Application of DiverseRank to Clustering

B.Tech. Project submitted in partial fulfillment
of the requirements for the degree of
Bachelors of Technology
in
CSE
by

Mayank Gupta

201101004

mayank.g@students.iiit.ac.in

Ankush Jain

201101010

ankush.jain@students.iiit.ac.in

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD - 500 032, INDIA

November 2014

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this project, titled
"Application of DiverseRank to Clustering"
by Mayank Gupta with Roll No. 201101004 and
Ankush Jain with Roll No. 201101010
has been carried out under my supervision for partial fulfillment of degree.

Signed: _____

(Adviser: Dr. P. Krishna Reddy)

Date: _____

“Winter is coming.”

— Ned Stark

Abstract

The process of clustering divides a set of data objects into smaller sets consisting of similar objects.

These clustering algorithms do not work efficiently in higher dimensional spaces because of the inherent sparsity of the data.

Projected clustering algorithms have been proposed to find the clusters in hidden subspaces. The identification of more relevant attributes is a research issue in the projected clustering.

In literature, an approach has been proposed to find the relevant subspaces using Frequent-Pattern Mining approaches. The frequent pattern can able to identify the item sets with high support. In frequent pattern based projected clustering approach, we observed that only frequent itemsets are considered.

In this paper, we propose an approach to refine clusters generated by the frequent pattern based approach and produce more accurate clusters. In the proposed approach, we identify the irrelevant objects in a given cluster by exploiting the notion of diversity. Based on the notion of diversity, which captures the extent the items in the pattern belong to multiple categories of a concept hierarchy, the objects the membership of an object is identified. The membership of each object decides the existance of the object in the cluster. This process iteratively refines each cluster. We conduct the experiments on the real world datasets and show that the proposed approach improves the quality of the cluster

Acknowledgements

We are grateful to Mr. M. Kumaraswamy and Dr. P. Krishna Reddy for their constant help and guidance throughout the project.

Contents

Dedicated to friends, family and pets ...

Chapter 1

Introduction

Clustering is a Data Mining technique, that, given a dataset, divides it into groups of objects such that the objects in a group will be similar to one another and different from the objects in other groups[17]. The objects are represented as a point, where each dimension corresponds to an attribute/feature and the feature value of each object determines its coefficient in the corresponding dimension. The data points are grouped into clusters based on the some similarity metric. The clustering algorithms have large number of applications such as marketing (e.g., customer segmentation), image analysis, bio- informatics, document classification, indexing, etc.

1.1 Issues with Clustering Techniques

Most clustering algorithms do not work efficiently in higher dimensional space because of the inherent sparsity of the data [3]. In high dimensional space, it is likely that the distance of any two points is almost the same for a large class of common distributions [5]. So, a clustering algorithm employ feature selection based approach know as subspaces clustering [11]. The goal of the subspace clustering is to find the particular dimensions on which the data points are correlated and pruning away the remaining dimensions that reduces the noise in the data.

The problem in these algorithms is that picking certain dimensions in advance can lead to a loss of information. Furthermore, in many real data, some points are correlated with respect to a given set of dimensions and others are correlated with respect to different dimensions. Thus, it may not always be feasible to prune off too many dimensions without considering the data at the same time incurring a substantial loss of information. Alternatively, the effects of dimensionality can be reduced by a dimensionality reduction

technique[6], however, in- formation from all dimensions is uniformly transformed and relevant information for some clusters may be reduced. Also, the clusters may be hard to understand.

1.2 Available Algorithms

The projected clustering[1] alogrithms overcome some of the issues of the subspace clustering. In projected clustering, a set of data points with an associated set of relevant dimensions are employed such that the data points are similar to each other in the subspace formed by the relevant dimensions, but dissimilar to data point outside the cluster. The widely used distance measures are more meaningful in projections of the high-dimensional space, where the object values are dense[9]. In other words, it is more likely for the data to form dense, meaningful clusters in a high-dimensional subspace. CLIQUE [3], PROCLUS [1], ORCLUS [2], and DOC [12] are some efforts in the projected clustering approaches.

These approaches suffer from the quality of the clusters and consumes lot of time. An effort is made to exploit the frequent pattern mining in the area of projected clustering to address the issue of quality clusters[18, 19]. However, the frequent pattern mining[4] approach identifies the itemsets (features/attributes) with high support. In frequent pattern based projected clustering approach, we observed that only frequent itemsets are considered and the infrequent itemsets are completely eliminated from the clustering process.

1.3 Our Approach

We propose an approach to refine the clusters generated by the frequent pattern based projected clustering approach and produce more qualitative clusters. We observed that a high support pattern includes few items and eliminates non-frequent items. As a result of elimination of some of the items, the influence of other items is not considered in the process of clustering. Due to elimination of some data values, the the quality of clusters gets affected. In the proposed approach, for a given cluster, we identify the mem- bership of each data point employing the notion called diversity. The diversity identify the non-members of the given cluster and remove the objects from the cluster. We conduct the experiment on the real world datasets and show that the proposed approach improves the quality of clusters.

Chapter 2

Analysis Of Existing Algorithms

2.1 CLIQUE, PROCLUS and ORCLUS

One of the first algorithms in the area of projected clustering algorithms is CLIQUE [3]. CLIQUE finds the dense regions (clusters) in a level-wise manner, based on the Apriori principle. However, this algorithm does not scale well with data dimensionality. In addition, the formed clusters have large overlap, and this may not generate clear disjoint partitions. PROCLUS[1] and ORCLUS[2] employ alternative techniques. They are much faster than CLIQUE and they can discover disjoint clusters. In PROCLUS, the dimensions relevant to each cluster are selected from the original set of attributes. ORCLUS is more general and can select relevant attributes from the set of arbitrary directed orthogonal vectors. PROCLUS fails to identify clusters with large difference in size and requires their dimensionality to be in a predefined range. ORCLUS may discover clusters that are hard to interpret.

2.2 DOC

DOC [12] is a density-based algorithm that iteratively discovers projected clusters in a data set. DOC discovers one cluster at a time. At each step, it tries to guess a good medoid for the next cluster to be discovered. It repeatedly picks a random point from the database and attempts to discover the cluster centered at random point. Among all discovered clusters, the cluster with the highest quality is selected. The process repeated for number of random points.

2.3 Frequent Pattern-based Algorithm/MinClus

The frequent itemset mining problem was first proposed in [4]. Frequent patterns are generated from the transactional databases for a given minSup. Subsequently, an efficient approach was proposed in FP-growth[8] to generate frequent patterns without generating the candidate items. In [18, 19], a projected clustering algorithm was proposed. The algorithm iteratively produces one cluster at a time by exploiting the frequent pattern mining approach. The top supported frequent patterns are employed to find the item sets and generate the clusters. A strength function is defined to arrange the clusters in to strong and weak clusters. The strong clusters are produced as the output and the objects in the weak clusters are added back to the database. The process is iterated till all the objects are grouped into the clusters. The FP-growth approach has been employed to generate the frequent patterns.

Chapter 3

MinClus and Diversity/DiverseRank

3.1 MinClus

The approach works based on the given a random medoid $p \in S$, the approach transform best projected cluster containing p, to the transactional databases. If the attribute of a record is bounded by p with respect to the width w (here, $w = 2$), an item for that attribute is added to the corresponding itemset. The appraoch observe that all frequent itemsets (i.e., combinations of dimensions) with respect to minsup $= \alpha * |S|$ are candidate clusters for medoid p. The problem of finding the best projected cluster for a random medoid p can be transformed to the problem of finding the best itemset in a transformation of S, where goodness is defined by the μ function (refer Equation 3.1).

Instead of discovering it in an non-deterministic way, a systematic data mining algorithm on S is applied. The association between the data items is identified by the frequent pattern mining algoirhtm. Due to association in the data items, the frequent pattern mining approach has been exploited in projected cluter approachs. Recently, there is a more efficient algorithm, the FP-growth method [8]. Here, the appraoch adopt frequent pattern mining for subspace clustering. However, the objective is to find the frequent itemset with maximum μ value, rather than finding all frequent subspaces. Assume that (I_{best}) is the itemset with the maximum μ value found so far and let $dim(I_{best})$ and $sup(I_{best})$ be its dimensionality and support, respectively.

Let I_{cond} be the current conditional pattern of the FP-growth process. Its support $sup(I_{cond})$ gives an upper bound for the supports of all patterns containing it. Moreover, the dimensionality of the itemsets that contain I_{cond} is at most $dim(I_{cond}) + l$, where l

is the number of items above the items in in the header table of the FP-Growth. The μ function helps to find the strong and weak clusters. Among the strong clusters, the one with highest μ score is prouced as the final cluster and the data in the other clusters are added back to the database for generation of the next clusters. The process is repeated till the now data point is left in the database. The final data points which are not part of any cluster is treated as outliers.

$$\mu(a, b) = a * (1/\beta) * b \tag{3.1}$$

where, a is support of the frequent pattern, $\beta \in (0, 1]$ reflects the importance of the projection, and b is the number of items in the frequent pattern.

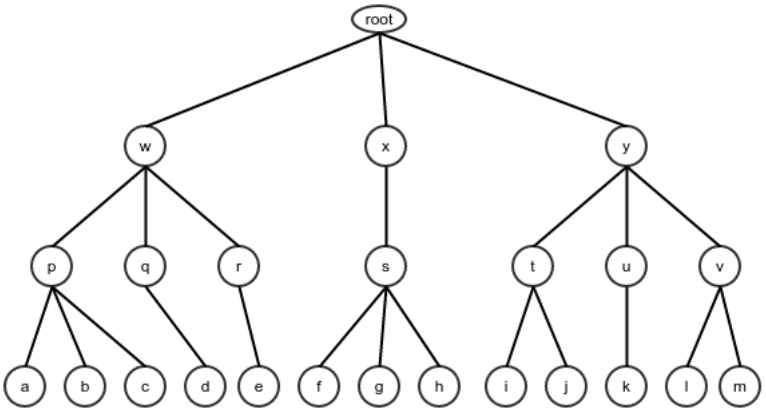

FIGURE 3.1: A Concept hierarchy

3.2 Diversity/DiverseRank

3.2.1 About Concept Hierarchies

A pattern contains data items. A concept hierarchy is a tree in which the data items are organized in a hierarchical manner. In this tree, all the leaf nodes are the items, the internal nodes represent the categories and the top node represents the root. The root could be a virtual node. Figure 3.1 is an example of the concept hierarchy.

Let C be a concept hierarchy. A node in C may be an item, category or root. The height of the root node is 0. Let n be a node in C. The height of n, is denoted as h(n), is equal to the number of edges on the path from root to n.

The concept hierarchies can be balanced and unbalanced. In the balanced concept hierarchy having height h has the same number of levels. The items at the given height are said to be at the same level. In an unbalanced concept hierarchy, the height of at least one of the leaf level node is different from the height of other leaf level nodes. The height of unbalanced concept hierarchy is equal to the height of the leaf level node having maximum height. In concept hierarchy C, all the lower-level nodes, except the root, are mapped to the immediate higher level nodes. We consider the concept hierarchies in which a lower level node is mapped to only one higher level node.

3.2.2 Diversity

The diversity of a pattern is based on the category of the items within it. If the items of a pattern are mapped to the same/few categories in a concept hierarchy, we consider that the pattern has low diversity. Relatively, if the items are mapped to multiple categories, we consider that pattern to be more diverse.

3.2.2.1 Initial Approach

For a given pattern diversity/DiverseRank is assigned based on the merging behavior in the corresponding concept hierarchy. If the pattern merges into few higher level categories quickly, it has low diversity/DiverseRank. Otherwise, if the pattern merges into one or a few high level categories slowly, it has relatively high diversity/DiverseRank value.

As an example, consider the concept hierarchy in Figure 3.1. For the pattern {a, b} the items a and b are mapped to the next level category p. In this case, the merging occurs quickly. For the pattern {a, d}, the item a is mapped to category p while item d is mapped to category q. Further, both th categories p and q are mapped to the category w. We say that the pattern {a, d} is more diverse than the pattern {a, b} as the merging is relatively slow in case of {a, d} as compared to {a, b}. Consider the pattern {a, i} which is relatively more diverse than the pattern {a, d} as both items merges at the root. The merging of {a, i} occurs slowly as comapred to {a, d}

Computing the Diversity of Patterns

We explain the process of calculating diverse rank of the pattern, called DRank, proposed in [[14], [15]], the balanced and unbalanced pattern as follows.

We extract the projection of Concept Hierarchy for Y(pattern) to compute the diversity.

Projection of Concept Hierarchy for $Y(P(Y/C))$: Let Y be P and C be concept hierarchy. The $P(Y/C)$ is the projection of C for Y which contains the portion of C. All the nodes and edges exists in the paths of the items of Y to the root, along with the items and the root, are included in $P(Y/C)$. The projection $P(Y/C)$ is a tree which represents a concept hierarchy concerning to the pattern Y.

Given two patterns of the same length, different merging behavior can be realized, if we observe how the items in the patterns are mapped to higher level nodes. That is, one pattern may quickly merge to few higher level items within few levels and the other patterns may merge to few higher level items by crossing more number of levels. By capturing the process of merging, we define the notion of diverse rank (drank). So, $drank(Y)$ is calculated by capturing how the items are merged from leaf-level to root in $P(Y/C)$. It can be observed that a given item set maps from the leaf level to the root level through a merging process by crossing intermediate levels. At a given level, several lower level items/categories are merged into the corresponding higher level categories.

Two notions are employed to compute the diversity of pattern: Merging Factor (MF), Level Factor (LF) and Adjustment factor (AF).

We explain about MF after presenting the notion of generalized pattern.

Generalized Pattern ($GP(Y, l, P(Y/C))$): Let Y be a pattern, h be the height of $P(Y/C)$ and l be an integer. The $GP(Y, l, P(Y/C))$ indicates the GP of Y at level l in $P(Y/C)$. Assume that the $GP(Y, l+1, P(Y/C))$ is given. The $GP(Y, l, P(Y/C))$ is calculated based on the GP of Y at level $(l+1)$. The $GP(Y, l, P(Y/C))$ is obtained by replacing every item at level $(l+1)$ in $GP(Y, l+1, P(Y/C))$ with its corresponding parent at the level l with duplicates removed, if any.

The notion of merging factor at level l is defined as follows.

Merging factor ($MF(Y, l, P(Y/C))$): Let Y be pattern and l be the level. The merging factor indicates how the items of a pattern merge from the level l+1 to the level $l(0l < h)$. If there is no change, the $MF(Y, l)$ is 1. If all items merges to one node, the $MF(X, l)$ value equals to 0. So, the MF value at the level l is denoted by $MF(Y, l, P(Y/C))$ which is equal to the ratio of the number of nodes in $(GP(Y, l, P(Y/C)1)$ to the number of nodes in $(GP(Y, l+1, P(Y/C)1)$.

$$MF(Y, l, P(Y/C)) = \frac{|GP(Y, l, P(Y/C)| - 1}{|GP(Y, l+1, P(Y/C)))| - 1} \tag{3.2}$$

We now define the notion of level factor to determine the contribution of nodes at the given level.

Level Factor ($LF(l, P(Y/C))$: For a given $P(Y/C)$, h be the height of $P(Y/C) = 0, 1$. Let l be such that $1l(h1)$. The LF value of $P(Y/C)$ at level l indicates the contribution of nodes at level l to $DRank$. We can assign equal, linear or exponential weights to each level. Here, we provide a formula which assigns the weight to the level such that the weight is in proportion the level number.

$$LF(l, P(Y/C)) = \frac{2 * (h - l)}{h * (h - 1)} \tag{3.3}$$

Adjustment factor $AF(Y, l, P(Y/C))$:Let Y be pattern and l be the level The Adjustment Factor (AF) at level l helps in reducing the drank by measuring the contribution of dummy edges/nodes relative to the original edges/nodes at the level l. The AF for a pattern Y at a level l should depend on the ratio of number of real edges formed with the children of the real nodes in $P(Y/E)$ versus total number of edges formed with the children of real and dummy nodes at l in $P(Y/E)$. The value of AF at a given height should lie between 0 and 1. If the number of real edges is equals to zero, AF is 0. If the pattern at the given level does not contain dummy nodes/edges, the value of AF becomes 1. Note that the AF value is not defined at the leaf level nodes as children do not exist. The AF for Y at level l is denoted as $AF(Y, l, P(Y/E))$ and is calculated by the following formula. The value for AF is less than 1 when the pattern is unbalanced otherwise it returns 1.

$$AF(Y, l, P(Y/C)) = \frac{\#\text{ of Real Edges of } P(Y, l, P(Y/E))}{\#\text{ of Total Edges of } P(Y, l, P(Y/E))} \tag{3.4}$$

where numerator is the number of edges formed with the children of the real nodes and denominator is the number of edges formed with the children of both real and dummy nodes at the level l in $P(Y/E)$.

The approach to compute drank of Y is as follows. We convert the concept hierarchy to extended concept hierarchy called, "extended concept hierarchy" by adding dummy nodes and edges. Next, we adjust the drank in accordance with the number of dummy nodes and edges using the notion called adjustment factor. So, the drank of Y is relative to the drank of the same pattern computed by considering all of its items are at the leaf level of the extended concept hierarchy. The dummy nodes and edges are added when the pattern is unbalanced.

Diverse rank of a frequent pattern $Y(drank(Y))$: Let Y be the pattern and C be the unbalanced concept hierarchy of height h. The drank of Y , denoted by $drank(Y)$, is given by the following equation:

$$drank(Y, C) = [MF(Y, l, P(Y/E))AF(Y, l, P(Y/E))] * LF(l, P(Y/E)) \tag{3.5}$$

where, h is the height of the $P(P/E)$, E is the extended unbalanced concept hierarchy, $MF(Y, l, P(Y/E))$ is the MF of Y at level l, $LF(l, P(Y/E))$ is the LF at level l and $AF(Y, l, P(Y/E))$ is the AF of Y at level l.

3.2.2.2 Improved Approach

We found some flaws in the initial approach, which were corrected in the new definition, which is described below.

To calculate diversity for a pattern, we consider subtree of the hierarchy that has only those leaf nodes that are present in the pattern. All nodes that don't connect to/are not the leaf nodes present in the pattern are removed. We call this tree e_1. The number of edges in such a tree, minus the minimum number of edges needed to form a tree of the same height as the hierarchy, with only those values as leaf nodes that appear in e_1 is the diversity of a pattern.

To normalize it we divide it with the difference of number of edges if the entire pattern merges at $root(E)$ and minimum number of edges needed to form a tree of same heightas the corresponding concept hierarchy whose number of leaf nodes is equal to number of distinct items in the original pattern.

$$dr(Y, C) = \frac{e' - e}{E - e} \tag{3.6}$$

As an example, consider the concept hierarchy in Figure 3.1.

- For the pattern {a,b}, E=6, e'=4 and e=4. So diversty=0.0
- For the pattern {a,d}, E=6, e'=5 and e=4. So diversty=0.5
- For the pattern {a,i}, E=6, e'=6 and e=4. So diversty=1.0

Chapter 4

Proposed Approach

Our approach uses Diversity to refine each cluster as it is formed. The MinClus algorithm generates clusters iteratively using a *Frequent Pattern* with high support. Right after a cluster is formed, we try to validate the membership of each element in the newly formed cluster. Any element that is deemed to be different from the rest of the cluster is put back into the dataset, and the MinClus algorithm will consider it while forming newer clusters in the future. Hence, an element is added to the cluster which is best suited for it. If a suitable cluster cannot be found for the element, it is listed as a singleton cluster.

The rationale and key idea behind using diversity is that all elements of the cluster will have similar diversity values with respect to the Concept Hierarchy. The concept hierarchy captures domain knowledge of the dataset and represents it in a tree-like form. Diversity is able to identify how different a data point is from other points in the cluster. If the data point is too different (exceeding a threshold), it clearly does not belong to the current cluster and is labelled as a non-member. All non-members are put back into the list of data points that are yet to be clustered.

For the purposes of refining, let us consider a cluster C containing n data points $\{o_1, o_2, \ldots, o_n\}$. We use diversity to determine the membership of o_i. The array containing the diversity values for each of the cluster elements is denoted as $\{d_1, d_2, \ldots, d_n\}$. Two approaches for this are proposed.

4.1 Diverse Square-Root Distance

We define a measure called *Diverse Square Root Distance (dsrd(k))* as:

$$dsrd(k) = \sqrt{\sum_{j=1}^{n} (d_k - d_j)^2} \tag{4.1}$$

$dsrd(k)$ represents the distance of the k^{th} element from the rest of the cluster. The basic idea is simple — remove all elements that are sufficiently distant from the rest of the cluster.

Algorithm 1: How to write algorithms

Data: C, D, δ

C is the current cluster;

D contains elements to be clustered;

δ is a user-defined threshold;

Result: Refined C

for $i \leftarrow 1$ *to* n **do**

  if $dsrd(i) > \delta$ **then**

    Remove i^{th} element from C;

  else

    Keep i^{th} element in C;

  end

end

return C;

4.2 Binning

For this approach, we sort the data elements in the cluster in increasing order, according to their diversity values. Henceforth, we will assume that the cluster C — $\{o_1, o_2, \ldots, o_n\}$ is in the increasing order of the elements' diversity values.

The idea behind this approach is similar - a cluster should consist of closely related diversity values. We divide the cluster into bins — given a threshold δ, we insert a partition at the point where absolute difference of two diversity values is greater than δ. Thus, if k such partitions are inserted, they divide the cluster into $k+1$ bins. We choose the largest of the bins formed as the representation of the current cluster and remove

all other bins (all elements of the removed bins will be considered by the algorithm for future clusters.

Algorithm 2: How to write algorithms

Data: C, D, δ

C is the current cluster;

D contains elements to be clustered;

δ is a user-defined threshold;

Result: Refined C

bins $\leftarrow \{\}$;

curBin $\leftarrow \{o_1\}$;

for $i \leftarrow 2$ *to* n **do**

  if $(d_i - d_{i-1}) > \delta$ **then**

    Remove i^{th} element from C;

    Add $currentBin$ to bins;

    $currentBin \leftarrow \{o_i\}$;

  else

    Add o_i to $currentBin$;

  end

end

Add $currentBin$ to $bins$;

Return longest bin from $bins$;

4.3 Overall Approach

1. **Convert dataset into transactional form**

   This involves dividing columns/dimensions with numerical values into discrete classes. For example, let's say we have a column/dimension called Age with values ranging from 1 to 70. We can replace these values with Age_{low} ($Age \leq 20$), Age_{mid} ($20 < Age \geq 50$) and Age_{high} ($Age > 50$).

   This is required since Frequent-Pattern based mining algorithms search for exact labels, and most often, relationships between data points depend on range of numerical values, rather than the exact value.

2. **Generate Frequent-Patterns**

   Various algorithms, including the classic Apriori [3] and FP-Growth[4] algorithms exist for this purpose. We use the Apriori algorithm in our implementation.

3. **Form a cluster using MinClus**

   The approach is described in Chapter 3.

4. **Refine cluster using one of the approaches.**
   Approaches described above.

5. **Repeat steps 2 to 4 until no more data points are left**

6. **[Optional] Merge weak clusters**
   The cluster-formation algorithm of MinClus does not know how many clusters need to be generated, and usually creates more clusters than the natural clusters. If the number of clusters is known, we can merge the clusters until the desired number is reached.

   The merging step is applied only when the user wants at most k clusters in the result. In this case, the strong clusters are merged in an agglomerative way until k clusters remain. Given clusters C_x and C_y , the merged cluster is $C_x \cup C_y$, its subspace is $D_x \cap D_y$, its spread is $R(C_x \cup C_y)$ and its μ value $\mu(|C_x \cup C_y|, |D_x \cap D_y|)$. A good cluster should have small spread and large value (i.e., large subspace), so we use both measures to determine the next pair to merge. We consider two rankings of the cluster pairs; one with respect to spread and one with respect to value. Then the pair with the highest sum of ranks in both orderings is merged.

   This process is repeated until only k clusters are left.

Chapter 5

Experiments

5.1 Work Done

- We studied relevant literature, as mentioned in the references
- We understood and implemented the Apriori Algorithm and FP-Growth algorithms in Python
- We understood and implemented both the implementions of Diversity in Python
- We implemented the entire MinClus clustering suite with support for merging
- We proposed and implemented six Outlier detection/Cluster refining methods
- We conducted extensive testing of our approach and that of the original MinClus approach, results of which are presented below.

5.2 Evaluation Metric

We view Clustering as a series of decisions - for each pair of items in the dataset, whether to put them in the same cluster or in a different cluster.

A true positive (TP) decision assigns two similar documents to the same cluster, a true negative (TN) decision assigns two dissimilar documents to different clus- ters. There are two types of errors we can commit. A (FP) decision assigns two dissimilar documents to the same clus- ter. A (FN) decision assigns two similar documents to dif- ferent clusters.

5.3 Results with Diversity (original approach)

5.3.1 Without merging

Table 5.1: Accuracy

Dataset	MinClus	DSRD	Binning
Iris	82.20	85.58	85.58
Seed	79.11	78.96	78.96
Zoo	88.33	88.55	81.18
Water	60.23	62.28	60.24

Table 5.2: Precision

Dataset	MinClus	DSRD	Binning
Iris	97.50	98.28	98.28
Seed	90.62	89.03	89.03
Zoo	94.02	91.28	88.94
Water	36.32	39.07	36.60

Table 5.3: Recall

Dataset	MinClus	DSRD	Binning
Iris	47.84	57.76	57.76
Seed	41.65	42.05	42.05
Zoo	55.03	57.96	24.89
Water	19.61	15.79	20.09

5.3.2 With merging

Table 5.4: Accuracy

Dataset	MinClus	DSRD	Binning
Iris	82.78	91.80	91.80
Seed	82.51	83.21	83.21
Zoo	91.02	94.65	90.47
Water	60.29	59.80	60.50

Table 5.5: Precision

Dataset	MinClus	DSRD	Binning
Iris	75.89	87.38	87.38
Seed	78.72	79.86	79.86
Zoo	90.95	90.02	81.06
Water	36.94	36.70	39.38

Table 5.6: Recall

Dataset	MinClus	DSRD	Binning
Iris	70.85	88.11	88.11
Seed	65.13	66.35	66.35
Zoo	69.61	87.45	78.82
Water	20.63	22.06	25.90

5.4 Results with Diversity (improved approach)

5.4.1 Without merging

Table 5.7: Accuracy

Dataset	MinClus	DSRD	Binning
Iris	82.20	82.43	77.54
Seed	79.11	79.96	75.06
Zoo	88.33	88.33	85.65
Water	60.23	63.35	60.27

Table 5.8: Precision

Dataset	MinClus	DSRD	Binning
Iris	97.50	98.74	98.34
Seed	90.62	95.76	91.00
Zoo	94.02	94.02	96.79
Water	36.32	39.96	36.37

Table 5.9: Recall

Dataset	MinClus	DSRD	Binning
Iris	47.84	47.89	33.17
Seed	41.65	41.74	27.93
Zoo	55.03	55.03	41.75
Water	19.61	11.43	19.53

5.4.2 With merging

Table 5.10: Accuracy

Dataset	MinClus	DSRD	Binning
Iris	82.78	90.94	93.24
Seed	82.51	85.00	83.41
Zoo	91.02	91.96	95.78
Water	60.29	62.05	60.18

Table 5.11: Precision

Dataset	MinClus	DSRD	Binning
Iris	75.89	86.21	89.82
Seed	78.72	77.50	80.45
Zoo	90.95	86.58	93.10
Water	36.94	38.86	36.71

Table 5.12: Recall

Dataset	MinClus	DSRD	Binning
Iris	70.85	86.69	89.92
Seed	65.13	77.51	66.38
Zoo	69.61	78.82	89.08
Water	20.63	16.57	20.59

Appendix A

Glossary and Basic Concepts

Frequent Pattern

Given transaction consisting of records of the form $\{A, B, C, \dots\}$ — a Frequent Pattern is the subset of some record that has a support $\geq minSupport$ (i.e. frequently appears as a part of many records).

Support

Support of a pattern is defined as

$$\frac{\text{Number of items a pattern is a subset of}}{\text{Total number of items}}$$

Apriori Algorithm

An algorithm for mining frequent patterns - inefficient but relatively simpler

FP-Growth Algorithm

Faster algorithm for mining frequent patterns - has a relatively very small memory footprint and takes only two passes over the entire dataset (as opposed to n passes taken by Apriori, where n is the dimensionality of the dataset).

Bibliography

[1] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. *SIGMOD* Rec., 28(2):6172, June 1999.

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. *SIGMOD* Rec., 29(2):7081, May 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD Rec., 27(2):94105, June 1998.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. *In Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB 94, pages 487499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[5] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? *In Proceedings of the 7th International Conference on Database Theory*, ICDT 99, pages 217235, London, UK, UK, 1999. Springer-Verlag.

[6] C. Faloutsos and K.-I. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD* Rec., 24(2):163174, May 1995.

[7] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: Current status and future directions. *Data Min. Knowl. Discov.*, 15(1):5586, 2007.

[8] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD* Rec., 29(2):112, May 2000.

[9] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? *In Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB 00, pages 506515, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[10] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264323, 1999.

[11] G. Moise, A. Zimek, P. Kruger, H.-P. Kriegel, and J. Sander. Subspace and projected clustering: Experimental evaluation and analysis. *Knowl. Inf. Syst.*, 21(3):299326, 2009.

[12] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD 02, pages 418427, New York, NY, USA, 2002. ACM.

[13] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846850, 1971.

[14] S. Srivastava, R. U. Kiran, and P. K. Reddy. Discovering diverse-frequent patterns in transactional databases. In *Proceedings of the 17th International Conference on Management of Data*, COMAD 11, pages 14:114:10. Computer Society of India, 2011.

[15] M. K. Swamy, P. K. Reddy, and S. Srivastava. Extracting diverse patterns with unbalanced concept hierarchy. In *Advances in Knowledge Discovery and Data Mining - 18th Pacific-Asia Conference*, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I, pages 1527. Springer-Verlag, 2014.

[16] UCI. Machine Learning Repository, `https://archive.ics.uci.edu/ml/index.html`, 2014 (accessed August, 2014).

[17] R. Xu and D. Wunsch, II. Survey of clustering algorithms. *Trans. Neur. Netw.*, 16(3):645678, 2005.

[18] M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM 03, pages 689692, Washington, DC, USA, 2003. IEEE Computer Society.

[19] M. L. Yiu and N. Mamoulis. Iterative projected clustering by subspace mining. *IEEE Trans. on Knowl. and Data Eng.*, 17(2):176189, 2005.