

# LINMA 2120 - Seminars in Applied Mathematics

## High dimensional time-series prediction with missing values

Quentin L  t  

November 2017

### 1 Introduction

Let us first give a definition of time series to explain the formal context in which we work.

**Definition 1.** Time series A time series is a sequence of data indexed by the time.

A time series is thus a realization of a stochastic process which can be seen as a family of random variables indexed by the time.

We also define here a property of a stochastic process that will be useful later : the Markov property.

**Definition 2.** Markov property We say that  $(Y_t)_{t \geq 0}$  is a Markov chain if for all  $t > 0$

$$\pi(y_t | y_{1:t-1}) = \pi(y_t | y_{t-1})$$

where  $\pi$  is the probability.

This means that all the information about the past is completely carried out in the  $Y_t$  at each step. With Markov chains, the joint distribution of observations take a fairly simple form :

$$\pi(y_{1:t}) = \pi(y_1) \cdot \prod_{j=2}^t \pi(y_j | y_{j-1})$$

### Notation

Time series can be multidimensional, i.e. be a vector of several quantities that evolve with time. In that case, every such quantities are called streams. In this work, we will denote time series as matrices for which each line correspond to a stream and each column to the measurements at a fixed time. For instance, we could have a time series  $Y \in \mathbb{R}^{n \times T}$ . In this case, the columns of  $Y$  will be noted in lowercase and bold form indexed on the time  $t$ , i.e.  $\mathbf{y}_t$ . The  $i$ -th row of  $Y$  will be noted  $Y_{i:}$  and correspond to the  $i$ -th stream.

### Imputation vs Interpolation

There are two ways of using the data provided to make inference on another value of the time series.

**Definition 3.** Imputation is using only the values of a fixed time to make predictions on another value of this particular time.

**Definition 4.** Interpolation is using the data of a single stream to infer on another value of this stream.

## 2 Classical methods

In this section, we present classical methods used for time series forecasting. These classical methods include AR (autoregressive) models and DLM (dynamic linear models) and are used for prediction. Let us explain sequentially these models.

### Autoregressive models

As explained in [1], the idea of this model is to represent each observation as a noisy linear combination of previous observations. Formally, let  $\mathcal{L}$  be the set of time indices that encodes the dependence between the observations over time. If  $\mathbf{x}_t \in \mathbb{R}^k$  is the measurement at time  $t$  and if the lag  $|\mathcal{L}| = p$ , the AR(p) model parametrized by the coefficient matrix  $\mathcal{W} = \{W^{(l)} \in \mathbb{R}^{k \times k} : l \in \mathcal{L}\}$  can be written as

$$\mathbf{x}_t = \sum_{l \in \mathcal{L}} W^{(l)} \mathbf{x}_{t-l} + \epsilon_t \quad (1)$$

where  $\{\epsilon_t\}_{t \in \mathbb{Z}}$  is a Gaussian noise vector with for simplicity  $\epsilon_t \sim \mathcal{N}(0, \sigma^2 I_k)$ .

This model is motivated by a theorem due to Wold that states that a stationary process can be represented by a MA( $\infty$ ) model, that is

$$\mathbf{x}_t = \sum_{i=1}^{\infty} \beta_i \epsilon_{t-i} + \epsilon_t$$

where  $\sum_{i=1}^{\infty} \beta_i < \infty$ . And if  $\{\mathbf{x}_t\}_t^{\infty}$  is invertible, we also have that it can be represented by an AR( $\infty$ ) model, that is

$$\mathbf{x}_t = \sum_{i=1}^{\infty} \alpha_i \mathbf{x}_{t-i} + \epsilon_t$$

With this theorem, it seems natural to use AR( $p$ ) models for prediction.

### Dynamic Linear Models

DLM is a simpler model of a more general framework called state space models.

**Definition 5.** State space model A state space model consist of two time series : a  $\mathbb{R}^p$ -valued time series  $\{\mathbf{x}_t\}$  and a  $\mathbb{R}^m$ -valued time series  $\{\mathbf{y}_t\}$  satisfying the following assumptions :

- $(\mathbf{x}_t)$  is a Markov chain
- Conditionnaly on  $(\mathbf{x}_t)$ , the  $\mathbf{y}_t$ 's are independent and the  $\mathbf{y}_t$ 's depends only on  $(\mathbf{x}_t)$

The idea behind this model is that the serie  $\mathbf{y}_t$  is determined by a latent process  $\mathbf{x}_t$ . The  $\mathbf{x}_t$ 's usually represent all the observable physical variables that have an influence on the variable of interest. For instance, if we try to predict the price of electricity in the future, the latent variables could be the temperature, the wind speed, the sunshine, ... Note that this  $(\mathbf{x}_t)$  is assumed to be a Markov chain.

In general, a state space model consists in two equations : an **obsvervation equation** which gives  $\mathbf{y}_t$  in function of  $\mathbf{x}_t$  at each step and an **evolution equation** which gives  $\mathbf{x}_t$  in function of  $\mathbf{x}_{t-1}$ . Both equations are pertubed by noise. This can be written as

$$\mathbf{y}_t = f_t(\mathbf{x}_t, v_t)$$

$$\mathbf{x}_t = g_t(\mathbf{x}_{t-1}, w_t)$$

To be complete we also have to specify the prior distribution for  $\mathbf{x}_0$ .

With that general state space model defined, it is possible to define the DLM which is a special case of it with linear functions and Gaussian noise.

**Definition 6.** DLM A dynamic linear model (DLM) is specified by a Normal prior distribution for the  $p$ -dimensional state vector at time  $t = 0$ ,

$$\mathbf{x}_0 \sim \mathcal{N}_p(m_0, C_0)$$

together with a pair of equations for each time  $t \geq 1$ ,

$$\begin{aligned} \mathbf{y}_t &= F_t \mathbf{x}_t + v_t, & v_t &\sim \mathcal{N}_m(0, V_t) \\ \mathbf{x}_t &= G_t \mathbf{x}_{t-1} + w_t, & w_t &\sim \mathcal{N}_p(0, W_t) \end{aligned}$$

where  $F_t$  and  $G_t$  are known matrices of order  $m \times p$  and  $p \times p$  respectively.

## Forecasting

Based on this model, how can we predict the future observation ? The problem is solved by use of the **Kalman filter**. This filter is based on the idea that, due to the markovian property of the state vector and the assumption of conditional independence of the observations, the density of the state and the observation knowing their previous values can be computed recursively. Moreover, because the equations are linear and the noise and prior distribution are Gaussian, it is possible to prove that the random vector  $(\mathbf{x}_0, \dots, \mathbf{x}_t, \mathbf{y}_0, \dots, \mathbf{y}_t)$  follows a multivariate Gaussian distribution. Therefore, only the mean and covariance matrix has to be computed. The Kalman filter gives the recursion equation for them :

**Theorem 1.** Let  $\mathcal{D}_t$  be the information provided by the first  $t$  observations  $\mathbf{y}_1, \dots, \mathbf{y}_t$ . Then, if

$$\mathbf{x}_{t-1} \sim \mathcal{N}(m_{t-1}, C_{t-1}),$$

where  $t \geq 1$ , then

(a) the one-step-ahead predictive density of  $\mathbf{x}_t$  given  $\mathcal{D}_{t-1}$  is Gaussian with parameters

$$\begin{aligned} a_t &= E(\mathbf{x}_t | \mathcal{D}_{t-1}) = G_t m_{t-1} \\ R_t &= \text{Var}(\mathbf{x}_t | \mathcal{D}_{t-1}) = G_t C_{t-1} G_t' + W_t \end{aligned}$$

(b) the one-step-ahead predictive density of  $\mathbf{y}_t$  given  $\mathcal{D}_{t-1}$  is Gaussian with parameters

$$\begin{aligned} f_t &= E(\mathbf{y}_t | \mathcal{D}_{t-1}) = F_t a_t \\ Q_t &= \text{Var}(\mathbf{y}_t | \mathcal{D}_{t-1}) = F_t R_t F_t' + V_t \end{aligned}$$

(c) the filtering density of  $\mathbf{x}_t$  given  $\mathcal{D}_t$  is Gaussian with parameters

$$\begin{aligned} m_t &= E(\mathbf{x}_t | \mathcal{D}_t) = a_t + R_t F_t' Q_t^{-1} e_t \\ C_t &= \text{Var}(\mathbf{x}_t | \mathcal{D}_t) = R_t - R_t F_t' Q_t^{-1} F_t R_t \end{aligned}$$

where  $e_t = \mathbf{y}_t - f_t$  is the forecast error.

This theorem gives us everything we need to predict the behaviour of the observations for a time series modeled by a DLM.

It can be seen that the complexity of this algorithm is  $\mathcal{O}(kn^2T + k^3T)$ . This rather high complexity can become prohibitive when dealing with high-dimensional data. For instance, it is noted in [2] that using a common package in  $R$  for DLM, problems with  $n \geq 32$  cannot be solved.

Another problem with the two classical methods that we have described is that it is not clear how missing data would be dealt with. Yet, missing data are common in many application involving time series and can be due to occlusions or malfunction of sensor, among others.

We will now present the novel approach to make predictions about high-dimensional time series published in [2].

### 3 Temporal Regularized Matrix Factorization

This section aims to describe the method developed in [2].

#### Framework

The authors place themselves in a framework similar to the state space model defined in definition 5. The difference is that here  $\mathbf{x}$  is not supposed to respect the Markov property. Moreover, the observation equation is assumed to be linear. Formally, this can be described by the following equations :

$$\mathbf{y}_t = F\mathbf{x}_t + \eta_t \quad (2)$$

$$\mathbf{x}_t = M_\Theta(\{\mathbf{x}_{t-l} : l \in \mathcal{L}\}) + \epsilon_t \quad (3)$$

The vector noise  $\eta_t$  and  $\epsilon_t$  are supposed to be Gaussian. These equations are parametrized by sets :  $\mathcal{L}$  that contains the number of previous state vectors on wich  $\mathbf{x}_t$  depends and  $\Theta$  that contains the weights associated to each vector in time.

#### Matrix factorization formulation

The primary obsevation made by the authors is that when we stack all  $\mathbf{x}_t$  in a matrix  $X$  and if we do the same with the  $\mathbf{y}_t$  in the matrix  $Y$ , we can see that  $Y \approx FX$ . Of course, noise has been omitted in this equation and this is why it is not an equality.

With this observation, our problem becomes a problem of matrix factorization which has been widely studied in the past. For instance, this is a key element in some recommender systems and the use of matrix factorization in this context to perform machine learning has been used in the well-known Netflix prize in 2007. In the context of the Netflix prize, the incomplete matrix of ratings was decomposed into a matrix for the users and a matrix for the items. Those two matrices were learned by minimizing an mean square error on the data.

A similar approach can be used in our context. If we denote by  $\mathbf{f}_i^\top$  the  $i^{\text{th}}$  row of the matrix  $F$  and by  $\Omega$  the set of observed entries, we could learn the  $X$  and  $F$  factors as

$$\min_{F, X} \sum_{(i,t) \in \Omega} (Y_{it} - \mathbf{f}_i^\top \mathbf{x}_t)^2 + \lambda_f \mathcal{R}_f(F) + \lambda_x \mathcal{R}_x(X) \quad (4)$$

where  $\mathcal{R}_f(F)$  and  $\mathcal{R}_x(X)$  are regularization factors on  $F$  and  $X$  respectively. These factors are important to prevent overfitting and to take into accounts possible dependencies between the different entries of the matrices. The link that has just been established between the general framework and matrix factorization approaches is important but it still does not tell us how to use equation 3 to account for the temporal dependencies. We also need to describe how forecasting could take place in this framework.

In classical MF applications like the one mentioned earlier for recommender systems, the regularization factors are often taken as the norm of the matrix (for example the Frobenius norm) to penalize high values of the learned elements. However, here, this choice would not be a good idea as it would not take into account the temporal dependencies that naturally exists in the  $\mathbf{x}_t$ s. In [2], the authors propose a choice of regularization factors which remedies that problem.

## Temporal regularization

The goal in this section is to describe a choice of regularization factor that is called  $\mathcal{T}_M(X|\Theta)$  that will promote the relationship between the  $\mathbf{x}_t$ s given in (3). Recall that  $\Theta$  in our model contains the information that links the  $\mathbf{x}_t$ s together. The idea proposed is to take the log-likelihood of the  $\mathbf{x}_t$ s knowing  $\Theta$ . This is formalized by the following equation :

$$\mathcal{T}_M(X|\Theta) = -\log \mathbb{P}(\mathbf{x}_1, \dots, \mathbf{x}_T|\Theta)$$

It can be seen that the lower this factor is, the more plausible it is to observe this set of  $\mathbf{x}_t$ s according to (3). Note that the logarithm in this equation is there for computational reasons only. If  $\Theta$  is given, then we can use directly  $\mathcal{T}_M(X|\Theta)$  as the regularization factor for  $X$ . When it is not given, a regularization for  $\Theta$  should be added to 4 which becomes

$$\min_{F, X, \Theta} \sum_{(i,t) \in \Omega} (Y_{it} - \mathbf{f}_i^\top \mathbf{x}_t)^2 + \lambda_f \mathcal{R}_f(F) + \lambda_x \mathcal{T}_M(X|\Theta) + \lambda_\theta R_\theta(\Theta) \quad (5)$$

It is important to note that with this formulation,  $\Theta$  can thus be learned from the data like  $F$  and  $X$ . This is another important advantage that TRMF has compared to other approaches like graph-based regularization. Once  $\Theta$  is learned, we can use it in our model and use equation (3) to predict that next latent vector  $\mathbf{x}_t$ , which can in turn be used to predict  $\mathbf{y}_t$  as  $F\mathbf{x}_t$ .

Another feature that was missing from the classical method was the possibility to deal missing entries. With the TRMF technique, we can simply estimate the missing entries of  $Y$  as  $Y_{it} = \mathbf{f}_i^\top \mathbf{x}_t$ . This is one big advantage of matrix factorization and it is central to many of its application like recommender systems.

Finally, the MF framework can also be used to perform clustering. Again, this is similar to what has already been proposed in classical use of MF. The  $\mathbf{f}_i$  vector can be interpreted as the latent embedding factor of the  $i^{\text{th}}$  time series in  $Y$  and thus clustering algorithms can directly be applied to these vectors.

## Extensions

This general TRMF framework can be conveniently extended to account for different applications dependent extensions. In this section, we present 3 useful ones.

**Known features** In the classical machine learning framework, every item is associated with a time invariant feature vector. In the study of time series, it is common that in addition to the time dependent data, some known features are available on the objects and it is important to be able to incorporate them in the model to improve performances. Our TRMF framework can be easily extended to account for this additional feature vector for item  $i$  denoted  $\mathbf{a}_i$  as follows :

$$\min_{F, X, \Theta} \sum_{(i,t) \in \Omega} (Y_{it} - \mathbf{a}_i^\top F \mathbf{x}_t)^2 + \lambda_f \mathcal{R}_f(F) + \lambda_x \mathcal{T}_M(X|\Theta) + \lambda_\theta R_\theta(\Theta) \quad (6)$$

This formulation has two main advantages :

- A new time series  $Y_{i't}$  can now be estimated without knowledge of any additional observation up to time  $T$ . We can simply and directly estimate it as  $Y_{i't} = \mathbf{a}_i'^\top F \mathbf{x}_t$  if the feature vector is available
- If  $\mathbf{a}_i \in \mathbb{R}^d$ , this has also the effect of reducing the dimensions of  $F$  from  $n \times k$  to  $d \times k$  which can improve the efficiency of the algorithm.

**Graph information among time series** Sometimes, a graph based information is available on the time series. This graph encodes relationship between them. The goal is to define a regularization on  $F$  that will tend to promote the structure defined by the graph. Given a graph  $G$  on the time series with weights  $G_{ij}$  on the edge  $i - j$ , we can take the following graph regularization factor :

$$\mathcal{G}(F|G, \eta) = \frac{1}{2} \sum_{i \sim j} G_{ij} \|\mathbf{f}_i - \mathbf{f}_j\| + \frac{\eta}{2} \sum_i \|\mathbf{f}_i\|$$

Our problem then becomes simply :

$$\min_{F, X, \Theta} \sum_{(i,t) \in \Omega} (Y_{it} - \mathbf{f}_i^\top \mathbf{x}_t)^2 + \lambda_f \mathcal{G}(F|G, \eta) + \lambda_x \mathcal{T}_M(X|\Theta) + \lambda_\theta R_\theta(\Theta) \quad (7)$$

**Temporal-regularized tensor factorization** It is also possible to extend our framework to tensor factorization with evolution over time. To take again the example of recommender system, it is sometimes useful to model the variation over time of the preferences of a user or to model some kind of fashion effect associated to a movie. In the classical MF framework, we thus incorporate time labelled data. If  $P$  denotes the matrix of latent embeddings of the users and  $Q$  of the movies, our model can be extended as :

$$\min_{P, Q, X, \Theta} \sum_{(i,j,t) \in \Omega} (Y_{ijt} - \langle \mathbf{p}_i, \mathbf{q}_j, \mathbf{x}_t \rangle)^2 + \lambda_p \mathcal{R}_p(P) + \lambda_q \mathcal{R}_q(Q) + \lambda_x \mathcal{T}_M(X|\Theta) + \lambda_\theta R_\theta(\Theta) \quad (8)$$

where  $\langle \mathbf{p}_i, \mathbf{q}_j, \mathbf{x}_t \rangle$  is defined as  $\sum_r p_{ir} q_{jr} x_{tr}$

## Autoregressive Temporal Regularization

TRMF have been described until here using a very general evolution equation (3). In this section, we will apply the TRMF framework on an autoregressive model for the latent embeddings. Let us recall equation (1) which defines the AR model for the time series  $\mathbf{x}_t$  parametrized by the lag set  $\mathcal{L}$  and the weights  $W^{(l)}$  :

$$\mathbf{x}_t = \sum_{l \in \mathcal{L}} W^{(l)} \mathbf{x}_{t-l} + \epsilon_t$$

We would like to define a regularization factor on  $X$  that promotes this model for the  $\mathbf{x}_t$ . A natural choice would be the following :

$$\mathcal{T}_{AR}(X|\mathcal{L}, \mathcal{W}, \eta) = \frac{1}{2} \sum_{t=m}^T \left\| \mathbf{x}_t - \sum_{l \in \mathcal{L}} W^{(l)} \mathbf{x}_{t-l} \right\|^2 + \frac{\eta}{2} \sum_t \|\mathbf{x}_t\|^2 \quad (9)$$

where  $m := 1 + \max(\mathcal{L})$  and the last term has been added to ensure strong convexity.

As explained previously, the TRMF framework allows us to learn the weight matrices  $W^{(l)}$  when they are not known. However, as  $W^{(l)} \in \mathbb{R}^{k \times k}$ , the number of weights to be learned would be  $|\mathcal{L}|k^2$  which can be large and lead to overfitting. We could reduce the amount of weights by assuming that the  $W^{(l)}$  are diagonal. This reduces the number of weights to  $|\mathcal{L}|k$ . This corresponds to assuming that each element of the observation vector  $\mathbf{y}_t$  is only influenced by the corresponding elements in the observations  $\{\mathbf{y}_{t-l}\}_{l \in \mathcal{L}}$

This has the effect to decouple each time series from each other (when we talk about each time series, we mean each line of the  $X$  or the  $Y$  matrices). Let us rewrite  $\mathcal{T}_{AR}$  in a way that enhances this decoupled structure. Let us denote  $\mathcal{W}$  as the  $k \times L$  matrix where the  $l$ -th column in the diagonal of  $W^{(l)}$  and note that for  $l \notin \mathcal{L}$ , the  $l$ -th column of  $\mathcal{W}$  is a zero vector. Let  $\bar{\mathbf{x}}_r^\top = [..., X_{rt}, ...]$  be the  $r$ -th row of  $X$  and  $\bar{\mathbf{w}}_r^\top = [..., \mathcal{W}_{rl}, ...]$  be the  $r$ -th row of  $\mathcal{W}$ . Then, equation (9) can be written as

$$\begin{aligned}\mathcal{T}_{AR}(X|\mathcal{L}, \mathcal{W}, \eta) &= \sum_{r=1}^k \mathcal{T}_{AR}(\bar{\mathbf{x}}_r|\mathcal{L}, \bar{\mathbf{w}}_r, \eta) \\ \mathcal{T}_{AR}(\bar{\mathbf{x}}_r|\mathcal{L}, \bar{\mathbf{w}}_r, \eta) &= \frac{1}{2} \sum_{t=m}^T \left( x_t - \sum_{l \in \mathcal{L}} w_l x_{t-l} \right)^2 + \frac{\eta}{2} \|\bar{\mathbf{x}}_r\|^2\end{aligned}$$

Let us also note that when  $L = 1$ , we have exactly the Dynamic Linear Model with a general weight matrix  $W^{(1)}$ .

### Optimizing TRMF with AR Temporal Regularization

Using  $\mathcal{T}_{AR}$  as the regularization factor for  $X$  in equation (5), we get the following optimization problem :

$$\min_{F, X, \Theta} \sum_{(i,t) \in \Omega} (Y_{it} - \mathbf{f}_i^\top \mathbf{x}_t)^2 + \lambda_f \mathcal{R}_f(F) + \lambda_x \sum_{r=1}^k \mathcal{T}_{AR}(\bar{\mathbf{x}}_r|\mathcal{L}, \bar{\mathbf{w}}_r, \eta) + \lambda_w R_w(\mathcal{W}) \quad (10)$$

The authors in [2] propose to solve (10) with alternating minimization. Therefore, the minimization of the parameters when the others are fixed reduces to well-known methods.

## Conclusion

In this section, we have presented a new method to deal with time varying data, i.e. Temporal Regularization Matrix Factorization. As its name indicates, this method is based on Matrix Factorization techniques and makes use of a particular regularization factor to account for temporal dependencies between the data.

The general framework was first discussed. Then, the method was specified in the case of an  $AR$  process for the latent embedding.

The main advantages of TRMF is that it is able to deal with missing data by estimating them with the latent embeddings provided by Matrix Factorization. This method has also the advantage of being more scalable than classical methods. Finally, with the framework described, the exact equation for the latent vector  $\mathbf{x}_t$  must not be known *a priori*.

## 4 Application

As stated above, TRMF can be used in various contexts that include time-varying data. One import sector that deals with this kind of data is the energy sector. For instance, in electricity market modelling, it is of tremendously important to be able to forecast the electricity demand or the production of renewable energy to be able to adapt the centralized production. In [3], this idea of using TRMF to forecast electrical quantities is developed. The framework used is similar to [2] while being a bit simpler. Matrix factorization is used to predict new values but the temporal regularization is ignored. The idea is that we have a dataset containing history of values of a certain electrical quantity. If we arrange this data in matrix form, future values can be predicted by minimizing the error of the matrix factorization on known entries, similarly to what was proposed for Matrix Factorization in the context of recommender systems.

To be slightly more formal, let us apply the idea on an example. We have data that corresponds to wind speed at each hour of the day from 1986 to 1995. We suppose that data of the year 1995 are missing and that we wish to recover or predict them. We build the matrix  $Y$  as the matrix with 3652 rows that correspond to the 3652 days

of the years 1986 to 1995 and 24 columns at each hour in these days. The entry  $Y_{i,t}$  is the wind speed that was recorded at hour  $t$  of day  $i$ . If the wind speed for the last hour of the last day is missing, the matrix look like

$$Y = \begin{bmatrix} Y_{1,1} & \dots & Y_{1,24} \\ \vdots & \ddots & \vdots \\ Y_{3652,1} & \dots & ? \end{bmatrix}$$

Let  $\Omega$  be the set of row-column pairs that have a known entry in the matrix  $Y$ . The matrices  $F \in \mathbb{R}^{3652 \times k}$  and  $X \in \mathbb{R}^{k \times 24}$ , where  $k$  is the length of the latent embedding and is a parameter of the method, are determined by the following optimization problem

$$\min_{F, X} \sum_{(i,t) \in \Omega} \left( Y_{i,t} - \mathbf{f}_i^\top \mathbf{x}_t \right)^2 + \lambda (\|\mathbf{f}_i\|^2 + \|\mathbf{x}_t\|^2)$$

This optimization problem can be solved by stochastic gradient descent. Then,  $?$  is estimated as  $? \approx \mathbf{f}_{3652}^\top \mathbf{x}_{24}$ . It is shown in [3] that this approach can yields better predictions than ARIMA (Autoregressive integrated moving average) models in certain situations.

It should be noted that the approach presented in [3], although using the same idea as [2] of Matrix Factorization for time-series prediction, does not develop any Temporal Regularization technique that could improve the results. Except the idea of the application to electrical quantities forecasting, it does not provide new ideas compared to the basic framework of matrix factorization for prediction that was developed during the Netflix prize.

## 5 Deep learning methods

Another class of methods designed to handle time varying data are methods based on deep learning. Deep learning techniques construct what is called a Neural Network, i.e. a model inspired by the human brain to perform machine learning tasks. In its simplest form, it embeds a directed-graph structure composed of several layers of nodes (neurons) provided with biases/thresholds and functions of activation (similarly to real neurons). Nodes are connected to each other by edges (axons) provided with weights.

Several people have designed Neural Networks to deal with times series prediction. In this section, we present two of such methods.

### Multi-directional Recurrent Neural Networks

This method is developed in [4].

## References

- [1] O. Anava, E. Hazan, and A. Zeevi, "Online time series prediction with missing data," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2191–2199. [Online]. Available: <http://proceedings.mlr.press/v37/anava15.html>
- [2] H. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization," *CoRR*, vol. abs/1509.08333, 2015. [Online]. Available: <http://arxiv.org/abs/1509.08333>
- [3] F. Lecron, "Using matrix factorisation for the prediction of electrical quantities," *CIREN - Open Access Proceedings Journal*, vol. 2017, pp. 2568–2571(3), October 2017. [Online]. Available: <http://digital-library.theiet.org/content/journals/10.1049/oap-cired.2017.1229>



- [4] J. Yoon, W. R. Zame, and M. van der Schaar, “Multi-directional recurrent neural networks: A novel method for estimating missing data,” 2017.