LanczosNet : Multi-Scale Deep Graph Convolutional Networks [1]

Pierre Simon Yuan Yin Zhongtian Xiao

Sorbonne Université

Introduction

Le graphe est une catégorie de données courante dans le monde réel. Modéliser le graphe avec les réseaux de neurones est l'une des problématiques les plus étudiées de l'apprentissage profond. Les tâches à résoudre concernent principalement l'apprentissage supervisé et l'apprentissage semi-supervisé pour la classification et la regression de nœuds ou/et de graphes. L'état de l'art du domaine est principalement constitué de deux familles de modèles : les réseaux récurrents et le réseaux convolutionnels.

Dans cet article, les auteurs ont proposé un réseau type convolutionnel appelé le LanczosNet, basé sur la théorie de traitement de signal des graphes, qui permet de construire un réseau capable de pondérer les informations multi-échelles.

Contexte

Étant donné un graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$ à N nœuds où A est une matrice d'ajacence, et les features de noeuds $X \in \mathbb{R}^{N\times F}$. La matrice d'affinité S du graphe peut être calculée à partir de la matrice A. Étant symétrique positive, la matrice S permet d'exprimer le graphe de manière efficace et d'en extraire des informations spectrales.

L'idée derrière le réseau convolutionnel de graphe est que d'abord on effectue une décomposition spectrale en prenant $S = V\Lambda V^\intercal$ où Λ comprend les valeurs propres et V les vecteurs propres qui représentent le domaine spectral. Cela permet de projeter les features du domaine original dans le domaine spectral et d'effectuer ensuite un filtrage.

Ensuite on procède au traitement de données directement dans le domaine spectral avec le filtre polynomial localisé (Localized Polynomial Filter) qui prend en compte pour chaque nœud les nœuds à τ pas près. L'une des modelisations possibles est le filtre polynomial de Tchebychev (Chebyshev polynomial filter) en prenant d'abord la récursion polynomiale de Tchebychev pour capturer les features à τ pas près :

$$\tilde{X}(t) = \begin{cases} X & \text{si } t = 0, \\ SX & \text{si } t = 1, \\ 2S\tilde{X}(t-1) & \text{sinon.} \end{cases}$$

Puis le filtrage est présenté comme suit :

$$Y = \sum_{t=1}^{\tau} \left[\tilde{X}(0) \cdots \tilde{X}(t-1)\right] W_t$$

Ceci est équivalent à appliquer directement un filtre global :

$$Y = \left[\tilde{X}(0) \cdots \tilde{X}(\tau-1)\right] W$$

Le LanczosNet est basé sur l'observation que chaque colonne du filtre de Tchebychev se trouve dans un sous-espace Krylov. Cela donne directement la motivation pour utiliser les décompositions dans les sous-espaces Krylov comme l'algorithme Lanczos.

Algorithme Lanczos

La matrice d'affinité peut être décomposée en $S \approx QTQ^\intercal$, où T est dans l'espace spectral, dans lequel on peut appliquer directement un filtre. L'utilisation de l'algorithme Lanczos est basée sur le fait que le filtre opère dans un sous-espace de Krylov. L'algorithme permet de calculer les informations spectrales T, et une base orthogonale de Krylov Q pour la transformation de Fourier et son inverse.

LanczosNet et AdaLanczosNet

Étant donné que le calcul pour agréger les informations à un paramètre d'échelle élevé, par exemple $S^{50}Z$, est trop coûteux, il est plus intéressant d'apprendre directement un filtre sur les informations spectrales, plus précisément les valeurs Ritz R ou la matrice tridiagonale T, en sortie de l'algorithme Lanczos. La décomposition $S \approx QTQ^\intercal \approx VRV^\intercal$ est une approximation de la matrice S prouvée dans le papier.

LanczosNet

Le LanczosNet est une instanciation du filtre polynomial localisé. Pour le filtre à échelles courtes $\mathcal{S}$, on utilise la matrice S exacte. Pour le filtre à échelles longues, on utilise l'approximation de la matrice $S \approx VRV^\intercal$. Le filtre spectral est ensuite modélisé par un réseau de neurones qui donne $\hat{R}$ filtrée en s'appliquant à R.

Le filtrage pour chaque couche est donc présenté comme suit :

$$Y = \left[X\ S^{\mathcal{S}_1}X \cdots\ S^{\mathcal{S}_{M_{\text{short}}}}X\ V\hat{R}(\mathcal{L}_1)V^\intercal X \cdots\ V\hat{R}(\mathcal{L}_{M_{\text{long}}})V^\intercal X\right] W$$

AdaLanczosNet

Dans l'AdaLancozNet, la matrice d'affinité S à approximiser est celle des features de nœuds en annulant les arêtes non présentes dans le graphe. Le filtre à échelle longue repose sur l'approximation $S \approx QTQ^\intercal$, et le filtre spectral est appliqué à la matrice tridiagonale T. De la même manière que dans le LanczosNet, un réseau de neurones filtre la matrice T et donne $\hat{T}$ pour former le filtrage suivant :

$$Y = \left[X\ S^{\mathcal{S}_1}X \cdots\ S^{\mathcal{S}_{M_{\text{short}}}}X\ Q\hat{T}(\mathcal{L}_1)Q^\intercal X \cdots\ Q\hat{T}(\mathcal{L}_{M_{\text{long}}})Q^\intercal X\right] W$$

Algorithmes

Algorithme 1 : LanczosNet

Données :
$X \in \mathbb{R}^{N\times F_0}$ features de nœuds,
$S \in \mathbb{R}^{N\times N}$ matrice d'affinité,
$V \in \mathbb{R}^{N\times K}$ vecteurs Ritz,
$R \in \mathbb{R}^{K}$ valeurs Ritz,
$\mathcal{S}$ paramètres d'échelles courtes,
$\mathcal{L}$ paramètres d'échelles longues
Résultat : $Y \in \mathbb{R}^{N\times F_c}$

$Y_0 \leftarrow X$;
pour $l = 1, 2, \cdots, c$ **faire**
  $\mathcal{Z} \leftarrow []$;
  $Z \leftarrow Y_{l-1}$;
  pour $j = 1, \ldots, \max(\mathcal{S})$ **faire**
    $Z \leftarrow SZ$;
    si $j \in \mathcal{S}$ **alors**
      $\mathcal{Z}$.append(Z) ;
  pour $i \in \mathcal{I}$ **faire**
    $\mathcal{Z}$.append($V\hat{R}(i)V^\intercal Y_{l-1}$) ;
  $Y_l \leftarrow \text{concat}(\mathcal{Z})W_l$;
  si $l < c$ **alors**
    $Y_l \leftarrow \text{dropout}(\sigma(Y_l))$;

Algorithme 2 : Algorithme Lanczos

Données :
$S \in \mathbb{R}^{N\times N}$ matrice d'affinité,
$x \in \mathbb{R}^{N}$ vecteur de base,
K nombre de pas Lanczos,
ϵ limite de norme
Résultat : $T \in \mathbb{R}^{K\times K}, Q \in \mathbb{R}^{N\times N}$

$\beta_0 \leftarrow 0, q_0 \leftarrow 0, q_1 \leftarrow \frac{x}{\|x\|}$;
pour $j = 1, 2, \cdots, K$ **faire**
  $z \leftarrow Sq_j$;
  $\gamma_j \leftarrow \langle q_j, z\rangle$;
  $z \leftarrow z - \gamma_j q_j - \beta_{j-1}q_{j-1}$;
  $\beta_j \leftarrow \|z\|$;
  si $\beta_j < \epsilon$ **alors**
    quitter ;
  $q_{j+1} \leftarrow \frac{z}{\beta_j}$
$Q \leftarrow [q_1, \ldots, q_K]$;
$T \leftarrow \begin{bmatrix} \gamma_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{K-1} \\ & & \beta_{K-1} & \gamma_K \end{bmatrix}$

Jeux de données

Nous avons effectué les expérimentations sur 4 jeux de données, dont 2 pour la régression, 2 pour la classification de nœuds.

Régression de graphe

Il s'agit de deux tâches de chimie quantique sur QM8, utilisé dans le papier, et QM9, le nouveau jeu, pour prédire les quantités de spectres électroniques et l'énergie pour chaque multi-graphe de molécule. QM9 dispose de 6 fois plus de molécules que QM8.

Classification de nœuds

La classification de nœuds est effectuée sur les jeux de données de Cora, un réseau de citation, et Terrorist Attack, un réseau de connections entre attentats. Les tâches ont pour objectif de prédire la classe d'un document ou le type d'attentat d'un événement.

Résultats

Les résultats des expérimentations sont obtenus avec les deux modèles LanczosNet et AdaLanczosNet que nous avons implémentés et d'autres modèles implémentés par les auteurs de l'article [a].

MAE	GGNN	DCNN	LNet	AdaLNet
QM8/Val	12.98e-3	10.18e-3	9.78e-3	9.69e-3
QM8/Test	12.85e-3	9.86e-3	9.62e-3	9.60e-3
QM9/Val	3.80	3.24	3.03	2.81
QM9/Test	3.84	3.31	3.04	2.93

Table 1 – Regression de graphe sur QM8 et QM9

Accuracy	GGNN	DCNN	LNet	AdaLNet
Cora/Val	81.7	87.3	81.6	81.3
Cora/Test	77.5	80.1	80.1	80.0
Terrorist Attack/Val	80.6	78.3	77.5	79.0
Terrorist Attack/Test	78.8	73.8	76.2	73.9

Table 2 – Classification semi-supervisée sur Cora et Terrorist Attack

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Références

[1] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard Zemel. Lanczosnet : Multi-scale deep graph convolutional networks. In *ICLR*, 2019.

a. `https://github.com/lrjconan/LanczosNetwork`