TP1 - Deep Learning

Réalisé par : GHALIM Nidal, HAMID Hajar, LAHOUIRI Hayat, NACIRI Nacira.

Encadré par : Pr. KHALFI Hamza.

Partie 1 : Formalisation mathématique

1.1 Jeu de données :

1. A quoi servent les ensembles d'apprentissage, de validation et de test ?

Les ensembles d'apprentissage, de validation et de test sont utilisés dans le processus d'apprentissage supervisé pour évaluer et ajuster un modèle de réseau de neurones.

- Ensemble d'apprentissage : Cet ensemble est utilisé pour entraîner le modèle. Les données d'apprentissage sont utilisées pour ajuster les poids et les paramètres du réseau de neurones en fonction des exemples fournis. Le modèle tente d'apprendre la relation entre les caractéristiques (x) et les étiquettes (y) de l'ensemble.

- Ensemble de validation : Après chaque itération d'entraînement, l'ensemble de validation est utilisé pour évaluer la performance du modèle. Cela permet de détecter le surapprentissage (overfitting) et de régler les hyperparamètres du modèle, tels que la taille du réseau, le taux d'apprentissage (le learning rate). L'ensemble de validation est essentiel pour éviter que le modèle ne s'ajuste trop bien aux données d'apprentissage sans généraliser correctement.

- Ensemble de test : Une fois que le modèle a été formé et ses hyperparamètres ont été ajustés, il est évalué sur l'ensemble de test, qui est indépendant de l'ensemble d'apprentissage et de l'ensemble de validation. Cela permet d'estimer

la performance réelle du modèle sur de nouvelles données non vues. Le résultat sur l'ensemble de test donne une indication de la capacité du modèle à généraliser et à prendre des décisions sur des données inconnues.

2. Quelle est l'influence du nombre N d'exemples ?

Le nombre N d'exemples dans le jeu de données a plusieurs influences sur l'apprentissage et la performance du modèle de réseau de neurones :

-> Plus le nombre N est élevé, plus le modèle a de données pour s'entraîner, ce qui peut potentiellement lui permettre d'apprendre des modèles plus complexes et de mieux généraliser sur de nouvelles données.

Cependant, un grand nombre de données peut également rendre l'entraînement plus coûteux en temps et en ressources computationnelles.

-> Un petit nombre de données peut entraîner un risque de surapprentissage, où le modèle s'ajuste trop bien aux données d'apprentissage mais ne généralise pas correctement sur de nouvelles données.

Donc, le choix du nombre optimal d'exemples dépend du problème spécifique et de la complexité du modèle. Il est souvent nécessaire de faire des compromis entre la taille du jeu de données et les ressources disponibles pour l'entraînement.

1.2. Architecture du réseau (phase forward)

3. Pourquoi est-il important d'ajouter des fonctions d'activation entre des transformations linéaires ?

Il est important d'ajouter des fonctions d'activation entre des transformations linéaires dans un réseau de neurones pour introduire de la non-linéarité. Les transformations linéaires, telles que les multiplications matricielles, peuvent seulement effectuer des opérations linéaires sur les données, ce qui signifie que le réseau serait limité à apprendre des relations linéaires simples entre les caractéristiques d'entrée et les prédictions de sortie.

Et donc, les fonctions d'activation introduisent la non-linéarité en permettant au réseau d'apprendre des relations plus complexes, ce qui le rend capable de

modéliser des données plus variées et de résoudre des problèmes plus complexes.

4.Quelles sont les tailles nx, nh, ny sur la figure 1 ? En pratique, comment ces tailles sont-elles choisies ?

Sur la figure 1, les tailles nx, nh et ny représentent les dimensions des couches du réseau de neurones :

- nx est la dimension de l'entrée, c'est le nombre de caractéristiques (ou dimensions) de l'entrée x.

- nh est la dimension de la couche cachée, c'est le nombre de neurones dans la couche cachée.

- ny est la dimension de la sortie, c'est le nombre de neurones dans la couche de sortie, ce qui dépend du type de problème que le réseau résout (par exemple, ny = 1 pour une classification binaire, ny = nombre de classes pour une classification multiclasse, etc.).

En pratique, ces tailles sont généralement choisies en fonction de la complexité du problème, de la quantité de données disponibles et de l'expérimentation. Il n'y a pas de règle unique pour déterminer ces dimensions, et il peut être nécessaire d'expérimenter avec différentes architectures pour trouver celle qui fonctionne le mieux pour un problème spécifique.

5.Que représentent les vecteurs $\hat{y}$ et y ? Quelle est la différence entre ces deux quantités ?

Les vecteurs $\hat{y}$ et y représentent les prédictions du modèle et les étiquettes réelles respectivement :

- $\hat{y}$ est la sortie prédite par le modèle après la phase forward. C'est ce que le modèle estime comme étant la réponse aux données d'entrée x.

- y est l'étiquette réelle associée aux données d'entrée x. C'est ce que l'on souhaite que le modèle prédise de manière précise.

La différence entre $\hat{y}$ et y est que $\hat{y}$ est la prédiction du modèle, tandis que y est la valeur réelle que l'on cherche à approcher ou à prédire avec le modèle.

6. Pourquoi utiliser une fonction SoftMax en sortie ?

On utilise une fonction SoftMax en sortie pour transformer les sorties du réseau en une distribution de probabilité sur les classes (dans le cas de la classification multiclasse).

La fonction SoftMax prend en entrée un vecteur de scores ou d'activations et renvoie un vecteur de probabilités, où chaque composante représente la probabilité d'appartenance à une classe particulière.

7. Ecrire les équations mathématiques permettant d'effectuer la passe forward du réseau de neurones, c'est-à-dire permettant de produire successivement $\tilde{h}$, h, $\tilde{y}$ et $\hat{y}$ à partir de x.

Les équations mathématiques pour effectuer la phase forward du réseau de neurones sont les suivantes :

- Transformation linéaire de la couche cachée : $\tilde{h} = W_h * x + b_h$

- Application de la fonction d'activation (par exemple, ReLU) à la couche cachée : $h = ReLU(\tilde{h})$

- Transformation linéaire de la couche de sortie :

$\tilde{y} = W_y * h + b_y$

- Application de la fonction SoftMax pour obtenir les prédictions finales :

$\hat{y} = SoftMax(\tilde{y})$

avec : W_h, W_y, b_h et b_y sont les poids et les biais des couches cachées et de sortie, et ReLU est la fonction d'activation Rectified Linear Unit.

1.3 Fonction de coût

8. Pendant l'apprentissage, on cherche à minimiser la fonction de coût. Pour l'entropie croisée et l'erreur quadratique, comment les $\hat{y}_i$ doivent-ils varier pour faire diminuer la loss ?

Pendant l'apprentissage, on cherche à minimiser la fonction de coût, et la manière dont les valeurs prédites (ŷi) doivent varier dépend de la fonction de coût que vous utilisez.

Pour :

-> L'entropie croisée pour la classification :Dans le cas de la classification, l'entropie croisée est couramment utilisée comme fonction de coût. Pour minimiser cette fonction de coût, les valeurs prédites ŷi doivent se rapprocher autant que possible des étiquettes réelles y. Plus précisément et par exemple, si l'étiquette réelle (y) est 1 pour une certaine classe, alors la valeur prédite (ŷi) doit être aussi proche que possible de 1. Si l'étiquette réelle est 0, alors la valeur prédite doit être aussi proche que possible de 0. L'entropie croisée pénalise les erreurs importantes en attribuant des coûts élevés aux prédictions incorrectes.

-> L'erreur quadratique pour la régression : L'erreur quadratique est couramment utilisée dans les problèmes de régression. Pour minimiser cette fonction de coût, les valeurs prédites (ŷi) doivent se rapprocher le plus possible des étiquettes réelles (y). Plus précisément, elles doivent être égales aux étiquettes réelles, car l'erreur quadratique est la différence au carré entre les prédictions et les étiquettes réelles. Les valeurs prédites doivent donc être ajustées pour minimiser cette différence au carré.

En résumé, pour minimiser l'entropie croisée, les valeurs prédites doivent se rapprocher des étiquettes réelles dans le cas de la classification, tandis que pour minimiser l'erreur quadratique, les valeurs prédites doivent être aussi proches que possible des étiquettes réelles dans le cas de la régression.

9. En quoi ces fonctions sont-elles plus adaptées aux problèmes de classification ou de régression ?

Ces fonctions de coût sont plus adaptées aux problèmes de classification ou de régression en raison de leurs propriétés mathématiques et de la nature des problèmes qu'elles visent à résoudre.

L'entropie croisée est appropriée pour les problèmes de classification, car elle pénalise fortement les prédictions incorrectes, ce qui est crucial dans la classification où vous cherchez à attribuer une classe spécifique à chaque exemple. L'erreur quadratique est adaptée à la régression, car elle mesure l'écart quadratique entre les valeurs prédites et les valeurs réelles, ce qui est approprié lorsque vous cherchez à prédire des valeurs numériques.

1.4 Méthode d'apprentissage

10. Quels semblent être les avantages et inconvénients des diverses variantes de descente de gradient entre les versions classique, stochastique sur mini-batch et stochastique online ? Laquelle semble la plus raisonnable à utiliser dans le cas général ?

La descente de gradient est un élément clé de l'apprentissage dans les réseaux de neurones. Différentes variantes de descente de gradient ont leurs avantages et inconvénients :

1. Descente de gradient classique :

- Avantages :

- Converge vers un minimum global du coût.

- Peut être plus stable car utilise l'ensemble des données à chaque itération.

- Inconvénients :

- Peut être lent pour de grands ensembles de données.

- Peut être coûteux en mémoire car nécessite de stocker l'ensemble des données d'entraînement en mémoire.

2. Descente de gradient stochastique sur mini-batch :

- Avantages :

- Converge plus rapidement car met à jour les poids fréquemment.

- Peut être utilisé pour de grands ensembles de données en exploitant les avantages du parallélisme.

- **Inconvénients :**

- Peut être moins stable et converger vers un minimum local.

- Les mises à jour des poids sont bruitées en raison de l'utilisation de mini-batch, ce qui peut nécessiter un taux d'apprentissage plus délicat.

3. Descente de gradient stochastique en ligne :

- Avantages :

- Converge encore plus rapidement car met à jour les poids à chaque exemple.

- Peut être efficace pour les ensembles de données en streaming.

- Inconvénients :

- Les mises à jour des poids sont très bruitées, ce qui peut rendre la convergence plus instable.

- Peut nécessiter un taux d'apprentissage plus petit pour éviter des oscillations.

En résumé, on trouve que la descente de gradient stochastique sur mini-batch est la plus raisonnable pour la plupart des cas d'utilisation. Vue qu'elle combine les avantages de la descente de gradient classique (convergence vers un minimum global) et de la descente de gradient stochastique en ligne (efficacité en termes de vitesse de convergence). De plus, la taille du mini-batch peut être ajustée pour équilibrer la vitesse de convergence et la stabilité.

11. Quelle est l'influence du learning rate sur l'apprentissage ?

L'influence du taux d'apprentissage sur l'apprentissage est cruciale. Un taux d'apprentissage trop élevé peut entraîner une divergence du modèle, tandis qu'un taux d'apprentissage trop faible peut entraîner une convergence lente ou la convergence vers un minimum local. Le choix du taux d'apprentissage est souvent empirique et peut nécessiter une recherche d'hyperparamètres.

12. Comparer la complexité (en fonction du nombre de couches du réseau) du calcul des gradients de la loss par rapport aux paramètres, en utilisant l'approche naïve et l'algorithme de backprop.

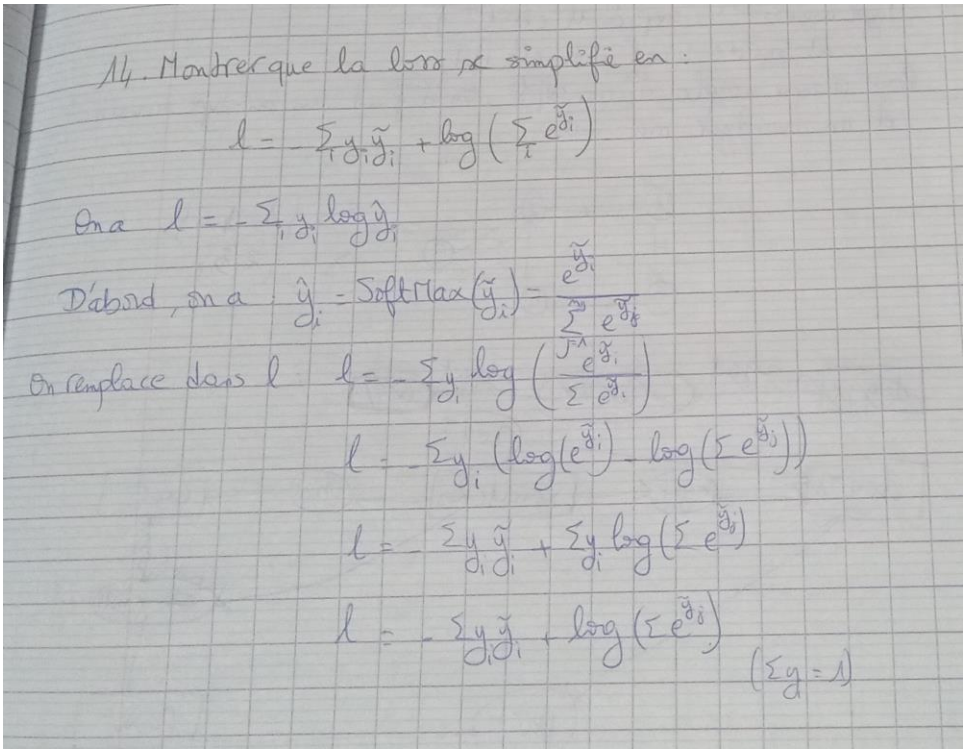
La complexité du calcul des gradients par rapport aux paramètres dépend du nombre de couches du réseau. L'approche naïve nécessite le calcul de tous les gradients pour chaque paramètre, ce qui peut être coûteux en termes de temps. En revanche, l'algorithme de backpropagation profite de la réutilisation des dérivées calculées précédemment, ce qui réduit considérablement la complexité du calcul.

13. Quel critère doit respecter l'architecture du réseau pour permettre la backpropagation ?

Pour permettre la backpropagation, l'architecture du réseau doit respecter la continuité des transformations et des dérivées. Les fonctions d'activation doivent

être différentiables, et les réseaux ne doivent pas contenir de boucles ou de dépendances cycliques.

14. La fonction SoftMax et la loss de cross-entropy sont souvent utilisées ensemble et leur gradient est très simple. Montrez que la loss se simplifie en :



15. Ecrire le gradient de la loss (cross-entropy) par rapport à la sortie intermédiaire $\tilde{y}$:

15. Ecrire le gradient de la loss (cross-entropy) par rapport à $\tilde{y}_i$

$$\frac{\partial \ell}{\partial \tilde{y}_i} = \frac{\partial}{\partial y_i}\left(-\sum_i y_i \tilde{y}_i + \log\left(\sum_i e^{\tilde{y}_i}\right)\right)$$

$$\frac{\partial \ell}{\partial \tilde{y}_i} = \frac{\partial}{\partial y_i}\left(-\sum_i y_i \tilde{y}_i\right) + \frac{\partial}{\partial y_i}\left(\log\left(\sum_i e^{\tilde{y}_i}\right)\right)$$

$$\frac{\partial \ell}{\partial \tilde{y}_i} = -y_i + \frac{e^{\tilde{y}_i}}{\sum e^{\tilde{y}_i}}$$

$$\frac{\partial \ell}{\partial y_i} = -y_i + \hat{y}_i$$