KDD99Cup dataset summery

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10% KDD | KDD | Stmp+http |
| Size | 73M | 725M | 101M |
| Training (normal) | 6.8M |  |  |
| Validation (normal) 1 & 2 | 2M & 2M |  |  |
| Validation (anomaly) | 14M |  |  |
| Test (normal) | 2M |  |  |
| Test (anomaly) | 42M |  |  |

Related works

* **LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection (Autoencoder, Anomaly detection,batch)[1]**

The EncDecAD (short for Encoder-Decoder scheme for Anomaly Detection) is Long Short Term Memory Networks(LSTMs) based anomaly detection architecture for time series. The utilization of LSTMs enabled the model to avoid missing the temporal information in the time series. The EncDecAD model consists of two parts, namely an encoder and a decoder, which have identical LSTMs based structure. The encoder takes a mini-batch of time series as input and outputs a hidden vector, and then follows the decoder, taking the hidden vector as input and trying to reconstruct the original input mini-batch as output. The reconstruction error is the main criterion of anomaly likelihood, while the model is only trained with normal instances, so for unacquainted anomalous data, the reconstruction error is considered to be relative higher.

According to their experimental results, the EncDecAD model could separate normal and anomalous points with a large margin. It works on different kinds of datasets, even non-periodic unpredictable data. However, The model is designed only for batch data, and need to train on the whole dataset. Time series data always comes continuously along with time, and sometime also with concept drift. If we consider the velocity and volume feature of time series data, an online incremental model is then necessary. Also, in the online fashion, it is challenging to learn from a massive stream of data the optimal number of features.

* **Online Incremental Feature Learning with Denoising Autoencoders**

**(Autoencoders over streams)**

The authors introduced an incremental feature learning algorithm to determine the optimal model complexity for streaming data based on the denoising autoencoder. The main idea is feature adding and merging. Specifically, it adds new features to minimize the objective function’s residual, and if features are redundant, then merge them to prevent overfitting and obtain a more compact feature representation. The result shows that it’s a good way to learn features from a large dataset by starting with a small set of initial features and automatically adjusting the number of features. And this method leads comparable or lower reconstruction and classification error than the stationary fashion.

* **Threaded ensembles of autoencoders for stream learning**

**(outlier detection over streams, streaming autoencoders, concept drift)**

This paper proposes a multi-threaded neural network model to deal with streaming data. On each thread running a autoencoder model and two buffers connecting data stream and model, in order to avoid model being idle or data stream being delayed. The thread ensemble enables the model with continuous learning capacity. They use a reconstruction error based criterion for anomaly detection, and maintain a buffer containing only normal data for decision of anomaly threshold. In order to distinguish between anomaly and concept drift, they check the points before and after a specific data point, while continuous founded anomalies indicate the possibility to be concept drift.

However, they didn’t talk about how the arrival rate of data stream and the fluctuations in the volume of data would impact the detection of anomalies. In addition, how emergent of the concept drift happens could also be a influence factor.

**(new/emergent class detection over streams)**

Structure

Abstract

1. Introduction
   1. Background, use case examples, importants
   2. Existing anomaly detection methods
   3. Challenges of current data\Time dependency
   4. Lack of model for time series and streaming data anomaly detection
   5. Autoencoder based data streaming AD model
2. Related works
3. Outlier detection on static data
   1. Distance based
   2. Density based
   3. Neighborhood based
   4. Waveform based
   5. One-class SVM
   6. HMM based

But those models do not consider the temporal dependency

1. Outlier detection for streams
2. Deep Learning approaches: unsupervised Autoencoder based models

(solved the multi-dimensional time series anomaly detection problem)

1. EncDecAD
2. TimeNet

But those model always needs have all data in advance, could not deal with data changes (they work with stationary data).

1. Neural network based online learning architecture
   1. Add & merge
   2. Threaded ensembles of autoencoders for streaming learning

But not exactly take time series as input (didn't apply any window, namely temporal combination)

1. From the super-/unsuper-/semisupervised learning perspective
2. Basic concepts
   1. Definition of a stream
   2. Method of processing the stream
   3. Definition of an outlier (point & window)
   4. Autoencoders | LSTMs
3. An autoencoder-based framework for unsupervised anomaly detection
   1. Overview / architecture of the framework
   2. Autoencoder component initialization
      1. EncdecAD based architecture (input, output, hidden layer)
      2. LSTM Input format
      3. Reconstruction error, anomaly score, parameters

Autoencoder might get outdated--> how to update the autoencoder

* 1. Online learning for batch-based outliers
     1. Updating stratergy

* + - Model
      1. Start from scratch -- if reconstruction error continuously being high
      2. Continue training with last-seen data -- if reconstruction error shortly high
    - Parameters(mu, sigma, threshold)
      1. Update, if prediction performance bad (e.g. F-beta low, miss alarm etc.)
      2. To avoid overfitting, previous parameter still as a part of the new parameter
    1. Maintenance of dataset for retraining
* Keep storing N batches streaming data in the buffer
* Store summarization of all seen batches
* Label data that the model mistakenly predicted, to pay more attention on them by retraining

1. Experimental setup
2. Datasets description
   1. Amount of anomaly
   2. Normal, anomaly proportion
   3. Periodicity
   4. Dimensionality
3. Initialization with first n batches streaming data
   1. Wait until accumulated enough data for initializing the model
   2. Split into normal sets
4. Sn: Training normal set
5. Vn1: Validation normal set1 for early stopping
6. Vn2: Validation normal set2 for parameter learning
7. Tn: for testing of training

And anomaly sets

1. Va: Validation anomaly set for parameter learning
2. Ta: for testing of training
   1. Dropout rate of autoencoder
   2. Save to disk
3. Streaming data generation
   1. Apache Kafka
   2. Kafka setup and configuration
   3. Deal with latency problem
4. Evaluation metric
   1. #False alarm, # miss alarm
   2. F-beta for performance
   3. Reconstruction error of normal data for model fitness of data
   4. \*Area under the curve based on anomaly score

1. Experiment results
2. Hyperparameter grid search
3. Generally performance
4. When updating is triggered
5. Reaction of concept drift, non-significant anomalies
6. Comparasion of performance before/after updatng, with/without updating
7. Runtime comparasion
8. Conclusion
   1. How online learning helps the model to adjust the stream trend
   2. Is the general performance comparable to troditional batch approaches
   3. Possiable reasons of suboptimal performance during experiment
   4. Future works

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 the model doesn’t fit the data all the time.

In this paper, we introduce a novel autoencoder based anomaly detection methods 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 from the reconstruction error. Experimental results shows that our model can sufficiently detect anomaly from data stream and update model online to fit the latest data.

**Key words**

LSTMs, autoencoders, anomaly detection, online learning

Introduction­

(Problem introduction, problem importance)

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 and outlier detection techniques proposed in literature, that solve this problem from variety perspectives, e.g. distance-based methods, clustering analysis, density-based methods etc.

(why problem hard, short coming of previous works)

There is no lack of approaches that perform really good for anomaly detection, however, most of them are focusing on batch data, 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 also 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-based classifier, 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.

(short coming of approaches similar to this paper, problems need to be solved, lead to inspiration & good ideas)

Malhotra et al. introduced similar autoencoder based anomaly detection approaches in [1],[2], and achieved good performance in multiple different 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 using incremental knowledge learning and model updating strategies based on the streaming data.

(high level ideas and concepts of approach in this paper)

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 as neurons, with also online learning ability for model updating. The model need only one pass of the streaming data. For each accumulated mini-batch of streaming data, the autoencoder try to reconstruct 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 criterions based on performance.

(summary of tested examples, summary of results)

(Todo: about label: semi-supervised, label for scoring, one-class for training)(Todo: we experimented with datasets: \_, \_ , \_. The model shows robustness )

Related works