kNN Ensembles with Penalized DTW for Multivariate Time Series Imputation

Stefan Oehmcke*†,Oliver Zielinski†,Oliver Kramer*

Computational Intelligence Group, Department of Computing Science*
Institute for Chemistry and Biology of the Marine Environment†
University Oldenburg, Germany
Email: {stefan.oehmcke, oliver.zielinski, oliver.kramer}@uni-oldenburg.de

***Abstract*—The imputation of partially missing multivariate time series data is critical for its correct analysis. The biggest problems in time series data are consecutively missing values that would result in serious information loss if simply dropped from the dataset. To address this problem, we adapt the k-Nearest Neighbors algorithm in a novel way for multivariate time series imputation. The algorithm employs Dynamic Time Warping as distance metric instead of point-wise distance measurements. We preprocess the data with linear interpolation to create complete windows for Dynamic Time Warping. The algorithm derives global distance weights from the correlation between features and consecutively missing values are penalized by individual distance weights to reduce error transfer from linear interpolation. Finally, efficient ensemble methods improve the accuracy. Experimental results show accurate imputations on datasets with a high correlation between features. Further, our algorithm shows better results with consecutively missing values than state-of-the-art algorithms.**

I. Introduction

The analysis of multivariate time series data is used for many applications in knowledge discovery. In most cases, these data consist of multiple concurrent measuring sensors over a certain amount of time - for example, a sensor platform in the ocean that provides data such as temperature, salinity, or wind speed for a scientific analysis. This analysis contributes to a greater understanding of ocean processes. In general, more complex analyses are now possible, because the storage and computational power is cheaply available.

Most methods for analysis for time series data require complete datasets, but data can be missing for various reasons. For example, a sensor could fail and the reason for failure may not depend on the process observed. If the missing data are dropped from the dataset, an analysis can become biased, less representative, or not possible at all. In contrast, if imputation algorithms fill the missing data with reasonable values, these effects diminish. For time series data imputation, algorithms such as linear or spline interpolation are commonly used [1].

However, existing imputation algorithms have various disadvantages. They often need heavily preprocessed data such as filters or some kind of smoothing. These preprocessings make assumptions about the dataset that require domain knowledge and could introduce bias. Multivariate data are often considered in a univariate way and thus take no advantage of feature correlations. Some algorithms, such as ARIMA [2], are not computationally feasible for huge datasets or high dimensionality. Moreover, standard algorithms often ignore multiple missing values at one point in time. Also, algorithms often only consider single missing values, while in reality sensors fail for a longer duration. The loss of information is small if one point is missing, but increases with consecutively missing data.

In this paper, we introduce an accurate and efficient algorithm to impute missing intervals in multivariate time series data. By combining the weighted k-Nearest Neighbors algorithm (`kNN`) with Dynamic Time Warping (`DTW`), our algorithm compares multiple points in time simultaneously. This is important for time correlated data because it enables our algorithm to learn with value movements. Furthermore, our algorithm computes global weights for every feature by considering the correlation coefficient of features. A novel penalty function calculates individual weights per missing interval by penalizing consecutively missing values. Finally, our algorithm creates an ensemble of imputation models with diversity methods such as bagging or by varying the penalty weights.

For evaluation, different degrees of missing data as well as different minimal missing data intervals are tested on various datasets. One dataset is taken from the Time Series Station (TSS) at Spiekeroog in Germany [3]–[5]. We use 15 more datasets from the UCR time series data mining archive by Keogh and Folias [6].

This paper is organized as follows. First, we discuss the related work in Section II. Next, we create a better formal understanding of missing data and its underlying mechanism in Section III. Section IV presents existing imputation algorithms such as linear interpolation and `kNN`. How to compare two time series with `DTW` is described in Section V. In Section VI we present our `DTWkNN` ensemble algorithm, which we evaluate experimentally in Section VII. Our conclusion can be found in Section VIII.

II. Related Work

In machine learning the imputation of values has mostly been considered for data without time correlation [7]–[9]. As a result, consecutively missing values also have not been analyzed. In general, the `kNN` algorithm has proven to be a robust imputation algorithm even if a larger part of data is missing in comparison to other algorithm such as CN2 and C4.5 [10].

Troyanskaya et al. [11] have used a time series representation of DNA data for imputation of missing parts. They have compared `kNN`, single value decomposition, and row average imputation. `kNN` has produced the best results. However, the authors have not tested with consecutively missing values or multivariate data, because of the application domain of DNA data.

Hsu et al. [12] have successfully combined `kNN` and `DTW` for DNA data. They have trained their model with complete data and interpolate single missing values. A focus of their work has been on comparing different variants of `DTW` for better prediction and computation performance. Again, consecutively missing values or multivariate data have not been considered.

A solution for the special case of lagged correlations in multivariate time series data has been proposed by Rahman et al. [13]. They have combined an extension of `kNN` that weights features based on their cross-correlation with missing values and a Fourier transformation to overcome multiple reasons for missing data. Consecutively missing values have not been addressed directly, but the authors state that the negative effect is stronger on the Fourier transformation results.

In the field of epistatic miniarray profiling, Pan et al. [14] have proposed an ensemble approach called EMDI. They combined multiple imputations from local and global algorithms such as `kNN`, local least squares imputation or matrix completion to one ensemble value. Their ensemble has calculated the model weights for the aggregation w.r.t. their individual amount of diversity in the ensemble. Nevertheless, they have only tested for univariate data and did not analyze effects of consecutively missing values.

Alippi et al. [15] have employed multiple linear and non-linear algorithms for on-line reconstruction of missing data. They have designed the algorithm for features with temporal or spatial redundancies. The authors have created a dependency graph that models the redundancy of features to simplify otherwise complex models by learning only with relevant features. Because of its focus on on-line datasets, only single missing values had been of interest.

III. Missing Data Theory

It is important to formalize multivariate missing data and how data can go missing for a better understanding of the imputation problem. The following definition for multivariate time series imputation problems is an adaption of the survey imputation definitions by Schafer [16]. A multivariate time series is represented as $n \times d$ matrix X with pattern $\boldsymbol{x}_t = (x_1, \ldots, x_d) \in \mathbb{R}^d$ as a d-dimensional feature vector at time $t \in \{1, \ldots, n\}$. To distinguish between observed and missing data there is a $n \times d$ missing indicator matrix M with $\boldsymbol{m}_t = (m_1, \ldots, m_d) \in \{0, 1\}^d$, whereby 1 indicates missing data and 0 shows that the value is available.

With help the of an indexing operator $\ominus$, all features of a pattern $\boldsymbol{x}_t$ marked with 1 in $\boldsymbol{m}_t$ can be extracted:

$$\boldsymbol{x}_t \ominus \boldsymbol{m}_t = (x_j : \forall j \in \{1, \ldots, d\} | m_j = 1). \qquad (1)$$

Now, the data can be separated into two sets: a set of completely observed features $X^{obs} = \{\boldsymbol{x}_1 \ominus \neg\boldsymbol{m}_1, \ldots, \boldsymbol{x}_n \ominus \neg\boldsymbol{m}_n\}$ and a set of missing features $X^{mis} = \{x_1 \ominus \boldsymbol{m}_1, \ldots, \boldsymbol{x}_n \ominus \boldsymbol{m}_n\}$.

The goal of an approximating model $f : X^{obs} \rightarrow X^{mis}$ is to minimize the sum of errors for a given vector $\boldsymbol{z}_t$ from X^{obs} in regard to the missing vector $\boldsymbol{y}_t$ from X^{mis}:

$$\min \sum_{t=1}^{n} \mathrm{Err}(\boldsymbol{y}_t, f(\boldsymbol{z}_t)). \qquad (2)$$

Whereby vector $\boldsymbol{y}_t$ would be known for evaluation. This is similar to the regression problem goal [17], but for each unique combination of $\boldsymbol{m}$ there exists a different regression problem, which reduces the available data for model training.

The occurrence of missing values depends on a process, which is usually hidden [1]. This process can be described as probability for the missing indicator M with the complete set X^{obs} and the missing set X^{mis} given. If the probability does not depend on the dataset X, but on an independent set of parameters ξ, it is called missing completely at random (MCAR):

$$\Pr(M | X^{obs}, X^{mis}, \xi) = \Pr(M | \xi). \qquad (3)$$

Furthermore, values are missing at random (MAR), when the missing indicator M depends on the complete set:

$$\Pr(M | X^{obs}, X^{mis}, \xi) = \Pr(M | X^{obs}, \xi). \qquad (4)$$

For example, a measurement x_1 is only taken if a certain threshold of value x_2 is passed: $\Pr(x_1 | x_2)$. If the missing indicator M is depending on both sets, the process is missing not at random (MNAR). In real datasets the missing process is rarely known and most algorithms assume MCAR or MAR.

We assume the MCAR mechanism in this paper. The set ξ contains two variables: the degree of missing data and the minimal interval of missing data. Here, the degree of missing data $r_d \in [0, 1]$ explains the ratio of missing values w.r.t. the total number $n \cdot d$ of feature values. Further, the minimal interval of missing data $r_i \in \mathbb{N}$ defines the least number of consecutively missing values (gaps). Normally, long missing intervals only occur with a high degree of missing values. More control over missing intervals is given with the second variable r_i.

IV. Imputation Algorithms

There are numerous algorithms for imputation [1]. One standard algorithm replaces missing features by their average values, which usually introduces bias to the dataset. Another easy-to-use algorithm is linear interpolation, which draws a straight line between start and end point of a gap. Linear interpolation performs well for short gaps in the data depending on its time resolution. With larger gaps the imputation is getting inaccurate. However, these methods are only univariate and ignore other features of the dataset. We define row i and column j positions for some matrix A as $A_{(i,j)}$.

The local instance based kNN algorithm fills the missing values with the average of similar patterns by comparing with the features that are not missing. For an unseen pattern $\boldsymbol{x}_t$, kNN imputation predicts the value x_j with $j \in \{1, \dots, d\}$ by averaging the k-nearest patterns [8], [17]:

$$f_{\mathrm{kNN}}(\boldsymbol{x}_t, \boldsymbol{m}_t, j) = \frac{1}{\mathrm{k}} \sum_{\boldsymbol{x}_{t'} \in \mathcal{N}_{\mathrm{k}}(\boldsymbol{x}_t, \boldsymbol{m}_t, j)} (\boldsymbol{x}_{t'} \ominus \boldsymbol{m}_t)_j. \quad (5)$$

The function $\mathcal{N}_{\mathrm{k}}(\boldsymbol{x}')$ returns the k-nearest patterns (k-arg min) from a set $L \subset X$ that have the same complete features ($\neg \boldsymbol{m}_{t^*} \rightarrow \neg \boldsymbol{m}_t$) plus the j-th feature available ($M_{(t^*,j)} = 0$):

$$\mathcal{N}_{\mathrm{k}}(\boldsymbol{x}_t, \boldsymbol{m}_t, j) = \text{k-arg}\min_{\boldsymbol{x}_{t'} \in L} \Delta(\boldsymbol{x}_t \ominus \neg \boldsymbol{m}_t, \boldsymbol{x}_{t'} \ominus \neg \boldsymbol{m}_t) \quad (6)$$

$$\text{with } L = \{\boldsymbol{x}_{t^*} : \forall t^* \in \{1, \dots, n\} | \neg \boldsymbol{m}_{t^*} \rightarrow \neg \boldsymbol{m}_t \wedge M_{(t^*,j)} = 0\} \quad (7)$$

kNN requires a distance measure Δ to get the set of k-nearest patterns. Often, the Euclidean distance in $\mathbb{R}^d$ is used as Δ. Only patterns with compatible missing vectors can be used for Euclidean distance calculation, because most metrics can not handle missing values. Normally, this algorithm does not consider the order of any time $t \in \{1, \dots, n\}$, but only the feature space without time component.

Theoretically, every machine learning algorithm for regression problems can perform imputation tasks. To achieve this, we need to train a new model for every unique missing combination of features [8]. In this paper, we used Random Forest [18] as state-of-the-art comparison to our algorithm, because its results are good even without extensive optimization and it is also an ensemble algorithm.

V. Dynamic Time Warping

First used in speech recognition [19] and then adapted for numerous data mining tasks, DTW is commonly used for all kinds of time series comparisons. Given two sequence patterns $\boldsymbol{s} = (s_1, \dots, s_\delta)^T$ and $\boldsymbol{s}' = (s'_1, \dots, s'_{\delta'})^T$ DTW calculates the optimal alignment between them. Figure 1 shows an exemplary alignment. Normally, a comparison is only made at the same time, e.g., with Euclidean distance $\sqrt{\sum_{i=1}^{\delta}(s_i - s'_i)^2}$. This comparison requires the sequence widths δ and δ' to be equal. In comparison, DTW finds a

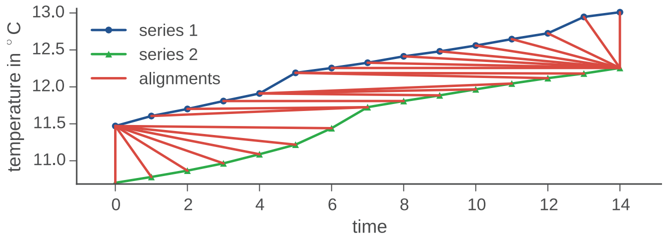

Figure 1. Example alignment of two temperature time series.

flexible alignment by creating a "warping" path between sequences and DTW allows them to have unequal widths.

Algorithm 1 shows how we calculate the minimal path distance. Initially, in Line 1 and 2 all entries of matrix D are set to infinity, except the first, which we set to zero. The matrix D contains the sums of minimal path distances with the minimal complete path distance at $D_{(|\boldsymbol{x}|,|\boldsymbol{x}'|)}$. This minimal path distance is the sum of the previous minimal path distance p, see Line 6, added to the current distance d. For the pointwise distance, often the Euclidean distance is used as distance measurement Δ. It is generally assumed that DTW is not computationally worthwhile, but this is actually not true. According to Ratanamahatana and Keogh [20] the runtime is closer to $O(\delta)$ than $O(\delta^2)$, if the algorithm uses lower bounding.

Algorithm 1: DTW-algorithm [19]

```
  input: $\boldsymbol{s}, \boldsymbol{s}'$, distance metric $\Delta$
1 $D \leftarrow$ create $(|\boldsymbol{s}|+1) \times (|\boldsymbol{s}'|+1)$ matrix full of $\infty$
2 $D_{(0,0)} \leftarrow 0$
3 for $l \leftarrow 1$ to $|\boldsymbol{s}|+1$ do
4     for $j \leftarrow 1$ to $|\boldsymbol{s}'|+1$ do
          // calculate distance and minimal path
5         $d \leftarrow \Delta(s_{l-1}, s'_{j-1})$
6         $p \leftarrow \min(D_{(l-1,j-1)}, D_{(l,j-1)}, D_{(l-1,j)})$
7         $D_{(l,j)} \leftarrow d + p$
8 return $D_{(|\boldsymbol{s}|,|\boldsymbol{s}'|)}$
```

VI. DTWkNN Ensemble

The initial idea of our algorithm is to combine kNN-imputation with DTW as distance measure Δ to utilize the correlation to time of datasets. This creates training problems with multivariate data that we solve with a pre-processing step, normalizing distances, applying correlation weighting and penalizing gaps. We call this algorithm DTWkNN. To further improve the imputation results, we employ an ensemble of DTWkNN models with the classical diversity method bagging and by varying the penalties for each model.

At first, for every time $t \in \{1, \dots, n\}$ a $\delta \times d$ feature window matrix $Z_t^\delta = [\boldsymbol{x}_{t-\delta}, \dots, \boldsymbol{x}_t]$ with width δ is created. Then, we calculate the DTW-distances between training and test set and use the average of the k nearest neighbors for imputation. This can easily be done for univariate time series data [12]. Even if some data are missing from the series, the windows can still be compared since `DTW` allows for different window widths.

However, a comparison is difficult if the features of a window are missing at different times. For example, a window Z_t^δ and its associated missing matrix $[\boldsymbol{m}_{t-\delta}, \ldots, \boldsymbol{m}_t] = [(0,1)_{t-2}, (1,0)_{t-1}, (0,0)_t]$ with width $\delta = 3$ and dimensionality $d = 2$ at time t are given. Because of the missing features, we need to either remove this window or interpolate it. We decided to use linear interpolation, because in a worst case scenario every time a pattern has a missing feature and by simply removing all these patterns, a comparison would not be possible. This also leads to windows that have always the same length, which can be used to speed up the comparisons [20]. It is important to note that this linear interpolation step is only applied to the feature windows and not to the labels, because this would result in overfitting to the linear interpolation.

The `kNN` algorithm performs depending on the weighting of features and the weighting of feature windows for the `DTWkNN` algorithm, respectively. For multivariate labels an individual weighting of feature windows per label is important. Because of this, we calculate the one-dimensional `DTW`-distance of two times t and t' for each feature j and then aggregate them to one distance measurement:

$$\sqrt{\sum_{j=1}^{d} \left(\mathtt{DTW}\left(Z_{(t,j)}^\delta, Z_{(t',j)}^\delta \right) \right)^2} \tag{8}$$

with $Z_{(t,j)}^\delta$ being the j-th feature of window matrix Z_t^δ. As a side effect the computation time decreases, because the equation calculates paths for d windows with width δ^2 instead of one window with width $\delta^{2 \cdot d}$. For example, if $d = 14$ and $\delta = 10$ only 1400 pointwise comparisons are required instead of 10^{28}.

By using `DTW` with preprocessed windows we also solve the problem of small training sets that occur with many unique missing data vectors $\boldsymbol{m}$. Before, only the subset of X could be used for training that had an missing indicator vector equal to the combined missing indicator vectors $\boldsymbol{m}_{t-\delta} \vee \ldots \vee \boldsymbol{m}_t$ of the to be compared window Z_t^δ. Further, this missing indicator vector had to include the currently sought label, because for training the label must be available. Now, because the feature windows contain no missing data, we can use all features for training. This is positive, because the imputations get more accurate when the model represents more of the whole dataset.

A. Normalized `DTW`-Distance

Features often have different scales and densities, which creates distortions in distance spaces with more than one feature. For instance, the TSS dataset contains the wind direction that scales between zero and 360 degrees and water temperature that scales between zero and 15°C. Without normalization, a difference in wind direction of 20° is huge in the calculated distance, even if the temperature does not change at all. Therefore, an imputation model created without normalization can be unreliable, if the features have large differences in scale and density.

A normalization solves this problem by dividing the distance of one feature by the standard deviation of this feature. The standard deviation vector $\sigma = (\sigma_1, \ldots, \sigma_d)$ is computed from the training set. If this is not possible, because too many values are missing or there is no deviation, we set the value to 1. This alters Equation 8 to:

$$\sqrt{\sum_{j=1}^{d} \left(\frac{\mathtt{DTW}\left(Z_{(t,j)}^\delta, Z_{(t',j)}^\delta \right)}{\sigma_j} \right)^2}. \tag{9}$$

B. Weighting Of Multivariate Outputs With Correlation

We normalize distances to create a uniform weighting for features, but not all features are equally important for label prediction. As indicator of feature importance we use the Pearson product-moment correlation coefficient [21], because a high correlation hints at a connection between two features. For example, water temperature and pressure are highly correlated, because with higher temperature the molecules in the water move faster, hence increasing the pressure. If we apply weights to the features according to their correlation with the label the accuracy of imputations should increase. According to Rahman et al. [13], using correlations is especially suitable if the MAR or MCAR mechanism applies.

The normalized Pearson product-moment covariance matrix C scales between minus and plus one, is symmetric, and $C_{(l,j)}$ describes the linear correlation between feature l and j. We can apply the weighting easily to our distances, because we divide the distance into a vector per feature as in Equation 8. Our final distance measurement Σ`DTW` has an additional parameter j' to represent the current label feature for the weight matrix W:

$$\Sigma\mathtt{DTW}(t, t', j') := \sqrt{\sum_{j=1}^{d} \left(\frac{\mathtt{DTW}\left(Z_{(t,j)}^\delta, Z_{(t',j)}^\delta \right)}{\sigma_j} \cdot W_{(t,j,j')} \right)^2}. \tag{10}$$

To create this $t \times d \times d$ weighting tensor W, we set:

$$W_{(t,j,j')} = |C_{(j,j')}|^\alpha. \tag{11}$$

We use the absolute values of the global correlation coefficient matrix C, because we want to emphasize correlation to other features regardless if they are positive or negative. In this context a global matrix C means that C is calculated for the entire dataset. This tensor W is 3-dimensional: the first dimension for time, the second stands for the feature in the sum, and the third represents the label feature. The exponent α is used to stretch and compress the weights. The higher the value of α, the more the weights are stretched. Finding the optimal value for α could be done with Cross-Validation or Gridsearch.

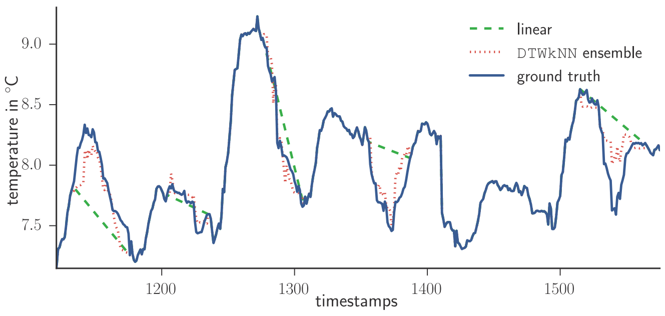

Figure 2. Huge gaps in data can result in poor imputes. The linear interpolation is far off the ground truth, but our method is not.

C. *Penalizing Of Missing Interval Interpolation*

Using interpolation to create feature windows introduces the same errors to these windows that are already present in the applied interpolation. Because of linear interpolation, bigger gaps in the data could be badly represented by the feature windows, as seen in Figure 2. The result will be wrongly matched feature windows by DTW, which leads to imputations that are biased towards the interpolated values.

We introduce a $n \times d$ penalty weight matrix P for the test dataset to counter the effect of badly shaped windows. A row entry will be written as $\boldsymbol{p}_t$ with $t \in \{1, \dots, n\}$. Algorithm 2 shows how the matrix P is calculated in detail. The more values have to be interpolated consecutively, the higher the penalty. To achieve this, we compute the width of the data gap and save it in gap width i at Line 6. Then, at Line 8, we assign the penalty $\frac{1}{i}$ to the weights of the gap. Additionally, at Line 12 and 13, we adapt the penalty weight according to the feature windows size.

Our weight tensor W from Equation 11 is updated to:

$$W_{(t,j,j')} = |C_{(j,j')}|^{\alpha} \cdot P^{\gamma}_{(t,j)} \tag{12}$$

In contrast to the globally used correlation weight C, the penalty weight is individual for every pattern $\boldsymbol{x}_t$ with $t \in \{t, \dots, n\}$. The exponent γ can be used to scale the penalty weight. If γ is a vector with length d, scaling per feature becomes possible.

D. *Efficient Ensemble Building*

Finding the global parameter settings for any model can be exhaustive. A setting might work well for one, but not for all dataset. An ensemble model F suffers less from this problem, because it aggregates l model outputs into one:

$$F(\boldsymbol{x}) = \frac{1}{l} \sum_{i=1}^{l} f_i(\boldsymbol{x}). \tag{13}$$

The quality of imputations for an ensemble model depends on the diversity between the l individual models [22]. We build a diverse ensemble with different parameters settings for each individual model by manipulating the penalty weight exponent and the size k of the neighborhood. The penalty weight exponent γ is drawn from a

Algorithm 2: Creating penalty weight matrix P

```
input: missing indicator matrix $M$, window width $\delta$,
       number of pattern $n$, number of features $d$
// invert $M$: missing$= 0$ and $not$-missing$= 1$
1  $P \leftarrow \sim M$
2  for $j \leftarrow 1$ to $d$ do
3      $i \leftarrow 0$                 // gap width $i$
4      for $t \leftarrow 0$ to $n$ do
           // increasing $i$ if value is missing
5          if $M_{t,j}$ then
6              $i \leftarrow i + 1$
           // apply penalty to gap
7          else if $i > 0$ then
8              $P_{(t-i,j)}, \dots, P_{(t,j)} \leftarrow \frac{1}{i}$
9              $i \leftarrow 0$
10 if $c > 0$ then   // last gap could be open-ended
11     $P_{(n-i,j)}, \dots, P_{(n,j)} \leftarrow \frac{1}{i}$
   // adapt penalty for each feature window
12 for $t \leftarrow 1$ to $n$  do
13     $\boldsymbol{p}_t \leftarrow \frac{\sum_{t'=t-\delta}^{t} \boldsymbol{p}_{t'}}{\delta}$
14 return $P$
```

normal distribution $\mathcal{N}(mean, std)$ for each ensemble member. Our algorithms draws its neighborhood size uniformly random between a start and end value (unif(start,end)). Additionally, we utilize bootstrap aggregating (bagging), a classic ensemble diversity method by Breiman [23]. With bagging a subset of the training set is drawn with replacement for every individual model as new training set. The amount of patterns n_F for training subsets depend on the sample rate s_r and the total number n of patterns in the dataset: $n_F = s_r \cdot n$.

While the accuracy of imputations increases, the runtime of this ensemble algorithm is not significantly higher than a single DTWkNN run. To achieve this, we first compute the normalized $m \times n \times d$ distance matrix with m the number of incomplete pattern, n the number of all patterns, and feature number d. We used brute-force for this, because of bagging, all distances would need to be calculated anyway. A distance matrix is given to each individual model, which can apply their weight matrix and choose their neighbors. Multiplying the weight matrix is a standard matrix operation and thus fast. Without separation of features and distance measurement in ΣDTW, we would need to compute an individual distance matrix for every ensemble member.

VII. Experimental Analysis

To validate our algorithm, we compared single and ensemble DTWkNN with linear interpolation, Random Forest, and linear interpolation preprocessed Random Forest (RF-linear). We choose linear interpolation to assure that our algorithm is in most cases better than its preprocessing.

Table I
Accumulated best R^2 scores for tested interval and missing rate combinations over all datasets.

intervals r_i	1				5				15				30				45				
missing rates r_d	.1	.2	.3	.4	.1	.2	.3	.4	.1	.2	.3	.4	.1	.2	.3	.4	.1	.2	.3	.4	$\sum$
fill with average	0	0	0	0	1	1	1	1	1	2	1	2	1	1	1	1	2	2	2	3	23
linear	4	3	3	2	1	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	17
Random Forest	5	3	3	1	5	3	3	1	5	4	2	1	7	4	2	1	8	6	2	1	67
RF-linear	1	2	3	5	3	3	4	5	4	2	1	2	1	2	2	1	1	0	2	3	47
DTWkNN	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3
DTWkNN ensemble	5	8	7	7	6	7	7	8	6	8	12	11	6	9	11	13	5	8	10	9	**163**

Also, Random Forest is a good choice, because it is a state-of-the-art machine learning algorithm with good out-of-the-box performance. To prove that the linear preprocessing alone does not solve the imputation problem, we also employ RF-linear.

In multivariate imputation we need a normalized validation score to enable aggregation of different scores per feature. Because of this, we employ the R^2 score, which scales in the interval $[-\infty, 1]$:

$$R^2(\boldsymbol{y}, \boldsymbol{x}, \bar{\boldsymbol{y}}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}{\sum_{i=1}^{n}(y_i - f(x_i))^2} \tag{14}$$

with the ground truth $\boldsymbol{y}$, the predicted values $f(\boldsymbol{x})$, and arithmetic mean $\bar{\boldsymbol{y}}$ of the tested dataset. A high score implies a well fitted model to the ground truth, but if it is 1, the model is potentially overfitted. The R^2 score is around zero, when $f(\boldsymbol{x})$ is equal to mean $\bar{\boldsymbol{y}}$, which means negative scores are worse than inserting mean values. We calculated the R^2 score for every feature and used the average over these d scores as dataset score.

We implemented `DTWkNN` ensemble in Python with a scikit-learn [24] interface. For the `DTW` distance measurement we created a Cython implementation. We also wrote a wrapper to adapt the scikit-learn version of Random Forest for imputation tasks. Additionally, we applied a linear interpolation preprocessing step to create the RF-linear models. This preprocessing step allow patterns to include interpolated values, but labels aren't interpolated, because this introduces to much bias.

A. Data & Design of Experiment

We selected 16 datasets for the experiment. The feature number d ranges from at least two to maximal 14 features. Our number of patterns per dataset ranges from 867 up to 2500. We took the TSS [5] dataset from 2015-03-26 22:10 until 2015-04-13 23:50, because this part included less sensor failures than others. Also, we smoothed the data with a ten-minute median filter to reduce noise and to further diminish missing data in the dataset. All other datasets are taken from the UCR time series data mining archive by Keogh and Folias [6]. Namely they are: synthetic, phone1, ballbeam, balloon, dryer2 sensor, foetal ECG, winding, glassfurnace, greatlakes, pgt50 alpha, pgt50 cdc15, pHdata, shuttle, water, and robot arm. The synthetic dataset was generated with very high time resolution, which is why only huge gaps in the data would have any impact. Because of this, we used every 50th pattern and thus reduced the data to 2000 patterns.

The missing rate r_d is tested with following set of values: $r_d \in \{.1, .2, .3, .4\}$. Higher missing rates are not considered, because if every second value is missing, information are so sparse that reacquiring the data might be a better choice. The minimal missing interval parameter r_i is set to: $r_i \in \{1, 5, 15, 30, 45\}$. This way we can test how the algorithms perform when at least 5% of the dataset are consecutively missing. We conducted 30 runs per combination to ensure generalizability. Each run we removed other values at random according to r_d and r_i from the dataset. The algorithms learn with the whole dataset and we optimized them beforehand. For the ensemble we adopt the optimized `DTWkNN` settings. The parameter ranges were extracted from preliminary studies as follows:

Model	Parameter	Value Ranges
Random Forest	num. estimators	200
	max. tree depth	1, 2, 5, 8, ∞
	max. features	sqrt, log2, all
`DTWkNN`	window width δ	1, 5, 10, 15
	penalty exp. γ	0, .0625, .125, .25, .5, 1
	covar. exp. α	0, .0625, .125, .25, .5, 1, 2
`DTWkNN` ensemble	num. estimators	50
	`kNN`'s k	unif(1,15)
	penalty std.	0, .03125, .0625, .125, .25, .5
	sample rate s_r	.125, .25, .5, .75, 1, 1.25, 1.5

B. Results of Experiment

In Table I we summed up the R^2 scores of all combinations for a better overview. There are in total 320 different combinations that come from 16 datasets, five missing interval and four missing rate settings. Whenever a model has the best mean minus standard deviation score for a combination it adds one to their counter. As expected, linear interpolation performs best with small missing intervals and later introduces too much bias. It is notable that Random Forest has most of its best cases with low missing rates. The linear preprocessing does only create best results in around 5% of all cases, because the preprocessed data introduced too much bias. With the penalty weight our algorithm counteracted this bias. In around 51% of all cases our `DTWkNN` ensemble scored best. The single `DTWkNN` model is in four cases better than

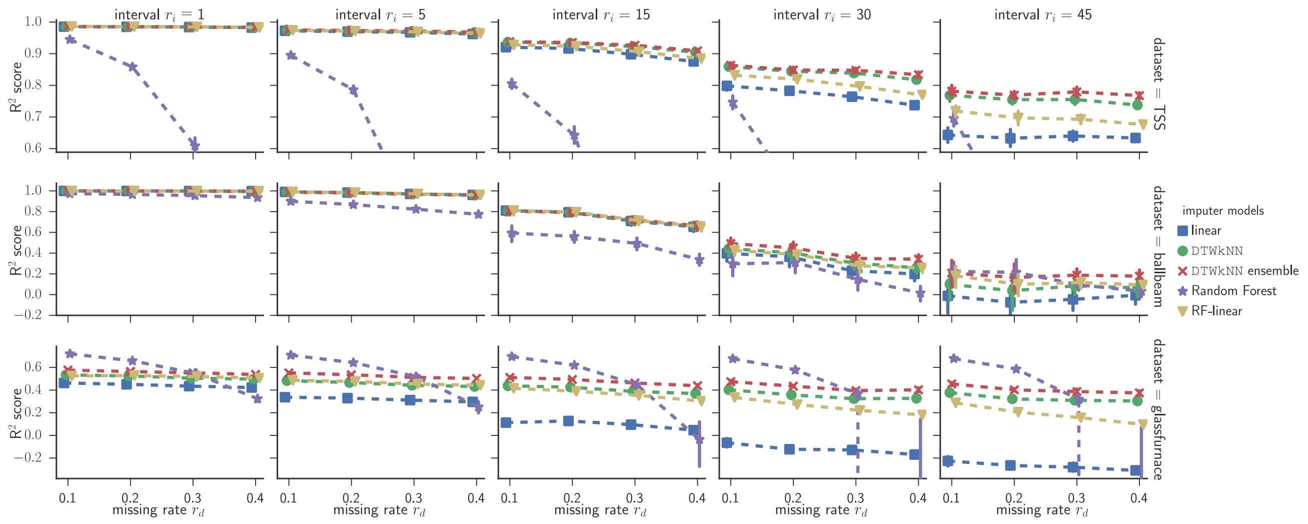

Figure 3. Performance on TSS, ballbeam, and glassfurnace dataset with different interval settings r_i. To improve visibility, results from different algorithms are slightly shifted on the x-axis.

the rest, which confirms that the ensemble imputation is better than a single one. The imputed values became more inaccurate than inserting the datasets mean for longer intervals in 25 cases .

Figure 3 shows results for dataset TSS, ballbeam, and glassfurnace over all tested intervals and missing rates. In general, the score for all imputation models starts around the same value for interval $r_i = 1$ and the models deviate with longer intervals. We see that imputation with linear interpolation delivers stable results across all tested parameters, but is low in score. Moreover, the single DTWkNN models are superior to linear interpolation and with increasing intervals this lead extends further. The same is true for the ensemble model of DTWkNN, it follows the same ups and downs as single DTWkNN, but with larger intervals it can retain a higher score. For Random Forest we observe that the model handles small missing rates well and sometimes produces better results than DTWkNN. With higher missing rates the score of Random Forest drops fast, because the dependency on compatible patterns limits the training set for the individual imputations. These compatible patterns become rare with high missing rates, which interferes with the training phase to the point where it is not possible to create good models. The Random Forest model with linear interpolation preprocessing does not have the problem of too few patterns for training. Nevertheless, the score is in most cases near to the linear interpolation score, which is not enough to surpass the other models.

We see in Table II scores for the datasets TSS, foetal ECG, greatlakes, and balloon. Only the best performing algorithms are shown. In the TSS dataset an interval of 15 and 45 translates to 2.5 and 7.5 hours, but our model is able to impute the values with high accuracy. The results for the foetal ECG dataset show that DTWkNN can perform well although the linear interpolation score is negative. Again, Random Forest starts with good performance in all datasets that diminishes with increasing missing rate. DTWkNN is outperformed for the balloon dataset. The reasons are probably the low dimensionality of this dataset and the poor time resolution, which prevents taking advantage of multivariate data or the time axis.

VIII. Conclusion

In this paper, we presented an imputation algorithm that is able to repair consecutively damaged multivariate time series data. Our algorithm creates an ensemble of models based on weighted kNN imputation with DTW as distance measurement with a linear interpolation preprocessing step. We penalize this preprocessing whenever values are consecutively missing. The ensemble ensures diversity among individual models with diversity methods such as varying penalty strength.

The experimental results on 16 datasets show that our algorithm outperforms standard and state-of-the-art methods in 51% of all cases. At higher missing rates, other algorithms such as Random Forest suffer from reduced training set availability while our DTWkNN ensemble can utilize the whole dataset. With larger minimal missing intervals, the introduced bias from preprocessing rises, but the feature window comparisons still yield reliable results because of our penalty function.

We designed our algorithm for highly correlated time series data. Thus, a certain redundancy of features has a positive effect on imputation quality when a higher missing rate is present. If the label of a pattern only correlates with other missing features from that pattern, the penalized distance matching can turn into random matching. Also, if the time resolution is too small, our algorithm can not create meaningful windows for DTW which can reduce accuracy.

Table II
R^2 scores of four datasets for interval lengths of 15 and 45. Highlighted in bold are the highest scores of a dataset for each combination of missing rate r_d and interval r_i.

Dataset	Imputation Model	$r_i =$	15				45			
		$r_d =$	0.1	0.2	0.2	0.4	0.1	0.2	0.3	0.4
TSS	Random Forest		0.81	0.64	0.11	<−1.00	0.70	0.35	−0.59	<−1.00
	RF-linear		0.93	0.92	0.91	0.88	0.72	0.70	0.69	0.68
	`DTWkNN` ensemble		**0.94**	**0.94**	**0.93**	**0.91**	**0.78**	**0.77**	**0.78**	**0.77**
foetal ECG	Random Forest		**0.91**	**0.87**	0.70	0.29	**0.89**	**0.81**	0.55	−0.19
	RF-linear		0.45	0.34	0.28	0.21	0.26	0.24	0.18	0.10
	`DTWkNN` ensemble		0.82	0.76	**0.72**	**0.62**	0.72	0.67	**0.59**	**0.62**
greatlakes	Random Forest		**0.77**	**0.75**	**0.68**	0.56	**0.73**	**0.67**	0.55	0.34
	RF-linear		0.66	0.66	0.64	0.61	0.49	0.51	0.46	0.43
	`DTWkNN` ensemble		0.72	0.71	**0.68**	**0.65**	0.64	0.62	**0.58**	**0.52**
balloon	Random Forest		**0.90**	**0.80**	**0.69**	**0.60**	**0.88**	**0.75**	**0.67**	**0.53**
	RF-linear		0.33	0.27	0.20	0.13	0.16	0.19	0.11	0.09
	`DTWkNN` ensemble		0.37	0.32	0.26	0.12	0.32	0.28	0.21	0.11

In future work, we will test other preprocessing steps such as splines or a Fourier transformation. In addition, we plan to integrate seasonality that has been detected into the penalty function and apply the penalty approach to other algorithms such as Random Forest.

Acknowledgments

We thank the ministry of science and culture of Lower Saxony for supporting us with the graduate school *Safe Automation of Maritime Systems (SAMS)*.

References

[1] M. Denk and M. Weber, "Avoid filling Swiss cheese with whipped cream: imputation techniques and evaluation procedures for cross-country time series," *IMF Working Papers*, pp. 1–27, 2011.

[2] R. Maronna, D. Martin, and V. Yohai, *Robust statistics*. John Wiley & Sons, Chichester. ISBN, 2006.

[3] R. Reuter, T. H. Badewien, A. Bartholomä, A. Braun, A. Lübben, and J. Rullkötter, "A hydrographic time series station in the Wadden Sea (southern North Sea)," *Ocean Dynamics*, vol. 59, no. 2, pp. 195–211, 2009.

[4] S. Garaba, T. Badewien, A. Braun, A.-C. Schulz, and O. Zielinski, "Using ocean colour remote sensing products to estimate turbidity at the Wadden Sea time series station Spiekeroog," *Journal of the European Optical Society-Rapid publications*, vol. 9, 2014.

[5] S. Oehmcke, O. Zielinski, and O. Kramer, "Event detection in marine time series data," in *KI 2015: Advances in Artificial Intelligence*. Springer, 2015, pp. 279–286.

[6] E. Keogh and T. Folias, "The UCR time series data mining archive," *Computer Science & Engineering Department, University of California, Riverside, CA. http://www. cs. ucr. edu/eamonn/TSDMA/index. html*, 2002.

[7] J. M. Jerez, I. Molina, P. J. García-Laencina, E. Alba, N. Ribelles, M. Martín, and L. Franco, "Missing data imputation using statistical and machine learning methods in a real breast cancer problem," *Artificial Intelligence in Medicine*, vol. 50, no. 2, pp. 105–115, 2010.

[8] S. García, J. Luengo, and F. Herrera, "Dealing with missing values," in *Data Preprocessing in Data Mining*, ser. Intelligent Systems Reference Library. Springer International Publishing, 2015, vol. 72, pp. 59–105.

[9] J. R. B. Jr., M. do Carmo Nicoletti, and L. Zhao, "Imputation of missing data supported by complete p-partite attribute-based decision graphs," in *International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, 2014, pp. 1100–1106.

[10] G. E. A. P. A. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied Artificial Intelligence*, vol. 17, no. 5-6, pp. 519–533, 2003.

[11] O. G. Troyanskaya, M. N. Cantor, G. Sherlock, P. O. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[12] H. Hsu, A. C. Yang, and M. Lu, "KNN-DTW based missing value imputation for microarray time series data," *Journal of Computers JCP*, vol. 6, no. 3, pp. 418–425, 2011.

[13] S. A. Rahman, Y. Huang, J. Claassen, and S. Kleinberg, "Imputation of missing values in time series with lagged correlations," in *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2014, Shenzhen, China, December 14, 2014*, 2014, pp. 753–762.

[14] L. Pan and J. Li, "K-nearest neighbor based missing data estimation algorithm in wireless sensor networks," *Wireless Sensor Network*, vol. 2, no. 2, pp. 115–122, 2010.

[15] C. Alippi, G. Boracchi, and M. Roveri, "On-line reconstruction of missing data in sensor/actuator networks by exploiting temporal and spatial redundancy," in *The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, June 10-15, 2012*, 2012, pp. 1–8.

[16] J. L. Schafer, *Analysis of incomplete multivariate data*. CRC press, 1997.

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. Springer, 2009, vol. 2, no. 1.

[18] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[19] J. B. Kruskal and M. Liberman, "The symmetric time-warping problem: from continuous to discrete," *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, pp. 125–161, 1983.

[20] C. A. Ratanamahatana and E. J. Keogh, "Three myths about dynamic time warping data mining," in *SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*. SIAM, 2005, pp. 506–510.

[21] K. Pearson, "Notes on the history of correlation," *Biometrika*, pp. 25–45, 1920.

[22] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*. Springer, 2000, pp. 1–15.

[23] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.