## Chapter 3

# Behavioral Clustering

Behavioral clustering is a broad term that refers to the task of automatically labeling and classifying user behavior. In a general context, clustering allows to identify sub-populations in a dataset, so that they can be represented by more compact structures for, e.g., classification and retrieval purposes. To this end, implicit interaction can provide current clustering methods with additional information. For instance, on the Web, clustering is usually deployed by using a single data source, which is often browsing usage information derived from server access logs. However, when it comes to getting deep information about user behavior, this representation is inadequate in such a dynamic environment.

In this chapter, two opportunities are identified to enhance behavioral clustering through implicit interaction research. First, fine-grained interactions can reveal valuable information that is not available in typical access logs; e.g., cursor movements, hesitations before clicking, etc. Second, user behavior has an intrinsic *sequential* nature, which is not considered on current clustering analysis, that can be exploited to simplify the structure of the data. Therefore, we propose two approaches for both opportunities: *1*) a novel methodology to model the website, i.e., finding interaction profiles according to how users behave while browsing, and *2*) a novel clustering algorithm to deal with sequentially-distributed data, whose suitability is illustrated in a human action recognition task.

### Chapter Outline

## 3.1 Introduction

A pervasive problem in science is to construct meaningful classifications of observed phenomena. Clustering can be seen as a compression technique to simplify the structure of the data, so that original objects can be represented by more compact structures that are better tailored for classification, storage, and retrieval purposes. The motivation to using these simplified structures can be as elemental as reducing the number of data samples to save space in large databases, such as web access logs, to more complex applications, such as detecting actions in hours of sensor data. The importance and interdisciplinary nature of clustering is evident through its vast literature; c.f. [Jain, 2010; Jain et al., 1999].

Two broad categories of clustering can be distinguished. In the first one, we have data from known groups as well as observations from entities whose group membership is unknown initially and has to be determined through the analysis of the data. On the other hand, the groups are themselves unknown a priori and the primary purpose of data analysis is to determine the groupings from the data, so that entities within the same group are in some sense more similar than those that belong to different groups. The latter category is the one we are tackling in this chapter.

We explore two novel approaches to (unsupervised) behavioral clustering, with a special emphasis on web page classification and human action recognition. On the one hand, in the context of page classification, currently the task of clustering web pages is approached in a similar way for both web documents and plain text documents. Even if it is known that web pages contain richer and implicit information associated to them [Poblete and Baeza-Yates, 2008], like the interactions that users perform while browsing. Thus, when facing a finer-grained understanding of user behavior and document analysis, server analytics are anything but accurate, being necessary to move toward the client side. As pointed out later, the first core contribution of this chapter is focused on this task.

On the other hand, the task of detecting actions from user behavior is not an easy one. Actions (or activities) are sequential by definition, and, while there are many works that solve sequential *supervised* machine learning problems (e.g. [Dietterich, 2002]), the unsupervised case had remained posing new challenges in the research community for years (e.g. [Trahanias and Skordalakis, 1989]). The second core contribution of this chapter consists in solving this problem.

### 3.1.1 Background

Cluster analysis provides an unsupervised classification scheme to efficiently organize large datasets [Duda et al., 2001]. Additionally, cluster analysis can

supply a means for assessing dimensionality [Agrawal et al., 1998] or identifying outliers [Leiva, 2011]. The fundamental data clustering problem may be defined as discovering "natural" groups, or clubbing similar objects together.

In this chapter, data clustering is seen as a data partitioning problem [Dubes, 1993; MacQueen, 1967; Yu, 2005] as opposed to the hierarchical approach [Fraley, 1996; Murtagh, 1984; Ward, 1963], since we are interested in a *partition* of the data and not in a *structure* (dendrogram) thereof.

Partitional clustering divides a dataset $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ of $n$ $d$-dimensional feature vectors into a set $\prod = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of $k$ disjoint homogeneous classes with $1 < k \ll n$. It is worth pointing out that the task of finding the optimum partition is formidable even for a computer, since this is an NP-hard problem. For example, if $k = 3$, we need to look at $3^{n-1}$ combinations. One way to tackle this problem is to define a criterion function that measures the quality of the clustering partition and then find a partition $\prod^*$ that extremizes such a criterion function.

The most popular algorithm for partitional clustering in scientific and industrial applications is by far the K-means (or C-means) algorithm, which can be considered as a simplified case of Expectation-Maximization (EM) clustering, and is described in the next section.

## 3.2 Revisiting the K-means Algorithm

The K-means algorithm is known for its simplicity, relative robustness, and fast convergence to local minima. K-means, including its multiple variants such as Fuzzy C-Means [Dunn, 1973], K-Medoids [Kaufman and Rousseeuw, 1990], etc., is based on the firm foundation of variance analysis. It requires the number of clusters $k$ to be an input parameter, which is tightly coupled to the nature of the involved task, though there are many studies for choosing $k$ automatically [Bezdek and Pal, 1998; Davies and Bouldin, 1979; Dunn, 1974; Hamerly and Elkan, 2001; Hubert and Arabie, 1985; Milligigan and Cooper, 1985; Sugar, 1998; Tibshirani et al., 2001]. The rough but usual approach is to try clustering with several values of $k$ and choose the one that contributes most to the minimization criterion. Nonetheless, a simple rule of thumb is setting the number of clusters to [Mardia et al., 1979]:

$$k \approx (n/2)^{1/2} \tag{3.1}$$

The criterion function that K-means tries to minimize is the Sum of Quadratic Errors (SQE), denoted simply as Energy or $J$ in the literature, which emphasizes the local structure of the data [Veenman et al., 2002]:

$$J = \sum_{j=1}^{k} H_j \tag{3.2}$$

where

$$H_j = \sum_{\boldsymbol{x} \in \mathcal{C}_j} \| \boldsymbol{x} - \boldsymbol{\mu}_j \|^2 \tag{3.3}$$

represents the heterogeneity (or distortion) of cluster $\mathcal{C}_j$, and

$$\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{\boldsymbol{x} \in \mathcal{C}_j} \boldsymbol{x} \tag{3.4}$$

is the cluster mean, with $n_j = |\mathcal{C}_j|$ being the number of samples in such cluster.

The most common implementation of this algorithm, generally attributed to Lloyd [1982], uses a minimum distance criterion, where in each iteration all samples are assigned to their closest cluster mean and convergence is achieved when the assignments no longer change. There exists, however, a more interesting version, often attributed to Duda and Hart [1973], which uses a sample-by-sample iterative optimization refinement scheme. At each step, the SQE is evaluated and the considered sample is reallocated to a different cluster if and only if that reassignment decreases $J$. Clearly, such a greedy optimization guarantees that the resulting partition corresponds always to a local minimum of the SQE. This refined version is explained as follows.

The variation in the SQE produced when moving a sample $\boldsymbol{x}$ from cluster $j$ to cluster $l$ can be obtained in a single computational step as [Duda et al., 2001]:

$$\Delta J(\boldsymbol{x}, j, l) = \frac{n_l}{n_l + 1} \| \boldsymbol{x} - \boldsymbol{\mu}_l \|^2 - \frac{n_j}{n_j - 1} \| \boldsymbol{x} - \boldsymbol{\mu}_j \|^2 \tag{3.5}$$

If this increment is negative, the new means, $\boldsymbol{\mu}'_j$, $\boldsymbol{\mu}'_l$ and the SQE, $J'$, can then be incrementally computed as follows [Duda et al., 2001]:

$$\begin{aligned} \boldsymbol{\mu}'_j &= \boldsymbol{\mu}_j - \frac{\boldsymbol{x} - \boldsymbol{\mu}_j}{n_j - 1} \\ \boldsymbol{\mu}'_l &= \boldsymbol{\mu}_l + \frac{\boldsymbol{x} - \boldsymbol{\mu}_l}{n_l + 1} \\ J' &= J + \Delta J(\boldsymbol{x}, j, l) \end{aligned} \tag{3.6}$$

### 3.2.1 Sequential Clustering

When clustering sequential data there exists a strong constraint, often related to time, that can be exploited to a great advantage. Nonetheless, by ignoring this constraint, classical clustering techniques fail to cope with the underlying

sequential data structure, as illustrated in Figure 3.1. What is more, previous research on sequential data clustering considered the objects to cluster as whole data sequences or previously determined subsequences thereof; c.f., Guralnik and Karypis [2001]; Lee et al. [2007]. Instead, we are interested in discovering *subtrajectories* within a single trajectory, so that we can obtain a simplified data structure preserving the underlying data sequentiality. From this point of view, approaches based on Hidden Markov Models (HMMs) have been proposed; e.g., [Bashir et al., 2007]. The downside of HMMs, however, is that they require complex training, and also can be prohibitive if processing power is a restriction, e.g., working on mobile devices. As such, we propose here a closed-form solution having a low computational cost in terms of performance, which translates to really fast convergence times, and provides consistent results in terms of accuracy: each run for a given number of classes always yields the same (well-formed) sequential clustering configuration.

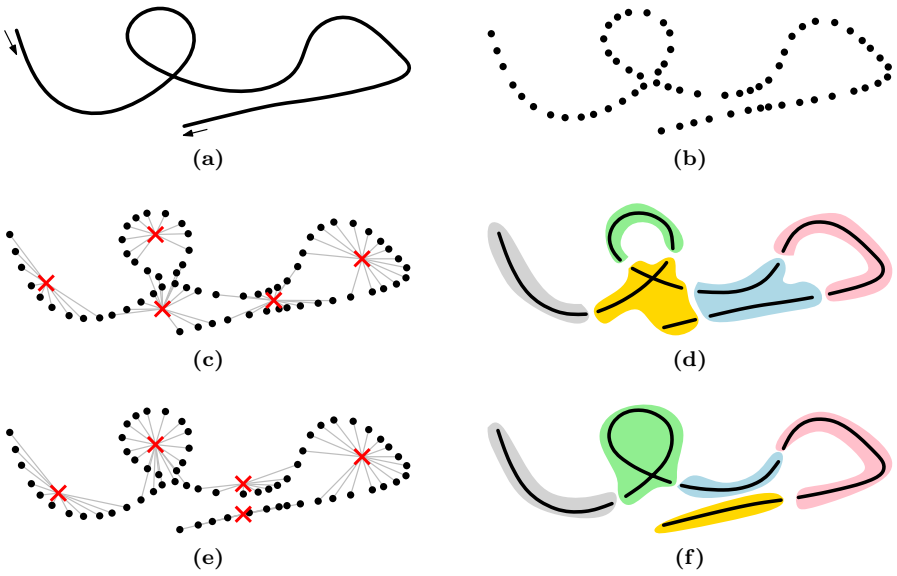



**Figure 3.1:** A 2D example. An arbitrary shape (3.1a) is digitized (3.1b) and reduced to 5 elemental units. Classical clustering algorithms do not deal with temporal information and, therefore, resulting units are ill-defined (3.1c), leading to an inconsistent configuration (3.1d). Our approach, however, provides a simple framework to easily cope with the sequentiality of the data (3.1e, 3.1f).

If the data in a dataset $X$ are sequentially given, it can be said that such data describe a *trace* or *trajectory* in the $d$-dimensional vector space where samples are represented:

$$X = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \tag{3.7}$$

We define a sequential clustering into $k$ classes as the mapping

$$\mathrm{b} : \{1, \ldots, k\} \mapsto \{1, \ldots, n\}$$

where $\mathrm{b}_j$ is the (left) *boundary* of cluster $j$; i.e., the index of the first sample in that cluster. See Figure 3.2 for a graphical example.

Using this convenient notation, the $j$-th (sequential) cluster of $X$ can be written as follows:

$$\mathcal{C}_j = \{\boldsymbol{x}_{\mathrm{b}_j}, \boldsymbol{x}_{\mathrm{b}_j+1}, \ldots, \boldsymbol{x}_{\mathrm{b}_j+n_j-1}\} \tag{3.8}$$

where $n_j$ can now be trivially computed as

$$n_j = \mathrm{b}_{j+1} - \mathrm{b}_j \tag{3.9}$$

This way, (3.3) and (3.4) can be rewritten as

$$H_j = \sum_{i=\mathrm{b}_j}^{\mathrm{b}_{j+1}-1} \| \boldsymbol{x}_i - \boldsymbol{\mu}_j \|^2 \tag{3.10}$$

$$\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{i=\mathrm{b}_j}^{\mathrm{b}_{j+1}-1} \boldsymbol{x}_i \tag{3.11}$$

and (3.5) and (3.6) can be directly used as such with this new formulation.

### 3.2.2 Warped K-Means

We propose a novel algorithm named *Warped K-Means* (WKM), inspired by the idea that the original data structure is delusively distorted, or "unfolded" (see Figure 3.2) to cope with the sequentiality restrictions. Our proposal is based on the trace segmentation (TS) technique for partition initialization (Figure 3.3), followed by a K-means-like optimization procedure (Figure 3.4).

As in classical K-means, WKM reallocates samples based on the analysis of effects on the objective function $J$, caused by moving a sample from its current cluster to a potentially better one. But now a hard sequentiality constraint is imposed. The first half of samples in cluster $j$ are only allowed to move to cluster $j-1$, and, respectively, the last half of samples are only allowed to move to cluster $j+1$. A sample will be reallocated if and only if the corresponding SQE increment is beneficial (i.e., negative). This process is iterated until no transfers are performed.

Because of this constraint, along with the sequential ordering of samples within each cluster, typically only the samples close to the cluster boundaries get reallocated. To take advantage of this observation, we introduce an optional parameter $\delta \in [0, 1]$ which allows us to fine-tune the WKM behavior and at the

same time achieve further reductions in computational cost. It allows testing only those samples that are more or less close to cluster boundaries. In the extreme case of $\delta = 0$ the algorithm is conservative: all samples in a cluster are visited to see if they should be reallocated. In the other extreme, if $\delta = 1$ WKM is optimistic: only the boundary and the last sample in each cluster will be checked. In general, the effect of $\delta$ is illustrated in Figure 3.2.

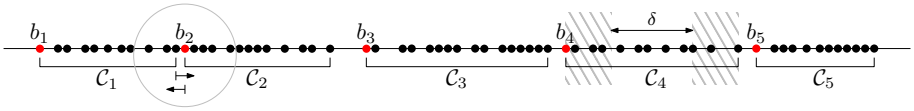



**Figure 3.2:** The basis of WKM. The algorithm provides an optional parameter $\delta$ which specifies the maximum amount of samples that will be inspected in each iteration. This way, if $\delta = 1$ only two samples are considered per iteration, while if $\delta = 0$ full search is carried out.

In sum, three key features differentiate our approach from other K-means based algorithms: initialization, visiting order, and sequentiality constraints.

**Algorithm: TS Boundary Initialization**

**Input**: Trajectory $X = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$; No. Clusters $k \geq 2$
**Output**: Boundaries $\mathrm{b}_1, \ldots, \mathrm{b}_k$

$L_1 = 0$
**for** $i = 2$ **to** $n$ **do** // Accumulated trace length
  $L_i = L_{i-1} + \| \boldsymbol{x}_i - \boldsymbol{x}_{i-1} \|$
$\lambda = \frac{L_n}{k}$ // Segment length
$i = 1$
**for** $j = 1$ **to** $k$ **do**
  **while** $\lambda\ (j-1) > L_i$ **do** // Interpolate
    $i{+}{+}$
  $\mathrm{b}_j = i$ // Define boundaries

**Figure 3.3:** Boundaries initialization. Each boundary is evenly allocated according to a piecewise linear interpolation on accumulated distances, resulting in a non-linearly distributed boundary allocation.

## Algorithm Overview

In each cluster $j$, the samples close to its boundary $\mathrm{b}_j$ are first visited to see if they can be advantageously reallocated to the previous cluster, $j - 1$ ("reallocate backwards" loop). Then, the samples close to the boundary of the next cluster, $j + 1$, are similarly considered ("reallocate forwards" loop). It is worth noting that with $\delta < 1$ the proportion of samples processed in each backward or forward sequential chunk is typically less than the number corresponding to the given value of $\delta$. This is because the reallocation process

**Algorithm: WKM**

**Input**: Trajectory $X$; No. Clusters $k \geq 2$ [; Proportion $\delta = 0.0$]
**Output**: Boundaries $\mathrm{b}_1, \ldots, \mathrm{b}_k$; Centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$; Distortion $J$

```
Initialize boundaries b_1, ..., b_k                          // Use TS (Figure 3.3)
for j = 1 to k do
    Compute μ_j, n_j, J                          // Use Eq. (3.2), (3.9), (3.10), and (3.11)
repeat
    transfers = false
    for j = 1 to k do
        if j > 1 then                                        // Reallocate backwards 1st half
            first = b_j;   last = first + ⌊(n_j/2)(1 − δ)⌋
            for i = first up to last do
                if n_{j−1} > 1 and ΔJ(x_i, j, j − 1) < 0 then
                    transfers = true
                    b_j += 1;   n_j += 1;   n_{j−1} −= 1
                    Update μ_j, μ_{j−1}, J                   // According to Eq. (3.6)
                else break
        if j < k then                                        // Reallocate forwards 2nd half
            last = b_{j+1} − 1;   first = last − ⌊(n_j/2)(1 − δ)⌋
            for i = last down to first do
                if n_j > 1 and ΔJ(x_i, j, j + 1) < 0 then
                    transfers = true
                    b_{j+1} −= 1;   n_j −= 1;   n_{j+1} += 1
                    Update μ_j, μ_{j+1}, J                   // According to Eq. (3.6)
                else break
until ¬transfers
```

**Figure 3.4:** Warped K-Means. A sample $\boldsymbol{x} \in \mathcal{C}_j$ is only allowed to move either to cluster $\mathcal{C}_{j-1}$ or $\mathcal{C}_{j+1}$. If a move proves advantageous, that is, the increment in SQE is negative, the sample is reallocated and the two cluster means involved in such a reallocation are incrementally recomputed according to (3.6). Otherwise, the next cluster is inspected, in order to preserve the clustering sequentiality.

is aborted as soon as the SQE does not improve for that chunk, in order to preserve the sequentiality of our clustering procedure.

Note also that if some samples are reallocated during the forward processing of cluster $j$, then we do not need to re-check them in the backward processing of cluster $j + 1$. This is easily verifiable with an auxiliary variable that stores the index of the last reallocated sample. This detail, however, is not shown in the WKM pseudo-code for the sake of clarity.

The computational cost of a complete iteration of WKM over the whole sequence $X$, depends on the number of samples $n$, the sample vector dimension $d$, and the number of clusters $k$. As previously discussed, it can also depend on the value of $\delta$. On the one hand, if $\delta = 1$, the complexity of WKM is reduced to $\Theta(kd)$ per iteration, in comparison to $\Theta(nkd)$ in the case of classical K-means.

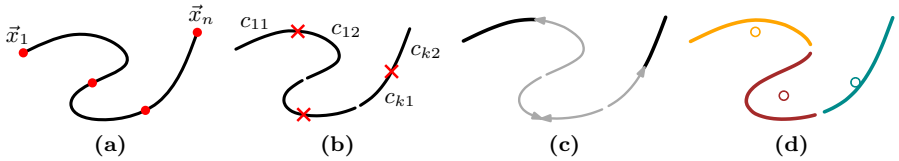



**Figure 3.5:** Graphical overview of Figure 3.4 for $k = 3$ clusters. [3.5a] Key points identification: the first and last key points always match the first and last data points ($\boldsymbol{x}_1$ and $\boldsymbol{x}_n$). [3.5b] Initial segmentation: crosses mark the $k$ segments' middle points. [3.5c] Point visiting order for reallocation: notice that chunks $c_{11}$ and $c_{kn}$ do not need to be inspected. [3.5d] Final clustering configuration: circles represent each segment's centroid.

On the other hand, if $\delta = 0$, the best- and worst-case complexities are $\Omega(kd)$ and $O(nd)$, respectively. Therefore, for all values of $\delta$ and in all cases, each iteration of WKM is expected to be (much) faster than conventional K-means algorithms. Moreover, according to empirical observations, the convergence tends to require less iterations than such classical K-means algorithms.

Overall, the main advantages of our proposal can be summarized as follows:

- **Consistent results**: It always guarantees the convergence to a good local minimum, i.e., a low distorted partition of the original dataset that preserves sequence ordering.
- **Robust solution**: Each run for a given $k$ always yields the same clustering configuration—thanks to the initialization algorithm and the minimization criterion for sample reallocation.
- **Low computational cost**: Much lower than that of classical K-means algorithms since, instead of the usual all-against-all search strategy, we only need to check two clusters in each step.
- **No extra mandatory parameters**: Our solution requires the same input data and parameters as in K-means, though an optional $\delta$ threshold can be specified to tune both the algorithm behavior and its cost.

As discussed by Leiva and Vidal [2011], the WKM algorithm is also suitable for online learning tasks over large datasets, due to the following facts: *1)* the computational cost of updating the centroids is independent of the number of samples and *2)* the final partition can be updated while new samples arrive without affecting too much the previous data structure.

## 3.3 Evaluation

In this section we evaluate behavioral clustering on two different tasks: web page classification and action recognition. In the former task, we are interested

in describing a website by how users interact within their contents. To this end, a classical clustering methodology is intuitively quite useful: different pages that trigger different behaviors should lie in different clusters, while pages with similar interactions are likely to be assigned to the same cluster. In the latter task, we are interested in characterizing human actions from raw sensor data. To this end, we look for data compression methods to reduce the number of samples for later action recognition. Here, a clustering methodology might also be quite useful, although preserving data sequentiality is of utmost importance—something that classical clustering methods fail to achieve.

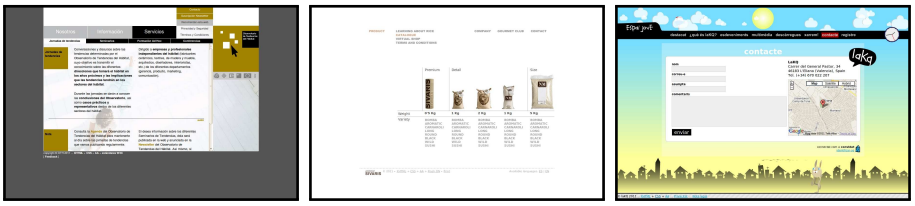
Notice that the goal of the page classification task is to describe the website as a whole, so there is no need to preserve data sequentiality. However, the goal of the action recognition task is to discover the most informative number of elementary samples that define a human action. Therefore, in this case it is clear that a better outcome is expected if we employ our WKM algorithm instead of classical methods.

### 3.3.1 Clustering Browsing Interactions

In the same way as web clustering engines organize search results by topic or document relevance, our method aims to organize websites by users' interaction semantics. Such semantics of interaction are characterized by a series of metrics (16 in total), which are computed by our mouse tracking tool and were described in Section 2.3.6, say, 1D metrics: browsing time, number of clicks, motion activity, and path length; and 2D metrics (with X and Y components): distance, range, entry point, exit point, centroid, and scroll reach. We hypothesize that if such metrics are consistent, they should generate clusters of (approximately) same precision for a given typology of pages. In addition, it is important to remark that metrics should be normalized. For instance, time and scrolling are often reported as relevant metrics [Claypool et al., 2001; Holub and Bielikova, 2010]. However, it is clear that longer/bigger pages will require both more time and scrolling, and hence they could lead to misleading results if one does not consider data normalization. Usually whitening the data (i.e., ensuring a distribution of each metric with mean 0 and variance 1) may be enough.

#### Method

We gathered interaction data for approximately a month on three *informational* websites (Figure 3.6), i.e., they are dedicated to the purpose of providing information to the users (like, e.g., news portals or corporate blogs). Most websites could fit in this type of website to some extent, so evaluating our approach on this typology should ensure a broad generalization scope. The characteristics of each corpus are summarized in Table 3.1.



**(a)** OTH (habitat trends) **(b)** NM (rice company) **(c)** LAKQ (social club)

**Figure 3.6:** Example screenshots from the corresponding websites of each evaluated dataset (see also Table 3.1).

| Codename | Size (MB) | # Logs | # URLs |
|---|---|---|---|
| OTH | 25.5 | 4803 | 63 |
| NM | 33.5 | 5601 | 43 |
| LAKQ | 7.4 | 1232 | 28 |

**Table 3.1:** Overview of evaluated datasets (see also Figure 3.6).

## Procedure

Users were selected by random sampling, which means that only a fraction of all visitors (with equal probability of selection) was collected. We set a tracking frequency of 24 fps. Each interaction log was stored in a MySQL database and then exported in XML format. Logs were modeled as normalized interaction-based 16-$d$ feature vectors (see Section 3.3.1). We took into account visits that lasted 0.5 hours at most, in order to discard bogus or spurious logs beforehand. Then, we applied the classical K-means algorithm to automatically group the logs in each corpus, using random convex combination as initialization method [Leiva and Vidal, 2010] to accelerate convergence. The optimal number of clusters for each corpus was determined as the marginally less distorted grouping in terms of the SQE, which is proportional to the intra-cluster (or within-class) variance; see, e.g., Figure 3.7. Once we had each log assigned to a cluster, we extracted the mean and standard deviation for the tracked interaction features, for later comparison and further analysis.

## Results

To illustrate the usefulness of the proposed framework we start by describing the profiles found in the OTH corpus. Table 3.2 summarizes the clustering results for this dataset. Then we discuss the main observations that relate to the other evaluated corpora.

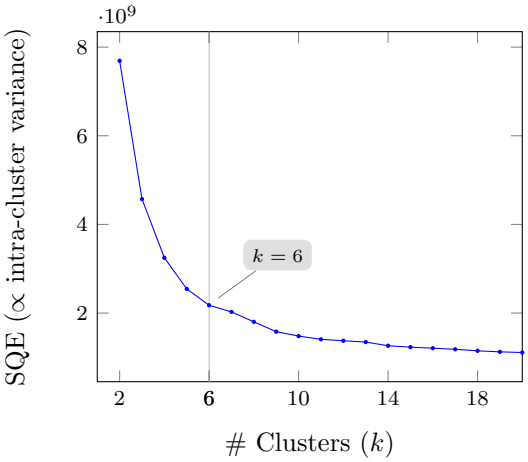



**Figure 3.7:** Clustering the OTH dataset. The intra-cluster variance (or energy, or SQE) decreases with increasing number of classes $k$. However, at some point the marginal gain will be smaller. Intuitively, this can be chosen as the number of clusters that better summarizes the dataset.

| Cluster # | Population | % | Energy (SQE) | % | Variance |
|---|---|---|---|---|---|
| 2 | 698 | 14 | $3.6{\cdot}10^8$ | 16 | $5.1{\cdot}10^5$ |
| 3 | 1347 | 28 | $4.7{\cdot}10^8$ | 22 | $3.5{\cdot}10^5$ |
| 6 | 2220 | 46 | $5.4{\cdot}10^8$ | 25 | $2.4{\cdot}10^5$ |
| **Avg. Total** | 4748 | 100 | $2.1{\cdot}10^9$ | 100 | $4.5{\cdot}10^5$ |

**Table 3.2:** Clusters found in the OTH dataset. Outliers were classified into three clusters (#1, #4, and #5), not reported here because they all represent near 10% of sample population.

**Profiles in OTH corpus** According to the 'elbow' criterion[1] (Figure 3.7), we found $k = 6$ to be the number of classes that better summarizes this dataset. However, three cluster were identified as outliers, which accounted for near 10% of the population. So actually we found three meaningful groups in this dataset. This fact reinforced the idea of using behavioral clustering for isolating sub-populations. Looking at these outliers we found that logs belonging to these clusters had unusual behaviors; e.g., 11.5 clicks on average (SD = 19.5), extremely long cursor trajectories of 12797.4 px (3130.9), and so on.

Pages in cluster #6 concentrated the biggest sub-population (46% of the data). We found short-term sessions of M = 30 s (SD = 132.9) with "one-click" browsing patterns of 1.1 clicks (0.6). Scrolling reached 40% (20) of the users' browser viewport and mouse range comprised 181.8 px (128.9) and 120.7 px (105.6) in horizontal and vertical axes, respectively. Thus, logs belonging to this cluster could be classified as "basic presence" pages, supporting somehow the evidence of the typology of the tracked pages (i.e., an informational website).

[1] In the literature, it is also mentioned as the 'gap statistic' [Tibshirani et al., 2001].

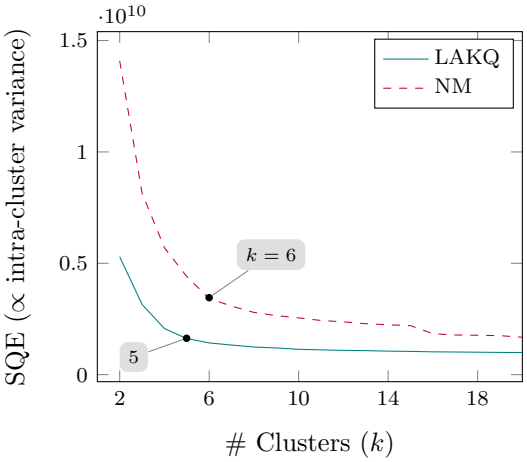



**Figure 3.8:** Clustering LAKQ and NM datasets, following the same criterion depicted in Figure 3.7. We identified 5 and 6 clusters to be the most informative number of classes to describe the pages in each dataset, respectively.

The population of cluster #3 was the least dispersed overall (16% of energy). In-page interactions lasted 45.7 s (125.96), issuing 1.5 clicks (1.6) per session on average. Pages belonging to this group were found to be browsed by relatively active users, e.g., mouse distance: 7.1 px (6.7), mouse motion: 16% (13), vertical scroll of 65% (23). Therefore, we hypothesize that these pages were be the most familiar for the users. Although we do not have such ground truth data to back up this claim.

Pages in cluster #2 showed metrics related to cluster #3, with similar power-law distributions. However, users in this cluster spent more browsing time, which was also more dispersed overall: 1.3 min (3.5), and clicked more: 2.33 (2.34). Pages were scrolled considerably more than the half of their browser's viewport: 76% (22). Together with the rest of considered metrics, this fact led us to conclude that pages in this cluster were the most interesting for the users.

**Profiles in NM and LAKQ corpora** Instead of performing a detailed analysis of each cluster found akin the OTH corpus as described above, we shall depict some interesting observations.

| Cluster | Population | % | Energy (SQE) | % | Variance |
|---|---|---|---|---|---|
| 1 | 159 | 13 | $2.5{\cdot}10^8$ | 15 | $1.6{\cdot}10^6$ |
| 4 | 632 | 53 | $4.1{\cdot}10^8$ | 24 | $6.4{\cdot}10^5$ |
| 5 | 346 | 29 | $4.5{\cdot}10^8$ | 27 | $1.3{\cdot}10^6$ |
| **Avg. Total** | 1178 | 100 | $1.6{\cdot}10^9$ | 100 | $1.3{\cdot}10^6$ |

**Table 3.3:** Clusters found in the LAKQ dataset. Two outliers (clusters #2 and #3) were identified.

Regarding Table 3.3, the biggest cluster (#4, 53% of the data) was surprisingly

not the most distorted overall. We observed that all meaningful clusters found were more or less similar in terms of dispersion, which is a convenient feature of K-means. What is specially interesting, however, is that vertical scrolling often overpassed 100% of the browser viewport. Taking also into account the mouse ranges, centroids, and entry/exit coordinates in these groups, we speculate that most visitors were using (moderately) large displays. This hypothesis was then verified by observing that the average screen resolution was 1208.9 (203.5) x 860.7 (118.8) px.

| Cluster | Population | % | Energy (SQE) | % | Variance |
|---|---|---|---|---|---|
| 2 | 1697 | 30 | $8.9{\cdot}10^8$ | 26 | $5.2{\cdot}10^5$ |
| 3 | 968 | 17 | $5.8{\cdot}10^8$ | 17 | $6.1{\cdot}10^5$ |
| 6 | 2132 | 38 | $9.1{\cdot}10^8$ | 26 | $4.2{\cdot}10^5$ |
| **Avg. Total** | 5557 | 100 | $3.4{\cdot}10^9$ | 100 | $6.1{\cdot}10^5$ |

**Table 3.4:** Clusters found in the NM dataset. Three outliers (clusters #1, #4, and #5) were identified.

As observed in Table 3.4, similar to the OTH dataset, we found three clusters (#2) in the NM dataset that were clear outliers. Again, we remark the usefulness of using behavioral clustering for isolating sub-populations in large datasets. On the other hand, though, the remaining clusters showed more consistent behaviors, comprising between 17% and 26% of the overall cluster energy. Overall, it was interesting to observe that the proposed metrics lead classical clustering to find the same number of classes as in the previously studied datasets. We elaborate more on this below.

## Discussion

Our study threw some interesting suggestions. First, using this clustering framework allows to focus on a small number of groups to describe the vast majority of the pages of a website. For instance, in the OTH corpus the 3 main clusters found represent 95% of the browsed pages. Similarly, by looking at the same number of clusters, we can explain 89% and 93% of the pages in LAKQ and NM datasets, respectively. Second, as previously commented, our method allows to describe web pages in a completely different way, i.e., from the user interactions' point of view, instead of the usual structure/content/usage triad. This knowledge has an interesting potential to be used to compare cross-site browsing behaviors, or predict interest of non-browsed pages. Third, using the information implicitly embedded in user's interactions may help webmasters to redesign the most important pages, in terms of in-page interactions. This way, if individual personalization is not possible, users could browse the site at the same performance level to a greater or a lesser extent. Fourth, we found

that the user sample we tracked at each website was often a mixture of distributions. This evidence encourages to be cautious in using logging tools or intuitions that assume a normal distribution for all users.

As observed, exploiting the browsing context from user behavior may serve as a useful complement to current web mining techniques. Further suitability of this work relates to any system that taps knowledge about the user, e.g.: information retrieval, relevance feedback, document organization, or usage inference, just to name a few. Armed with this awareness, one could carry out novel research studies on user modeling and related applications.

### 3.3.2 Classifying Human Actions

In this case, we chose a straightforward classification task to test the WKM algorithm in isolation. We wanted to test how data sequentiality may affect the performance of a recognizer. To this end, we used the Localization Data for Person Activity dataset [Kaluža et al., 2010] form the UCI Machine Learning Repository [Asuncion and Newman, 2007]. In this corpus, 164860 data points were captured from 5 people wearing 5 active RFID tags (both ankles, belt, and chest). Up to 11 human actions were represented as a time series of `x,y,z` coordinates of such 5 body parts.

Note that, while there is an important number of works tackling the problem of classifying human actions, we chose this corpus to show the capabilities of WKM as a simple and accurate compression tool for a complex, real-world task. To this end, each human action is represented as a vector of a fixed number of "elementary actions", where each elementary action is, in turn, a cluster mean vector obtained by clustering the original sequence of action samples (`x,y,z` coordinates). Once each action is represented as a fixed dimension vector, many simple classifiers can be adequately used, among which we chose the well-known Nearest-Neighbor (NN) classifier.

#### Method

To characterize each activity, the `x,y,z` coordinates of all sensors were merged into a single 12-dimensional feature vector sample $\boldsymbol{x} = (x_1, y_1, z_1, \ldots, x_4, y_4, z_4)^T$. So a trajectory was defined as the sequence $X = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, where $n$ is the number of samples in $X$.

Unfortunately, the dataset did not include the same number of instances per sensor. Therefore some of the composed trajectories had extremely different number of 12-dimensional vectors (e.g., some had just two vectors and others had more than 800). We needed thus to build a more comparable dataset; so, while composing each trajectory we verified that it had at least 10 samples.

Eventually we obtained 125 trajectories of 162 samples on average (SD=138.6), belonging to one of the following 5 classes: 'falling', 'lying', 'on-all-fours', 'sitting', and 'walking'. There were 25 trajectories per class. The features of the dataset used in the experiments are depicted in Table 3.5.

| | |
|---|---|
| **Trajectories** | 125 |
| **Mean samples per trajectory** | 162 |
| **Dimension of sample vectors** | 12 |
| **Classes (actions)** | 5 |
| **Number of trajectories per class** | 25 |

**Table 3.5:** Features of the dataset used in the WKM experiments.

**Vector Representation** We ran our implementation of WKM to cluster each trajectory into a variable number of segments ($k \in \{2, 4, \ldots, 20\}$) and with different cluster proportions ($\delta \in \{0, 0.2, \ldots, 1\}$). We also compared WKM with two well-known versions of K-means: the classical *Duda&Hart*'s algorithm [Duda and Hart, 1973] and the popular *Lloyd*'s version [Lloyd, 1982], using both random and TS initializations. When initializing randomly we performed up to 5 times each experiment, in order to mitigate the effects of chance, and computed the average values.

The cluster means obtained by $k$-clustering each action data sequence were stacked into a $3 \cdot 4 \cdot k$ dimensional feature vector, i.e., a $12\,k$-dimensional vector. For those trajectories with less samples than the desired number of segments, (i.e., when $k > n$) we used singleton clusters instead (i.e., $k = n$) and the missing dimensions were filled with zeros. As we will see below, this fact had clear repercussions when classifying some trajectories with $k > 10$ (ten was the minimum number of vectors in all trajectories), specially in terms of classification error.

**Nearest Neighbor Classifier** The simple and well-known 1-NN classifier with Euclidean distance was adopted to classify vector-represented action trajectories. As previously pointed out, each class was represented by a number of prototype trajectories. Each test trajectory was classified into the class of its nearest neighbor prototype.

In these experiments, we employed the C++ ANN library [Mount and Arya, 1998] for NN searching, with its basic, exact search option. Given the relatively small number of available trajectories overall, we adopted the leaving-one-out training and testing procedure.

## Results

The first experiment was aimed at studying the behavior of different algorithms when minimizing SQE and increasing the number of clusters. Results are shown

in Figure 3.9. As expected, in all cases SQE decreases monotonically with increasing number of clusters. It is interesting to note that K-means algorithms achieve a (slightly) lower SQE than WKM, which is explained by the lack of sequentiality restrictions, that otherwise WKM imposes on the data.

In the next experiment we studied the ability of different clustering algorithms to behave as data preprocessors, in order to obtain simplified vector-represented trajectories for classification purposes. We considered the case when a sensor trajectory is segmented into just one single cluster ($k = 1$) as the baseline; that is, each trajectory is represented by a 12-dimensional vector corresponding to the average of all its trajectory samples. In that case, the classification error was as low as 9.6%, which is reasonable given the nature of the activities involved (e.g., the position of "lying" and "sitting" should differ greatly at least in the average `z` coordinate of each sensor).

Results for other values of $k$ are shown in Figure 3.10. As expected, certain segmentations performed better than others for each algorithm, but a particularly adequate number of elementary actions seems to be 6 in most cases. Interestingly, WKM is the method that better puts this fact forward. We observed that accuracy degraded noticeably for $k > 10$, to the point that for $k = 20$ error rates were above 50% for all classifiers—for the reason explained in Section 3.3.2. Also, as observed, the randomly initialized versions were the worst performers.

In order to better understand the impact of the $\delta$ threshold of WKM, we repeated the previous experiment for different values of this threshold. Figure 3.11 shows the influence of $\delta$ in the recognition accuracy. We see that by tuning this parameter WKM results can be further improved, with a best result of 3.2% error rate for six elementary actions. Finally, regarding the computational cost of each algorithm, as shown in Figure 3.12, WKM behaves much better than its peers.

Table 3.6 summarizes the results discussed so far. Classical K-means algorithms do not help overcoming the trivial baseline (just one cluster). In contrast, WKM achieved a recognition accuracy of 97%, which represents a 66% improvement over the baseline. WKM is borderline statistically significantly better than all compared methods [$\chi^2_{(7,N=125)} = 4.44, p = .07$]. Most interestingly, the improvements introduced by WKM are achieved along a huge computational cost reduction (more than one order of magnitude) with respect to K-means algorithms. We can conclude that WKM was the best performer among its peers, and that results confirmed our expectations.

## Discussion

As can be observed in the figures, WKM gives very competitive error rates at a low computational cost. Therefore, our experimental results show that WKM is

**Figure 3.9:** Sum of squared errors against number of segments. Each value is averaged for all trajectories (activity × person × trial). As expected, the segmentations achieved by WKM have higher distortion than those of classical K-means, since the former imposes a strong sequential restriction, while the latter does not.

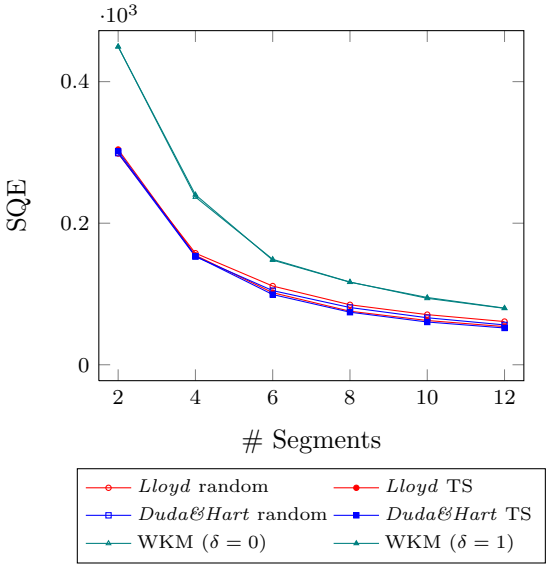
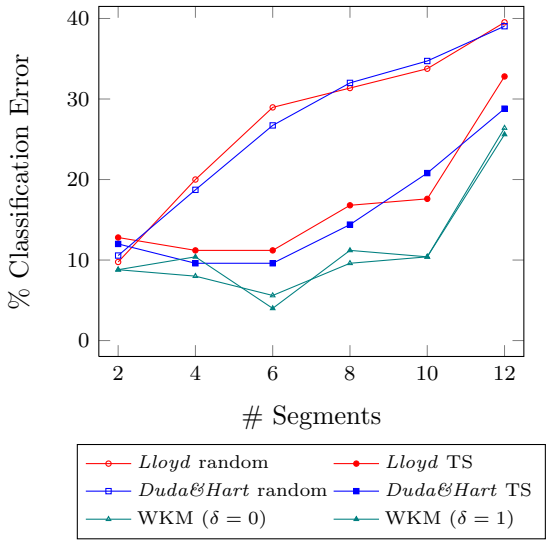



**Figure 3.10:** We performed variations to three alternatives for clustering trajectories: The *Duda&Hart*'s algorithm and the *Lloyd* version, using both random initialization and trace segmentation, and the WKM algorithm using two extreme distortion thresholds.



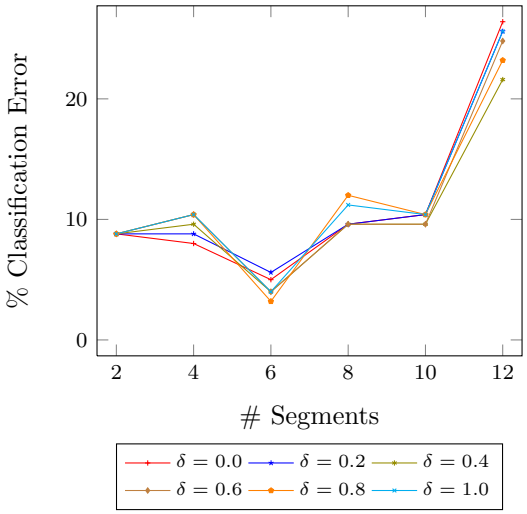



**Figure 3.11:** WKM classification error. We used different $\delta$ thresholds for each tested number of segments. The best accuracy was achieved when using $k = 6$ for all threshold values.

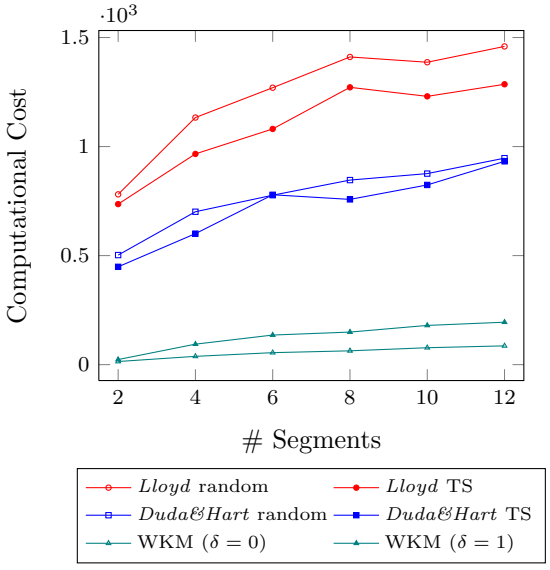



**Figure 3.12:** Computational cost against number of segments. Cost is estimated as number of times Eq. (3.5) is executed. For *Lloyd* versions, cost was computed as the number of times each algorithm tested if cluster means did change.

| Algorithm | Best $k$ | % Error | Cost |
|---|---|---|---|
| Baseline | 1 | 9.6 | — |
| *Lloyd* random | 2 | 9.8 | 796 |
| *Lloyd* TS | 6 | 11.2 | 1080 |
| *Duda&Hart* random | 2 | 10.6 | 448 |
| *Duda&Hart* TS | 6 | 9.6 | 778 |
| WKM $\delta = 0.0$ | 6 | 5.6 | 54 |
| WKM $\delta = 0.8$ | 6 | **3.2** | 71 |
| WKM $\delta = 1.0$ | 6 | 4.0 | 135 |

**Table 3.6:** Summary of sequential clustering results. Bold value indicates that it is the best result among all methods being compared.

an interesting approach for lowering both classification error and computational cost regarding to using other comparable clustering alternatives.

It is worth pointing out that all algorithms initialized with TS allow to find the "natural" number of classes. However, as shown in Figure 3.10, for WKM this number in turn corresponds to the lowest classification error rate in all cases (see also Table 3.6).

Additionally, we have shown that WKM ensures monotonic improvement and finite assignments in a sequential fashion, which translates to convergence to a good local minimum in which trajectory segments are well-defined. This can be leveraged in some interesting applications, as we shall expose as follows.

**Online Handwriting** Our clustering technique can be used as a preprocessing step for online text recognition. As illustrated in Figure 3.1, the obtained (well-formed) segments capture pen-stroke regularities which can be advantageously exploited by existing handwritten recognition approaches to increase character recognition accuracy [Leiva and Vidal, 2012].

**Eye/Mouse Tracking** This algorithm entails a reliable contribution to clustering eye movements on aggregated data; e.g., both heatmaps and areas of interets (AOIs) are computed by distance-based clusters, and therefore they do not distinguish between long-time fixations of a single person or short-time fixations of a group of people.

**Motion Segmentation** The storage and transmission of motion tracking content is a problem due to their tremendous size and the noise caused by imperfections in the capture process. Thus, one could use our method for a more compact representation of these (large) data.

In general, any discipline that would handle ordered data sequences could benefit from our approach; e.g., human motion classification from surveillance cameras or automatic video key frame extraction.

## 3.4 Conclusions and Future Work

This chapter has covered behavioral clustering, a broad term that refers to the task of automatically labeling and classifying user behavior, which was evaluated on two different tasks with a series of real-world datasets.

In the first task we were able to discover "hidden" profiles on websites, according to how users behave while browsing. We have demonstrated that this technique can be used to organize and describe websites from the user interactions' point of view. This technique can also be used as a measure of similarity between web pages, to evaluate their design in an automated fashion, or to discover outliers. We believe that this work opens a new door to novel approaches on web behavior studies.

Lines of future work regarding web page classification according to (implicit) interaction metrics include inferring behavior of non-browsed pages and finding related websites based on user interactions. The metrics we used for clustering are related to cursor activity, because cursor data are easy to collect and no special instrumentation is required on client side. However, user interaction is inherently multimodal. Thus, other related input signals such as eye movements could (and should) be taken into consideration, and be incorporated to more sophisticated web profiles. This way, one may complement studies of quantitative/qualitative nature, improving thus the usability and usefulness of websites, and being able to extend this methodology to related fields such as web applications or software products.

In the second task, we have presented a novel revisitation of the K-means algorithm, specially suited for sequentially distributed data. We have successfully used this approach to automatically identify human actions derived from raw sensor data. By taking into account that data are sequentially given, our proposal, WKM, behaves much better than classical clustering algorithms. One obvious reason why using a cluster representation may have advantages over working with raw sensor data is the evident size reduction, which in turn may enhance the ease of storage, transmission, analysis, and indexing. Moreover, extending this notion to the analysis of trajectories reverts in another significant advantage: having a good and compact representation of a data sequence makes it more invariant to noise or distortions in such data. This fact has been backed up by our experimental results, lowering both classification error and computational cost regarding to using other comparable clustering alternatives.

As stated in this chapter, a critical step for (adequately) clustering sequential data with WKM is the initialization of segment boundaries. We used the TS

technique, although other algorithms that ensure a sequential distribution may be also helpful. For instance, we could use an equispaced boundary initialization instead. Future work will be focused on removing the (optional) $\delta$ parameter from the algorithm, and instead learning automatically the best value for a given cluster configuration. Further research on WKM will be leaned toward an optimum procedure of choosing the number of clusters. We hope that our work may encourage researchers and practitioners to apply this algorithm to a wealth of new problems and/or domains.

# Bibliography of Chapter 3


R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pp. 94–105, 1998.

A. Asuncion and D. J. Newman. UCI machine learning repository, 2007. Available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

F. I. Bashir, A. A. Khokhar, and D. Schonfeld. Object trajectory-based activity classification and recognition using Hidden Markov Models. *IEEE Transactions on Image Processing*, pp. 1912–1919, 2007.

J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *IEEE Transactions on System, Man and Cybernetics*, 28(3):301–315, 1998.

M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *Proceedings of the 6th international conference on Intelligent user interfaces (IUI)*, pp. 33–40, 2001.

D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(4):224–227, 1979.

T. G. Dietterich. Machine learning for sequential data: A review. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pp. 15–30, 2002.

R. Dubes. *Handbook of Pattern Recognition & Computer Vision*, chap. Cluster analysis and related issues, pp. 3–32. World Scientific Publishing Co., Inc., 1993.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*, chap. Unsupervised Learning and Clustering, pp. 517–599. John Wiley & Sons, 2001.

J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

J. C. Dunn. A cluster separation measure. *Journal of Cybernetics*, 4:95–104, 1974.

C. Fraley. Algorithms for model-based gaussian hierarchical clustering. Tech. Report 311, Department of Statistics, University of Washington, 1996.

V. Guralnik and G. Karypis. A scalable algorithm for clustering sequential data. In *Proceedings of IEEE International Conference on Data Mining*, pp. 179–186, 2001.

G. Hamerly and C. Elkan. Learning the k in k-means. In *Proceedings of the seventeenth annual conference on neural information processing systems (NIPS)*, pp. 281–288, 2001.

M. Holub and M. Bielikova. Estimation of user interest in visited web page. In *Proceedings of the 19th international conference on World wide web (WWW)*, pp. 1111–1112, 2010.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8): 651–666, 2010.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):1–60, 1999.

B. Kaluža, V. Mirchevska, E. Dovgan, M. Luštrek, and M. Gams. An agent-based approach to care in independent living. In *Proceedings of the International Joint Conference on Ambient Intelligence (AmI)*, pp. 177–186, 2010.

L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD)*, pp. 593–604, 2007.

L. A. Leiva. Mining the browsing context: Discovering interaction profiles via behavioral clustering. In *Adjunct Proceedings of the 19th conference on User Modeling, Adaptation, and Personalization (UMAP)*, pp. 31–33, 2011.

L. A. Leiva and E. Vidal. Assessing users' interactions for clustering web documents: a pragmatic approach. In *Proceedings of the 21st ACM conference on Hypertext and Hypermedia (HT)*, pp. 277–278, 2010.

L. A. Leiva and E. Vidal. Revisiting the K-means algorithm for fast trajectory segmentation. In *Proceedings of the 38th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2011.

L. A. Leiva and E. Vidal. Simple, fast, and accurate clustering of data sequences. In *Proceedings of the 17th international conference on Intelligent User Interfaces (IUI)*, pp. 309–310, 2012.

S. Lloyd. Least squares quantization in PCM. *IEEE Transations on Information Theory*, 28(2):129–137, 1982.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

G. W. Milligigan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

D. Mount and S. Arya. ANN: library for approximate nearest neighbor searching, 1998. Available at http://www.cs.umd.edu/~mount/ANN/.

F. Murtagh. A survey of recent advances in hierarchical clustering algorithms which use cluster centers. *Computing Journal*, 26(1):354–359, 1984.

B. Poblete and R. Baeza-Yates. Query-sets: Using implicit feedback and query patterns to organize web documents. In *Proceedings of the 17th International Conference on World Wide Web (WWW)*, pp. 41–50, 2008.

C. Sugar. *Techniques for Clustering and Classification with Applications to Medical Problems*. PhD thesis, Department of Statistics, Stanford University, 1998.

R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(2):411–423, 2001.

P. Trahanias and E. Skordalakis. An efficient sequential clustering method. *Pattern Recognition*, 22(4):449–453, 1989.

C. J. Veenman, M. J. T. Reinders, and E. L. Baker. A maximum variance cluster algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1273–1280, 2002.

J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of American Statistics Association*, 58(301):235–244, 1963.

J. Yu. General C-means clustering model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1197–1211, 2005.