Grpah Mining HW3 - Nettack

109065510 姜林寬

首先看懂助教的隨機攻擊模型，由於Nettack是灰箱攻擊，無法得知要攻擊之已訓練模型裡的參數，所以我們需要自己設計一個GCN作為surrogate model，藉由攻擊這個模型得到反饋以優化自己的攻擊模型，助教已經做好了這個模型，直接拿來用就行了，不過由於要算擾動的edge對目標模型的影響，我多加了 **surrogate.get_weight()** 來取得surrogate model第一層與第二層的weight，之後實作attack model時用來評估。

```
model.attack(features, adj, labels, idx_train, w1, w2, target_node, \
            n_added=1, n_perturbations=n_perturbations)
```

在熟讀Adversarial Attacks on Neural Networks for Graph Data這篇論文後大致了解主要的流程，首先是對收進來的參數進行一些前處理，接著要先準備好loss function用來評估攻擊效果，利用從surrogate model獲得的weight算出logits，這是NN算出數值丟入最終激活函數前的向量，任務目標只要讓原本正確的類別分類錯誤，所以取正確類別的值減掉錯誤類別中概率最高的那類當作loss：

```
surrogate_losses = [logits_start[label_u] - logits_start[best_wrong_class]]
```

由於作業可以使用direct attack，所以對於能擾動的edge不用多做限制，Graph中所有node皆可以對其潛在邊進行擾動，上課提到擾動架構的攻擊效果會比擾動特徵好，所以我就只做改變特徵。在選擇擾動邊時需要參考的指標是struct score，其公式與surrogate_losses類似，將擾動後正確類別之logits減掉錯誤類別中最大的logits：

```
struct_scores = logits_for_correct_class - best_wrong_class_logits
```

從以上算法可知我們每次在選擇擾動邊時都要選擇分數最低的(這點跟原論文相反，這裡我為了讓score與surrogate_losses的格式相似改動了順序)，接著就是根據n_perturbations依序選當下最高struct_scores的擾動邊，n_perturbations的算法是根據助教的寫法，為**目標node的degree數+2**，在這個攻擊預算下去跑一個for迴圈，每次for迴圈首先先對所有潛在擾動邊去算出擾動後的adjacency matrix’ ，利用adjacency matrix’乘以已經訓練好的surrogate model所得到的weight可以得到每個class的logits，再拿logits去算每個擾動邊的struct_scores。

找到合適的擾動邊後就翻轉0,1存到modified_adj，由於adjacency matrix是對稱的，所以擾動要改matrix的兩個位置：

```
for node1,node2 in structure_perturbations:
    modified_adj[node1, node2] = 1 - modified_adj[node1, node2]
    modified_adj[node2, node1] = 1 - modified_adj[node2, node1]
```

之後用modified_adj取代原本的adj matrix，訓練後可以得到特定node的logits進而得知有沒有攻擊成功，到這裡為止雖然我沒有統計過確切正確率是多少，我將作業的五筆公開資料重複實驗，發現偶爾會有攻擊失敗的情況發生，所以再重新從原始paper以及他公開的方法找解法。

有一步是原本作業沒有要求的，就是考慮擾動前後的Graph結構之distribution，由於這個distribution會符合power law，作者在選擾動邊時會選擇alpha變動小的，也就是power law相似的，所以參考作者計算alpha的算法，將alpha的變動也考慮進去，做法是每次擾動前後都會存一份整張graph，然後計算alpha進而求得log likelihood，對這個值設一個門檻變成filter，對可擾動邊進行篩選，將不符合的擾動邊去除，不考慮動這些邊，從剩下邊選最適合的擾動邊，然後再把這個變動加上現有的圖進行下一階段的選擇。由於每次加一條邊就要重算alpha與log likelihood，所以這裡才會需要numba的幫助。

在加入考慮alpha變動的條件後，整體效果是變好了(一樣以同樣五組攻擊目標重複跑大量實驗來比較)，我原本的想法是有alpha限制應該會讓能攻擊的邊選擇變少，限定在不會太明顯的邊，這樣攻擊效果會變差，但經過實測以後攻擊成功率卻是提高的，在此我提出我認為的可能：

- 多考慮了擾動前後的分佈，所過濾出的edges set整體攻擊品質是往上提升，所以從中選擇大都不會太差
- 因為Nettack是greedy的，每步都選當下分數最好的，且每次的選擇都會影響下次選擇，所以可能會錯過global optima，第一次選擇分數最好的不一定最後就是最佳解
- 在一開始就過濾掉品質不夠好的擾動邊可能可以提升整體攻擊效果