INF8225 – Intelligence artificielle  
Techniques probabilistes et d’apprentissage

Travail pratique 2  
Réseau de neurones

Carl-Vincent Landry-Duval - 1748935  
Luc Coubariaux - 1670433

Soumis à : Alexandre Piché

18 février 2018

# Partie I

1. Pseudocode pour l’algorithme de rétropropagation :

Quelques définitions de constantes nécessaires :

* Posons : Largeur du réseau de neurone. Prédéfini comme étant 100 unités;
* Posons L : Nombre de couches cachées du réseau de neurone;
* Posons : Nombre de cycles d’apprentissage réalisés;
* Posons : Matrice de poids de dimension pour la couche ;
* Posons : Matrice de poids de dimension pour la couche ;
* Posons : Vecteur de valeurs d’entrée de dimension pour la couche ;
* Posons : Vecteur avec 1 ajouté à la fin de dimension pour la couche ;
* Posons : Vecteur de valeurs d’entrée initial de dimension . Synonyme de ;
* Posons : Valeur attendue en sortie pour le vecteur d’entrée ;
* Posons : Taux d’apprentissage du réseau de neurone;
* Posons : Rapport entre la taille de l’ensemble d’entraînement et de la taille l’ensemble destiné à l’apprentissage.

Quelques définitions de fonctions :

Pour des fins de simplicité, nous considérons que la fonction exponentielle appliquée à un vecteur applique la fonction exponentielle sur chaque élément du vecteur. Les opérations arithmétiques élémentaires tel que l’addition, la soustraction, la multiplication et la division s’appliqueront de façon similaire. Pour distinguer la multiplication du produit matriciel, l’opérateur sera utilisé pour indiquer le produit matriciel.

* Posons : Fonction de calcul de la perte globale;
* Posons : Fonction pour obtenir la réponse;
* Posons : Fonction pour obtenir le vecteur ;

Calcul du gradient de la perte :

Pseudocode rétropropagation :

* Poser  : Ensemble d’entraînement aléatoirement ordonné formé par les paires
* Poser  : Les premiers éléments de ;
* Poser  : Ensemble des matrices crées avec des poids aléatoires, incluant le vecteur à sa fin;
* Répéter pour un nombre d’époques :
  + Pour chaque dans faire :
    - ;
    - Pour allant de 1 à :
      * ;
    - ;
    - ;
    - ;
    - Pour j allant de à 1 :
      * ;
      * ;
    - ;

1. Optimisations possibles :

Afin d’optimiser le traitement d’un grand nombre de données d’entraînement, il est possible d’utiliser plusieurs techniques :

1. Utiliser les minibatchs afin de paralléliser l’apprentissage tout en réduisant le nombre de modifications aux poids du réseau de neurones par cycles;
2. Paralléliser les multiplications de matrices, vecteurs et l’application de la fonction de régression logistique;
3. Utiliser une architecture réseau distribuée et/ou utiliser des cartes graphiques afin d’accélérer le parallélisme;
4. Réduire le nombre d’époques tout en augmentant le taux d’apprentissage;
5. Réduire le nombre de couches cachées du réseau de neurones;

# Partie 2

Nous avons choisi de noter nos architectures de la manière suivante [<taille d'entrée=784>, <taille des couches intermédiaires, >, <taille de sortie=10>].

Au cours de nos expériences, nous avons remarqué que les réseaux de taille plus petite apprennent plus vite mais ont tendance à avoir une précision finale inférieure. Un exemple est celui du réseau sans couche intermédiaire [784, 10], mais on observe cela aussi via la progression de [784, 64, 10], [784,512,10], [784,1024,10], de précision respective 87.8%, 89.1%, 89.1%, et les réseaux à plusieurs couche intermédiaire comme [784, 64, 64, 10] et [784, 512, 512, 10]. On peut supposer que les réseaux ne respectant pas cette tendance ont besoin d'un entrainement plus long.

D'autre part les tailles de couches inégales ([784, 512, 64, 10], [784, 512, 1024, 10], [784, 1024, 512, 10]) ne semblent pas meilleurs ou inférieurs à ceux de couches intermédiaires de taille égale mentionnés plus tôt.

# Annexe A – Graphes de résultats

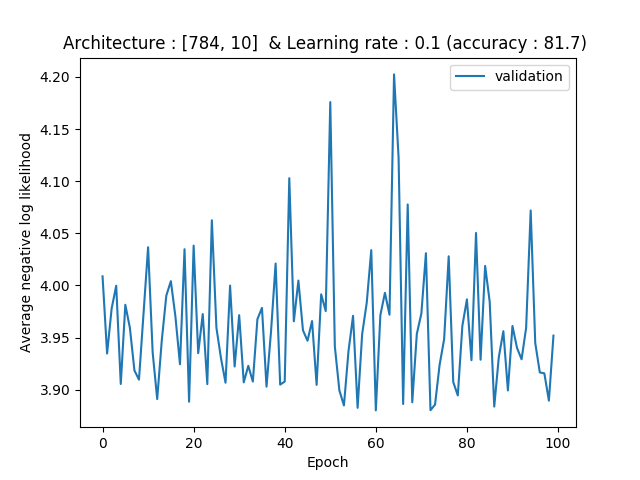


Figure : Log vraisemblance pour un réseau de neurone sans couches cachées et un taux d'apprentissage de 0.1

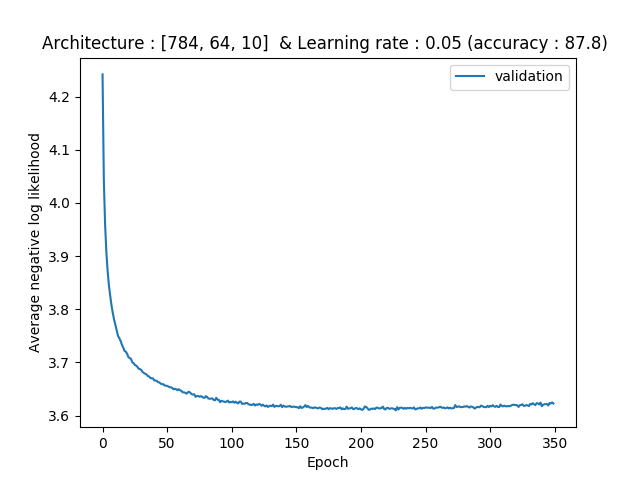


Figure : Log vraisemblance pour un réseau de neurone avec une couche cachée de 64 neurones et un taux d'apprentissage de 0.05

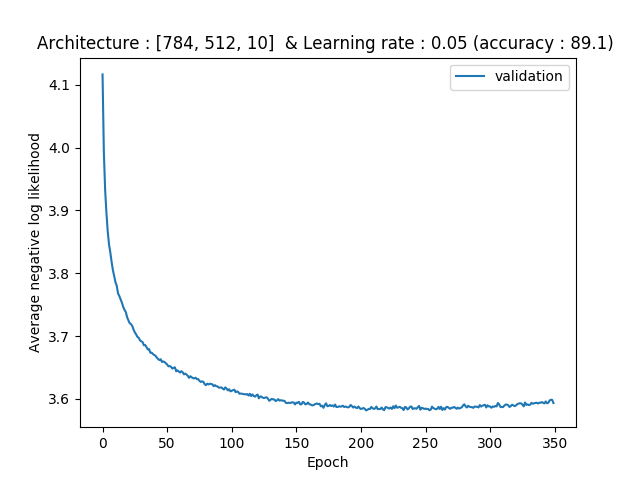


Figure : Log vraisemblance pour un réseau de neurone avec une couche cachée de 512 neurones et un taux d'apprentissage de 0.05

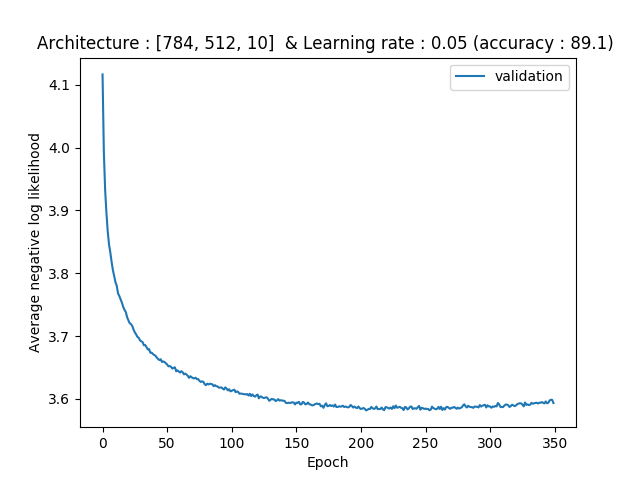


Figure : Log vraisemblance pour un réseau de neurone avec une couche cachée de 1024 neurones et un taux d'apprentissage de 0.05

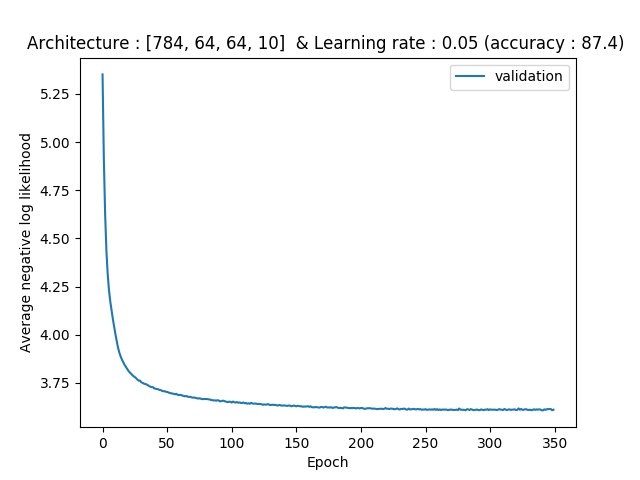


Figure : Log vraisemblance pour un réseau de neurone avec deux couches cachées de 64 neurones et un taux d'apprentissage de 0.05

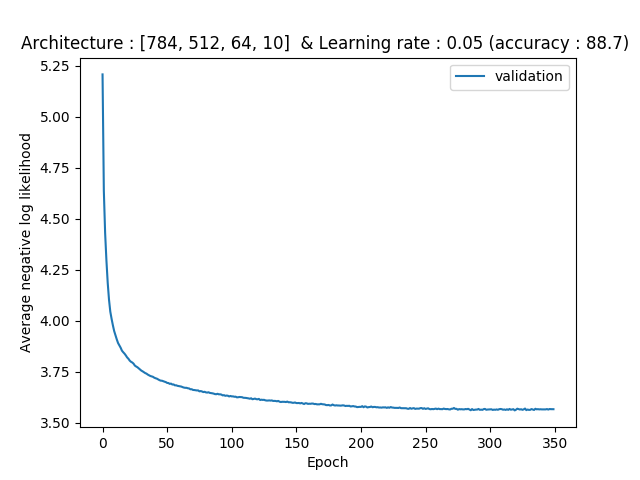


Figure : Log vraisemblance pour un réseau de neurone avec deux couches cachées de 512 et 64 neurones et un taux d'apprentissage de 0.05

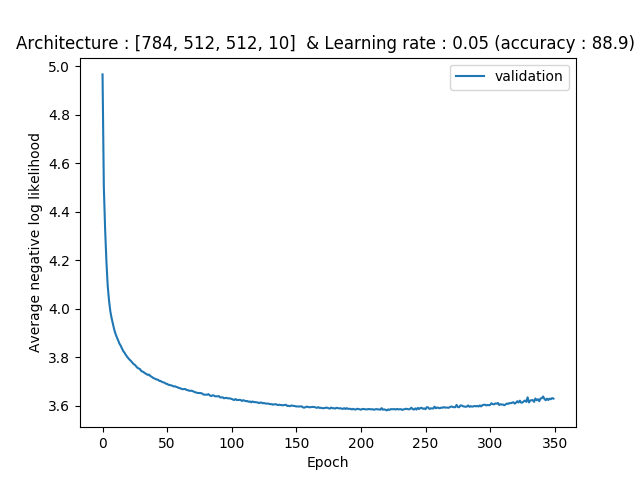


Figure : Log vraisemblance pour un réseau de neurone avec deux couches cachées de 512 neurones et un taux d'apprentissage de 0.05

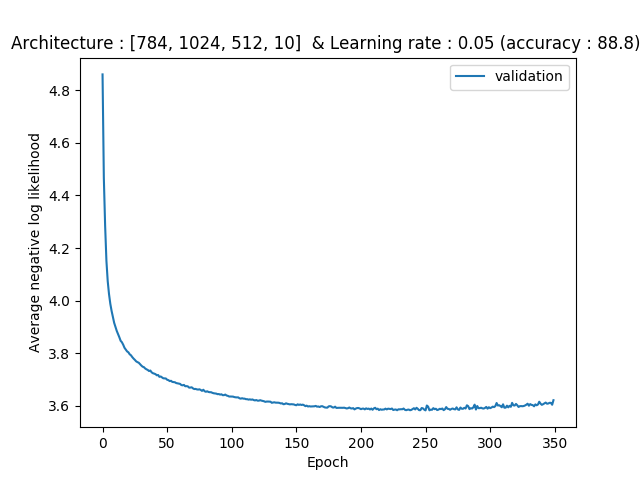


Figure : Log vraisemblance pour un réseau de neurone avec deux couches cachées de 1024 et 512 neurones et un taux d'apprentissage de 0.05

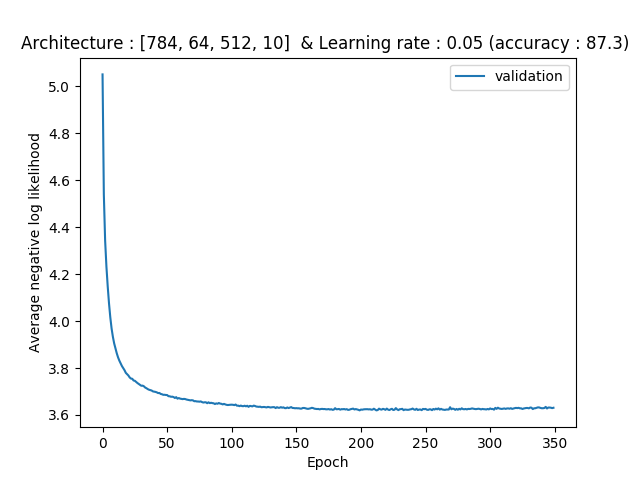


Figure : Log vraisemblance pour un réseau de neurone avec deux couches cachées de 64 et 512 neurones et un taux d'apprentissage de 0.05

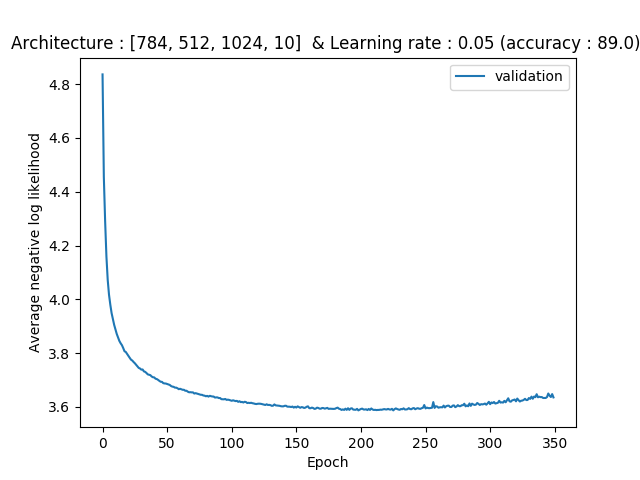


Figure : Log vraisemblance pour un réseau de neurone avec deux couches cachées de 512 et 1024 neurones et un taux d'apprentissage de 0.05