Unsupervised Context-aware User Preference Mining

Fei Li[1], **Katharina Rasch**[2], **Sanjin Sehic**[1], **Schahram Dustdar**[1] and **Rassul Ayani**[2]
[1] Information Systems Institute
Vienna University of Technology
Vienna, Austria
{li, ssehic, dustdar}@infosys.tuwien.ac.at
[2] School of Information and Communication Technology (ICT),
Royal Institute of Technology (KTH), Sweden
Stockholm, Sweden
{krasch, ayani}@kth.se

Abstract

In pervasive environments, users are situated in rich context and can interact with their surroundings through various services. To improve user experience in such environments, it is essential to find the services that satisfies user preferences in certain context. Thus the suitability of discovered services is highly dependent on how much the context-aware system can understand users' current context and preferred activities. In this paper, we propose an unsupervised learning solution for mining user preferences from the user's past context. To cope with the high dimensionality and heterogeneity of context data, we propose a subspace clustering approach that is able to find user preferences identified by different feature sets. The results of our approach are validated by a series of experiments.

Pervasive environments are rich in context information and services. Users in such environments need services that suit to their current context and preferred activities. As more and more sensors are deployed to collect context information, the task of interpreting context becomes increasingly challenging (Lim and Dey 2010). Consequently, the usefulness of service discovery (Li, Sehic, and Dustdar 2010; Rasch et al. 2011) or service recommendation (Adomavicius et al. 2005) is largely limited by the ability to understand user preference.

In the literature, context attributes used for learning user preferences are chosen by either empirical assumption (Munoz-Organero et al. 2010) or dimension reduction (Krause, Smailagic, and Siewiorek 2006) that renders a small set of features. However, these approaches are infeasible in a broad and ever-growing spectrum of context information. They fail to acknowledge the large variety of features needed to describe different user preferences. Therefore, they often struggle to find services that fulfill user requirements accurately.

Context modeling (Baldauf, Dustdar, and Rosenberg 2007; Strang and Linnhoff-Popien 2004) can define context in an extensible and dynamic way. These models are useful when reasoning based on a priori knowledge, but fall short in acquiring implicit knowledge from past context. For identifying user preferences, most previous work applies supervised learning approaches that are targeted at identifying a set of known classes, usually activities (Ferscha et al. 2004). These approaches, however, are also challenged in a service-rich environment. Because of the highly diverse service invocation possibilities, they require a large amount of training data and a long learning process.

This paper addresses the aforementioned challenges by modeling and analyzing context and services in a high-dimensional data space. Our main contribution is a subspace clustering approach that is specialized to work on high-dimensional context data. In the clustering process, we accommodate for different data types, different densities and context attributes of varying importance. Different to most known solutions for learning from context, our solution is an unsupervised learning process that requires minimal a priori knowledge. The output of clustering is a set of user preferences presented as subspace clusters, which can then be used for service discovery. Our experimental results demonstrate that the proposed approach is able to achieve very good clustering results, and in turn, provide highly reliable understanding of users' preferred context and activities.

The paper is organized as follows: We first give a background introduction to modeling services and preferences in high dimensional context spaces. Next, we discuss the challenges of applying subspace clustering to context data, followed by a detailed presentation of our preference mining approach. Afterwards, the proposed approach is evaluated by series of experiments. The paper concludes with discussions and future work.

1 Background

1.1 Service in Hyperspace Analogue to Context

Hyperspace Analogue to Context (HAC)(Li, Sehic, and Dustdar 2010; Rasch et al. 2011) is a concept to model context as multidimensional space. It effectively captures continuously changing information from various context sources. We first give a short introduction to core concepts in HAC.

- HAC is a space $H =< d_1, d_2...d_n >$, where each dimension d_i denotes a type of context. In HAC, a dimension is the meta data to describe the data type and value set for a specific type of context, e.g. location, time, and status of a device.

- The *context* of an object o is a point $c^o =< v_1, v_2...v_n >$. A change of c^o is considered as the object moving in H.
- A *scope* $S =< D, V >$, where $D \subset H$ is the dimension set that characterizes the scope, and V is the value set allowed for dimensions in D. In a hyperspace of continuous real dimensions, a scope is a hypercube projected on D. We define a context c to be in a scope S as: $c \in S$, when the projection of c on D belongs to V.

Using the concept of scope, we can describe two important aspects in user preference mining: *user preference* and *context-aware service*.

Preferences are the scopes a user would like to be situated in. Preferences may describe common situations such as the user's preferred indoor temperature, but also less common ones, for example a user's habit of watching a specific TV channel every Sunday evening. The goal of preference mining is to suggest relevant services that lead to preferred scopes. In this process, some preferences may be more important to the user than others; for example a "no fire" preference is obviously more important than another preference concerned with comfortable lighting. The significance of a preference is decided by the dimensions involved and the frequency of the described situation. In HAC, each dimension is associated with a predefined weight w_i, where $\sum w_i = 1$.

A context-aware service can be invoked in a certain scope and invoking it will change the user context. Thus, a service $s =< S^{Pre}, S^{Eff} >$ is characterized by two context scopes.

- Precondition $S^{Pre} =< D^{Pre}, V^{Pre} >$ is the invocation condition of service s. When user is in S^{Pre}, service s becomes one possible choice.
- Effect $S^{Eff} =< D^{Eff}, V^{Eff} >$ is the possible context after invoking a service.

If a service s is successfully executed, a transition of user context $c \rightarrow c'$ happens such that $c \in S^{Pre}$ and $c' \in S^{Eff}$. An example service which turns on the oven and heats it up to a desired temperature could be described as having input context $S^{Pre} =< d_{oven} = [off] >$ and $S^{Eff} =< d_{oven} = [on], d_{ovenTemp} = [120 \ldots 250] >$.

1.2 Clustering and context data

Intuitively, a user is comfortable with a situation if he is frequently in similar context. The narrow ranges of some context attributes define the preferable situation. Clustering – as a widely applied approach to partitioning data into sets based on a certain similarity measure – seems to be a natural candidate to abstract user preference from large amounts of context data. However, in the literature, clustering is not a favorable approach to context data analysis (Lim and Dey 2010). One reason is that context data is intrinsically high dimensional. Classic clustering approaches, e.g. kNN (k-Nearest Neighbors), usually rely on various types of distance measures, which are effective on low-dimensional data. But on high-dimensional data the "Curse of Dimensionality" becomes a significant problem because "as dimensionality increases, the distance to the nearest data point approaches the distance to the farthest data point" (Beyer et al. 1999). Consequently, distance measures become ineffective. Dimension reduction techniques, such as Principle Component Analysis (PCA) (Wold 1987), can derive a single reduced set of dimensions for all objects, but at the same time they lose information of locally correlated dimensions. The local relationships between dimensions are of great importance in context data analysis since user preferences are based on different sets of correlated context attributes.

Aimed at high-dimensional data, subspace clustering or projected clustering (Kriegel, Kröger, and Zimek 2009; Parsons, Haque, and Liu 2004) has gathered considerable attention in the last years. The goal of subspace clustering is to find a list of clusters, each a pair $< D, O >$, where D is a set of data attributes[1], and O is a set of data objects. Compared to classic clustering methods, subspace clustering recognizes *local feature correlations* and finds similarities of data with regard to different subsets of dimensions. Subspace clustering is applied to gene expression analysis, product recommendation, text document analysis, etc. We can observe a high accordance between the context scope in HAC and the goal of subspace clustering, prompting us to investigate the potential to apply subspace clustering on context data. However, most applications of subspace clustering use data consisting of homogeneous dimensions, e.g. gene data, which is not the case for context data. The heterogeneity of data poses a strong limitation on choosing an effective clustering method — similarity calculation, essential in many clustering approaches (Ester et al. 1996) and usually based on certain distance measures across multiple dimensions, is not applicable. Context data is heterogeneous in several ways:

- *Dimension semantics.* Each dimension in context data presents unique semantics. The semantics decide the data type of each dimension, which can be numerical or nominal, continuous or discrete. It is not meaningful to measure the distance between two multidimensional points if the dimensions are semantically different. Although we can always define certain distance measure on each dimension, in many cases the measure is subjective and arbitrary, and eventually results in a loss of information.
- *Distribution.* Context data on each dimension is in a specific value domain and of a specific distribution. Even just for numerical dimensions, there is no single normalization method applicable to all of them. Failing to recognize the different distributions between dimensions may result in bad clusters, or in our case, wrong preferences.
- *Significance.* Dimensions are not equally important. One prominent example is that the dimensions indicating emergency or security issues are more important than a comfortable ambiance. The importance of dimensions will directly influence the importance of preferences. When clustering context data to find user preferences, dimensional significance must be taken into consideration.

A sample of context data is presented in Table 1. The sample is an excerpt of a user's weekend life. Each dimension

[1]In this paper we use *attribute* and *dimension* interchangeably.

Object ID	Clock time	Location	In or out	AC	Indoor temperature ($^\circ$C)	TV	Door lock
o_0	07:30	bedroom	in	on	22	off	locked
o_1	09:15	out	out	off	30	off	locked
o_2	10:30	out	out	off	30	off	locked
o_3	11:15	living room	in	on	22	on	locked
o_4	12:30	kitchen	in	on	21	off	locked
o_5	13:45	bedroom	in	on	22	off	locked
o_6	15:00	out	out	off	29	off	locked
o_7	17:00	living room	in	on	21	on	locked
o_8	18:30	kitchen	in	on	22	on	locked
o_9	20:00	living room	in	on	21	on	locked
o_{10}	22:00	bedroom	in	on	22	off	locked

Table 1: Sample context data

presents different semantics, has a different data type and value range. For example, temperature and time are discrete and numerical, while all others are nominal. Some clusters in one dimension can easily be observed. For example, on temperature dimension, $\{o_1, o_2, o_6\}$ is a cluster with temperature around 30°C, and other objects form another cluster around 21°C. We can even observe two subspace clusters. Objects $\{o_1, o_2, o_6\}$ on dimensions of location, AC and temperature suggest that when user is out, the AC is often off. Other objects also suggest another cluster specifying that when the AC is on, the temperature is usually set to around 21°C. We can also observe the association that when the user is in the house, usually the AC is on and the temperature is set to around 21°C. This relationship is, however, not explicitly demonstrated in the sample data because this subspace cluster is obscured by the three different rooms in the location dimension. Some important pattern could be missing if we use sensory data directly as input for subspace clustering. This problem reveals the hierarchical nature of some context, e.g. the three rooms are all indoor. We solve this problem in data preparation by adding feature dimensions. For location, we can add another dimension called *in or out* and set all indoor rooms to value *in*. Similarly, we can describe time by multiple dimensions such as *day of a week* and *time in a day*.

2 Mining user preferences by subspace clustering

Our goal of applying subspace clustering is to find user preferences in large amounts of context data. Given the aforementioned characters of context data, we adopt a framework suggested by FIRES (FIlter REfinement Subspace clustering) (Kriegel et al. 2005), which distinguishes a step of preclustering from generating subspace clusters. In each step, we propose a method for processing heterogeneous context data and the end result of the process are subspace clusters representing user preferences. The most significant advantage of this framework is that it allows to accommodate for dimensional heterogeneity.

In the first step, we need to find 1-dimensional (1D) clusters in each dimension. To this end we propose an efficient, density-based method to deal with highly differing densities among dimensions. In the subspace cluster generation step, the basic idea is to merge 1D clusters from different dimensions based on the data objects they share. We transform the clustering problem to frequent itemset mining. Thus the data in each 1D cluster is regarded as the appearances of one item. We modify the FP-Tree based *FP-Growth* algorithm (Han, Pei, and Yin 2000) for frequent itemset mining. Because the algorithm merges one item each time, we can differentiate dimensions in the process. Furthermore, the algorithm is efficient without costly candidate generation as in the classical a priori approach (Agrawal and Srikant 1994). In contrast to the second step in FIRES, which produces approximations of subspace clusters, our second step produces subspace clusters in a deterministic manner.

2.1 Preclustering

The aim of preclustering is to find 1D clusters hidden in each dimension. The quality of the found 1D clusters has a major influence on the quality of the overall clustering results: 1D clusters that are not found in the preclustering step, will also not be represented in the final clustering result.

Nominal dimensions are already discrete. Each value represented in the data is considered as 1D cluster containing all data objects having that value. For numeric dimensions, further processing is necessary. Each numeric dimension may contain multiple 1D clusters of varying size and local densities plus various noise points that are not part of any cluster. Popular density-based clustering algorithms, e.g. DBScan (Ester et al. 1996), often struggle when clusters with different densities are present in the data. In contrast, our own density-based clustering algorithm Binmerge is able to find clusters with highly varying local densities.

In order to identify hidden clusters, Binmerge makes use of the Local Outlier Factor (LOF) described by Breuning et al. (Breunig et al. 2000) The LOF of a point p is a measure of how likely p can be considered an outlier. If $LOF(p)$ is less than 1.0, then p lies within a cluster. For values higher than 1.0, the likeliness that p is an outlier increases with increasing $LOF(p)$. Binmerge uses this information to identify potential cluster points. The local density of the points in the hidden cluster around p can be approximated using the $\text{k-distance}(p)$, the distance of p to its k-nearest neighbors.

Let U be the set of all data points that are not yet clustered. In the first step of Binmerge, the $\mathrm{k\text{-}distance}(p)$ and the $LOF(p)$ for all $p \in U$ are calculated. Let U_c be the set of all points $q \in U$ with $LOF(q) < 1.0$. Now the following algorithm is performed until U_c is empty: First, select from U_c the point q with the smallest $\mathrm{k\text{-}distance}(q)$. The point q indicates a not yet found cluster whose density can be approximated using $\mathrm{k\text{-}distance}(q)$. Second, in order to find all points belonging to that cluster, the data interval covered in U is divided into consecutive, non-overlapping bins with width equal to $\mathrm{k\text{-}distance}(q)$ and all $p \in U$ are sorted into these bins. Third, the bins are then merged with each other; two non-empty bins can be merged with each other if they are either adjacent or are separated by maximum k empty bins. Bins are merged until no further merging is possible. Fourth, the remaining bin containing the highest number of points is considered a 1D cluster and is added to the result set. All points in this cluster are removed from U and U_c.

Through dividing and merging the data set using different $\mathrm{k\text{-}distances}$, clusters with highly varying densities can be found. It is important to start by finding the cluster with the highest local density, i.e. the lowest $\mathrm{k\text{-}distance}$. Otherwise it could happen that points from a high-density cluster are wrongfully included in an adjacent low-density cluster. Even though multiple iterations through the data set are necessary in Binmerge, the data set will get increasingly smaller since points contained in found clusters are removed.

2.2 Generating subspace clusters

Due to limited space, this section only introduces the general idea of FP-Growth algorithm and details our modification. For more detailed mechanism and rationale of the algorithm, and the methods to create FP-Tree, readers should refer to Han et al. (Han, Pei, and Yin 2000). The result of preclustering is sorted in descending order of *support*, which is the number of data objects in each cluster. The sorted list is called *header table*, denoted as ht. From the header table, an FP-Tree is generated. FP-Tree is an item prefix tree preserving all the frequent itemset information in a concise tree structure. The recursive FP-Growth algorithm merges itemsets by pruning the FP-Tree. The algorithm requires only one user input parameter, the minimal support of the resulting itemset. The original FP-Growth algorithm keeps all possible combinations of dimensions discovered during the clustering process. This results in some redundant clusters, whose information is contained in clusters of higher dimensionality, making it difficult to interpret the results in application scenarios.

When merging 1D clusters into subspace clusters, we generally prefer clusters of higher dimensionality, because they can represent more associations between dimensions. In context-aware applications, clusters of higher dimensionality can characterize user preference more accurately. On the other hand, a cluster containing low number of objects, even with high dimensionality, indicates the association is not frequent enough. Therefore it should be considered a bad cluster. A measure of cluster quality must take into consideration both the number of dimensions as well as the number of objects in the cluster. Since our clustering algorithm is

Algorithm 1 wTreeGrowth

```
 1: initialize
 2:     create FP-Tree $T$ from $ht$
 3:     initialize subspace cluster $C^m = null$
 4: end initialize
 5: procedure wTreeGrowth($T$, $ht$, $C^m$)
 6: $result = C^m$
 7: if only one branch in $T$ then
 8:     for $i = ht.length$ to 1 do
 9:         $C^{m+1} = result.merge(ht[i].C^1)$
10:         if $C^{m+1}.quality > result.quality$ then
11:             $result = C^{m+1}$
12:         end if
13:     end for
14:     if $|result.dimensions| >= minDim$ & $|result.support| >= minSupport$ then
15:         add $result$
16:     end if
17: else
18:     for $i = ht.length$ to 1 do
19:         $C^{m+1} = C^m.merge(ht[i].C^1)$
20:         if $C^{m+1}.quality > C^m.quality$ then
21:             Create new header table $ht'$ based on $C^{m+1}$
22:             if $ht'$ is not $null$ then
23:                 Create new tree $T'$
24:                 wTreeGrowth($T'$, $ht'$, $C^{m+1}$)
25:             else
26:                 $result = C^{m+1}$
27:             end if
28:         end if
29:         if $|result.dimensions| >= minDim$ & $|result.support| >= minSupport$ then
30:             add $result$
31:         end if
32:     end for
33: end if
34: end procedure
```

specialized for context data, the weights of individual context dimensions must also have a substantial influence when calculating cluster quality. We propose a cluster quality defined in (1) to meet all of the above requirements.

$$Quality(C^m) = |C| \times u^{-|D|} \times \prod_{d_i \in D} \frac{1}{1 - w_i} \qquad (1)$$

In (1), C is the object set and D is the dimension set. $|C| \times u^{-|D|}$ stresses the importance of extending dimensionality against reducing the support. $u \in (0, 1)$ is a user input parameter to regulate the importance of dimensionality. $\prod \frac{1}{1-w_i}$ indicates that for context data, attributes are not equal. Thus clusters of different dimension set, even with same support and dimension number, could be valued differently in the process of generating further subspace clusters.

Our modified FP-Growth algorithm based on this quality measure is called *wTreeGrowth*, illustrated in Algorithm 1. In the algorithm, a 1D cluster is denoted as C^1, and correspondingly, a cluster of m dimensions is C^m.

wTreeGrowth is a recursive algorithm that generates subspace clusters of good quality by selectively merging with 1D clusters. $C^{m+1} = C^m.merge(C^1)$ means to add C^1 to the current subspace cluster. The dimensionality of C^m is extended, while the intersection of data objects from C^m and C^1 are kept in the new subspace cluster C^{m+1}. From Line 8 to Line 16, when there is only one branch left in the tree, the algorithm tries to extend the current C^m with as many dimensions as possible. A 1D cluster will be skipped only when it decreases the quality of resulting subspace cluster (Line 10). When multiple branches exist in the tree (Line 18 to Line 32), there are multiple possibilities to extend the current C^m. If the subspace cluster, created by merging with the next header table entry, has higher quality than the current one, the merged cluster is kept and the header table is reduced in the recursive invocation of the algorithm.

3 Experiments and discussion

3.1 Cluster evaluation measures

For evaluating effectiveness of subspace clustering, there is so far no generally agreed metrics, particularly regarding projected clusters on subspaces. In this paper, we adopt the measures proposed recently by Mueller et al. (Müller et al. 2009) All of them are normalized to $[0, 1]$ where 1 is considered the best. We explain shortly the meaning of each measure and our modifications on some of them with respect to the character of context data. For more details, readers should refer to the original publications.

The first two measures evaluate results with regard to data objects, but do not consider the projection to subspaces.

Entropy: The entropy measures the homogeneity of found clusters with respect to hidden clusters. The more similar found and hidden clusters are, the better the entropy will be.

F1: This measure evaluates how well hidden clusters are presented. A found cluster should cover as many objects as possible in some hidden clusters, and as little as possible in other clusters.

Following measures reveal clustering results at a finer grain with considerations to dimensional projections. The basic elements under inspection are *subobjects*, which are the projection of each data object on each dimension, denoted as o_{ij} where i is the dimension id and j is the object id.

Relative Nonintersecting Area (RNIA): RNIA measures the overall coverage of hidden subobjects with respect to found clusters. Let all hidden subobjects be SO^h, and all found subobjects be SO^f. Let U be the union of these two subobject sets and I be their intersection. $RNIA = \frac{U-I}{U}$ and since $U \supset I$, $RNIA = \frac{|U|-|I|}{|U|}$.

Clustering Error (CE): CE extends RNIA by considering one-to-one mappings between hidden and found clusters. CE measures mapping of each found cluster to at most one hidden cluster and vice versa. For each mapping the intersection of subobjects is determined and the union of all these intersections is $\bar{I}$. We can calculate the CE value by replacing I in RNIA with $\bar{I}$.

Since RNIA and CE do not consider the dimensions of subspace clusters, we modified them to incorporate weights of dimensions in their evaluation. We call these modified measures *wRNIA* and *wCE*. Each subobject is associated with weight of its dimension. The modification has following effect on measures: more precise clustering results w.r.t. important dimensions is, better the measures are. Given a set of subobjects SO, we define $wsize(SO)$ in (2). wRNIA and wCE measures are the result of replacing each $|SO|$ in RNIA and CE respectively with $wsize$.

$$wsize(SO) = \sum_{o_{ij} \in SO} \frac{1}{1 - w_i} \tag{2}$$

3.2 Synthetic data generation

In order to test our algorithms, we generated synthetic data following the method described in (Aggarwal et al. 1999). Since the original method supports only fixed-size numeric dimensions, small adaptions are necessary. We generate nominal dimensions by creating a finite set between 5 and 20 random nominal values. Numeric dimensions are generated by choosing 0 as the minimum value of the dimension and any value between 100 and 1000 as the maximum value. Numeric cluster points are drawn from a Gaussian distribution as in the original method, whereas nominal values are assigned a single nominal value set for this cluster.

We generate clusters containing on average 200 points, which is the cluster size at which our algorithm stabilizes. The noise ratio is set to 0.2, meaning that 20% of all points are not contained in any cluster. In a typical pervasive environment, e.g. smart home, we expect that the noise ratio will be even smaller, since nearly every data point is indicative of a user preference, even if it is just on one dimension like indoor temperature. The number of overall dimensions is set to 10, and the percentage of nominal dimensions is 60%. We expect that in pervasive environments a majority of dimensions are nominal, e.g. dimensions describing the status of devices, and only some dimensions, such as temperature, are numerical. We hide 10 clusters, which is equivalent to detecting 10 user preferences. Each cluster contains on average three dimensions, but as we will show later, the quality of our results is not influenced by the number of cluster dimensions.

3.3 Estimating parameters

Different algorithm parameters have been introduced in the previous sections. The minimum support of a cluster describes the number of data points constituting a minimal cluster. We have achieved good results in all of our experiments by setting minSupport to 0.025, which means that only clusters that contain at least 2.5% of all data points are considered. The minimum support is manually set to 10 if it was calculated as a smaller value, in order to avoid that arbitrary noise points create clusters.

The parameter k influences the behaviour of Binmerge; it determines the number of neighbours that are included when calculating the LOF of a point and controls the process of merging bins. A very good approximation of k is the minimal mimSupport of a cluster, in our case 10. When setting k like this, the LOF calculation will take into account exactly those neighbours that could constitute a minimal cluster.

The third parameter is u, which regulates the importance of the total number of dimensions when calculating the cluster quality. We have found that $u = 0.6$ yields the best results in our experiments.

3.4 Experimental results

All of the experiments were performed on a desktop PC with an Intel Core 2 Duo CPU with 3 GHz and 4 GB RAM running Linux, and were run for 25 replications.

Figure 1 shows how the average size of clusters influences the results of our clustering algorithm. A steady increase of the quality of the clustering results can be seen for small cluster sizes with less than 200 points per cluster. From 200 points the results stabilize and high quality clusters are found. A user's comfort situations can thus be reliably found if there were approximately 200 data points indicative of this situation recorded. Since each change in the users context constitutes a new data point, this limit is reached quickly, especially for common user preferences such as indoor temperature and lighting.

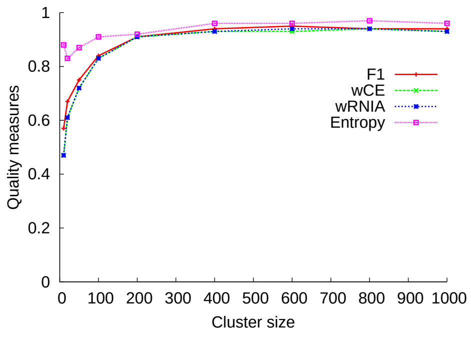

Figure 1: Influence of the average cluster size

It can also be seen from Figure 1 that the stabilized entropy is constantly higher than 0.9, meaning that the hidden clusters are found with minimal fragmentation into multiple clusters. This assumption is further supported by the fact that the values for wCE and wRNIA are very similar for all results, suggesting that our algorithm is very successful in finding exactly one cluster for each hidden cluster.

The values for wCE and wRNIA are however lower than the entropy, indicating that some data objects are not assigned to the correct clusters. This is typically the case when the 1D clustering of numeric dimensions fails to correctly cluster some values. Figure 2 shows consequently, that the wRNIA and wCE increase if there are less numerical and more nominal dimensions present.

In pervasive environments, dozens or even hundreds of different context dimensions may be present. Figure 3 shows how our algorithm also fares well when the overall number of dimensions is increased. It can be seen, that the cluster qualities are very stable even with increasing dimension count. Especially for dimension numbers bigger than 30, many dimensions will not be used in any clusters. The results prove that our algorithm is very successful in separating these noise dimensions and correctly identifying the

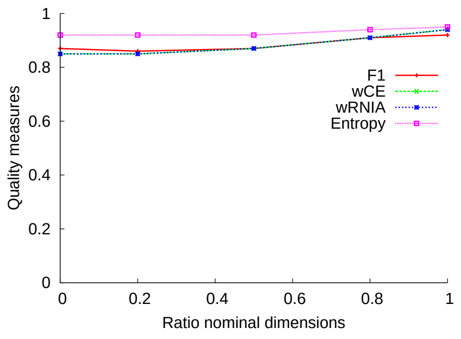

Figure 2: Influence of the percentage of nominal dimensions

cluster dimensions. We have also found that results for a varying average number of cluster dimensions is very stable; high-quality clusters are found both for low and for high-dimensional clusters.

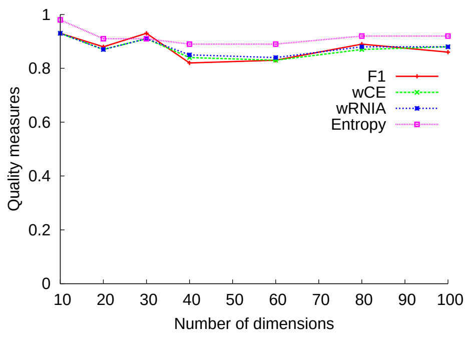

Figure 3: Influence of the number of overall dimensions

Figure 4 shows that our algorithm is most successful if between 4 and 10 clusters are hidden in the data. For higher cluster numbers, all measures except for the entropy start to decrease. This indicates that the clusters are still identified correctly, however the clustering algorithm is becoming less effective in assigning all objects to their correct clusters. This happens because the 1D numeric clustering algorithm degrades with an increasing number of clusters on each dimension; and indeed we can see much more stable results when the number of nominal dimensions is increased.

We also looked at the influence of the ratio of unclustered noise points on the clustering results. Our algorithm achieves very stable clustering results for up to 40% noise points. With more noise points, the number of overall points and thereby also the minimal support for each cluster has increased so drastically, that the minimal support is bigger than many of the actual hidden clusters, so only some clusters can be found. However, as pointed out earlier, the expected noise ratio in the scenarios we envision is much lower, since nearly all data points are indicative of a user preferences

Finally we were interested in the execution time of our algorithm. Figure 4 shows the execution times for the varying cluster sizes already represented in Figure 5. The total

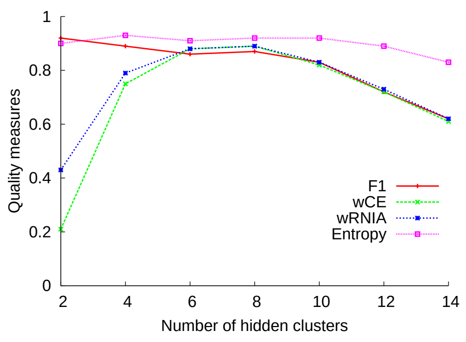

Figure 4: Influence of the number of clusters

number of data points to be clustered increases from 1250 points for a cluster size of 100 to 12500 points for a cluster size of 1000. As expected, the total execution time increases with the growing number of data points. Nevertheless, even for 12500 points, the clustering finishes under 5 seconds. As the clustering method is meant to be used offline, the performance is acceptable.

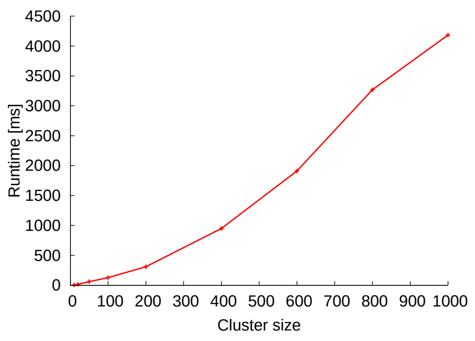

Figure 5: Influence of cluster size on runtime

4 Related work

To the best of our knowledge, our work is the first to use subspace clustering to analyze context data. By exploiting subspace clustering on high-dimensional, heterogeneous data, we can discover hidden preferences in context data with minimal user input.

Our user preference mining approach distinguishes itself substantially from other subspace clustering approaches. FIRES (Kriegel et al. 2005) is a generic subspace mining framework that features a 1D clustering step. We adopt this framework but devised a completely different dimension merging approach for context data. In addition, the cluster quality in FIRES is $\sqrt{|C|} * |D|$, which we found much less effective on context data than our cluster quality. HSM (Heterogeneous Subspace Mining) (Müller, Assent, and Seidl 2009) is the first known approach that tackles heterogeneous data. HSM uses SCY-tree (Subspace Clusters with in-process-removal of redundancY) (Assent et al. 2008) to perform dimension merging. The approach mixes the steps of 1D clustering and dimension merging, thus the clustering algorithm is sensitive to the order of dimensions when building the SCY-tree, which is not the case in our FP-Tree based approach. Furthermore, HSM only deals with data type heterogeneity. In contrast, our user preference mining approach addresses the heterogeneity of attribute distribution using a dedicated 1D clustering approach for data of differing densities. Also different to existing approaches, we incorporate a quality measure in the mining process to emphasize important dimensions.

Service discovery in pervasive environments usually makes use of context to define a scope of search for suitable services. Bellavista et al. (Bellavista et al. 2006) proposed a user-centric service view, and Park et al. (Kyung-Lang Park, Yoon, and Shin-Dug Kim 2009) uses the concept of virtual personal space. Both of them resemble the scopes of user preferences, but our user preferences can be much more fined grained than their service discovery scopes and require no pre-configured preference description. In Li et al. (Li, Sehic, and Dustdar 2010) and Rasch et al. (Rasch et al. 2011), we have presented the concepts of HAC, and based on it, we proposed a context-driven service discovery framework. In that work, we assumed user preferences are predefined input for service discovery algorithm. Our user preference mining approach can be complementary to previous service discovery work. But most importantly by modeling service in the same data space, service discovery is transformed to evaluating the spacial relationships between preferences, service effects and user context.

5 Conclusion

In this paper, we introduced a model to associate service and context coherently in one data space. Based on the model, we propose a subspace clustering approach on high-dimensional and heterogeneous context data to find meaningful user preferences. The effectiveness of the proposed approach is proved experiments on synthetic data.

Our future work will be mainly focused on improving and validating our approach in field tests. More specifically, we will carry out extensive experiments of the user preference mining algorithm on real-world data. As pervasive computing becomes an essential technology paradigm in our daily life, the potentials of our approach are beyond context-aware activity recognition and service discovery. We will continue to develop the data model and algorithm for sensory data, which can be found in many application scenarios like e-health, wearable computing and smart buildings.

Acknowledgments

This work is sponsored by Pacific Controls Cloud Computing Lab (PC^{3}L), a joint lab between Pacific Controls L.L.C., Dubai, United Arab Emirates and the Distributed Systems Group of the Vienna University of Technology.

References

Adomavicius, G.; Sankaranarayanan, R.; Sen, S.; and Tuzhilin, A. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems* 23(1):103–145.

Aggarwal, C. C.; Procopiuc, C.; Wolf, J. L.; Yu, P. S.; and Park, J. S. 1999. A framework for finding projected clusters in high dimensional spaces. In *ACM SIGMOD Conference*, 61–72.

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, 487–499. Citeseer.

Assent, I.; Krieger, R.; Muller, E.; and Seidl, T. 2008. INSCY: Indexing Subspace Clusters with In-Process-Removal of Redundancy. 719–724.

Baldauf, M.; Dustdar, S.; and Rosenberg, F. 2007. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing* 2(4):263–277.

Bellavista, P.; Corradi, A.; Montanari, R.; and Toninelli, A. 2006. Context-aware semantic discovery for next generation mobile systems. *Communications Magazine, IEEE* 44(9):62–71.

Beyer, K.; Goldstein, J.; Ramakrishnan, R.; and Shaft, U. 1999. When Is "Nearest Neighbor" Meaningful? In Beeri, C., and Buneman, P., eds., *Database Theory — ICDT'99*, volume 1540 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 217–235.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. Lof: identifying density-based local outliers. *SIGMOD Rec.* 29:93–104.

Ester, M.; Kriegel, H.-P.; Sander, J.; and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E.; Han, J.; and Fayyad, U. M., eds., *Second International Conference on Knowledge Discovery and Data Mining*, 226–231. AAAI Press.

Ferscha, A.; Mattern, F.; Tapia, E.; Intille, S.; and Larson, K. 2004. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. 3001:158–175–175.

Han, J.; Pei, J.; and Yin, Y. 2000. Mining frequent patterns without candidate generation. *ACM SIGMOD Record* 29(2):1–12.

Krause, A.; Smailagic, A.; and Siewiorek, D. P. 2006. Context-Aware Mobile Computing: Learning Context-Dependent Personal Preferences from a Wearable Sensor Array. *IEEE Transactions on Mobile Computing* 5(2):113–127.

Kriegel, H.-P.; Kroger, P.; Renz, M.; and Wurst, S. 2005. A generic framework for efficient subspace clustering of high-dimensional data. 8 pp.

Kriegel, H.-P.; Kröger, P.; and Zimek, A. 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data* 3(1):1–58.

Kyung-Lang Park; Yoon, U.; and Shin-Dug Kim. 2009. Personalized Service Discovery in Ubiquitous Computing Environments. *Pervasive Computing, IEEE* 8(1):58–65.

Li, F.; Sehic, S.; and Dustdar, S. 2010. COPAL: An adaptive approach to context provisioning. In *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, 286–293. IEEE.

Lim, B. Y., and Dey, A. K. 2010. *Toolkit to support intelligibility in context-aware applications*. New York, New York, USA: ACM Press.

Müller, E.; Assent, I.; and Seidl, T. 2009. *HSM: Heterogeneous Subspace Mining in High Dimensional Data*, volume 5566 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 497–516.

Müller, E.; Günnemann, S.; Assent, I.; and Seidl, T. 2009. Evaluating clustering in subspace projections of high dimensional data. *Proc. VLDB Endow.* 2:1270–1281.

Munoz-Organero, M.; Ramíez-González, G.; Munoz-Merino, P.; and Delgado Kloos, C. 2010. A Collaborative Recommender System Based on Space-Time Similarities. *Pervasive Computing, IEEE* 9(3):81–87.

Parsons, L.; Haque, E.; and Liu, H. 2004. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl.* 6(1):90–105.

Rasch, K.; Li, F.; Sehic, S.; Ayani, R.; and Dustdar, S. 2011. Context-driven personalized service discovery in pervasive environments. *World Wide Web* 1–25–25.

Strang, T., and Linnhoff-Popien, C. 2004. A context modeling survey.

Wold, S. 1987. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2(1-3):37–52.