Period-aware local modelling and data selection for time series prediction

Article *in* Expert Systems with Applications · October 2016

DOI: 10.1016/j.eswa.2016.04.024

Marcin Bernaś
University of Silesia in Katowice
26 PUBLICATIONS **48** CITATIONS

SEE PROFILE

Bartlomiej Płaczek
University of Silesia in Katowice
31 PUBLICATIONS **97** CITATIONS

SEE PROFILE

Period-aware local modelling and data selection for time series prediction

Marcin Bernas[*] and Bartłomiej Płaczek

Institute of Computer Science, University of Silesia, Będzińska 39, 41-200 Sosnowiec, Poland

marcin.bernas@gmail.com*, placzek.bartlomiej@gmail.com

Research highlights

- The introduced algorithm selects useful data for improved training of local models.
- A hybrid usefulness-related distance is proposed for training data selection.
- Data usefulness is evaluated by taking into account periodicity of time series.
- Autocorrelation function and Renyi entropy is used to reduce number of parameters.
- The proposed method offers lower prediction error than the state-of-the-art local and global models.

Abstract

The paper tackles with local models (LM) for periodical time series (TS) prediction. A novel prediction method is introduced, which achieves high prediction accuracy by extracting relevant data from historical TS for LMs training. According to the proposed method, the period of TS is determined by using autocorrelation function and moving average filter. A segment of relevant historical data is determined for each time step of the TS period. The data for LMs training are selected on the basis of the k-nearest neighbours approach with a new hybrid usefulness-related distance. The proposed definition of hybrid distance takes into account usefulness of data for making predictions at a given time step. During the training procedure, only the most informative lags are taken into account. The number of most informative lags is determined in accordance with the Kraskov's mutual information criteria. The proposed approach enables effective applications of various machine learning (ML) techniques for prediction making in expert and intelligent systems. Effectiveness of this approach was experimentally verified for three popular ML methods: neural network, support vector machine, and adaptive neuro-fuzzy inference system. The complexity of LMs was reduced by TS preprocessing and informative lags selection. Experiments on synthetic and real-world datasets, covering various application areas, confirm that the proposed period aware method can give better prediction accuracy than state-of-the-art global models and LMs. Moreover, the data selection reduces the size of training dataset. Hence, the LMs can be trained in a shorter time.

Keywords: local models, time series prediction, data reduction, segmentation, k-nearest neighbours, soft computing

[*] corresponding author. Tel.: +48 517 435 509

1. Introduction

Time series (TS) prediction is an active research topic, due to its application potential in many areas of science and industry. The TS prediction algorithms play a major role in decision-making processes for various applications, e.g., stock markets, climate changes, industrial management, and transportation. Over the past decade, much effort has been devoted to the fusion and improvement of conventional TS prediction models (Lin et al., 2011 and Mehdi & Bijari, 2011). Machine learning (ML) is an alternative approach to the TS prediction problem (Štěpnička et al., 2013). Recently, special attention was paid to the ML methods that are based on local prediction models. A local model (LM) is built "just in time", i.e., when a prediction is required, by using historical data that are similar to current observations (Kaneko et al., 2010). The LMs take advantage of the divide and conquer principle by splitting the global prediction problem into several sub-problems and adjusting a LM for a specific sub-problem (Martinez-Rego et al., 2011 and Wu & Lee, 2015).

These approach do not take under consideration the periodic character of TS during selection of the historical data (nearest neighbours) that are used for model training. In this paper a novel prediction method is proposed, which detects periodic changes in a TS and utilizes the information about TS periodicity, to extract relevant data for LM training.

The proposed prediction method consists of the following main steps. First, a period of the analysed TS is determined by using autocorrelation function and moving average filter (Box & Jenkins, 2008). Second, a usefulness relation is extracted from the TS. The usefulness relation enables selection of relevant data for training the LM at a given step of the TS period. Third, when a prediction query has to be processed, the similar historical data (k-nearest neighbours) are searched by taking into account a hybrid usefulness-related distance. Forth, the most informative lags for the selected data subseries are extracted in accordance with the Kraskov mutual information criteria (Kraskov & Stogbauer, 2004). Fifth, training of the LM is performed based on the selected data and finally, the prediction is made.

The novelty of the proposed method lies in selection of the training data that are expected to be useful for prediction making at a given step of TS period. There are two main contributions of this work: a ranking-based algorithm for extraction of the usefulness relation from periodical TS, and a definition of the usefulness-related hybrid distance, which enables selection of relevant historical data to train the LMs. Effectiveness of the proposed approach was confirmed in experiments with real-world and synthetic TS. The prediction accuracy was compared with that of the seasonal ARIMA (Box & Jenkins, 2008) as well as the LMs without period-aware training data selection (Wu & Lee, 2015).

The paper is organized as follows. Related works are reviewed in Section 2. Section 3 describes details of the proposed method. An example of training data extraction from periodical TS is presented in Section 4. Section 5 includes presentation and discussion of the experimental results. Finally, conclusions are given and future research directions are outlined in Section 6.

2. Related works and contribution

2.1 Prediction methods

TS prediction has been an active research area over last decade. The variety of applications brings a need for universal prediction tools as well as dedicated models that could be applied for a given problem. The dedicated models are commonly used in such areas as weather forecast (Liang et al., 2012), transportation (Płaczek, 2013) or medicine (Wu et al., 2003).

The development of the dedicated models requires detailed knowledge of the predicted processes. However, if detailed knowledge is not available, the development of dedicated model becomes a difficult task. In such case, the prediction model can be constructed based on historical data, by using ML algorithms and statistical data analysis to find a relation that enable prediction of future values. The historical data usually has the form of discrete TS that contains data points observed at constant time intervals.

If the prediction is made one time interval ahead into the future, it is called one-step or single-step forecasting (Gooijer & Hyndman, 2006). This type of prediction is used in real time applications, e.g., stock market exchange (Zatlavi et al., 2014) or traffic control (Bernas et al., 2015). A multi-step prediction refers to estimation of future values for more than one time interval ahead. Such prediction is used in long-term analysis, e.g., to forecast the climate change (Linag et al., 2012).

Over the past decade, major advances have occurred in statistical models and ML methods for TS prediction. In the literature, several linear prediction approaches were proposed. ARIMA model is one of the most popular prediction methods. This method can be used when the considered TS is stationary and no data are missing (Weigend & Gershenfeld, 1993). Several extensions of ARIMA have been proposed that enable applications for different types of TS (Box & Jenkins, 2008). These extensions include the seasonal models

(Khashei et al., 2012). The major drawback of those approaches is the pre-assumed linearity of the model and sensitivity to outliers (Khashei & Bijari, 2011).

Other statistical methods, like spectral analysis (Brillinger, 2011), Markov process (Zhou et al., 2014) and Kalman filter (Lin et al., 2012) are based on the probability theory and require prior knowledge of the underlying process.

The ML methods have been introduced to enable extraction of underlying characteristics for a predicted process without the prior knowledge and human intervention (Štěpnička et al., 2013). The most widely used ML method is based on artificial neural networks (ANNs). ANN became one of the most important nonparametric nonlinear TS prediction models. Main advantage of ANNs is the capability of flexible nonlinear function approximation with a desired accuracy (Cybenko, 1989). As a nonparametric and data-driven model, ANNs do not require additional assumptions before the model generation (Zhang et al., 1998).

Various problems and challenges are associated with ANNs. The selected weights and thresholds have mayor impact on the prediction results. When training the ANN, local optima can be found instead of the global optimum. Wang, Zeng, and Chen (2015) have proposed an adaptive differential evolution algorithm to select appropriate initial connection weights and thresholds for ANN. Kocadag˘lı and Aşıkgil (2014) have used a Bayesian inference approach to train an ANN. Kourentzes, Barrow, and Crone, (2014) have suggested that a hybrid ANNs ensemble may improve robustness and accuracy of prediction at the cost of increased complexity. Nevertheless, ANN is still considered as a 'black-box' and does not provide intuitive description of the prediction process (Lai, Fan, Huang, & Chang, 2009).

Other ML techniques that have been successfully applied for TS prediction include the adaptive neural fuzzy inference system (ANFIS) and the support vector machines (SVMs). ANFIS allows a set of IF-THEN rules and membership functions of fuzzy sets to be constructed based on the historical data (Jyh-Shing,1993 and Jang et al., 1997). This inference system integrates the best features of ANNs and fuzzy logic to handle the non-linearity and uncertainty in real-world processes (Piero, 2000 and Lee & Ouyang, 2003). SVMs have found many applications in classification, pattern recognition and regression analysis (Suykens & Vandewalle, 1999). Over the years, multiple variations of this method have been proposed. Partial least squares SVM method combines the partial least squares based feature selection with support vector machine for information fusion (Yang et al., 2011). This method was proposed to identify complex nonlinearity and correlations among financial indicators. Ensemble learning proposed by Kang et al. (2010) improves the performance of SVM-based classification and prediction algorithms. Fuzzy sets adaptation (Chaudhuri & Kajal, 2011) is capable of handling uncertainty and imprecision in prediction of corporate data. It is effective in finding a subset of optimal features and parameters. Other examples of SVM applications to financial predictions can be found in (Lin et al., 2011) and (Chen et al., 2010). Least squares SVM (LS-SVM) uses linear instead of quadratic programming, thus it reduces computational complexity of the original SVM algorithm (Gestel et al., 2003). LS-SVM involves mapping the data to a space of features, in which a function is constructed that can be used for TS prediction (Huang & Shyu, 2010).

The prediction models based on ML can be categorized into two classes: LMs, and global models. A global model is trained only once and then the same model is used for making many predictions (at different time instances). A LM is trained independently for each prediction case (Martinez et al., 2011). The LMs are usually trained by using a relatively small number of historical data subseries (nearest neighbours) that are similar to an input sequence (query) for which the prediction has to be made. The main issues of local modelling are efficient model building (Kaneko et al., 2010) and selection of lags that provides useful information. In Hastie et al. (2008) a lag selection method was proposed which uses t-statistics of estimated coefficients. Several distance measures are commonly used: Euclidean distance, hash function transformation (Chang et al., 2012) or fuzzy measures (Smith & Oswald, 2003). Mutual information criteria were used for informative lags selection (Božić et al., 2013 and Wu & Lee, 2015).

2.2 Time series pre-processing

In the related literature, several pre-processing methods have been proposed in order to improve the accuracy of TS prediction. Among these methods, a popular approach is to utilize various representations of TS and segmentation methods.

Spectral representation is based on frequency density function of TS. The Fourier or wavelet transform is commonly used for transforming TS to the spectral representation (Shumway & Stoffer, 2010). This representation can be used directly for creating a prediction model or to find the TS period, however the transformation can cause a loss of precision, due to a bias.

In case of state representation of TS, a state vector is used. The state vector is defined as a set of first order differential equations (Franklin, Powell, & Emami-Naeini, 2002). This representation, combined with Klaman filtering, was applied to the linear dynamical system and linear Gaussian state space model (Barber, 2012). The main limitation of this solution is related to its lineal character. In (Thrun, Burgard, & Fox, 2005) a modification of the Kalman filter was proposed to address nonlinear TS.

Fuzzy time series (FTS) represent the data by means of fuzzy relations. The data are fuzzified to obtain the FTS representation. Subsequently, a prediction model is created by determining fuzzy relations between data using learning methods (Yu & Huarng, 2010). The determination of appropriate partitions for building fuzzy relations is one of the main challenges (Askari and Montazerin, 2015 and Yu and Huarng, 2010). In (Askari and Montazerin, 2015) it was proposed to use fuzzy clusters for selecting the partitions. Complex solutions for FTS with multiple variables are addressed in (J. Dabrowski and J. Villiers, 2015).

Granular TS representation (Al-Hmouz, Pedrycz, & Balamash, 2015) is often used with fuzzy time series (Lu et al., 2014 and Wang et al., 2014). In case of the above mentioned ML methods for time series prediction, the application of granular representation involves four steps (Wu and Lee (2015). In the first step, the local context of the user query is found by using the k-nearest-neighbours method or fuzzy c-means method. Secondly, the appropriate number of lags is selected by applying mutual information criteria to measure the relevance of data. Thirdly, a set of training patterns is extracted from the data. Finally, the training patterns are fed to a ML algorithm. The drawback of such approach is that the nearest-neighbours method tends to fail, while tackled with noisy data.

Another pre-processing method considers time series segmentation, which allows us to categorize big TS data according to a defined clustering pattern. An extended up to date review of such methods can be found in (Zolhavarieh et. al., 2014). There are many clustering methods: hierarchical, k- and c-means, and based on pattern discovery. Hierarchical clustering builds a nested hierarchy of related time periods. The method enables analysis of TS on various hierarchical levels, at the cost of quadratic computational complexity (Lin et. al., 2002).

Partitioning clustering is a partitioning method, where each partition is represented by at least one object. The partition is crisp if each object belongs to exactly one cluster, or fuzzy otherwise. The crisp representations are build based on k-means method, where various distances can be used. In case of fuzzy solution, the implementations of *c*-means algorithm are used. Additional approach to TS segmentation is based on application of fuzzy *c*-medoids algorithm (Izakian et. al., 2015). These heuristic algorithms define a cluster as some shape e.g. spherical-shaped cluster. Recently more focus was given on density-based clustering methods. The idea of density-based methods is to extend a cluster as long as the density (number of objects or data points) in range exceeds some threshold. Several density-based methods were developed that additionally define order of clusters, e.g., DBSCAN or OPTICS (Denton A., 2004). Finally, the newest group of the methods covers clusters pattern discovery (or motif discovery). The patterns are usually searched based on frequency or shape. An extensive analysis of these methods was presented by Marschall, T. and Rahmann S. (2009).

A main disadvantage of the above described pre-processing methods is that they are based on spectral, fuzzy and state representations. Such representations cause the possibility that important information is lost during the TS transformation. A second drawback is that the segmentation methods can find false (not existing) patterns in noised TS, which may cause incorrect predictions.

2.3 Original contribution

In this paper a novel method is introduced, which enables TS pre-processing and data selection for training LMs. Instead of using the spectral methods, the period of TS is detected based on the Box and Jenkins (2008) analysis. The proposed approach enables detection of the strongest periodical pattern in TS. The detected period is used by the new pre-processing algorithm, which allows us to determine usefulness relation (*UR*). The *UR* enables selection of the historical data that are useful for making predictions at particular steps of the TS period. In contrast to other segmentation methods, the *UR* takes under consideration not only similarities of data sequences (Huang et al. , 2011) but also the expected prediction error. Moreover, each data point in TS is processed independently, so no generalisation is performed at this step. According to the proposed method, useful segments of training data are selected by means of a new usefulness-related hybrid distance. The data selection is performed based on the k-NN approach combined with the novel usefulness-related distance, which takes into consideration the periodicity of TS. In this paper, the proposed approach is compared against complex k-NN implementations (Wu and Lee (2015) that have been suggested recently for LMs creation.

The introduced *UR* can be computed in parallel to the execution of the prediction procedure, thus the prediction method can be applied for real-time systems. The proposed pre-processing and data selection method enables effective applications of various ML techniques for periodical TS prediction in expert and intelligent systems. Effectiveness of this approach was experimentally tested for three popular ML methods: ANN (Zhang et al., 1998) (Adhikari et al., 2011), ANFIS (Jyh-Shing et al., 1993), and SVM (Lin et al., 2011).

It is worth to note, that the proposed TS pre-processing algorithm does not change the data representation. Thus, in contrast to the methods based on spectral transformation, fuzzification or granulation, the introduced approach does not involve any loss of the information, which is contained in historical TS. This allows all useful data to be utilised for LM training.

3. Proposed method

This section provides a detailed description of the proposed method. The novelty of the approach lies mainly in the pre-processing stage. Therefore, this part is thoughtfully described with examples. Main steps of the proposed method are presented in Fig. 3.1. In the first step, regularities in TS are found by using the approach proposed by Box & Jenkins (2008), which is based on autocorrelation function and moving average filter. Then, the usefulness relation (*UR*) is extracted. The *UR* extraction algorithm is based on similarity and prediction error rankings. The introduction of *UR* was motivated by an observation that useful training data are usually found in a relatively narrow time span. *UR* extraction aims at selection of the data that are useful for making prediction at a given step of TS period.

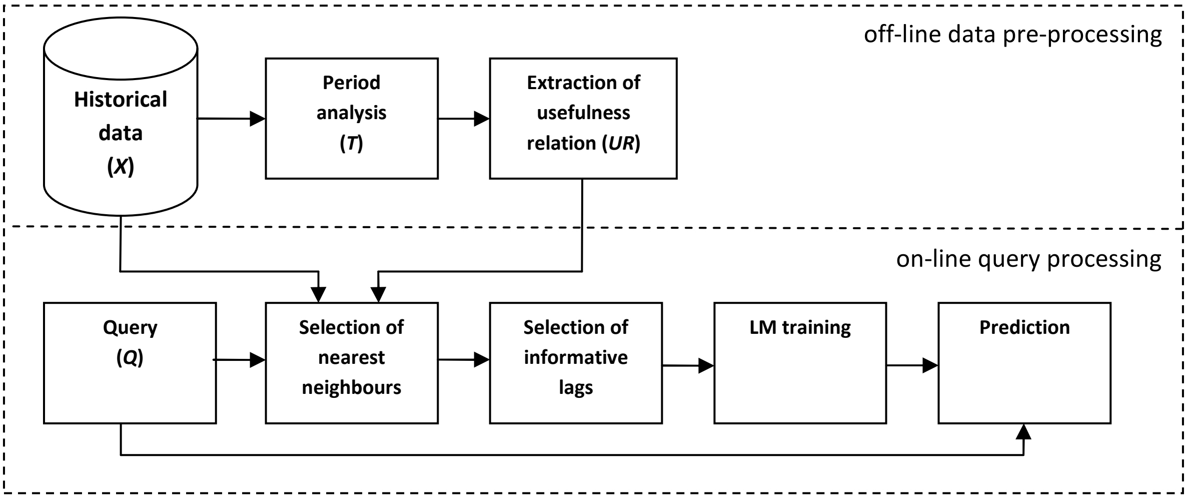

off-line data pre-processing
Historical data (*X*)
Period analysis (*T*)
Extraction of usefulness relation (*UR*)
on-line query processing
Query (*Q*)
Selection of nearest neighbours
Selection of informative lags
LM training
Prediction

Fig. 3.1. The proposed model overview.

The TSs, analyzed in the paper, have a discrete form. TS is defined as a series of observations X=[x_0, x_1, ..., x_t,..., x_n]. Time step between any two adjacent observations is constant. The series X can also be represented as a set of subseries $S_{j,w} = [x_j, x_{j-1},..., x_{j-w}]$, where j defines a j-th time point and w is a number of the past observations (lags):

$$S_{n,w} = [x_n, x_{n-1},..., x_{n-w}]$$

...

$S_{j,w} = [x_j, x_{j-1},..., x_{j-w}]$

...

$S_{w,w} = [x_w, x_{w-1},..., x_0]$

For a test query $Q_{z,w} = [x_z, x_{z-1},..., x_{z-w}]$ the prediction is denoted as $\hat{x}_{z+s}$, where *s* is the considered number of steps ahead into the future, and *z* is the time point of prediction ($z > n$). In order to process the test query, a training data set (*LS*) has to be selected from *X*. The *LS* is used to build a local model *G* by using the ML methods.

3.1 Period analysis

At the first step of the proposed method, a time interval *T* (period) is searched for which some regularities in TS can be distinguished. The period analysis is based on the Box & Jenkins approach (Box & Jenkins, 2008), thus it is assumed that no missing values are present in TS.

It is should be noted here that the spectral based method (Shumway & Stoffer, 2010) was also considered to find the longest period. However, for the analysed real-world TS it was hard to distinguish the period without extended user-assisted calibration process.

The period of TS is found using autocorrelation function (*ACF*). Let x_t denote the value of *X* at time *t*. The values of *ACF* function for series *X* describe correlations between x_t and x_{t-h} , where *h* defines a lag (*h* = 1, 2, ..., T_{max}). The maximum search period T_{max} is not longer than *card*(*X*)/4, where *card* denotes cardinality of the TS. The analysis of longer periods can give unreliable results (Venables & Ripley, 2002). In practice, the value of *ACF* is calculated as follows:

$$ACF(X,h) = \frac{AVF(X,h)}{\max\limits_{i \in 1,2,...,T_{\max}} (AVF(X,i))}, \tag{3.1}$$

$$AVF(X,h) = \sum_{i=\max(1,-h)}^{\min(n-h,n)} [x_{i+h} - \bar{x}][x_i - \bar{x}],$$

where: $\bar{x}$ is mean value of *X* , *n* is equal to card(X), and x_i is an element of the TS ($x_i \in X$).

To ensure that the obtained result is reliable, the TS has to be at least weakly stationary. This means that the values of mean and variance are constant and the auto covariance between x_t and x_{t+h} depends only on the lag *h* (*h* is a finite integer value), therefore the stationary component of TS is extracted to find the period. To this end, first differences $x'_t = x_t - x_{t-1}$, for t=1,..., *n* are used instead of the original time series *X*. This is a common method for obtaining a de-trended and weakly stationary TS. In the proposed method, the analysis has to be performed for a number of data points within a single period, thus the periods longer than T_{min} are searched. Results of preliminary experiments show that periods shorter than T_{min} = 8 are insufficient to find useful training data in real-world TS. Examples of the *ACF* analysis for a real-world road traffic TS an a synthetic TS are presented in Fig. 3.2.

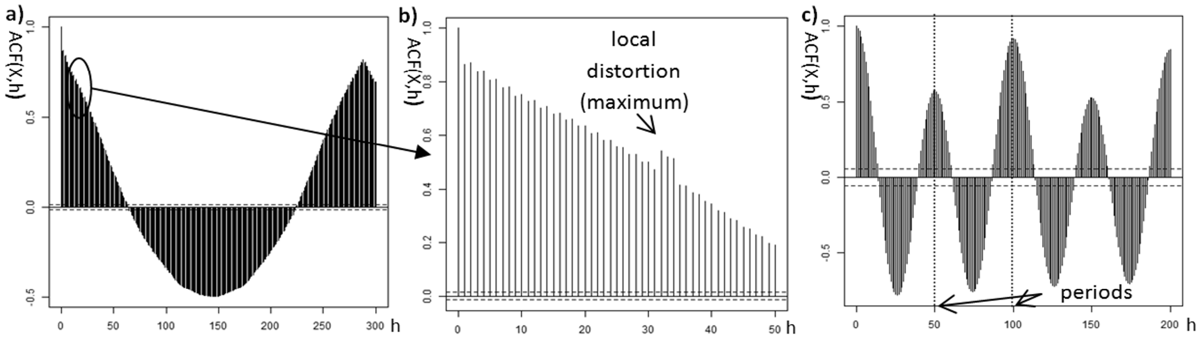


Fig. 3.2. ACF analysis for: a, b) traffic volume TS, c) synthetic TS.

Figures 3.2 a and 3.2 b illustrate the *ACF* values obtained for road traffic data, where the day period contains 288 measurements collected in time intervals of 5 minutes. The presence of distortions and noise results in local maxima of the *ACF* values (Fig. 3.2 b). Such maxima can be improperly detected as periods. To reduce the influence of noise, the moving average is used. If a TS with multiple periods is considered (Fig. 3.2 c) then the longest period has to be searched. The above assumptions are taken into account by the period finding algorithm (Algorithm 1). According to this algorithm, the correlation value for the recognized period *T* has to be above a given threshold ($ACF_{threshold}$).

Algorithm 1: Period finding

```
{pre-processing}
set $ACF_{threshold}$ parameter to $0.2$
Create X' as first difference of X
Perform moving average filtering on $X'$ with window size $T_{min}$
{ACF calculation}
initialize $TA$ array by zeros
for $i$ := $T_{min}$ to $T_{max}$ do
    $TA$[i]:=ACF($X'$,$i$);
{period determination}
$T$:=null; $max$:= $ACF_{threshold}$ ;
for $i$ := $T_{min}$ to $T_{max}$ do
```

```
    if TA($i$)>max then
        if $TA[i-1]<TA[i]$ and $TA[i+1]<TA[i]$ then
            T:=$i$ and max:= ACF($X'$,$i$)
```

Based on the results presented in (Box & Jenkins, 2008) and (Venables & Ripley, 2002), the $ACF_{threshold}$ was set to 0.2. While analyzing TS it is possible that no period can be found. Thus, if there is no local maximum with value above $ACF_{threshold}$, the *UR* cannot be constructed. The information about period *T* is necessary for extraction of *UR* in the next step of the proposed method.

3.2. Extraction of usefulness relation

In the proposed method a binary *UR* is introduced that determines, which historical data from the time series are useful for making a prediction at a given time step of the period. If the data registered at time step *v* of the period are useful for making the prediction at time step *u* then the pair (*u*, *v*) is an element of the usefulness relation.

The usefulness relation is extracted from a learning time series by using Algorithm 2. The inputs of this algorithm include: the learning time series $X = [x_0, ..., x_m, x_{m+1}, ..., x_n]$, the length of the period *T*, value $p(0)$ which identifies time step of the period for the first time point in time series *X*, and parameter *w* which determines number of lags in subseries. An example of the time series is presented in Fig. 3.3. Note that for this example $T = 15$ and $p(0) = 10$.

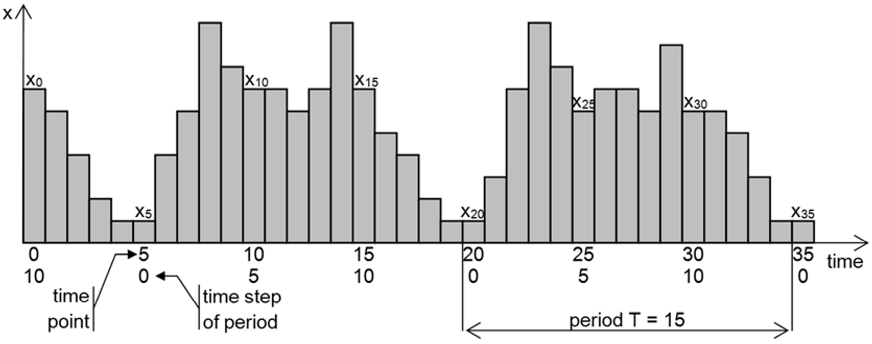


Fig. 3.3. Example of time series.

The learning time series *X* is divided into two parts at *m*-th data point. For the experiments reported in this paper it was assumed that $m = \lfloor n/2 \rfloor$. During extraction procedure, the data from the right part of the time series $[x_{m+1}, ..., x_n]$ are interpreted as current measurements. The left part of the time series $[x_0, ..., x_m]$ is considered as a set of historical measurements that can be used for making a prediction. It means that Algorithm 1 evaluates usefulness of the data from $[x_0, ..., x_m]$ for making the prediction at time points $m + 1, ..., n$.

The usefulness evaluation is based on an insight that the historical data are useful if they are similar to current measurements and contribute to accurate prediction of future values. Therefore, the data usefulness is evaluated using two criteria: similarity with current conditions and error of the prediction made on the basis of the selected data. According to these criteria, two rankings of the data points are created: similarity ranking and prediction error ranking. Usefulness of a data point is calculated based on its positions in both rankings.

3.2.1. Similarity ranking

In order to create the similarity ranking, a measure of distance between current and historical data is analysed. More specifically, distances are calculated between pairs of subseries $S_{i,w} = [x_i, x_{i-1},..., x_{i-w}]$ and $S_{j,w} = [x_j, x_{j-1},..., x_{j-w}]$ that include the historical and the recent measurements respectively. The parameter w determines the number of lags. According to the above discussed assumptions, time points $i = w, w + 1, ..., m$ are considered as the historical ones, and time points $j = m + 1, m + 2, ..., n$ correspond to the current situation. In this study, the distance between $S_{i,w}$ and $S_{j,w}$ (denoted in Algorithm 1 by $distance(i, j)$) is evaluated based on the Euclidean norm. Such approach to measuring the similarity is commonly used in the related literature (Gooijer, 2006).

For each time point j, a list L is created which includes the time points $i = w , w + 1, …, m$. This list is then sorted in ascending order according to $distance(i, j)$. In this way the similarity ranking is constructed. Let $index(i)$ denote the position of i in list L, i.e., $index(i) = 0, 1, ..., size(L) – 1$, where $size(L)$ is the number of elements in the list L. The positions of the time points i in the ranking are used to determine their scores. A time point i gets score of 1 if it is on the first place in the ranking ($index(i) = 0$). In opposite situation, if time point i is on the last place ($index(i) = size(L) – 1$), it gets score of 0. Thus, the score for time point i is calculated using the formula:

$$score(i) = 1 - \frac{index(i)}{size(L) - 1}. \tag{3.2}$$

The scores determined on the basis of the similarity ranking are accumulated in array SIM for particular time steps of the period. This operation (line 7 of Algorithm 2) requires finding the time step of period $v \in \{0, 1, ..., T – 1\}$, which corresponds to time point i. To this end, the following formula is used:

$$p(i) = i + p(0) - T \cdot \left\lfloor \frac{i + p(0)}{T} \right\rfloor, \tag{3.3}$$

where $p(0)$ is the time step of period, which corresponds to the first time point in time series X. It is assumed that $p(0)$ is known. Note that according to Eq. (3.3) $p(i)$ is the remainder of division of $i + p(0)$ by T.

Finally, the scores in array *SIM* are accumulated for time steps $v = 0, 1, \ldots, T - 1$ and normalized into interval [0, 1] (lines 8 - 10 in Algorithm 2). Thus, if the normalized score $SIM[j, v]$ is close to 1 then it means that the measurements made at v-th time step of the period are usually similar to the data registered at time point j.

Algorithm 2: Extraction of usefulness relation

```
1   {smilarity ranking}
2   initialize $SIM$ array by zeros
3   for $j := m + 1$ to $n$ do
4       create list $L$ of time points ($i = w, w + 1, ..., m$)
5       sort $L$ according to $distance(i, j)$ in ascending order
6       for $i := w + 1$ to $m$ do
7           $SIM[j, v(i)] := SIM[j, v(i)] + 1 - \mathrm{index}(i) / (\mathrm{size}(L) - 1)$
8       $sim\_max := \max(SIM[j, 0], ..., SIM[j, T - 1])$
9       for $v := 0$ to $T - 1$ do
10          $SIM[j, v] := SIM[j, v] / sim\_max$
11  {prediction error ranking}
12  initialize $PRE$ array by zeros
13  for $j := m + 1$ to $n - 1$ do
14      create list $L$ of time points ($i = 0, 1, ..., m$)
15      sort $L$ according to $error(i, j)$ in ascending order
16      for $i := w + 1$ to $m$ do
17          $PRE[j, p(i)] := PRE[j, p(i)] + 1 - \mathrm{index}(i) / (\mathrm{size}(L) - 1)$
18      $pre\_max := \max(PRE[j, 0], ..., PRE[j, T - 1])$
19      for $v := 0$ to $T - 1$ do
20          $PRE[j, v] := PRE[j, v] / pre\_max$
21  {usefulness relation}
22  for $j := m + 1$ to $n - 1$ do
23      for $v := 0$ to $T - 1$ do
24          $PRE[j, v] := PRE[j, v] \cdot SIM[j, v]$
25  initialize $UR$ array by zeros
26  for $j := m + 1$ to $n - 1$ do
27      for $v := 0$ to $T - 1$ do
28          $UR[p(j), v] := UR[p(j), v] + PRE[j, v]$
29  for $v := 0$ to $T - 1$ do
```

```
30      $ur\_max$ := max($UR$[0, $v$], ..., $UR$[$T$ – 1, $v$])
31      for $u$ := 0 to $T$ – 1 do
32         $UR$[$u$, $v$] := $UR$[$u$, $v$] / $ur\_max$
33   calculate threshold $\alpha$
34   for $v$ := 0 to $T$ – 1 do
35      for $u$ := 0 to $T$ – 1 do
36         if $UR$[$u$, $v$] $\geq \alpha$ then $UR$[$u$, $v$] = 1 else $UR$[$u$, $v$] = 0
```

3.2.2. Prediction error ranking

Error of the prediction, which can be made based on selected data, is the criterion of the second ranking. In this ranking, the time points are considered in pairs: *i* (as a historical base for making the prediction) and *j* (as current time). In order to evaluate the impact on the prediction accuracy, it is assumed that the prediction is based only on a single observation, which was made at time step *i*. For such assumptions the predicted value $\hat{x}_{j+1}$ can be determined as $\hat{x}_{j+1} = x_{i+1}$. The correct result of the prediction made at time point *j* is x_{j+1}. Thus, absolute error of the prediction, which is based on the data registered for time step *i*, can be evaluated by using the following formula:

$$error(i, j) = \left| \; x_{i+1} - x_{j+1} \; \right| \; . \tag{3.4}$$

For a given time step *j*, the error ranking is created by sorting the list of time points (i = 0, 1, …, m) in ascending order, according to the value of *error*(i, j). The subsequent operations are analogous to those discussed for the similarity ranking. They include calculations of scores for time points, accumulated scores for time steps of the period, and the normalized scores. The scores for time points i = 0, 1, …, m are calculated according to Eq. (3.4). Array *PRE* is used to accumulate the scores for particular time steps of the period v = 0, 1, …, T – 1 (line 17 in Algorithm 2). The accumulated scores are then normalized into interval [0, 1] (lines 18 - 20 in Algorithm 2).

The normalized scores obtained from the prediction error ranking should be interpreted in the following way: *PRE*[*j*, *v*] close to 1 means that a low prediction error is usually achieved for time point *j* when using the data registered at *v*-th time step of the period as the prediction input. A high error can be expected when the prediction at time point *j* is based on the observations made at *v*-th time step of period, for which *PRE*[*j*, *v*] is close to 0.

3.3. Usefulness of data

According to the proposed approach, data are considered as being useful for making prediction if they are similar with current measurements and ensure low prediction error. Therefore, it is legitimate to use the fuzzy logic *AND* operator for combining the scores, calculated on the basis of the similarity ranking and the prediction error ranking, into single usefulness measure. It should be noted that the normalized scores defined in previous sections enable identification of two fuzzy sets. The score $SIM[j, v]$ is interpreted as a membership grade of time step v in a fuzzy set containing the time steps at which the measurements are usually similar to those registered at time point j. Analogously, scores $PRE[j, v]$ are interpreted as membership values for a fuzzy set of the time steps that usually provide low prediction error for time point j.

In order to determine the fuzzy set of measurements that satisfy both criteria (similarity with current measurements and low prediction error), the scores $SIM[j, v]$ and $PRE[j, v]$ are combined using algebraic product as the fuzzy *AND* operator (line 24 in Algorithm 2). These calculations are repeated for all time points $j = m + 1, \ldots, n - 1$. The results obtained for particular time points j are then accumulated in array *UR* to determine usefulness of the data registered at v-th time step of the period for making prediction at time step $v(j)$ (line 28 in Algorithm 2). At the next step, the elements of array *AU* are normalized into interval [0, 1] (lines 30-32 in Algorithm 2). This normalization is performed independently for each value of u ($u = 0, ..., T - 1$). The obtained value of array element $UR[u, v]$ should be interpreted as a degree to which the historical data collected at v-th time step of the period are useful for making prediction at time step u.

Finally, a threshold α is used to obtain binary values of the elements in array *UR* (lines 34-36 in Algorithm 2). The threshold α is selected by using the method based on Renyi's entropy (Maszczyk & Duch, 2008), which was originally designed for thresholding of gray-level images. The main concept behind this method is to use the entropy as a measure of the amount of information contained in the resulting binary array. Therefore, a threshold value is selected which maximizes the entropy. During preliminary experiments, the above method has provided better results than other threshold selection methods that are available in the literature (Sezgin & Sankur, 2004).

Output of the introduced algorithm is the binary array *UR*, which represents the usefulness relation. Value 1 of an element $UR[u, v]$ means that the ordered pair (u, v) is an element of the usefulness relation. Zero elements in array *UR* indicate those pairs (u, v) that do not belong to this relation.

3.4. Selection of nearest neighbours

The *k* nearest neighbours' method aims to find subseries in a historical data set that are similar to the analyzed ones and thus describe the possible future values. For a given query $Q_{z,w}$ at time point *z*, the *k* nearest neighbours are defined as the subseries $\{ S_{t_i,w} : i = 1, 2, ..., k\}$ with lowest values of a distance measure. Several distance measures are commonly used: Euclidean distance, hash function transformation (Chang et al., 2012) or fuzzy measures (Smith & Oswald, 2003). In this paper, the measure based on the *UR* is proposed.

The *UR* describes which times steps *v* of the period T are useful for making prediction at time step *u*. Fig. 3.4 shows an example of *UR* obtained for synthetic two-period sinusoid TS.

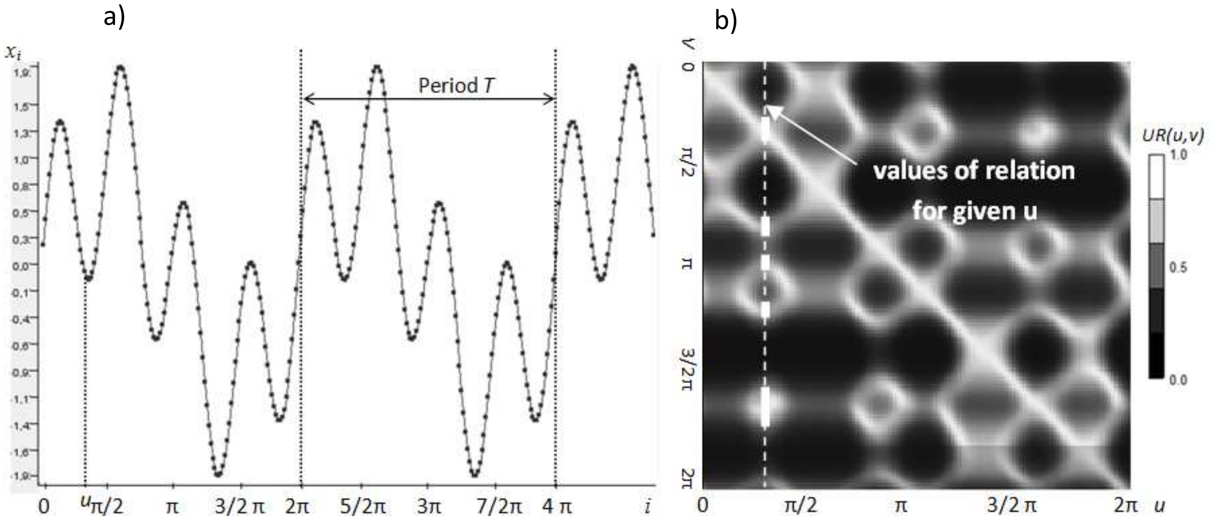


Fig. 3.4. The *UR* for synthetic TS: a) TS plot, b)*UR* relation before binarization.

The *UR* is used for selecting the time points that are expected to provide an accurate prediction. Therefore, the following hybrid usefulness-related distance is proposed:

$$R(S_{j,w}, Q_{z,w}) = \frac{E_U(S_{j,w}, Q_{k,w}) + E_U(S'_{j,w}, Q'_{z,w})}{2} - UR(p(j), p(z)), \qquad (3.5)$$

where:

UR - usefulness relation,

p - time step function defined in Eq. (3.3),

$S_{j,w}$ - the historical subseries,

$Q_{z,w}$ - query, for which the prediction is made,

$S_{j,w}'$, - first order difference of the subseries,

$Q_{i,w}'$ - first order difference of the query,

E_U - Euclidean distance normalized to interval [0,1].

The *UR* returns value 0 or 1, thus the hybrid distance *R* takes values within interval [-1, 1]. Based on distance *R*, the *k* nearest neighbours (subseries) are selected from $\{S_{j,w}: j = w, w + 1, ..., n\}$. Set $TE=\{t_i: i = 1, 2, ..., k\}$ includes the time points in historical database, where the *k* nearest neighbours, i.e., the subseries $S_{t_i,w}$ with lowest values of distance *R*, are found.

3.6. Selection of informative lags

A selected subseries $S_{t_i,w}$ includes data registered at time point t_i and at *w* previous time points (lags). All these values could be used for training a prediction model (Khashei & Bijari, 2011). However, this approach can lead to many issues. With growing data size it is not possible to estimate the maximum processing time and in consequence the model cannot be implemented in real time systems (Wu & Lee, 2015). Furthermore, the usage of all lags may lead to over fitting (Kraskov et al., 2004). On the other hand, if the number of lags is not large enough, the prediction accuracy will be decreased. Therefore, only the relevant, informative lags have to be considered for further processing.

In the proposed method, the mutual information function (Kraskov et al., 2004) is used to determine the influence of the set of lags on the predicted value. The mutual information function was successfully applied for LMs in (Wu & Lee, 2015). For the sake of clarity, the selected subseries are represented by the matrix shown in Fig. 3.5.

Lag *Neighbour*	0 (B_0)	1 (B_1)	2 (B_2)	…	w (B_w)	-s (D)
$S_{t_1,w}$ (l_1)	x_{t_1} ($b_{0,1}$)	x_{t_1-1} ($b_{1,1}$)	x_{t_1-2}		x_{t_1-w}	x_{t_1+s} (d_1)
$S_{t_2,w}$ (l_2)	x_{t_2} ($b_{0,2}$)	x_{t_2-1} ($b_{1,2}$)	x_{t_2-2}		x_{t_2-w}	x_{t_2+s} (d_2)
$S_{t_3,w}$ (l_3)	x_{t_3} ($b_{0,3}$)	x_{t_3-1} ($b_{1,3}$)	x_{t_3-2}		x_{t_3-w}	x_{t_3+s} (d_3)
…			↖ **B submatrix**			
$S_{t_k,w}$ (l_z)	x_{t_k} ($b_{0,z}$)	x_{t_k-1} ($b_{1,z}$)	x_{t_k-2}		x_{t_k-w}	x_{t_k+s} (d_z)

Fig. 3.5. Matrix representation of the selected subseries.

The analysed set of lags is represented by submatrix *B*. The last column of the matrix (vector *D*) contains values x_{t_i+s}, *i* = 1,..., *k* that correspond to the predictions made for *s* steps forward.

The rows of the matrix are called instances (I_i , i = 1, ..., k). Information function is used to analyse the dependency between subset of lags ($B \subset \{B_0, B_1, ..., B_w\}$) and vector D. The information function is defined by Eq. 3.6.

$$I(B,D) \approx \psi(g) - \frac{1}{g} - \frac{1}{k}\sum_{i=1}^{z}\left[\psi(n_B(i)) + \psi(n_D(i))\right] + \psi(k)\,, \tag{3.6}$$

$$\psi(x) \approx \psi(x-1) + \frac{1}{x},\ \ \psi(1) = -c\,,$$

where: g is a parameter of the method, k denotes the number of neighbours, n_B, n_D are support functions, and c = 0.5772156 is the Euler-Mascheroni constant.

Eq. (3.6) was proposed by Kraskov et al. (2004) and provides a rough estimation of the mutual information function of two random variables proposed by Guiasu & Silviu (1977), which is based on joint probability density function.

The support functions $n_B(i)$ and $n_D(i)$ determine the number of instances enclosed respectively in a B- and D-hyper-rectangle. An example of interval representation of the hyper-rectangles is presented in Fig. 3.6.

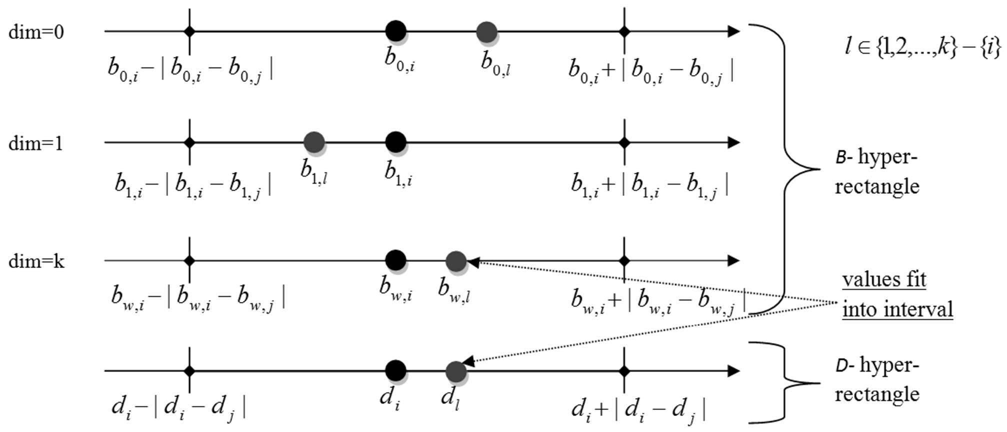

dim=0
$b_{0,i} - |b_{0,i} - b_{0,j}|$
$b_{0,i}$
$b_{0,l}$
$b_{0,i} + |b_{0,i} - b_{0,j}|$
$l \in \{1,2,...,k\} - \{i\}$
dim=1
$b_{1,i} - |b_{1,i} - b_{1,j}|$
$b_{1,l}$
$b_{1,i}$
$b_{1,i} + |b_{1,i} - b_{1,j}|$
B- hyper-rectangle
dim=k
$b_{w,i} - |b_{w,i} - b_{w,j}|$
$b_{w,i}$
$b_{w,l}$
$b_{w,i} + |b_{w,i} - b_{w,j}|$
values fit into interval
$d_i - |d_i - d_j|$
d_i
d_l
$d_i + |d_i - d_j|$
D- hyper-rectangle

Fig. 3.6. Construction of B- and D-hyper-rectangle.

In order to construct the hyper-rectangles, a ranking based on the maximum norm E_N is used:

$$E_N(I_i, I_l) = \max\{|b_{0,i} - b_{0,l}|, |b_{1,i} - b_{1,l}|, ..., |b_{w,i} - b_{w,l}|, |d_i - d_l|\} \tag{3.7}$$

The ranking is created by sorting instances I_l , $l \in \{1,2,...,w\} - \{i\}$ in ascending order, according to $E_N(I_i, I_l)$. The aim is to find j-th instance (I_j), which is on g-th position in the ranking. The hyper-rectangles are determined by i-th and j-th instance.

Subsequently, it is verified if the instances I_l, $l \in \{1,2,...,w\} - \{i\}$ belong to the B- and D-hyper-rectangle. The instance I_l belongs to the hyper-rectangle, only if all its values fit into separate intervals (Fig. 3.6). The functions $n_B(i)$ and $n_D(i)$ determine the number of instances $I_l, l \in \{1,2,...,w\} - \{i\}$ that belong to the created hyper-rectangle.

The lag selection process is controlled by parameter *g*. In (Stogbauer et al., 2004) it was suggested that g = 6 is a good choice. The preliminary experiments have confirmed that this value ensures a high performance. The final value of information function is calculated based on Eq. (3.6). This function is utilized in the proposed algorithm for finding the useful lags (Algorithm 3), where *w* is a number of input lags. The number of informative lags (*y*) is determined during calibration procedure.

Algorithm 3: Useful lag selection

```
{initialize data}
initialize  empty $L_\lambda$ array of length $w$ by zeros
create empty B set
create vector with values of prediction D=$\lfloor x_{t_1+s}, x_{t_2+s},...,x_{t_z+s} \rfloor$
{main loop}
for lags:= 1 to $y$ do
    initialize  empty $L_{\lambda val}$ array of length $w$ by zeros
    for i:=0 to w do
        {calculation of information criteria}
        if $L_\lambda$[i]=0 then // candidate lag for selection
            BT = B $\cup$ $B_i$
            $L_{\lambda val}$[i]=I(BT,D) according to Eq. (3.6)
    find index $i_{sel}$ of maximal element of $L_{\lambda val}$ array
    add selected lag to table $L_\lambda$:  $L_\lambda[i_{sel}]=1$
    add selected lag to set B:  $B = B \cup B_{i_{sel}}$
{create a list of lags}
create empty set LL
for i:=1 to w do
    if  $L_\lambda[i]$=1 then LL=LL $\cup$ $i$
```

The result of Algorithm 3 is a list of *y* informative lags ($LL=\{\lambda_i$, i=1,..., $y\}$). The proposed algorithm uses a greedy approach to find the optimal lags. Sorjamaa et al. (2007) has proved that the greedy approaches, based on forward selection and backward elimination, perform equally well. In this study, the forward selection was used because during initial experiments it was observed that this approach is faster ($y << w$).

Finally, the number of lags is reduced to y. The selected lags maximize the value of mutual information function. The selected data are used as a training set (LS) for the local prediction model. The LS is a set of pairs that include the subseries with selected lags $S_{t_i,y}$ and the prediction values x_{t_i+s} for s steps ahead:

$$\text{LS}=\{(S_{t_i,y}, x_{t_i+s}): \text{i} = 1..\text{k}\}, \tag{3.8}$$

where $S_{t_i,y} = [x_{t_i-\lambda_1}, x_{t_i-\lambda_2,...,} x_{t_i-\lambda_y}]$.

In a similar way, the reduction of query $Q_{k,w}$ is performed. As a result, query $Q_{z,y}$ is obtained, which includes only the selected lags.

4. Example of data selection for LM training

This section includes a detailed example of the proposed method application for synthetic TS (noised sinusoid of unknown period). The synthetic TS considered in this example (X) and the prediction query (Q) are defined as follows:

$$X = [-0.07, 0.97, 0.59, -0.63, -0.93, 0.03, 0.93, 0.55, -0.59, -0.99, -0.03, 1.00, 0.60, -0.56, -1.03, 0.02, 0.88, 0.54, -0.64, -0.86] \tag{4.1}$$

$$Q_{24,4} = [-0.01, 0.90, 0.67, -0.67, -0.91]$$

Algorithm 1 was used to find period T of the analysed TS. The ACF values calculated for X are illustrated in Fig. 4.1. The results of ACF analysis show that the period T of TS equals 5.

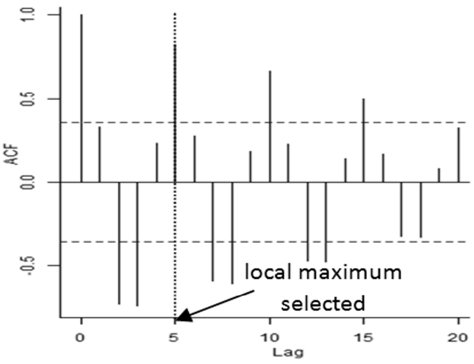


Fig. 4.1. Period detection based on Algorithm 1.

Let us assume that the prediction has to be made for a single step ahead (s = 1) and length of the analysed subseries is w = 4. Then, X is represented as a set of subseries $S_{i,4}$ i = 4, ..., 18 (Tab. 4.1).

Table 4.1. The calculations example for ranking construction.

i	p(i)	$S_{i,w}$					x_{i+s}	$E(S_{j,4}, S_{i,4})$		error(*i*,*j*) $\lvert x_{j+s,4}, x_{i+s,4}\rvert$	
		x_i	x_{i+1}	x_{i+2}	x_{i+3}	x_{i+4}		j=17	j=18	j=17	j=18
4	4	-0.93	-0.64	0.59	0.98	-0.08	0.04	3.02	1.84	0.68	0.91
5	0	0.04	-0.93	-0.64	0.59	0.98	0.93	3.01	3.05	1.58	1.80
6	1	0.93	0.04	-0.93	-0.64	0.59	0.56	1.82	3.02	1.20	1.42
7	2	0.56	0.93	0.04	-0.93	-0.64	-0.60	**0.14**	1.85	**0.05**	0.27
8	3	-0.60	0.56	0.93	0.04	-0.93	-1.00	1.87	**0.13**	0.35	**0.13**
9	4	-1.00	-0.60	0.56	0.93	0.04	-0.04	3.02	1.87	0.61	0.83
10	0	-0.04	-1.00	-0.60	0.56	0.93	1.01	3.01	3.02	1.65	1.87
11	1	1.01	-0.04	-1.00	-0.60	0.56	0.60	1.89	3.09	1.25	1.47
12	2	0.60	1.01	-0.04	-1.00	-0.60	-0.57	**0.16**	1.97	**0.08**	0.30
13	3	-0.57	0.60	1.01	-0.04	-1.00	-1.04	1.86	**0.17**	0.39	**0.17**
14	4	-1.04	-0.57	0.60	1.01	-0.04	0.03	3.06	1.85	0.68	0.90
15	0	0.03	-1.04	-0.57	0.60	1.01	0.88	3.08	3.09	1.53	1.75
16	1	0.88	0.03	-1.04	-0.57	0.60	0.54	1.89	3.06	1.19	1.41
17	2	0.54	0.88	0.03	-1.04	-0.57	-0.65	-	-	-	-
18	3	-0.65	0.54	0.88	0.03	-1.04	-0.87	-	-	-	-

The thick line in Tab. 4.1 separates current subseries from historical subseries (see Sect. 3.2). The Euclidean distances are calculated between pairs of historical subseries ($S_{i,w}$, i = 4,...,16) and the current ones ($S_{j,w}$, j = 17,...,18). Additionally, the prediction error (*error*(i, j)) is calculated as the absolute difference $|x_{i+s} - x_{j+s}|$ (see Alg. 2). All necessary calculations are illustrated in Tab. 4.1 for two subseries: $S_{17,w}$ and $S_{18,w}$. The calculation process for the scores *SIM* and *PRE* is presented in Tab. 4.2 for subseries $S_{17,w}$. The rankings are created by sorting the distances (Tab. 4.1) in ascending order (2-nd and 6-th row in Tab. 4.2). Scores *SIM* and *PRE* are calculated based on the positions in rankings, as discussed in Sect. 3.2 (4-th and 8-th row in Tab. 4.2). First ranking takes into account the similarity between the subseries. In this ranking, those subseries are favoured that are more similar (closer) to the selected subseries j = 17. In the considered example, the highest similarity scores (*SIM*) for j = 17 have the subseries $S_{7,4}$ and $S_{12,4}$. The second ranking for prediction error is constructed by using the error measure (6-th row in Tab. 4.2).

Table 4.2. Similarity and prediction ranking for $u = p(j = 17) = 2$.

No	**Similiarity ranking**													
1	$S_{i,4}$ i=	7	12	6	13	8	16	11	5	10	9	4	14	15
2	$E(S_{17,4}, S_{i,4})$	0.14	0.16	1.82	1.86	1.87	1.89	1.89	3.01	3.01	3.02	3.02	3.06	3.08
3	v=p(i)	**2**	**2**	1	3	3	1	1	0	0	4	4	4	**0**
4	SIM[2, *v*]	1.00	0.92	0.83	0.75	0.67	0.58	0.50	0.42	0.33	0.25	0.17	0.08	0.00
	Prediction error ranking													
5	$S_{i,5}$ i=	7	12	8	13	9	14	4	16	6	11	15	5	10
6	$\lvert x_{17+s,4} - x_{i+s,4}\rvert$	0.05	0.08	0.35	0.39	0.61	0.68	0.68	1.19	1.20	1.25	1.53	1.58	1.65
7	v=p(i)	**2**	**2**	3	3	4	4	4	1	1	1	0	**0**	**0**
8	PRE[2, *v*]	1.00	0.92	0.83	0.75	0.67	0.58	0.50	0.42	0.33	0.25	0.17	0.08	0.00

Based on the example in Tab 4.2, it should be concluded that for the prediction made at time step $u = p(17) = 2$, the most useful data are those registered at time step 2 ($v = 2$), and the data observed for time step $v = 0$ are the least useful.

Finally, the fuzzy AND operation is performed for values of two rankings. In order to determine the *UR* value, the result of AND operation is normalized to interval [0, 1]. In the analysed example, the values of *UR* are calculated for $u = 2$ (Tab. 4.3 a).

Table 4.3. Part of *UR* relation for $u = 2$ and $u = 3$.

a) u=2					
UR(2,v)	v=0	v=1	v=2	v=3	v=4
SIM(2,v)	0.25	0.64	0.96	0.71	0.17
PRE(2,v)	0.08	0.33	0.95	0.79	0.58
UR(2,v)	**0.02**	**0.23**	**1.00**	**0.62**	**0.11**
b) u=3					
UR(3,v)	v=0	v=1	v=2	v=3	v=4
SIM(3,v)	0.22	0.19	0.63	0.96	0.69
PRE(3,v)	0.08	0.33	0.79	0.96	0.58
UR(3,v)	**0.02**	**0.07**	**0.54**	**1.00**	**0.43**

Using the same analysis for subseries $S_{18,4}$, where $u = p(j = 18) = 3$, allows us to calculate the values for next row of *UR* (Table. 4.3. b). To calculate the *UR* values, at least one subseries for each $u = 0..T - 1$ have to be processed. The calculated values are presented in Fig. 4.2. In this example, the rows for $u = 2$ and $u = 3$ are marked by dashed line.

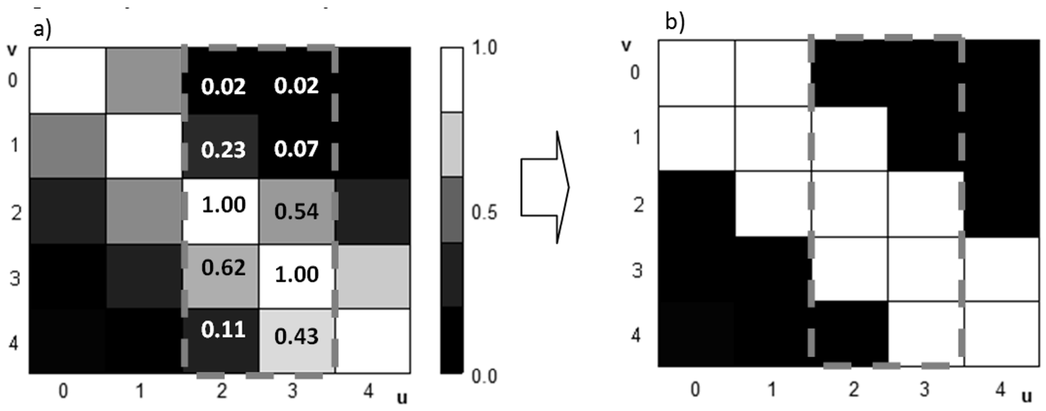


Fig. 4.2. The *UR* for noised sinusoid with period $T = 5$: a) estimated relation, b) after binarization.

The thresholding (binarization) of *UR* is performed by using the Renyi entropy approach (Fig. 4.2 b). The *UR* is necessary for calculating the hybrid usefulness-related distance *R* (Eq. 3.5). Based on distance *R* the selection of the *k* subseries that are the nearest to query $Q_{24,4}$ is performed. For the analysed example, the number of the nearest neighbours (k) equals 4. The necessary calculations are presented in Tab. 4.4. E_u is the Euclidean distance normalized to interval [0, 1]. The distance E_u is calculated based on the subseries as well as their first order differences.

Table 4.4. Calculation of the hybrid usefulness- related distance *R*

i	$S_{i,w}$					V	UR (4,*v*)	E_u $(S_{i,4}, Q_{24,5})$	E_u $(S'_{i,4}, Q'_{24,5})$	R
	0	1	2	3	4					
4	**-0.93**	**-0.64**	**0.59**	**0.979**	**-0.1**	**4**	**1**	**0.04**	**0.07**	**-0.94**
9	**-1.00**	**-0.60**	**0.56**	**0.93**	**0.04**	**4**	**1**	**0.06**	**0.09**	**-0.93**
14	**-1.04**	**-0.57**	**0.60**	**1.01**	**-0**	**4**	**1**	**0.07**	**0.11**	**-0.91**
8	**-0.60**	**0.56**	**0.93**	**0.04**	**-0.9**	**3**	**1**	**0.59**	**0.52**	**-0.45**
18	-0.65	0.54	0.88	0.03	-1	3	1	0.60	0.52	-0.44
13	-0.57	0.60	1.01	-0.04	-1	3	1	0.62	0.54	-0.42
10	-0.04	-1.00	-0.60	0.56	0.93	0	1	0.60	0.65	-0.37
15	0.03	-1.04	-0.57	0.60	1.01	0	1	0.62	0.67	-0.36
5	0.04	-0.93	-0.64	0.59	0.98	0	1	0.63	0.68	-0.35
7	0.56	0.93	0.04	-0.93	-0.6	2	0	0.96	0.83	0.90
17	0.54	0.88	0.03	-1.04	-0.6	2	0	0.97	0.86	0.92
12	0.60	1.01	-0.04	-1.00	-0.6	2	0	1.00	0.88	0.94
6	0.93	0.04	-0.93	-0.64	0.59	1	0	0.98	0.99	0.99
16	0.88	0.03	-1.04	-0.57	0.6	1	0	0.98	1.00	0.99
11	1.01	-0.04	-1.00	-0.60	0.56	1	0	0.99	0.99	0.99
$Q_{24.5}$	-0.91	-0.67	0.67	0.90	0	u=4				

According to the obtained values of distance *R*, the subseries $S_{4,w}$,$S_{9,w}$, $S_{14,w}$, and $S_{8,w}$ ($t_i \in \{4,9,14,8\}$) are selected as the nearest neighbours for further processing. Within the selected subseries, the optimal lags are searched by using Alg. 3. For the sake of simplicity, the lag cardinality (*y*) in this example is limited to 2 and parameter of the Kraskov's method (*g*) equals 2. The selected subseries are presented in the form of a matrix *B* (Tab. 4.5), where columns B_i, i = 0, ..., 4 contain the values of lags. Additionally, column *D* in Tab 4.5 contains predicted values x_{t_i+s}:

$$[x_{4+1}, x_{9+1}, x_{14+1}, x_{8+1}] = [\ 0.04, -0.04, 0.03, -1.00]. \qquad \text{(x)}$$

Table 4.5. Matrix used to find informative lags

	B_0	B_1	B_2	B_3	B_4	D
I_1	-0.93	-0.64	0.59	0.98	-0.08	0.04
I_2	-1.00	-0.60	0.56	0.93	0.04	-0.04
I_3	-1.04	-0.57	0.60	1.01	-0.04	0.03
I_4	-0.60	0.56	0.93	0.04	-0.93	-1.00

In the first step of the lag selection algorithm, the lag with the highest information value is selected. The value of information function is calculated for every lag separately. In case of a lag, which is equal to 0 ($B = B_0$), the matrix is narrowed to two columns, so the values E_n (Eq. 3.7) can be calculated easily for each instance. E.g., for instance I_1 the values of E_n are calculated as follows:

$$\begin{gathered} E_n(I_1, I_2) = \max(|-0.93 - -1.00|, |0.04 - -0.04|) = 0.08 \\ \underline{E_n(I_1, I_3) = \max(|-0.93 - -1.04|, |0.04 - 0.03|) = 0.11} \\ E_n(I_1, I_4) = \max(|-0.93 - -0.60|, |0.04 - -1.00|) = 1.04 \end{gathered} \tag{4.2}$$

The second instance ($g = 2$) in this ranking is I_3, therefore this instance is selected to build *B*- and *D*-hyper-rectangle. In this example, B-hyper-rectangle corresponds to the interval: [-0.93 - |-0.93 - -1.04|, -0.93 + |-0.93 - -1.04|] = [-1.04, -0.82], and D-hyper-rectangle is determined as: [0.04 - |0.04 - 0.03|, 0.04 + |0.04 - 0.03|] = [-0.03, 0.05].

Based on the B-hyper-rectangle, the values of function n_B and n_D are calculated. The calculations of these functions are illustrated in Fig. 4.3. The functions determine the number of instances that are inside the hyper-rectangle (interval in case of one lag).

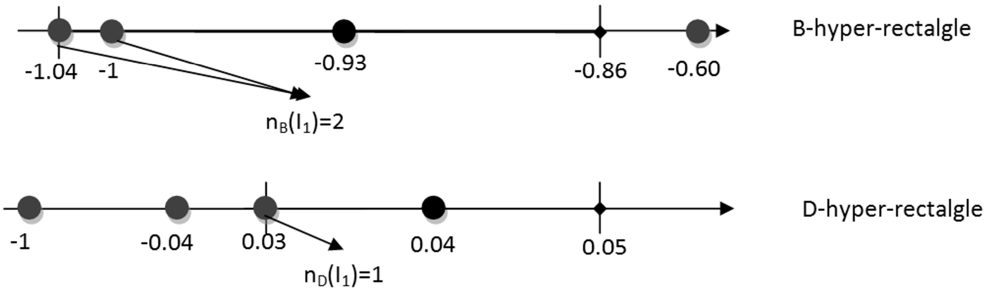


Fig. 4.3. Construction of hyper-rectangles for one lag.

Similarly, the n_B and n_D values are calculated for all instances. The results are as follows:

$$n_B(1) = 2, \quad n_D(1) = 1, \quad n_B(2) = 2, \quad n_D(2) = 2, \quad n_B(3) = 2, \quad n_D(3) = 1, \quad n_B(4) = 3, \quad n_D(4) = 2. \tag{4.3}$$

Finally, the value of information function is calculated by using Eq. (3.6):

$$\begin{aligned} I(B,D) &\approx \psi(g) - \frac{1}{g} - \frac{1}{z}\sum_{i=1}^{z}\left[\psi(n_B(i)) + \psi(n_D(i))\right] + \psi(m) = \\ &= \psi(2) - \frac{1}{2} - \frac{1}{4}\sum_{i=1}^{m}\left[\psi(n_B(i)) + \psi(n_D(i))\right] + \psi(4) = \\ &= -0{,}07722 - 0.5 - 0.25(-1.2843) + 0.5061 = 0.25 \end{aligned} \tag{4.4}$$

The value of information function for B_0 is equal to 0.25. In a similar manner, the value of information function is calculated for every lag: $I(B_0, D) = 0.25$, $I(B_1, D) = \mathbf{0.45}$, $I(B_2, D) = 0.33$, $I(B_3, D) = 0.12$, $I(B_4, D) = \mathbf{0.45}$. The highest value of the information function was obtained for B_1 and B_4. According to the proposed algorithm, the first lag, which has the highest value of the information function is selected, i.e., lag 1. The selected lag is added to matrix *B*. To select the second informative lag, the remaining lags ($i = 0,2,3,4$) are considered: $I(B_1 + B_0, D) = 0.58$, $I(B_1 + B_2, D) = \mathbf{0.70}$, $I(B_1 + B_3, D) = 0.70$, $I(B_1 + B_4, D) = 0.58$.

After two steps of the algorithm, lags 1 and 2 are selected (LL = {1, 2}). Therefore, the learning set LS for query $Q_{24,2}$ = [-0.67, 0.67] is determined as LS = {([-0.64, 0.59], 0.04), ([-0.60, 0.56], -0.04), ([-0.57, 0.60], 0.03), ([0.56, 0.93], -1.00)}. The selected LS is used by ML algorithms: NN, ANFIS and LS-SVM that allows the LM to be created. After training procedure, the LM is used to make the prediction for query $Q_{24,2}$.

5. Experiments

Verification of the proposed method was conducted using KNIME environment with support of R language and forecast libraries (Berthold et al., 2009). The proposed method was implemented in two variants. The first variant (denoted hereinafter as U-LM-*x*) is an exact implementation of the proposed approach, which was described in Section 3. This variant combines the LM training with the novel elements (training data selection based on the hybrid usefulness-related distance and lag selection based on the Kraskov information criteria). The second variant (U-*x*) is considered to verify the effectiveness of the proposed training data selection based on the binary *UR*. The U-*x* variant does not include the informative lag selection and *k* nearest neighbour finding, thus it is less complex. In this variant, the *UR* is directly applied for selection of the historical subseries. It means that the subseries $S_{i,w}$ is taken into account for prediction query $Q_{z,w}$, only if condition $UR(p(i), p(z)) = 1$ is satisfied. The *UR* is determined for period *T*, thus exactly *T* LMs are constructed in this variant (one model for each time step of the period).

As it was already mentioned in Sect. 1, in this study the three popular ML techniques are considered: ANN, ANFIS, and LS-SVM. Therefore, *x* in names of the prediction methods (U-LM-*x* and U-*x*) stands for the ML algorithm, which is used to construct the local prediction model (ANN, SVM or ANFIS). The ANN was implemented by using the probabilistic neural network based on dynamic decay adjustment, where data are labelled using constructive training (Štěpnička et al., 2013). The implementation of LS-SVM is based on the approach presented by Suykens & Vandewalle (1999) that derives a linear function, which best approximates the training data (LS). Finally, ANFIS was implemented, which generates a set of fuzzy rules from the given set of training patterns by applying input space partitioning (Chang et al. , 2011). Details of this implementation can be found in (Suykens & Vandewalle, 1999), (Yang et al., 2011) and (Kang et al., 2010).

Accuracy of the proposed method was compared against several state-of-the-art approaches. The first compared approach is the LM (LM-*x*) proposed by Wu & Lee (2015). Note that three different LM-*x* models are considered as *x* stands for ANN, SVM or ANFIS. The ML techniques were also applied without any modifications, as global prediction models (denoted as ANN, SVM, and ANFIS). Finally, seasonal ARIMA was used in the prediction experiments (Khashei et al., 2012). The parameters of seasonal ARIMA were selected on the basis of the Akaike information criterion and the Schwarz Bayesian criterion. More detailed information about ARIMA can be found in (Weigend & Gershenfeld, 1993) and (Khashei & Bijari, 2011).

In order to examine the advantages and weaknesses of the proposed approach, seven various TS were selected for the experiments. The effectiveness of the proposed method strongly depends on correct detection of period *T* for the analysed TS. The accuracy of period detection was verified in preliminary experiments that were conducted on a number of synthetic TS with 5000 data points. The synthetic TS were generated as sinusoids with various amount of additive Gaussian noise (Comp-Engine, 2015). The experimental results obtained for two representative synthetic TS are discussed later in this section. The considered synthetic TS are denoted here as Synth-1 and Synth-2. Synth-1 corresponds to single sinusoid of period $T = 10$ with higher amount of noise. Synth-2 represents a composition of two sinusoids with periods 100 and 50 with lower amount of noise.

The experimental set of data also includes five real-world TS. Two of them represent traffic volume that was measured over one year period from 2012 to 2013. The traffic data were collected at a road intersection in medium size city of Gliwice, Poland (Bernas et al., 2015). Traffic volume was measured in five-minute intervals for weekdays (Traffic-1) as well as for Sundays (Traffic-2). The third real-world TS (Dollar) represents the historical Euro to Dollar exchange rate in years from 1999 to 2015. The data were obtained from European Central Bank (ECB, 2015). Fourth TS (Sunspot) contains monthly mean number of relative sunspots registered from 1749 to 1983, which was collected at Swiss Federal Observatory, Zurich until 1960, and then Tokyo Astronomical Observatory (Andrews et al., 1985). Finally, the fifth TS (Riverflow) describes quarter-monthly river inflows of Lac St-Jean Reservoir, between 1953 and 1982 (Thompstone & Herzberg, 1985).

5.1. Calibration

During calibration process, the *UR* was extracted for each test TS. To this end, the analyzed TS were divided into two separate subseries of equal lengths, called historical sequence and verification sequence. The first sequence is used to calibrate the model, while the second one is used to evaluate the prediction accuracy of the model. The parameters of LMs (*w*, *y* and *k*) correspond to initial lag cardinality, number of selected informative lags and number of nearest neighbours. These parameters are optimized by means of cross-validation. Although the method can be applied to make predictions for an arbitrary number of time steps (*s*), the single-step prediction ($s = 1$) is considered here without loss of generality. The calibration process and the accuracy evaluation were performed by taking into account two error metrics: the root mean squared error (RMSE), and the mean absolute error (MAE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left[x_t - \hat{x}_t\right]^2}\ , \tag{5.1}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{t=1}^{N}\left|x_t - \hat{x}_t\right|, \tag{5.2}$$

where: N is the total number of the analyzed test queries, x_t denotes the real value at time point t, and $\hat{x}_t$ stands for result of prediction for time point t. The aim of the calibration process was to find a set of parameters, for which the RMSE value is minimized.

An example of the calibration process for traffic-1 TS is presented in Fig. 5.1. The results shown in Fig. 5.1 were obtained by using the U_LM_ANN algorithm. In the analysed example (Fig. 5.1), comparable values of RMSE are achieved for $y = 3$ and $y = 6$. It was noticed that close to the optimal parameter value, the error level is not changing rapidly. The optimal value of k is 90. For such settings, the low prediction error was obtained with parameter w above 10. This observation can be utilized to estimate upper boundary for values of the optimized parameters. Based on these results, the following parameter values were selected for further experiments: $y = 3$, $w = 15$ and $k = 90$.

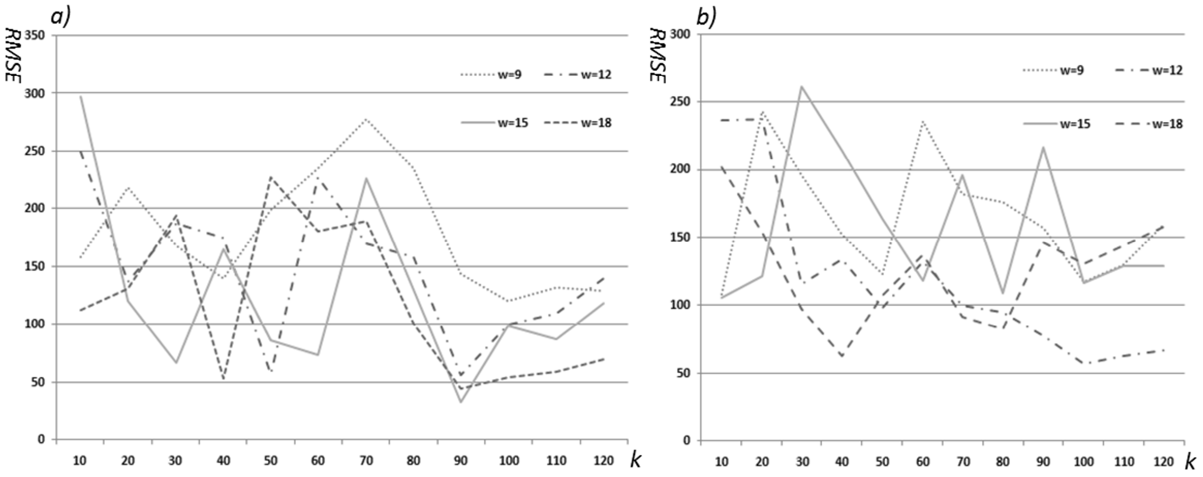


Fig. 5.1. Model calibration for Traffic 1 TS: a) y = 3, b) y = 6.

Each examined prediction method is calibrated separately. The results of the calibration are summarized in Tab. 5.1. The parameters were calibrated simultaneously. It was noticed that for complex real-world TS more lags (above 4) are required to build effective prediction model. The lowest number of lags was necessary for the synthetic TS (Synth 1 and Synth 2), while the highest number of lags was required in case of the traffic volume prediction. It was also observed that the amount of noise influences the optimal number of nearest neighbours (*k*). In case of the TS with low amount of noise (Synth 2), the optimal number of nearest neighbours is lower than for the more noisy TS (Synth 1).

Table 5.1. Calibration results for U-LM-*x* models

Data set	Model	Parameters			RMSE
		w	y	k	
Synth 1	U-LM-ANN	4	3	140	93.29
	U-LM-SVM	9	6	110	31.66
	U-LM-ANFIS	10	6	170	49.23
Synth 2	U-LM-ANN	8	3	30	101.88
	U-LM-SVM	6	4	10	45.07
	U-LM-ANFIS	5	3	40	92.34
Traffic 1	U-LM-ANN	15	3	90	3.76
	U-LM-SVM	10	9	80	1.9
	U-LM-ANFIS	8	3	70	1.0
Traffic 2	U-LM-ANN	19	11	160	8.33
	U-LM-SVM	10	4	100	5.22
	U-LM-ANFIS	4	3	140	7.34
Dollar	U-LM-ANN	13	6	120	8.71
	U-LM-SVM	8	6	200	10.60
	U-LM-ANFIS	5	3	170	10.51
Riverflow	U-LM-ANN	10	5	80	531.18
	U-LM-SVM	9	6	110	412.31
	U-LM-ANFIS	7	4	100	213.32
Sunspot	U-LM-ANN	11	5	120	22.17
	U-LM-SVM	8	5	150	10.42
	U-LM-ANFIS	7	4	130	16.84

The number of lags selected by using the Kraskov information criteria (*y*) is lower than the number of lags that were used to build the *UR* (*w*). Thus, the Kraskov information criteria allow us to simplify the LM by reducing the set of lags. In consequence, it is possible to decrease the number of inputs in the prediction model and speed up the training process. The reduction of lag number is especially visible for ANFIS and ANN models.

The calibration results for U-*x* models are presented in Tab. 5.2. In case of these models, the *UR* is directly used for the selection of training data. Thus, only the number of lags (*w*) has to be tuned. The obtained results show that the optimal values of parameter *w* are similar for the two variants of the proposed method. Therefore, the calibration process for U-LM-*x* models can be simplified, i.e., parameter *w* can be selected first and then the remaining parameters can be optimized independently.

Table 5.2. Calibration results for U-x models

Data set	Model	w	RMSE
Synth 1	U-ANN	**5**	70.21
	U-SVM	**9**	41.73
	U-ANFIS	**9**	72.73
Synth 2	U-ANN	**7**	103.23
	U-SVM	**5**	99.20
	U-ANFIS	**6**	112.69
Traffic 1	U-ANN	**13**	6.72
	U-SVM	**9**	7.01
	U-ANFIS	**9**	8.21
Traffic 2	U-ANN	**15**	8.23
	U-SVM	**13**	7.13
	U-ANFIS	**6**	8.38
Dollar	U-ANN	**13**	10.54
	U-SVM	**9**	10.62
	U-ANFIS	**9**	10.03
Riverflow	U-ANN	**11**	683.53
	U-SVM	**8**	521.93
	U-ANFIS	**7**	582.64
Sunspot	U-ANN	**10**	21.37
	U-SVM	**8**	11.31
	U-ANFIS	**9**	16.98

The calibration results firmly show that U-LM-*x* models can be adapted to historical data more precisely (in terms of RMSE) than the simplified models (U-*x*). Nevertheless, the tuning of three parameters for U-LM-*x* models simultaneously is more complex task than the calibration of one parameter for U-x model.

Sun et. al (2013) have observed the stability of parameters close to optimal solution. Thus, the stability of the obtained parameters can be analysed to avoid local minimums. This remark is especially helpful in case of three parameters to tune, where it can be used to decrease the computational complexity. In order to further reduce the computational complexity of tuning process, the evolutionary algorithms can be used, e.g., the PSO method (Xionga e. Al. ,2015). In case of analysed TS the upper value of parameters did not exceed 19, 11 and 200 for *w*, *y* and *k*, respectively. Thus, these values could be used as upper boundary in the tuning process.

While tuning the global models based on ANN, LS-SVM and ANFIS, it was observed that the optimal lag number is significantly higher than this for the same ML method used as LM. In case of global models, this parameter was on average three times bigger. Therefore, the LMs can be trained significantly faster not only due to the reduced training data set, but also due to the lower complexity of LM.

5.2. Experimental results

The proposed method is suitable for TS that describe some periodic processes. The *UR* can be determined only if period *T* of the TS is found. In case of synthetic TS (Synth 1, Synth 2) the period *T* is known in advance and Algorithm 1 correctly determined it. Results of initial experiments on synthetic TS show that the length of TS and the amount of noise influence the effectiveness of Algorithm 1. The period *T* was correctly determined if the analysed TS contains at least eight to nine full periods. If TS contains much more than eight periods then the results of period detection are more resilient to the noise. The robustness of the period finding algorithm was also analysed for the five above mentioned real-world TS.

The obtained *UR*s show which data are useful for the prediction making at particular steps of the TS period. It should be noted here that Figs. 5.2 - 5.4 present the *UR*s before binarisation. Strong periodical changes in TS result in high values that appear at diagonals of the plots in Fig. 5.2. For the Synth 2 TS, the high values outside the diagonal correspond to the second (shorter) period. The regular and sharp pattern obtained for Synth 2 is a result of smaller amount of the Gaussian noise. Due to the present noise, the high values are scattered around the diagonal in the *UR* plots. If no noise is present, the maximum *UR* elements create a thin diagonal line pattern.

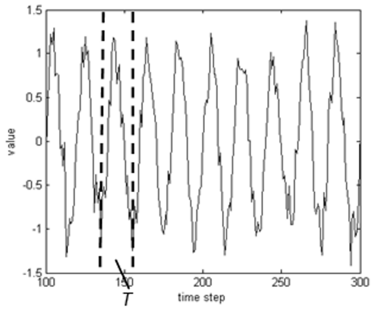
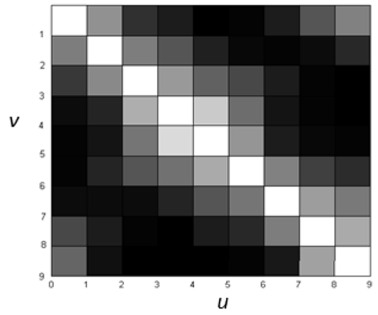
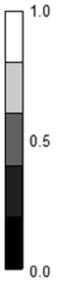
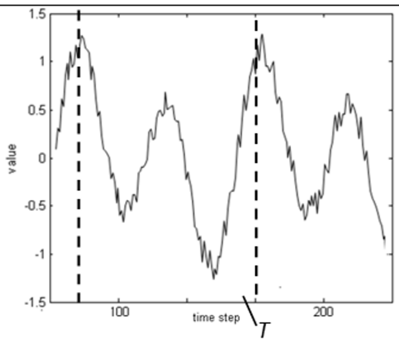
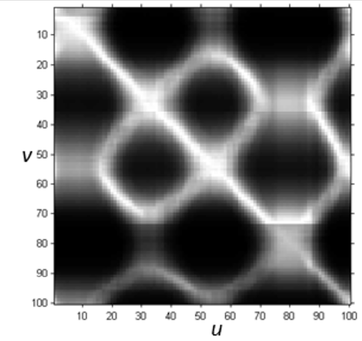
TS	a) overview	b) UR relation	Scale
Synth 1			
Synth 2			

Fig. 5.2. The synthetic time series: a) overview, b) *UR* plot.

Figures 5.3 and 5.4 show the results of *UR* extraction for the real-world TS. The traffic volume TS (Traffic1 and Traffic2) are presented in Fig 5.3. In these TS, the cyclic changes of traffic volume have the period of one day. Algorithm 1 has correctly recognized the one-day period, which corresponds to 288 time steps. The *UR*s determined for traffic volume in weekdays (Traffic1) and during Sundays (Traffic2) are similar. The traffic at night (for *u* between 210 and 50) and during daytime (for *u* between 50 and 210) differs significantly and is clearly separated in the *UR* plot. Additionally, in case of Sunday traffic (Traffic2), the traffic volumes during daytime vary significantly: the morning traffic is more intensive than in the afternoon. In Fig. 5.3 the above mentioned areas of *UR* plot were separated by dashed lines. The dashed lines separate the time intervals for which the predictions will be made by using different segments of the historical TS.

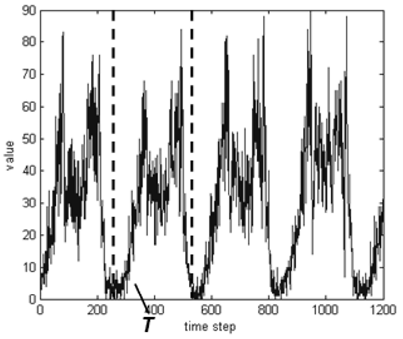
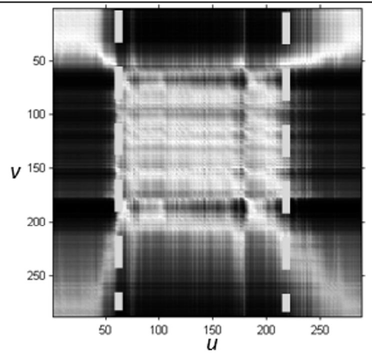
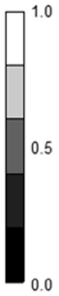
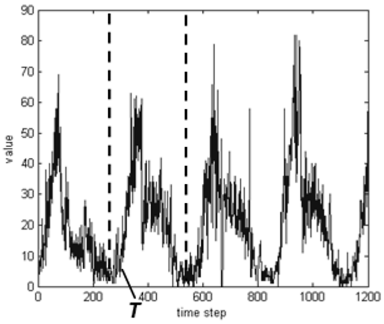
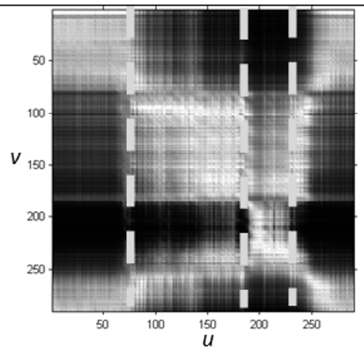
<table>
<tr><th>TS</th><th>a) overview</th><th>b) UR relation</th><th>Scale</th></tr>
<tr><td>Traffic
1</td><td>
</td><td>
</td><td rowspan="2">
</td></tr>
<tr><td>Traffic
2</td><td>
</td><td>
</td></tr>
</table>

Fig. 5.3. The traffic volume TS: a) overview, b) *UR* plot.

The last three real-world TS and their *UR* plots are presented in Fig. 5.4. The recognized period for Sunspot TS equals 110 time steps. Based on the information in (Andrews et al., 1985), it can be concluded that the period is recognised correctly. Two time intervals can be distinguished in the *UR* plot for the Sunspot TS. One part of the TS (for *u* up to 60) has an increasing trend, while in the second part the trend is decreasing (for *u* above 60). According to the obtained *UR* plot for time steps that correspond to $u \leq 60$ the LMs will be trained based on the historical data from the segment indicated by *v* values between 0 and 30. This fact shows that the proposed approach significantly reduce the size of training data set

Less regular cyclic patterns are observed in the River-flow and Dollar TS. The detected period for these TSs is 49 and 366 respectively. Both these values approximately correspond to one year period (expressed in number of weeks and number of days). The period for Dollar is more precisely defined because the analysed TS was much longer. In case of the River-flow TS the data collected only for 12 periods were taken into account. Intuitively, the river flows depend on the seasons. For winter weeks, the flow differs significantly from those in other seasons. This fact is reflected in the *UR* plot, where the dashed lines separate the time interval corresponding to the winter weeks. In case of the Dollar TS it is hard to clearly distinguish data segments in the *UR* plots. The exchange rate is influenced by many time-independent factors, thus the *UR* plot for the annual period is blurred. Nevertheless, some annual cyclic pattern can be observed in the TS (the diagonal line).The obtained *UR* allows us to ignore the data that are not useful for training local prediction models. These data segments are indicated in the *UR* plot by dashed rectangles.

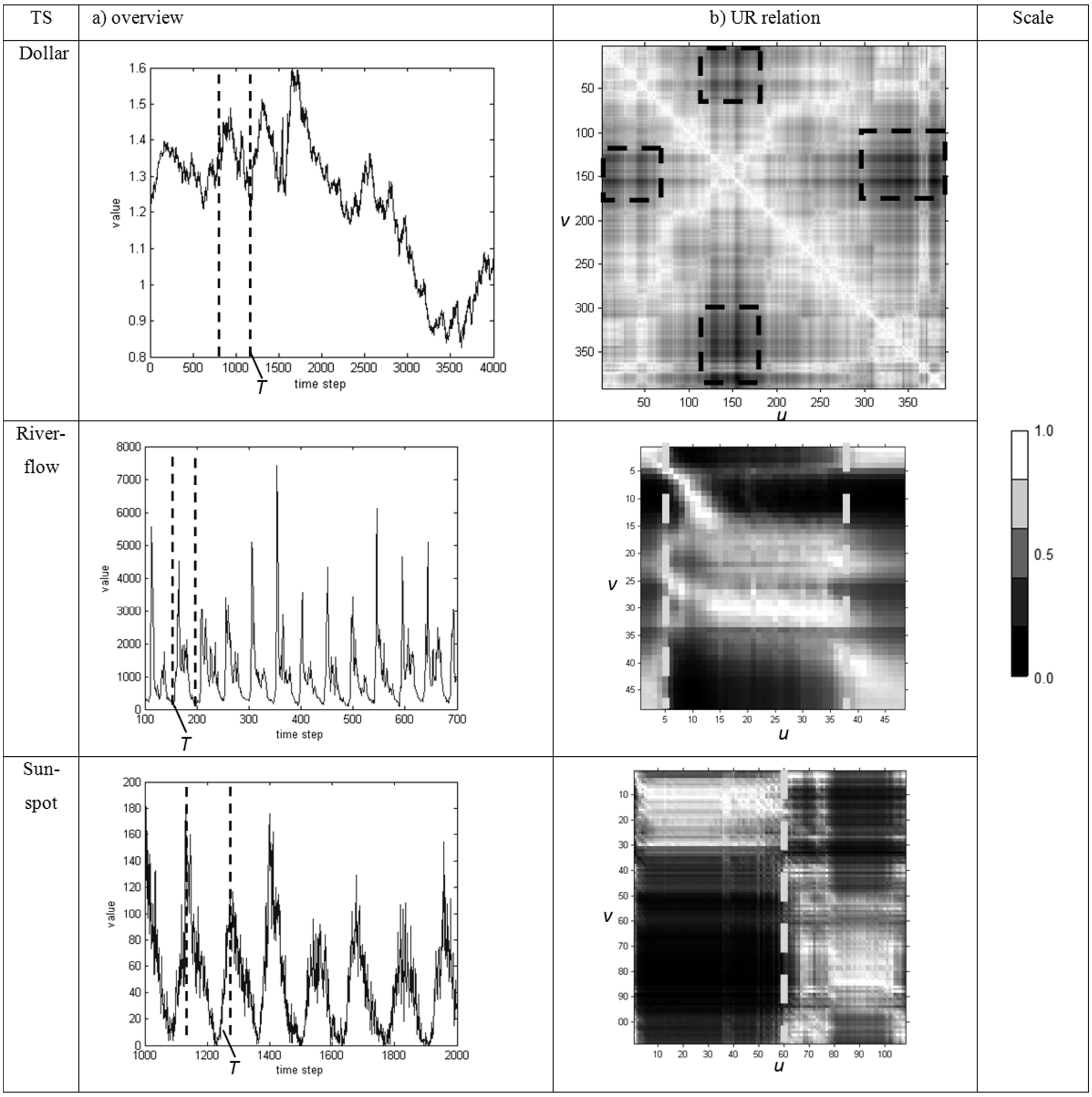


Fig. 5.4. Dollar, river flow, and sunspot TS: a) overview, b) *UR* plot.

The results obtained for Dollar TS shows the strengths and weakness of proposed method. If strong cyclic patterns in the analysed TS can be found then the *UR* relation enables significant reduction of the training data set. However, if the cyclic pattern is week then almost all historical data in TS have to be taken into account while training the LM. In such case, the extra cost of *UR* calculation may be not compensated by the reduced cost of model training.

The *UR*s described above, as well as the method parameters were determined based on the cross validation approach. Then, the obtained results were used for training the proposed model (U-LM-*x*) and its modification (U-*x*). The prediction accuracy of these models was compared against this of the LMs proposed in (Wu & Lee, 2015), global models based on the ML algorithms (ANN, SVG and ANFIS) and seasonal ARIMA. As discussed earlier in this section, the analysis of prediction accuracy is based on two error metrics: MAE (Tab. 5.3) and RMSE (Tab. 5.4).

Table 5.3. MAE values for the compared methods

Method	Time series						
	Dollar	Traffic-1	Traffic-2	Riverflow	Sunspot	Synth-1	Synth-2
ARIMA	8.51	6.16	5.43	716.18	22.58	84.65	59.54
ANN	11.29	7.00	6.33	547.74	21.46	89.73	96.33
LM-ANN	9.68	7.02	6.42	651.66	24.20	60.59	97.67
U-ANN	9.51	5.31	5.42	545.33	19.04	62.41	96.33
U-LM-ANN	7.21	6.38	6.01	641.82	20.30	82.27	104.19
SVM	10.46	8.91	6.65	404.05	10.37	61.23	98.09
LM-SVM	8.36	9.00	6.45	394.52	10.17	57.82	79.41
U-SVM	8.21	6.32	5.95	368.99	11.44	33.78	92.04
U-LM-SVM	7.54	8.87	5.60	345.30	9.43	37.56	52.96
ANFIS	29.05	11.25	7.34	927.32	25.38	94.14	116.23
LM-ANFIS	12.40	10.00	5.80	399.25	25.86	67.85	114.64
U-ANFIS	8.70	7.50	6.41	456.96	15.78	58.09	95.18
U-LM-ANFIS	7.55	6.25	6.53	332.89	22.92	47.26	92.84

Table 5.4. RMSE values for the compared methods

Method	Time series						
	Dollar	Traffic-1	Traffic-2	Riverflow	Sunspot	Synth-1	Synth-2
ARIMA	11.15	8.24	7.83	1116.07	22.85	93.35	67.58
ANN	14.25	12.18	9.61	1037.41	28.23	108.04	110.83
LM-ANN	12.07	8.90	9.28	1120.45	34.05	75.45	130.17
U-ANN	11.82	7.03	8.97	1037.35	23.43	77.71	110.83
U-LM-ANN	9.77	7.85	9.03	1060.33	30.84	100.17	125.35
SVM	12.23	8.95	7.62	763.64	12.54	64.91	109.75
LM-SVM	11.13	8.97	8.02	724.36	12.64	59.86	106.52
U-SVM	11.00	7.50	7.17	641.35	12.17	46.42	101.39
U-LM-SVM	10.65	8.97	6.91	581.26	10.59	39.99	67.25
ANFIS	29.92	11.32	9.84	1775.81	30.15	110.20	182.56
LM-ANFIS	12.67	11.18	7.29	665.72	34.43	82.65	127.70
U-ANFIS	10.74	10.61	8.93	690.96	18.74	76.10	121.56
U-LM-ANFIS	10.61	6.37	8.13	525.63	25.11	66.48	113.85

The lowest values of the prediction errors are underlined in Tab. 5.3 and Tab. 5.4. The minimum values of MAE and RMSE were obtained for the same methods. The exceptions are the traffic volume TS (Trafic-1 and Traffic-2). In case of these two TS, the lowest value of MAE was achieved using the U-ANN approach, while minimum RMSE was obtained for U-LM-ANFIS and U-LM-SVM methods.

The proposed LM achieved higher prediction accuracy than the global models (ANN, SVM and ANFIS). The average error was lower by 5.4%, 21.7%, 39.0% in terms of MAE and by 7.9%, 19.7%, 41.4% in terms of RMSE, respectively for the considered ML algorithms.

The proposed U-LM-*x* models allow us to obtain lower error values than the state-of-the-art LM-*x* models: MAE was reduced by 2.3%, 16.0% and 20.1% and RMSE was decreased by 2.7%, 17.7% and 18.0% for the particular ML algorithms. The proposed approach can achieve better results than the state-of-the-art LMs as it takes into account the periodicity of TS. It should be also noted that the difference between U-LM-*x* and LM-*x* models lies in the training data selection. In case of LM-*x* models, the training data are selected from the entire historical TS by using the *k*-NN approach with Euclidean distance. In the proposed solution (U-LM-*x*) the training data are searched in selected segments of TS, by taking into account the hybrid usefulness-related distance. If the number of historical subseries in the selected data segments is insufficient then this method utilizes the sequences from outside of the segments determined by *UR*. However, if the historical TS is multiple times longer than the searched sequence then the data not belonging to the selected segments can be ignored during the searching procedure. In case of big data sets, this approach speeds up the training process proportionally to the amount of ignored data.

The experimental results also show that the proposed methods (U-LM-*x*) based on SVM and ANFIS achieve 21% and 4% lower average error than the seasonal ARIMA in terms of both MAE and RMSE. In case of U-LM-ANN the obtained average error was higher by 7% (MAE) and 17% (RMSE) when comparing with the seasonal ARIMA. However, in case of the Dollar TS, which is the least periodical one, the U-LM-ANN model achieved better prediction accuracy than all the remaining models. It was observed that the ANN algorithm allows the low prediction error to be obtained for the TS with weakly periodic pattern.

The direct application of data selection based on *UR* with ML methods (U-*x* models) offers lower prediction error 14%, 13% and 35% (in terms of MAE and RMSE) in comparison to the global models. However the U-*x* models proved to be inferior to U-LM-*x*. These results show that the pre-processing based on *UR* improves the accuracy of the global prediction model. The determination of *UR* allows us to create exactly *T* quasi-LMs that can be trained offline. This enables fast processing of the prediction queries. The quasi-LM is selected by taking into account the time step defined by the query. The additional advantage of this approach is the simplified calibration procedure. Only one parameter need to be tuned (number of lags). The UR relation can be updated periodically, when new historical data are collected.

When considering the prediction accuracy for the different ML techniques, it is hard to say if one technique is generally better than the rest. SVM provides the best results while predicting the synthetic TS, traffic volumes in Sundays and sunspot numbers. These TS are characterized by strong periodic patterns. In case of TS with weaker periodic patterns (weekday's traffic and river flow), ANFIS proved to be the best technique. Finally, the ANN gave the best results when predicting the TS of dollar to euro exchange rate, for which the periodicity is not so evident.

An important benefit of using the LMs is reduced complexity of the model and lower number of lags that are used for model training. For instance, in case of the ANN model, which was trained for the Dollar TS, the number of inputs (lags) is 14 and the number of neurons is 30. In contrast, the U-LM-ANN model has 5 inputs, 11 neurons and uses limited number of sequences for training (120). Thus, the training time is significantly shorter for the LM. The reduction of model complexity was observed for all considered TS. The highest reduction was obtained for the synthetic TS (Synth 1 and Synth 2) as well as for the traffic TS (Traffic 1 and Traffic 2).

The proposed extraction of *UR* at the pre-processing stage can considerably reduce the amount of data that have to be analysed for finding the nearest neighbours and processed by ML methods. In the analysed examples, the highest data reduction, about 90%, was achieved for the synthetic sinusoid TS, while the minimal reduction (12%) was achieved in case of Dollar TS. The percent of reduced data is closely related to the strength of the periodic components and the amount of noise in TS. The above results are important for applications that require real-time data processing, e.g., in intelligent transport systems.

An advantage of the proposed approach is that the interpretation of the *UR* is intuitive. Moreover, the *UR* extraction algorithm does not change the data representation , hence there is no loss of the information contained in TS. The higher accuracy of the prediction can be obtained if the data for model training are searched in segments identified by the *UR*.

The above experimental results demonstrate advantages of the proposed approach. However, this approach has some limitations. Firstly, it can be applied for TS, where some periodic patterns can be detected. Secondly, the method requires a pre-processing stage to extract the *UR*. The computational complexity of the *UR* extraction algorithm is $O(n^2)$. The extraction of *UR* and model calibration can be executed periodically off-line, by using historical data.

Another disadvantage is related to the fact that the local prediction model has to be constructed independently for each query. An alternative solution is to use the second variant of the proposed method (U-*x* models) with quasi-LMs that can be trained offline. The number of the quasi-LMs corresponds to the period TS.

Other possible research directions that can improve the accuracy of the method include application of fuzzy UR relation and modification of the information criteria for lag selection.

6. Conclusion

Prediction of TS is a practical issue of a great interest to both industry and academia. Many researchers use the ML techniques, e.g., fuzzy logic, support vector machines, evolutionary computations, to predict the values of TS. In this paper, the period-aware approach to local modelling was proposed for TS prediction. According to the period-aware approach, usefulness of data is evaluated by taking into account the period of TS. Based on the usefulness analysis, data are selected for training a LM. Results of the experiments show that the proposed approach improves the prediction accuracy of the LMs when some periodic patterns in TS can be found. The result obtained for dollar TS shows that the prediction error is reduced even if the periodic pattern is very weak.

The modification of the Box & Jenkins algorithm (Box & Jenkins, 2008) allows us to find the strongest periodic pattern required for building *UR*. The algorithm was tested for various TS and proved robust to noise.

The proposed algorithm extracts the *UR* form historical (training) TS and enables selection of the data that are useful for making prediction at a given time step of the TS period. The *UR* is build using similarity and prediction error ranking. It is calculated for each data point separately, thus can be used for various TS

A definition of hybrid distance was introduced, which takes into account the data usefulness. The usefulness-related hybrid distance is used to find nearest neighbours for learning a LM. Experimental results show that a higher accuracy of the prediction is obtained if the nearest neighbours are searched among the useful data that are identified by the proposed *UR*. This effect can be explained by the fact that for the selected data the probability of finding a neighbour, which will give a wrong prediction, is lower than for entire TS. Moreover, the application of Kraskov information criteria allowed us to select lags that contain relevant information. As a result, the LMs were significantly simplified by reducing the number of inputs.

The extraction of *UR* is time-expensive however it is executed off-line, before running the prediction algorithm. In practice, the extraction of *UR* could be performed periodically at the time when the computational resources of the system are utilized to a small extent. Therefore, the extra time cost of the extraction is negligible in comparison with the benefit of improved prediction accuracy. Note that in the presented experiments the extraction of *UR* was executed only once for a given dataset and its results were used in the prediction during test stage.

In this study, the proposed period-aware local modelling approach was experimentally evaluated for prediction models obtained by using three different ML methods: neural network, adaptive neuro-fuzzy inference system, and least squares support vector machine. The application of LMs was based on the method proposed in (Wu & Lee, 2015). Effectiveness of the period-aware approach was also verified by using the LMs for which the training data were selected based on the Kraskov's mutual information criteria. Moreover, the prediction accuracy of LMs was compared against that of the seasonal ARIMA. The experimental results are encouraging. They show that the proposed period-aware local modelling approach provides more accurate predictions for periodic TS than the seasonal ARIMA, global models, and the state-of-the-art LMs without period-awareness.

The drawback of proposed model is that it requires a sound period detection algorithm. The detected period should not change in time. Additionally, the computational cost of this method is higher due to the *UR* calculation. However, the extra cost can be compensated by significant training data reduction in case of big datasets. The simplified U-x model allows us to prepare quasi-LMs in advance. This approach is especially useful for real time processing. Alternatively, the models could be trained on demand and saved for further use by next queries.

Further research will be conducted to examine the proposed approach for noise-insensitive TS. Additionally, the *UR* will be extracted using distance time warping for TS, where period changes over time. In order to decrease the time overhead of local modelling, faster algorithms will be explored. Such algorithms can utilize the fact that when using the proposed *UR* one can decrease the number of LMs that can be pre-calculated and tested in parallel, so that the computational burden can be shifted to the pre-processing stage.

References

Adhikari, R. & Agrawal, R. K. (2011). A homogeneous ensemble of artificial neural networks for time series forecasting. International Journal of Computer Applications, Vol. 32 (7), pp. 1–8

Al-Hmouz, R., Pedrycz, W., Balamash, A. (2015). Description and prediction of time series: A general framework of granular computing. Expert Systems with Applications, 42 (10), pp. 4830–4839

Andrews, D. F. & Herzberg, A. M. (1985). Data: A Collection of Problems from Many Fields for the Student and Research Worker. New York: Springer-Verlag, ISBN: 978-1-4612-9563-1

Askari, S., Montazerin, N. (2015). A high-order multi-variable fuzzy time series forecasting algorithm based on fuzzy clustering. Expert Systems with Applications, 42 (4), pp. 2121–2135

Barber, D. (2012). Bayesian reasoning and machine learning, Cambridge University Press

Bernaś, M., Płaczek, B., Porwik, P. & Pamuła, T. (2015): Segmentation of vehicle detector data for improved k-nearest neighbours-based traffic flow prediction. IET Intelligent Transport Systems, Vol. 9 (3), pp. 264 – 274

Berthold, M. R., et al. (2009). KNIME - the Konstanz information miner: version 2.0 and beyond. AcM SIGKDD explorations Newsletter, 2009, Vol. 11 (1), pp. 26-31

Box, G. & Jenkins, G. (2008). Time series analysis: forecasting and control. Wiley, Hoboken, N.J., ISBN: 978-0-470-27284-8.

Božić, M., Stojanović, M., Stajić, Z. & Floranović, N. (2013). Mutual information-based inputs selection for electric load time series forecasting. Entropy, 15, pp. 926–942

Brillinger, D. (2011) The Spectral Analysis of Stationary Interval Functions. Selected Works in Probability and Statistics, Springer, pp. 25-55

Chang, H., Lee, Y., Yoon, B., Baek, S. (2012): Dynamic near-term traffic flow prediction: system oriented approach based on past experiences. IET Intelligent Transport Systems, 2012, 6, (3), pp. 292-305

Chaudhuri, A. & Kajal, D. (2011). Fuzzy support vector machine for bankruptcy prediction. Applied Soft Computing, Vol. 11 pp. 2472-2486

Chen, M., Chen C. & Chang Y. (2010). Comparison of support vector machine and support vector regression an application to predict financial distress and bankruptcy. IEEE conf on Service Systems and Service Management (ICSSSM), pp.1-6

Comp-Engine (2015) Time Series Repository webpage: http://www.comp-engine.org/timeseries/time-series_data

Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems. Vol. 2. pp. 303-314

Dabrowski, J.J., Villiers, J.P. (2015). Maritime piracy situation modelling with dynamic Bayesian networks. Information Fusion, 23 (0), pp. 116–130

Denton, A., (2004) Density-based clustering of time series subsequences. In Proceedings of the IEEE International Conference on Data Mining (ICDM '04)

Du W., Leung S.Y.S., Kwong C.K. (2014). Time series forecasting by neural networks: A knee point-based multiobjective evolutionary algorithm approach. Expert Systems with Applications, 41 (18), pp. 8049–8061

European Central Bank (ECB) statistics (2015). Euro foreign exchange reference rates. https://www.ecb.europa.eu/stats/exchange

Frank, R., Davey, N., Hunt, S. (2001). Time series prediction and neural networks. Journal of Intelligent and Robotic Systems, 31 (1-3), pp. 91–103

Gestel, T., Baesens, B., Suykens, J. & Espinoza, M. (2003), Bankruptcy prediction with least support vector machine classifiers, Computational Intelligence for Financial Engineering, pp. 1-8, doi: 10.1109/CIFER.2003.1196234

Gooijer, J. G. & Hyndman, R. J. (2006). 25 years of time series forecasting. International Journal of Forecasting, 22 (3), pp. 443–473

Górecki T., Łuczak M. (2015). Multivariate time series classification with parametric derivative dynamic time warping. Expert Systems with Applications, 42 (5), pp. 2305–2312

Hastie, T., Tibshirani, R. & Friedman J. (2008). The elements of statistical learning. Springer, New York, ISBN 978-0-387-84858-7.

Huang, Z., Ouyang, H., Tian, Y. (2011): Short-Term Traffic Flow Combined Forecasting Based on Nonparametric Regression. In: International Conference on Information Technology, Computer Engineering and Management Sciences ICM 2011, 2011, vol. 1, pp. 316-319

Huang, Z., & Shyu, M. L., (2010). k-NN based LS-SVM framework for long-term time series prediction. In: Proceedings of IEEE international conference on information reuse and integration. pp. 69–74

Izakian, H., Pedrycz, W. , Jamal, I..(2015). Fuzzy clustering of time series data using dynamic time warping distance. Engineering Applications of Artificial Intelligence. Vol. 39, 235–244

Jang, J. S., Sun, C. T. & Mizutani, E. (1997). A computational approach to learning and machine intelligence. Neuro-Fuzzy and Soft Computing. Prentice-Hall, Inc., Upper Saddle River, New Jersey.

Jyh-Shing, R. J. (1993). ANFIS : Adaptive-network-based fuzzy inference system. IEEE Transactions on systems, man, and cybernetics, vol. 23 No.3, pp. 665–685.

Joo, T.W., Kim S.B. (2015). Time series forecasting based on wavelet filtering. Expert Systems with Applications, 42 (8), pp. 3868–3874

Kaneko, N., Matsuzaki, S., Ito, M., Oogai, H. & Uchida, K. (2010). Application of improved local models of large scale database-based online modeling to prediction of molten iron temperature of blast furnace. ISIJ International, Vol. 50 (7), pp. 939–945

Kang, D. K. & Myoung-jong K. (2010). Performance enhancement of SVM ensembles using genetic algorithms in Bankruptcy prediction. 3rd international conference on advanced computer theory and engineering, Vol. 2, pp. 154-158

Khashei, M., Bijari, M.(2011). A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. Applied Soft Computing. Vol. 11 (2), pp. 2664–2675

Kocadagˇli O., Aşikgil B. (2014). Nonlinear time series forecasting with Bayesian neural networks

Expert Systems with Applications, 41 (15), pp. 6596–6610

Kourentzes N., Barrow D.K., Crone S.F. (2014). Neural network ensemble operators for time series forecasting. Expert Systems with Applications, 41 (9), pp. 4235–4244

Kraskov, A. H. & Stogbauer, P (2004). Grassberger. Estimating mutual information. Physical Review E, 69 (6), pp. 66-138

Lennox, K. P., Dahl, D. B., Vannucci, M., Day, R. & Tsai, J. (2010). A Dirichlet Process Mixture of Hidden Markov Models For Protein Structure Prediction. The Annals of Applied Statistics 2010, Vol. 4 (2), pp. 916–942 DOI: 10.1214/09-AOAS296

Lin, F., Yeh, C. & Lee, M. (2011). The use of hybrid manifold and support vector machines in the prediction of business failure, Knowledge based system, Vol. 24, pp. 95-101

Lin, J., Chen, C. & Peng, C.(2012) Kalman filter decision systems for debris flow hazard assessment. Natural Hazards, Vol. 60 (3), pp. 1255-1266

Martinez-Rego, D., Fontenla-Romero, O. & Alonso-Betanzos, A. (2011). Efficiency of local models ensembles for time series prediction. Expert Systems with Applications, Vol. 38 (6), pp. 6884–6894

Maszczyk, T. & Duch, W. (2008). Comparison of Shannon, Renyi and Tsallis Entropy used in Decision Trees, Lecture Notes in Computer Science, Vol. 5097, pp. 643-651

Piero, P. B. (2000). Adaptive neural fuzzy inference systems (ANFIS): Analysis and applications. Lecture notes

Płaczek, B. (2013). A Traffic Model Based on Fuzzy Cellular Automata. Journal of Cellular Automata (ISSN: 1557-5969), Vol. 8(3-4) pp. 261-282

Lai R.K., Fan C.-Y., Huang W.-H., Chang P.-C. (2009). Evolving and clustering fuzzy decision tree for financial time series data forecasting. Expert Systems with Applications, 36 (2, Part 2), pp. 3761–3773

Lee, S. J. & Ouyang, C. S. (2003). A neuro-fuzzy system modeling with self-constructing rule generation and hybrid svd-based learning IEEE. Transactions on Fuzzy Systems, Vol. 11 (3), pp. 341–353

Lennox, K. P., Dahl, D. B., Vannucci, M., Day, R. & Tsai, J. W. (2010). A Dirichlet Process Mixture Of Hidden Markov Models For Protein Structure Prediction. The Annals of Applied Statistics, Vol. 4, No. 2, 916–942, doi: 10.1214/09-AOAS296

Liang, X., Xu, M., Yuan, X., Ling, T., Choi, h., Zhang, F., Chen, L., Liu, S., Su, S., Qiao, F., He, Y., Wang, J., Kunkel, K., Gao, W., Joseph, E., Morris, V., Yu, T., Dudhia, J. & Michalakes, J. (2012). Regional Climate–Weather Research and Forecasting Model. Bull. Amer. Meteor. Soc., Vol. 93, pp. 1363–1387, doi: http://dx.doi.org/10.1175/BAMS-D-11-00180.1

Lin, J., Keogh, E., and Truppel, W. (2003). Clustering of streaming time series is meaningless. In Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD ’03), pp. 56–65, San Diego

Lu, W., Pedrycz, W., Liu, X., Yang, J., Li, P. (2014) The modeling of time series based on fuzzy information granules. Expert Systems with Applications, 41 (8), pp. 3799–3808

Marschall, T., Rahmann, S. (2009). Efficient exact motif discovery. Bioinformatics, Vol. 25 (12),pp. 356-364

Sezgin, M. & Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging, Vol. 13 (1), pp. 146 –165.

Silviu, G. (1977). Information Theory with Applications. McGraw-Hill

Smith, B. L. & Oswald, R. K. (2003). Meeting Real–Time Traffic Flow Forecasting Requirements with Imprecise Computations. Computer-Aided Civil and Infrastructure Engineering, 2003, Vol. 18, (3), pp. 201-213

Sorjamaa, A., Hao, J., Reyhani, N. & Ji, Y. (2007). Lendasse. Methodology for long-term prediction of time series. Neurocomputing, Vol. 70 (16-18), pp. 2861–2869

Štěpnička, M., Cortez, P., Donate, J. P. & Štěpničková L. (2013). Forecasting seasonal time series with computational intelligence: On recent methods and the potential of their combinations. Expert Systems with Applications, Vol. 40 (6), pp. 1981–1992

Stogbauer, H., Kraskov, A., Astakhov, S.A. & Grassberger P. (2004). Least-dependent-component analysis based on mutual information. Physical Review E, Vol. 70, pp. 66-123

Sun, W., Wang, J., Fang., Y. (2013) Consistent Selection of Tuning Parameters via Variable Selection Stability. Journal of Machine Learning Research, 14, 3419-3440

Suykens, J. A., Vandewalle, J. (1999). Least squares support vector machine classifiers. Neural Processing Letters, Vol. 9 (3), pp. 293–300

Thompstone, R. M., Hipel, K. W. & McLeod, A.I. (1985). Forecasting quarter-monthly riverflow, Water Resources Bulletin, Vol. 21 (5), pp.731-741

Thrun, S., Burgard, W., Fox, D. (2005). Probabilistic robotics, Intelligent robotics and autonomous agents series. Mit Press 2015

Wang L., Liu X., Pedrycz W., Shao Y. (2014). Determination of temporal information granules to improve forecasting in fuzzy time series. Expert Systems with Applications, 41 (6), pp. 3134–3142

Wang, L., Zeng, Y., Chen, T. (2015) Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. Expert Systems with Applications, 42 (2), pp. 855–863

Weigend, A. S. & Gershenfeld, N.A. (1993), Time Series Prediction: Forecasting the Future and Understanding the Past. ISBN-10: 0201626020

Wu, L., Cheng, H., Su, Y. & Feng, H. (2003). Mathematical model for on-line prediction of bottom and hearth of blast furnace by particular solution boundary element method. Applied Thermal Engineering. Vol. 23 (16), pp. 2079–2087

Wu S. F. & Lee S. J. (2015). Employing local modeling in machine learning based methods for time-series prediction. Expert Systems with Applications. Vol. 42 (1), pp. 341–354

Xionga, T., Baob Y., Hub, Z. , Raymond Chiongc (2015). Forecasting interval time series using a fully complex-valued RBF neural network with DPSO and PSO algorithms. Information Sciences, Vol. 305, , 77–92

Yang, Z., You, W. Ji, G. (2011), Using partial least squares and support vector machines for bankruptcy prediction, Expert system with applications, Vol. 38, pp. 8336-8342

Yu, T.H.-K., Huarng, K.-H. (2010). A neural network-based fuzzy time series model to improve forecasting. Expert Systems with Applications, 37 (4), pp. 3366–3372

Venables, W. N. & Ripley, B. D. (2002). Modern Applied Statistics with S. Fourth Edition. Springer-Verlag

Zatlavi, L., Kenett, D. Y., Ben-Jacob, E. (2014). The design and performance of the adaptive stock market index. Algorithmic Finance, Vol. 3 (3-4), pp. 189-207

Zhang, G., Patuwo, B. E. and Hu, M. Y. (1998). Forecasting with artificial neural networks: the state of the art. International Journal of Forecasting, Vol. 14, pp. 35-62

Zhou, G., Guo, B., Gao, X., Zhao, D., & Yan, Y. (2014). Research on the System Reliability Modeling Based on Markov Process and Reliability Block Diagram. Advances in Intelligent Systems and Computing Vol. 298, pp 545-553

Zolhavarieh, S., Aghabozorgi, S., and Teh, Y. (2014). A Review of Subsequence Time Series Clustering. The Scientific World Journal. Vol. 2014 (2014), Article ID 312521, http://dx.doi.org/10.1155/2014/312521