

# GT DMD : notes de lecture sur le papier « Graph Attention Networks » (ICLR 2018), Julien Velcin

Papier de Petar Velickovic et Pietro Lio (Cambridge) + 2 personnes de l'UAB (Barcelone) + 2 du MILA à Montréal (groupe de Y. Bengio)

6 Art en mars 2021 ~ 3300 citations

Approche non spectrale pour projeter les nœuds d'un graphe attribué

Le reproche aux approches spectrales est qu'elles dépendent trop de la structure (générale).

La structure de base est une couche appelée Graph Attention Layer

Soit  $\vec{h}_i$  le vecteur de caractéristiques attachées au nœud  $i$  du graphe, alors on cherche à transformer ce vecteur de dimension  $F$  (le # de features) vers un nouvel espace de dimension  $F'$ , avant d'utiliser un mécanisme d'attention vis-à-vis de autres nœuds du graphe :

Si  $i$  et  $j$  sont deux nœuds du graphe :

$$e_{ij} = \alpha(\underbrace{W \cdot \vec{h}_i}_{\text{proj. de } h_i}, \underbrace{W \cdot \vec{h}_j}_{\text{proj. de } h_j \text{ dans un nouvel espace}})$$

mécanisme d'attention ) j'assume le rôle de « contexte »

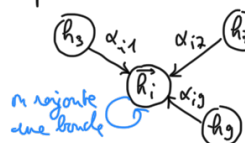
avec  $a: F \times F \rightarrow \mathbb{R}$   
 $(h_i, h_j) \mapsto e_{ij}$

L'idée est de calculer  $e_{ij}$  uniquement sur les nœuds voisins de  $i$  (ce qu'ils appellent l'attention « masquée »)

Il suffit enfin de normaliser afin d'obtenir les coefficients d'attention :

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})} \quad (\text{softmax})$$

La dernière étape consiste alors à agréger l'information des voisins suivant  $\alpha_{ij}$



$$\vec{h}'_i = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W \vec{h}_j\right)$$

Ce calcul correspond à une tête d'attention.

On peut généraliser à  $K$  têtes (multitask), ce qui donne :

$$\vec{h}'_i = \parallel_{k=1}^K \sigma\left(\sum_j \alpha_{ij}^k W^k \vec{h}_j\right)$$

matérialisation

Pour la dernière couche, on moyenne au lieu de concaténer :

$$\vec{h}'_i = \sigma\left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in N_i} \alpha_{ij}^k W^k \vec{h}_j\right) \text{ qui est bien dans un espace à } F' \text{ dimensions}$$

Zoom sur le mécanisme d'attention :

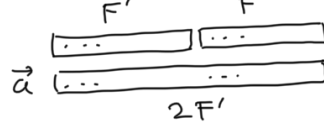
On parle ici de self-attention additive :

( - self-attention car le mécanisme porte sur un même objet en entrée  
 ( - self-attention car le mécanisme porte sur un même objet en entrée  
 ( - self-attention car le mécanisme porte sur un même objet en entrée )

} - additive, suivant la terminologie introduite par Vaswani avec le Transformer  
 L'attention additive a été utilisée par Bahdanau en traduction auto (ICLR 2015).  
 D'autres mécanismes existent, par ex. (scaled) dot product de Vaswani.  
 On a alors :

- 1) concaténer les représentations (transformées)
- 2) apprendre un vecteur de poids  $\vec{a}$
- 3) passer le résultat dans une fonction non linéaire, par ex. ici leaky ReLU

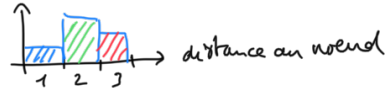
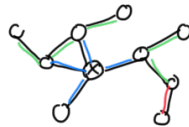
$$e_{ij} = \text{leakyReLU}(\vec{a}^T [W\vec{h}_i || W\vec{h}_j])$$



Le nombre de paramètres à apprendre :  $[(F \times F') + 2F] * K$   
 (avec  $K$  têtes d'attention)

Distinction avec la littérature :

- GCN : on retrouve un GCN si on fixe des poids d'attention uniformes sur le voisinage (immédiat)  
 Les avantages soulignés dans le papier :
  - importance différente des nœuds du voisinage (basé sur les caractéristiques)
  - interprétabilité
  - inductif
- Graph Attention Models (Almi & Haja et al., NeurIPS 2018) : dans la lignée des travaux DeepWalk, ces modèles cherchent à estimer une distribution sur l'importance de la distance dans le graphe pour le tirage de promenades.



- GraphSAGE (Hamilton et al., NeurIPS 2017)  
 pas de fonctionnement clair, mais résultats xp (bien) meilleurs  
 GraphSAGE considère tous les voisins de la même manière (pas de poids)

XP Tâches de classification (single / multi labels)

2 types d'XP réalisées :

- transductif : durant l'apprentissage, utilise l'information de tous les nœuds
- inductif : le test s'effectue sur des graphes qui n'ont pas du tout été vus

L'architecture du réseau dépend du type choisi :

Transductif :

2 couches GAT avec 8 têtes d'attention avec  $F'=8$  (couches 1 à 64 features)

1 tête avec  $C$  features

= # classes pour la tâche à résoudre

régularisation L2 pendant l'apprentissage

avec ELU

Dropout sur : - l'entrée des couches

- coeff  $\propto$  (activation)

→ a pour effet d'échantillonner le voisinage

On note des différences (subtiles) suivant les datasets, ex.  $\lambda$  de L2 différent par task

Inductif:

3 couches GAT avec  $\left. \begin{array}{l} 4 \text{ têtes avec } F' = 256 \\ 4 \text{ têtes avec } F' = 256 \\ 6 \text{ têtes avec } F' = 128 \end{array} \right\} + \text{ELU}$   
+ sigmoid (car classif multilabels)

Ni régularisation ni dropout

Skip connections

batches de 2 graphes (4000 nœuds)

Les data:

fonction de coût : cross-entropy

initialisation Glorot (ASTAT 2010)

optimiseur Adam SGD avec de l'early stopping

Datasets:

pour le transductif : 3 bases d'articles scientifiques (CORA, Citeseer, PubMed)  
les features = B.o.W  
le graphe = citations

compétiteurs : GCN, Node2Vec, DeepWalk, L.P

pour l'inductif : PPI = 24 graphes d'interactions entre protéines

compétiteurs : différentes versions de GraphSAGE (Hamilton 2017)

Notes supplémentaires :

leaky ReLU : proba  $\downarrow$  de gradient vanishing (comme ReLU)



résout le PB de « dying ReLU »  
(grande partie du réseau « morte »)  
accélère l'apprentissage

Quelques papiers récents basés sur GAT:

- G2T: generating fluent descriptions for knowledge graphs (SIGIR'20)  
GAT utilisé comme baseline
- GAT-Based User representation learning for unifying robust reco and fraudster detection (SIGIR'20)  
même système d'attention inspiré par GAT dans le GCN