AMAL - TP 7
Réseaux convolutifs (1D)

Nicolas Baskiotis - Benjamin Piwowarski - Laure Soulier

2020-2021

Introduction (brève, cf cours)

Un réseau convolutif (CNN pour "Convolutional Neural Network") est un type d'architectures essentiellement utilisé pour *la classification séquences* (1D), d'images (2D) ou, en général, de données présentant des symétries. Dans les cas les plus standards, ces symétries font que la façon d'étudier un individu est invariant par rapport *à une translation*. Par exemple, détecter une bouche dans une image ne dépend pas de la position de cette bouche dans l'image ; pour le texte, détecter que l'on parle d'un animal roux ne dépend pas non plus de la position de cet extrait dans le texte.

1 Détection de sentiments

Dans la suite de ce TP, nous allons faire des expériences avec des CNN en 1D pour classifier des séquences mais le principe de fonctionnement est général.

Question 1

Vous utiliserez une segmentation variable avec sentencepiece, en utilisant le fichier `tp7_preprocess.py`. Lancez dès maintenant le processus en spécifiant la taille du vocabulaire souhaitée.

En 1D (figure 1), la convolution peut être vu comme une transformation linéaire qui porte sur `kernel_size`=3 éléments de la séquence (3 dans la figure) répétée en se déplaçant de `stride`=1 éléments à chaque fois.

Chaque transformation linéaire transforme un vecteur de taille `kernel_size` × `in_channels` en un vecteur de taille `out_channels` où `in_channels` est la dimension des données en entrée, et `out_channels` la dimension en sortie. Cette dimension en sortie peut être vu comme le nombre de filtres sur des séquences de taille `kernel_size`. Dans ce TP, vous utiliserez le module `torch.nn.Conv1d` de PyTorch qui implémente ces convolutions de manière efficace.

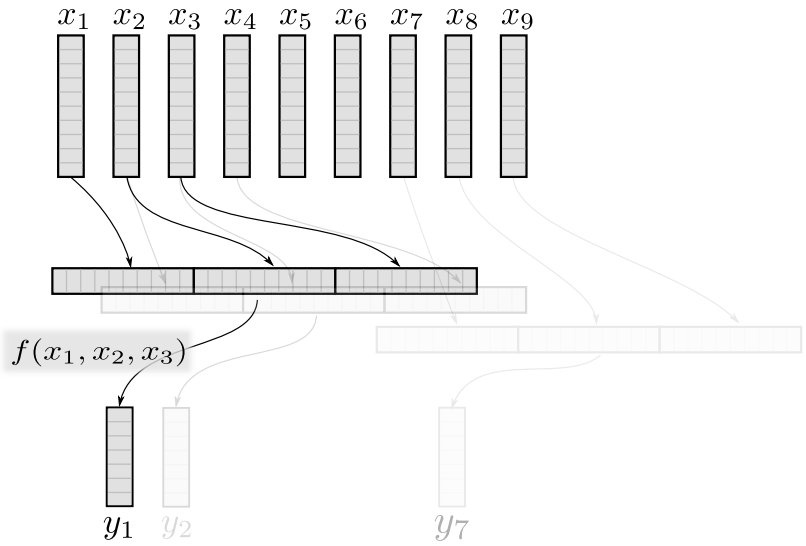

FIGURE 1 – Une convolution 1D (avec $f(x_1, x_2, x_3) = (x_1 \oplus x_2 \oplus x_3)A + b$) **ou** un opérateur de pooling (ex. pour le max-pooling : $f(x_1, x_2, x_3) = (\max_{i=1\ldots3} x_{i1}, \ldots, \max_{i=1\ldots3} x_{id})$)

Une autre couche commune dans les CNNs correspond à un opérateur de *pooling*. Son fonctionnement est similaire à l'opérateur de convolution en tout point sauf sur la fonction utilisée pour transformer. Pour le pooling, il s'agit d'une agrégation qui est effectuée au niveau de chaque composante des vecteurs d'entrée. Par exemple, le *max-pooling* renvoie le maximum (`torch.nn.MaxPool1d`).

L'interaction entre l'opérateur de convolution et de max-pooling est la suivante :

1. La convolution permet de détecter un ensemble de motifs (par exemple, "un bon film", "très bon film" ou bien "un film déplorable") sur des séquences courtes (ici 3 mots) ;
2. Le *max-pooling* permet résumer l'information capturée par les filtres sur une sous-séquence de taille plus grande. Par exemple, un max-pooling de taille 5 et de stride 2 va résumer les détections pour les sous-séquences $(y_1, \ldots, y_7)$, puis $(y_3, \ldots, y_9)$, etc.

Il est ensuite possible de répéter ces opérations un certain nombre de fois, afin de détecter des motifs de taille et complexité croissante. Par exemple, on peut détecter "j' ai beaucoup"/"j' ai énormément"/"ai pas beaucoup" et "ce film"/"cet acteur" (4 filtres) sur la première couche de convolution. La seconde couche de convolution va combiner les informations pour détecter un jugement sur un film ou un acteur.

La dernière couche est généralement un maximum global (sur chaque composante de sortie), suivie d'un classifieur linéaire.

Les données utilisées proviennent de Sentiment140. Les classes sont 0 = negative, 2 = neutral, 4 = positive (convertis en 0, 1 et 2 dans les données pré-traitées).

Question 2

Après avoir inspecté le type de données, vous proposerez des architectures convolutives que vous comparerez au niveau du taux de bonne classification ; vous aurez pris soin de mesurer la performance d'un algorithme trivial qui renvoie systématiquement la classe majoritaire et rapporterez la performance relative à ce modèle.

2 Étude du comportement d'un CNN

Afin d'étudier ce que fait le CNN, nous allons nous intéresser à la dernière couche avant le maximum global ; plus particulièrement, nous allons chercher les sous-séquences (dans le jeu de *train*) qui activent le plus chaque filtre de sortie.

Exemple Pour cela, il faut tout d'abord déterminer à quelle position (dans le texte) correspond chaque sortie. Par exemple, si on considère une convolution avec une taille de noyau 3 et un stride de 1, alors la 1ème sortie correspondra au texte entre les positions 1 et 3, la 2ème à celui entre les positions 2 et 4, etc. Si on ajoute un max-pooling (noyau de taille 2, stride 2), alors la 1ère sortie correspond au texte entre les positions 1 et 4, la 2ème aux positions 3 et 6, etc.

Il faut maintenant généraliser. Pour cela, nous allons considér que la ième opération (convolution/pooling) est définie seulement par la taille du noyau w_i (*kernel width*) et le déplacement s_i (*stride*). Nous nous intéressons aux deux valeurs qui caractérisent l'ensemble des transformations jusqu'a l'opération $i-1$:

— La longueur des entrées correspondant à une sortie W_{i-1}.
— Le déplacement S_{i-1} dans les entrées correspondant à un déplacement unitaire dans les sorties.

Question 3

Donner la formule de récurrence qui permet, étant donné W_i, S_i de déterminer W_{i+1} et S_{i+1} sachant w_{i+1} et s_{i+1}.

Question 4

Soit $(y_1, \ldots, y_L)$ la sortie du CNN. Une fois ce calcul fait, donnez la formule qui, étant donné la position j de la sortie y_j, permet de déterminer les indices correspondant dans la séquence d'entrée.

Question 5

Finalement, parcourez les données du jeu de *train*, et trouvez les sous-séquences qui activent le plus chaque caractéristique de sortie.