

MENINGIX : Séparation de deux spirales par un réseau de neurones multicouches

Le but de ce projet est de séparer deux populations distinctes de points : les points rouges et les points bleus, chaque série de points étant située sur une spirale.

Pour ceci, on souhaite utiliser un réseau de neurones auquel on va faire "apprendre" où sont les points rouges et où sont les points bleus. Ce réseau de neurones devra ensuite être capable de séparer correctement les deux populations.

Points et spirales

Chaque point est repéré par ses coordonnées x et y et par sa couleur : rouge ou bleu

La spirale contenant les points bleus est une courbe paramétrique d'équation suivante :

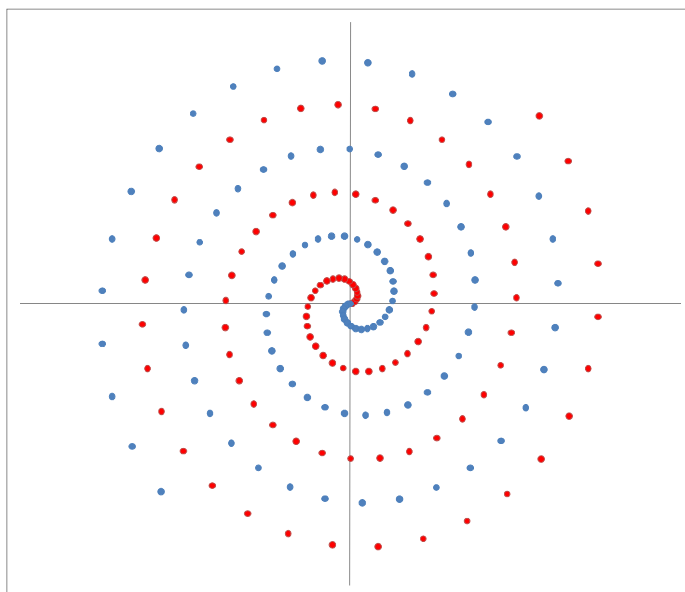
$$\begin{cases} x = t.\cos(t) \\ y = t.\sin(t) \end{cases}$$

pour t de 0 à $+\infty$. Les points bleus correspondent aux valeurs de t commençant à 0 avec un pas de 0,2. ($t = 0$ pour le premier point, $t = 0,2$ pour le deuxième, $t = 0,4$ pour le troisième, et ainsi de suite).

Suivant le même principe, les points rouges sont des échantillons de la spirale :

$$\begin{cases} x = -t.\cos(t) \\ y = -t.\sin(t) \end{cases}$$

, pour les valeurs discrètes de t commençant à 0 et évoluant avec un pas de 0,2.



Les deux spirales sont symétriques.

Réseau de neurones

Le neurone formel

Les neurones informatiques utilisés pour ce type de réseau sont en fait de simples opérateurs mathématiques qui reçoivent plusieurs valeurs en entrée et produisent une valeur en sortie (tiens, mais on dirait bien une fonction...). Ces valeurs sont des nombres réels. Les valeurs en entrée sont nommées x_i (il y a de 1 à n entrées), et la valeur de sortie est nommée y .

Les valeurs en entrée d'un neurone lui sont fournies par l'intermédiaire de synapses. Chaque synapse est caractérisée par un poids w_i .

Le neurone dispose d'une fonction de transfert f lui permettant de calculer la valeur de y en fonction des x_i et de w_i selon la formule suivante :

$$y = f\left(\sum_i w_i \cdot x_i\right)$$

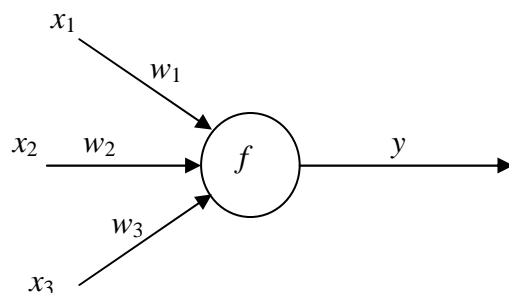


fig 1 : un neurone formel avec entrées, poids synaptiques et sortie

La fonction de transfert f à utiliser pour ce projet est la fonction tangente hyperbolique :

$$th(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Illustration sur l'exemple précédent : le neurone de la figure 1 calcule la somme S de ses entrées pondérées de la manière suivante :

$$S = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$$

$$\text{Puis calcule } y : y = f(S) = \frac{e^S - e^{-S}}{e^S + e^{-S}}$$

Cette valeur de sortie peut être directement interprétée ou utilisée comme entrée d'un autre neurone.

Réseau de neurones multicouches

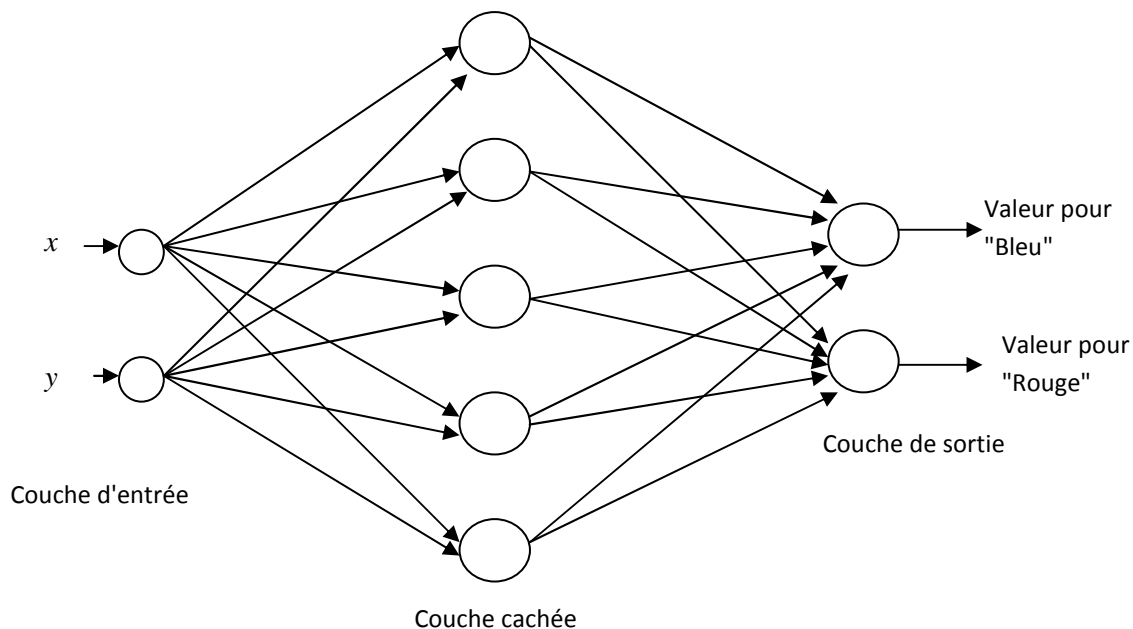
Un réseau de neurones multicouches consiste en plusieurs couches de neurones reliées les unes aux autres. On utilise une couche dite d'entrée qui recevra les valeurs du problème, avec un neurone pour chaque valeur à traiter. Dans notre cas, la couche d'entrée aura 2 neurones, un pour la valeur x , un pour la valeur y . Ces neurones d'entrée n'ont pas de poids d'entrée et fournissent en sortie les valeurs x et y .

On trouve ensuite 1 ou 2 couches dites cachées, comportant un certain nombre de neurones. La détermination de ce nombre reste délicate, car on ne sait pas exactement comment procède le réseau pour faire son apprentissage : il vous appartiendra donc de faire des essais, en sachant qu'une couche cachée comporte entre 3 et 8 neurones (au delà, le temps de calcul risque de s'en ressentir).

Enfin, on trouve une couche de sortie comportant autant de valeurs que de possibilité de résultat. Dans notre cas, la couche de sortie comportera 2 neurones, un indiquant quelle est la valeur associée à "Bleu" et un autre indiquant quelle est la valeur associée à "Rouge". Les valeurs recueillies sur les neurones de sortie indiquent grosso-modo la probabilité que le réseau associe aux entrées. Par exemple, si on fournit au réseau en entrée les valeurs $x=2$, $y=-1$ et qu'il répond par les valeurs 0,6 (pour rouge) et 0,03 (pour bleu), il indique que la probabilité que ce point appartienne à la spirale rouge est 20 fois plus élevée que la probabilité qu'il appartienne à la spirale bleue.

Les neurones d'une même couche ne sont pas connectés entre eux, mais prennent en entrée les sorties de tous les neurones de la couche précédente.

Illustration d'un réseau à une couche cachée de 5 neurones pour notre exemple :



Rappel : les neurones de la couche d'entrée produisent à leur sortie la même valeur que celle qu'ils ont sur leur entrée.

Principe de calcul

Le calcul de la réponse du réseau de neurone est séquentielle : on propage les valeurs couche par couche depuis l'entrée vers la sortie du réseau. Ainsi, à partir de deux valeurs x et y fournies en entrée, on calcule les entrées des neurones de la couche cachée. Ces neurones calculent une valeur en sortie, à l'aide de la formule indiquée dans la section précédente. Ces valeurs en sortie constituent les entrées des neurones de la couche de sortie. Ces neurones de la couche de sortie calculent à leur tour une valeur en sortie correspondant à la réponse du réseau.

Algorithme d'apprentissage

Les poids des synapses (flèches reliant les neurones) sont les seuls éléments du réseau de neurones que l'on peut modifier, et c'est sur ces poids synaptiques que l'on va agir pour modifier la réponse du réseau en lui indiquant quelle est l'erreur qu'il commet dans la réponse qu'il donne pour un jeu de valeurs en entrée.

Au départ, les poids synaptiques sont initialisés avec des valeurs aléatoires réelles (à vous de trouver le bon intervalle pour les poids initiaux !).

On présente au réseau un couple (x,y) pour lequel on connaît la valeur de la réponse attendue (par exemple, on sait que le point $(2,5)$ est "bleu". Le réseau fournit une réponse (forcément fautive au départ), par exemple : 0,2 pour bleu et 0,5 pour rouge. A partir de l'erreur sur la réponse fournie et celle attendue (on attendait 1 pour bleu et 0 pour rouge), on calcule, par **rétropropagation du gradient**, les nouvelles valeurs des poids synaptiques. Mais cette mise à jour ne concerne que l'exemple fourni pour ces valeurs de x et y , il faut donc recommencer !

L'algorithme de rétropropagation du gradient

Voici une version presque implémentée de l'algorithme de rétropropagation du gradient. Il en existe de nombreuses démonstrations mathématiques, mais elles ne nous intéresseront pas ici

vecteur_x représente les entrées fournies au réseau

vecteur_y représente la sortie attendue

S_i représente la valeur *calculée* en sortie du neurone i de la couche de sortie

Y_i représente la valeur *attendue* en sortie du neurone i de la couche de sortie

O_i représente la valeur *calculée* en sortie du neurone i d'une couche cachée

Δ_i représente un delta (différence) qui sera utilisée pour calculer la mise à jour effective des poids synaptiques du réseau.

w_{ij} représente le poids synaptique entre les neurones i et j

x_{ij} représente la valeur transportée par la synapse entre les neurones i et j

Répéter

Prendre un exemple (vecteur_x, vecteur_y) et calculer la sortie du réseau pour ce vecteur_x

```
Pour tout neurone de sortie i // couche de sortie
    di <- si(1-si)(yi-si)
finPour

Pour chaque couche de q-1 à 1
    Pour chaque neurone i de la couche courante
        di = oi(1-oi) * Somme [pour k appartenant aux
indices des neurones prenant en entrée la sortie du neurone i] de dk*w_ki
    finPour
finPour

Pour tout poids w_ij
    w_ij <- w_ij + epsilon*di*x_ij
finPour
finRépéter
```

(source : <http://alp.developpez.com/tutoriels/intelligence-artificielle/reseaux-de-neurones/#LV>)

Pour la mise à jour des poids (ligne en gras de l'algorithme), le facteur epsilon permet de ne pas appliquer en totalité la mise à jour calculée, car cela forcerait le réseau à n'apprendre que l'exemple courant. Un epsilon faible permet de faire une mise à jour partielle des poids. Il faudra répéter cette mise à jour partielle à travers de nombreux exemples pour que le réseau converge vers un état final.

Phase d'apprentissage

Dans cette phase, le but est de faire évoluer les poids de manière à ce que le réseau soit capable de reconnaître et séparer les spirales. Il faudra donc lui présenter au hasard des exemples de points issus de la courbe et appliquer pour chaque exemple l'algorithme de rétropropagation. L'apprentissage s'arrête lorsqu'on estime que le réseau n'apprend plus (c'est à dire que la mise à jour des poids devient très faible). On ne sait pas à l'avance combien d'exemples il faut présenter avant que l'apprentissage ne soit effectué. Il est même possible que le réseau n'apprenne plus (mise à jour des poids faibles) et que les réponses qu'il donne ne soient pas forcément correctes !).

L'idée est donc de regarder, lors de la mise à jour des poids pour un exemple, quelle est la modification de poids la plus élevée. Si elle est faible (seuil à déterminer par vous-même !), alors l'apprentissage s'arrête.

Phase de généralisation

Les poids du réseau étant maintenant figés, il est temps de regarder comment se comporte le réseau pour des entrées quelconques : on lui présente maintenant des couples (x,y) de coordonnées quelconques et on affiche le résultat en bleu ou en rouge (ou avec un mélange de couleurs par exemple si la réponse n'est pas tranchée) pour voir comment le réseau a appris les deux spirales. Bien entendu, généralisation implique que l'on va fournir au réseau des entrées qu'il n'aura pas appris lors de la phase précédente, c'est là tout l'intérêt de cette technique !

Aide à la modélisation du problème

Modéliser un neurone et une couche du réseau par une structure n'est pas forcément une bonne idée : en fait, un neurone ne fait pas grand-chose à part un calcul. Il peut être intéressant de **modéliser directement une couche** en indiquant :

- L'ensemble des valeurs qu'elle aura en entrée (en effet, tous les neurones d'une même couche reçoivent les mêmes valeurs x_i des neurones de la couche précédente);
- L'ensemble des poids associé à chaque neurone;
- L'ensemble des sorties calculées par les neurones de la couche (cet ensemble est en fait l'ensemble des entrées de la couche suivante).

Ainsi une couche du réseau serait représentée par :

```
double _entrees[N]; // où N est le nombre de neurones de la couche précédente
```

```
double _poids[N][P]; // tableau des poids, où P est le nombre de neurones de la couche actuelle
```

```
double _sorties[P]; // valeurs de sortie de la couche actuelle
```

Cahier des charges

Votre programme devra permettre de :

- Créer un réseau multicouches avec au maximum 2 couches cachées, et entre 3 et 8 neurones par couche cachée;
- Réaliser un apprentissage ;
- Montrer le résultat obtenu en phase de généralisation;
- Charger / sauvegarder une configuration connue : cela vous permettra de montrer rapidement un résultat sans repasser par une phase d'apprentissage qui peut être longue et pas forcément couronnée de succès : vous aurez de très nombreux tests à mener avant d'arriver à un résultat satisfaisant;

Dans votre rapport de projet, vous devrez également inclure une partie sur la campagne de tests que vous aurez effectuée, et conclure sur les choix finaux en termes de nombre de couches cachées et nombre de neurones pour le réseau qui donne, à votre avis, le meilleur résultat.

Vous devrez également vous documenter, **c'est normal**, il est difficile d'expliquer toute la théorie des réseaux de neurones dans un sujet de projet. Pensez donc à indiquer quelles sont les sources que vous aurez exploitées pour mener à bien ce projet ambitieux !

Planning du projet

Distribution du projet

Semaine du 29 Mars au 2 Avril

Constitution des trinômes

Chaque groupe devra fournir à M. FLASQUE avant le Mercredi 5 Mai une liste des **trinômes** formés dans le groupe. Cette liste peut être fournie par mail à : flasque@efrei.fr. Au delà de cette date, les trinômes seront attribués d'office.

TP avancement de projet

4 heures (par groupe) y seront consacrées le Mercredi 14 Avril.

Soutenance de projet

4 heures (par groupe) y seront consacrées le Mercredi 12 Mai.

Vous ferez une démonstration de votre programme, l'enseignant qui fait passer la soutenance vous posera des questions sur le programme ainsi que sur la manière dont vous vous êtes réparti les différentes tâches. Le rapport de projet devra être remis à votre enseignant lors de la soutenance.

Pour la soutenance, un ordre de passage sera disponible sur Campus EFREI.