Thesis structure

Wednesday, March 21, 2018 4:18 PM

Introduction

1. Back ground, use case examples, importants
2. Existing anomaly detection methods
3. Lack of model for time series and streaming data anomaly detection
4. Autoencoder based data streaming AD model

Related works

1. Common batch models
    a. Distance based
    b. Density based
    c. Neighborhood based
    d. Waveform based
    e. One-class SVM
    f. HMM based

   But those models do not consider the temporal dependency

2. Autoencoder based models
    a. EncDecAD
    b. TimeNet

   But those model always needs have all data in advance, could not deal with data changes

3. Neural network based online learning architecture
    a. Add & merge
    b. Threaded ensembles of autoencoders for streaming learning

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

Model

1. Model architecture
    a. EncdecAD based architecture (input, output, hidden layer)
    b. LSTM Input format
    c. Reconstruction error, anomaly score, parameters

2. Online learning
    a. Updating straregy
        i. Model
            1) Start from scratch -- if reconstruction error continuously being high
            2) Continue training with last-seen data -- if reconstruction error shortly high
            - Todo: Add & merge LSTM neurons

    ii. 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

b. Maintainance of dataset retraining
    i. Keep storing N batches streaming data in the buffer
    ii. Store summarization of all seen batches
    iii. Label data that the model mistakenly predicted, to pay more attention on them by retraining

Experimental setup

1. Datasets description
    a. Amount of anomaly
    b. Normal, anomaly proportion
    c. Periodicity
    d. dimensionality
2. Initialization with first n batches streaming data
    a. Wait until accumulated enough data for initializing the model
    b. Split into normal sets
        i. Sn: Training normal set
        ii. Vn1: Validation normal set1 for early stopping
        iii. Vn2: Validation normal set2 for parameter learning
        iv. Tn: for testing of training

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

Experiment results

1. Hyperparameter grid search
2. Generally performance
3. When updating is triggered
4. Reaction of concept drift, non-significant anomalies
5. Comparasion of performance before/after updatng, with/without updating
6. Runtime comparasion

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