In Proc. 20th International Conference on Scientific and Statistical Database Management (SSDBM 2008), Hong Kong, China, 2008, pp. 580-585.

ELKI: A Software System for Evaluation of Subspace Clustering Algorithms

Elke Achtert, Hans-Peter Kriegel, Arthur Zimek

Institute for Informatics, Ludwig-Maximilians-Universität München
`http://www.dbs.ifi.lmu.de`
{`achtert,kriegel,zimek`}`@dbs.ifi.lmu.de`

Abstract. In order to establish consolidated standards in novel data mining areas, newly proposed algorithms need to be evaluated thoroughly. Many publications compare a new proposition – if at all – with one or two competitors or even with a so called "naïve" *ad hoc* solution. For the prolific field of subspace clustering, we propose a software framework implementing many prominent algorithms and, thus, allowing for a fair and thorough evaluation. Furthermore, we describe how new algorithms for new applications can be incorporated in the framework easily.

1 Introduction

In an active research area like data mining, a plethora of algorithms is proposed every year. Most of them, however, are presented once and never heard about again. On the other hand, newly proposed algorithms are often evaluated in a sloppy way taking into account only one or two partners for comparison of efficiency and effectiveness, presumably because for most algorithms no implementation is at hand. And if an implementation is provided by the authors, a *fair* comparison is nonetheless all but impossible due to different performance properties of different programming languages, frameworks, and, last but not least, implementation details. Eventually, an evaluation based on implementations of different authors is more likely to be a comparison of the efforts of different authors in efficient programming rather than truly an evaluation of algorithmic merits.

Recently, an understanding for the need for consolidation of a maturing research area is rising in the community as illustrated by the discussions about the repeatability of results for SIGMOD 2008, the Panel on performance evaluation at VLDB 2007, and the tentative special topic of "Experiments and Analyses Papers" at VLDB 2008.

In the software system described in this paper, we try to facilitate a fair comparison of many subspace clustering algorithms based on experimental evaluation. The framework provides the data management independently of the tested algorithms. So all algorithms are comparable on equal conditions. The implementation aims at effectiveness in a balanced way for all algorithms. But even more important is an intuitive and easy-to-understand programming style to invite contributions in the future when the framework is made available open source.

2 An Overview on the Software System

A wealth of data-mining approaches is provided by the almost "classical" open source machine learning framework Weka [1]. We consider Weka as the most prominent and popular environment for data mining algorithms. However, the focus and strength of Weka is mainly located in the area of classification, while clustering approaches are somewhat underrepresented.

The same holds true for another framework for data mining tasks: YALE [2]. This is a rather complex environment that completely incorporates Weka. The main focus of YALE is in supporting "rapid prototyping", i.e. to ease the definition of a specific data mining task as a combination of a broad range of available methods. While Weka is restricted to use numerical or nominal features (and in some cases strings), YALE does also extend the range of possible input data.

Although both, Weka and YALE, support the connection to external database sources, they are based on a flat internal data representation. Thus, experiments assessing the impact of an index structure on the performance of a data mining application are not possible using these frameworks.

On the other hand, frameworks for index structures, such as GiST [3], do not provide any precast connection to data mining applications.

To connect both worlds, we demonstrate the Java Software Framework ELKI (**E**nvironment for Deve**L**oping **K**DD-Applications Supported by **I**ndex Structures). ELKI comprises on the one hand a profound and easily extensible collection of algorithms for data mining applications, such as item-set mining, clustering, classification, and outlier-detection, and on the other hand ELKI incorporates and supports arbitrary index structures to support even large, high-dimensional data sets. But ELKI does also support the use of arbitrary data types, not only feature vectors of real or categorical values. Thus, it is a framework suitable to support the development and evaluation of new algorithms at the cutting edge of data mining as well as to incorporate experimental index structures to support complex data types.

ELKI intends to ease the development of new algorithms by providing a wealth of helper classes and methods for algebraic and analytic computations, and simulated database support for arbitrary data types using an index structure at will.

2.1 The Environment: A Flexible Framework

As a framework, our software system is flexible in a sense, that it allows to read arbitrary data types (provided there is a suitable parser for your data file or adapter for your database), and supports the use of any distance or similarity measure (like some kernel-function) appropriate for the given data type. So far, many implementations of data mining algorithms – especially subspace clustering algorithms – still rely on the numeric nature of feature vectors as underlying data structure. Our framework is already one step ahead and ready to work on complex data types. Generally, an algorithm needs to get provided a distance of some sort. Thus, distance functions connect arbitrary data types to arbitrary algorithms.

The architecture of the software system separates data types, data management, and data mining applications. So, different tasks can be implemented independently. A new data type can be implemented and used by many algorithms, given a suitable distance function is defined. An algorithm will perform its routine irrespectively of the data handling which is encapsulated in the database. A database may facilitate efficient data management via incorporated index structures.

2.2 Available Algorithms

While the framework is open to all kind of data mining applications, the main focus in the development of ELKI has been on clustering and especially subspace clustering (axis-parallel as well as arbitrarily oriented). Available general clustering algorithms are SLINK [4], k-means [5], EM-clustering [6], DBSCAN [7], Shared-Nearest-Neighbor-Clustering [8], OPTICS [9], and DeLiClu [10]. There are axis-parallel subspace and projected clustering approaches implemented like CLIQUE [11], PROCLUS [12], SUBCLU [13], PreDeCon [14], HiSC [15], and DiSH [16]. Furthermore, some biclustering or pattern-based clustering approaches are supported like δ-bicluster [17], FLOC [18] or p-cluster [19], and correlation clustering approaches are incorporated like ORCLUS [20], 4C [21], HiCO [22], COPAC [23], ERiC [24], and CASH [25]. The improvements on these algorithms described in [26] are also integrated in ELKI.

2.3 Development of Subspace Clustering Algorithms

Often, the main difference between clustering algorithms is the way to assess the distance or similarity between objects or clusters. So, while other well known and popular software systems like Weka [1] or YALE [2] predefine the Euclidean distance as only possible distance between different objects to use in clustering approaches (beside some kernel functions in classification approaches), ELKI allows the flexible definition of any distance measure. This way, subspace clustering approaches that differ mainly in the definition of distance between points (like e.g. COPAC and ERiC) can use the same algorithmic routine and become, thus, highly comparable in their performance.

Distance functions are used to perform range queries on a database object. Any implementation of an algorithm can rely on the database object to perform range queries with an arbitrary distance function and needs only to ask for k nearest neighbors not being concerned with the details of data handling.

A new data type is supposed to implement the interface `DatabaseObject`. A new algorithm class suitable to certain data types `O` needs to implement the Interface `Algorithm<O extends DatabaseObject>`. The central routine to implement the algorithmic behavior is `void run(Database<O> database)`. Here, the algorithm is applied on an arbitrary database consisting of objects of a suitable data type. The database supports operations like

```
<D extends Distance<D>>
List<QueryResult<D>>
kNNQueryForObject(O queryObject,
                  int k,
                  DistanceFunction<O,D> distanceFunction)
```

performing a k-nearest neighbor query for a given object of a suitable data type `O` using a distance function that is suitable for this data type `O` and provides a distance of a certain type `D`. Such a query method returns a list of `QueryResult<D>` objects encapsulating the database id of the collected objects and their distance to the query object in terms of the specified distance function.

A new subspace clustering algorithm may therefore use a specialized distance function and implement a certain routine using this distance function on an arbitrary database.

2.4 Support of Arbitrary Index-Structures

As pointed out above, while existing frameworks for index-structures, such as GiST [3], do not provide any precast connection to data mining applications, well-known data-mining frameworks like Weka [1] or YALE [2] do not support the internal use of index structures.

Our software system ELKI supports the use of arbitrary index structures in combination with, e.g., a clustering algorithm. Already available within ELKI are metric index-structures like MTree [27], MkCoPTree and its variants MkTabTree and MkMaxTree [28], and MkAppTree [29] and spatial index-structures like RStarTree [30], DeLiCluTree [10], and RdkNNTree, an extension from [31] for $k \geq 1$.

Index structures are encapsulated in database objects. These database objects facilitate range queries using arbitrary distance functions. Algorithms operate on database objects irrespective of the underlying index structure. So the implementation of an algorithm, as pointed out above, is not concerned with the details of handling the data which can be supported by arbitrary efficient procedures.

This is interesting because the complexity of algorithms is often analyzed theoretically on the basis of index structures but often, if implementations are provided, an index structure is not included and cannot be incorporated in the particular implementation.

2.5 Setting Up Experiments

The integration of several algorithms into one software framework also allows for setting up complex experiments comparing different algorithms in an easy way and on equal terms. We plan to use the framework for extensive comparisons of a broad range of subspace clustering algorithms.

2.6 Availability and Documentation

The framework ELKI is available for download and use via

`http://www.dbs.ifi.lmu.de/research/KDD/ELKI/`.

There is provided an extensive documentation of the implementation and usage as well as examples to illustrate how to expand the framework by integrating new algorithms.

3 Conclusion

The software system ELKI presents a large collection of data mining applications (mainly clustering and – axis parallel or arbitrarily oriented – subspace and projected clustering approaches). Algorithms can be supported by arbitrary index structures and work on arbitrary data types given supporting data classes and distance functions. We therefore expect ELKI to facilitate broad experimental evaluations of algorithms – existing algorithms and newly developed ones alike.

References

1. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. 2nd edn. Morgan Kaufmann (2005)
2. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid prototyping for complex data mining tasks. In: Proc. KDD. (2006)
3. Hellerstein, J.M., Naughton, J.F., Pfeffer, A.: Generalized search trees for database systems. In: Proc. VLDB. (1995)
4. Sibson, R.: SLINK: An optimally efficient algorithm for the single-link cluster method. The Computer Journal **16**(1) (1973) 30–34
5. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematics, Statistics, and Probabilistics. Volume 1. (1967) 281–297
6. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B **39**(1) (1977) 1–31
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. KDD. (1996)
8. Ertöz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: Proc. SDM. (2003)
9. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. SIGMOD. (1999)

10. Achtert, E., Böhm, C., Kröger, P.: DeLiClu: Boosting robustness, completeness, usability, and efficiency of hierarchical clustering by a closest pair ranking. In: Proc. PAKDD. (2006)
11. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. SIGMOD. (1998)
12. Aggarwal, C.C., Procopiuc, C.M., Wolf, J.L., Yu, P.S., Park, J.S.: Fast algorithms for projected clustering. In: Proc. SIGMOD. (1999)
13. Kailing, K., Kriegel, H.P., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: Proc. SDM. (2004)
14. Böhm, C., Kailing, K., Kriegel, H.P., Kröger, P.: Density connected clustering with local subspace preferences. In: Proc. ICDM. (2004)
15. Achtert, E., Böhm, C., Kriegel, H.P., Kröger, P., Müller-Gorman, I., Zimek, A.: Finding hierarchies of subspace clusters. In: Proc. PKDD. (2006)
16. Achtert, E., Böhm, C., Kriegel, H.P., Kröger, P., Müller-Gorman, I., Zimek, A.: Detection and visualization of subspace cluster hierarchies. In: Proc. DASFAA. (2007)
17. Cheng, Y., Church, G.M.: Biclustering of expression data. In: Proc. ISMB. (2000)
18. Yang, J., Wang, W., Wang, H., Yu, P.S.: δ-clusters: Capturing subspace correlation in a large data set. In: Proc. ICDE. (2002)
19. Wang, H., Wang, W., Yang, J., Yu, P.S.: Clustering by pattern similarity in large data sets. In: Proc. SIGMOD. (2002)
20. Aggarwal, C.C., Yu, P.S.: Finding generalized projected clusters in high dimensional space. In: Proc. SIGMOD. (2000)
21. Böhm, C., Kailing, K., Kröger, P., Zimek, A.: Computing clusters of correlation connected objects. In: Proc. SIGMOD. (2004)
22. Achtert, E., Böhm, C., Kröger, P., Zimek, A.: Mining hierarchies of correlation clusters. In: Proc. SSDBM. (2006)
23. Achtert, E., Böhm, C., Kriegel, H.P., Kröger, P., Zimek, A.: Robust, complete, and efficient correlation clustering. In: Proc. SDM. (2007)
24. Achtert, E., Böhm, C., Kriegel, H.P., Kröger, P., Zimek, A.: On exploring complex relationships of correlation clusters. In: Proc. SSDBM. (2007)
25. Achtert, E., Böhm, C., David, J., Kröger, P., Zimek, A.: Robust clustering in arbitrarily oriented subspaces. In: Proc. SDM. (2008)
26. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: A general framework for increasing the robustness of PCA-based correlation clustering algorithms. In: Proc. SSDBM. (2008)
27. Ciaccia, P., Patella, M., Zezula, P.: M-Tree: an efficient access method for similarity search in metric spaces. In: Proc. VLDB. (1997)
28. Achtert, E., Böhm, C., Kröger, P., Kunath, P., Pryakhin, A., Renz, M.: Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In: Proc. SIGMOD. (2006)
29. Achtert, E., Böhm, C., Kröger, P., Kunath, P., Pryakhin, A., Renz, M.: Approximate reverse k-nearest neighbor search in general metric spaces. In: Proc. CIKM. (2006)
30. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-Tree: An efficient and robust access method for points and rectangles. In: Proc. SIGMOD. (1990) 322–331
31. Yang, C., Lin, K.I.: An index structure for efficient reverse nearest neighbor queries. In: Proc. ICDE. (2001)