Final Project (SNA第四組)
r01922164 李揚　　r03922032王珮恂　　r02922062 郭冠呈

我覺得報告還是以閱讀性以及清處為重點，我們把細節放在我們的附帶檔案。
附帶檔案:
歷史回顧(我們的hw2報告): hw2.pdf
Features總整理:　features.pdf

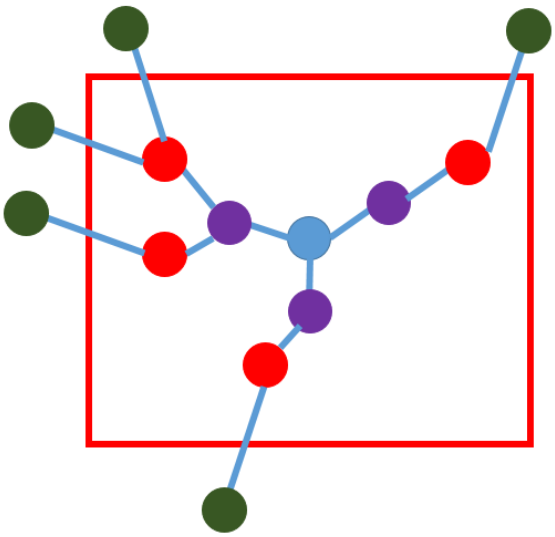
目標:
做一個有效的link prediction在hw2上

資料集:
助教提供的hw2的資料。

討論與結果方別放在不同方法的章節裡。

Result:

Method	F1	MAP
Our current result	**0.140762**	**0.141015**
No. 1 result in hw2	0.126635	0.135865
our result in hw2	0.124259	0.121458



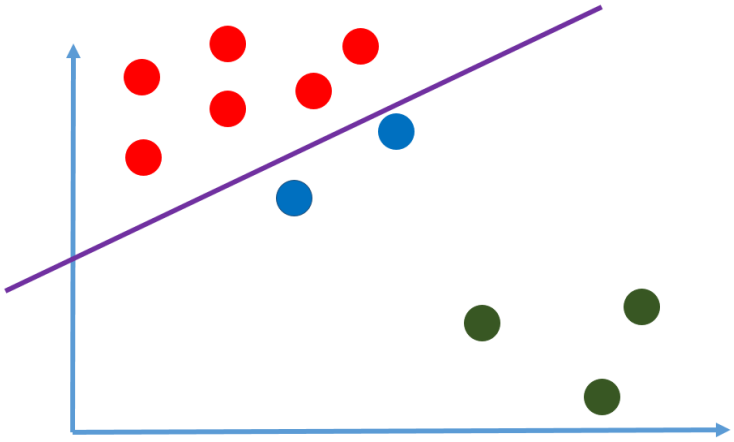
Learning Project

實驗架構:
之前我們的實驗架構其實已經有了learning的部分，但我們改成了之前第18組所提到的two-level sampling的方法。

我用左邊的圖去詮釋two-level sampling的細節。**藍色的節點**是我的seeds，**紫色的節點**是seeds的鄰居，**紅色的節點**是對於我的seeds的第二層鄰居，綠色的點則是第二層鄰居以外的點。在two-level sampling裡面，綠色的點是不會被考慮進來的，也就是被我捨棄的點。

然後再training的時候，**紫色的點**是我的training裡的positive，**紅色的點**是我的training裡的negative。

在testing的時候，我只猜**紅色的點**。
分析與討論"2-level sampling的物理意義":
2-level sampling表面上看起來可能有點奇怪，因為我是把training裡的negative在testing裡當candidates來猜。因此，一個夠複雜的model，會對我所有的candidates都當作不是link。[注意，這不是overfit!!!這是另外一個問題，問題來自於我把potential是positive的當作negative。雖然實際上原本的方法也有這樣的問題，但這方法的疑問可能更大，因為"所有的candidate"都是"negative"]
但是，我可以藉由model去學，到底哪些點"最像是training裡的positive link"。
ex:假設我現在是在2維空間下做logistic regression。如下圖所示:



假設我現在用logistic regression學到一條**紫色線**，他成功的把positive的**紅色點**分隔出來。成功的把其他的點(negative)的點分隔到另一邊。

即使如此，我知道**藍色的點**比較可能成為positive。(這部分可以很簡易的算logistic regresion的score較為接近0的得知)。綠色的點不太可能成為positive。所以我仍舊可以選出最有可能成為link的點。

但是，這樣有什麼好處呢，以下我就對這種方法的優缺做基本的分析。

優點:
2-level sampling的negative是比較有辦法增加鑑別度的。依照我之前的方法，sample(negative隨機從不是link裡面的挑出來)進來的點跟training之間的差別可能用一個簡單的common neighbor即可分辨，如此一來並沒有辦法讓我學到其他的東西。
實際上，由實驗證明，2-level-sampling確實是比較好的。以下比較都是用"同樣的"features。

	F1	MAP
original method	0.059909	0.078108
2-level-sampling	0.130268	0.129754

缺點:
這缺點還蠻明顯的，2-level sampling直接捨棄掉了**綠色的點**，而且**綠色的點**也不算少。尤其是在異性link的方面，綠色點最後成為link的機率並不低，對於異性來說大約有45.5%的link來自於**綠色的點**

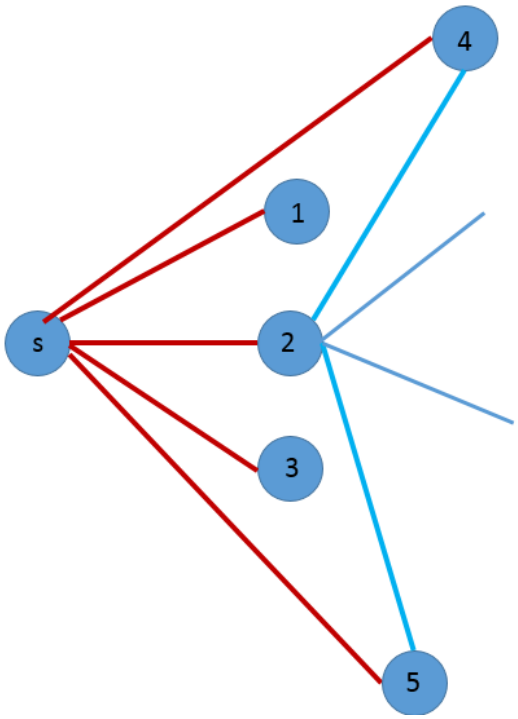
	link是common neighbor的比例(2-level sampling會報考慮成candidates的比例)
同性	80.4%
異性	54.5%

這也是我們接下來想要做Random walk的原因之一。

Feature Engineering

我們這邊不會一個一個講所有的features，但我會再另外一份文件(Features總整理)說明所有的features，我這邊列出，根據我們實驗，最有用的features，並做解釋。
Effective Neighbors:
在common neighbor這個index 裡面，每一個neighbor對於score的重要性是一模一樣的，Adamic index和resource allocation index，release了每一個neighbor都是一樣的assumption，但是是根基於"經驗"，而非"統計"，所以並沒有辦法客製化到各個問題上。
於是，我們提出了一個index:"Effective Neighbors index"以下簡稱en index，是使用統計，而非經驗，所創作出來的index。
概念如下:
對於每一個common neighbor，我們直接去統計，這個neighbor的link有多少機率是我現在seeds的link。
舉例來說:
假設現在s點是我的目標seeds然後我要推薦candidates給他，於是乎我要計算每個點的重要性。以2號點為例，我就去算2號點的所有links當中，有多少links是會連到我的目標

seed s。如此一來，我可以統計出不同common neighbor的重要性。詳細公式如下圖所示



$$EN(s,2) = \frac{|common\ neighbor(s,2)|}{|neighbor(2)|} = \frac{2}{5}$$

$$En(s,n) = \sum_{i \in common\ neighbors(s,n)} EN(s,i)$$

除此之外，我們可以發現en index並不是一個對稱的index。也就是說，2對於s的en index，和s對於2的en index是完全不一樣的值。通常來講，我們一定是希望兩邊都有好的effective neighbors才會想做推薦，而非只有單邊好就做推薦，於是我們把這種非對稱的index做相乘的動作，在此叫做reciprocal effective neighbor index，簡稱r_en index。

注意，我們這邊所提出的方法雖然用到統計，但可以發現，**計算複雜度沒有因此變得複雜**，實際上會跟Adamic一樣，cost是對於candidates算common neighbor還有每一個common neighbor算degree。這複雜度是一模一樣的，也就是說，這是一個很快的演算法。

以下，我把所有的non learning index的準度做個比較。

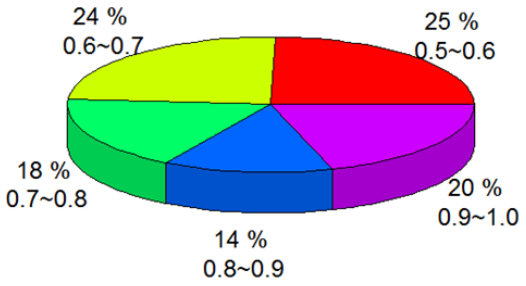
Index	**F1**	**MAP**
Reciprocal Effective Neighbors	0.078714	0.11739
Effective Neighbors	0.079374	0.115094

Adamic-adar	0.076740	0.113445
Common neighbor	0.072296	0.103799

可以發現，我們提出的index比大家公認最好的adamic的index還要來的好。並且遠比最基本的common neighbor還來的好。而且我們是提出一與adamic相同複雜度的方法。

Sex proportion features:

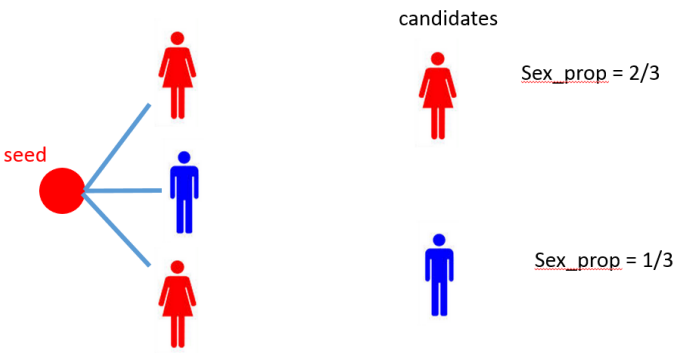
我們在data裡面觀察，發現，大部分的人都偏好與單一的性別互動。(Hw2裡我們也提到大部分的人喜歡跟異性互動)。我們為此畫了圓餅圖，如下圖所示。



有20%的人他的較為偏好的性別占了超過0.9(90%)，有34%的人他的較為偏好的性別佔了超過0.8(80%)。

於是乎，我們設計了sex proportion的features，簡稱sex_prop。

方法如下:



如左圖所示，對於我的seed我對於他之前所建的link的性別比例去做統計，candidates的sex_prop的值就是這個seed過去認識的candidates的性別的比例。

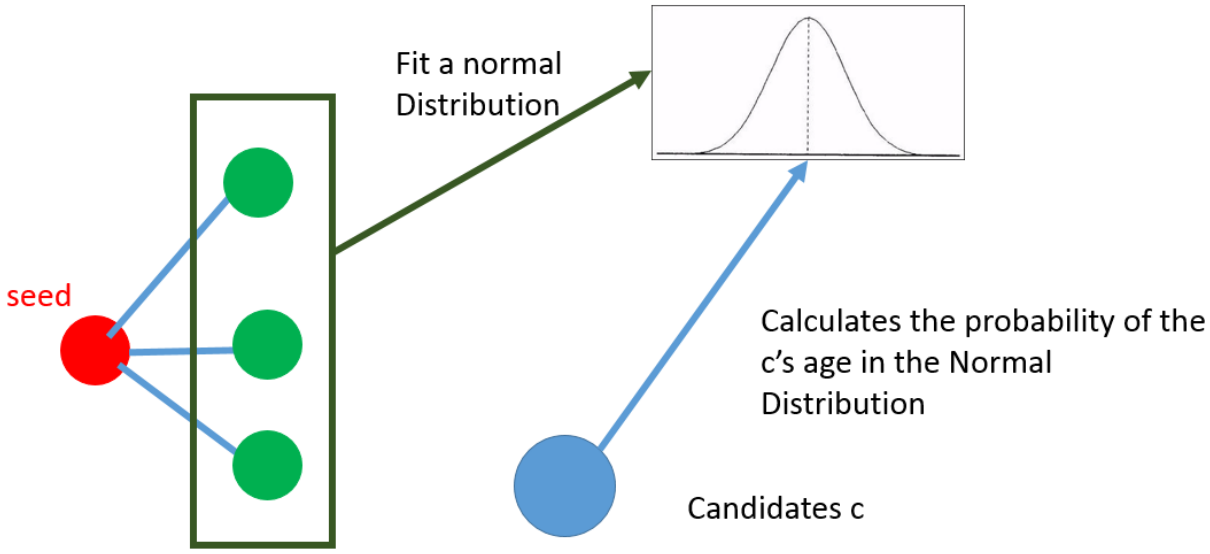
age normal distribution features:
大部分的人喜歡跟自己年齡相近的人做朋友。但是，我們又對這件事情做了更深入的分析。

1. 即使大部分的人喜歡跟自己年齡相近的人做朋友，但是可分為，比自己大一點以及比自己小一點兩種情形。例如:根據我們的分析，男生會偏好跟自己小一點的人做朋友，女生則會偏好比自己大一點的人做朋友

Average age difference with the link:
-0.374

Average age difference with the link:
0.264

2. 第二點觀察是，有些人可以做朋友的年齡range很大，有些則很小。如果單純用年齡差距，也無法model這件事情。
因此，我們提出了age's normal distribution features，簡稱age_norm。方法如下:



我們根據**seed**點過去**建立的link**，去fit一個normal distribution，對每一個candidate去算他有多少機率出現在這個normal distribution。(可以直接利用pdf，我們是切100個bins)。如此一來，對於range比較大的seeds，即使年齡差距較大，也較不會接受太大的處罰，且這樣真正學到seed最喜歡的年齡分布，而非只有差距。

degree features:

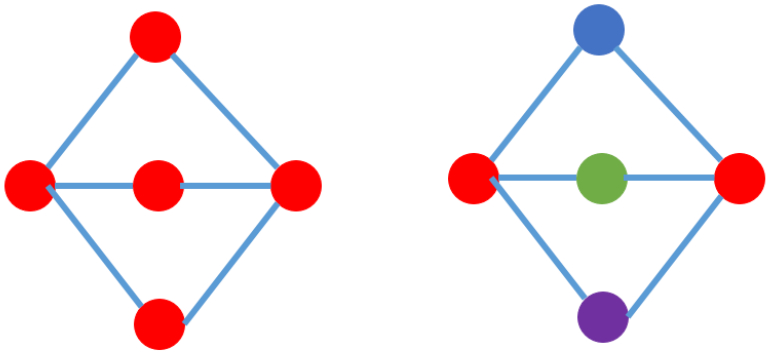
我們認為degree也是很重要的features。所以把degree的difference，還有針對兩邊的degree。我們都把它做成features
全部features的解釋可以看"Features總整理"檔案。
加入這些features之後，我們的performance又有了大幅的進步。

	F1	MAP
Original RF	0.130268	0.129754
After adding these features	0.139583	0.140986

Use Community to Further improve our results:
我們大致有三種方法去利用分出來的community。

1. 直接分群去做訓練，照各種不同的cluster下去分開訓練
2. 直接把各個cluster當作binary features丟進去學
3. 假設同個cluster的common neighbor 較為重要。並且設計以下的features。

其中第三個方法比較需要解釋，我們是參考一篇發自www companion 2012的論文。
"Using community information to improve the precision of link prediction methods"
概念如下:
同樣是屬於同一個cluster的兩個nodes。他們如果share越多同一個cluster的nodes，我應該要給他更高的分數。
Ex:
兩個很喜歡打球的人，如果共同認識很多一樣喜歡打球的人，可能會更有可能成為朋友。
反之，兩個很喜歡打球的人，共同認識喜歡攝影、喜歡畫畫的朋友，比較可能是巧合。
示意圖如下:



這張圖裡，不同顏色代表不同的cluster
同樣是shared三個common neighbors，但左邊的圖都是share同一個cluster的，右邊則不是。
因此，我會給左邊的圖的情形更高的分數。

於是，這邊paper提出了依照上面敘述的現象的修改版本的common neighbor和resource allocation index。

$$CN1(a,b) = CN(a,b) + \sum_{i \in \Gamma(a,b)} |C(i) \cap C(a) \cap C(b)|.$$

$$RA1(a,b) = \sum_{i \in \Gamma(a,b)} \frac{1 + |C(i) \cap C(a) \cap C(b)|}{d(i)}.$$

第一個修改後common neighbor的式子，第二個是修改後resource allocation index的式子。在shared common neighbor與link兩邊的nodes的cluster相同時，都會得到一點點的加分。

不同的cluster方法:

如果仔細觀察我們的"Features總整理"可以發現，上述的情形，其實已經被我們的features model進來了。我們已經把兩邊的性別、年齡，放進我們的features當中，所以在Random Forest的架構下，如果這樣cluster的分別重要的話，它已經會自動把我的資料分成一個一個cluster。
因此，我們要用上面敘述三個方法的cluster，我們是based on “結構性(path,structure)”的分群，而非"特徵性(features)"的分群。
而我們使用的方法大致可分為兩個方面

1. Community Detection
2. Role Detection

greedy method to optimize modularity。

Method	F1	MAP
1	0.135347	0.136536
2	0.135347	0.136536
3	0.140186	0.141015

RolX:
我們在報告時有提到，我們想要藉由Matrix Factorization來做Role Detection，不過我們發現在2012年的時候，這種想法已經被提出來過了，發表在KDD。論文的title是:

"Rolx: structural role extraction & mining in large graphs"。

這篇論文一樣是使用Matrix Factorization的概念，只不過他比我們在報告時提出的想法更為精細，我的想法(單邊Matrix Factorization)必須要先做community detection而且無法加入features。
而這篇論文提出的想法卻可以把role detection一次到位，而且可以加入額外的features。
Rolx的演算法如下:

1. 對於每一個nodes生一個feature matrix V。
2. 然後使用non-negative matrix factorization去裂解這個V。式子如下:

$$\text{argmin}_{G,F} \|V - GF\|_{fro}, \text{s.t. } G \geq 0,\ F \geq 0$$

假設有f個features，N個使用者。那V就會是一個N*f的陣列。假設我們現在要分成r個roles，G是一個n*r的陣列，代表著各個role會屬於各個角色的機會。F是一個r*f的陣列，代表各個角色對於每一個features的貢獻度。

Result:
我們依照Rolx的方法分成十個roles，然後用上述的第2,3個方法去做實驗，只可惜沒有進步。但是對之後的logistic regression，有些微的幫助。

Method	F1	MAP
2	0.139449	0.140690
3	0.139012	0.140763

Random Walk with simple features:
根據我們survey的結果，Random Walk with restart(Personal Pagerank)，是眾多論文都同意的一個好方法，一方面他快，而且又不需要煩惱negative sampling的問題(甚至也不用去選candidates)。並且可以學到一些結構性的事情。
之前有一組有用Random walk with restart，做到MAP 0.116的成果。但加上一些features和確實地把此方法跑到收斂，可以再提升這個方法最後的表現。

Random walk with restart (avoiding meet the both column_49==0):
我們把c49 皆為0的去掉，並且跑到第100萬回合，就可以發現，random walk可以達到不錯的效果。Random walk是我們目前所有實驗除了logistic regression，可以在一小內跑完，而且用最少記憶體的演算法。而且他仍舊可以達到0.123990的MAP。我覺得是一個蠻不錯的方法。

F1	MAP
0.125438	0.123990

Random walk with restart with sex_prop features:

F1	MAP
0.125327	0.123969

Logistic Regression:
接下來，我們嘗試Logistic Regression。
我們要嘗試著使用Linear Model去達到好的表現，另外，Logistic Regression是一個的optimization是使用SGD like的演算法，可以較短時間內，得到我們想要的結果。而且，使用Logistic Regression一樣可以製造出比之前hw2第一名更好的prediction，卻節省了可觀的時間。(我們這邊的Logistic Regression在五分鐘內即可跑完。(training+prediction))

feature scaling問題:
我們發現feature scaling會對logistic regression的影響非常大。
不scaling很理所當然地得到很糟糕的結果(MAP/F1不到0.05)，因為有些features數值的range相差很大
如果使用傳統的Standard scaling會達到一個很差的結果(F1/MAP都在0.1以下)。我們猜測，是因為有些features(ex: common neighbors, adamic...等等)，如果是負的會嚴重影響到它的意義。而且因為我們是用2_level的sample，所以即使是有link，這些features變成負的機率並不低。
但是改成min-max scaling就可以達到相對較好的結果(F1/MAP都在0.12以上)。

Features選擇問題
我們這邊Logistic regression目前只有使用比較不會出問題的數值性features(ex: effective neighbors,katz … etc)，而沒有使用分類式的

features(ex:sex_s,sex_n,degree_s,degree_n,...)。因為分類式的features對於logistic regression意義不大，而且容易造成overfit。

Best Global Learner:

F1	MAP
0.128873	0.129843

另外，Logistic Regression使我們有機會去實作10000個model，也就是說，我們可以針對每一個人去特別train一個model。因為Logistic Regression是SGD like的演算法，所以，我們不會花太多時間在training上(只會跟資料的比數成線性關係)，實際上，即使是要train一萬個model，training+ prediction的時間仍舊不會超過五分鐘。

Best individual learner:

F1	MAP
0.133699	0.132157

即使沒有比我們之前RF的結果來的好，不過在F1方面已經贏過hw2的第一名。
Logistic Regression在一個很短的時間內，就可以達到了一個還算是不錯的結果。

Individual Learner with Rolx

F1	MAP
0.133168	0.132535

總結:
在第一階段(final presentation之前)，我們主要專注在提升最終可以得到的表現，結果也得到非常大幅的進步(比之前第一名進步12%)。
第二階段我們專注於分群和效率，在分群方面雖然突破不大，但確實有為最後的結果做一些進步。
我們專注於效率較佳的Random Walk並且達到了比之前第一名好的結果。
我們也專注於研究效率更好的Logistic Regression，在不到五分鐘，就可以得到比hw2第一名更好的結果。
我們的方法總結:

Method	F1	MAP
Ensemble	0.141003	0.141015
RF	0.139583	0.140986
RF+ community Detection	0.140186	0.141015
Random Walk	0.125438	0.123990
Logistic regression(global)	0.128873	0.129843
Logistic Regression(individual)	0.133699	0.132157
hw2 No.1 result	0.126635	0.135864

其中Random Walk，Logistic Regression(global/inidividual)都遠比其他演算法來的快。