Projet informatique
Réseaux neuronaux et machine learning

Denis Priou – v3.1 (24/06/2019)

Durant la PED de langage C, nous avons découvert le fonctionnement élémentaire d'un neurone formel et d'un petit réseau de neurones. Nous avons aussi vu une formulation simplifiée des méthodes de descente de gradient et de rétropropagation des erreurs. Ces concepts ont été mis en œuvre sur un exemple très simple, celui de la reconnaissance d'un type d'iris parmi les trois possibles (setosa, versicolor, virginica), dans une population de 150 échantillons.

L'objectif de ce projet est de formuler ces concepts de façon plus générale et de les mettre en œuvre sur des données et sur des réseaux de neurones plus complexes. Il s'agit d'un premier passage à l'échelle. Nous manipulerons environ un millier de neurones répartis dans trois couches (entrée, cachée, sortie). Nous appliquerons les méthodes du machine learning à la reconnaissance de caractères et utiliserons des jeux de données contenant plusieurs dizaines de milliers d'échantillons. Pour plus de détail sur les méthodes employées, on pourra consulter l'article intitulé « Gradient-Based Learning Applied to Document Recognition » (Yann LeCun et al., Proc. of the IEEE, nov. 1998).

1 – Reconnaissance de caractères

Nous allons construire un réseau de neurones capable de reconnaître des caractères manuscrits ou typographiés (plus précisément, des chiffres : 0, 1, 2, ..., 9), fournis au réseau sous la forme d'une image (un bitmap). Nous allons tester ce réseau sur un jeu de données de référence, le « MNIST dataset » (voir http://yann.lecun.com/exdb/mnist/). Nous pourrons appliquer le même code informatique à un jeu de données constitué plus récemment, le « Fashion-MNIST », contenant les images de vêtements Zalando (voir https://www.kaggle.com/zalando-research/fashionmnist et https://github.com/zalandoresearch/fashion-mnist). Dans ces images, il s'agit de reconnaître les différents types de vêtement.

1.1 – Le jeu de données MNIST

Le jeu de données MNIST a été conçu pour tester l'efficacité de différents algorithmes d'intelligence artificielle à reconnaître des chiffres manuscrits. Chaque image est un bitmap de 28 pixels par 28 représentant un chiffre (0, 1, ..., 9). Chaque pixel du bitmap correspond à une intensité en niveaux de gris allant de 0 à 255. Le MNIST dataset est constitué d'un jeu d'apprentissage de 60000 caractères et d'un jeu de test de 10000 caractères. Tous ces caractères ont été écrits à la main par différents « cobayes » humains et ont été numérisés et pré-traités. La diversité graphologique et l'aspect « données réelles » font de ce jeu de données un test standard pour les algorithmes de reconnaissance de caractères (voir figure 1).

Le jeu de données MNIST est fourni sous la forme de quatre fichiers :

- Le fichier `train-images-idx3-ubyte`, qui contient les bitmaps du jeu d'apprentissage : ce fichier débute par une clé « magique » de 32 bits (`0x00000803`), suivie de 3 entiers de 32 bits indiquant le nombre de bitmaps (60000), leur largeur (28 colonnes) et leur hauteur (28 lignes) ; on trouve ensuite 60000 séquences de 28 x 28 octets non signés (les 60000 bitmaps, stockés ligne par ligne) ;
- Le fichier `train-labels-idx1-ubyte`, qui contient la liste des chiffres représentés dans les bitmaps du jeu d'apprentissage (dans le même ordre) : ce fichier débute par une clé magique

de 4 octets (`0x00000801`), suivie du nombre de valeurs sur 4 octets (60000) ; on trouve ensuite la valeur de ces 60000 chiffres (octets non signés).

- Les fichiers `t10k-images-idx3-ubyte` et `t10k-labels-idx1-ubyte` contiennent respectivement les bitmaps du jeu de test et la valeur des chiffres représentés : leur structure est identique à celle des fichiers d'apprentissage, à ceci près que le deuxième entier de 32 bits vaut 10000 (et non 60000).

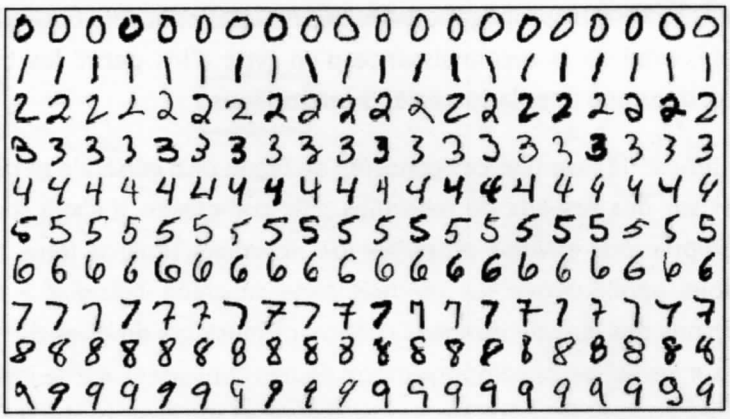

Figure 1 – Quelques exemples de caractères tirés du jeu de données MNIST

1.2 – Le jeu de données Fashion-MNIST

Le jeu de données Fashion-MNIST a été conçu pour renouveler les travaux portant sur le jeu originel ci-dessus. Les performances des algorithmes standard sur le jeu de données MNIST dépassent souvent 97% et peuvent atteindre 100%. Il s'agissait, pour les auteurs du Fashion-MNIST, de proposer un jeu de données structurées exactement de la même façon, informatiquement parlant, ce qui facilite le recyclage du code développé pour le MNIST. Les éléments techniques permettant la lecture des données MNIST, décrits dans la section précédente, peuvent donc être repris à la lettre.

La reconnaissance des motifs dans les images Fashion-MNIST est cependant plus difficile : on peut assez facilement atteindre des taux de l'ordre de 90%. Aller au-delà nécessite un travail plus innovant.

Les images du jeu Fashion-MNIST ont l'allure de la figure 2. On y trouve 10 types différents de vêtements, selon la classification suivante : T-shirt/top (0), Trouser (1), Pullover (2), Dress (3), Coat (4), Sandal (5), Shirt (6), Sneaker (7), Bag (8), Ankle boot (9).

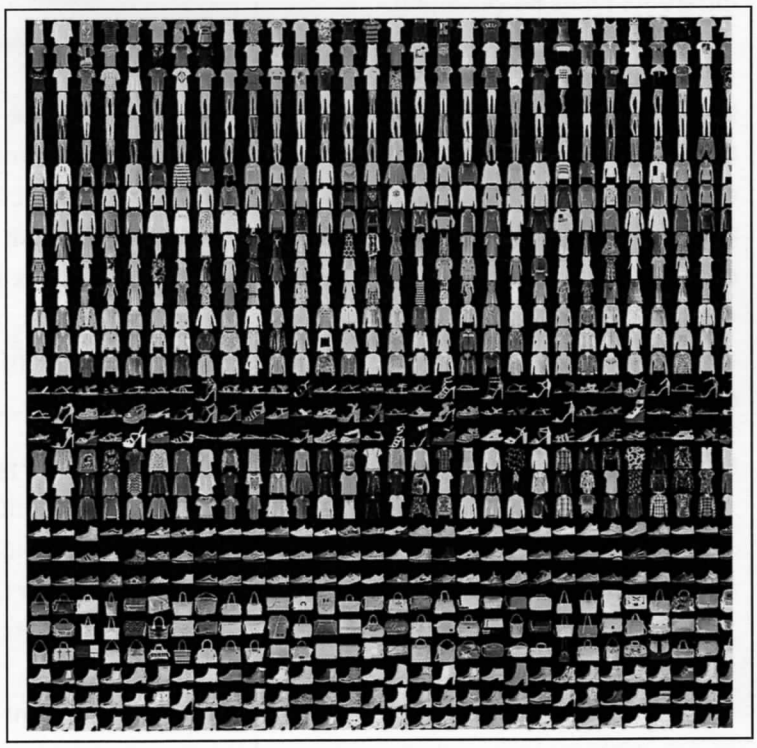

Figure 2 – Quelques exemples de vêtements tirés du jeu de données Fashion-MNIST

2 – Réseau de neurones retenu pour traiter les jeux de caractères

Définissons la structure et le fonctionnement du réseau neuronal qui traitera les données MNIST et Fashion-MNIST : nombre de couches, nombre de neurones par couche, méthodes de traitement, etc. Nous en profiterons pour reformuler en toute généralité les concepts fondamentaux présentés dans le chapitre précédent : neurone formel, couche et réseau de neurones, algorithme de rétro-propagation des erreurs, algorithme de descente de gradient. Nous définirons au passage quelques notations utiles dans la suite de ce chapitre.

2.1 – Structure retenue pour le réseau neuronal

Notre réseau de neurones prend en entrée les bitmaps du jeu de données, l'un après l'autre, et formule pour chacun une prédiction (est-ce un zéro ? est-ce un chiffre 1 ? etc.). Ce réseau comporte trois couches : la couche d'entrée, la couche cachée et la couche de sortie.

La couche d'entrée reçoit les données d'entrée, à savoir le bitmap à analyser : chaque neurone de cette couche prend en entrée la valeur de l'un des pixels du bitmap. La couche d'entrée comporte donc (NB_LIGNES x NB_COLONNES) neurones, où NB_LIGNES et NB_COLONNES désignent respectivement le nombre de lignes et le nombre de colonnes d'un bitmap. Nous complétons cette couche par un neurone de biais : la couche d'entrée comprend donc (1 + NB_LIGNES x NB_COLONNES) neurones. Le neurone de biais a pour indice 0 ; les neurones d'indice 1, 2, ..., (NB_LIGNES x NB_COLONNES) reçoivent la valeur de chacun des (NB_LIGNES x NB_COLONNES) pixels. On note $N_{entrée}$ le nombre total de neurones de la couche d'entrée. Pour le jeu de données MNIST, $N_{entrée} = 785$.

Il est envisageable de procéder à un *maxpooling* des images, avant passage dans le réseau de neurones. Il s'agit de diminuer la taille des images en fusionnant 4 pixels voisins (en carrés), et en retenant le pixel de valeur maximale. Si l'on supprime préalablement la première et la dernière colonne, ainsi que la première et la dernière ligne de l'image, sa taille passe de 28 pixels par 28 à 26 pixels par 26, puis à 13 pixels par 13. Le nombre de neurones en entrée est alors égal à $N_{entrée} = 170$. On pourra comparer l'efficacité du réseau de neurones avec et sans *maxpooling*, ainsi que le temps d'exécution, toutes choses égales par ailleurs.

La couche de sortie produit dix valeurs, à savoir la probabilité pour le bitmap fourni en entrée du réseau représente le chiffre 0, le chiffre 1, ..., le chiffre 9. Cette couche comporte donc 10 neurones. On note N_{sortie} le nombre de neurones de la couche de sortie, pour conserver aux explications leur caractère de généralité.

La couche cachée n'a pas de taille contrainte, à la différence des deux autres couches. Le nombre de neurones cachés, que l'on note $N_{caché}$, sera compris entre $N_{entrée}$ et N_{sortie}. On testera l'efficacité prédictive du réseau en faisant varier ce nombre. La couche cachée contiendra un neurone de biais, d'indice 0, et ($N_{caché}$–1) neurones « actifs » (c'est-à-dire recevant les informations produites par la couche d'entrée).

2.2 – Couche d'entrée

La couche d'entrée est représentée dans la figure 3. Cette couche comprend un nombre de neurones égal à $N_{entrée}$ = (1 + NB_LIGNES x NB_COLONNES). Le neurone d'indice 0 est le neurone de biais : il fournit systématiquement la valeur 1 à la couche cachée. Les neurones d'indice 1 à NB_COLONNES fournissent à la couche cachée la valeur des pixels de la première ligne du bitmap. Les neurones d'indice (NB_COLONNES+1) à (2 x NB_COLONNES) fournissent la valeur des pixels de la deuxième

colonne, et ainsi de suite. La valeur des pixels de la dernière colonne est fournie par les neurones d'indice [(NB_LIGNES–1)xNB_COLONNES + 1] à (NB_LIGNESxNB_COLONNES).

Toutes ces informations sont transmises aux dendrites des neurones de la couche cachée. Dans la figure 3, les dendrites reliant les neurones d'entrée au neurone caché d'indice 1 sont représentés en bleu. Les poids synaptiques associés sont w_{01}, w_{11}, ..., $w_{(NB\_LIGNESxNB\_COLONNES)(1)}$. Les dendrites du neurone caché d'indice j sont représentés en rouge. Les poids synaptiques associés sont w_{0j}, w_{1j}, ..., $w_{(NB\_LIGNESxNB\_COLONNES)(j)}$. Enfin, les dendrites du dernier neurone d'entrée, d'indice $N_{entrée} - 1$ = (NB_LIGNES x NB_COLONNES), figurent en vert. Les poids synaptiques associés sont $w_{(0)(Ncaché-1)}$, $w_{(1)(Ncaché-1)}$, ..., $w_{(NB\_LIGNESxNB\_COLONNES)(Ncaché-1)}$. De façon générale, le neurone d'entrée « i » transmet ses informations au neurone caché « j » via sa dendrite de poids synaptique w_{ij}.

Convention de notation ! Dans la suite de ce chapitre, on utilisera systématiquement la lettre « i » pour désigner l'indice d'un neurone de la couche d'entrée. La valeur de « i » pourra donc varier de 0 à ($N_{entrée}$–1). On notera systématiquement X_i la valeur transmise par le neurone d'entrée d'indice i à la couche cachée. On utilisera systématiquement le lettre « j » pour désigner l'indice d'un neurone de la couche cachée. La valeur de « j » pourra donc varier de 0 à ($N_{caché}$–1).

Pour le jeu de données MNIST, NB_LIGNES et NB_COLONNES sont égaux à 28 (sans *maxpooling*) : la couche d'entrée comporte donc 1 + 28 x 28 = 785 neurones. Avec *maxpooling*, les images font 13 pixels par 13 : la couche d'entrée comporte donc 1 + 13 x 13 = 170 neurones.

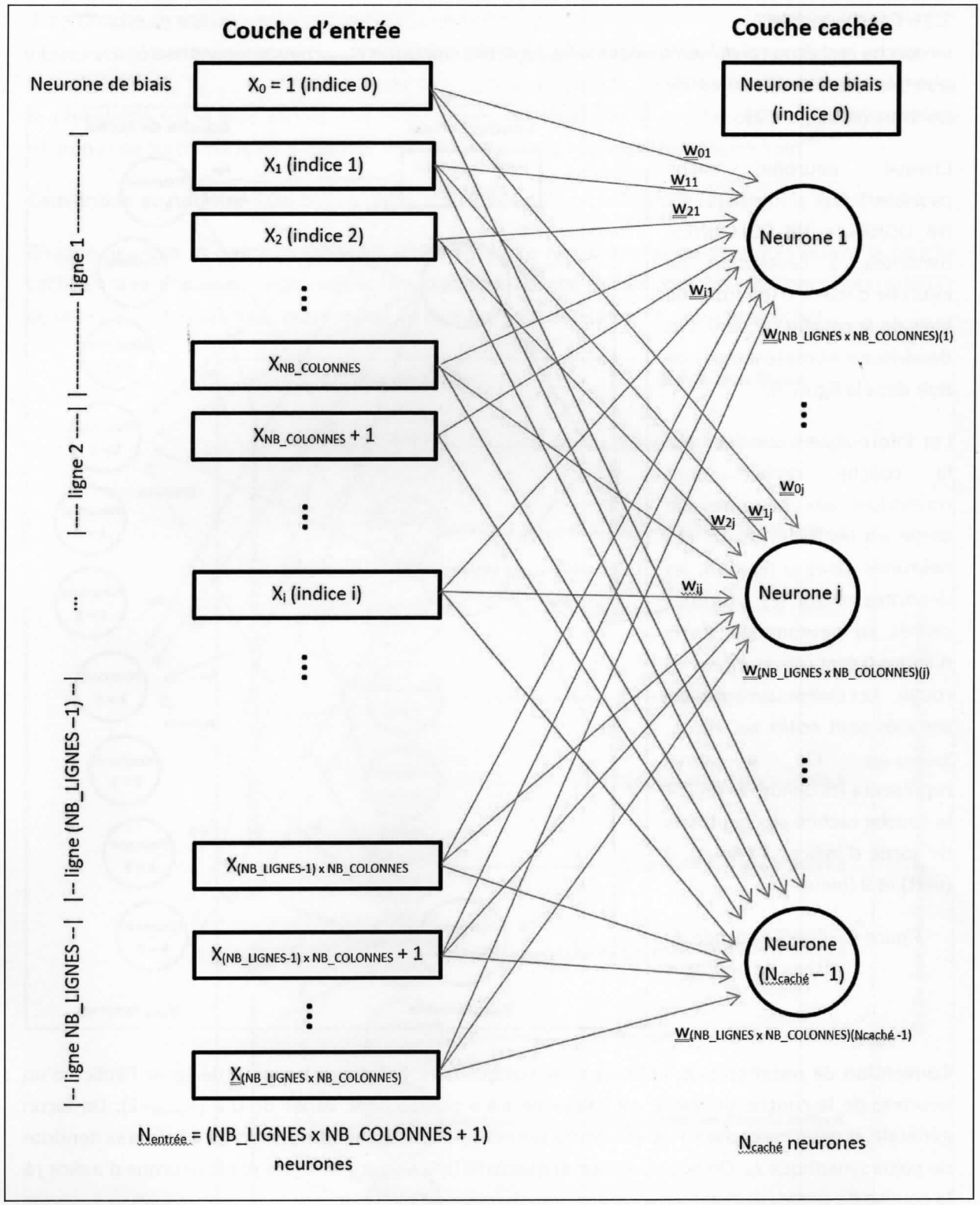

Figure 3 – Couche d'entrée du réseau neuronal

2.3 – Couche cachée

La couche cachée est représentée dans la figure 4. Elle comporte $N_{caché}$ neurones connectés à la couche d'entrée via les dendrites de poids w_{ij} (voir figure 2).

Chaque neurone caché possède $N_{entrée}$ = (1 + NB_LIGNES x NB_COLONNES) dendrites, à l'exception du neurone d'indice 0 (neurone de biais de la couche cachée). Ces dendrites sont représentées en noir dans la figure 4.

Les informations calculées par la couche cachée sont transmises aux neurones de sortie via les dendrites de ces neurones. Dans la figure 4, les dendrites reliant les neurones cachés au neurone de sortie d'indice 0 sont représentées en rouge. Les poids synaptiques associés sont notés z_{00}, z_{10}, ..., $z_{(Ncaché-1)(0)}$. On a aussi représenté les dendrites reliant la couche cachée aux neurones de sortie d'indices 3 (violet), 6 (vert) et 9 (bleu).

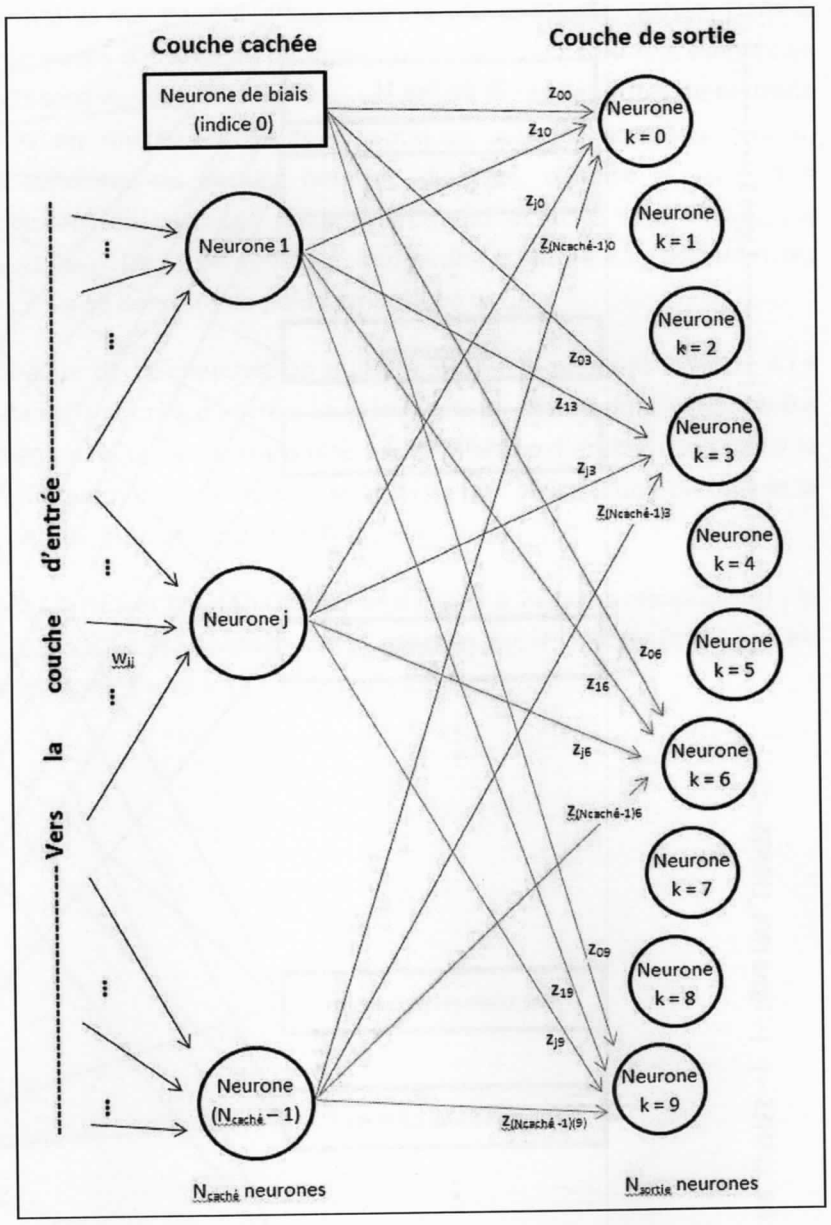

Figure 4 – Couche cachée du réseau de neurones

Convention de notation ! On utilisera systématiquement la lettre « k » pour désigner l'indice d'un neurone de la couche de sortie. La valeur de « k » pourra donc varier de 0 à (N_{sortie}–1). De façon générale, le neurone caché « j » transmettra ses informations au neurone de sortie « k » via sa dendrite de poids synaptique z_{jk}. On notera systématiquement U_j la valeur transmise par le neurone d'indice j à la couche de sortie.

Pour le jeu de données MNIST, le nombre de dendrites vaut 785 x $N_{caché}$ (sans *maxpooling*). Si la couche cachée comporte 300 neurones, ce nombre vaut 235500. Avec *maxpooling*, et si la couche cachée comporte 66 neurones, ce nombre vaut 170 x $N_{caché}$ = 11220.

2.4 – Couche de sortie

La couche de sortie comporte N_{sortie} (= 10) neurones qui produisent, pour le bitmap fourni en entrée, la probabilité $\hat{y}_0$, $\hat{y}_1$, ..., $\hat{y}_9$ d'être le chiffre 0, 1, ..., 9 (voir figure 5). Le chiffre final prédit est celui dont la probabilité est la plus élevée. Les informations calculées par la couche cachée parviennent aux neurones de sortie via leurs dendrites de poids synaptiques z_{jk}, représentées en noir.

Convention de notation ! On notera $\hat{y}_k$ la probabilité qu'a un bitmap de représenter le chiffre « k ».

Chaque neurone de sortie possède $N_{caché}$ dendrites. Le nombre total de dendrites reliant la couche cachée à la couche de sortie vaut donc 10 x $N_{caché}$. Si la couche cachée comporte 300 neurones (MNIST), ce nombre vaut 3000. Avec *maxpooling*, ce nombre vaut 660 (pour 66 neurones cachés).

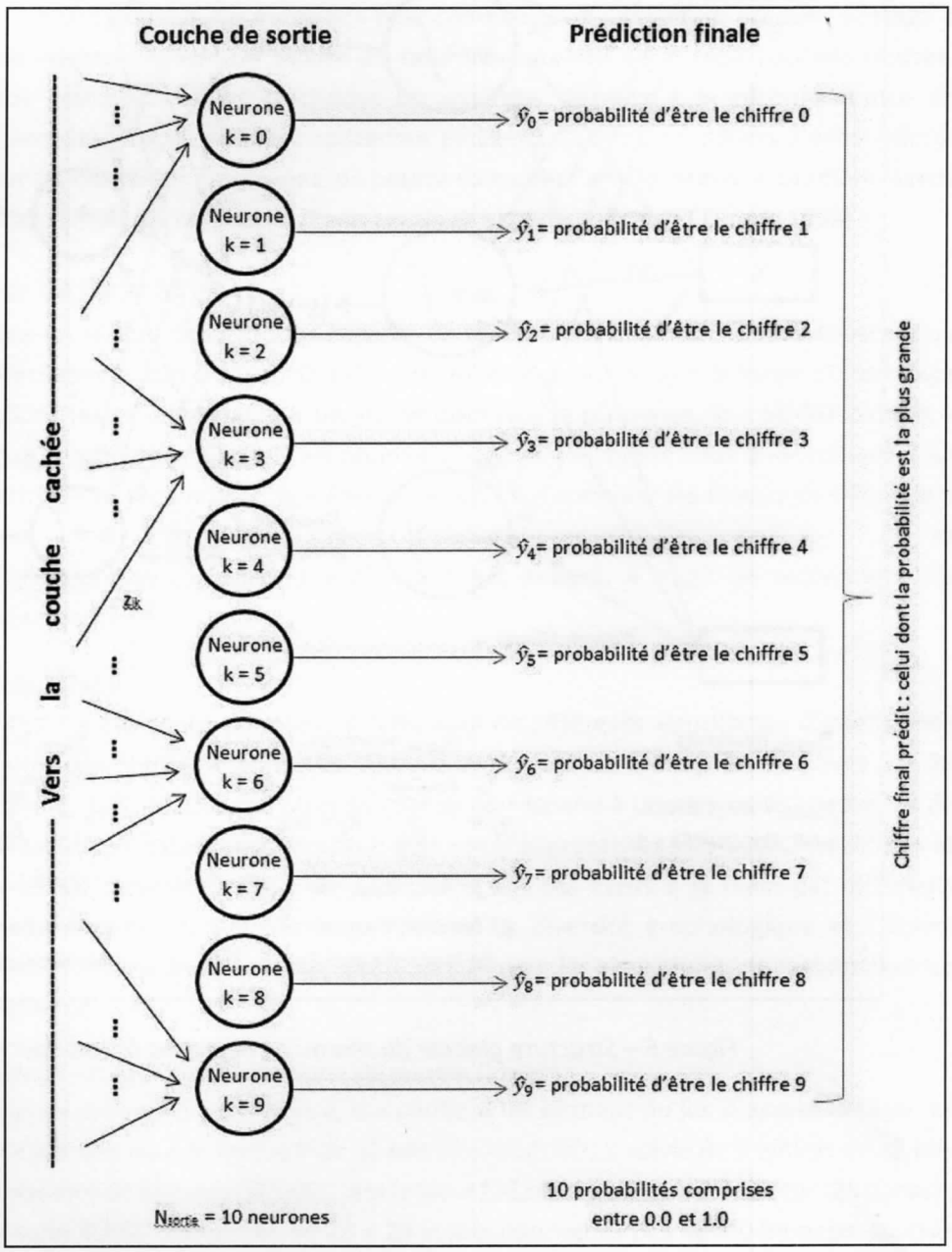

Figure 5 – Couche de sortie du réseau de neurones.

La structure globale du réseau de neurones est résumée dans la figure 6. Elle permet de décrire les algorithmes appliqués pour reconnaître les caractères du jeu de données MNIST.

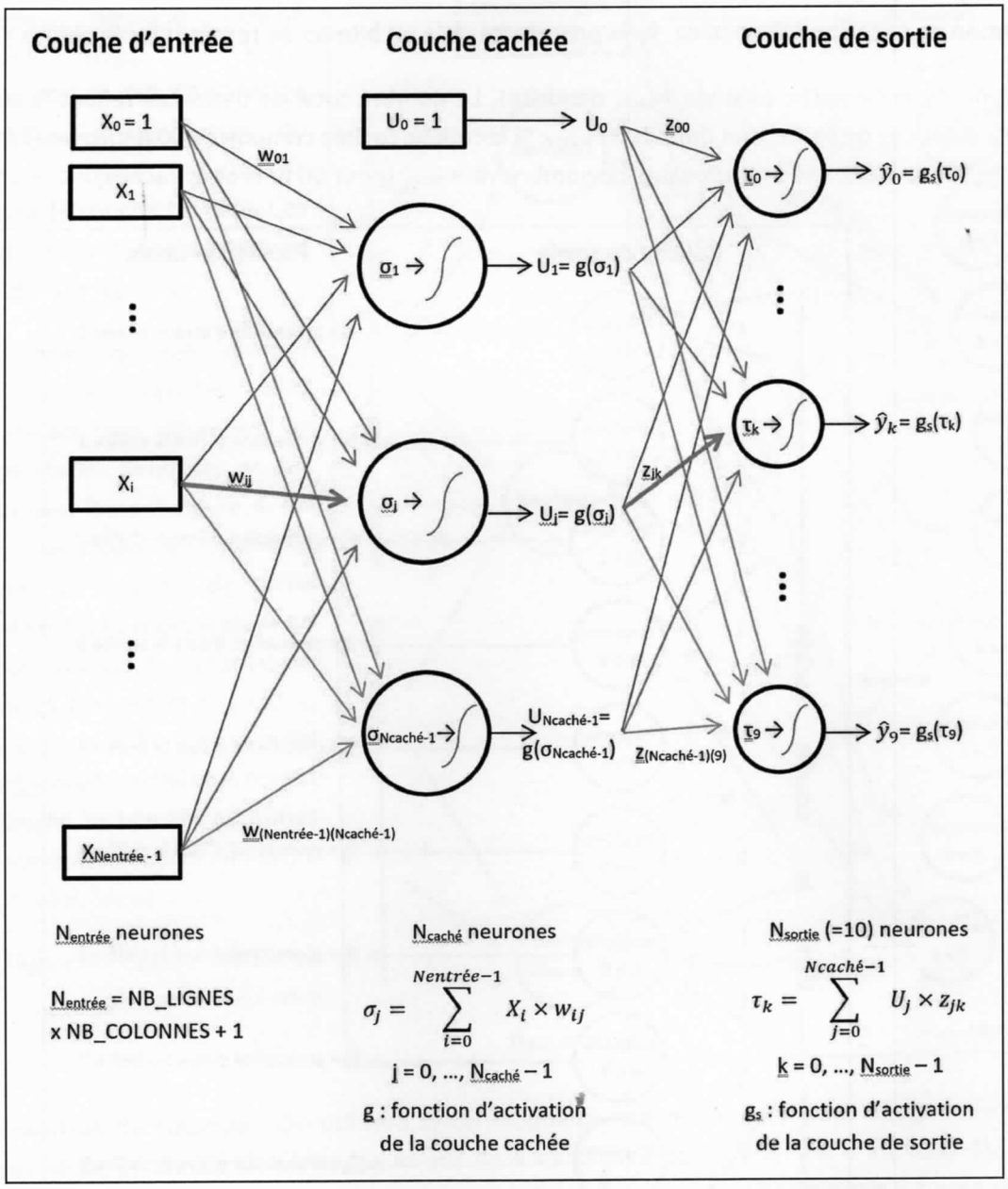

Figure 6 – Structure globale du réseau de neurones à trois couches.