4 MARS 2018

Travail Pratique 1
-
Mon application en Intelligence Artificielle

Travail individuel

ÉRIC COGOLUENHÈS
NI : 111162638

Introduction

L'objectif principal de ce travail est d'approfondir les concepts vus en cours autour de la définition de l'intelligence artificielle (IA), la reconnaissance d'une d'entres-elles et la compréhension basique de son fonctionnement .
Le choix du logiciel pour réaliser cette activité s'est porté sur le « playground TensorFlow » qui est une petite librairie de réseaux de neurones qui a un but éducatif. L'application utilisée dans le « monde réelle », avec les librairies complètes de réseaux de neurones est la librairie « TensorFlow ».
L'identification de cette application web, directement utilisable depuis son navigateur de recherche internet, m'est venue de la conférence de Hugo Larochelle (chercheur, responsable de Google Brain, Google) sur l'« Intelligence artificielle : la révolution de l'apprentissage profond » lors des 3 jours organisés par l'ITIS (Institut Technologique de l'information et sociétés) : Objectif numérique 2017 - vers des sociétés numériques et durables.
En effet, il avait appuyé sa démonstration sur cette application accessible et éducative sur l'apprentissage profond.
Cet écrit se décomposera en trois parties. La première permettra d'identifier en quoi cette application utilise l'intelligence artificielle et, si oui, quelle technique de celle-ci. La seconde partie décriera comment l'intelligence artificielle a permis de résoudre le problème proposé par l'application. Enfin, la troisième partie aura pour objectif d'expliquer le(s) principe(s) et le fonctionnement de la technique d'intelligence artificielle utilisée dans l'application en s'appuyant sur quelques exemples.

1. En quoi consiste l'intelligence artificielle dans cette application ?

Comme rapidement évoqué ci-dessus, « Playground TensorFlow » est une visualisation d'une petite librairie de réseaux de neurones. Elle n'est qu'un aperçu, simplifié à des fins éducatives, de la librairie complète « TensorFlow ». Cette dernière est utilisée par plusieurs compagnies comme DeepMind ou encore Intel.
Comme évoqué précédemment, elle est utilisable depuis un navigateur de recherche internet. Cette application web permet de réaliser des simulations à partir de cette petite librairie de réseaux de neurones.

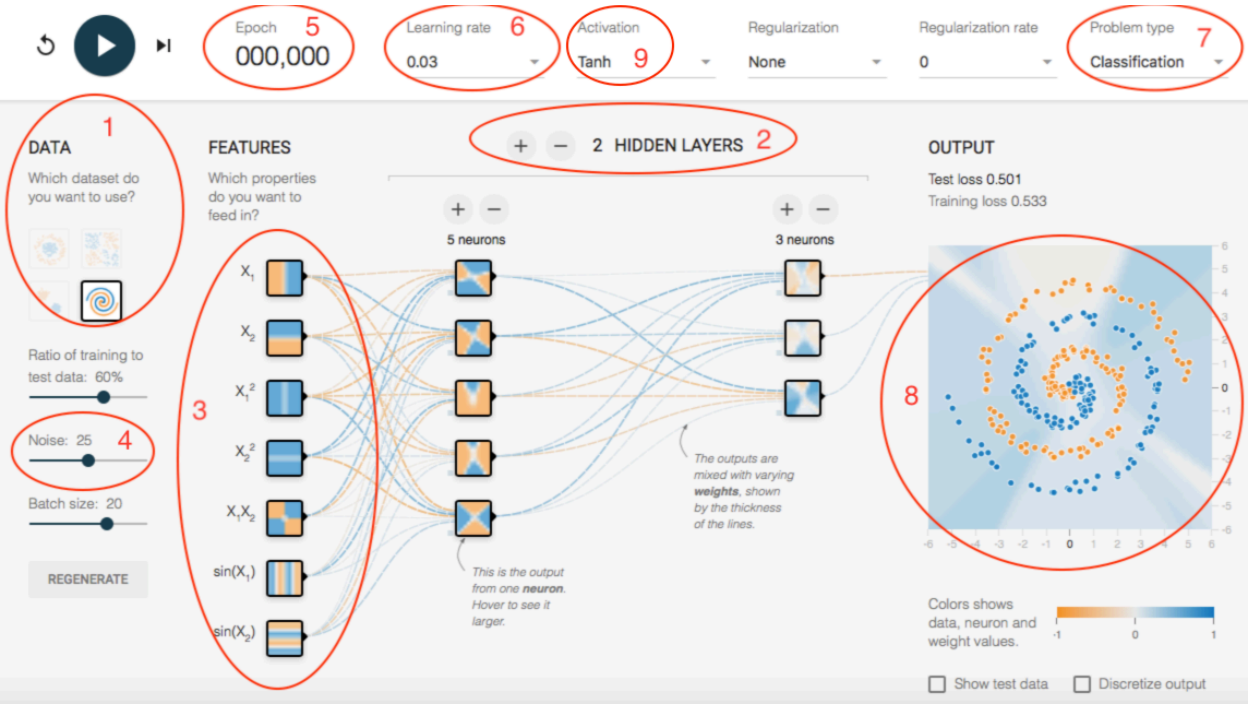

IMAGE 1 - COPIE ÉCRAN DU SIMULATEUR « PLAYGROUND TENSORFLOW »

Comme on peut le voir sur la copie écran ci-dessus, le simulateur permet de jouer avec plusieurs paramètres, parmi lesquels :

1. les données à utiliser,
2. le nombre de couches cachées de réseaux de neurones artificiels à utiliser,
3. La couche d'entrée avec les neurones d'entrée,
4. le niveau de bruit, ou plutôt le nombre de parasites dans nos données,
5. le nombre de récurrences ou de tests,
6. le taux d'apprentissage,
7. le type de problème à résoudre,
8. la visualisation du résultat de test, la couche de sortie,
9. le choix de technique heuristique à utiliser pour résoudre le problème, qui permet de définir les neurones artificiels des couches cachées (image 1, référence 2).

Ce simulateur se fonde sur une technique d'intelligence artificielle qui s'appelle l'apprentissage profond et sur une approche par réseaux de neurones.

Cette technique consiste principalement à construire un programme informatique qui apprend à partir des données qui lui sont fournies. Son modèle est basé sur la façon dont nous imaginons que le cerveau humain fonctionne.
Pour cela, il faut d'abord créer et connecter un ensemble de neurones artificiels (logiciels et/ou opérations mathématiques) qui peuvent communiquer ensemble. Ensuite, un problème est soumis au réseau de neurones artificiels. Le réseau tente de le résoudre un grand nombre de fois et, à chaque test de résolution, le système renforce les connections qui conduisent à des succès et diminue ceux qui mènent à des échecs.

2. Pourquoi a-t-on utilisé l'intelligence artificielle ?

Cette simulation permet de résoudre différents problèmes en fonction des données utilisées (image 1, référence 1) et du type de problème (image 1, référence 7) choisi. Et cela avec la possibilité de déterminer et de changer les paramètres à utiliser pour le résoudre (image 1, références 2 à 6 et 9).

On aurait pu résoudre ces problèmes de manière totalement aléatoire, par exemple avec la recherche en profondeur ou la recherche en largeur, mais cela aurait demandé beaucoup plus de temps et de puissance de calcul.

Si on essaie de représenter ce problème par un espace d'états, on aurait :
- notre ensemble de noeuds qui seraient nos neurones artificiels : N = {neurone N1, neurone N2, …, neurone Nx}
- Nos opérations seraient les connections positives ou négatives entre les neurones de différentes couches de neurones (d'entrée, cachées et de sortie). En effet, l'une des règles serait que les opérations ne sont possibles qu'entre deux neurones de couches différentes. A = {(X1,N1C2), (X1, N2C2), (X2, N1C2), etc.}
- Les neurones d'entrée seraient plusieurs noeuds de départ (image 1, référence 3), puisque chaque neurone d'entrée peut être utilisé pour commencer le test. S = {X1, X2, etc.}
- L'état final est le résultat du test ou encore la neurone de sortie (image 1, référence 8), résultante de tous les cas positifs ou négatifs. GD = *Output*.

Dans l'état final (la couche de sortie - image 1, référence 8), les points sont colorés en orange ou en bleu en fonction de leurs valeurs d'origine. La couleur d'arrière plan indique ce que le réseau prédit pour une zone particulière. L'intensité de la couleur montre à quel point la prédiction est confiante.

Ici la technique utilisée est une technique de recherche heuristique grâce à laquelle on va tester différents espaces d'états afin de définir l'état final le plus proche possible du jeu de données fournies au départ. En effet, lors de chaque test, à chaque noeud ou neurone, on va identifier les opérations (ou connections) positives (en bleu sur la simulation), c'est à dire qui ont le plus de chance de mener à un succès. À chaque neurone (noeud), une valeur pourra être attribuée aux différentes opérations (liens) possibles en fonction de leur poids positif ou négatif. Cela va permettre d'estimer sa valeur pour aller du noeud courant à l'état final (celui qui sera le plus susceptible de mener à une solution positive).

Cette technique peut sembler proche des techniques vues en cours, malgré quelques différences fondamentales.
En effet, pour le Hill-Climbing (simple et le plus escarpé), le meilleur d'abord et la technique A*, il faut choisir la meilleure opération menant à un état meilleur que l'état courant.
Pour cela, on se base sur une valeur heuristique pour chaque opération disponible à chaque noeud. Cette valeur est déterminée afin de définir l'évaluation de l'état suivant à l'état actuel et en fonction de sa capacité à nous mener le plus rapidement à l'état final.

Dans la simulation, l'idée est de prendre un grand nombre de tests qui vont passer par différents neurones (noeuds) et connections (opérations), jusqu'à l'état final.
À chaque test, le système va évaluer les connections et la couche de sortie (état final) positivement ou négativement en fonction de sa véracité (bleu si positif, orange si négatif, de plus en plus intense selon la confiance en la prédiction)

Ainsi dans notre simulation, le réseau de neurones utilise les tests pour automatiquement changer la valeur heuristique de ses connections et ainsi changer les règles et améliorer sa précision.

3. Comment est construite l'intelligence artificielle dans cette application ?

La couche d'entrée (image 2, référence 2) du réseau contient des neurones codant les valeurs des pixels d'entrée. Les pixels d'entrée sont l'ensemble des données utilisées pour la simulation (image 2, référence 1).

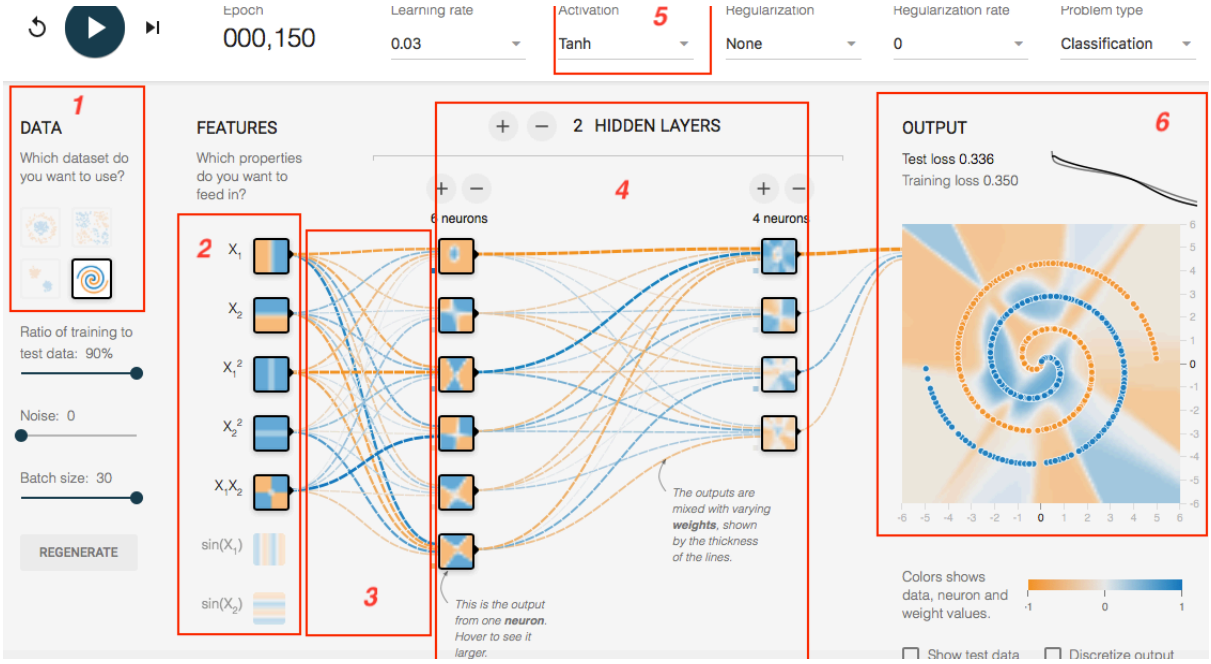

IMAGE 2 - COPIE ÉCRAN D'UNE NOUVELLE SIMULATION DANS « PLAYGROUND TENSORFLOW »

Nous pouvons remarquer que nous n'avons pas un neurone par pixel de l'image analysée, mais quatre neurones qui composent cette couche d'entrée (image 2, référence 2). En effet, le simulateur permet aussi de modifier le paramètre du nombre de neurones de la couche d'entrée (image 2, référence 2) en cliquant sur chacun d'entre eux pour les utiliser ou non.

Chaque neurone est une fonction à travers laquelle va être testé le pixel analysé en entrée.
Il pourra être testé positivement par la fonction (s'il est dans la zone bleue de la fonction) ou négativement (zone orange de la fonction). Les neurones d'entrée donnent une valeur de 1,0 représentant le bleu ou de -1,0 pour l'orange, selon l'endroit de la valeur de sortie avec la fonction qui représente le neurone. Plus la couleur est foncée sur la représentation visuelle de la fonction du neurone (image 2, référence 2) et plus la valeur de sortie est proche de -1,0 pour orange et 1,0 pour bleu. Entre ces valeurs, cela représente des nuances entre les deux couleurs et l'incertitude plus ou moins forte selon l'éloignement de la valeur de sortie de la fonction du neurone avec le -1,0 ou le 1,0.

Une fois que le pixel à analyser est passé par la couche d'entrée, la résultante de la fonction du neurone utilisé lors de cette couche d'entrée va passer par les couches cachées (image 2,

référence 4), en passant par des liens différents en fonction du poids de chacun de ceux-ci (image 2, référence 3). En effet, nous pouvons noter qu'entre les couches de neurones, chacun d'entre eux est lié par des lignes bleues ou oranges, plus ou moins épaisses et nuancées. Ces liens, tout comme les couches cachées de neurones, vont évoluer tout au long de la simulation, leur valeur heuristique permettant d'améliorer la précision du réseau de neurones après chaque test de pixel.

Les couches suivantes sont ce que l'on appelle les couches cachées (image 2, référence 4). Il n'est pas facile de résumer le processus de conception des couches cachées avec quelques règles simples. Elles sont déterminées par de nombreuses heuristiques de conception développées par des chercheurs en réseaux de neurones. Elles permettent d'obtenir le comportement souhaité pour les réseaux utilisés. Dans notre exemple de simulation, elles ont été déterminées par l'heuristique « Tanh » (image 2, référence 5) qui utilise la fonction tanh(x) (image 3).

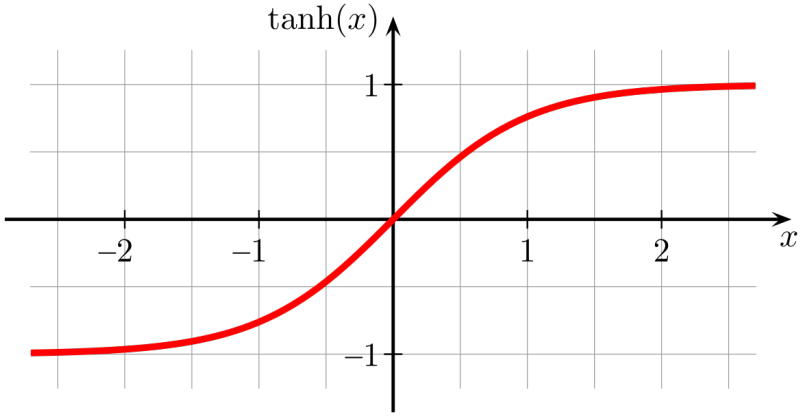

IMAGE 3 - FONCTION TANH(X)

La couche de sortie du réseau de neurones (image 2, référence 6) représente les données d'entrée, qui sont les points bleus ou oranges dépendant de leurs valeurs de base. La couleur en arrière plan évolue tout au long de la simulation. Elle varie en fonction de ce que le système de réseau de neurones prédit pour la zone testée. L'intensité de la couleur montre le niveau de certitude de la prédiction du système. Lorsque le système atteint un niveau de certitude proche de |1.0|, la couche de sortie n'évolue plus car tout nouveau test passe par les neurones et les connections qui lui assurent de trouver la bonne valeur en sortie.

Les réseaux de neurones sont utilisés dans de nombreux secteurs de notre vie.
Ils peuvent notamment être utilisés pour la reconnaissance des textes écrits manuellement par l'homme dans les machines de la poste qui lisent les adresses sur le courrier papier afin de les acheminer au bon endroit. Dans cet exemple, il est souvent utilisé beaucoup plus de neurones dans la couche d'entrée et énormément d'échantillons de tests afin que le réseau de neurones puisse améliorer son efficacité et tendre vers une certitude proche de 1.0. Les réseaux de neurones sont aussi très efficaces pour la reconnaissance d'images et la classification des images pour les recherches sur internet. Il faudra aussi un grand nombre de jeux de tests afin de s'approcher d'un niveau de certitude le plus proche possible de 1.0.

Conclusion

Comme nous avons pu le voir tout au long de cette analyse, le « Playground TensorFlow » est un outil de simulation très facile d'accès et pourtant suffisamment puissant pour permettre de mieux comprendre le fonctionnement et la dynamique d'un système de réseaux de neurones. Tout au long de notre travail, nous nous sommes efforcés d'expliquer son fonctionnement de manière accessible et synthétique. Cependant, nous aurions pu, si nous avions eu plus de temps essayer d'expliquer les différentes fonctions heuristiques qui peuvent être utilisées dans les couches cachées du réseau de neurones.

Références / Bibliographie

TensorFlow Playground :
http://playground.tensorflow.org/#activation=tanh&batchSize=30&dataset=spiral&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=6,4&seed=0.79477&showTestData=false&discretize=false&percTrainData=90&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=true&cosY=false&sinY=true&collectStats=false&problem=classification&initZero=false&hideText=false

TensorFlow
https://www.tensorflow.org

Site web ITIS : Programme de objectif numérique 2017 : vers des sociétés numériques et durables
https://www.itis.ulaval.ca/cms/lang/fr/pid/216790

Wiki : Définition de l'Intelligence artificielle Wikipedia
https://fr.wikipedia.org/wiki/Intelligence_artificielle

Neural Networks and Deep Learning
Andrej Karpathy
http://neuralnetworksanddeeplearning.com/index.html

Deep Learning : An MIT Press Book
Ian Goodfellow and Yoshua Bengio and Aaron Courville
@book{Goodfellow-et-al-2016, title={Deep Learning}, author={Ian Goodfellow and Yoshua Bengio and Aaron Courville}, publisher={MIT Press}, note={\url{http://www.deeplearningbook.org}}, year={2016}}
http://www.deeplearningbook.org