Horizon 2020 Framework Programme
Grant Agreement: 732328 – FashionBrain

Document Information

Deliverable number: D1.4
Deliverable title: Software Requirements: SSM library for time series modelling and trend prediction
Deliverable description: Most modern algorithms of State Space Models (SSM) for time series analysis and probabilistic inference will be summarised in this deliverable and will be used as a basis for future software developments in the project. The output will be available for project internal and public use.
Due date of deliverable: 31/12/17
Actual date of deliverable: Revised version submitted 20/04/18
Authors: Duncan Blythe, Alan Akbik, Roland Vollgraf
Project partners: Zalando
Workpackage: WP1
Workpackage leader: Roland Vollgraf
Dissemination Level: Public

Change Log

Version	Date	Status	Author (Partner)	Description/Approval Level
1	31/12/17	Final	Zalando	Public

Abstract

This deliverable outlines a suite of algorithms and sample code for a time-series package to be integrated into MonetDB. We review traditional approaches to time-series analysis and outline a state-of-the-art recurrent neural method using long-short term memory networks and sequential importance resampling.

1 Introduction

Time-series analysis is the task of modeling sequential data where the sequence reflects the unfolding of time. In the fashion industry, these time-series may arise in reference to customers (clicks-per-customer on types of item), in reference to market demand (purchases fluctuating over time in a given sector), or in reference to products (number of mentions over time of specific products on social media channels).

Typical tasks in time-series analysis [16] include:

- **Forecasting (F)**: predicting a subsequent value of a quantity given past values.
- **Distributional modeling (DM)**: modeling the full process giving rise to empirical time-series.
- **Dependency analysis (DA)**: understanding the relationship between two time-series from data.
- **Missing value estimation (MV)**: imputing missing or unavailable time-series points from observed points.

In the fashion industry **F** may be used to predict load and consequently perform load balancing to cope with the phenomenon of fluctuating sales. **DM** may be used to check how well a specific customer's data corresponds to modeling assumptions and also as preprocessing step to a subsequent data scientific pipeline.

DA may be used to understand the inter-relationship between diverse business sectors, such as country, gender and type-of-product, and how these evolve over time. **MV** be used to infer lost or unavailable measurements of specific customers at a range of time points.

Difficulties typically hampering traditional approaches to the above tasks include:

1. Models are often overly simple, e.g. employing strong linear and/or Gaussian assumptions [1, 13, 21].

2. Models usually ignore changing trends over time, i.e. non-stationarities and cannot be adapted to model long-range dependencies [13, 18, 27].

3. Models lack a probabilistic component, not yielding estimates with confidence estimates of their certainty [11, 21].

In fashion e-commerce these difficulties are particularly urgent since:

1. Time-series such as transactional click-data exhibit a high-degree of non-linearity and non-Gaussianity.

2. Fashion data is intrinsically seasonally and responds to emerging trends. Thus non-stationarity must be incorporated into models.

3. Weighted decisions, which allow a strategy to applied probabilistically, in the absence of certainty, is vital to yielding financially optimal decisions.

The aim of this deliverable is to outline a suite of state-of-the-art algorithms time-series algorithms for incorporation into the MonetDB framework, allowing time-series measures for **F**, **DM**, **DA** and **MV** to be employed in state-of-the-art information retrieval.

The proposals must thus fulfill the following **requirements**:

A. The algorithms must cope with problems 1 to 3.

B. The framework must be self-contained and unified as to admit plausible incorporation into MonetDB within the scope of the project.

C. Sample code should be provided in `python` as a blueprint for the MonetDB solutions.

In this report we summarize a suite of algorithms and explain how they fulfill these requirements:

A. We use a state-of-the-art probabilistic state-space model based on recurrent neural networks (RNNs) and an accompanying training framework to solve problems 1 to 3.

B. The RNN model has the universal approximation property (is highly flexible) and may be trained in a unified manner to solve **F**, **DM**, **DA** and **MV**; the only difference between the tasks consists in data-preparation. This allows for simple adaptation to the MonetDB framework.

C. An open-source `python` library `probrnn` is freely available for download and may be used as a basis for further progress in the project.

2 Background

A time-series [16] is an indexed sequence of random variables $X_1, X_2, \dots, X_t := X_{1:t}$ which is distributed according to some underlying join probability distribution: X

$$X_{1:t} \sim P(X_1, X_2, \dots, X_t)$$

Typically several sample paths of such a time-series are observed: $X^i_1, X^i_2, \dots, X^i_t$ for $i = 1, \dots, n$ and t an arbitrary end time point.

2.1 Forecasting (F)

Formally, forecasting consists of producing a good estimate of $E(X_t|X_1, \dots, X_{t-1})$, or more generally $P(X_t|X_1, \dots, X_{t-1})$.

2.2 Distribution modeling (DM)

Formally, distribution modeling consists of estimating $P(X_1, X_2, \dots, X_t)$.

2.3 Dependency analysis (DA)

Dependency analysis consists in estimating values of one time-series from another:

$$P(Y_t|X_{1:t}, Y_{1:t-1})$$

2.4 Missing value estimation (MV)

In missing value estimation we are given a sample path with missing values:

$$X^i_1, X^i_2, ?, ?, X^i_5, ?, \dots, X^i_t$$

The task is to estimate these missing values.

3 Prior Work

Traditionally methods for **F**, **DM**, **DA** and **MV** have typically drawn from one or more of the following restrictive **assumptions**:

AS1. The best estimate is based on a linear relationship.

AS2. The noise innovations around the true model are Gaussian.

AS3. The marginal distributions across time are constant (stationarity).

AS4. Time-dependence ranges over a finite number of time-steps (no long-range dependence).

AS5. Negligible loss in performance is achieved by neglecting the confidence of an estimate.

3.1 Forecasting (F)

The simplest of models for time-series forecasting are the linear autoregressive model [16] (AR):

$$X_t = \sum_{i=1}^{k} \varphi_i X_{t-i} + c + E_t$$

and the moving average model [16] (MA):

$$X_t = \sum_{i=1}^{k} \theta_i E_{t-i} + \mu + E_t$$

The Greek letters apart from E are parameters of the model and E is typically assumed to be white noise.

These models are particularly restrictive, clearly using AS1.–AS5.. Often hard and fast estimates are based on this model, so that in practice AS5. is also applied.

In order to overcome this restrictiveness, innumerable extensions of these models have been proposed, with increasingly complex acronymcs: ARMA, GARMA, ARIMA, FARIMA, ARCH, GARCH, FIGARCH, NGARCH and so forth [16, 20, 2, 22, 4].

ARMA combines the advantages of AR and MA models, thus is more flexible, but still subject to AS1.–4.. ARIMA extends ARMA using a specific non-stationarity assumption, thus partially solving AS3. but still subject to the remaining assumptions. FARIMA extends ARIMA in a way to partially solve AS5. but in a parametric fashion. The other variants add a property know as "autoregressive conditional heteroskedasticity" solving the non-stationarity problem in a slightly different way taylored especially to financial data. NGARCH solves the non-linearity problem using a highly specific model of non-linearity. All of these extensions contain an implicit Gaussianity assumption, AS2.

3.2 Distribution modeling (DM)

Standard approaches to distribution modeling make parametric assumptions about the joint distribution $\mathrm{P}(X_1, \ldots, X_t)$. For example one may assume that $X_{1:t}$ is distributed according to a multivariate Gaussian [12]:

$$X_{1:t} \sim \mathrm{N}(\mu, \Sigma)$$

. More flexible approaches relax the strict Gaussianity assumption, for instance assuming a mixture of Gaussians [3]:

$$X_{1:t} \sim \sum_i w_i \mathrm{N}(\mu_i, \Sigma_i)$$

or by assuming a fully non-parametric model, using for instance a kernel density estimator [23].

Although these latter methods are in principle flexible enough to handle arbitrary distributions, they do not scale well to high dimensional data. The former methods, on the other hand are overly restrictive.

3.3 Dependency analysis (DA)

Traditional approaches to dependency analysis use highly parametric assumptions about the type of of dependency, assuming linearity (correlation analysis) [6], Gaussianity (Pearson correlation analysis) [6], or specific assumptions about parameters (LASSO [25], elastic net [28]). Few of these dependency analysis methods fully relax the linearity assumption, allowing for arbitrary dependencies to be modeled – general methods such as mutual information require a huge dataset size to be of practical interest [7]. Moreover, few of these methods are specifically tailored to time-series analysis, where the dependency between time-points can hamper analysis of the strength of dependencies. Accordingly in the study of time-series many authors use a model discussed in the context of **F**and examine the performance of estimators, such as correlation conditional on this modeling assumption.

3.4 Missing value estimation (MV)

There are two varieties of missing value estimation methods: parametric and non-parametric. In the parametric case, one typically assumes a model, and performs expectation-maximization (EM) to estimate the missing values, performing updates using the modeling assumption [9].

In the non-parametric or semi-parametric framework typically one-assumes that the data-generating distribution is degenerate in some-sense and uses this degeneracy assumption to obtain estimates for missing values which are valid given for a range of models; this variety includes compressed sensing [10] and low-rank matrix completion [5].

4 Proposals

We now outline our suite of time-series algorithms. Let $f(X_t, h_t)$ be the recurrence relation of a long-short term memory recurrent neural network (LSTM) [17], with an embedding layer to the input of the LSTM. Then we make the following approximation:

$$P(X_t | X_1, \ldots, X_{t-1}) \approx P(X_t | X_{t-1}, h_t)$$

where:

$$h_t = f(X_{t-1}, h_t)$$

LSTMs help to avoid the assumptions AS1.–5., with the following **solutions**:

SOL1. They are capable of modeling arbitrary non-linear relationships [24].

SOL2. In combination with a binning approach and a softmax layer on the output, they may cope with arbitrary noise distributions [14].

SOL3. They do not *a priori* assume stationarity, since the time-step is implicit in h_t.

SOL4. Long range dependence may be modeled in a manner superior to traditional neural networks, potentially spanning hundredes of time-steps [17].

SOL5. They may be parameterized in a probabilistic manner as described above [19, 15].

We now describe how training the network f to model $P(X_t|X_{t-1}, h_t)$ may be used to perform **F**, **DM**, **DA** and **MV**. A schematic of our approach is displayed in Figure 1.

4.1 Forecasting (F)

The sample is split into windows $X^i_1, \ldots, X^i_t$ and the neural network is trained to maximize $P(X^i_t|X^i_{t-1}, h_t)$. If the windows are consecutive in time, the final hidden state from a previous window is passed to the next window as the initial state.

4.2 Distribution modeling (DM)

The LSTM is trained as before but with a learnt and constant initial hidden state. The approximate joint is then given by:

$$\begin{aligned} P(X_{1:t}) &= \prod_{i=1}^{t} P(X_i|X_{1:i-1}) \\ &\approx \prod_{i=1}^{t} P(X_i|h_i) \end{aligned}$$

This approach is known as neural autoregressive distribution estimation [26].

4.3 Dependency analysis

The LSTM network is trained to predict the next value of the sequence:

$$X_1, Y_1, X_2, Y_2, \ldots, X_t$$

At inference time, a particle filtering approach is applied using sequential importance resampling [8] to predict Y_t from previous values.

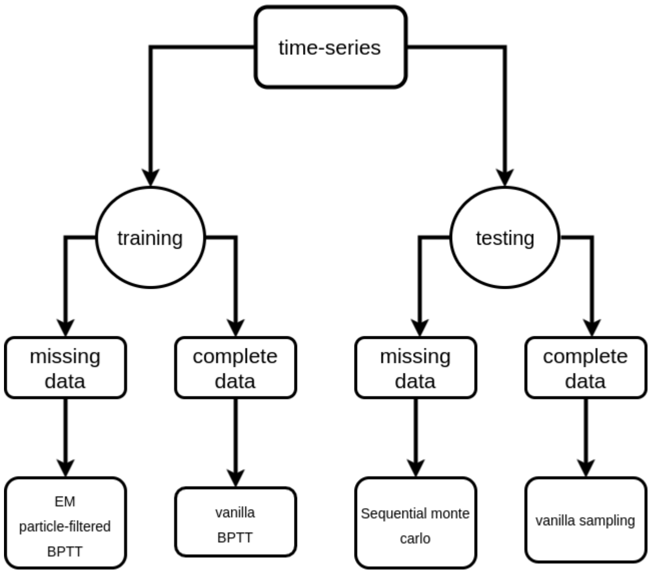

Figure 1: Schematic of the proposed approach

4.4 Missing value estimation (MV)

A particle filter in combination with expectation maximization on the LSTM model is used to maximize the probability of the observed sample paths.

5 Code

Code is available here.

Installation:

```
git clone git@github.com:zalandoresearch/probrnn.git
cd probrnn/
make install
```

Testing:

```
make clean
make test
```

Set up **F** data:

```
from probrnn import data
import numpy as np

x = np.random.randn(100000)
datastruct = data.TimeSeries(x)
```

Set up **DM** data:

```
from probrnn import data
import numpy as np

x = np.random.randn(1000, 10)
datastruct = data.NadeWrapper(x)
```

Set up **DA** data:

```
from probrnn import data
import numpy as np

x = np.random.randn(1000)
y = np.random.randn(1000)
z = np.zeros(0000)
z[::2] = x
z[1::2] = y
datastruct = data.TimeSeries(z)
```

Get a forecasting model:

```
from probrnn import models

model = models.TimeSeriesPrediction(datastruct, params=params)
```

Get a distribution estimation model:

```
from probrnn import models

model = models.NADE(datastruct, params=params)
```

Do the training, save the model, and the log the training.

```
training = models.Training(model, "test_model", "test_log.json")
callback = lambda err, i, _: print "loss: {err};".format(err=err)
training.train(callback)
```

Same thing but with missing values:

```
from probrnn import inference

imputer = lambda a, b: inference.NaiveSIS(a, b)
training = models.Training(
    model,
    "test_model",
    "test_log.json",
    imputer=imputer
)
training.train(callback)
```

Fill in missing values at test time:

```
x[np.random.choice(len(x), replace=False, size=50)] = np.nan
estimate = imputer(model, x).estimate()
```

6 Conclusion

We have provided an overview of traditional approaches to time-series analysis and outlined a suite of state-of-the-art algorithms using probabilistic state-space recurrent neural networks. The simplicity and generality of the approach should allow efficient incorporation in the MonetDB codebase.

References

[1] Hirotugu Akaike. Fitting autoregressive models for prediction. *Annals of the institute of Statistical Mathematics*, 21(1):243–247, 1969.

[2] Michael A Benjamin, Robert A Rigby, and D Mikis Stasinopoulos. Generalized autoregressive moving average models. *Journal of the American Statistical association*, 98(461):214–223, 2003.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

[5] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[6] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[7] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[8] Nando De Freitas, C Andrieu, Pedro Højen-Sørensen, M Niranjan, and A Gee. Sequential monte carlo methods for neural networks. In *Sequential Monte Carlo methods in practice*, pages 359–379. Springer, 2001.

[9] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[10] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[11] James Durbin and Siem Jan Koopman. Time series analysis of non-gaussian observations based on state space models from both classical and bayesian perspectives. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(1):3–56, 2000.

[12] David A Freedman. *Statistical models: theory and practice*. cambridge university press, 2009.

[13] Clive WJ Granger. Causality, cointegration, and control. *Journal of Economic Dynamics and Control*, 12(2-3):551–559, 1988.

[14] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[15] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[16] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Søren Johansen. Statistical analysis of cointegration vectors. *Journal of economic dynamics and control*, 12(2-3):231–254, 1988.

[19] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[20] Alan Pankratz. *Forecasting with univariate Box-Jenkins models: Concepts and cases*, volume 224. John Wiley & Sons, 2009.

[21] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[22] Peter M Robinson. *Time series with long memory*. Advanced Texts in Econometrics, 2003.

[23] Murray Rosenblatt et al. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[24] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150, 1995.

[25] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[26] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016.

[27] Paul Von Bünau, Frank C Meinecke, Franz C Király, and Klaus-Robert Müller. Finding stationary subspaces in multivariate time series. *Physical review letters*, 103(21):214101, 2009.

[28] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.