Time Series Prediction from Multiple Factors

Perrine Cribier-Delande[1,2], Raphael Puget[2] and Vincent Guigue[1],
Ludovic Denoyer[1] *

1- MLIA, Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

2 - Renault, DEA-IR, Technocentre, 1 avenue du Golf 78084 Guyancourt , France

Abstract. We propose a new neural architecture to predict time series, each depending on multiple underlying factors. Our method typically applies to spatio-temporal prediction of missing series where only certain locations and times are observed. The model is based on an encoder-decoder structure where the multiple factors are projected into a latent space which is learned by combining the latent factors coming from multiple observed series. We show on several spatio-temporal datasets that our method is able to predict missing series, not only for observed factors values, but also for new ones (e.g new locations or times).

1 Introduction

Modeling time series is an important issue in machine learning with many concrete applications that received a lot of attention during the last decades. Usually, the latter problem is handled mainly as an imputation problem (i.e., completing missing values in an observed series as in [1]) or as a forecasting one (i.e., being able to predict the "future" of any new series as in [2]). We focus on a different task: predict missing times series conditioned on latent factors of variations. More precisely, we consider the problem where time series are organised among two different factors of variation[1]. For instance, such a setting can correspond to spatio-temporal time series where each series is associated with a particular location (factor 1) and has been captured for a particular day (factor 2). In this context, it is usual to have no observations for some combinations of factors values particularly for acquisition cost reasons. A critical problem is thus to be able to generate the **missing series** based on the observed ones.

The main difficulty in this setting is to learn from observations the influence of each factor in the time series. This problem is usually denominated as the *disentanglement* problem in the literature and has been studied in different application domains, and particularly in computer vision. For instance, multiple models have been proposed to generate images based on class and view [3].

Recent works on disentanglement and generation are using two techniques : Variational AutoEncoder (VAEs, [4, 5]) and Generative Adversarial Network (GANs, [6]). More recently, disentanglement techniques have been extended to video and speech where the temporal component is one of the considered factors. For instance, [7] presents a VAE-based generative model, that explicitly

*This work is partially supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 780754, "Track & Know".

[1]Our method can easily apply to more than two factors.

learns to disentangle content and dynamic to perform "feature sweeping" and generates speech with a different voice than the one of the original speaker. Using adversarial training, [8] and [9] predict the next frame of a video by using different auto-encoders architectures and disentangling dynamics and content. Also, [10] used Kalman filters in the latent space of embeddings to generate videos of a physical system such as a pendulum or a bouncing ball.

In contrast, our approach is focused on data of a different nature, i.e., classical time series, and we propose to study a different setting. Indeed, we tackle two tasks: i) the *missing series* prediction problem where missing series correspond to unobserved series for known values of the two factors (e.g., for known locations, and past days) and the ii) *new series* prediction problem where one aims at predicting series for unknown factor values (e.g., for new locations and/or new days) based on the observation of a single series. To this end, we propose firstly a novel neural model where each series can be predicted based on embeddings representing the different factors. Secondly, we extend this model to consider new and unobserved factor values by completing our architecture with an encoder able to compute a factor's embedding just based on the only observation of any new single time series. We demonstrate that our model can successfully predict missing time series on three different datasets, and achieve good performance in comparison to oracle and non-oracle baseline methods.

2 Notations and Tasks

We consider the problem where one has access to a set of temporal series $x_{i,j}$ with i and j corresponding to the values of two different factors of variations characterising the observed series. When considering for instance a city subway network, $x_{i,j}$ corresponds to the amount of ticket validations each hour during day i in a particular subway station j. For sake of simplicity, each temporal series will be of fixed size T such that $x_{i,j} \in \mathbb{R}^T$. We also consider that training series are provided for specific combinations of $i \in [1..N]$ and $j \in [1..M]$ as illustrated in Figure 1 (left). The observed series are indexed through a mask m such that $m_{i,j} = 1$ if $x_{i,j}$ is observed (and is in the training set), and $m_{i,j} = 0$ if $x_{i,j}$ is not observed[2]. We consider two different tasks:

Missing Series Prediction: The objective of this task is to predict the values of the temporal series that are not in the training set. This task can be seen as a matrix completion problem where missing entries are temporal series. It typically applies to applications, where for cost reasons, time series have not been collected on all combinations of factors.

New Series Prediction: The objective of this task is to predict series for new factor values i.e., new values of i or j that do not appear in the observed set of series. It corresponds to a setting where one observes a new series[3] $x_{\hat{i},j}$ such that $j \in [1..M]$ but where $\hat{i} \notin [1..N]$. For instance a series for a new subway station

[2]Note that in the experimental section, this split will also include a validation set for hyper-parameters tuning.

[3]It also applies to the problem where $x_{i,\hat{j}}$ is the newly observed series.

that has never been observed in the past. In this case, the goal is to predict all the missing series for that particular factor's value $\hat{i}$, i.e., all the $x_{\hat{i},k}$ for $k \neq j$.

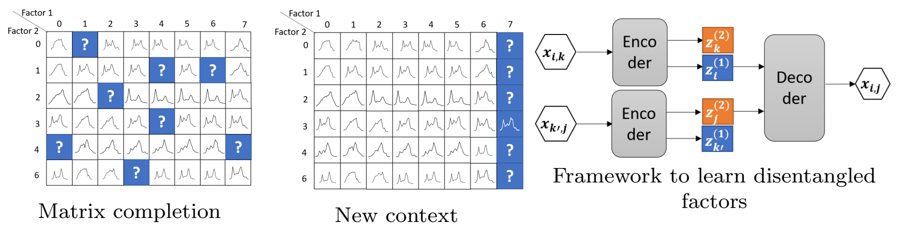

Fig. 1: [Left] Our dataset with the different splits corresponding to our 2 tasks. [Right] Architecture to learn disentangled factors.

3 Model

Both settings can be solved if we are able to capture from the training set the influence of each factor i and j in $x_{i,j}$. We now detail how we propose to achieve this disentanglement for both instances of the problems described above. The principle of our approach is to compute a representation of each factor of variation in a latent space, and to use these representations to infer any series values. Let us consider two latent spaces $\mathbb{R}^{\ell_1}$ and $\mathbb{R}^{\ell_2}$ of size ℓ_1 and ℓ_2. Each value of the 1st factor i (respectively of the 2nd factor j) will be represented by a vector $z_i^{(1)}$ (respectively $z_j^{(2)}$). Based on these representations, the series $x_{i,j}$ will be inferred through a decoder function $d : \mathbb{R}^{\ell_1} \times \mathbb{R}^{\ell_2} \rightarrow \mathbb{R}^T$. Any series $x_{i,j}$ can then be computed through $d(z_i^{(1)}, z_j^{(2)})$.

Let us consider $e_1 : \mathbb{R}^T \rightarrow \mathbb{R}^{\ell_1}$ a function that, given any time series is able to compute the representation of the first factor of this series i.e., given a series $x_{i,j}$, $e_1(x_{i,j})$ computes the representation $z_i^{(1)}$. Similarly, $e_2 : \mathbb{R}^T \rightarrow \mathbb{R}^{\ell_2}$ will take in charge the second factor such that $e_2(x_{i,j})$ computes $z_j^{(2)}$. Given a new series $x_{\hat{i},\hat{j}}$, considering that there exists two series $x_{\hat{i},k}$ and $x_{k',\hat{j}}$ that are observed, $x_{\hat{i},\hat{j}}$ can then be predicted through $x_{\hat{i},\hat{j}} = d(e_1(x_{\hat{i},k}), e_2(x_{k',\hat{j}}))$. The knowledge of d, e_1 and e_2 allow us to predict any new series provided that at least another series is observed for each factor values $\hat{i}$ and $\hat{j}$. Learning such a model enable us to solve the two particular tasks presented above.

Learning through disentanglement

Learning d, e_1 and e_2 is not trivial and typically cannot be achieved through a classical auto-encoding loss applied on each individual training series since, in that case, the model would be unable to disentangle which information comes from the first factor of variation, and which information comes from the second. We thus propose to rely on a learning objective that have been proposed in the disentanglement literature [9]. The principle of the learning method is to minimise the loss between a predicted series by using embeddings values computed

Name	Description	#locations	#days
STIF	Number of people that enters Paris's subway using smart cards	298	89
Energy	Hourly consumption of energy in the North East of the USA [4]	10	1877
NO_2	Hourly measurement of NO_2 in the air of Madrid [5]	24	2669

Table 1: Time series datasets with two explanatory factors

from other series as illustrated in Figure 1 (right).

Let us consider a triplet $\tau = (x_{\hat{i},\hat{j}}, x_{\hat{i},k}, x_{k',\hat{j}})$ containing only series observed at train time i.e., $m_{\hat{i},\hat{j}} = m_{\hat{i},k} = m_{k',\hat{j}} = 1$. Given this triplet, one can compute a loss function $\mathcal{L}(\tau, d, e_1, e_2) = \Delta(d(e_1(x_{\hat{i},k}), e_2(x_{k',\hat{j}})), x_{\hat{i},\hat{j}})$ where Δ is a loss function measuring the distance between the two series, such as mean-squared error . The final functions d^*, e_1^* and e_2^* are thus obtained by solving:

$$d^*, e_1^*, e_2^* = \arg\min_{d,e_1,e_2} \frac{1}{|\mathcal{T}|} \sum_{(x_{\hat{i},\hat{j}}, x_{\hat{i},k}, x_{k',\hat{j}}) \in \mathcal{T}} \Delta(d(e_1(x_{\hat{i},k}), e_2(x_{k',\hat{j}})), x_{\hat{i},\hat{j}}) \tag{1}$$

where $\mathcal{T}$ is the set of all triplets that can be built from the training set. When considering that d, e_1 and e_2 are differentiable parameterised functions (e.g neural networks), this problem can be solved by classical stochastic gradient descent techniques.

Inference: At test time, a series $x_{\hat{i},\hat{j}}$ can be predicted through $x_{\hat{i},\hat{j}} = d(e_1(x_{\hat{i},k}), e_2(x_{k',\hat{j}}))$, provided that some $x_{\hat{i},k}$ and $x_{k',\hat{j}}$ are observed. However, it may happen that multiple series $x_{\hat{i},k}$ could be observed for different values of k (and respectively for k' and $x_{k',\hat{j}}$). In this case, we propose two inference methods that are compared in the next section:

Series averaging: The predicted series are the average of the series predicted by using all the combinations of observed $x_{\hat{i},k}$ and $x_{k',\hat{j}}$.

Embedding averaging: The predicted series are computed by averaging at the embedding levels i.e., by averaging the values of $e_1(x_{\hat{i},k})$ and $e_2(x_{k',\hat{j}})$ on the observed series, and by using these vectors as inputs to the decoding function d.

4 Experiments

4.1 Dataset

We tested our models on three different datasets. They all consist in daily measures of some quantity on several locations. The data is aggregated hourly, giving 24 time steps for each series ($T = 24$). The first factor corresponds to the location of the series and the second factor corresponds to the day when the series was acquired. Table 1 describes the different datasets. These datasets correspond to very different applications: from public transportation traffic to energy consumption and air quality measurements. However, they share common properties: stationarity for week days and specific operating modes for the week-ends. Each time series value was normalised between 0 and 1 by using the min/max values after removing the 99.9 percentile to remove outliers.

[4]https://www.kaggle.com/robikscube/hourly-energy-consumption/

[5]https://www.kaggle.com/decide-soluciones/air-quality-madrid/

4.2 Baselines and Models

We compare our models to different baselines:

Baseline average (BL avg) This baseline is considered for both tasks. It predicts $x_{\hat{\imath},\hat{\jmath}}$ by averaging the observed time series sharing a common factor $\hat{\imath}$ or $\hat{\jmath}$.

Baseline p-Nearest Neighbour (BL NN(p)) This intuitive approach is dedicated to the 2^{nd} task. Given $x_{\hat{\imath},k}$, we aim at predicting all other $x_{\hat{\imath},\cdot}$. The principle is to find the p series that are the closest to $x_{\hat{\imath},k}$ from $\{x_{\cdot,k}\}$. Let's define $\mathcal{U}$ the set of indices of these series: $\{x_{u,k}\}_{u\in\mathcal{U}}$. Then, $x_{\hat{\imath},\hat{\jmath}}$ is predicted with the average of the p series $\{x_{u,\hat{\jmath}}\}_{u\in\mathcal{U}}$

Oracle p-Nearest Neighbour (Oracle NN(p)) The third baseline is an oracle since it requires to access the ground truth. $x_{\hat{\imath},\hat{\jmath}}$ is simply estimated by averaging its p nearest neighbours in the trainset.
p is a parameter of the baselines we optimise.

For our model, we explore different possibilities to build the functions d, e_1 and e_2. The decoding function will be either a Multi-Layer Perceptron or a one-dimensional convolutional neural network. Concerning the encoding function, we also relied on both MLP and CNN. We also investigate the specific case where e_1 and e_2 are matrices of embeddings (i.e., one vector for each possible factor value) which only applies for the *missing series* problem since the *new series prediction* task needs to be able to compute values for new factors.

Hyper-parameters have been chosen on a validation subset of 20% of the series; the evaluation has been made on the test set, comprised of another 20% of the series[6].

4.3 Experimental Results

Table 2 presents all results for the first task using the MSE over the predicted series vs the ground truth. Performances are consistent across the different use cases: the oracle provides a nearly perfect reconstruction while our proposals outperform the baselines. In the transductive setting (when using embeddings matrices), we need to provide to the model exact information on the real day and location, therefore it always outperforms the inductive framework (where e_1 and e_2 are not embeddings matrices).

Table 3 illustrates the MSE on the *new series prediction* task. Our results are still consistent except for new locations on the Energy and NO_2 datasets: this can be explained by the very low number of locations preventing the model to capture the relevant information.

5 Conclusion & Perspectives

We have proposed a new approach for predicting time series when the underlying observations are organised following two factors of variations. This approach ap-

[6]MLP design counts 1 or 2 layers with 128 or 256 hidden neurons and tanh activation functions. CNN relies on 32 or 64 filters of size 3, 5 or 7 depending on the dataset. Parameters are optimised by random search on the validation set. Results are reported on the test set.

		STIF		Energy		NO2	
Encoder	Decoder	Series avg	Emb. avg	Series avg	Emb. avg	Series avg	Emb. avg
Oracle NN(p)		0.001(p=9)		0.00008 (p=11)		0.0031 (p=5)	
BL avg (fact. 1)		0.0179		0.0060		0.0213	
BL avg (fact. 2)		0.0121		0.0089		0.0099	
Emb	MLP	**0.0028**		**0.0020**		**0.0098**	
Emb.	CNN	0.0053		0.0048		0.0101	
MLP	MLP	0.0084	0.0084	0.0046	0.0047	0.0106	0.0105
CNN	CNN	0.0073	0.0073	0.0060	0.0060	0.0104	0.0105

Table 2: MSE errors for task 1 (matrix completion setting).

			STIF		Energy		NO2	
	Encoder	Decoder	Series avg	Emb. avg	Series avg	Emb. avg	Series avg	Emb. avg
New loc.	BL NN		0.0123 ($p = 150$)		0.0081 ($p = 3$)		0.0107 ($p = 5$)	
	BL avg (factor 2)		0.0125		0.0087		0.0112	
	MLP	MLP	**0.0059**	0.0059	0.0118	0.0118	0.0107	0.0107
	CNN	CNN	0.0075	0.0076	**0.0099**	0.102	**0.0101**	0.0102
New day	BL NN		0.0143 ($p = 5$)		0.093 ($p = 900$)		0.0207 ($p = 60$)	
	BL avg (factor 1)		0.0145		0.0089		0.0210	
	MLP	MLP	0.0063	0.0061	0.0076	0.0076	0.0133	0.0133
	CNN	CNN	0.0061	**0.0059**	0.0065	**0.0063**	0.0129	**0.0128**

Table 3: MSE errors for task 2: new locations above, new days below.

plies to different types of applications, and particularly to spatio-temporal problems where series are organised based on locations and schedule. While simple, our approach offers good performance on the studied datasets, including the STIF dataset which corresponds to a real-world application. Future extensions of this method includes dealing with series of variable length, and performing evaluation on problems with more than two factors of variations.

References

[1] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. *NIPS*, 2015.

[2] A. Ziat, E. Delasalles, L. Denoyer, and P. Gallinari. Spatio-temporal neural networks for space-time series forecasting and relations discovery. *ArXiv*, abs/1804.08562, 2018.

[3] M. Chen, L. Denoyer, and T.Artières. Multi-view data generation without view supervision. *arXiv*, abs/1711.00305, 2017.

[4] D J Rezende, S Mohamed, and D Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv*, abs/1401.4082.

[5] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, 2013.

[6] I. J. Goodfellow, J. Pouget-Abadie, M.i Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[7] Y. Li and S. Mandt. A deep generative model for disentangled representations of sequential data. *arXiv*, abs/1803.02991, 2018.

[8] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. *ArXiv*, abs/1706.08033, 2017.

[9] E. Denton and V. Birodkar. Unsupervised learning of disentangled representations from video. *NIPS*, 2017.

[10] Fraccaro M., Rezende D. J., Zwols Y., Pritzel A., Eslami S.M.A., and Viola F. Generative temporal models with spatial memory for partially observed environments. In *ICML*, 2018.