Détection d’anomalies grâce à VGAE et GAN

Rapport de Bio-Inspired Machine Learning

Angéline Marc, Nathan Corroller, Thomas Huguenel
20/10/2024

Sujet

Le but de ce projet était de mettre en place l'un des trois sujets suivants : prédiction de lien, détection d'anomalie ou classification. De notre côté, nous avons décidé de travailler sur la détection d'anomalie. Pour cela, il nous était demandé de travailler sur un graphe représentant des aéroports et les liens entre ces aéroports.

La détection d'anomalies sur un jeu de données consiste à prédire si certaines valeurs dans les colonnes ou les liens du graphe sont erronés. Pour notre part, nous avons décidé de nous concentrer sur la colonne « Population » des villes. Cependant, nous aurions aussi pu détecter des erreurs sur le nom et l'emplacement des villes mais également vérifier les liens, en déterminant si deux villes étaient reliées de manière anormale.

Caractérisation d'une anomalie

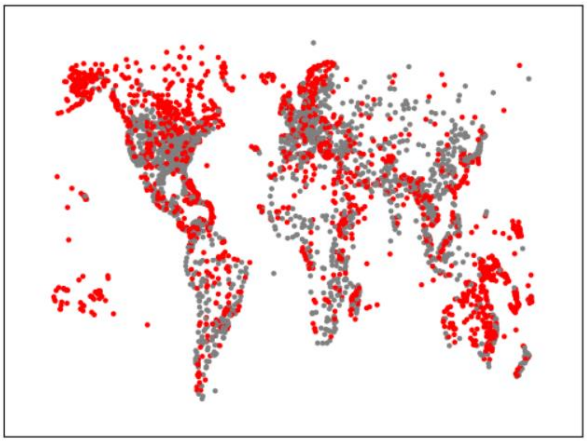
Nous avions dans un premier temps besoin de nous donner une idée du nombre et du type d'erreurs que nous pouvions détecter. Pour cela, nous avons observé manuellement le jeu de données et nous avons réalisé qu'un nombre important de villes avaient une population définie à 10 000 habitants. Nous avons alors acté qu'il pouvait s'agir d'une anomalie. En effet, on peut imaginer que certains chiffres n'aient pas été renseignées dans le dataset car la ville concernée a une population faible ou qui n'est pas connue. La valeur 10 000 aurait alors été entrée par défaut pour ces villes. De plus, on remarque que certaines villes ont une population incorrecte comme Lille en France ou Portland aux Etats-Unis qui se retrouvent avec un nombre d'habitants anormal par rapport aux données réelles. Il est aussi possible de voir que certaines villes différentes mais avec un nom identique possède la même population.

Nous avons déterminé manuellement certaines anomalies mais par soucis de complexité et pour ne pas vérifier la démographie de chacune des 3363 villes présentes dans le jeu de données, nous avons décidé de nous concentrer principalement sur les villes ayant une population à 10 000 habitants.

Après un calcul rapide, nous avons obtenu les valeurs suivantes nous donnant une estimation du nombre d'anomalies présentes.

```
Pourcentage de villes avec 10000 habitants: 49.75%, soit 1673 sur 3363 villes
```

Avec ces informations, nous pouvons tracer une carte représentant les nœuds du graphe où les villes marquées avec 10 000 habitants sont dessinées en rouge tandis que les autres sont de couleur grise.



Modèles utilisés

Pour permettre la réalisation de la détection d'anomalies, nous avons mis en place et comparé deux modèles, un VGAE et un GAN.

Dans les deux cas, nous commençons par récupérer toutes les données du graphe et de les normaliser pour ne pas avoir de trop grands écarts de valeur sur les valeurs numériques et nous encodons les autres avec un encodage *One-hot* pour pouvoir les concaténer ensuite dans des tenseurs. Nous séparons également les données en trois pour obtenir des jeux d'entrainement, de validation et de test grâce à la fonction *train_test_split_edges* de torch_geometric.

VGAE

Définition

Le premier modèle que nous avons implémenté est un Auto-Encodeur Variatif de Graphes (VGAE). Il s'agit d'une variante de VAE conçue pour travailler avec des données sous forme de graphe qui combine les concepts des auto-encodeurs et des réseaux de neurones graphiques (GNN). Un auto-encodeur est un réseau de neurones composé d'un encodeur et un décodeur. L'encodeur prend en entrée la structure du graphe et compresse les informations pour obtenir une représentation latente. Le décodeur quant à lui a pour but de reconstruire le graphe d'origine à partir de la représentation.

Implémentation

```
Entraînement validation :
Epoch: 000, AUC: 0.8757, AP: 0.8866
Epoch: 010, AUC: 0.8848, AP: 0.8942
Epoch: 020, AUC: 0.9007, AP: 0.9098
Epoch: 030, AUC: 0.8878, AP: 0.9047
Epoch: 040, AUC: 0.8995, AP: 0.9102
Epoch: 050, AUC: 0.8840, AP: 0.9015
Epoch: 060, AUC: 0.8947, AP: 0.9120
Epoch: 070, AUC: 0.9223, AP: 0.9352
Epoch: 080, AUC: 0.9323, AP: 0.9438
Epoch: 090, AUC: 0.9339, AP: 0.9463
Epoch: 100, AUC: 0.9405, AP: 0.9513
```

```
Epoch: 110, AUC: 0.9489, AP: 0.9577
Epoch: 120, AUC: 0.9545, AP: 0.9619
Epoch: 130, AUC: 0.9544, AP: 0.9621
Epoch: 140, AUC: 0.9547, AP: 0.9631
Epoch: 150, AUC: 0.9551, AP: 0.9644
Epoch: 160, AUC: 0.9559, AP: 0.9655
Epoch: 170, AUC: 0.9584, AP: 0.9671
Epoch: 180, AUC: 0.9587, AP: 0.9676
Epoch: 190, AUC: 0.9595, AP: 0.9684
Test final :
Epoch: 200, AUC: 0.9643, AP: 0.9721
```

Il était dans un premier temps nécessaire de réaliser un mappage des nœuds du graphe ainsi que d'extraire les arêtes sous forme de paires d'indices pour pouvoir rentrer les informations un objet *Data*. Nous avons ensuite défini notre encodeur avec trois couches de convolution avant

de l'entrainer en donnant le tout à un VGAE. En utilisant l'optimiseur Adam à un taux d'apprentissage à 0.005, nous obtenons une AUC autour des 96% et une précision moyenne proche des 97% sur la phrase de test. Ces valeurs satisfaisantes nous permettent d'évaluer la qualité des prédictions dans notre graphe.

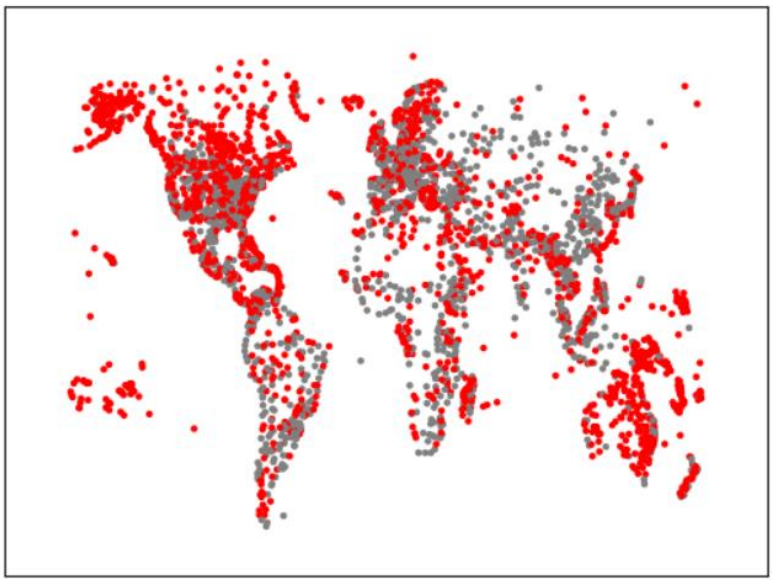
Il a fallu dans un second temps, décoder ces informations pour obtenir le nom des villes avec la prédiction de leur population. Pour cela, nous avons défini un modèle avec une couche linéaire et une de ReLu que nous entrainons avec l'optimiseur SGD et MSELoss pour la fonction de perte avec un taux d'apprentissage de 0.1.

Pour terminer, on dénormalise la population et compare les valeurs prédites avec les valeurs originales présentes dans le dataset.

Résultats

Finalement, notre modèle parvient à identifier que 56% des villes de notre graphe ont une population anormale, dont 47,43% des villes qui possèdent une population de 10 000 habitants. Ces résultats peuvent varier de plus ou moins 3% selon les exécutions.

```
Pourcentage de villes avec des anomalies détectées: 56.05%
Pourcentage de villes avec anomalies ayant 10000 de population: 47.43%
Pourcentage de villes avec 10000 habitants: 49.75%
```



GAN

Définition

Nous avons également décidé d'implémenter un modèle de Réseaux Antagonistes Génératifs (GAN) pour prédire les anomalies. Un GAN est composé de deux réseaux de neurones principaux : un générateur et un discriminateur.

- Le générateur prend en entrée un vecteur de bruit aléatoire et génère un vecteur qui imite les caractéristiques d'une ville en respectant les distributions apprises à partir des données d'entraînement. L'objectif du générateur est de créer des villes "fausses" qui

ressemblent aux villes réelles du dataset dans le but de tromper l'autre réseau, le discriminateur.

- Le discriminateur, quant à lui, prend en entrée un vecteur représentant une ville réelle, provenant du jeu de données ou un vecteur généré par le générateur. Son rôle est de distinguer les villes réelles des villes générées et son objectif est de maximiser la précision avec laquelle il parvient à différencier les données réelles des données imitées.

Le GAN est entraîné de manière que le générateur devienne de plus en plus performant pour tromper le discriminateur, tandis que ce dernier devient de plus en plus efficace pour distinguer les faux exemples.

Une fois le GAN entraîné, il peut être utilisé pour détecter des anomalies dans le dataset des villes. L'idée est que le générateur ait appris la distribution des villes "normales" à partir des données d'entraînement. Si une ville présente des caractéristiques très différentes de celles apprises par le générateur, elle pourra être considérée comme étant une anomalie.

Implémentation

Nous commençons par définir les hyperparamètres utilisées pour entrainer les deux réseaux en définissant le taux d'apprentissage à *0.0001*, la dimension du vecteur aléatoire pris par le générateur en entrée à 150, ainsi que le nombre d'itérations à 50. Le GAN que nous avons réalisé est inspiré du DCGAN présenté sur le site Pytorch et plus précisément sur la page de présentation du DCGAN (Deep Convolutional Generative Adversarial Networks). La structure des différentes couches du générateur et du discriminateur provient de ce modèle.

En ce qui concerne la phase d'entrainement, nous avons décidé d'utiliser l'optimiseur Adam et la fonction de perte « BCE with logits Loss » pour les deux modèles. Cette fonction est utilisée lorsque que l'on a une sortie binaire, ce qui est le cas avec le discriminateur qui retourne si une donnée est générée ou réelle. Nous obtenons après entrainement une précision de 87.26%.

Pour terminer, on dénormalise la population et compare les valeurs prédites avec les valeurs originales présentes dans le dataset.

Résultats

Nous obtenons pour le GAN des résultats légèrement différents du VGAE avec 58,55% de villes identifiées comme anomalies donc 49,66% d'entre-elles qui possèdent 10 000 habitants. Pour une raison qui nous est inconnue, l'entrainement du modèle nous donne toujours la même précision malgré les différentes exécutions. Nous pensons que nous avons commis une erreur lors du calcul de la précision ou lors du formatage des données pour l'utilisation du modèle car après plusieurs recherches, la partie de notre code qui correspond à l'entrainement des deux modèles nous semble juste. De plus, étant donné que l'accuracy du GAN est de 87.26%, il n'est pas normal d'avoir presque toutes les villes à 10 000 habitants qui sont correctement prédites. Nous ne pouvons donc pas conclure sur la réelle efficacité de ce modèle pour la détection d'anomalie.

```
Pourcentage de villes avec des anomalies détectées: 58.55%
Pourcentage de villes avec anomalies ayant 10000 de population: 49.66%
Pourcentage de villes avec 10000 habitants: 49.75%
```

Comparaison des résultats du VGAE avec plusieurs paramètres

Les meilleurs résultats obtenus en termes de précision et de fiabilité étant ceux du modèle VGAE, nous avons décidé de nous baser dessus pour faire des comparatifs qui sont présentés dans le tableau ci-dessous.

Fonction de perte	Optimiseur	ETA	Pourcentage de villes anormales	Pourcentage de villes à 10000 habitants anormales
MSELoss	SGD	0.1	57.95	49.12
	Adam	0.0001	40.29	33.33
	RProp	**0.0001**	**58.64**	**49.75**
L1Loss	SGD	0.1	58.55	49.66
	Adam	0.0001	53.88	45.38
		0.001	38.72	32.02
	Rprop	**0.0001**	**58.64**	**49.75**

Conclusion

Nous avons mis en place un VGAE et un GAN pour faire de la prédiction d'anomalie sur la population de notre jeu de données. Cependant, comme nous l'avons évoqué précédemment, nous ne pouvons pas vraiment conclure sur l'utilisation du GAN car nous n'avons pas réussi à trouver l'erreur qui bloquait notre accuracy. De plus, nous trouvons notre VGAE surperformant dans certaines configurations. En effet, selon les hyperparamètres, fonction de perte et d'optimisation, notre décodeur nous permet de trouver 100% des villes ayant 10 000 habitants. Ce qui, selon nous, n'est pas normal car notre VGAE n'a pas une précision de 100%.