonore étant à valeurs réelles, sa transformée est symétrique. On ne considère donc que les M = 257 premiers points.

**Densité spectrale de puissance et filtres passe bande**

Il faut ensuite calculer la densité spectrale de puissance de la fenêtre, définie par :

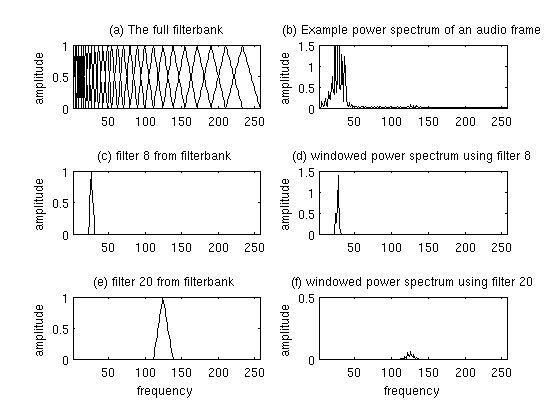
On peut maintenant filtrer cette fonction avec filtres passes bandes en échelle mel. Ce filtrage renvoie un nombre réel par filtre qui donne une estimation de la puissance spectrale du signal dans une tranche de fréquence.

Figure : Illustration de l’application des filtres 8 et 20 à un spectre donné

Les filtres sont définis de la manière suivante :

On pose d’abord une fréquence minimum et maximum que l’on passe en échelle mel. On choisit respectivement 0 *Hz* et la fréquence de Nyquist (8 *kHz*), ce qui donne 0 *mel* et 2840 *mel*.

La conversion de *Hertz* vers *mel* se fait de la manière suivante :

On calcule ensuite la fréquence centrale de chaque filtre en les espaçant linéairement entre 0 et 2840 *mel*. En repassant ces valeurs en Hertz, on obtient des filtres espacés logarithmiquement selon l’échelle des fréquences.

Pour pouvoir être appliquées au spectre du signal, il faut aligner ces valeurs sur les paramètres de la FFT :

Chaque valeur représente la fréquence correspondant au pic d’un filtre. On définit le m-ième filtre par :

Une fois filtrées, on obtient les puissances suivantes :

**Transformée en cosinus discrète (DCT-II)**

La dernière étape sert à essayer d’éliminer les caractéristiques dépendant du locuteur en calculant les coefficients cepstraux. Le cepstre d’un signal est défini de la manière suivante :

Le cepstre est donc le spectre d’un spectre. Les harmoniques dépendant du locuteur de la fréquence fondamentale sont donc transformées un coefficient cepstral d’ordre supérieur (figure 6b). La transformation inverse des coefficients les plus bas est montrée en figure 6c et celle des coefficients les plus hauts en 6d. En conservant uniquement les premiers coefficients, on élimine les parasites crées par l’appareil vocal sur le son. Les 13 premiers coefficients forment donc les MFCCs. On n’utilise pas une TFD mais plutôt une transformée en cosinus discrète car celle-ci est plus efficace lorsqu’il s’agit de compresser des données.

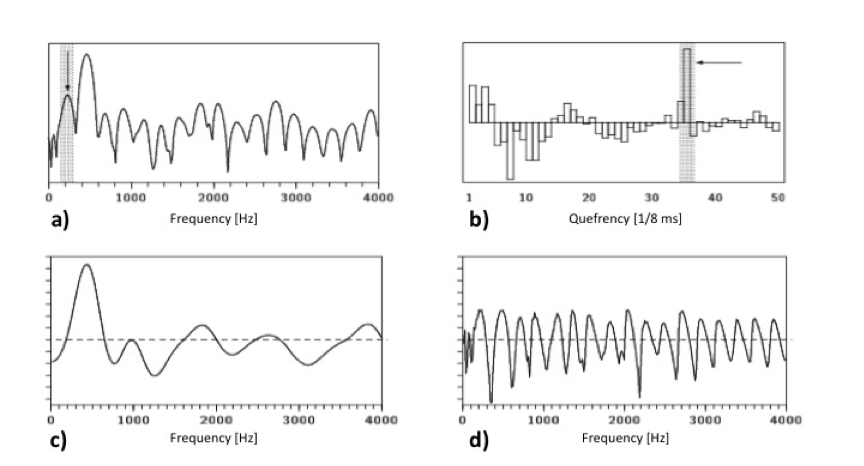


Figure : Effet du calcul du cepstre sur la fondamentale

### c. Extraction du vecteur caractéristique

Une fois les 13 coefficients spectraux calculés, on calcule leurs dérivées premières et secondes.

Le vecteur caractéristique est composé de ces trois vecteurs mis bouts à bouts pour un total de 39 coefficients.

Ces opérations ont deux effets primordiaux :

Premièrement, on passe d’une liste de 0.025\*16000=400 valeurs par fenêtre à seulement 39. Ceci permet de réduire de façon significative la durée d’apprentissage du réseau de neurones.

D’autre part, les données extraites sont plus représentatives du phonème prononcé et permettent un apprentissage plus simple et des résultats plus précis.

## 3) Reconnaissance de plusieurs locuteurs

Il existe de deux manières pour créer un système utilisable par plusieurs locuteurs. La première consiste à avoir une base de données contenant suffisamment d’exemple prononcés par une large gamme de voix et la seconde de demander à l’utilisateur de prononcer un ensemble de phrases donné pour calibrer le système, notamment en mesurant les différences avec une prononciation de référence pour établir une proportionnalité entre la voix de l’utilisateur et celle utilisée pour former la base de donnée. La seconde méthode est généralement plus efficace mais demande plus d’effort pour l’utilisateur.

Lorsque l’on calcule un MFCC, le passage au domaine fréquentiel et les filtrages effectués permettent d’éliminer une grande quantité d’éléments dépendant du locuteur, mais l’efficacité n’est pas suffisante. En utilisant la deuxième méthode donnée, on peut récupérer certaines informations sur le timbre vocal de l’utilisateur et ainsi calibrer certains coefficients, notamment les fréquences centrales de la banque de filtre.

Toutefois, les MFCCs se montrent peu efficaces pour reconnaitre des voix aigües. Couplé avec une base de données très maigre, le système développé ne pourra probablement pas reconnaitre beaucoup de locuteurs.

**II- Modèle acoustique**

Un fois le vecteur caractéristique obtenu, on utilise un modèle acoustique pour détecter le phonème prononcé dans la fenêtre étudiée. Les techniques utilisées à cet effet sont multiples, les principales étant les automates de Markov à états cachés (HMM : Hidden Markov Models) et les réseaux de neurones. Dans cette dernière catégorie, il existe plusieurs variantes que nous présenterons brièvement avant de détailler celle que nous avons choisie.

**1) Les réseaux de neurones**

Un réseau de neurones artificiels, est un ensemble d'algorithmes dont la conception est à l'origine très schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.

Il existe une multitude de réseaux de neurones chacun ayant ses avantages et inconvénients.  
Les plus performants théoriquement sont ceux utilisant les méthodes du deep learning. En effet, dans la première partie, nous avons vu comment calculer les vecteurs caractéristiques d’un fichier audio grâce à une méthode pouvant paraître compliquée à trouver. De plus, il serait nécessaire de créer un autre algorithme si nous voulions classifier d’autres types de données, tel que des images. L’apprentissage profond permet de remplacer ce processus laborieux par un algorithme n’ayant aucune connaissance spécifique du problème étudié. Ces réseaux sont toutefois très complexes et nécessitent beaucoup de temps et de puissance de calcul pour arriver à un résultat.

Nous avons préféré les réseaux de neurones aux HMM tout d’abord parce que ils constituent une part importante de la recherche actuelle et qu’ils sont présent dans de plus en plus de logiciels et trouvent des applications dans une large gamme de domaines. Pour la reconnaissance vocale, ils ont complètement remplacé les HMM grâce aux avantages suivants :

* Leur capacité d’entrainement. Les réseaux peuvent apprendre n’importe quelle association d’entrées-sorties. Ceci est utile pour entrainer le réseau à classifier des échantillons de voix dans des catégories de phonèmes.
* Leur généricité. Les réseaux ne font pas que mémoriser la base de données, ils apprennent leur forme sous-jacente. Ils peuvent donc généraliser ce qu’ils ont appris à de nouveaux exemples. C’est essentiel en reconnaissance vocale car les sons prononcés ne sont jamais exactement les mêmes.
* Leur non-linéarité. Les réseaux peuvent déterminer une régression non linéaire et non paramétrique correspondant à la base de données, ce qui leur permet d’apprendre des motifs en apparence très complexes.

Parmi tous les algorithmes existant pour résoudre le problème, on distingue trois grandes catégories :

* Les algorithmes supervisés. On soumet au réseau une entrée et on compare la réponse à la sortie désirée puis on adapte le réseau de sorte à minimiser l’erreur produite. Un tel algorithme nécessite une base de données pour laquelle on aurait une sortie désirée pour chaque entrée.
* Les algorithmes non supervisés. On soumet au réseau une liste d’entrées et celui-ci se charge de classifier les entrées en un certain nombre de catégories. Pour de la reconnaissance vocale, il faut donner au réseau des échantillons de voix sans indiquer ce qu’ils signifient.
* Les algorithmes semi-supervisés. Ces algorithmes utilisent un ensemble de données étiquetées et non-étiquetés. Il se situe entre les deux autres.

Les réseaux les plus adaptés à la reconnaissance artificielle sont les réseaux de neurones récurrents car ils prennent en compte le comportement temporel des données. Cependant, les algorithmes d’apprentissage sont supervisés et nous voulions un algorithme non supervisé puisqu’il facilitait la création de la base de données.

Nous avons donc opté pour les réseaux de Kohonen qui répondait à ce critère.

**2) Réseaux de Kohonen**

### a. Présentation

Les réseaux de Kohonen, ou carte auto adaptative (SOM : Self Organizing Map), initialement conçus pour la reconnaissance vocale, permet de réaliser des tâches de réduction de dimension et de classification. Nous les utiliserons pour classifier les vecteurs caractéristiques selon des critères sélectionnés dans la base de données lors de la phase d’apprentissage

Un tel réseau se présente sous la forme d’un tableau à 2 dimensions dont chacune des cases contient un vecteur à 26 dimensions. Une case est appelée neurone et le vecteur qui lui correspond est son poids.

Après l’apprentissage, cette carte fournira une représentation discrète de l'espace d'entrée, permettant d’indexer des données via leurs coordonnées dans le tableau. La structure topologique de l’espace de départ est conservée lors de la réduction de dimension.

### b. Algorithme d’apprentissage

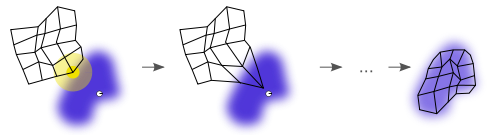


Figure : Apprentissage d'un SOM

Sur la figure 6, on peut voir une illustration de l’apprentissage d’un SOM de dans. La forme bleue représente les données et le disque blanc est le vecteur en train d’être traité. Le poids du neurone gagnant ainsi que celui de son voisinage (en jaune) est adapté. Au fur et à mesure de l’avancée de l’algorithme, la carte classifie de mieux en mieux l’ensemble des données.

Ce réseau est donc fondé sur une méthode d’apprentissage non supervisé. Il se distingue par l’utilisation d’une fonction de voisinage qui conserve la topologie de l’espace de départ et par un apprentissage compétitif, en opposition à un apprentissage par correction de l’erreur: ici, les neurones répondant à une entrée seront « récompensés ».

L’initialisation des valeurs de poids pour chaque neurone se fait aléatoirement. Ensuite, l’algorithme est itératif :

A chaque itération, on soumet un vecteur choisi aléatoirement dans la base de données à la carte auto adaptative. Selon les valeurs des poids des neurones, il en sortira un neurone gagnant, appelé BMU (Best Matching Unit). On adapte alors tous les neurones présents dans le voisinage du BMU, en rapprochant leur poids du vecteur d’entrée. Ceci leur permettra de répondre encore mieux à d’autres entrées semblables à celle-ci. Ainsi, c'est toute la région de la carte autour du neurone gagnant qui se spécialise.

A chaque itération, le voisinage d’un neurone se rétrécit et la modification des poids devient de moins en moins importante.

**Formalisation mathématique**

et désignent l’itération actuelle et le nombre d’itérations.  
 désigne le nombre de neurones dans la grille.  
 désigne le poids du neurone dans la grille.  
 désigne la position du neurone sur la grille.  
 désigne la norme euclidienne.

1. On donne une valeur aléatoire aux poids de chaque vecteur.

2. On sélectionne un vecteur d’entrée.

3. On note un neurone tel que. est le BMU.

4. On adapte les poids des neurones dans le voisinage du BMU selon la formule suivante :

Avec,  
 donnant le rayon du voisinage d’un neurone à l’étape  
 et donnant le coefficient d’apprentissage à l’étape.

5. On incrémente et on recommence depuis l’étape 2 tant que.

Une variante de l’algorithme consiste à parcourir tous les vecteurs de la base de données à chaque itération.

### c. Paramètres variables

Le choix des paramètres de l’algorithme est très important. On en compte 7 :

* Les dimensions de la carte
* Le nombre d’itérations
* Les deux valeurs extrêmes du rayon
* Les deux valeurs extrêmes du coefficient d’apprentissage

De plus, l’apprentissage se déroule souvent en deux fois. Une première fois avec peu d’itérations et un rayon et un coefficient élevé pour rapprocher la carte de l’ensemble des données. Puis une seconde fois de manière plus fine et avec un nombre d’itérations beaucoup plus élevé.

### d. Représentation graphique

### Pour pouvoir visualiser les résultats du réseau, on calcule généralement l’U-Matrice du réseau. Cela consiste à calculer la distance moyenne séparant le poids d’un neurone de ceux de ses voisins puis de l’afficher sur une carte. Plus cette distance sera élevée, plus le neurone sera représenté avec une couleur foncée.

### Une telle représentation permet de rapidement visualiser les groupes formés pour ensuite les classer.

Cependant, lorsque les délimitations entre les groupes sont moins marquées, il est parfois impossible d’attribuer à un neurone de la carte une seule catégorie. On peut donc créer une troisième carte sur laquelle il y aura les catégories possibles auxquelles peut appartenir un neurone.

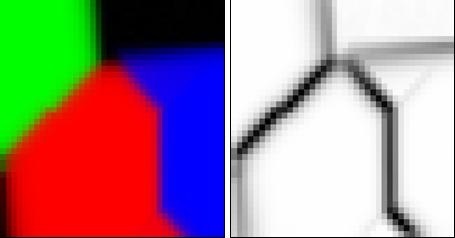


Figure : Un SOM classant des vecteurs de. A gauche, les poids représentés en format RGB et à droite, l’U-Matrice du réseau.

**III – Conversion phonèmes – français**

**IV- Configuration et expériences**

Le système total possède un nombre important de paramètres pour la plus part définis grâce des expérimentations. Les valeurs choisies pour le calcul des MFCCs ont toutes étés données en partie I et sont des valeurs classiques utilisées par un grand nombre de logiciels de reconnaissance vocal.

Pour la première phase de test, nous avons essayé de reconnaitre une base de données contenant la prononciation des chiffres de zéro à trois. Cela représente 10 phonèmes différents rendant l’apprentissage du réseau assez rapide et nous permettant de réaliser de nombreux tests avec des paramètres différents et ainsi voir leur influence sur les résultats obtenus.

Nous sommes ensuite passés à une base de données constituée de deux phrases contenant chacune la totalité des phonèmes du français. Une fois l’apprentissage fait, la reconnaissance était efficace pour les voyelles mais peu pour les consonnes. Nous en avons déduit que le problème venait en partie de la base de données. En effet, les voyelles étaient naturellement très présentes dans les phrases choisies tandis que certaines consonnes n’apparaissaient qu’une seule fois (les phonèmes peuvent être classés en voyelles et consonnes, voir en annexe).

Nous avons donc changé de base de données pour en choisir une qui contienne chaque couple de consonne puis voyelle possible. Celle-ci regroupe donc tous les phonèmes choisis de manière plus ou moins égale.

Les phonèmes choisis sont au nombre de 30. En effet, bien qu’il existe 36 phonèmes en français, la différence entre certains est très fine voire même inexistante chez certains locuteurs. De plus, nous avons choisi de rajouter un phonème représentant un silence. La liste est donnée en annexe.

Le réseau de Kohonen sera une carte de vecteurs de.

Les fonctions donnant le coefficient d’apprentissage et le rayon en fonction de l’étape sont celles définies en II-3)b. avec les paramètres suivants.

|  |  |  |
| --- | --- | --- |
| Variable | Première phase | Seconde phase |
|  | 150 | 10000 |
|  | 0.5 | 0.3 |
|  | 0.1 | 0.001 |
|  | 13.6 | 6.8 |
|  | 6.8 | 0.001 |

Tableau : Valeurs des paramètres pour chacune des phases d’apprentissage

**V- Annexes**

La liste des phonèmes utilisés est :

|  |  |
| --- | --- |
| Phonème utilisé - type | Phonème(s) remplacés en notation internationale (API) |
| I - voyelle | /i/ |
| ET - voyelle | /e/ |
| AI - voyelle | /ɛ/ |
| A - voyelle | /a/ /ɑ/ |
| AU - voyelle | /ɔ/ |
| O - voyelle | /o/ |
| OU - voyelle | /u/ |
| U - voyelle | /y/ |
| E - voyelle | /ø/ /ə/ |
| EU - voyelle | /œ/ |
| AN - voyelle | /̃ɑ/ |
| ON - voyelle | /̃ɔ/ |
| IN - voyelle | /̃ɛ/ /̃œ/ |
| P - consonne | /p/ |
| T - consonne | /t/ |
| K - consonne | /k/ |
| B - consonne | /b/ |
| D - consonne | /d/ |
| G - consonne | /g/ |
| F - consonne | /f/ |
| S - consonne | /s/ |
| CH - consonne | /ʃ/ |
| V - consonne | /v/ |
| Z - consonne | /z/ |
| J - consonne | /ʒ/ |
| L - consonne | /l/ |
| R - consonne | /ʁ/ |
| M - consonne | /m/ |
| N - consonne | /n/ /ɲ/ |