État d'avancement

Projet Informatique Individuel – Émilie ROGER
ENSC – Promotion 2018

1. Travail préliminaire

1.1. Prise en main de Python

J'ai commencé par télécharger WinPython sur mon ordinateur. Il s'agit de la version 3.5.2.3Qt5 disponible sur le site *WinPython* (http://winpython.github.io).

Ensuite, j'ai utilisé le tutoriel « Apprenez à programmer en Python » disponible sur le site *OpenClassrooms* (https://openclassrooms.com). Je l'ai effectué à 55 % afin de savoir comment écrire les classes, les boucles et les fonctions.

1.2. Prise en main de TensorFlow

J'ai débuté par l'installation de TensorFlow en suivant les instructions disponibles sur le site *TensorFlow* (https://www.tensorflow.org).

Sur ce site, j'ai pu également suivre les tutoriels « Getting Started With TensorFlow » et « MNIST For ML Beginners ».

Le premier m'a permis d'apprendre à utiliser TensorFlow dans les grandes lignes. Le second m'a permis de mieux appréhender le « Machine Learning » et d'utiliser la base de données MNIST (Mixed National Institute of Standards and Technology) qui contient des chiffres écrits de façon manuscrite.

1.3. Spécifications

Pour déterminer les spécifications générales et détaillées de mon projet, je suis partie des exigences définies dans mon cahier des charges.

1.3.1. Organisation du code

J'ai conceptualisé une classe principale « Network » qui auraient deux héritiers : une pour les réseaux complétement connectés et une pour les réseaux convolués.

Au niveau des attributs de notre classe, celle-ci aurait au moins une liste de couches et une matrice contenant les poids des neurones de chaque couche.

Au niveau des méthodes de notre classe, cette dernière aurait un constructeur, une méthode pour déterminer les poids des neurones du réseau à partir de la couche d'entrée et la couche de sortie théorique, et une méthode qui déterminerait la couche de sortie prévue à partir d'une couche d'entrée donnée.

1.3.2. Interactions avec l'utilisateur

Au niveau des interactions, l'utilisateur pourra définir quel type de réseau il souhaite, notamment s'il veut un réseau complétement connecté ou convolué, combien il veut de couches dans son réseau et quelles sont les dimensions des couches. Il pourra également entraîner son réseau, puis une fois entraîné, le tester.

Toutes les interactions se feront *via* la console de Python, une fois la bibliothèque importée. Les données d'entrées et de sortie seront simples, comme des opérateurs logiques, par exemple.

1.3.3. Fonctionnalités

Mon programme est simplement destiné à prévoir une sortie à partir d'une entrée donnée. Pour cela, il suffit de définir un réseau de neurones et de l'entraîner un certain nombre de fois.

Les fonctionnalités se résument donc à l'entraînement et à la prévision.

1.4. Autres

Pour une meilleure compréhension de mon projet, j'ai lu les 2 articles envoyés par mon tuteur de stage et j'ai parcouru les 2 sites web qu'il m'avait conseillé d'aller voir. J'ai également demandé à mon tuteur de stage quelques précisions pour commencer le codage et la validation de mon cahier des charges.

Par ailleurs, j'ai fait quelques recherches, notamment sur le perceptron et les réseaux convolués. Pour finir, j'ai participé à la conférence du 2 mars 2016, organisée par l'IRASCA dans le cadre de la journée nationale de l'intelligence artificielle proposée par FranceIA (http://www.economie.gouv.fr/France-IA-intelligence-artificielle).

2. Codage

2.1. Codage d'une couche

Le codage d'une couche complétement connectée a été effectué dans la première phase de codage. Pour cela, j'ai décidé de créer une classe « Layer ».

Cette classe fut très simple à mettre en œuvre. Pour l'instant, il s'agit juste d'un constructeur prenant en paramètres :

- le nom de la couche
- le nombre de neurones de la couche (c'est-à-dire la taille / dimension de la couche)
- la fonction d'activation de la couche (ici soit la fonction sigmoïde soit la fonction ReLU)
- l'intervalle initial des amplitudes pour les poids des connexions à la sortie des neurones de la couche

Le constructeur récupère ses paramètres et crée ainsi un dictionnaire. Celui-ci est accessible *via* la méthode « get » qui renvoie tel quel le dictionnaire.

2.2. Codage du réseau

Au niveau du réseau, le codage s'est principalement tourné vers des réseaux dans lesquels les couches sont complètement connectées. À ce jour, je n'ai pas encore commencé à coder pour des couches convoluées et n'ai donc pas de classe héritée.

Les attributs de la classe sont :

- le nombre de couches du réseau
- une liste contenant les fonctions d'activation de chaque couche
- une liste contenant les couches du réseau, c'est-à-dire la valeur de chaque neurone du réseau
- une liste contenant les poids des connexions entre chaque neurone de chaque couche
- une liste contenant les derniers termes d'erreur calculés pour les connexions entre chaque neurone de chaque couche

Les méthodes de la classe sont :

- un constructeur qui prend en argument une liste de dictionnaires, ces derniers correspondant aux couches du réseau
- une méthode qui « entraîne » le réseau, c'est-à-dire qui calcule les poids à donner aux connexions entre les neurones des couches de façon à ce que l'erreur soit la plus faible
- une méthode qui « teste » le réseau, c'est-à-dire qui calcule la sortie obtenue avec le réseau actuel

2.3. Tests

Actuellement, je n'ai testé mon programme que pour des opérateurs logiques. Par exemple, je donne comme entrée : [[[0, 0], [0]], [[0, 1], [1]], [[1, 0], [1]], [[1, 1], [0]]] correspondant à l'opérateur logique XOR.

Voici une capture d'écran montrant la sortie avant entraînement et après entraînement pour un XOR :

```
In [1]: runfile('C:/Users/Lewan/Documents/GitHub/pii-
neural-network/Codes/network.py', wdir='C:/Users/Lewan/
Documents/GitHub/pii-neural-network/Codes')
Avant :
[0, 0]  ->  [-0.01179719]
[0, 1]  ->  [-0.02113963]
[1, 0]  ->  [-0.01276967]
[1, 1]  ->  [-0.02274472]
Après :
[0, 0]  ->  [-0.00061914]
[0, 1]  ->  [ 0.99419879]
[1, 0]  ->  [ 0.99091372]
[1, 1]  ->  [ 0.01152627]
```

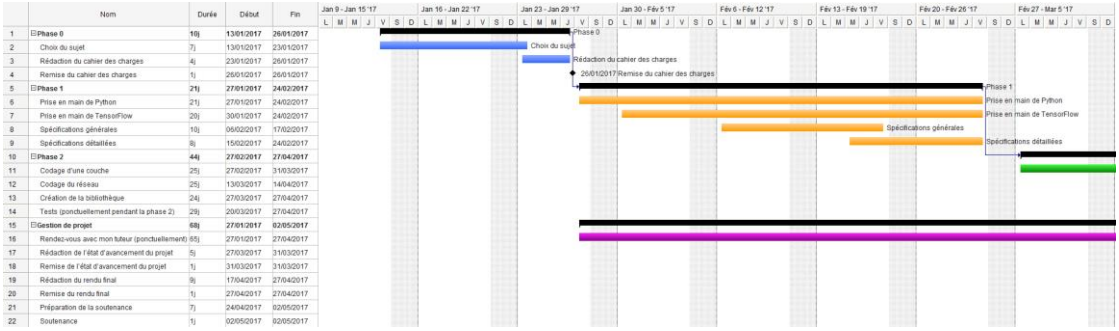
L'ensemble du code est disponible sur GitHub à l'adresse suivante :

https://github.com/remilieam/pii-neural-network

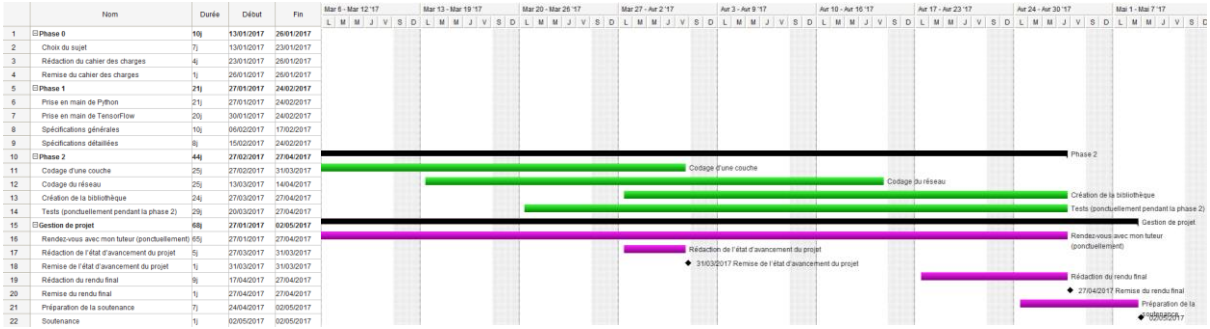
3. Bilan

Pour l'instant, le planning est respecté. La phase 1 et le codage de la couche sont terminés, comme prévu, et j'ai fait la moitié du codage du réseau : il me manque les réseaux convolués, mais j'ai *a priori* fini les réseaux totalement connectés, ce qui est cohérent avec mon planning.

Ma planification est donc toujours la même. Je n'ai donc effectué aucune modification dans le planning initial, rappelé ci-dessous :



	Nom	Durée	Début	Fin
1	Phase 0	10j	13/01/2017	26/01/2017
2	Choix du sujet	7j	13/01/2017	23/01/2017
3	Rédaction du cahier des charges	4j	23/01/2017	26/01/2017
4	Remise du cahier des charges	1j	26/01/2017	26/01/2017
5	Phase 1	21j	27/01/2017	24/02/2017
6	Prise en main de Python	21j	27/01/2017	24/02/2017
7	Prise en main de TensorFlow	20j	30/01/2017	24/02/2017
8	Spécifications générales	10j	06/02/2017	17/02/2017
9	Spécifications détaillées	8j	15/02/2017	24/02/2017
10	Phase 2	44j	27/02/2017	27/04/2017
11	Codage d'une couche	25j	27/02/2017	31/03/2017
12	Codage du réseau	25j	13/03/2017	14/04/2017
13	Création de la bibliothèque	24j	27/03/2017	27/04/2017
14	Tests (ponctuellement pendant la phase 2)	29j	20/03/2017	27/04/2017
15	Gestion de projet	68j	27/01/2017	02/05/2017
16	Rendez-vous avec mon tuteur (ponctuellement)	65j	27/01/2017	27/04/2017
17	Rédaction de l'état d'avancement du projet	5j	27/03/2017	31/03/2017
18	Remise de l'état d'avancement du projet	1j	31/03/2017	31/03/2017
19	Rédaction du rendu final	9j	17/04/2017	27/04/2017
20	Remise du rendu final	1j	27/04/2017	27/04/2017
21	Préparation de la soutenance	7j	24/04/2017	02/05/2017
22	Soutenance	1j	02/05/2017	02/05/2017



	Nom	Durée	Début	Fin
1	Phase 0	10j	13/01/2017	26/01/2017
2	Choix du sujet	7j	13/01/2017	23/01/2017
3	Rédaction du cahier des charges	4j	23/01/2017	26/01/2017
4	Remise du cahier des charges	1j	26/01/2017	26/01/2017
5	Phase 1	21j	27/01/2017	24/02/2017
6	Prise en main de Python	21j	27/01/2017	24/02/2017
7	Prise en main de TensorFlow	20j	30/01/2017	24/02/2017
8	Spécifications générales	10j	06/02/2017	17/02/2017
9	Spécifications détaillées	8j	15/02/2017	24/02/2017
10	Phase 2	44j	27/02/2017	27/04/2017
11	Codage d'une couche	25j	27/02/2017	31/03/2017
12	Codage du réseau	25j	13/03/2017	14/04/2017
13	Création de la bibliothèque	24j	27/03/2017	27/04/2017
14	Tests (ponctuellement pendant la phase 2)	29j	20/03/2017	27/04/2017
15	Gestion de projet	68j	27/01/2017	02/05/2017
16	Rendez-vous avec mon tuteur (ponctuellement)	65j	27/01/2017	27/04/2017
17	Rédaction de l'état d'avancement du projet	5j	27/03/2017	31/03/2017
18	Remise de l'état d'avancement du projet	1j	31/03/2017	31/03/2017
19	Rédaction du rendu final	9j	17/04/2017	27/04/2017
20	Remise du rendu final	1j	27/04/2017	27/04/2017
21	Préparation de la soutenance	7j	24/04/2017	02/05/2017
22	Soutenance	1j	02/05/2017	02/05/2017

4. Annexes

Pour mieux comprendre les réseaux de neurones complétement connectées (perceptron multi-couches) et savoir comment les coder, j'ai fait un document montrant les calculs utilisés. Celui-ci est joint en annexe de ce document (extension pdf).

Par ailleurs, les classes « Network » et « Layer » sont également jointes au projet (extension python).