Anomaly Detection in Multivariate Time Series Using Generative Adversarial Networks

Emil Andersson

LUND UNIVERSITY

Faculty of Engineering, Centre for Mathematical Sciences, Mathematical Statistics

MSc Thesis TFRT-9999
ISSN 0280–5316

Faculty of Engineering, Centre for Mathematical Sciences, Mathematical Statistics
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

A condensed description of my work.

Acknowledgements

These people helped me a lot with my work.

Contents

1 Introduction

Unexpected events, or anomalies, are happening everywhere in daily life. In many cases the anomalies are unexpected events in time that are not similar to what has happened in the past. Anomaly detection in time series has historically been a common topic for research with univariate data in smaller amounts. Detecting unexpected heart beats is a clear example of this kind. However, with the rise of the Internet of Things (IoT), where numerous sensors simultaneously measure values, and with the rapid increase of computational power, the research into how to find anomalies in multivariate data has increased. The data that should be processed is high-dimensional, the amount of data is very big and in many cases the anomalies are not labeled. These kind of sensor systems appear in many real-world applications such as space shuttles, health care monitoring and in factories. In those systems, a lot of multivariate time series data is generated continuously, and it is often of relevance to predict unexpected events to be able to handle the problems before they appear. This could be in the form of predictive maintenance in factories to find problems before a failure.

1.1 Anomaly detection in time series

When handling time series it is important to catch the temporal dependencies between time steps. Anomalies can be defined as a lack of temporal continuity in the long or short-term history. In this thesis anomalies will be divided into anomalies within a time series, which could be either *contextual* or *collective*, and anomalies between the time series in the multivariate setting. Contextual anomalies refer to the short-term anomalies when values at specific time stamps suddenly change with respect to their temporally adjacent values. Collective anomalies refer to the long-term anomalies when entire time series or large subsequences within a time series have unusual shapes. When collective anomalies should be found it is useful to use a rolling window approach where a multivariate window of past time values in all dimensions are used for each point in time. Each window instead of each point is then classified as either an anomaly or not. When working with multivariate time series it is possible to detect anomalies in each univariate time series seperately. However, the cross-correlation between different time series could also be helpful in discovering anomalies. [Aggarwal, 2013]

Preprocessing the data and extracting useful features from raw time series will make the classification easier. The feature extraction phase is often considered just as important as the classification algorithm.

The characteristics of the anomalies are often unknown before they happen. They are most often uncommon and appear only in 1-5% of the data which means that a lot of annotated data is needed to be able to use supervised methods. Hence the approach to solving the problem is generally unsupervised where all the data that the model is trained on is seen as non-anomalous.

In this thesis the focus is on reconstructing the data and measuring how good the different models can reconstruct the data in a window of different sizes. Since the model is trained on non-anomlaous data, the anomalies are harder to reconstruct and will have a larger reconstruction error. The resulting error can either be used as a feature or a direct measure of how anomalous the sequence is.

1.2 Related work

Overview

Prediction-based

- VARIMA
- Neural Networks (CNN/RNN/LSTM/GRU)
- Using PCA to get univariate forecasting

Distance and density-based

- Dynamic Time Warping
- Mahalanobis distance
- Isolation Forest

Probabilistic

- Hidden Markov Model
- Dynamic Bayesian Networks

Linear Model

- Principal Component Analysis (PCA)

- Partial Least Squares (PLS)
- Matrix Factorization
- Support Vector Machines

Reconstruction based

- LSTM Encoder-Decoder (LSTM-ED)
- Auto Encoder (AE)
- Variational Auto Encoder (VAE)
- Generative Adversarial Networks (GAN)
- Deep Autoencoding Gaussian Mixture Model (DAGMM)
- Hierarchical Temporal Memory (HTM)

Transformation to other representations

- Leveraging Trajectory Representations of Time Series
- Numeric Multidimensional Transformations
- Discrete Sequence Transformations

1.3 Problem formulation

2

Theory

2.1 Temporal models

Time series analysis can be done in the frequency-domain or the time-domain. Classical time series analysis in the time-domain is based on Box-Jenkins work on regression modelling [Box et al., 1970]. In Box-Jenkins modelling an autoregressive integrated moving average-process (ARIMA-process) that models the temporal dependency is identified and fitted to data. This model can then be used for forecasting time series. However, ARIMA-processes are linear and can only handle data that can be made stationary. Autoregressive conditional heteroskedasticity (ARCH) models

Neural network

Occam's razor

black-box

Recurrent Neural Networks

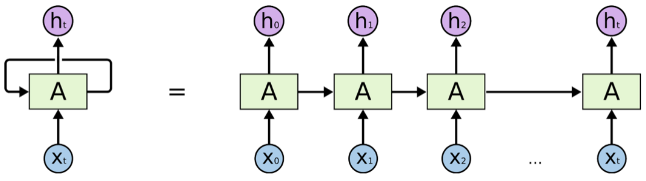

Figure 2.1 RNN cell. [*Understanding LSTM Networks – colah's blog* n.d.]

Recurrent Neural Networks (RNNs) are a group of neural network that, in contrary to other neural networks, have loops in them. This gives them a memory which makes them good at handling sequential data. In Figure 2.1 an unrolled RNN is shown, illustrating how the loops of the RNN can be interpreted as a sequence of conventional neural networks. Each module, A, takes an input, X_t, and the output from the previous module, h_{t-1}, and puts it through an activation function, tanh, (see Figure 2.2) which generates an output, h_t. The output is then passed to the next module and is given by

$$h_t = tanh(W \cdot [h_{t-1}, X_t]) \tag{2.1}$$

where W is the weight.

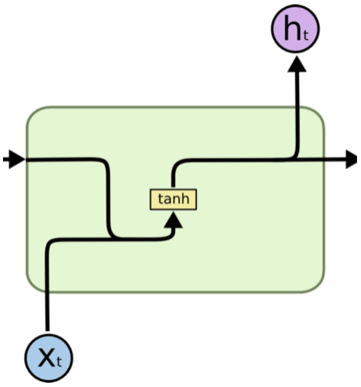

Figure 2.2 RNN cell. [*Understanding LSTM Networks – colah's blog* n.d.]

The weights in multilayer architectures, such as RNNs, can be trained by a back-propagation procedure [Rumelhart et al., 1986]. The idea is to compute the gradient of an objective function with respect to the weights of the network. By using the chain rule for derivatives and repeatedly working backwards in the network with respect to the output of a given module the gradient is propagated from the last module to the beginning of the network.

In practice, RNNs fail to contain long-term information due to vanishing or exploding gradients and are not robust to input noise [Bengio et al., 1994]. When using backpropagation in neural networks with more than one layer, and in the case of RNNs several modules, the chain rule includes increasingly many multiplications when the number of modules increase. Since the activation function is in the span [-1,1] the gradients could become very small over long distances.

Long Short-Term Memory The repeating module in the basic RNN cell can be substituted for Long Short-Term Memory (LSTM) modules that can contain long-term information without vanishing gradients [Hochreiter and Schmidhuber, 1997]. The LSTM module contains four interacting layers instead of a single tanh layer (see Figure 2.3). In each module a cell state is updated and in addition to the output, the cell state is sent into the next module. The cell state is updated with

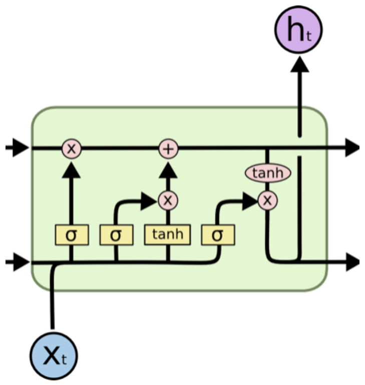

Figure 2.3 LSTM cell. [*Understanding LSTM Networks – colah's blog* n.d.]

2.2 Deep generative models

Generative Adversarial Networks

Probabilistic computations that arise in maximum likelihood estimation and similar methods are in many cases difficult to approximate. Generative Adversarial Network (GAN) is a framework that bypass some of these difficulties. A GAN consists of two major models; a generative model and a discriminative model. The generative model produces a sample and the discriminative model determines if the sample is from the distribution of the generative model or the distribution of the real data. These two models compete and the generative models improves with the goal of generating more realistic data and the discriminative model improves with the goal of being able to differentiate between real and generated data.

In the framework, both the generator and the discriminator are neural networks. Random noise is sampled from a latent space and forward propagated through the generator, the generator can then be trained with backpropagation and gradient descent to generate data that follows the wanted distribution. This procedure enables approximations to be made without using Markov chains and approximate inference.

The generative capacity of Generative Adversarial Networks (GANs) have previously been shown to be effective in image analysis for generating images that follow a certain distribution. Recent work has shown that it could also be used to effectively generate multivariate time series. In anomaly detection this generative ability could be used to reconstruct samples of time series by first mapping them back to the latent space. The samples that doesn't follow the learned distribution reconstructs with an error and can be classified as anomalies.

The GAN framework is making it possible to use a wide variety of cost functions that measure the distance between the real and the generated distribution. But they also have several disadvantages.

Wasserstein GAN

3

Methods

4

Results

4.1 Data

5

Discussion and conclusions

Bibliography

Aggarwal, C. C. (2013). *Outlier analysis (book)*, pp. 1–446. ISBN: 9781461463962. DOI: `10.1007/978-1-4614-6396-2`.

Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning Long-Term Dependencies with Gradient Descent is Difficult". *IEEE Transactions on Neural Networks* **5**:2, pp. 157–166. ISSN: 19410093. DOI: `10.1109/72.279181`. URL: `https://ieeexplore.ieee.org/document/279181/`.

Box, G. E. P., G. M. Jenkins, and G. C. Reinsel (1970). *Time series analysis : forecasting and control*. John Wiley, p. 746. ISBN: 9780470272848.

Hochreiter, S. and J. Schmidhuber (1997). "Long Short-Term Memory". *Neural Computation* **9**:8, pp. 1735–1780. ISSN: 08997667. DOI: `10.1162/neco.1997.9.8.1735`.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning representations by back-propagating errors". *Nature* **323**:6088, pp. 533–536. ISSN: 00280836. DOI: `10.1038/323533a0`. URL: `http://www.nature.com/articles/323533a0`.

Understanding LSTM Networks – colah's blog (n.d.). URL: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.