Detection d'anomalies contextuelles dans un graphe attribué

Rémi Vaudaine, Baptiste Jeudy, Christine Largeron

Univ Lyon, UJM-Saint-Etienne, CNRS, Institut d Optique Graduate School
Laboratoire Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France
{remi.vaudaine,baptiste.jeudy,christine.largeron}@univ-st-etienne.fr

Résumé. La détection d'anomalies dans des données relationnelles modélisées par un graphe s'est avérée très utile dans un large éventail de domaines, par exemple pour détecter des comportements frauduleux sur des plateformes en ligne ou des intrusions sur des réseaux de télécommunication. Cependant, la plupart des méthodes existantes utilisent des prédicteurs pré-construits à partir du graphe et n'exploitent pas nécessairement des informations locales. Pour surmonter ces limites, nous proposons CoBaGAD, un détecteur d'anomalies, basé sur le contexte, qui exploite les informations locales pour détecter les noeuds anormaux d'un graphe de manière semi-supervisée. Ce modèle de réseaux de neurones, inspiré du Graph Attention Network (GAT), avec un mécanisme d'attention personnalisé permet de créer des représentations des noeuds, de les agréger et de classer les noeuds non étiquetés en normal ou anormal. Les résultats expérimentaux ont montré que CoBaGad surpasse les méthodes de pointe en terme de rappel et de précision. [1]

1 Introduction

La détection d'anomalies est un domaine de recherche intense ces dernières décennies, aussi bien pour les données relationnelles (Akoglu et al. (2015)) que vectorielles (Mehrotra et al. (2017)). Même s'il n'existe pas de définition consensuelle de ce qu'est une anomalie, elle est souvent décrite comme un écart par rapport à la norme. Dans un ensemble de graphes, l'anomalie peut concerner un des graphes alors que dans un seul graphe, elle peut se situer au niveau d'un sous-graphe, des liens ou encore des sommets. C'est ce dernier cas qui sera considéré dans la suite de cet article.

Les méthodes proposées dans la littérature visent rarement à trouver le même type d'anomalies mais elles peuvent être classées en différentes catégories selon le type de graphe qu'elles traitent. Le premier type est le graphe simple dans lequel des algorithmes de clustering tels que Scan (Xu et al. (2007)) ou PAICAN (Bojchevski et Günnemann (2018)) identifient les classes tout en détectant les anomalies. Ces anomalies sont généralement des ponts entre les communautés comme dans (Wang et Davidson (2009)). De son côté, OddBall (Akoglu et al. (2010)) est basé sur des lois de puissance et détecte des noeuds du graphe dont les caractéristiques

1. Cet article est une traduction de Vaudaine et al. (2021)

s'écartent de ces lois de puissance, par exemple des noeuds à très haut degré ou dont le voisinage est particulièrement dense. D'autres méthodes (Müller et al. (2013) Perozzi et Akoglu (2018) Perozzi et al. (2014)) ont été proposées pour traiter les graphes attribués où les noeuds sont caractérisés par des attributs. Ainsi, dans FocusGO (Perozzi et al. (2014)) ou (Perozzi et Akoglu (2018)) les noeuds aberrants appartiennent structurellement à une communauté mais s'écartent de ses attributs.

Dans le cadre de cet article, nous introduisons la notion d'anomalie contextuelle, qui à notre connaissance n'a pas été étudiée dans la littérature, bien qu'ellle soit relativement fréquente en pratique, et souvent associée à la fraude ou à la corruption. Par exemple, si une entreprise a un PDG ou un comptable qui a un compte dans un paradis fiscal, alors elle a plus de risque d'être elle même frauduleuse. Pour définir formellement ce type d'anomalie, nous considèrons qu'il existe un petit sous-graphe inconnu (un contexte), et un noeud distingué dans ce sous-graphe, tel qu'à chaque fois que ce sous-graphe apparaît dans les données, alors le noeud correspondant au noeud distingué est une anomalie. Nous appelons ce sous-graphe inconnu un contexte et les anomalies correspondantes « anomalies basées sur le contexte ou contextuelles». Des exemples illustratifs sont fournis dans la table 1.

Nous proposons de détecter ces anomalies contextuelles dans un cadre transductif semi-supervisé : les données sont constituées d'un seul graphe pour lequel une proportion, éventuellement petite, des noeuds est étiquetée (soit « anomalie » ou « normal ») et les autres noeuds n'ont pas d'étiquettes. Le but est de trouver les étiquettes des noeuds non étiquetés.

Pour ce faire, nous proposons CoBaGAD (Context-Based Graph Anomaly Detector), un réseau de neurones permettant d'apprendre automatiquement une représentation adaptée des noeuds et, simultanément de détecter les anomalies. Inspiré des modèles de type Graph Attention Networks (GAT), CoBaGAD possède un mécanisme d'attention permettant de construire des représentations plus adaptées à la tâche à résoudre, comme l'ont confirmé les expérimentations que nous avons réalisées sur différents types de graphes avec différents cas d'anomalies pour comparer les performances de CoBaGad avec les méthodes de l'état de l'art telles que GAT, GCN, GraphSage et une combinaison de Node2vec et LOF.

L'article est organisé de la manière suivante. Nous définissons le problème et présentons notre méthode de détection d'anomalies contextuelles dans les graphes attribués dans la section 2. Ensuite, nous décrivons notre protocole d'évaluation et les expérimentations réalisées pour évaluer la capacité de notre méthode à détecter ces anomalies dans la section 3.

2 Notre méthode CoBaGAD

Notre objectif est de détecter des anomalies contextuelles qui sont des noeuds du graphe dont le contexte local présente un arrangement singulier. Plus formellement, soit $G(V, E, X)$ un graphe défini par un ensemble de n noeuds $V = \{v_i\}$, un ensemble d'arêtes $E = \{e_{ij}\}$ et une matrice de caractéristiques $X \in \mathbf{R}^{n \times F}$. Chaque ligne $\vec{x}_i$ dans la matrice X est le vecteur de caractéristiques catégorielles du noeud v_i (sexe, genre, etc). On considère qu'il existe un petit sous-graphe inconnu (un contexte), et un noeud distingué dans ce sous-graphe, tel qu'à chaque fois que ce sous-graphe apparaît dans les données, alors le noeud correspondant au noeud distingué est une anomalie (plusieurs exemples sont fournis dans la section 3).

Nous utilisons un cadre transductif : les données consistent en un seul graphe pour lequel une proportion des noeuds est étiquetée (soit « anomalie » ou « normal ») et les autres noeuds

n'ont pas d'étiquettes. Le but est de trouver les étiquettes des noeuds non étiquetés. Cependant et surtout, les conditions (c'est-à-dire le contexte) qui rendent un noeud anormal ne sont pas connues lors de l'apprentissage du modèle. L'idée principale de notre méthode est d'apprendre simultanément un classifieur à deux classes avec des mécanismes d'attention. Pour cela, les informations locales sont agrégées pour déterminer si un noeud est normal ou non. Les paramètres du réseau sont appris avec une fonction de perte standard (entropie croisée) de façon semi-supervisée. CoBaGad comporte quatre étapes :

Transformation affine : La première étape est une transformation affine suivie d'une fonction non linéaire σ. Cette fonction σ est appliquée élément par élément. Les paramètres sont la matrice $W \in \mathbf{R}^{F \times F'}$ et un vecteur ligne $b \in \mathbf{R}^{1 \times F'}$. La matrice d'identité est notée $\mathbb{1}$.

$$\Lambda = \sigma\left(XW + \mathbb{1}b\right) \tag{1}$$

Cette étape permet de transformer les attributs des noeuds $\vec{x}_i$ indépendamment pour chaque noeud v_i. La $i$ème ligne de Λ est la nouvelle représentation du noeud v_i, et notez que puisque la matrice W n'est pas nécessairement carrée, cette nouvelle représentation peut avoir plus ou moins de caractéristiques que $\vec{x}_i$.

Couche d'attention : La couche d'attention se compose de k têtes d'attention (k est un hyper-paramètre). Pour chaque tête d'attention $c \in \{0, ..., k-1\}$, nous effectuons une transformation linéaire locale suivie d'un agrégat pondéré. Tout d'abord, une transformation linéaire locale $W_c \in \mathbf{R}^{F' \times F'}$ est appliquée sur les features :

$$\Lambda_c = \Lambda W_c \tag{2}$$

Ensuite pour chaque lien $(i, j) \in E$, la valeur $e_{i,j,c}$ est calculée par :

$$e_{i,j,c} = LeakyReLU\left((\vec{\lambda}_{i,c} \odot \vec{\lambda}_{j,c}).\vec{u}_c\right) \tag{3}$$

où $\odot$ est le produit Hadamard, $\vec{\lambda}_{i,c}$ et $\vec{\lambda}_{j,c}$ sont resp. la $i$ème et $j$ème ligne de Λ_c, et $\vec{u}_c \in \mathbf{R}^{F'}$ est un vecteur de paramètres. Cela correspond à la similarité du produit scalaire pondérée par $\vec{u}_c$.

Les poids d'attention $\alpha_{i,j,c}$ sont définis comme une version normalisée des $e_{i,j,c}$ tel que pour chaque noeud v_i et chaque tête c, ils soient positifs et de somme égale à 1 :

$$\alpha_{i,j,c} = \frac{\exp(e_{i,j,c})}{\sum_{k \in N(i)} \exp(e_{i,k,c})} \tag{4}$$

où $N(i)$ est l'ensemble des voisins du noeud v_i.

L'étape suivante consiste à calculer pour chaque noeud v_i, une combinaison convexe des $\vec{\lambda}_{j,c}$ pour tous les voisins v_j de v_i en utilisant ces poids d'attention. Ces poids peuvent être considérés comme l'intensité de la relation entre les noeuds adjacents. Enfin, la nouvelle représentation $\vec{h_i}$ du noeud v_i est donnée par la concaténation de toutes les représentations données par les k têtes d'attention, i.e., chaque noeud v_i est représenté par un vecteur ligne de $F'k$ composantes :

$$\vec{h_{i,c}} = \sum_{j \in N(i)} \alpha_{i,j,c} \vec{\lambda}_{j,c} \text{ et } \vec{h_i} = \sigma'\left(\|_{c=0}^{k-1} \vec{h_{i,c}}\right) \tag{5}$$

où σ'est une fonction d'activation et $\|$ est la concaténation.

Le réseau peut comporter plusieurs de ces couches d'attention, la sortie de chaque couche étant l'entrée de la couche suivante.

Classification : Pour détecter les anomalies, nous considérons que chacune des têtes de la dernière couche est un classificateur à 2 classes (donc chaque $\vec{h_{i,c}} \in \mathbf{R}^2$) et nous combinons ces classificateurs en prenant le argmax. c'est-à-dire que si la composante maximale dans le vecteur $\vec{h_i}$ est dans un indice impair, v_i est classé comme une anomalie. Si le maximum est dans un index pair, alors c'est un noeud normal.

Les paramètres qui doivent être appris sont : W, b, et pour chacune des k têtes d'attention dans chaque couche d'attention : la matrice W_c et le vecteur $\vec{u_c}$. Les hyper-paramètres sont F', le nombre de couches d'attention et le nombre k de têtes dans chaque couche d'attention. Les fonctions d'activation σ et σ' peuvent également être choisies par l'utilisateur.

CoBaGAD diffère du GAT par deux points majeurs. Tout d'abord, nous avons ajouté une transformation affine globale (Eq.1) pour projeter les caractéristiques d'origine dans un nouvel espace. Cette opération améliore la capacité de détection des anomalies et permet de réduire la dimension du problème. Ensuite, nous utilisons un mécanisme d'attention personnalisé indiqué dans l'équation Eq.3. Plutôt que de concaténer les nouvelles représentations d'une paire de noeuds, nous calculons une similarité entre elles avec le produit Hadamard pondéré par $\vec{u_c}$.

3 Expériences

Jeux de données : Comme il n'existe pas de jeux de données réelles publiquement accessibles, pour évaluer expérimentalement notre modèle et le comparer à l'état de l'art, nous avons introduit artificiellement des anomalies contextuelles dans de nombreux graphes différents, réels ou synthétiques. La table 1 donne le nom, le nombre de noeuds et le nombre d'arêtes de ces graphes. G_0 est un graphe aléatoire d'Erods-Renyi. G_1 et G_4 sont également générés, respectivement avec Dancer (Largeron et al. (2017)) et LFR (Lancichinetti et al. (2008)), qui imitent le comportement de graphes du monde réel. De plus, nous avons choisi des graphes réels qui sont courants dans la littérature : Polblogs [2], Cora [3] et Facebook [4]. La génération d'anomalies dans les graphes est une contribution à part entière trop longue pour figurer dans ce papier. Pour plus de détail sur le générateur de graphes avec anomalies que nous avons développé, nous renvoyons le lecteur à Vaudaine et al. (2021).

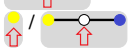
Dans nos expériences, nous avons introduit dans chaque graphe un faible pourcentage, entre 4% et 6% de noeuds qui sont considérés comme anormaux parce qu'ils apparaissent dans un des contextes décrits dans le tableau 1 où $\bar{Y}$ signifie que les anomalies sont des noeuds de couleur $Jaune$ alors que B signifie que les anomalies ont au moins un voisin dont la couleur est $Bleue$. Par exemple, A_6 représente des noeuds de couleur jaune (Y) ou ayant au moins un voisin de couleur bleue (B) et au moins un voisin de couleur jaune (Y).

Il convient de rappeler que notre algorithme CoBaGAD ne sait pas comment les anomalies ont été créées et son but est de reconnaître ces anomalies sans cette connaissance contextuelle.

Protocole expérimental : Tous les noeuds d'un graphe appartiennent soit à l'ensemble des anomalies, soit à l'ensemble des noeuds normaux. Dans une configuration transductive,

2. http ://konect.cc/networks/dimacs10-polblogs/
3. https ://relational.fit.cvut.cz/ dataset/CORA
4. https ://snap.stanford.edu/data/egonets-Facebook.html

Graphe	Nom	Noeuds	Arêtes
G_0	Erdos-Renyi	10000	24907
G_1	Dancer	10000	189886
G_2	Facebook	4039	88234
G_3	Polblogs	1224	16715
G_4	LFR	1000	5622
G_5	Cora	1433	5429

Anomalie	Définition	Contexte
A_0	$B \wedge G$	
A_1	$(B \wedge G) \vee (B \wedge R)$	
A_2	$(B \wedge G) \vee (Y \wedge R)$	
A_3	$\bar{Y}$	
A_4	$\bar{Y} \wedge B$	
A_5	$Y \wedge \bar{Y} \wedge B$	
A_6	$\bar{Y} \vee (Y \wedge B)$	

TAB. 1 – *(Gauche) Caractéristiques des jeux de données. (Droite) Types d'anomalies.* $B =$ *blue,* $G =$ *green,* $R =$ *red,* $Y =$ *yellow,* $P =$ *purple.* $\bar{Y}$ *signifie que les anomalies ont la couleur* Y*.* B *signifie que les anomalies ont au moins un voisin de couleur* B*. Le noeud distingué dans le graphe contexte est indiqué avec une flèche.*

les noeuds sont divisés en ensembles d'apprentissage, de validation et de test. Le jeu d'apprentissage est constitué de 50% du total des anomalies auquelles on ajoute autant de noeuds normaux qu'il y a d'anomalies. Il en est de même pour le jeu de validation et celui de test, chacun défini avec 25% d'anomalies et complété de 25% de noeuds normaux du graphe. Le code et les jeux de données sont disponibles sur GitHub [5].

Nous comparons notre algorithme, CoBaGAD, avec des méthodes de pointe en classification de noeuds : Graph Convolution Networks (Kipf et Welling (2017)) (GCN), Graph Attention Networks (Veličković et al. (2018)) (GAT), GraphSAGE (Hamilton et al. (2017)) avec agrégateur de moyennes et, une approche de détection d'anomalies non supervisée basée sur une combinaison de Node2vec (Grover et Leskovec (2016)) et LOF (Breunig et al. (2000)) dont les résultats ne seront pas présentés car la précision était très faible comme on pouvait s'y attendre (entre 0% et 20%) mais ils figurent dans le GitHub [6]. Bien que difficilement calculable, toutes ces méthodes utilisant des réseaux de neurones pour les graphes ont une complexité du même ordre. Aussi, pour chaque méthode d'apprentissage en profondeur, nous apprenons une seule couche. Compte tenu du type de motif étudié, ajouter plus de couches ne semble pas pertinent. Pour CoBaGAD, nous utilisons GELU (Hendrycks et Gimpel (2020)) comme fonction d'activation σ et softmax comme fonction d'activation σ', $F' = 2$, $k = 2$ car nous apprenons deux classifieurs à 2 classes pour les anomalies et les noeuds normaux. Pour GAT et GraphSAGE, nous utilisons les mêmes paramètres. Pour GCN, nous utilisons le filtre *localpool*, softmax comme fonction d'activation et une sortie de dimension 2. Pour chaque algorithme, nous avons essayé deux versions : sans boucle et avec (en ajoutant la matrice identité à la matrice d'adjacence) et nous présentons les meilleurs résultats. Les modèles sont entrainés avec 1000 époques avec l'optimiseur Adam et un taux d'apprentissage de $5e-3$. Les poids des réseaux sont conservés lorsque la précision sur l'ensemble de validation est la plus élevée (early-stopping). Comme fonction de perte, nous utilisons l'entropie croisée catégorique standard définie par : $L(Y^{true}, Y^{pred}) = -\sum_{j=1}^{k}\sum_{i}^{N}(y_{ij}^{true} \times log(y_{ij}^{pred}))$.

5. https ://github.com/vaudaine/Detection-of-contextual-anomalies-in-attributed-graphs

6. Un plongement du graphe est réalisé via Node2vec avec $p = 1, q = 1$ dans un espace de dimension 128. Cette nouvelle représentation de chaque noeud v_i est concaténée avec son vecteur caractéristique $\vec{x_i}$. Enfin, ce vecteur concaténé alimente un classificateur LOF afin de détecter des anomalies.

3.1 Résultats

Pour chaque modèle, chaque jeu de données et chaque type d'anomalie, les expérimentations sont réalisées 12 fois en changeant le découpage entre apprentissage/validation/test. Nous choisissons les trois meilleurs résultats lors de la validation et rapportons la moyenne et l'écart-type de précision obtenus sur l'ensemble de test dans le tableau 2. Pour des contraintes d'espace, le rappel sur les anomalies, le rappel et la précision sur les nœuds normaux ne sont pas rapportés. Ils sont en effet très élevés pour tous les graphes et tous les types d'anomalies (la plupart du temps supérieurs à 98%) et aucune différence significative ne peut être observée entre les méthodes. Ces résultats sont cependant disponibles dans le matériel supplémentaire.

A_0	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.98 \pm 0.03}$	$\mathbf{0.96 \pm 0.03}$	$\mathbf{0.95 \pm 0.03}$	$\mathbf{0.87 \pm 0.09}$	$\mathbf{0.85 \pm 0.13}$	0.92 ± 0.12
GAT	0.96 ± 0.04	0.83 ± 0.02	0.33 ± 0.08	0.55 ± 0.08	0.59 ± 0.13	$\mathbf{0.93 \pm 0.1}$
GCN	0.34 ± 0.02	0.12 ± 0.01	0.11 ± 0.0	0.18 ± 0.02	0.31 ± 0.03	0.26 ± 0.02
GraphSage	0.53 ± 0.01	0.56 ± 0.03	0.63 ± 0.05	0.44 ± 0.08	0.36 ± 0.08	0.52 ± 0.03

A_1	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.96 \pm 0.03}$	$\mathbf{0.98 \pm 0.02}$	$\mathbf{0.78 \pm 0.1}$	$\mathbf{0.73 \pm 0.05}$	$\mathbf{0.72 \pm 0.15}$	$\mathbf{0.85 \pm 0.12}$
GAT	0.74 ± 0.3	0.55 ± 0.21	0.51 ± 0.14	0.63 ± 0.07	0.46 ± 0.19	0.8 ± 0.18
GCN	0.34 ± 0.02	0.19 ± 0.03	0.13 ± 0.02	0.19 ± 0.03	0.34 ± 0.04	0.26 ± 0.01
GraphSage	0.46 ± 0.04	0.5 ± 0.02	0.51 ± 0.06	0.39 ± 0.06	0.38 ± 0.07	0.47 ± 0.03

A_2	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.61 \pm 0.04}$	$\mathbf{0.62 \pm 0.06}$	$\mathbf{0.58 \pm 0.2}$	$\mathbf{0.47 \pm 0.14}$	$\mathbf{0.64 \pm 0.15}$	$\mathbf{0.72 \pm 0.06}$
GAT	0.52 ± 0.02	0.42 ± 0.1	0.29 ± 0.03	0.35 ± 0.06	0.3 ± 0.03	0.51 ± 0.11
GCN	0.27 ± 0.02	0.12 ± 0.0	0.14 ± 0.01	0.2 ± 0.04	0.25 ± 0.02	0.23 ± 0.03
GraphSage	0.33 ± 0.01	0.38 ± 0.02	0.44 ± 0.03	0.44 ± 0.05	0.33 ± 0.03	0.38 ± 0.01

A_3	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	0.99 ± 0.01	$\mathbf{0.99 \pm 0.01}$	0.95 ± 0.05	$\mathbf{0.91 \pm 0.13}$	0.89 ± 0.15	0.93 ± 0.02
GAT	0.82 ± 0.07	0.79 ± 0.03	0.78 ± 0.07	0.55 ± 0.03	0.43 ± 0.01	0.61 ± 0.09
GCN	0.95 ± 0.02	0.97 ± 0.02	0.91 ± 0.07	0.8 ± 0.21	$\mathbf{0.94 \pm 0.08}$	0.78 ± 0.04
GraphSage	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.97 \pm 0.04}$	0.71 ± 0.1	0.68 ± 0.24	$\mathbf{0.97 \pm 0.03}$

A_4	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.9 \pm 0.1}$	$\mathbf{0.95 \pm 0.04}$	$\mathbf{0.97 \pm 0.04}$	$\mathbf{0.8 \pm 0.07}$	0.62 ± 0.1	$\mathbf{0.89 \pm 0.09}$
GAT	0.5 ± 0.07	0.77 ± 0.15	0.77 ± 0.03	0.67 ± 0.23	0.51 ± 0.1	0.43 ± 0.05
GCN	0.44 ± 0.01	0.7 ± 0.03	0.71 ± 0.12	0.6 ± 0.12	$\mathbf{0.65 \pm 0.07}$	0.35 ± 0.02
GraphSage	0.46 ± 0.03	0.73 ± 0.02	0.72 ± 0.02	0.58 ± 0.11	0.61 ± 0.04	0.41 ± 0.03

A_5	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.84 \pm 0.02}$	$\mathbf{0.9 \pm 0.03}$	$\mathbf{0.82 \pm 0.07}$	$\mathbf{0.61 \pm 0.03}$	0.51 ± 0.23	$\mathbf{0.84 \pm 0.22}$
GAT	0.69 ± 0.11	0.74 ± 0.07	0.71 ± 0.09	0.57 ± 0.07	0.52 ± 0.09	0.46 ± 0.1
GCN	0.35 ± 0.01	0.67 ± 0.04	0.62 ± 0.07	0.6 ± 0.04	0.46 ± 0.13	0.23 ± 0.02
GraphSage	0.34 ± 0.01	0.68 ± 0.0	0.68 ± 0.06	0.51 ± 0.04	$\mathbf{0.56 \pm 0.06}$	0.3 ± 0.04

A_6	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.9 \pm 0.01}$	$\mathbf{0.89 \pm 0.05}$	$\mathbf{0.54 \pm 0.27}$	$\mathbf{0.62 \pm 0.2}$	0.26 ± 0.09	0.4 ± 0.04
GAT	0.55 ± 0.12	0.59 ± 0.12	0.39 ± 0.05	0.48 ± 0.1	0.27 ± 0.03	0.39 ± 0.03
GCN	0.38 ± 0.05	0.26 ± 0.09	0.2 ± 0.07	0.21 ± 0.01	0.32 ± 0.13	0.46 ± 0.11
GraphSage	0.67 ± 0.04	0.58 ± 0.05	0.48 ± 0.06	0.49 ± 0.16	$\mathbf{0.48 \pm 0.06}$	$\mathbf{0.65 \pm 0.14}$

TAB. 2 – *Precision de la détection des anomalies A_0-A_6 sur les graphes (G_0-G_5) sur l'échantillon test.* ***gras*** *: meilleur résultat.*

Ils montrent que notre algorithme atteint les meilleures performances dans 33 cas sur 42, tous types d'anomalies et de graphes confondus. Plus précisément, pour A_0, A_1 et A_2, notre méthode surpasse toujours les autres concurrents (sauf pour A_0, G_5 où la différence avec le meilleur modèle, GAT, n'est pas significative). Il surpasse GAT, notre principal concurrent, d'au moins 2% sur A_0, G_0 jusqu'à 62% sur A_0, G_2. Les méthodes basées sur l'attention (GAT et CoBaGaD) s'avèrent meilleures que les autres (GCN, GraphSage, Node2vec + LOF).
Les anomalies A_3 à A_6 contiennent le motif $\bar{Y}$ ce qui signifie que les nœuds anomalies doivent être jaunes. Cela signifie que les informations sur le nœud lui-même sont requises. A_3 est un motif très simple où les anomalies sont définies par le motif le plus simple : les nœuds sont simplement jaunes. Dans ce cas, nous pouvons supposer qu'il est facile pour de nombreux

algorithmes de bien détecter ces nœuds. En fait, GraphSAGE affiche de bonnes performances pour la plupart des graphes sauf G_3 et G_4. Les résultats de GCN sont plus homogènes mais les résultats sont généralement moins bons que ceux fournis par GraphSAGE. Alors que GAT ne montre pas de bonnes performances, notre méthode CoBaGaD s'avère la plus homogène sur tous les graphes et montre de très bons résultats en général. Ensuite, pour A_4 à A_6, au fur et à mesure que le motif se complexifie, CoBaGAD reste la seule méthode qui, à quelques exceptions près, détecte correctement les anomalies.

4 Conclusion

Nous avons défini un nouveau type d'anomalie basé sur le contexte. Ensuite, nous avons présenté Context Based Graph Anomaly Detector, CoBaGAD, une extension des Graph Attention Networks qui se concentre sur la détection de telles anomalies. Grâce à des expériences transductives intensives, nous démontrons la capacité de notre méthode à identifier de telles anomalies et à surpasser les algorithmes de pointe.
Différentes améliorations peuvent être envisagées dans le cadre de travaux futurs, telles que l'attribution d'un score d'anormalité au lieu d'une classification binaire ou l'étude d'anomalies définies par des contextes de plus grand diamètre ; ce qui impliquera l'utilisation de réseaux avec plus de couches pour augmenter le "champ de vision".

Ce travail est financé en partie par IDEXLYON ACADEMICS projet ANR-16-IDEX-0005 de l'Agence Nationale pour la Recherche.

Références

Akoglu, L., M. McGlohon, et C. Faloutsos (2010). Oddball : Spotting Anomalies in Weighted Graphs. In *PAKDD*, pp. 410–421.

Akoglu, L., H. Tong, et D. Koutra (2015). Graph based anomaly detection and description : a survey. *Data Min. Knowl. Discov. 29*(3), 626–688.

Bojchevski, A. et S. Günnemann (2018). Bayesian robust attributed graph clustering : Joint learning of partial anomalies and group structure. In *AAAI Conference*, pp. 2738–2745.

Breunig, M., H.-P. Kriegel, R. T. Ng, et J. Sander (2000). Lof : Identifying density-based local outliers. In *ICMD*, pp. 93–104.

Grover, A. et J. Leskovec (2016). Node2vec : Scalable feature learning for networks. In *SIGKDD*, pp. 855–864. ACM.

Hamilton, W. L., R. Ying, et J. Leskovec (2017). Inductive Representation Learning on Large Graphs. In *NeurIPS*, Volume 30, pp. 1024–1034.

Hendrycks, D. et K. Gimpel (2020). Gaussian error linear units (gelus).

Kipf, T. et M. Welling (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Lancichinetti, A., S. Fortunato, et F. Radicchi (2008). Benchmark graphs for testing community detection algorithms. *Phys. Rev. E 78*.

Largeron, Mougel, Benyahia, et Zaïane (2017). Dancer : dynamic attributed networks with community structure generation. *Knowl Inf Syst 53*, 109–151.

Mehrotra, K. G., C. Mohan, et H. Huang (2017). *Anomaly Detection Principles and Algorithms*. Springer International Publishing.

Müller, E., P. Sánchez, Y. Mülle, et K. Böhm (2013). Ranking outlier nodes in subspaces of attributed graphs. In *ICDEW*, pp. 216–222.

Perozzi, B. et L. Akoglu (2018). Discovering Communities and Anomalies in Attributed Graphs : Interactive Visual Exploration and Summarization. *ACM TKDD 12*(2), 1–40.

Perozzi, B., L. Akoglu, P. Iglesias Sánchez, et E. Müller (2014). Focused clustering and outlier detection in large attributed graphs. In *SIGKDD*, pp. 1346–1355.

Vaudaine, R., B. Jeudy, et C. Largeron (2021). Detection of contextual anomalies in attributed graphs. In *19th International Symposium on Intelligent Data Analysis, IDA 2021*.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, et Y. Bengio (2018). Graph Attention Networks. *International Conference on Learning Representations*.

Wang, X. et I. Davidson (2009). Discovering contexts and contextual outliers using random walks in graphs. In *ICDM*, pp. 1034–1039.

Xu, X., N. Yuruk, Z. Feng, et T. A. J. Schweiger (2007). Scan : A structural clustering algorithm for networks. In *SIGKDD*, pp. 824–833.

Summary

Graph anomaly detection have proved very useful in a wide range of domains. For instance, for detecting anomalous accounts on online platforms, intrusions and failures on communication networks or suspicious and fraudulent behaviors on social networks. However, most existing methods often rely on pre-selected features built from the graph, do not necessarily use local information. To overcome these limits, we present CoBaGAD, a Context-Based Graph Anomaly Detector which exploits local information to detect anomalous nodes of a graph in a semi-supervised way. We use Graph Attention Networks (GAT) with our custom attention mechanism to build local features, aggregate them and classify unlabeled nodes into normal or anomaly. Finally, we show that our algorithm is able to detect anomalies with high precision and recall and, outperforms state-of-the-art baselines.