
Laboratoire sur les RÉSEAUX DE NEURONES ARTIFICIELS

Objectifs

- Introduction aux réseaux de neurones artificiels;
- Familiarisation avec les étapes de conception et ses principes de mise en œuvre;
- Connaissance d'outils-types en Matlab pour les réseaux de neurones artificiels de type rétropropagation standard.

Directives

- Ce laboratoire est évalué sur 5%. Le travail consiste à remplir les questions demandées dans ce compte-rendu et de le remettre à la fin du laboratoire.
- Les scripts Matlab pour les réseaux de neurones artificiels sont donnés dans la librairie "nnet;nndemos" placé dans le répertoire de Matlab (sur PC: W:\Matlab\Toolbox; sur Macintosh: "NOVA.APP;App;Maths;Matlab 4.2.c.1;toolbox"). Les scripts pour le laboratoire sont donnés dans la librairie "rna" placé dans le répertoire du cours GEI 457. Utilisez Matlab 5 (il y a quelques problèmes avec la version 5.2).
- Ce laboratoire peut être fait de façon individuelle ou en équipe de deux. Je suis aussi disponible pour répondre à vos questions, mais seulement lors de la journée du laboratoire. Ceci n'est pas un examen, ni même un quiz. L'objectif est que vous vous familiarisiez avec les réseaux de neurones artificiels et leurs concepts.
- Vous pouvez joindre les impressions des graphes et des résultats demandés en les brochant à votre compte-rendu. Chacun des graphes et des impressions remis doit porter votre nom et numéro de matricule (pas écrit à la main: modifiez les scripts pour que ces informations puissent être sur l'impression). Vous pouvez copier vos résultats dans le formulaire de réponses "Labo RNA.x" en format Word 98, Word 6 ou Word 5 placés dans la librairie "rna", dans le répertoire du cours GEI 457.
- Veuillez bien identifier votre copie de compte-rendu de laboratoire.

MATRICULE	NOM	PRÉNOM	SIGNATURE

Bon laboratoire!

Partie 1 Fonctions de base à connaître

Dans les scripts que vous allez utiliser, un certain nombre de scripts sont fréquemment utilisés. Des descriptions sommaires sont données ici afin de faciliter votre compréhension. Pour plus d'explications sur les RNA à propagation avant et la loi de rétropropagation standard, veuillez vous référer à votre manuel de cours.

initff: Création d'un RNA à propagation avant jusqu'à trois couches. Initialise les poids et les biais du RNA. Cette fonction prend la matrice de vecteurs d'entrées P , la fonction de transfert de chaque couche F , et retourne la matrice de poids W et la matrice des biais b pour chaque couche. Ex. pour un RNA à 8 unités cachées, 4 unités de sorties, deux couches, avec des fonctions sigmoïdales:

```
[W1,b1,W2,b2]=initff(P,8,'logsig',4,'logsig');
```

Il est important que P contienne les valeurs maximale et minimale de chaque entrée afin de calculer les meilleurs poids et les biais initiaux. Sinon, il est possible de donner l'intervalle en question. Ex. pour une première unité d'entrée définie sur l'intervalle ± 10 et la seconde sur l'intervalle de 0 à 5:

```
[W1,b1,W2,b2]=initff([-10 10; 0 5],8,'logsig',4,'logsig');
```

Si T est la matrice de sortie, on peut aussi utiliser:

```
[W1,b1,W2,b2]=initff(P,8,'logsig',T,'logsig');
```

simuff: Propagation avant d'un vecteur d'entrée à un RNA à propagation avant jusqu'à trois couches. Cette fonction prend le vecteurs d'entrées p , la matrice de poids W et la matrice des biais b pour chaque couche, ainsi que la fonction de transfert de chaque couche F , et retourne les sorties a de chaque couche. Ex.:

```
a=simuff(p,W,b,'logsig'); %1 couche
[a1,a2]=simuff(p,W1,b1,'logsig',W2,b2,'logsig'); %2 couches
[a1,a2,a3]=simuff(p,W1,b1,'logsig',W2,b2,'logsig',W3,b3,
                'logsig'); %3 couches
```

deltalog: Dérivée de l'erreur (le *delta vector*) pour une couche du RNA à propagation avant. La fonction **deltalog(a,e)** retourne ce vecteur pour une couche de sortie utilisant la sigmoïde comme fonction d'activation, e étant le vecteur d'erreur du RNA. La fonction **deltalog(a,d,W)** retourne ce vecteur pour une couche cachée, avec d étant le vecteur delta de la couche de sortie.

logsig: Fonction d'activation de type sigmoïde (entre 0 et 1).

learnbp: Retourne les variations de poids et de biais. Cette fonction prend le vecteurs d'entrées p , le vecteur delta d et le rythme d'apprentissage lr . Ex.:

```
[dW,db]=learnbp(p,d,lr);
```

learnbpm: Retourne les variations de poids et de biais. Cette fonction prend le vecteurs d'entrées p , le vecteur delta d , le rythme d'apprentissage lr , le momentum mc , et les variations de poids et de biais pour le cycle précédent. Ex.:

```
[dW,db]=learnbpm(p,d,lr,mc,dW,db).
```

trainbp: Fonction qui effectue l'apprentissage d'un RNA à propagation avant, jusqu'à trois couches. Cette fonction prend la matrice de poids W et de biais b , les fonctions de transfert F , pour chacune des couches. Ces arguments sont suivis des vecteurs¹ d'entrée P , les vecteurs de sorties T , et les paramètres d'apprentissage tp . Cette dernière variable indique la fréquence de présentation des résultats, le nombre maximum d'époques d'entraînement, la tolérance sur la somme des erreurs au carré jugée acceptable (pour indiquer quand l'entraînement peut être arrêté) et le rythme d'apprentissage. La fonction retourne les nouvelles matrices de poids et de biais, le nombre d'époques d'apprentissage et les erreurs enregistrées tr . Ex.:

```
tp = [display_frequency max_epoch err_goal lr];
[W,b,epochs,tr]= trainbp(W,b,'F',P,T,tp); % 1 couche
[W1,b1,W2,b2,epochs,tr]= trainbp(W1,b1,'F1',W2,b2,'F2',P,T,tp);
% 2 couches
```

trainbpm: Fonction qui effectue l'apprentissage d'un RNA à propagation avant, jusqu'à trois couches, avec momentum.

trainbpx: Fonction qui effectue l'apprentissage d'un RNA à propagation avant, jusqu'à trois couches, avec momentum et rythme d'apprentissage variable.

trainlm: Fonction qui effectue l'apprentissage d'un RNA à propagation avant, jusqu'à trois couches, avec des variantes de type rétropropagation.

Note: On utilise habituellement la variable R pour indiquer le nombre de couches, Q pour le nombre de sorties, et S_x pour indiquer le nombre d'unités sur la couche x .

¹ Si on désire faire du *batch learning*, on soumet les matrices de vecteurs d'entrée et les matrices de vecteurs de sortie. Sinon, on peut soumettre les vecteurs un par un.

Partie 2 Démonstration des principes de base - Effet de la rétropropagation

• Ouvrez le script “bp1”. Ce script définit un problème simple utilisant un RNA à une entrée, une sortie. Le neurone de sortie utilise une fonction de transfert sigmoïdale, et la sortie obtenue correspond à $a = \text{logsig}(w \cdot p + b)$, avec w , le poids sur le lien entre le neurone d’entrée et le neurone de sortie, b , le biais, p , l’activation du neurone d’entrée.

NOTE IMPORTANTE: Le biais b correspond en fait à la position centrale de la fonction d’activation $g(x)$ sur l’axe des x . On le qualifie souvent de seuil ou *threshold*. Le fait de varier b en cours d’apprentissage permet de translater la fonction d’activation (i.e., modifier son seuil). Pour y arriver, on modélise le biais sous la forme d’un poids, liant un neurone ne recevant aucune entrée et dont l’activation est maximale et toujours fixe, au neurone qui reçoit le biais pour évaluer son activation.

Représentez le RNA par un schéma simple:

On donne deux vecteurs d’apprentissage à ce RNA: si on donne -3 en entrée, le RNA doit retourner 0.4; si on donne 2 en entrée, le RNA doit retourner 0.8. Deux variables doivent être déterminées par apprentissage dans ce RNA, w et b , en fonction de l’erreur. Sachant que la loi d’apprentissage utilise la dérivée de l’erreur en fonction des poids et biais du RNA, on peut représenter pour ce problème l’erreur du RNA en fonction de w et b . Activez le script pour voir cette représentation. On peut voir une représentation en trois dimensions et une représentation des contours de la surface d’erreur (plus les lignes sont rapprochées, plus la variation de l’erreur est grande). Notez qu’habituellement, il n’est pas possible de tracer de telles représentations, car le nombre de poids et de biais est beaucoup plus grand (la surface d’erreur est alors représentée en N -dimensions, où N est le nombre de poids et de biais).

La première étape pour l’entraînement du RNA est d’initialiser le poids et le biais avec **initff**. Leurs valeurs initiales respectives sont illustrées par un ‘+’ sur la représentation de contours. Lorsque le script vous demandera quels poids et biais vous voulez utiliser, prenez les valeurs du guide de l’usager (choix 2).

Ensuite, le script initialise les paramètres d’apprentissage (la fréquence d’affichage, le nombre maximum d’époques, la tolérance désirée de l’erreur (en terme de somme des carrés des erreurs) et le rythme d’apprentissage). Le RNA est ensuite prêt à être entraîné en utilisant **tbp1**, une procédure similaire à **trainbp** mais plus spécifique à

des RNA à une couche (i.e., sans couche cachée). Au cours de l'apprentissage, w et b sont ajustés pour déplacer le RNA vers une erreur minimale.

Quel est le nombre d'époques d'apprentissage requis?

Quelle est la précision atteinte?

Comment faire pour l'améliorer?

Le script montre ensuite un graphe de la somme du carré des erreurs, et les valeurs d'activation pour les deux entrées données lors de l'apprentissage.

Quelles sont les valeurs de sortie pour -3 et 2 appliquées en entrée?

Avec l'aide des valeurs de w et de b résultantes, retrouver mathématiquement les sorties obtenues (Note: puisque le neurone de sortie utilise une fonction sigmoïdale définie entre 0 et 1, vous ne pouvez obtenir de réponses négatives – ceci est un signe d'erreur de calcul).

Si on changeait dans le script la tolérance désirée de la somme des erreurs au carré pour 0.001, quelles seraient le nombre d'époques d'apprentissage (en prenant les mêmes poids et biais initiaux) et les valeurs de sorties obtenues pour les vecteurs d'entrée?

Qu'en déduisez-vous?

Si vous placez le point de départ sur le plateau en $w = -0.4$, $b = 4$, que se passe-t-il?

Partie 3 Surface d'erreur non-linéaire

La surface d'erreur d'un RNA n'a habituellement pas un seul minimum, et il est possible que le RNA converge vers un minimum local et y reste pris. L'objectif est plutôt d'atteindre le minimum global de la surface d'erreur, mais ceci n'est pas garanti.

Le script "bp2" réalise l'apprentissage d'un RNA à une entrée, une sortie, avec huit vecteurs d'apprentissage. Pour cette application, la surface d'erreur a un minimum en plein centre, et deux vallées sur les côtés mènent à des minimums locaux (en haut à droite, et en bas à droite). Si vous utilisez le poids et le biais initiaux du guide de l'utilisateur, vous constaterez que le RNA converge vers le minimum global.

Quel est le rythme d'apprentissage et le nombre d'époques requis pour réaliser l'entraînement du RNA?

Répétez l'entraînement avec $w = -0.1$, $b = -2.4$.

Que se passe-t-il? Pourquoi?

En conclusion, il faut toujours prendre en considération que le RNA puisse converger vers un minimum local, et ceci dépend, entre autres, du point de départ du RNA sur la surface d'erreur. Ce point de départ dépend de l'initialisation des poids et des biais du RNA.

Partie 4 Effet du rythme d'apprentissage

Un paramètre très important pour l'entraînement du RNA est le rythme d'apprentissage. Il n'existe pas de formule qui permet de fixer ce paramètre pour une application donnée. Le meilleur moyen est de faire quelques tests, et de comprendre l'effet de ce paramètre sur les performances d'apprentissage du RNA.

Avec le script "bp3", vous pouvez voir l'effet du rythme d'apprentissage sur la convergence du RNA en fonction de la surface d'erreur. Pour les prochaines questions, repartez toujours de la position par défaut.

Si le rythme d'apprentissage est à 2, que se passe-t-il?

Si le rythme d'apprentissage est à 8, que se passe-t-il? Pourquoi?

Si le rythme d'apprentissage est à 0.16, que se passe-t-il? Combien de fois devez-vous réactiver l'entraînement (avec Go) avant d'arriver au minimum global de la surface d'erreur?

Lorsque vous voulez utiliser un RNA pour un problème donné, vous devez établir un compromis entre un rythme d'apprentissage élevé pour un entraînement rapide, et suffisamment petit pour converger vers un minimum. Ce choix n'est pas toujours facile à réaliser: des variantes existent où le rythme d'apprentissage est modifié en cours d'apprentissage en fonction de la variation de l'erreur (si la variation de l'erreur est petite, alors on augmente le rythme d'apprentissage; si elle est grande, alors on diminue le rythme d'apprentissage).

Partie 5 Apprentissage avec momentum

L'utilisation du momentum lors de l'apprentissage permet de diminuer la sensibilité du RNA aux petits détails présents dans la surface d'apprentissage, comme les minimums locaux. Comme expliqué dans le volume de cours, le momentum ajoute à la variation des poids et biais une fraction (déterminée par le paramètre momentum) de la variation des poids et biais de l'époque précédente. On peut comparer cet effet à une accélération (ou un momentum!). Pour mieux comprendre son effet, le script "bp4" réalise l'entraînement d'un RNA à une entrée, une sortie. En partant toujours de la position par défaut, répondez aux questions suivantes:

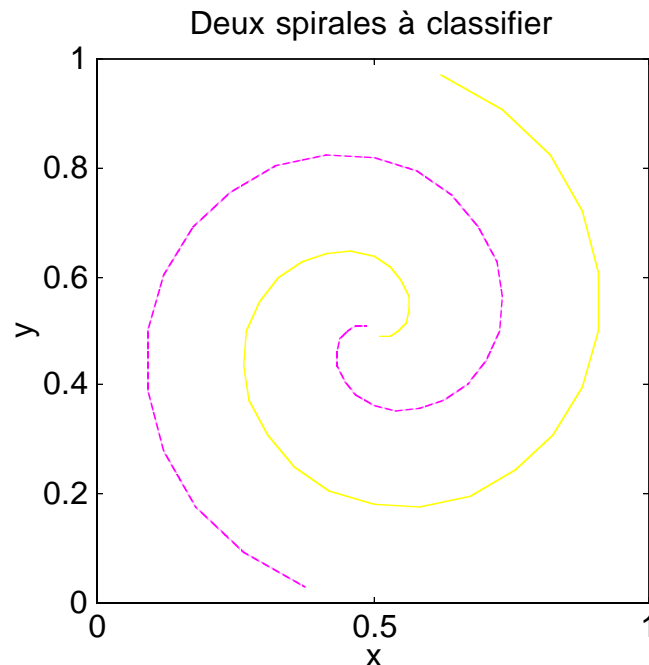
Que se passe-t-il lorsque le facteur de momentum est nul?

Que se passe-t-il lorsque le facteur de momentum est 1?

*Quelle est la valeur de momentum permettant d'atteindre le minimum global?
Combien de fois devez-vous réactiver l'entraînement (avec Go) pour y arriver?*

Partie 6 Distinction entre deux spirales - Problème théorique

Passons maintenant à une première application, soit la distinction de deux spirales à partir de points (x,y) donnés en entrée. Dans une telle application, il serait difficile d'élaborer des règles permettant de modéliser cette classification.



Pour ce problème, quelle est le nombre d'entrées et de sortie du RNA?

Le script “spirale” sera utilisé ici pour l’entraînement d’un RNA pour cette application. Puisque ce problème est non-linéaire, un RNA à deux couches est nécessaire. En plus de l’influence des poids initiaux du RNA, du rythme d’apprentissage et du momentum, il reste à déterminer le nombre d’unités cachées. Ce nombre doit être suffisamment grand pour permettre au RNA d’établir la fonction de correspondance désirée, mais pas trop grand pour permettre la généralisation du RNA. Comme critère pour déterminer si un vecteur fut appris par le RNA ou non, la dernière partie du script examine si l’erreur sur chacun des vecteur est supérieure à 0.5 (soit le niveau servant de distinction entre les deux spirales).

En gardant le rythme d’apprentissage et le momentum aux valeurs données dans le script ($lr = 0.15$ et $mc = 0.9$) (soyez patient!):

Essayez avec 5 unités cachées. Le RNA est-il en mesure de distinguer les deux spirales? Imprimez vos résultats en cours d’apprentissage, et le nombre de vecteurs non appris. Indiquez vos résultats sur le formulaire de réponses (réf.: Directives page 1).

Essayez avec 9 unités cachées. Le RNA est-il en mesure de distinguer les deux spirales? Imprimez vos résultats en cours d’apprentissage, et le nombre de vecteurs non appris. Indiquez vos résultats sur le formulaire de réponses (réf.: Directives page 1).

Expliquez pourquoi, si on répète l’apprentissage avec une même configuration, les résultats diffèrent?

Partie 7 Application de reconnaissance de caractères

Développer des machines capables de lire des caractères peut être très utile dans le traitement automatique du courrier ou pour lire des chèques. La présente application démontre comment un RNA à rétropropagation peut arriver à reconnaître des caractères simples.

Le RNA sera conçu et entraîné pour reconnaître les 26 lettres de l’alphabet. On suppose avoir à notre disposition un système qui centre chaque lettre dans son champ de vision et la digitalise. Le résultat est une représentation dans un plan de 5 par 7 de la présence (1) ou non (0) d’encre. L’objectif est de classifier les entrées digitalisées pour indiquer laquelle des 26 lettres est reconnue. Pour chacun des

vecteurs d'entrées, la sortie désirée consiste à assigner un 1 au neurone pour la lettre correspondante, et 0 pour les autres. Consultez le fichier "prprob" (dans le répertoire pour le laboratoire ou dans le toolbox *nnet* de Matlab) pour visualiser les vecteurs pour chacune des lettres de l'alphabet. Il est à noter que la digitalisation peut comporter du bruit, et il sera important de valider cet aspect au cours de l'entraînement du RNA et par rapport aux performances de l'apprentissage.

Combien de neurones d'entrées et de neurones de sorties sont nécessaires pour le RNA?

Le RNA que nous utiliserons aura deux couches, soit une seule couche cachée. Les fonctions de transfert des neurones sont de type sigmoïdal. Une fois le RNA initialisé, le script "appcr" effectue les opérations suivantes

- Entraînement du RNA avec les données du fichier "prprob", pour un maximum de 5000 époques;
- Copie des poids du RNA pour créer un deuxième RNA avec lequel on effectuera un apprentissage avec trois ensembles de données bruitées pour un maximum de 300 époques. La tolérance sur la somme des erreurs au carré est plus grande;
- Réapprentissage du deuxième RNA avec les données sans bruit;
- Validation des deux RNA avec différents niveaux de bruit, et impression des résultats.

Avant de commencer, ajoutez votre nom et numéro de matricule dans le script.

En gardant le rythme d'apprentissage et le momentum constant, tentez de trouver le nombre d'unités cachées minimales pour que le RNA soit en mesure de respecter le critère d'arrêt d'entraînement (basé sur la somme du carré des erreurs SSE) dans le nombre d'époques prescrit.

Imprimer les résultats obtenus (nombre d'unités cachées, somme du carré des erreurs à la fin de l'entraînement du premier RNA, graphe de comparaison des erreurs pour les deux RNA). Indiquez vos résultats sur le formulaire de réponses (réf.: Directives page 1).

Avec le script “testCharac”, vous pouvez aussi utiliser les RNA entraînés pour traiter des entrées que vous auriez définies. Il suffit de modifier la variable “lettre” dans ce script pour définir votre entrée, enregistrer le script et donner la commande (dans la fenêtre *Command*):

```
testCharac(W1,b1,W2,b2) %Pour le premier RNA
```

```
testCharac(W1n,b1n,W2n,b2n) %Pour le deuxième RNA
```

Imprimer les résultats affichés pour la lettre donnée dans “testCharac”, et indiquer la lettre identifiée par le RNA. Compléter le script pour retourner la lettre en question. Indiquez vos résultats sur le formulaire de réponses (réf.: Directives page 1). Essayez avec les lettres M et U et commentez les résultats obtenus.