

Méthodes Formelles

TP apprentissage neuronal

Valentin Guerlesquin
Shulong Ling

1^{er} février 2007

Table des matières

1	Objectif du TP	1
2	Apprentissage d'une fonction	2
2.1	Données préliminaires	2
2.2	Construction du réseau de neurones	2
3	Apprentissage d'une commande	2
3.1	Une première approche	2
3.1.1	Premiers enregistrements, premier réseau de neurones	2
3.1.2	Vérification de l'apprentissage	3
3.2	Retour d'expérience, réflexion	3
3.2.1	Faites ce que je dis, pas ce que je fais	3
3.3	L'exemple à suivre	3
3.3.1	Procédure d'élaboration des échantillons d'apprentissage	4
3.3.2	Réseau de neurones	4
4	Conclusions	4

1 Objectif du TP

Il s'agit de se familiariser avec les réseaux de neurones en utilisant le logiciel SNNS.

Au cours du TP, on tentera de construire et d'entraîner deux réseaux de neurones. Le premier devra apprendre une fonction mathématique, le second devra apprendre à piloter un robot.

2 Apprentissage d'une fonction

2.1 Données préliminaires

On dispose de plusieurs fichiers sources. Ceux-ci nous permettent :

- de générer un fichier interprétable et affichable par GNUPLOT, représentant une surface gaussienne.
- de générer un fichier d'échantillonnage de cette surface, destiné à l'apprentissage.
- de générer des fichiers de patterns utilisables par le logiciel SNNS

2.2 Construction du réseau de neurones

A l'aide du logiciel SNNS (en fait, de son implémentation en Java), on construit un réseau de neurones à deux couches, avec trois neurones dans la couche cachée. La couche d'entrée dispose de deux neurones, la couche de sortie, un. La fonction est du type :

$$f : x, y \mapsto z$$

Il s'agit d'une fonction continue, qui, d'après le cours, doit être représentable avec un réseau de deux couches.

Un premier essai avec une couche cachée de trois neurones nous donne un résultat satisfaisant. En effet, l'ajout d'autres neurones du même type, ou l'ajout de couches cachées supplémentaires, nous donne un résultat similaire. Pour l'apprentissage, nous avons utilisé 30% des données disponibles.

3 Apprentissage d'une commande

On souhaite maintenant construire un réseau de neurones capable d'apprendre à piloter le robot `tobor`. Le robot `tobor` dispose de deux capteurs. L'un pour la distance à la cible, l'autre pour l'angle de la cible par rapport à la direction courante. Le robot se pilote avec deux variables : vitesse linéaire, et vitesse angulaire.

3.1 Une première approche

On utilise le mode "utilisateur" du robot virtuel `tobor` pour générer des données d'apprentissage. Il s'agit de contrôler les valeurs des vitesses angulaire et linéaire afin d'atteindre une cible. Une fois la cible atteinte, celle ci réapparaît à un autre endroit du plan de "jeu".

3.1.1 Premiers enregistrements, premier réseau de neurones

Après avoir compilé les fichiers qui vont bien, on commence la manipulation du robot. Les grands enfants que nous sommes réussissent à atteindre la cible un certain nombre de fois, afin de générer, l'espérons nous, suffisamment de données d'apprentissage.

On ne sait pas trop quelle type de fonction permet la commande du robot. Cependant, le cours nous dit que n'importe quelle fonction peut être approchée avec une précision arbitraire en utilisant un réseau de neurones à trois couches. Le résultat aisément obtenu lors de la première partie nous laisse penser que l'utilisation de deux couches cachées de 8 neurones chacune sera largement suffisante.

3.1.2 Vérification de l'apprentissage

Une fois "appris", le comportement du robot est testé sur le logiciel tobor. Celui ci s'avère être un abruti complet. Il tourne en rond, bêtement¹...

3.2 Retour d'expérience, réflexion

Dans ce genre de situations (lorsque ça ne marche pas du premier coup), il est parfois bon d'aller voir ce qu'a fait le voisin. Il s'avère que, selon les binômes, le nombre de neurones et de couche varie. De même, le pourcentage de données d'apprentissage diffère. Cependant, tout les robots semblent aussi idiots. Il est sans doute nécessaire d'engager une réflexion sur notre façon de procéder.

En effet, si notre réseau a "mal appris", c'est probablement parce que nous ne sommes pas des exemples à suivre.

3.2.1 Faites ce que je dis, pas ce que je fais

Par défaut, lorsque nous pilotons le robots, nous adoptons un comportement qui tiens compte d'éléments autres que ceux vues par celui ci. Par exemple, il nous arrive de rejoindre la cible en effectuant une rotation de plus de 180 degrés. En effet, si notre robot atteinds la cible, avec une vitesse angulaire positive (disons, dans le sens trigonométrique), et que la nouvelle cible se trouve sur notre droite, on aura tendance à poursuivre notre rotation plutôt que de l'inverser. Ce comportement, que l'on pourrait qualifier d'"au plus simple" (par opposition au "plus court"), fait intervenir des données d'entrées qui ne sont pas prises en compte par notre réseau de neurones : la vitesse angulaire angulaire du robot (et, dans une moindre mesure, sa vitesse linéaire).

3.3 L'exemple à suivre

Pour pallier à ce problème, on envisage deux solutions :

- Soit l'on apprend à notre réseau de neurones la flemmardise, en introduisant comme valeurs d'entrées l'état du robot. Cela implique une modification du réseau (4 entrées), mais il faut aussi générer des fichiers de traces différents (4 entrées, deux sorties).
- Soit l'on s'applique lors de la génération des fichiers de traces.

¹Mettons de coté ce jugement de valeur pour décrire plus précisément le comportement. En fait, plutôt que de se diriger dans la bonne direction, il semblerait que le robot "cherche" son chemin, mais certes, en tournant en rond. On pourrait en conclure que pour conduire, il faut plus de 20 neurones...

Quitte à faire des robots, autant leur épargner les défauts typiquement humains. On va donc s'appliquer.

3.3.1 Procédure d'élaboration des échantillons d'apprentissage

Pour que notre réseau de neurones apprenne, on va adopter une attitude plus simple. Lorsqu'une nouvelle cible apparaît, on adopte le comportement suivant :

1. Vitesse linéaire nulle
2. Rotation pour se mettre dans l'axe, en tournant "au plus court" (angle de rotation inférieur à 180°)
3. Vitesse linéaire non nulle pour atteindre la cible

On effectue un enregistrement d'une dizaine de cibles.

3.3.2 Réseau de neurones

Après une petite réflexion, on pense que les fonctions de vitesse angulaire et de vitesse linéaire sont continues. En effet, des fonctions du type

$$\omega(\alpha) = \alpha$$

et

$$v(d) = d + k_0$$

sembleraient convenir. Les fonctions de contrôle du robot sont donc probablement continues. Un réseau de deux couches devrait donc suffir.

On établit donc un réseau de deux couches, comprenant une couche cachée de 5 neurones. Avec 50% d'échantillonnage, la courbe de validation n'est pas satisfaisante. Cependant, un test nous permet de vérifier que le réseau sait piloter correctement le robot.

En adoptant le même comportement d'apprentissage, et avec des couches cachées de trois neurones, nos collègues obtiennent également des résultats satisfaisants.

4 Conclusions

On est capable d'enseigner à un réseau de neurones des fonctions plus ou moins complexes. Dans le premier cas d'étude, on disposait d'un échantillonnage de la fonction très pertinent (les valeurs étaient directement issues de la fonction). Notre réseau de neurones n'a eut aucun mal à apprendre la fonction. Dans le deuxième cas d'étude, l'apprentissage s'est basé sur notre exemple (la façon dont nous, humains, pilotons le robot). Nous ne sommes pas de bons professeurs : des facteurs extérieurs influencent notre conduite (paresse, esthétique ...) rendant les données d'apprentissage incohérentes.

On en conclura qu'il est essentiel d'avoir des données d'apprentissages conformes à ce que l'on souhaite enseigner.