Meeting

0609

- 週五交雅淳學姊論文的修改完的版本
- Survey paper
- Attack paper
  - Nettack實驗文字/表格/圖
  -
  - 在跑amount=3,7,等數據 並 加入碩論
  - Metattack改code，正在跑，不確定效果
  - BUT.....jupyter有點問題正在修QQ

0526

- 實驗
  - Nettack大致跑完，部分還在tune
  - 其他攻擊方法試試
- 雅淳學姊論文&REVIEW
  - 已和apple & 巧莛學姊討論過
  - 有一個不確定

0512

- 論文
  - Apple建議修改
  - 文字細修&更新版本
- 實驗
  - 嘗試跑其他robustness的實驗
  - Nettack
    - 看Nettack paper&code
    - 參考其他paper實作Nettack的方法

        - 要完成的實驗：

Nettack	Cora	1	5	10	20
s&f perturbations	RLGAT-ASR				
	GCN-ASR				
Nettack	Citeseer	1	5	10	20
s&f perturbations	RLGAT-ASR				
	GCN-ASR				
Nettack	PubMed	1	5	10	20
s&f perturbations	RLGAT-ASR				
	GCN-ASR				

        - 跑的時間要等一陣子且不確定效果如何
    - Metattack
        - 要看paper&code
        - 把自己的model兜上去

old

- **原本初版的idea&novelty**
    - idea
        - adversarial training
        - GCN
        - 網路學習擾動量
        - iterative產擾動
        - 更多鄰居資訊考量
- **建議可以修正的方向**
    - 手法比較偏attack
    - 缺乏structure上的考量
    - 防禦現有的graph attack手法，而非純FGSM
    - baselines應該使用SOTA
        - 例如在node classification的accuracy，應該不是直接跟原本GCN比較，而是後面提出的最新的方法
        - 例如defense上面，在adversarial training上最好的(有找到一些也大部分有更好，但沒有放上去)，或甚至不侷限在adverssarial training上面的方法
        - 對實驗的討論不夠多，結果不夠說服人
- **目前版本的idea&novelty**
    - 在node perturbation上
        - 使用原本三種方式對features產perturbation
    - 在structure perturbation上
        - 參考Adversarial Attack on Graph Structured Data (透過DQN的方式找到兩個nodes去加減邊，以達到攻擊效果)
        - 預期做到：做DQN一樣選擇兩個點作為加減邊依據

        - state: 目前的graph
        - action: 分兩次各選一個點
        - reward: 更改後的graph去train之validation的accuracy
        - 作為adversarial training的方法
        - 選兩個點->翻轉邊->train GCN(adversarial trianing)->validation accuracy
        - 設定每次actions數量
- journal version論文準備進度
    - Introduction
        - 修改部分敘述
        - 待：contributions修改
    - Related Work
        - 依照reviewers意見，按照survey paper架構重寫一遍
        - cite更全面的graph attack & adversarial training papers
        - 提及：
            - graph embedding methods
            - adversarial attack on graphs
            - adversarial defense on graphs
        - 待：RL在graph方面的survey
    - Proposed Method
        - 原本的adversarial training方式
            - 網路學習擾動量
            - iterative產擾動
            - 更多鄰居資訊考量
        - 待：RL架構
            - flip k edges to enhance the accuracy/robustness of the node classifier
    - Experiments
        - 原本的實驗：
            - 分類準確度
                - classification accuracy
                - clean samples
            - 參數敏感性
                - iterative步數
                - 鄰居階層數量
                - 產生的擾動的penalty大小
            - 防禦能力
                - 針對FGSM攻擊，準確度的下降程度
        - 待：
            - 找問題：
                - gp再Citeseer上不夠好

        - gp的設定不夠精細：網路、penalty等
    - RL相關實驗
    - 對目前attack手法的防禦能力(*survey)
    - 和其他adversarial training的手法比較
- **實驗部分**

Method	Cora	Citeseer	PubMed
LP	68.0	45.3	63.0
DeepWalk	67.2	43.2	65.3
SemiEmb	59.0	59.6	71.1
Planetoid	75.7	64.7	77.2
GCN	81.4	69.3	79.0
GraphAT	82.5	73.4	75.2
GraphVAT	82.6	73.7	75.3
GraphSGAN	83.0	73.1	-
GAT + gp	83.6	72.3	78.2
GVAT + gp	83.7	72.3	78.4
IGAT	83.6	73.9	78.1
IGVAT	82.7	74.1	78.0
GAT + k-hop (K=2)	**84.1**	74.3	80.3
GVAT + k-hop (K=2)	83.0	74.3	77.8
IGAT + k-hop (K=2)	83.1	74.4	80.7
IGVAT + k-hop (K=2)	82.9	**74.7**	**80.9**

Table 2: Summary of results in terms of classification accuracy (%) on three citation datasets.

-

		Cora	Citeseer	PubMed
$\epsilon = 0.001$	GCN	-2.2	-2.7	-1.5
	GraphAT	-1.2	**-2.2**	-1.9
	IGAT+2-hop	**-0.5**	-2.7	**-1.2**
$\epsilon = 0.005$	GCN	-7.3	-14.0	-11.4
	GraphAT	-5.4	**-8.0**	**-5.6**
	IGAT+2-hop	**-4.4**	-9.1	-8.3
$\epsilon = 0.01$	GCN	-15.0	-34.0	-27.2
	GraphAT	-10.6	-16.4	-9.8
	IGAT+2-hop	**-9.7**	**-15.8**	**-5.8**

Table 3: Decrease of accuracy(%) on three datasets.

    - 要再把像GraphSGAN的SBVAT、OBVAT以及latent版本的AT放進去表格
  - Dataset
    - citation networks * 3
    - large graph?
  - RL

```
====== begin of s2v configuration ======
| msg_average = 0
======   end of s2v configuration ======
# train: 13500  # test: 1500
loss: 0.32309 acc: 0.88000: 100%|
average training of epoch 0: loss 0.60715 acc 0.81393
loss: 0.24697 acc: 0.90000: 100%|
average test of epoch 0: loss 0.32467 acc 0.88267
----saving to best model since this is the best valid loss so far.----
loss: 0.22634 acc: 0.92000: 100%|
average training of epoch 1: loss 0.28475 acc 0.89941
loss: 0.16270 acc: 0.96000: 100%|
average test of epoch 1: loss 0.21167 acc 0.93133
----saving to best model since this is the best valid loss so far.----
loss: 0.12939 acc: 0.98000: 100%|
average training of epoch 2: loss 0.23782 acc 0.92230
loss: 0.10685 acc: 0.98000: 100%|
average test of epoch 2: loss 0.19641 acc 0.92733
----saving to best model since this is the best valid loss so far.----
loss: 0.14827 acc: 0.94000: 100%|
average training of epoch 3: loss 0.17027 acc 0.94178
loss: 0.00765 acc: 1.00000: 100%|
average test of epoch 3: loss 0.14735 acc 0.94333
----saving to best model since this is the best valid loss so far.----
loss: 0.19938 acc: 0.96000:  56%|
```

- dqn的理解
- code本身的理解(c++ & Python)
- code classification修好了可以跑，但attack部分還不行，有cuda相容性問題，至今仍在解決中
- 仍在改寫code中 T_____T
  - a. code本身cuda環境的問題(解決中)
  - b. DQN的QNet從s2v(C++)改成GCN去建embedding
  - c. reward設定改掉: acc下降越多越高->validation accuracy越高

- 原本想參考的code:
  - Adversarial Attack on Graph Structured Data
  - 在最後的attack(DQN) part遇到以下問題：
    - `undefined symbol: __cudaRegisterFatBinaryEnd`

```
Namespace(adj_norm=1, base_model_dump='../../dropbox//scratch/results/graph_classification/components/nodes-90-100-p-0.02-c-1-3-lv-2/epoch-best', batch_size=50, bilin_q=0, cross_rate=0.1, ctx='cpu', data_folder='../../dropbox//data/components', dataset=None, del_rate=0, dropout=0.5, er_p=0.02, feat_dim=0, feature_dim=None, fold=1, frac_meta=0, gm='mean_field', hidden=32, idx_start=None, latent_dim=64, learning_rate=0.001, logfile=None, max_c=3, max_lv=2, max_n=100, meta_test=0, min_c=1, min_n=90, mlp_hidden=64, mutate_rate=0.2, n_graphs=5000, n_hops=2, num_class=None, num_epochs=1000, num_instances=None, num_mod=1, num_steps=100000, out_dim=0, phase='train', population_size=100, rand_att_type='random', reward_type=None, rounds=10, save_dir='./saved', saved_model=None, seed=1, targeted=0, weight_decay=0.0005)
Traceback (most recent call last):
  File "grad_attack.py", line 17, in <module>
    from q_net import NStepQNet, QNet, greedy_actions
  File "/mnt/sda/wwj/graph_adversarial_attack/code/graph_attack/q_net.py", line 22, in <module>
    from modules.custom_mod import JaggedMaxModule
  File "/mnt/sda/wwj/graph_adversarial_attack/code/graph_attack/../common/modules/custom_mod.py", line 2, in <module>
    from functions.custom_func import JaggedLogSoftmax, JaggedArgmax, JaggedMax
  File "/mnt/sda/wwj/graph_adversarial_attack/code/graph_attack/../common/functions/custom_func.py", line 3, in <module>
    from _ext import my_lib
  File "/mnt/sda/wwj/graph_adversarial_attack/code/graph_attack/../common/_ext/my_lib/__init__.py", line 3, in <module>
    from ._my_lib import lib as _lib, ffi as _ffi
ImportError: /mnt/sda/wwj/graph_adversarial_attack/code/graph_attack/../common/_ext/my_lib/_my_lib.so: undefined symbol: __cudaRegisterFatBinaryEnd
```

    - try:
      - python 3.6
      - pytorch 0.3.1 (code)
      - pytorch 0.4.1
      - cuda 10.0 (server)
      - cuda 10.1
- Graph + RL
  - state:

        - 每個node去看各自的一階二階鄰居的embedding
        - 何向南：uniform sample -> 在state上的影響力？
    - 把何向南的code換成PyTorch：大致完成但還沒驗證
    - Graph: PyTorch geometric
    - RL的customized environment：大致上完成
    - RL的training：還在進行中
    - 問題:
        - choose two nodes 的方法
        - Total reward 定義： 包含最後一次的val acc 和 modify次數(越低越好)
        - 停下的條件：已達到最大modify次數上限or val acc提升幅度達某值
- future
    - RL
    - baselines (AT, RL...)
    - defense existing methods
    - 架構圖 & 演算法

link prediction

- Adversarial Attack on Graph Structured Data
    - cuda 10.1 to 9.1
        - follow official instructions
        - and failed
        - now: cuda 11.2
        - compiler: 10.1
    - 目前詢問的解法也失敗
    - 需要的原因：減小計算量，否則large graph會爆炸
- link prediction survey
    - attack, defense, adversarial training
    - https://docs.google.com/spreadsheets/d/1efCgKljusikXaRNompziPuVo91qqYQA13AYWtrQ1VRc/edit?usp=sharing
- RL code
    - 有bug需解決 (待換在server上看有沒有問題)
        - load customized environment

- followed the official instructions on gym GitHub but got the following error:

```
[149]  1 !pip install gym-graph

       Requirement already satisfied: gym-graph in ./gym-graph (0.0.1)
       Requirement already satisfied: gym in /usr/local/lib/python3.6/dist-packages (f
       Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages
       Requirement already satisfied: pyglet<=1.5.0,>=1.4.0 in /usr/local/lib/python3.
       Requirement already satisfied: cloudpickle<1.7.0,>=1.2.0 in /usr/local/lib/pyth
       Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.6/dist-p
       Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages
```

```
1 import gym
2 import gym_graph
3 # env = gym.make('gym_graph:graph-v0')
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-150-661229369324> in <module>()
      1 import gym
----> 2 import gym_graph
      3 # env = gym.make('gym_graph:graph-v0')

ModuleNotFoundError: No module named 'gym_graph'
```

  - action space
    - n * n
    - preprocess
      - attack: similarity edges
      - defense: otherwise
      - adversarial training: ?
  - state space
    - 2 1-hop neighbors & 2 2-hop neighbors of each node (shape=[2708,14])
- baslines
  - adversarial training
    - 何向南
    - GraphSGAN
    - SBVAT OBVAT
    - GCN and traditional methods

41	GCN	81.5%	YES

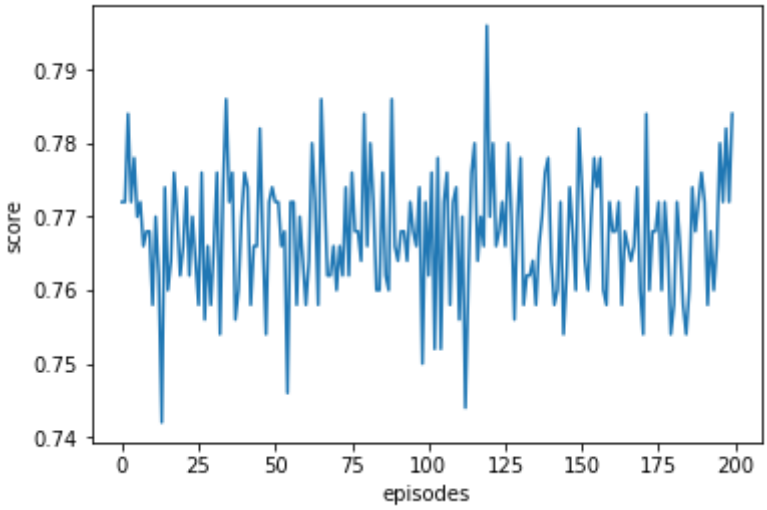
RANK	MODEL	ACCURACY	TRAINING SPLIT	VALIDATION	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	SSP	90.16%			×	Optimization of Graph Neural Networks with Natural Gradient Descent			2020
2	SplineCNN	89.48%			×	SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels			2017
3	GCN-LPA	88.5%			×	Unifying Graph Convolutional Neural Networks and Label Propagation			2020
4	AS-GCN	87.4%			✓	Adaptive Sampling Towards Fast Graph Representation Learning			2018
5	NodeNet	86.80%			×	NodeNet: A Graph Regularised Neural Network for Node Classification			2020
6	DFNet-ATT	86%			×	DFNets: Spectral CNNs for Graphs with Feedback-Looped Filters			2019
7	GResNet (GAT)	85.5%			×	GResNet: Graph Residual Network for Reviving Deep GNNs from Suspended Animation			2019
8	AdaGCN	85.46%			×	AdaGCN: Adaboosting Graph Convolutional Networks into Deep Models			2019
9	PPNP	85.29%		YES	✓	Predict then Propagate: Graph Neural Networks meet Personalized PageRank			2018
10	DifNet	85.1%			×	Get Rid of Suspended Animation Problem: Deep Diffusive Neural Network on Graph Semi-Supervised			2020

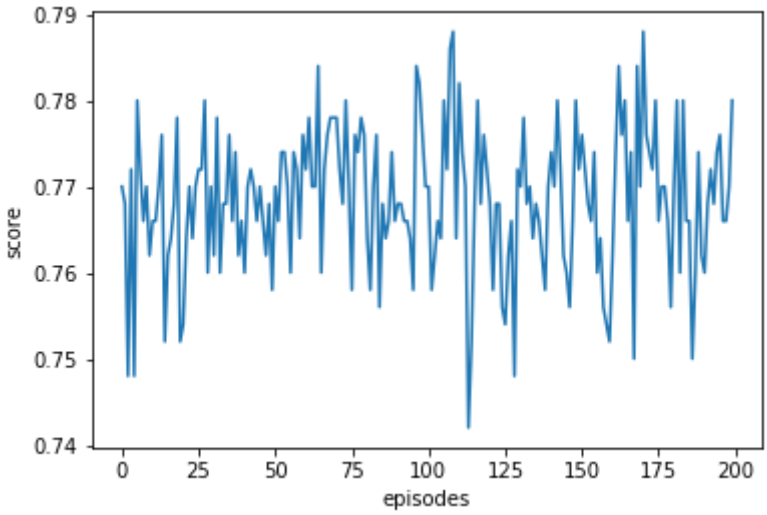
- Optimization of Graph Neural Networks with Natural Gradient Descent

1 SSP 90.16%

- RL code
  - [x] 把server cuda環境復原好(10.1,tf-gpu==2.3.0)
  - [x] 重寫環境兜上去，先用選擇一個node的方法去train
  - [x] replay buffer有bug還在解決中
  - [ ] 刻不能跑那篇的選擇兩個node的方法
- link prediction survey
- baselines survey
  - on node classification
  - but not adversarial training

- RL Graph Adversarial Training
  - simple version

        - 選2 nodes去做
        - 最多改10 edges
        - next state: new graph
    - 目前狀況
        - on Cora dataset
        - loss 有穩定下降後停在某處徘徊
        - reward (val acc)沒有顯著進步 (cleanGCN: 0.76)
        - 

        - 

        - 繼續調參、調整設定中，網路再複雜些
        - 在改寫何教授的code+兜上去

- Writing
    - Story
        - **AT的必要性：** adversarial training 在面對adversarial attack的時候，相較於detection，會是比較適用的手段，因為在training的時候已經訓練model有一定的防禦能力，而在目前幾篇graph上面的adversarial training也可以看到比較好的成效
        - **統一架構：** Training with adversarial goals and Training with adversarial examples (from survey paper: A Survey of Adversarial Learning on Graph)

        - 基於何教授的方法改進，在features上面用更進階的方式做adversarial training；而在edge perturbations上，則使用reinforcement learning中的DQN來做，除了
            - 1.能作為discrete問題的解
            - 2.透過把validation的效果作為reward，試圖解決node classification上training跟testing效果差距略大的問題
            - 3.加減邊在training時作為AT的手法，在真實情況下，讓DQN遇到一graph的時候可以知道如何加減邊來完成訓練並達到更好的效果
        - reinforcement learning 用在graph的adversarial training上，並同時考量feature 和edge perturbation，以建構出來的訓練模型
    - Reinforced Negative Sampling over Knowledge Graph for Recommendation
    - Improving the Robustness of Wasserstein Embedding by Adversarial PAC-Bayesian Learning
    - Graph Adversarial Training: Dynamically Regularizing Based on Graph Structure
    - Adversarial Personalized Ranking for Recommendation
- Experiments
    - RL: 改幾條邊、state看的是什麼(目前: embedding)、參數實驗...
    - GraphVAT: 原本的那些實驗
    - 試著比較：RL跟直接選degree大的邊做flipping的adversarial training效果差別等相關實驗
    - baslines: survey
        - 防禦別人的adversarial attack
        - 純node classification方法疊加上去效果是否提升
    - 2020的survey也只有目前找到的那些AT方法
    - large dataset
    - 其他

上次的問題

- 寫成PyTorch的時候純GraphVAT出來的結果和原本tf code不同
    - 僅在torch.geometric裡面定義Conv層和tf code可能有些微不同，但其他皆同
- 解決方法
    - 找錯or不同處
    - 確定tf code運作正常
    - 換把tf的code和DQN接，重寫部分code
    - 確定純GraphVAT出來的validation accuracy結果合理

Training RL

- state

    - (2708, 14) 每個node看到自己兩個鄰居的embedding
- action
    - (n1, n2) 一次選取2 nodes所決定的邊
    - (n1, n2) 原本就存在 -> drop edge
    - (n1, n2) 原本不存在 -> add edge
- reward
    - 加減邊後的graph拿去train GraphVAT
    - 以validation accuracy作為reward

Problem

還是很抖動，拉長epoch數看似會好一點，但也不是穩定的。 而且要train蠻久的。

目前有嘗試調整

- learning rate
- epsilon_decay rate
- batch_size
- replay
- epochs
- 網路大小
    - state目前是(2708,14)，有可能網路不夠大學不起來？
- 調整可以改動的邊數: 10,5,3
- 改動N條邊(N次動作)的memory(s,a,r,s')
    - r 同
    - s, r, s' 同 (X)
- 原本有想要改state，但直接改成adj會OOM
- trace rl-graph-attack paper 的 code 看有沒有什麼可以改

原本純GraphVAT on Cora: validation accuracy=0.80 純GCN: 0.79

batch_size++ 拉長epochs

DQN

- State
    - an edge (n1, n2)
    - (n1, n2) 可能原本存在or不存在

    - n1, n2 取法：
        - 從目前被分錯的nodes中random選出兩點
        - 從目前被分錯的nodes中random選出一點，另一點random取得
        - 從目前被分錯的nodes中random選出一點，另一點從第一點的鄰居random取得
- Action
    - 0 or 1
        - 0: 不要flip
        - 1: 要flip: add or drop edge
- Reward
    - 和原本純GraphVAT的validation accuracy比較
    - reward = reward - original_val_acc
    - 0.00xx -> *1000
- Episode reward
    - modify 10次 (action=1 累計10次) 之 reward 加總
- Memory
    - 原本是10條edges同一個reward(上次改的)
    - 先改回每條有自己的reward
- Result Time:1/10, state: [314, 2024], action: [1], reward: 0.0080 Time:2/10, state: [137, 2110], action: [1], reward: 0.0080 Time:3/10, state: [334, 1485], action: [1], reward: 0.0080 Time:4/10, state: [441, 2582], action: [1], reward: 0.0080 Time:5/10, state: [151, 481], action: [1], reward: 0.0060 Time:6/10, state: [362, 2009], action: [1], reward: 0.0060 Time:7/10, state: [264, 2038], action: [1], reward: 0.0060 Time:8/10, state: [138, 144], action: [1], reward: 0.0060 Time:9/10, state: [331, 1143], action: [1], reward: 0.0060 Time:10/10, state: [202, 2063], action: [1], reward: 0.0060 episode: 0/300, score: reward->**67.9999589920044** and count->10, epsilon: 1.0
- Question
    - acc只是一點點增長
        - 相差的*1000、10條邊加總
        - 所以回推效果要/10000
        - e.g. Episode reward=150, 150/10000=0.015，代表原本validation上的效果提升1.5%
    - 300 episodes 需要兩個小時或更久，但episode數量感覺要再多才會收斂
    - 限制一定要10次才會結束一個episode，是否必要
    - state 取點的方式還要再想
        - 不管怎樣初始都還有random成分在
    - DropEdges (ICLR, 2020)
        - paper
        - https://github.com/tkipf/pygcn/blob/master/pygcn/train.py
    - Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications

0303

- initial state: 分最差的node
- state: self and neighbors' embedding
- action: choose a neighbor
- reward: validation accuracy

survey

- fake nodes on adv
  - generate X using GAN
  - RL
    - s: graph
    - a: add edges & label of fake nodes
    - r: attack效果

todo

- methods
- RL-S2V NIPA code
- defense against other attack methods

0310

架構

- State
  - 分錯的點 和 他的鄰居 和 emb和他近但不是鄰居的點 的embedding
- Action
  - drop edge: 刪掉目前的某個鄰居
  - add edge: 和最近但沒連的點連接
  - AT training strategies: 選擇其中一種作為AT訓練的超參數
- Reward
  - 做完N次以後的validation accuracy

目前進度

改了drop/add edges的地方，正在train一版純structure的。 改features的部分這兩天會再加進去。

嘗試：

1. GCN不能有isolated nodes，所以如果剩下一個鄰居且選到要刪掉與這個鄰居的edge的時候，會出現不動作的情況，但是計入總動作次數

2. 若只有兩個鄰居，但選到要刪掉第三個鄰居：目前先用random從前兩個選一個來刪掉=仍會做動作，只是變成從現有的選擇中隨機選擇
3. 下一個state可以從分的最錯的點來選，或是前一個點有做過動作的點來選，前者會遇到的點都差不多，後者可能會限制他的效果
4. 超過三個鄰居的節點因為需要隨機選三個做為action，所以如果同一個node，同一個action，不見得會drop掉同一條edge
5. 目前add edge只連最靠近的那個點
6. 因為邊數有一萬多，總動作次數會再嘗試更多10,20,50...，目前在跑20的
7. 只看embedding好像state蠻小的，也把feature加進去state

老師feedback:

1. 實驗有一個可以是拔掉自己方法部分的實驗
2. 找至少兩篇要投的期刊上面相關論文去cite
3. 兜起來的時候確定甚麼都不做的時候可以到暑假的效果orz
4. 方法的地方可以一邊寫一邊講為甚麼要這樣做
5. 實驗碩論要四個、投稿要三個
6. **如果前後加減同一條邊的話，先不要理他**

下禮拜進度：

1. 純structure 的結果
2. 兩個合在一起 的結果 **(至少要跟暑假的表現持平!!!!)** 2-hop+iter

我自己的進度：寫字 暑假實驗記錄

1. 確定DQN的training.....(哀
2. 能有機會的話跑citeseer或pubmed
3. RL相關文獻! 還是不要提到fake nodes好ㄌ(自打臉
4. 下下週：把一二章也慢慢去加完整
5. RL的validation和testing在哪XD
6. 最後表上的結果應該是rl reward最好的那個策略 出現的validation acc
7. novelty!!! 說服的點

每天要做啥

1. 確定cora最好的hyperparameters set
2. 如何選擇 或把RL和2hop+iter合起來?
3. 確定RL 的 replay buffer等等可以調的東西到底有多少，batch size=32, replay buffer 256? 每次丟掉一點點就好不要直接清空?
4. train到最後應該會樂單一，就是會是表現最好的那幾種策略

5. tf1 到底有沒有寫錯QQ
6. 先把所有的小標題確定好架構orz
7. 寫第四章///
8. "a b" 去查，才會知道這兩個字邏輯上可不可以放在一起
9. 重點放在前或後，不要想到甚麼寫甚麼
10. 不要寫超級長的句子
11. 要寫WE
12. 主動被動交叉
13. 用詞可以更精確，更專業der
14. 先等patrick給我意見，再繼續寫？
15. 取三次一個或是一次取三個(才不會重複)
16. TMUX來跑 不要關掉網頁就壞掉

training notes

0312

```
Epoch: 0206 tr_obj= 0.8154 tr_acc= 0.9857 tr_loss= 0.2491 tr_l2= 696.2010 tr_vat= 0.000000 tr_gat= 0.436409
Epoch: 0206 va_obj= 1.2534 va_acc= 0.8000 va_loss= 0.6873 va_l2= 696.2478 va_vat= 0.000000 va_gat= 0.436047
Epoch: 0206 te_obj= 1.2373 te_acc= 0.8360 te_loss= 0.6712 te_l2= 696.2478 te_vat= 0.000000 te_vat= 0.436047
------- time= 0.53169 ------
Epoch: 0207 tr_obj= 0.8120 tr_acc= 0.9857 tr_loss= 0.2481 tr_l2= 696.2478 tr_vat= 0.000000 tr_gat= 0.431398
Epoch: 0207 va_obj= 1.2513 va_acc= 0.8040 va_loss= 0.6867 va_l2= 696.0844 va_vat= 0.000000 va_gat= 0.432969
Epoch: 0207 te_obj= 1.2348 te_acc= 0.8370 te_loss= 0.6703 te_l2= 696.0844 te_vat= 0.000000 te_vat= 0.432969
------- time= 0.52639 ------
Epoch: 0208 tr_obj= 0.8136 tr_acc= 0.9857 tr_loss= 0.2474 tr_l2= 696.0844 tr_vat= 0.000000 tr_gat= 0.436367
Epoch: 0208 va_obj= 1.2527 va_acc= 0.8040 va_loss= 0.6866 va_l2= 695.7090 va_vat= 0.000000 va_gat= 0.436468
Epoch: 0208 te_obj= 1.2361 te_acc= 0.8320 te_loss= 0.6700 te_l2= 695.7090 te_vat= 0.000000 te_vat= 0.436468
------- time= 0.52736 ------
Epoch: 0209 tr_obj= 0.8120 tr_acc= 0.9857 tr_loss= 0.2467 tr_l2= 695.7090 tr_vat= 0.000000 tr_gat= 0.434869
Epoch: 0209 va_obj= 1.2516 va_acc= 0.8060 va_loss= 0.6868 va_l2= 695.0722 va_vat= 0.000000 va_gat= 0.434614
Epoch: 0209 te_obj= 1.2350 te_acc= 0.8320 te_loss= 0.6702 te_l2= 695.0722 te_vat= 0.000000 te_vat= 0.434614
------- time= 0.53437 ------
Epoch: 0210 tr_obj= 0.8182 tr_acc= 0.9857 tr_loss= 0.2462 tr_l2= 695.0722 tr_vat= 0.000000 tr_gat= 0.448957
Epoch: 0210 va_obj= 1.2579 va_acc= 0.8060 va_loss= 0.6861 va_l2= 694.5256 va_vat= 0.000000 va_gat= 0.449121
Epoch: 0210 te_obj= 1.2414 te_acc= 0.8330 te_loss= 0.6696 te_l2= 694.5256 te_vat= 0.000000 te_vat= 0.449121
------- time= 0.52949 ------
Epoch: 0211 tr_obj= 0.8131 tr_acc= 0.9857 tr_loss= 0.2457 tr_l2= 694.5256 tr_vat= 0.000000 tr_gat= 0.440315
Epoch: 0211 va_obj= 1.2530 va_acc= 0.8020 va_loss= 0.6864 va_l2= 693.9240 va_vat= 0.000000 va_gat= 0.439207
Epoch: 0211 te_obj= 1.2364 te_acc= 0.8350 te_loss= 0.6699 te_l2= 693.9240 te_vat= 0.000000 te_vat= 0.439207
------- time= 0.52674 ------
Epoch: 0212 tr_obj= 0.8093 tr_acc= 0.9857 tr_loss= 0.2447 tr_l2= 693.9240 tr_vat= 0.000000 tr_gat= 0.435120
Epoch: 0212 va_obj= 1.2510 va_acc= 0.8040 va_loss= 0.6867 va_l2= 693.3435 va_vat= 0.000000 va_gat= 0.435333
Epoch: 0212 te_obj= 1.2345 te_acc= 0.8370 te_loss= 0.6702 te_l2= 693.3435 te_vat= 0.000000 te_vat= 0.435333
------- time= 0.52447 ------
Epoch: 0213 tr_obj= 0.8069 tr_acc= 0.9929 tr_loss= 0.2438 tr_l2= 693.3435 tr_vat= 0.000000 tr_gat= 0.432895
Epoch: 0213 va_obj= 1.2501 va_acc= 0.8060 va_loss= 0.6871 va_l2= 692.9330 va_vat= 0.000000 va_gat= 0.433039
Epoch: 0213 te_obj= 1.2334 te_acc= 0.8380 te_loss= 0.6704 te_l2= 692.9330 te_vat= 0.000000 te_vat= 0.433039
------- time= 0.53757 ------
Epoch: 0214 tr_obj= 0.8031 tr_acc= 0.9929 tr_loss= 0.2431 tr_l2= 692.9330 tr_vat= 0.000000 tr_gat= 0.427033
Epoch: 0214 va_obj= 1.2469 va_acc= 0.8060 va_loss= 0.6873 va_l2= 692.6039 va_vat= 0.000000 va_gat= 0.426527
Epoch: 0214 te_obj= 1.2300 te_acc= 0.8370 te_loss= 0.6704 te_l2= 692.6039 te_vat= 0.000000 te_vat= 0.426527
------- time= 0.52652 ------
Early stopping...
Optimization Finished!
Test set results: cost= 1.23598 accuracy= 0.83700 time= 1.38609
(rlvat) wwj@kotzujian:/mnt/sda/wwj/GraphAT-master$
```

跑vat-2-hop-iter，beta改0.5可以到的結果 (tf cpu 1.14.0) (env:rlvat)

可以看到最後validation都是比testing爛的....

0315

```
In [25]: t = time.time()
         create_col_indices(adj)
         time.time()-t
Out[25]: 0.25051045417785645

In [26]: cl[47]
Out[26]: array([ 163, 1579], dtype=int32)

In [27]: t = time.time()

         cl[47] = np.append(cl[47], 163)
         cl[47] = np.sort(cl[47])

         cl[163] = np.append(cl[163], 47)
         cl[163] = np.sort(cl[163])

         time.time()-t
Out[27]: 0.00028967857360839844
```

這個時間差很多啊...還是寫成下面的寫法好ㄌ

0317

:::info :bulb: 增減邊和改feature整合進大型RL並有實驗結果 :::

- 完成架構
  - Natural DQN + GVAT
  - Double DQN + GVAT
- 細節設定
  - state:
    - emb (7, 7)
    - fea (7, 1433)
    - emb+fea (7, 1440)
    - adj (7, 2707)
  - action:
    - add
    - drop
    - strategies (2hop+iter, 2hop, iter)
  - reward:
    - r + R
    - R

    - 個別嘗試&個別的好壞
- 目前
    - 花費時間
        - train each step: 很久
        - train each episode: 5~12 hours
    - 修正整體架構的問題 tf1
    - ce_loss最高者會被當作下一次的state
    - if train each episode: (不合理)
        - 從選過最多次的strategy 配合 加減邊 下去做
        - 每次episode開始就決定好target_nodes，initialized weights一樣，但ce_loss最高者不完全相同 -> sample
    - if train each step: (目前)
        - 每次action完重新train，下一個state是看新的emb
        - 考慮要不要加入adjacency
        -
    - reward的定義：
        - 原本GVAT acc約在0.79~0.80，目前以此為基準去給reward
        - 因為看total reward，有可能沒辦法學到某個strategy可以讓表現更好(表現更好的效果也沒有到特別突出)
        - 想&改成別的定義，不要直接單用R或R+r
    -
- next week
    - 比較合理的設定 & 實驗結果

嘗試

trajectory

- 每次train最不好的點不一定完全相同，reward的高低取決於選到的點(state)、選到的action，abcde->0.3和a
- 動作 最大值 100，不動作 不計入次數 但計入100，總動作多少次(達到?效果停止)，計入reward的計算->需要定

0324

state兩種設計

1. 每個node跑一次，個別吐出node各自需要做的動作
2. 一次吐出所有node要做的事<-目前的

reward兩種設計

1. 分對的數量
2. validation accuracy

目前

- [x] check tensorflow GCN&DDQN init&fixed
- [x] DDQN memory 因應目前架構修改
- [x] 嘗試一些可以不要變慢的方法: del session等
  - [ ] -> 但還是會慢下來，最初跑一個動作6~ 8s，過了150次動作以後約停留在9~13s
- [ ] q network loss有在下降但reward仍然浮動不定

problems

1. 時間：目前的架構在100次actions以後會漸漸變慢
2. 網路：可能目前簡單dnn學不起來
3. 調參：lr, learning的頻率
4. reward: 需要再多嘗試各種設計
   - 看那個點(做了那個動作之後)分錯與否or ce_loss，一起考量到reward當中，可以把美個點的reward區別出來
   - reward如果相減
     - 可以看動作好壞 但少了真實的acc(base)

next week

- 繼續try
- 計算整體時間總共要花多久
- [ ] reward 不要140一樣，可能5,10,20,25個點共享一個reward就好
- [ ] memory要包含多一點不同的reward才會學道不同東西
- [ ] 奇威：(a,b)added跟(b,a)dropped要怎麼區別出好壞、還是會被當作沒這件事發生？
- [ ] 比何向南好、還是比上次的自己好？(可能訂一個reward-baseline)
- [ ] 存model weights(DDQN

0331

換了一個dataset: Citeseer

<- 好可怕，然後他就爆了

- 可以採到比之前所有更好的結果0.74x(0.742 or 0.740) -> 0.756
- 有點像是 隨機踩會train的比完全不要drop add來得高...(汗)
- 不會持續高 因為鄰居的隨機性？

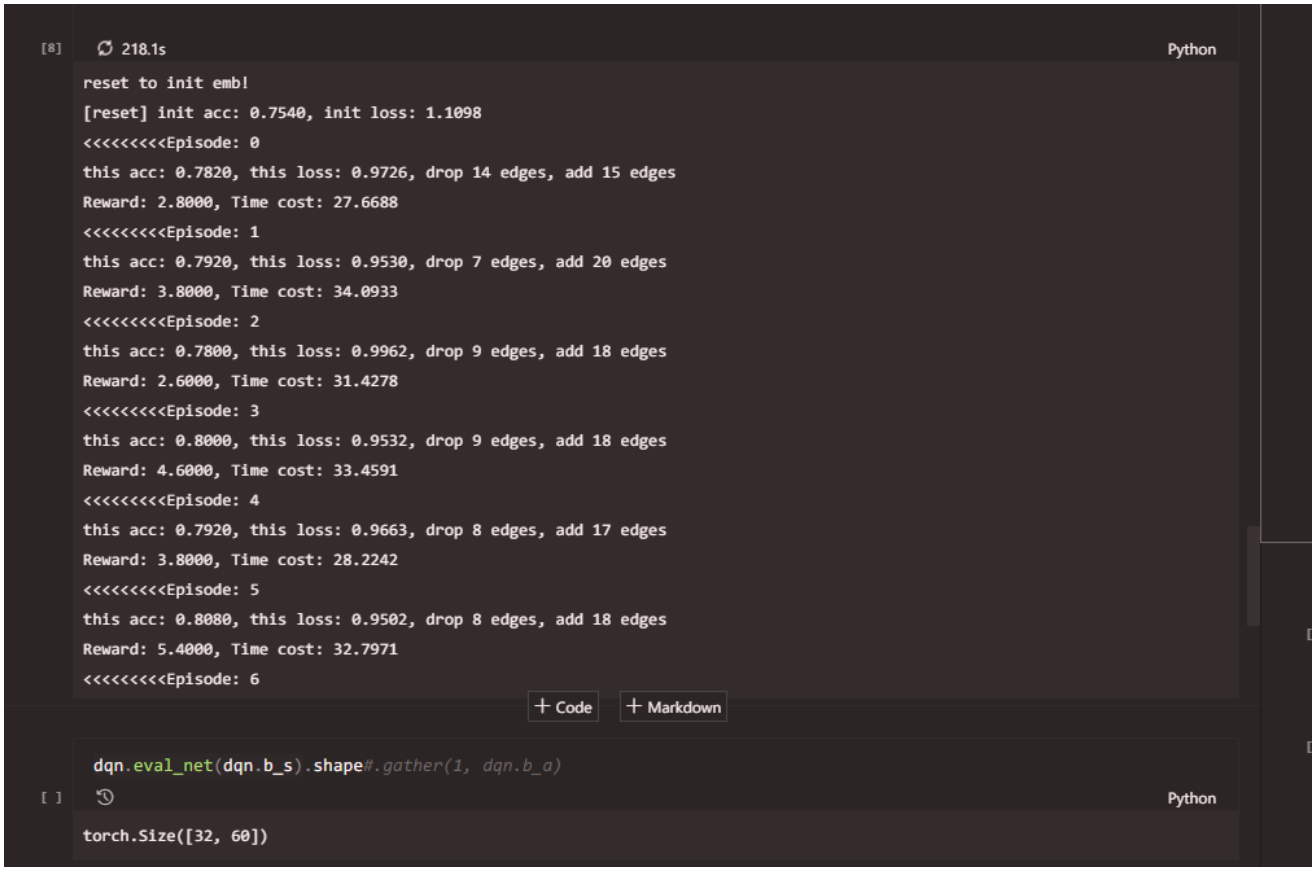
- 想法：
  - 把最高那次留起來，紀錄改了哪些drop/add edges，以及是哪些nodes使用了strategy(可以這樣冂)
  - 作為一種"使用聰明的方法試圖找到比較好的training/validation結果"去作為rl的用意
  - **BUT!!!完全沒有strategy!!!**
- 有strategy的話，可以超過向南，但是不會超過只有strategy的(持平或更低)，很偶爾會踩到0.75QQ!!

```
<<<<<<<<<Episode: 35
action shape: (120,)
this acc: 0.7500, this loss: 1.7924, drop 27 edges, add 20 edges
Reward: 0.8000, Time cost: 50.7330
start to learn!
```

- 恆~~~

```
<<<<<<<<<Episode: 15
this acc: 0.7520, this loss: 1.7606, drop 13 edges, add 26 edges
Reward: 1.2000, Time cost: 44.8154
start to learn!
<<<<<<<<<Episode: 16
```

PubMed

- 有時候會好: 0.79->0.812
  - 

- 但最後也差不多是會贏向南，但不會贏自己(會踩到，但不會一直)
  https://zhuanlan.zhihu.com/p/260703124 https://arxiv.org/pdf/1512.07679.pdf

](https://i.imgur.com/0LKw1vm.png =600x600)

0331

- 要做的實驗
- 要嘗試的
  - cora、pubmed tune出好的結果
    - pubmed training有蠻明顯的效果但在testing上會不如training
  - state多看一點東西：
    - 上一次是否分對、和鄰居的KL距離(要有均值)等
  - reward 的設定：
    - validation效果好，但不見得反映在testing上
- 困難
  - 遇到太多次跑到最後停掉
  - 所以一直在換&改了各種寫法
- 老師的建議
  - tune到好的那個就當作method
  - sess.close()
  - 把memory定期清掉
- 要做的事
  - 如何取得training後的結果(最後N次？不要只有一次
  - sample memory一直都是隨機抽？還是有透過q value篩選(要再看詳細一點
  - testing有跑出好結果的action要存起來!!!存好當時的training跟testing 結果
  - 步數實驗 想想要怎麼做會比較好
  - 五次更新 還是一直看原本的？
  - state多考量一點東西，例如鄰居的距離、標準化，或是和正確答案的距離標準化
  - 實驗 + 對抗poisoned graph的部分？
    - 現成的poisoned graph在哪裡？
  - 每一段要怎麼敘述比較好，比較有統一且完整的故事？
  - 盡量讓RLGVAT可以是最好的，f- e- 都其次@@
  - cora的strategy要不要換成 gat+khop
  - pubmed可以挑挑看 training不要那麼高的方式(要有個衡量標準)

0407

目前

- state: emb + 看min_max(與鄰居的距離)
- action: training nodes +/-/features
- reward: val_acc 提升程度 + val nodes和鄰居距離減少程度

0414

0421

0428

- 改上次提到的圖&更新結果
- 交給老師跟Apple初版
- 這幾天細修&想其他實驗