Gottfried Wilhelm Leibniz Universität Hannover
Institut für Verteilte Systeme
Distributed Computing & Security Group

Master thesis
Informatics (M.Sc.)

Anomaly detection in streaming data using autoencoders

Student:	B.Sc. Bin Li
First Supervisor:	Prof. Dr. Eirini Ntoutsi
Second Supervisor:	Prof. Dr. Wolfgang Nejdl
Date:	May 1, 2018

Declaration of Authorship

I hereby certify that this thesis has been composed by me and is based on my own work, unless stated otherwise. No other person's work has been used without due acknowledgement in this thesis. All references and verbatim extracts have been quoted, and all sources of information have been specifically acknowledged.

Hanover, May 1, 2018

B.Sc. Bin Li

Contents

List of Figures

List of Tables

Codeverzeichniss

Abstract

Data stream is a data format appears in plenty of big data research scenarios, for example, manufactural sensors, production line data etc. Here anomaly detection plays an important role for use cases like predictive maintenance, event detection, and could potentially avoid large amount of financial costs. However, different from traditional anomaly detection tasks, anomaly detection in streaming data is especially difficult while data comes along the time with latent changes, so that a single static model doesn't fit streaming data all the time. In this paper, we propose a novel autoencoder-based anomaly detection approach specially designed for streaming data. The model takes mini-batches of data from the stream as input, and try to reconstruct it using autoencoder, and the anomaly likelihood is informed by the reconstruction error. Experimental results suggests that our model can sufficiently detect anomaly from data stream and update model online to fit the latest data property.

Chapter 1

Introduction

Anomaly detection is an important problem in data mining, and widely used in the manufacturing industry, commercial world, internet company etc. It could avoid or reduce lose in many scenarios like machine health monitoring, credit card fraud detecting and spam email classification, and could also be used as a preprocessing step to remove anomalies for datasets. There are already plenty of anomaly detection techniques proposed in previous literatures, that solve this problem from variety perspectives, e.g. distance-based methods, clustering analysis, density-based methods etc.

There is no lack of anomaly detection approaches that perform good with respect to different kinds of data, however, majority of them are batch model, which means, all data should be available in advance. This becomes a shortcoming under today's big data background. With the rapid development of hardware in the last decade, the situation of data acquisition and analysis has significantly been changed. Specifically, the IoT application. Assume that we collect data from sensors attached to IoT devices, the data comes continuously and everlasting. During data analysis, we should always consider the volume and velocity of data, which means, on one hand, with traditional batch classifiers, the infinity data stream will lead to out of memory, on the other hand, streaming data usually comes with a high speed that leaving the system few processing time. In addition, the statistical property of data may also change over time, which is formally called 'concept drift'. The model should always learn new knowledge from the stream and update its definition of normal and anomalous automatically. To this end, an anomaly detection system for streaming data should be able to 1) be initialized with only a small subset, 2) process streaming data and make prediction in real-time, 3) adapt data evolution over time.

Malhotra et al. introduced an autoencoder based anomaly detection approaches in [1],[2], and achieved good performance in multiple time series dataset. However, in this approach, they assume that the whole datasets are available beforehand, and didn't considered the aforementioned online learning difficulties. Hence, we enhanced this kind of autoencoder based anomaly detection approaches with the online learning ability by inplementing incremental model updating strategies based on the streaming data.

In this paper, we introduce a novel and robust incremental autoencoder-based anomaly detection model, which designed specifically for time series data in a streaming fashion using Long Short-Term memory (LSTM) units, with also online learning ability for model updating. For each accumulated mini-batch of streaming data, the autoencoder reconstructs it with previous knowledge learned from normal data. Anomaly data (never used for training) is expected to cause significant larger reconstruction error than normal data. In addition, the model update itself online according to performance-based criterions.

Chapter 2

Related works

There are already pretty much researches on anomaly detection, some of them referred to streaming data and online learning. In this section, we list some widely used classical machine learning-based approaches, as well as some autoencoder based researches, and finally provious work on neural network incremental learning.

2.1 Classical machine learning based approaches

As an important component of data mining and machine learning, anomaly detection has been investigated using plenty efficient models. When talking about anomaly detection, the most intuitive solution may be detection of outliers from a dense cluster, or to find thoes data points that have obvious different property as their neighbors and so on. Within thoes large batch of classical methods, the Local Outlier Factor (LOF) and One-class Support Vector Machine (OCSVM) are two commonly used models in real-world use cases.

LOF model

In anomaly detection, the LOF is a common density-based approach. LOF shares some concepts with DBSCAN such as ‘core distance’ and ‘reachability distance’, in order to estimate local density. Here, points with substantially lower local density than their neighbors are considered as anomalies. LOF shows competitive performance in many anomaly detection tasks, especially when dealing with data with unevenly density distribution. However, when getting a numerical factor from LOF model, it is actually hard to define a threshold automatically for the judgement of anomaly.

OCSVM model

Another widely used model is the domain based One-class Support Vector Machine. As an unsupervised one-class classifier, OCSVM takes only normal data as input, and

generates a decision surface to separate them from the anomaly states. By analyzing anomalies, the datasets are always bias to the normal part, and anomaly appear only rarely. So, this kind of one-class classifiers avoid making balance between the two classes. Besides, they also take advantage of classical support vector machine, with the help of kernel method, they can also deal with linearly not separable data. However, in the mean time, the choosing a proper kernel becomes a hard point of OCSVM. A suboptimal kernel function can seriously impact the performance.
Although classical machine learning approaches can handle most of the normal anomaly detection, there a still a lack of pervasive models that fit different kinds of data characters. Moreover, only few of those approaches could be directly or after some modification used for time series and streaming data, while they ignore the temporal dependency between samples.

2.2 Autoencoder-based anomaly detection approaches

LSTMs-Autoencoders are originally widely used for text generation. Text data are usually embedded into vector as input of autoencoder. And the tasks are either generate temporal relevant text on the decoder side or learn text representation in the hidden layer [1]. As text data are relevant in the sense of words within a sentence or between sentences, it is similar to the streaming data temporal dependency problem.

Sutskever et al. [10] use a deep LSTMs-based sequence to sequence model for language translation. In their work, the deep LSTMs encoder take single sentence as input, and learn a hidden vector of a fixed dimensionality, and then a different LSTMs decoder decodes it to the target sentence. As a translation task, they found that this encoder-decoder architecture can capture long sentences and sensible phrases, especially they achieved better performance with deep LSTMs in compare with shallow LSTMs. In addition, a valuable found is, reversing the order of words in the input sentence makes the optimization problem much easier and achieved better performance. The LSTMs based model outperforms non-LSTMs model on the long input sentence cases (more than 35 words) since its long-term memory ability.

Li et al. [4] did similar research on long paragraph text and even entire document generation using LSTMs-autoencoders. Their main contribute is the hierarchical sentences representation. The model learns words level, sentence level and paragraph level information with each respectively a LSTMs layer, so that the model captures very long-term temporal information. Moreover, they introduced an attention based hierarchical sequence to sequence model that connect the most relative part between encoder and decoder like the works around a final punctuation. They experiment with documents over 100 words, the results show that hierarchical and attention-based hierarchical LSTMs learns even better long-term temporal information than standard LSTMs-encoder-decoder models.

As autoencoders achieves great successes in text data and speech processing, they are also used on time series anomaly detection in terms of temporal dependently data. These models train autoencoders with only normal data, and anomaly data as unknown patterns. Then the autoencoder can only reconstruct normal patterns, large reconstruction error indicates anomaly. An early work [8] uses the vanilla autoencoder to detect abnormal status of the electric power system. In order to capture temporal information, they applied sliding window on the raw data as input. As anomaly scoring method, they evaluated each sliding window with respect to their reconstruction error. As some measures in the autoencoder output vectors are more sensible to anomalies than others, they use the average absolute deviation of reconstruction error as anomaly score. And the anomaly threshold is chosen by large amount of experiments over normal data.

An important reason of using autoencoder for anomaly detection is its ability of dealing with high-dimensional. Sakurada et al. [9] experimented with time series data that consist of 10-100 variables with no linear correlation. Comparing with reconstruction using PCA or Kernel PCA techniques, using the autoencoder reconstruction error is more easily to recognize anomalies.

In further researches, Malhotra et al. [7][5] develop the application of LSTMs-autoencoder in sequence learning into anomaly detection problem. They proposed stacked LSTM networks model to learn high level temporal patterns. They show that LSTMs outperforms normal RNNs based anomaly detection model and avoid facing to the gradient vanishing problem. They also detect anomaly based on the reconstruction error. The scoring function is based on the parameters of a estimated normal distribution of a validation set. Their experiments show that the model performs good in variety kinds of datasets. A variation of this model [6] has been proved that achieves better performance in the anomaly detection tasks, while they tell that using a constant as input of decoder instead of read time series value improves the performance of model.

2.3 Online incremental learning with autoencoders

Zhou et al. [11] proposed an online incremental updating method for denoising autoencoders by modifying the hidden layer neurons in order to deal with the non-stationary streaming data properties. The kern ideal are two steps, adding hidden layer neurons to capture new knowledge, and merging hidden layer neurons if information is redundant. Their experimental result shows comparable or better reconstruction result than non-incremental approaches with only few data used during initialization. And they show that their incremental feature learning methods performs more adaptively and robustly to highly non-stationary input distribution.

Dong et al. [2] proposed a 2-step anomaly detection mechanism with incremental autoencoders. The implemented the system with ensembled autoencoders in multithreads

to leverage parallel computing when large volumes of data arrive. Besides their 2-step mechanism check anomaly in the first step and verify anomaly data with previous and subsequent data (to differ between anomalous state and concept drift) to reduce false-positive rate in anomaly detection. In the experimental results, they show that their model outperforms commonly used tree-based anomaly detection model especially when concept drift presents and speed up the online processing speed with mini-batch learning and online learning in multithreads.

In this work, we implement a LSTMs-autoencoder based incremental streaming data anomaly detection model. The LSTMs-autoencoder is close to the model by Malhotra et al. [5], and we design an online model updating strategy as well as the dataset used for model updating.

Chapter 3

Preliminaries

3.1 Definition of stream

Assume that there are some devices or data warehouse that provide data continuously with a specific velocity V (here we only take about numerical data). An input window is defined as

$$X^{(i)} = \{x_1^{(i)}, x_2^{(i)}, ..., x_T^{(i)}\} \tag{3.1}$$

where $X^{(i)}$ is the i^{th} input window with length T, and $x_t^{(i)}$ is the t^{th} instance in the window, which is a multi-dimensional vector.

In each single time, a mini-batch B consists of multipe windows is used for model training or testing, and we assume that the data stream is infinity, so that mini-batches can always be accumulated from data stream. The equation 3.2 is a mini-batch start from window $X^{(i)}$ with batch size MB. And equation 3.3 describes the data stream.

$$B^{(i)} = \{X^{(i)}, x^{(i+1)}, ..., x^{(i+MB-1)}\} \tag{3.2}$$

$$S = \{..., B^{(i)}, B^{(i+MB)}, B^{(i+2MB)}, ...\} \tag{3.3}$$

3.2 Defination of anomlaies

Pointwise
A data point (instance) is anomalous if this point is distant from other observations according to some specific measurement metrics. This is used in fine-grained anomaly detection tasks, that need to find out every single anomalous instance, e.g. credit card fraud detection, spam email detection.

Window-based
A window is anomalous if the window contains one or more anomalous data points. For most of the window-based anomaly detection algorithm, they only calculate the anomaly

score of a given window, it's hard and sometimes not necessary to find out which data points in this window are those anomalies.

3.3 LSTMs

Recurrent neural networks(RNNs) are widely used for speech, video recognition and prediction due to its recurrent property that captures the temporal dependency between data in compare with other feed forward networks. However, the volume of RNN's memory is limited, and vanishing gradient is also a difficulty by training RNNs. Therefore, the long short-term memory networks (LSTMs) are a kind of reinforced RNN that is able to remember meaningful information in arbitrary time interval. A LSTM network is a recurrent neural network with neurons being LSTM units. Figure 3.1 shows a classical structure of a LSTM unit. LSTMs are able to capture long-term memory while there are a forget gate and a update gate in the LSTM unit, that select necessary previous information and new coming information according to the input data at each time step. The information is transferred to the next step within the cell state. Besides, each LSTM units also output its value by going through a softmax function.

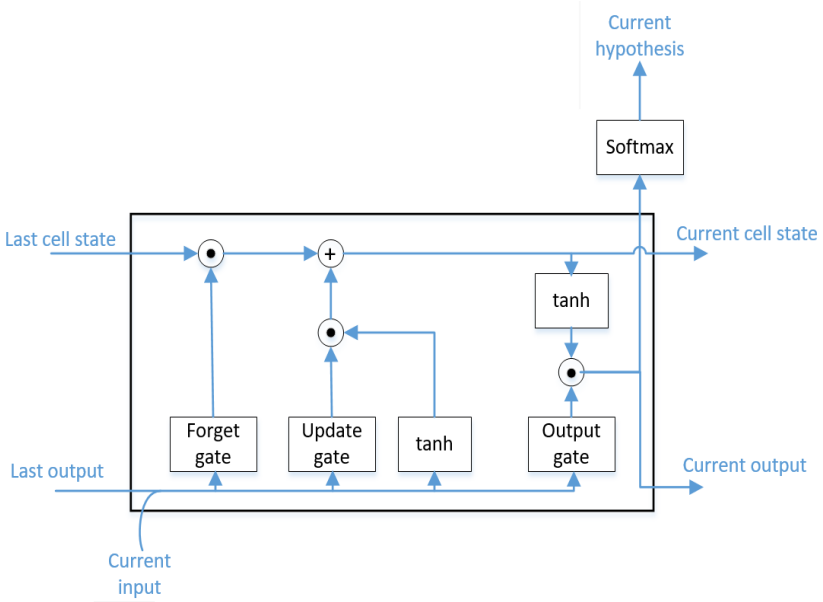


Figure 3.1: The LSTM unit

A LSTM unit can be unfolded over time, as shown in Figure 3.2 on the next page. The LSTM unit takes a data window as input, namyly takes one instance at a time. Therefore, the LSTM unit extracts useful and drop useless temporal information from the window of data.

Deep LSTM RNNs are built by stacking multiple LSTM layers. Note that LSTM RNNs are already deep architectures in the sense that they can be considered as a feed-forward neural network unrolled in time where each layer shares the same model parameters. It has been argued that deep layers in RNNs allow the network to learn at different time

scales over the input[3]. Figure 3.3 is a example of stacked deep LSTM neural network, there are 3 LSTM layers, each can be unfolded into 5 time steps, so the LSTMs take a window in length 5 as input and the output is in same size.

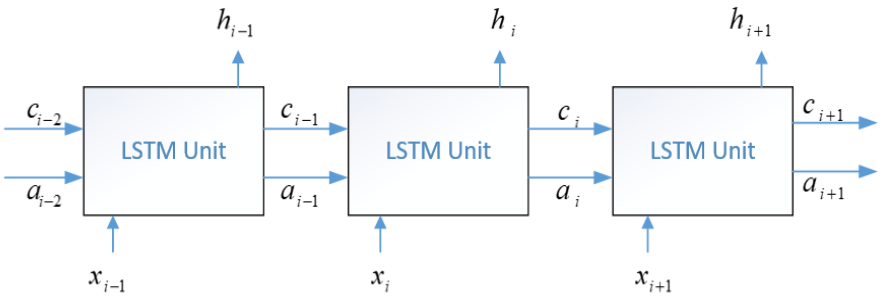
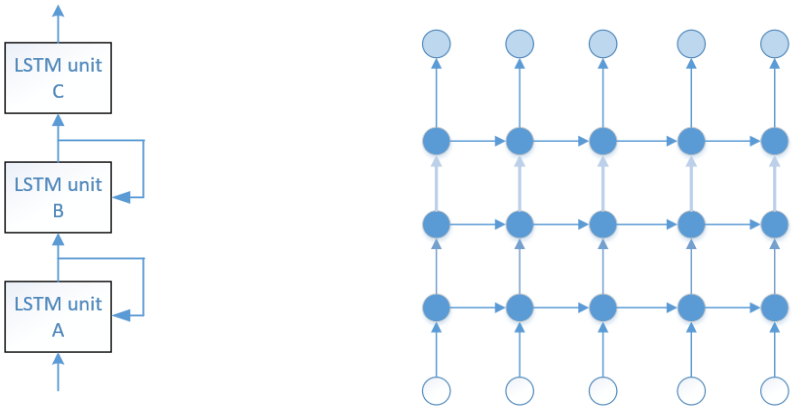


Figure 3.2: Unfolded LSTM unit



(a) Deep folded LSTMs

(b) Deep unfolded LSTMs. Each horizontal dark dot chain is an unfoldered LSTM unit over time, hollow dots and grey dots are windows of inputs and outputs.

Figure 3.3: Deep LSTMs

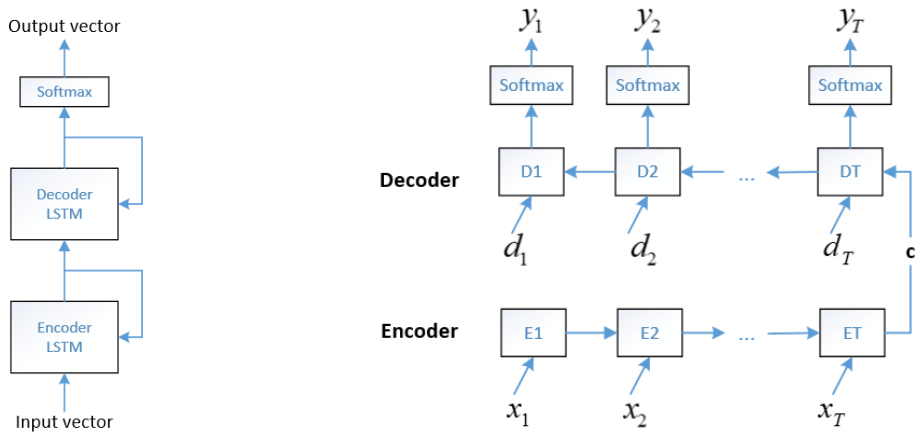
3.4 Autoencoders

An autoencoder is an artificial neural network with symmetrical structure. Normally an autoencoder has at least one hidden layer that consists of less neurons than input and output layers. And the basic aim of autoencoders is to reconstruct its own input and learn a lower dimensional representation (encoding) of input data in the hidden layer. Moreover, the autoencoders are also used for anomaly detection by measuring the reconstruction error between inputs and predictions. Normally the component between input layer and hidden layer is called encoder, and the symmetrical component between hidden layer and output layer is called decoder. For input χ, the objective function is to find weight vectors

for encoder and decoder to minimize the reconstruction error (3.4).

$$\begin{aligned} \Phi &: \chi \rightarrow H \\ \Psi &: H \rightarrow \chi \\ \phi, \Psi &= argmin \, \|\chi - (\Psi \circ \Phi)\chi\| \end{aligned} \tag{3.4}$$

LSTMs-autoencoder has the same encoder-decoder architecture, while the neurons are LSTM units and connected in the way described in section 3.3. Figure 3.4 is a basic LSTMs-based autoencoder architecture with single LSTM layer on both encoder and decoder side. Our incremental LSTMs-autoencoder is based on this structure. The model takes window with length T as input (one instance in each step). The cell state carries sequence information and is passed through LSTM unit over time. When the encoder reaches the last encoder state, namely ET in Figure 3.4b, the cell state is actually the fix length embedding of the input window, and copied to the decoder as initial cell state of decoder, so that the input information is also transferred to the decoder. And the decoder predict the window in reversed order in order to make the optimization problem easier. To be notice is, different from aforementioned deep LSTMs in section 3.3, the encoder outputs at each time step are not directly used as inputs of decoder, while between the encoder and decoder is actually not the same logical connection as stacked LSTMs. Here, the outputs of encoder are ignored, and there are different works contributes to the research of decoder inputs. Cho et al. [1] feeds the input sequence to the decoder for a learning phrase representation task, Malhotra et al. [5] feed to decoder LSTM unit at each time step the prediction of last time step as input, and in a extended work [6] they feed the decoder always a constant vector for an anomaly detection task, because the finial cell state already carries all relevant information to represent the input window. In our model, we feed the decoder a constant vector.



(a) Folded LSTMs-Autoencoder (b) Unfolded LSTMs-Autoencoder

Figure 3.4: LSTMs-Autoencoder

3.5 Streaming data generator: Apache Kafka

We utilize Apache Kafka as the streaming platform. Kafka is a widely used Publish/Subscribe architecture streaming system. It different from classical message queue technique with its fault tolerant, durable and large capacity properties. In the experimental setting, our data source is static databases, Kafka generate real-time data stream pipeline as data source publish records to the specific topic (topic is the data category mechanisms used in Kafka), and furthermore the stream of records will be consumed by different consumers like our analysis model, visualization model etc. This configuration can be easily scaled up to more complicated and demanding real world use cases. Each record in the Kafka stream pipeline is in the form of [Key, Value, Timestamp], where keys are used for positioning and values carry the data record.

Chapter 4

Proposed model

4.1 Framework overview

The proposed model is a full flow from data stream generation, anomaly detection with autoencoder-based model and online model incremental updating. Apache Kafka is used as the stream generator as shown in Figure 4.1. The first received batches of streaming data are used for decision of model hyperparameters and the model initialization. Hyperparameters includes the hidden layer size, batch size, input window length as well as the number of epochs. Once the hyperparameters are learned, an autoencoder will be constructed and initialized with random weights. A subset of the streaming data is used for initial model training (only normal data used for training). Furthermore, the model is used for online anomaly detection, and will be retrained when the retraining condition is triggered. As aforementioned, topic is the data category mechanisms in Kafka. The streaming data are published to a topic, and the prediction results are send back to another Kafka topic for visualization.

The Consumer 2 in Figure 4.1 is actually the core component of the LSTMs-autoencoder model. Once the initialized model is available, the online phase is then start. As shown in Algorithm 1, if a batch of streaming data is available, the model will start do prediction, evaluation, and check whether current batch is useful to store for later retraining.

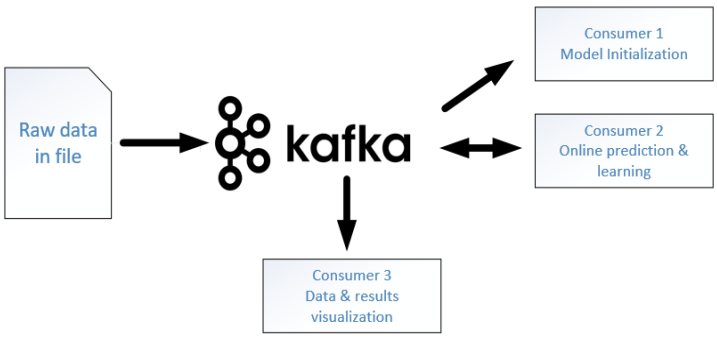


Figure 4.1: Data stream pipeline

Algorithm 1: Pipeline

```
input: performanceThreshold, retrainDataSize
needRetraining = False;
retrainBuffer = [ ];
while Batch data available do
    batch, label = getBatchData();
    if len(retrainBuffer) == retrainDataSize then
        retrain(retrainBuffer);
    else
        pred = predict(batch);
        result = evaluation(pred,label);
        if result >= performanceThreshold then
            continue;
        else
            if label == "normal" then
                retrainBuffer.append([B, label]);
            else
                continue;
            end
        end
    end
end
```

4.2 LSTMs-Autoencoder initialization

4.2.1 Encoder-decoder architecture

The LSTMs-Autoencoder is consist of two LSTM units, one as encoder and the other one as decoder. The encoder inputs are fix length vectors with shape <MB, T, D>, where MB is the number of data windows contained in a mini-batch, T is the numbers of data points within each data window, and D represents the number of data dimensionality. Here, MB and T are learned as hyperparameter in the initialization phase. And on the decoder side, it will output exactly the same format data vector for each mini-batch. The LSTM unit copies its cell state for itself as one of the cell input at next timestamp. At the last timestamp of encoder, the cell state of LSTM unit is the hidden representation of the input data vector and copied to the decoder unit as initial cell state, so the hidden information can be passed to the decoder. The size of hidden layer representation vector, namely the size of cell state is another hyperparameter need to be learn in the initialization phase. The larger the hidden vector, the more information can be captured during the process, so it is a feature highly depends on the data. Similar to previous study [10], we also train the encoder and decoder with time series in reverse order. For example, if the input data fragment are data points from timestamp t1 to t2, then the decoder will predict data point at t2 at first, and then back to t1 step by step, while this trick makes the gradient escarpment between last state of encoder and first state of decoder smaller and easier to learn.

In order to let the whole process happen online, the model initialization also utilizes streaming data. Once a small subset of streaming data is available, hyperparameters are learned, and then another dataset that consists only of normal data is collected from stream used for training.

4.2.2 Anomaly detection mechanism

(Todo: reconstruction error, anomaly scoring, initialized scoring parameters used for online phase)

4.3 Online learning

However, if we consider using the model for streaming data, the autoencoder might get outdated because of the relative small and simple initialization dataset and concept drift happed along with time. So the update of model is necessary. The main contribution of this paper is the incremental learning setting of the autoencoder model.

4.3.1 Retraining trigger

4.3.2 Retraining dataset

Chapter 5

Experimental setup

5.1 Datasets

5.2 Parameter tuning

Chapter 6

Experimental results

6.1 Grid search

6.2 Anomaly detection performance

6.3 Retraining

6.3.1 Reaction of concept drift

6.3.2 Comparison: with and without retraining

Chapter 7

Conclusion

Bibliography

[1] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. 2014.

[2] Yue Dong and Nathalie Japkowicz. Threaded ensembles of autoencoders for stream learning. 2017.

[3] Michiel Hermans and Benjamin Schrauwen. Training and analyzing deep recurrent neural networks. 2013.

[4] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. 2015.

[5] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. 2016.

[6] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Timenet: Pre-trained deep recurrent neural network for time series classi

cation. 2017.

[7] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. 2015.

[8] Marco Martinelli, Enrico Tronci, Giovanni Dipoppa, and Claudio Balducelli2. Electric power system anomaly detection using neural networks. 2004.

[9] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. 2014.

[10] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. 2014.

[11] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoder. 2012.