Consignes et critères d'évaluation pour la remise du problème sur la rétropropagation de l'erreur pour réseaux à décharges et la classification

Jean ROUAT
Département de génie électrique et génie informatique
Université de Sherbrooke
courriel : Jean.Rouat@usherbrooke.ca

Neuro-computationel GEI723
Automne 2022

5 janvier 2022

Résumé

On donne l'énoncé ainsi que les critères d'évaluation du problème. Ce problème permet de travailler la compréhension des modifications à apporter à l'algorithme de la rétropropagation de l'erreur pour le faire fonctionner avec des réseaux à décharges. L'implémentation des équations des neurones à décharges est aussi étudiée.

1 Introduction

L'article de référence est 'Surrogate gradient learning in spiking neural networks' par EO Neftci, H Mostafa & F Zenke, paru dans la revue internationale IEEE SIGNAL PROCESSING MAGAZINE. Cet article est disponible dans le répertoire du problème.

ATTENTION : Le but du problème est de vous faire avancer dans la compréhension et l'apprentissage de la matière. Il n'est pas nécessaire de maîtriser parfaitement

le contenu de l'article donné en référence ci-dessus. On accordera une attention particulière à l'analyse du fonctionnement de VOS réseaux et des résultats en lien avec la compréhension et l'acquisition de la matière.

1.1 Compétences et Objectifs

1. Comprendre l'impact du choix de la forme de la dérivée associée à une décharge dans l'algorithme de rétropropagation de l'erreur ;
2. Savoir résoudre et implémenter les équations différentielles qui caractérisent un réseau de neurones ;
3. Être capable d'étudier et d'analyser l'impact des méta-paramètres sur les résultats ;
4. Être capable de définir l'architecture d'un réseau de neurones à décharges avec apprentissage par rétropropagation de l'erreur pour une application en classification ;
5. Être capable de faire le lien entre la littérature scientifique proposée dans ce problème et l'implémentation informatique en langage python.

2 Travail à réaliser

1. L'ensemble du travail se fait à partir de l'environnement qui a été installé en début de session sous Anaconda et qui utilise les librairies pytorch ainsi que d'autres paquets tels qu'introduits dans le fichier d'exemple donné en laboratoire et dans le fichier de code à trous. ATTENTION, le code à trous nécessite l'utilisation de la version 0.11.0 du package "sparse" afin d'assurer la compatibilité [1].
2. Afin de développer la solution finale, il sera nécessaire de réaliser plusieurs apprentissages et configurations de réseaux de neurones, ce qui peut-être long et peut requérir plusieurs heures de traitement sur un CPU conventionnel. Il est donc recommandé de développer des solutions préliminaires qui sont apprises sur des sous-ensembles plus petits de données. À la toute fin, une fois satisfaits de votre réseau vous pourrez utiliser toutes les données et observer ainsi l'impact de la taille de la base des données sur les résultats.

1. Avec pip préciser "sparse==0.11.0"

Il vous est demandé de travailler avec 3 jeux de données (apprentissage, validation et test).

2.1 Partie I : Résolution des équations différentielles

En utilisant la méthode d'intégration par Euler ou la méthode exacte (intégration préalable « à la main » et utilisation de la solution sous forme numérique), proposez une solution à temps discret (vous référer par exemple au procédural sur les modèles) des 2 équations suivantes :

$$\frac{dI_i(t)}{dt} = -\frac{I_i(t)}{\tau_I} + \sum_{j=1}^{N} w_{ij} S_j(t) \tag{1}$$

$$\frac{dU_i(t)}{dt} = \frac{-U_i(t) + I_i(t)}{\tau_U} \tag{2}$$

Avec i l'indice du neurone, $I(t)$ le courant du neurone, $U(t)$ le potentiel membranaire du neurone, w_{ij} le poids synaptique du neurone j vers le neurone i, S_j la séquence d'impulsions de sortie du neurone j. N est le nombre de neurones j qui sont pré-synaptiques au neurone i. Veuillez noter qu'il s'agit d'un modèle de neurone de type à intégration et décharge avec fuite dont les entrées sont situées au niveau du courant.

2.2 Partie II : Lecture de l'article de référence

1. Dans un premier temps, lire l'article *Surrogate Gradient Learning in Spiking Neural Networks* de façon rapide et superficielle afin d'en comprendre les principes généraux (maximum 1 heure de lecture) ;
2. Ensuite, comparer les équations de la partie I à celles de l'article. En quoi sont-elles différentes ? Pourquoi ? Pour quel type de réseau de neurones s'appliquent-elles ?
3. Puis faire une lecture plus approfondie pour comprendre les éléments principaux de la section sur « Methods for training RNN »
4. Bien étudier les sous-sections Smoothed SNNs et Surrogate gradients
5. Il n'est pas nécessaire de lire la suite de l'article à partir de la section APPLICATIONS.

2.3 Partie III : Code à "trous"

Le code à trous vous sera transféré APRÈS que vous aurez fourni un schéma conceptuel démontrant votre stratégie de résolution du problème (éléments techniques et méthodologie). ATTENTION, le code à trous est fourni à titre d'exemple. Il ne faut pas procéder exactement comme dans le code à trous. Par exemple, le code fourni n'utilise pas de données de validation. À vous de faire en sorte qu'il le fasse. Ce code a été développé par Ismaël Balafrej, étudiant au PhD au sein du groupe de recherche NECOTIS (NEuro-COmputationnel et Traitement Intelligent des Signaux) et propose une méthode originale d'apprentissage de réseaux de neurones à décharges. Cette méthode intègre principalement les travaux de 3 laboratoires de recherche [2].

1. E. O. Neftci, H. Mostafa, Friedemann Zenke & Ganguli (2019), Surrogate Gradient Learning in Spiking Neural Networks, IEEE Signal Processing Magazine, Nov. 2019, pp. 51–63, preprint : https://arxiv.org/pdf/1901.09948.pdf ;
2. S. B. Shrestha & G. Orchard (2018), SLAYER : Spike Layer Error Reassignment in Time NIPS 2018
3. G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein & W. Maass, Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets, arxiv.org/abs/1901.09049, January 2019, preprint : https://arxiv.org/pdf/1901.09049.pdf

Modifiez le code à trous pour créer et tester dans un premier temps un réseau de neurones à décharges avec une seule couche cachée pour la classification de MNIST [3] En plus de la ReLu qui est déjà fournie on vous demande d'implémenter au moins 2 autres façons différentes de substituer le gradient et 2 autres formes de la Relu (données aux équations ci-dessous). Vous devez au préalable compléter les lignes de code avec les équations des neurones. Utilisez la configuration pour faire l'analyse de l'impact de la forme et des paramètres de ces dérivées. Il est important de détailler la façon dont vous faites l'évaluation ainsi que l'interprétation et la compréhension.

2. la lecture des articles de : S. B. Shrestha & G. Orchard et de G. Bellec et al. n'est pas nécessaire. L'information est fournie uniquement pour ceux d'entre vous intéressés à approfondir dans le futur ce domaine

3. Le code à trou fournit un exemple avec 2 couches et l'utilisation de la ReLu ainsi que de sa dérivée (pour implémenter le gradient substitué).

2.4 Partie IV : Analyse, évaluation et compréhension de l'impact des formes et paramètres des dérivées des fonctions substituées

La configuration proposée dans le code à trous comprend 2 couches de neurones (128 neurones dans la couche cachée et un nombre de neurones dans la couche de sortie égal au nombres de chiffres à classer). On demande d'utiliser cette configuration initiale pour étudier et analyser toutes les implémentations de la dérivée (on inclut aussi l'exemple déjà fourni dans le code à trous) de la non-linéarité (fournir un tableau donnant les résultats de classification en fonction de la forme et des paramètres des dérivées). Chaque forme d'implémentation de la dérivée comprend des paramètres qui sont variables (α pour la ReLu, paramètres multiplicatifs ou de translation des entrées, etc.) ; étudier l'impact de ces paramètres sur la convergence et les résultats. Dans le code à trous, un exemple particulier de Relu est donné. Toutefois, pour ce problème considérez la forme plus générale de la Relu :

$$z = max(0, y) + \alpha.min(0, y)$$

Il est possible d'étudier alors 3 types de Relu :
$\alpha = 0$ est la Relu donnée dans l'exemple ;
$\alpha = 0.01$ est la Relu avec fuite ;
$\alpha = -1$, est la Relu de type $|\ |$.
Il serait même possible d'inclure la recherche du paramètre α dans l'algorithme d'apprentissage pour trouver la meilleure configuration de la Relu. On ne vous demande toutefois pas d'aller aussi loin.

Vous pouvez aussi choisir les couches auxquelles vous attribuez les formes différentes de Relu lorsque ceci semble justifié.

2.5 Partie V : Analyse, évaluation et compréhension de l'impact des méta-paramètres

Cette fois-ci vous avez toute latitude pour choisir le nombre de neurones, de couches cachées, etc. Choisir la configuration initiale (avec le gradient substitué que vous choisissez pour le reste des expériences) qui vous semble la plus judicieuse pour faire la suite du problème. On vous demande de modifier les méta-paramètres du réseau (nombre de couches, de neurones, nombre d'itérations, le taux d'apprentissage, le pas de discrétisation ($\mathbf{\Delta t}$) des équations), la taille des lots

pour essayer d'améliorer les performances de classification. Reportez les résultats dans un tableau, analysez et interprétez.

2.6 Partie VI : Réseau de type "Apprentissage Extrême"

L'expérience de cette section cherche à minimiser l'apprentissage et à le limiter à quelques couches ou neurones seulement. Choisir l'architecture que vous préférez et qui comprend au moins 2 couches cachées. Modifiez cette architecture pour :

1. que la première couche cachée de neurones ne soit pas apprise. C'est à dire que les connexions entre les neurones d'entrée et les neurones de la première couche cachée sont fixées de façon aléatoire. Utilisez l'apprentissage pour les autres couches ;
2. Étudiez une autre configuration où les couches cachées ne sont pas apprises ;

Comparez ensuite ces réseaux avec celui de la section précédente. Quel est le gain obtenu avec un réseau complètement entrainé par rapport à un réseau "extrême" ? Choisir la configuration "extrême" qui vous convient et modifiez les paramètres (nombre de neurones dans la première couche cachée, nombre de couches, taux d'apprentissage) de ce réseau pour en améliorer les résultats. Vous pouvez étudier l'impact d'autre paramètres. Les performances de ce réseau sont elles si différentes d'un réseau complètement appris ? Quels en sont les avantages et inconvénients ?

3 Critères d'évaluation

1. **Partie I : Résolution des équations différentielles (10%)** ;
   — Partie I : Proposez une solution à temps discret ;
2. **Partie II : Lecture de l'article de référence (5%)** ;
   Comparer les équations, en quoi elles sont différentes, pourquoi, pour quel type de réseau ?
3. **Partie III : Code à "trous" (10%)** ;
   Créer et tester un réseau de neurones à décharges avec une seule couche cachée ;
   (En plus de la ReLu fournie : 2 autres formes de ReLu et au moins 2 autres substitution du gradient).

Analyse de l'impact de la forme et des paramètres de ces dérivées.
Détailler l'évaluation ainsi que l'interprétation et la compréhension.

4. **Partie IV : Analyse, évaluation et compréhension de l'impact des formes et paramètres des dérivées (15%)** ; Même type de question mais cette fois-ci pour un réseau à 2 couches cachées ;
   Avec un tableau donnant la classification en fonction de la forme et des paramètres des dérivées.
   Impact des paramètres sur la convergence et les résultats.
   Couches auxquelles vous attribuez les formes différentes de ReLu lorsque ceci semble justifié.
5. **Partie V : Analyse, évaluation et compréhension de l'impact des méta-paramètres (16%)** ; Tableau reportant les résultats, analyse et interprétation pour :
   Nombre de couches, de neurones, nombre d'itérations, le taux d'apprentissage, le pas de discrétisation (Δt), la taille des lots.
6. **Partie VI : Réseau de type "Apprentissage Extrême" (20%)** ; Réseau avec au minimum 2 couches cachées :
   Première couche non apprise ;
   Autre configuration pour laquelle certaines couches cachées ne sont pas apprises.
   Quel est le gain obtenu par rapport à un réseau complètement appris ?
   Étudier l'impact des paramètres sur un réseau "extrême" : nombre de neurones dans la première couche cachée, nombre de couches, taux d'apprentissage.
   Les performances de ce réseau sont elles si différentes d'un réseau complètement appris ?
   Quels en sont les avantages et inconve ?nients ?
7. **ÉVALUATION DU CODE INFORMATIQUE (24%)**
   (a) Fichier LISEZ-MOI et en-tête de fichiers / 4 points ;
   (b) Commentaires dans le code permettant la compréhension / 2 points
   (c) Reproductibilité des résultats du poster /12 points
   (d) Facilité du changement entre les 3 types de dérivées / 2 points
   (e) Clarté et qualité du code / 4 points
8. **Qualité du Français ou utilisation abusive du Franglais : (peut conduire jusqu'à - 5%)**

Le poster doit inclure les informations permettant au lecteur de comprendre vos solutions afin de reproduire vos résultats. Éventuellement fournir une table des paramètres utilisés

4 Notation des cibles d'apprentissage

— A dépassé les objectifs : 95% et plus
— A atteint les objectifs : 80% et plus
— Moyen, est en dessous des objectifs mais correct : 65%
— Est en dessous des objectifs : 50%
— Est très largement en dessous des objectifs : 40% et moins

5 Instructions pour la rédaction du poster

Les instructions sont les mêmes que pour le problème 1 (Importance de placer sur le poster tous les éléments permettant de voir que tous les résultats ont été obtenus, 1 vidéo d'au maximum 20mns pour vous permettre de présenter et de défendre votre proposition). Vous pouvez utiliser 2 pages pour le poster et 3 pages pour les annexes.

6 Remise du poster et du code informatique qui y est relié

À la lecture du poster, l'évaluateur doit être en mesure de connaître les contributions de chaque personne. Pour chaque section et/ou figure et/ou table du poster, on doit indiquer les initiales des personnes. Pour le code informatique, les consignes sont aussi les mêmes : identifier dans le code les contributions respectives. Par ailleurs, les instructions sont les même que pour le problème 1. Remettre l'ensemble du travail pour **jeudi le 9 décembre 2021 à 12h00 - date et heure à valider en classe** dans l'outil devoir sur moodle. **Attention :** tout retard sera pénalisé. le serveur de dépôt indiquera l'heure à laquelle le dépôt a été effectué. Si vous avez un conflit d'horaire, SVP en aviser immédiatement le professeur.

7 ATTENTION

Ce problème est personnalisé et il y a plusieurs façons de le résoudre. Il est de la responsabilité des équipes de consulter régulièrement l'équipe pédagogique afin de valider avec celle-ci l'évolution des apprentissages. Le problème est défini pour permettre d'assimiler et de comprendre la matière exposée en cours. La démarche adoptée pour résoudre ce problème est souvent plus importante que les résultats finaux. Cette démarche devrait maximiser l'apprentissage.

Remerciements

À Ismaël Balafrej pour avoir préparé les notebooks python permettant d'intégrer les derniers résultats de la recherche en apprentissage par rétropropagation de l'erreur avec des neurones à décharges. À Ahmad El Ferdaoussi pour la mise à jour des notebooks python.