Preface

We are honored to welcome you to the 2nd International Workshop on Advanced Analytics and Learning on Temporal Data (AALTD), which is held in Riva del Garda, Italy, on September 19th, 2016, co-located with The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2016).

The aim of this workshop is to bring together researchers and experts in machine learning, data mining, pattern analysis and statistics to share their challenging issues and advance researches on temporal data analysis. Analysis and learning from temporal data cover a wide scope of tasks including learning metrics, learning representations, unsupervised feature extraction, clustering and classification.

This volume contains the conference program, an abstract of the invited keynotes and the set of regular papers accepted to be presented at the conference. Each of the submitted papers was reviewed by at least two independent reviewers, leading to the selection of eleven papers accepted for presentation and inclusion into the program and these proceedings. The contributions are given by the alphabetical order, by surname.

The keynote given by Marco Cuturi on "Regularized DTW Divergences for Time Series" focuses on the definition of alignment kernels for time series that can later be used at the core of standard machine learning algorithms. The one given by Tony Bagnall on "The Great Time Series Classification Bake Off" presents an important attempt to experimentally compare performance of a wide range of time series classifiers, together with ensemble classifiers that aim at combining existing classifiers to improve classification quality. Accepted papers spanned from innovative ideas on analytic of temporal data, including promising new approaches and covering both practical and theoretical issues.

We wish to thank the ECML PKDD council members for giving us the opportunity to hold the AALTD workshop within the framework of the ECML/PKDD Conference and the members of the local organizing committee for their support.

The organizers of the AALTD conference gratefully thank the financial support of the Université de Rennes 2, MODES and Universidade da Coruña.

Last but not least, we wish to thank the contributing authors for the high quality works and all members of the Reviewing Committee for their invaluable assistance in the
selection process. All of them have significantly contributed to the success of AALTD 2106.

We sincerely hope that the workshop participants have a great and fruitful time at the conference.

Andrés M. Alonso
Benjamin Bustos
Ahlame Douzal-Chouakria
Simon Malinowski
Pierre-François Marteau
Edoardo Otranto
Romain Tavenard
José A. Vilar
Conference Program

AALTD’16 takes place on September 19, 2016 in Riva Del Garda Congress Center (room 1000B).

Morning talks

- Session I (9:00 am – 10:40 am)
  - 9:00 am – 9:20 am: Welcome speech by workshop chairs
  - 9:20 am – 9:40 am: Transfer Learning for Time Series Classification in Dissimilarity Spaces
  - 9:40 am – 10:00 am: Missing Data Prediction in Multi-source Time Series with Sensor Network Regularization
  - 10:00 am – 10:20 am: A time series two-sample test based on comparing distributions of pairwise distances
  - 10:20 am – 10:40 am: Scaling up Vector Autoregressive Models With Operator-Valued Random Fourier Features

- Coffee break (10:40 am – 11:00 am)

- Session II (11:00 am – 12:40 pm)
  - 11:00 am – 11:40 am: Invited talk by Tony Bagnall
  - 11:40 am – 12:00 pm: EAST representation: fast discovery of discriminant temporal patterns from time series
  - 12:00 pm – 12:20 pm: Discovering relationships in climate-vegetation dynamics using satellite data
  - 12:20 pm – 12:40 pm: Progressive Temporal Window Widening

- Lunch break (12:40 pm – 2:20 pm)
Afternoon talks

- **Session III (2:20 pm – 4:00 pm)**
  - 2:20 pm – 3:00 pm: Invited talk by Marco Cuturi
  - 3:00 pm – 3:20 pm: Recurrent Neural Networks for Modeling Company-Product Time Series
  - 3:20 pm – 3:40 pm: Data Augmentation for Time Series Classification using Convolutional Neural Networks
  - 3:40 pm – 4:00 pm: Node Classification in Dynamic Social Networks

- **Coffee break (4:00 pm – 4:20 pm)**

- **Challenge Session (4:20 pm – 6:00 pm)**
  - 4:20 pm – 4:40 pm: Presentation of the challenge by challenge chairs
  - 4:40 pm – 5:00 pm: Assessing pattern recognition or labeling in streams of temporal data (Task 2 evaluation)
  - 4:40 pm – 6:00 pm: Presentation of competing methods by their authors
The Great Time Series Classification Bake Off

Tony Bagnall
School of Computing Sciences, University of East Anglia

Abstract

Time series classification (TSC) problems are differentiated from traditional classification problems because the attributes are ordered. In the last five years there have been a large number of new TSC algorithms proposed in the literature. These algorithms have been evaluated on subsets of the 47 data sets in the University of California, Riverside (UCR) time series classification archive. The archive has recently expanded to 85 data sets, over half of which have been donated by researchers at the University of East Anglia (UEA). The data has been relaunched as the UCR/UEA TSC archive and hosted on the website www.timeseriesclassification.com. The relaunch of the archive provides a timely opportunity to thoroughly evaluate recent algorithms on a larger number of datasets.

I present a rough taxonomy of approaches to TSC and then detail the workings of 18 recently proposed algorithms. We have implemented these algorithms in a common Java framework and compared them against two standard benchmark classifiers (and each other) by performing 100 resampling experiments on each of the 85 datasets. We use the results of some 30 million experiments to test several hypotheses relating to whether the algorithms are significantly more accurate than the benchmarks and each other. Our results indicate that only 9 of these algorithms are significantly more accurate than both benchmarks and that one classifier, the Collective of Transformation Ensembles, is significantly more accurate than all of the others. All of our experiments and results are reproducible: we release all of our code, results and experimental details and we hope these experiments form the basis for more robust testing of new algorithms in the future.

I will then present some TSC case studies to emphasise the strengths and weaknesses of alternative approaches and discuss where the future direction of my research into TSC lies.
Regularized DTW Divergences for Time Series

Marco Cuturi
Graduate School of Informatics, Kyoto University

Abstract

I will present in this talk the families of Global Alignment kernels for time series, and show that they can be seen as a DTW optimization problem endowed with an entropic regularization. I will discuss why this regularization is beneficial from a learning perspective, in the sense that GA kernels (and the divergence that can be derived from it by taking minus its logarithm) are smooth functions of the entries, and can thus be efficiently optimized to compute, as an application, Fréchet means of time series.
Scaling up Vector Autoregressive Models With Operator-Valued Random Fourier Features.

Romain Brault$^{1,3}$, Néhémie Lim$^2$, and Florence d’Alché-Buc$^3$

$^1$ IBISC, Université d’Évry-Val-d’Essonne, Évry, 91000, France
romain.brault@telecom-paristech.fr

$^2$ Department of Statistics, University of Washington, Seattle, WA 98195, USA
nehemyl@uw.edu

$^3$ LTCI CNRS, Télécom ParisTech, Université Paris-Saclay, France
florence.dalche@telecom-paristech.fr

Abstract. We consider a nonparametric approach to Vector Autoregressive modeling by working in vector-valued Reproducing Kernel Hilbert Spaces (vv-RKHS). The main idea is to build vector-valued models (OKVAR) using Operator-Valued Kernels (OVK). As in the scalar case, regression with OVK boils down to learning as many weight parameters as data, except that here, weights are vectors. To avoid the inherent complexity in time and in memory to deal with kernels, we introduce Operator-Valued Random Fourier Features (ORFF) that extend Random Fourier Features devoted to scalar-valued kernels approximation. Applying the approach to decomposable kernels, we show that ORFF-VAR is able to compete with OKVAR in terms of accuracy on stationary nonlinear time series while keeping low execution time, comparable to VAR. Results on simulated datasets as well as real datasets are presented.

1 Introduction

Time series are ubiquitous in various fields such as climate, biomedical signal processing, videos understanding to name but a few. When linear models are not appropriate, a generic nonparametric approach to modeling is relevant. In this work we build on a recent work about Vector Autoregressive models using Operator-Valued Kernels [1,2]. Vector autoregression is addressed in a vector-valued Reproducing Kernel Hilbert Space with the important property to allow for couplings between outputs. Given a $d$-dimensional time series of $N$ data points $\{x_1, \ldots, x_N\}$, autoregressive models based on operator-valued kernels have the form $x_{t+1} = h(x_t) = \sum_{\ell=1}^{N-1} K(x_t, x_\ell)c_\ell$ where coefficients $c_\ell \in \mathbb{R}^d, \ell = 1, \ldots, N - 1$ are the model parameters. A naive approach for training such a model requires a memory complexity $O(N^2d^2)$, which makes the method prohibitive for large-scale problems. To scale up standard algorithms, we define an approximated operator-valued feature map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ that allows to approximate the aforementioned model $h$ in the RKHS by the following function: $\tilde{h}(x_t) = \Phi(x_t)^*\theta \approx h(x_t)$. The features maps are matrices of size $D \times d$ where $D$ controls the quality of the approximation, $d$ is the dimension of the
inputs and \( \theta \) is here the parameter vector to learn. This formulation allows to reduce the memory complexity to \( O((N-1)D + (N-1)d) \) which is now linear w.r.t. the number of data points. The principle used for building the feature map extends the idea of Random Fourier Features to the operator-valued case [3,4].

2 Operator-Valued Kernels for Vector Autoregression

Assume that we observe a dynamical system composed of \( d \) state variables at \( N \) evenly-spaced time points. The resulting discrete multivariate time series is denoted by \( x_{1:N} = \{x_1, \ldots, x_N\} \) where \( x_t \in \mathbb{R}^d \) denotes the state of the system at time \( t \), \( t = 1, \ldots, N \). It is generally assumed that the evolution of the state of the system is governed by a function \( h \), such that \( x_t = h(x_{t-p}, \ldots, x_{t-1}) + u_t \) where \( t \) is a discrete measure of time and \( u_t \) is a zero-mean noise vector. Then \( h \) is usually referred to as a vector autoregressive model of order \( p \). In the remainder of the paper, we consider first-order vector autoregressive models, that is \( p = 1 \). In a supervised learning setting, the vector autoregression problem consists in learning a model \( \hat{h} : \mathbb{R}^d \rightarrow \mathbb{R}^d \) from a given training set \( S_N = \{(x_1, x_2), \ldots, (x_{N-1}, x_N)\} \subset \mathbb{R}^d \times \mathbb{R}^d \). In the literature, a standard approach to vector autoregressive modeling is to fit a VAR model. The VAR(1) model reads : \( h(x_t) = Ax_t \) where \( A \) is an \( d \times d \) matrix whose structure encodes the temporal relationships among the \( d \) state variables.

However, due to their intrinsically linear nature, VAR models fail to capture the nonlinearities underlying realistic dynamical systems. This paper builds upon the recent work of [2] where the authors introduced a family of nonparametric nonlinear autoregressive models called OKVAR (Operator-valued Kernel-based Vector AutoRegressive). OKVAR models rely on the theory of operator-valued kernels [5], which provides a versatile framework for learning vector-valued functions [6,7,8]. Those models can be regarded as natural extensions of VAR models to the nonlinear case.

Next, we provide key elements of the theory of vector-valued Reproducing Kernel Hilbert spaces (vv-RKHS) of functions from \( \mathbb{R}^d \) to \( \mathbb{R}^d \).

Definition 1 (Matrix-valued kernels). A function \( K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d} \) is said to be a positive semidefinite \( \mathbb{R}^{d \times d} \)-valued kernel if:

1) \( \forall x, z \in \mathbb{R}^d, K(x, z) = K(z, x)^* \),

2) \( \forall m \in \mathbb{N}, \forall \{(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}^d, i = 1, \ldots, m\}, \sum_{i,j=1}^m y_i^* K(x_i, x_j) y_j \geq 0 \).

Furthermore, for a given \( \mathbb{R}^{d \times d} \)-valued kernel \( K \), we associate \( K \) with a unique vv-RKHS \( (\mathcal{H}_K, \langle \cdot, \cdot \rangle_{\mathcal{H}_K}) \) of functions from \( \mathbb{R}^d \) to \( \mathbb{R}^d \). The precise construction of \( \mathcal{H}_K \) can be found in [6]. In this paper, we assume that all functions \( h \in \mathcal{H}_K \) are continuous. Then \( K \) is called an \( \mathbb{R}^d \)-Mercer kernel. Similarly to the case of scalar-valued kernels, working within the framework of vv-RKHS allows to take advantage of representer theorems for a class of regularized loss functions such as ridge regression. More precisely, we consider \( h \), a nonparametric vector autoregressive model of the following form assuming we have observed \( N \) data
points. Given \( x_t \) the state vector at time \( t \), we have
\[
\hat{x}_{t+1} = \sum_{\ell=1}^{N-1} K(x_t, x_\ell) c_\ell,
\]
where \( x_1:N = \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^d \) is the observed time series, \( K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{d \times d} \) is a matrix-valued kernel and \( c_1, \ldots, c_{N-1} \in \mathbb{R}^d \) are the model parameters. We call OKVAR any model of the above form. In [2], the authors developed a family of OKVAR models based on appropriate choices of kernels to address the problem of network inference where both the parameters \( c_\ell, \ell = 1, \ldots, N - 1 \) and the OVK itself are learned using a proximal block coordinate descent algorithm under sparsity constraints. In the following, we will not consider the kernel learning problem and will use a simple ridge loss. We will also illustrate our approach to a well known class of OVK, called decomposable or separable kernels [6,9] that were originally introduced to solve multi-task learning problems[10].

Other kernels may also be considered as developed in [11].

Proposition 1 (Decomposable kernels). Let \( k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R} \) be a scalar-valued kernel and \( B \in \mathbb{S}_d^+ \) a positive semidefinite matrix of size \( d \times d \). Then function \( K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{d \times d} \) defined for all \( (x, z) \in \mathbb{R}^d \times \mathbb{R}^d \) as \( K(x, z) = k(x)B \) is a matrix-valued kernel.

A common choice for the scalar-valued kernel is the Gaussian kernel: \( k_{\text{Gauss}}(x, z) = \exp(-\|x-z\|^2/(2\sigma^2)) \) for any \( x, z \in \mathbb{R}^d \) and \( \sigma > 0 \). The corresponding decomposable kernel is referred to as \( K_{\text{dec}} \) and is as follows: \( K_{\text{dec}}(x, z) = k_{\text{Gauss}}(x)B \) with \( B \in \mathbb{S}_d^+ \). While the model parameters \( c_\ell \)'s are estimated under sparsity constraints in [2], here we consider the classic kernel ridge regression setting where the loss function to minimize is: \( J(h) = \frac{1}{N-1} \sum_{\ell=2}^{N} \|h(x_{\ell-1}) - x_{\ell}\|_2^2 + \lambda \|h\|^2_{HK} \)
with \( \lambda \geq 0 \) and \( \|h\|^2_{HK} = \sum_{\ell=1}^{N-1} c_\ell K(x_\ell, x_\ell) c_\ell \). The optimization problem is solved using a L-BFGS-B which is well suited for optimization problems with a large number of parameters, and is wildly used as a training algorithm on small/medium-scale problems. However, like standard kernel methods, OKVAR suffers from unfavorable computational complexity both in time and memory since it needs to store the full Gram matrix, preventing its ability to scale to large data sets and making it really slow on medium scale problem. We argue that this obstacle can be effectively overcome: in the following we develop a method to scale up OKVAR to successfully tackle medium/large scale autoregression problems.

3 Operator-Valued Random Fourier Features

We now introduce our methodology to approximate OVKs. Given a translation-invariant kernel \( K(x, z) = K_0(x - z) \), we approximate \( K \) by finding an explicit feature map such that \( \Phi(x)^* \Phi(z) \approx K_0(x - z) \). The idea is to use a generalization of Bochner’s theorem for the OVK family that states that any translation-invariant OVK can be written as the Fourier transform of a positive operator-valued measure. More precisely, we build on the following proposition first proven in [7]. More details can be found in [11].

Proposition 2 ([7]). Let \( K \) be a shift-invariant \( \mathbb{R}^d \)-Mercer kernel. Suppose that \( \forall i, j = 1, \ldots, d, \ K_0(\cdot)_{ij} \in L^1(\mathbb{R}^d, dx) \) where \( dx \) denotes the Lebesgue measure on
Define the matrix $C(\omega)$ such that for all $\omega \in \mathbb{R}^d$,

$$C(\omega)_{ij} = \int_{\mathbb{R}^d} \exp(i\langle \delta, \omega \rangle) K_0(\delta)_{ij} d\delta = F^{-1}[K_0(\cdot)_{ij}](\omega).$$

(1)

Then (i) $C(\omega)$ is a non-negative operator for all $\omega \in \mathbb{R}^d$, (ii) $C(\cdot,\omega) \in L^1(\mathbb{R}^d, d\omega)$, and (iii) for all $\delta \in \mathbb{R}^d$, $K_0(\delta)_{ij} = \int_{\mathbb{R}^d} \exp(-i\langle \delta, \omega \rangle) C(\omega)_{ij} d\omega$.

In the following, suppose that $K_0 = k_0(\cdot)A$ is a decomposable kernel. Decomposable kernels belong to the family of translation-invariant OVK. From proposition 2 we see that $C(\omega)_{ij} = F^{-1}[k_0(\cdot)](\omega)A_{ij}$. We decompose $A$ as $A = BB^*$, note that $A$ does not depend on $\omega$, and we denote $\bigoplus_{j=1}^D z_j$ the $Dm$-long column vector obtained by stacking vectors $z_j \in \mathbb{R}^m$. Then we define an approximate feature map for $K_0$, called Operator-Valued Random Fourier Feature (ORFF) map [11] as follows: for all $x \in \mathbb{R}^d$,

$$\tilde{\Phi}^{\text{dec}}(x) = \frac{1}{\sqrt{D}} \bigoplus_{j=1}^D \left( \begin{array}{c} \cos \langle x, \omega_j \rangle B^* \\ \sin \langle x, \omega_j \rangle B^* \end{array} \right), \quad \omega_j \sim F^{-1}[k_0],$$

which can also be expressed as a Kronecker product of a scalar feature map with an operator:

$$\tilde{\Phi}^{\text{dec}}(x) = \tilde{\phi}(x) \otimes B^*, \quad \text{where} \quad \tilde{\phi}(x) = \frac{1}{\sqrt{D}} \bigoplus_{j=1}^D \left( \begin{array}{c} \cos \langle x, \omega_j \rangle \\ \sin \langle x, \omega_j \rangle \end{array} \right), \quad \omega_j \sim F^{-1}[k_0]$$

is a scalar-valued feature map. In particular, if $k_0$ is a Gaussian kernel with bandwidth $\sigma^2$, then $F^{-1}[k_0] = \mathcal{N}(0, 1/\sigma^2)$ as proven in [3]. More examples on different OVK can be found in [11] as well as a proof of the uniform convergence of the kernel approximation defined by $\hat{K}(x,z) = \tilde{\Phi}(x)^*\tilde{\Phi}(z)$ towards the true kernel. In the case of vector autoregression, we consider a model $\hat{h}$ of the form: $x_{t+1} = \tilde{\Phi}(x_t)^* \theta$. That model is referred to as ORFFVAR in the remainder of the paper. Now, given the operator-valued feature map, we get a linear model, and we want to minimize the loss function

$$\mathcal{J}(\theta) = \frac{1}{N-1} \sum_{t=2}^N \left\| (\tilde{\phi}(x_{t-1})^* \otimes B) \theta - x_t \right\|^2_2 + \lambda \|\theta\|_2^2$$

(2)

with $\lambda \geq 0$. [11] proposed to formulate the learning problem as a Stein equation when dealing with decomposable kernels, and then used an appropriate solver [12]. We opted here for a more general algorithm, which is a variant of the doubly stochastic gradient descent [13]. In a few words, this algorithm is a stochastic gradient descent that takes advantage of the feature representation of the kernel allowing the number of features to grow along with the number of points. [13] show that the number of iterations needed for achieving a desired accuracy $\varepsilon$ using a stochastic approximation is $\Omega(1/\varepsilon)$, making it competitive compared to other stochastic methods for kernels such as NORMA [14] and its OVK adaptation ONORMA [15]. We propose here in Algorithm 1, an extension of the
doubly stochastic gradient descent of [13] to OVK. Additionally we consider a batch approach w.r.t. the data and the features, and make it possible to ‘cap’ the maximum number of features. The inputs of the algorithm are: $X$ the input data, $Y$ the targets, $K_0(\cdot)$ the kernel used for learning, $\gamma_t$ the learning rate (see [13] for a discussion on the selection of a proper learning rate), $T$ the number of iterations, $n$ the size of data batch, $b$ the size of the feature batch, and $D$ the maximum number of features. Note that if $K_0$ is a scalar kernel, $D = T$, $b = 1$ and $n = 1$, we retrieve the algorithm formulated in [13].

Data: $X$, $Y$, $K_0$, $\gamma_t$, $\lambda$, $T$, $n$, $D$, $b$  

**Result:** Find $\theta$

Let $D_b = D/b$ and find $B$ and $\mu = F^{-1} [k_0];$

for $i = 1$ to $D_b$ do

$\theta_{i,\omega} = 0;$

end

for $t = 1$ to $T$ do

$A_t = X_t \times Y_t$, a random subsample of $n$ data from $X \times Y;$

$h(X_t) =$ predict $(X_t, \theta^t, K_0);$  // Make a prediction.

$\Omega_t \sim \mu(\omega)$ with seed $i$, where $i = ((t - 1) \mod D_b) + 1;$  // Sample $b$

features from $\mu(\omega)$.

for $\omega \in \Omega_t$ // Update the parameters from the gradient.

$\theta_{t+1} = \theta_t - \gamma_t \left( \frac{1}{|A_t|} \sum_{(x,y) \in A_t} \frac{B \exp(i(\omega, x))(h(x) - y)}{\sqrt{D}} + \lambda \theta_{t,\omega} \right);$

end

end

**Algorithm 1:** Block-coordinate mini-batch doubly SGD.

In addition, the convergence of the algorithm can be speeded-up by preconditioning by the Hessian of the system.

4 Numerical Performance

**Simulated data.** To assess the performance of our models, we start our investigation by generating discrete $d$-dimensional time series $(x_t)_{t \geq 1}$ as follows

\[
\begin{align*}
\begin{aligned}
x_1 &\sim \mathcal{N}(0, \Sigma_x) \\
x_{t+1} &= h(x_t) + u_{t+1}, \forall t > 0.
\end{aligned}
\end{align*}
\]  

(3)

where the residuals are homoscedastic and distributed according to $u_t \sim \mathcal{N}(0, \Sigma_u)$. We study two different kinds of noise: an isotropic noise with covariance $\Sigma_u = \sigma_u^2 I_d$ and an anisotropic noise with Toeplitz structure $\Sigma_{u,ij} = \nu^{|i-j|}$, where $\nu$ lives in $(0, 1)$. We generated $N = 1000$ data points and used a sequential cross-validation (SCV) with time windows $N_t = N/2$ to measure the Mean Squared Error (SCV-MSE) of the different models. Next, we compare the performances
of VAR(1), OKVAR and ORFFVAR through three scenarios. Across the simulations, the topological structures of the underlying dynamical systems are encoded by a matrix $A$ of size $5 \times 5$. All entries of $A$ are set to zero except for the diagonal where all coefficients are equal to 0.9 for Settings 1 and 3 and 0.5 for Setting 2. Then five off-diagonal coefficients are drawn randomly from $\mathcal{N}(0,0.3)$ for Settings 1 and 3 and $\mathcal{N}(0,0.5)$ for Setting 2. We check that all the eigenvalues of $A$ are less than one to ensure the stability of the system. More specifically, we picked the following values of parameters for each scenario:

- **Setting 1: Linear model.** $h(x_t) = Ax_t$, $\nu = 0.9$ and $\sigma_u = 0.9$.
- **Setting 2: Exponential model.** $h(x_t) = A \exp(x_t)$ where $\exp$ is the element-wise exponential function, $\nu = 0.09$ and $\sigma_u = 0.09$.
- **Setting 3: Sine model.** $h(x_t) = A \sin(x_t)$ where $\sin$ is the element-wise sine function, $\nu = 0.9$ and $\sigma_u = 0.009$.

ORFFVAR is instantiated with $D = 25$ random features in presence of a white noise while we set $D = 50$ in case of a Toeplitz noise. We summarize the computational efficiency and the statistical accuracy of the models in Table 1. Throughout all the experiments, we set $B$ as the identity matrix of size $d \times d$. This reflects the absence of a prior on the structure of the data. A further study on the influence of the choice of $B$ can be found in [8].

In Setting 1, we observe that OKVAR does not provide any advantage over VAR(1) as expected since the data were generated according to a linear VAR(1) model. Note that OKVAR takes orders of magnitude more time to achieve the same performance as VAR(1) while ORFFVAR performs equally well with a competitive timing. In nonlinear scenarios (Settings 2 and 3), OKVAR and ORFFVAR consistently outperform VAR(1). Noticeably, ORFFVAR reaches the accuracy of OKVAR with the computation time of VAR(1).

<table>
<thead>
<tr>
<th>Setting</th>
<th>model</th>
<th>noise</th>
<th>SVC-MSE</th>
<th>variance</th>
<th>time</th>
<th>SVC-MSE</th>
<th>variance</th>
<th>time</th>
<th>SVC-MSE</th>
<th>variance</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR(1)</td>
<td>White</td>
<td>0.914979</td>
<td>0.573248</td>
<td>0.002467(s)</td>
<td>0.001275</td>
<td>0.000994</td>
<td>0.001697(s)</td>
<td>0.009534</td>
<td>0.006003</td>
<td>0.001697(s)</td>
<td></td>
</tr>
<tr>
<td>Toeplitz</td>
<td>1.091096</td>
<td>1.267880</td>
<td>0.004822(s)</td>
<td>0.017014</td>
<td>0.013498</td>
<td>0.002050(s)</td>
<td>0.116901</td>
<td>0.127396</td>
<td>0.001702(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORFFVAR</td>
<td>White</td>
<td>0.91661</td>
<td>0.572096</td>
<td>0.000994(a)</td>
<td>0.001003</td>
<td>0.000647</td>
<td>0.001284(a)</td>
<td>0.009536</td>
<td>0.005998</td>
<td>0.002777(a)</td>
<td></td>
</tr>
<tr>
<td>Toeplitz</td>
<td>1.071801</td>
<td>1.269678</td>
<td>0.001022(a)</td>
<td>0.012635</td>
<td>0.008837</td>
<td>0.002114(a)</td>
<td>0.11664</td>
<td>0.127395</td>
<td>0.000934(a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OKVAR</td>
<td>White</td>
<td>0.958790</td>
<td>0.591934</td>
<td>0.104706(s)</td>
<td>0.001100</td>
<td>0.000731</td>
<td>0.027099(s)</td>
<td>0.009227</td>
<td>0.005717</td>
<td>0.014458(s)</td>
<td></td>
</tr>
<tr>
<td>Toeplitz</td>
<td>1.410669</td>
<td>1.312049</td>
<td>0.289046(a)</td>
<td>0.013854</td>
<td>0.008777</td>
<td>1.850688(a)</td>
<td>0.360133</td>
<td>0.135760</td>
<td>0.034176(a)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Sequential cross-validation MSE and computation times for VAR(1), ORFFVAR and OKVAR on synthetic data (Settings 1, 2 and 3).

**Influence of the number of random features.** Next, we investigate the impact of $D$, the number of random features for ORFFVAR. To this end, we generated $N = 10000$ data points following eq. (3), with exponential nonlinearities and white noise as in Setting 2. We performed a sequential cross-validation on a window of $N/2$ data. As expected the error decreases with the number of random features $D$ (Table 2). For the same computation time ($D = 25$) as VAR(1), ORFFVAR achieves an SCV-MSE that is twice as small.
Table 2. SVC-MSE with respect to \( D \) the number of random features for ORFFVAR.

**Real data.** We now investigate three real datasets. The performances of the models on those datasets are recorded in Table 3. Throughout the experiments, the hyperparameters are set as follows: the bandwidth of the Gaussian kernel \( \sigma \) is chosen as the median of the Euclidean pairwise distances and the regularization parameter \( \lambda \) was tuned on a grid. The number of random features \( D \) and the parameters in Algorithm 1 were picked so as to reach the level of accuracy of OKVAR/VAR.

**Macrodata:** this dataset is part of the python library Statmodels\(^4\). It contains 204 US macroeconomic data points collected on the period 1959–2009. Each data point represents 12 economic features. No pre-processing is applied before learning. We measure SVC-MSE using a window of 25 years (50 points). We instantiated Algorithm 1 as follows: \( \gamma_t = 1, \lambda = 10^{-3}, D = 100, T = 2 \) and \( b = 50 \) for ORFF and \( \lambda = 0.00025 \) and \( \sigma = 11.18 \) for OKVAR.

**Gesture phase:** this dataset\(^5\) is constructed using features extracted from seven videos with people gesticulating. We present the results for videos 1 and 4, consisting in 1069 data points and 31 features. Data are normalized prior to learning. We measure SVC-MSE using a time window of 200 points. We implemented ORFFVAR with \( \gamma_t = 1, \lambda = 10^{-3}, D = 100, T = 2 \) and \( b = 50 \).

**Climate:** this dataset\(^6\) contains monthly meteorological measurements of 18 variables (temperature, CO2 concentration,...) collected at 135 different locations throughout the USA and recorded over 13 years, thus resulting in 135 time series of dimension 18 and length 156. Data are standardized at each station. A unique model is learned for all stations. SVC-MSE is measured on a window of 1872 points, corresponding to the data of all the 135 stations over one year. Specifically, we set the parameters of ORFFVAR as follows: \( \gamma_t = 1, \lambda = 10^{-6}, D = 100, T = 1 \) and \( b = 100 \).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>N</th>
<th>d</th>
<th>( D = 1 )</th>
<th>( D = 5 )</th>
<th>( D = 10 )</th>
<th>( D = 25 )</th>
<th>( D = 50 )</th>
<th>( D = 100 )</th>
<th>VAR(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVC-MSE</td>
<td>0.005342</td>
<td>0.001111</td>
<td>0.000991</td>
<td>0.000962</td>
<td>0.000949</td>
<td>0.000944</td>
<td>0.000944</td>
<td>0.000944</td>
<td>0.001660</td>
</tr>
<tr>
<td>variance</td>
<td>0.008639</td>
<td>0.000793</td>
<td>0.000660</td>
<td>0.000618</td>
<td>0.000608</td>
<td>0.000605</td>
<td>0.000605</td>
<td>0.000605</td>
<td>0.001363</td>
</tr>
<tr>
<td>time</td>
<td>0.001191</td>
<td>0.002384</td>
<td>0.003614</td>
<td>0.018469</td>
<td>0.038229</td>
<td>0.069294</td>
<td>0.019634</td>
<td>0.019634</td>
<td>0.001363</td>
</tr>
</tbody>
</table>

Table 3. SVC-MSE and computation times for ORFFVAR, VAR(1) and OKVAR on real datasets.

5 Discussion

Operator-Valued Random Fourier Feature provides a way to approximate OVK and in the context of time series, allows for nonlinear Vector Autoregressive

\(^4\) https://github.com/statsmodels/statsmodels

\(^5\) https://archive.ics.uci.edu/ml/datasets/Gesture+Phase+Segmentation
models that can be efficiently learned both in terms of computing time and memory. We illustrate the approach with a simple family of Operator-valued kernels, the so-called decomposable kernels but other kernels may be used. While we focused on first-order autoregressive models, we will consider extensions of our models for higher orders. In this work, the kernel hyperparameter $B$ is given prior to learning, however it would be interesting to learn $B$ as in OKVAR. Thus, a promising perspective is to use these models in tasks such as network inference and search for causality graphs among the state variables for large-scale time series [1,2].

References

Data Augmentation for Time Series Classification using Convolutional Neural Networks

Arthur Le Guennec\(^1\), Simon Malinowski\(^2\), and Romain Tavenard\(^1\)

\(^1\) LETG-Rennes COSTEL / IRISA – Univ. Rennes 2
\(^2\) IRISA – Univ. Rennes 1

Firstname.Lastname@irisa.fr

Abstract.

Time series classification has been around for decades in the data-mining and machine learning communities. In this paper, we investigate the use of convolutional neural networks (CNN) for time series classification. Such networks have been widely used in many domains like computer vision and speech recognition, but only a little for time series classification. We design a convolutional neural network that consists of two convolutional layers. One drawback with CNN is that they need a lot of training data to be efficient. We propose two ways to circumvent this problem: designing data-augmentation techniques and learning the network in a semi-supervised way using training time series from different datasets. These techniques are experimentally evaluated on a benchmark of time series datasets.

1 Introduction

Classification of time series (TS) has received a large interest over the last decades within the data mining and machine learning communities. It finds potential application in many fields as biology, medicine, or finance. Two main families of methods can be found in the literature for time series classification: distance-based and feature-based methods. Distance-based methods work directly on raw time series and make use of similarity measures between raw time series to perform classification. The most used similarity measures are Dynamic Time Warping (DTW) or Euclidean Distance (ED). The combination of DTW and \(k\)-nearest-neighbors (\(k\)-NN) is known to be a very efficient approach for TS classification [13]. Feature-based methods rely on extracting, from each time series, a set of feature vectors that describe it locally. Then, these feature vectors are most of the time quantized to form a Bag-of-Words (BoW). Each time series is finally represented by a histogram of word occurrences, which is then given to a classifier. Feature-based approaches for TS classification mostly differ in the kind of features extracted [2, 14, 15, 16]. These techniques are most of the time faster than distance-based methods and reach competitive accuracy.

Recently, deep learning has been widely used in many domains such as computer vision and speech processing for instance. Convolutional neural networks in particular have proved to be very efficient for image classification [8, 10]. These networks can be seen as a feature-based approach as the first layers of the...
network extract features from the training data while the last layers use these features to perform classification. CNN need huge training sets to be efficient, as they are composed of a lot of parameters (up to millions). In time series classification, UCR datasets are commonly used to evaluate the performance of different approaches. This benchmark is composed of relatively small datasets, which has led the authors of [17] not to use them to evaluate their proposed method based on CNN. In this paper, we study two approaches to improve the performance of CNN for time series classification even with small datasets. We first design data-augmentation techniques adapted to time series. We also study how to learn the CNN in a semi-supervised way (i.e. features are learned in an unsupervised manner whereas the classifier uses supervision). This technique allows us to learn our model’s convolution filters using training time series from several different datasets, leading to larger training sets. The network is then used for classification on a particular dataset once learned.

This paper is organized as follows. State-of-the-art approaches for time series classification are described in Section 2. Section 3 details how we design a CNN for time series classification and the two approaches we propose to improve its performance when faced with small datasets. Finally, Section 4 is dedicated to experiments that assess the performance of the proposed approaches.

2 State-of-the-art on time series classification

Time series classification methods can be split in two main categories: distance-based and feature-based methods. Distance-based methods rely on computing point-to-point distances between raw time series. These distances are coupled with a classifier like k-NN or Support Vector Machines (SVM) for instance. The two main similarity measures used for time-series classification are Dynamic Time Warping (DTW) and Euclidean Distance (ED). DTW has been shown to be more efficient for time series classification, as it can handle temporal distortion [13]. However, it has a higher computational cost than ED. In addition, the kernel derived from DTW is not semi-definite positive. In [7], Cuturi derived Global Alignment Kernel (GAK) from DTW. GAK is a robust similarity measure that can be used within kernel methods.

Feature-based methods extract local features from time series. Many different works have been proposed that differ in the kind of features that are extracted. For instance, Wang et al. [16] use wavelet coefficients. Fourier coefficients are used in [14]. SIFT features (widely used for image description) adapted to time series have been considered in [2]. Once these features are extracted, they are quantized into words and time series are represented by a histogram of word frequencies. These histograms are then given to a classifier. Symbolic Aggregate Approximation (SAX) symbols can also be used [15].

Recently, CNN have been used for time series classification in [17] and [6]. Authors of [17] claim that their network cannot be efficiently used for small datasets because of overfitting issues. In [6], authors propose different data-
augmentation methods to artificially increase the number of training samples. They show that these techniques improve classification accuracy.

3 Convolutional Neural Networks for Time Series Classification

Artificial neural networks are classification models that are composed of elementary units called neurons. Each neuron is associated with weights and the set of weights from all neurons in the model constitute the model parameters. A typical architecture for neural networks is the fully connected multilayer neural network architecture where neurons are organized in layers and each neuron from a given layer takes the outputs of all the neurons from the previous layer as its inputs. With such architectures, using more layers enables to consider more complex decision boundaries for the classification problem, at the cost of having to learn more parameters for the model.

Convolutional neural networks form a class of artificial neural networks that incorporate some translation invariance in the model. The principle on which these models rely is to learn convolution filters that efficiently represent the data. Such models have been successfully used in the computer vision community [3, 4] where spatial convolutions are cascaded to summarize spatial content of images. Besides translation invariance, these models are known to be less prone to overfitting than other artificial neural networks because they tend to have much fewer weights to optimize than fully connected models. CNN can be used as feature extractors to feed any kind of classifier such as fully connected multilayer neural networks or Support Vector Machines [3, 4].

In this paper, we use CNN with temporal convolutions for time series classification. In the following, we detail the different parts of our model as well as strategies to alleviate the possible lack of labelled time series.

Model. Our CNN model, denoted $t$-leNet in the following, is a time-series specific version of leNet model [11]. leNet has proved successful for image classification. It is made of two convolutions layers, each followed by a sub-sampling step performed through max pooling. Finally, fully connected layers enable to match extracted features with class labels to be predicted. The convolutional part of our model is presented in Fig. 1: a first convolution with 5 filters of temporal span equal to 5 is used, followed by a max pooling of size 2. Then, a second convolution layer is made of 20 filters with the same time span as the previous ones and a final max pooling of size 4 is used.

Data augmentation. As presented above, CNN tend to suffer less from overfitting than fully connected networks. However, it has been shown in the computer vision community that such models could still benefit from data augmentation methods [4]. This family of techniques aims at building synthetic data by transforming existing labelled samples so as to help the model learn the range of intra-class invariances one could observe. For example, when performing image classification, it is likely that flipping an image will not change its class. In practice, such synthetic samples will be added to training sets so as to enrich
them. In this section, we present the set of data augmentations we have used for time series.

A first method that is inspired from the computer vision community [8, 10] consists in extracting slices from time series and performing classification at the slice level. This method has been introduced for time series in [6]. At training, each slice extracted from a time series of class $y$ is assigned the same class and a classifier is learned using the slices. The size of the slice is a parameter of this method. At test time, each slice from a test time series is classified using the learned classifier and a majority vote is performed to decide a predicted label. This method is referred to as window slicing (WS) in the following.

The last data augmentation technique we use is more time-series specific. It consists in warping a randomly selected slice of a time series by speeding it up or down, as shown in Fig. 2. The size of the original slice is a parameter of this method. Fig. 2 shows a time series from the “ECG200” dataset and corresponding transformed data. Note that this method generates input time series of different lengths. To deal with this issue, we perform window slicing on transformed time series for all to have equal length. In this paper, we only consider warping ratios equal to $\frac{1}{2}$ or 2, but other ratios could be used and the optimal ratio could even be fine tuned through cross-validation on the training set. In the following, this method will be referred to as window warping (WW).

**Dataset mixing.** An important finding that has enabled the efficient use of huge amounts of data to learn deep models consists in pre-training the models. Indeed, it has been shown that the standard back-propagation procedure used to learn model parameters heavily suffers from the vanishing gradient phenomenon. To improve on this, pre-training each layer in an unsupervised manner (as an
auto-encoder) enables to reach regions of the parameter space that are better seeds for the back-propagation procedure [3]. Building on this, we suggest to pre-train our model in an unsupervised manner using time series from various datasets. Once this pre-training step is done, convolution filters are kept and the supervised part of the model is trained separately for each dataset. This method is called dataset mixing (DM) in the following.

4 Experimental evaluation

All the experiments presented in this section are conducted on public datasets from the UCR archive [9] using the torch framework [5]. When training neural networks, we use a learning rate of 0.01 with a decay of 0.005.

Impact of data augmentation. We first compare the efficiency of the data augmentation methods presented in Section 3. For this evaluation, we report dataset types as found in [1] that splits UCR datasets into 7 types: Device, ECG, Image outlines, Motion, Sensor, Simulated, and Spectro. Performance is studied through scatter plots of compared error rates and Win / Tie / Lose scores, and one-sided Wilcoxon signed rank test is used to assess statistical significance of observed differences (p-values are reported on the figures).

Fig. 3 indicates that both WS and WW methods help improve classification performance. For these experiments, the size of the slices for WS is set to 90% of the size of the original time series, while the size of the slices that are warped in WW is set to 10% of the size of time series. Observed improvement for WW cannot be considered significant at the 5% level because of error rate dispersion that is much higher than for the WS data augmentation method. However, one interesting point is that for WW, we can derive a rule of thumb of when to use it: the method almost always improve performance except for datasets of type Image outlines for which it should be avoided. Finally, when compared to state-of-the-art methods, both WS and WW methods improve on basic distance-based method performance but are still dominated by PROP ensemble method (cf. Fig. 4).
Fig. 3: Impact of WS and WW on classification performance (both axes correspond to error rates). Win / Tie / Lose scores indicate that this kind of data augmentation tends to improve classification accuracy ("Win" means that the y-axis method has lower error rate) and this difference can be considered significant at the 5% level for t-leNet-WS.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PROP</th>
<th>MCNN</th>
<th>t-leNet-WS + FC</th>
<th>t-leNet-WS DM + SVM</th>
<th>t-leNet-WS DM + FC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChlorineCon</td>
<td>0.360</td>
<td>0.203</td>
<td>0.188</td>
<td>0.129</td>
<td>0.203</td>
</tr>
<tr>
<td>ECG5000</td>
<td>–</td>
<td>0.061</td>
<td>0.001</td>
<td>0.002</td>
<td>0.002</td>
</tr>
<tr>
<td>ECGFiveDays</td>
<td>0.178</td>
<td>0.000</td>
<td>0.007</td>
<td>0.002</td>
<td>0.000</td>
</tr>
<tr>
<td>FacesUCR</td>
<td>0.063</td>
<td>0.063</td>
<td>0.108</td>
<td>0.052</td>
<td>0.113</td>
</tr>
<tr>
<td>Gun_Point</td>
<td>0.007</td>
<td>0.000</td>
<td>0.007</td>
<td>0.006</td>
<td>0.000</td>
</tr>
<tr>
<td>Plane</td>
<td>0.000</td>
<td>–</td>
<td>0.048</td>
<td>0.010</td>
<td>0.038</td>
</tr>
<tr>
<td>wafer</td>
<td>0.003</td>
<td>0.002</td>
<td>0.003</td>
<td>0.002</td>
<td>0.003</td>
</tr>
</tbody>
</table>

Table 1: Error rates obtained using dataset mixing.

**Dataset mixing and semi-supervised learning.** We now turn our focus on the impact of dataset mixing (DM). As presented above, this method consists in learning the convolutional part of our model in an unsupervised manner by mixing time series from several datasets. For this set of experiments, we have selected 7 datasets that share similar time series length. These datasets are listed in Table 1 that provides obtained error rates. Note that WS method is used in these experiments. In this table, we report results obtained by using either a set of fully connected layers or a SVM to perform per-dataset supervised classification and one can observe that both methods obtain similar results. Second, our methods reach competitive performance when compared to PROP [12] that is an ensemble classifier considered as one of the most efficient baselines for time series classification these days, and MCNN [6], a CNN for time series.
5 Conclusion

In this paper, we have designed a Convolutional Neural Network for time series classification. To improve the performance of this CNN when faced with small training sets, we propose two approaches to artificially increase the size of training sets. The first one is based on data-augmentation techniques. The second one consists in mixing different training sets and learning the network in a semi-supervised way. We show that these two approaches improve the overall classification performance. As a future work, we intend to improve the warping approach by considering more warping ratios and use more datasets to learn better feature extractors.
References

Assessing pattern recognition or labeling in
streams of temporal data

Pierre-François Marteau1,
IRISA, Université Bretagne Sud, Campus de Tohannic, Vannes, France
pierre-francois DOT marteau AT univ-ubs DOT fr

Abstract. In the data deluge context, pattern recognition or labeling in streams is becoming quite an essential and pressing task as data flows inside always bigger streams. The assessment of such tasks is not so easy when dealing with temporal data, namely patterns that have a duration (a beginning and an end time-stamp). This paper details an approach based on an editing distance to first align a sequence of labeled temporal segments with a ground truth sequence, and then, by back-tracing an optimal alignment path, to provide a confusion matrix at the label level. From this confusion matrix, standard evaluation measures can easily be derived as well as other measures such as the "latency" that can be quite important in (early) pattern detection applications.

1 Introduction

In the big data and Internet of things era, data is widely generated through streams at an increasing rate. Such data is provided by sensor applications, measured in network monitoring and traffic management system, available as log records or click-streams in web exploration, sequence of email, blogs data, RSS feeds, social networks, and so many other sources. Applications requiring pattern recognition in such streams is becoming more and more demanding, and, consequently, the question of assessing such applications is particularly essential. Among various type of stream data we are focusing on stream of temporal data, also called timestamped data stream, for which data elements (samples) are associated with a time index. The aim of this paper is, given a ground truth sequence of time-stamped labeled segments, to propose a consolidated and extended assessment framework for pattern labeling or recognition in streams.

Fig. 1. Example of a ground truth sequence.

We consider in this paper that the ground truth sequence is given as a sequence of time-stamped labels, each label being characterized by a begin and an
end time-stamp, as depicted in figure 1. It should be stressed that, to ensure as much as possible the independence of the assessment framework to the nature of the stream, such sequence does not keep any reference to the data that are conveyed by the stream except for the time location of the labeled segments.

The output of the pattern recognition or labeling tasks in stream that we intend to assess are expected to provide the same kind of sequences. The proposed algorithm will output

- a confusion matrix from which traditional assessment measures such as precision, recall or F-measures can be derived,
- an estimation of the recognition latency that can be an issue for real-time application,
- an estimation of the average relative duration of the matched labeled segments, that can be of some utility when assessing labeling tasks.

The contribution of this paper is the presentation of a dynamic programming algorithm dedicated to the alignment of such sequences. We insists that it is not one another stream segmentation or labeling method, but a proposal for the assessment of stream segmentation or labeling methods when time occurrence of the labels is a critical issue. The requirement for a ground-truth sequence is a strong constraint that indeed limit the application of the proposed assessment framework, but ensures in general a better acceptance and quality of the assessment since experts are supposedly maintained into the process.

2 Previous work and problem statement

Stream classification or labeling refers essentially to three distinct tasks [1][2]

- stream classification: the task consists in affecting a class label to a given stream. For instance, categorizing a RSS feed into some pre-defined categories such as sport, economy, society, culture, etc, is typically a stream classification problem.
- stream segment classification: here we consider that the streams are segmented (the segments are known), and the task consists in affecting a class label to each segment. Considering a general RSS feed, naturally segmented into time-stamped news items, the task is to affect to each news item a class label (e.g. sport, economy, society, etc.).
- pattern spotting and recognition in streams: here again we consider that the stream is also segmented but the segments are only known for training through manual segmentation and labeling. Hence the segments need to be localized along the time axis, with a beginning and an end time-stamps. This is the case for example when addressing human data such as speech data which can be segmented into phonemes, syllables, words, or any other acoustic or linguistic parts. This tasks is also referred to as temporal classification [3].
This paper addresses the assessment of approaches that tackle the third task, obviously the most difficult one since it requires to identify segment frontiers and segment classification as a whole. Formally, such approaches will typically output a sequence of labeled time-stamped segments (SLTSS).

Let \( L \) be the set of labels, and \( T \) a time segment. A labeled-segment \( s \) will be defined as a triplet \((l, t_b, t_e) \in L \times T \times T\) with the constraint that \( t_b \leq t_e \). A sequence of labeled time-stamped segments \( S \) will be denoted as \( S = s_1s_2 \cdots s_n = (l_i, t_{b_i}, t_{e_i})_{i=1 \cdots n} \). We will note \( S(i) \) the ith labeled segment in the sequence \( S \).

Furthermore, we impose an ordering of the segments inside the sequence, such that: \( \forall i, t_{b_i} \leq t_{b_{i+1}} \) and \( t_{e_i} \leq t_{e_{i+1}} \). From this definition, it should be noted that segments may have different lengths and successive segments can be partially overlapping or disjoint.

The problem of assessing pattern spotting and recognition systems in streams comes down to the alignment of a predicted SLTSS with a ground-truth SLTSS, with the constraint that only time overlapping segment can be matched.

If we were dealing uniquely with sequences of symbolic labels, an edit distance, such as the Levenshtein [4] or the Smith and Waterman distance [6] proposed in bioinformatics among others, would be perfectly adapted and has been used for various tasks in bioinformatics, natural language processing, etc, [2]. However, the time-stamps that delimit the temporal segments introduce a kind of fuzziness in the matching of segments that need to be coped with. It is quite striking that, to our knowledge, no dynamic programming algorithm has been yet proposed to solve this temporal alignment problem at a labeled segment level (straightforwards techniques at frame or sample levels are usually used). Although edit distance based measures have been design to cope with temporal segment sequences such as the time warp edit distance (TWED) [5], these measures does not cope with the label attached to the temporal segments. But more importantly, they are not suited for the alignment of SLTSS we are dealing with, mainly because they enable the alignment of disjoint time segments which has no meaning in our context. The purpose of the temporal alignment algorithm that we detail hereinafter intends to bridge this apparent gap.

3 Dynamic programming algorithm for the alignment of a pair of SLTSS

Following the mathematical definition of the Levenshtein edit distance designed to align two strings we define \( \delta_{\text{SLTSS}} \) as an edit distance allowing to align two SLTSS \( S_1, S_2 \) (of length \(|S_1|\) and \(|S_2|\) respectively) is given by \( \delta_{\text{SLTSS}}(S_1(|S_1|), S_2(|S_2|)) \) where
\[ \delta_{SLTSS}(S_1(i), S_2(j)) = \begin{cases} 
C_0 \cdot \max(i,j) & \text{if } \min(i,j) = 0, \\
\delta_{SLTSS}(S_1(i-1), S_2(j)) + C_0 & \\
\delta_{SLTSS}(S_1(i), S_2(j-1)) + C_0 & \\
\delta_{SLTSS}(S_1(i-1), S_2(j-1)) + C_m(S_1(i), S_2(j)) & \text{otherwise.}
\end{cases} \] (1)

where \( \delta_{SLTSS}(S_1(i), S_2(j)) \) is the distance between the first \( i \) segments of \( S_1 \) and the first \( j \) segments of \( S_2 \), \( C_0 > 0 \) is a constant penalty value corresponding to a segment insertion or deletion and \( C_m(S_1(i), S_2(j)) \geq 0 \) is the local cost associated to the correct matching of segment \( S_1(i) \) with segment \( S_2(j) \) or a substitution of the first segment by the second.

Fig. 2. An SLTSS example with contiguous, overlapping and disjoint segments.

The fact that streams can be partially labeled, meaning that labeled segments can be possibly not contiguous (they can be overlapping but also separated by "blank" or "unlabeled" segments), introduces some difficulty. One way to deal with such kind of stream is to introduce a dedicated label NL (for no label) to make the labeling covering the whole time axis. An example of the SLTSS we are working with is given in Figure 2. By convention, a NL segment starts at the end of the previous (disjoint) segment and end at the beginning of the following (disjoint) segment.

Fig. 3. Matching situation involving a ground truth A labeled segment aligned with a predicted A labeled segment.

The main difference with the Levenstein’s distance is the local matching cost that is defined for \( \delta_{SLTSS} \) as:
$C_m(S_1(i), S_2(j)) = \begin{cases} 
\infty & \text{if time intervals } [t_{bi}; t_{ei}] \text{ and } [t_{bj}; t_{ej}] \text{ are disjoint}, \\
\infty & \text{if } l_i \neq l_j \text{ and } (l_i = \text{NL} \text{ or } l_j = \text{NL}) \\
\text{and } [t_{bi}; t_{ei}] \text{ and } [t_{bj}; t_{ej}] \text{ overlap} \\
(C_0 \text{ if } l_i \neq l_j \text{ and } l_i \neq \text{NL} \text{ and } l_j \neq \text{NL} \\
\text{and } [t_{bi}; t_{ei}] \text{ and } [t_{bj}; t_{ej}] \text{ overlap (substitution),} \\
1.0 - \frac{\min(t_{ei}, t_{ej}) - \max(t_{bi}, t_{bj})}{\max(t_{ei}, t_{ej}) - \min(t_{bi}, t_{bj})} & \text{otherwise (correct match)} 
\end{cases} 

(2)

Basically, the local matching cost is evaluated according to the following cases:

- it is infinite if the segments do not overlap or if the labels are distinct with one being equal to NL (we prohibit the matching of non overlapping segments or NL labeled segments with non NL labeled segments),
- it equals $C_0$ if labels are different and distinct to NL,
- otherwise it reduces to $1.0$ minus the degree of overlapping of the two segments. Hence, if the two segments are equal (fully overlapping), the cost is null, and if the segments share a single sample, then the cost is $1.0$.

Figure 3 shows an example of a partial matching with an overlap that will produce a matching cost verifying $0 < C_m < 1.0$. The $\infty$ cost penalty prohibits the matching of non overlapping segments or segments that are differently labeled. Hence, the substitution operation (switching a label by another) is only allowed if the segments overlap, with a cost of $C_0$.

To favor the matching operation over the substitution, insertion and deletion operations, the local penalty cost $C_0$ is set up to $2.0$, the double of the matching cost with overlap worse case.

In addition, the average latency estimation (LAT) and the average match duration (DUR) can be estimated. When a match is detected, the latency and the duration measures of the matched segments are defined accordingly to Figure 3. The latency is defined as the difference of the matched mid-segment time locations, while the match duration is the length of the overlap between the two matched segments.

The evaluation of the errors is performed by back-tracing the best path provided by the recursive equation 2. This process is a bit tricky in the presence of repetitions or multiple errors. Repetitions as exemplified in figure 4 and possibly multiple errors such as the double errors depicted in figure 5 will produce false positive (FP) and false negative (FN) errors. More precisely one of the repetitions will correspond to a correct match operation while the other occurrences
of the repetition will correspond to a deletion (or insertion) operation and will be accounted for FP errors. Similarly, in the presence of multiple errors on a set of overlapping segments, one of the error will be accounted for 1 FN and 1 FP (substitution), while the other errors will be accounted for 1 FP error each. For instance, in figure ??, the first error will account for 1 FN and 1 FP (substitution A -¿ B), the second error will account for 1 FP.

The implementation of $\delta_{SLTSS}$ is described in a simplified form in Algorithm 1, which is decomposed in three blocks depicted in Algorithms 2, 3 and 4. Following are the conventions used to described these algorithms:

- the vector or matrix indexes start from 0: basically the first element of a vector $V$ is located at $V[0]$. The same applies for matrix elements.
- If $L = \{l_1, l_2, ..., l_{|L|}\}$ is the set of labels (with NL \notin L), $l_i$ will correspond to the $i^{th}$ row or column of the confusion matrix (referred to as CM in the algorithms). $idx(l) = i > 0$ will correspond to the index of label $l$ in the CF matrix ($idx(l_i) = i$).
- row and column at index 0 in the CM matrix will correspond to NL (no label).

Algorithm 1 returns a confusion matrix (obtained by back-tracing the best alignment provided by the dynamic programming algorithm), the number of repetitions, the average match duration $DUR$, the average match latency $LAT$.

From this confusion matrix it is easy to evaluate assessment measures such as:

- Micro Average Accuracy (MAA): $1/|L| \sum_{i=1}^{L}(TP_i + TN_i)/(TP_i + FP_i + TN_i + FN_i)$,

1 The code will be made available for the community at the earliest feasible opportunity.
Algorithm 1 $\delta_{SLTSS}$: alignment of a ground-truth ($S_1$) and a predicted ($S_2$) labeled segment sequences

1: procedure $\delta_{SLTSS}(S_1, S_2)$
2: Double $D[|S_1| + 1][|S_2| + 1]$; \(\triangleright\) the distance matrix
3: Integer $CF[|L| + 1][|L| + 1] = Zeros(|L| + 1, |L| + 1)$; \(\triangleright\) the confusion matrix
4: Integer $match\_count = 0$;
5: for $i = 0$ to $|S_1|$ do $D[i][0] = i \cdot C_0$; \(\triangleright\) $D$ initialization
6: for $j = 0$ to $|S_2|$ do $D[0][j] = j \cdot C_0$;
7: // $D$ Calculation
8: for $i = 1$ to $|S_1|$ do
9: for $j = 1$ to $|S_2|$ do
10: $D[i][j] = \min\{D[i - 1][j] + C_0, D[i][j - 1] + C_0, D[i - 1][j - 1] + C_m(S_i(i), S_j(j))\}$;
11: // BACK-TRACE
12: $i = |S_1|$, $j = |S_2|$;
13: $REP = 0$, $DUR = 0$, $LAT = 0$;
14: while $i > 0$ and $j > 0$ do
15: $m = \min(D[i - 1][j], D[i][j - 1], D[i - 1][j - 1])$;
16: switch($m$)
17: case ($D[i - 1][j - 1]$): MATCH/SUBST\_BLOCK;
18: break;
19: case ($D[i][j]$): DELETE1\_BLOCK;
20: break;
21: case ($D[i - 1][j]$): DELETE2\_BLOCK;
22: break;
23: case ($D[i - 1][j - 1]$): MATCH/SUBST\_BLOCK;
24: break;
25: $DUR = DUR/match\_count$;
26: $LAT = LAT/match\_count$;
27: return $CF$, $REP$, $DUR$, $LAT$;

Algorithm 2 MATCH/SUBST\_BLOCK

1: if $S_i(i)$ and $S_j(j)$ have the same label ($l_i(i) = l_j(j)$) then \(\triangleright\) correct match
2: $match\_count + 1$;
3: $DUR = DUR + overlap(S_i(i), S_j(j))$;
4: $LAT = LAT + latency(S_i(i), S_j(j))$;
5: $CF[idx(l_i(i))][idx(l_j(j))] + 1$; \(\triangleright\) $idx(l_i(i))$ is the raw or column index for $l_i(i)$
6: $i = i - 1$, $j = j - 1$;

Algorithm 3 DELETE1\_BLOCK

1: if $S_j(j) \neq NL$ then \(\triangleright\) false negative error
2: $CF[0][idx(l_j(j))] + 1$;
3: else NL deletion in GT
4: $j = j - 1$;

- the Micro Average Precision (MAP): $1/|L| \sum_{i=1}^{L} TP_i / (TP_i + FP_i)$,
- the Micro Average Recall (MAR): $1/|L| \sum_{i=1}^{L} TP_i / (TP_i + FN_i)$,
Algorithm 4 DELETE2 BLOCK

1: if $S_2(j) \neq NL$ then ⊿ false positive error
2: $CF[idx(l_1(i))] [0] + +;$
3: else NL deletion in PRED
4: $i = i - 1;$

- Micro Average F1 (MAF1): $MAF1 = 2.MAP.MAR/(MAP+MAR)$.

The algorithmic complexity of $\delta_{SLTSS}$ is clearly $O(|S_1|.|S_2|)$, or simply $O(N^2)$ for sequences of length $N$.

4 Example

Table 1 presents a simple example based on a pair of short sequences. The set of admissible labels is $L = \{1, 2, 3, 4, 5\}$. NL is the "no label" extra label. Table 2 present the back-trace process output along a best alignment path. Finally, table 3 gives the confusion matrix provided by the backtracking. Clearly, if we except the NL label, three correct matches have been detected, 1 label of the ground truth has not been predicted (label 3), which results in a false negative error, 1 mismatch (substitution) has been detected between labels 2 (GT) and 4 (PRED) and 1 prediction has led to a false positive error (label 5 has been repeated in the predicted sequence). The "best" alignment provided by the algorithm is given in Figure 6.

<table>
<thead>
<tr>
<th>index</th>
<th>Label</th>
<th>$t_b$</th>
<th>$t_e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NL</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>NL</td>
<td>31</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>NL</td>
<td>51</td>
<td>88</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>50</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>NL</td>
<td>91</td>
<td>95</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>96</td>
<td>106</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>107</td>
<td>152</td>
</tr>
<tr>
<td>7</td>
<td>NL</td>
<td>153</td>
<td>174</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>175</td>
<td>195</td>
</tr>
<tr>
<td>9</td>
<td>NL</td>
<td>196</td>
<td>203</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>204</td>
<td>254</td>
</tr>
</tbody>
</table>

On this example, the mean latency and mean duration have been respectively evaluated to 9.16 and 35 time units.

5 Conclusions

The assessment of temporal pattern labeling or recognition tasks in streams is not trivial. It requires to align sequences of labeled time-stamped segments
Table 2. Back-trace of the alignment process. PRED means predicted sequence, GT means ground truth sequence. The local alignments are given line by line and each segment is presented as: sequence_index (label | t_b − t_e)

<table>
<thead>
<tr>
<th>GROUND TRUTH</th>
<th>PREDICTION</th>
<th>EVENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 (1</td>
<td>204-254)</td>
<td>10 (1</td>
</tr>
<tr>
<td>4 (4</td>
<td>153-203)</td>
<td>9 (NL</td>
</tr>
<tr>
<td>4 (4</td>
<td>153-203)</td>
<td>8 (2</td>
</tr>
<tr>
<td>3 (2</td>
<td>102-152)</td>
<td>7 (NL</td>
</tr>
<tr>
<td>3 (2</td>
<td>102-152)</td>
<td>6 (2</td>
</tr>
<tr>
<td>2 (5</td>
<td>51-101)</td>
<td>5 (5</td>
</tr>
<tr>
<td>1 (NL</td>
<td>46-50)</td>
<td>4 (NL</td>
</tr>
<tr>
<td>1 (NL</td>
<td>46-50)</td>
<td>3 (5</td>
</tr>
<tr>
<td>1 (NL</td>
<td>46-50)</td>
<td>2 (NL</td>
</tr>
<tr>
<td>1 (NL</td>
<td>46-50)</td>
<td>1 (NL</td>
</tr>
<tr>
<td>0 (3</td>
<td>0-45)</td>
<td>0 (NL</td>
</tr>
<tr>
<td>0 (3</td>
<td>0-45)</td>
<td>0 (NL</td>
</tr>
</tbody>
</table>

Table 3. Confusion matrix provided by the back-trace of the alignment process.
GT stands for ground-truth and PR for predicted

<table>
<thead>
<tr>
<th>GT/PR</th>
<th>NL 1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL</td>
<td>1 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0 1</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>4</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>5</td>
<td>1 0 0 0 1</td>
</tr>
</tbody>
</table>

Ground-Truth

Predictition

Fig. 6. Alignment provided by the algorithm for the pair of SLTSS used in the example. In red/bold the mismatch, in orange/bold the insertion/deletion, in green the correct matches.

that either partially overlap or are disjoint. We have proposed to use an editing distance to find first an optimal alignment (basically an alignment with a minimal cost) dedicated to this specific alignment problem. Following backward this optimal alignment allows for the inventory of the various kind of mismatch (substitutions, false positive or false negative) errors as well as the correct matches. The choice of the local alignment costs has been made to favor correct label matches and to penalize mismatched labels.

This algorithm provides a confusion matrix from which standard evaluation measures can be easily derived. Furthermore, it evaluates the average latency and the relative average duration of the correct matches, which can be useful to...
"rank" applications that require pattern spotting and recognition (e.g. speech or gesture recognition applications) as soon as possible, or a precise temporal location of the labels.

This algorithm requires a ground-truth sequence in input, which is indeed a limitation to its use. In the other hand, since experts are generally involved in the construction of the ground-truth, it ensures a human control of the quality of the assessment procedure and possibly a better acceptance.

References

Recurrent Neural Networks for Modeling Company-Product Time Series

Katsiaryna Mirylenka, Christoph Miksovic, and Paolo Scotton

IBM Research – Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland
{kmi,cmi,psc}@zurich.ibm.com

Abstract. Given the increasing amount of data related to information technology (IT) inventories and products purchased by companies, it is advantageous to analyze this information and model the IT install base of companies. Such an analysis and modeling reveal latent connections between deployed IT products and helps to discover discriminative features of the IT install base structure. This allows one to efficiently compare products and companies, apply appropriate marketing strategies towards similar sets and recommend future products.

In this paper we verify the temporal correlation of product time series. We formalize the notion of a company in terms of its IT install base and study the applicability of language modeling techniques emerging in natural language processing to the task of company-product modeling. The analysis is done using $n$-gram models and Recurrent Neural Networks (RNN) over a corpus of more than 800k companies. We assess various configurations of the modeling techniques and select the best model using the perplexity measure.

The results of this study demonstrate that RNN with one hidden layer and product embeddings of size 200 is optimal for the company IT install base representation in terms of goodness of fit of the model. Additionally, RNN provides meaningful features for products, which can be used to focus marketing campaigns on specific install base structures and enhance a sales recommendation system.

1 Introduction

During the past decade, the amount of marketing intelligence data, provided by specialized companies, has been growing dramatically. This data has proven to be extremely useful to get market insights in several contexts such as, for example, new markets development or white space determination. For a given set of companies, the data provides insights into the type of IT equipment they own and how this equipment is distributed across their physical locations. Usually the data also contains timestamps related to the equipment acquisition and the confidence of its presence.

In this work, we first demonstrate the sequential property of our data by means of hypothesis testing. Then, we evaluate different architectures of Recurrent Neural Networks (RNN) for the task of company-product modeling. We
choose the best model in terms of goodness of fit or best predictive power. This model is used to extract product representations that can be applied for similarity search queries and marketing recommendations.

The main contributions of this work are the following:
1. We formalize the problem of modeling company install bases considering time series of product appearances, such that state-of-the-art unsupervised techniques from Natural Language Processing (NLP) can be applied.
2. We experimentally verify the sequential nature of the product appearance time series by applying statistical hypothesis testing based on the binomial distribution.
3. We assess the applicability of language modeling via RNN to our data. We demonstrate that RNN with one hidden layer and product embeddings of size 200 fits the data best and, additionally, provides discriminative embeddings of products.

The paper is organized as follows. In Section 2, we show the details of the available competitive install base data and provide a formalization of our problem. Section 3 discusses related approaches and their applicability. The application of RNN is presented in Section 4. Finally, Section 5 provides the comparison between different RNN architectures and discusses learned product embeddings.

## 2 Preliminaries and Problem Formalization

As a base for this work, we rely on the install base information provided by HG Data Company, Inc. [4].

For each company assessed, the following information is provided: the type of IT products available at each site of the company\(^1\), the confidence of the information, and first confirmation date of the product presence. Product descriptions are organized in a hierarchical fashion containing three levels. The highest level of the hierarchy contains *Category Parents*, giving a high-level description of the product type, for example, “Data Center Solution” or “Hardware (Basic)”\(^2\). The *Categories* level contains finer-grained descriptions, such as “Printers” or “Midrange Computers”. Finally, each *Category* contains the *Products*, which constitutes the lowest level.

In this study we chose to focus on the *Category* layer. Therefore for each company we consider the product categories\(^2\) associated with a company. In our version of the HG Data Company database, there are 91 distinct categories. Out of those categories, we restrict our study to hardware and low-level hardware-management software categories. We end up with 38 distinct categories.

We create our corpus for model training based on a set of product category appearance time series, which will be referred to as product time series in the remainder of this paper.

\(^1\) More precisely, the database reveals whether a given product type is present, but without providing quantities.

\(^2\) In the remainder of the paper, we use the terms products and product categories interchangeably, meaning product categories.
More formally, we consider a set of \( N \) companies represented in the HG Data Company database \( C = \{c_0, \ldots, c_{N-1}\} \). Each company \( c_i \) has a time series of products \( A_i \) of length \( k_i \). These products belong to the set of all possible products \( A = \{a_0, \ldots, a_{M-1}\} \). That is:

\[
\forall c_i \in C : c_i \mapsto A_i = (a_{i_0}, \ldots, a_{i_{k_i}-1}) \subset A.
\]

The products from \( A_i \) are sorted by the time of their appearance in the IT install base of a company.

Given this formalization, the goal of this paper is to model company-product time series to provide recommendations about the set of possible future products. Additionally, we would like to learn the most representative features of products \( B \), based on their temporal proximity. The features should be representative in terms of goodness of fit of the generative model of company-product data.

**Sequential nature of the data.** To assess correlations between the consecutive values of product time series, we use statistical hypothesis testing. We verify whether the frequencies of product \( n \)-grams are statistically significantly higher than in the case of independent (1-gram) product observations. Therefore, the significance test depends on the actual frequencies of the individual products, the number of observed \( n \)-grams, and the frequencies of \( n \)-grams. The test statistic shows whether the frequency of an \( n \)-gram could be generated randomly from 1-grams.

In this random 1-gram model, every symbol in the sequence is drawn independently from a categorical distribution. The probabilities of the individual symbols can be estimated from their occurrences in the company-product time series. We calculate the frequencies of each product \( Fr(a_1), Fr(a_2), \ldots, Fr(a_M) \) and the corresponding probabilities \( Pr[a_i] \approx Fr(a_i)/T \), where \( T \) is the total number of products over all company product time series.

The random generation is connected to the fact that all products of a sequence are temporally independent and distributed as Bernoulli trials with the corresponding probabilities observed in the time series. We verify whether the frequency of a sequence \( X = (x_1, x_2, \ldots, x_k) \) in this case is equal to \( Pr[X] = Pr[x_1, x_2, \ldots, x_k] = Pr[x_1] \cdot Pr[x_2] \cdot \ldots \cdot Pr[x_k] \). The distribution of a frequency of a sequence \( X \) is binomial with the parameters \( T \) and \( Pr(X) \):

\[
Fr_X \sim B(T, Pr(X)).
\]

In consequence, the frequency \( Fr_X \) of \( X \) is a random variable with known distribution that is based on the properties of the dataset (frequencies of the products), exactly as required for significance testing. We formulate the hypothesis as follows:

- \( H_0 \): null hypothesis – the frequency \( Fr_X \) of a \( k \)-gram is not significant, and the \( k \)-gram does not have a sequential nature.
- \( H_1 \): alternative hypothesis – the frequency \( Fr_X \) of a \( k \)-gram is significant. It is unlikely that the frequency comes from the random i.i.d case.

After setting the significance level \( \alpha \), which corresponds to the probability of rejecting \( H_0 \) when it is true, we test the hypothesis by checking the following \( p \)-value:

\[
p\text{-value} = Pr[B(T, Pr(X)) \geq Fr_X].
\]
If $p_{\text{value}} \leq \alpha$, the null hypothesis $H_0$ is rejected, and the frequency of $k$-gram is considered to be statistically significant. After applying the hypothesis testing in Section 5, we demonstrate that our company-product time series have strong temporal correlation, rejecting therefore $H_0$.

3 Related work

Several approaches for company-product modeling have been proposed and aim at product recommendations. The first group of methods includes various co-clustering algorithms, for example the PaCo algorithm [13] and the OCuLaR algorithm for overlapping co-clustering [3]. These algorithms do not consider the sequential nature of data and are therefore not related to our work.

Of the sequential algorithms applicable to our problem, we can cite various ‘heavy hitter approaches’, such as association rules and conditional heavy hitters [10], [8]. These algorithms are able to model time series in a Markovian fashion to produce product predictions, but they are not able to build hidden structures of the IT install base of companies. Moreover, they do not provide high-dimensional product representations.

The third group of techniques comes from the NLP domain. One of the key tasks in NLP is language modeling. The goal is to learn representations for hierarchies of concepts, starting from words, phrases, and sentences, to more sophisticated concepts, such as documents and topics. Inspired by our initial results in modeling company-product data with Latent Dirichlet Allocation [9], we decided to consider other approaches from the NLP domain. In recent years, a lot of research has been devoted to the advancement and improvement of language processing methods, including Deep Neural Network (DNN) approaches. We assume that our company install base model consists of the following layers: a layer of companies, a layer of product categories and a hierarchy of latent structures inside the install base. Given this assumption, these company layers can be mapped to NLP concepts for application in DNN methods. Considering NLP terminology, we associate companies with documents and product categories with words. The vocabulary of words consists of the product categories in our scope. All companies that we consider in our analysis form the corpus of company documents. We further assume that products or product embeddings can be grouped into latent topics, which then construct specific and discriminative features $\mathcal{B}_i$ for each company $c_i$, $0 = 1, 2, ..., N - 1$.

Currently DNNs are the core of the state-of-the-art techniques for the tasks related to NLP. Mikolov et al. [7] [6] use a simplified architecture of neural networks that allow the use of very large training datasets and the building of accurate word embeddings in Euclidean space of high dimensionality in a very efficient manner. The word embeddings can afterwards be used directly for clustering without any transformation or aggregation as features. They can also be aggregated to represent a document as a vector in a smaller space using, for example, the Fisher Kernel Framework (probabilistic modeling of the cor-
pus of documents using a mixture of Gaussians [5]) similarly as described by Clinchant et al. [2].

If labeled data is provided, classification can be done with a convolutional neural network, where both embedding vectors are refined and the classifier is learned similarly as in the approach proposed by Severyn et al. [11]. As our problem is unsupervised, we focus in the direction of unsupervised DNN, such as learning of word embeddings and generative text modeling using Recurrent Neural Networks. RNN achieve state-of-the-art performance on language modeling, which is what we are interested in, and on the tasks of speech recognition and machine translation [14]. This technique applies a distinct view on the data flow, using information not about independent instances but taking into account the sequential nature of data. Therefore, recurrent networks are sensitive to the past inputs and can adapt to them. State-of-the-art performance on language modeling task is achieved by RNN with Long Short-Term Memory (LSTM) units with the dropout regularization method [14]. We will apply this method to our company-product modeling using a time series input, \( A_i \), and analyze its performance.

4 Solution Approach

In the general case of recommender system building, the goal is to learn the parameters of a scoring function \( f(c_i, a_j) \). The function \( f(\cdot) \) takes as input a company \( c_i \), its current product time series \( A_i \) and a possible new product \( a_j \), and provides the score of recommending the product \( a_j \) for company \( c_i \). This scoring function can be linear, logistic, a multilevel neural network or can belong to another family of functions. The domain of values of a scoring function can be binary, real numbers or it can provide a probability that the offering will be successful in the future with a certain set of company and product parameters. If historical data about all the features of a company and its offerings is available, several time slices of data can be used to do supervised learning of the parameters of a scoring function. As in our case we do not yet have enough data from our internal sources for supervised learning, we focus on the methods of unsupervised learning. Once we have additional historical and labeled data in the future, we can use these representations of companies and products as an input to supervised learning techniques.

In the current setting, we can model and learn the semantic information about companies in terms of their IT install base, which is based on the fact that similar products should be close in the \( L \)-dimensional space, where \( L \) is the size of the product embeddings.

In this work, we consider two types of unsupervised models: \( n \)-gram modeling and RNN-based modeling. The modeling methods are evaluated using the measures of goodness of fit of a model.

Model adaptation and parameter estimation. For RNN we use various architectures as parameters of the model. We select the parameters by minimizing the perplexity level of a model. The average per-product perplexity is
calculated on a test set using the product time series \( A_i \) with the total number of products \( n \) from the test set. Perplexity\(^3\) shows how well the probability distribution, defined by a model, predicts testing data and is calculated as follows:

\[
\text{Perplexity} = \exp \frac{1}{n} \sum_{i=1}^{n} \ln P(a_i),
\]

where \( P(\cdot) \) is the probability distribution that is induced by the model. The lower the perplexity, the better the model. The embeddings of products \( B \) are computed using the RNN models with the lowest perplexity. Instead of the original nominative time series \( A_i \), a company now can be represented via product embeddings trained on RNN. We also use the perplexity to assess \( n \)-gram models.

5 Experimental evaluation

The experiments are done for 860k companies and 38 product categories\(^4\). The companies belong to 83 industries, such as “Health Services”, “Agricultural Services” and others.

First, we estimate the percentage of statistically significant bi- and trigrams to check the sequential nature of our data. As described in Section 2, we use a significance level \( \alpha = 0.05 \). For \( T = 4 \times 10^6 \), 3650k bigrams and 3249k trigrams, we found that 69% of the bigrams and 43% of the trigrams have \( p_{value} < \alpha \), therefore leading to the rejection of the null hypothesis. This means that these bi- and trigrams are statistically significant showing the clear sequential nature of our data.

Second, we estimate perplexity values of both the \( n \)-gram and the RNN model for our data. We use 70% of the initial corpus for training, 10% for parameter validation and 20% for model testing. As a baseline, we calculate the perplexities of ‘naïve’ initial times series representations, using 1-gram, bi- and trigram models. The resulting perplexities are correspondingly equal to 19.5, 16.4 and 15.5. In addition to the hypothesis testing result, we observed that longer contexts lead to better perplexity of \( n \)-gram models.

We use different architectures of RNN models by varying the number of hidden layers and the number of LSTM nodes, which is equivalent to the embedding size. We have tested 12 architectures of RNN with the number of layers \( N_{\text{layers}} = \{1, 2, 3\} \) and the size of product embeddings \( N_{\text{embed}} = \{10, 100, 200, 300\} \). We used the RNN model implementation from the ‘tensorflow’ package [1]. Training is done for 14 epochs over the training data. The resulting perplexity values for each RNN architecture are shown in Figure 1.

---

\(^3\) We use the terms perplexity and average perplexity per product interchangeably.

\(^4\) The full list of product categories is available at: \[http://www.hgdata.com/Technologies-We-Track\].
The lowest perplexity achieved throughout all the assessed RNN models is 11.6 for the test set, which corresponds to 1 hidden layer and an embedding size of 200. The perplexity is better than the baseline values. The RNN-based representations of products learned by the best model are represented in Figure 2 using t-SNE [12] 2D projections.

In this figure, we see certain intuitive “clusters”, such as, for example, “collaboration - communication technologies - commerce and contact center” or “mainframes - midrange”. As future work, the quality of this product representation will be evaluated in the context of a recommendation system.

\[\text{We shortened the original names of product categories for the sake of better visualization. ‘HW’ stands for hardware, ‘SW’ for software, ‘OS’ for operating systems.}\]
6 Conclusions

We have demonstrated that unsupervised modeling techniques borrowed from the NLP domain are able to capture intrinsic hidden install base structures in our company-product time series. The predictive capabilities are demonstrated by the resulting perplexity. To the best of our knowledge, this is the first time such techniques are used in this context. We have evaluated different model architectures and demonstrated that RNN with 1 hidden layer and an embedding size of 200 fits our data best.

As future work, we will assess other deep neural network architectures starting from lower levels of product descriptions. This presents the challenges of a much larger data set, which, in turn, is useful for training more sophisticated models. Another important work is to validate the results based on the outcome of a marketing recommendation tool. We will also compare the efficiency of RNN-based representations with the state of the art recommender system approaches, such as matrix factorization algorithms.

References


A time series two-sample test based on comparing distributions of pairwise distances

Pablo Montero-Manso and José A. Vilar
Research Group on Modeling, Optimization and Statistical Inference (MODES), Department of Mathematics, Computer Science Faculty, University of A Coruña

Abstract. A new test statistic based on comparing empirical distributions of pairwise distances is proposed to check the homogeneity of two groups of time series. By considering distributions instead of specific features of the pairwise distances, the proposed method facilitates the identification of a suitable time series distance in order to increase the discriminatory power of the test. An extensive simulation study shows the performance of the proposed test compared to test statistics based on other distance features such as means or nearest neighbor.

1 Introduction

This paper deals with the problem of testing homogeneity of generating processes for two sets of observed time series. Specifically, let \( \{X_1, \ldots, X_n\} \) and \( \{Y_1, \ldots, Y_m\} \) be \( n \) and \( m \) independent realizations of real-valued processes \( X \equiv \{X_t : t \in \mathbb{Z}\} \) and \( Y \equiv \{Y_t : t \in \mathbb{Z}\} \), respectively. Our goal is to test the null hypothesis of equality of \( X \) and \( Y \), i.e. these time series exhibit the same behavior. Assuming equal length \( T \) for all observed series and denoting by \( F_X \) and \( F_Y \) the \( T \)-dimensional distributions of random vectors \( (X_1, \ldots, X_T) \) and \( (Y_1, \ldots, Y_T) \), respectively, the hypothesis test can be formally stated as

\[
\begin{cases} 
H_0 : F_X = F_Y \\
H_1 : F_X \neq F_Y 
\end{cases}
\] (1)

Therefore, the problem is initially posed in terms of a two-sample test for equal distribution in high dimension. Note that two-sample problems involving partial realizations of time series frequently arise in applications from different fields, ranging from financial settings, such as testing for Monday effect in stock markets [11], to genetics [16] or brain signal analysis [13]. Nevertheless, classical multivariate approaches to handle (1) could face serious drawbacks in a time series framework due to the specific characteristics associated to series. Overall, dimensionality is much greater than the sample size (\( p \gg n \)). Frequently, the observed time series have different length. In many problems the interest lies in the temporal behavior while differences in terms of location or dispersion tend to be considered as nuisance factors, thus series are normalized prior to analysis. Other relevant issues such as phase difference tend to be ignored too. In short, this kind of particularities limit the effectiveness of commonly used multivariate
methods, requiring specific time series approaches, as can be seen in related problems such as time series clustering or classification.

Likewise distances or dissimilarities between time series objects play a key role in many successful methods of time series classification [10], intuitively one would also expect that knowledge on the distances between time series should be useful to address the two-sample problem (1). In fact, distances are also the base of successful nonparametric multivariate two-sample tests, (mainly because they are suited for the $p \gg n$ scenario) [3, 6, 9, 17], and an increasing research effort has been put on generalizing the class of distances that may be used with these methods [9, 15, 18]. We argue that different distances are expected to perform better than others, e.g. the supremum/infimum norm outperforms the Euclidean norm in multivariate scenarios where the difference lies only in the location of one of the variables (see Section 3.2 in [7]).

These evidences and the fact that distances are often used to convey invariance to time series nuisance factors (see [2] for a review of invariance types required in several time series domains) directed us to study the behavior in two-sample problems of different combinations of distances and test statistics. There are two main contributions in this work. First, a new test statistic based on comparing the empirical distributions of pairwise distances between series coming from the same sample and series from the different samples is proposed. The statistic takes advantage of considering the whole distributions instead of only a few specific moments, and exhibits a good performance in different simulation scenarios compared to other alternative test procedures. Second, supported by an extensive numerical study, it is also shown the effect of choosing suitable distances to increase the power of the test. In particular, it is observed that the proposed method produces good results for a large set of distances.

2 Two-sample distance based methods

The methods to be compared are all based on distances. Other methods that do not incorporate distances such as [19] are expected to underperform when extra domain knowledge is introduced through a distance.

The nearest neighbors [8] method is selected among variants and similar methods such as [7] due to its direct association with the k-nearest neighbors method used in classification, which makes easier its interpretation. The so-called “energy” two-sample test [17] is one of the most popular distance-based two-sample methods and its core concept is also used to develop a dependency test, all part of the class of $\varepsilon$-statistics. Including the energy test is especially interesting due to its similarity with the proposed method, both of them based on distributional properties of interpoint distances. A short description of these procedures is given below.

Nearest neighbors. This test, introduced in [8], is based on counting, for each element $i$, the amount of elements belonging to the same sample as $i$ among its $r$ nearest neighbors.
Let \( \mathcal{Z} \equiv \{ \mathbf{Z}_i, i = 1, \ldots, n + m \} \) be the pooled sample including all \( n + m \) series \( \mathbf{X}_1, \ldots, \mathbf{X}_n \) and \( \mathbf{Y}_1, \ldots, \mathbf{Y}_m \). Define \( I_i \) as the indicator function such as 
\[ I_i(r) = 1, \text{ if } \mathbf{Z}_i \text{ and the } r\text{-th nearest neighbor to } \mathbf{Z}_i \text{ belong to the same sample, and } I_i(r) = 0 \text{ otherwise}. \]

The statistic of the nearest neighbors test is given by
\[
T = \sum_{i=1}^{n+m} \sum_{j=1}^{r} I_i(j)
\]

The statistic \( T \) has an asymptotic normal distribution, but in practice parameters of the distribution are difficult to obtain analytically [17], thus the null distribution of the statistic is approximated via the permutation procedure. Under the alternative hypothesis, values of the statistic are expected to be greater than under the null, so the test rejects the null for large values of the statistic.

The restriction to use the Euclidean distance is removed by Henze [9] so other distances can be used. The test depends on the selection of the \( r \) parameter, failing to achieve nominal significance levels in some scenarios, as reported in [17] and confirmed in our experiments. It also assumes continuity of distributions. Weighted versions of the statistic have been considered to tune the sensibility of the test against certain alternatives [14].

**Energy test.** This test is based on Euclidean interpoint distances between different samples and within the same sample, proposed by Székely and Rizzo [17] and by Baringhaus and Franz [1]. The term energy comes from the similitude with Newton's gravitational potential energy. The statistic takes the form
\[
E = \frac{nm}{n+m} \left( \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \| \mathbf{X}_i - \mathbf{Y}_j \| - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \| \mathbf{X}_i - \mathbf{X}_j \| - \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \| \mathbf{Y}_i - \mathbf{Y}_j \| \right)
\]
with \( \| \cdot \| \) denoting the Euclidean norm. The restriction of Euclidean distance is later reduced to the class of continuous negative definite functions [18].

The null distribution of the statistic is calculated using the permutation procedure and the test rejects the null for large values of \( E \).

It is worthy to point out the strong similarity between the energy statistic and the kernel maximum mean discrepancy (kernel MMD) statistic proposed by Gretton et al. [6], which is based on comparing mean kernel embeddings of distributions. Sedljenovic et al. [15] state the equivalence between distance-based and kernel-based statistics for the two-sample problem.

### 3 A new two-sample test based on pairwise distances

Let \( b_{ij} \) be the interpoint distances between samples, \( b_{ij} = \| \mathbf{X}_i - \mathbf{Y}_j \| \), \( 1 \leq i \leq n, 1 \leq j \leq m \), and \( w_{1i}, \ldots, w_{ni}, w_{n+1,i}, \ldots, w_{n+m,n+i} \) be the joint interpoint distances within samples, \( w_{ij} = \| \mathbf{X}_i - \mathbf{X}_j \|, 1 \leq i, j \leq n \) and \( w_{ij} = \| \mathbf{Y}_{i-n} - \mathbf{Y}_{j-n} \|, n+1 \leq i, j \leq n+m \). The value of the proposed statistic \( D \) is the Cramer-von Mises type statistic between the empirical distribution.
functions of \( b \) and \( w \). In the pooled sample of \( b \) and \( w \), define \( r_1, \ldots, r_{nm} \) as the ranks of \( b \) and \( s_1, \ldots, s_{n^2+m^2} \) as the ranks of \( w \), then the statistic is defined as:

\[
D = \frac{U}{nm(n^2 + m^2)(nm + (n^2 + m^2))} - \frac{4nm(n^2 + m^2) - 1}{6(nm + (n^2 + m^2))},
\]

with

\[
U = nm \sum_{i=1}^{nm} (r_i - i)^2 + (n^2 + m^2) \sum_{j=1}^{n^2+m^2} (s_i - i)^2.
\]

The null distribution of \( D \) is obtained via the permutation procedure.

As mentioned, there is a strong similarity between the energy statistic and the proposed statistic, \( D \). To motivate our approach, an intuition of the difference in performance between the energy statistic and the proposed method is given.

In some cases, the test statistics based on the means of within- and between-groups interpoint distances can perform worse than the ones based on the whole distributions of these distances. Figure 1a shows within- and between samples interpoint densities for series generated from AR(0.3) and AR(0.8) processes. Densities instead of distributions are depicted for a clearer view of the effect.

The energy statistic compares the average of the means of the within-groups distances against the mean of the between-groups distances. In some scenarios such as the one presented here, the value of this statistic may not be large enough to reject the null hypothesis, and larger sample sizes would be required. Even though the mean-based energy statistic is small, comparing interpoint densities can produce better discriminatory power. This effect is illustrated in Figure 1b, where the within-groups distances are joint. The energy statistic essentially compares the means of these two densities while our method directly compares the distributions. It can be seen that when comparing against between-groups distances, joining withinA and withinB distances may result in an very small mean difference while the densities difference can still be significative.

Scenarios where the mean-based approach works better than the distribution method can indeed appear, for instance when sampling distributions only differ in location. In these scenarios, the energy statistic outperforms the \( D \) statistic in the same way that a two-sample mean test is superior to a general distribution test when only the means are different.

When we introduce a complex distance, the resulting interpoint distances may separate groups in more ways than the different location scenario, so a test robust against this possibility is desirable, even at the cost of discriminatory power in the previously mentioned scenario.

4 Simulation study

In this section, the empirical power of the nearest neighbors test with parameter \( r = 3 \) (3-NN), the energy test (Energy) and the proposed test based on the distributions of interpoint distances (D-D) are compared under the following scenarios:
A time series two-sample test based on distributions of distances

Fig. 1: (a) Densities of within and between samples Euclidean distances for AR(0.3) (A) and AR(0.8) (B) processes. (b) Comparing joint within group and between group interpoint distances densities with the means used by the energy statistic.

- E1: First order autoregressive processes with coefficients -0.9, -0.6, -0.3, 0, 0.3, 0.6, 0.9. In all cases, the error term follows a standard normal distribution, the length of the series is 50, and the distance used is the Euclidean.
- E2: Practitioners may be only interested in the temporal behavior of the series, thus they standardize data to have mean 0 and variance 1. In order to measure the sensibility of the tests in this setting, we have considered the same experiment as E1, but adjusting the models to have the same marginal variance.
- E3: The third experiment is conducted to assess the effect of using different dissimilarities specifically designed for time series. Besides considering particularly interesting comparisons of AR(1) models, some GARCH models are also included to examine more complex dependence structures.
- E4: Real data examples taken from the UCR Time Series archive [5] using different dissimilarities. The selected datasets cover a range of series’ lengths from 96 to 720 and do not require either very small or very large sample sizes to produce meaningful results.

A detailed description of the dissimilarities used in the experiments can be seen in [12]. The dissimilarities are selected as representatives of broad groups, such as PDC for the Complexity-based dissimilarities group, AR.LPC.CEPS for the Autoregressive Model-based, CID and DTWARP for their complexity and time warping invariances, etc. Note that this study is not intended as a comparison of dissimilarities but as evidence of the benefits of introducing specialized dissimilarities in the time series context. Sample size was $n = m = 20$ for E1, E2 and E3. E4 has sample sizes of $n = m = 15$ except for the ScreenType and TwoPatterns datasets, both with $n = m = 25$. Experiments were replicated 500 times to approximate the rejection proportion. Real datasets experiments were replicated using permutations from the pooled training and testing subsets.
compared classes were always the first and second one when the datasets had more than 2 classes.

Experiments E1 and E2 are summarized in Table 1, where rejection rates at $\alpha = 0.05$ are shown. Results from E1 show that the proposed method is the best in this scenario, except for the comparison AR(-0.6) vs AR(-0.3), where 3-NN takes a slight advantage, and AR(-0.3) vs AR(0.3), where 3-NN fairly outperforms D-D. The energy test achieves less power than D-D in all considered scenarios. One representative example is the comparison of AR(0.6) vs AR(-0.3) and AR(0.6) vs AR(-0.6). Despite being more separate in their coefficients, the first scenario achieves greater power for both energy and D-D methods. This is explained by the difference of variance of the processes, greater in the first scenario. Results from E2 are free of this effect, and it is observed that the changes in variance drastically reduce the power of the tests, illustrating the difficulty of comparing normalized time series. The results of the nearest neighbor approach are superior on average to the ones obtained with the proposed test, although there exist comparisons where D-D performs better than 3-NN. The energy statistic leads to the lowest powers. Note that the rejection rates under the null are very close to the nominal size for all statistics and experiments.

Experiments E3 and E4 are summarized in Table 2, rejection rates are also for $\alpha = 0.05$. The effect of considering different dissimilarities in the normalized autoregressive scenario (AR(0.2) vs AR(0)) is especially illustrative. When using the Euclidean distance, all methods achieve very low discriminatory power. The power is greatly improved by including distances like SPEC.LLR and AR.LPC.CEPS, which measure dissimilarity in terms of spectra and cepstral coefficients, respectively. The optimal properties of these distances to deal with these models are inherited by the test statistics, resulting in higher powers. It is worthy to remark that, if the Euclidean distance is considered, the 3-NN method is the best one, but Energy and D-D statistics outperform 3-NN when SPEC.LLR and AR.LPC.CEPS are used. This result, which can be extrapolated to the whole of E2, shows that a dissimilarity may differentiate samples in ways which some tests are not sensible to. Note that energy outperforms D-D, for instance the standardization coupled with the coefficients extracted by the AR.LPC.CEPS dissimilarity creates the ideal location-separated scenario mentioned in section 3.

The D-D method shows better performance in the non-normalized GARCH scenarios. Series in the experiment marked with $'$ in Table 2 have length 200. In this scenario, 3-NN fails to achieve the nominal value, requiring fine-tuning of the parameter $r$. This effect occurs for other GARCH experiments of length 200, not shown here. In the normalized, length 1000 GARCH scenario, marked with $^*$ in Table 2, using other distances increases the discriminatory power. D-D is the method taking the most advantage of the new distances. Notably, 3-NN benefits more than Energy with SPEC.LLR, while the opposite happens with AR.LPC.CEPS.

On the real data scenarios, the use of non-Euclidean distances improves the rejection rates with few exceptions. The results also show that the effect of a
<table>
<thead>
<tr>
<th>Scenario</th>
<th>3-NN Energy</th>
<th>D-D</th>
<th>*3-NN *Energy</th>
<th>*D-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR(-0.9) vs AR(-0.9)</td>
<td>0.05</td>
<td>0.06</td>
<td>0.06</td>
<td>0.05</td>
</tr>
<tr>
<td>AR(-0.9) vs AR(-0.6)</td>
<td>0.97</td>
<td>1.00</td>
<td>0.18</td>
<td>0.09</td>
</tr>
<tr>
<td>AR(-0.9) vs AR(-0.3)</td>
<td>1.00</td>
<td>1.00</td>
<td>0.29</td>
<td>0.13</td>
</tr>
<tr>
<td>AR(-0.9) vs AR(0)</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.16</td>
</tr>
<tr>
<td>AR(-0.9) vs AR(0.3)</td>
<td>1.00</td>
<td>1.00</td>
<td>0.84</td>
<td>0.22</td>
</tr>
<tr>
<td>AR(-0.9) vs AR(0.6)</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.26</td>
</tr>
<tr>
<td>AR(-0.9) vs AR(0.9)</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.38</td>
</tr>
<tr>
<td>AR(-0.6) vs AR(-0.6)</td>
<td>0.06</td>
<td>0.11</td>
<td>0.11</td>
<td>0.06</td>
</tr>
<tr>
<td>AR(-0.6) vs AR(-0.3)</td>
<td>0.26</td>
<td>0.82</td>
<td>0.39</td>
<td>0.08</td>
</tr>
<tr>
<td>AR(-0.6) vs AR(0)</td>
<td>0.98</td>
<td>0.99</td>
<td>0.91</td>
<td>0.10</td>
</tr>
<tr>
<td>AR(-0.6) vs AR(0.3)</td>
<td>1.00</td>
<td>0.10</td>
<td>1.00</td>
<td>0.15</td>
</tr>
<tr>
<td>AR(-0.6) vs AR(0.6)</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.27</td>
</tr>
<tr>
<td>AR(-0.6) vs AR(0.9)</td>
<td>0.06</td>
<td>0.04</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>AR(-0.3) vs AR(0)</td>
<td>0.11</td>
<td>0.14</td>
<td>0.10</td>
<td>0.06</td>
</tr>
<tr>
<td>AR(-0.3) vs AR(0.3)</td>
<td>0.43</td>
<td>0.16</td>
<td>0.43</td>
<td>0.07</td>
</tr>
<tr>
<td>AR(-0.3) vs AR(0.6)</td>
<td>0.98</td>
<td>0.99</td>
<td>0.92</td>
<td>0.10</td>
</tr>
<tr>
<td>AR(-0.3) vs AR(0.9)</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.19</td>
</tr>
<tr>
<td>AR(0) vs AR(0)</td>
<td>0.07</td>
<td>0.12</td>
<td>0.10</td>
<td>0.07</td>
</tr>
<tr>
<td>AR(0) vs AR(0.3)</td>
<td>0.10</td>
<td>0.25</td>
<td>0.40</td>
<td>0.08</td>
</tr>
<tr>
<td>AR(0) vs AR(0.6)</td>
<td>1.00</td>
<td>0.52</td>
<td>0.18</td>
<td>0.64</td>
</tr>
<tr>
<td>AR(0) vs AR(0.9)</td>
<td>1.00</td>
<td>0.05</td>
<td>0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>AR(0) vs AR(0.6)</td>
<td>1.00</td>
<td>0.14</td>
<td>0.11</td>
<td>0.06</td>
</tr>
<tr>
<td>AR(0) vs AR(0.9)</td>
<td>1.00</td>
<td>0.14</td>
<td>0.11</td>
<td>0.06</td>
</tr>
<tr>
<td>AR(0.3) vs AR(0.3)</td>
<td>1.00</td>
<td>0.33</td>
<td>0.14</td>
<td>0.35</td>
</tr>
<tr>
<td>AR(0.6) vs AR(0.6)</td>
<td>0.06</td>
<td>0.04</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>AR(0.6) vs AR(0.9)</td>
<td>0.96</td>
<td>0.04</td>
<td>0.06</td>
<td>0.05</td>
</tr>
<tr>
<td>AR(0.9) vs AR(0.9)</td>
<td>0.04</td>
<td>0.06</td>
<td>0.06</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 1: E1 and E2 rejection rates results, * columns have the normalized versions of the series.

dissimilarity is not uniform across the studied methods, e.g., while all methods benefit from the PDC dissimilarity [4] in the RefrigeratorDevices dataset, the 3-NN with PDC decreases its power in the TwoPatterns dataset, when compared with the Euclidean.

The UCR time series datasets are used for classification, classes are presumed to be separable. Nevertheless the two-sample problem may include different scenarios, such as comparing groups consisting of the same two classes, but in different proportions. In the last row of Table 2, mixtures of the first classes of two datasets in different proportions are compared, showing that 3-NN method is not particularly suited for this scenario.

5 Conclusions and further research

We have shown the effect of the chosen dissimilarity when testing homogeneity of generating processes for two sets of time series. Compared to the standard mul-
tivariate approach based on the Euclidean distance, our experiments show that a well-selected time series dissimilarity can substantially improve the discriminatory power. Dissimilarities can separate samples in ways which existing two-sample methods do not take full advantage of. The proposed test compares the whole empirical distributions of pairwise distances within- and between-groups, thus providing a more complete information than simply using a number of specific moments of these distributions. Compared to other alternative statistics, our proposal attained the best performance in many of the simulated scenarios, and produced competitive results in the rest. This good behavior is expected to ease the selection of a proper dissimilarity for the purpose of testing.

The results presented here open the possibility of performing time series two-sample tests in situations where the discriminatory power is low.

Possible lines of research include studying the different ways of comparing interpoint distributions, such using nonparametric densities instead of empirical distributions, and the automatic selection of dissimilarities in the context of hypothesis testing.

### References


### Table 2: E3 and E4 rejection rates results.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Dissimilarity</th>
<th>3-NN Energy D-D</th>
<th>EUCL</th>
<th>SPEC.LLR</th>
<th>AR.LPC.CEPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(AR(0.2) vs AR(0.0))</td>
<td>EUCL</td>
<td>0.06</td>
<td>0.03</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>(AR(0.2) vs AR(0.0))</td>
<td>SPEC.LLR</td>
<td>0.62</td>
<td>0.92</td>
<td>0.83</td>
<td></td>
</tr>
<tr>
<td>(AR(0.2) vs AR(0.0))</td>
<td>AR.LPC.CEPS</td>
<td>0.76</td>
<td>0.95</td>
<td>0.91</td>
<td></td>
</tr>
<tr>
<td>(GARCH $\omega=0.2 \alpha=0.05 \beta=0.7$) vs (GARCH $\omega=0.1 \alpha=0.1 \beta=0.8$)</td>
<td>EUCL</td>
<td>0.05</td>
<td>0.07</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>(GARCH $\omega=0.2 \alpha=0.1 \beta=0.7$) vs (GARCH $\omega=0.1 \alpha=0.15 \beta=0.7$)</td>
<td>EUCL</td>
<td>0.1</td>
<td>0.27</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>*(GARCH $\omega=0.1 \alpha=0.7 \beta=0.2$) vs (GARCH $\omega=0.05 \alpha=0.65 \beta=0.15$)</td>
<td>EUCL</td>
<td>0.00</td>
<td>0.60</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>*(GARCH $\omega=0.1 \alpha=0.7 \beta=0.2$) vs (GARCH $\omega=0.05 \alpha=0.65 \beta=0.15$)</td>
<td>EUCL</td>
<td>0.04</td>
<td>0.03</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>*(GARCH $\omega=0.1 \alpha=0.7 \beta=0.2$) vs (GARCH $\omega=0.05 \alpha=0.65 \beta=0.15$)</td>
<td>EUCL</td>
<td>0.14</td>
<td>0.08</td>
<td>0.39</td>
<td></td>
</tr>
<tr>
<td>*(GARCH $\omega=0.1 \alpha=0.7 \beta=0.2$) vs (GARCH $\omega=0.05 \alpha=0.65 \beta=0.15$)</td>
<td>AR.LPC.CEPS</td>
<td>0.10</td>
<td>0.30</td>
<td>0.43</td>
<td></td>
</tr>
<tr>
<td>*(GARCH $\omega=0.1 \alpha=0.7 \beta=0.2$) vs (GARCH $\omega=0.05 \alpha=0.65 \beta=0.15$)</td>
<td>INT.PER</td>
<td>0.09</td>
<td>0.06</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>ScreenType</td>
<td>EUCL</td>
<td>0.1</td>
<td>0.1</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>ScreenType</td>
<td>INT.PER</td>
<td>0.21</td>
<td>0.09</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>RefrigerationDevices</td>
<td>EUCL</td>
<td>0.06</td>
<td>0.04</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>RefrigerationDevices</td>
<td>PDC</td>
<td>0.95</td>
<td>0.97</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>ShapletSim</td>
<td>EUCL</td>
<td>0.05</td>
<td>0.04</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>ShapletSim</td>
<td>CID</td>
<td>0.86</td>
<td>0.98</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>TscSegmentation1</td>
<td>EUCL</td>
<td>0.12</td>
<td>0.06</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>TscSegmentation1</td>
<td>DTWARP</td>
<td>0.86</td>
<td>0.73</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td>TwoPatterns</td>
<td>EUCL</td>
<td>0.45</td>
<td>0.2</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>TwoPatterns</td>
<td>PDC</td>
<td>0.15</td>
<td>0.22</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>TwoPatterns</td>
<td>SPEC.LLR</td>
<td>0.4</td>
<td>0.58</td>
<td>0.60</td>
<td></td>
</tr>
<tr>
<td>ElectricDevices</td>
<td>EUCL</td>
<td>0.1</td>
<td>0.21</td>
<td>0.48</td>
<td></td>
</tr>
<tr>
<td>ElectricDevices</td>
<td>PDC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ShapletSim &amp; TwoPatterns Mixture (0.6,0.4) vs (0.8,0.2)</td>
<td>SPEC.LLR</td>
<td>0.33</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
A time series two-sample test based on distributions of distances


Discovering relationships in climate-vegetation dynamics using satellite data

Christina Papagiannopoulou¹, Diego Miralles³,², Mathieu Depoorter², Niko E.C. Verhoest², Wouter Dorigo¹, and Willem Waegeman¹

¹ Depart. of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Belgium
christina.papagiannopoulou, willem.waegeman@ugent.be
² Laboratory of Hydrology and Water Management, Ghent University, Belgium
mathieu.depoorter, niko.verhoest, diego.miralles@ugent.be
³ Depart. of Earth Sciences, VU University Amsterdam, the Netherlands
⁴ Depart. of Geodesy and Geo-Information, Vienna University of Technology, Austria
wouter.dorigo@geo.tuwien.ac.at

Abstract. Advances in satellite Earth observation have resulted in the development of consistent global historical records of environmental and climatic variables, forming enormous amounts of multivariate time series. In this work we present a novel machine learning framework for detecting relationships between climatic time series and vegetation indices. Our pipeline consists of several components, including data fusion from various databases, time series decomposition techniques, feature construction methods and predictive modeling. Experimental results indicate that with this pipeline it is possible to detect patterns that express relationships in large-scale climate data.

Keywords: time series analysis, feature construction, shapelets

1 Introduction

Earth Observation (EO) satellite data provide a wealth of information about the dynamics of our planet in recent decades. Independent sensors on different platforms monitor vegetation, soils, oceans and atmosphere, collecting optical, thermal, microwave, altimetry, or gravimetry information. Composite records of environmental and climatic variables now span up to 35 years, enabling the study of climate-vegetation interactions over multi-decadal scales. Such records can be interpreted as long multivariate time series with different spatial and temporal resolutions. Due to their volume and complexity, the resulting datasets pose important challenges with respect to data processing and data analysis, leading to a need for developing novel methods.

Vegetation is a major player in the global climate system by affecting the water, the energy and the carbon cycles. Plants alter climate through the transfer of water vapor from land to atmosphere, direct effects on the surface net radiation, exchange of carbon dioxide with the atmosphere, or changes in roughness
length affecting wind speed and direction. Given the crucial role of vegetation in climate, and the influence of climate on vegetation dynamics, understanding how vegetation will respond to projected climatic changes is crucial to narrow down the uncertainty in the predictions of global warming. A first and necessary step, however, is to investigate the sensitivity of vegetation to past-time climate variability. Simple correlation statistics have led to important steps forward in understanding the link between climate and vegetation while considering a few datasets (e.g. [8, 13]). But to fully use this available stream of information in constant expansion, new and more sophisticated approaches are required.

In this article we present a novel framework for finding climatic drivers that affect vegetation. We will consider a data-driven approach, analyzing satellite time series that span the entire globe and three decades. In a first step, we describe how we combine data from different sources. In a second step, we reformulate the problem of discovering relationships in climate-vegetation dynamics as a machine learning problem, where vegetation is considered as the target time series, while climate information sources serve as predictor time series. We apply time series decomposition techniques to the target vegetation time series and the various predictor climatic time series to isolate seasonal cycles, trends and residuals in order to remove unwanted correlations that originate from seasonal and trend effects. Subsequently, we explore various techniques for constructing high-level features from climatic time series using techniques that are similar to shapelets [14]. We employ standard machine learning techniques, as nonlinear autoregression method, in order to search for shapelets that are predictive with respect to the residuals of vegetation time series. As shown in the experimental results, this approach allows us to discover novel insights w.r.t. climate-vegetation dynamics, moving beyond the state-of-the-art in this application domain.

2 Methodology

2.1 Data collection and fusion

Since we aim to disentangle the effect of past-time climate variability on global vegetation, data sets have been selected from the current pool of satellite and in situ observations. The environmental and climate variables are collected on the basis of meeting a series of spatiotemporal requirements: (a) to span multi-decadal records, (b) to have a global coverage, and (c) to be available at an adequate spatial and temporal resolution. All these data sets span the study period 1981-2011 at the global scale, and have been converted to a common monthly temporal resolution and 1° × 1° latitude-longitude spatial resolution. To do so, we have used averages to re-sample original data sets found at finer native resolution, and linear interpolation to resample coarser-resolution ones. Five different climatic and environmental drivers of vegetation dynamics have been considered: precipitation, temperature, radiation, snow depth (i.e. snow water equivalents) and surface soil moisture. Rather than using a single data set for each of these variables, the approach has been to collect and utilise the
largest possible number of data sets meeting the above-mentioned requirements (see Supplementary Material for details).

For vegetation, we use the satellite remote sensed products of Normalized Difference Vegetation Index (NDVI), a graphical indicator which is used to assess whether the target being observed contains live green vegetation or not. Data from the Global Inventory Modeling and Mapping Studies (GIMMS) data set has been used, which is one of the most commonly used NDVI data sets [10] covering a wide time interval of 30 years (1981-2011).

2.2 Nonlinear autoregressive model

In this paper we will analyze climate-vegetation dynamics in a machine learning setting, where for each pixel the vegetation time series will be considered as target time series, using the notation \( Y_t \), and the other time series will serve as predictor time series, using the notation \( X_t \). This will result in a regression problem with moving windows (NDVI is a continuous variable). We hypothesize that relationships between climate and vegetation are expected to be highly nonlinear, we will replace the linear Vector Autoregressive (VAR) models with nonlinear machine learning models. We have chosen to use random forests [1], a well-known nonlinear machine learning method that has shown its merits in diverse application domains. We will evaluate the performance in terms of explained variance, \( R^2 \), defined as:

\[
R^2 = 1 - \frac{RSS}{TSS}
\]

where \( RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \) is the residual sum of squares, \( TSS = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2 \) the total sum of squares, the \( y_i \) and \( \hat{y}_i \) the real and the predicted value respectively, \( \bar{y}_i \) the mean and \( n \) the number of the examples. In our analysis, we will treat each pixel on earth as a separate problem, because vegetation is assumed to have a very local pattern.

Our approach is visualized in Figure 1. For a given value of the target time series \( Y_t \) at time stamp \( t \), we investigate properties of the different predictor time series \( X_t \) by considering a moving window of the months before time stamp \( t \). Within those windows, we intend to construct higher-level features, which represent properties in each of the climate time series that are predictive w.r.t. the vegetation time series.

Before constructing those features, we first decompose the target and predictor time series into trends, seasonal cycles and anomalies. This is an important step, because the trend and seasonal component of the vegetation time series are not influenced by climatic features. In what follows we will work further with the anomalies, and the goal will be to forecast those using information from the predictor time series.

Many methods for decomposing time series have been proposed in the literature [3,11]. We decided to use an additive model without break-points, since it is

\(^5\) http://www.sat-ex.ugent.be/supplementary_material.pdf
conceptually-simple, while delivers satisfactory results in a reasonable amount of time. In a first step, time series $Y_t$ is at every pixel de-trended linearly based on the entire study period, using a simple linear regression model, $Y_t \approx \beta_1 \times t + \beta_0 = T_t$. In this way we obtain the de-trended time series, $D_t = Y_t - T_t$. In a second step, the seasonal cycle $S_t$ is estimated as a monthly expectation, taking the multi-year average for each month of the year. In a last step, the anomalies are calculated by subtracting the corresponding monthly expectation from the de-trended time series, $R_t = D_t - S_t$. The same time series decomposition method is followed for all predictor time series as well (see Supplementary material for details).

### 2.3 Feature construction from shapelets

We intend to identify patterns in the windows of Figure 1 that are predictive w.r.t. the anomalies of the target vegetation time series. To this end, we will analyze subsequences of the moving window specified for time stamp $t$, a technique that is similar to so-called shapelets [14]. Techniques for finding shapelets have been mainly applied to the problem of time series classification while various metrics are used for the evaluation of their quality [6, 7, 14].

Unlike most applications of feature selection, we are in our analysis less interested in discovering individual features. In contrast, we rather intend to know which types of climatic drivers affect vegetation in different regions of the world. The predictor variables will hence be grouped based on the type of the climate
variables. More specifically, three main categories are distinguished, namely radiation, water (including precipitation, soil moisture and snow-water equivalents) and temperature. Then, we run ridge regression and random forests separately for each group and we examine the value of $R^2$, investigating which group of predictors explains better the variability of the NDVI residuals. Each group of predictors is assessed separately because climate drivers are highly correlated and our goal is to investigate their predictive power for each region separately.

### 2.4 Overview of features extracted from shapelets

Besides the application to discovering relationships between time series instead of time series classification, another difference with shapelet construction papers is that we will not be interested in the shapelets itself, but aggregates that can be derived from them. In particular we will focus on three categories: lags, cumulatives and extreme indices. They are described below in more detail.

**Lags:** Vegetation responds to meteorological and environmental changes at different time scales. Since vegetation, soil and atmosphere have a memory, and because vegetation may require some time to adapt to environmental changes, it is necessary to explore potential lag-time responses to gain understanding of the relation between plants and their environment. While the concept of introducing lag times in the study of this relationship is not new (see e.g. [4]), it has become more extended in recent studies [2, 12]. Given the flexibility of our machine learning approach to incorporate a large number of climate-based predicting features, a large number of lag-times can be applied to the different climate variables. We experimented with time-lags covering a range of $\ell = 0, 1, ..., 12$ months (where $\ell = 0$ is the current month) for all the driving variables.

**Cumulatives:** Vegetation dynamics may not necessarily reflect the climatic conditions from (e.g.) three months ago, but the average of the (e.g.) three antecedent months. This integrated response to antecedent environmental and climatic conditions is referred here as ‘cumulative’ response. Note that, unlike in the case of lagged variables, cumulative variables include always up to present-time climate conditions.

**Extreme indices:** Over the last few years, many research studies have been performed on climate extremes [9, 17]. The fact that many daily data sets are freely available make the calculation of extreme indices easier. In recent years, 27 recommended indices related to temperature and precipitation have been developed [5, 15]. Related to the vegetation response, extremes on climate variables that cause a different behavior of the terrestrial ecosystems have been investigated [16]. In our work, we have calculated different monthly indices on the raw data as well as on the residuals (including the trend - see Supplementary material).

### 3 Experimental results

Putting all pieces of the above pipeline together, we end up with a dataset that has 5319 features generated on thirty-year time series with a monthly resolution.
We will analyze 13,097 land pixels independently, covering 130GB on disk. We use the implementation of scikit-learn for the random forest regressor, setting the number of trees equal to 100 and the maximum number of features per node to the square root of the total number of features. The model is assessed by means of five-fold cross-validation.

We apply ridge regression, Pearson correlation coefficients and a random forest-based Autoregressive model using as features the past values of the NDVI residuals (only the green moving window in Fig. 1) as baseline methods. We perform ridge regression using nested five-fold cross-validation for the tuning of the parameter $\lambda$ on the same data sets. Pearson correlation coefficients are calculated on the training sets and the feature with the highest value is selected. Then, we calculate the squared Pearson correlation value of the selected feature on the test data.

Figure 2, on the top, shows the result of the random forest model applied to the total number of features. As one can observe, the model has better predictive skill in Australia, in the bulk of Africa and in a portion of North and South America. Compared to the other three approaches, the results obtained by the random forest model are a big improvement. Ridge regression also performs well for the regions where the random forest yields a high $R^2$, in contrast to the other regions, where the map is mostly colored white. However, ridge regression in general leads to substantially worse results for almost all regions of the world. This result confirms that the relationships between the climatic variables are non-linear. The filter approach based on Pearson correlations, which is the current state-of-the-art in modelling climate-vegetation dynamics performs very
Fig. 3: Performance comparison of the three feature groups. **Blue:** Regions where water related features explain better the variance of the NDVI residuals. **Red:** Regions where temperature features outperform the other two groups. **Green:** Regions where radiation features give better result. **White:** All models give low $R^2$. Poorly. Finally, the Autoregressive model performs really low in almost all of the pixels. As such, we can conclude that an interplay between different variables is needed to model vegetation dynamics.

In Fig. 3 we classify each pixel to one of the three groups: temperature, radiation and water, according to the explained variance obtained from each group. It is clear that in regions where the performance of the total model is high, the water-related group of features give better results in comparison with the other two groups, radiation and temperature. In addition, in regions where radiation outperforms the other two groups, the performance of the total model is poor. Our results are quite consistent with the ones in [8], except for the result in rain forests. This is explained by the fact that NDVI is very constant in these regions, and thus none of the three groups (radiation, water, temperature) is, a priori, a clear winner for the explanation of this little NDVI variability that exists.

4 Conclusions

In this paper we presented a machine learning framework for detecting relationships in climate-vegetation dynamics. We used a wide collection of data and we created a unique database that bundles all publicly-available datasets with an appropriate spatial and temporal resolution. As such, we hope that this paper can inspire the machine learning and data mining communities to explore a new application domain with enormous potential for developing novel methods.

Our preliminary results are quite encouraging, and therefore in upcoming work the implications of our results from a climate and biophysical perspective will be examined. The framework that we propose also allows to answer other
questions w.r.t. climate-vegetation dynamics, such as the influence of lags, and
the quality of the various data sources. We have already obtained interesting
preliminary results in that direction, but those results are excluded due to lack
of space.

References

2. T. Chen et al. Using satellite based soil moisture to quantify the water driven
variability in NDVI: A case study over mainland Australia. *Remote Sensing of
3. R. B. Cleveland et al. Stl: A seasonal-trend decomposition procedure based on
& Row, 1984.
5. M.G. Donat et al. Updated analyses of temperature and precipitation extreme
indices since the beginning of the twentieth century: The hadex2 dataset. *Journal of
7. A. Mueen et al. Logical-shapelets: an expressive primitive for time series classifica-
tion. In *Proc. of the 17th ACM SIGKDD international conference on Knowledge
discovery and data mining - KDD ’11*, page 1154, New York, New York, USA, aug
8. R.R. Nemani et al. Climate-Driven Increases in Global Terrestrial Net Primary
9. N. Nicholls and L. Alexander. Has the climate become more variable or extreme?
10. C.J. Tucker et al. An extended AVHRR 8-km NDVI dataset compatible with
MODIS and SPOT vegetation NDVI data. *International Journal of Remote Sens-
11. J. Verbesselt et al. Detecting trend and seasonal changes in satellite image time
discovery and data mining - KDD ’09*, page 947, New York, New York, USA, jun
15. X. Zhang et al. Indices for monitoring changes in extremes based on daily temper-
ature and precipitation data. *Wiley Interdisciplinary Reviews: Climate Change*,
16. J. Zscheischler et al. A few extreme events dominate global interannual variability
17. F.W. Zwiers et al. Climate extremes: challenges in estimating and understanding
recent changes in the frequency and intensity of extreme climate and weather
Node Classification in Dynamic Social Networks

Yulong Pei, Jianpeng Zhang, George H.L. Fletcher, and Mykola Pechenizkiy

Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, the Netherlands
{y.pei.1,j.zhang.4,g.h.l.fletcher,m.pechenizkiy}@tue.nl

Abstract. Classifying nodes in networks is different from traditional classification tasks since the i.i.d. assumption does not hold. In a dynamic scenario, nodes/edges may change during time which makes node classification more difficult. There have been a number of studies on node classification in networks in recent years but one limitation exists: how to make use of the temporal information. In this paper, we propose the dynamic Factor Graph Model (dFGM), which is an extension of Factor Graph Models, for the problem of node classification in dynamic social networks. dFGM can capture not only node attributes and correlations but also the temporal information for node classification. We conduct experiments on a real-world data set and the experimental results demonstrate the effectiveness of our proposed method.

1 Introduction

Network structures are ubiquitous nowadays and more and more data can be organized in networks with dependency relationships. Generally, nodes in networks can be associated with labels and these labels may come in many forms, e.g., demographic labels, interests, affiliations, etc. Assigning labels to unlabeled nodes in the graph is the node classification problem. However, the increasing number of the network applications and the complicated relationships between graph nodes have made the labels of the graph data expensive and/or difficult to obtain. Therefore, the problem of node classification in networks has attracted extensive attention recently.

Different from traditional classification tasks, the independent and identically distributed (i.i.d.) assumption does not hold for node classification in networks and methods should take the structure dependency into account. There have been a number of studies on node classification in networks in recent years [7, 9] and these methods can be categorized into two types [1]: (1) methods based on iterative application of traditional classifiers using structural properties as features and (2) methods which propagate the labels via random walks. However, there is one major limitation in existing studies. In specific, most of these studies focus on static networks. In fact, many real-world networks are dynamic and nodes/edges in the networks may change during time. In such dynamic scenario, temporal information can also play an important role in classifying nodes.
There are some methods which have been proposed to classify nodes in dynamic networks [2, 8]. Li et al. propose a method which can learn the latent feature representation and capture the dynamic patterns [2]. However, this method requires data from all the historical snapshots to classify nodes in the next snapshot while in practice some labels in previous data may be missing or incorrect. Yao et al. uses SVM to classify nodes in each snapshot and combines the support vector from last snapshot and current training data for classification [8]. But this operation depends heavily on the performance of SVM and using support vector from previous snapshot may also lose useful dynamic information.

Aiming to overcome the limitations, in this paper we propose the dynamic Factor Graph Model (dFGM) for node classification in dynamic social networks. In detail, the dynamic graph data is organized in the format of a series of graph snapshots and to model the graph snapshots, three types of factors, named node factor, correlation factor and dynamic factor, are designed in the dFGM based on node features, node correlations and temporal correlations, respectively. Node factor and correlation factor can capture the global and local properties of the graph structures while the dynamic factor can make use of the temporal information. To validate the effectiveness of the dFGM, a real-world data set DBLP is used for the experiments.

The main contributions of our work can be summarized as follows:

– We propose the dynamic Factor Graph Model (dFGM) for node classification in dynamic social networks and this model can capture node attributes, correlations and temporal information.
– We evaluate the proposed dFGM on a real-world data set and the experimental results demonstrate the effectiveness of our model compared with other methods on two evaluation metrics: accuracy and error in probability.

The rest of this paper is organized as follows. Section 2 introduces the related work and Section 3 formally defines the problem. Section 4 introduces the proposed dynamic Factor Graph Model. And then in Section 5 we discuss the experiments and analysis. Finally, in Section 6 we draw the conclusions.

2 Problem Definition

In this section, several necessary definitions are introduced and then the formal definition of the node classification problem is presented. In this paper, we assume edges to be undirected and also assume the nodes are fixed and the edges may change over time.

Definition 1 Partially labeled network. Given a fixed finite non-empty label set \( R \), a partially labeled network is \( G_L = \{V_L, V_U, E, X, r\} \) where (1) \( V_L \) is a set of labeled nodes and \( V_U \) is a set of unlabeled nodes with \( V_L \cup V_U = V \) and \( V_L \cap V_U = \emptyset \); (2) \( E \) is the set of edges, i.e., \( E \subseteq V \times V \); (3) \( X \) is an attribute matrix associated with nodes in \( V \) where each row corresponds to a node \( v \), each column an attribute, and an element \( x_{ij} \) denotes the value of the \( j^{th} \) attribute of node \( v \); and (4) \( r \) is a mapping function which maps each labeled node to a label, i.e., \( r : V_L \to R \).
Definition 2 Partially labeled dynamic network. Let $V$ be a finite set of nodes, a partially labeled dynamic network $\mathcal{G} = \{G^t| t = 1,\ldots,T\}$ consists of a series of graph snapshots $G^t = \{V_L^t, V_U^t, E^t, X^t, r^t\}$, where each snapshot $G^t$ is a partially labeled network as defined in Definition 1, and $V_L^t \cup V_U^t = V$.

Given a partially labeled dynamic network $\mathcal{G}$, our goal is to infer the labels of all the unlabeled nodes in the network. Formally,

Problem 1 Node classification in dynamic networks. Given a partially labeled dynamic network $\mathcal{G} = \{G^t| t = 1,\ldots,T\}$, learn a set of predictive functions $\{f_1, \ldots, f_T\}$ where for $1 \leq t \leq T$, $f^t : V \rightarrow R$ such that (1) $\forall v \in V_L^t$, $f^t(v) = r(v)$, and (2) $\forall v \in V_U^t$, $f^t(v) \in R$ is the correct label assignment.

Fig. 1. A four-node example in three snapshots of the dFGM. The white circles in the lower layers denote the nodes, the colored circles in the upper layers denote the labels and the squares are the factors. For these colored circles in the upper layer, the blue circles mean that these nodes are labeled and the black ones are unlabeled. The white, grey and black squares denote the node, correlation and dynamic factors, respectively.

3 Dynamic Factor Graph Model

In this section, we present the dynamic Factor Graph Model (dFGM), which builds on the traditional factor graph models, for node classification in dynamic
social networks. Intuitively the class/label of a node in a dynamic social network will be determined by three factors including: (1) node attributes, i.e., the node’s global and local characteristics at current time step. This factor corresponds to the node features extracted from the network; (2) node correlations, i.e., the relationships or interactions between nodes in a snapshot. This factor corresponds to the social relations of nodes in the networks; (3) previous labels. We assume that the label of a node will not change abruptly. This factor can capture the dynamic information in the network. Based on these intuitions, we propose the \textit{dFGM} which consists of three factors, named node factor, correlation factor and dynamic factor, and they correspond to the node attributes, node correlations and previous labels, respectively. In detail, these three factors are defined as follows.

**Node factor** \( g(r_i, x_i) \). This factor represents the posterior probability of the label \( r_i \) given the feature \( x_i \) of node \( v_i \).

**Correlation factor** \( c(r_i, N(r_i)) \). This factor reflects the correlation between nodes, where \( N(r_i) \) is the set of correlated labels to \( r_i \). There are multiple ways to define the set of correlated labels and in this study, \( N(r_i) \) denotes the labels of neighbors of node \( v_i \).

**Dynamic factor** \( d(r^t_i, r^{t-1}_i) \). This factor denotes the correlation between the labels of one node in two consecutive snapshots.

An example of the \textit{dFGM} with three factors is shown in Fig 1. In this example, there are two labels: 1 and 2. For the labeled nodes, the probability values of the ground label will be 1.0 or 0.0. For the unlabeled nodes, the probability values will be real numbers. Based on all the factors introduced above, the joint distribution of labels \( R \) given the graph \( G \) can be defined as

\[
P(R|G) = \prod_{t} \prod_{i} \; g(r_i, x_i) c(r_i, N(r_i)) d(r^t_i, r^{t-1}_i) \tag{1}
\]

These factors can be instantiated in different ways. We choose the exponential-linear function to instantiate the factors because it can simplify the model learning. In detail, the node factor is defined as

\[
g(r_i, x_i) = \frac{1}{Z_1} \exp \{ \alpha^T \phi(r_i, x_i) \} \tag{2}
\]

where \( Z_1 \) is the normalizing factor, \( \alpha \) is the weighting vector, and \( \phi \) is a vector of feature function. Similarly, the edge factor is defined as

\[
c(r_i, N(r_i)) = \frac{1}{Z_2} \exp \{ \sum_{r_j \in N(r_i)} \beta^T I_{\text{corr}}(r_i, r_j) \} \tag{3}
\]

where \( Z_2 \) is the normalizing factor, \( \beta \) is the weighting vector, and \( I_{\text{corr}} \) is the indicator function for node correlations and defined as

\[
I_{\text{corr}}(r_i, r_j) = \begin{cases} 0, & \text{if } e_{ij} \notin E \\ 1, & \text{if } e_{ij} \in E \end{cases} \tag{4}
\]

Then dynamic factor is defined in the same way

\[
d(r^t_i, r^{t-1}_i) = \frac{1}{Z_3} \exp \{ \gamma^T I_{\text{dyn}}(r^t_i, r^{t-1}_i) \} \tag{5}
\]
where $Z_2$ is the normalizing factor and $\gamma$ is the weighting vector. $I_{\text{dyn}}(r_i^t, r_i^{t-1})$ is the indicator function for the dynamic information and defined as

$$I_{\text{dyn}}(r_i^t, r_i^{t-1}) = \begin{cases} 0, & \text{if } r_i^t \neq r_i^{t-1} \\ 1, & \text{if } r_i^t = r_i^{t-1} \end{cases} \quad (6)$$

To learn this model, we write the joint probability defined in Eq (1) as

$$P(R|G) = \frac{1}{Z} \prod_t \prod_i \exp\{\theta^T s_i^t\} = \frac{1}{Z} \exp\{\theta^T \sum_t \sum_i s_i^t\} = \frac{1}{Z} \exp\{\theta^T S\} \quad (7)$$

where $Z = Z_1 Z_2 Z_3$ is the normalizing factor, $\theta$ is the parameter configuration, i.e., $\theta = (\alpha, \beta, \gamma)$ and $S$ is the concatenation of the factor functions, i.e., $S = (\phi(r_i, x_i)^T, I_{\text{corr}}(r_i, r_j)^T, I_{\text{dyn}}(r_i^t, r_i^{t-1})^T)^T$. Thus, model learning is to estimate the parameter configuration $\theta$. To solve this problem, we use the labeled data to infer the unknown labels. In specific, we use $R|RL$ to denote the predicted labels inferred from the known labels and define the log-likelihood function $O(\theta)$ as

$$O(\theta) = \log p(R^L|G) = \log \sum_{R|RL} \frac{1}{Z} \exp\{\theta^T S\}$$

$$= \log \sum_{R|RL} \exp\{\theta^T S\} - \log Z = \log \sum_{R|RL} \exp\{\theta^T S\} - \log \sum_{R} \exp\{\theta^T S\}$$

Then gradient descent method is used to solve this optimal problem:

$$\frac{\partial O(\theta)}{\partial \theta} = \frac{\partial (\log \sum_{R|RL} \exp\{\theta^T S\} - \log \sum_{R} \exp\{\theta^T S\})}{\partial \theta}$$

$$= \frac{\sum_{R|RL} \exp\{\theta^T S\} \cdot S - \sum_{R} \exp\{\theta^T S\} \cdot S}{\sum_{R|RL} \exp\{\theta^T S\}} = E_{p_a}(R|RL, G) S - E_{p_a}(R, G) S$$

Since the graphical structure in $dFGM$ can be arbitrary and may contain cycles, we use Loopy Belief Propagation (LBP) [3] for the model learning in this paper similar to [6] due to its effectiveness in handling graphs with cycles. The learning algorithm is summarized in Algorithm 1.

4 Experiments

We conduct experiments to validate the performance of $dFGM$ on a subset of DBLP data set. Conferences from six research communities, including artificial intelligence and machine learning, algorithm and theory, database, data mining, computer vision, information retrieval, have been extracted. In specific, we extract the co-author relations in these conferences from 2001 to 2010 and data in each year is organized in a graph snapshot. Each author represents a node in the network and if two authors collaborated on a paper, there will be an edge between these two nodes. The features are extracted from each snapshot using DeepWalk [4] due to its generalization in graph mining tasks.

http://dblp.uni-trier.de/xml/
Algorithm 1 Model learning for dFGM

Input: learning rate $\eta$
Output: learned parameters

while not converge do
    Calculate $\mathbb{E}_{p(R|L,G)} S$ and $\mathbb{E}_{p(R,G)} S$ using LBP
    Calculate the gradient $\nabla_\theta$ of $\theta$ according to Eq. (9)
    Update parameters $\theta$ with the learning rate $\eta$ according to $\theta_{new} = \theta_{old} - \eta \nabla_\theta$
end while

To compare our proposed dFGM with existing methods, three types of baseline methods have been used:

- **Feature-based classification.** We use the Logistic Regression (LR) and Support Vector Machine (SVM) as the baseline in the feature-based classification.

- **Link-based classification.** Two methods have been employed in the link-based classification type. The first method is majority voting method with dynamic information (MV+dynamic). In detail, if a node is labeled in previous snapshot, the predicted label is copied from previous one. Otherwise, the node is labeled by majority voting from neighbors in current snapshot. The second one is the collective classification (CC) [5]

- **Factor graph models (FGM).** To validate the effectiveness of the temporal information, we also compare dFGM with FGM using only the features (FGM_feat) and FGM using both features and correlations (FGM_corr).

Note that features used in these methods are the same extracted using Deep-Walk and the correlations used here are same to the link-based classification methods.

4.1 Evaluation Metrics

Two types of evaluation metrics have been used in the experiments: accuracy and error in probability. The accuracy is defined as $\text{accuracy} = \frac{n}{N}$ where $n$ is the number of instances correctly classified (the label with highest probability matches the ground-truth label), and $N$ is the total number of instances in the test data.

To better evaluate the performance, we also use another evaluation metric, namely the error in probability. This metric is beneficial in two aspects: (1) it can match the output of dFGM which is the probability of labels; (2) it can evaluate the prediction of multiple labels. The error in probability is defined as

$$\text{error} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{c} |\hat{p}_i^j - p_i^j|$$ (10)

where $N$ is the number of instances in the test data and $c$ is the number of labels. $\hat{p}_i^j$ and $p_i^j$ are the predicted probability and ground probability of label $j$ for user $i$, respectively. The ground probability of label $j$ for user $i$ is the ratio of the number of papers published by user $i$ in community $j$ to the total number of papers published by user $i$. 


Table 1. Comparison of node classification performance in DBLP data set.

<table>
<thead>
<tr>
<th>Methods</th>
<th>accuracy</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature-based</td>
<td></td>
<td></td>
</tr>
<tr>
<td>classification</td>
<td>LR</td>
<td>0.3157±0.0031</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>0.4983±0.0029</td>
</tr>
<tr>
<td>Link-based</td>
<td>MV+dynamic</td>
<td>0.5049±0.0011</td>
</tr>
<tr>
<td>classification</td>
<td>CC</td>
<td>0.7935±0.0033</td>
</tr>
<tr>
<td>Factor Graph Models</td>
<td>FGM_feat</td>
<td>0.2684±0.0024</td>
</tr>
<tr>
<td></td>
<td>FGM_corr</td>
<td>0.8360±0.0328</td>
</tr>
<tr>
<td></td>
<td>dFGM</td>
<td>0.8410±0.0058</td>
</tr>
</tbody>
</table>

4.2 Results

The performance of $dFGM$ and other methods are shown in Table 1 and we use 70% data as the training set and 30% as the test set. From the results, some conclusions can be drawn: (1) the $dFGM$ outperforms other methods in both evaluation metrics which shows the effectiveness of our proposed model and the importance of the dynamic information; (2) since FGM is used to model correlations in graphs, if the correlation information is removed, i.e., in FGM_Feat, the performance will be extremely poor even compared with traditional classification methods, e.g., SVM; (3) link-based classification methods (MV+dynamic and CC) perform better than feature-based methods (LR and SVM), and it demonstrates the importance of correlations in graph classification problem.

It is worth noting that the improvement on accuracy is very small. This is because in accuracy calculation, the predicted labels are the labels with maximum probability. For example, assume CV is the correct label and the probability of label CV is 0.8 predicated by model A and 0.95 predicted by model B, although model B performs better (it gives a more precise prediction), A and B have the same predicted label for accuracy metric.

Furthermore, we analyze the influence of size of training data in $dFGM$. The size of training data is set from 10% to 90% and the results are shown in Fig. 2 and Fig. 3. Overall, better results can be obtained when more training data is given and they demonstrate the robustness of the $dFGM$. Moreover, note that when the size of training data is relatively small (e.g., less than 50% in Fig 2 and less than 30% in Fig 3), the performance of $dFGM$ is not good because estimates of correlation and dynamic information become less reliable with decreased training set size which will influence the performance of $dFGM$.

5 Conclusions

In this paper, we proposed the $dFGM$ method to classify nodes in dynamic social networks. To capture the temporal information, graph factors based on node attributes, node correlations and dynamic information are integrated in $dFGM$. Experiments have been conducted on a real-world data set which demonstrate the effectiveness of our method. We also analyzed the influence of feature dimension and size of training data. As future work, we will take the UGC information.
into consideration for node classification in the dynamic scenario. In addition, with rapid increase of network size, it will be interesting to study more effective and efficient method for larger scale networks.

Acknowledgement The research of Pei is supported by the Netherlands Organisation for Scientific Research (NWO).

References

EAST Representation: Fast Discriminant Temporal Patterns Discovery From Time Series

Xavier Renard1,3, Maria Rifqi2, Gabriel Fricout3 and Marcin Detyniecki1,4

1Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6, Paris, France.
2Université Panthéon Assas, Univ Paris 02, LEMMA, Paris, France.
3ArcelorMittal Research, Maizières-lès-Metz, France.
4Polish Academy of Sciences, IBS PAN, Warsaw, Poland.

Abstract. Mining discriminant temporal patterns is one problem for the time series classification currently led by the shapelet. We expose this issue from the perspective of a standard feature-space classification task. This approach is enabled by the recent observation that most enumerable subsequences from a time series are redundant and can be discarded. In addition to its simplicity the approach has state-of-the-art classification performances with extremely fast computations. It also provides a flexible framework with interesting perspectives.

1 Introduction & related work

Two main approaches exist to discover localized and phase independent discriminant temporal patterns from time series: the shapelets and the derivation of features from intervals of the series [2]. We focus here on the shapelet principle. We invite the reader to refer to the seminal article [12] for details, to summarize the shapelet discovery relies on three steps.

Step 1 Exhaustive enumeration of the subsequences from a set of time series.
Step 2 Evaluation of the subsequences. The minimal euclidean distances (MED) between subsequences and time series of the dataset are computed to get the discriminatory power (usually the information gain) of each subsequence.
Step 3 The subsequences that most consistently separate the time series with respect to their classes are conserved.

Several variations have been proposed after the original shapelet tree that learns a tree of successive discriminant shapelets at a price of a large time complexity. The shapelet transform [7] has become a classical instantiation of the shapelet principle: the shapelet discovery is performed in one pass and the MED between time series and shapelets feed a classifier. Among the other approaches we can cite the logical shapelets to discover conjunction or disjunction of shapelets [8], the fast shapelets [9] and the learnt shapelets [4].

The shapelet has two limitations: the time complexity and, from our point of view, the independent evaluation of the discriminatory power of each subsequence. The first limitation results from the discovery complexity in $O(L^3N^2)$
with \( L \) the time series length and \( N \) the number of time series in the dataset [12]. The second limitation lies in the Step 2 of subsequence evaluation (based on the information gain or a similar metrics): each shapelet has to be sufficient to discriminate a set of time series. Even if a set of characteristic subsequences is very discriminant, the shapelet discovery would fail to find it. To reduce the time complexity, several improvements have been proposed [12,9] but it remains large. It has been recently observed that most subsequences extracted from a time series are redundant. A drastic random sub-sampling among the subsequences at Step 1 instead of an exhaustive enumeration has been shown effective to reduce the time complexity while preserving the classification performances [3,10,5].

In this work, we rely on this observation to propose a flexible representation called EAST (Enumerate And Select discriminant Temporal patterns). It aims at improving the two aforementioned limitations (time complexity and independent evaluation). We postulate that each subsequence extracted from a set of time series should be considered as a feature and the exhaustive set of subsequences forms a feature vector. The redundancy is eliminated by the random sub-sampling of the subsequences. A relevant subset of features (i.e. subsequences) for the classification is conserved after a feature selection stage. The originality of the approach resides in the problem formulation that enables the use of well-established feature selection techniques to perform a powerful discovery of discriminant temporal patterns. We show, with instances of our proposition, that the classification performances already reach the state-of-the-art while being extremely fast (evaluation of a few thousands subsequences at most). We also demonstrate its scalability: the number of subsequences to evaluate is independent of the number of time series in the dataset.

2 Proposition: discriminant temporal pattern discovery

We have a training set \( D \) of time series \( T_n \) with \( n \in [1, 2, \ldots, N] \) where \( T_n = [t_n(1), \ldots, t_n(|T_n|)] \). \( T_n \) has a length \( |T_n| \in [L_{min}, \ldots, L, \ldots, L_{max}] \) with \( L_{min}, L, L_{max} \in \mathbb{N}^* \), where \( L_{min} \) is the smallest time series of \( D \) and \( L_{max} \) is the longest one. A subsequence \( s \) of length \( l \) at a starting position \( j \) in a time series \( T_n \) is noted as \( s_{j+l}^l(T_n) = [t_n(j), \ldots, t_n(j+l-1)] \). \( S \) is the set of all the subsequences \( s \) that it is possible to extract from \( D \), whatever their lengths and starting positions are.

The problem is framed in the field of time series classification: each time series \( T_n \) has exactly one class label \( y(T_n) \in \mathbb{Y} \). In this work, our concern is the discovery of meaningful temporal patterns to perform time series classification. We make the assumption that there exists a strongly-discriminant set of patterns \( Z = \{z_1, \ldots, z_p, \ldots, z_P\} \) with \( p, P \in \mathbb{N}^* \) and \( |z_1|, |z_p|, \ldots, |z_P| \in [1; L_{max}] \) where \( z_p \) is discriminant of one class or a subset of classes of \( \mathbb{Y} \). \( Z \) is strongly-discriminant in that it contains all the possible subsequences, which taken independently or not, are discriminant enough to solve the classification problem. The transformation of \( T_n \) using \( Z \) produces a feature vector \( X_n \), such that a classifier \( f \) is able to learn a mapping \( f(X_n) \rightarrow y(T_n) \). The transformation is based on a distance.

63
However $Z$ is unknown. Our objective is to discover from $D$ a set $\hat{Z} = \{\hat{z}_1, \ldots, \hat{z}_j, \ldots, \hat{z}_J\}$ of patterns that produces a feature vector $X^{\hat{Z}}$ such that the classification performance of $f(X^{\hat{Z}})$ is as close as possible of the one of $f(X)$ with $X$ obtained with a transformation based on $Z$.

2.1 Proposition

To determine $\hat{Z}$ we propose to combine a random enumeration of subsequences from $D$ with a feature selection stage to retain a relevant set of patterns with respect to the classification task.

Step 1: random sub-sampling to handle subsequence redundancy

The first step of our proposition relies on a random sampling $\hat{S}$, among all the subsequences $S$, because of the efficiency of this principle mentioned in introduction. Each subsequence $s_j^{l+1}(T_n)$ is given the same probability to be picked, whatever its time series, position and length. We demonstrate later that the number of subsequences $q = |\hat{S}|$ to draw to obtain a given classification performance is not related to the size of the dataset $D$ (i.e. the number of time series).

Step 2: learning the representation by selecting a set of discriminant subsequences

Once $\hat{S}$ is drawn we need to discover the set $\hat{Z} \subset \hat{S}$ that maximizes the classification performance. We propose to formalize the problem from the perspective of a standard feature-space classification task. The minimal euclidean distance $d_{\text{min}}$ between a subsequence $s$ with $|s| = l$ from $T_1$ and a time series $T_2$ such that:

$$d_{\text{min}}(s, T_2) = \min([d(s, s_1^{l+1}(T_2)), \ldots, d(s, s_1^{T_2}_{|T_2|+l}(T_2))])$$

$d_{\text{min}}$ is calculated between each subsequence of $\hat{S}$ and every time series of $D$. The result is a feature space $X$ (Fig. 1) where the distances to the subsequences of $\hat{S}$ are the attributes (columns) and the time series of $D$ are the instances (rows). The number of columns of $X$ (i.e. number of attributes) may still be large and it is very likely that it contains numerous meaningless features: no selection has been performed yet with respect to the classification problem. In other terms, irrelevant subsequences $s \in \hat{S}$ are still present in the feature space $X$.

To reduce $\hat{S}$ to $\hat{Z}$ and derive a feature space $X^{\hat{Z}}$ relevant to train a classifier $f(X^{\hat{Z}}(T_n)) \rightarrow y(T_n)$ we use the feature vector formalization of the problem to exploit classical feature selection approaches. They allow to efficiently identify relevant attributes in a feature space with respect to a classification task. Advanced feature selection techniques offer the possibility to discover both single discriminant subsequence and sets of subsequences where each subsequence is characteristic of a class or a subclass, while the whole set is discriminant. Numerous feature selection techniques exist, the approaches used in this work are presented in the experimentation section (we use them as black boxes).
Fig. 1: EAST principle workflow. After a drastic subsequences sub-sampling (1), the distances between subsequences and time series (2) form a feature space of reasonable size (3) on which advanced feature selection techniques can be applied to discover discriminant set of subsequences (4).

The overall principle of the proposed approach to discover discriminant temporal patterns is summarized Fig. 2. A classifier can be trained with $X\hat{Z}$. The result of the training is both a set $\hat{Z}$ of patterns and a classifier $f$. To perform the classification of new instances, time series are transformed into a feature vector according to $\hat{Z}$ and the classification is performed with $f$.

1: $\hat{S} \leftarrow$ Draw $q$ subsequences from time series from $D$
2: $X \leftarrow$ Calculate $d_{\min}$ between time series from $D$ and subsequences from $\hat{S}$
3: $X\hat{Z}, \hat{Z} \leftarrow$ Perform feature selection on $X$ with respect to labels of $Y$

Fig. 2: Learning of the EAST representation in 3 key steps: random sub-sampling, minimal euclidean distance and feature selection.

3 Experimentation

The objective of the experimentation is to evaluate the relevance of advanced feature selection in a standard feature space for the temporal pattern discovery over the classical selection scheme used by the shapelet (usually the information gain). The classification performances are observed together with the time complexity required by the pattern discovery with several configurations. For this purpose, EAST is instantiated with several feature selection approaches and classifiers for various values $q$ of subsequences drawn. The experimentation performed for this work is framed into the classical UCR univariate time series classification framework (45 datasets from this repository are used).

With EAST, the feature selection stage is open to any approach. To perform the experimentation we use some of them. Feature selection is an established field: we do not contribute but instead we rely on it. Also, we don’t advocate one approach is better than another. Feature selection methods are usually classified into three groups: filters, wrappers and embedded methods [6]. For the experimentation we select one wrapper, the Recursive Feature Elimination with cross-validation associated with a linear SVM (named RFE+SVM), and two embedded methods, the Randomized Logistic Regression (RLR) and the Ran-
dom Forest (RF). For the RLR we test two classifiers: a SVM with a RBF kernel and a random forest (respectively named RLR+SVM and RLR+RF). These approaches are able to learn combinations or sets of features (i.e. subsequences). On the contrary, the shapelet approach, which usually makes use of the information gain that is part of the filters, is unable to learn such sets or combinations. For the random draw of $\hat{S}$ several values $q = |\hat{S}|$ are tested: $q \in [50, 100, 500, 1000, 2000, 5000]$.

The results are compared with the current leading shapelet approach, the shapelet ensemble (SHPT) [2]. The authors state that shapelet ensemble performs identically or better than other shapelet approaches. We reproduce here their results. We also compare the results with the random-shapelet (RSHPT) [10] that has the same selection stage than the classical shapelet but on a small fraction of the exhaustive set of subsequences. The same number of subsequences is picked for the random-shapelets and for the EAST instantiations.

A strict evaluation protocol is required to assess the EAST representation and the random-shapelets because they contain a random generation step. We rely on the evaluation protocol proposed in [1] for a proper way to analyze the performances of randomized algorithms. Each single test of the EAST representation and the random-shapelets is reproduced 10 times to evaluate the variability. The complete description of the evaluation protocol of the provided results is described in additional material [11].

3.1 Results

Classification performances

Fig. 3: Comparison of the approaches with the Nemenyi test. Groups of approaches not significantly different ($\alpha = 0.05$) are connected. CD is the critical difference.

The classification performances of the proposed approach are significantly similar to the ones obtained by the shapelet ensemble (Figure 3 & 4). These performances are obtained with only 2000 subsequences drawn for the best implementation of the proposition that is an infinitesimal fraction of subsequences evaluated by the exhaustive shapelet ensemble; the largest tested UCR
dataset reaches $5.10^8$ subsequences. The relevance of advanced feature selection approaches over independent discriminant subsequences is also shown by the experimentation. With the same number of subsequences drawn, our proposition systematically outperforms a random sampling associated with the classical shapelet evaluation procedure based upon the information gain. The parameter $q$ is obviously critical, but until a certain point: with this experimentation we observe no statistical difference in the classification performances between the best performing configuration with $q = 2000$, $q = 5000$ and those of the shapelet ensemble, the state-of-the-art (Figure 3). It is also worth noting the low standard deviation in the performances, in particular for the RLR+SVM approach, with a maximum of 2.7% (for one single dataset) with most standard deviation below or around 1% (for $q = 2000$). Raw results of the experimentation are available in additional material [11].

| Score | 29 | 27 | 25 | 18 | 16 | 15 | 12 | 11 | 8 | 6 | 5 | 3 | 3 | 1 | 1 | -4 | -10 | -11 | -11 | -15 | -18 | -25 | -25 | -25 | -25 | -25 |
|-------|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|
| Time complexity For the approaches used for this experimentation, most of the time is spent in the distance calculations between each subsequence enumerated and the time series. This fact is illustrated figure 6 and is especially true for large datasets and with values of $q$ increasing: the time spent in the feature selection becomes insignificant. We use this specificity to compare the time complexity of the approaches and avoid implementation or hardware bias. |
The EAST representation enumerates a fixed number of subsequences, in this work the maximal value is \( q = 5000 \). The typical shapelet approach performs an exhaustive enumeration. Figure 7 shows that the exhaustive shapelet discovery (SHPT) evaluates subsequences sets several order of magnitude larger than the EAST approach while having comparable classification performances. For the datasets used in the experimentation, the exhaustive number of subsequences to extract varies from 20,100 (Italy Power Demand) to 524,800,000 (Star Light Curves). For all the datasets and for significantly similar classification performances our proposition uses \( q = 2000 \) subsequences. On the bigger dataset this is less than 0.0002\% of the exhaustive number of subsequences. The exhaustive number of subsequences depends on \( L \) (time series length) and \( N \) (number of time series in \( D \)). We demonstrate that the number of subsequences to draw from \( S \) to determine \( \hat{Z} \) is not dependent of \( N \). This allows a considerable gain for the training phase on datasets with numerous time series. The following lemma is demonstrated in additional material [11].

**Lemma** The probability of drawing a relevant subsequence \( \hat{z}_j \) for the classification task is independent of the number of time series \( N \) in \( D \).

![Fig. 6: Time spent in the distance calculations vs. Time spent in the feature selection for EAST. For small datasets (ItalyPowerDemand), the feature selection requires a similar amount of time than the distance calculations. For larger datasets (Coffee, Car) feature selection becomes insignificant in front of distance computations. We use this specificity to compare the time complexity of the approaches based on the number of distance computations and avoid implementation or hardware bias.](image)

![Fig. 7: Number of subsequences evaluated by EAST and the exhaustive shapelet discovery (log scale). EAST enumerates a constant number of subsequences over the datasets with comparable classification accuracies than the shapelet ensemble that generates subsequences sets several orders of magnitude larger.](image)
4 Conclusion

This work evaluates advanced feature selection relevance to discover discriminant temporal patterns for time series classification. This approach is enabled by the previous observation that most subsequences in a time series are redundant and can be discarded. We state that each subsequence represented by its distance to the time series is a feature in a feature vector on which classical feature selection can be applied. The experimentation on 45 datasets of the UCR shows significantly similar classification performances to the state-of-the-art with a time complexity drastically reduced. Moreover the scalability of the approach is demonstrated. The proposed approach may allow the discovery of sophisticated patterns, such as multivariate patterns, thanks to the use of advanced feature selection: this study is our next step.

References

Missing Data Prediction in Multi-source Time Series with Sensor Network Regularization

Weiwei Shi∗, Yongxin Zhu ∗, Xiao Pan∗, Philip S. Yu†, Bin Liu∗, and Yufeng Chen‡

∗School of Microelectronics, Shanghai Jiao Tong University, China
∗Shijiazhuang Tiedao University, China
†Department of Computer Science, University of Illinois at Chicago, USA
‡Shandong Power Supply Company of State Grid, China

{iamweiweishi,zhuyongxin}@sjtu.edu.cn
smallpx@stdu.edu.cn, psyu@uic.edu, liubinpoem@gmail.com,
chenyufeng169@sina.com

Abstract. Raw multi-source time series usually contain missing values, which can hardly meet the requirement of precise analysis. To address the problem, we propose a novel method named Matrix factorization with Sensor Network Regularization (MSNR). In the paper, we consider two sensors in a sensor network to be correlated if one sensor has a strong correlation with the other one. On the contrary, if one sensor has a weak correlation with another one, we regard the two sensors as uncorrelated. The proposed method aims to predict missing data by minimizing the difference between a sensor and its correlated sensors or to maximize the difference between a sensor and its uncorrelated sensors via matrix factorization. In the process of matrix factorization, we impose the sensor network regularization terms to constrain the objective functions of matrix factorization. To treat the correlated or uncorrelated sensors differently, we further improve the objective functions by incorporating similarity functions. Extensive experiments on real-world data sets demonstrate that the proposed approach MSNR can effectively improve the performance of missing data prediction in multi-source time series, even when the missing ratio is as high as 90 percent.

Keywords: missing data prediction, matrix factorization, multi-source time series

1 Introduction

Multi-source time series are ubiquitous in many real-world applications, such as electric equipment monitoring, weather forecasting, environment state monitoring, security surveillance, and so on [1, 2]. In most applications, multiple sensors are used to generate time series data, and they usually share one common goal. In this paper, the sensors sharing one common goal are treated as a sensor network. Unfortunately, the raw time series in a sensor network usually contain missing values due to the harsh working conditions or uncontrollable factors. A
large collection of data mining and statistical methods have been proposed to predict the missing values of time series [3]. Grabocka et. al. proposed a matrix factorization method to classify multiple time series. Their work aims at extracting latent factors based on observed entries [4]. However, these methods either focus on predicting the missing data in the time series from one single source or could not effectively deal with the missing data prediction problem of the time series from multiple sources. In this paper, aiming at solving the above problems, we propose MSNR, a matrix factorization with sensor network regularization method, that utilizes the correlation information among the different sensors in a sensor network to improve the accuracy of missing data prediction in multi-source time series. Moreover, to treat the correlated or uncorrelated sensors differently, we further improve the sensor network regularization terms of the objective function by incorporating similarity functions. The experimental results reveal that our proposed method shows superior performance to the state-of-the-art algorithms.

2 Proposed Methods

2.1 Low Rank Matrix Factorization

Let $X$ be the multi-source time series collected from $N$ different data sources, and the $j$th entity in time series data from the $i$th source can be denoted as $X_{ij}$ for $i = \{1, 2, 3, \ldots, N\}$, $j = \{1, 2, 3, \ldots, M\}$. As the original matrix $X$ might contain a great number of missing values, we only need to factorize the observed entities in $X$. Hence, we have an optimization problem based on Singular Value Decomposition (SVD):

$$
\min_{S,V} \frac{1}{2} \| W \circ (X - SV^T) \|_F^2 + \frac{\lambda_1}{2} \| S \|_F^2 + \frac{\lambda_2}{2} \| V \|_F^2,
$$

(1)

where $S \in \mathbb{R}^{N \times L}$, $V \in \mathbb{R}^{M \times L}$ with $L < \min(N, M)$, $\lambda_1, \lambda_2 > 0$, $W$ is an indicator matrix, and $\circ$ denotes the Hadamard product. Two regularization terms $\| S \|_F^2$ and $\| V \|_F^2$ are added in order to avoid overfitting. Gradient based approaches can be applied to find a minimum due to their effectiveness and simplicity [5].

2.2 Model 1: Correlated Sensors based Regularization

In a sensor network, although the different sensors are assigned different tasks, they usually share one common goal and there might exist strong correlation among some of the sensors. If one sensor has a strong correlation with another one, we call the two sensors are correlated.

As $S$ denotes the latent sensor matrix and there might be strong correlation among correlated sensors, we propose the first missing data prediction model based on matrix factorization technique with the following optimization problem:

$$
\min_{S,V} \mathcal{L}(X, S, V) = \frac{1}{2} \| W \circ (X - SV^T) \|_F^2 + \frac{\lambda_1}{2} \| S \|_F^2 + \frac{\lambda_2}{2} \| V \|_F^2 + 
\frac{\alpha}{2} \sum_{i=1}^{N} \| S_i - \frac{\sum_{c \in C(i)} H(i,c) * \rho_{i,c} S_c}{\sum_{c \in C(i)} H(i,c)} \|_F^2,
$$

(2)
where \( \alpha \) is the penalty factor and \( \alpha > 0 \), \( C(i) \) denotes the set of the correlated sensors of the \( i \)th sensor and \( |C(i)| \) is the total number of these correlated sensors. The included scaling factor \( \rho_{i,c} \) aims at matching the scale difference between the \( i \)th sensor and the \( c \)th sensor. In the first model, we incorporate one sensor network regularization term, i.e., the correlated sensors based regularization term. Concretely, if the correlated sensors are \( C(i) \), we deduce that the state of the \( i \)th sensor is correlated to the average state of \( C(i) \). The function \( H(i, c) \) measures the similarity between the \( i \)th sensor and the \( c \)th sensor. From this improved regularization item, we know that if the \( c \)th sensor is very correlated to the \( i \)th sensor, the value of \( H(i, c) \) will be large, i.e, it contributes more to the state of the \( i \)th sensor.

2.3 Model 2: Uncorrelated Sensors based Regularization

The first model we propose imposes a correlated sensors based regularization term to constrain the matrix factorization. From the opposite view, if one sensor has a weak correlation with another one, we call the two sensors are uncorrelated. And we also employ another sensor network regularization term, i.e., the uncorrelated sensors based regularization term, to build the second model. Since uncorrelated sensors share weak correlation, we attempt to add one constrain term to maximize the distance between the \( i \)th sensor and its uncorrelated sensors. Consequently, the optimization problem in Equation (2) is updated as:

\[
\min_{S,V} \mathcal{L}'(X, S, V) = \frac{1}{2} \|W \circ (X - SV^T)\|_F^2 + \frac{\lambda_1}{2} \|S\|_F^2 + \frac{\lambda_2}{2} \|V\|_F^2 - \frac{\alpha'}{2} \sum_{i=1}^{N} \|S_i - \sum_{c' \in C'(i)} H(i, c') \cdot \rho_{i,c} S_{c'} \|_F^2. \tag{3}
\]

where \( \alpha' \) is the penalty factor and \( \alpha' > 0 \), \( C'(i) \) denotes the set of the uncorrelated sensors of the \( i \)th sensor.

2.4 Similarity Function

The proposed regularization terms in Equation (2) and Equation (3) require a function \( H \) to measure the similarity between two sensors, which is a key component of the proposed method. Vector Space Similarity (VSS) is applied to measure the similarity between two sensors \( i \) and \( c \):

\[
H_{VSS}(i, c) = \frac{\sum_{j \in o_i \cap o_c} X_{ij} \cdot X_{cj}}{\sqrt{\sum_{j \in o_i \cap o_c} X_{ij}^2} \sqrt{\sum_{j \in o_i \cap o_c} X_{cj}^2}}, \tag{4}
\]

where \( o_i \) and \( o_c \) is the subset of \( x_i \) and \( x_c \). The entities in \( o_i \) and \( o_c \) are observed.

Another way to measure the similarity between two sensors \( i \) and \( c \) is based on Gaussian Kernel (GK):

\[
H_{GK}(i, c) = \exp(-\frac{\sum_{j \in o_i \cap o_c} (X_{ij} - X_{cj})^2}{2\sigma^2}). \tag{5}
\]
Another commonly used function Pearson Correlation Coefficient (PCC) is employed to take the different scales between two sensors into consideration. The details of PCC could be found in [6].

Dynamic Time Warping (DTW) is a well-known technique to compare two time series with different length. The strategy is to find a warping path $\mathbf{W}$ that minimize the warping cost. This path and the relevant details can be found using dynamic programming [7]. To make it consistent that a larger value of $H$ means that sensors $i$ and $c$ are more correlated, the reciprocal of DTW is employed as the similarity function:

$$H_{\text{DTW}}(i, c) = \frac{1}{\text{DTW}(o_i, o_c)}.$$  \hfill (6)

Furthermore, to better reveal the necessity of incorporating similarity functions, a constant function (CF) $H_{\text{CF}}(i, c) = C$ is also employed as the baseline function in the paper.

### 3 Experiments

In this section, to demonstrate the effectiveness of the proposed method MSNR, we conduct extensive experiments on two real-world data sets, which include the Motes data set [8] and the Diagnostic Gases data set [6].

To evaluate all the methods fairly, we incrementally simulate the data missing of the two data sets with an increasing missing ratio. For example, to increase the missing ratio from 0.10 to 0.15, we randomly move 5% of the total data from the observed data set to the missing data set. In this way, the subsequent missing data set always contains the missing data of the previous one.

From the Equation (2) and (3), we know that the constant value $C$ will not change the value of the equations. Thus, $C$ could be simply set as 1. Besides, the parameters $\lambda_1$ and $\lambda_2$ are both set equal to $\lambda$ in this paper. The parameter $|C(i)|$ determines how many correlated or uncorrelated sources should be incorporated into the optimization functions. To make the paper more concise, we denote the method MSNR based on Model 1 as MSNR$_{M1}$ and use MSNR$_{M2}$ to indicate the method MSNR based on Model 2. We compare the proposed method MSNR with many baseline methods in predicting the values of the missing samples in multi-source time series. The comparison methods used include: Linear Interpolation (LI), Non-negative Matrix Factorization (NMF) [9], Probabilistic Matrix Factorization (PMF) [10], Bayesian PMF (BPMF) [11], Support Vector Machine (SVM) [6] and Simplified MSNR (SM), which is a simplified version of MSNR with $\alpha = 0$. To evaluate the performance of the proposed method, root mean squared error (RMSE) is used to measure the prediction quality [6]:

$$\text{RMSE} = \sqrt{\frac{\sum_{i,j}(1 - W_{ij})(X_{ij} - \hat{X}_{ij})^2}{\sum_{i,j}(1 - W_{ij})}},$$  \hfill (7)

where $X_{ij}$ is the observed value and $\hat{X}_{ij}$ is the corresponding predicted value. $W$ is the indicator matrix.
3.1 Experimental Results

Fig. 1 shows the experimental results of the proposed methods and baseline methods on the Motes and the Diagnostic Gases data sets. The logarithm of RMSE is shown in the vertical axis to present the experimental results more clearly. First, we take the first model $\text{MSNR}_{M1}$ into consideration. We can see that $\text{MSNR}_{M1}$ consistently outperforms the other baseline methods. For instance, as for the Motes data set, when the missing ratio $\epsilon$ is equal to 0.4, $\text{MSNR}_{M1}$ achieves the lowest RMSE 2.56, which is about 85% lower than PMF. Even when the missing ratio exceeds 60%, the RMSE of $\text{MSNR}_{M1}$ is still within a reasonable range. As for the Diagnostic Gases data set, the proposed method also obtains the best performance, while the other baseline methods show barely satisfactory results even when the missing ratio is as low as 0.1.

Moreover, given the second model $\text{MSNR}_{M2}$, we can observe that the RMSE of $\text{MSNR}_{M2}$ is generally a bit lower than $\text{MSNR}_{M1}$ for the Motes data set. However, as for the Diagnostic Gases data set, the performance of $\text{MSNR}_{M2}$ is not as good as that of $\text{MSNR}_{M1}$. As the Motes data set generates from 54 sensors, the Motes data set has a much higher chance of containing uncorrelated sensors. Thus $\text{MSNR}_{M2}$ shows better performance for the Motes data set. On the contrary, the Diagnostic Gases data set is collected from only five sensors. As a consequence, it is much more important for the Diagnostic Gases data set to find the correlated sensors, thus $\text{MSNR}_{M1}$ shows superior performance. Nevertheless, as $\text{MSNR}_{M1}$ and $\text{MSNR}_{M2}$ show better performance than the baseline methods, they are both alternative models in the proposed method.

Furthermore, it is noteworthy that the only difference between the proposed method and SM is whether the sensor network regularization terms are incorporated or not. The SM method shows larger RMSE than both $\text{MSNR}_{M1}$ and $\text{MSNR}_{M2}$ based on the experimental results in the Fig. 1. As a consequence, we may safely deduce that the incorporated network regularization terms mainly contribute to the superior performance of the proposed method.
Table 1. Performance of MSNR$_{M1}$ with different similarity functions and missing ratio $\epsilon$.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>VSS</th>
<th>GK</th>
<th>PCC</th>
<th>DTW</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2.48</td>
<td>2.49</td>
<td>2.41</td>
<td>2.48</td>
<td>2.50</td>
</tr>
<tr>
<td>0.4</td>
<td>2.61</td>
<td>2.62</td>
<td>2.55</td>
<td>2.62</td>
<td>2.59</td>
</tr>
<tr>
<td>0.6</td>
<td>2.94</td>
<td>2.93</td>
<td>2.90</td>
<td>2.97</td>
<td>2.93</td>
</tr>
<tr>
<td>0.7</td>
<td>3.08</td>
<td>3.01</td>
<td>2.98</td>
<td>2.95</td>
<td>3.09</td>
</tr>
<tr>
<td>0.8</td>
<td>3.07</td>
<td>3.13</td>
<td>2.97</td>
<td>2.86</td>
<td>3.06</td>
</tr>
<tr>
<td>0.9</td>
<td>3.06</td>
<td>3.16</td>
<td>2.98</td>
<td>2.96</td>
<td>3.09</td>
</tr>
</tbody>
</table>

Motes data set

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>VSS</th>
<th>GK</th>
<th>PCC</th>
<th>DTW</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>40.47</td>
<td>24.66</td>
<td><strong>16.72</strong></td>
<td>24.93</td>
<td>56.45</td>
</tr>
<tr>
<td>0.4</td>
<td>52.02</td>
<td>51.16</td>
<td><strong>21.40</strong></td>
<td>72.26</td>
<td>54.02</td>
</tr>
<tr>
<td>0.6</td>
<td>62.20</td>
<td>85.82</td>
<td><strong>56.29</strong></td>
<td>92.23</td>
<td>56.44</td>
</tr>
<tr>
<td>0.7</td>
<td>55.51</td>
<td>49.92</td>
<td>63.19</td>
<td><strong>33.40</strong></td>
<td>74.21</td>
</tr>
<tr>
<td>0.8</td>
<td>80.84</td>
<td>99.06</td>
<td>91.03</td>
<td><strong>58.66</strong></td>
<td>91.01</td>
</tr>
<tr>
<td>0.9</td>
<td>63.47</td>
<td>46.84</td>
<td>99.34</td>
<td><strong>42.50</strong></td>
<td>62.58</td>
</tr>
</tbody>
</table>

Diagnostic Gases data set

3.2 Similarity Functions Impact Discussion

Due to the lack of space, we only give the impact discussion of the similarity functions in the first model MSNR$_{M1}$. Similar results are observed for the second model. The similarity function $H$ aims at finding the set of correlated sensors $C(i)$ or the set of uncorrelated sensors $C'(i)$. $H$ directly determines which sensors are correlated or uncorrelated with the $i$th sensor and the weights of the sensor network regularization terms. Thus, we mainly focus on the analysis of the similarity functions in this subsection. As Table 1 shows, when the missing ratio is below 0.6, PCC obtains lower RMSE for both of the two data sets. PCC takes the different scales among various sensors into consideration, which might contribute to its better performance. However, DTW achieves far superior performance to the other functions when the missing ratio exceeds 0.6. We deduce that DTW can better measure the similarity between two time series when the missing ratio is high, as it utilizes all the observed entities in the raw time series. Nevertheless, both PCC and DTW are alternative similarity functions in the proposed method. In addition, we observe that the constant function CF shows barely satisfactory results, which further demonstrates the necessity and importance of employing an appropriate similarity function.

3.3 Parameters Impact Discussion

In this subsection, we also only give the analyses of the parameters of the first model MSNR$_{M1}$, which is shown in Fig. 2.

Firstly, the parameter $|C(i)|$ denotes the total number of the correlated sensors with the $i$th sensor, which plays a very important role in the proposed method. Taking the Motes data set for example, when $|C(i)|$ is set as 4, the RMSE is equal to 2.76. However, when $|C(i)|$ is equal to 11, the method achieves the lowest RMSE 2.38, which is reduced by about 14%. Specially, the optimum RMSE is about 58% lower than the worst RMSE for the Diagnostic Gases data.
Fig. 2. Impact of Parameters.

set. We deduce that an oversized $|C(i)|$ will bring in more noise while too small a $|C(i)|$ will be not enough to constrain the matrix factorization. Thus, an appropriate value of $|C(i)|$ is of great importance in the proposed method.

Then, the impact of the dimension of $L$ on the performance is also shown in the figure. On the whole, the RMSE is consistently below a reasonable value. Concretely, for the Motes data set, the RMSE is equal to 2.65 when $L$ is set as 1, while the lowest RMSE 2.38 is obtained when $L$ is equal to 4. Nevertheless, based on the experimental results, we may safely set $L = 4$ and $L = 1$ for the Motes data set and the Diagnostic Gases data set respectively. Hence, the dimension of the latent factors $L$ also plays an important part in the proposed method.

Next, the impact of $\alpha$ on the performance is presented. $\alpha$ controls how much information of the sensor network should be incorporated into the optimization problem. In general, as Fig. 2 shows, RMSE not only shows little variation but also is consistently below a relatively low value for all of the different $\alpha$ values. We can observe that the best performance is achieved when $\alpha$ is equal to 0.5 and 0.6 for the two data sets respectively. We deduce that too small an $\alpha$ would greatly decrease the influence of the sensor regularization term on the matrix factorization. On the other hand, if we employ too large an $\alpha$, the sensor regularization term would dominate the learning processes. So, an appropriate coefficient $\alpha$ could further improve the performance of the proposed method.

Finally, the penalty coefficient $\lambda$ is optimized. Based on the experimental results, we can reasonably set $\lambda = 0.4$ and $\lambda = 0.3$ for the Motes data set and the Diagnostic Gases data set respectively.

4 Conclusion

In this paper, we have proposed a novel method MSNR for predicting the missing data in the time series from multiple sources. The method incorporates sensor network regularization terms to minimize the difference between one sensor and its correlated sensors or to maximize the distance between one sensor and
its uncorrelated sensors during matrix factorization. As expected, the proposed method MSNR exhibits higher precision in terms of lower RMSE than classical methods and state-of-the-art matrix factorization based approaches. We deduce that the incorporated network regularization terms mainly contribute to the superior prediction result.

Acknowledgment
This paper is sponsored in part by the National High Technology and Research Development Program of China (863 Program, 2015AA050204), State Grid Science and Technology Project (520626140020, 14H100000552, SGCDK00PJJS1400020), State Grid Corporation of China, the National Research Foundation, Prime Ministers Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, and the National Natural Science Foundation of China (No.61373632).

References
Transfer Learning for Time Series Classification in Dissimilarity Spaces

Stephan Spiegel
DAI Lab, Berlin Institute of Technology
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
spiegel@dai-lab.de
http://www.dai-lab.de/en

Abstract. For many time series classification problems the amount of labeled data is insufficient to achieve satisfying accuracy. However, there exists an abundance of unlabeled time series data, which does not follow the same class labels or generative distribution as the labeled data. We propose to employ this unlabeled data to improve the performance of the supervised learning task, which is also known as transfer learning. In this work use arbitrary UCR time series or random sequences to embed a given set of labeled data into dissimilarity space, leading to enriched feature representations that facilitate statistical learning procedures. Our results show that transfer learning increases the accuracy of time series classification in dissimilarity spaces, which in turn has been shown to outperform the popular 1NN-DTW time series classification approach.

Keywords: Transfer Learning, Dissimilarity Spaces, Time Series

1 Introduction

In time series mining, the Dynamic Time Warping (DTW) distance is a commonly and widely used dissimilarity measure [14]. Its popularity and widespread use are owing to the fact that, in contrast to Euclidean distance (ED), the DTW distance works well for time series with local scaling invariance [1]. The popular combination of the 1-Nearest-Neighbor (1NN) classifier with the DTW distance has been shown to achieve high classification accuracy on time series from various application domains [4].

The 1NN-DTW approach is very intuitive since humans typically compare objects by means of analogies in their structure [6]. But, structural descriptions (such as local scalings captured by DTW) do not match well with statistical learning procedures, which are most powerful for vectorial object representations [5]. Dissimilarity spaces are a promising way to combine structural and statistical pattern recognition approaches [12], where structural descriptions are used to compare objects, leading to a set of pairwise dissimilarities from which vectors can be derived for the purpose of statistical learning [5]. Early work on graphs [2, 15] has shown that the dissimilarity space approach has significant potential to outperform classifiers that directly operate in the graph domain.
Recent work on time series classification [7–9] has proposed to consider DTW distances as feature vectors for standard machine learning methods. The results [7–9] have shown that time series classification in dissimilarity spaces (using the SVM model) is superior to the ‘exceptionally hard to beat’ 1NN-DTW approach [1, 10, 17]. To furthermore increase the classification accuracy of the dissimilarity space approach, it has been proposed to add cDTW and SAX distances to the feature vectors [9] as well as to extract mutually independent features by means of PCA [7, 8]. Related work [3, 16] has confirmed that the SVM model achieves better generalization performance with prior feature extraction.

In this work, we aim at extending the idea of ‘time series classification in dissimilarity spaces’ [7, 8] by means of transfer learning [11]. More precisely, we employ self-taught learning, where unlabeled data is used for the supervised classification task [13]. Self-taught learning does not assume that the unlabeled data follows the same class labels or generative distribution, making it widely applicable to many practical learning problems. In this study, we employ a large number of unlabeled temporal sequences and randomly generated processes to embed a given set of time series and to improve their classification.

The rest of the paper is structured as follows. Section 2 introduces our proposed approach. Section 3 presents empirical results, which are discussed in Section 4. Finally, we conclude with future work in Section 5.

2 Approach

In supervised learning, we are usually given a labeled dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ that consists of $x$ training examples $\mathcal{X} = X_1, \ldots, X_x$ and $y$ test examples $\mathcal{Y} = Y_1, \ldots, Y_y$. The goal is to predict the correct labels for all test examples $\mathcal{Y}$ by generalizing from our training examples $\mathcal{X}$. Most time series classification tasks [4] assume that the test and training examples $X_1, \ldots, X_x, Y_1, \ldots, Y_y \in \mathbb{R}^n$ have the same length or number of dimensions $n$, although this is no necessary requirement for pairwise (dis)similarity comparisons.

In earlier work [7, 8] we have proposed to solve time series classification in dissimilarity spaces. For this purpose, we have used a selected subset of $\alpha$ training examples $\mathcal{A} \subseteq \mathcal{X}$ to embed our original dataset $\mathcal{D}$ into a new feature space. Given a time series $T \in \mathcal{D}$ we obtain its feature representation $(d(T, A_1), \ldots, d(T, A_\alpha))$ by computing the dynamic time warping distance $d(\cdot, \cdot)$ to all training examples contained in $\mathcal{A} = \{A_1, \ldots, A_\alpha\}$. The crux of dissimilarity spaces is that the derived feature vectors reside in Euclidean space and can be used by powerful statistical learning procedures.

In this work aims at studying transfer learning for time series classification in dissimilarity spaces. More precisely, we evaluate the classification accuracy of dissimilarity spaces that were constructed from a set of $\beta$ unlabeled time series $\mathcal{B} = \{B_1, \ldots, B_\beta\}$ that do not follow the same class labels and generative distribution as the labeled data $\mathcal{D}$. Given a time series $T \in \mathcal{D}$ we now obtain its feature representation $(d(T, B_1), \ldots, d(T, B_\beta))$ by computing the dynamic time warping distance $d(\cdot, \cdot)$ to all examples in $\mathcal{B}$.
Of course, we can also concatenate the feature vectors of dissimilarity spaces that were constructed from different datasets. For example, we can use a subset of labeled training examples $A$ as well as some unlabeled time series $B$ plus a set of random sequences $H$ to embed a time series $T \in D$ into the following dissimilarity space: $(d(T, A_1), \ldots, d(T, A_\alpha), d(T, B_1), \ldots, d(T, B_\beta), d(T, H_1), \ldots, d(T, H_\eta))$. In that way, we combine available knowledge from various domains, which can be transferred to our supervised learning task. Please note that we never use the test set for embedding in order to keep it independent from the training set.

**Table 1. Dissimilarity Matrices**

<table>
<thead>
<tr>
<th>Combination of Dissimilarity Matrices</th>
<th>Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>FordA - Train</td>
<td>FordA - Test</td>
</tr>
<tr>
<td>$1320 \times 1320$</td>
<td>$1320 \times 3601$</td>
</tr>
<tr>
<td>$A_x (500 \times 500)$</td>
<td>$A_y (500 \times 500)$</td>
</tr>
<tr>
<td>FordB - Train/Test</td>
<td></td>
</tr>
<tr>
<td>$4446 \times 1320$</td>
<td>$4446 \times 3601$</td>
</tr>
<tr>
<td>$B_x (500 \times 500)$</td>
<td>$B_y (500 \times 500)$</td>
</tr>
<tr>
<td>Ham - Train/Test</td>
<td></td>
</tr>
<tr>
<td>$214 \times 1320$</td>
<td>$214 \times 3601$</td>
</tr>
<tr>
<td>$C_x (431 \times 500)$</td>
<td>$C_y (431 \times 500)$</td>
</tr>
<tr>
<td>Herring - Train/Test</td>
<td></td>
</tr>
<tr>
<td>$128 \times 1320$</td>
<td>$128 \times 3601$</td>
</tr>
<tr>
<td>$D_x (512 \times 500)$</td>
<td>$D_y (512 \times 500)$</td>
</tr>
<tr>
<td>Coffee - Test/Train</td>
<td></td>
</tr>
<tr>
<td>$56 \times 1320$</td>
<td>$56 \times 3601$</td>
</tr>
<tr>
<td>$E_x (286 \times 500)$</td>
<td>$E_y (286 \times 500)$</td>
</tr>
<tr>
<td>Wafer - Test/Train</td>
<td></td>
</tr>
<tr>
<td>$7174 \times 1320$</td>
<td>$7174 \times 3601$</td>
</tr>
<tr>
<td>$F_x (152 \times 500)$</td>
<td>$F_y (152 \times 500)$</td>
</tr>
<tr>
<td>Random Process 1</td>
<td></td>
</tr>
<tr>
<td>$500 \times 1320$</td>
<td>$500 \times 3601$</td>
</tr>
<tr>
<td>$G_x (500 \times 500)$</td>
<td>$G_y (500 \times 500)$</td>
</tr>
<tr>
<td>Random Process 2</td>
<td></td>
</tr>
<tr>
<td>$500 \times 1320$</td>
<td>$500 \times 3601$</td>
</tr>
<tr>
<td>$H_x (500 \times 500)$</td>
<td>$H_y (500 \times 500)$</td>
</tr>
</tbody>
</table>
Table 1 illustrates our approach for the FordA dataset, which consists of 1320 training and 3601 test time series of length 500. If we embed the FordA dataset with all 1320 training examples then we obtain two dissimilarity matrices $Ax$ and $Ay$ of size $1320 \times 1320$ and $1320 \times 3601$. The columns in $Ax$ and $Ay$ represent our generated feature vectors for the FordA training and test examples respectively. Note that each entry in $Ax$ and $Ay$ corresponds to a time series distance, which is the result of finding an optimal path in a $500 \times 500$ warping matrix.

Table 1 furthermore illustrates the dissimilarity matrices for an embedding with two sets of random processes as well as with the FordB, Ham, Herring, Coffee, and Wafer time series [4], using both training and test examples. For each embedding, we denote the size of the resulting dissimilarity matrices (in black color) as well as the size of the corresponding warping matrices (in brackets and gray color). Given all the illustrated dissimilarity matrices (A to H) we can either consider their feature vectors individually or combine their column vectors to a new feature representations. The left side of Table 1 shows the combinations that we evaluate in Section 3.

3 Results

The goal of our evaluation is to assess the time series classification accuracy in consideration of various different dissimilarity spaces, which were constructed from the combinations of labeled training examples, unlabeled time series, and random sequences.

In our experiments we consider the complementary datasets Toe1 and Toe2 as well as FordA and FordB, which are part of the UCR time series archive [4]. For each of the four datasets we assess the classification accuracy for an embedding with the corresponding training set, the respective complementary dataset, arbitrary unlabeled time series, and random (auto-regressive) processes. Furthermore, we assess the classification accuracy for different combinations of the resulting feature vectors, as illustrated by our example in Table 1. The individual or combined feature vectors can subsequently be used as an input for standard statistical learning procedures. In our experiments we employ a linear SVM (using the quadratic programming algorithm of the Matlab optimization toolbox) to solve the classification problem.

Table 2 shows the classification errors for the traditional 1-Nearest-Neighbor classifier (with ED and DTW) in comparison to our proposed approach, using transfer learning in dissimilarity spaces. More precisely, we present classification results for various different dissimilarity spaces that were constructed by means of individual datasets or their combination. In Table 2, an embedding with a combination of different time series is symbolized by a sequence of capital letters, where each letter represent an individual dataset. For instance, the sequence $AB$ describes the combination of the label training examples $A$ and the corresponding complementary dataset $B$. Please note that the classification performance does not depend on the ordering of the combined datasets and does not change for different permutations of the constructed feature vectors.
Furthermore, Table 2 presents the classification results for an embedding with arbitrary datasets \((C=\text{Ham}, D=\text{Herring}, E=\text{Coffee}, \text{and } F=\text{Wafer})\) from the UCR time series archive \([4]\) as well as the two sets of auto-regressive processes \((G=\text{Random Process 1 and } H=\text{Random Process 2})\) that were generated by two different parameters settings\(^1\).

The results in Table 2 shows how transfer learning from complimentary, arbitrary, or random time series influences the classification error. We discuss our interpretation of the empirical results in Section 4.

<table>
<thead>
<tr>
<th>Table 2. Classification Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1NN in Time Series Space</td>
</tr>
<tr>
<td>(\text{Toe1} \quad \text{Toe2} \quad \text{FordA} \quad \text{FordB})</td>
</tr>
<tr>
<td>0.3200 0.1920 0.3410 0.4420</td>
</tr>
<tr>
<td>0.2280 0.1620 0.4380 0.4060</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SVM in Dissimilarity Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{Toe1} \quad \text{Toe2} \quad \text{FordA} \quad \text{FordB})</td>
</tr>
<tr>
<td>0.1404 0.1615 0.2885 0.2990</td>
</tr>
<tr>
<td>0.1140 0.2231 0.2458 0.2624</td>
</tr>
<tr>
<td>0.0877 0.1923 0.2438 0.2525</td>
</tr>
<tr>
<td>0.1930 0.1769 0.3396 0.3707</td>
</tr>
<tr>
<td>0.2632 0.3385 0.4135 0.4497</td>
</tr>
<tr>
<td>0.1798 0.1692 0.3291 0.3804</td>
</tr>
<tr>
<td>0.0877 0.2000 0.2413 0.2486</td>
</tr>
<tr>
<td>0.2675 0.3231 0.4452 0.4568</td>
</tr>
<tr>
<td>0.1316 0.2000 0.3107 0.3328</td>
</tr>
<tr>
<td>0.1711 0.2077 0.3035 0.3342</td>
</tr>
<tr>
<td>0.1535 0.1769 0.2399 0.2371</td>
</tr>
<tr>
<td>0.1447 0.1846 0.2291 0.2368</td>
</tr>
<tr>
<td>0.2237 0.2923 0.3213 0.3377</td>
</tr>
<tr>
<td>0.2719 0.1769 0.4263 0.4054</td>
</tr>
<tr>
<td>0.1930 0.2385 0.3121 0.3276</td>
</tr>
<tr>
<td>0.1491 0.2231 0.2363 0.2428</td>
</tr>
<tr>
<td>0.1535 0.1923 0.2205 0.2280</td>
</tr>
</tbody>
</table>

\(^1\) Set \(G\) and \(H\) each contain 500 random sequences that were generated by an auto-regressive progress \(x_i = ax_{i-1} - bx_{i-2} + c\eta\) of second order, which was initialized with \(x_1 = x_2 = 0\) and stop after 500 time steps. The parameter settings for \(G\) and \(H\) are \(a = 1.8, b = 0.972, c = 0.64\) and \(a = 1.85, b = 0.917, c = 0.76\) respectively. The noise \(\eta\) is a vector of uniformly distributed random numbers in the interval \((0, 1)\)
4 Discussion

Having explained our approach and experimental setup, we are eventually in the position to discuss the empirical results presented in Table 2. In the following Table 3 we compare the classification errors of (i) the traditional 1-Nearest-Neighbor approach using either ED or DTW as competitor - 1NN Baseline [4], (ii) the standard dissimilarity spaces approach using only training examples for embedding - DSS Standard [7, 8], and (iii) our proposed transfer learning in dissimilarity spaces approach using the best combination of constructed feature vectors - DSS Transfer.

<table>
<thead>
<tr>
<th>Toe1</th>
<th>Toe2</th>
<th>FordA</th>
<th>FordB</th>
<th>Toe1</th>
<th>Toe2</th>
<th>FordA</th>
<th>FordB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2280</td>
<td>0.1620</td>
<td>0.3410</td>
<td>0.4060</td>
<td>0.1404</td>
<td>0.1615</td>
<td>0.2885</td>
<td>0.2990</td>
</tr>
<tr>
<td>(+38.42%)</td>
<td>(+0.31%)</td>
<td>(+15.41%)</td>
<td>(+26.36%)</td>
<td>(+61.54%)</td>
<td>(-4.44%)</td>
<td>(+35.34%)</td>
<td>(+43.84%)</td>
</tr>
</tbody>
</table>

As shown in Table 3, for the Toe1 dataset the DSS Standard approach achieved a performance increase of about 38%, while our proposed DSS Transfer approach even achieved a performance increase of more than 61% (relative to the 1NN Baseline). In the case of Toe1, the lowest classification error was given by combining the feature vectors of dissimilarity spaces A and B, which were constructed from the corresponding Toe1 training examples and the complementary Toe2 training and test set (see Table 2).

Our proposed DSS Transfer approach furthermore outperformed the other techniques for the FordA and FordB dataset. For these two datasets, the lowest classification error was given by combining the feature vectors of all examined dissimilarity spaces A−H (refer to Table 2). However, the DSS (Standard and Transfer) approach were not able to achieve a performance increase for the Toe2 dataset, which may be due to the already quite small 1NN Baseline classification error (for Toe2).

In general, the results in Table 2 show that transfer learning from arbitrary or even random dissimilarity spaces is often able to achieve significantly higher classification accuracy than learning from the original training examples. This is astonishing since arbitrary or random dissimilarity representations contain no domain knowledge that relates to the original classification problem. In that sense, transfer learning can be imagined as solving the classification of cats and dogs by means of knowledge about pigs and cows or random mammals.

For instance, Table 2 shows that in the case of Toe1 our DSS Transfer approach achieved a classification error of 0.1140 using only knowledge about the complementary Toe2 set, which equates to a performance increase of exactly 50% with respect to the 1NN Baseline approach using all of the available domain knowledge. Furthermore, in the case of FordB our DSS Transfer approach achieved a classification error of 0.3276 using a combination of random sequences.
from set $G$ and $H$, which is a performance increase of more than 19% in comparison to the *1NN Baseline* approach. Of course, there are also time series datasets, such as *Herring*, which yield dissimilarity representations that result in a performance decrease (for all considered classification problems).

In the following, we discuss how the dimension of the dissimilarity spaces influences the classification accuracy. Table 1 shows that the dimension of our individual dissimilarity spaces correlates with the number of time series that were used for embedding. The question is whether more data leads to better dissimilarity representations and lower classification error? Figure 1 illustrates the classification accuracy for *Toe1* and *Toe2* as a function of the dataset size.

![Classification error for Toe1 and Toe2 with varying dataset size](image)

*Fig. 1.* Classification error for *Toe1* and *Toe2* with varying dataset size, where the time series for the dissimilarity embedding were randomly selected from set $G$ and $H$.

According to our experimental results shown in Figure 1, the classification error converges to a certain lower limit with growing dataset size or embedding dimension. However, the error curves are strongly fluctuating, meaning that in certain cases additional information worsens the classification accuracy and, therefore, more data is not necessarily better. Earlier work [7, 8] suggested to employ dimensionality reduction techniques to identify those time series that add useful information to the dissimilarity representation.
5 Conclusion

We have picked up the idea of time series classification in dissimilarity spaces \cite{7,8} and extended this approach to transfer learning, which allows us to generate enriched dissimilarity representations by considering additional time series that are totally unrelated to the original supervised learning task. Our results show that the proposed approach is able to considerably improve classification accuracy, depending on the additional time series used for the embedding.

In general, it would be advantageous if we could determine the information gain of additional time series datasets beforehand. In future work, we aim to investigate the influence of the (original and transfer) data distribution on the performance increase achieved by the enriched dissimilarity representations.

References

Progressive Temporal Window Widening

David Tolpin
PayPal, dtolpin@paypal.com

Abstract. This paper introduces a scheme for data stream processing which is robust to batch duration. Streaming frameworks process streams in batches retrieved at fixed time intervals. In a common setting a pattern recognition algorithm is applied independently to each batch. Choosing the right time interval is tough — a pattern may not fit in an interval which is too short, but detection will be delayed and memory may be exhausted if the interval is too long. We propose here Progressive Window Widening, an algorithm for increasing the interval gradually so that patterns are caught at any pace without unnecessary delays or memory overflow.

This algorithm is relevant to computer security, system monitoring, user behavior tracking, and other applications where patterns of unknown or varying duration must be recognized online in data streams. Modern data stream processing frameworks are ubiquitously used to process high volumes of data, and adaptive memory and CPU allocation, facilitated by Progressive Window Widening, is crucial for their performance.

Keywords: temporal data streams, sliding windows, stream processing

1 Introduction

We consider here the problem of windowed data stream processing [7]. A data stream is a real-time, continuous, ordered sequence of items. In the windowed setting, the arriving data are divided into windows, either by time interval or by data size, and a pattern recognition algorithm, based on a data mining or machine learning approach, is applied to each window to discover exact or approximate patterns appearing in the window [6]. Here, we view a pattern recognition algorithm as a black box function on stream fragments. For example, a pattern can be an episode — a partially ordered sparse subsequence [10], the language of the text, or the most likely goal of the sequence of actions in the fragment.

Windowed data stream processing is frequently used in computer security [12, 11, 14], user behavior tracking [2], sensor data analysis for system monitoring [3], and other applications. The right choice of window size is crucial for efficient data processing and timely response. Data are divided either into physical windows, by time interval, or into logical, or count-based, windows, by data size or number of records in a single window [7, 6].

The choice of either physical or logical windows depends both on properties of the data stream and on the objective of the data processing algorithm. Logical
windows are more naturally handled by machine learning algorithms with inputs of fixed size [6], while physical windows allow both more efficient processing and faster online response [9, 16, 15]. This paper explores selecting a window size for physical, interval-based windows. The dilemma behind selecting a window size which inspired this research is

– whether to choose a smaller window and sacrifice context, such that no single window contains a complete pattern,
– or to increase the window size at the cost of increased consumption of computational resources and delayed response.

This dilemma is relevant to many applications of data stream processing, but in particular to security applications [12, 11, 14], where an adversary aware of the maximum window time interval can escape the detection algorithm by introducing delays between data stream entries (such as transactions or web site accesses) which exceed the interval and prevent detection. Even if the maximum duration of a pattern is known in advance, setting the window size to exceed the maximum duration means that recognition of any shorter pattern will be delayed.

To address this dilemma, we introduce an algorithm which we call Progressive Window Widening (PWW). PWW processes the data stream through an array of sliding windows of increasing physical size, such that shorter patterns are recognized sooner, however windows covering longer patterns are also applied to the stream. Despite employing several window sizes in parallel, PWW still remains efficient in CPU and memory consumption. The paper proceeds as follows: first, necessary preliminaries are introduced in Section 2. Then, the algorithm is described and analysed (Sections 3 and 4), as well as evaluated empirically (Section 5). Finally, related work is reviewed, and contribution and future research are discussed (Sections 6 and 7).


2 Preliminaries

2.1 Batched Stream Processing

In batched stream processing, which we adopt in this paper as a lower level for PWW, stream data arrives in batches — sequences of fixed duration. Several batches can be combined into a window of size equal to the total size of the batches composing the window. Along with batch size (or duration, used interchangeably here), a batch is characterized by its length, the number of atomic elements, or records, it contains. For example, a one-minute batch of web site log stream may contain 1000 entries — we shall say that the size, or duration of the batch is 1 minute, and the length of the batch is 1000 entries.

Further on, we extend the note of batched stream processing by stating that a data stream with batch duration $t$ may be transformed into a data stream with batch duration $kt$ by concatenating each $k$ consecutive batches together.
Denoting a batch of the original stream with batch duration $t$ by $B_{i,l}$ and a batch of the combined stream with batch duration $kt$ by $B_{i+1,j}$ for some $i$, $j$, and $l$, one may write ($\circ$ stands for batch concatenation):

$$B_{i+1,j} = B_{i,kj-k+1} \circ B_{i,kj-k+2} \circ \cdots \circ B_{i,kj} \quad \forall j \in \mathbb{N}^+ \quad (1)$$

2.2 Sliding Windows

Depending on the overlay between windows, one discerns between tumbling (there are gaps between windows), jumping (the windows are adjacent), and sliding (overlapping) windows [7]. PWW is based on sliding windows; the next window starts earlier than the current window terminates.

Sliding windows have several uses. We are interested in one particular case: sliding windows with a half-size overlap; the feature we are interested in is described by Lemma 1:

**Lemma 1** A sequence of sliding windows of size $2b$ with overlap $b$ covers any interval of size at most $b$.

A corollary of Lemma 1 is that if we want to recognize patterns of duration at most $t$, it is sufficient to use sliding windows of size $2t$ with half-size overlap.

3 Progressive Window Widening

We introduce here **Progressive Window Widening**, an algorithm for progressive widening of temporal windows. To define the algorithm efficiently, we rely on an auxiliary notion of $L_{\text{max}}$ — the maximum length of a data sequence which may contain a pattern. For example, if a game player must complete each game round in 20 moves, then any pattern pertaining to a single round must be contained within 20 moves. Alternatively, $L_{\text{max}}$ can be chosen such that the probability of a random occurrence of the pattern in a data sequence of length $L_{\text{max}}$ is sufficiently low [8].

The algorithm processes the data stream in parallel, through multiple asynchronous sliding windows of different sizes.

3.1 Algorithm Outline

PWW (Algorithm 1) performs the following operations:

1. Recursively combines pairs of adjacent batches, doubling batch duration of each stream and creating a stream with batches of double duration (line 3).
2. Runs a detection algorithm in a sliding window on each stream (line 6).
3. While combining batches, discards subintervals of combined batches which cannot intersect a yet unseen pattern (see Section 3.2 for detailed explanation).
Algorithm 1 Progressive Window Widening
1: procedure PWW(S -- stream, t -- batch duration)
2: Sleep(t)
3: Create stream $S'$ from $S$ with batch duration $2t$ (see Algorithm 2)
4: Call PWW($S'$, $2t$) asynchronously
5: for each sliding window $W$ in $S$ do
6:  if patterns present in $W$ then
7:    Output detected patterns

As the algorithm runs, multiple batched streams are created, and sliding windows move through each of the streams (Figure 1). The algorithm relies on asynchronous recursive calls to PWW (line 4). Asynchronous calls are possible because the processing of each stream is independent. Such asynchronous execution is particularly suitable for modern multi-core multi-node cluster architectures: different invocations of PWW may be executed on different cores or different nodes in the cluster.

Note that extra streams are created (lines 2–4) and processed (line 5) with exponentially increasing delays, since a window can be processed only upon termination of the window’s interval.

3.2 Combining batches

An integral part of PWW is the optional discarding of a subinterval while combining two subsequent batches. For every stream of batches of duration $t$, the algorithm waits $2t$ time units for 2 batches to arrive. Then, a stream of base duration $2t$ is formed by combining the batches (Algorithm 2). PWW combines batches by concatenation (line 2). If the length of the combined batch is greater than $2L_{\text{max}}$, the middle part of the combined batch is discarded (Figure 2), leaving subsequences of length $L_{\text{max}}$ at both ends of the batch (lines 3–4). Consequently, no batch in any stream is longer than $2L_{\text{max}}$. The subintervals may
be discarded because a combined batch at the next level coincides with a sliding window at the current level, so new patterns may be discovered only between batches, rather than within a single batch.

4 Algorithm Analysis

We launch an unbounded number of parallel processes, and want to show that PWW runs in computationally bounded resources. The work that the algorithm performs is assumed to take place inside a pattern recognition algorithm run on each sliding window. Let us denote the resources (a combination of memory and amount of work) required to run a certain pattern recognition algorithm on window of length $l$ by $R(l)$. Then, the following theorem holds:

**Theorem 1** Denote by $t$ the batch duration of the initial, uncombined stream. Assume that the maximum length of a batch of the initial stream does not exceed $2L_{\text{max}}$. Then the average resources $\rho$ per unit time required to run PWW are bounded by a constant:

$$\rho \leq \frac{2R(4L_{\text{max}})}{t}. \quad (2)$$

In practice, the number of parallel streams may be bounded. However, even if unbounded, average resources required to run the algorithm are bounded.

5 Case Study: Detecting Remote Shells in a System Call Stream

In this case study, we monitor an online stream of system calls from a network-connected server, and want to detect possible invocations of remote shells as soon as possible. System call sequences corresponding to remote shell invocations can be interspersed with unrelated activities.

For simplicity, we limit detection to a single episode which may correspond to accepting a network connection and then launching a shell communicating with the remote user through the connection:

1. accept $fd=x \Rightarrow y$
2. $dup \; fd=y \Rightarrow 0 \mid dup \; fd=y \Rightarrow 1 \mid dup \; fd=y \Rightarrow 2$
3. $execve \; exe=z$
In the above pseudocode, system call name is followed by `name=value` argument pairs and then by return value preceded by `=>`. In a matching system call sequence $y$ must have the same value in lines 1 and 2, three system calls in line 2 may be executed in any order, and $x, z$ may take any value. For example, sequence

```
accept fd=5 => 6, dup fd=6 => 2, dup fd=6 => 1, dup fd=6 => 0, execve exe=sh
```

matches the episode.

For the empirical evaluation we use a sequential version of PWW which facilitates easy estimation of the amount of work. We set $L_{max} = 100$ because malicious code is often transmitted in a single packet with only a few dozens of instructions. We use a stream of 10,000 system calls recorded on a Linux machine, into which we inject episode instances with varying delays between instructions.

We find that:

- The detection delay (Figure 3) is proportional to the episode duration with factor 0.5.
- The amount of work (Figure 4) approaches but stays below bound (2).

![Fig. 3. Detection delay.](image1)

![Fig. 4. Amount of work.](image2)

Both results are in accordance with the algorithm analysis. If a fixed window were used, either the average detection delay would grow, or some episodes were left undetected.

The source code, data, and results for the case study are available at [https://bitbucket.org/dtolpin/pww-paper-case-studies](https://bitbucket.org/dtolpin/pww-paper-case-studies). The evaluation notebook can be viewed in the browser at [http://tinyurl.com/jgknulz](http://tinyurl.com/jgknulz).

## 6 Related Work

While Progressive Window Widening can be implemented from scratch on low-level data streams, the algorithm was inspired and relies in implementation on batched stream processing. Batch stream processing was introduced in Comet [9]. Apache Spark offers Spark Streaming [16, 15], a powerful implementation of programming model *discretized streams*. Discretized streams, which enable efficient batch processing in parallel architectures, is the enabling lower level for PWW.

PWW uses varying window sizes to accommodate for differences in data. Another approach in batched stream processing is to use *adaptive window size*. 

91
Adaptive window algorithms is a field of active research [17, 4, 5, 13]. However, this research represents a different approach, in which the window size is changed sequentially and adaptively, for future windows based on earlier seen data. In PWW, several windows of fixed sizes are applied in parallel, in a parameter-free manner suitable for simple and robust implementation. Windows of doubling size were proposed for processing data streams in earlier work [1], however the approach employed in PWW is significantly different in that temporal windows of unbounded doubling durations are applied in parallel, while still ensuring efficient use of resources.

7 Contribution and Future Research

This paper introduced the Progressive Window Widening algorithm for data stream processing using temporal sliding windows. The algorithm

- solves the dilemma of smaller window size at a cost of inability to recognize longer patterns versus larger windows but slower response;
- works in parallel, in a manner suitable for modern multi-core multi-node cluster architectures;
- uses computational resources efficiently, imposing only a constant factor overhead compared to an algorithm based on a single window size.

The basic algorithm described in the paper brings a solution to the stated problem. At the same time, the algorithm design poses a number of questions and opens several research directions.

- Many adaptive window algorithms are, unlike PWW, essentially sequential. Modern data frameworks provide an opportunity to exploit the parallelism for more flexible and efficient adaptation.
- Doubling of batch durations is chosen in PWW due to simplicity of implementation and analysis. A different allocation of window sizes, either data-independent or adaptive, may bring better theoretical performance and practical results.
- PWW relies on batched stream processing, however it is only loosely coupled with the underlying computing architecture, which is both an advantage and a drawback. A tighter coupling with lower-level stream processing may be helpful.

Along with others, these directions are deemed to be important for future research.

References


