SUBCLU

Subspace Clustering for High-Dimensional Data

Many areas of study produce data that can benefit from data mining techniques, such as clustering. The data produce can however be very high-dimensional and therefore more difficult to process and interpret. Different methods exist to reduce this difficulty but have their own drawbacks. For example, the technique of dimensionality reduction attempts to map the entirety of the feature space to a subspace of lower-dimension and relevant attributes. However, this may result in the loss of an attributes intuitive meaning, causing it to be more difficult to draw conclusions from produced clusters. Additionally, the clusters produced may exist only in one unique subspace, resulting in information that may correspond to alternate subspaces to be lost. And finally, dimensionality reduction may not provide for any beneficial results at all. A second technique of projected clustering which also may result in data points clustering differently in particular subspaces. This may be seen illustrated in the figure below.

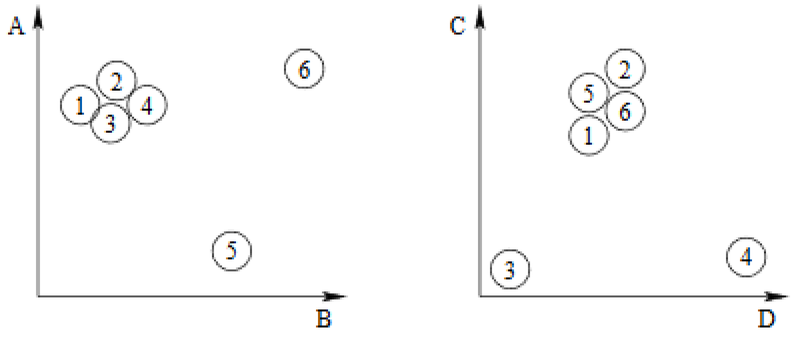

Figure 1: Drawback of projected clustering

The algorithm SUBCLU was presented by Karin Kailing, Hans-Peter Kriegel, and Peer Kröger as a means of clustering high-dimensional data based on a property called monotonicity. This property states that if a cluster exists in a subspace, a cluster also exists in each subspace that comprises the original. This does not necessarily hold that the cluster in the original subspace is identical to the cluster in the comprising subspaces. However, if there exists a density- connected set in a subspace, each of its comprising subspaces must also contain that density-connected set. This is the down-ward closure that forms the basis of the algorithm.

Similar to the apriori algorithm used to form frequent item sets, the SUBCLU algorithm begins by performing clustering on each of the one-dimensional spaces of the data set corresponding to a single attribute. This clustering is done using the density-based clustering algorithm DBSCAN in order to make use of the monotonicity of clusters. Then, if the 1-D subspaces have clusters detected through DBSCAN, they are used to form candidate subspaces in 2-D. The subspace that produced the fewest clustered points is then marked as the optimal subspace for the next round of clustering. This reduces the runtime of the DBSCAN clustering by focusing on a subset of the overall database where clusters are already known to exist. The process then repeats with candidate subspace generation and pruning, optimal subspace detection, and density-based clustering until no clusters are detected in the candidate subspaces. Pseudocode for the algorithm and candidate generation will be featured in the following figures.

```
SUBCLU($DB, eps, MinPts$)
    $S_1 := \emptyset$
    $C_1 := \emptyset$
    for each $a \in Attributes$
        $C^{\{a\}} = $ DBSCAN($DB, \{a\}, eps, MinPts$)
        if($C^{\{a\}} \neq \emptyset$)
            $S_1 := S_1 \cup \{a\}$
            $C_1 := C_1 \cup C^{\{a\}}$
        end if
    end for
    // In a second step, $k+1$-dimensional clusters are built from $k$-dimensional ones:
    $k := 1$
    while($C_k \neq \emptyset$)
        $\texttt{CandS}_{k+1}$ := GenerateCandidateSubspaces($S_k$)
        for each $cand \in \texttt{CandS}_{k+1}$
            bestSubspace := $\min_{s \in S_k \wedge s \subset cand} \sum_{C_i \in C^s} |C_i|$
            $C^{cand} := \emptyset$
            for each cluster $cl \in C^{\texttt{bestSubspace}}$
                $C^{cand} := C^{cand} \cup$ DBSCAN($cl, cand, eps, MinPts$)
                if ($C^{cand} \neq \emptyset$)
                    $S_{k+1} := S_{k+1} \cup cand$
                    $C_{k+1} := C_{k+1} \cup C^{cand}$
                end if
            end for
        end for
        $k := k + 1$
    end while
end
```

Figure 2: SUBCLU pseudocode

```
GenerateCandidateSubspaces($S_k$)
    CandS$_{k+1}$ := $\emptyset$
    for each $s_1 \in S_k$
        for each $s_2 \in S_k$
            if ($s_1$ and $s_2$ differ in exactely one attribute)
                CandS$_{k+1}$ := CandS$_{k+1} \cup \{s_1 \cup s_2\}$
            end if
        end for
    end for
    // Pruning of irrelevant candidate subspaces
    for each $cand \in$ CandS$_{k+1}$
        for each $k$-element $s \subset cand$
            if ($s \not\subset S_k$)
                CandS$_{k+1}$ = CandS$_{k+1} \setminus \{cand\}$
            end if
        end for
    end for
end
```

Figure 3: Pseudocode for candidate subspace generation and pruning

The SUBCLU algorithm was tested by its authors against the CLIQUE algorithm, which uses a grid-based subspace clustering technique, on several sets of synthetic data. The results were greatly in the favor of SUBCLU in terms of overall accuracy of cluster detection. The algorithm was also applied to a database on gene expression and producing meaningful results thus proving its prowess for real-world applications as well. The algorithm also scales well with a runtime of O(n^2) with respect to database size, dimensionality of the data, and dimensionality of the clusters.