Mining sensor data from complex systems

Vespier, U.

Chapter 6

Subsequences Clustering for Events Modeling

6.1 Introduction

In this chapter, we investigate how to build a model of traffic activity events, such as passing vehicles or traffic jams, from measurements data collected in the context of Infrawatch, the Structural Health Monitoring project discussed in Section 2.4.

It has been shown that the structural stress caused by heavy loads is one of the main causes of bridge deterioration. Because of this, we focus here on modeling traffic activity events in the strain measurements, such as passing vehicles or traffic jams. The produced model can then be employed for real-time event classification or detection of anomalous responses from the bridge.

A single moving vehicle is represented in the strain measurements as a bump-shaped peak (see Figure 6.1 (right)) with an intensity proportional to the vehicle's weight and a duration in the order of seconds. On the other hand, events like traffic jams represent significantly larger time spans and cause an overall increase in the average strain level, due to the presence of many slow-moving vehicles on the bridge. Because we are dealing with events of

varying nature, straightforward algorithms based on peak detection will not suffice.

In order to model all the different kinds of traffic events represented in the strain data, we investigate the effectiveness of time series subsequence clustering [41, 49, 30, 43], which essentially employs a sliding window technique to split the data stream in individual subsequences (observations), which can then be clustered. However, the naive implementation of subsequence clustering (SSC) using a sliding window and k-Means is controversial, as it is prone to producing undesirable and unpredictable results, as was previously demonstrated and analyzed in several publications, e.g. [49, 30, 43]. Indeed, within our strain data application, we notice some of the mentioned phenomena, although not all. We provide an analysis of how the different phenomena can be explained, and why some of them are not present in the data we consider. Finally, we introduce a novel *Snapping* distance measure which, employed in SSC based on k-Means, removes the artifacts and produces a correct clustering of the traffic events. We believe that the proposed distance measure can lead to a rehabilitation of SSC methods for finding characteristic subsequences in time series.

6.2 InfraWatch and the Strain Sensor Data

In this section, we describe what the strain data looks like and how the different types of traffic activities are represented in the strain measurements, in order to motivate the technical solutions employed in Section 6.3.

As mentioned, we focus on modeling traffic events, such as vehicles passing over the bridge or traffic jams, represented in the strain measurements of the InfraWatch data. The data is being sampled at 100 Hz which amounts to approximately $8.6 \cdot 10^6$ measurements per sensor per day. As the sensor network is highly redundant, and the different strain sensors are fairly correlated or similar in behavior, we selected one sensor that is reliable and low in measurement-noise (less than 1.0 $\mu m/m$). The strain gauge considered is placed at the bottom of one of the girders in the middle of a 50 meter span

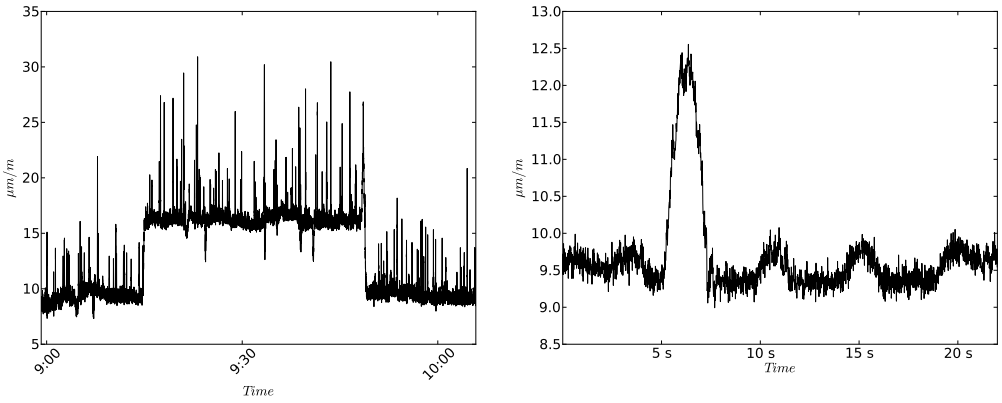

Figure 6.1: Detailed plots of strain, showing a traffic jam during rush hour (left) and individual vehicles (right).

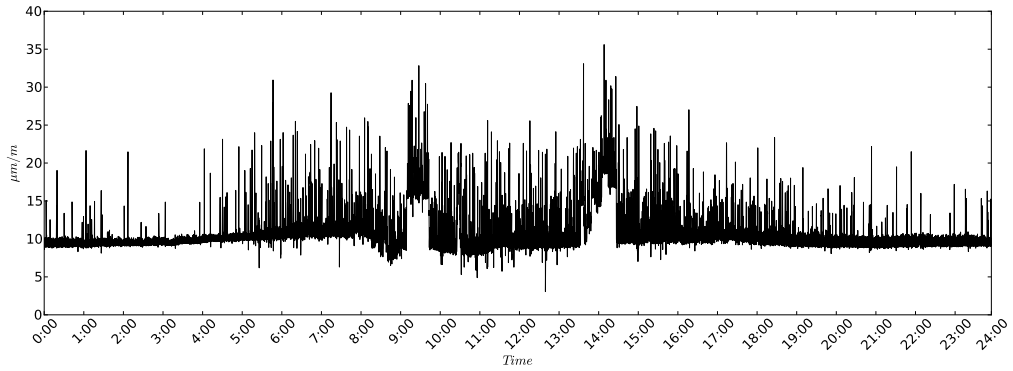

Figure 6.2: One full week day of strain measurements. All y-axis units in this chapter are in $\mu m/m$ (μ-strain).

near one end of the bridge. The strain data is thus related to this portion of the infrastructure. Every load situated on this span will have a positive effect, with loads in the middle of the span contributing more to the strain than loads near the supports of the span. Figure 6.2 shows an overall plot of the measurements for a single (week)day.

At the time scale of Figure 6.2, it is not possible to identify short-term changes in the strain level (except for notable peaks), such as individual vehicles passing over the span. However, long-term changes are clearly visible. For instance, there is a slightly curved trend of the strain baseline which slowly develops during a full day, which is due to changes in temperature,

slightly affecting both the concrete and gauge properties. The sudden rise of the average strain level between 9am and 10am is caused by a traffic jam over the bridge (as verified by manual inspection of the video signal). A traffic jam involves many slowly moving vehicles, which causes high vehicle densities. This in turn produces a heavy combined load on the span, and the strain measurements record this fact accordingly. Figure 6.1 (left) shows a detailed plot of the traffic jam event.

Short-term changes, on the other hand, can be identified when considering a narrower time window, in the order of seconds. A passing vehicle is represented in the data by a bump-shaped peak, reflecting the load displacement as the car moves along the bridge's span. Figure 6.1 (right) shows a time window of 22 seconds where the big peak represents a truck while the smaller ones are caused by lighter vehicles such as cars.

The examples above show how different traffic events, though all interesting from a monitoring point of view, occur with different durations and features in the strain data. Our aim is to characterize the different types of traffic the bridge is subjected to by analyzing short fragments of the strain signal, in the order of several seconds. The remainder of this chapter is dedicated to the clustering of such subsequences obtained by a sliding window.

6.3 Subsequence Clustering for Traffic Events Modeling

In this section, we introduce the rationale behind the subsequence clustering technique. We review the known pitfalls of SSC considering the features of the strain data and we show how its naive application produces results affected by artifacts. We finally propose a novel distance measure for SSC designed to remove the artifacts.

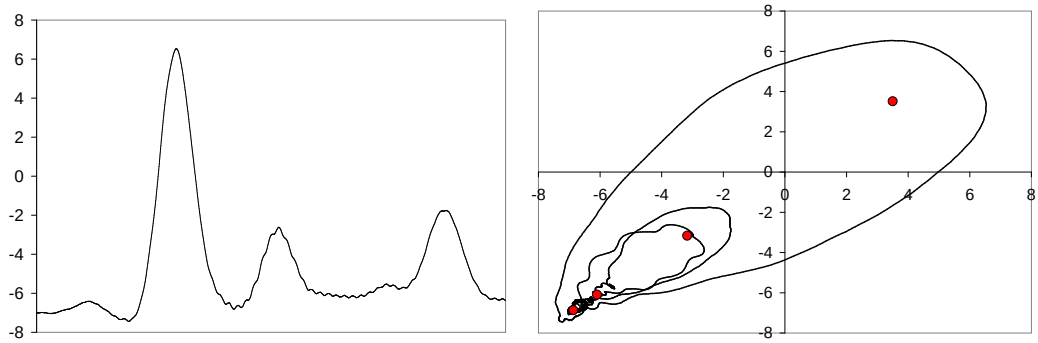

Figure 6.3: Two plots of the same data, showing the original data as a function of time (left), and a projection on two selected dimensions in w-space and the four prototypes generated by k-Means (red circles). Clearly, the sliding window technique creates a trajectory in w-space, where each loop corresponds to a bump in the original signal.

6.3.1 Subsequence Clustering

Subsequence clustering is a time series clustering techniques based on the concept of *subsequences set*:

Definition (Subsequences Set). The subsequences set $D(X, w) = \{S_{i,w} \mid 1 \leq i \leq m - w + 1\}$ is the set of all the subsequences extracted by sliding a window of length w over the time series X.

The subsequences set $D(X, w)$ contains all possible subsequences of length w of a time series X. The aim of *subsequence clustering* is discovering groups of similar subsequences in $D(X, w)$. The intuition is that, if there are repeated similar subsequences in X, they will be grouped in a cluster and eventually become associated to an actual event of the application domain.

6.3.2 Subsequence Clustering equals Event Detection?

Subsequence clustering is an obvious and intuitive choice for finding characteristic subsequences in time series. However, in a recent paper by Keogh et al. [49], it was shown that despite the intuitive match, SSC is prone to a number of undesirable behaviors that make it, in the view of the authors,

unsuitable for the task at hand. Since then, a number of papers (e.g. [43] and [30]) have further investigated the observed phenomena, and provided theoretical explanations for some of these, leading to a serious decline in popularity of the technique. In short, the problematic behavior was related to the lack of resemblance between the resulting cluster prototypes and any subsequence of the original data. Prototype shapes that were observed were collections of smooth functions, most notably sinusoids, even when the original data was extremely noisy and angular. More specifically, when the time series were constructed from several classes of shorter time series, the resulting prototypes did not represent individual classes, but rather were virtually identical copies of the same shape, but out of phase. Finally, it was observed that the outcome of the algorithm was not repeatable, with different random initializations leading to completely different results.

The unintuitive behavior of SSC can be understood by considering the nature of the subsequence set $D(X, w)$ that is the outcome of the initial sliding window step. Each member of $D(X, w)$ forms a point in a Euclidean w-dimensional space, which we will refer to as w-space, illustrated in Figure 6.3. As each subsequence is fairly similar to its successor, the associated points in w-space will be quite close, and the members of $D(X, w)$ form a trajectory in w-space. Figure 6.3 shows an example of a (smoothed) fragment of strain data, and its associated trajectory in w-space (only two dimensions shown). Individual prototypes correspond to points in w-space, and the task of SSC is to find k representative points in w-space to succinctly describe the set of subsequences, in other words, the trajectory. Figure 6.3 (right) also shows an example of a run of k-Means on this data. As the example demonstrates, the prototypes do not necessarily lie along the trajectory, as they often represent an (averaged) curved segment of it.

So how does SSC by k-Means fare on the strain data from the Hollandse Brug? Experiments reported in Section 6.4 will show that not all the problematic phenomena are present in clustering results on the strain data. In general, cluster prototypes do resemble individual subsequences, although some smoothing of the signal as a result of averaging does occur, which is only logical. The relatively good behaviour can be attributed to some cru-

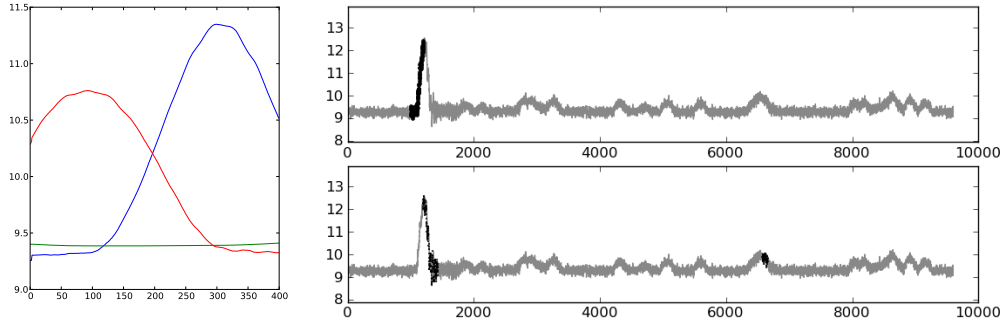

Figure 6.4: Multiple representation of events. The left plot shows the prototypes computed by the classic k-Means. The right plot shows, in black, the portion of the data assigned to the two bump-shaped prototypes.

cial differences between the nature of the data at hand, and that used in the experiments of for example [49, 43]. Whereas those datasets typically were constructed by concatenating rather short time series of similar width and amplitude, the strain data consists of one single long series, with peaks occurring at random positions. Furthermore, the strain data shows considerable differences in amplitude, for example when heavy vehicles or traffic jams are concerned. There remains however one phenomenon that makes the regular SSC technique unsuitable for traffic event modeling: the clustering tends to show multiple representations of what is intuitively one single event (see Figure 6.4 for an example). Indeed, each of the two bump-shaped prototypes resembles a considerable fraction of the subsequences, while at the same time having a large mutual Euclidean distance. In other words, our notion of traffic event does not coincide with the Euclidean distance, which assigns a large distance to essentially quite similar subsequences. In the next section, we introduce an alternative distance measure, which is designed to solve this problem of misalignment.

6.3.3 A Context-Aware Distance Measure for SSC

As showed in the previous section, applying SSC to the strain data employing the classic k-Means leads to undesirable multiple representations of the

same logical event. The problem is that comparing two subsequences with the Euclidean distance does not consider the similarity of their local contexts in the time series. Below we introduce a novel distance measure which finds the best match between the two compared subsequences in their local neighborhood.

Given a time series X and two subsequences $S_{p,w} \in X$ and S_{fixed} of length w, we consider not only the Euclidean distance between S_{fixed} and $S_{p,w}$, but also between S_{fixed} and the neighboring subsequences, to the left and to the right, of $S_{p,w}$. The minimum Euclidean distance encountered is taken as the final distance value between $S_{p,w}$ and S_{fixed}.

Formally, given a shift factor f and a number of shift steps s, we define the neighbor subsequences indexes of $S_{p,w}$ as:

$$NS = \left\{ \quad p + \frac{fw}{s} \cdot i \quad \middle| \quad -s \leq i \leq s \quad \right\}$$

The extent of data analyzed to the left and to the right of $S_{p,w}$ is determined by the shift factor while the number of subsequences considered in the interval is limited by the shift steps parameter. The *Snapping* distance is defined as:

$$Snapping(S_{p,w}, S_{fixed}) = \min\{Euclidean(S_{i,w}, S_{fixed}) \mid i \in NS\} \qquad (6.1)$$

We want to employ the *Snapping* distance in a SSC scheme based on k-Means. k-Means is a well known clustering/quantization method that, given a set of vectors $D = \{x_1, \ldots, x_n\}$, aims to find a partition $P = \{C_1, \ldots, C_k\}$ and a set of centroids $C = \{c_1, \ldots, c_k\}$ such that the sum of the squared distances between each x_i and its associated centroid c_j is minimized.

The classic k-Means heuristic implementation looks for a local minimum by iteratively refining an initial random partition. The algorithm involves four steps:

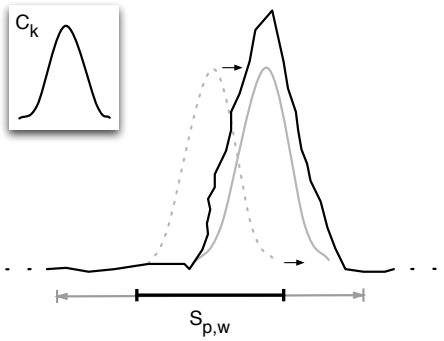

Figure 6.5: A subsequence $S_{p,w}$ is compared against the centroid C_k. The minimum Euclidean distance between C_k and the neighbor subsequences of $S_{p,w}$, including itself, is taken as a distance. Here, the best match is outlined in gray at the right of $S_{p,w}$.

1. (*initialization*) Randomly choose k initial cluster prototypes $c_1, \ldots, c_k$ in D.

2. (*assignment*) Assign every vector $x_i \in D$ to its nearest prototype c_j according to a distance measure. The classic k-Means uses the Euclidean distance.

3. (*recalculation*) Recalculate the new prototypes $c_1, \ldots, c_k$ by computing the means of all the assigned vectors.

4. Stop if the prototypes did not change more than a predefined threshold or when a maximum number of iterations has been reached, otherwise go back to step 2.

In our SSC scheme, the set of vectors D to be clustered is the subsequences set $D(X, w)$, where X is a time series and w the sliding window's length. In the assignment step, we employ the *Snapping* distance defined in Equation 6.1. Moreover, we force the initialization step to choose the random subsequences such that they do not overlap in the original time series. Figure 6.5 illustrates the intuition behind the *Snapping* distance measure in the context of k-Means clustering. In the next section, we evaluate this SSC scheme on the InfraWatch strain data.

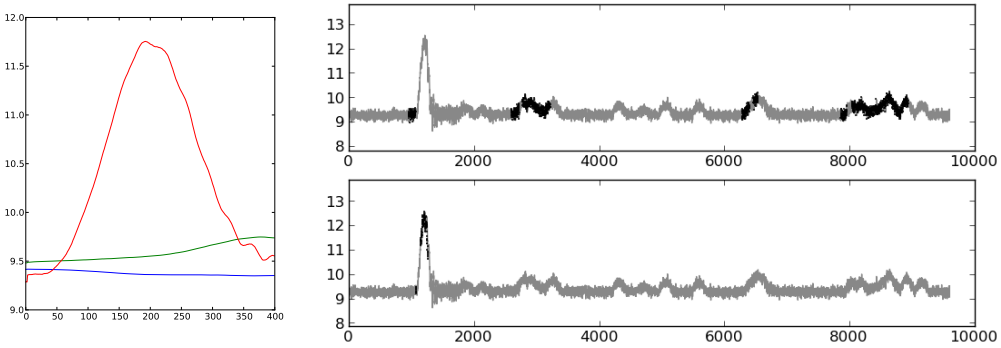

Figure 6.6: Improved results using the *Snapping* distance (see Figure 6.4).

6.4 Experimental Evaluation

In this section, we introduce the experimental setting and we discuss the results of applying the SSC scheme defined in Section 6.3.3 to the strain data.

We considered the following strain time series: `100Seconds` has been collected during the night in a period of low traffic activity across the Hollandse Brug, and consists of 1 minute and 40 seconds of strain data sampled at 100 Hz. The series contains clear traffic events and does not present relevant drift in the strain level due to the short time span. A more substantial series, `FullWeekDay`, consists of 24 hours of strain measurements sampled at 100 Hz, corresponding to approximately 9 millions values. The data has been collected on Monday 1st of December 2008, a day in which the Hollandse Brug was fully operational. All the traffic events expected in a typical weekday, ranging from periods of low activity to congestion due to traffic jams, are present in the data. The temperature throughout the chosen day varied between 4.9 and 7.7 degrees. Figure 6.2 shows an overall plot of the data.

In order to run the defined k-Means SSC scheme, we need to fix a number of parameters. The window length w has been chosen to take into account the structural configuration of the bridge and the sensor network. Considering the span in question is 50 meters long, and a maximum speed of 100 km/h,

a typical vehicle takes in the order of 2.5 seconds to cross the span. In order to capture such events, and include some data before and after the actual event, the window length was set to 400, which corresponds to 4 seconds. The number of clusters k directly affects how the resulting prototypes capture the variability in the data. For the `100Seconds` data, we found $k = 3$ a reasonable choice because, considering its short duration, the time series does not present drift in the strain baseline and the variability in the data can be approximated by assuming three kind of events: no traffic activity (baseline) and light and heavy passing vehicles. On the other hand, the `FullWeekDay` data presents much more variability, mostly due to the drift in the measurements which vertically translates all the events to different levels depending on the external temperature. Moreover, traffic jams cause underlying variability in the data. In the `FullWeekDay`, we found $k = 10$ to be large enough to account for most of the interesting, from an SHM point of view, variations in the time series, though we will also show the result with $k = 4$ for comparison. The f parameter affects the size of the neighborhood of subsequences considered by the *Snapping* distance. As the neighborhood gets smaller, the *Snapping* distance converges to the Euclidean. A big neighborhood, on the other hand, could include subsequences pertaining to other events. We experimented with $f = 0.25$, $f = 0.5$ and $f = 0.75$, yelding comparable outcomes. The presented results were all computed using $f = 0.5$. The shift steps parameter imposes a limitation on the number of Euclidean distances to compute for each comparison of a subsequence with a centroid; we fix it to $s = 10$.

6.4.1 Results

Given the chosen parameters, we run both the classic k-Means SSC and the *Snapping* distance variant on the `100Seconds` and `FullWeekDay` data.

Figure 6.6 depicts the results obtained by applying the k-Means SSC based on the *Snapping* distance on the `100Seconds` data. Comparing this with the results using the Euclidean distance on the same data in Figure 6.4, in this case, the big bump-shaped peak, caused by a heavy passing vehicle, is represented by a single prototype, while the remaining prototypes model

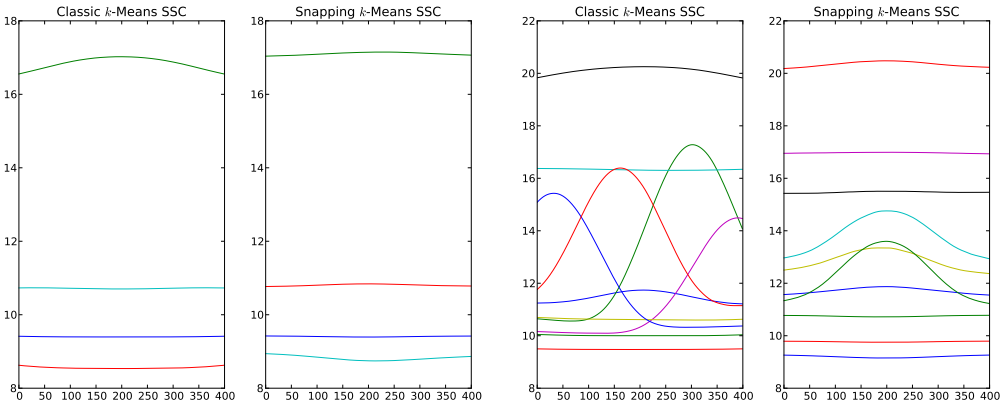

Figure 6.7: Prototypes produced by applying k-Means respectively with Euclidean and Snapping distance on the `FullWeekDay` data, for both $k = 4$ (left) and $k = 10$ (right).

lighter passing vehicles and the strain baseline (whose assignments are not shown in the picture).

Figure 6.7 shows the resulting prototypes obtained from the `FullWeekDay` data for $k = 4$ (left) and $k = 10$ (right). The prototypes computed for $k = 4$ by both the classic and revised k-Means SSC are really similar. Setting $k = 4$ does not account for all the variability in the `FullWeekDay` data and the resulting prototypes try to represent the different strain levels more than the actual events. In this case, the effect of considering the neighborhood of each subsequence, as done by the *Snapping* distance, is dominated by the presence of large differences in the strain values.

The prototypes for $k = 10$ better describe the variability in the data and represent both the different strain levels as well as the individual events (peaks). In this case, the classic k-Means SSC introduces double representations of the same logical events. This is avoided in our revised solution, thus better representing the variability in the data: every prototype now models a different strain level or event, as shown in Figure 6.7 (right).

Although Figure 6.7 gives an idea of the differences between the prototypes produced by the classic k-Means SSC and the *Snapping* version, it does not

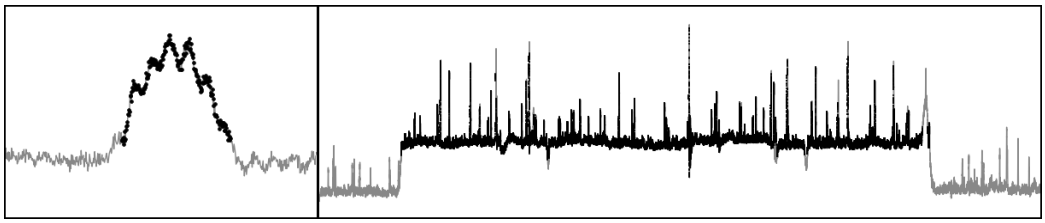

Figure 6.8: Two examples of events represented by individual prototypes. The central point of an associated subsequence is drawn in black.

show how the data is subdivided across them. Figure 6.8 shows two examples, at different time scales, of events associated to a single prototype. The plot on the left shows a heavy passing vehicle (in black), while the plot on the right shows all the subsequences considered part of a traffic jam event.

6.4.2 A Scalable Implementation

Given the amount of data generated by the sensor network, it is important to have a very scalable implementation of our clustering method. Therefore, we have developed a parallelized version based on the MapReduce framework using Hadoop [101]. Indeed, the main bottleneck in clustering lies in calculating the (snapping) distances between every subsequence and the cluster centers, which need to be read from disk. With MapReduce, we can distribute the data reads over a cluster of machines.

An overview of the resulting system is shown in Figure 6.9. In the first stage, we 'massage' the data to prepare it for the clustering phase. Since the computing nodes work independently, they need to be passed complete subsequences, including the lead-in and lead-out, in single records. First, we read the measurements of a single sensor for every timestamp, and its value is *mapped* to the initial timestamp ts of every subsequence in which it occurs. Then, all measurements for a specific ts are *reduced* to a complete subsequence.

In the clustering phase, we first select k random centroids. Then, each subsequence is mapped to the nearest centroid, using the snapping distance,

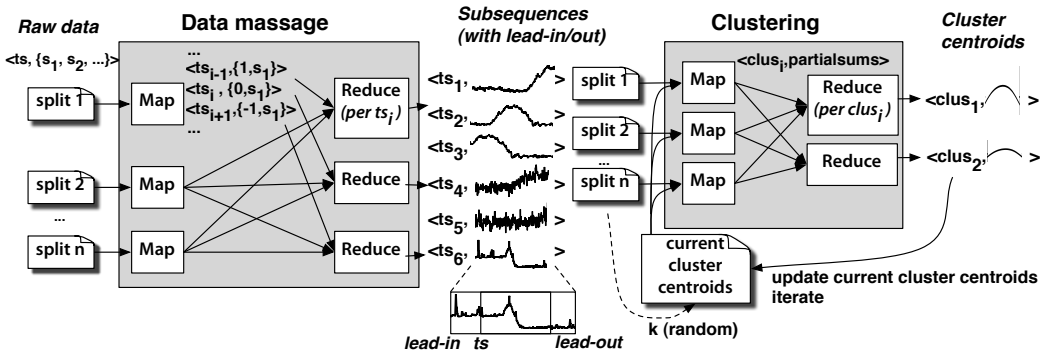

Figure 6.9: MapReduce implementation of our clustering method. Every map or reduce task can be run on any available computing core.

together with the combined points mapped by the same mapper. The reducer receives all points mapped to a certain cluster and calculates the new cluster centroid. This is repeated n times or until the clusters converge. The k-Means implementation is an adapted version of k-Means found in the Mahout library.[1]

We evaluated this implementation on a relatively small cluster of 5 quad-core computing nodes. Compared to a sequential implementation that loaded all the `FullWeekDay` data in memory, it yielded a 6-fold speedup in spite of the extra I/O overhead. Moreover, it scales linearly, even slightly sub-linearly, to time series of several months. For example, one month of data was clustered (using 10 iterations) in less than 14 hours, and can be sped up further by simply adding more nodes.

6.5 Conclusion

In this chapter, we have focused on the problem of identifying traffic activity events in strain measurements, produced by a sensor network deployed on a highway bridge. Characterizing the bridge's response to various traffic events represents an important step in the design of a complete SHM solution, as it will permit implementations of real-time classification or anomaly discovery

[1]Mahout — Scalable Machine Learning Library. `http://mahout.apache.org`

techniques.

The proposed solution is based on subsequence clustering, a technique shown to be prone to undesired behaviors and whose outcome is strongly dependent on the kind of data it is applied to. In view of this, we studied SSC in relation to the features of the strain data, showing that only some of the documented pitfalls (i.e., multiple representations) occur in our case. To solve this, we introduced a context-aware distance measure between subsequences, which also takes the local neighborhood of a subsequence into account. Employing this *Snapping* distance measure, we showed that SSC by k-Means returns a correct modeling of the traffic events.