CY Cergy Paris Université

Composante Sciences et Techniques

Master Informatique - 1ère année

Systèmes Intelligents et Communiquants

Mention Informatique et Ingénierie des Systèmes Complexes

Rapport TP4

Intelligence Artificielle +

Auteur :

LABADY Sterley Gilbert

29 mars 2023

1 Introduction

Le but de ce TP est d'implémenter et entraîner un réseau de neurones récurrent (RNN) sur une tache de classification. Nous construisons et entraînons un RNN sur une tâche de classification de textes.

2 Données

La base de données contient 10000 reviews de films étant classifiées comme "positives" ou "négatives". Chaque texte est une séquence de mots (w1 , . . . , wT) de longueur variable. Nous allons utiliser une couche de représentation (word embedding) pré-entraînée pour obtenir une entrée (x1 , . . . , xT) où chaque xi correspond à un encodage sémantique du mot wi dans un espace de dimension 50. Nous entraînons notre réseau récurrent à classifier chaque texte en prédisant s'il s'agit d'une review positive y = 1 ou négative y = 0. Nous séparons notre dataset en 9000 textes pour l'apprentissage et 1000 textes pour la validation.

3 Implémentation

Traitement des données : Nous avons utilisé la bibliothèque Gensim pour prétraiter les reviews et les représenter sous forme de séquences de vecteurs. Les représentations vectorielles ont été utilisées pour transformer les mots en vecteurs. Les reviews ont été complétées pour avoir une longueur fixe de 50 mots.

Class RNN : La class RNN est implémentée en utilisant PyTorch, avec une architecture comprenant 5 couches cachées avec une taille 64. Le modèle prend en entrée les séquences de vecteurs et produit en sortie une prédiction de classe (positive ou négative). Le modèle est entraîné avec la fonction de perte d'entropie croisée et l'optimiseur Adam.

Entraînement : Le modèle a été entraîné sur 50 époques, avec une taille de lot de 64 et un taux d'apprentissage de 0.001. Les performances du modèle ont été évaluées en utilisant l'ensemble de test, avec une mesure de l'accuracy de la classification.

Test et Résultats : Nous avons ensuite testé le modèle sur un ensemble de test. Le modèle a atteint un taux d'accuracy de 61%.

4 Analyses

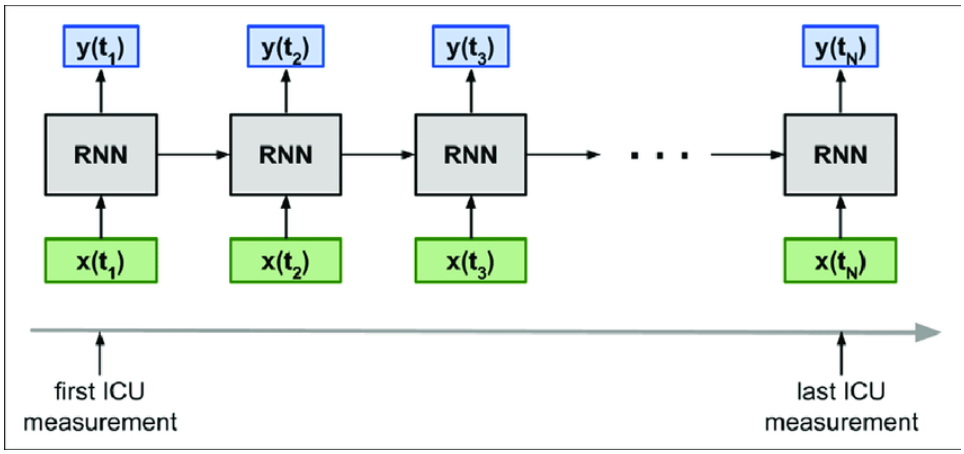
- Précision sur les données d'apprentissage : 64
- Précision sur les données de validation : 61

La précision du classifieur sur les données d'apprentissage est légèrement supérieure à celle sur les données de validation, ce qui est attendu, car le modèle s'ajuste aux données d'apprentissage pendant l'entraînement. Les taux d'accuracy indiquent que le modèle peut encore être amélioré. Des architectures plus avancées, telles que LSTM, pourraient améliorer les performances.

En conclusion, les précisions du classifieur RNN sur les données d'apprentissage et de validation montrent qu'il n'y a pas de surapprentissage significatif. Pour améliorer les performances, nous recommandons d'explorer d'autres architectures de modèles.

5 Questions / Réponses

Si vous deviez utiliser un RNN pour apprendre à générer une séquence (y0 , . . . , yT) en fonction d'une entrée xt=0 , quelle architecture de RNN utiliseriez vous ? (faire un schéma similaire au schéma de droite figure 1)
R. Si vous souhaitez utiliser un RNN pour générer une séquence (y0, ..., yT) à partir d'une entrée unique xt=0, nous pouvons utiliser une architecture de RNN avec une couche d'entrée et plusieurs couches cachées récurrentes et une sortie. L'entrée unique xt=0 est d'abord transformée en une représentation vectorielle. La class RNN calcule le nouvel état caché (h_t). L'état caché (h_t) est ensuite passé à travers une couche entièrement connectée (FC) pour générer la première sortie y0. Le procédé est répété plusieurs fois, en utilisant à chaque fois le même vecteur d'entrée incorporé et les états cachés mis à jour pour générer les sorties y1, y2, ..., yT.


y(t1)
y(t2)
y(t3)
y(tN)
RNN
RNN
RNN
RNN
x(t1)
x(t2)
x(t3)
x(tN)
first ICU measurement
last ICU measurement

Quel est le nombre de paramètres dans le réseau en fonction de la dimension d de la couche cachée du RNN ?
R.
n : la taille de la couche d'entrée (dimension du vecteur d'entrée) : 50
d : la taille de la couche cachée (dimension de l'état caché) : 64
l : Nombre de couche : 5
m : la taille de la couche de sortie (nombre de classes ou de dimensions de la sortie) : 2
Pour Win : Nombre de paramètres : n * d
Pour (Wrec) : Nombre de paramètres : (l - 1)(d * d)
Pour (Wout) : Nombre de paramètres : d * m
Total de paramètres = (n * d) + (l - 1)(d * d) + (d * m)
Total de paramètres = (50 * 64) + (5 - 1) * (64 * 64) + (64 * 2)
Total de paramètres = 3200 + 16384 + 128
Total de paramètres = 19712

Quel sont les intérêts d'utiliser une représentation sémantique des mots ?
R. Les embeddings de mots réduisent considérablement la dimensionnalité des données. Les embeddings de mots permettent aux modèles d'apprendre et de généraliser à partir de contextes similaires. Les mots ayant des contextes similaires auront des vecteurs proches.

Comment pourriez-vous améliorer la précision de votre modèle ?
R. Rechercher et ajuster les hyperparamètres du modèle, tels que la taille de la couche cachée, le nombre de couches, le taux d'apprentissage et la taille du lot, peut conduire à une meilleure performance. Remplacer le RNN simple par des architectures plus avancées, telles que les LSTM (Long Short-Term Memory). Parfois, un entraînement plus long peut aider le modèle à converger vers une meilleure solution.