Project Number 288094

eCOMPASS

eCO-friendly urban Multi-modal route PlAnning Services for mobile uSers

STREP
Funded by EC, INFSO-G4(ICT for Transport) under FP7

eCOMPASS – TR – 044

Time Series Segmentation

T. Diamantopoulos, D. Kehagias

September 2013

Project Number 288094

eCOMPASS

eCO-friendly urban Multi-modal route PlAnning Services for mobile uSers

STREP
Funded by EC, INFSO-G4(ICT for Transport) under FP7

eCOMPASS – TR – 044

Time Series Segmentation

T. Diamantopoulos, D. Kehagias

September 2013

Time Series Segmentation

1 Introduction

This document provides information about the idea of segmenting a time series to identify segments that have constant trend. When trying to predict the next value of a series, as in a traffic scenario, it is common that the series has different behavior (e.g. trend) on different segments. With respect to traffic prediction scenarios, we first analyze the data for the city of Berlin and plot their mean to understand the nature of the series. After that, we demonstrate a way of segmenting the time series of the roads. Thus, finally we are able to apply well known time series algorithms (i.e. Space Time Auto-Regressive Integrated Moving Average (STARIMA)[1, 2]) in different segments.

2 Taking a look into traffic time series

Figure 1 depicts the mean of all road timeseries for the first and the second Monday of the dataset. The x axis corresponds to time intervals. Since each time interval is 5 minutes, the total number of intervals for 24 hours are 288[1].

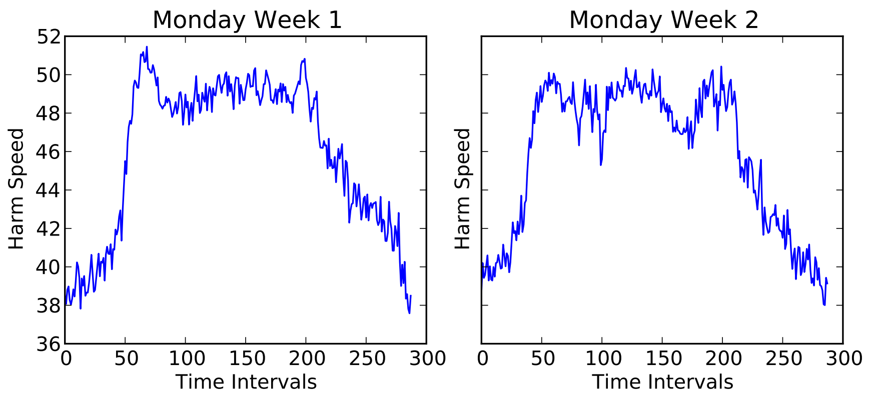

Figure 1: Example mean time series for the Mondays of week 1 and week 2.

[1]24 hours are $24 \cdot 60 = 1440$ minutes, so the number of 5-min intervals is $1440/5 = 288$.

As one can clearly observe in Figure 1, there are different trends that concern different times of the day. Around 7:00 in the morning (interval 72[2]), the traffic starts having higher values since more and more cars speed on the large roads of Berlin. The traffic is trend-free (actually not entirely trend-free but these trends will be analyzed later) for some period until 18:00 in the afternoon (interval 204[3]) and then it lowers again in the evening.

As one can clearly see in the above figure, the time series can be segmented to provide with clear representative segments. Concerning our traffic prediction scenario, the segmentation has to be automatic in order to be applicable on different datasets. Furthermore, since the data from the second week corresponds to input (testing) data, and is given in real time, it is necessary that the segmentation is done at an online manner. These problems will be approached in the following section.

3 Segmenting the time series

Time series segmentation is a well known problem in current literature [3, 4]. As noted in [4], there are three algorithms for time series segmentation: the sliding window algorithm, the top-down algorithm, and the bottom-up algorithm. The sliding window algorithm iterates over the time series values and for each new value it uses a sliding window containing past values in order to check whether certain error criteria are met to keep the value in the current window or if it should spawn a new segment. The top-down algorithm finds all possible partitioning of the time series and splits it using certain error criteria, while the bottom-up algorithm works by creating small segments and merging adjacent segments according to the same criteria.

Although all algorithms are known to be effective on different scenarios, not all of them are applicable in a real-time scenario such as the traffic one since the online criterion is not met. In specific, only the sliding window algorithm can be used. The pseudocode of the algorithm is shown in Figure 2. Note that Figure 2 illustrates a version of the algorithm different from that shown in [4], since the former adds also the `MinElementsPerSegment` parameter which sets a minimum size for each segment of the time series. One can clearly see that the algorithm is online since it iterates the time series once from its starting to its ending point. For each new segment, the algorithm has an `anchor` which is the starting point of the segment. Every time a new value comes, variable `i` increases and the error of the subseries containing the segment and the new value (`T[anchor:anchor+i]`) is checked against a threshold (`MaxError`). The values keep adding to the segment as long as the threshold is not surpassed. If the threshold is surpassed, the segment is saved (or simply returned for the online case), and a new segment is created by setting the anchor to the next value of the series.

[2]Note that 7:00 in the morning in Berlin corresponds to UTC time 6:00, which corresponds to $6 \cdot 60 = 360$ minutes from zero, thus the interval will indeed be $360/5 = 72$.

[3]Similarly, 18:00 is 17:00 UTC, i.e. $17 \cdot 60 = 1020$ minutes, so the interval is $1020/5 = 204$.

```
def SegmentTimeSeries(T, MaxError, MinElementsPerSegment):
  SegmentedT = []
  anchor = 0
  while anchor < len(Timeseries):
    if anchor + MinElementsPerSegment <= len(means):
      i = MinElementsPerSegment
    else:
      i = len(means) - anchor
    while (error(T[anchor:anchor+i]) < MaxError) and i<len(T)+1:
      i += 1
      SegmentedT.append(T[anchor:anchor + i])
      anchor += i
```

Figure 2: The sliding window algorithm for time series segmentation.

The application of the algorithm on the means of Figure 1 is shown in Figure 3 for both the Monday of the first week (training) and the second week (testing).

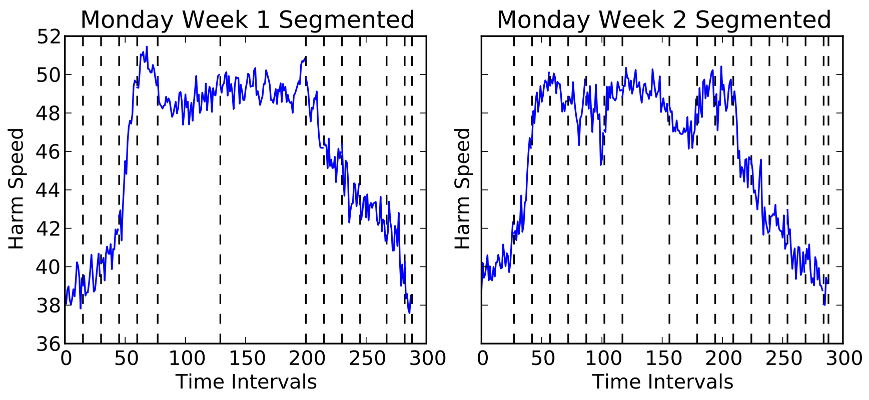

Figure 3: Segmented mean time series for the Mondays of week 1 and week 2.

As shown in Figure 3, the algorithm seems quite effective since it splits the time series to reasonable segments.

Upon applying the algorithm to the time series of each specific road (since the example of Figures 1 and 3 contains the mean time series of all roads), one can train different models for different training segments. The problem, however, lies in testing; selecting which of the models to use on each testing segment is a non-trivial issue. We implemented three different approaches towards this issue.

The first approach is the simplest. Intuitively, one can say that the segments for the Monday of the first week shall be similar to the ones for the Monday of the second week. For example, if there is heavy traffic at the morning hours of the first week, this is also expected to happen at the same hours of the second week. Thus, a simple solution indicates using the same segmentation for the testing set that we used for the input set, i.e. having predefined segments for the input sets corresponding to the respective intervals on the training set.

The aforementioned approach could yield satisfactory results in most cases. However, heavily relying on the periodicity of traffic is not optimal since, as shown in Figure 3, the segmentation for each set (training and testing) could be quite different. For example, in the first week one can observe a sudden rise in harmonic average speed within intervals 50 to 60. This rise, however, is observable in the second week within intervals 40 to 50. Ideally, we would like to map these two segments to each other. Thus, the problem is reduced to identifying which training segment corresponds to the input (testing) segment at hand. This is accomplished by introducing a metric that measures the similarity between the segment at hand and each of the training segments.

Immediate intuition indicates using correlation metrics. However, the segments may have different lengths, which makes the use of these metrics difficult. Instead we use two different approaches, one based on the mean and standard deviation of the segments, and one based on their trends. At first, one can simply compare two series T_1 and T_2 using their mean and standard deviation. The distance between the time series can then be defined as:

$$D(T_1, T_2) = |\mu_1 - \mu_2| + |\sigma_1 - \sigma_2| \tag{1}$$

where μ_1, μ_2 are the means and σ_1, σ_2 are the standard deviations of time series T_1 and T_2 respectively.

Although means and standard deviations are fundamental statistics of time series, their modeling potential is limited. Totally different time series may have similar means and even similar standard deviations. A metrics that is actually highly relevant to the problem at hand is the trend of a time series. Modeling the trend is not an easy procedure. In terms of this work, we decided to refrain from creating highly complex models (e.g. an ARMA) since the segmented time series are (and should be) too simple to support such models. Furthermore, complex models would be inefficient in such a scenario. Instead, we approximated the trend of the time series by finding the line that is close enough to all values of the series segment. The (deliberately simple) model is:

$$x = \alpha t + b \tag{2}$$

where x is the speed value for time t and α, β are the parameters of the model which are found using linear least squares approximation[4]. Thus, each time series segment is actually represented by a line denoting its trend. Selecting the

[4]The method is actually quite fast, not only because of having only two parameters, but also since the segments are generally small.

training segment that is closest to the testing segment at hand is a matter of combining parameters α_1 and β_1 of the series T_1 to the respective parameters α_2 and β_2 of the series T_2. This is accomplished by first finding the 5 nearest series given the distance of the β's, and then ending up to the series with the nearest α. Intuitively, the 5 nearest series given the distance of the β's have similar values with the series at hand (so they should have similar means and standard deviations as in the former method). Adding the constraint of the minimum distance of α's indicates the time series should also have similar slopes, i.e. trends.

4 Results

We set up a suitable scenario using training data for 24 hours from Monday of the first week and testing data for 24 hours from Monday of the second week. Following the methodology for creating the series (as in [2]), we tested our implementations for 1 to 6 intervals ahead of present time. We set `MaxError` to 15 and `MinElementsPerSegment` to 15 (see Figure 2) as these values provided optimal results. The total RMSE values are shown in Table 1.

Table 1: RMSE of Different Lag STARIMA implementations

Lag STARIMA	Lag STARIMA with predefined segments	Lag STARIMA with segments based on mean/std	Lag STARIMA with segments based on trend
2.982	3.015	3.066	3.052

As shown in Table 1, having a single model for all intervals proved optimal. Our three approaches perform slightly worse without, however, much difference.

5 Conclusion

Our analysis indicated that using different models for different time series segments is reasonable. The intuition of the segmentation step is generally well supported as to the different nature of each segment. When taking into account the statistics of each segment, including its trend, one can train a better model (in the sense that it should fit better).

However, the results of our analysis indicate no significant improvement over our algorithm. Achieving an improvement would probably involve fine-tuning the parameters (e.g. `MaxError` and/or `MinElementsPerSegment`) of our algorithms automatically. In addition, one could use different segmentation methods, or even create segments on a more coarse-grained level (e.g. mean speed as in the figures) and use them for all different roads (although this could produce better results only if all road time series are similar, which is not the case in general).

References

[1] Yiannis Kamarianakis and Poulicos Prastacos. Space-time modeling of traffic flow. *Comput. Geosci.*, 31(2):119–133, March 2005.

[2] Themistoklis Diamantopoulos, Dionysios Kehagias, Felix G. König, and Dimitrios Tzovaras. Investigating the effect of global metrics in travel time forecasting. To appear in *Proceedings of 16th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[3] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. In *In an Edited Volume, Data mining in Time Series Databases. Published by World Scientific*, pages 1–22. Publishing Company, 1993.

[4] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296, 2001.