CentraleSupélec
Projet long du cursus Supélec
Encadré par : J. Tomasik et A. Rimmer

Dessines-moi un mouton
Genrative Adversial Network

François Bouvier d'Yvoire
Mathieu Delmas
Romain Poirot
Paul Witz

Etude des reseaux de neurones en perceptrons
avec application au concept des Generative Adverserial Network

Années 2017-2018

Dessines-moi un mouton
Genrative Adversial Network

Résume

Resume

Mots clefs

Mots clefs

Table des matières

Chapitre 1

Introduction aux réseaux de neurones et premières applications

1.1 Outils utilisés pour le projet

Description des outils mis en place pour le projet et de certains choix techniques.

1.2 Réseaux de neurones par couches : le perceptron

1.2.1 Structure du réseau

1.2.2 apprentissage par rétropropagation

La rétro-propagation des erreurs est un type d'apprentissage supervisé pour les perceptrons multicouches. Il consiste à calculer l'influence de chaque paramètre sur la sortie et à les mettre à jour en fonction de cette influence.

Les paramètres que l'on fait évoluer sont les poids et les biais. La formule de mise à jour est la suivante :

$$W(t+1) = W(t) + \eta \frac{\partial E}{\partial W}$$

avec η le pas de convergence, $\frac{\partial E}{\partial W}$ la matrice de terme général $\frac{\partial E}{\partial W_{i,j}}$
Pour pouvoir mettre à jour les poids, il faut donc calculer les $\frac{\partial E}{\partial W_{i,j}}$.
A la couche k l'influence des poids est donnée par :

$$\frac{\partial E^p}{\partial W_k} = \frac{\partial F}{\partial W}(W_k, X_{k-1}) \frac{\partial E^p}{\partial X_k}$$

1.3 Application au problème du XOR

Lorsque l'on souhaite travailler sur des algorithmes d'apprentissages par ordinateur il est recommandé de les essayer sur des problèmes connus afin d'en vérifier les performances.
Le problème du XOR est l'un des plus classiques car il apporte de nombreuses difficultés.

L'objectif du XOR est de séparer le plan complexe en quatres cadrants, $(x > 0, y > 0)$, $(x > 0, y < 0)$, $(x < 0, y > 0)$ et $(x > 0, y < 0)$. On restreint le plan à $[-1; 1]^2$. Les sorties attendues par le réseau de neurones sont 1 pour les points tel que $x * y > 0$ et -1 pour les points tels que $x * y < 0$.
Le premier intérêt de ce problème est qu'il est non linéaire, c'est à dire que l'on ne peut pas tracer une droite séparant le plan en 2 qui répond à celui-ci.

C'est en se basant sur la résolution du XOR que nous avons construits notre structure de réseau et vérifié la cohérence de notre code. La littérature propose comme réseau le plus simple pour ce problème une couche cachée de 2 neurones, avec 2 entrées (x et y) et 1 sortie dans $[-1, 1]$. Nous avons étudié également quelques autres formes de réseaux pour comparer les résultats.

Notion de résultats La notion de résultats nécéssite d'être correctement défini afin de pouvoir être interprété correctement, et surtout comparé à d'autres résultats obtenus par nous même ou par d'autres personnes.
Dans le cas des ses apprentissages, le réseau classe les objets que l'on donne en entrée. Généralement l'on défini le résultat par rapport à un pourcentage de succès dans cette classification. Pour l'obtenir on commence par définir une erreur relative, c'est à dire une distance entre la sortie cible et la sortie obtenue. Puis l'on applique un seuil afin de définir si oui ou non le réseau à correctement classifié l'entrée.
Dans le cas du XOR on met en place un seuil de 0.5.
On cherche également à évaluer la vitesse d'apprentissage, ainsi on calcule le pourcentage de succès du réseau à intervalle régulier au cours de l'apprentissge. Les réseaux étant soumis à une forte composante aléatoire (l'ordre d'apprentissage, ainsi que l'initialisation des poids), on effectue des apprentissages dans les mêmes conditions plusieurs fois afin d'obtenir des courbes moyennes, et des intervalles de confiances justifiant nos résultats.

Réseau en $2 \to 2 \to 1$ Les résultats obtenus au début sur ce réseau le plus simple semblait tout à fait aléatoire et nous on permit de détecter des erreurs de traductions des équations de rétropropagations en code Python. Nous avons finalement pu obtenir des résultats satisfaisants comme le montre la figure **??**. Cependant ce résultat n'était pas obtenu dans l'intégralité des apprentissages, nous fournissant des résultats très différents, comme sur la figure **??**. La littératures et en particulier les rapports des années précédentes [**?**] et [**?**] nous on montré que dans le XOR n'était effectivement pas juste dans 100% des cas.
Nous avons donc établis soumis le réseau à de nombreux apprentissages, en faisant varier les paramètres ainsi que la forme du réseaux. Voici les résultats les plus intéressants :

Conclusion sur le XOR Instabilité du réseau $2 \to 2 \to 1$ et comparaison avec le $2 \to 4 \to 1$ et le $2 \to 2 \to 2 \to 1$
Pas d'apprentissage très petit par rapport à la littérature
Influence des fonctions d'activation

Bibliographie

[1] K. Steenbergen, F. Janssen, J. Wellen, R. Smets, T. Koonen, "Fast wavelength-and-time slot routing in hybrid fiber-access networks for IP-based services", in *IEEE LEOS Symposium*, Delft, The Netherlands, October 2000.

[2] K. Nichols, V. Jacobson, L. Zhang, "A two-bit differentiated services architecture for the Internet", *IETF RFC 2638*, July 1999.

[3] http ://www.omniorb.org