Traffic Analysis Nice

May 20, 2021

1 Introduction

With the data at disposition of different areas of the city, the goal of the project is to analyze the traffic of Nice area in the year 2019 and 2020. The aim is to identify different seasonal and weekly paths of the traffic over the periods considered, and try to study the impact of the restrictions due to Covid2019 in the traffic dynamic. The data are used with respect their temporal order, daily time series are created with respect to the variables at disposition. In the first section the techniques and algorithms used are presented with a theoretical approach. In the second section Promenade Des Anglais data are considered, after having explain the pre-process implemented result are showed for both 2019 and 2020.

2 Methodology

In order to identify seasonal and weekly traffic paths, using daily time series, clustering procedures are proposed. Clustering is the task of dividing the time series data into a number of groups such that data in the same groups are more similar, present analogous behaviours during the day, than those in other groups. To better identify the path represented by a group (cluster) a barycenter, also call centroid, is computed. Since I am dealing with temporal data I have to define a strategy in order to compare different series both for assign each series in a cluster and update the centroid. I would need a dissimilarity measure that is able to "match" a point of time series x even with "surrounding" points of time series y. The Euclidean distance, assumes the $i-th$ point of the x series is aligned with the $i-th$ point of the y series, it will produce a pessimistic dissimilarity measure. Two clustering procedures are discussed below: Soft-Dynamic Time Warping with K-means algorithm and Global Alignment kernel (GAK) inside of a kernel K-means algorithm. The two procedure are connected each other and can be used together as showed later.

In addition to futher highlight long-term trends in the traffic behaviour a simple moving average calculation is taken in account. This techniques allows to analyze temporal data by creating a series of averages of different subsets of the full data set that delete short-term fluctuations.

2.1 Soft-Dynamic Time Warping and K-means algorithm.

Dynamic Time Warping is a technique to measure similarity between two temporal sequences considering not only the temporal alignment but every binary alignment of the two series. For example a similar traffic condition based on different variables could be recognized in different hours of the day in two different series. The calculation of the DTW similarity involves a dynamic programming algorithm that tries to find the optimum warping path between two series under certain constraints.

Given two multivariate series that corresponds to two different days:

$x \in R^{d \times n}$ and $y \in R^{d \times n}$ valued in R^d (d dimension of the multivariate time series).

Consider a function in order to compare different points of the two series ($x_i \in R^d$ and $y_j \in R^d$) $d \ : R^d \times R^d \Rightarrow R$, such as $d(x_i, y_j) = (\sum_{i,j=1}^{n} (|x_i - y_j|^p)$, where usually $p = 2$ and $d(x_i, y_j)$ is the quadratic Euclidean distance between two vectors.

A matrix of similarity is computed:

$\Delta(x, y) := [d(x_i, y_j)]_{i,j} \in R^{n \times n}$

$\Delta(x, y)$ can also be defined as local cost matrix, such a matrix must be created for every pair of series compared.

The DTW algorithm finds the path that minimizes the alignment between x and y by iteratively stepping through $\Delta(x, y)$, starting at $[d(x_i, y_j)]_{1,1}$ (bottom left) and finishing at $[d(x_i, y_j)]_{n,n}$ (upper right). and aggregating the cost [1]. At each step, the algorithm finds the direction in which the cost increases the least allowing 3 elementary moves $\rightarrow$, $\uparrow$, $\nearrow$. To improve the discrimination between different warped paths a chosen constraint can be specyfied. This constraint typically consists in forcing paths to lie close to the diagonal of the local cost matrix [2].

By considering $A_{n,n} \subset \{0, 1\}^{n,n}$ the set of all binary alignment between two time series x and y of the same lenght n, the DTW similarity measure reads as follow:

$$DTW(x, y) = \min_{A \in A_{n,n}} \langle A, \Delta(x, y) \rangle \tag{1}$$

This creates a warped "path" between x and y that aligns each point in x to the nearest point in y.

However I can not define dynamic time warping as a distance because does not satisfy the triangular inequality, moreover it is not differentiable everywhere due to the min operator.

Soft-Dynamic Time Warping is a variant of DTW that is differentiable. It uses the log-sum-exp formulation [3]:

$$DTW^{\gamma}(x, y) = -\gamma \log \sum_{A \in A_{n,n}} exp(-\frac{\langle A, \Delta(x, y) \rangle}{\gamma}) \ \ where \ \ \gamma \geq 0 \tag{2}$$

Despite considering all alignments and not just the optimal one, soft-DTW can be computed in quadratic time $O(d \times n^2)$ as DTW, however as DTW soft-DTW does not satisfy the triangular inequality. Soft-DTW is a symmetric similarity measure, it supports multivariate series as DTW, and it can provide differently smoothed results by means of a user-defined parameter γ.

The "path" created between x and y is smoother than the one created with DTW. Soft-DTW depends on a hyper-parameter γ that controls the smoothing. As showed in equation (3) DTW corresponds to the limit case when γ =0.

[1] Sardà-Espinosa:"Comparing Time-Series Clustering Algorithms in R Using the dtwclust Package" chapter 2

[2] Sakoe,Chiba: "Dynamic programming algorithm optimization for spoken word recognition," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 26(1), pp. 43–49

[3] Cuturi,Blondel:"Soft-DTW: a Differentiable Loss Function for Time-Series"

$$DTW^{\gamma}(x,y) = \begin{cases} \min_{A \in A_{n,n}} \langle A, \Delta(x,y) \rangle, & \gamma = 0 \\ -\gamma \log \sum_{A \in A_{n,n}} exp(-\frac{\langle A, \Delta(x,y) \rangle}{\gamma}), & \gamma \geq 0 \end{cases} \tag{3}$$

SoftDTW is then using with Centroid-based clustering, such as K-means and K-medoids. These algorithms use an iterative way to create the clusters by moving data points from one cluster to another, based on a distance measure, starting from an initial partitioning. [4].

Stepping into time series clustering, even if the number of data points is substantially large (ts time series with n daily observations and d dimension) K-means remains computationally attractive. The complexity of each iteration of the K-means algorithm performed on ts time series is $O(k \times ts \times n \times d)$ where k is the number of cluster. The time complexity considered to be "linear", due to $k, n \ll ts$ is one of the reasons for the popularity of the K-means clustering algorithm [5].

The soft-DTW is used in K-means algorithm to assign the series to the clusters and to upload the centroid of the cluster. Centroid in a cluster correspond to the multivariate time series $\in R^{d \times n}$ that minimizes the sum of the similarity measures between that time series and all time series inside the cluster. Given the ts multivariate time series each of them composed by n daily observations the algorithm work as follow:

```
Algorithm  $k - meansclustering \ (T, K)$
  $Input : T = (t_1, t_2, ..., t_{ts})$ set of daily time series where the generic series $t_i \in R^{d \times n}$
  $Input : k$ the number of clusters
  $Output : c_1, c_2, ..., c_k$ (partion of time series in cluster, centroid of each cluster $c_i \in R^{d \times n}$)
  $p = 0$
  Randomly choose $k$ series $\in R^{d \times n}$ and make them as initial centroids $(c_1^{(0)}, c_2^{(0)}, ..., c_k^{(0)})$
  repeat
    Assign each time series to the nearest cluster using $\arg\min_k DTW^{\gamma}(c_k^{(p)}, t_i)$
    $p = p + 1$
    // Centroid update
    for $j = 1$ to $k$ do
     Update the centroid $c_j^{(p)}$ by considering the subset of series $(t_1, t_2, ..., t_t$ where $t \leq ts)$ that
    end for
  until
 $c_j^{(p)} \approx c_j^{(p-1)} \quad j = 1, 2, ..., k$
  Return $c_1, c_2, ..., c_k$
```

Sometimes different initializations of the centroids lead to very different final clustering results. To overcome this problem the K-means algorithm is run 5 times with different centroids randomly placed at different initial positions. The final results will be the best output of the 5 times consecutive runs in terms of inertia.

[4]K-medoids clustering is very similar to the K-means clustering algorithm. The major difference between them is that while a cluster is represented with its center in the K-means algorithm, it is represented with the most centrally located data point in a cluster in the K-medoids clustering

[5]Soheily-Khah:" Generalized k-means based clustering for temporal data under time warp" Chapter 3

2.2 Global Alignment Kernel and Kernel K-Means

Kernel methods use a kernel function in order to separate high dimensional data. Given two time series x,$y \in R^{d \times n}$ A kernel $k(,)$ is defined as

$$k(x, y) = \langle\, \rho(x), \rho(y) \rangle_{\nu} \tag{4}$$

where an appropriate non linear mapping $\rho : R^{d \times n} \rightarrow \nu$ is used to compute the dissimilarity measure $k(x, y)$ in some (unknown) embedding space ν (also know as latent space). This approach is called the "kernel trick". It is used to map time series in a higher-dimensional feature space ν by computing the inner products between the images of pairs of data using a nonlinear function ρ. The "kernel trick" perform computations in the embedding space ν by the inner product of the transformed vectors $\rho(x), \rho(y)$ in the higher dimensional space. For example, one can compute distance between x and y in ν as

$$\begin{aligned} \|\rho(\mathbf{x}) - \rho(\mathbf{y})\|_{\nu}^{2} &= \langle \rho(\mathbf{x}) - \rho(\mathbf{y}), \rho(\mathbf{x}) - \rho(\mathbf{y}) \rangle_{\nu} \\ &= \langle \rho(\mathbf{x}), \rho(\mathbf{x}) \rangle_{\nu} + \langle \rho(\mathbf{y}), \rho(\mathbf{y}) \rangle_{\nu} - 2 \langle \rho(\mathbf{x}), \rho(\mathbf{y}) \rangle_{\nu} \\ &= k(\mathbf{x}, \mathbf{x}) + k(\mathbf{y}, \mathbf{y}) - 2k(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{5}$$

such computations are used in the Kernel K-means algorithm as illustrated later [6]. In Kernel K means before clustering, time series are mapped to a higher-dimensional feature space ν using a nonlinear functionρ, and then kernel K-means partitions the series by linear separators in the new space. Kernel methods require only a user-specified kernel $k(,)$, such as similarity function over pairs of data in raw representation.

The Global Alignment Kernel (GAK) is a kernel that operates on time series. Considering two time series x,$y \in R^{d \times n}$ I indicate with π a generic warping function, composed by a pair of integral vectors: π_1 for x and π_2 for y. π maps the generic alignment between two series from $\Delta(x, y)_{1,1}$ (bottom left) to $\Delta(x, y)_{n,n}$ (upper right) by allowing 3 elementary moves $\rightarrow$, $\uparrow$, $\nearrow$ as previoulsy showed. The (GAK) for a given bandwith σ is defined as:

$$k_{GAK}^{\gamma}(x, y) = \sum_{\pi \in \mathcal{A}(n,n)} \prod_{i=1}^{|\pi|} \exp\left(- \frac{\left\| x_{\pi_1(i)} - y_{\pi_2 i} \right\|^2}{2\sigma^2} \right) \tag{6}$$

where $|\pi|$ is the length of the generic alignment and the discrepancy between any two points x_i and y_i observed in x and y is the squared euclidean distance.

As reported in [7] the bandwidth σ can be set as a multiple of a simple estimate of the median distance of different points observed in different time-series of the training set, scaled by the square root of the length of time-series in the training set (In my case all daily multivariate time series have the same lenght previously indicated with n). The suggested estimation is

$\sigma \in \{0.1, 1, 10\} median||x - y||\sqrt{n}$

and it is available in **tslearn** [8].

[6]Dhillon,Guan,Kulis :"Kernel k-means, Spectral Clustering and Normalized Cuts"

[7]Cuturi:"Fast Global Alignment Kernels"

[8]Tavernard et.Al: "Tslearn, A Machine Learning Toolkit for Time Series Data"

The(GAK) is related to softDTW, it can be defined as the exponentiated soft-minimum of all aligment distances:

$$k^{\gamma}_{GAK}(x,y) = \sum_{A \in A_{n,n}} exp(-\frac{\langle A, \Delta(x,y) \rangle}{\gamma}) \tag{7}$$

where the γ hyperparameter that controls the smoothness in softDTW is linked with σ by $\gamma = 2\sigma^2$. Through this relation GAK can be used to estimate γ hyperparameter of softDTW.

Both GAK and softDTW, that scales in $O(d \times n^2)$, incorporates the set of all possible alignments $A \in A_{n,n}$ in the cost matrix $\Delta(x,y)$. For this reason they provide richer statistics than the minimum of that set, which is instead the unique quantities considered by DTW. However only Global Alignment Kernel (GAK) is positive definite [9].

Given a kernel $k^{\gamma}_{GAK}(,)$ and as input a set of daily time series $T = (t_1, t_2, ..., t_{ts})$ where the generic daily series $t_i \in R^{d \times n}$, the $ts \times ts$ matrix $K = (k^{\gamma}_{GAK}(t_i, t_j))_{ij}$ is called the Kernel matrix. Furthermore, the positive definiteness of kernel function translates $k^{\gamma}_{GAK}(,)$ in practice into the positive definiteness of the so called Kernel matrix K [10]. .

Positive definite Kernel Matrix positive can be used in kernel K-means algorithm.

The aim of kernel Kmeans algorithm is to partition the set of daily time series $T = (t_1, t_2, ..., t_{ts})$ in k clusters denoted as $\{C_j\}_{j=1}^{k}$. The algorithm introduce also weight for each time series denoted by w. A first significant difference (when compared to k-means) is that clusters centers are never computed explicitly, hence time series assignments to cluster are the only kind of information available once the clustering is performed. The algorithm only finds the "best" cluster representative in the embedding space. For the generic cluster j the "best" cluster representative is computed as:

$$m_j = \frac{\sum_{t_i \in C_j} w(t_i)\rho(t_i)}{\sum_{t_i \in C_j} w(t_i)} \tag{8}$$

The squared euclidean distance from $\rho(t_i)$ to m_j used to update the partitioning of the time series at every step read as:

$$\begin{aligned} \|\rho(t_i) - m_j\|^2_{\nu} &= \langle \rho(t_i) - m_j, \rho(t_i) - m_j \rangle_{\nu} \\ &= \langle \rho(t_i), \rho(t_i) \rangle_{\nu} + \langle m_j, m_j \rangle_{\nu} - 2 \langle \rho(t_i), m_j \rangle_{\nu} \end{aligned} \tag{9}$$

where the inner product of time series are computed using kernel function $k^{\gamma}_{GAK}(,)$ and contained in the kernel matrix K

The kernel K-means algorithm finally reads as follow:

[9] Blondel,Mensch,Vert:"Differentiable Divergences Between Time Series"
[10] Cuturi:"Positive Definite Kernels in Machine Learning" pag 9

```
Algorithm  $Kernelk-meansclustering\ \ (T,K)$
  $Input: K\ \ Kernel\ \ Matrix$
  $Input: k\ \ the\ \ number\ \ of\ \ clusters$
  $Output: C_1, C_2, ..., C_k\ \ (partion\ \ of\ \ time\ \ series\ \ in\ \ clusters)$
  $p = 0$
  $Randomly\ \ assign\ \ each\ \ series\ \ to\ \ a\ \ cluster\ \ (C_1^{(0)}, C_2^{(0)}, ..., C_k^{(0)})$
  repeat
    $For\ \ every\ \ cluster\ \ \{C_j\}_{j=1}^k\ \ find\ \ the\ \ "best"\ \ cluster\ \ representative\ \ m_j = \arg\min_m \sum_{t_i \in C_j} w(t_i)||\rho$
    $Update\ \ the\ \ partitioning\ \ of\ \ time\ \ series\ \ by\ \ considering\ \ \ the\ \ "best"\ \ cluster\ \ representative\ \ pre$
    $p = p + 1$
  until
 $C_j^{(p)} \approx C_j^{(p-1)}\quad j = 1, 2, ..., k$
  $Return\ \ c_1, c_2, ..., c_k$
```

A Second difference with respect to k-means is that clusters generated by kernel k-means are phase dependent rather than in shape. For instance considering two days that present same level of traffic congestion (same shape) reached in different hours (different phase) can be assigned to different clusters. This is because Global Alignment Kernel is not invariant to time shifts, as demonstrated in [11] for the closely related soft-DTW, however the difference in time shift becomes less emphatic when considering series of the same lenght.

As mentioned before GAK and softDTW are closely related and can be used together. GAK can be used to tune the hyperparameter γ in softDTW while the latter can be used to computed the cluster center once kernel k-means is computed. The barycenter corresponds to the time series that minimizes the sum of the distances between that time series and all the time series in the cluster.

2.3 Simple moving average

The moving average is commonly used with time series to smooth random short-term variations and to highlight other components such as trend present in the data. This technique is used to highlight the traffic trend of long periods (see most trafficated months), deleting the impact of the traffic peak in the morning and in afternoon. The simple moving average is the unweighted mean of the previous M data points. The selection of M (sliding window) depends on the amount of smoothing desired. By increasing the value of M improves the smoothing at the expense of accuracy. For a sequence of values, the simple moving average at time period t reads as follows:

$$SMA_t = \frac{x_t + x_{t-1} + x_{t-2} + ... + x_{M-(t-1)}}{M} \tag{10}$$

3 Promenade Des Anglais

The data at disposition are collected from 9 detectors positioned on the Promenade in both directions. 3 detectors collect the traffic in the north direction and 6 detectors collect the traffic in the

[11] Janati, Cuturi, Gramfort: "Spatio-Temporal Alignments: Optimal transport through space and time,"

south direction (facing the sea). Two period of time are registered from 01/06/2019 to 31/12/2019 and from 06/02/2020 to 30/12/2020. Two variables are reported for each lane of the road: the flow (n° veh/h) indicated as $q(x,t)$ and the occupancy rate of the detector in percentage indicated as $o(x,t)$. The data are recorded every minute. Rough data needs to be treated and aggregated. Here a list of operation computed in order to study the dynamics in both intervals time 2019 and 2020:

1. Duplicate removal.
2. Mark as missing observations of the occupancy rates that are greater than 50%.
3. Mark as missing observations of the flow that are greater than 60 veh/h and have an occupancy rate equal to 100%.
4. For every detector all lanes are aggregated together (any lane containing NaN, cause the resulting output column flow or occupancy to be NaN as well).
5. By looking at the fundamental diagram outlier observations are removed.
6. Fill jumps in the series due to the omission of data with Nans values for both flow and occupancy rate. 1430 omissions in 2019 period and 2401 omissions in 2020 period.
7. 6-minute averages aggregation to mitigate the impact of traffic lights.
8. Compute the percentage of missing values in the series in order to decide which detectors can be used in the analysis.
9. Fill Nans values of usable detector with an imputation procedure.

To apply the clustering procedure the NaNs values must be filled. The imputation procedure takes in account the temporal order of observations, thus if there are too many consecutive NaNs to fill the procedure is not applied. The procedure reads as follow: 1. If an isolated Nan value is recognized (both previous and next observations are present), the observation before the Nan value is propagate forward to fill the missing value.

2. If the number of consecutive Nan values is lower than 20 (2 hours). The consecutive Nans are filled with a linear method based on the temporal alignment of the previous observations available. Otherwise if the number of consecutive Nans exceed 2 hours the imputation is not performed.

Time series, flow and occupancy rate, are preprocessed using normalization over all periods (51360 observations for 2019 and 66480 observations for 2020 6-minute averages). This scaler is such that each output time series is in the range [0,1] allowing to have identical scales for time series with originally different scales (*veh/h* and \$ %\$):

$$q_{norm}(t,x) = \frac{q(t,x) - MIN}{MAX - MIN}$$

.

$$o_{norm}(t,x) = \frac{o(t,x) - MIN}{MAX - MIN}$$

.

The MinMaxScaler function from the Python machine learning library **Scikit-learn** [12] trasforms each values of the time series proportionally within the range [0,1] preserving the shape. Density

[12]Pedregosa et.Al "Scikit-learn: Machine Learning in Python"

and flow scaled series are not amplitude invariant, they do not have the same standard deviation (reached instead by using standardization).

The inverse_transform method of MinMaxScaler, that undo the scaling of a data point according to feature_range,

$$q(t,x) = q(t,x)_{norm} * (MAX - MIN) + MIN$$

$$o(t,x) = o(t,x)_{norm} * (MAX - MIN) + MIN$$

is used in the centroids representations to have a better comprehension of paths, no more in the [0,1] range.

Despite the $MIN \quad MAX$ for $q(t,x)$ and $o(t,x)$ are defined over the years 2019 and 2020, the clustering procedure is applied to the ts daily multivariate time series with n daily observations (where n is equal to 240 observations with 6-minute averages). The choice of scaling the two varaibles with respect to the entire periods not with respect to the single daily time series is done to preserve variability with respect to different days of the week. For example I simple assume that the MAX flow reached on Sunday or Saturday is really low compared to MAX flow reached on another working day.

To create the ts daily normalized multivariate time series a simple **reshape** procedure is applied:

$[ts * n, 2] \implies [ts, n, d]$ where $d = 2$ is the dimensionality of the daily multivariate time series (flow and occupancy rate).

3.1 2019

3.1.1 C601

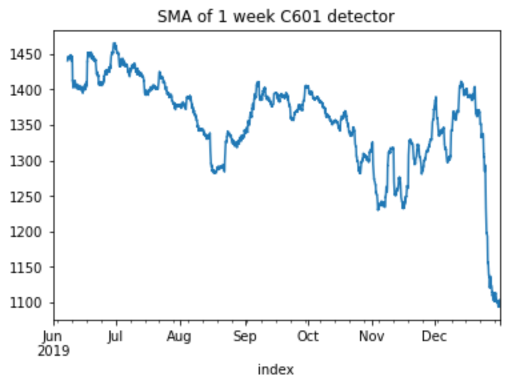

Figure 3.1 Simple moving average 2019 period

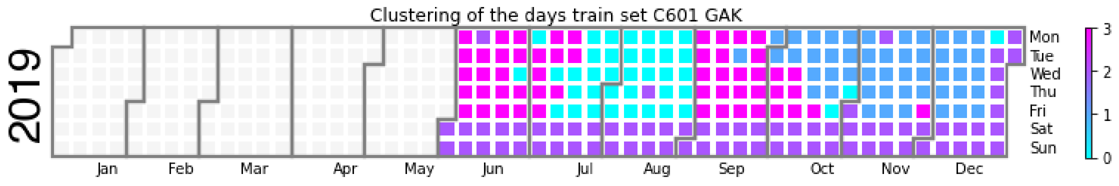

Figure 3.2 Kernel K-means, four clusters 2019 period

centroids path after Kernel K-Means

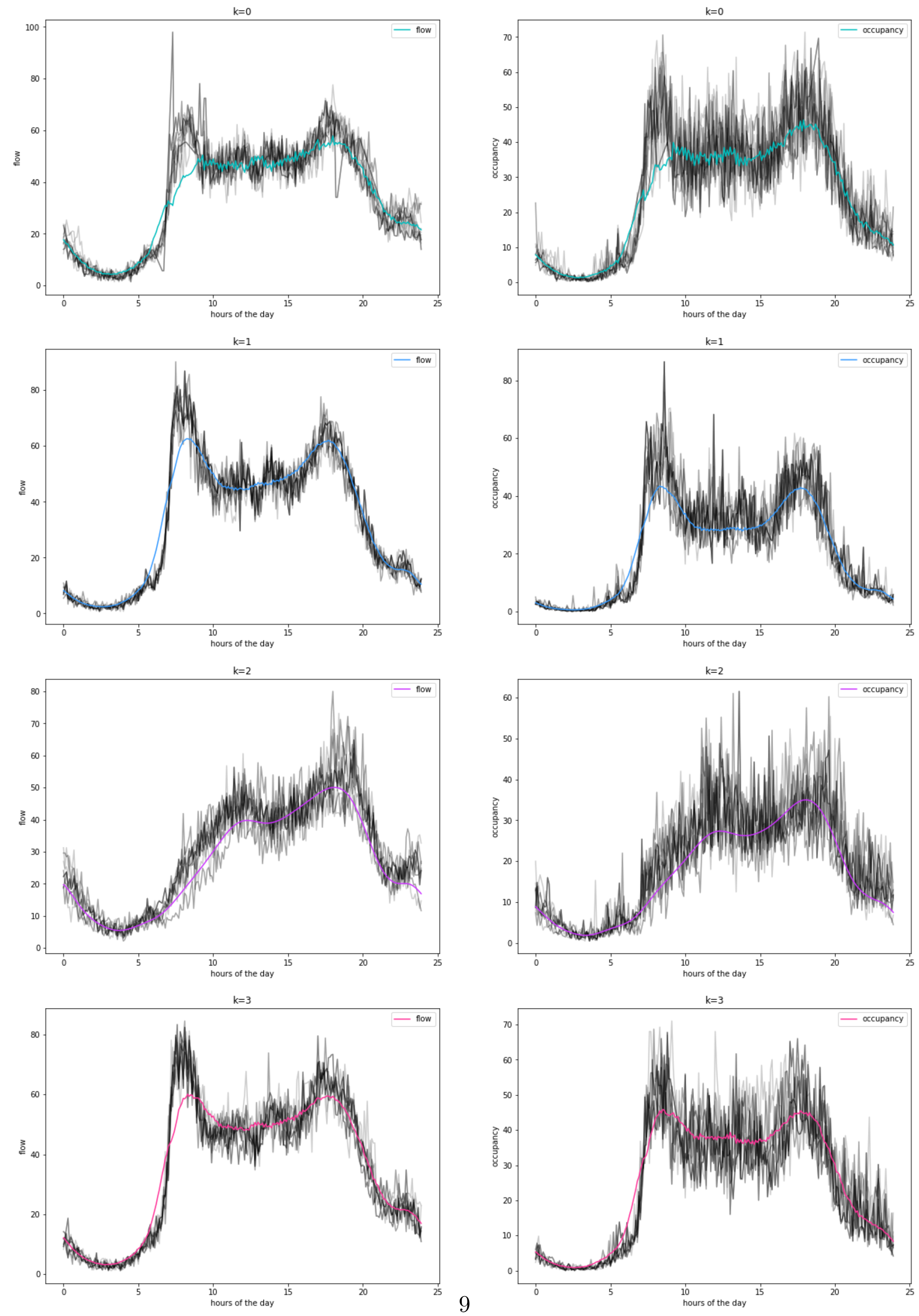

Figure 3.3 Centroids of Kernel K-means, four clusters 2019 period, computed with softDTW

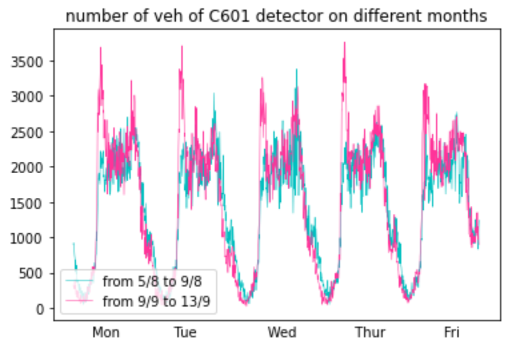

Figure 3.4 Comparison of working days in a week of August and September 2019

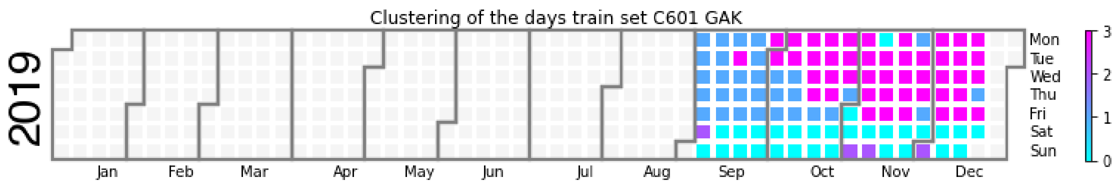

Figure 3.5 Kernel K-means, four cluster from 2/9/2019 to 21/12/2019

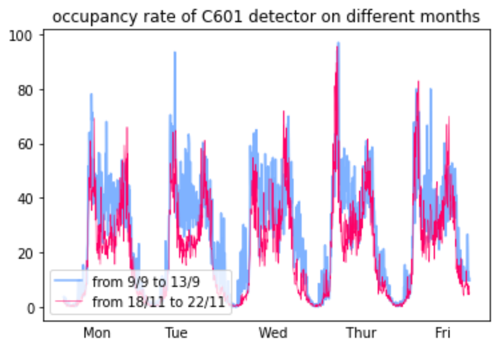

Figure 3.5 Comparison of working days in a week of September and November 2019

3.1.2 C009/C094

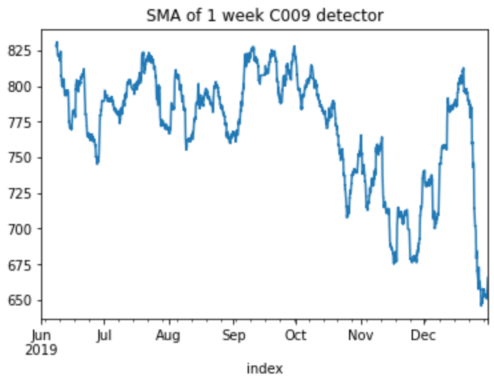

Figure 3.5.5 Simple moving average 2019 period

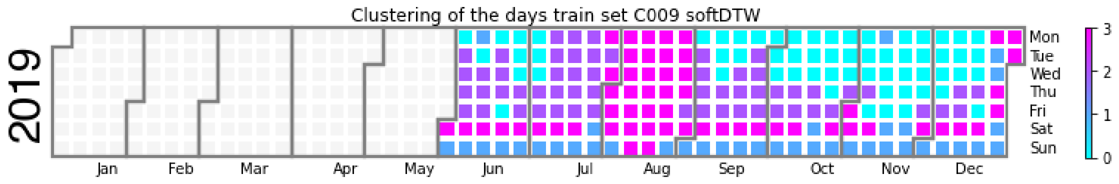

Figure 3.6 K-means, four clusters 2019 period

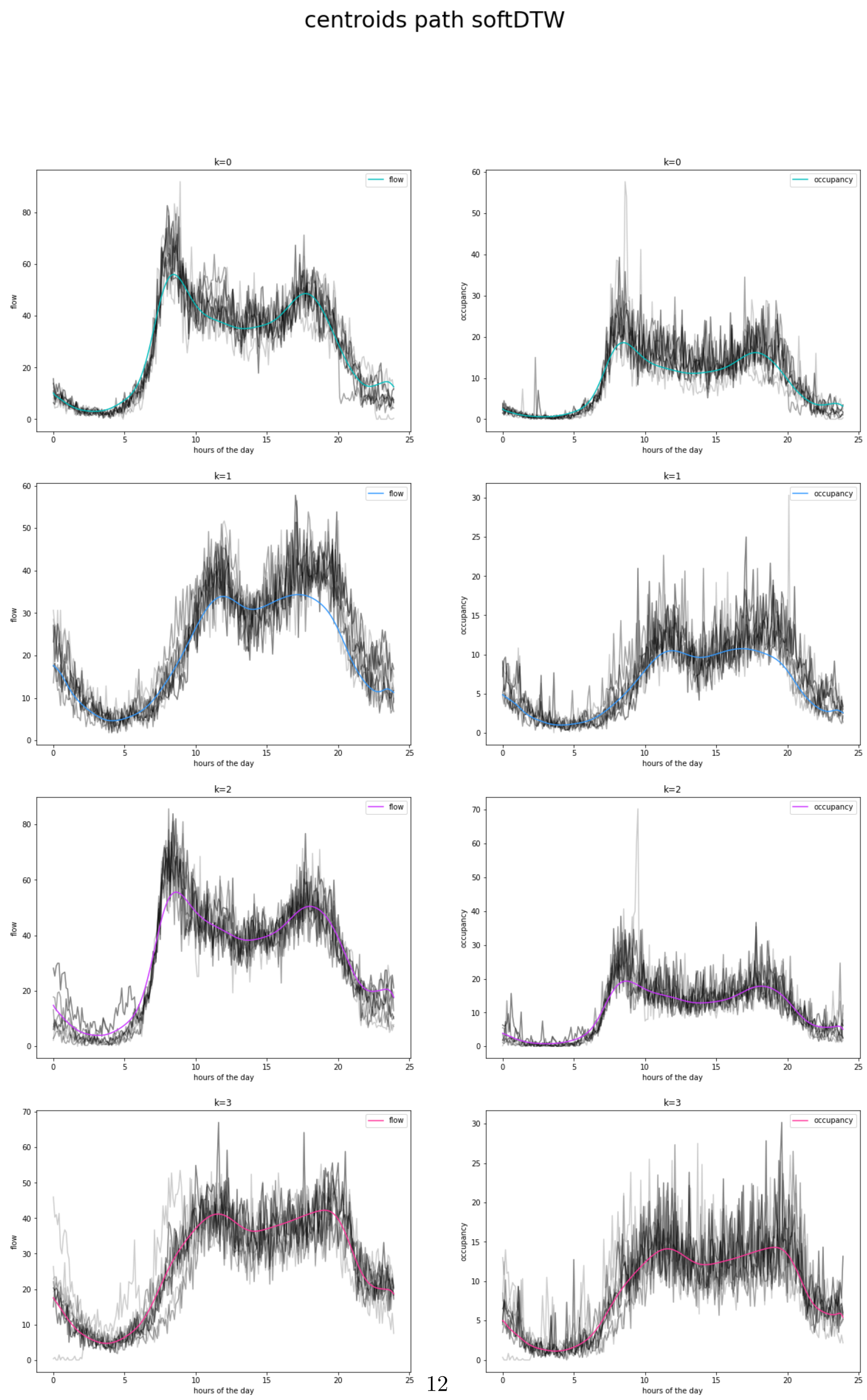
centroids path softDTW
k=0
k=1
k=2
k=3
flow
occupancy
hours of the day

Figure 3.7 Centroids of K-means, four clusters 2019 period

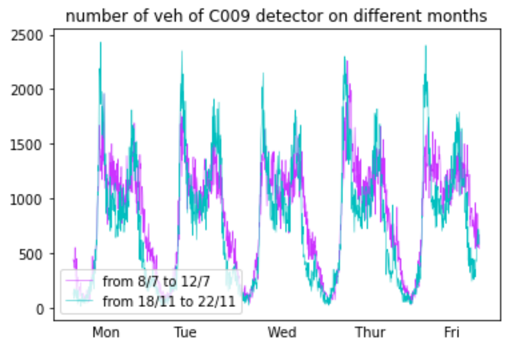

Figure 3.8 Comparison of working days in a week of July and November 2019

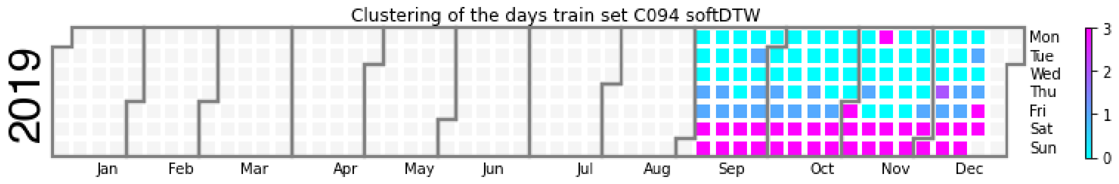

Figure 3.9 K-means,four cluster from 2/9/2019 to 21/12/2019

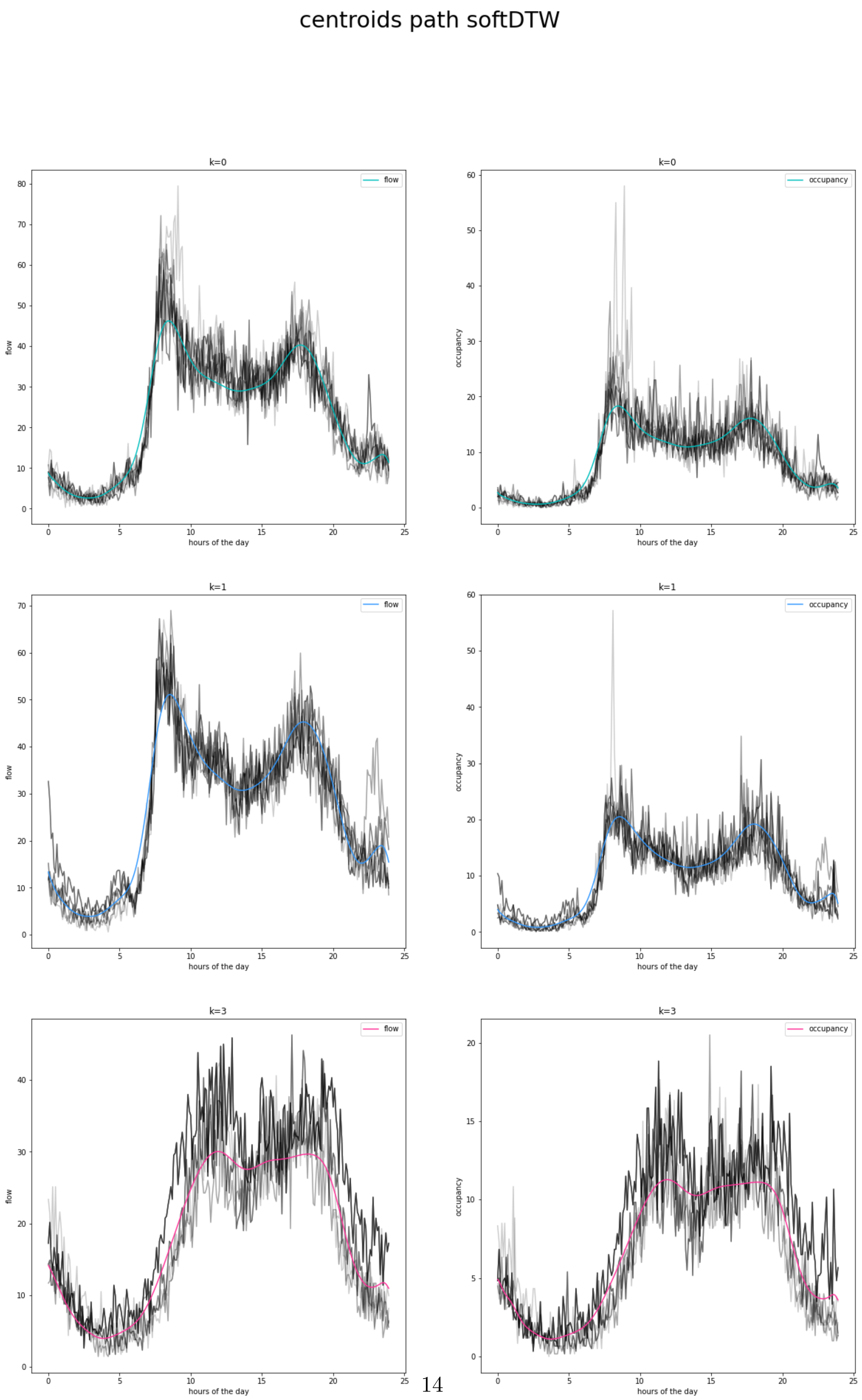
centroids path softDTW
k=0
flow
occupancy
k=1
flow
occupancy
k=3
flow
occupancy
hours of the day

Figure 3.10 Centroids of K-means, four clusters from 2/9/2019 to 21/12/2019

3.2 2020

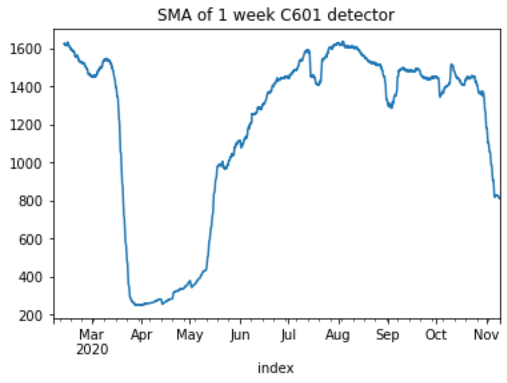

Figure 3.11 Simple moving average 2020 period

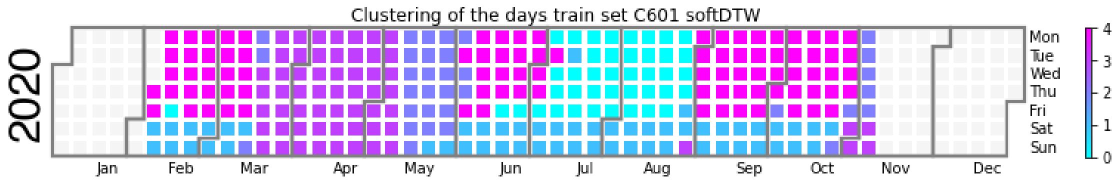

Figure 3.12 K-means, five clusters 2020 period

centroids path softDTW

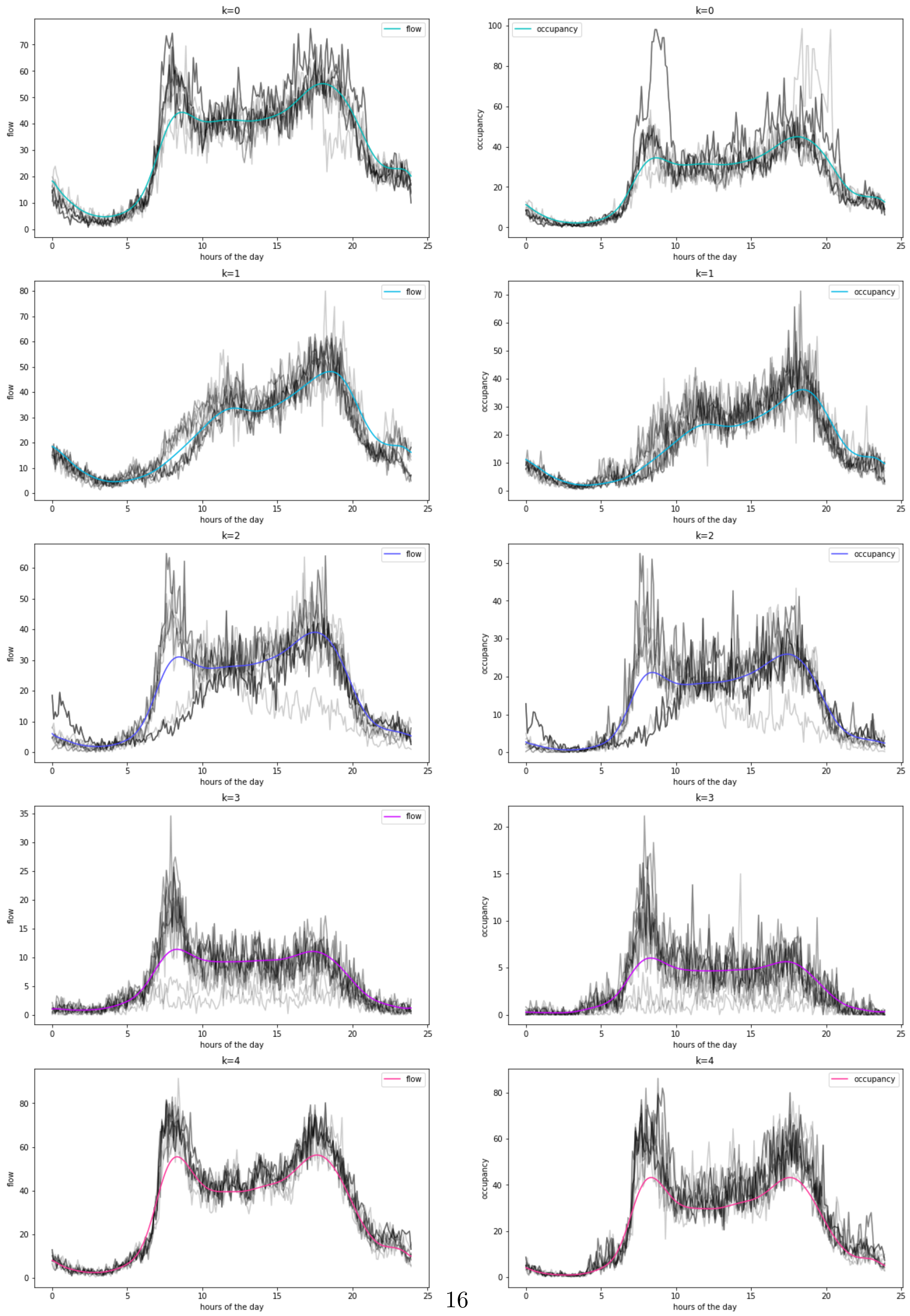

Figure 3.13 Centroids of K-means, five clusters 2020 period

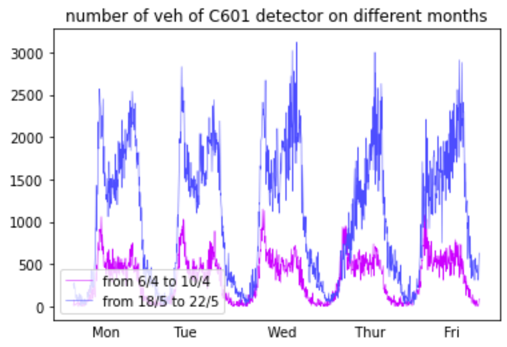

Figure 3.14 Comparison of working days in a week of April and May 2020

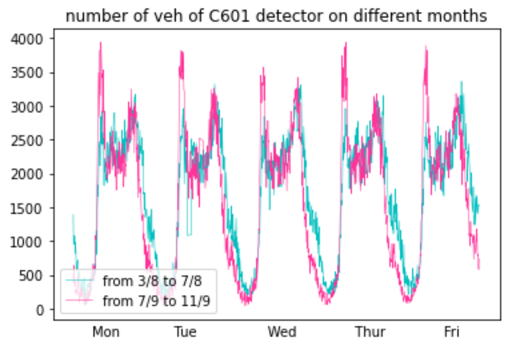

Figure 3.15 Comparison of working days in a week of August and September 2020

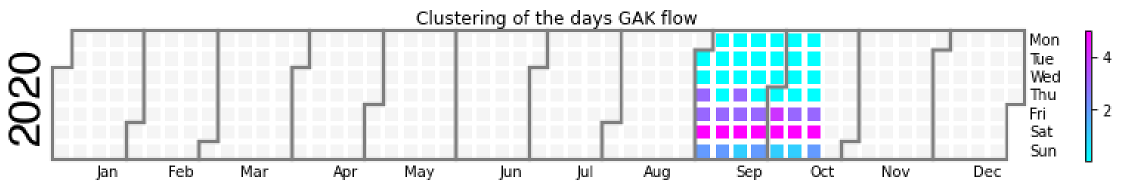

Figure 3.16 kernel K-means on flow of all detectors, six clusters from 1/9/2020 to 17/10/2020

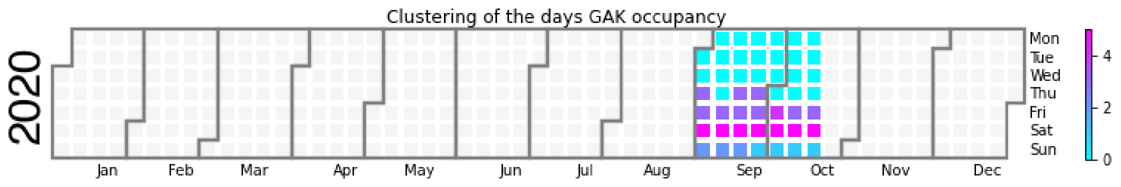

Figure 3.17 kernel K-means on occupancy rates of all detectors, six clusters from 1/9/2020 to 17/10/2020

3.3 2019 vs 2020

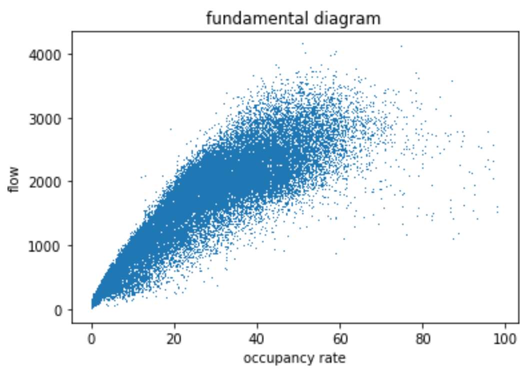

Figure 3.18 Fundamental diagram 2019

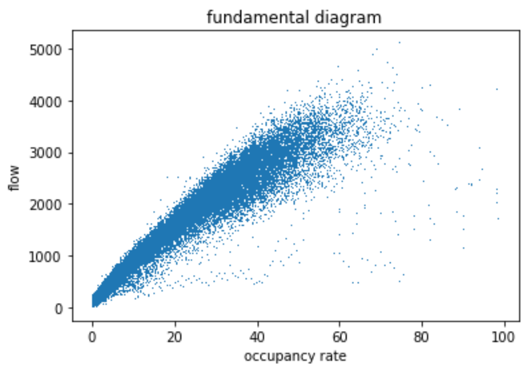

Figure 3.19 Fundamental diagram 2020

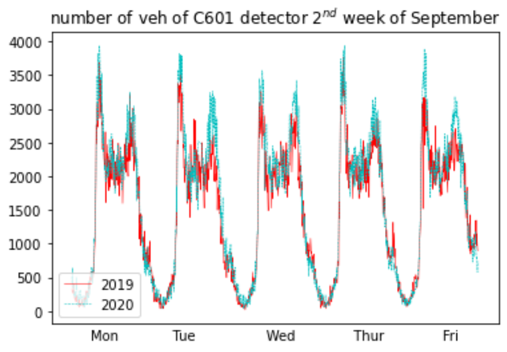

Figure 3.20 Comparison September 2020 and September 2019

3.4 Number of Optimal Clusters, K

K-means requires number of clusters K, as clustering parameter. Getting the optimal number of clusters is very significant in the analysis. If,for example, K is too high each time series starts representing a own cluster.

There is no a unique approach for finding the right number of clusters, I take in account a clustering quality measure, the soft-DTW similarity measure between nearest centroids and an empirical method to fix a minimum number of time series in each cluster.

Since Soft-DTW is differentiable it could be also used as a function to evaluate the cohesion inside each cluster and the separation with respect to the nearest cluster. The silhouette coefficient is a measure of how similar a time series is to its own cluster (cohesion) compared to other clusters (separation). The silhouette can be computed with the soft-DTW metric, it takes values in the range [-1, 1] [13].

Assume that the time series have been clustered via K-means. For time series $t_i \in C_k$ (time series i in the cluster C_k) $a(t_i)$ is the mean distance between time series t_i and all other time series in the same cluster:

$a(t_i) = \frac{1}{|C_k|-1} \sum_{j \in C_k} DTW^{\gamma}(t_i, t_j)$.

$b(t_i)$ is defined as the mean dissimilarity of the time series $t_i \in C_k$ to the nearest cluster C_z (where $C_k \neq C_z$) as the mean of the distance from the time series t_i to all time series $\in C_z$:

$b(t_i) = \min_{k \neq z} \frac{1}{|C_z|} \sum_{j \in C_z} DTW^{\gamma}(t_i, t_j)$.

Finally the silhouette coefficient for a time series is computed as follow:

$s(t_i) = \frac{b(t_i) - a(t_i)}{max\{a(t_i), b(t_i)\}} \quad if \quad |C_k| > 1$

The coefficients for every time series are averaged to have a global measure. The Mean Silhouette Coefficient for all time series is computed for the different number of clusters considered in the K-means algorithm to see how the number of clusters afflicts the analysis.

I consider also the soft-DTW similarity measure between the nearest clusters, by taking in account their centroids and applying equation (2) :

- When $K = 2$ becomes $DTW^{\gamma}(C_1, C_2)$ where C_1 and C_2 are centroids of the clusters.
- When $K = d$ where $d \geq 2$, $DTW^{\gamma}(C_j, C_z)$ is applied to all possible binary combinations of centroids forming a symmetric matrix of dimension $(d \times d)$ in which the minimum similarity between two centroids is selected.

As the number of clusters K increases the nearest clusters become closer each other until values of soft-DTW are stabilazed **??**.

In addition as the number of clusters increases is more difficult to interpret the traffic behaviour captured by each cluster. To deal with this problem, following an empirical method [14], I fixed a lower bound for the minimum number of observations in each cluster approximately at $\sqrt{\frac{3}{2} * N}$. Where N is the number of daily time series (365). If the number of time series within a cluster is lower than the fixed bound the cluster does not generalize well a particular traffic path, thus the interpretability of the cluster is too difficult.

The code is implemented with the Python machine learning library for time series **tslearn** [15].

[13]Rousseeuw: "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis" pag 53-65
[14]Zhang et.Al: "An empirical study to determine the optimal k in Ek-NNclus method" chapter 1
[15]Tavernard et.Al: "Tslearn, A Machine Learning Toolkit for Time Series Data"

4 References

[1] Pedregosa, Varoquaux, Gramfort, Michel, Thirion. “Scikit-learn: Machine Learning in Python”.

[2] Sardà-Espinosa. “Comparing Time-Series Clustering Algorithms in R Using the dtwclust Package”.

[3] Sakoe, Chiba. “Dynamic programming algorithm optimization for spoken word recognition,” IEEE Transactions on Acoustics, Speech and Signal Processing.

[4] Cuturi, Blondel. “Soft-DTW: a Differentiable Loss Function for Time-Series”.

[5] Saeid Soheily-Khah. “Generalized k-means based clustering for temporal data under time warp”.

[6] Rousseeuw. “Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis”.

[7] Zhang, Bouadi, Martin. “An empirical study to determine the optimal k in Ek-NNclus method”.

[8] Tavenard, Faouzi, Vandewiele, Divo, Androz, Holtz, Payne, Yurchak. “Tslearn, A Machine Learning Toolkit for Time Series Data”.

[9] Treiber, Martin, Arne Kesting. “Traffic flow dynamics.”

[10] The code implemented is available at https://github.com/NicolaRonzoni/Multivariate-Time-series-clustering.git