Project 1

Alignment-free network classification and comparison based on global features

Jieao Zhu, ID: 110168391 Shihcheng Tsai, ID: 110161848 Weiliang Xing, ID: 108211104

**Abstraction:**

There exist great challenges to measure similarity between two graphs. One way to do that is to compare the graphs node to node with alignment methods [1]. The alignment-free methods, in comparison, may have better performance. In this project, we proposed a method to find similarity of different graphs. Training sets of graphs are generated from PAM, ER, GEO, and DDM graph families and their global features are extracted and feature vectors for each graph are built. We then use various classifier for training and testing. Results show that 1) features based on nodes and edges with space normalization in SVM classifier could achieve good performance of classification. We also tested with ??? with ??? performance. Also, ???local features are used with results ???.We could classify testing graph with accuracy as high as ???. For similarity measurement about two graphs, we achieved ???. Experiments shows good performance and effectiveness for our methods.

**Introduction:**

Graphs and relative researches draw a log of attention in recent years, with its importance in data representation and network analysis in computational biology [2]. In many applications network comparison is needed, with a key challenge to properly measure similarity between two networks. For example, network similarity need to be compared among different species [3]. However, the network comparison is naively intractable with lots of heuristic methods developed currently [4][5].

There are mainly two categories in network comparison. One is alignment-based comparison and the other is alignment-free comparison. Alignment-based comparison tries to find a mapping between two graphs, with normal methods for node-to-node comparison. Applications include protein structure identification [6], evolutionary conserved region of biological networks among species [7], etc. Alignment-free comparison, however, develops comparison methods based on global signatures of the graphs, namely, compare and discover overall topological similarity without evaluating detailed structural information [5]. Compared to alignment-based comparison methods, alignment-free comparison methods are normally computationally less expensive due to lack of the procedure for graph mapping [5]. However, quality control is needed to define a proper score representing overall similarity between two networks.

The methods in alignment-free comparison are normally divided into two basic categories: 1) comparison using global network properties; and 2) comparison using local network properties. Global network properties are properties that using overall topological properties to represent the graph and differentiate different graphs; while local properties extract more detailed information. Currently the most famous trends are towards investigating local properties based on graphlet, including methods like Graphlet Correlation Distance (GCD) [8] and NetDis [9], giving the reason that global properties-based methods usually capture limited information of networks [5][8]. We doubt that in this project. With proper extraction of proper features and property method design, there maybe potential good performance for methods based on global properties.

To measure similarity between graphs and compare them, we should first properly classify graphs into different families they originally belong to. Actually, among various techniques, developing computational methods for graph classification to identify the basic property of graphs is becoming a focuses theme. [10] The main challenge for graph classification is how to convert graphs into numeric features with proper classification methods [11].  Thus the graph comparison problem could be converted to be stated as follows: Given a groups of graphs that belong to different families F1, F2, … Fk, extract proper features into vector for machine learning that could predict or classify future test graphs into correct family.

In this project, we will investigate to build feature vectors with series of topological features for classification with SVM and Gaussian Naive Bayes classifier. There are two categories of features we use: local and global. For local features, the advantages include: 1) It is fast; we could finish feature extraction in limited time; 2) It is efficient, for larger graph, we don’t need to search over all graphs and we can explore part of graph; 3) we could use kernel methods to find the similar between two graphs; 4) we could focus on specific patterns to compare between two graphs with many features like shortest path, cycle patterns, and sub-graph, etc. Another features belong to global features. Using global features, we have advantages to obtain a set of signatures to represent a graph; It is also scalable compared to state-of-the-art graph kernel methods. In this project, we will make use of advantages of global features to efficiently classify graphs and compare the similarity. We will utilize features on classification and eventually choose the features to measure the similarity of two graphs as efficient as possible. Various experiments are implemented using normalization techniques and different classifier kernels. The performance is measured based on existing testing methodology [5].

**Goals:**

There are several goals in this project to incrementally sharpen our method to achieve better performance:

1)    Properly choose features with proper classifier kernels for classification;

2)    Measure performance of the method;

**Dataset**

In this project, we produce graphs from four families: PAM, ER, GEO and DDM. Compared to seven families mentioned in project proposal, we use only four families because there are only four families implemented in proposal testing framework. Actually, we are trying to implement more robust method that no matter how many families are provided, we could make a meaningful result. Graphs are generated using networkx in Python. Networkx is a solid Python package for graph analysis and graph generation. We will also use its built-in functions for feature generation. Totally 400 graphs with certain nodes number randomly chosen from 1000 to 2000 with 100 graphs in each family. Also density is randomly chosen for more practical condition. Among them half of them are used for training and rest for test. Below is part of graphs with basic parameter from graph generations:

|  |  |  |  |
| --- | --- | --- | --- |
| Family | Nodes | Edges | Density |
| PAM | 1760 | 8775 | 0.0056 |
| PAM | 1056 | 5255 | 0.0094 |
| DDM | 1839 | 3952 | 0.0023 |
| DDM | 1511 | 3158 | 0.0046 |
| ER | 1380 | 9510 | 0.0099 |
| ER | 1188 | 7113 | 0.0099 |
| GEO | 1541 | 6272 | 0.0053 |
| GEO | 1177 | 3608 | 0.0052 |

Table 1 Part of graphs from dataset.

**Feature Selection**

For better understanding and modularization of experiments, we divided features into five groups to build feature vectors separately:

Group 1: Features based on shape:

F1: Transitivity: The fraction of all triangles in an undirected graph. A triangle is a sub-graph structure with two edges sharing one vertex; We use it to present all possible triangles in graphs;

F2: Cycle basis: The minimum collection of cycle. Because cycles in a graph could be composed from many other smaller cycles, we collect such smaller cycles as cycle basis;

F3: Square clustering coefficient: The probability that two neighbors of node v sharing the same neighbor different from v. It measures the fraction of all possible squares that exist in the graph.

Group 2: Features based on nodes and edges:

F4: Percentage of isolated nodes: an isolated node in a graph is a node with degree zero. The degree of a node is the number of its neighboring edges. We collect the rate of isolated nodes to the total number of nodes in the graph. This feature is meaningful because in certain family like DDM, there is a clear tendency that the graph is not always connected with certain isolated nodes existing;

F5: Percentage of end nodes: a node with degree of one is an end node. We collect the rate of end nodes to the total number of nodes in the graph;

F6: Average degree: the average number of the degree of all nodes in graphs.

F7: Average neighbor degree: the average degree of the neighborhood of each node;

F8: Density: The density for undirected graphs is

 = \frac{2m}{n(n-1)},

where https://networkx.github.io/documentation/latest/_images/math/b1f5ca5a538abe6036ed478902bb5a03ef05f0c2.png is the number of nodes and https://networkx.github.io/documentation/latest/_images/math/2a6f9b0b4119433a3eae475d163e0694af1ca408.png is the number of edges in https://networkx.github.io/documentation/latest/_images/math/a3b876a880363bc604f5b4e0f360b95ffc7f4738.png.

F9: Average page rank: The page rank is the ranking of a node in the graph based on the structure of the incoming links. We record average page rank of the graph.

F10: Giant connected ratio: Giant graph is a sub-graph that is connected and has maximum number of nodes. The ratio of giant graph is the ratio of the number of nodes in graph graph to total number of nodes in the graph.

Group 3: Feature based on path:

F11: Average path length: the reciprocal of the averaged total path length between node and every other node that is reachable [11]. We use average of such feature.

F12: Energy: Square sum of the eigenvalues of the adjacency matrix.

F13: Betweenness centrality of nodes: The total shortest-path between centrality for nodes. Betweenness centrality of a node v is the sum of the fraction of all-pairs shortest paths that pass through v [12].

F14: Betweenness centrality of edges: The total shortest-path between centrality for edges.

Group 4: Features based on Eigenvalues:

F15: Number of eigenvalues: The number of distinct eigenvalues of an adjacency matrix;

F16: Second largest eigenvalue: The value of the second largest eigenvalue of the adjacency

matrix.

F17: energy: squared sum of eigenvalues of the adjacency matrix.

Group 5: Combined Features:

We combine all above features into one feature vector for a total observation.

**Normalization**:

Before training the classifier, we need to normalize extracted features. Feature vector built from raw features could represent a specific graph, but may be meaningless to represent the signature of the family of that graph. Another reason is that the dataset has graphs generated with different nodes, edges, density; to minimize the effect of such variable, normalization is needed. Here we will use range normalization and z-normalization. The range-normalization value of each feature will be calculated as , where min and max is the minimum and maximum value for certain feature x. The z-normalization value of each feature will be calculated as , where is the mean and is standard deviation of the feature x [11]. After normalization, the accuracy of classification is expected to improve. After normalization, the feature vector of each graph is build, with length i as number of features in the vector. The feature vector will then be sent to classifiers. An example of feature vectors is show in below table:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Feature | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 |
| Raw | 0.018 | 7016 | 0.001 | 0 | 0 | 9.97 | 23.13 | 0.05 | 1.0 |
| Space- | 0.034 | 0.38 | 0.009 | 0 | 0 | 0.42 | 0.55 | 0.3 | 0.49 |
|  |  |  |  |  |  |  |  |  |  |
| Feature | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 |  |
| Raw | 0.0005 | 1.0 | 0.31 | 0.05 | 2.15 | 1.58 | 1.59 | 2281 |  |
| Space- | 0.38 | 0.51 | 0.59 | 0.34 | 0.31 | 0.48 | 0.50 | 0.79 |  |
|  |  |  |  |  |  |  |  |  |  |

Table 2. The example of a feature vector with 17 features combining all 4 groups mentioned below in a graph.

**Classifier**

We use mainly two classifiers to classify our data, SVM and GNB. SVM, or Support vector machine, uses supervised learning models with associated learning algorithms, which is commonly used for regression analysis and classification [13]. SVM can compare with k-dimension features and find the similarities fast because it uses Kernel function that applies inner-product to calculate the similarity. We use two Kernel operations, “linear” and “RBF”, to compare with two strings in SVM kernel. GNB, or Gaussian Naive Bayes, is a classic but efficient algorithm for classification. Here classifiers are implemented by using sklearn package which is based on libsvm [14] for all of the kernel. Another reason we use GNB is because we extract global features from every graph family, and the value of each feature would not change dramatically.

**Results**

Experiment 1:

In this experiment, nodes are randomized for each group:

Below table is Accuracy for each family:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | success | fail | accuracy |
|  | G1+space | 154 | 46 | 0.77 |
|  | G2+space | 198 | 2 | 0.99 |
| linear | G3+space | 168 | 32 | 0.84 |
|  | G4+space | 116 | 84 | 0.58 |
|  | G5+space | 200 | 0 | 1 |
|  | G1+z | 124 | 76 | 0.62 |
|  | G2+z | 106 | 4 | 0.53 |
|  | G3+z | 124 | 76 | 0.62 |
|  | G4+z | 106 | 94 | 0.53 |
|  | G5+z | 162 | 38 | 0.81 |
|  | G1+space | 156 | 44 | 0.78 |
| RBF | G2+space | 190 | 10 | 0.95 |
|  | G3+space | 68 | 32 | 0.84 |
|  | G4+space | 116 | 84 | 0.58 |
|  | G5+space | 200 | 0 | 1 |
|  | G1+z | 124 | 76 | 0.62 |
|  | G2+z | 108 | 92 | 0.54 |
|  | G3+z | 124 | 76 | 0.62 |
|  | G4+z | 106 | 94 | 0.53 |
|  | G5+z | 162 | 38 | 0.81 |
|  | G1+space | 200 | 0 | 1 |
|  | G2+space | 72 | 128 | 0.36 |
| GNB | G3+space | 96 | 104 | 0.48 |
|  | G4+space | 138 | 62 | 0.69 |
|  | G5+space | 98 | 102 | 0.49 |
|  | G1+z | 136 | 64 | 0.68 |
|  | G2+z | 100 | 100 | 0.5 |
|  | G3+z | 164 | 36 | 0.82 |
|  | G4+z | 132 | 68 | 0.66 |
|  | G5+z | 100 | 100 | 0.5 |

The corresponding result is more clearly shown in Figure 1:

Figure 1. Results from Experiment one. G1-G5 means different group types with different normalization methods (space normalization and z-normalization) along with different kernels (linear, RBF, and GNB). Read bars shows the group has better performance than else.

In Figure 1, Group 5 with all combined features are normally used for comparison. Different groups are tested with different normalization and classifier methods. From the experiment we found that Group 2 with space normalization in linear and RBF mode achieve better accuracy, which are above 95%. Though Group 5 seems always achieve best performance (100%), when considering the efficiency, Group 2 may shows greater potential for classification.

Experiment 2:

In this experiment, nodes is fixed for each group.

G2+ space succed: 1051 fail 49 rate: 0.9554545454545454

G3+sp succed: 826 fail 274 rate: 0.7509090909090909

G4+sp succed: 628 fail 472 rate: 0.5709090909090909

G5+sp succed: 1040 fail 60 rate: 0.9454545454545454

Linear:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PAM | ER | GEO | DDM |
| G1 + space- | 72% Er: 14 to 2 | 36% Er: 32 to 1 | 100 % 0 | 100% 0 |
| G2 + space- | 100% 0 | 100% 0 | 96% Er:2 to 4 | 100% 0 |
| G3 + space- | 100% 0 | 54% Er: 24 to 1 | 100% 0 | 84% Er: 8 to 3 |
| G4 + space- | 52% Er: 24 to 2 | 36%Er: 32 to 1 | 56% Er:22 to 12 | 88% Er:6 to 3 |
| G5 + space- | 0 | 0 | 0 | 0 |
| G1 + z- | 76% | 36% | 80% | 56% |
| G2 + z- | 64% | 100% | 32% | 16% |
| G3 + z- | 4% | 100% | 68% | 76% |
| G4 + z- | 48% | 56% | 20% | 88% |
| G5 + z- | 80% | 100% | 100% | 44% |

RBF:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PAM | ER | GEO | DDM |
| G1 + space- | 76% | 36% | 100 % | 100% |
| G2 + space- | 84% | 100% | 96% | 100% |
| G3 + space- | 100% | 52% | 100% | 84% |
| G4 + space- | 52% | 36% | 56% | 88% |
| G5 + space- | 100 | 100 | 100 | 100 |
| G1 + z- | 76% | 36% | 80 % | 56% |
| G2 + z- | 68% | 100% | 32% | 16% |
| G3 + z- | 4% | 100% | 68% | 76% |
| G4 + z- | 48% | 56% | 20% | 88% |
| G5 + z- | 80% | 100% | 100% | 44% |

GNB:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PAM | ER | GEO | DDM |
| G1 + space- | 100% | 100% | 100 % | 100% |
| G2 + space- | 0% | 0% | 48% | 96% |
| G3 + space- | 100% | 0% | 0% | 92% |
| G4 + space- | 76% | 24% | 88% | 88% |
| G5 + space- | 0 | 0 | 100% | 96% |
| G1 + z- | 64% | 12% | 96 % | 100% |
| G2 + z- | 0% | 0% | 100% | 100% |
| G3 + z- | 92% | 52% | 100% | 84% |
| G4 + z- | 48% | 36% | 92% | 88% |
| G5 + z- | 0% | 0% | 100% | 100% |

Experiment 2: nodes is fixed:

Below table is Accuracy for each family with standard deviation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nodes = ? | PAM | ER | GEO | DDM |
| G1 + space- |  |  |  |  |
| G2 + space- |  |  |  |  |
| G3 + space- |  |  |  |  |
| G4 + space- |  |  |  |  |
| G5 + space- |  |  |  |  |
| G1 + z- |  |  |  |  |
| G2 + z- |  |  |  |  |
| G3 + z- |  |  |  |  |
| G4 + z- |  |  |  |  |
| G5 + z- |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | PAM | ER | GEO | DDM |
| Nodes = 200~300 | G1 + space- |  |  |  |  |
| G2 + space- |  |  |  |  |
| G3 + space- |  |  |  |  |
| G4 + space- |  |  |  |  |
| G5 + space- |  |  |  |  |
| G1 + z- |  |  |  |  |
| G2 + z- |  |  |  |  |
| G3 + z- |  |  |  |  |
| G4 + z- |  |  |  |  |
| G5 + z- |  |  |  |  |
| Nodes = ? | G1 + space- |  |  |  |  |
| G2 + space- |  |  |  |  |
| G3 + space- |  |  |  |  |
| G4 + space- |  |  |  |  |
| G5 + space- |  |  |  |  |
| G1 + z- |  |  |  |  |
| G2 + z- |  |  |  |  |
| G3 + z- |  |  |  |  |
| G4 + z- |  |  |  |  |
| G5 + z- |  |  |  |  |
| Nodes = ? | G1 + space- |  |  |  |  |
| G2 + space- |  |  |  |  |
| G3 + space- |  |  |  |  |
| G4 + space- |  |  |  |  |
| G5 + space- |  |  |  |  |
| G1 + z- |  |  |  |  |
| G2 + z- |  |  |  |  |
| G3 + z- |  |  |  |  |
| G4 + z- |  |  |  |  |
| G5 + z- |  |  |  |  |

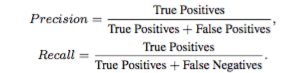
Experiment 3: edge density?

Experiment 4: rewire rate:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| rewire = 0(next table 10, 20, 30, 40,50,60,70,80,90,) | G1 + space- |  |  |  |  |
| G2 + space- |  |  |  |  |
| G3 + space- |  |  |  |  |
| G4 + space- |  |  |  |  |
| G5 + space- |  |  |  |  |
| G1 + z- |  |  |  |  |
| G2 + z- |  |  |  |  |
| G3 + z- |  |  |  |  |
| G4 + z- |  |  |  |  |
| G5 + z- |  |  |  |  |

Experiment 5: Similarity measurement

To measure how similar given two tested graph, we will use systematic area under precision-recall curve (AUPR) as measurement [5]. That is, if the two graphs are generated from same graph family, the graph pair is true, otherwise, false. For a given distance threshold , if two graphs is , they are considered positive, otherwise, negative. Then the AUPR is calculated given below equations:



(@jieao, @kevin, check this: <http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html>

)

To evaluate the AUPR, we considered (1) graphs with same sizes and densities; (2) graphs with different sizes and densities; (3) repeated above using different rewire rate.

@jieao, kevin, Below is an example: (到这个阶段应该选其中的某几个作为最优可能方案而不是全部) （table: AUPR-rewire rate）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000nodes, 0.01density(as example) | 0 | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% |
| G1 + space- |  |  |  |  |  |  |  |  |  |  |
| G2 + space- |  |  |  |  |  |  |  |  |  |  |
| G3 + space- |  |  |  |  |  |  |  |  |  |  |
| G4 + space- |  |  |  |  |  |  |  |  |  |  |
| G5 + space- |  |  |  |  |  |  |  |  |  |  |
| G1 + z- |  |  |  |  |  |  |  |  |  |  |
| G2 + z- |  |  |  |  |  |  |  |  |  |  |
| G3 + z- |  |  |  |  |  |  |  |  |  |  |
| G4 + z- |  |  |  |  |  |  |  |  |  |  |
| G5 + z- |  |  |  |  |  |  |  |  |  |  |

**Discussion and conclusion:**

In this project, we are trying to efficiently finish classification and similarity comparison using global feature vectors. Various different settings are used, including different kernels (SVM and GNB), different mode (linear and RBF), different normalization methods (space- and z-normalization), and different groups of features to generate better performance. The graphs are initially transformed into a feature-vector, which then be used for classification. The feature-vector, as we set the assumption, could represent the similarity of graphs in the same family, while could also differentiate graphs from different families. From experiment 1, we found that compared to other settings, especially to the group which combine all features, features that are grouped based on nodes and edges (namely, Group 2) with space normalization in linear and RBF mode achieve better accuracy, which are above 95%. This implies that more important rankings for properties related to nodes and edges. Also, there may exist tendency that as more global features are added into feature vector, better performance is achieved. This is reasonable because each feature describes the graph from specific angle, and more detailed description could better picture what graph is essentially. However, this unavoidably slows the whole classification process, because many features cost; for example, finding maximal cliques is NP-hard, which could consume lots of CPU for practical larger graph. Thus choosing proper features that describe the graph with enough accuracy is the best tradeoff. In current research, there are many discussions about this area. Some groups prefer local-feature to implement alignment-free methods, using Graphlet structure, etc. [5][9]; some group argue that global-features are enough to describe the graph [11]. Local-feature could efficiently find features for graphs locally, but the implementation is difficult and may misleading the result if local control is lost; global-features could be implemented easier, but how to finding proper features combination is still under discussion. Li, Geng, et al. [11] used as many as 20 features to achieve acceptable accuracy, while some of them cost when graphs are very large. Here from experiment 1, we preliminarily determine that features related to nodes and edges have better potential to achieve good accuracy. Although the fact exists that the four graph families may have essentially great structural differences, for example, DDM family always generates isolated nodes, while PAM family not, our research may provide another direction to find more efficient way for graph classification and similarity comparison.

For experiment 2/3/4: according to what result, we found what interesting points, so get what conclusion; compared to what existing machine learning, we argue/believe that what things.

**Documents**

The source code could be found in <https://github.com/allan888/alignment_free_network_comparison>

Please follow instruction in Readme.md to setup the experiments.

**Acknowledges**

  This work is tutored by Professor Robert Patro. Also the teaching assistant of class CSE 549, Hirak Sarkar helps us a lot. We give great thanks to Pro. Robert Patro and Hirak Sarkar for problem understanding, project processing and discussion.

**References**

1.     Robert Patro. CSE549 Project Ideas.pdf (2015).

2.     Tong, Amy Hin Yan, et al. "Global mapping of the yeast genetic interaction network." science 303.5659 (2004): 808-813.

3.     Singh, Rohit, Jinbo Xu, and Bonnie Berger. "Global alignment of multiple protein interaction networks with application to functional orthology detection." Proceedings of the National Academy of Sciences 105.35 (2008): 12763-12768.

4.     Cook, S. A. (1971). The complexity of theorem-proving procedures. In Proceedings of the third annual ACM symposium on Theory of computing, pages 151–158. ACM.

5.     Yaveroğlu, Ömer Nebil, Tijana Milenković, and Nataša Pržulj. "Proper evaluation of alignment-free network comparison methods." Bioinformatics(2015): btv170.

6.     Zhang, Yang, and Jeffrey Skolnick. "TM-align: a protein structure alignment algorithm based on the TM-score." Nucleic acids research 33.7 (2005): 2302-2309.

7.     Liao, Chung-Shou, et al. "IsoRankN: spectral methods for global alignment of multiple protein networks." Bioinformatics 25.12 (2009): i253-i258.

8.     Yaveroğlu, Ömer Nebil, et al. "Revealing the hidden language of complex networks." Scientific reports 4 (2014).

9.     Ali, Waqar, et al. "Alignment-free protein interaction network comparison."Bioinformatics 30.17 (2014): i430-i437.

10.  Moonesinghe, H. D. K., et al. "A probabilistic substructure-based approach for graph classification." Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on. Vol. 1. IEEE, 2007.

11.  Li, Geng, et al. "Effective graph classification based on topological and label attributes." Statistical Analysis and Data Mining: The ASA Data Science Journal 5.4 (2012): 265-283.

12.   <https://networkx.github.io>

13. <https://en.wikipedia.org/wiki/Support_vector_machine>

14. Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: A library for support vector machines." ACM Transactions on Intelligent Systems and Technology (TIST)2.3 (2011): 27.