Notes de cours
Semaine 17

Cours Turing

Neurones

Biologique

Un neurone biologique est une cellule nerveuse qui reçoit, traite et transmet des informations sous forme de signaux électriques.

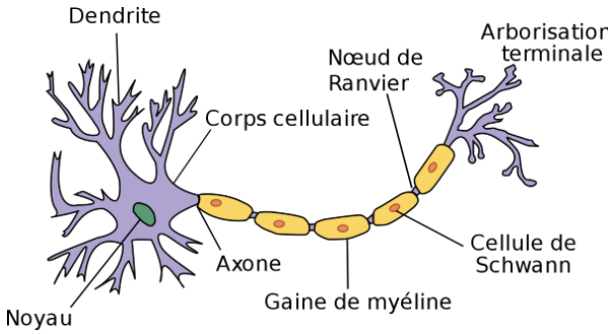

Figure 1: Schéma d'un neurone biologique

Un neurone transmettra un signal électrique à d'autres neurones en fonction des signaux électriques qu'il reçoit. Différents neurones auront des réactions différentes au même signal électrique.

D'une manière simplifiée, un neurone enverra un signal électrique à ses voisins en sortie si la somme des signaux électriques reçus est supérieure à un certain seuil.

Artificiel

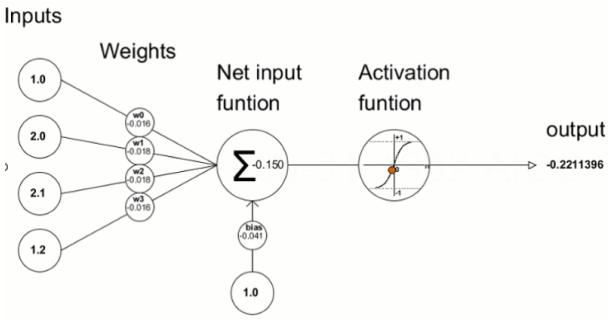

Figure 2: Schéma d'un neurone artificiel "informatique"

En informatique, nous représentons un neurone par une fonction f qui prend en entrée un vecteur $\vec{x}$ et qui renvoie un scalaire y. Cette fonction f dépend de paramètres internes au neurone : un vecteur de poids $\vec{w}$ et un biais b.

$$y = f(\vec{x}) = \sigma(\sum_{i=1}^{n} w_i x_i + b) \tag{1}$$

où σ est une fonction d'activation et w_i le poids de $\vec{w}$ associé à l'entrée x_i.

Les fonctions d'activation utilisées sont généralement des fonctions non linéaires, par exemple :

- La fonction sigmoïde : $\sigma(x) = \frac{1}{1+e^{-x}}$
- La fonction tangente hyperbolique : $\sigma(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- La fonction ReLU : $\sigma(x) = \max(0, x)$

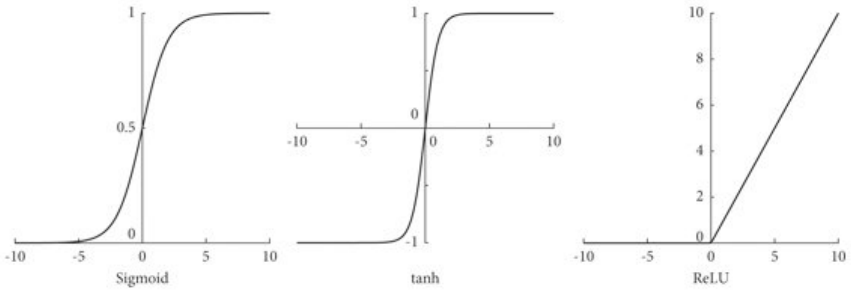

Figure 3: Différentes fonctions d'activation

Exemple numérique

Soit un neurone avec les poids $\vec{w} = [w_1, w_2, w_3, w_4, w_5] = [6, 7, 8, 9, 10]$ et un biais $b = 11$ qui utilise la fonction d'activation ReLU.

Si on lui donne en entrée le vecteur $\vec{x} = [1, 2, 3, 4, 5]$, on obtiendra en sortie :
$\sigma(\vec{x} \cdot \vec{w} + b) = \sigma(1 \cdot 6 + 2 \cdot 7 + 3 \cdot 8 + 4 \cdot 9 + 5 \cdot 10 + 11) = \sigma(141) = max(0, 141) = 141$

Par contre, si on lui donne en entrée le vecteur $\vec{x} = [-1, -1, -1, -1, -1]$, on obtiendra en sortie :
$\sigma(\vec{x} \cdot \vec{w} + b) = \sigma(-1 \cdot 6 - 1 \cdot 7 - 1 \cdot 8 - 1 \cdot 9 - 1 \cdot 10 + 11) = \sigma(-29) = max(0, -29) = 0$

On peut voir que cette fonction réagira différemment en fonction des entrées.
En informatique, un réseau de neurones est simplement un ensemble de neurones interconnectés.

Pourquoi les réseaux de neurones sont si populaires ?

Les réseaux de neurones sont très populaires, car ils sont très efficaces pour approximer des fonctions complexes en utilisant uniquement des données.
Les réseaux de neurones doivent être entrainés afin de pouvoir résoudre un problème. C'est-à-dire que les paramètres internes du réseau de neurones vont être optimisés en fonction des données d'entrainement.

Si nous désirons entrainer un réseau de neurones pour qu'il puisse distinguer des images de chats et des images de chiens, nous allons lui donner en entrée des images de chats et des images de chiens et nous allons lui dire si l'image est un chat ou un chien.
Ce processus va être répété de nombreuses fois, car le processus d'entrainement est itératif.

Nous allons désormais voir les éléments permettant d'entrainer un réseau de neurones.

Les éléments d'un entrainement de réseau de neurones

Un réseau de neurones

La structure du réseau est à définir, combien de couches, combien de neurones par couches ? Autant d'hyperparamètres à définir.
Des hyperparamètres sont des paramètres qui ne sont pas modifiés par le réseau de neurones mais qui sont définis par l'utilisateur.

De manière générale, plus un réseau de neurones contient de paramètres, plus il sera capable d'approximer des fonctions complexes.

Le réseau de neurones sera vu comme une fonction f qui prend en entrée un vecteur X et qui renvoie un vecteur de sortie $\hat{Y}$.

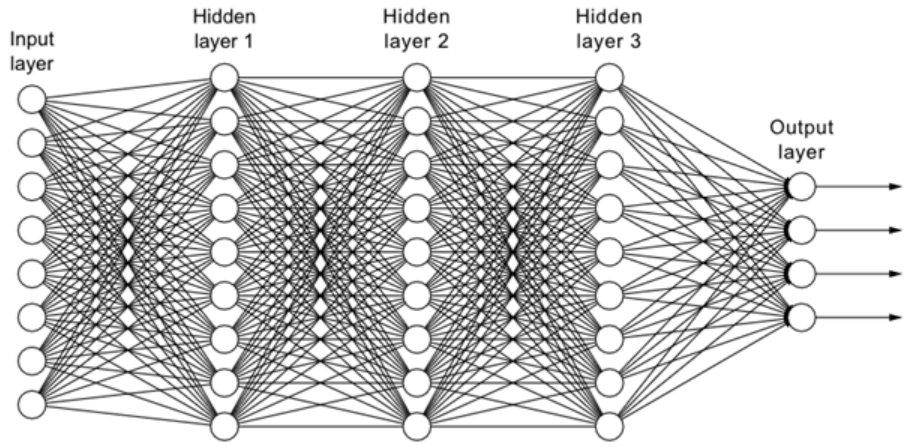

Figure 4: Un réseau de neurones

Le plus important est d'avoir une architecture compatible avec les dimensions des données d'entrainement, que ce soit les données d'entrée ou les données de sortie.

Les données

Les données sont les exemples que l'on va donner au réseau de neurones.
X est un ensemble de données d'entrée et Y est un ensemble de données de sortie.

Plus le réseau de neurones aura de données d'entrainement, plus il sera capable de résoudre des problèmes complexes.
Plus le réseau de neurones possède de paramètres, plus il aura besoin de données d'entrainement.

Les données d'entrainement sont généralement divisées en 3 ensembles :

- L'ensemble d'entrainement (training set) : Cet ensemble est utilisé pour entrainer le réseau de neurones.
- L'ensemble de validation (validation set) : Cet ensemble est utilisé pour analyser les performances du réseau de neurones pendant l'entrainement.
- L'ensemble de test (test set) : Cet ensemble est utilisé pour analyser les performances du réseau de neurones après l'entrainement.

Pourquoi ne pas utiliser l'ensemble de test pour l'entrainement ?
Car les réseaux ont tendance à mémoriser les données d'entrainement, donc si on utilise l'ensemble de test pour l'entrainement, le réseau de neurones sera très performant sur l'ensemble de test, mais nous ne pourrons plus savoir s'il sera capable de généraliser à de nouvelles données.

La fonction de coût

La fonction de coût permet de mesurer l'erreur entre la sortie $\hat{Y}$ du réseau de neurones et la sortie attendue Y.
Il existe plusieurs fonctions de coût, comme l'erreur quadratique moyenne (mean squared error), l'erreur absolue moyenne (mean absolute error), l'entropie croisée (cross entropy), etc.

L'erreur quadratique moyenne est la moyenne des carrés des différences entre les valeurs prédites et les valeurs réelles.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{2}$$

L'erreur absolue moyenne est la moyenne des valeurs absolues des différences entre les valeurs prédites et les valeurs réelles.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \tag{3}$$

L'entropie croisée (cross entropy), qui est souvent utilisée pour les problèmes de classification, est une mesure de la différence entre deux distributions de probabilités.

$$cross\_entropy = -\sum_{i=1}^{n} y_i \cdot \log(\hat{y_i}) \tag{4}$$

Au départ, les paramètres du réseau de neurones sont initialisés aléatoirement, la sortie du réseau de neurones sera aléatoire et donc la fonction de coût sera très élevée. L'objectif de l'entrainement est de modifier les paramètres du réseau de neurones afin de minimiser la fonction de coût.

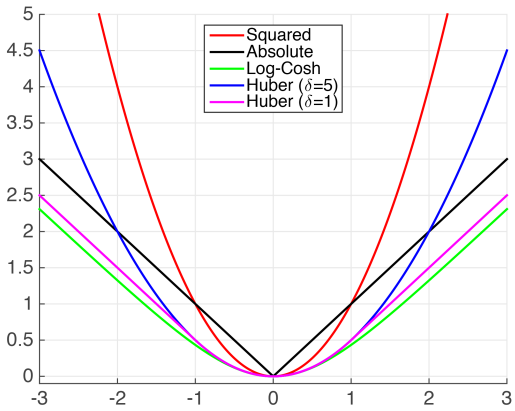

Figure 5: Visualisation de différentes fonctions de coût

La fonction de coût permettra d'analyser si le réseau de neurones est en train de s'améliorer ou non.

L'optimiseur

L'optimiseur permet de modifier les paramètres du réseau de neurones afin de minimiser la fonction de coût.
Il existe plusieurs optimiseurs, le plus connu est la descente de gradient.
Dans l'idée de la descente de gradient, on va modifier les paramètres du réseau de neurones dans le sens opposé du gradient de la fonction de coût.

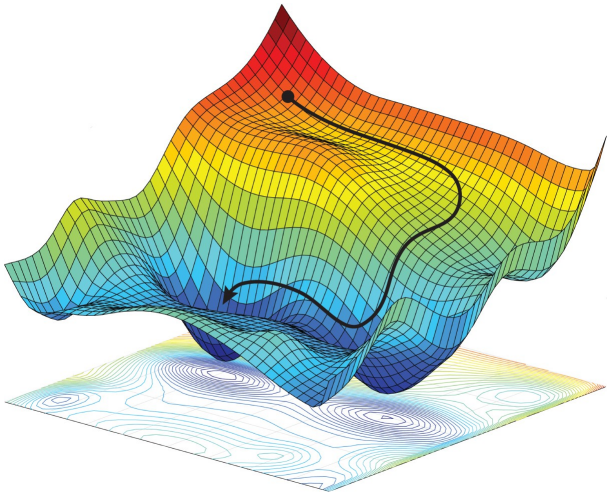

Figure 6: Visualisation d'une descente de gradient

$$\vec{w} = \vec{w} - \alpha \nabla_{\vec{w}} MSE \tag{5}$$

Où α est le taux d'apprentissage.

Les étapes de l'entrainement

Les étapes de l'entrainement sont les suivantes :

1. On donne au réseau de neurones un exemple d'entrée X et on calcule la sortie $\hat{Y}$.
2. On calcule la fonction de coût LOSS(Y, $\hat{Y}$).

3. On calcule le gradient de la fonction de coût par rapport aux paramètres du réseau de neurones.
4. On modifie les paramètres du réseau de neurones dans le sens opposé du gradient de la fonction de coût.

Une époque (epoch) est une itération sur l'ensemble des données d'entrainement, généralement, on effectue plusieurs époques.

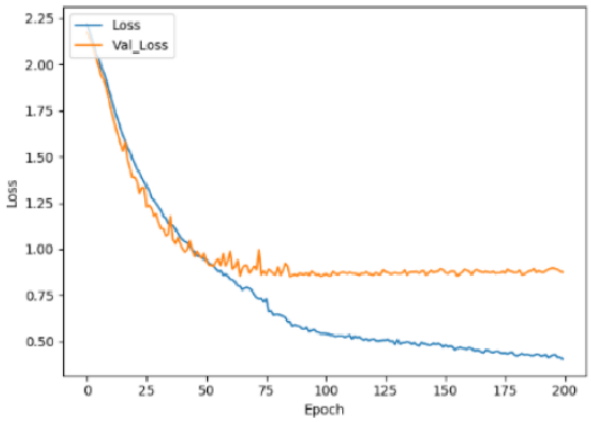

Figure 7: Evolution de la fonction de coût en fonction du nombre d'époques; à partir de l'époque 50, la fonction de coût sur l'ensemble de validation stagne, alors que celle d'entrainement continue de diminuer.

Pytorch

La librairie Pytorch est une librairie d'apprentissage machine (machine learning) basée sur la librairie Torch. Elle est principalement utilisée pour le traitement de données et l'entrainement de réseaux de neurones.

Pour installer Pytorch, il faudra utiliser la commande suivante :

```
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu
```

Les tenseurs

Les tenseurs sont des tableaux multidimensionnels qui peuvent être manipulés par des opérations mathématiques. Cette librairie ressemble beaucoup à la librairie numpy mais permet d'effectuer des calculs sur des cartes graphiques.

Les couches

Il existe plusieurs types de couches dans Pytorch, par exemple :

- Linear
- Conv2d
- MaxPool2d
- ...

La couche Linear

La couche Linear est une couche qui permet de calculer une fonction linéaire qui est donnée par la formule :

$$y = x \cdot W^T + b \tag{6}$$

où x est un vecteur d'entrée, W est une matrice de poids et b est un vecteur de biais.

La dimension de W est $[n_{out}, n_{in}]$ et la dimension de b est $[n_{out}]$.

Une multiplication matricielle est une opération qui prend en entrée une matrice A de dimension $[n, m]$ et une matrice B de dimension $[m, p]$ et qui renvoie une matrice C de dimension $[n, p]$ telle que :

$$C_{i,j} = \sum_{k=1}^{m} A_{i,k} \cdot B_{k,j} \tag{7}$$

La couche Conv2d

La couche Conv2d est une couche qui permet de calculer une convolution 2d.

Soit I une image de dimension $[n, m]$ et F un filtre de dimension $[k, l]$, alors la convolution C de I par F est une image de dimension $[n - k + 1, m - l + 1]$ telle que :

$$C_{i,j} = \sum_{a=1}^{k} \sum_{b=1}^{l} I_{i+a-1,j+b-1} \cdot F_{a,b} \tag{8}$$

La couche MaxPool2d

La couche MaxPool2d est une couche qui permet de réduire la taille d'une image en prenant la valeur maximale dans une fenêtre de taille $[k, l]$.
Généralement, on utilise une fenêtre de taille $[2, 2]$ et un pas de $[2, 2]$ ce qui signifie qu'un seul pixel sur quatre sera conservé.

Elle permet de réduire l'utilisation de mémoire et de calculer plus rapidement les convolutions.

Points clés

- Le but d'un réseau de neurones est d'approximer une fonction f à partir de neurones. Le même réseau de neurones peut approximer différentes fonctions.
- Plus un réseau contient de neurones, plus il sera capable d'approximer des fonctions complexes.
- La qualité et la quantité des données d'entrainement sont très importantes pour entrainer un réseau de neurones.
- Un réseau de neurones possède énormément de paramètres, il est généralement impossible d'expliquer son fonctionnement. C'est un modèle boîte noire.
- Il est impossible de prédire comment un réseau de neurones va réagir à des données qu'il n'a jamais vu.
- L'entrainement d'un réseau de neurones est un processus itératif qui consiste à modifier les paramètres du réseau de neurones afin de minimiser la fonction de coût. C'est un processus dont l'élégance et l'efficacité sont discutables.

L'explosion de la popularité des réseaux de neurones convolutifs

Depuis quelques années, les réseaux de neurones convolutifs sont devenus très populaires et se sont immiscés dans de nombreux domaines.

Ce succès est dû à plusieurs facteurs.

Les couches convolutives

Entrainer un réseau de neurones utilisant des couches Conv2d est beaucoup plus efficace que d'entrainer un réseau de neurones avec des couches Linear. En effet, une couche Conv2d utilise beaucoup moins de paramètres qu'une couche Linear.

De plus, une couche Conv2d prend en compte la structure de l'image et l'aspect local des données : dans une couche linear toutes les entrées sont analysées en même temps.
Les couches convolutives exploitent la localité de l'information et sont particulièrement efficaces pour :

1. L'analyse d'images
2. L'analyse de texte
3. L'analyse audio
4. ...

Dans ce genre de données, les données sont souvent structurées et l'information est locale, en effet dans une phrase, les mots proches sont plus liés que les mots éloignés.
Chat-GPT produit par exemple du texte en produisant un mot à la fois, chaque mot doit être cohérent avec les mots précédents.

L'augmentation de la puissance de calcul

La puissance de calcul des ordinateurs augmente régulièrement, ce qui permet d'entrainer des réseaux de neurones plus complexes ou les mêmes réseaux de neurones plus rapidement.
Mais le facteur le plus important est l'utilisation de cartes graphiques pour effectuer des calculs. En effet, la carte graphique (Graphics Processing Unit en anglais (GPU)) est une unité de calcul très efficace pour effectuer des calculs en parallèle, ce qui possible dans les réseaux de neurones convolutifs.

Pour ces calculs, les cartes graphiques sont beaucoup plus efficaces que les processeurs.

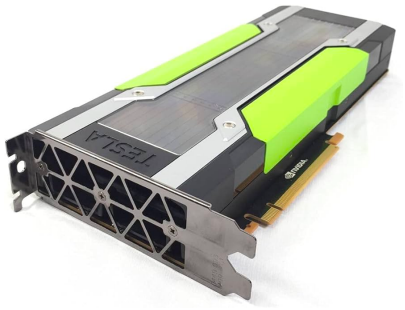

Figure 8: Une carte graphique avec laquelle il n'est pas possible de jouer à des jeux vidéos

Dans la figure 8, nous pouvons voir une carte graphique, une Nvidia Tesla, qui n'a même pas de sortie vidéo.

Il est aussi intéressant de noter que les cartes graphiques sont également utilisées pour le minage de cryptomonnaies.

L'augmentation de la quantité de données

Les données sont devenues de plus en plus accessibles, en effet, il est possible de trouver des données sur internet, de les collecter, de les acheter, etc.

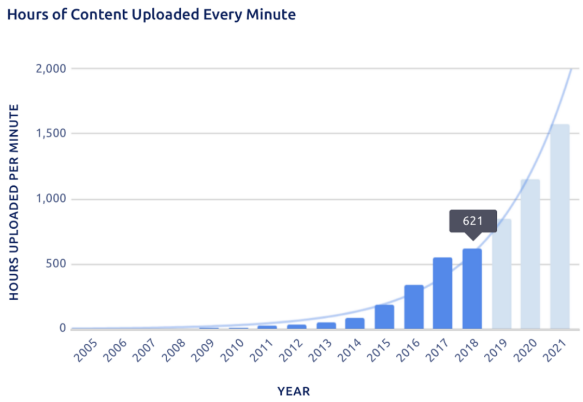

Figure 9: Heures de contenu mis en ligne par minute en fonction des années

Dans la figure 9, nous pouvons voir l'augmentation de la quantité de données sur Youtube.

Les entreprises comme Google, Facebook, Amazon, etc. possèdent énormément de données et les utilisent pour "améliorer leurs services".
Ils utilisent également ces données pour générer des profils utilisateurs et les vendre à des entreprises tierces cherchant à faire de la publicité ciblée.

Ils utilisent également ces données pour entraîner de nouveaux modèles de réseaux de neurones, par exemple Chat-GPT. Nous utilisons la suite office de Microsoft dans les gymnases vaudois, Microsoft a un partenariat avec OpenAI, l'entreprise qui a développé GPT-2 et Chat-GPT.

Il n'est pas impossible que vos données soient utilisées par des entreprises sans que vous le sachiez.

Des résultats impressionnants

Les CNNs ont explosé en popularité, car la performance de ces modèles dans diverses tâches a été démontrée :

1. Conduite autonome
2. Analyse d'image médicale
3. Jeu de Go
4. Deepfakes 1
5. Deepfakes 2
6. ChatGPT 1
7. ChatGPT 2
8. Midjourney
9. Midjourney
10. Reconnaissance de visage
11. ...