Data Science Homework 5 Report

梁湘梅　110062661

2022 年 6 月 24 日修訂

I. 介紹

本次作業主題是實作 MERIT，MERIT 是一種 self-supervised learning 模型，它首先對輸入的 Graph 進行 Graph Augmentations 生成兩個 Graph View，再利用 Online Network 和 Target Network 進行學習，並設計了 Cross-network 和 Cross-view 來計算模型的 loss。

根據作業要求我們需要實作 Graph Augmentations 中的 Node Feature Masking(NFM)與 Cross-network 和 Cross-view 來計算模型的 Loss。

我們使用作業提供的 Cora 作為訓練資料，使用的 Python 版本為 python3.7，作業系統為 windows 10，GPU 為 NVIDIA GeForce GTX 1650 with Max-Q Design。

實驗結果：Accuracy =0.826, Time=2580s。

II. 程式內容說明

在 NFM 的部分，我們需實做下列函式：

$$\text{aug\_feature\_dropout}(input\_feat, \text{drop\_percent} = p)$$

aug_feature_dropout() 會將輸入的 Feature Matrix ($input\_feat$) 以隨機的方式將 $p \times input\_feat.shape[1]$ 個 Node Feature 去除。完整程式碼在 **aug.py** 中。

在aug_feature_dropout()中，我們首先須對傳入的$input_feat$進行 Deep Copy 以建立一份完全獨立的變數(aug_input_feat)，確保我們之後的操作不會修改到原始資料。接著，計算出我們要修改的 Node Feature 數量：

$$drop\_feat\_num = (p \times input\_feat.shape[1]).$$

再來，我們以 Random Sample 的方式選出$drop_feat_num$個要修改成$\vec{0}$的 Index($drop\_indx$)，將 aug_input_feat 內被選中的 Index ($drop\_indx$) 其 Element 修改成 $\vec{0}$。最後回傳修改後的 aug_input_feat。

在計算模型的 Loss 的部分，我們需實做下列函式：

$$\text{forward}(self, aug\_adj\_1, aug\_adj\_2, aug\_feat\_1, aug\_feat\_2, sparse)$$

forward()的輸入是兩個 Graph View 的 Adjacency Matrix 以及 Features Matrix，利用 MERIT 提供的 Online Network 以及 Target Network 計算出 Cross-view 以及 Cross-network Contrastive Loss 作為優化的損失函數輸出。這裡我們會對forward()的程式碼進行說明，完整程式碼在 **merit.py** 中。

在forward()中，我們先做 Online Network 的預測，對兩個 Graph View 的 Adjacency Matrix 以及 Features Matrix 進行 Encoding ($self.online\_encoder$)與 Predict ($self.online\_predictor$)。接著，

進行 Target Network 的計算，將兩個 Graph View 的 Adjacency Matrix 以及 Features Matrix 進行 Encoding($self.target\_encoder$)，這裡因為 Target Network 是使用 Moment Updating(詳細 updating 方式請參考原文的 3.2 節)不需要計算 Gradient，所以這裡我們可以將 Gradient 的計算關掉，提升執行速度，縮短訓練時間。

最後，參考 MERIT 論文的 Equation(4) ~ Equation(11)對 Online Network 的預測結果($online\_pred\_1, online\_pred\_2$)和 Target Network 的結果 ($target\_output\_1, target\_output\_2$) 計算 Contrastive Loss。因為這裡要計算的 Cross-view 以及 Cross-network Contrastive Loss 計算方式較複雜，所以我們將其拆分出來，使用函式$contrastive_loss_cross_view(pred_1, pred_2, target)$實作 Cross-view Contrastive Loss；函式$contrastive_loss_cross_network(pred_1, pred_2, target)$實作 Cross-network Contrastive Loss。

本次實驗的 Accuracy 為 0.824，執行時間為 4315 秒。

III. 對程式進行修改

我嘗試將 NFM 從「隨機刪除固定數量的 Node Features」改成「隨機刪除不固定數量的 Node Features」，aug_feature_dropout_2() 其輸入的參數和原 aug_feature_dropout() 一樣。aug_feature_dropout_2()給予每個 Node Feature 一個隨機**機率**(介於 0~1 之間)，刪除**機率**$< p$的 Node Features，因為**機率**是隨機生成的，所以每次刪除的 Node Features 數量會不一定一樣。此修改的實驗的 Accuracy 為 0.827，執行時間為 4168 秒。此修改的變化對實驗結果的影響並不大。

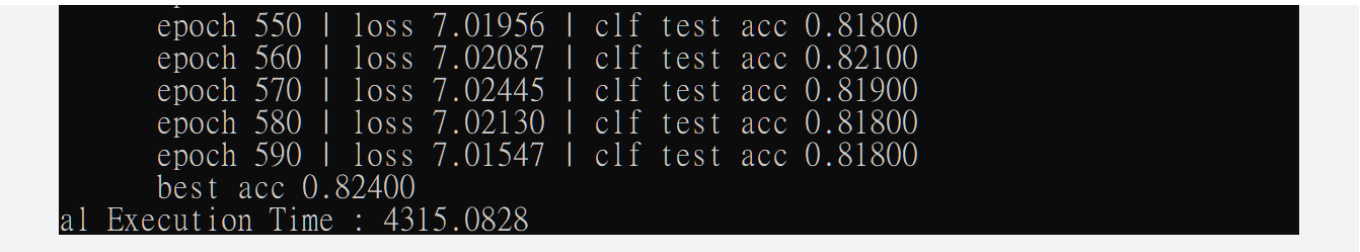
接著，我嘗試將 Edge Modification 改為只 drop $p/2$ edges。此修改的實驗的 Accuracy 為 0.825，執行時間為 2592 秒。此修改沒有改變 Accuracy，但大幅縮短了程式的執行時間。作者原本的 Edge Modification 方式是 drop $p/2$ edges 且 add $p/2$ edges，希望防止過度改變圖，同時可以保證 Graph Augmentations 具有足夠的複雜性，但從結果來看這並沒有讓模型的 Accuracy 上升，只增加了圖的複雜性且造成執行時間變長。

最後，結合 NFM 和 Edge Modification 的修改實驗的 Accuracy 為 0.826，執行時間為 2580 秒。

IV. 實驗結果

1. 未修改的程式執行時間為 4315 秒，Accuracy = 0.824。

程式執行結果截圖：



2. 修改 NFM 函式後程式執行時間為 4168 秒，Accuracy = 0.827。

程式執行結果截圖：

```
epoch 550 | loss 7.01846 | clf test acc 0.82300
epoch 560 | loss 7.01812 | clf test acc 0.81800
epoch 570 | loss 7.02432 | clf test acc 0.82100
epoch 580 | loss 7.01793 | clf test acc 0.82100
epoch 590 | loss 7.01559 | clf test acc 0.81900
best acc 0.82700
Total Execution Time : 4168.0011
```

3. 修改 Edge Modification 後程式執行時間為 2592 秒，Accuracy = 0.825。

程式執行結果截圖：

```
epoch 550 | loss 7.00459 | clf test acc 0.81500
epoch 560 | loss 7.00149 | clf test acc 0.81700
epoch 570 | loss 7.00476 | clf test acc 0.81800
epoch 580 | loss 7.00362 | clf test acc 0.81500
epoch 590 | loss 7.00355 | clf test acc 0.81500
best acc 0.82500
Total Execution Time : 2591.9847
```

4. 修改 NFM 函式和 Edge Modification 後程式執行時間為 2580 秒，Accuracy = 0.826。

程式執行結果截圖：

```
epoch 550 | loss 7.00032 | clf test acc 0.81900
epoch 560 | loss 7.00112 | clf test acc 0.82000
epoch 570 | loss 7.00459 | clf test acc 0.82100
epoch 580 | loss 7.00023 | clf test acc 0.82200
epoch 590 | loss 7.00143 | clf test acc 0.81800
best acc 0.82600
Total Execution Time : 2580.5713
```

V. 學習重點

在 Graph Augmentations 的處理上，很多的方法都有將隨機性加入其中，雖然依照權重或其他指標將重要的邊、點留下，讓模型更專注於這些資訊上也是一個方法，但因為是 self-supervised learning，我們期望以較少的訓練資料得到不錯的結果，所以在 Graph Augmentations 上加入隨機性以增加資料多樣性。而之所以採 drop edges 和 add edges 是因為我們希望能同時保留生成的資料與原資料的一致性，也就是讓圖形維持固定數量的邊，但這同時也會增加計算的時間，須謹慎選用。

VI. 參考資料

1. 上課講義
2. HW5 作業說明
3. MERIT https://github.com/GRAND-Lab/MERIT
4. 圖神經網絡自監督學習之 MERIT https://zhuanlan.zhihu.com/p/484136556