Clasificaction tipo de relaciones entre Sistemas Autónomos

modelos

Para bordar este problema se utiulizaron .....

Experimentos

Para abordar este problema se tomarón dos enfoque diferentes.

- Clasificación Binaria: Se tomaron p2c y c2p como una misma clase y las otras siendo p2p.
- Clasificación Multiclase: Las relaciones p2c y c2p se tomaron como clases diferentes.

Experimento 1:

- GNN -> GCN
- Predictor -> DOtProduct y MLP
- Optimizador -> Adam
- Función de pérdida -> CrossEntropyLoss
- Split de edges para entrenamiento y validacion
- Stochastic Gradient Descent (SGD) con un learning rate de XXX.

Figure 1: Resultados.

- Problemas con el modelo:
  - Posible overfitting.

Experimento 2:

- GNN -> GraphSAGE
- Predictor -> MLP
- Optimizador -> Adam
- Función de pérdida -> CrossEntropyLoss
- Split de edges para entrenamiento y validacion
- Neighbou sampling.

Figure 2: Resultados.

- ¿Por que ocupamos GraphSAGE y no GCN?

Como estamos entreando de forma transductive, es decir ocupando un mismo grafo para entrenar y validar, puede ocurrir que se este overfiteando el grafo y por eso obteniendo buenos resultados, estod ebdo a que al probar con epoch muuuy grandes los valores de loss todo el rato eran muy

similares, y lo que se espera obtener en este caso es q llegase a un punto dond ela loss era similar , pero luego empezaran adiverger, que seria el punto en donde el modelo se esta empezando a aprender los datos de memoria, pero esto no paso???. Para evitar esto se decidio ocupar otras forma sd eentrenamiento para majear que el modelo pudeese generalizar y no se estuviese validando mal (porq al final mas q na es problema de q noe stamos validando con datos diferentes). Entrena runasdo ampling.

Experimento 3:

- GNN -> GraphSAGE
- Predictor -> MLP
- Optimizador -> Adam
- Función de pérdida -> CrossEntropyLoss
- Split de edges para entrenamiento y validacion
- Otro sampling

Figure 3: Resultados.

Experimento 4:

Agregar paquetes de flujo

Figure 4: Resultados.

Destacar que en nuestro enfoqeu fue de Regresion, es decir la salida final de los modelos corresponde a un valor uncaente al cual luego se calcula un umbral para clasificar en las clases correspondientes. Esto porque l tener clases desbalanceadas, el modelo desidiria en solo clasificar un solo tipo (la q es mayor) y aun asi no se tienen resultados tan malos. Por esto no se tomo ese enfoque ya que es una de als fallas a la que queda propenso.

GCN VS GraphSAGE

- Comparamos el caso de GCN con GraphSAGE usando como Predictor el DotProduct y ocupando todas las features:

GCN:

- Con 100 epoch, un % de train de 0.6 se obtiene una accuracy de 0.81, sin embargo se puede observar overfitting mas o menos desde el epoch 70.
- Ocupando un modelo que agrega entre cada capa GNN dropout se octiene una accuracy 0.8. No es muy diferentes.
- Drop out ayuda harto, sin embargo hace q no sea tan smoth el converger.

GraphSAGE:

- Con epoch 100, un % de train de 0.6 se obtiene una accuracy de 0.9 mas o menos.
- Para que haya overfitting (y sea visible en los graficos) se encesita que ocupemos un bajo porcenaje de train ej:0.1 y alto numero de entrenamiento de epoch ej :300. ahi claramnete se ve el verffitiing.

Podemos ver que GraphSAGE super en performance a GCN. Tambien podemos ver que para ambos modelos el overfitting se presenta en diferentes casos, esto por la naturaleza de los calculos de cada uno en la aggregación. Pues en GCN para obtener un mensaje al nodo , se necesitan todos sus vecinos haciendo que se afacil reconocer un nodo de otro ya si mas fcil de que se reconosca overfitting, en vambio en graphSAGE el mensaje al nodo en como un promedio de los vecinos, por lo que es mas dificil que se reconosca overfitting.

Se realizo la mism acomparación pero en vez de DotProduct se ocupo MLPPredictor. En el caso de GCN La accuracy subió a un 0.8222(0.9221 cmabiando porte capas) y GraphSAGE a un 0.92 (con el mismo numero entre capas) 0.9546 (cambiando nuemero de capas). Sin embargo con 100 epoch no cambiaba mucho al final. Quise ver que ocurria si ocupaba un porcentaje muy bajo de train, para evr si habia overfitting, pero lo que psao fue q igual habia buenos resultados...??? ocupe el modelo sin droppout y ahi se podia ver un poco el overfitting. con una cantidad de X ejmplos de edge para train.

[Cachar porque no me esta encajando el numero de true y falses en el train msakk al ahcer split del datset]

como sea Es mejro con MLP que con DotProduct, por lo que se ocupara MLP en los siguientes experimentos.

OCUPAR TODAS LAS FEATURES ES NECESARIO?

Luego para ver que tan esenciales en la tarea era los features recolectados se porbo con graphSAGE y MLP como predictor (con ahsta el momento los mejores resultados que s e ahan obtenido), con 100 epoch y un % de train de 0,6 .

Se partio ocupando unicamente como atributos de los ndoos su grado in y grado out por nodo. Obteniendo una Accuracy de 0.8724 con DotProduct y 0.9290 con MLPpredcitor. Con esto vemos que si bien se dan buenos rsutltodos agregar las feeatures Mejora la accuracy. ¿Pero son de ayuda las que le estamos pasando? Prque si bien ya tenemos que son una ayuda luego de hacer una exploracion de esta (En anexo mas info) podemos notar que para muchos features par ala mayria de los nodos no se tiene información y por ende no serian muy relevantes?.

PAra esto se decidio incluir unicamete aquellas features cuya información para todos los nodos estuviera sobre 80%. Es decir existe info de la feature para el 80 % de los nodos. Estoso consistieron en :

- AS_rank_numberAsns
- AS_rank_customer
- AS_rank_peer
- peeringDB_ix_count
- peeringDB_fac_count
- cti_top

Con estos usando GraphSAGE y MLP como predictor se obtuvo una accuracy de 0.9452 lo que s eve que tener todas las otras valores lo mejora pero no tanto, es decir no son tan relevantes para dicha tarea.

otros caso fue elegir unicamente .....[COMPLETAR]...

————————————————————

IDEAS FALLIDAS

- Dentrro de otras ideas que se intentaron pero no funcionaron fue crear grafos a partir de los recolectores RRC, de sta forma tener diferentes grafos a partri d elos cuales algunos se elijan para training y otro diferente para testeo. Sin embargo falle, no se si retomar o no.
- otra idea fue que RIPEstat tineen una API de la cual se puede obtener información, sin embrago se demora demasiado para la cantidad de nodos ques e tienen por grafos.

————————————————————

Continuando con los experimentos una de los problemas que mas miedo tenia era que se entrenara y se esttuviera overfitteoando el grafo, porque no tenemso mas grafo que el de esa fechaa, pues esos datos (los atributos corresponden a sacados por otro paper) . Es decir estabamos tomando un enfoque inductivo. Es por esto que una podible solución que se nos ocurrio fue proabr diferentes tecnicas de samplingm¿, area que aun sigue en investigación e inovacion en el area de GNN.

Para esto se partio con random neighbour sampling[explicado en XXX marco teorico] y se obtuvo una accuracy de 0.9414 , lo que no mejoro mucho los resultados obtenidos anteriormente. ya desde el segundo epoch hay overfitiing.

se decidió probar con otro tipo dxxxe meodo el cual corrspondiente a ClusterGCN [explciado marco teorico seccion XXX] obteniendo una accuracy 0.9696 !!!!!!!

Luego para comparar GNN con otros metods para resolver dicha tarea, se probo resolver la atrae de clasificación con PAgeRank, DeepWalk y BGP2Vec. Obteniendo Resultados XX, 0.9270 (entrenando) y XX respectivamente.

La idea es ver que tan bien se comporta el modelo en comparación con otros metodos que se han ocupado para resolver la misma tarea.

Caso	Accuracy
GCN + DotProductPredictor	0.81
GraphSAGE + DotProductPredictor	0.9
GCN + MLPPredictor	0.9221
GraphSAGE + MLPPredictor	0.9546
GraphSAGE + MLPPredictor + in_degree y out_degree	0.9290
GraphSAGE + MLPPredictor + features sobre 80%	0.9452
GraphSAGE + MLPPredictor + Random neighbour sampling	0.9414
GraphSAGE + MLPPredictor + ClusterGCN	0.9696
PageRank	0.8746
DeepWalk	0.9270

Caso	Accuracy
BGP2Vec	X
Crear de 0	0.9068