Third Laboratory

Community Detection

Complex Networks

Universitat Rovira i Virgili

Master in Artificial Inteligence

2^{nd} Semester

Author:
Jorge Rodriguez Molinuevo

June 4, 2017

1 Community Detection

1.1 Used Algorithms

For this block, several community detection algorithms has been used. This are the algorithms used:

- Random walks based approach called **Walktrap**. The general idea is that if you perform random walks on the graph, then the walks are more likely to stay within the same community because there are only a few edges that lead outside a given community. Walktrap runs short random walks of 3-4-5 steps (depending on one of its parameters) and uses the results of these random walks to merge separate communities in a bottom-up manner. Them the modularity score to select where to cut the dendrogram.

- **Label Propagation**, is a simple approach in which every node is assigned one of k labels. The method then proceeds iteratively and re-assigns labels to nodes in a way that each node takes the most frequent label of its neighbours in a synchronous manner. The method stops when the label of each node is one of the most frequent labels in its neighbourhood. It is very fast but yields different results based on the initial configuration.

- **Louvain Method** for community detection, which is based in a bottom-up optimization of the modularity. The Louvain Method has two phases that are repeated iteratively. First, each node in the network is assigned to its own community. Then for each node i, the change in modularity is calculated for removing i from its own community and moving it into the community of each neighbour j of i.

  In the second phase of the algorithm, it groups all of the nodes in the same community and builds a new network where nodes are the communities from the previous phase. Any links between nodes of the same community are now represented by self loops on the new community node and links from multiple nodes in the same community to a node in a different community are represented by weighted edges between communities. Once the new network is created, the second phase has ended and the first phase can be re-applied to the new network.

- Community detection by **Edge Betweenness** is a hierarchical decomposition process where edges are removed in the decreasing order of their edge betweenness scores. This is motivated by the fact that edges connecting different groups are more likely to be contained in multiple shortest paths simply because in many cases they are the only option to go from one group to another. This method yields good results but is very slow because of the computational complexity of edge betweenness calculations and because the betweenness scores have to be re-calculated after every edge removal.

All this algorithm are used in R where they are already implemented in the igraph package. This way the partitions are obtained and their modularity is calculated, also with R, them radatools is used to compare partitions, with the Compare_Partitions tool.

1.2 Implementation and deliverables

For this exercise three scripts has been done, an R script for modularity and partition calculations, called "iGraphCommunity.R", the optimal parameters are automatically selected, however for the walktrap algorithm it is recommended to run it several times, since the the results is stochastic and can vary, them select the partition with the higher modularity.

To automate the comparison of the partitions a ".bat" partition has been created to call the executable "Compare_Partitions.exe" with the correct values, called "J_compare_partitions.bat". Also an auxiliary Python script has been used to created LaTeX tables from the obtained results.

The created partitions has been saved in a folder called "Partitions", the different algorithms used can be identified by the ending of the partition, each name correspond to a method in the following way:

- Label Propagation: *_label_prop.clu.*
- Random Walk: *_walktrap.clu.*
- Louvain Method: *_louvain.clu.*
- Edge Betweenness: *_betweenness.clu.*

The comparison between algorithms have been done throw Jaccard Index, Normalized Arithmetic Mutual Information (NAMI) and Normalized Variation of Information(NVI). This results can be found in Section 3.

The plot of the obtained partitions can be found in the folder named 'plots', every networks appears by each name plus the algorithm used for calculating communities.

2 Results and conclusions

When taking a look to the values of modularity at Table 1, first thing we notice is the value for '*star.net*' this is because of the kind of network, which is just a node with multiple edges to other nodes not connected between them, for this special case the results are 0. Also notice that in general the results are quite similar, apart from some exceptions, such as '$cat\_cortex_sim.net$', where the results for Betweenness based algorithms is less successful to optimize modularity.

So, all for algorithms are able to optimize correctly modularity, yet maximizing the modularity doesn't necessary mean that the real communities are detected. For toy and model networks the communities detected are nearly the same as the ones given, as can be seen in Section 3. However on real problems the modularity is not the truth necessarily, in Tables 15, 14, 13 and 12 can be notice how the obtained partitions are no so close to real ones, neither of the algorithms is able get the truth, however for some the values are high: label propagation is able to get really good results for Zachary's network and football; and Louvain's algorithm over-performed the rest in cortex network.

It is important to point out the computational cost of this algorithms, which is shown in Table 2 measured in needed time. The algorithms Louvain, Label Propagation and Random Walk needed very similar times, and since the times were measured only 2 times and them averaged, it cannot be said if one is faster then the rest, nevertheless, the times fo Edge Betweenness are so high that it is not necessary to do it, it was needed 3.5 hours for airport network.

To sum up, all the algorithm yielded good results, some better than others in some cases, however based in the time needed the Edge Betweenness algorithm under-performed, taking all into account.

3 Tables

Network Name	Louvain Method	Label Propagation	Random Walk	Edge Betweenness
256_4_4_2_15_18_p.net	0.7818	0.7818	0.7818	0.7818
256_4_4_4_13_18_p.net	0.6974	0.6634	0.6974	0.6974
rb125.net	0.6169	0.5917	0.6121	0.6107
airports_UW.net	0.6460	0.4405	0.6051	0.5913
cat_cortex_sim.net	0.3719	0.3709	0.3709	0.1590
dolphins.net	0.5185	0.4308	0.4888	0.5194
football.net	0.6046	0.5807	0.6029	0.5996
zachary_unwh.net	0.4188	0.3991	0.3532	0.4013
20x2+5x2.net	0.5426	0.5416	0.5416	0.5426
graph3+1+3.net	0.3672	0.3672	0.3672	0.3672
graph4+4.net	0.4408	0.4429	0.4429	0.4429
star.net	0	0	0	0

Table 1: Modularities values for each network and algorithm.

Network Name	Louvain Method	Label Propagation	Random Walk	Edge Betweenness
256_4_4_2_15_18_p.net	0.0303	0.003	0.016	26.2822
256_4_4_4_13_18_p.net	0.0045	0.002	0.0186	37.8807
rb125.net	0.0025	0.002	0.0035	0.5392
airports_UW.net	0.0371	0.0441	1.2471	13250.88
cat_cortex_sim.net	0.002	0.002	0.007	1.8993
dolphins.net	0.001	0.001	0.0015	0.0657
football.net	0.002	0.001	0.0035	1.1867
zachary_unwh.net	0.002	0.001	0.0015	0.1429
20x2+5x2.net	0.002	0.002	0.0015	0.0862
graph3+1+3.net	0.002	0.001	0.001	0.0025
graph4+4.net	0.001	0.001	0.001	0.002
star.net	0.001	0.0015	0.001	0.0015

Table 2: Execution time for each algorithm divided by networks.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
$20x2 + 5x2.clu$	$20x2 + 5x2$_label_prop.clu	1.0000	1.0000	0.0000
$20x2 + 5x2.clu$	$20x2 + 5x2$_louvain.clu	0.9412	0.9401	0.0354
$20x2 + 5x2.clu$	$20x2 + 5x2$_walktrap.clu	1.0000	1.0000	0.0000
$20x2 + 5x2.clu$	$20x2 + 5x2$_betweenness.clu	0.9412	0.9401	0.0354

Table 3: Partitions comparissons in network $20x2 + 5x2$.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
$graph3 + 1 + 3.clu$	$graph3 + 1 + 3$_label_prop.clu	0.5000	0.5295	0.3303
$graph3 + 1 + 3.clu$	$graph3 + 1 + 3$_louvain.clu	1.0000	1.0000	0.0000
$graph3 + 1 + 3.clu$	$graph3 + 1 + 3$_walktrap.clu	1.0000	1.0000	0.0000
$graph3 + 1 + 3.clu$	$graph3 + 1 + 3$_betweenness.clu	0.5000	0.5295	0.3303

Table 4: Partitions comparissons in network $graph3 + 1 + 3$.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
$graph4 + 4.clu$	$graph4 + 4$_label_prop.clu	1.0000	1.0000	0.0000
$graph4 + 4.clu$	$graph4 + 4$_louvain.clu	0.4615	0.6002	0.3538
$graph4 + 4.clu$	$graph4 + 4$_walktrap.clu	1.0000	1.0000	0.0000
$graph4 + 4.clu$	$graph4 + 4$_betweenness.clu	1.0000	1.0000	0.0000

Table 5: Partitions comparissons in network $graph4 + 4$.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
star.clu	star_label_prop.clu	1.0000	1.0000	0.0000
star.clu	star_louvain.clu	1.0000	1.0000	0.0000
star.clu	star_walktrap.clu	0.0000	0.0000	1.0000
star.clu	star_betweenness.clu	1.0000	1.0000	0.0000

Table 6: Partitions comparissons in network *star*.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
256_4_4_2_15_18_p.clu	256_4_4_2_15_18_p_label_prop.clu	1.0000	1.0000	0.0000
256_4_4_2_15_18_p.clu	256_4_4_2_15_18_p_louvain.clu	1.0000	1.0000	0.0000
256_4_4_2_15_18_p.clu	256_4_4_2_15_18_p_walktrap.clu	1.0000	1.0000	0.0000
256_4_4_2_15_18_p.clu	256_4_4_2_15_18_p_betweenness.clu	1.0000	1.0000	0.0000

Table 7: Partitions comparissons in network 256_4_4_2_15_18_p.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
256_4_4_4_13_18_p.clu	256_4_4_4_13_18_p_label_prop.clu	0.2698	0.7184	0.2344
256_4_4_4_13_18_p.clu	256_4_4_4_13_18_p_louvain.clu	0.9048	0.9529	0.0254
256_4_4_4_13_18_p.clu	256_4_4_4_13_18_p_walktrap.clu	0.9048	0.9529	0.0254
256_4_4_4_13_18_p.clu	256_4_4_4_13_18_p_betweenness.clu	0.9048	0.9529	0.0254

Table 8: Partitions comparissons in network 256_4_4_4_13_18_p.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
$rb125-1.clu$	*rb125_label_prop.clu*	0.2577	0.6703	0.3101
$rb125-1.clu$	*rb125_louvain.clu*	0.6167	0.8554	0.1223
$rb125-1.clu$	*rb125_walktrap.clu*	0.6203	0.8340	0.1400
$rb125-1.clu$	*rb125_betweenness.clu*	0.5711	0.8175	0.1587

Table 9: Partitions comparissons in network $rb125-1$.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
$rb125-2.clu$	*rb125_label_prop.clu*	0.3489	0.9027	0.1235
$rb125-2.clu$	*rb125_louvain.clu*	0.2703	0.8267	0.2111
$rb125-2.clu$	*rb125_walktrap.clu*	0.2523	0.8225	0.2157
$rb125-2.clu$	*rb125_betweenness.clu*	0.2733	0.8394	0.1970

Table 10: Partitions comparissons in network $rb125-2$

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
$rb125-3.clu$	*rb125_label_prop.clu*	0.3509	0.9067	0.1194
$rb125-3.clu$	*rb125_louvain.clu*	0.2659	0.8241	0.2152
$rb125-3.clu$	*rb125_walktrap.clu*	0.2533	0.8275	0.2115
$rb125-3.clu$	*rb125_betweenness.clu*	0.2746	0.8441	0.1929

Table 11: Partitions comparissons in network $rb125-3$.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
cat_cortex_sim.clu	*cat_cortex_sim_label_prop.clu*	0.2572	0.0000	0.3341
cat_cortex_sim.clu	*cat_cortex_sim_louvain.clu*	0.6711	0.7853	0.1323
cat_cortex_sim.clu	*cat_cortex_sim_walktrap.clu*	0.5841	0.7108	0.1761
cat_cortex_sim.clu	*cat_cortex_sim_betweenness.clu*	0.3139	0.5343	0.4622

Table 12: Partitions comparissons in network *cat_cortex_sim*.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
$dolphins-real.clu$	*dolphins_label_prop.clu*	0.4662	0.5516	0.2154
$dolphins-real.clu$	*dolphins_louvain.clu*	0.3631	0.5636	0.2581
$dolphins-real.clu$	*dolphins_walktrap.clu*	0.4796	0.5652	0.2045
$dolphins-real.clu$	*dolphins_betweenness.clu*	0.4370	0.5997	0.2199

Table 13: Partitions comparissons in network $dolphins-real$.

Real	*Obtained*	*JaccardIndex*	*NAMI*	*NVI*
$football-conferences.clu$	*football_label_prop.clu*	0.7583	0.9102	0.0914
$football-conferences.clu$	*football_louvain.clu*	0.7004	0.8909	0.1095
$football-conferences.clu$	*football_walktrap.clu*	0.7115	0.8879	0.1127
$football-conferences.clu$	*football_betweenness.clu*	0.6639	0.8797	0.1202

Table 14: Partitions comparissons in network $football-conferences$.

$Real$	$Obtained$	$JaccardIndex$	$NAMI$	NVI
$zachary\_unwh - real.clu$	$zachary\_unwh\_label\_prop.clu$	0.8858	0.8372	0.0639
$zachary\_unwh - real.clu$	$zachary\_unwh\_louvain.clu$	0.4754	0.6176	0.2359
$zachary\_unwh - real.clu$	$zachary\_unwh\_walktrap.clu$	0.3498	0.5467	0.3169
$zachary\_unwh - real.clu$	$zachary\_unwh\_betweenness.clu$	0.4712	0.6177	0.2515

Table 15: Partitions comparissons in network zachary.