Classification of Sparse Time Series via Supervised Matrix Factorization

Josif Grabocka, Alexandros Nanopoulos, Lars Schmidt-Thieme
{josif, nanopoulos, schmidt-thieme}@ismll.uni-hildesheim.de
Information Systems and Machine Learning Lab, University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim, Germany

Abstract

Data sparsity is an emerging real-world problem observed in a various domains ranging from sensor networks to medical diagnosis. Consecutively, numerous machine learning methods were modeled to treat missing values. Nevertheless, sparsity, defined as missing segments, has not been thoroughly investigated in the context of time-series classification. We propose a novel principle for classifying time series, which in contrast to existing approaches, avoids reconstructing the missing segments in time series and operates solely on the observed ones. Based on the proposed principle, we develop a method that prevents adding noise that incurs during the reconstruction of the original time series. Our method adapts supervised matrix factorization by projecting time series in a latent space through stochastic learning. Furthermore the projected data is built in a supervised fashion via a logistic regression. Abundant experiments on a large collection of 37 data sets demonstrate the superiority of our method, which in the majority of cases outperforms a set of baselines that do not follow our proposed principle.

1 Introduction

Time-series classification has attracted considerable research interest during the recent decades. However the problem of classifying sparse time series has not been generally analyzed. In the context of our study sparsity is defined as the absence of time-series segments at random. The absence of segments is due to either improper recordings or deliberate avoidance of it. For instance in sensor networks, devices like wireless transmitters often experience operational failures that result in sparse signals. Recording might also be deliberately skipped, as in medical diagnosis, where high costs lead to partial (sparse) array of measurements over time (Rostamizadeh, Agarwal, and Bartlett 2011).

Dynamic time warping (DTW) is a distance measure that identifies time-shifted patterns among series. DTW is used for the nearest neighbor (NN) time-series classification (Berndt and Clifford 1996). It has been presented that the combination of DTW and NN achieved hard-to-beat results (Ding et al. 2008). Unfortunately DTW is not directly applicable to sparse time series, since the warping path procedure cannot operate with missing segments. An intuitive way to overcome this can be done in two steps: (i) Approximately *reconstructing* the full form of a sparse series, (ii) then apply NN classifier with DTW on the full form.

In this two-step approach, reconstruction (a.k.a. imputation) can be either intra-series (Opfer and Knott 2001) or collaborative (Marwala 2009). Intra-series approaches reconstruct the missing segment via analyzing the patterns in the other present segments of the series. Such strategies, as will be shown, perform non-optimally because of the unpredictable patterns observed on typical time series data sets. Collaborative techniques advocate sharing the structure information with other series instances in the same data set. Specifically, missing segments are reconstructed by mimicking pattern structures from other time series, during the same time interval as the missing segment. Yet, reconstruction techniques may produce noisy approximations of the original series segments. As our results will demonstrate, classification performance deteriorates for this reason.

Motivated by the aforementioned drawbacks of the reconstruction techniques, we propose a novel principle to handle sparsity (Section 3). Instead of focusing on improving reconstruction, our principle aims at classifying sparse time series using only observed segments, without any need for reconstructing the missing segments.

Based on this principle, we propose a new method (Section 4) that adapts supervised matrix factorization, which collaboratively extracts latent (hidden) features of time series by considering only observed values. An important aspect of matrix factorization, is that it creates a latent feature space where all time-series instances jointly share information. This latent space is built via a stochastic learning technique which is specifically designed for sparse data sets (Koren, Bell, and Volinsky 2009). Additionally, our method performs supervised factorization, which improves separability of latent data by enforcing a classification accuracy loss function (Menon and Elkan 2010). In our implementation we introduce an all-vs-one logistic regression loss function. Abundant experiments on a large collection of 37 data sets demonstrate the superiority of our method, which in the vast majority of cases outperforms a set of baselines that do not follow our proposed principle (Section 5).

2 Related Work

Time series classification has been highly elaborated by machine learning researchers mostly in the two recent decades. Various state-of-art classifiers have been adopted to the time series domain, ranging from neural networks (Kehagias and Petridis 1997), Bayesian networks (Pavlovic, Frey, and Huang 1999) and Support Vector Machines (SVM) (Eads et al. 2002). Distance based nearest neighbor empowered with a distance similarity technique called Dynamic Time Warping (DTW) overcomes the drawbacks of other classifiers by detecting shifts of patterns in time. Recent studies show that DTW is a hard-to-beat method in terms of classification accuracy (Ding et al. 2008). DTW computes the minimum warping distance between two series by constructing a matrix whose cells contain accumulated similarity-aligned path cost (Keogh and Pazzani 2000). Global constraints on the warping path can be applied to restrict the subset of visitable cells around the diagonal, called warping window (Keogh and Ratanamahatana 2005). As mentioned in Introduction, in contrast to our proposed method, DTW cannot directly handle sparsity.

Imputation is a long studied field in statistics and computer science involved with the filling/reconstruction of missing values of data set. Several imputation techniques have been developed, where one of the most popular is model based imputation, which offers a collaborative approach for imputing missing segments using information from other data instances (Marwala 2009). Spline interpolations, on the other hand, are used to complete missing curvature segments by preserving the continuity and smoothness of the curve (Opfer and Knott 2001). In comparison, our method avoids reconstruction at all.

Matrix Factorization (**MF**) exploits hidden features of a matrix by learning latent matrices whose dot product represent the original (Srebro, Rennie, and Jaakola 2005; Gemulla et al. 2011). Unification of different MF techniques has been formalized as minimizing a generalized Bregman divergence (Singh and Gordon 2008). Stochastic gradient descent learning has been successfully applied for unsupervised factorization of data sets with missing values (Koren, Bell, and Volinsky 2009). For instance, in recommender systems various MF strategies have been recently adopted to collaborative filtering of sparse user-item ratings with the aim of recommending unrated pairs (Rendle and Schmidt-Thieme 2008). Supervision, in the context of matrix factorization, enables the linear separation of classes in the projected data space (Menon and Elkan 2010). The objective function of the factorization has been shaped to enforce specific properties of the latent projected data, for example, geometric structure of affinity can be used to position neighbor instances closer in the projected space (Cai et al. 2011). MF has been also applied to the analysis of time series, for instance in music transcription (Smaragdis and Brown 2003), or EEG preprocessing (Rutkowski, Zdunek, and Cichocki 2007). Our study extends the prior works by adopting supervised factorization to sparse time series.

3 Proposed Principle and Method Outline

3.1 Proposed Principle

We define our data set to contain N time series over M time points. Such data set can be represented as a sparse matrix, denoted $\mathbf{X} \in (\mathbb{R} \cup \{\cdot\})^{N \times M}$, where $\{\cdot\}$ denotes the sparse (*null*) cells. Each row of $\mathbf{X}$ corresponds to one time series. The respective class labels can be defined alongside as $\mathbf{Y} \in \{0,1\}^{N \times L}$, where L is the count of distinct label values. In this matrix each class label is a row vector where all values are zero except the index of the class label value, which is set to one. Our proposed principle aims at classifying unlabeled time series *directly*, relying solely on observed time series matrix cells, denoted as $O = \{(i,j) \mid X_{i,j} \neq \{\cdot\}, 1 \leq i \leq N, 1 \leq j \leq M\}$.

3.2 Method Outline

Our proposed method is composed of two steps: (i) supervised projection to a latent space and (ii) classification in this latent space. During the first step, the data set is projected in a latent space of K dimensions using supervised MF. Supervised factorization has been shown to boost the accuracy of classifiers on the projected space (Menon and Elkan 2010). In our study, we apply a polynomial kernel Support Vector Machine (SVM) (Cortes and Vapnik 1995) in order to classify the projected data. Section 4 describes in details the technique employed to handle the supervised factorization.

4 Supervised Matrix Factorization (SMF)

Matrix factorization approximates the original data set by learning latent features matrices whose dot product can approximately reconstruct the original matrix as shown in Equation 1. In our problem description the time series matrix $\mathbf{X}$ are approximated by the dot product of two factor matrices $\mathbf{U} \in \mathbb{R}^{N \times K}$ and $\mathbf{V} \in \mathbb{R}^{M \times K}$, approximating only the observed cells O. Meanwhile K is the dimensionality of the latent space. Firstly, $\mathbf{U}$ is the latent representation of time series, where each row $\mathbf{U}_{\mathbf{i},:}$ defines the latent features of the original time series $\mathbf{X}_{\mathbf{i},:}$. Similarly $\mathbf{V}$ is the latent representation of time points.

$$\mathbf{X} \approx_O \mathbf{U}\mathbf{V}^T \tag{1}$$

Supervised factorization adds another objective, Equation 2, by conditioning the latent instances of $\mathbf{U}$ to be linearly separable w.r.t class labels $\mathbf{Y}$. Let us initially define a matrix $\mathbf{W} \in \mathbb{R}^{K \times L}$ to be a set of linear logistic regression weights, one column per class label. We mutually restrict $\mathbf{U}$ and $\mathbf{W}$ to be built such that the logistic value of their dot product can accurately predict the labels $\mathbf{Y}$, where σ is the cell-wise logistic function $\sigma(z) = \frac{1}{1+e^{-z}}$.

$$\mathbf{Y} \approx \sigma(\mathbf{U}\mathbf{W}^T) \tag{2}$$

The objective of our matrix factorization is to learn optimal latent representation matrices, denoted $\mathbf{U}^*$, $\mathbf{V}^*$, $\mathbf{W}^*$, which best approximate the matrices $\mathbf{X}$ and $\mathbf{Y}$ via a loss function optimization.

4.1 Supervised Matrix Factorization

The objective function to be minimized, shown in Equation 3, is composed of three loss terms. Firstly the reconstruction loss, denoted L_R, makes sure that the latent feature matrices $\mathbf{U}$ and $\mathbf{V}$ could reconstruct $\mathbf{X}$. Secondly the supervised classification accuracy loss term, denoted L_{CA}, enforces the latent time series representation $\mathbf{U}$ to be corrected such that a set of all-vs-one logistic regression weights $\mathbf{W}$ can maximize the classification accuracy on the data set. Finally the regularization term, denoted Reg, ensures that the latent matrices do not overfit. We introduce a parameter μ as a switch that controls the trade-off between the importance of reconstruction and the impact of classification accuracy.

$$\begin{aligned}(\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*) &= \underset{U,V,W}{\operatorname{argmin}}\ \mu L_R(\mathbf{X}, \mathbf{U}\mathbf{V}^{\mathbf{T}}) \\ &+ (1-\mu) L_{CA}(\mathbf{Y}, \sigma(\mathbf{U}\mathbf{W}^{\mathbf{T}})) \\ &+ Reg(\mathbf{U}, \mathbf{V}, \mathbf{W}) \end{aligned} \quad (3)$$

Reconstruction loss is defined in Equation 4 as the sum of square errors in predicting the observed cells of $\mathbf{X}$.

$$\begin{aligned} L_R(\mathbf{X}, \mathbf{U}\mathbf{V}^{\mathbf{T}}) &= \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_O^2 \\ &= \sum_{(i,j)\in O} \left(\mathbf{X}_{i,j} - \sum_{k=1}^{K} \mathbf{U}_{i,k} \mathbf{V}_{k,j}^T \right)^2 \end{aligned} \quad (4)$$

Meanwhile the classification accuracy loss, defined in Equation 5, is expressed as the cumulative linear logistic regression loss in classifying the training set of time series Y through the weights W.

$$\begin{aligned} L_{CA}(\mathbf{Y}, \sigma(\mathbf{U}\mathbf{W}^{\mathbf{T}})) = -\sum_{i=1}^{N}\sum_{l=1}^{L} \mathbf{Y}_{i,l} \ln \sigma \left(\sum_{k=1}^{K} \mathbf{U}_{i,k} \mathbf{W}_{k,l}^T \right) \\ +(1-\mathbf{Y}_{i,l}) \left(1 - \ln \sigma \left(\sum_{k=1}^{K} \mathbf{U}_{i,k} \mathbf{W}_{k,l}^T \right) \right) \end{aligned} \quad (5)$$

Equation 6 introduces the regularization terms, each defined with a squared Euclidean norm. This loss assures that the latent matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$ do not overfit.

$$Reg(\mathbf{U}, \mathbf{V}, \mathbf{W}) = \lambda_U \|\mathbf{U}\|^2 + \lambda_V \|\mathbf{V}\|^2 + \lambda_W \|\mathbf{W}\|^2 \quad (6)$$

4.2 Stochastic Learning

The objective function (3) contains only convex terms therefore gradient descent is an obvious choice for optimization. In sparse matrices *stochastic* gradient descent is preferred to the full gradient descent (Rendle and Schmidt-Thieme 2008), because it accelerates the convergence of the latent projection (Koren, Bell, and Volinsky 2009). Matrix cells are picked in random order and the partial error of a single cell is propagated backward to the latent matrices for correction. In order to repair the error of a single observed cell, the respective row and column of the latent matrices is corrected in the negative direction of the error. For instance, in order to repair the loss of cell $\mathbf{X_{i,j}}$ then row $\mathbf{U}_{\mathbf{i},:}$ and column $\mathbf{V}_{:,\mathbf{j}}^{\mathbf{T}}$ are corrected. A stepwise correction of a row requires the stepwise updates of all its column values, K columns for reconstruction loss or L columns for classification accuracy loss. In order to apply the gradient update of every cell value in the latent representations it is a prerequisite to initially compute the partial derivatives of the loss equations (4) and (5). The computed derivatives are shown in equation 7-10.

$$\frac{\partial L_{R_{i,j}}}{\partial U_{i,k}} = -2 \left(\mathbf{X}_{i,j} - \sum_{k=1}^{K} \mathbf{U}_{i,k} \cdot \mathbf{V}_{k,j}^T \right) \mathbf{V}_{k,j}^T \quad (7)$$

$$\frac{\partial L_{R_{i,j}}}{\partial V_{k,j}^T} = -2 \left(\mathbf{X}_{i,j} - \sum_{k=1}^{K} \mathbf{U}_{i,k} \cdot \mathbf{V}_{k,j}^T \right) \mathbf{U}_{i,k} \quad (8)$$

$$\frac{\partial L_{CA_{i,l}}}{\partial U_{i,k}} = -\left(\mathbf{Y}_{i,l} - \sigma \left(\sum_{k=1}^{K} \mathbf{U}_{i,k} \mathbf{W}_{k,l}^T \right) \right) \mathbf{W}_{k,l}^T \quad (9)$$

$$\frac{\partial L_{CA_{i,l}}}{\partial W_{k,l}^T} = -\left(\mathbf{Y}_{i,l} - \sigma \left(\sum_{k=1}^{K} \mathbf{U}_{i,k} \mathbf{W}_{k,l}^T \right) \right) \mathbf{U}_{i,k} \quad (10)$$

4.3 Algorithm

Algorithm 1 demonstrates a method that updates the latent matrices until the cumulative reconstruction and classification accuracy loss function cannot be further minimized. There are two major parts of the update mechanisms, the reconstruction updates and the classification accuracy updates. During each iterative epoch all observed cells of $\mathbf{X}$, denoted by O as in Section 3.1, are randomly taken into consideration. The loss error reconstructing each cell is used as a scale to correct back the latent matrices $\mathbf{U}$ and $\mathbf{V}^T$ via the partial derivatives. Updates are issued based on the reconstruction loss $L_{R_{i,j}}$ for a cell i, j with respect to all K values of the i-th row of $\mathbf{U}$ and j-th column of $\mathbf{V}^T$. Similar mechanism is followed for propagating backwards the error associated with the classification accuracy loss, with one difference, since label data are not sparse, we do not need stochastic selections of the cells to process. The learning rate parameter, denoted η, controls the trade-off between run time and convergence accuracy.

Once we have built the latent representation matrix $\mathbf{U}$, we can use it as the new data set for classification. Upon algorithm completion $\mathbf{U}$ has two main properties, it is both good estimate of the observed data segments and is also well set apart by the classifier loss, therefore it is suitable for being classified via an SVM classifier. It has been shown that combining supervised factorization and an additional classifier in the produced latent space, results in better classification accuracy compared to either unsupervised factorization or classification without additional use of classifiers in the latent space (Menon and Elkan 2010).

Algorithm 1 Stochastic Gradient Descent Learning

Input: Sparse $\mathbf{X}$
Output: $\mathbf{U}, \mathbf{V}, \mathbf{W}$

Initialize $\mathbf{U}, \mathbf{V}, \mathbf{W}$ uniformly random between $(-\epsilon, +\epsilon)$
$previousLoss \leftarrow MaxValue$
$currentLoss \leftarrow \left\{L_R(\mathbf{X}, \mathbf{U}\mathbf{V}^T) + L_{CA}(\mathbf{Y}, \sigma(\mathbf{U}\mathbf{W}^T))\right\}$
while $currentLoss < previousLoss$ **do**
  for $(i, j) \in O$ in random order **do**
    for $k = 1$ to K **do**
      $\mathbf{U}_{i,k} \leftarrow \mathbf{U}_{i,k} - \eta \cdot \frac{\partial L_{R_{i,j}}}{\partial U_{i,k}}$
      $\mathbf{V}^T_{k,j} \leftarrow \mathbf{V}^T_{k,j} - \eta \cdot \frac{\partial L_{R_{i,j}}}{\partial V^T_{k,j}}$
    end for
  end for
  for $i = 1$ to N **do**
    for $l = 1$ to L **do**
      for $k = 1$ to K **do**
        $\mathbf{U}_{i,k} \leftarrow \mathbf{U}_{i,k} - \eta \cdot \frac{\partial L_{CA_{i,l}}}{\partial U_{i,k}}$
        $\mathbf{W}^T_{k,l} \leftarrow \mathbf{W}^T_{k,l} - \eta \cdot \frac{\partial L_{CA_{i,l}}}{\partial W^T_{k,l}}$
      end for
    end for
  end for
  $previousLoss \leftarrow currentLoss$
  $currentLoss \leftarrow \left\{L_R(\mathbf{X}, \mathbf{U}\mathbf{V}^T) + L_{CA}(\mathbf{Y}, \sigma(\mathbf{U}\mathbf{W}^T))\right\}$
end while
return $\mathbf{U}, \mathbf{V}, \mathbf{W}$

5 Experiments

In order to compare the classification accuracy of our supervised matrix factorization (**SMF**) method we selected the following baselines which address the sparsity reconstruction/imputation based on different perspectives. All methods reconstruct a sparse time series as a prerequisite to NN plus DTW classifier.

- **Model Based Imputation (MBI)**: is a popular and effective collaborative approach for imputing missing data (Marwala 2009), (Raghunathan et al. 2001). In case a data set contains k features, then MBI builds k different classifiers, each trained to predict a different feature as target. Each classifier treats the observed values of the target feature as training set and the missing data as test set (Marwala 2009). Support Vector Regression was preferred due to the superior performance exhibited in our experiments compared to regression trees.[1] Comparing our method towards MBI will show whether avoiding reconstruction improves classification accuracy.
- **Matrix Factorization and Reconstruction (MFR)**: is another direct approach, which builds latent representation matrices similar to SMF without the classification accuracy loss term in the optimization loss function (Koren, Bell, and Volinsky 2009). Finally, the aim is to predict the missing values as dot product of the latent matrices. The relative performance of SMF against MFR will demonstrate whether the supervision added in our factorization loss has additive benefits.
- **Spline Interpolation**: Splines are effective intra-series alternatives for reconstructing missing segments. Cubic Spline Interpolation (**CSP**), interpolate a missing segment by fitting a local cubic polynomial (spline), assuring that the first derivative (tangential) and the second derivative (speed of curvature change) are preserved at the connection points to observed neighboring segments. Similarly B-Spline Interpolation (**BSP**), fits a third degree piecewise polynomial into a missing interval by using neighbor observed points as control points or de Boor points (Opfer and Knott 2001). The relative performance of SMF against spline interpolation will clarify whether collaborative reconstruction is superior to the intra-series approach of splines.

5.1 Experimental Setup

The experiments were conducted on a massive setup of 37 time series data sets belonging to the UCR collection[2]. In absence of publicly available sparse time series data, sparsification was simulated by randomly removing segments of continuous time points. Each segment has length equal to 10% of the whole series, where the starting point of the segment is picked uniformly random. Every data set was subject to a 5-fold cross validation split, three folds used for training, one for validation and one fold for testing. The final accuracy on the test sets was averaged among splits. The SMF hyperparameters to be tuned were: the regularization coefficients $\lambda_U, \lambda_V, \lambda_W$ of equation 6, the number of latent dimensions K, μ of Equation 3, the learning rate η of Algorithm 1, and the SVM complexity parameter C. In order to reduce the computational time, the same λ values were used for all latent matrices and the SVM learns a fixed degree polynomial kernel. To easy experiment reproducibility we report the following ranges of parameters: Learning rates η one of $\{0.01, 0.001, 0.0001\}$, K around 30% of the number of features, λ one of $\{0.0001, 0.001, 0.01, 0.1\}$, μ one of $\{0.5, 0.6, 0.7, 0.8\}$, SVM's C one of $\{0.125, 0.25, 0.5, 1, 2, 4\}$, while the degree of polynomial kernel was fixed to 4.

To help the interpretation of our results, we introduce a metric for categorizing the data sets in the examined collection. The metric, denoted as $E2D$, is defined as $E2D = \mathcal{E}(Euclidean) - \mathcal{E}(DTW)$, where $\mathcal{E}(Euclidean)$ denotes the error percentage of 1-NN classifier using Euclidean distance (hereafter simply referred to as *Euclidean-NN*). Similarly $\mathcal{E}(DTW)$ denotes the error percentage of 1-NN using DTW (hereafter referred to as *DTW-NN*). Only for this categorization, the provided (non-sparse) train/test splits in the UCR collection were used. Based on the E2D metric, the UCR data sets can be categorized into three types:

- **Type-1 Data Sets**: $E2D \leq 0$. *Euclidean-NN*, is more, or equally, accurate w.r.t. *DTW-NN*, implying data has no patterns shifted in time. Nearest neighbor classifier is a

[1] Expectation Maximization (**EM**) was also a candidate, however it performed poorly in our experiments. EM implementations suggest that the number of training instances be more than the number of features (Schafer 1997), but time series data sets do often have more features (time points) than instances.

[2] www.cs.ucr.edu/~eamonn/time_series_data

dominant factor affecting classification accuracy, while *DTW-NN* is expected to perform non-optimally.

- **Type-2 Data Sets**: $0 < E2D \leq \alpha$. *DTW-NN* is slightly superior w.r.t *Euclidean-NN*, implying patterns are slightly shifted in time. In order for the types to contain about equal number of data sets, we set α to 10. SMF is not supposed to handle time shifting, however we still expect SMF to be competitive because the disadvantage in considering time shifting is counteracted by the advantage in handling sparsity without needing reconstruction/imputation. Meanwhile the accuracies of DTW-based methods are expected to deteriorate because of the additive reconstruction error.
- **Type-3 Data Sets**: $E2D > \alpha$. *DTW-NN* has large superiority, therefore time series contain patterns significantly shifted in time. Since SMF is not modeled to consider large time-shifts, we expect worse results than baselines.

5.2 Results

Table 1 presents the results of the experiments. Three degrees of sparsity were applied to every data set $20\%, 40\%$ and 60%. Result values denote classification error percentages, where **bold** results show the winning method per degree of sparsity. Moreover values shown with circle ($^\circ$) denote cases where the difference between the average error of the winning method and second best method is adequately large, such that the standard deviations span of these two errors do not overlap. The results are interpreted in accordance with the $E2D$-based categorization as follows:

- **Type-1 Results**: In data sets where patterns are not shifted in time. As expected from above, SMF has a clear dominance winning 36.33 (fraction due to ties) out of 45 experiments.
- **Type-2 Results**: In accordance to our expectation, SMF's superior handling of sparsity counteracts the initial light accuracy advantage of the baselines that handle time shifts. It wins 22.83 out of 36.
- **Type-3 Results**: The existence of significant time shifts provides superiority to the baselines. Therefore SMF could only win 7 out of 30 experiments.

Overall Results: SMF wins in the vast majority of experiments as demonstrated in the *Total Wins* row of the table.

Below, we demonstrate experiments on two data sets with varying levels of sparsity. Results for a Type-1 data set, Adiac, are depicted in Figure 1, showing that SMF constantly results in the lowest error ratio on all degrees of sparsity. The second graph shown in Figure 3 illustrates CBF, a Type-3 data set. As expected, SMF is not the winning method in all sparsity ratios. However due to the advantage of avoiding reconstruction, SMF recovers the handicap of not detecting time shifts and therefore it is highly competitive against the winning baselines.

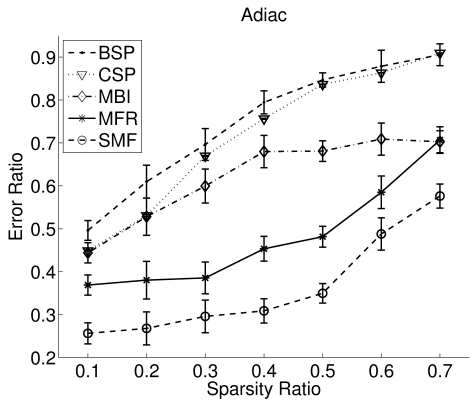

Figure 1: Type-1 data set 'Adiac': Error Ratios per Different Degrees of Sparsity, ($0.x$ is $x \cdot 100\%$). Vertical bars denote standard deviations.

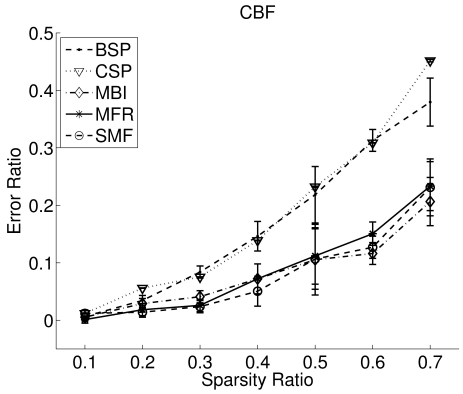

Figure 2: Type-3 Data Set 'CBF': Error Ratios per Different Degrees of Sparsity. Vertical bars denote standard deviations.

6 Conclusions

We introduced a new principle in analyzing the problem of sparse time series classification which, in contrast to other approaches, avoids reconstructing missing segments and presented a stochastic learning algorithm which operates only on observed data. The proposed method relies on a customized supervised matrix factorization technique that projects sparse time series into a dense latent space.

Large experimentations were conducted on 37 time series data sets, where the superiority of the supervised factorization was clearly distinguishable compared to examined baseline, which incur noise during the reconstruction of the missing segments. Due to avoiding reconstruction and collaboratively building a latent data projection, our method exhibited superiority on vast majority of experiments.

Despite being superior in treating sparsity, the supervised factorization technique is a generic method suited for feature vector data instances, where each feature is treated equally in the objective function. It is not specifically tailored to time series, where the features/points are correlated to the neighbor points, therefore it is not designed to consider patterns which might be heavily shifted in time. As a future work, we plan to incorporate a time shift detection mechanism into the

Table 1: **Classification Error Percentages of Different Algorithms on Sparse Data Sets**

(i) E2D is defined in section 5.1
(ii) All baselines BSP, CSP, MBI, MFR reconstruct the sparse dataset, which is later classified by 1-NN with DTW

Data Set	E2D	20% Sparsity					40% Sparsity					60% Sparsity				
		BSP	CSP	MBI	MFR	SMF	BSP	CUB	MBI	MFR	SMF	BSP	CUB	MBI	MFR	SMF
Type-1 Data Sets																
EC2	-11	24	22	20	23	**16**	29	29	24	26	°**15**	38	32	26	**23**	24
MOT	-4	11	12	10	**8**	**8**	17	15	13	14	**12**	22	25	**17**	18	**17**
BEE	-3	47	55	53	57	**43**	57	57	**48**	50	50	70	70	**58**	**58**	**58**
ECF	-3	9	11	7	4	°**0**	22	26	11	9	°**3**	33	42	18	16	°**11**
SO2	-3	14	13	9	7	**5**	23	25	12	13	°**8**	32	32	**13**	14	**13**
UWY	-3	42	40	45	40	°**34**	51	51	52	45	°**40**	63	61	60	52	°**48**
WAF	-2	2	3	*Runn.*	2	°**0**	4	4	4	2	°**1**	5	7	5	3	°**2**
ADI	-1	60	53	53	37	°**26**	79	76	68	48	°**35**	88	87	71	59	°**49**
GUN	-1	12	19	23	10	**6**	17	25	29	19	**9**	33	36	27	28	**23**
ITA	-1	8	7	5	5	**4**	12	12	**5**	6	**5**	16	16	**3**	8	6
UWX	-1	32	31	33	30	°**27**	43	42	43	36	°**32**	59	55	54	45	°**39**
UWZ	-1	38	38	40	37	°**31**	50	46	47	42	°**37**	61	58	56	50	°**44**
CHL	0	52	52	43	37	°**2**	57	57	49	41	°**7**	56	57	52	38	°**19**
OLI	0	60	62	°**22**	38	43	55	77	°**15**	37	42	68	70	°**27**	57	63
SWE	0	40	37	32	26	°**18**	63	61	44	41	°**27**	77	78	53	55	°**43**
Type-1 Wins		0	0	1	0.5	**13.5**	0	0	2.5	0	**12.5**	0	0	3.33	1.33	**10.33**
Type-2 Data Sets																
DIA	+3	6	5	3	**1**	**1**	17	11	6	**0**	1	33	26	9	4	**3**
WOR	+3	49	48	48	**40**	41	69	68	60	55	**50**	83	80	72	69	°**62**
MED	+5	36	30	30	**28**	32	41	41	40	36	**35**	49	52	**43**	**43**	**43**
SON	+5	14	12	7	6	°**3**	22	24	9	8	**5**	31	33	13	15	°**8**
SYM	+5	6	5	9	**4**	5	12	12	15	**6**	9	22	21	22	14	**11**
FAF	+5	31	39	19	9	**8**	49	70	25	20	**15**	67	64	**25**	**25**	31
FIS	+5	49	45	45	35	°**17**	69	67	51	48	°**23**	77	77	56	58	°**41**
50W	+6	52	51	48	41	°**37**	71	70	62	61	°**47**	86	84	72	75	°**63**
COF	+7	41	33	14	11	**5**	43	43	16	13	**5**	59	44	**12**	25	18
OSU	+7	56	53	51	**43**	45	64	67	62	54	**50**	73	75	72	66	**62**
FAA	+9	26	28	13	**8**	12	52	60	28	23	**21**	73	78	38	42	°**34**
TWO	+9	**3**	**3**	7	5	16	17	20	21	°**13**	31	41	46	39	°**31**	48
Type-2 Wins		0.5	0.5	0	**5.5**	**5.5**	0	0	0	3	**9**	0	0	1.83	1.83	**8.33**
Type-3 Data Sets																
SYN	+11	12	10	4	**3**	4	27	25	**5**	7	12	50	48	**10**	14	19
LI2	+12	26	27	25	**20**	34	°**25**	43	30	29	36	44	48	°**33**	40	43
FAU	+14	29	28	14	°**9**	12	54	58	28	23	°**21**	72	77	40	39	**36**
CBF	+15	3	6	3	2	**1**	15	14	7	7	**5**	31	30	**11**	15	13
CRY	+15	45	46	40	**36**	47	65	67	55	°**48**	60	78	82	68	°**58**	74
LI7	+15	40	43	37	**31**	48	53	59	49	**46**	49	63	72	**46**	49	60
TWE	+16	6	6	3	1	**0**	17	18	8	4	**2**	30	32	11	9	**6**
CRZ	+17	43	42	41	°**32**	49	59	62	52	**49**	63	75	80	67	**61**	70
CRX	+20	43	41	40	°**32**	52	57	67	53	**47**	59	78	81	67	°**60**	72
TRA	+24	9	9	21	°**8**	17	18	°**16**	28	20	29	36	40	37	**28**	33
Type-3 Wins		0	0	0	**8**	2	1	1	1	**4**	3	0	0	**4**	**4**	2
Cumulative Results																
Total Wins		0.5	0.5	1	14	**21**	1	1	3.5	7	**24.5**	0	0	9.16	7.16	**20.66**

Data Set Name Acronyms

EC2:ECG200, MOT:MoteStrain, 50W:50Words, BEE:Beef, TWE:TwoLeadECG, UW*:uWaveGestureLibrary_*,
ECF:ECGFiveDays, MED:MedicalImages, ITA:ItalyPowerDemand, CHL:ChlorineConcentration, WOR:WordsSynonyms,
TRA:Trace, SWE:SwedishLeaf, DIA:DiatomSizeReduction, SON:SonyAIBORobotSurface, SO2:SonyAIBORobotSurfaceII
FAF:FaceFour, FIS:Fish, COF:Coffee, OSU:OSULeaf, FAA:FaceAll, TWO:Two_Patterns, GUN:Gun-Point, SYM:Symbols
SYN:SyntheticControl, WAF:Wafer, OLI:OliveOil, ADI:Adiac, FAU:FacesUCR, LI*:Lightning*, CR*:Cricket*, CBF:CBF

latent feature extraction (factorization) algorithm.

7 Acknowledgement

Funded by the Seventh Framework Programme of the European Comission, through project REDUCTION (# 288254). www.reduction-project.eu

References

Berndt, D. J., and Clifford, J. 1996. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*. 229–248.

Cai, D.; He, X.; Han, J.; and Huang, T. S. 2011. Graph regularized nonnegative matrix factorization for data representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 33(8):1548–1560.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine Learning* 20(3):273–297.

Ding, H.; Trajcevski, G.; Scheuermann, P.; Wang, X.; and Keogh, E. J. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB* 1(2):1542–1552.

Eads, D.; Hill, D.; Davis, S.; Perkins, S.; Ma, J.; Porter, R.; and Theiler, J. 2002. Genetic Algorithms and Support Vector Machines for Time Series Classification. In *Proc. SPIE 4787; Fifth Conference on the Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation; Signal Processing Section; Annual Meeting of SPIE*.

Gemulla, R.; Nijkamp, E.; Haas, P. J.; and Sismanis, Y. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In Apté, C.; Ghosh, J.; and Smyth, P., eds., *KDD*, 69–77. ACM.

Kehagias, A., and Petridis, V. 1997. Predictive modular neural networks for time series classification. *Neural Networks* 10(1):31–49.

Keogh, E. J., and Pazzani, M. J. 2000. Scaling up dynamic time warping for datamining applications. In *KDD*, 285–289.

Keogh, E. J., and Ratanamahatana, C. A. 2005. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.* 7(3):358–386.

Koren, Y.; Bell, R. M.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* 42(8):30–37.

Marwala, T. 2009. *Computational Intelligence for Missing Data Imputation, Estimation, and Management: Knowledge Optimization Techniques*. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 1st edition.

Menon, A. K., and Elkan, C. 2010. Predicting labels for dyadic data. *Data Min. Knowl. Discov.* 21(2):327–343.

Opfer, G., and Knott, G. D. 2001. Interpolating cubic splines. *Journal of Approximation Theory* 112(2):319–321.

Pavlovic, V.; Frey, B. J.; and Huang, T. S. 1999. Time-series classification using mixed-state dynamic bayesian networks. In *CVPR*, 2609–. IEEE Computer Society.

Raghunathan, T. E.; Lepkowski, J. M.; Hoewyk, J. V.; and Solenberger, P. 2001. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology* 27:85–95.

Rendle, S., and Schmidt-Thieme, L. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In Pu, P.; Bridge, D. G.; Mobasher, B.; and Ricci, F., eds., *RecSys*, 251–258. ACM.

Rostamizadeh, A.; Agarwal, A.; and Bartlett, P. L. 2011. Learning with missing features. In Cozman, F. G., and Pfeffer, A., eds., *UAI*, 635–642. AUAI Press.

Rutkowski, T. M.; Zdunek, R.; and Cichocki, A. 2007. Multichannel EEG brain activity pattern analysis in time-frequency domain with nonnegative matrix factorization support. *International Congress Series* 1301:266–269.

Schafer, J. 1997. *Analysis of Incomplete Multivariate Data*. London: Chapman and Hall.

Singh, A. P., and Gordon, G. J. 2008. A unified view of matrix factorization models. In *ECML/PKDD (2)*, 358–373.

Smaragdis, P., and Brown, J. C. 2003. Non-negative matrix factorization for polyphonic music transcription. In *In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 177–180.

Srebro, N.; Rennie, J. D. M.; and Jaakola, T. S. 2005. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, 1329–1336. MIT Press.