

Cahier d'expériences

Exploration de la notion de méta-apprentissage

Dans quelle mesure un système apprenant peut prendre conscience de ses performances et altérer son comportement ?

Yann Boniface, Alain Dutech, Nicolas Rougier
Matthieu Zimmer

25 juin 2012

Table des matières

Table de matière	3
Introduction	4
Workflow	6
Expérience A1	7
Expérience A2	13
Expérience A3	19
Expérience A4	25
Expérience B1	30
Expérience B2	34
Expérience B3	38
Expérience B4	42
Expérience B5	46
Expérience C1	50
Expérience C2	54
Expérience C3	58
Expérience C4	62
Expérience D1	65
Expérience D2	69
Expérience D3	73
Expérience D4	77
Expérience F1	81
Expérience F2	85
Expérience F3	89
Expérience F4	93
Expérience F5	97
Expérience F6	100
Expérience G1	104

Expérience G2	108
Expériences FG	112

Introduction

SoN apprend plus vite

En regardant les résultats des expériences Expérience A1, Expérience A2 et Expérience A3, on peut s'interroger sur l'expression “apprendre plus vite”.

Si on regarde l'erreur de classification (graphes de droite) on voit :

- expA1 : au début, SoN classifie mieux mais à partir de l'épisode 180, ce n'est plus le cas.
- expA2 : FoN classifie toujours mieux que SoN.
- expA3 : il semble que FoN classifie toujours un peu mieux que SoN.

Par contre, il est vrai que les courbes d'erreur normalisées peuvent faire penser que SoN apprend plus vite. Les graphes montrent que SoN apprend rapidement à “recopier” la couche cachée de FoN (mais c'est une input). Par contre il apprend beaucoup moins facilement à retrouver les entrées, surtout dans le cas des données manuscrites (ce qui paraît logique, car on n'est plus dans des relations bijectives entre les couches d'entrée et cachées de FoN). Pour ce qui est d'apprendre les valeurs de la couche de sortie, on est assez surpris de constater que l'apprentissage est plus rapide que pour la couche cachée.

Au final, on peut se demander ce qu'on entend par “apprendre plus vite” pour SoN.

Expériences B

Des représentations pertinentes pour différencier les différentes classes en entrées semblent effectivement apprises relativement vite car, quand on fige les poids de la couche cachée de FoN, le changement de tâche (qui sont toujours des permutations sur la couche de sortie) est appris rapidement. Quand on ne fige pas les poids, j'imagine que la couche cachée essaie de se reconfigurer, ce qui est plus lent, et sans doute d'autant plus difficile que certains neurones doivent saturer et que leur dérivée est donc quasi-nulle, ce qui ralenti le changement de valeur des poids.

SoN apprend lentement pour la même raison : la couche cachée essaie de se reconfigurer, ce qui ne doit pas être rapide pour au moins quelques un de ses neurones (ceux qui saturent)... Pour preuve l'Expérience B4 montre que SoN arrive à se reconfigurer quand ses propres poids sont bloqués.

Si on prend des tâches qui ne sont plus de simples permutations, ça semble marcher quand même (cf Expérience B5).

Expériences C

Deux questions ont motivé la reprise de cette expérience et le fait de jouer un peu avec les paramètres des réseaux.

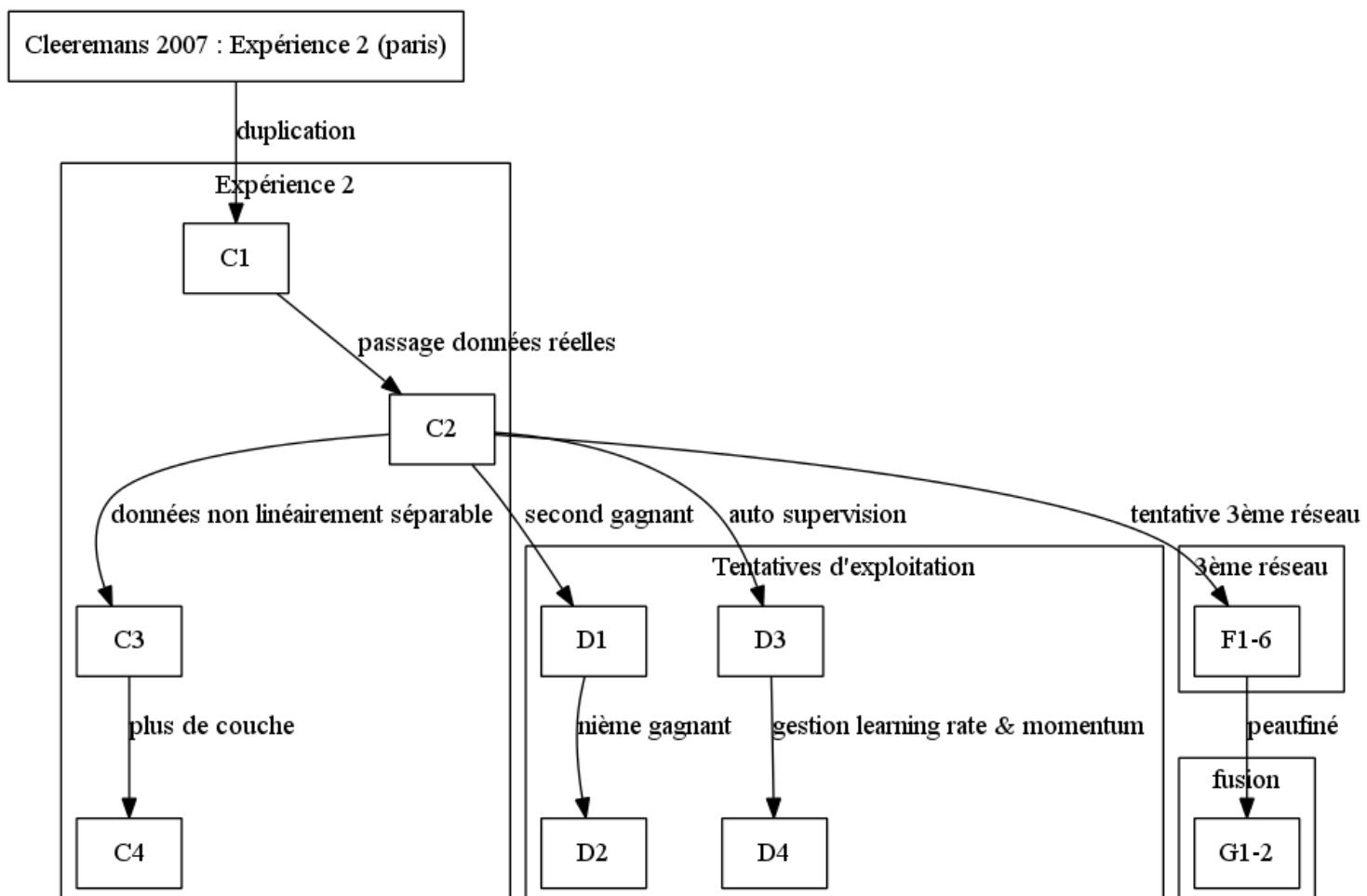
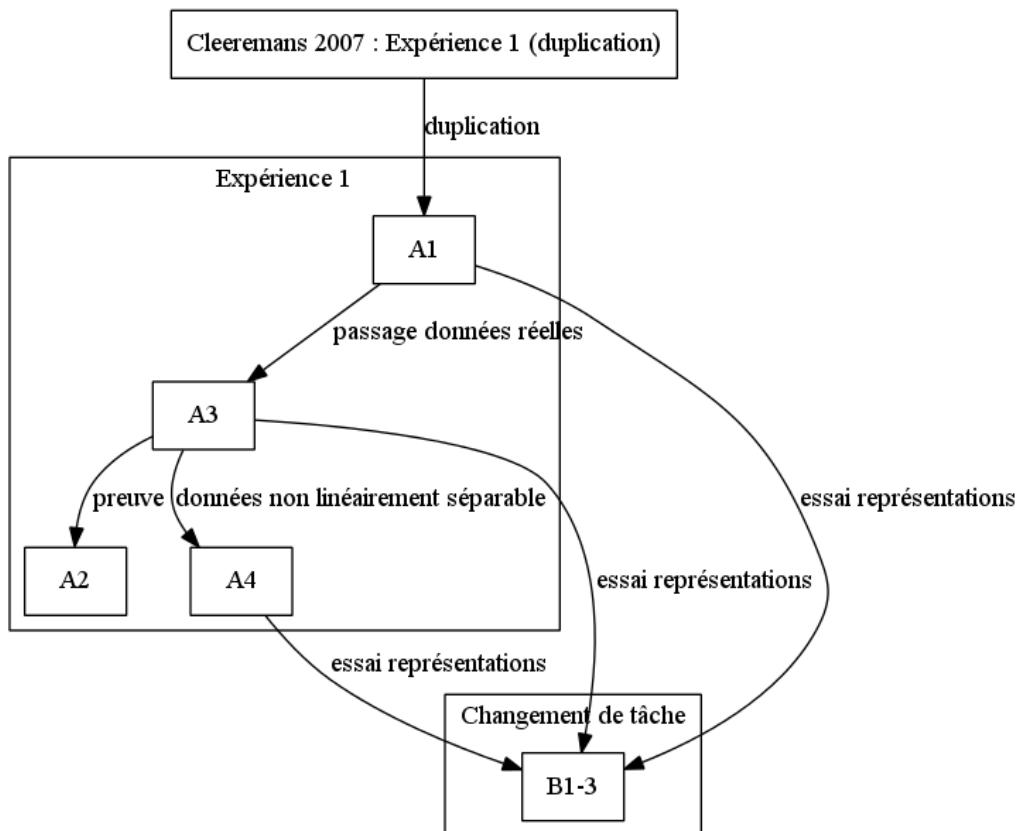
D'une part, nous voulions en savoir plus sur le comportement du réseau SoN. Quand on a vu qu'il se “contentait” de parler “low” pendant les premiers épisodes (tant que FoN fait plus de 50% d'erreur) puis “high” ensuite (quand le taux de succès de FoN dépasse 50%), nous nous sommes dit que c'était le plus comportement le plus simple. Mais pas forcément le plus efficace. Ainsi, dans un deuxième temps, comme nous étions persuadés qu'avec 100 neurones dans la couche cachée, le SoN devrait être capable d'apprendre “par coeur” les cas où FoN se trompe, nous avons expérimenté avec les paramètres d'apprentissage jusqu'à trouver les résultats présentés qui confirme ce que nous pensions.

L'?? ne fonctionne pas car les données de la couche cachée ne discriment pas de façon non ambiguë les entrées. Il y a sans doute une généralisation effectuée par la couche cachée du FoN.

Expériences D

Les Expérience D3 et Expérience D4 sont intéressantes, elles montrent qu'il est possible pour une architecture de “monitorer” son comportement de manière à influencer les paramètres d'apprentissage pour augmenter les performances.

Workflow



Expérience A1

Objectif

Comprendre de quelles manières peuvent émerger des représentations et métaréprésentations dans un réseau de neurone connexionniste, plus particulièrement sur des perceptrons multicouches.

Reproduction et approfondissement des résultats de la première expérience de l'article Cleeremans Alex (2007).

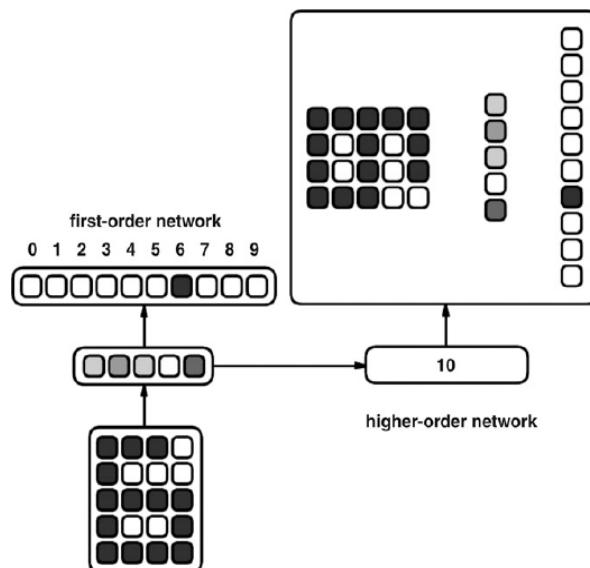
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 20 neurones d'entrées. Il est composé d'une couche cachée de 5 neurones.

Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

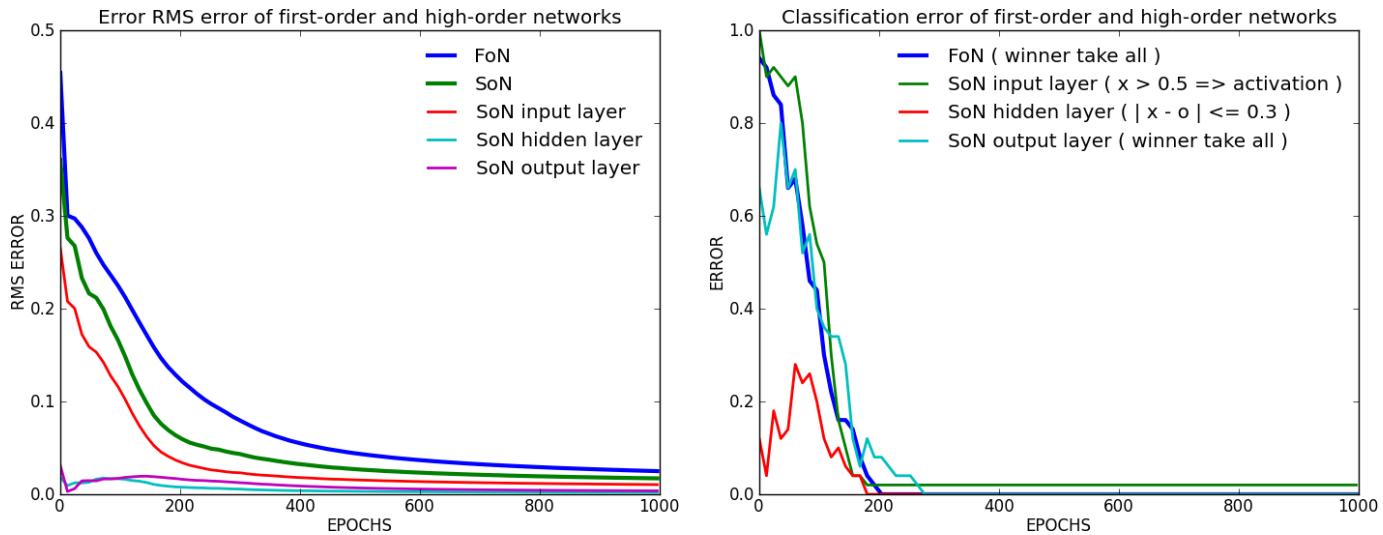


Paramètres

- momentum : 0.9 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- 10 chiffres différents présentés
- apprentissage 10 (formes) x 1000 (époques)
- utilisation de biais
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



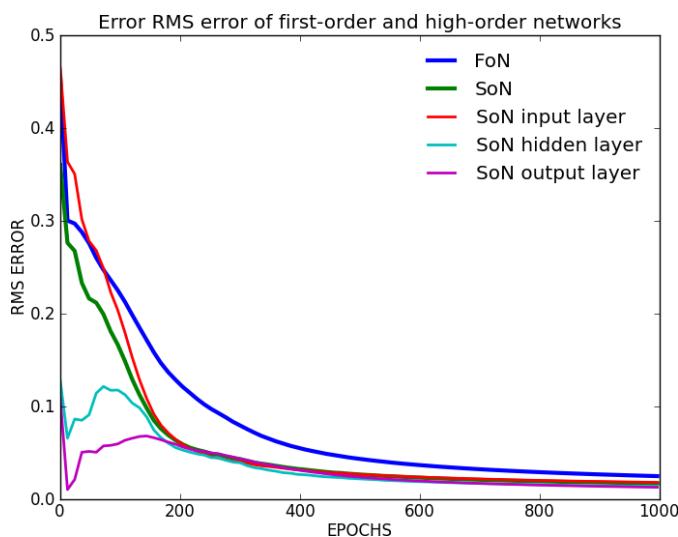
Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer
- l'erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n'est pas possible (ce n'est pas de la classification, mais une duplication), il y a donc un seuil d'erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- le premier réseau réussit à apprendre sa tâche de classification
- le second réseau réussit à apprendre sa tâche de duplication
- la couche cachée et la couche de sortie ne posent aucun problèmes d'apprentissage
- les performances du second réseau dépendent principalement de sa capacité à reproduire les entrées
- le second réseau apprend plus rapidement que le premier

Secondaires RMS indépendant par couche



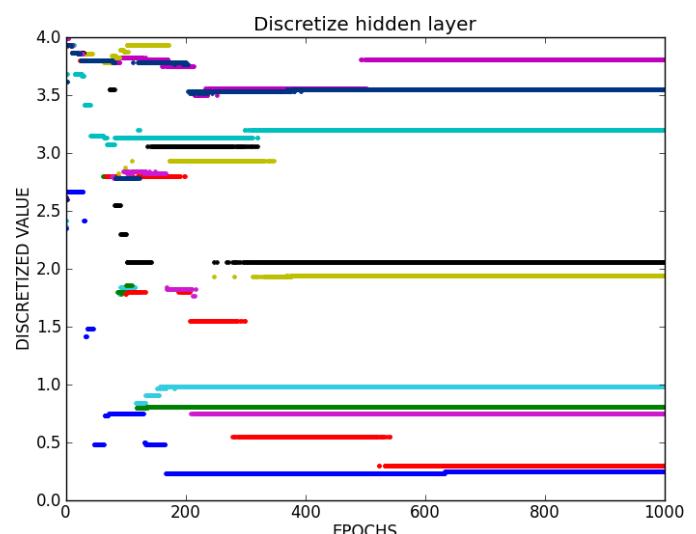
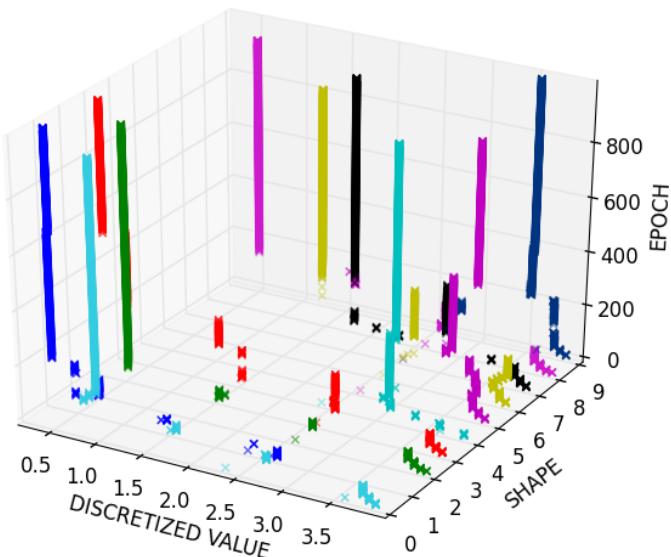
Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseau) sur les couches à reproduire
- les erreurs de SoN sont calculées en divisant par le nombre de neurone de la couche à reproduire (et non par le nombre total de neurones)

Conclusion

- le SoN a plus de mal à apprendre la couche cachée que celle de sortie

Secondaires Discrétisation de la couche cachée du premier réseau



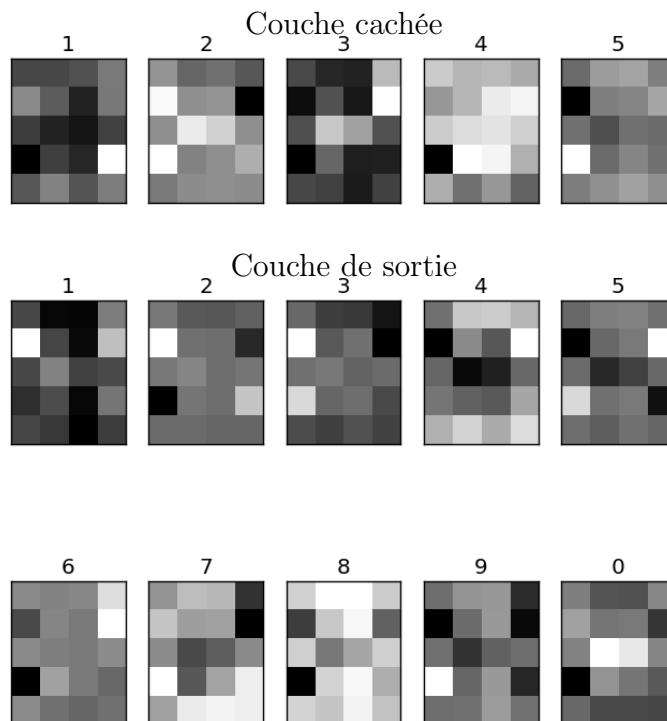
Notes

- une couleur équivaut à un chiffre présenté
- une valeur discrétisée correspond à un certain encodage de la couche cachée (cf Algorithmes Discrétisation)

Conclusion On peut voir ici, qu'un simple perceptron permet de résoudre le problème ; puisqu'à la fin, aucunes des couleurs n'est superposées à une autre.

Les neurones se stabilisent très rapidement (autour de la 50^{ième} époque en moyenne), le tout permettant au second réseau d'avoir des entrées très peu variables, favorisant son apprentissage.

Secondaires Représentations au travers des poids du premier réseau

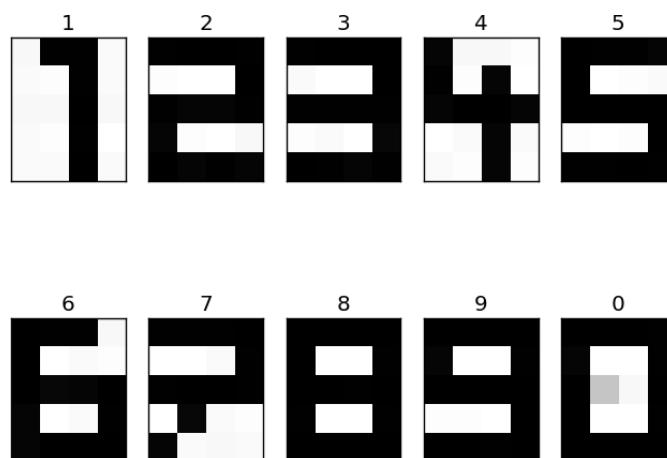


Notes

- plus une case est noire, plus sa présence est importante pour le chiffre en question
- plus une case est blanche, plus son absence est importante
- c'est seulement lorsqu'elle est grise qu'elle n'est pas prise en compte

Conclusion Au vu du peu d'entrées différentes présentées au réseau (10), on peut voir qu'il fixe 3 à 4 points précis pour classifier un nombre.

Secondaires Prototypes à l'intérieur des neurones du second réseau qui essaient de redonner les entrées



Notes

- Il sagit de la moyenne des réponses du second réseaux sur toutes les entrées

Conclusion Le peu d'entrées permet l'apprentissage par-coeur de chaque forme. À tel point qu'on ne peut pas vraiment parler de prototype.

Conclusion

Comme il l'est dit dans Cleeremans Alex (2007), cette architecture tente de résoudre les problèmes des réseaux connexionnistes classiques à avoir un semblant de conscience.

À savoir :

- qu'ils ne savent pas qu'ils peuvent se trouver dans différents états, et qu'ils ne traitent pas leurs propres états : d'où la présence de ce second réseau qui tente de traiter ses états et d'apprendre qu'il en a plusieurs
- des représentations restant bloquées dans la chaîne de causalité de la tâche à apprendre : d'où le fait que ce second réseau n'affecte pas l'apprentissage du premier (ie. pour ne pas retomber dans la chaîne de causalité)

Au vu des résultats secondaires qui nous montre que la tâche est vraiment facile, dans l'Expérience A3, nous nous sommes intéressés à la même architecture mais sur des données réelles.

Quant à l'Expérience B1, nous avons expérimenté les représentations et le transfert de tâche.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number \text{ of neurons on the output layer} \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Discrétisation Pour la couche cachée $hiddenNeuron$ de n neurones, un neurone pouvant être encodé par $number_cutting$ valeurs différentes :

$$\sum_{i=0}^n number_cutting^i \times cutting(hiddenNeuron[i])$$

Exemple $400 \leftarrow [0 ; 0, 25] [0 ; 0, 25] [0, 25 ; 0, 5] [0, 5 ; 0, 75] [0, 25 ; 0, 5]$
 $400 \leftarrow 0 \times 4^0 + 0 \times 4^1 + 1 \times 4^2 + 2 \times 4^3 + 1 \times 4^4$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction \text{ sigmoide} \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionisme.

Expérience A2

Objectif

Comprendre de quelles manières peuvent émerger des représentations et métaréprésentations dans un réseau de neurone connexionniste, plus particulièrement sur des perceptrons multicouches.

Dans l'optique d'éprouver la première expérience de l'article Cleeremans Alex (2007) sur des données plus réelles, il sagit, dans un premier temps, de montrer qu'une augmentation du nombre de neurones, et qu'un simple agrandissement des chiffres en entrées, n'affecte pas fondamentalement le fonctionnement du réseau.

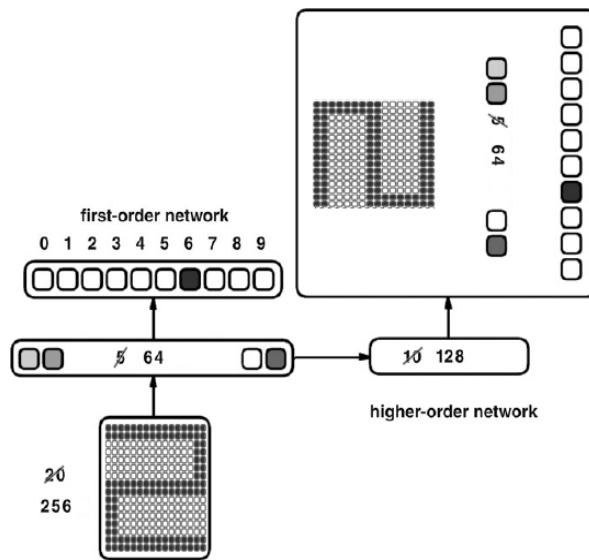
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 64 neurones.

Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

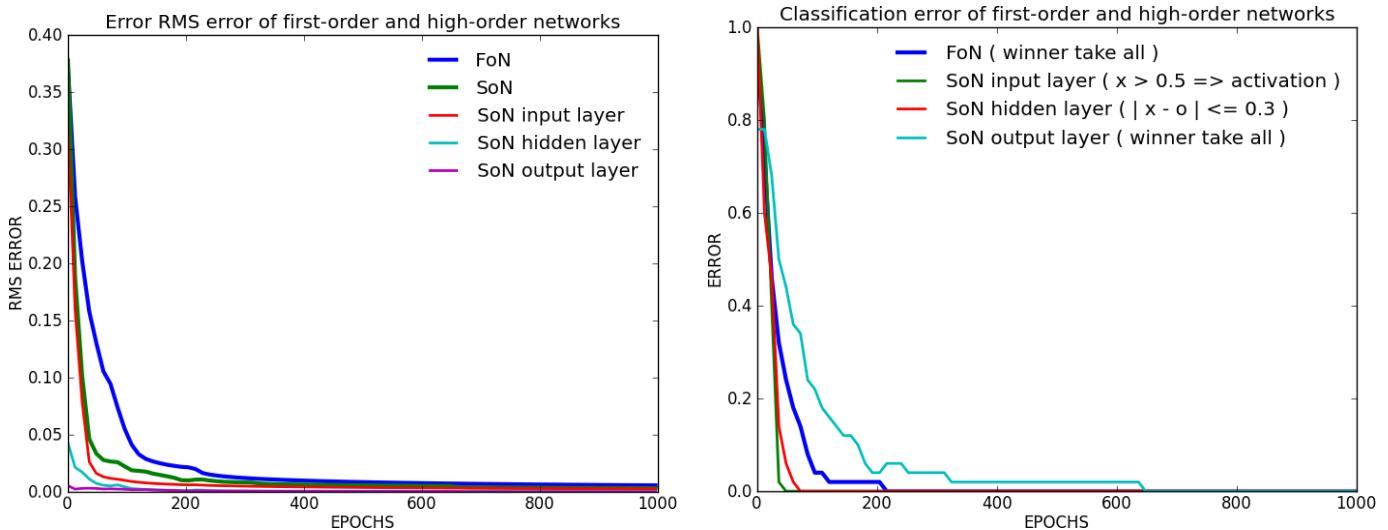


Paramètres

- momentum : 0.9 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- **10 formes** de chiffres différents présentées (shuffle)
- apprentissage 10 (formes) x 1000 (époques)
- utilisation de biais
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



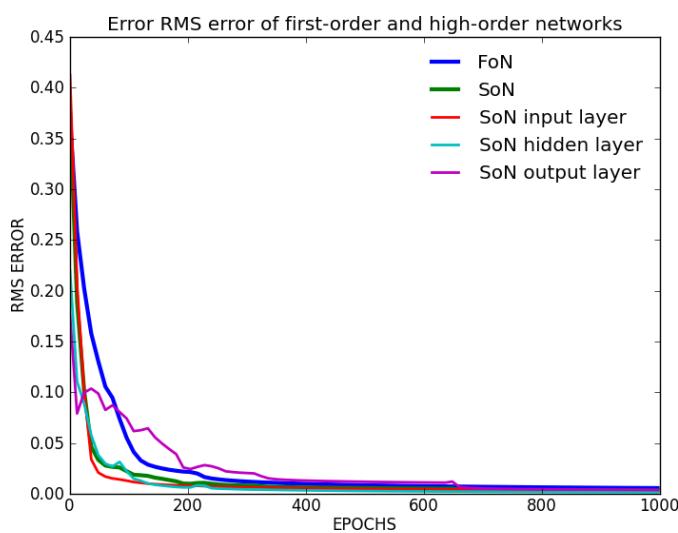
Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer
- l’erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n’est pas possible (ce n’est pas de la classification, mais une duplication), il y a donc un seuil d’erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- le premier réseau réussit à apprendre sa tâche de classification
- le second réseau réussit à apprendre sa tâche de duplication
- la couche cachée et la couche de sortie ne posent aucun problèmes d’apprentissage
- les performances du second réseau dépendent principalement de sa capacité à reproduire les entrées
- le second réseau apprend plus rapidement que le premier
- l’augmentation du nombre de neurone ne change pas les tendances des courbes

Secondaires RMS indépendant par couche



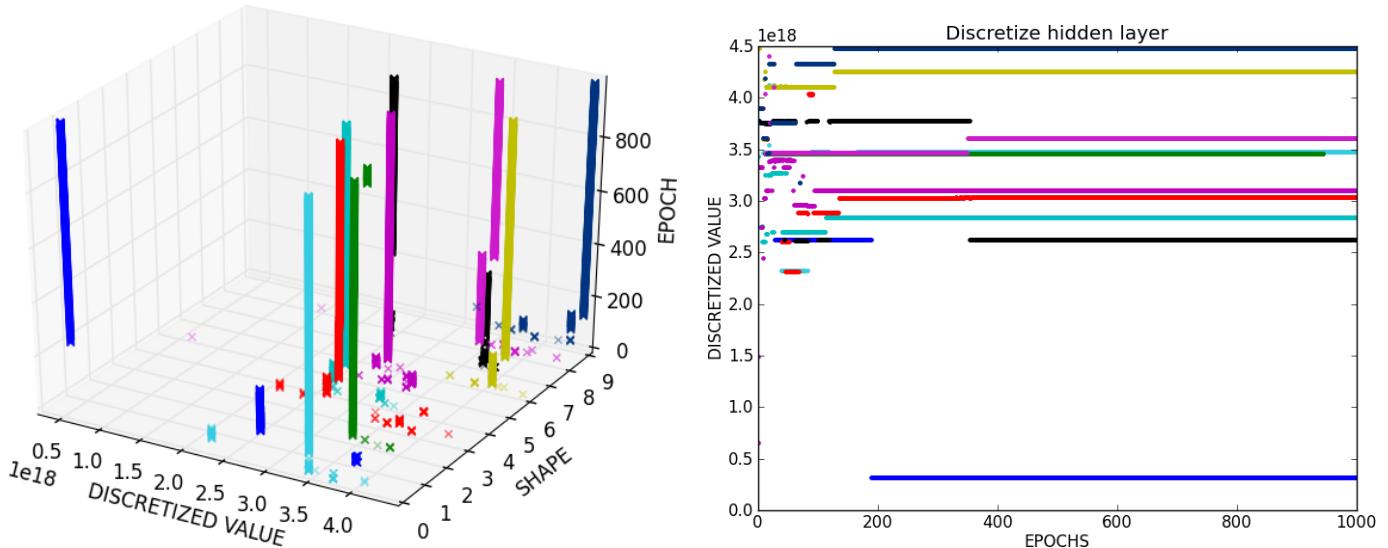
Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- les erreurs de SoN sont calculées en divisant par le nombre de neurone de la couche à reproduire (et non par le nombre total de neurones)

Conclusion

- le SoN a plus de mal à apprendre la couche de sortie contrairement à l'Expérience A1

Secondaires Discrétisation de la couche cachée du premier réseau



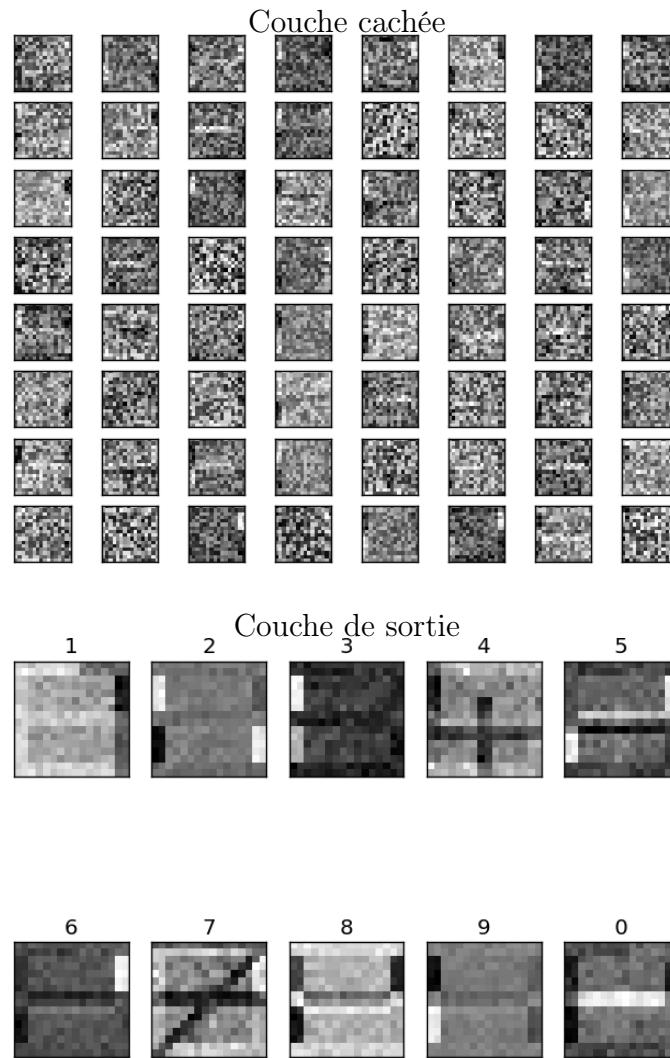
Notes

- une couleur équivaut à un chiffre présenté
- une valeur discrétisée correspond à un certain encodage de la couche cachée (cf Algorithmes Discrétisation)

Conclusion On peut voir ici, qu'un simple perceptron permet de résoudre le problème ; puisqu'à la fin, aucunes des couleurs n'est superposées à une autre.

Les neurones se stabilisent très rapidement (autour de la 50^{ième} époque en moyenne), le tout permettant au second réseau d'avoir des entrées très peu variables, favorisant son apprentissage.

Secondaires Représentations au travers des poids du premier réseau

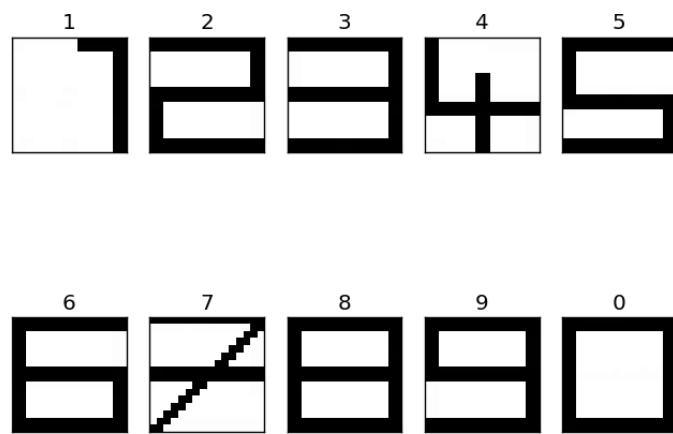


Notes

- plus une case est noire, plus sa présence est importante pour le chiffre en question
- plus une case est blanche, plus son absence est importante
- c'est seulement lorsqu'elle est grise qu'elle n'est pas prise en compte

Conclusion Il est intéressant de remarquer qu'au vu du peu d'entrée, le réseau cible des neurones très précis. Pour le chiffre 2, par exemple, seul 3 lignes sont ciblées.

Secondaires Prototypes à l'intérieur des neurones du second réseau qui essaient de redonner les entrées



Notes

- Il s'agit de la moyenne des réponses du second réseaux sur toutes les entrées

Conclusion Le peu d'entrées permet l'apprentissage par-coeur de chaque forme. À tel point qu'on ne peut pas vraiment parler de prototype.

Conclusion

L'augmentation du nombre de neurones dans le réseau, et l'augmentation de la taille des entrées ne modifient pas l'attitude du réseau.

Au vu des résultats secondaires qui nous montre que la tâche est vraiment facile, dans l'Expérience A3, nous nous sommes intéressé à la même architecture mais sur des données réelles.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number \text{ of neurons on the output layer} \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Discrétisation Pour la couche cachée *hiddenNeuron* de n neurones, un neurone pouvant être encodé par *number_cutting* valeurs différentes :

$$\sum_{i=0}^n number_cutting^i \times cutting(hiddenNeuron[i])$$

Exemple $400 \leftarrow [0 ; 0,25] [0 ; 0,25] [0,25 ; 0,5] [0,5 ; 0,75] [0,25 ; 0,5]$
 $400 \leftarrow 0 \times 4^0 + 0 \times 4^1 + 1 \times 4^2 + 2 \times 4^3 + 1 \times 4^4$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction \text{ sigmoide} \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionisme.

Expérience A3

Objectif

Comprendre de quelles manières peuvent émerger des représentations et métaréprésentations dans un réseau de neurone connexionniste, plus particulièrement sur des perceptrons multicouches.

Réalisation de la première expérience de l'article Cleeremans Alex (2007) sur des données réelles.

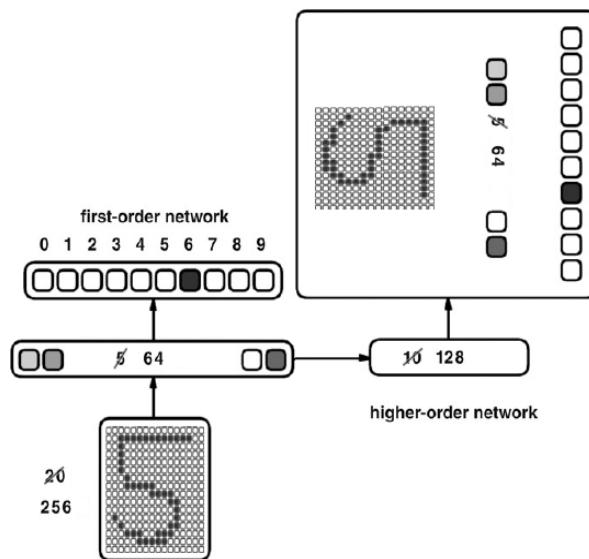
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 64 neurones.

Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

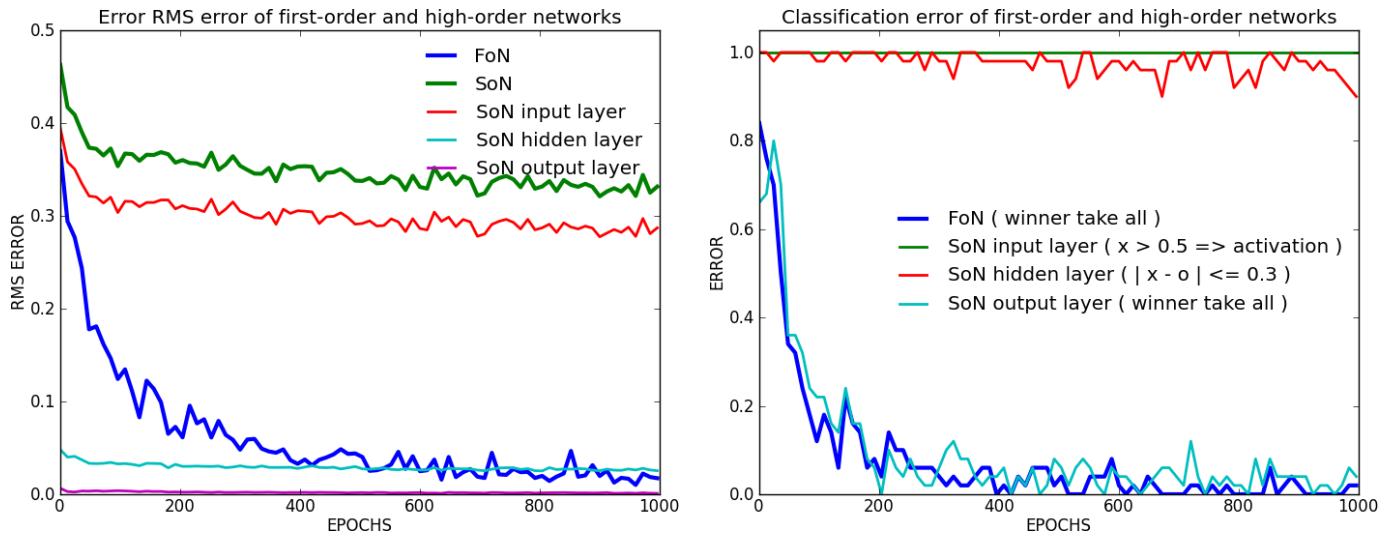


Paramètres

- momentum : 0.9 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 10 (formes) x 1000 (époques)
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais

Résultats

Principaux Analyse des performances



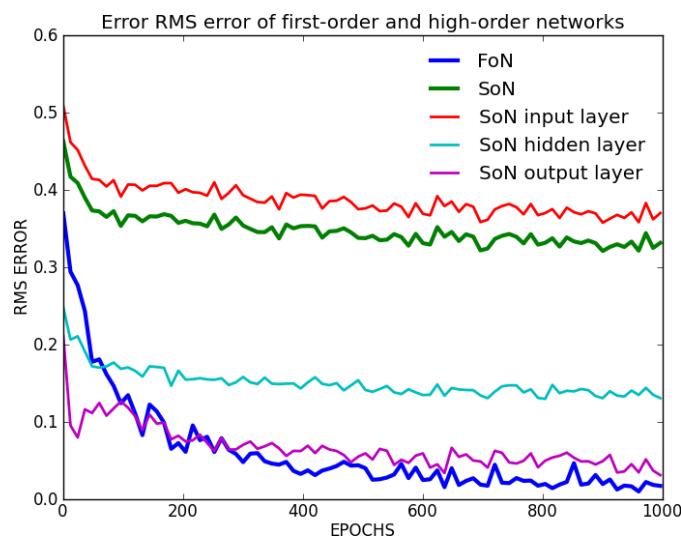
Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer
- l'erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n'est pas possible (ce n'est pas de la classification, mais une duplication), il y a donc un seuil d'erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- le premier réseau réussit à apprendre sa tâche de classification
- le second réseau **ne réussit pas** à apprendre sa tâche de duplication (seul la couche cachée et de sortie sont appris)
- la couche cachée et la couche de sortie posent peu de problèmes d'apprentissage
- le second réseau a du mal à reproduire les entrées
- le second réseau apprend maintenant (vs Expérience A1, Expérience A2) moins rapidement que le premier

Secondaires RMS indépendant par couche



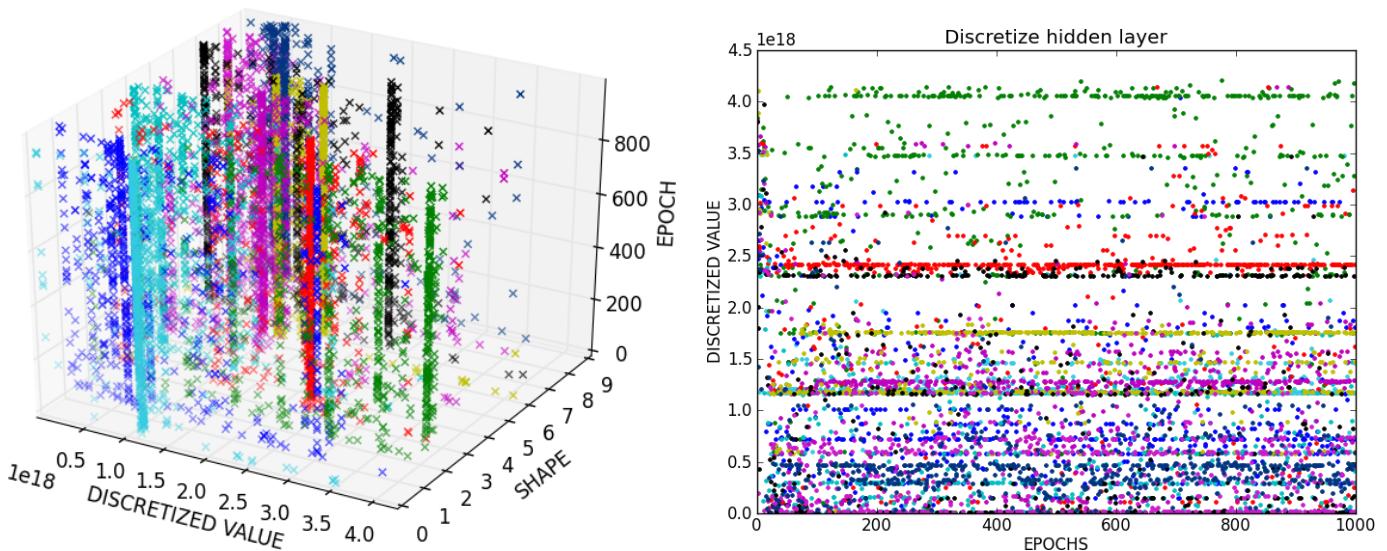
Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseau) sur les couches à reproduire
- les erreurs de SoN sont calculées en divisant par le nombre de neurone de la couche à reproduire (et non par le nombre total de neurones)

Conclusion

- le SoN a plus de mal à apprendre la couche cachée que celle de sortie

Secondaires Discrétisation de la couche cachée du premier réseau



Notes

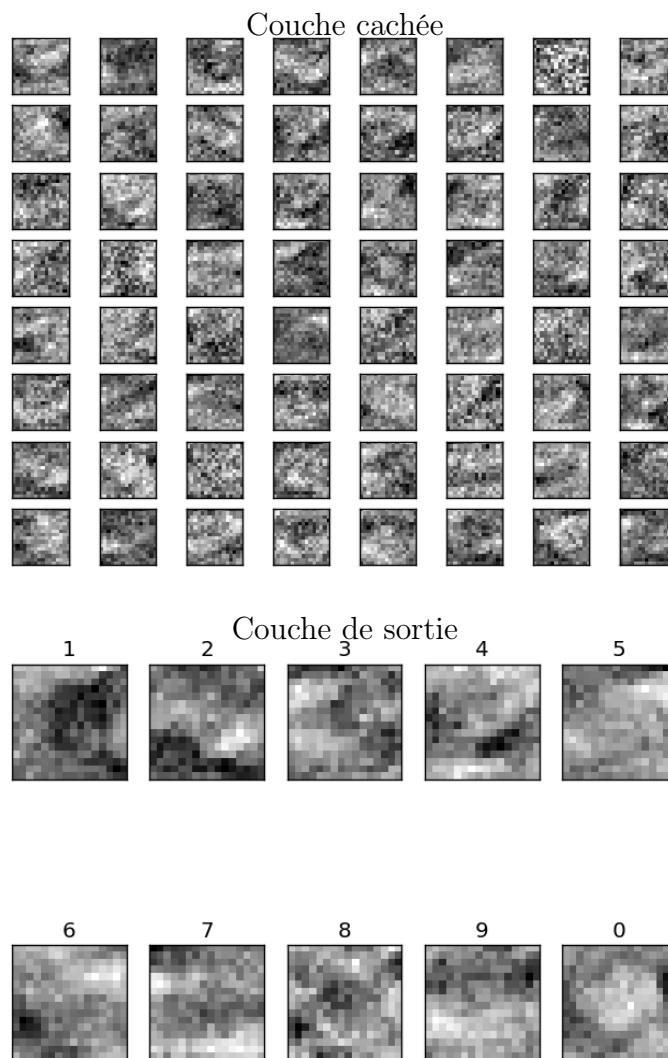
- une couleur équivaut à un chiffre présenté
- une valeur discrétisée correspond à un certain encodage de la couche cachée (cf Algorithmes Discrétisation)

Conclusion Nous pouvons maintenant expliquer les raisons provoquant le mauvais apprentissage des entrées par le second réseau :

Pour un seul chiffre présenté, il existe différentes valeurs de la couche cachée le représentant (factorisation dans la couche cachée du premier réseau). Le second réseau ne peut donc pas apprendre toutes les entrées.

Une même valeur discrétisée peut correspondre à plusieurs couleurs (différents nombres) ce qui augmentent la difficulté d'apprentissage.

Secondaires Représentations au travers des poids du premier réseau

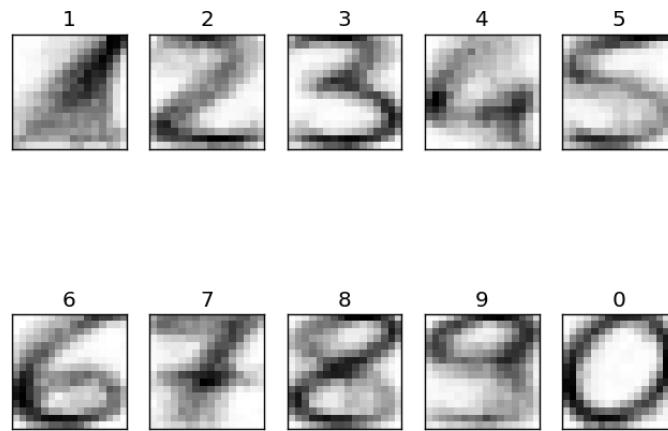


Notes

- plus une case est noire, plus sa présence est importante pour le chiffre en question
- plus une case est blanche, plus son absence est importante
- c'est seulement lorsqu'elle est grise qu'elle n'est pas prise en compte

Conclusion On arrive, sans trop de mal, à distinguer les chiffres.

Secondaires Prototypes à l'intérieur des neurones du second réseau qui essaient de redonner les entrées



Notes

- Il s'agit de la moyenne des réponses du second réseaux sur toutes les entrées

Conclusion Il est intéressant de constater que malgré la difficulté de la tâche, ces prototypes émergent.

Et que contrairement à ce qu'indique les performances, les chiffres sont bien distincts.

On peut, par ailleurs, conjecturer que plus la couche cachée du premier réseau est petite, plus les prototypes seront factorisés (ie. personnels au réseau).

Conclusion

Comme il l'est dit dans Cleeremans Alex (2007), cette architecture tente de résoudre les problèmes des réseaux connexionnistes classiques à avoir un semblant de conscience.

À savoir :

- qu'ils ne savent pas qu'ils peuvent se trouver dans différents états, et qu'ils ne traitent pas leurs propres états : d'où la présence de ce second réseau qui tente de traiter ses états et d'apprendre qu'il en a plusieurs
- des représentations restant bloquées dans la chaîne de causalité de la tâche à apprendre : d'où le fait que ce second réseau n'affecte pas l'apprentissage du premier (ie. pour ne pas retomber dans la chaîne de causalité)

Lors de l'Expérience A4, nous avons confronter la même architecture à des données non linéairement séparables.

Dans l'Expérience B2, nous avons expérimenté les représentations et le transfert de tâche.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number \text{ of neurons on the output layer} \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Discrétisation Pour la couche cachée *hiddenNeuron* de n neurones, un neurone pouvant être encodé par *number_cutting* valeurs différentes :

$$\sum_{i=0}^n number_cutting^i \times cutting(hiddenNeuron[i])$$

Exemple $400 \leftarrow [0 ; 0,25] [0 ; 0,25] [0,25 ; 0,5] [0,5 ; 0,75] [0,25 ; 0,5]$
 $400 \leftarrow 0 \times 4^0 + 0 \times 4^1 + 1 \times 4^2 + 2 \times 4^3 + 1 \times 4^4$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction \text{ sigmoide} \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience A4

Objectif

Comprendre de quelles manières peuvent émerger des représentations et métaréprésentations dans un réseau de neurone connexionniste, plus particulièrement sur des perceptrons multicouches.

Réalisation de la première expérience de l'article Cleeremans Alex (2007) sur des données réelles et non linéairement séparables.

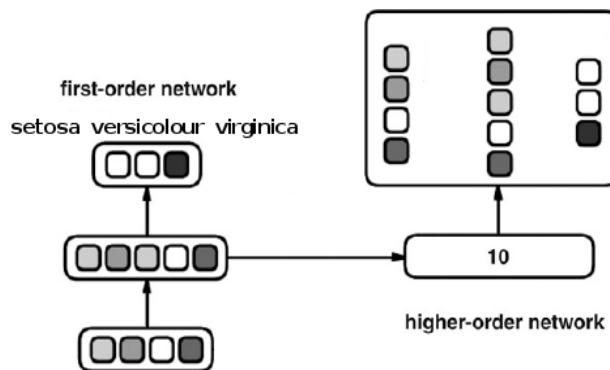
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des fleurs caractérisées par 4 neurones d'entrées qui représentent taille et largeur de la pétalement et la sépale. Il est composé d'une couche cachée de 5 neurones.

Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

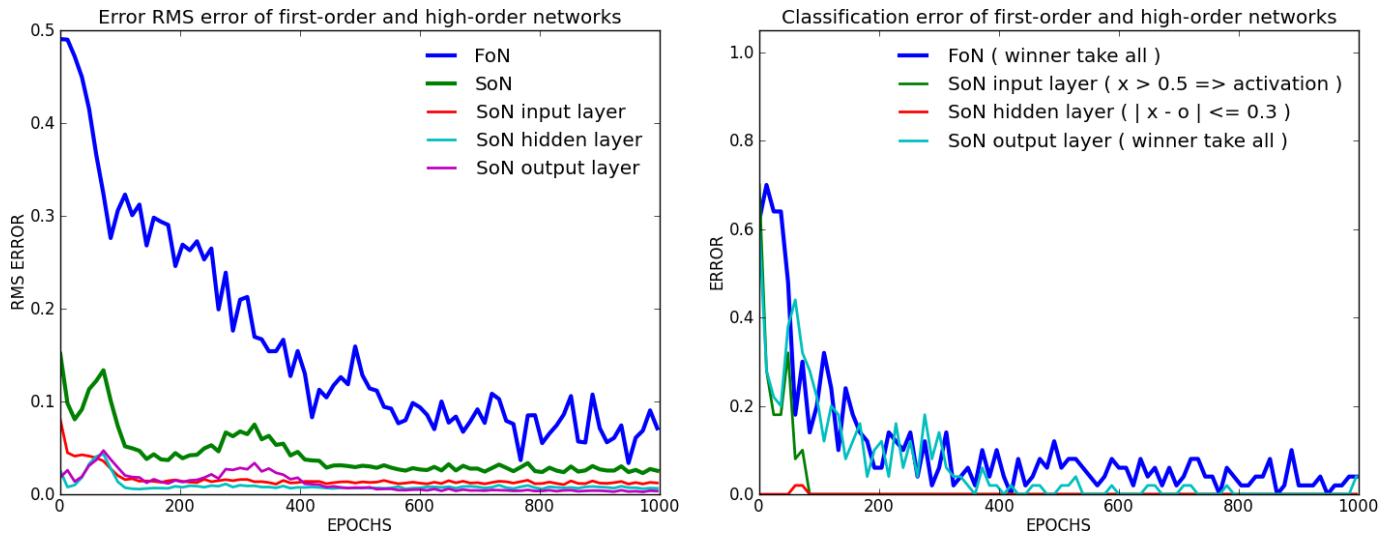


Paramètres

- momentum : 0.9 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- 150 fleurs différentes présentées Fisher (1988)
- apprentissage 10 (formes) x 1000 (époques)
- utilisation de biais
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées réelles sur [0 ; 1]
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



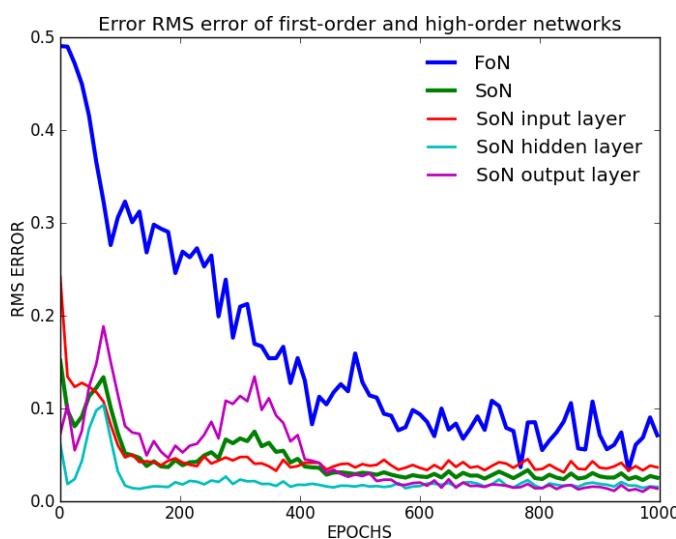
Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer
- l'erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n'est pas possible (ce n'est pas de la classification, mais une duplication), il y a donc un seuil d'erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- le premier réseau réussit à apprendre sa tâche de classification
- le second réseau réussit à apprendre sa tâche de duplication
- la couche cachée et la couche de sortie posent peu de problèmes d'apprentissage
- les performances du second réseau dépendent principalement de sa capacité à reproduire les entrées
- le second réseau apprend plus rapidement que le premier

Secondaires RMS indépendant par couche

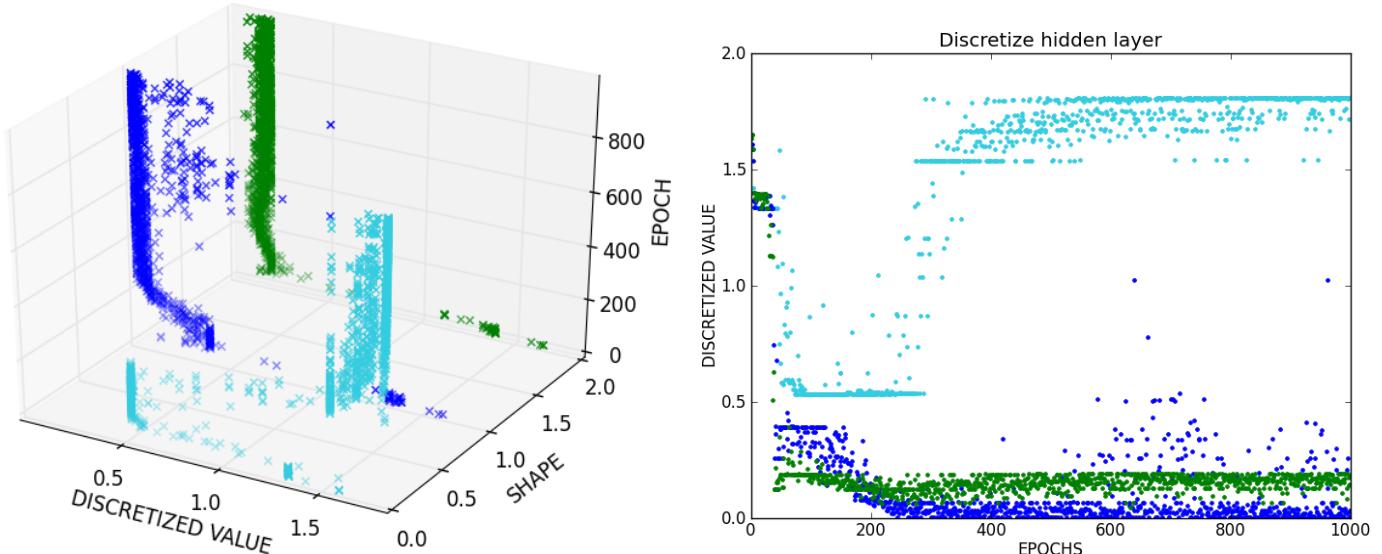


Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseau) sur les couches à reproduire
- les erreurs de SoN sont calculées en divisant par le nombre de neurone de la couche à reproduire (et non par le nombre total de neurones)

Conclusion

Secondaires Discrétisation de la couche cachée du premier réseau



Notes

- une couleur équivaut à une fleur présentée
- une valeur discrétisée correspond à un certain encodage de la couche cachée (cf Algorithmes Discrétisation)

Conclusion Les données n'étant pas linéairement séparables, leur passage dans une seule couche rend la tâche d'apprentissage du second réseau plus complexe.

On peut voir plusieurs points de différentes couleurs aux mêmes endroits (ie. pour la même couche cachée).

Conclusion

Le passage sur des données non linéairement séparables se passent bien.

Comme il l'est dit dans Cleeremans Alex (2007), cette architecture tente de résoudre les problèmes des réseaux connexionnistes classiques à avoir un semblant de conscience.

À savoir :

- qu'ils ne savent pas qu'ils peuvent se trouver dans différents états, et qu'ils ne traitent pas leurs propres états : d'où la présence de ce second réseau qui tente de traiter ses états et d'apprendre qu'il en a plusieurs
- des représentations restant bloquées dans la chaîne de causalité de la tâche à apprendre : d'où le fait que ce second réseau n'affecte pas l'apprentissage du premier (ie. pour ne pas retomber dans la chaîne de causalité)

Dans l'Expérience B3, nous avons expérimenté les représentations et le transfert de tâche.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number \text{ of neurons on the output layer} \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Discrétisation Pour la couche cachée *hiddenNeuron* de n neurones, un neurone pouvant être encodé par *number_cutting* valeurs différentes :

$$\sum_{i=0}^n number_cutting^i \times cutting(hiddenNeuron[i])$$

Exemple $400 \leftarrow [0 ; 0, 25] [0 ; 0, 25] [0, 25 ; 0, 5] [0, 5 ; 0, 75] [0, 25 ; 0, 5]$
 $400 \leftarrow 0 \times 4^0 + 0 \times 4^1 + 1 \times 4^2 + 2 \times 4^3 + 1 \times 4^4$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction \text{ sigmoide} \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

- Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.
- Fisher, R. (1988). Iris plants database. 150 (50 in each of three classes).
- Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience B1

Objectif

Comprendre de quelles manières peuvent émerger des représentations et métaréprésentations dans un réseau de neurone connexionniste, plus particulièrement sur des perceptrons multicouches.

À partir de la première expérience de l'article Cleeremans Alex (2007), expérimenter les représentations.

Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 20 neurones d'entrées. Il est composé d'une couche cachée de 5 neurones.

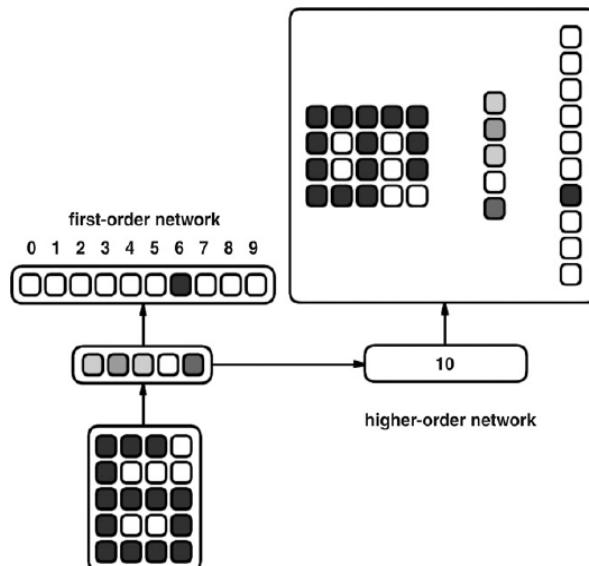
Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Toutes les 1000 époques, l'objectif de la tâche est changé (par une autre combinaison).

Nous avons déroulé l'expérience en 2 fois, une normal, et une fois où on fige les poids entre les entrées et la couche cachée du premier réseau après les 1000 premières époques.

Schéma

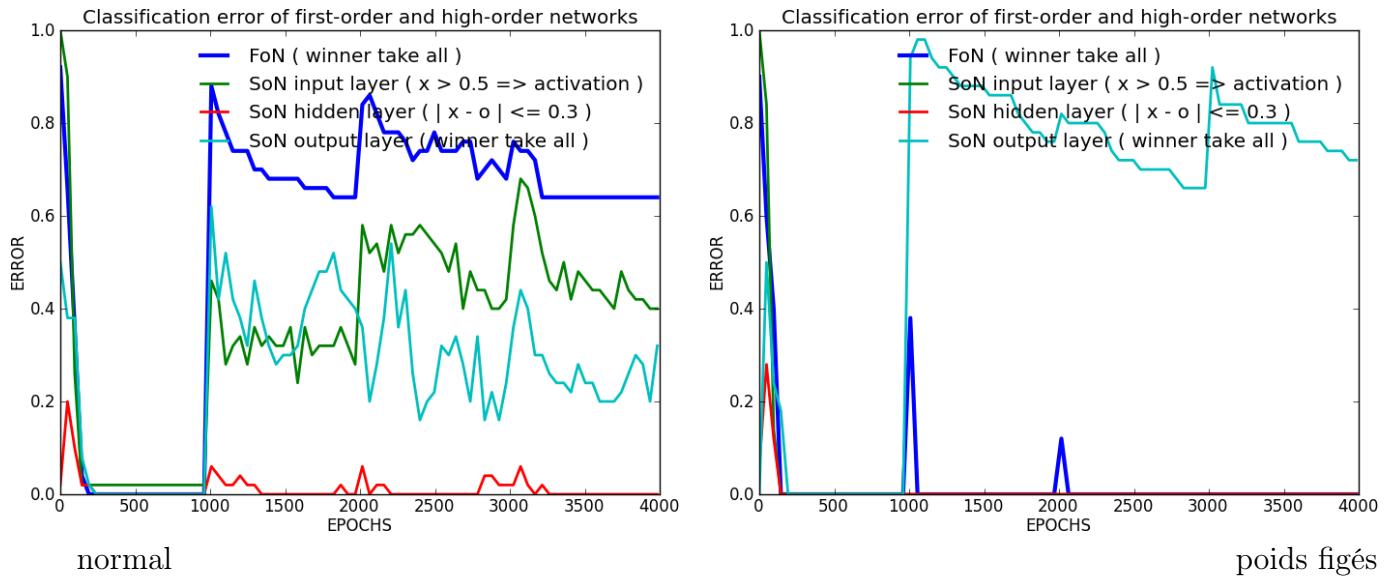


Paramètres

- momentum : 0.9 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- 10 chiffres différents présentés
- apprentissage 10 (formes) x 4000 (époques)
- utilisation de biais
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



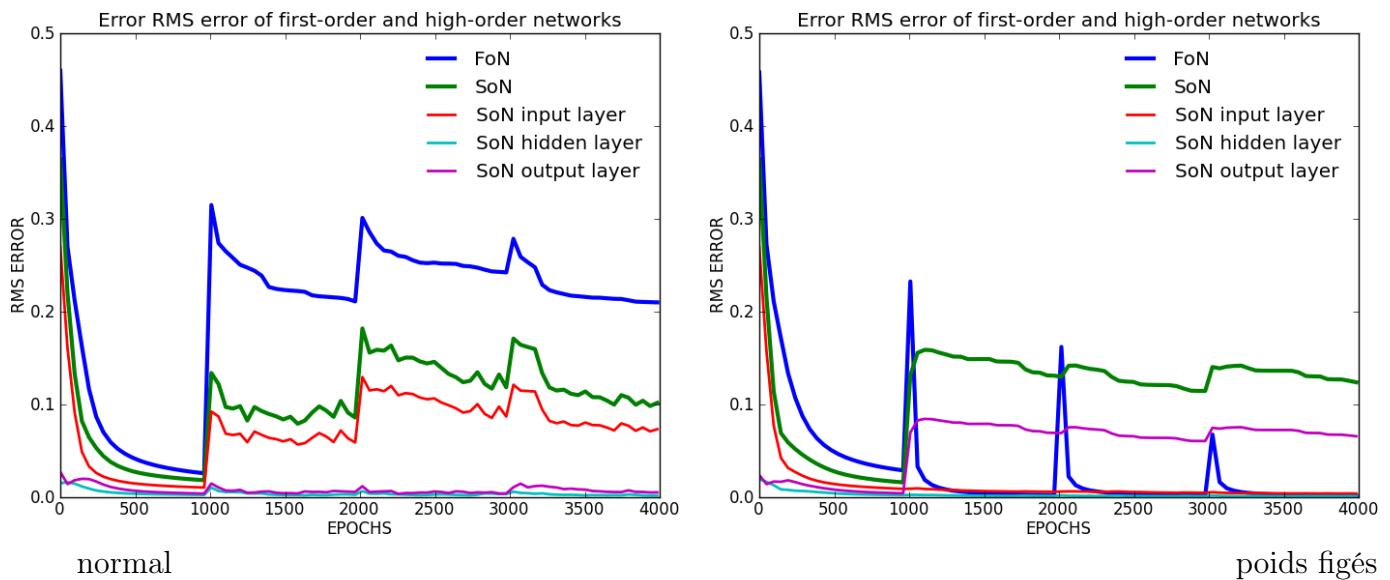
Notes

- l'erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n'est pas possible (ce n'est pas de la classification, mais une duplication), il y a donc un seuil d'erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- lorsque l'on fige les poids, le premier réseau arrive à se réadapter aux nouvelles tâches, mais s'ils ne sont pas figés, il n'y arrive pas
- la reproduction de la couche cachée par le second réseau ne pose pas de problème
- lorsque l'on ne fige pas les poids, le second réseau est lui aussi instable
- lorsque les poids sont figés, la couche de sortie a du mal à être totalement réapprise par le second réseau

Secondaires Analyse des performances RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer

Conclusion Ces courbes sont a peu près équivalentes à celles de classifications.

Conclusion

Le blocage des poids dans la couche cachée prouve la présence de représentations stables permettant le transfert de tâche.

Ces représentations contiennent ce qui caractérise la forme sans encore lui mettre un sens dessus. Il sait à quoi ressemble un 1 (dans la couche cachée) et lui met un donne le sens 1 (dans la couche de sortie). Il est donc facile de lui dire que finalement, cette représentation correspond à un 2, sans qu'il ait à réapprendre la forme entièrement.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience B2

Objectif

Comprendre de quelles manières peuvent émerger des représentations et métaréprésentations dans un réseau de neurone connexionniste, plus particulièrement sur des perceptrons multicouches.

À partir de la première expérience de l'article Cleeremans Alex (2007), expérimenter les représentations sur des données réelles.

Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par (16x16) neurones d'entrées. Il est composé d'une couche cachée de 64 neurones.

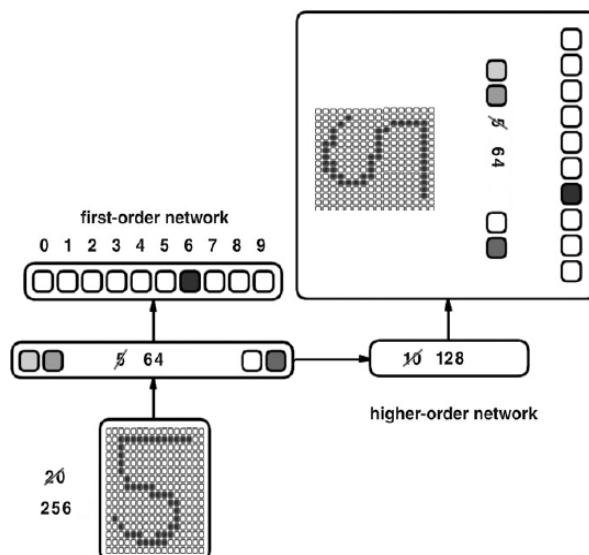
Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Toutes les 1000 époques, l'objectif de la tâche est changé (par une autre combinaison).

Nous avons déroulé l'expérience en 2 fois, une normal, et une fois où on fige les poids entre les entrées et la couche cachée du premier réseau après les 1000 premières époques.

Schéma

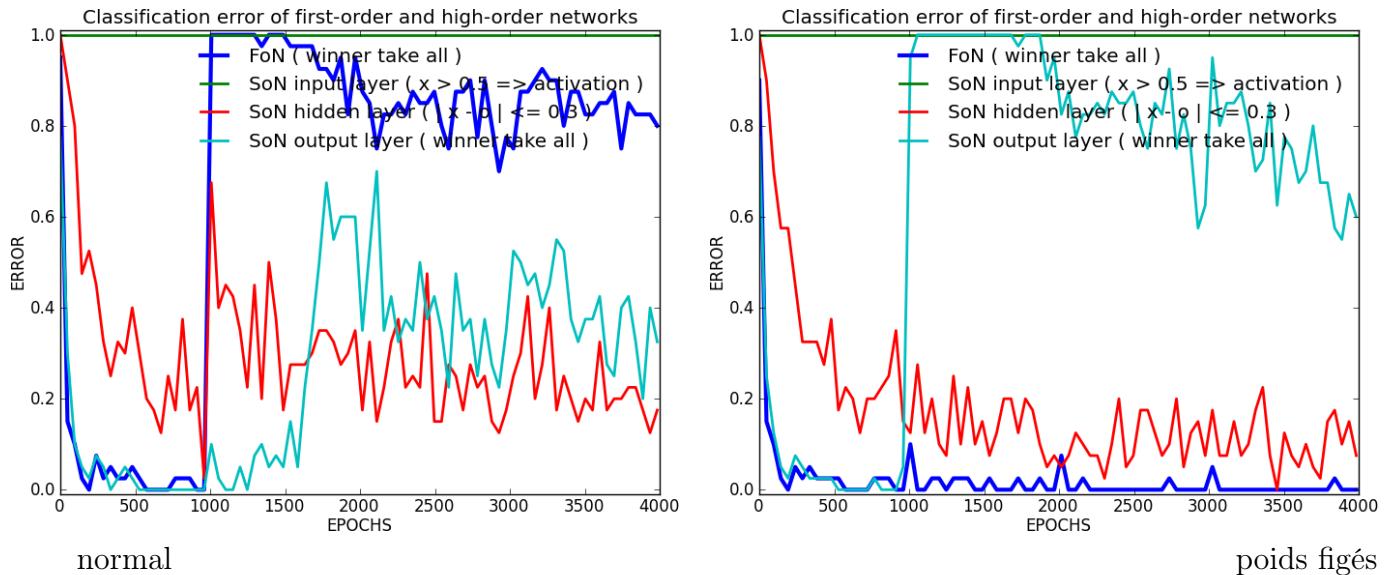


Paramètres

- momentum : 0.5 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 10 (formes) x 4000 (époques)
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais

Résultats

Principaux Analyse des performances



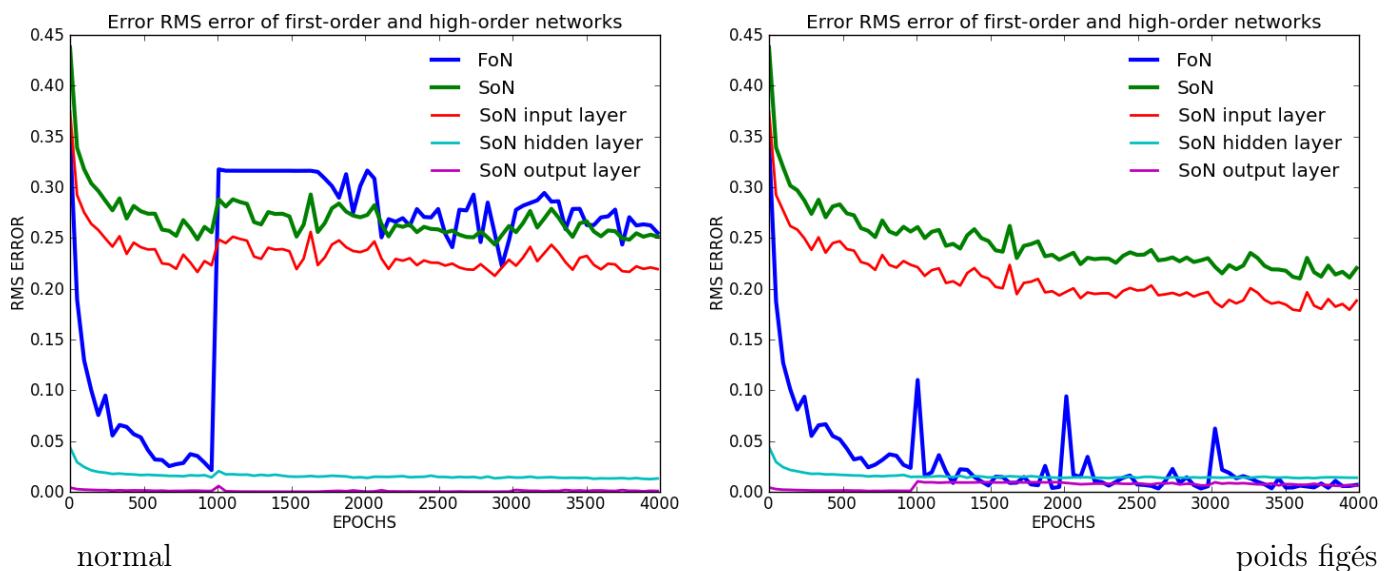
Notes

- l'erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n'est pas possible (ce n'est pas de la classification, mais une duplication), il y a donc un seuil d'erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- lorsque l'on fige les poids, le premier réseau arrive à se réadapter aux nouvelles tâches, mais s'ils ne sont pas figés, il n'y arrive pas
- la reproduction de la couche cachée par le second réseau ne pose pas de problème
- lorsque l'on ne fige pas les poids, le second réseau est lui aussi instable
- lorsque les poids sont figés, la couche de sortie a du mal à être totalement réapprise par le second réseau

Secondaires Analyse des performances RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer

Conclusion Ces courbes sont a peu près équivalentes à celles de classifications.

Conclusion

Le blocage des poids dans la couche cachée prouve la présence de représentations stables permettant le transfert de tâche.

Ces représentations contiennent ce qui caractérise la forme sans encore lui mettre un sens dessus. Il sait à quoi ressemble un 1 (dans la couche cachée) et lui met un donne le sens 1 (dans la couche de sortie). Il est donc facile de lui dire que finalement, cette représentation correspond à un 2, sans qu'il ait à réapprendre la forme entièrement.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience B3

Objectif

Comprendre de quelles manières peuvent émerger des représentations et métaréprésentations dans un réseau de neurone connexionniste, plus particulièrement sur des perceptrons multicouches.

À partir de la première expérience de l'article Cleeremans Alex (2007), expérimenter les représentations sur des données linéairement séparables.

Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des fleurs caractérisées par 4 neurones d'entrées qui représentent taille et largeur de la pétalement et la sépale.

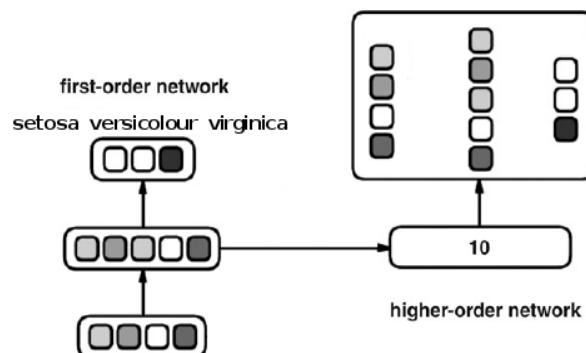
Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Toutes les 1000 époques, l'objectif de la tâche est changé (par une autre combinaison).

Nous avons déroulé l'expérience en 2 fois, une normal, et une fois où on fige les poids entre les entrées et la couche cachée du premier réseau après les 1000 premières époques.

Schéma

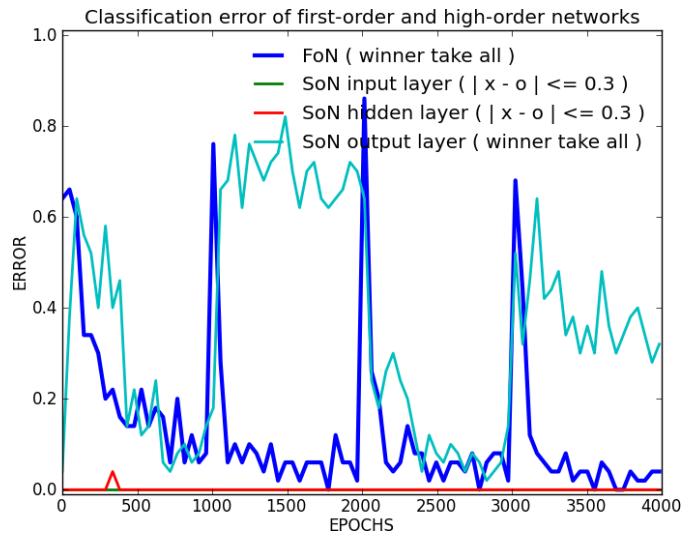
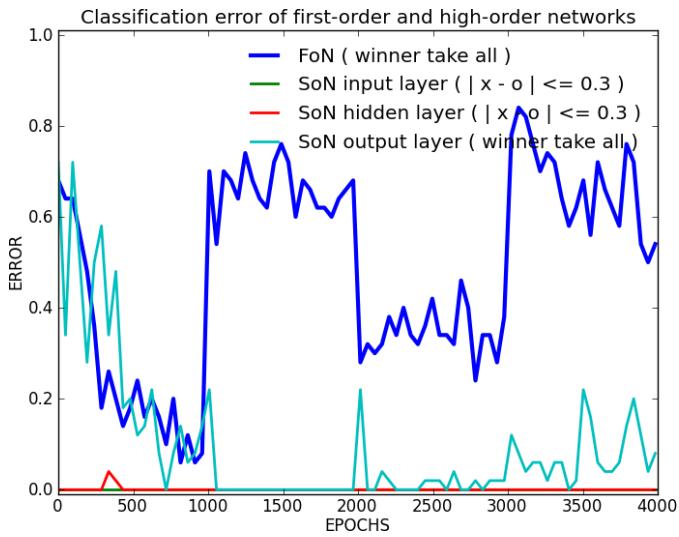


Paramètres

- momentum : 0.9 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- 10 chiffres différents présentés
- apprentissage 10 (formes) x 4000 (époques)
- utilisation de biais
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



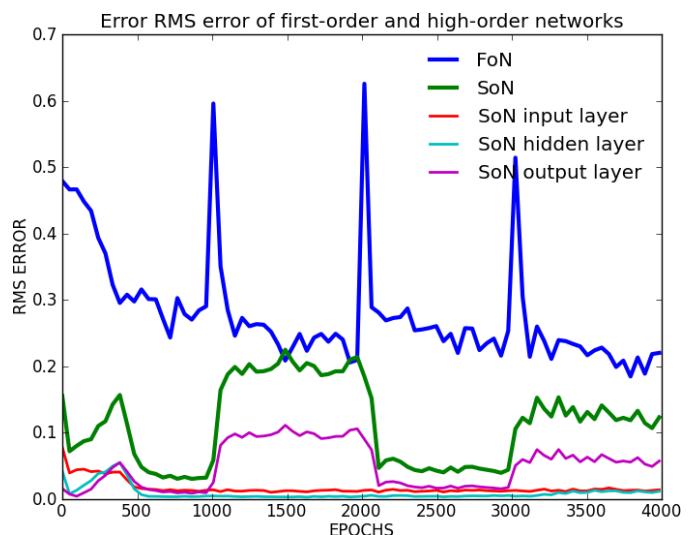
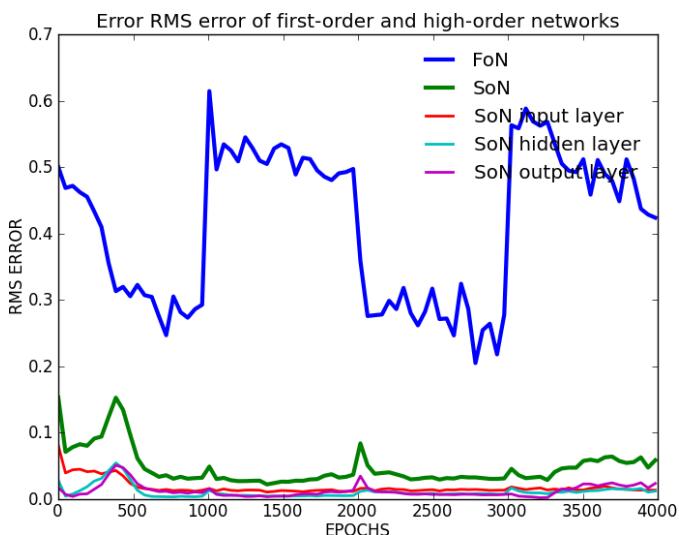
Notes

- l'erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n'est pas possible (ce n'est pas de la classification, mais une duplication), il y a donc un seuil d'erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- lorsque l'on fige les poids, le premier réseau arrive à se réadapter aux nouvelles tâches, mais s'ils ne sont pas figés, il n'y arrive pas
- la reproduction de la couche cachée par le second réseau ne pose pas de problème
- lorsque l'on ne fige pas les poids, le second réseau est lui aussi instable
- lorsque les poids sont figés, la couche de sortie a du mal à être totalement réappris par le second réseau

Secondaires Analyse des performances RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer

Conclusion Ces courbes sont a peu près équivalentes à celles de classifications.

Conclusion

Le blocage des poids dans la couche cachée prouve la présence de représentations stables permettant le transfert de tâche.

Ces représentations contiennent ce qui caractérise la fleur sans encore lui mettre un nom dessus. Il sait à quoi ressemble cette fleur (dans la couche cachée) et lui donne un nom (dans la couche de sortie). Il est donc facile de lui dire que finalement, cette fleur correspond à une autre, sans qu'il ait à la réapprendre entièrement.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience B4

Objectif

Il sagit d'une extension de l'Expérience B2 pour montrer qu'avec des changements de tâche : le second réseau est lui aussi plus enclin à apprendre lorsque ses poids sont figés.

Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par (16x16) neurones d'entrées. Il est composé d'une couche cachée de 64 neurones.

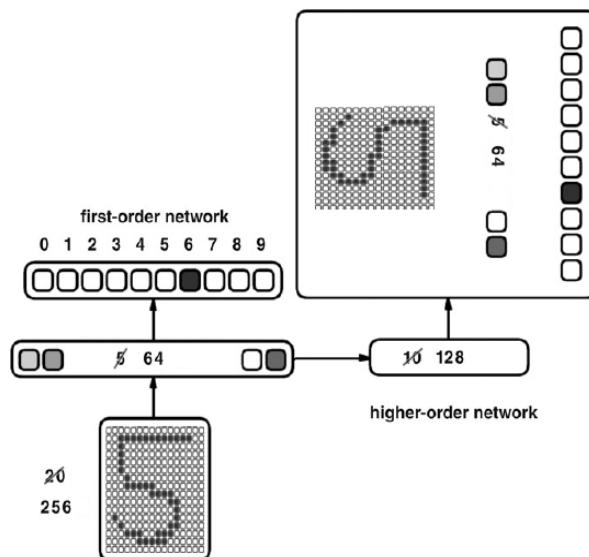
Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Toutes les 1000 époques, l'objectif de la tâche est changé (par une autre combinaison).

Nous avons déroulé l'expérience en 2 fois, une fois où on fige les poids entre les entrées et la couche cachée du premier réseau après les 1000 premières époques, et une seconde où on fige les poids des couches cachées (entrée - couche cachée) des 2 réseaux.

Schéma

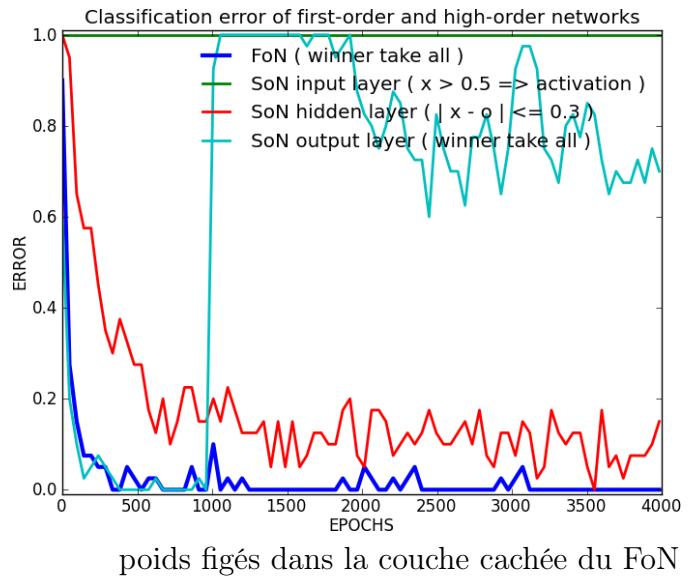
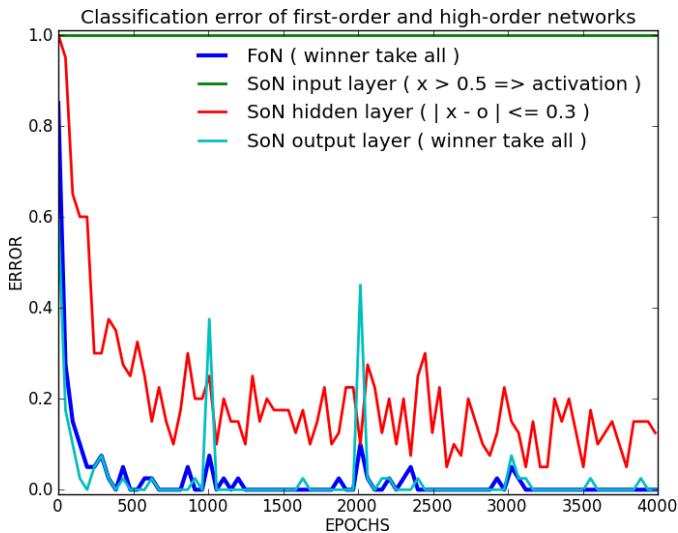


Paramètres

- momentum : 0.5 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 10 (formes) x 4000 (époques)
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais

Résultats

Principaux Analyse des performances



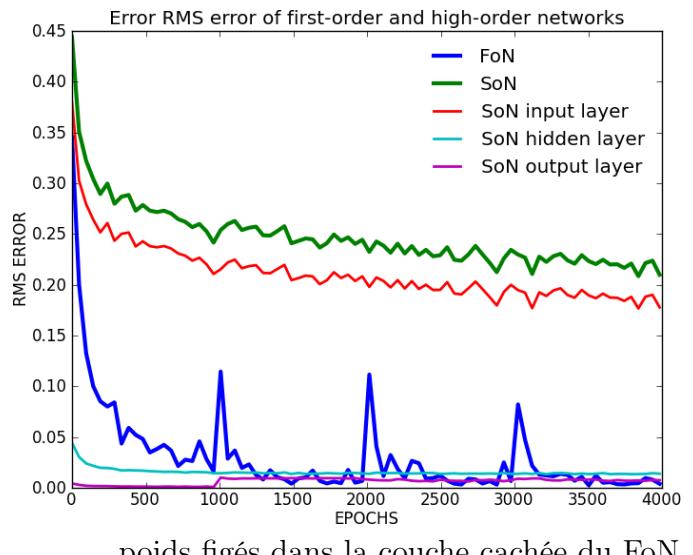
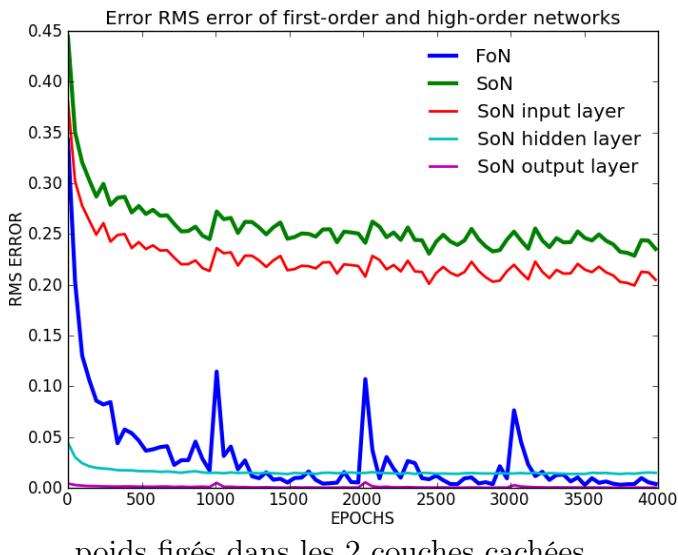
Notes

- l'erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n'est pas possible (ce n'est pas de la classification, mais une duplication), il y a donc un seuil d'erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- lorsque l'on fige les poids, le premier réseau arrive à se réadapter aux nouvelles tâches
- lorsqu'on fige également les poids du SoN, il est capable d'apprendre à dupliquer la sortie du FoN

Secondaires Analyse des performances RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer

Conclusion Ces courbes sont a peu près équivalentes à celles de classifications. Même avec un RMS très faible pour la couche de sortie, dans la courbe de droite, cela suffit au réseau pour se tromper dans l'erreur de classification.

Conclusion

Preuve que le SoN subit le même phénomène (il faut bloquer des poids pour qu'il puisse s'adapter) que le FoN lors du changement de tâche.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience B5

Objectif

Il sagit d'une extension de l'Expérience B2 pour montrer que même un changement de tâche, qui n'est pas simplement une simple combinaison, fonctionne.

Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par (16x16) neurones d'entrées. Il est composé d'une couche cachée de 64 neurones.

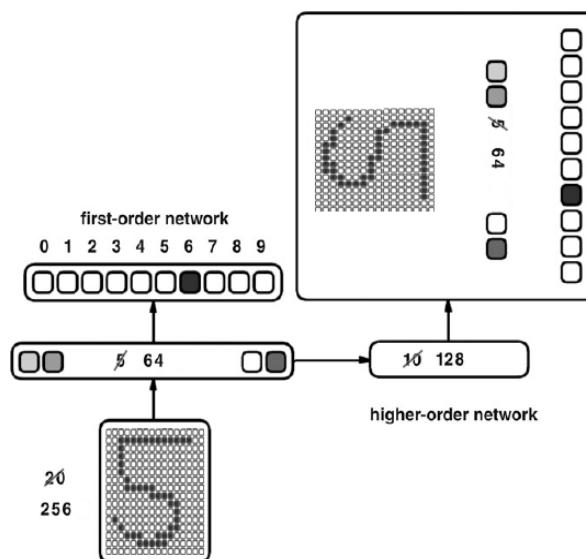
Un second réseau de perceptron multicouche apprend à dupliquer toutes les couches du premier réseau en n'ayant que sa couche cachée en entrée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

À l'époque 1000, l'objectif de la tâche est changé, il doit qualifier les formes par pair/impair (8 neurones sont alors supprimés).

Nous avons déroulé l'expérience en 2 fois, une normal, et une fois où on fige les poids entre les entrées et la couche cachée du premier réseau après le changement de tâche.

Schéma

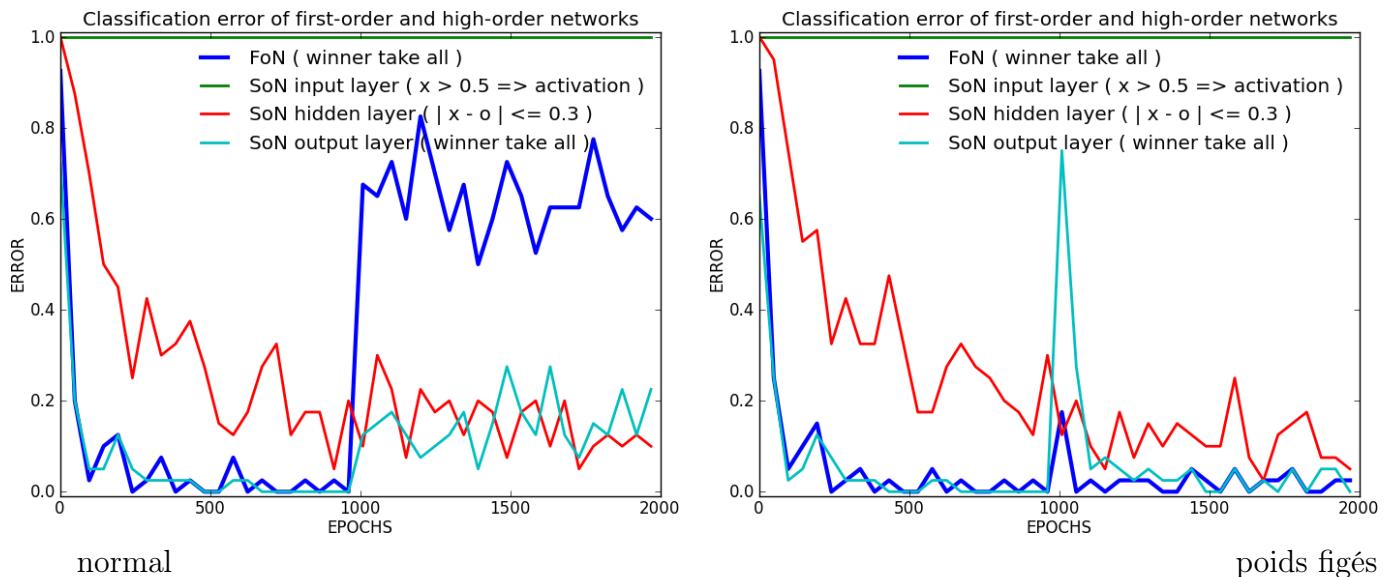


Paramètres

- momentum : 0.5 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 10 (formes) x 2000 (époques)
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais

Résultats

Principaux Analyse des performances



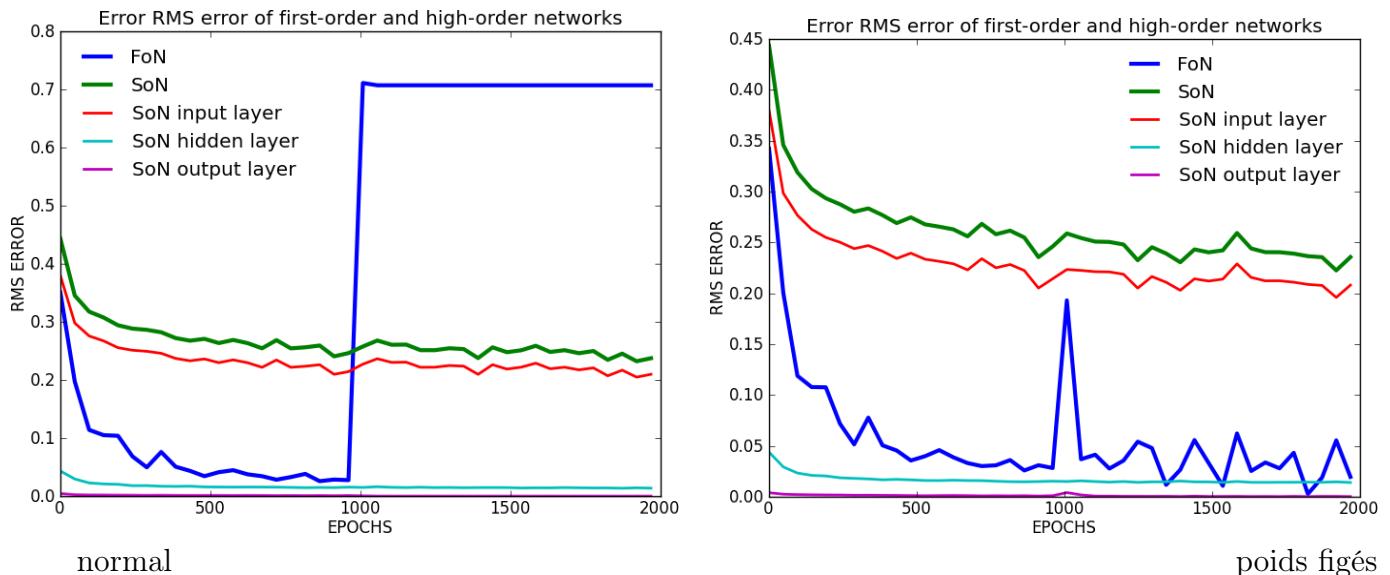
Notes

- l'erreur de classification représente le taux de mauvaises réponses pour les 10 formes présentées sur une époque
- pour SoN input layer et hidden layer, un winner-take-all n'est pas possible (ce n'est pas de la classification, mais une duplication), il y a donc un seuil d'erreur qui ne doit pas être dépassé par un seul neurone

Conclusion

- lorsque l'on fige les poids, le premier réseau arrive à se réadapter à la nouvelle tâche, mais s'ils ne sont pas figés, il n'y arrive pas
- lorsque l'on ne fige pas les poids, le second réseau est lui aussi instable

Secondaires Analyse des performances RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- les courbes SoN layer représentent les erreurs (du second réseaux) sur les couches à reproduire
- la courbe RMS verte (SoN) est la somme des 3 courbes SoN layer

Conclusion Ces courbes sont a peu près équivalentes à celles de classifications.

Conclusion

L'Expérience B2 peut être étendu sur des changements de tâche plus complexe.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience C1

Objectif

Comprendre de quelles manières un réseau de neurone connexioniste peut parier sur ses propres résultats à partir de ses représentations personnelles.

Reproduction et approfondissement des résultats de la seconde expérience de l'article Cleeremans Alex (2007).

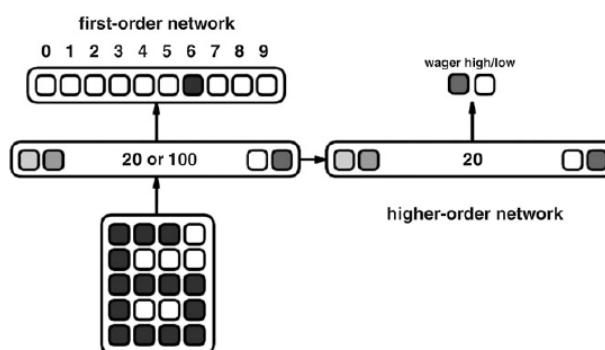
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 20 neurones d'entrées. Il est composé d'une couche cachée de 20 ou 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

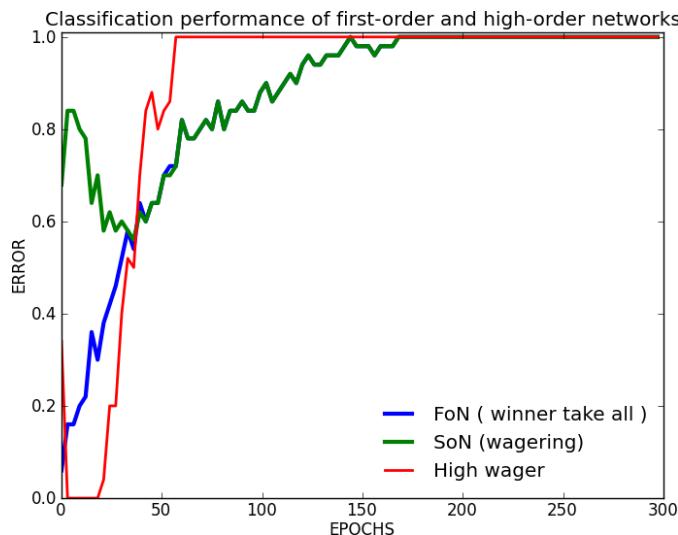


Paramètres

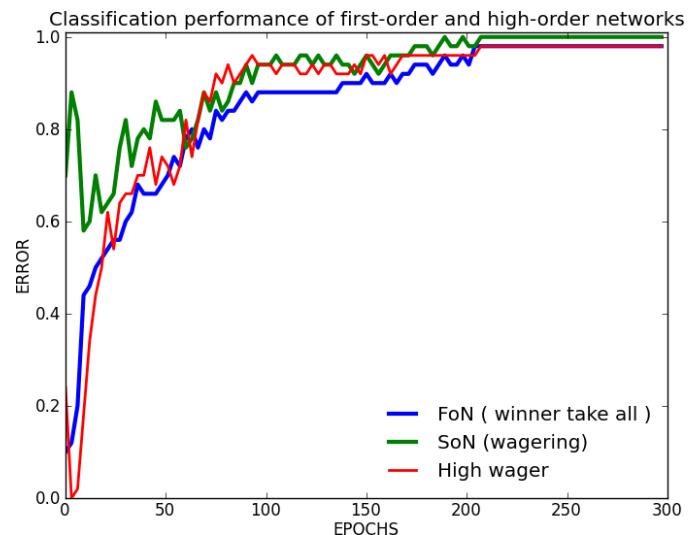
- momentum : 0.5 sur les 2 réseaux
- taux d'apprentissage : 0.15 sur les 2 réseaux
- 10 chiffres différents présentés
- apprentissage 10 (formes) x 300 (époques)
- utilisation de biais
- sigmoïde à température 1
- poids initialisés sur [-1 ; 1] pour les 2 réseaux
- taux d'apprentissage constant
- entrées valent 0 ou 1

Résultats

Principaux Analyse des performances



20 neurones en couche cachée



100 neurones en couche cachée

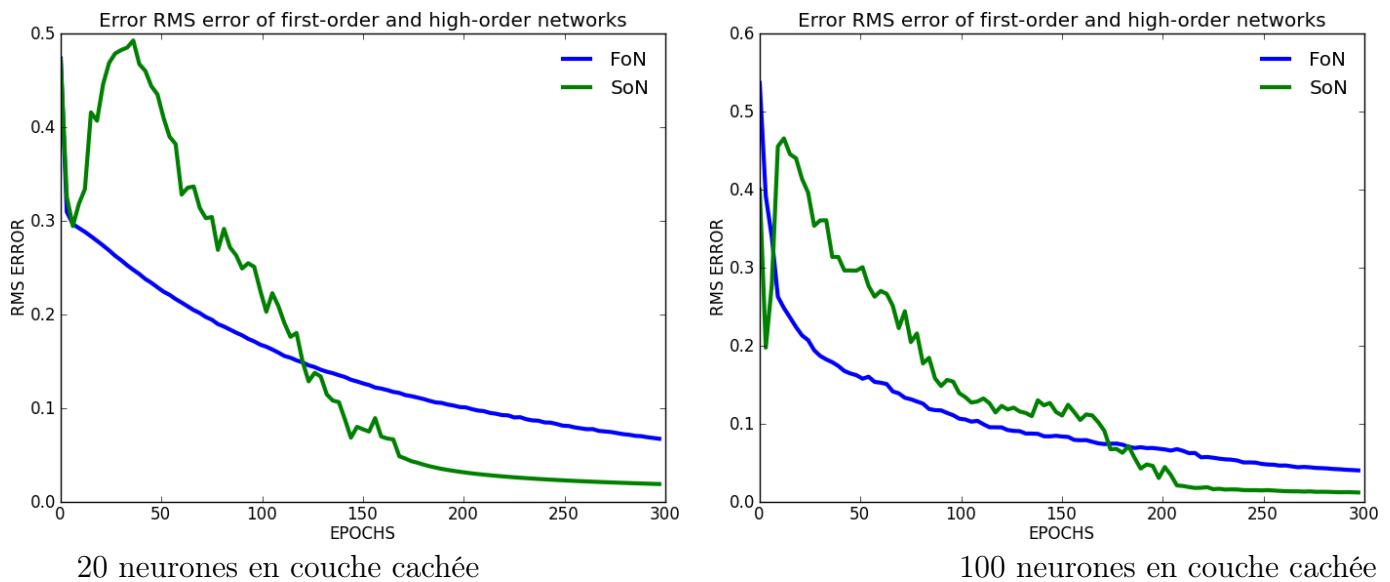
Notes

- la courbe rouge représente le taux de paris hauts du second réseau
- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 10 formes présentées sur une époque

Conclusion

- dans les 2 cas, le premier réseau réussit à apprendre sa tâche de classification
- lorsque le nombre de neurones dans la couche cachée est faible, l'apprentissage de la tâche est optimal, il ne peut plus être amélioré
Ainsi le second réseau se contente de parier haut.
- lorsque le nombre de neurones dans la couche cachée est élevé, l'apprentissage peut être amélioré, le second réseau le remarque et peut accorder son taux de paris hauts avec le taux de succès du premier réseau.
Remarquons tout de même qu'avec l'amélioration du second réseau, on ne peut pas dépasser un réseau optimal, seulement l'égaler

Secondaires RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)

Conclusion On remarque que le second réseau apprend et désapprend. De plus, il reste toujours sous la barre des 0.5.

Conclusion

Comme il l'est dit dans Cleeremans Alex (2007), cette architecture tente de résoudre les problèmes des réseaux connexionnistes classiques à avoir un semblant de conscience.

À savoir :

- qu'ils ne savent pas qu'ils peuvent se trouver dans différents états, et qu'ils ne traitent pas leurs propres états : d'où la présence de ce second réseau qui tente de traiter ses états et d'apprendre qu'il en a plusieurs
- des représentations rentant bloquées dans la chaîne de causalité de la tâche à apprendre : d'où le fait que ce second réseau n'affecte pas l'apprentissage du premier (ie. pour ne pas retomber dans la chaîne de causalité)
- qu'ils n'ont pas les connaissances conscientes des raisons de leurs décisions : on essaye de les y sensibiliser avec les paris

Par ailleurs, cette architecture ouvre des possibilités d'amélioration de l'apprentissage. Elle permet au réseau de détecter lui même un nombre de neurones trop important dans la couche cachée. On pourrait, par exemple, imaginer un réseau autorégulant son nombre de neurone.

L'expérience suivante (Expérience C2) retente l'expérience sur des données réelles.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience C2

Objectif

Comprendre de quelles manières un réseau de neurone connexioniste peut parier sur ses propres résultats à partir de ses représentations personnelles.

Réalisation de la seconde expérience de l'article Cleeremans Alex (2007) sur des données réelles.

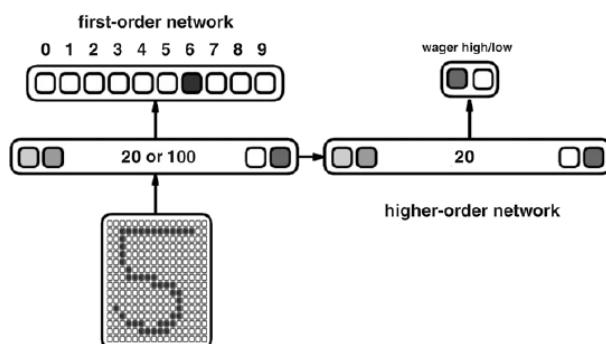
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 20 ou 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

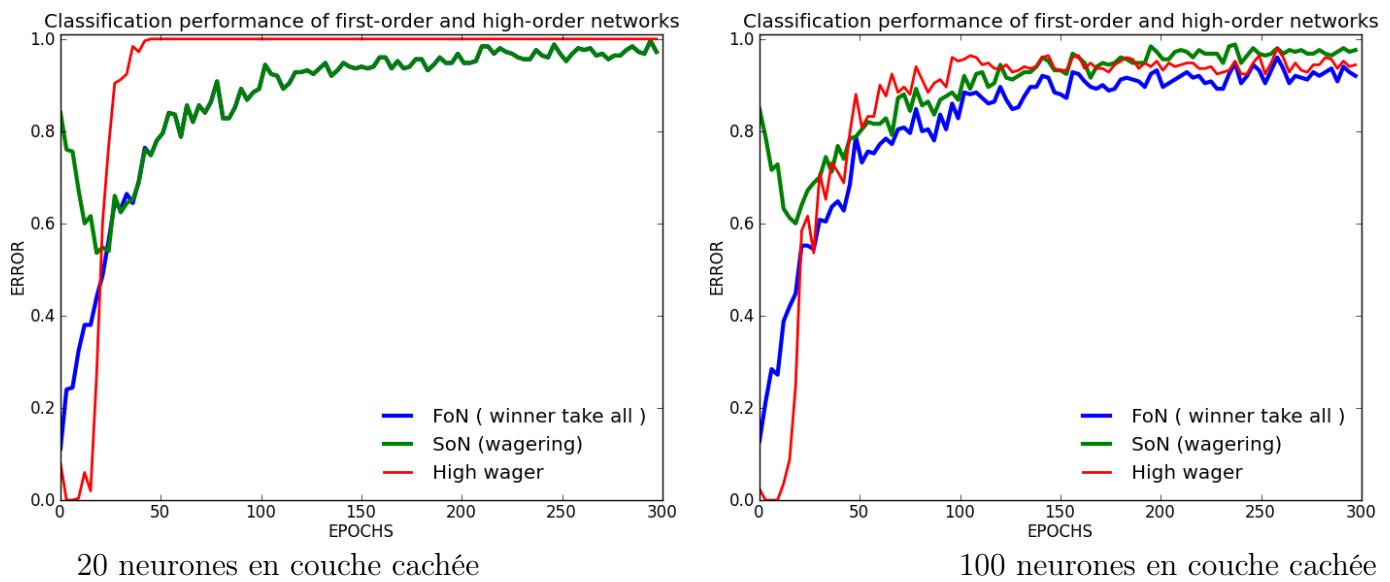


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le second réseau
- taux d'apprentissage : 0.15 sur les 2 réseaux
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 50 (formes) x 300 (époques)
- utilisation de biais
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le second réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



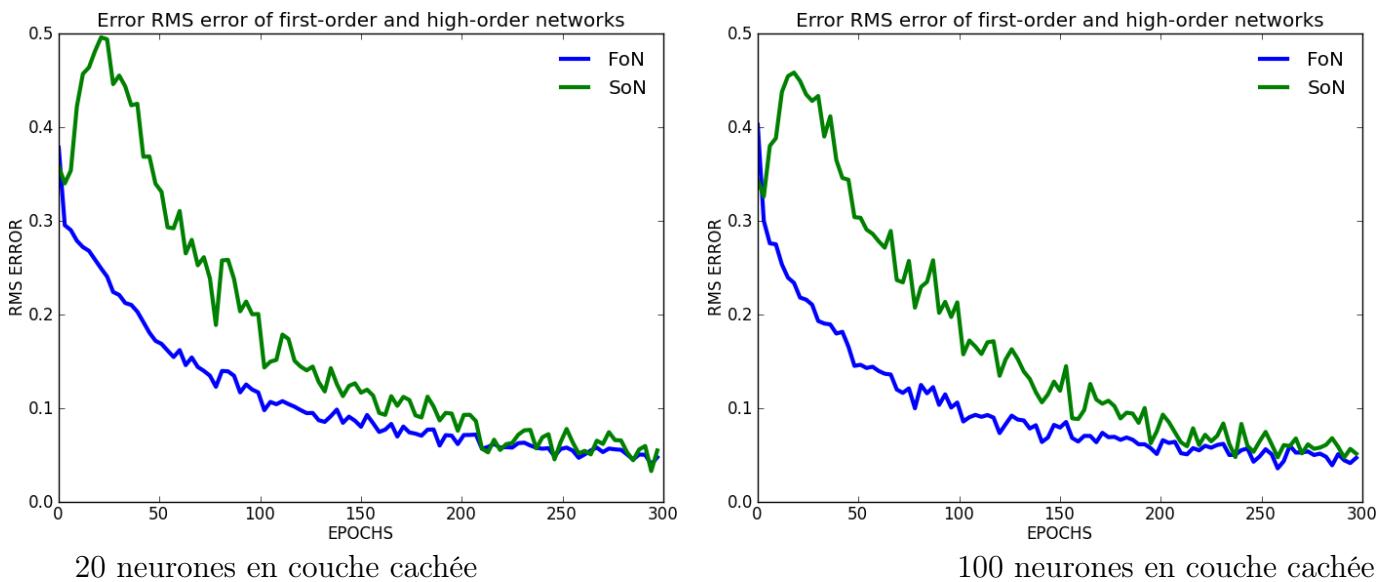
Notes

- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque
- la courbe rouge représente le taux de paris hauts du second réseau

Conclusion

- dans les 2 cas, le premier réseau réussit à apprendre sa tâche de classification
- lorsque le nombre de neurones dans la couche cachée est faible, l'apprentissage de la tâche est optimal, il ne peut plus être amélioré
Ainsi le second réseau se contente de parier haut.
- lorsque le nombre de neurones dans la couche cachée est élevé, l'apprentissage peut être amélioré, le second réseau le remarque et peut accorder son taux de paris hauts avec le taux de succès du premier réseau.
Remarquons tout de même qu'avec l'amélioration du second réseau, on ne peut pas dépasser un réseau optimal, seulement l'égaler

Secondaires RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)

Conclusion On remarque que le second réseau apprend et désapprend. De plus, il reste toujours sous la barre des 0.5.

Conclusion

Comme il l'est dit dans Cleeremans Alex (2007), cette architecture tente de résoudre les problèmes des réseaux connexionnistes classiques à avoir un semblant de conscience.

À savoir :

- qu'ils ne savent pas qu'ils peuvent se trouver dans différents états, et qu'ils ne traitent pas leurs propres états : d'où la présence de ce second réseau qui tente de traiter ses états et d'apprendre qu'il en a plusieurs
- des représentations rentant bloquées dans la chaîne de causalité de la tâche à apprendre : d'où le fait que ce second réseau n'affecte pas l'apprentissage du premier (ie. pour ne pas retomber dans la chaîne de causalité)
- qu'ils n'ont pas les connaissances conscientes des raisons de leurs décisions : on essaye de les y sensibiliser avec les paris

Le passage sur des données réelles ne modifient pas le comportement du réseau.

Par ailleurs, cette architecture ouvre des possibilités d'amélioration de l'apprentissage. Elle permet au réseau de détecter lui-même un nombre de neurones trop important dans la couche cachée. On pourrait, par exemple, imaginer un réseau autorégulant son nombre de neurone.

Toutes les expériences F et G découlent de cette architecture.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience C3

Objectif

Comprendre de quelles manières un réseau de neurone connexioniste peut parier sur ses propres résultats à partir de ses représentations personnelles.

Réalisation de la seconde expérience de l'article Cleeremans Alex (2007) sur des données réelles non linéairement séparables.

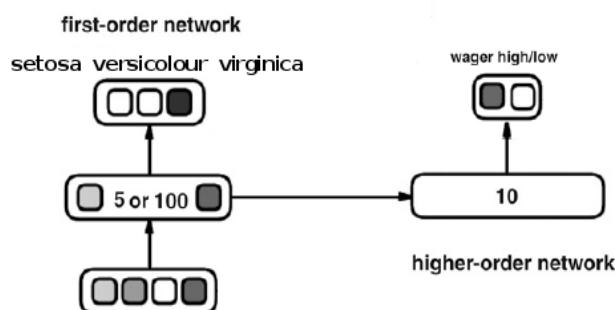
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des fleurs caractérisées par 4 neurones d'entrées qui représentent taille et largeur de la pétale et la sépale. Il est composé d'une couche cachée de 5 ou 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

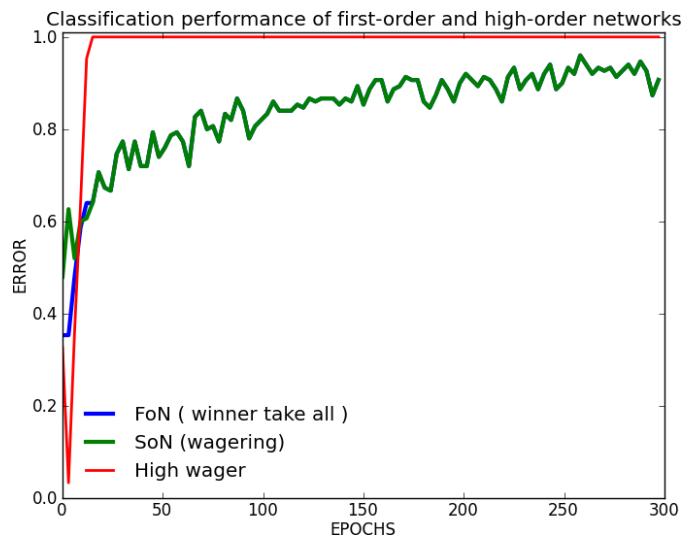


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le second réseau
- taux d'apprentissage : 0.15 sur les 2 réseaux
- **150 formes** de fleurs différents présentées (shuffle) Fisher (1988)
- apprentissage 50 (formes) x 300 (époques)
- utilisation de biais
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le second réseau
- taux d'apprentissage constant
- entrées réelles sur [0 ; 1]
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



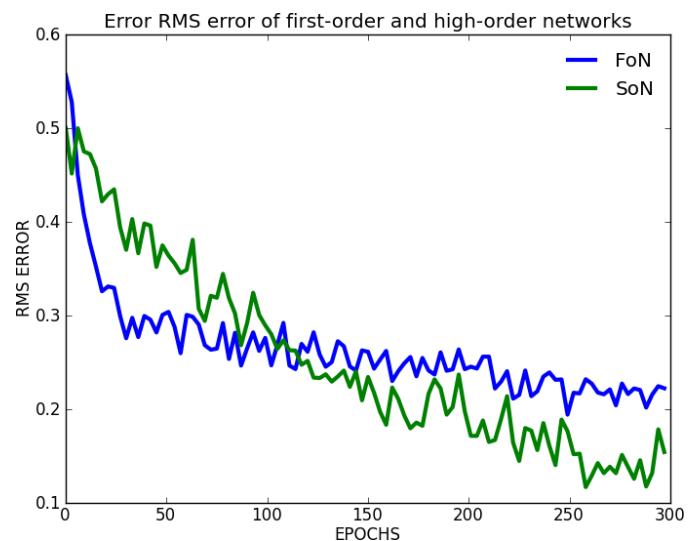
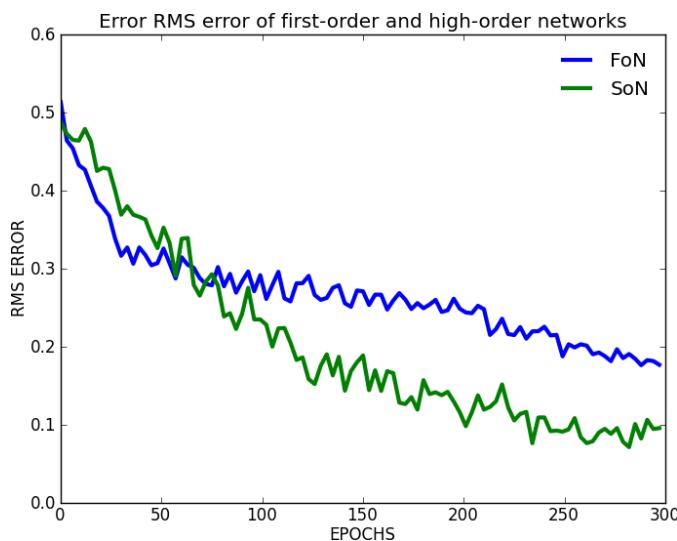
Notes

- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque
- la courbe rouge représentent le taux de paris hauts du second réseau

Conclusion

- dans les 2 cas, le premier réseau réussit à apprendre sa tâche de classification
- dans les 2 cas, le second réseau n'arrive pas à tirer parti de ses représentations, il se contente simplement de parier haut au bout d'un moment

Secondaires RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)

Conclusion Il n'y a plus de pique du second réseau comme dans les Expérience C1 et Expérience C2.

Conclusion

Comme il l'est dit dans Cleeremans Alex (2007), cette architecture de résoudre les problèmes des réseaux connexionnistes classiques à avoir un semblant de conscience.

À savoir :

- qu'ils ne savent pas qu'ils peuvent se trouver dans différents états, et qu'ils ne traitent pas leurs propres états : d'où la présence de ce second réseau qui tente de traiter ses états et d'apprendre qu'il en a plusieurs
- des représentations rentant bloquées dans la chaîne de causalité de la tâche à apprendre : d'où le fait que ce second réseau n'affecte pas l'apprentissage du premier (ie. pour ne pas retomber dans la chaîne de causalité)
- qu'ils n'ont pas les connaissances conscientes des raisons de leurs décisions : on essaye de les y sensibiliser avec les paris

Lors du passage sur des données réelles non linéairement séparables, l'utilité du second réseau s'écroule, les représentations ne sont plus exploitées.

Pour tenter de résoudre ce problème, l'Expérience C4 augmente le nombre de couche du premier réseau.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Fisher, R. (1988). Iris plants database. 150 (50 in each of three classes).

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience C4

Objectif

Comprendre de quelles manières un réseau de neurone connexioniste peut parier sur ses propres résultats à partir de ses représentations personnelles.

Réalisation de la seconde expérience de l'article Cleeremans Alex (2007) sur des données réelles non linéairement séparables à l'aide de plusieurs couches dans le perceptron.

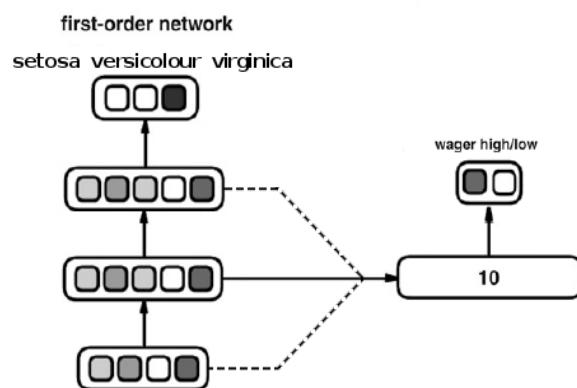
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des fleurs caractérisées par 4 neurones d'entrées qui représentent taille et largeur de la pétale et la sépale. Il est composé de plusieurs couche cachée de 5 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de ses couches cachées et de sa couche d'entrée.

L'apprentissage du second réseau, n'affecte pas le premier réseau.

Schéma

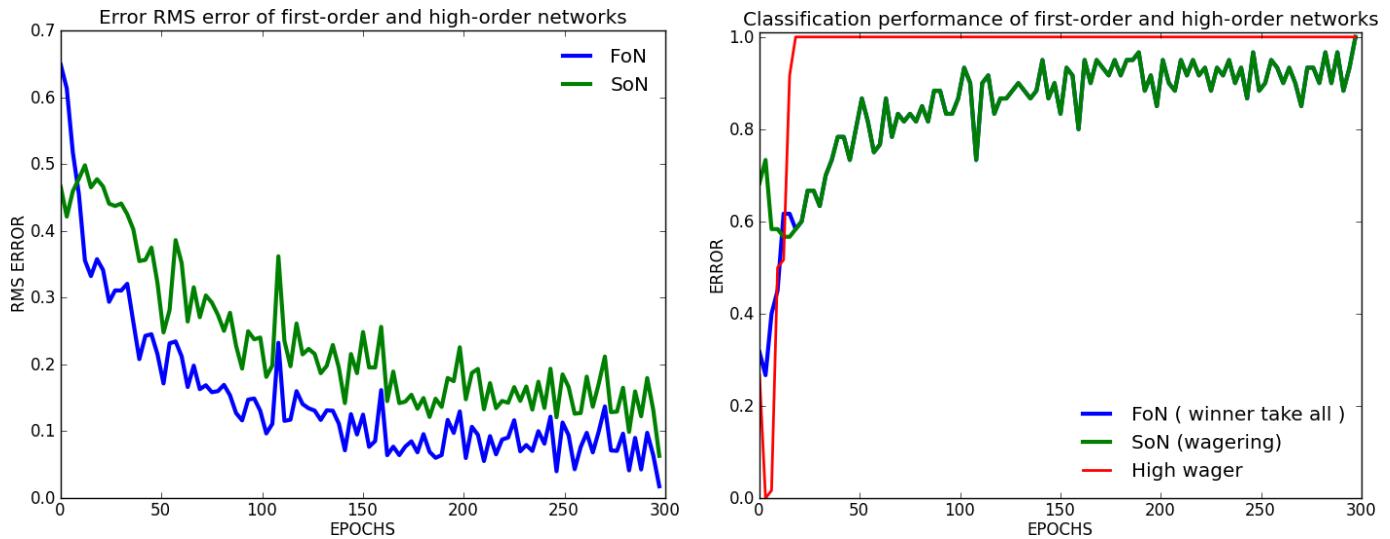


Paramètres

- momentum : 0.9 sur les 2 réseaux
- taux d'apprentissage : 0.1 sur les 2 réseaux
- 10 chiffres différents présentés
- apprentissage 10 (formes) x 1000 (époques)
- utilisation de biais
- poids initialisés sur [-0.25 ; 0.25]
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



Notes

- la courbe rouge représentent le taux de paris hauts du second réseau
- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion Peut importe l'architecture choisi (2 couches cachées, 1 couches cachée, 2 couches cachées + entrées), les performances sont sensiblement les mêmes.

Le second réseau n'arrive pas à tirer parti des représentations du premier réseau et se contente de partier hauts à chaque fois au bout d'un moment.

Conclusion

Lors du passage sur des données réelles non linéairement séparables, l'utilité du second réseau s'écroule, les représentations ne sont plus exploitées, même avec plusieurs couches.

Il faut continuer les essais pour réussir à le faire fonctionner.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience D1

Objectif

En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement.

Architecture

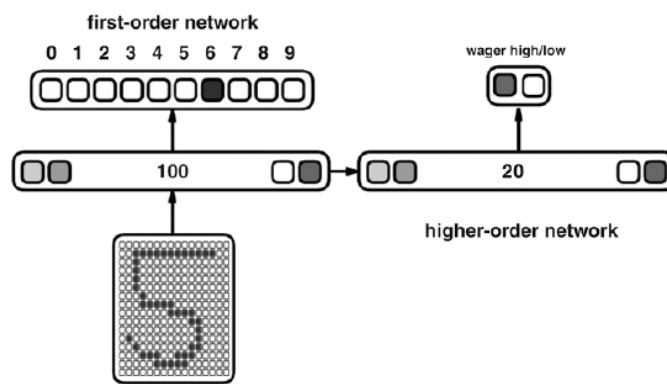
Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Lorsque le second réseau parie haut, la réponse du premier réseau est gardée, à l'inverse, lorsqu'il parie bas, c'est le second neurone le plus élevé du premier réseau qui sera la réponse.

Schéma

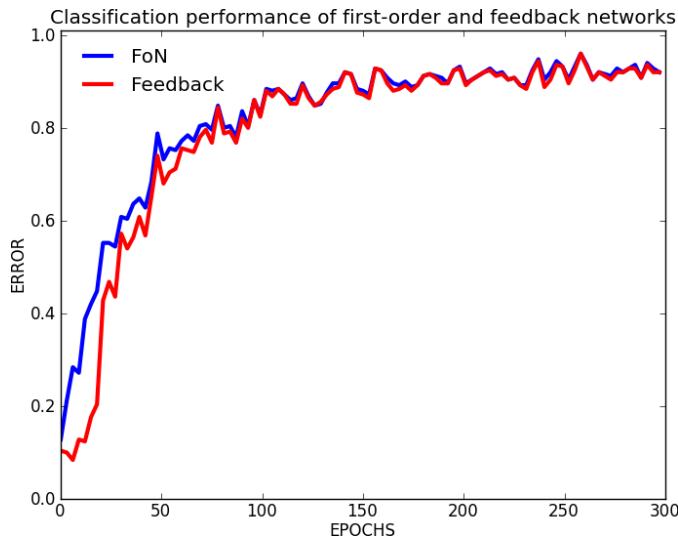


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le second réseau
- taux d'apprentissage : 0.15 sur les 2 réseaux
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 50 (formes) x 300 (époques)
- utilisation de biais
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le second réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



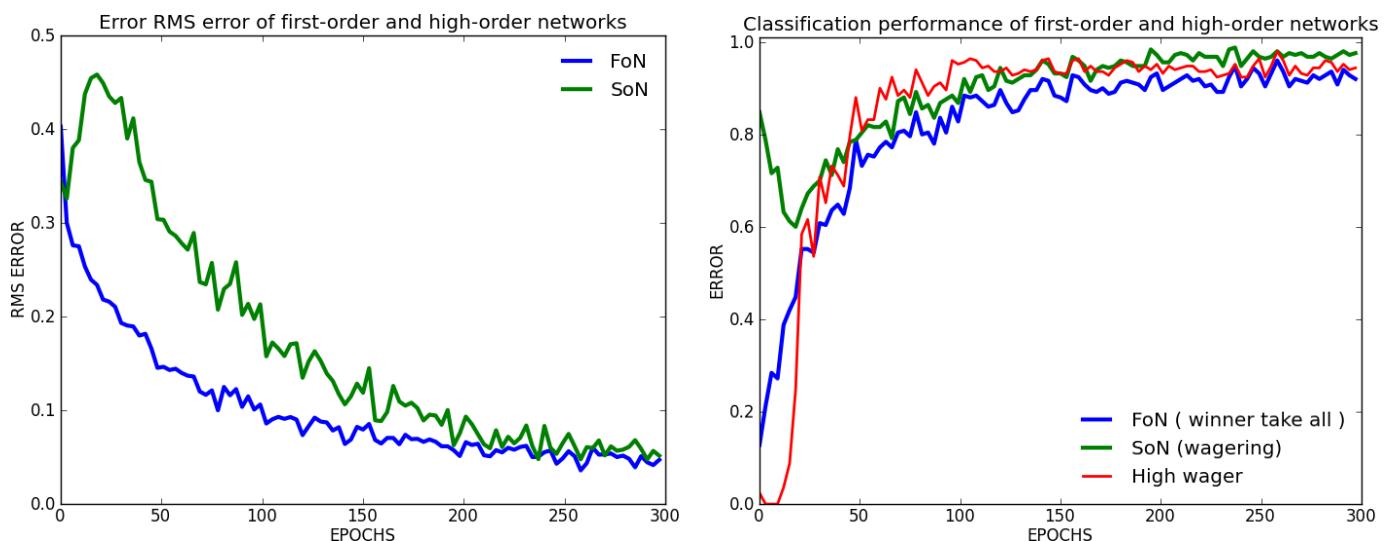
Notes

- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- l'architecture n'arrive pas à obtenir une augmentation de performance, ce qui veut dire que la bonne réponse ne se trouve pas dans le second neurone le plus élevé (lorsque la réponse n'est pas dans le premier).

Secondaires Analyse des performances sous-jacentes



Notes

- la courbe rouge représentent le taux de paris hauts du second réseau
- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque
- formule utilisée pour RMS (cf. Formules RMS)

Conclusion Les performances du second réseau sont respectables ceci prouve bien l'hypothèse précédente.

Conclusion

Lorsque la réponse n'est pas dans le neurone le plus élevé, elle ne se trouve pas non plus toujours dans le second neurone le plus élevé.

Cette idée nous ouvre le chemin vers l'Expérience D2 qui tentera d'enregistrer la profondeur de la bonne réponse.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience D2

Objectif

En partant de la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement.

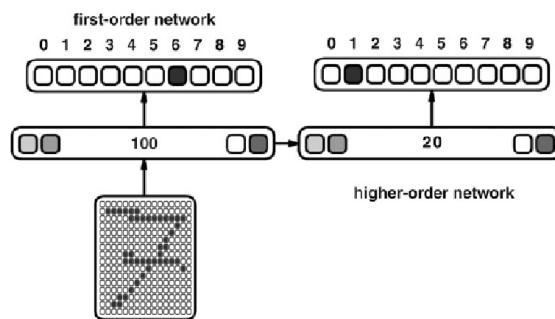
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron multicouche apprend à suivre une **intuition** représentant la “distance” entre l'indice du neurone gagnant le winner-take-all et l'indice du neurone ayant la bonne réponse.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

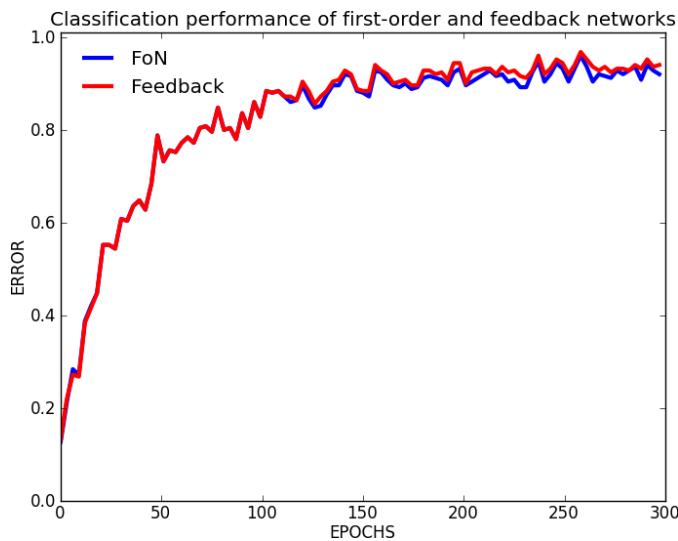


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le second réseau
- taux d'apprentissage : 0.15 sur les 2 réseaux
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 50 (formes) x 300 (époques)
- utilisation de biais
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le second réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



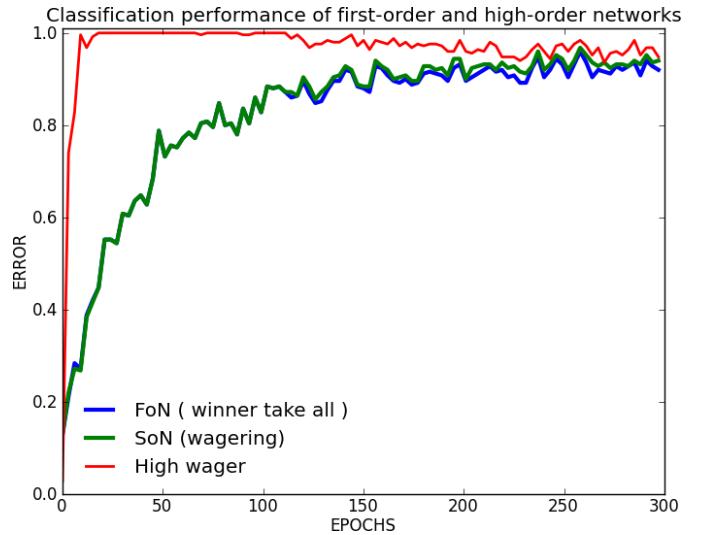
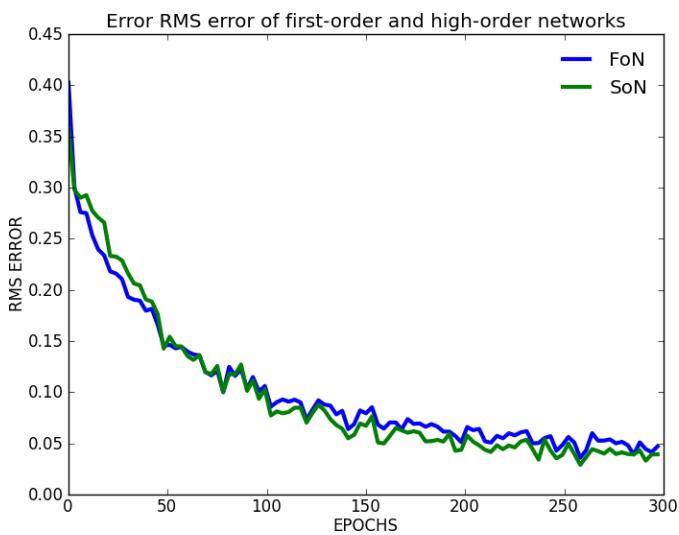
Notes

- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- l'architecture n'arrive qu'à une très faible augmentation de performance

Secondaires Analyse des performances sous-jacentes



Notes

- la courbe rouge représentent le taux de paris hauts (ie. premier neurone activé) du second réseau
- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque
- formule utilisée pour RMS (cf. Formules RMS)

Conclusion Cette fois-ci les performances du second réseau sont bien plus mitigées. N'ayant plus seulement 2 sorties mais 10, il ne dépasse que de peu le premier réseau.

Conclusion

L'augmentation de performance est moindre, à cause d'une tâche plus ardue pour le second neurone.

Avant de fermer ce chemin, il faudrait pousser un peu plus les recherches au niveau des paramètres choisis pour les réseaux.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience D3

Objectif

En partant de la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement.

Architecture

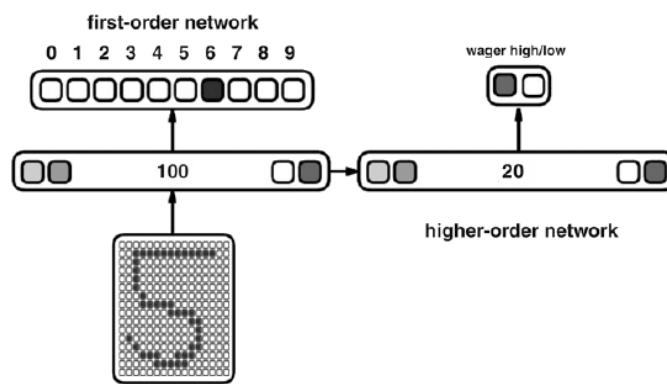
Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Lorsque le second réseau parie bas, le taux d'apprentissage du premier réseau est augmenté et le momentum est diminué. À l'inverse, lorsqu'il parie haut, c'est que la forme est déjà apprise, le taux d'apprentissage sera faible et le momentum élevé.

Schéma

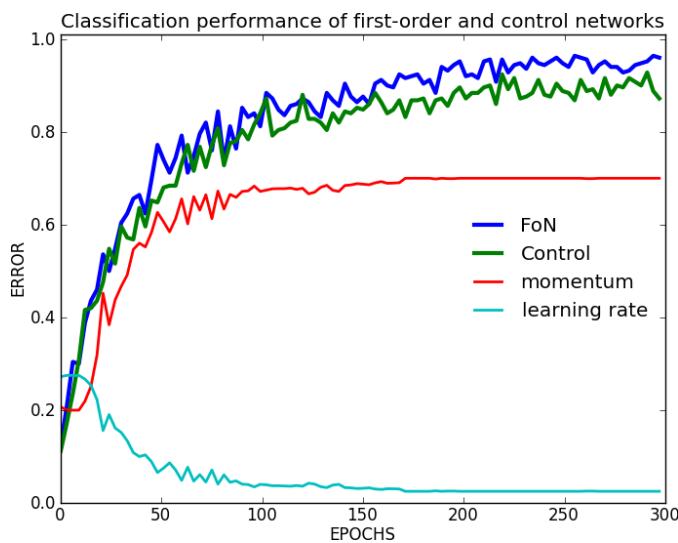


Paramètres

- momentum : 0. sur le second réseau
- taux d'apprentissage : 0.15 sur les second réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 50 (formes) x 300 (époques)
- utilisation de biais
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le second réseau
- entrées valent 0 ou 1
- sigmoïde à température 1

Résultats

Principaux Analyse des performances



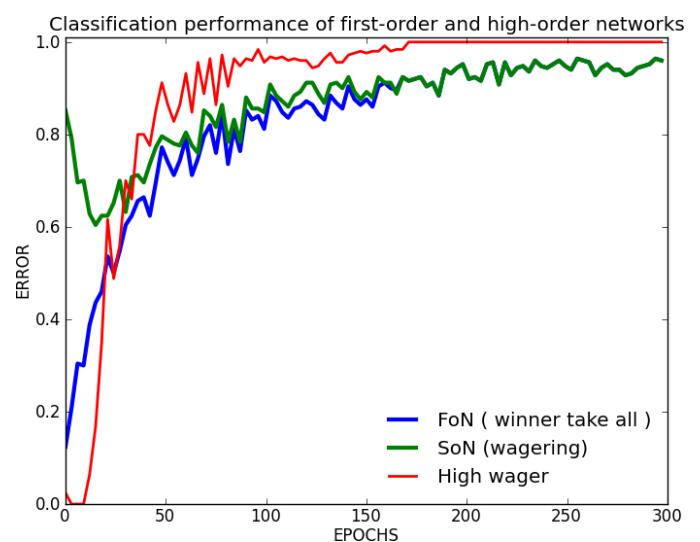
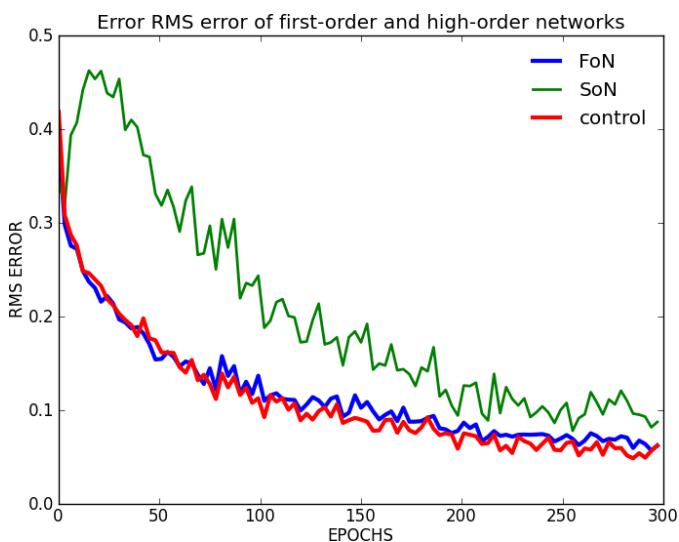
Notes

- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque
- momentum et learning rate sont la moyenne des paramètres appliqués au premier réseau sur une époque

Conclusion

- On obtient une hausse de performance
- Comme la règle le défini, à la fin (ie. paris hauts) le momentum est élevé et le taux d'apprentissage faible

Secondaires Analyse des performances sous-jacentes



Notes

- la courbe rouge représentent le taux de paris hauts du second réseau
- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

- formule utilisée pour RMS (cf. Formules RMS)

Conclusion Il faut se méfier du RMS, car même si l'erreur globale sur un neurone est plus faible avec control, durant la classification on cherche juste le neurone le plus actif.

Conclusion

L'expérience est plutôt concluante, le bémol vient du fait que c'est l'humain qui règle le comportement du réseau : le taux d'apprentissage/momentum à utiliser dans tel ou tel cas.

Dans l'Expérience D4, nous allons tenter de laisser le réseau le faire lui-même.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience D4

Objectif

En partant de la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement.

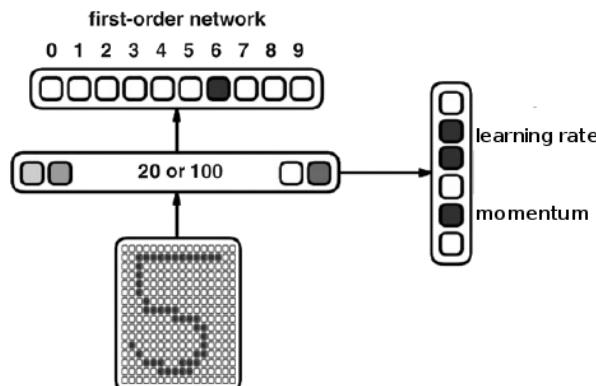
Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron apprend par renforcement le taux d'apprentissage et le momentum idéale en fonction des représentations du premier réseau. Ces paramètres sont encodé sur 3 neurones de façon binaire. Au final, le taux d'apprentissage est défini sur [0.1 ; 0.35] et le momentum sur [0.2 ; 0.85].

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Schéma

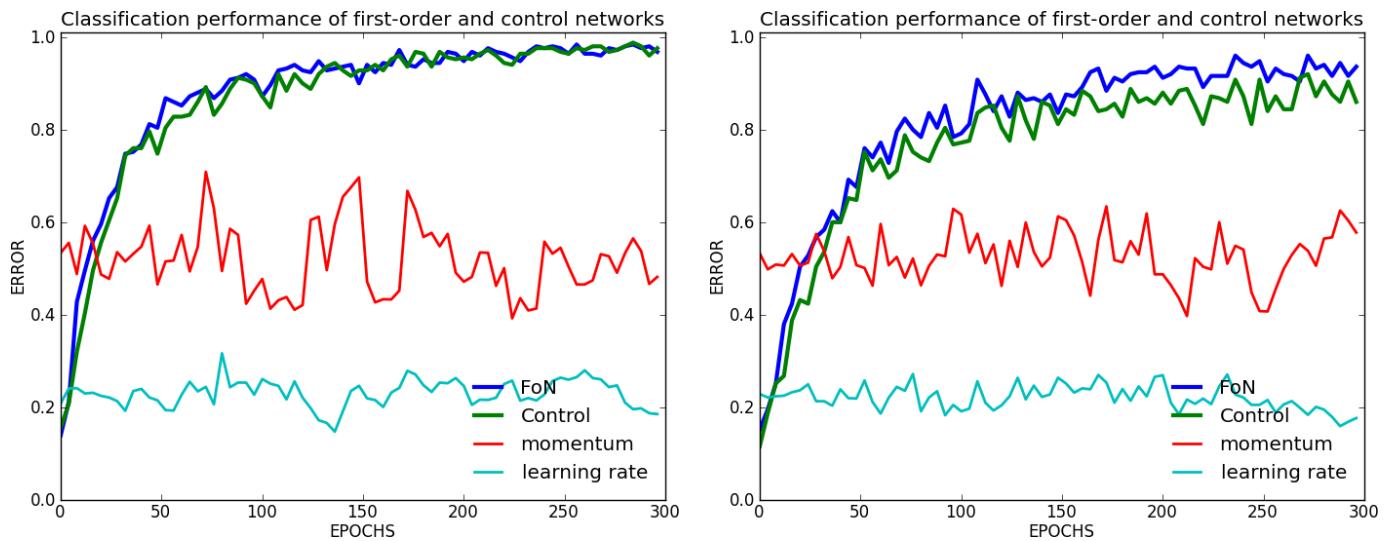


Paramètres

- momentum : 0.9 sur les 2 réseaux
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- apprentissage 50 (formes) x 300 (époques)
- poids initialisés sur [-0.25 ; 0.25]
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais

Résultats

Principaux Analyse des performances



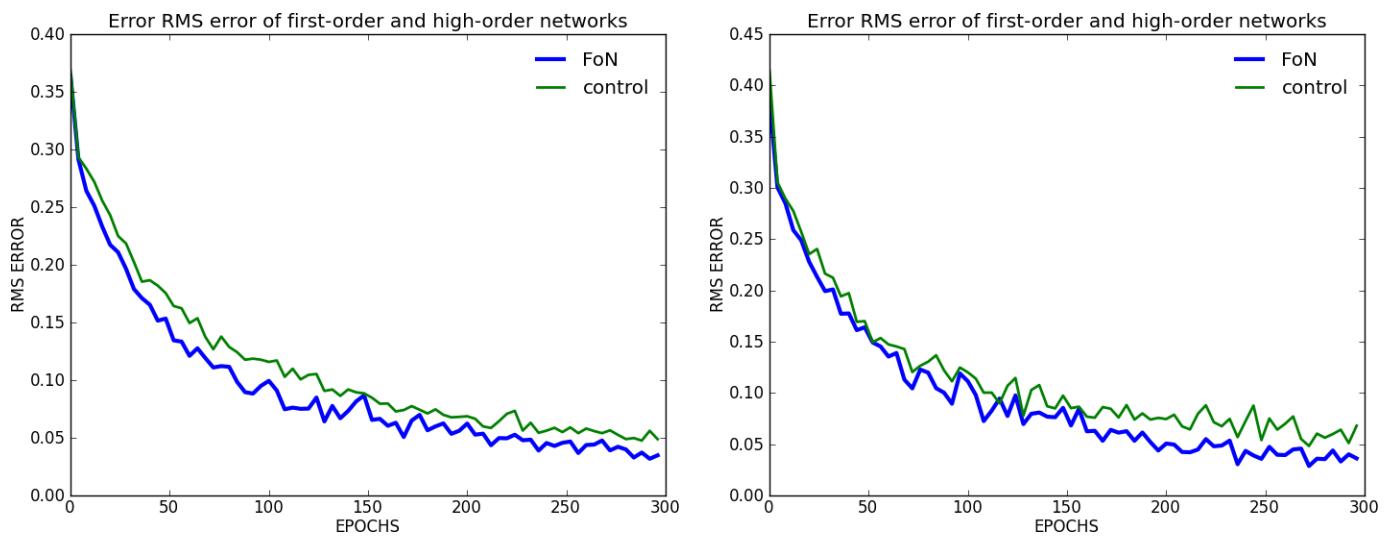
Notes

- la performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque
- momentum et learning rate sont la moyenne des paramètres appliquées au premier réseau sur une époque

Conclusion

- il y a une légère hausse de performance par rapport à un réseau de base, elle est plus accentuée lorsque le nombre de neurones dans la couche cachée est élevé

Secondaires Analyse des performances RMS



Notes

- formule utilisée pour RMS (cf. Formules RMS)

Conclusion Le RMS ne nous dit pas grand chose de plus que la performance de classification.

Conclusion

L'expérience est plutôt concluente, mais le système d'encodage/décodage du perceptron à apprentissage par renforcement peut probablement être amélioré.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience F1

Objectif

En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement, à partir d'un 3^{ème} réseau.

Architecture

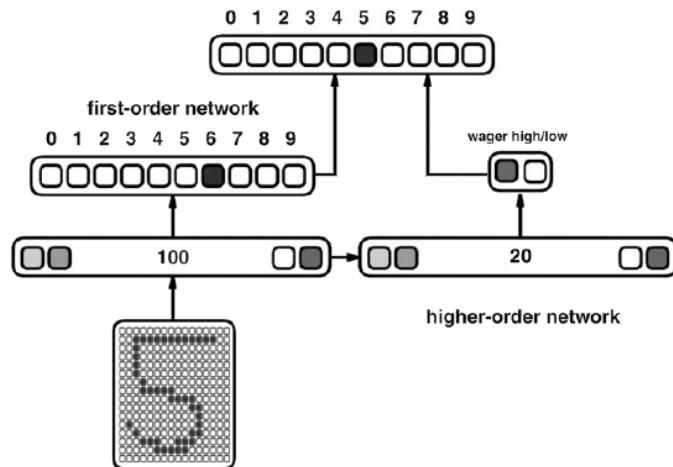
Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Un troisième réseau de perceptron apprend la solution à partir des sorties des 2 premiers réseau. Son apprentissage n'affecte aucun des 2 sous réseaux.

Schéma

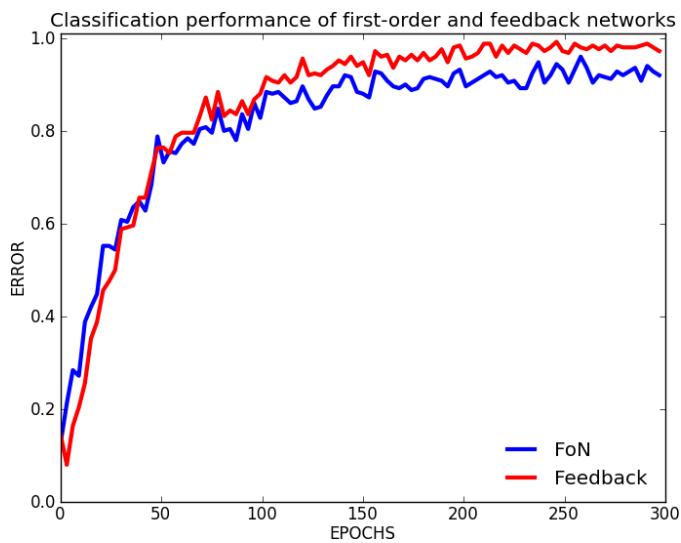


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage : 0.15 sur le premier réseau
- taux d'apprentissage : 0.1 sur 2^{ème} et 3^{ème} réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais
- apprentissage 50 (formes) x 300 (époques)

Résultats

Principaux Analyse des performances



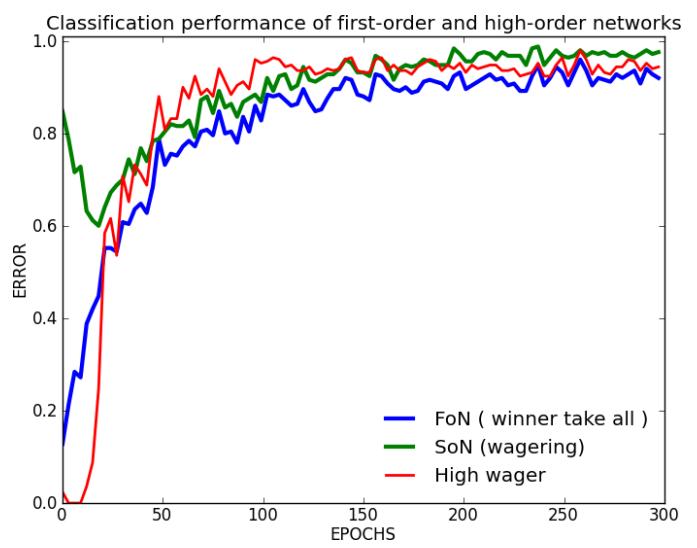
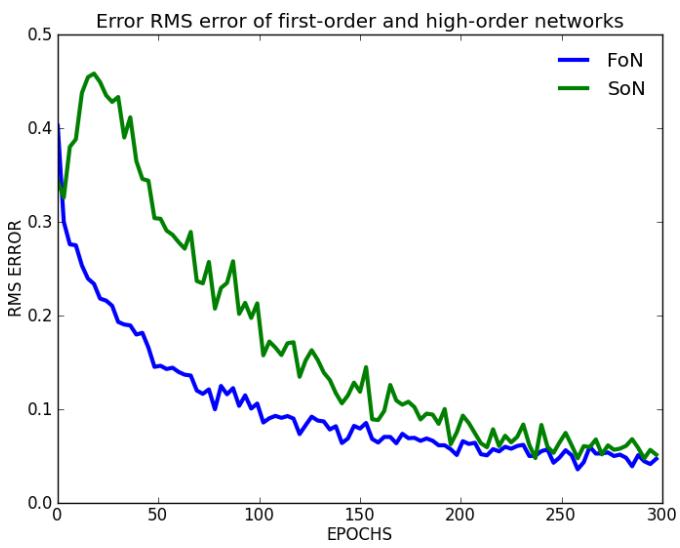
Notes

- La performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- il faut attendre les 50 premières époques pour que le 3^{ème} réseau puisse profiter d'une hausse de performance

Secondaires Analyse des performances sous-jacentes



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- la courbe rouge représente le taux de paris haut du second réseau

Conclusion Les bonnes performances du second réseau permettent une l'augmentation de performance du 3^{ème} réseau. On retrouve toujours la chute puis la remontée du second réseau.

Conclusion

La mise en place de ce 3^{ème} réseau permet une légère hausse de performance. Il est intéressant de voir qu'uniquement grâce à un chiffre donné (sortie du premier réseau) et grâce à un paris (sortie du 2^{ème} réseau), on arrive quand même à avoir une hausse de performance.

Pour résoudre le problème de temps d'apprentissage des 50 premières époques et du 3^{ème} réseau supplémentaire, l'Expérience G1 tente de supprimer ce 3^{ème} réseau et de fusionner les sorties.

Évidemment, comme nous l'avons montré dans les expériences C, il faut un nombre élevé de neurone dans la couche cachée du premier réseau. Cette hausse de performance ne peut dépasser les performances d'un réseau optimal.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience F2

Objectif

En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement, à partir d'un 3^{ème} réseau.

Architecture

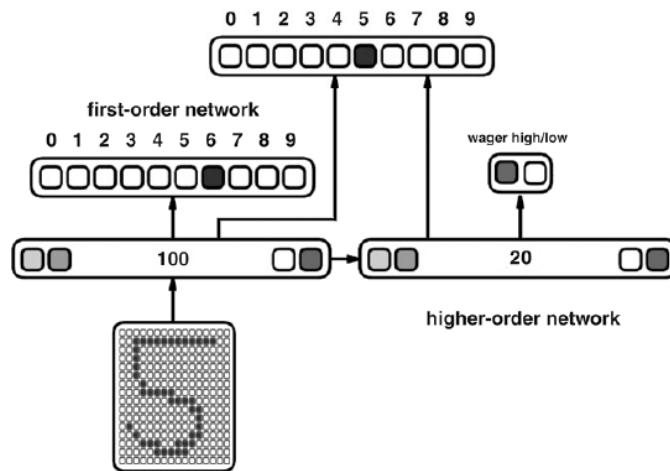
Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Un troisième réseau de perceptron apprend la solution à partir des couches cachées des 2 premiers réseau. Son apprentissage n'affecte aucun des 2 sous réseaux.

Schéma

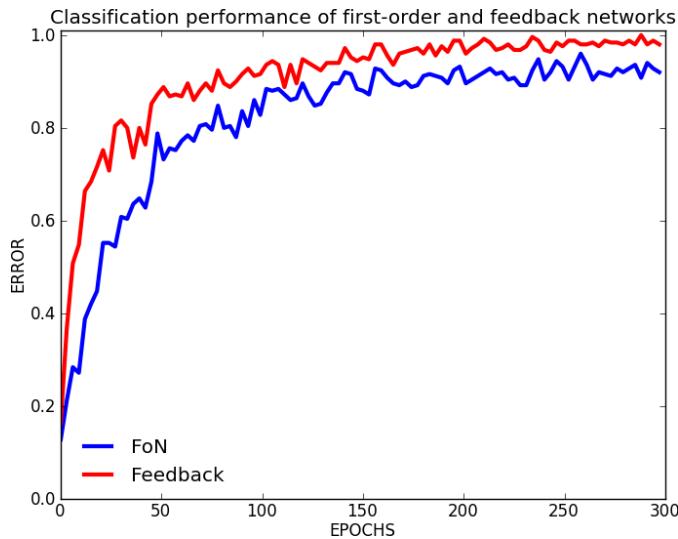


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage : 0.15 sur le premier réseau
- taux d'apprentissage : 0.1 sur 2^{ème} et 3^{ème} réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais
- apprentissage 50 (formes) x 300 (époques)

Résultats

Principaux Analyse des performances



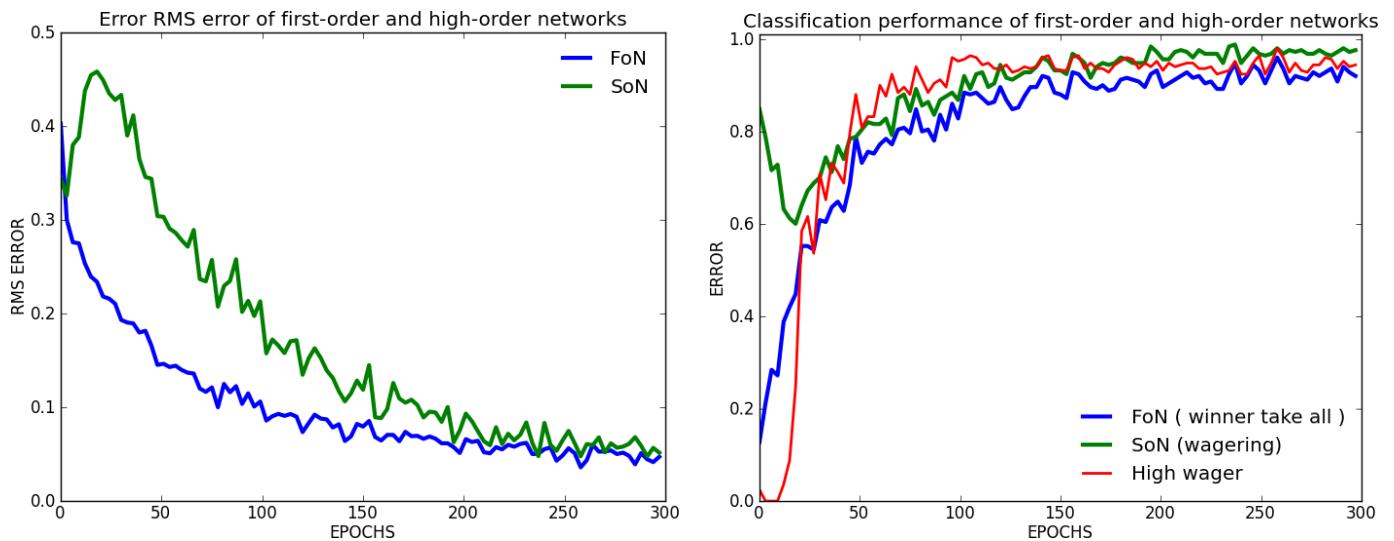
Notes

- La performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- Il y a une très belle hausse de performance toute au long de l'apprentissage

Secondaires Analyse des performances sous-jacentes



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- la courbe rouge représente le taux de paris haut du second réseau

Conclusion Les bonnes performances du second réseau permettent une l'augmentation de performance du 3^{ème} réseau. On retrouve toujours la chute puis la remontée du second réseau.

Conclusion

La mise en place de ce 3^{ème} réseau permet une belle hausse de performance. Il est intéressant de voir que le dernier perceptron exploite à la fois les **représentations** et les **méta-représentations**.

Évidemment, comme nous l'avons montré dans les expériences C, il faut un nombre élevé de neurone dans la couche cachée du premier réseau. Cette hausse de performance ne peut dépasser les performances d'un réseau optimal.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience F3

Objectif

En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement, à partir d'un 3^{ème} réseau.

Architecture

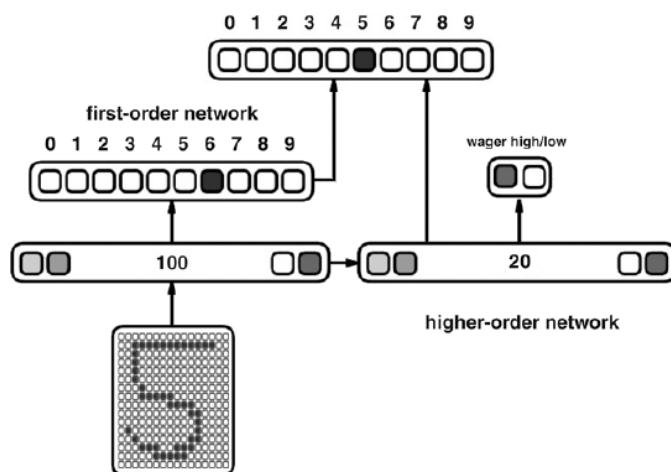
Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Un troisième réseau de perceptron apprend la solution à partir de la couche de sortie du premier réseau et de la couche cachée du second réseau. Son apprentissage n'affecte aucun des 2 sous réseaux.

Schéma

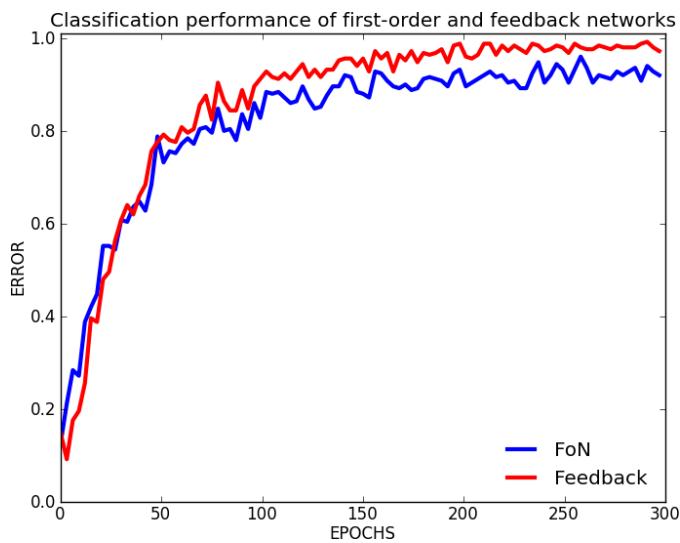


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage : 0.15 sur le premier réseau
- taux d'apprentissage : 0.1 sur 2^{ème} et 3^{ème} réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais
- apprentissage 50 (formes) x 300 (époques)

Résultats

Principaux Analyse des performances



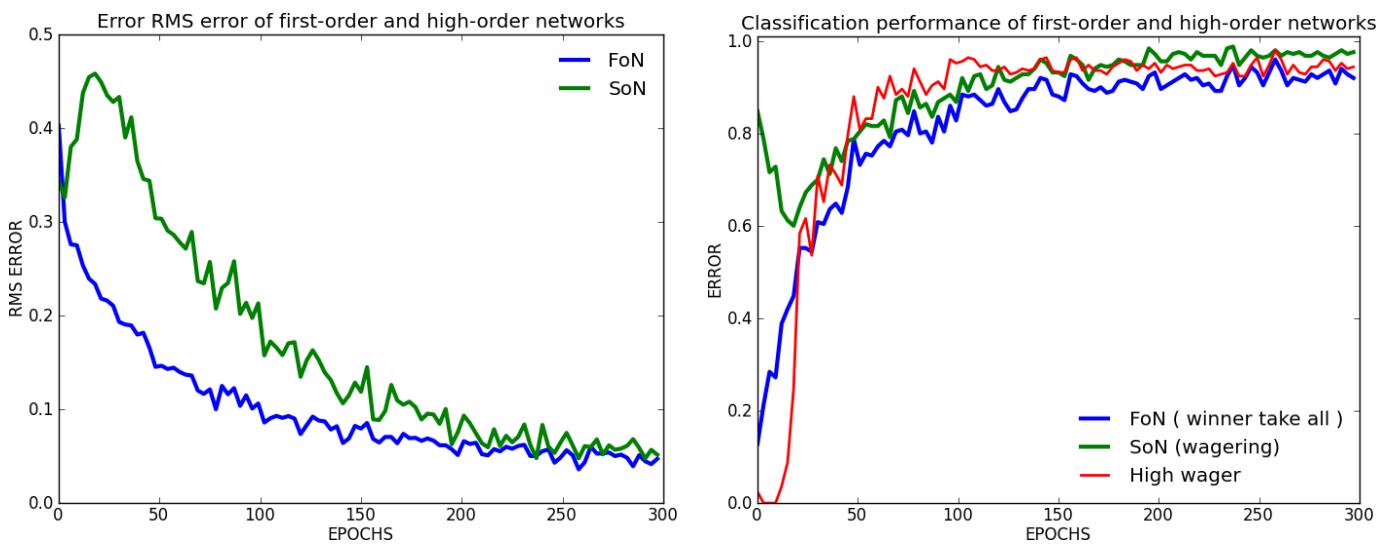
Notes

- La performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- il faut attendre les 50 premières époques pour que le 3^{ème} réseau puisse profiter d'une hausse de performance

Secondaires Analyse des performances sous-jacentes



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- la courbe rouge représente le taux de paris haut du second réseau

Conclusion Les bonnes performances du second réseau permettent une l'augmentation de performance du 3^{ème} réseau. On retrouve toujours la chute puis la remontée du second réseau.

Conclusion

La mise en place de ce 3^{ème} réseau permet une légère hausse de performance.

Pour résoudre le problème de temps d'apprentissage des 50 premières époques et du 3^{ème} réseau supplémentaire, l'Expérience G2 tente de supprimer ce 3^{ème} réseau et de fusionner les sorties.

Évidemment, comme nous l'avons montré dans les expériences C, il faut un nombre élevé de neurone dans la couche cachée du premier réseau. Cette hausse de performance ne peut dépasser les performances d'un réseau optimal.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience F4

Objectif

En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement, à partir d'un 3^{ème} réseau.

Architecture

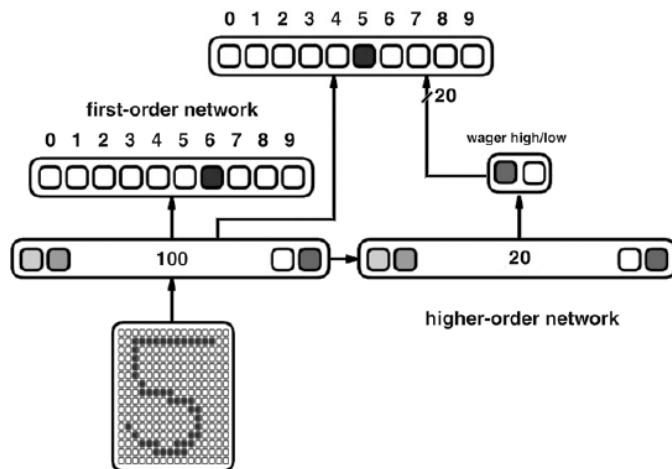
Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Un troisième réseau de perceptron apprend la solution à partir de la couche de cachée du premier réseau et de la couche de sortie du second réseau. Son apprentissage n'affecte aucun des 2 sous réseaux.

Schéma

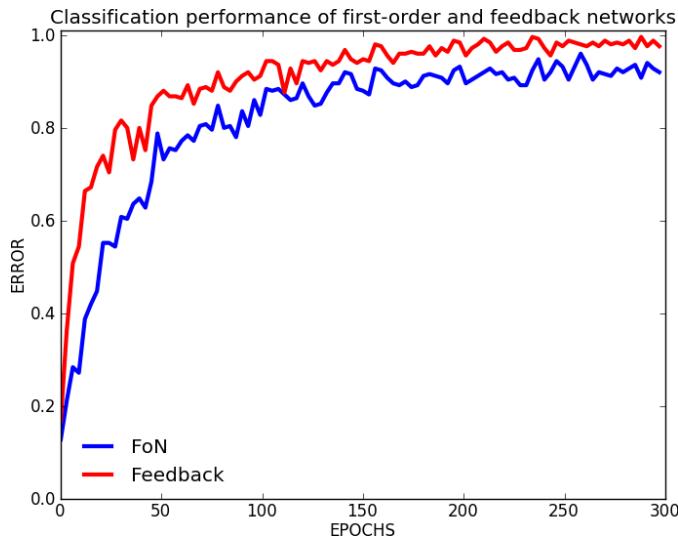


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage : 0.15 sur le premier réseau
- taux d'apprentissage : 0.1 sur 2^{ème} et 3^{ème} réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais
- apprentissage 50 (formes) x 300 (époques)

Résultats

Principaux Analyse des performances



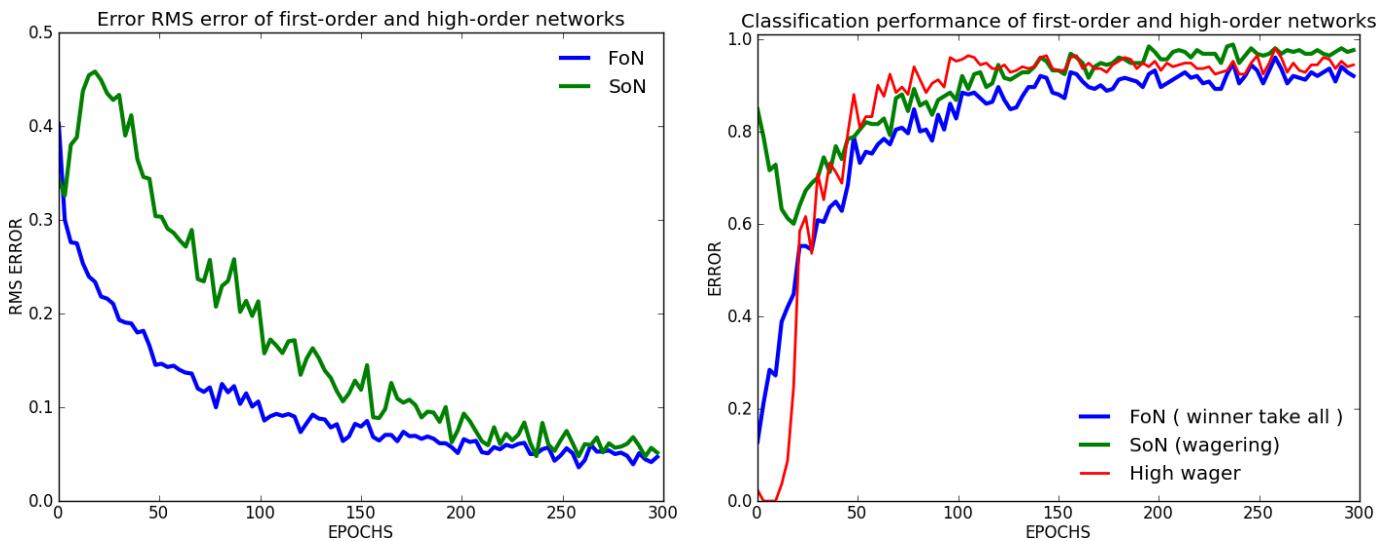
Notes

- La performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- Il y a une très belle hausse de performance toute au long de l'apprentissage

Secondaires Analyse des performances sous-jacentes



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- la courbe rouge représente le taux de paris haut du second réseau

Conclusion Les bonnes performances du second réseau permettent une l'augmentation de performance du 3^{ème} réseau. On retrouve toujours la chute puis la remontée du second réseau.

Conclusion

La mise en place de ce 3^{ème} réseau permet une belle hausse de performance.

Évidemment, comme nous l'avons montré dans les expériences C, il faut un nombre élevé de neurone dans la couche cachée du premier réseau. Cette hausse de performance ne peut dépasser les performances d'un réseau optimal.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience F5

Objectif

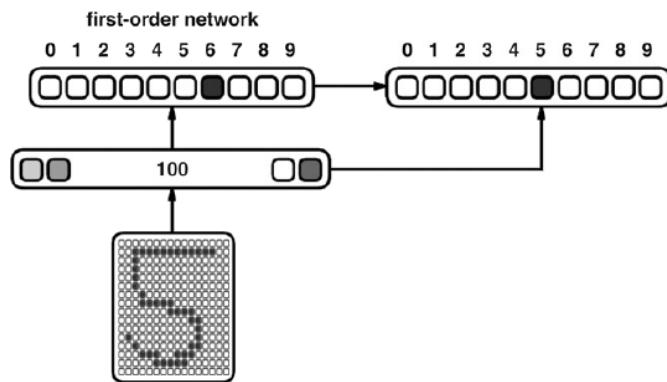
En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement, à partir d'un 3^{ème} réseau.

Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un deuxième réseau de perceptron apprend la solution à partir de la couche cachée et de la couche de sortie du premier réseau. Son apprentissage n'affecte pas le premier réseau.

Schéma

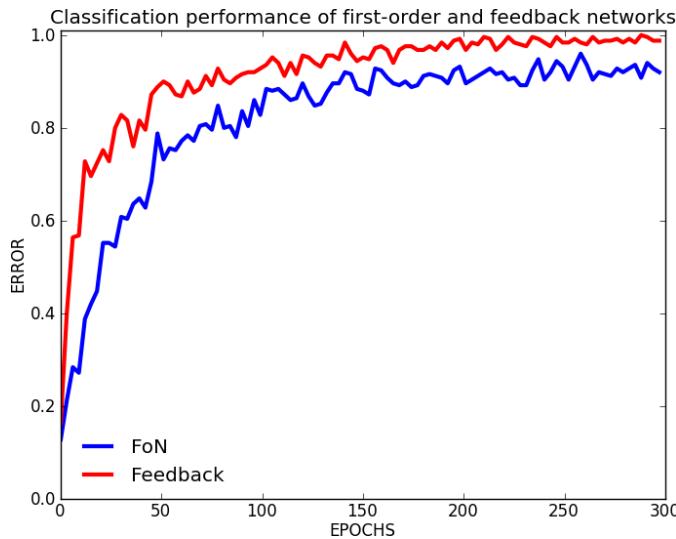


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le 2^{ème} réseau
- taux d'apprentissage : 0.15 sur le premier réseau
- taux d'apprentissage : 0.1 sur 2^{ème} réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le 2^{ème} réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais
- apprentissage 50 (formes) x 300 (époques)

Résultats

Principaux Analyse des performances



Notes

- La performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- Il y a une très belle hausse de performance toute au long de l'apprentissage

Conclusion

Cette belle hausse de performance est très étrange car elle **remet en cause l'utilité du second réseau et des paris.**

Formules

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\left\{ \begin{array}{l} f : \text{fonction sigmoïde} \\ x_i : \text{valeur du neurone } i \\ d_i : \text{valeur désirée pour le neurone } i \\ a_i : \text{somme pondérée des poids du neurone } i \end{array} \right.$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. *doi :10.1016/j.neunet.2007.09.011*.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience F6

Objectif

En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement, à partir d'un 3^{ème} réseau.

Architecture

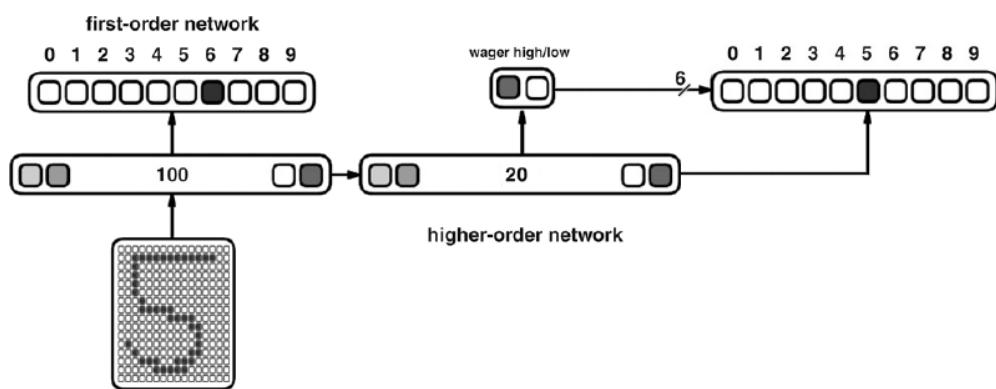
Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones.

Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Un troisième réseau de perceptron apprend la solution à partir de la couche cachée et de la couche de sortie du second réseau. Son apprentissage n'affecte aucun des 2 sous réseaux.

Schéma

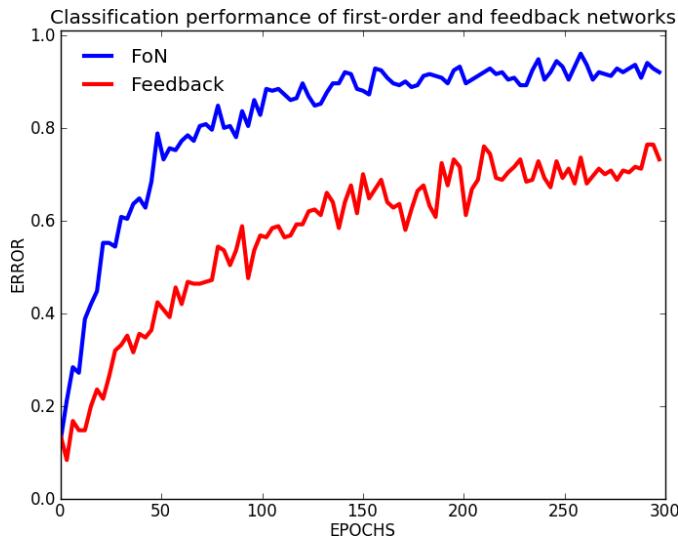


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage : 0.15 sur le premier réseau
- taux d'apprentissage : 0.1 sur 2^{ème} et 3^{ème} réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le 2^{ème} et 3^{ème} réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais
- apprentissage 50 (formes) x 300 (époques)

Résultats

Principaux Analyse des performances



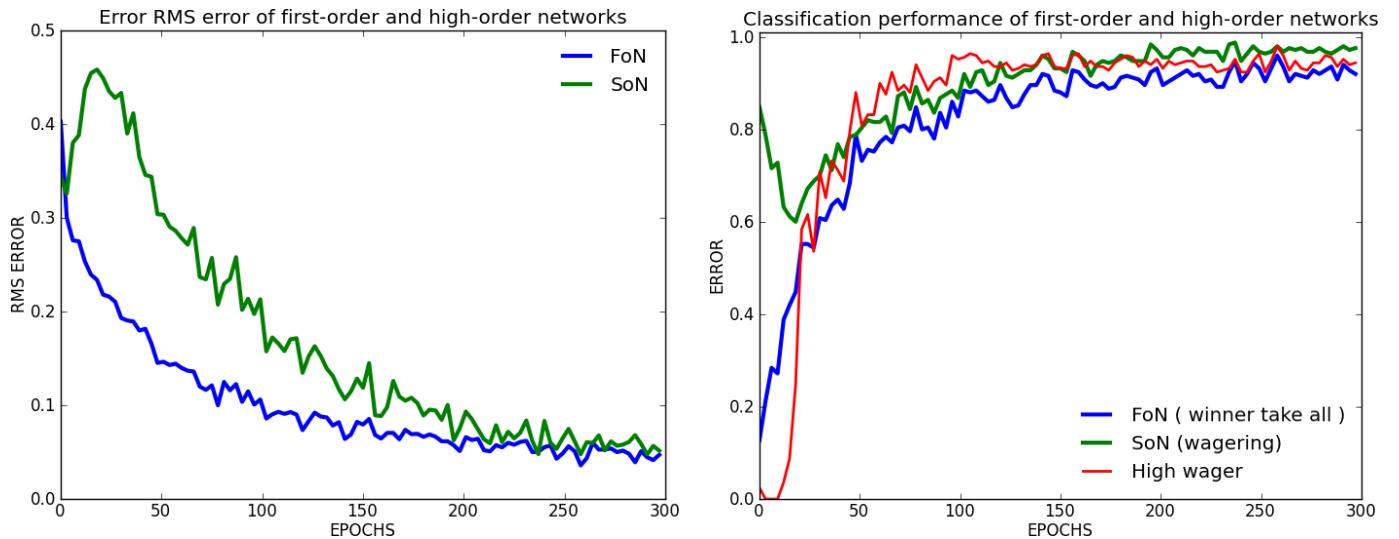
Notes

- La performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- il n'y a jamais de hausse de performance, la perte d'information est trop élevée

Secondaires Analyse des performances sous-jacentes



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- la courbe rouge représente le taux de paris haut du second réseau

Conclusion Même les bonnes performances du second réseau ne permettent une augmentation du 3^{ème} réseau. On retrouve toujours la chute puis la remontée du second réseau.

Conclusion

Il n'y a jamais de hausse de performance avec cette architecture, la perte d'information subit à travers les paris dans le second réseau est trop élevée pour pouvoir retrouver les chiffres.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience G1

Objectif

En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement, en fusionnant des réseaux.

Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones. Sa couche de sortie utilise les 100 neurones (de sa couche cachée) et 20 neurones (venant du second réseau).

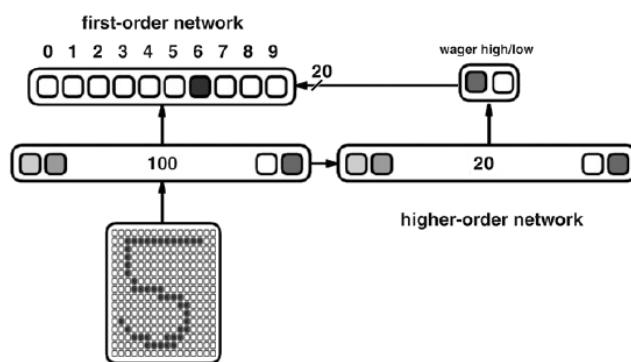
Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Lorsque le second réseau parie haut, la réponse du premier réseau est gardée, à l'inverse, lorsqu'il parie bas, c'est le second neurone le plus élevé du premier réseau qui sera la réponse.

L'apprentissage entre les 2 couches de sorties n'affecte pas les poids du second réseau.

Schéma

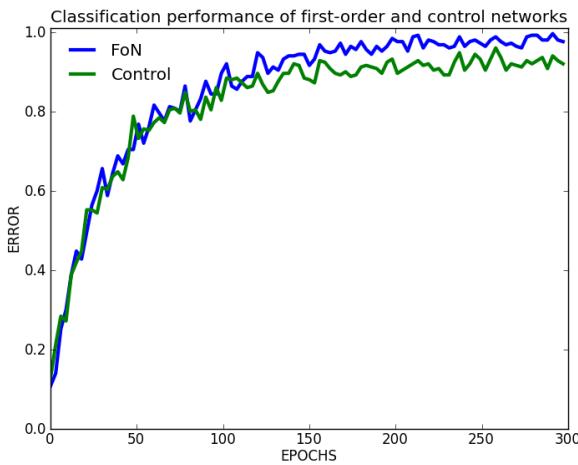


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le 2^{ème} réseau
- taux d'apprentissage : 0.15 sur le premier réseau
- taux d'apprentissage : 0.1 sur 2^{ème} réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le 2^{ème} réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais
- apprentissage 50 (formes) x 300 (époques)

Résultats

Principaux Analyse des performances



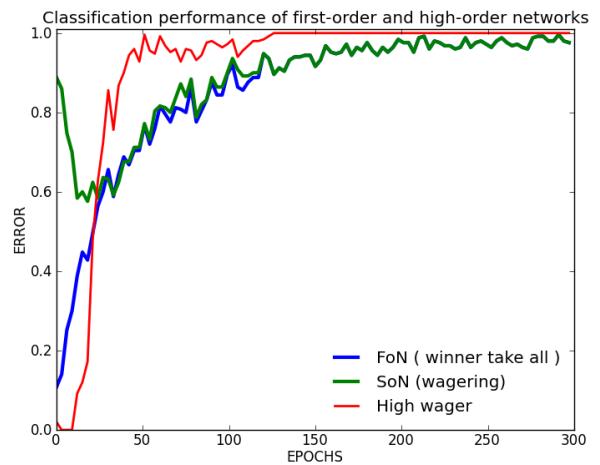
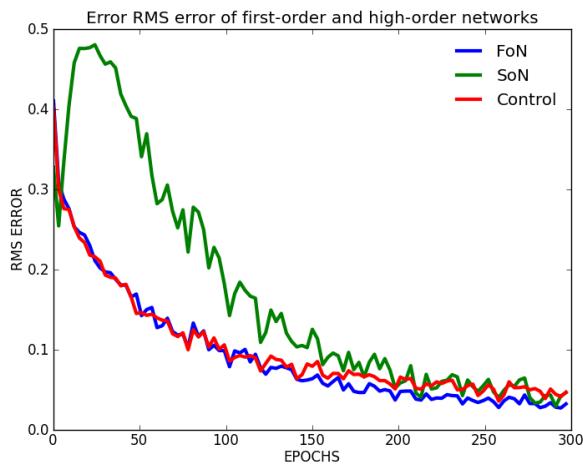
Notes

- La performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- Il y a une hausse de performance honorable.

Secondaires Analyse des performances sous-jacentes



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- la courbe rouge représente le taux de paris haut du second réseau

Conclusion Les performances du second réseau ne sont pas des plus convaincantes.

Conclusion

La mise en place de cette architecture permet une légère hausse de performance pour un nombre de connexions supplémentaire très faible.

Évidemment, comme nous l'avons montré dans les expériences C, il faut un nombre élevé de neurone dans la couche cachée du premier réseau. Cette hausse de performance ne peut dépasser les performances d'un réseau optimal.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expérience G2

Objectif

En utilisant la seconde architecture de Cleeremans Alex (2007), comprendre de quelles manières un réseau de neurone connexionniste peut, à partir de ses propres paris sur son résultat, améliorer son comportement, en fusionnant des réseaux.

Architecture

Description Un premier réseau de perceptron multicouche apprend à discréteriser des chiffres représentés par 256 (16x16) neurones d'entrées. Il est composé d'une couche cachée de 100 neurones. Sa couche de sortie utilise les 100 neurones (de sa couche cachée) et 20 neurones (venant du second réseau).

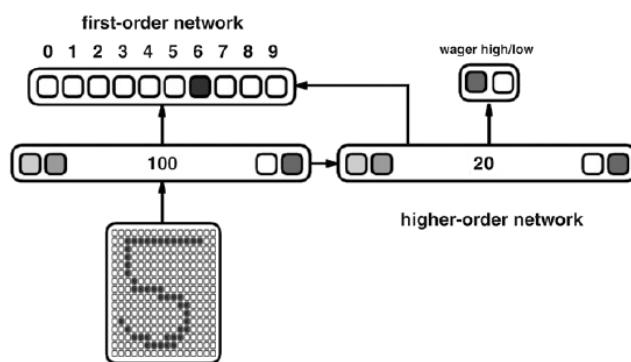
Un second réseau de perceptron multicouche apprend à parier sur la qualité de la réponse du premier réseau à partir de sa couche cachée.

L'apprentissage du second réseau, n'affecte pas les poids entre la couche d'entrée et la couche cachée du premier réseau.

Lorsque le second réseau parie haut, la réponse du premier réseau est gardée, à l'inverse, lorsqu'il parie bas, c'est le second neurone le plus élevé du premier réseau qui sera la réponse.

L'apprentissage entre les 2 couches de sorties n'affecte pas les poids du second réseau.

Schéma

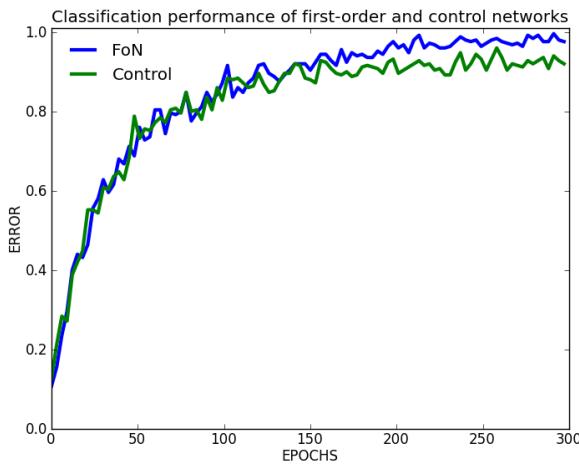


Paramètres

- momentum : 0.5 sur le premier réseau
- momentum : 0. sur le 2^{ème} réseau
- taux d'apprentissage : 0.15 sur le premier réseau
- taux d'apprentissage : 0.1 sur 2^{ème} réseau
- **1600 formes** de chiffres différents présentées (shuffle) Semeion Research Center (1994)
- poids initialisés sur [-1 ; 1] pour le premier réseau
- poids initialisés sur [-0.25 ; 0.25] pour le 2^{ème} réseau
- taux d'apprentissage constant
- entrées valent 0 ou 1
- sigmoïde à température 1
- utilisation de biais
- apprentissage 50 (formes) x 300 (époques)

Résultats

Principaux Analyse des performances



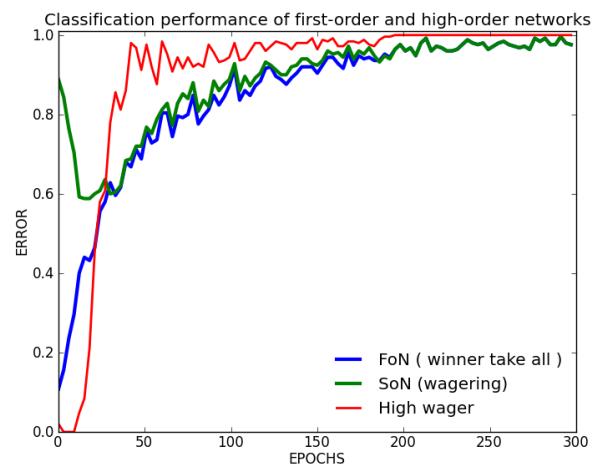
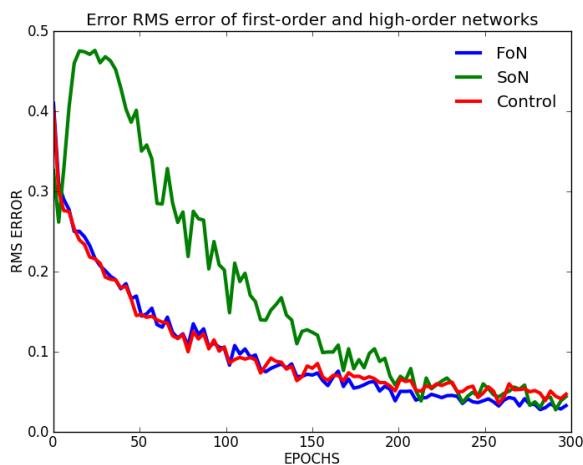
Notes

- La performance de classification représente le taux de bonnes réponses (winner-take-all) pour les 50 formes présentées sur une époque

Conclusion

- Il y a une hausse de performance honorable.

Secondaires Analyse des performances sous-jacentes



Notes

- formule utilisée pour RMS (cf. Formules RMS)
- la courbe rouge représente le taux de paris haut du second réseau

Conclusion Les performances du second réseau ne sont pas des plus convaincantes.

Conclusion

La mise en place de cette architecture permet une légère hausse de performance pour un nombre de connexions supplémentaire très faible.

Évidemment, comme nous l'avons montré dans les expériences C, il faut un nombre élevé de neurone dans la couche cachée du premier réseau. Cette hausse de performance ne peut dépasser les performances d'un réseau optimal.

Formules

RMS Pour une époque e :

$$rms_e = \sqrt{\frac{1}{n} \sum_{i=1}^n (o_{i,e} - d_i)^2}$$

with $\begin{cases} n : number of neurons on the output layer \\ o_{i,e} : value obtained for the i^{th} neuron at the e^{th} epoch \\ d_i : value desired for the i^{th} neuron \end{cases}$

Descente de gradient Touzet (1992)

Construction de l'erreur :

$$\begin{aligned} y_i &= f'(a_i) \times (d_i - x_i) \text{ si } i \text{ neurone de sortie} \\ y_i &= f'(a_i) \times \sum_k (w_{ki} \times y_k) \text{ si } i \text{ neurone cache} \end{aligned}$$

Mise à jour des poids :

$$w_{ij}(t+1) = w_{ij}(t) + learning_rate \times y_i \times x_j + momentum \times (w_{ij}(t) - w_{ij}(t-1))$$

Variables :

$$\begin{cases} f : fonction sigmoide \\ x_i : valeur du neurone i \\ d_i : valeur desire pour le neurone i \\ a_i : somme pondere des poids du neurone i \end{cases}$$

References

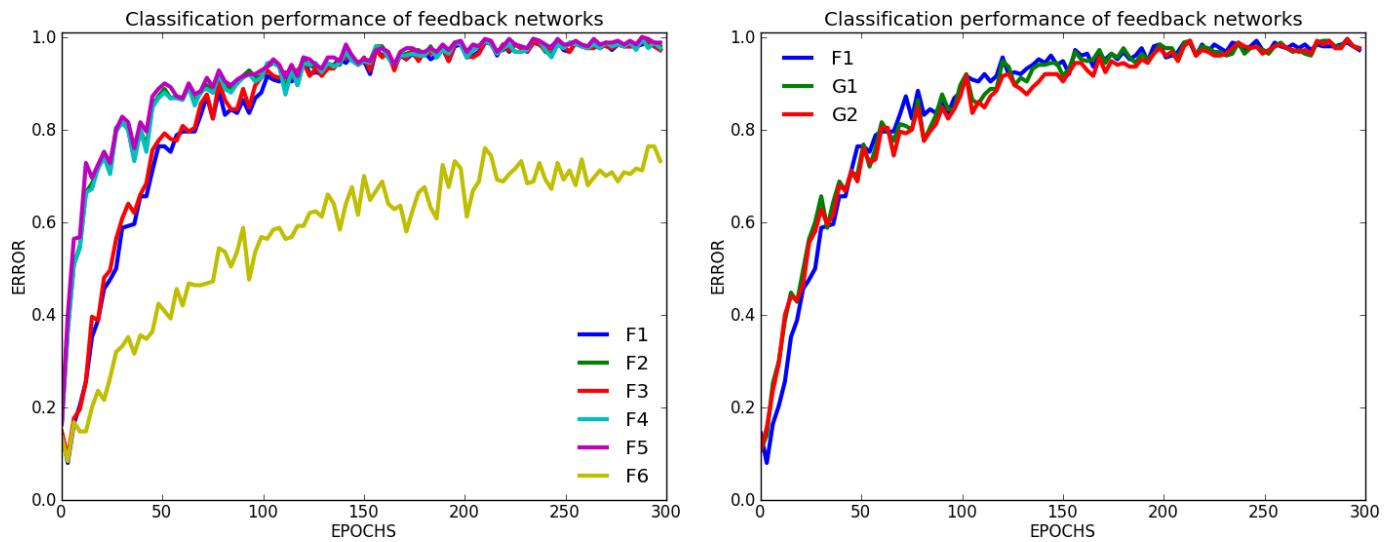
Cleeremans Alex, Timmermans Bert, P. A. (2007). Consciousness and metarepresentation : A computational sketch. doi :10.1016/j.neunet.2007.09.011.

Semeion Research Center, o. S. o. C. (1994). Semeion handwritten digit data set. 1593 handwritten digits from around 80 persons.

Touzet, C. (1992). Les réseaux de neurones artificiels - introduction au connexionnisme.

Expériences FG

Comparaison rapide des performances des expériences F et G.



Il faut prendre en compte que les expériences G s'exécutent plus rapidement que les F puisqu'elles n'ont pas de 3ème réseau de perceptron.