University of Vienna
Faculty of Computer Science
Ass.-Prof. Dr. Nils M. Kriege
Dipl.-Ing. Lukas Miklautz
Johannes Stangl, BSc

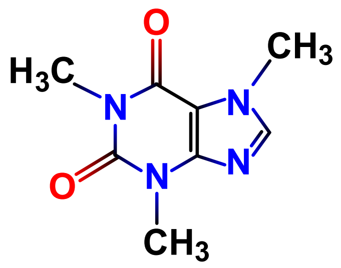
Data Mining: Programming Project

WS 2020/21

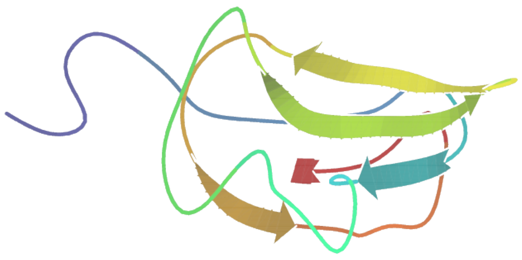
General Remarks:

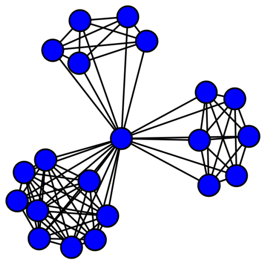
- This project should be conducted in teams of 4 to 5 students.
- The final deadline for the submission is at 23:59 o'clock on 24.01.2021. Please upload your solutions on Moodle. No deadline extension is possible. An intermediate submission of your projects current state will be at 23:59 o'clock on 29.11.2020, where you should already have finished most of task 1.
- After the final deadline on 24.01.2021, the peer review process on Moodle will start and you have to evaluate the other groups' work until 07.02.2021 at 23:59 p.m. You can find all the further details about the peer review process on Moodle.
- For answering the questions, please use Python and utilize the provided Python templates. Your code must be well-documented and readable; see the Style Guide for Python[1] for common style conventions. You may use jupyter notebooks for your report in which you explain and visualize your results, but the main code base should be in separate python files.
- Upload your solutions as a zip archive with the following naming scheme **team_X.zip**, where X is your teams number. The archive should contain your code as well as a PDF or HTML report describing your approach including your assumptions, results and a description how to compile and run your code. Additionally, you should prepare a 10 minute video for each task, in which you briefly introduce your team and present your results.
- Mark and cite external sources you are using in the code and report.
- You can use the designated forum to ask questions about the project. If needed, we will have a weekly online meeting (about 30 minutes) for all groups to discuss problems that you encountered.
- If you have problems do not hesitate to contact Johannes Stangl (johannes.stangl@univie.ac.at)/Lukas Miklautz (lukas.miklautz@univie.ac.at) or post a question on the dedicated Moodle forum.

[1] `https://www.python.org/dev/peps/pep-0008/`



(a) Small molecule


(b) Enzyme


(c) IMDB ego-network

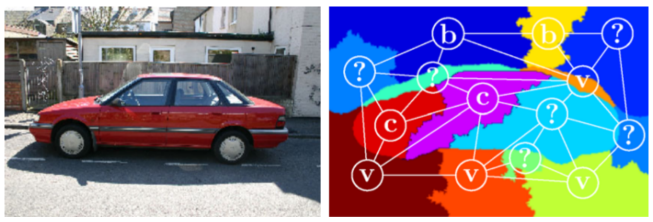

(d) Semantic image processing

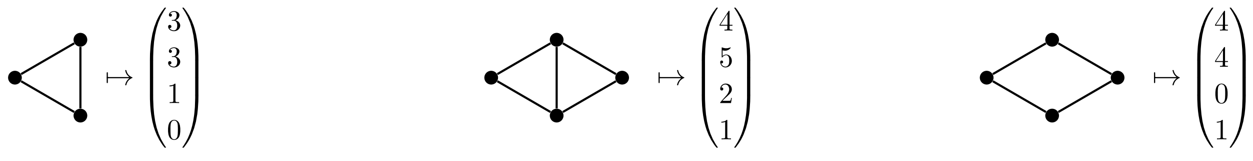
Figure 1: Examples of the graph from different domains.[2]

Graph Data Sets

Graphs are a versatile data structure and can be used to represent real-world objects from various domains. Your task will be to analyze one of the following data sets, which are available in the benchmark collection TUDataset at `graphlearning.io`. Experimental results of classification experiments for all the data sets can be found in [15] and [17].

- NCI1: Drug discovery projects typically have the goal to find a small molecule with a specific well-defined effect. Since structurally similar molecules often have a similar effect, predicting, e.g., toxicity or biological activity, based on structural properties is a promising approach. Small molecules can be naturally be represented as graphs, where nodes take the places of atoms and edges that of chemical bonds. This data set contains small molecules that were tested experimentally regarding their effect on a specific type of cancer. The class labels encode whether the molecule is able to inhibit the growth of cancer cells. The data set NCI1 was used in [20] and is a balanced subset of the original data set available at PubChem [1], see also [21].
- ENZYMES: In this data set every graph represents an enzyme. Nodes represent secondary structure elements (SSE) and are annotated by their type, i.e., helix, sheet or turn. Two nodes are connected by an edge if they are neighbors along the amino acid sequence or one of three nearest neighbors in space. Here, the task is to assign enzymes to one of the 6 EC top-level classes, which reflect the catalyzed chemical reaction. The data set was introduced in the publication [12], where also the graph model was proposed.
- IMDB-BINARY: This data set was derived from the Internet Movie Database (IMDb). Nodes represent actors and there is an edge between them if they appear in the same movie. Two collaboration graphs were derived based on movies belonging either to the genre Action or Romance. From these, ego-networks for the individual actors were obtained. The ego-network is the graph induced by the node representing the actor and all its neighbors. The class labels encode the genre of the collaboration graph. The data set was introduced in [22].

[2]Source Figure1b: *Graph Kernels*, S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, K.M. Borgwardt; JMLR (40), 2010. Source Figure1d: *Propagation kernels: efficient graph kernels from propagated information*, M. Neumann, R. Garnett, C. Bauckhage, K. Kersting; Machine Learning, 2015.



(a) F (b) G (c) H

Figure 2: Examples of a simple vector representation for the graphs F, G and H, where the components of the vector are the number of vertices, edges, triangles and cycles of length four, respectively.

- MSRC_21: This data set is related to a computer vision task. Here, images are modeled as graphs, where nodes represent continuous regions. Two nodes are connected by an edge if the two regions share a border. Nodes are labeled with semantic information, e.g., whether the region belongs to a building, a tree or a car, if available, see Figure 1d for an example. The task is to predict the main theme of the image. The data set was used in [18].

Note that the nodes may be annotated by labels, e.g., representing the atom or SSE type. For simplicity please ignore the real-valued node attributes of the ENZYMES data set.

Alternative Data Sets: The above data sets were selected from the repository `graphlearning.io`. You are also allowed to use different graph data sets. A interesting source is the *Open Graph Benchmark* website, were you can compare your results with researchers across the globe.[3] You may also generate your own graph data sets, if you come up with a problem that can be adequately modeled as a graph classification task. Information on the used file format can be found at `graphlearning.io`.

Vector Representations

There are various ways to represent graphs as vectors, see Figure 2 for an example. For the first task, we will provide vector representations obtained in the following ways. You will learn more about these techniques in the second part of the Data Mining course.

Graphlet The graphlet method computes a feature vector, where each component counts the number of occurrences of a specific connected subgraph containing three nodes. Every such subgraph is uniquely assigned to one component based on its structure and node labels.

Weisfeiler-Lehman This method is based on node label refinement. In each iteration of the refinement process each node is assigned a new label that uniquely encodes its current label as well as the multiset of labels of its neighbors. Hence, the extent of the neighborhood taken into account increases with every iteration. The vector representation are the stacked node label histograms for the first h iterations, where h is a parameter of the method.

Shortest Paths This approach first solves the all-pair-shortest-paths problem and then encodes all shortest paths in a vector representation as follows. Each component of the vector is associated with a triple (s, l, t), where s is the label of the start node, t the label of the end node and l the length of the shortest path between the start and end node. Each component counts the number of shortest paths of that type occurring in the graph.

[3] `https://ogb.stanford.edu`, see task *Graph Property Prediction.*

Task -1 Exploratory Data Analysis (50 P)

For each data sets we will provide you with vector representations from graph kernels that you can use as a starting point for your project. Based on the data set you should then conduct the following tasks:

(a) Make yourself familiar with the domain of graph property prediction and the data set that you have chosen. Choose one of the clustering algorithms listed in the Appendix (end of the document) or search for interesting algorithms that you think might be suited to your data set. If you decide to choose an algorithm not covered in the lecture, please consult us beforehand. You need to announce your choice of data set and algorithm at least until the 03.11.2020 at 23:59 o'clock on the dedicated Moodle forum.

(b) Implement the algorithm that you chose in Python[4] and compare your results with the algorithms implemented in Environment for DeveLoping KDD-Applications Supported by Index-Structures (ELKI). If you chose an algorithm, that is not implemented in ELKI compare your implementation with the code of the original authors. Check if your implementation is valid by comparing your results on some of the example benchmark datasets at the ELKI website.[5] (32 points)

(c) After validating your implementation, analyze the clustering results of your implemented algorithm on your chosen graph data set. For evaluating your clustering results you can use the metrics implemented in scikit-learn, which are shown in the clustering performance evaluation section.[6] Additionally, you should perform an exploratory data analysis (Feature Engineering, Dimensionality Reduction, Clustering, Outlier Detection, Visualisation, ...) with methods already implemented in sklearn and related libraries and investigate your data set. You can use the provided notebook for getting started with the exploratory data analysis. (18 points)

For the exploratory data analysis you could compare which cluster structures you can find in the different graph kernel representations and hand engineered features. Some example questions that you might want to explore are: How can the found clusters be interpreted. Do they correspond to the ground truth labels or something else? How much do the different clustering results differ or agree (e.g. by calculating the NMI between cluster solutions in different vector representations)? Based on this, can you already construct some hypothesis which features are well suited for classification and which classes will be hard to predict?
Further questions that would be interesting to explore are: If the found clusters do not correspond to the ground truth labelling, what other features might be interesting to add to make those clusters more explicit? What structures can you find besides the ground truth labelling? Which graph specific visualization techniques could be used?

To complete Task 1, summarize your results in an intermediate progress report (pdf or html file e.g. with jupyter notebooks) and upload a 10 minute video presentation of your projects current state.

[4]If you are aiming for an efficient implementation you may also implement the core algorithm in e.g. C++ or Java and only write a python interface to it.

[5]`https://elki-project.github.io/datasets`

[6]`http://scikit-learn.org/stable/modules/clustering.html`

Task -2 Graph Property Prediction (50 P)

In this task we consider classification problems for the data sets already used in the previous task. We use the software provided at `graphlearning.io`. Please follow the instructions provided at `https://github.com/chrsmrrs/tudataset` for installation and see our Collab Notebook for further information.

(a) Make yourself familiar with evaluation methods for classification and, in particular, the cross validation based approach implemented in the software package by following the tutorial at `https://chrsmrrs.github.io/datasets/docs/evaluation/`.

(b) Perform classification experiments with the baseline graph kernels including those used for clustering in Task 1. Try to relate your findings to your results for the clustering. Besides comparing the clustering quality regarding the class labels to the classification accuracy, it could be interesting to study specific classes or the predictions for individual elements. (13 points)

(c) Design your own graph kernel or graph neural network. Kernels can be easily combined and all kinds of graph and node properties can be used. The closure properties of kernels may help you to design new kernels, which are computed functionally or from the feature vectors for graphs. Make yourself familiar with graph and node properties used in the various disciplines such as social network analysis and structural graph theory. For example, the following properties could be of interest:

Graph properties

- Number of nodes and edges
- Degree histogram or distribution
- Node label histogram
- Connectivity
- Planarity
- Bipartiteness
- Treewidth
- ...

Node properties

- Centrality measures: Eigenvector, PageRank, degree, betweenness, closeness etc.
- Clustering coefficient
- ...

For further ideas of possible properties see `https://en.wikipedia.org/wiki/Graph_property`. Several graph properties can conveniently be computed with libraries such as NetworKit [7] or NetworkX. [8] You can convert the graphs from the tudataset to NetworkX graphs using the method displayed at the very end of the tutorial from (a). You may design a new kernel using some of these properties or combine them with existing graph kernels. Think about whether these properties provide information that is implicitly already contained in the individual baseline graph kernels or capture additional structural characteristics. (25 points)

(d) Experimentally evaluate your graph kernel. Try to improve your graph kernel based on your findings. (12 points)

To complete Task 2, summarize your results in a final report (pdf or html file e.g. with jupyter notebooks) and upload a 10 minute video presentation of your projects final results.

[7] `https://networkit.github.io/`

[8] `https://networkx.github.io/`

Appendix

Implement one of the following algorithms and compare your results with the algorithms implemented in the Environment for DeveLoping KDD-Applications Supported by Index-Structures (ELKI):

Correlation Clustering

- CASH [2]
- COPAC [6]
- ERiC: Exploring Relationships among Correlation Clusters [5]
- HiCO: Mining Hierachies of Correlation Clusters [7]
- LMCLU [13]
- ORCLUS: Arbitrarily ORiented projected CLUSter generation [9]

Subspace (axis-parallel) clustering algorithms

- CLIQUE [10]
- DiSH: Detecting Subspace cluster Hierachies [4]
- DOC: Density-based Optimal projective Clustering [19]
- HiSC: Finding Hierarchies of Subspace Clusters [3]
- P3C: A Robust Projected Clustering Algorithm [16]
- PreDeCon: Subspace Preference weighted Density Connected Clustering [11]
- PROCLUS: PROjected CLUStering [8]
- SUBCLU: Density connected Subspace Clustering [14]

References

[1] The PubChem project, http://pubchem.ncbi.nlm.nih.gov/

[2] Achtert, E., Böhm, C., David, J., Kröger, P., Zimek, A.: Robust clustering in arbitrarily oriented subspaces. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA. pp. 763–774 (2008), https://doi.org/10.1137/1.9781611972788.69

[3] Achtert, E., Böhm, C., Kriegel, H., Kröger, P., Müller-Gorman, I., Zimek, A.: Finding hierarchies of subspace clusters. In: Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18-22, 2006, Proceedings. pp. 446–453 (2006), https://doi.org/10.1007/11871637_42

[4] Achtert, E., Böhm, C., Kriegel, H., Kröger, P., Müller-Gorman, I., Zimek, A.: Detection and visualization of subspace cluster hierarchies. In: Advances in Databases: Concepts, Systems and Applications, 12th International Conference on Database Systems for Advanced Applications, DASFAA 2007, Bangkok, Thailand, April 9-12, 2007, Proceedings. pp. 152–163 (2007), https://doi.org/10.1007/978-3-540-71703-4_15

[5] Achtert, E., Böhm, C., Kriegel, H., Kröger, P., Zimek, A.: On exploring complex relationships of correlation clusters. In: 19th International Conference on Scientific and Statistical Database Management, SSDBM 2007, 9-11 July 2007, Banff, Canada, Proceedings. p. 7 (2007), https://doi.org/10.1109/SSDBM.2007.21

[6] Achtert, E., Böhm, C., Kriegel, H., Kröger, P., Zimek, A.: Robust, complete, and efficient correlation clustering. In: Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA. pp. 413–418 (2007), https://doi.org/10.1137/1.9781611972771.37

[7] Achtert, E., Böhm, C., Kröger, P., Zimek, A.: Mining hierarchies of correlation clusters. In: 18th International Conference on Scientific and Statistical Database Management, SSDBM 2006, 3-5 July 2006, Vienna, Austria, Proceedings. pp. 119–128 (2006), https://doi.org/10.1109/SSDBM.2006.35

[8] Aggarwal, C.C., Procopiuc, C.M., Wolf, J.L., Yu, P.S., Park, J.S.: Fast algorithms for projected clustering. In: SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA. pp. 61–72 (1999), http://doi.acm.org/10.1145/304182.304188

[9] Aggarwal, C.C., Yu, P.S.: Finding generalized projected clusters in high dimensional spaces. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA. pp. 70–81 (2000), http://doi.acm.org/10.1145/342009.335383

[10] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA. pp. 94–105 (1998), http://doi.acm.org/10.1145/276304.276314

[11] Böhm, C., Kailing, K., Kriegel, H., Kröger, P.: Density connected clustering with local subspace preferences. In: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK. pp. 27–34 (2004), https://doi.org/10.1109/ICDM.2004.10087

[12] Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S.V.N., Smola, A.J., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics **21 Suppl 1**, i47–i56 (Jun 2005). https://doi.org/10.1093/bioinformatics/bti1007

[13] Haralick, R.M., Harpaz, R.: Linear manifold clustering in high dimensional spaces by stochastic search. Pattern Recognition **40**(10), 2672–2684 (2007), https://doi.org/10.1016/j.patcog.2007.01.020

[14] Kailing, K., Kriegel, H., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004. pp. 246–256 (2004), https://doi.org/10.1137/1.9781611972740.23

[15] Kriege, N.M., Johansson, F.D., Morris, C.: A survey on graph kernels. Appl. Netw. Sci. **5**(1), 6 (2020). https://doi.org/10.1007/s41109-019-0195-3

[16] Moise, G., Sander, J., Ester, M.: P3C: A robust projected clustering algorithm. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China. pp. 414–425 (2006), https://doi.org/10.1109/ICDM.2006.123

[17] Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: Tudataset: A collection of benchmark datasets for learning with graphs. In: ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020) (2020), www.graphlearning.io

[18] Neumann, M., Garnett, R., Bauckhage, C., Kersting, K.: Propagation kernels: efficient graph kernels from propagated information. Machine Learning pp. 1–37 (2015), http://dx.doi.org/10.1007/s10994-015-5517-9

[19] Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.M.: A monte carlo algorithm for fast projective clustering. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002. pp. 418–427 (2002), http://doi.acm.org/10.1145/564691.564739

[20] Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**, 2539–2561 (2011)

[21] Wale, N., Watson, I.A., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. Knowl. Inf. Syst. **14**(3), 347–375 (2008)

[22] Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1365–1374. KDD '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2783258.2783417