Who-To-Follow Link Prediction

數據所碩一 楊子萱

April 22, 2023

Abstract

本次的作業為follower-followee network datasets的link prediction任務，而在於模型的架構上使用GAE的方式，針對edge index透過Encoder的方式取得Node information，再透過Decoder的方式將information取出，得到最終Edge的機率。在於三個dataset中最高的AUC為0.9008，最高的Average Precision為0.9140，而在三個dataset中AUC平均分數為0.8714，Average Precision平均分數為0.9009，也都是近乎9成的準確性。

1 Introduction

Link prediction是指在一個Graph Network中，根據網絡中已有的節點和邊，預測兩個節點之間存在邊的機率。通常該問題被用於社交網絡、生物網絡、電子商務等領域，舉例來說，今天若是有兩個人擁有共同的朋友，則會有較高的機率被預測出來未來可能會成為朋友，若是有一個族群的人常購買某種商品，若是今天有一位新加入的樣本，具有相同的興趣或愛好，則也有很高的機率會購買此種商品，因此也常用於推薦朋友、預測用戶購買商品等任務。

在圖深度學習領域，有很多方法可以用於link prediction問題。最常見的方式為將原始的node feture透過embedding的方式將information轉化成為較低的dimension，然後透過每一個人的feature vector去計算node間的simialrity。而上述的方式也可透過多種的方法實踐，例如普通的Neuron Network、GCNconv，或是帶有時間性的LSTM、GRU等。除了embedding的方式外，也有許多例如Random walk、Random walk with restart(RWR)等方法。

本次的實作中，使用了在link prediction常見的GAE方法，在Encoder中使用GCN針對edge index以及node feature去取得Node information，在Decoder中針對用於判斷loss的edge去將information取出並加總，得到最終是否存在該Edge的機率，而其中也嘗試了多種的變化，但可能因為對於Graph本身不完全熟悉，因此嘗試過後的結果並沒有達到較高的準確性，但在於報告中也會逐一解釋做了哪些的嘗試。

2 Proposed method: GAE

GAE是Graph Autoencoder，是一種透過Encoder & Decoder，將Graph node feature embedding和Reconstruct的Unsupervised learning。Encoder會將原始Graph data嵌入到一個低維空間中，而Decoder則是將feature還原回原始Graph data。目標是最小

化原始與重構Grpah間的差異。

在GAE模型中，Encoder & Decoder通常都是基於圖卷積神經網絡（GCN）構建的。GCN是一種專門用於處理Grpah的捲積神經網路，能夠透過捲積的方式去聚合鄰居節點的資訊，因此能夠捕捉Graph中的局部結構和全局結構，從而取得豐富的Graph information。

GAE除了link prediction外還可以應用於多種Grpah task，例如node classfiction、node clustering。該模型已經被證明在社交網絡、生物網絡和化學分子網絡等領域中表現良好。同時，GAE也為其他Graph技術提供了一個基礎框架，例如基於變分自編碼器（VAE）和基於生成對抗網絡（GAN）的圖形嵌入方法，以及GraphSAGE。

在這個章節中，會詳細的解說本次任務中效果最好的GAE model方法，包含模型的架構以及激勵函數的套用，以及其餘做過的嘗試，與嘗試後卻沒有較好成效的推論原因。

2.1 Encoder

$$z_v = ENC(A, X, \theta_{ENC}) \tag{1}$$

A為相鄰矩陣，X為node feature，θ_{ENC}為Encoder中會使用到的權重矩陣 {W0、W1}，在本次是透過兩層的GCNconv做information embedding，其中是傳入node feature以及edge index，模型會根據edge index去組成相對應的相鄰矩陣，而在於激勵函數的部分使用LeakyReLU。

2.1.1 Discussion

在於激勵函數的套用上，測試過多種的激勵函數對於模型訓練的影響，其中包含有ReLU、LeakyReLU、Sigmoid、Tanh，發現到在於正確性來講LeakyReLU的效果最好，認為因為LeakyReLU相比於ReLU上，在於負值是給予一定程度的懲罰像，並非直接收斂到0，因此能夠降低負值對整體的影響，卻也能夠保留負值的資訊，而Sigmoid雖然也能夠保留負值的資訊並轉為正的，但對於正值卻也大大的降低使得整體預測力下降，而Tanh雖然相對於Sigmoid有zero-counter的優勢，但卻比Sigmoid收斂得更快導致預測力下降。

在於Edge index到底是要輸入含有positive edge index以及negative edge index，還是僅輸入positive edge index，我認為也是蠻值得討論的一個項目，因為在於測試的結果上將全數的edge index輸入的效果較好，但在於直覺的理解上，應該是要在真實的edge上學習Graph的information才會具有意義。 但是當僅丟入positive edge index時，容易造成training AUC非常高，而validation的AUC卻相當的差，我認為是因為若是將僅丟入positive edge index，容易學習到正樣本，而對於其餘的樣本則會效果極差，因此需要在其中混入負樣本，去降低positive edge的information過高而導致其他值過低，被近乎收斂至0的問題。

在於模型的堆疊上使用兩層的GCNconv做information embedding，測試做一層以及三層，發現到一層的GCN也能夠收斂到與兩層GCN相同的效果，差別在於說一層的收斂速度會比較慢，因此需要更多的epoch與運算時間去驗證最終的成果，

而在於三層的部分，發現到整體的準確性會下降，推測是在於information的值在於normalized後皆小於1，而在於多層的聚合後值變得越來越小，且本身的feature matrix就非常的sparse，因此導致forward後多數的feature vector近乎收斂到0，而使得準確性下降。

2.2 Decoder

$$y_v = z_v * z_v.T \tag{2}$$

Decoder則是將Embedding後的information reconstruct成相鄰矩陣，並取用該Edge index的兩個node即可獲得預測是否會存在該條邊的機率值，而在於程式撰寫上由於減少計算量，因此只取用對應的Edge index去做聚合得到最終結果。

2.2.1 Discussion

在本次的模型建構上僅聚焦在於Encoder時的node embedding，並未對Decoder多做更多的嘗試稍嫌可惜，僅簡單的聚合node representation作為edge出現的機率，也許可以嘗試將node representation去計算node間的similarity，或是透過MLP的轉化得出edge出現的機率也是之後可以在嘗試的目標。

2.3 Train & Predict

上面的小節講述了Encoder & Decoder的詳細實作方法，而在本小節將講述訓練是如何進行，以及在於其中發現不同訓練方式會導致的差異。

Train.

1. 將model設置為訓練模式，並清空gradient
2. 將training data的全部的edge index以及node feature傳入Encoder中得到z
3. negative sampling與positive edge數量相同的negative edge
4. 並將positive edge與negative edge合併，以及兩者的label也合併
5. 將合併後的egde index以及node feature傳入Decoder中得到output
6. output與label透過Binary Cross Entropy計算loss
7. 根據loss值更新參數

Predict.

1. 將model設置為評估模式
2. 將測試集的edge index以及node feature傳入Encoder中得到z
3. 最終再將z與edge期望測試的邊做decoder得到是否存在該條邊的機率

2.3.1 Discussion

在於一開始的訓練時，僅使用老師提供的全部edge index作為訓練的positive & negative edge，因此在於training set AUC皆達到將近95$\tilde{9}$8的準確性，卻在validation set效果非常的差，而在後續的training時，加入negative sampling，發現大大的提升預測的準確率，因此發現說若是只讓模型觀察單一一種的negative edge，則會使得模型只學習到一種的negative pattern，因此negative sampling會在每個epoch加入不同的negative edge，去增加模型觀察到的負樣本，使得模型學習到的information更加完整，再加上因為要去平衡正負樣本的數量，因此每次都是放入數量相同的正負樣本。

在於loss的計算上則是採用Binary Cross Entropy with logit這個function做計算，因為在於計算時若是先透過一層的sigmoid，再去計算Binary Cross Entropy，則預測效果會較差，因為當output經過一層的sigmoid則容易收斂成0，而在於Binary Cross Entropy with logit則會將sigmoid與exponential一併計算，能夠使得模型更加的穩定，預測效果也會更加的好。

3 Try and Error

在於這章節將會講述到在本次預測任務中，嘗試希望套用的模型與其他資料前處理的方式卻未得到更高準確性的方法，與考量後可能的原因。

3.1 Data Prepossessing

在於訓練資料的前處理上主要是針對Graph node feature做前處理，而簡單的透過PCA、t-SNE針對特徵做降維的動作，但最終的成效都比起未做任何處理效果來的較差，因此根據降維做了相關的文獻回顧，發現到PCA與LDA類同都是針對特徵做線性轉換，因此當data struture是處於nonlinear manifold則會容易喪失掉Grpah strucutre information[4]，而在於Manifold learning的t-SNE雖然解決了curse of dimensionality高維資料中的距離關係不能完整在低維空間中保留以及各個分群會有聚集在一起無法區分的擁擠問題（crowding problem），但貌似較少使用t-SNE in graph可能是在於降維應用上不同領域還是有不同常用的方法，因此較少查到相關資料。

在於前處理上也測試了不同的normalized方法，最終成效最佳的為Row-normalizes，即使得一個node的所有feature總和為1，也測試過作Column-normalizes即將所有具有這個feature的總和變成1，但是由於有許多的feature同時具有的人數很多，因此若是做normalize則每一格的值則變為非常小，因此在通過GCN時變得容易收斂為0，而無法成功的訓練下去，也嘗試過同時具有多數node的feature刪除也依舊無法改善，因此最終使用Row-normalizes。

3.2 Different Model

在於模型的建立上，也嘗試使用GrapghSAGE[1]或者是NESS[3]去測試，但可能在於實作上沒有完全正確的還原論文中的input方式，或者是使用尚未正確理解清楚，因此正確性比起簡單的GAE來講較差，未來可以根據已嘗試的部分再多加嘗試。

NESS. Node Embeddings from Static Subgraph是透過Contrastive learning的方式去比較多個子圖的相似性已去達成最終的預測，實踐方式是將一個Grpah拆分成多張subgraph，然後再透過GAE使得subgraph去獲得graph representations再重新重構出整張graph，並去計算subgraph與重構後的subgraph間的差異，將多張subgraph的loss加總，並且儲存subgraph representations兩兩計算Paired Contrastive loss，再透過loss更新參數，而在Contrastive loss計算上採用normalized temperaturescaled cross entropy loss (NT-Xent)，最終再根據訓練完的subgraph representations做聚合取得graph representations。

但在於測試時，可能是對於Contrastive learning在於計算loss上某個環節中出錯，或者是在於subgraph的切分上數量太少，導致近乎所有的點預測出來的值相近，無法看出不同點間是否存在的機率，認為在於subgraph的切分方式能夠再多加研究，因為測試發現不能完全根據平時的拆分方式將圖直接的拆分，可能可以將較相近的node切再一起，除了能夠增加node間的關聯性，也可以降低明明Between centrality值高或Closeness centralit高的node卻被拆分成Outlier的情形，使得整體的information無法透過這些點聚合。

Algorithm 1 NESS

Input: Graph $G = (V, E)$, adj. $\boldsymbol{A}$, node features $\boldsymbol{X}$
Output: Node representation $\boldsymbol{Z}$
Initialize: Encoder Enc
Partition G into k static $subgraphs = [\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_k]$ by using random edge split.
for $epoch = 1$ **to** max_epocs **do**
  $loss = 0, z\_list = []$
  for a_k **in** subgraphs **do**
    $\boldsymbol{z}_k = Enc(\boldsymbol{X}, \boldsymbol{a}_k)$
    $\hat{\boldsymbol{a}}_k = \sigma(\boldsymbol{z}_k \boldsymbol{z}_k^T)$
    $loss = loss + recon\_loss(\boldsymbol{a}_k, \hat{\boldsymbol{a}}_k)$
    $z\_list.append(\boldsymbol{z}_k)$
  end for
  if contrastive loss == True **then**
    Get combinations of $\boldsymbol{z}_k$'s: $[(\boldsymbol{z}_1, \boldsymbol{z}_2), ...]$
    Compute contrastive loss $\mathcal{L}_c$, based on Eq. 3
    $loss = loss + \mathcal{L}_c$
  end if
  Update encoder Enc
end for
Return $\boldsymbol{Z} = aggregate(\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_k)$

Figure 1: Algorithm1 NESS.

GraphSAGE. 由於大部分的link prediction model都是Transductive learning，意思是根據目前已經具有知道的node資訊去作出推論，並且預測是否存在邊的機率，但是在於現實社會中大多數的node皆是新的僅能夠拿到其相對應的attribute information [2]，因此發展出一種Inductive learning，從已有數據中歸納出pattern來，應用於新的Graph和index。SAGE為SAmple and aggreGatE，而其與平時的GCN不同點在於，GCN是根據整張圖去聚合Neighborhood information，而在於SAGE則會去sampling Neighborhood再透過不同方式做聚合(ex：Sum, Mean, LSTM)。此方法透過Minibatch的方式去訓練整張圖，因此能夠擴展到訓練Large Graph。

因為在於觀察到的GrapgSAGE多應用於link prediction任務，因此使用方式也與上面章節的Train方式相同，但在於預測準確性上卻沒有原始來得好，因此可能在於應用的方法是否有需要再轉化，需要再多閱讀相關的文章，並且在針對詳細的link prediction task做細緻的了解，才能夠在於不同model的應用時，能夠瞭解其中細部的差異。

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

Figure 2: Algorithm2 GraphSAGE.

4 Experiment

4.1 Experiment setup

在本次的測試中，將dataset透過RandomLinkSplit的方式拆分成Train、Val、Test，使用上述的GAE模型透過Binary Cross Entropy with logit計算loss，並計算Training set、Validation set以及Testing set的AUC，本次測試用於20-core server with 512GB memory，再加上NVIDIA Ampere A100的GPU。

4.2 Results on datasets

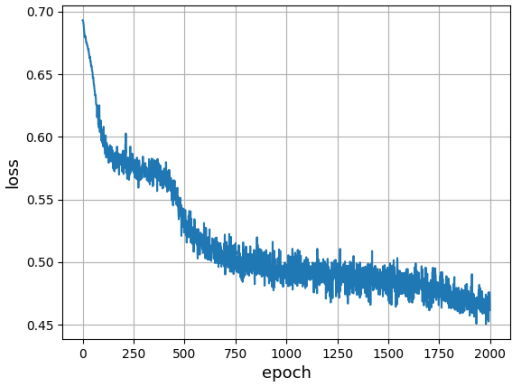

Figure 3: loss on training

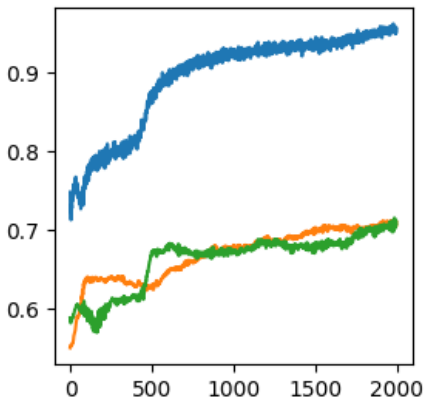

Figure 4: AUC on train&val&test

可以透過上圖看到，儘管Traing的loss不斷在降低，且Traing set的AUC近乎95%以上，而在於validation set以及testing set上卻很差，可以看出模型已經學習到大部分的Traing set資訊卻在於遇到新的Edge時，沒有辦法正確地去預測到edge是否存在，因此可能會需要改善負樣本或是未出現的樣本在於training時的比例去使得模型在於學習時能夠不要overfitting。

	dataset1	dataset2	dataset3
AUC	0.8685	0.9008	0.8592
Average Precision	0.8926	0.9140	0.8962

Table 1: Total Result of link prediction.

本次實驗的架構與訓練方式在於dataset2中表現最好，而在dataset3中表現最差，但在於Average Precision中皆能夠達到近乎0.9的準確性，顯示雖然模型的架構簡單，卻也能夠很大程度的去收集Graph的資訊，因此未來努力的目標是更加地去瞭解Graph到底該如何地去被訓練才能夠使得Graph在於正負樣本中能夠正確的學到資訊。

5 Conclusion

在本次的link prediction作業中嘗試了多種不同的前處理方式以及模型期望去正確預測是否存在邊的機率，測試的結果發現GAE對於Graph information的收集與Reconstructured成效最好，最能夠去獲取最多圖的資訊，且在於node embedding上各種不同的GCN最主要目的都是在於去將不同的node間資訊聚合，顯示說neighborhood information對於node來說非常的重要，而到底是全部直接性的聚合，還是再透過不同方式訓練也值得在多加探討。

最終，我認為針對Graph如何地去拆分，如何的去建構negative sample，雖然不是在於model本身做改變，但是也在於訓練時是與model同等重要的一環，畢竟放入的資料要能夠代表資料本身，也才能夠使得學習會有成果，以去推展至大型的真實世界網路，或者是動態的社群網路，因為社群的建立是日新月異，而新的節點會不斷的產生。

References

[1] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[2] Yu Hao, Xin Cao, Yixiang Fang, Xike Xie, and Sibo Wang. Inductive link prediction for nodes having only attribute information. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, jul 2020.

[3] Talip Ucar. Ness: Learning node embeddings from static subgraphs, 2023.

[4] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):40–51, 2007.