Pattern-Oriented Hierarchical Clustering

Tadeusz Morzy, Marek Wojciechowski, Maciej Zakrzewicz

Poznan University of Technology
Institute of Computing Science
ul. Piotrowo 3a, 60-965 Poznan, Poland
Tadeusz.Morzy@put.poznan.pl
Marek.Wojciechowski@cs.put.poznan.pl
Maciej.Zakrzewicz@cs.put.poznan.pl

Abstract. Clustering is a data mining method, which consists in discovering interesting data distributions in very large databases. The applications of clustering cover customer segmentation, catalog design, store layout, stock market segmentation, etc. In this paper, we consider the problem of discovering similarity-based clusters in a large database of event sequences. We introduce a hierarchical algorithm that uses sequential patterns found in the database to efficiently generate both the clustering model and data clusters. The algorithm iteratively merges smaller, similar clusters into bigger ones until the requested number of clusters is reached. In the absence of a well-defined metric space, we propose the similarity measure, which is used in cluster merging. The advantage of the proposed measure is that no additional access to the source database is needed to evaluate the inter-cluster similarities.

1 Introduction

Clustering is one of the most popular data mining methods [2] [3] [4] [5] [6] [7] [8] [9] [11]. It consists in discovering interesting data distributions and patterns in very large databases. Given \( k \) data points in a \( d \)-dimensional metric space, the problem of clustering is to partition the data points into \( n \) clusters such that the data points within a cluster are closer (more similar) to each other than data points in different clusters. Clustering is often used for market segmentation, in which the customers are divided into groups based on the similarity of their characteristics.

Clustering algorithms typically determine \( n \) partitions that optimize some criterion function. The most commonly used criterion is the square-error criterion defined as follows:

\[
E = \sum_{i=1}^{n} \sum_{p \in c_i} \left| p - m_i \right|^2
\]  

(1)

* This work was partially supported by the grant no. KBN 43-1309 from the State Committee for Scientific Research (KBN), Poland.

© Springer-Verlag Berlin Heidelberg 1999
where \( m_i \) is the mean of cluster \( c_i \), \( p \) is a data point, and \( E \) is the square error. Many clustering algorithms employ the idea of hierarchical clustering, which consists in merging pairs of similar clusters to form new larger clusters.

Traditional clustering approaches deal with points in \( d \)-dimensional space. However, not all types of information can be represented in this form. In many applications, users operate on databases of event sequences, such as customer purchase history given in Figure 1, where an ordered set of purchased products is stored for each customer. Let us consider the general problem of the similarity of sequences. It seems that e.g. the sequences \( a \rightarrow b \rightarrow c \) and \( b \rightarrow c \rightarrow d \) are similar since the both contain the same subsequence \( b \rightarrow c \). But what can we say about the similarity of the sequences e.g. \( a \rightarrow b \) and \( c \rightarrow d \)? We claim that these two sequences also can be considered similar, if there are many other sequences in the database, that contain them both, e.g. \( a \rightarrow b \rightarrow c \rightarrow d \), \( a \rightarrow c \rightarrow b \rightarrow d \), etc. Therefore, we assume that two sequences are similar if either they contain the identical subsequences, or their subsequences have the tendency to co-occur together in some other sequences.

For example, the customer 103 is more similar to the customer 104 than to the customer 105 since the pair 103-104 has a common subsequence (\( tv_set \rightarrow vcr \rightarrow cassette \) ) while the pair 103-105 has no common subsequences. We are interested in clustering sequences based on such intuitive similarity measure. We notice that this problem cannot be solved using traditional clustering methods because: 1. the sequences are variable-length, 2. the sequences cannot be represented in a \( d \)-dimensional metric space, and 3. no natural distance function is available.

In this paper we address and solve the problem of partial clustering of sequential data. We are interested in discovering an arbitrary number of possibly overlapping clusters that hold the customers, whose behavior is similar to each other. We refer to our clustering method as to partial clustering, because we allow the customers who are not similar to any other not to be covered by any cluster, and we allow a customer to belong to more than one cluster. To perform the partial clustering of sequential data, we employ the idea of sequential pattern discovery [1] [10]. A sequential pattern is a frequently occurring subsequence of database sequences. Sequential pattern discovery consists in finding all sequential patterns, whose frequency is above some user-defined minimum value. Thus, the discovered sequential patterns represent the most common subsequences within the database sequences and can be used to determine their similarity. For example, the sequential patterns that can be discovered in the database from Figure 1 are: ‘\( tv_set \rightarrow vcr \rightarrow cassette \)’ (contained in two database sequences), ‘\( book \rightarrow c\_disk \)’ (also contained in two), etc. Each of those sequential patterns says that a number of customers bought one product, later on they bought some other product, and so on.

The presented algorithm uses sequential patterns discovered in the database to generate both the clustering model and cluster contents. For example, our algorithm executed on the database from Fig. 1 with \( n=2 \) gives two clusters based on the following clustering model:

- Cluster 1: described by the patterns: \( tv_set \rightarrow vcr \rightarrow cassette \) and \( bicycle \rightarrow b\_ball \); the cluster contains the following customers: 102, 103, 104,
- Cluster 2: described by the patterns: \( book \rightarrow c\_disk \) and \( lamp \rightarrow pillow \); the cluster contains the following customers: 101, 105.