Reconnaissance de chiffres manuscrits par réseaux de neurones artificiels

1 Reconnaissance de chiffres manuscrits

La reconnaissance automatique de chiffres manuscrits est un des problèmes ayant beaucoup inspiré de travaux en apprentissage automatique. Le site de Yann Le Cun (http ://yann.lecun.com) regorge d'une quantité d'information sur ce problème ainsi que de résultats obtenus par différents modèles d'apprentissage statistique sur la base de données MNIST (`http://yann.lecun.com/exdb/mnist/`).

Le but de ce projet de programmation C est de construire un réseau de neurones artificiels capable d'apprendre à reconnaître des chiffres manuscrits représentés par des images en niveaux de gris.

Pour cela, une base (compressée) de chiffres manuscrits est mise à disposition à l'url suivante :

`http://www.cmi.univ-mrs.fr/~liva/DONNEES/digits_train.csv.bz2`

Chaque ligne de ce fichier correspond à la suite de l'intensité (normalisée pour prendre une valeur entre -1 et 1) des pixels de l'image représentant le chiffre manuscrit et la dernière entrée de chaque ligne correspond au chiffre lui-même. Chaque image est de taille 16x16 et chaque ligne comporte donc 256+1 valeurs.

L'évaluation du système programmé se fera en le confrontant à une base de données de chiffres manuscrits indépendante de la base d'apprentissage fournie et la qualité du système sera mesurée par le taux de bonne classification, c'est-à-dire la proportion d'images affectées correctement aux chiffres qui leur correspondent.

Une description plus détaillée du fonctionnement des réseaux de neurones que l'on considère, les *perceptrons multi-couches*, et de leur apprentissage est fournie dans la section 3.

Quelques particularités des images. Les données à traiter correspondent à des images de chiffres manuscrits. Entre deux images représentant le même chiffre, il peut y avoir, en plus des variations de style d'écriture, des phénomènes de translation, de rotation, d'amincissement, etc. (par exemple, un même '7' peut être centré dans une image et être légèrement décalé sur la droite dans une autre). Il peut être intéressant d'essayer de tenir compte, par le moyen de pré-traitements appliqués aux données, de ce type de phénomènes. Par ailleurs, on peut également penser à utiliser des outils de traitement d'image permettant d'avoir une représentation insensible à certaines invariances standard (e.g., transformée de Fourier).

2 Modalités

Ce projet est à réaliser par groupes de 2 étudiants au maximum.

Le projet donnera lieu à un rapport de 10 pages au maximum dans lequel seront expliqués les différents choix de programmation effectués, les structures de données principales, les fonctions majeures, et les raisons qui ont motivé leur choix.

Des commentaires critiques sur l'apprentissage par réseaux de neurones, l'efficacité de la descente de gradient (stochastique), l'impact du nombre de neurones portés par la couche cachée seront

fournis. Quelques résultats rapportant les taux de bonne classification du réseau de neurones seront donnés dans ce rapport.

Le rapport doit être rendu le **vendredi 23 avril 2010** et une soutenance sur machine aura lieu le **lundi 26 avril 2010** (horaires à préciser).

La répartition des points est la suivante :
- **10 points** pour le rapport (aussi étonnant que cela puisse paraître, une majorité des points portera toujours sur votre rapport dans votre future vie d'étudiant) ;
- **5 points** pour le programme en lui-même, c'est-à-dire sa structuration, la syntaxe, l'indentation, l'algorithmique programmée, son ergonomie, etc.
- **5 points** pour la soutenance ;
- **2 points** si le programme permet d'obtenir un taux de bonne reconnaissance entre 95% et 97% ;
- **3 points** si le programme permet d'obtenir un taux de bonne reconnaissance supérieur à 97%.

Si le travail est parfaitement réalisé de bout en bout, le projet est donc noté sur 23 !

3 Perceptron multi-couches

3.1 Apprentissage automatique et classification supervisée

L'apprentissage automatique est un domaine de l'informatique qui s'intéresse à l'étude et la proposition de méthodes informatiques capables d'« apprendre », comme les êtres humains. Plusieurs domaines d'application peuvent bénéficier de l'apprentissage automatique : le reconnaissance automatique de chiffres manuscrits, la reconnaissance de visages, la détection d'émotions, les interfaces homme-machine, la recommandation de films, les jeux de plateau comme les échecs, les dames, Othello, et, plus récemment, le Go, etc.

Un des thèmes principaux de l'apprentissage automatique est la *classification supervisée*, où la tâche abordée est celle de pouvoir associer automatiquement une classe ou une catégorie à un objet, et c'est justement dans ce cadre de travail que se situe ce projet de reconnaissance automatique de chiffres manuscrits.

Plus précisément, le problème de la classification supervisée se décrit comme suit. Soit un échantillon fini $S = \{(X_i, Y_i)\}_{i=1}^{n}$ d'observations étiquetées[1], où $X_i \in \mathcal{X}$ est la description d'un objet (par exemple, le vecteur des niveaux de gris d'une image représentant un chiffre, et donc $\mathcal{X} = \mathbb{R}^d$) et $Y_i \in \mathcal{Y}$ l'étiquette de cet objet (dans le cas de la reconnaissance de chiffres manuscrits $\mathcal{Y} = \{0, 1, \ldots, 9\}$). L'objectif est de construire à partir de l'*ensemble d'apprentissage* S une fonction de $\mathcal{X}$ dans $\mathcal{Y}$ capable de prédire correctement les bonnes étiquettes Y à associer à des objets X *non présents* dans S : cet objectif traduit l'essence même de la *généralisation* (i.e. on veut être capable de prédire correctement sur de nouvelles données, et donc de généraliser des connaissances extraites à partir de S à de nouvelles observations).

Une approche intuitive et statistiquement justifiée si l'on fait de bonnes hypothèses (les détails ne sont pas donnés dans ce document mais les étudiants intéressés peuvent consulter des livres/documents sur ce qu'on appelle la théorie de l'apprentissage statistique) est de construire un modèle/une fonction qui est capable de prédire correctement les associations (X, Y) pour les données de S. Il s'agit du *principe de minimisation du risque empirique* (ERM) et il peut se formaliser comme la recherche d'une fonction $\hat{f}$ au sein d'une famille de fonctions $\mathcal{F}$ qui minimise le

[1] Pour être complet, on fait souvent l'hypothèse que cet échantillon est constitué de n variables aléatoires indépendamment et identiquement distribuées suivant une loi de probabilité D fixe mais inconnue.

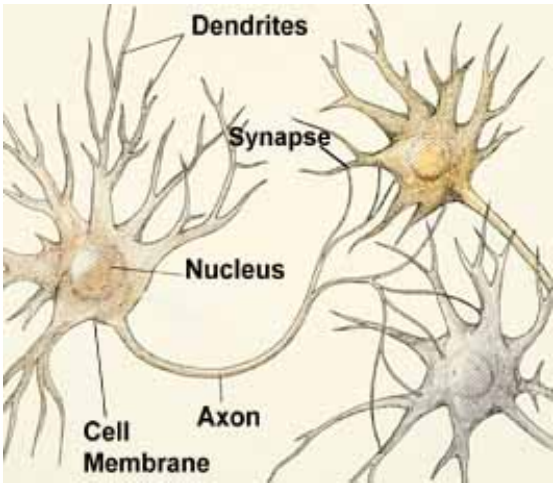

Fig. 1 – Réseau de neurones biologique, dont s'inspirent les réseaux de neurones artificiels.

risque empirique $\hat{R}(f, S)$ défini par :

$$\hat{R}(f, S) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[f(X_i) \neq Y_i],$$

où $\mathbb{I}[A]$ est la fonction indicatrice qui vaut 1 si A est vrai et 0 sinon. En somme, le principe ERM préconise de choisir $\hat{f}$ de la manière suivante :

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{R}(f, S).$$

C'est ce principe d'induction que nous allons mettre en œuvre avec les réseaux de neurones artificiels. Nous allons en fait nous limiter à une famille particulière de réseaux de neurones, les perceptrons multi-couches ; ces réseaux de neurones constituent ainsi notre famille $\mathcal{F}$.

3.2 Modèle

Les réseaux de neurones artificiels constituent des familles de fonctions construites sur des réseaux d'unités de calcul très simples, désignées par la suite par *neurones* ou *cellules*, dont on dit qu'elles s'activent ou s'inhibent. Elles transmettent leur degré d'activation ou d'inhibition aux cellules auxquelles elles sont connectées, ce qui entraîne de nouveaux calculs, définissant à leur tour d'autres degrés d'activation/inhibition, etc. Comme le nom l'indique, ces fonctions sont fortement inspirées des réseaux de neurones que l'on peut trouver chez les êtres vivants (voir Figure 1). Cependant, la complexité du fonctionnement des réseaux de neurones biologiques est telle que les réseaux de neurones artificiels n'en sont que des versions extrêmement simplifiées.

Le perceptron multi-couches (PMC) est un type particulier de réseaux de neurones artificiels. Sa spécificité est son fonctionnement dit *en passe avant*, parce que l'information qui y circule ne le fait que dans un sens ; il n'y a donc pas de boucles au niveau des connexions neuronales. Le schéma de la Figure 2 représente un perceptron multi-couches avec 2 neurones sur la couche d'entrée, une couche cachée, qui peut contenir autant de neurones que l'on veut, et une couche de sortie contenant également 2 neurones, et dont l'interprétation permet de réaliser la tâche abordée. Si l'on se pose un problème de classification binaire (des objets de deux classes sont alors à distinguer les uns des autres) alors le premier neurone de sortie correspond à la première classe et le second à le seconde classe : si l'activation du réseau de neurones en réponse à la présentation d'un objet conduit à une activation plus forte du premier neurone que du second neurone alors l'objet est associé à la première classe et inversement.

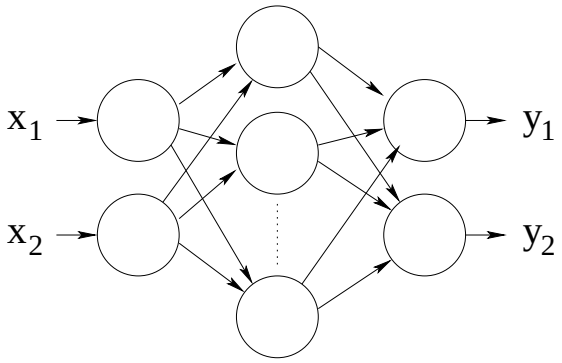

Fig. 2 – Perceptron multi-couches : l'information circule de la couche d'entrée (à gauche), à la couche de sortie (à droite), en passant par la couche cachée.

Les sections suivantes présentent en détail les paramètres qui entrent en jeu dans la définition d'un PMC, la manière dont l'activation du réseau se fait et, évidemment, comment l'apprentissage d'un PMC est réalisé.

3.3 Paramètres et activation d'un PMC

Un PMC, comme tout autre réseau de neurones, est paramétré non seulement par le nombre de neurones qui le constituent (sur les couches d'entrée, cachée et de sortie), le graphe de connectivité, mais également par les poids des connexions qui lient les neurones entre eux. Suivant l'inspiration biologique des PMC, deux neurones qui sont connectés avec un arc muni d'un poids positif s'activent simultanément et inversement. Des neurones réalisant des tâches similaires sont donc censés s'activer ensemble.

L'objectif premier de l'apprentissage par PMC est de régler automatiquement les poids des connexions à partir d'un ensemble d'observations étiquetées.

Un autre paramètre qui entre en jeu dans la définitiion d'un PMC est la fonction d'activation ou fonction de transfert associée à chaque neurone : cette fonction constitue précisément le calcul élémentaire réalisé par chaque neurone. Dans les versions originelles des perceptrons, on utilisait une fonction d'activation de type « seuil », c'est-à-dire une fonction qui prenait uniquement deux valeurs, 0 ou 1, selon que la force des signaux arrivant au neurone dépassait un certain seuil ou non. Malheureusement, la fonction $x \mapsto \mathbb{I}[x \leq \theta]$ pour $\theta \in \mathbb{R}$ n'est pas dérivable et induit des problèmes numériques pour l'apprentissage des poids du PMC. Afin de pallier ce problème de non-dérivabilité, plusieurs fonctions, dites sigmoïdes (car prenant la forme de S), dont la forme s'approche de la fonction seuil peuvent être utilisées. Dans ce projet, nous utilisons la fonction σ suivante :

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

La figure 3 représente en détail un PMC avec les poids w_{ji} (w_{ij} désigne le poids de la connexion du neurone i vers le neurone j), le mode de calcul de l'activation d'un neurone, et fournit une représentation de la fonction σ

De manière plus détaillée :
– les cellules biais sont des neurones qui ne reçoivent pas de signaux mais dont l'activation est toujours égale à 1 ;

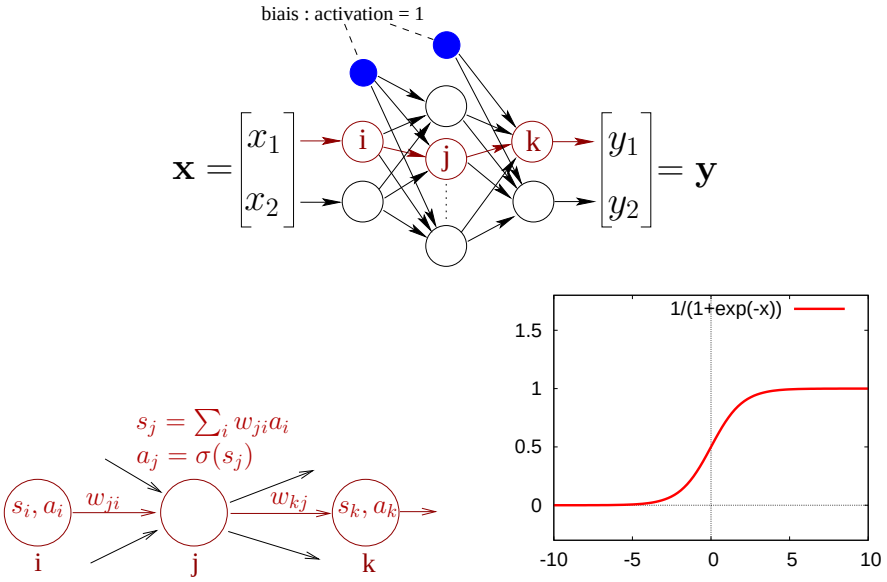

Fig. 3 – Détails des paramètres d'un PMC, mode de calcul de l'activation d'un neurone et fonction sigmoïde σ.

- les cellules de la couche d'entrée transmettent sans les modifier les signaux qu'elle reçoivent ; en d'autre terme, la fonction d'activation ou de transfert utilisée par ces neurones est la fonction identité $h(x) = x$;
- les cellules de la couche cachée et de la couche de sortie calculent leur activation de la même manière ; par exemple, pour le neurone j :
  1. l'intensité du signal reçu par j est calculée par :
  $$s_j = \sum_{i \in \text{entrée}} w_{ji} a_i;$$
  2. son activation est alors calculée par :
  $$a_j = \sigma(s_j).$$

3.4 Apprentissage d'un PMC

Comme signalé précédemment, l'objectif premier de l'apprentissage est celui de déterminer automatiquement les poids des connexions entre les neurones. Ici, nous ne fournissons pas les détails des calculs qui mènent aux équations de mise à jour proposées mais signalons simplement que l'algorithme est un algorithme de descente de gradient adapté à la structure particulière d'un PMC.

Avant de fournir le détail de ces calculs, on relève à nouveau que la classification d'un objet en fonction de l'activation d'un PMC se fait à la lecture de l'activation des neurones de la couche de sortie. Une convention usuelle est d'attribuer à un objet la classe k si c'est le neurone k de la couche de sortie qui a l'activation la plus élevée. Ainsi, on souhaite qu'à chaque objet X à classer de classe k, le vecteur de sortie Y calculé par le PMC vérifie cette propriété. C'est ce critère qui permet de définir l'algorithme d'apprentissage par rétro-propagation du gradient.

Nous adoptons la convention suivante. Pour un problème de classification à K classes, le vecteur à prédire pour un exemple X de classe k est le vecteur $Y = [0.1, \ldots, 0.1, 0.9, 0.1, \ldots, 0.1]$ de taille K où le 0.9 est en k-ième position. L'algorithme d'apprentissage est décrit dans la table 1.

Tab. 1 – Algorithme de rétro-propagation du gradient stochastique

– Répéter jusqu'à convergence
  – pour chaque exemple (X_p, Y_p) faire
    1. propager l'information (calculer l'activation du PMC)
    2. pour chaque neurone de sortie k calculer

$$\delta_k \leftarrow a_k(1 - a_k)(y_{pk} - a_k)$$

    3. pour chaque neurone caché j calculer

$$\delta_j \leftarrow a_j(1 - a_j) \sum_{k \in \text{sortie}} \delta_k w_{kj}$$

    4. calculer les pas Δw^t_{ji} et Δw^t_{kj} par $(\eta > 0)$

$$\Delta w^t_{ji} = +\eta \delta_j a_i, \qquad \text{(c. entrée à c. cachée)}$$
$$\Delta w^t_{kj} = +\eta \delta_k a_j \qquad \text{(c. cachée à c. sortie)}$$

    5. mettre à jour les w_{ji} et les w_{kj} par

$$w_{ji} \leftarrow w_{ji} + \Delta w^t_{ji} \qquad \text{(c. entrée à c. cachée)}$$
$$w_{kj} \leftarrow w_{kj} + \Delta w^t_{kj} \qquad \text{(c. cachée à c. sortie)}$$

L'algorithme s'appelle algorithme de rétro-propagation du gradient parce que l'information utilisée – le gradient – pour mettre les poids à jour est calculée depuis la sortie du PMC jusqu'à l'entrée, et donc en sens inverse de la propagation normale de l'information.

4 Travail à faire

Le but du projet est de programmer un PMC, son apprentissage et de l'exécuter sur un ensemble d'images de chiffres manuscrits qui n'ont pas été vus lors de l'apprentissage.

Le programme réalisé devra respecter les contraintes suivantes.

Modularité. Le programme doit être décomposé en plusieurs modules, définis par les fonctionnalités qu'elle proposent. En particulier, des modules évidents sont celui permettant les calculs concernant le PMC, celui permettant le chargement depuis un fichier un PMC ainsi que les vecteurs de chiffres manuscrits, un autre qui permet le calcul des statistiques de bonne classification du PMC appris. Évidemment, la compilation du projet doit se faire grâce à l'utilitaire make et un fichier makefile approprié. Tout projet qui n'est pas muni d'un fichier makefile se verra attribuer la note de 0.

Persistance. Le programme doit permettre d'enregistrer/charger un PMC dans/depuis un fichier. Le jour de la soutenance, il ne sera pas possible d'exécuter l'apprentissage du PMC, faute de temps ; les étudiants devront charger en mémoire un PMC qui aura été appris auparavant.

Ergonomie. Le programme doit être simple d'utilisation. Il doit permettre à l'utilisateur de choisir entre exécuter un apprentissage, utiliser un PMC déjà appris, enregistrer un PMC qui vient d'être appris, régler les paramètres de l'apprentissage (le pas d'apprentissage $\eta > 0$, notamment), définir une structure de PMC, etc.