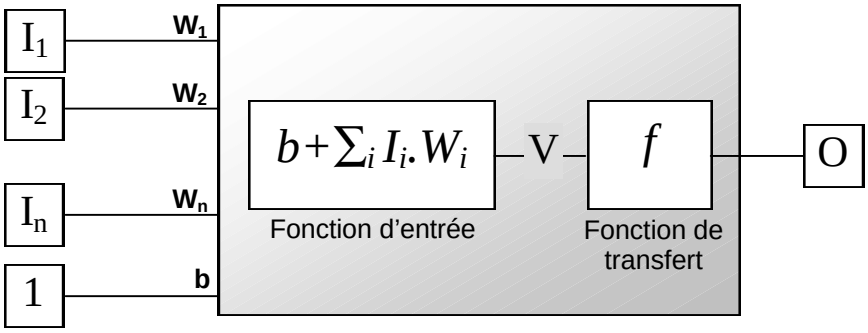
PLAN DE L'EXPOSÉ

PRÉVISION DE TEMPÉRATURE PAR RÉSEAUX NEURONAUX

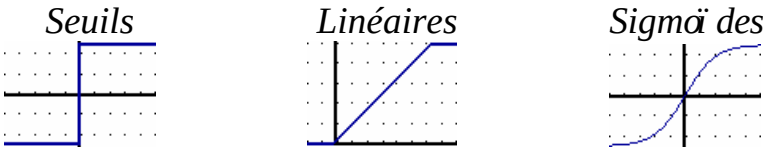
Les réseaux de neurones artificiels

- Introduits par Mc Culloch et Pitts en 1943
- Inspirés du modèle biologique :
  - → **Parallélisme** massif
  - → Notion de **poids synaptiques**
  - → Possibilité d'**apprentissage**

LE NEURONE ARTIFICIEL



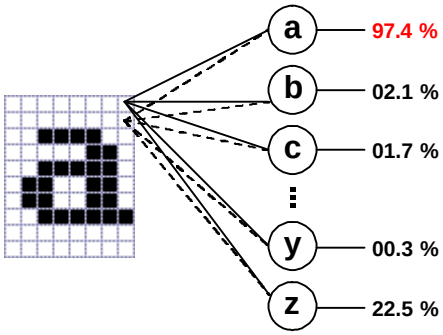
- *n* **entrées** I_k affectées d'un **poids** *synaptique* W_k
- **Fonction d'entrée** $\rightarrow$ **potentiel** V (état interne)
- **Fonction de transfert** $\rightarrow$ **sortie** O



- **Réseau** : ensemble de neurones interconnectés

UNE PREMIÈRE APPLICATION

– Reconnaissance de caractères –

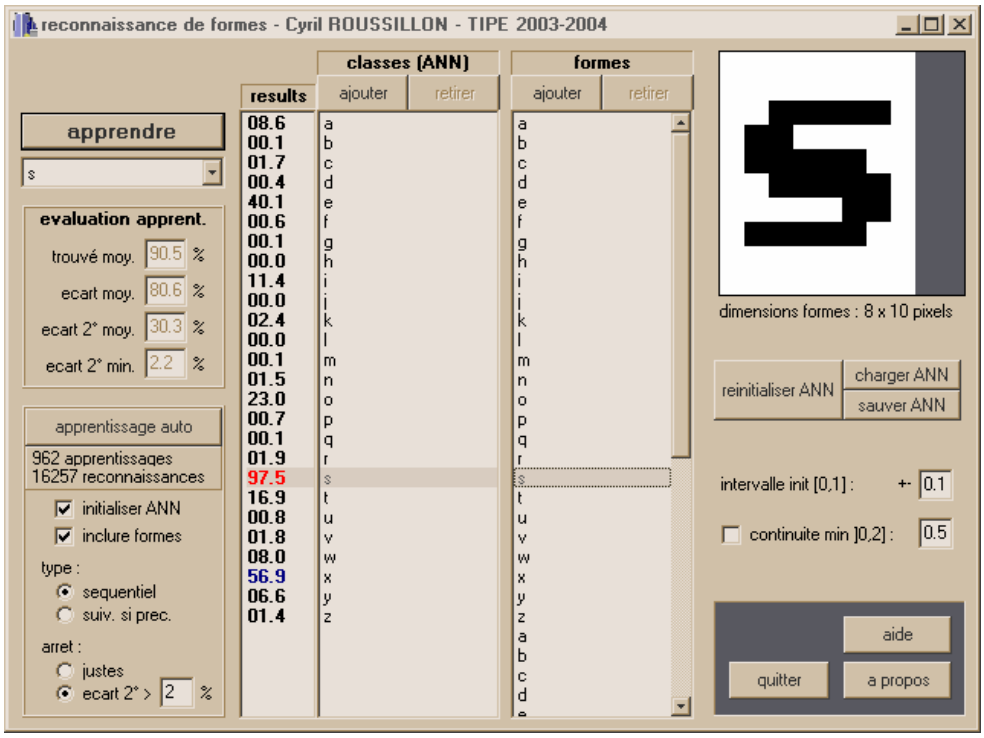


- **Une couche** de neurones, chaque neurone relié à **tous les pixels**
- Chaque neurone est spécialisé dans la reconnaissance d'un caractère
- Fonction de transfert : **sigmoïde** → réel entre 0 et 1
- Sa sortie indique la **probabilité** que la forme présentée soit la sienne

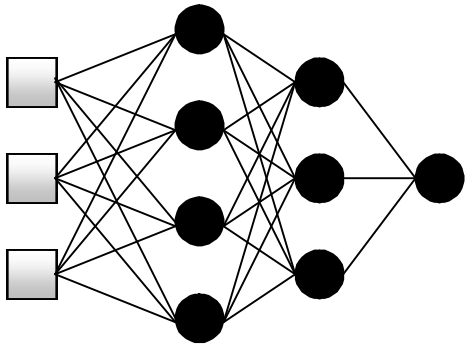
APPRENTISSAGE : SUPERVISÉ

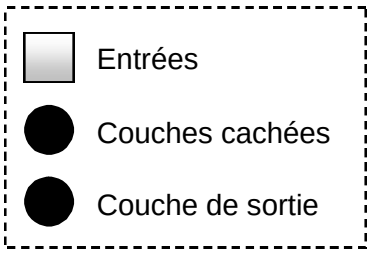
- **Correction** si le réseau s'est trompé
  → augmenter les poids des pixels activés pour la forme théorique
  → les diminuer pour la forme trouvée par le réseau (fausse)
  → proportionnellement à la différence entre sorties théorique et trouvée

- **Renforcement** si le réseau a trouvé la bonne forme
  → augmenter le poids des pixels activés
  → d'autant plus que l'écart avec la forme en deuxième position est faible



LE RÉSEAU MULTICOUCHES





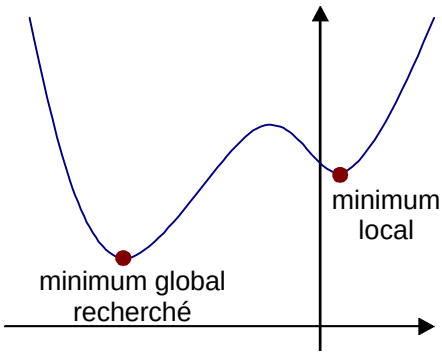
- Neurones organisés en **couches**
- Chaque neurone → reçoit sorties des neurones de la couche précédente
  → envoie sa sortie aux neurones de la couche suivante

Comment réaliser alors un apprentissage supervisé pour les neurones cachés ?

RÉTROPROPAGATION DU GRADIENT

Application de la méthode universelle d'optimisation de **descente de gradient**

→ Descendre la pente d'une fonction d'erreur pour en trouver un minimum



Reste à exprimer cette pente en fonction de paramètres connus du réseau :

$$\Delta W_{ijk}(t+1) = ç.S_{i\text{-}1,k}.ä_{ij} \quad \text{avec} \begin{cases} d_{nj} = f_n'(V_{nj}).(T_j - O_j) \\ d_{ij} = f_i'(V_{ij}).\sum_{l=1}^{m_{i+1}} d_{i+1,l}.W_{i+1,l,j} \quad \text{si } i < n \end{cases}$$

QUALITÉS D'APPROXIMATEUR

Le **but** de tout réseau de neurones est d'approximer une fonction

Les réseaux de neurones sont des **approximateurs** :

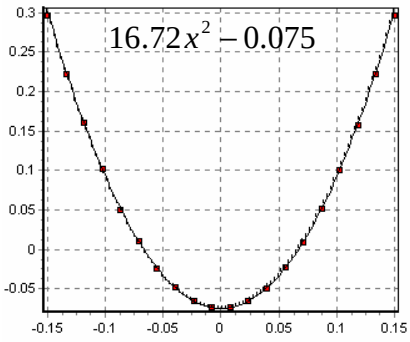
→ **universels**

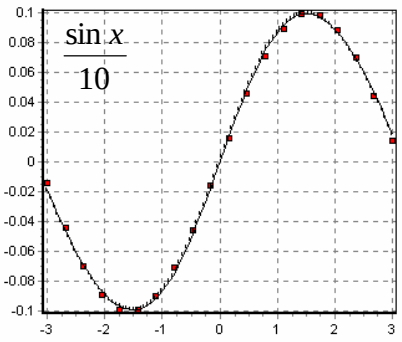
→ plus **parcimonieux** que les polynômes

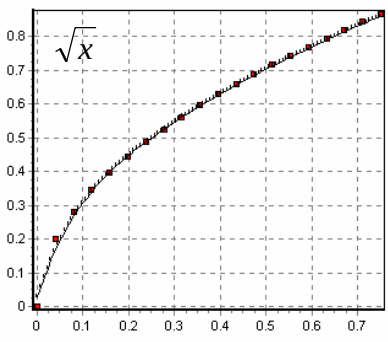
Les réseaux de neurones sont donc économes en exemples

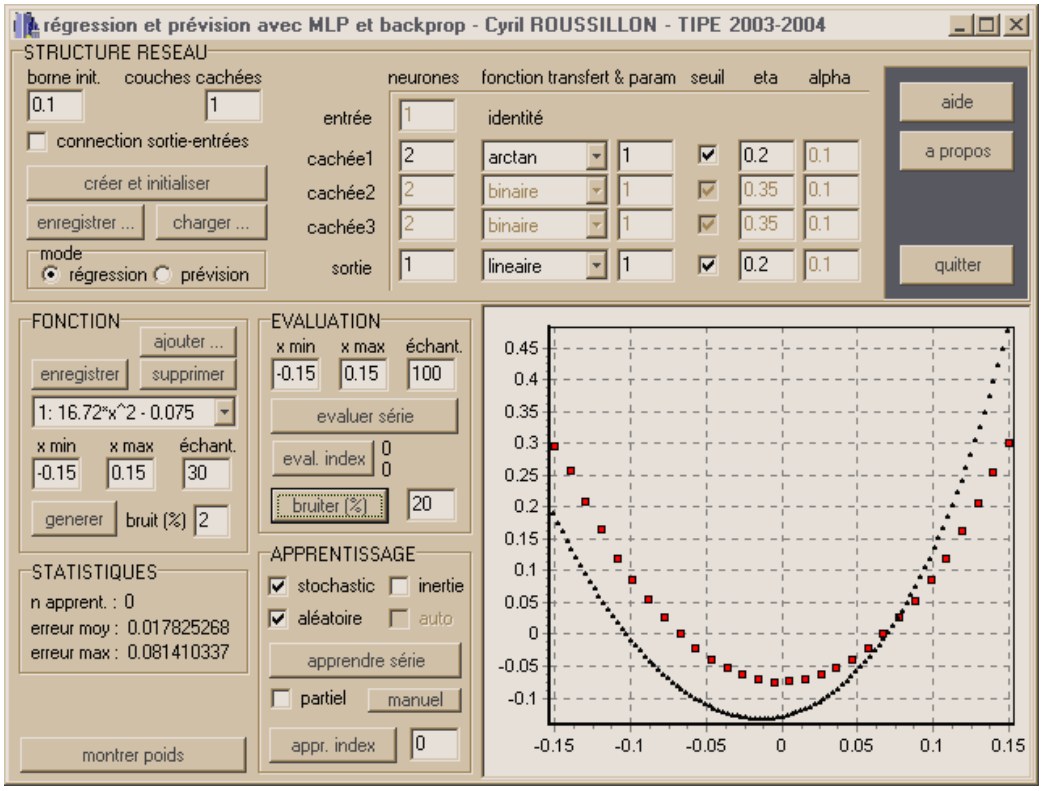
RÉGRESSION

- Intérêt pratique limité, mais intérêt **pédagogique** important

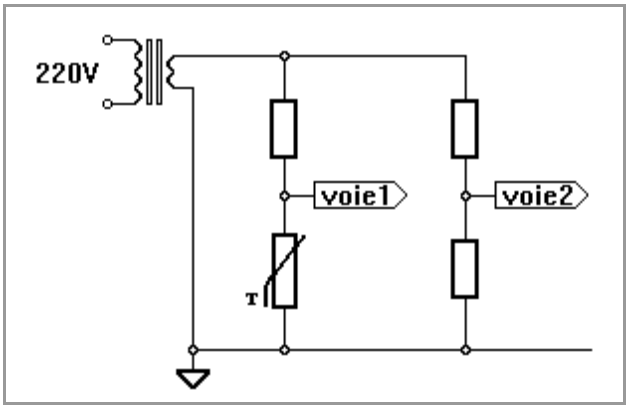


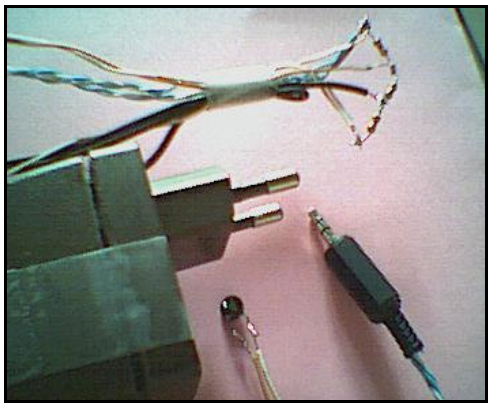




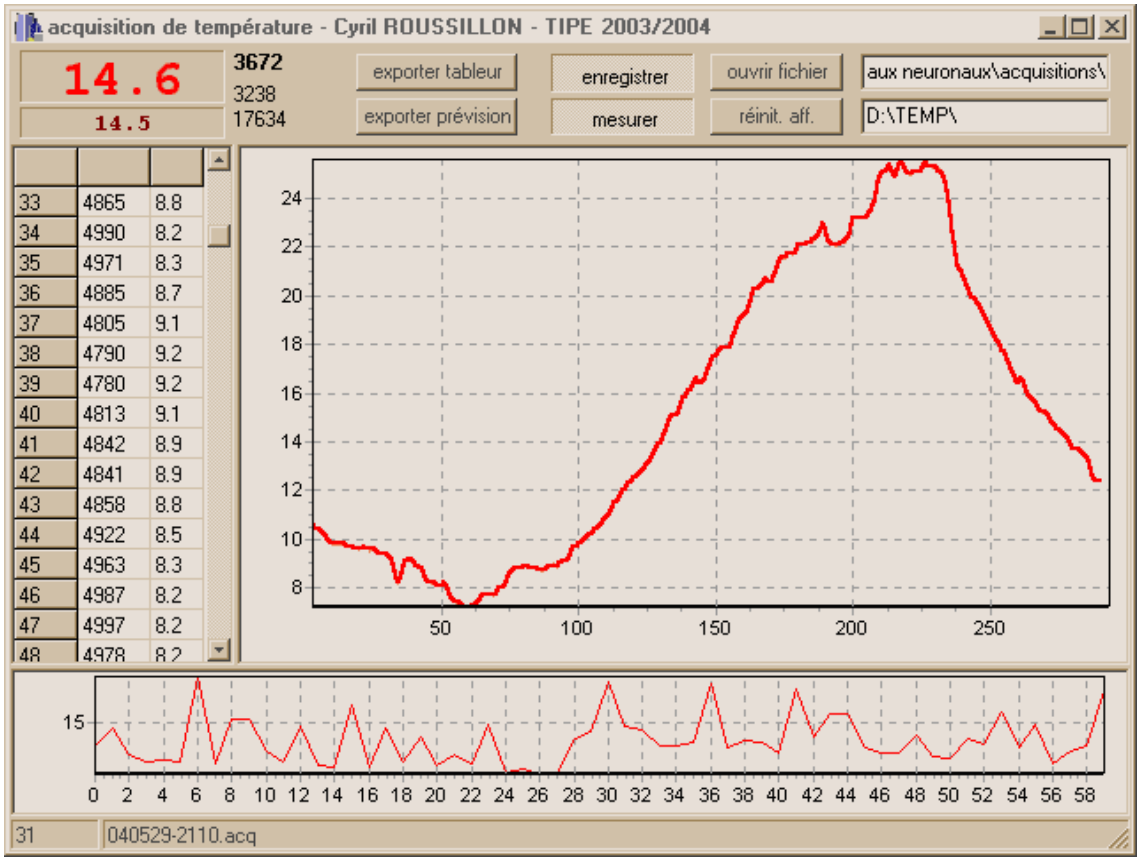


ACQUISITION DE TEMPÉRATURE

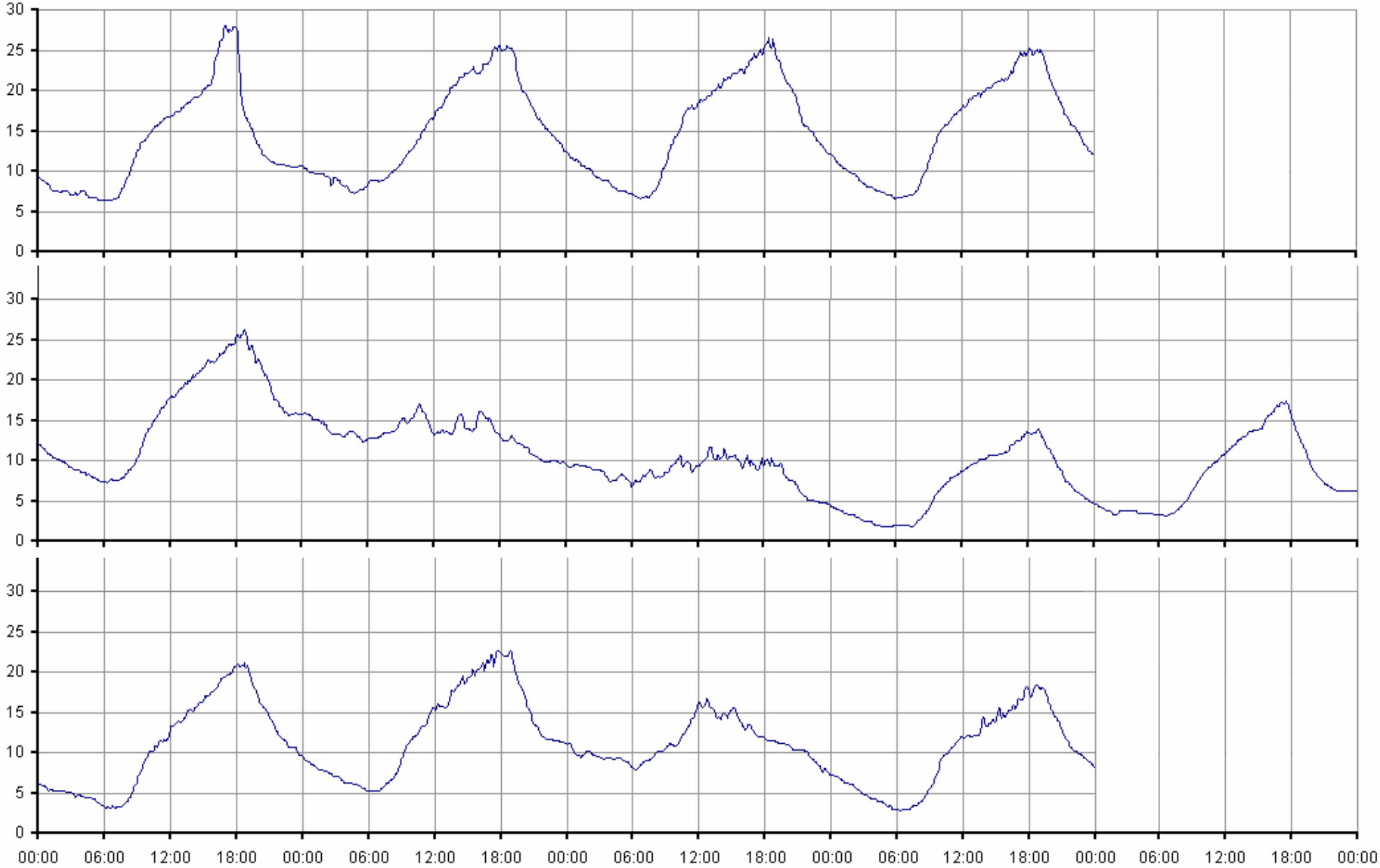




- Capteur : **thermistance** CTN
- CAN 16 bits : **carte son** (entrée LINE)
- Sinusoïde 50 Hz pour neutraliser le condensateur d'entrée
- Enregistrement dans un fichier .WAV dans la mémoire RAM
- Lecture du fichier → amplitude

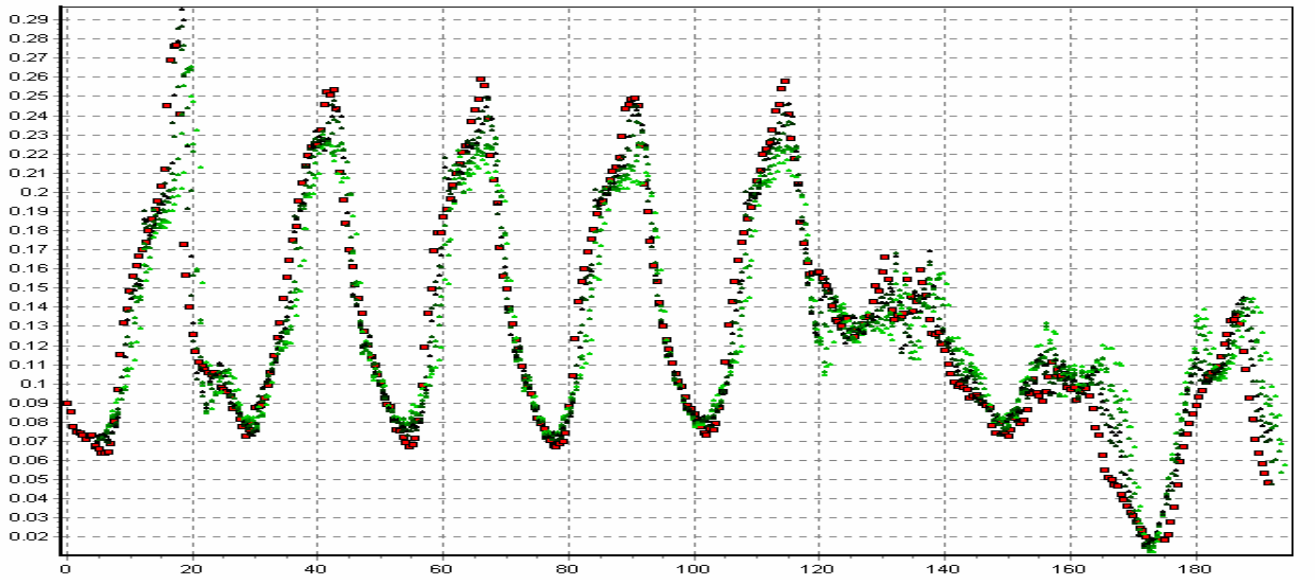


Température extérieure de 13 jours consécutifs



PRÉVISION DE TEMPÉRATURE

- *n* entrées contenant les valeurs précédentes de la température
- Prévision à un pas de temps, que l'on peut prolonger



- Amélioration possible en fournissant d'autres données au réseau

CONCLUSION

- Réseaux neuronaux prometteurs pour certaines applications
- Mais ne pas négliger méthodes classiques

Texte complet de l'exposé

Introduction. Les réseaux neuronaux ont été introduits par deux biophysiciens américains, McCulloch et Pitts en 43. Ils reprennent quelques grands principes du modèle biologique, à savoir qu'ils sont constitués d'entités autonomes réalisant une fonction très simple, mais très fortement interconnectées, ce qui rend le traitement du signal massivement parallèle. Ils reprennent également la notion de coefficients synaptiques, qui déterminent la force de l'interaction entre les neurones, ainsi que la possibilité de modifier ces coefficients afin de faire réaliser au réseau la fonction voulue.

1.1. Donc on peut représenter un neurone artificiel de cette façon. Il possède n entrées, à chacune desquelles est affecté un poids. Le neurone va commencer par en faire la somme pondérée, c'est sa fonction d'entrée, ce qui va donner son état interne. Puis il va appliquer sa fonction de transfert, qui va produire sa sortie, qu'il va envoyer aux autres neurones. Cette fonction peut prendre différentes formes, ce peut être une fonction à seuil, linéaire ou encore sigmoïdale, c'est-à-dire une approximation dérivable d'un seuil. Un réseau de neurones est alors simplement un ensemble de neurones interconnectés.

1.2. Pour exemple, j'ai commencé par réaliser un programme de reconnaissance de formes. Le réseau est simplement constitué d'une couche de neurones ; chaque neurone est relié à tous les pixels, et est responsable de la reconnaissance d'une forme. La fonction utilisée est une sigmoïde, qui va donner un réel entre 0 et 1, et on va donc entraîner le réseau pour que cette sortie représente la probabilité que la forme présentée au réseau soit celle dont est responsable le neurone.
L'apprentissage est de type supervisé, c'est-à-dire que l'on connaît la solution, ce qui permet de calculer l'erreur que commet le réseau afin de corriger les poids. Si le réseau s'est trompé, on va effectivement procéder à une correction, en augmentant le poids des pixels activés pour la forme réelle, et en les diminuant pour ceux de la forme donnée par le réseau, le tout proportionnellement à l'erreur commise. Et si le réseau a trouvé la bonne forme, on va tout de même procéder à un renforcement de l'apprentissage, en augmentant le poids des pixels activés, mais cette fois-ci de manière inversement proportionnelle avec la forme arrivant en seconde position, puisque cet écart représente la fiabilité de la reconnaissance.
J'ai réalisé ce programme en C++, comme tous les autres qui vont suivre, d'une part parce que c'est un langage que je connaissais bien, et d'autres part parce que cela m'a permis d'utiliser le logiciel BORLAND C++Builder, qui permet de réaliser une interface graphique très pratique, comme vous pouvez le constater.
Ce réseau donne des résultats satisfaisants, puisqu'il parvient à distinguer les 26 lettres de l'alphabet, même parmi plusieurs polices différentes.

1.3. Je me suis ensuite intéressé aux réseaux multicouches, qui sont de loin les plus utilisés. Comme leur nom le laisse présumer, les neurones sont organisés en couches : chaque neurone reçoit les sorties de tous les neurones de la couche précédente, et envoie sa sortie à tous les neurones de la couche suivante.
Mais alors, si on prend un neurone caché, on ne connaît pas sa sortie théorique (on sait juste ce que l'on veut en sortie du réseau). On ne peut donc pas calculer l'erreur de ce neurone, et on peut ainsi se demander comment réaliser un apprentissage supervisé pour ce neurone.

2.1. En fait la solution a été trouvée seulement dans les années 80, dans la méthode dite de rétropropagation du gradient. C'est en fait une simple application de la méthode d'optimisation de descente de gradient, qui consiste à localiser un minimum d'une fonction d'erreur en suivant son gradient. On peut s'imaginer la méthode par une bille que l'on poserait sur la courbe d'erreur, et qui descendrait sa pente pour s'arrêter dans un minimum. Alors bien sûr on peut rester bloquer dans un minimum local, alors que l'on recherche un minimum global, c'est pourquoi diverses améliorations ont été apportées, par exemple dans l'image de la bille on peut prendre en compte son inertie, son élan lui permettant de passer sur de petits minima locaux.
Il faut ensuite pouvoir calculer cette pente, c'est-à-dire l'exprimer en fonction de paramètres connus (comme les poids, les potentiels, les fonctions de transfert). En fait ça se fait très bien, la démonstration se trouve dans le dossier, elle tient en une page, et on trouve que la modification préconisée des poids est proportionnelle à l'entrée (sortie de la couche précédente) et à un coefficient, qui pour la couche de sortie vaut le produit de la dérivée de la fonction de transfert par la différence entre sortie théorique et sortie du réseau, et pour les autres couches est calculé en fonction des coefficients de la couche suivante, c'est-à-dire que l'on rétropropage l'erreur.

2.2. Ensuite je me suis intéressé aux qualités d'approximateur des réseaux de neurones, puisqu'en fait le but de tout réseau de neurones en apprentissage supervisé est d'approximer une fonction, vectorielle (on veut associer une certaine sortie à des entrées), et les réseaux de neurones présentent l'avantage d'être à la fois des approximateurs universels (comme les polynômes), mais aussi des approximateurs plus parcimonieux que les polynômes, c'est-à-dire qu'à précision donnée, le nombre de paramètres ajustables croît linéairement avec le nombre de variables, contre exponentiellement pour les polynômes. Ceci est dû en fait à leur non-linéarité par rapport aux paramètres.
Et l'on imagine aisément que pour ajuster un certain nombre de paramètres, il faut un nombre d'exemples grand devant le nombre de paramètres, donc les réseaux neuronaux se révèlent économes en exemples, qui peuvent être longs ou coûteux à récolter.

3.1. J'ai donc réalisé un réseau multicouches utilisant la méthode de rétropropagation du gradient, et j'ai commencé pour le tester à l'appliquer à la régression. En effet cela présente un intérêt pratique limité, puisque l'avantage que j'ai évoqué tout à l'heure d'une parcimonie accrue n'est valable que pour un grand nombre de variables : pour 1 ou 2 variables il n'y a pas de différence entre une évolution linéaire ou exponentielle. En revanche cela représente un gros intérêt pédagogique pour tester le réseau, puisque l'on voit immédiatement la vitesse et la qualité de l'apprentissage : la courbe doit se rapprocher des points.
Et donc au début l'apprentissage ne semblait pas fonctionner, mais le réseau s'est en fait simplement révélé très sensible aux coefficients d'apprentissage (le êta dans l'expression de tout à l'heure), et finalement le réseau donne de bons résultats puisqu'il parvient à approcher différentes fonctions avec une bonne application.

3.2. Pour appliquer mon réseau à la prévision de température, j'ai commencé par réaliser un capteur de température. Je suis allé au plus simple : le capteur est constitué d'une simple thermistance que l'on insère dans un pont diviseur de tension, et l'on acquiert le signal grâce à la carte son de l'ordinateur. On envoie une sinusoïde issue du secteur afin de neutraliser le condensateur d'entrée de la carte son, et on enregistre le son dans un fichier (un composant de C++Builder le fait tout seul), puis on l'ouvre (grâce à la structure du format WAV que j'ai trouvée sur Internet) pour l'interpréter.
J'ai donc écrit un logiciel qui automatise tout, et cela m'a permis d'obtenir la température extérieure de plusieurs jours, ce qui va constituer la base d'exemples d'apprentissage et de test, pour entraîner et tester mon réseau.

3.3. Le réseau utilisé pour la prévision de température est le même que pour la régression, la différence c'est qu'il y a cette fois n entrées (d'où l'intérêt des réseaux neuronaux avec beaucoup d'entrées). Ces entrées contiennent les précédentes valeurs de la température, et le réseau doit prédire la valeur suivante. C'est donc une prévision à un pas de temps, mais que l'on peut prolonger en réinjectant en entrée les valeurs déjà prédites.
Comme on peut le voir ici les résultats sont satisfaisants, puisqu'à une échéance de 3 heures, l'erreur dans le pire cas est de 3 ou 4 degrés Celsius, mais la plupart du temps l'erreur reste inférieure à 1 ou 2 degrés Celsius. On peut encore améliorer le résultat en donnant au réseau des données telles que l'heure, l'ensoleillement ou l'hygrométrie, qui ont une influence sur la température et donc peuvent aider le réseau.

Conclusion. Les réseaux neuronaux sont donc prometteurs pour certaines applications : la reconnaissance de formes est utilisée pour détecter les nappes de pétrole dans l'océan ou pour détecter les défauts dans les matériaux, et la prévision de température peut être utilisée dans une régulation de chauffage, dans le but de réaliser des économies d'énergie, ou dans la prévision des pics d'ozone.
Mais ce n'est pas pour autant qu'il faut négliger les méthodes plus classiques qui ont fait leurs preuves, car les réseaux neuronaux ont le défaut d'être une boîte noire, c'est-à-dire qu'une fois que le réseau a appris à résoudre le problème, la connaissance se trouve dans ses coefficients synaptiques, mais on a beaucoup de difficultés à interpréter ces coefficients pour déterminer sur quoi se base le réseau, et la résolution du problème par le réseau de neurones ne nous apporte à nous aucune connaissance sur le problème traité.