Pâris Douady

Projet de Vie Artificielle – Perceptron multicouches multi-classes probabiliste

Introduction

Le but de ce projet est de comprendre le fonctionnement de réseaux neuronaux simples, et de voir leur capacité de prédiction et de reconnaissance de différents types de données.
Dans notre cas nous étudierons 2 ensembles de données : MNIST (Chiffres) et les Silhouettes Cal101.

I. Présentation du modèle

L'architecture de base utilisée est un réseau à 4 couches :

- 784 neurones d'entrées
- 1000 neurones intermédiaires
- Activation Sigmoïde
- 100 neurones intermédiaires
- Activation Sigmoïde
- 10 neurones de sorties (ou 101 dans le cas de Cal101)
- Softmax + Entropie Croisée

Nous étudierons ensuite diverses modifications de cette architecture pour comprendre la dynamique de ce type de réseau, ou afin d'aider à la visualisation. Par exemple en changeant l'activation, la fonction de coût ou le nombre de neurones intermédiaires.

L'architecture du code adoptée est inspiré de la librarie Keras en python, c'est à dire que chaque type de « couche » possède une classe associée qui calcule indépendamment la propagation avant et arrière. Ces couches incluent les activations, la fonction softmax, les fonctions de coût etc…
Toutes les couches se trouvent dans le package « layers ». Le package layers.activations contient les activations, le package layers.flat contient les couches « classiques » telle que la transformation affine, la normalisation par batch (Batchnorm) etc.. Enfin, le package layers.loss contient les fonctions de coût.

A noter que la classe la plus utile est la classe « math.Matrix » qui permet de simplifier l'écriture du code en le rendant plus facile à lire. Par exemple une multiplication d'une matrice par sa transposée s'écrit simplement Matrix B = A.T().mult(A) ;

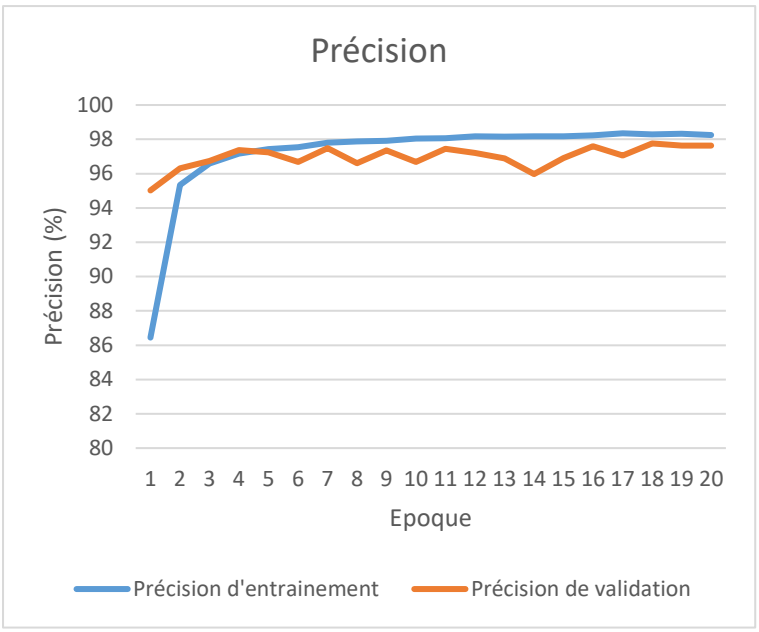
La stratégie adoptée pour optimiser les paramètres du réseau n'est pas simplement la descente de gradient par mini-batch mais l'optimisation « RMSProp ». En effet celle-ci converge plus rapidement à une solution correcte, et réduit donc le nombre d'époques nécessaires pour que le réseau trouve son optimum local. Le principe est de faire une moyenne mobile de la magnitude de chaque paramètre, puis de les diviser par cette magnitude.

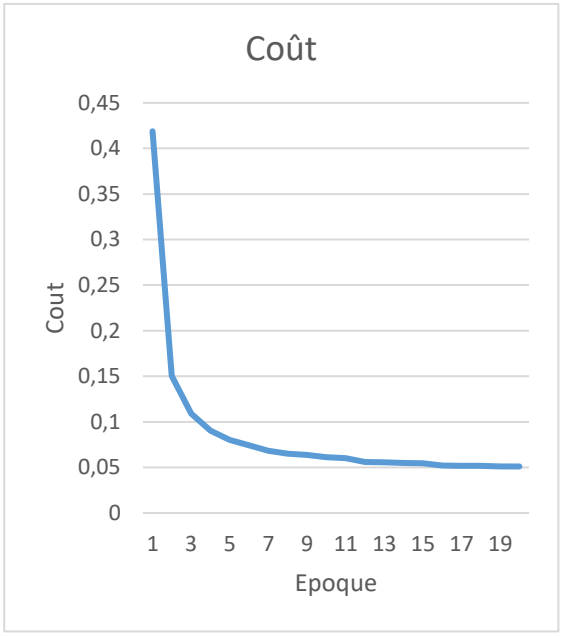
La régularisation utilisée pour éviter le surapprentissage est la régularisation L2, c'est-à-dire que la somme des carrés des éléments des matrices de poids multiplié par un coefficient est ajoutée au coût.

II. Résultats et analyse

Pour la base de données MNIST, le meilleur résultat obtenu sur les 10.000 données de test après avoir fait une recherche d'hyper paramètres est de 97,70% de précision. A noter que le code de la classe MainMnistForward devrait reproduire exactement ce résultat.

Voici quelques graphes sur le déroulement de l'entraînement :





On peut voir le coût ne fais que descendre, c'est donc que la descente de gradient s'effectue correctement, et la précision d'entrainement ne fais que monter, sans rester trop loin de la précision de validation et en atteignant pas les 100%. Il n'y a donc pas de surentraînement grâce à la régularisation.

Voici aussi la matrice de confusion des données d'entraînement pour le modèle présenté ci-dessus :

Chiffre	0	1	2	3	4	5	6	7	8	9	Correct
0	4914	0	10	2	1	7	7	0	2	3	
1	2	5656	5	3	6	2	6	10	6	3	
2	1	6	4918	19	1	2	2	21	4	0	
3	1	2	4	4987	0	9	0	0	1	11	
4	0	2	3	1	4834	1	6	3	0	8	
5	1	0	1	32	0	4439	4	0	0	3	
6	3	0	5	0	2	6	4919	0	6	0	
7	0	5	6	4	2	3	0	5127	0	39	
8	8	7	13	46	1	32	7	11	4823	22	
9	2	0	3	7	12	5	0	3	0	4899	

Prédiction

On peut voir que les 4 erreurs les plus communes sont de prédire un 8 à la place d'un 3, de prédire un 7 à la place d'un 9, de prédire un 8 à place d'un 5 et enfin de prédire un 5 à la place d'un 3.

On peut aussi grâce à cette matrice calculer la précision et la sensibilité (recall) pour chaque chiffre :

Chiffre	0	1	2	3	4	5	6	7	8	9
Précision	99,64%	99,61%	98,99%	97,77%	99,49%	98,51%	99,35%	99,07%	99,61%	98,22%
Sensibilité	99,35%	99,25%	98,87%	99,44%	99,51%	99,08%	99,55%	98,86%	97,04%	99,35%

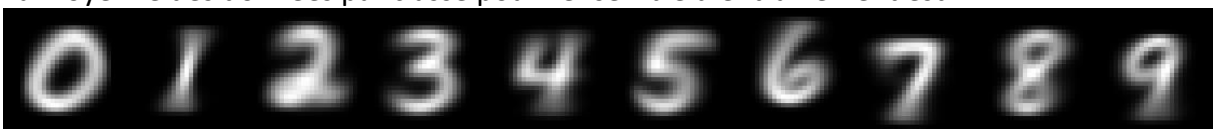
On peut voir que le chiffre 3 est peu précis, mais l'on est très sûr que c'est un 3 si il est prédit en tant que tel. En revanche le chiffre 8 a une haute précision mais on ne peut pas être entièrement sûr que c'est bien un 8 si un 8 est prédit.
Autrement dit, les classes les plus problématiques qui ressortent sont dans l'ordre la classe 3 (faible précision), 8 (faible sensibilité) et 9 (faible précision) et 5 (faible précision & sensibilité). Cela semble logique dût à leurs ressemblances.

Les données les plus difficile de l'ensemble d'entraînement (trié par plus faible probabilité de la vraie classe) sont :

1 -> 2 8 -> 9 7 -> 2 4 -> 9 1 -> 7 1 -> 9 8 -> 6 1 -> 7 4 -> 6 0 -> 6

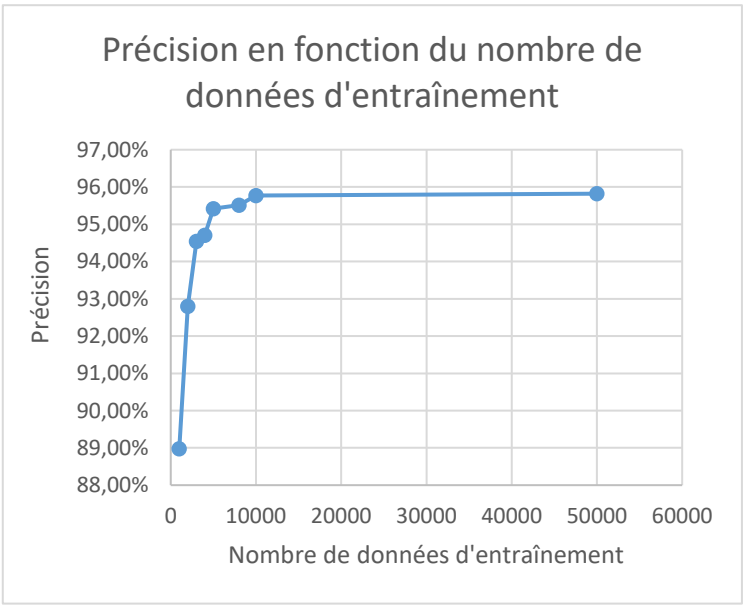
Le premier chiffre correspond à la classe prédite par le réseau, et le second chiffre la véritable classe.

La moyenne des données par classe pour l'ensemble d'entraînement est :



On reconnait bien les chiffres, c'est un autre indicateur que le réseau classifie bien les images.

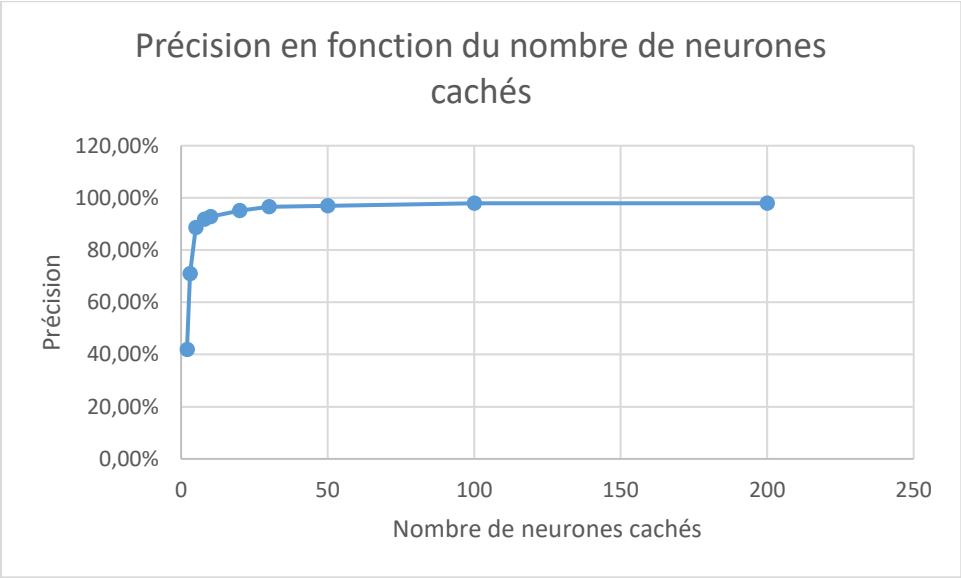
On peut aussi regarder le score à l'ensemble de test en fonction du nombre de données d'entraînement avec la même architecture au bout du même nombre de mise à jour :



On a donné au réseau le droit de regarder 50000 images en tout, c'est-à-dire qu'avec 1000 images d'entraînement 50 époques sont effectués, et avec 50000 images 1 seul époque est effectuée.

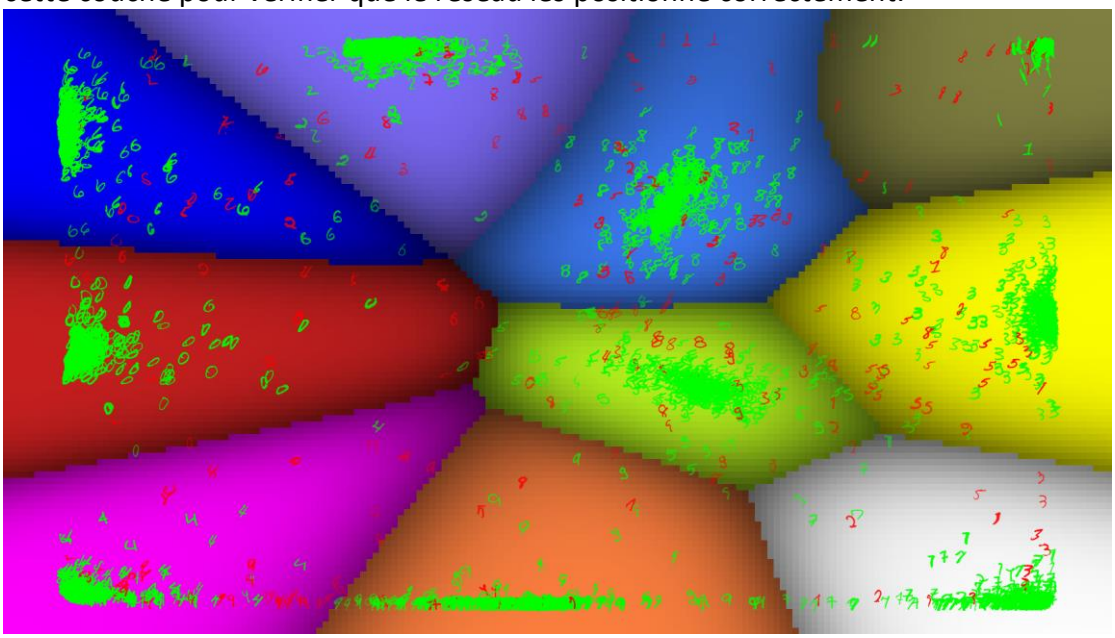
On voit ce à quoi on s'attends, une progression logarithmique en fonction du nombre de données d'entrainement.

Nous pouvons ensuite regarder l'intérêt de la couche cachée, en ne faisait qu'un réseau à 3 couches en augmentant le nombre de neurones cachées. On met ici 16000 données d'entraînement et 5 époques d'entraînement.



Pour la base de données Cal101 Silhouettes, le meilleur résultat obtenu sur les 1000 données de test après une recherche d'hyper paramètres est de 58,70%. Le code de la classe MainCalForward devrait reproduire ce résultat.

En bonus, voici une visualisation d'un réseau à l'architecture suivante : 784-1000-2-100-10. Sur cette image, on visualise tous les points de la 3ème couche, chaque couleur correspond à la classification d'un nombre en fonction de son activation (les axes X et Y correspondent à la valeur de l'activation de chaque neurone) pour cette couche et son intensité correspond à la confidence de la prédiction. On ajoute aussi le mapping d'un certain nombre de données sur cette couche pour vérifier que le réseau les positionne correctement.



Les chiffres en rouge sont les chiffres classifiés incorrectement et les chiffres en vert sont classifiés correctement.