1) Background.

1) 2,000 time-series: each day is one time slot?
2) What's the type of logs? SQL or plain text? Is data well formatted? How is data formatted?

Kaizen is a Data Collection Platform developed by CopperTree. It collects the data from the Building Automation System (BAS) to MongoDB. The data come from different sensors and controllers which include**, temperature, humidity, pressure, voltage, flow, power** and so on.

There are two types of time series trend logs, one is **Polling** and the other is **COV**. The first one means they have a fixed sampling period, like 5mins, 10 mins, and 15mins. Cov is 'Change Of Value', which means the Kaizen only sample the data when the value is changed. Usually, the set point trend log will be saved as COV.

Each trend log is saved in MongoDB as JSON format with 2 files. Metadata and their time series data. Metadata contains info for each trend log like sample interval, units, id and so on. I used **pymongo** library to extract the data from MongoDB. Then, using JSON and Pandas library to transform the time series data to Pandas Dataframe for future analysis.

1. Long Time Series Trend Clustering

2.

In the beginning, the idea is quite simple. I tried to cluster the time series data for future exploration and anomaly detection. The idea is like suppose some trend logs are in the same cluster for a long time if the next time period one trend log's change trend is away from the other trend log. This trend log would be regarded as an anomaly trend log.

Time-series data clustering usually has four components: dimensionality reduction methods, distance measures, clustering algorithms and clustering evaluation measures.

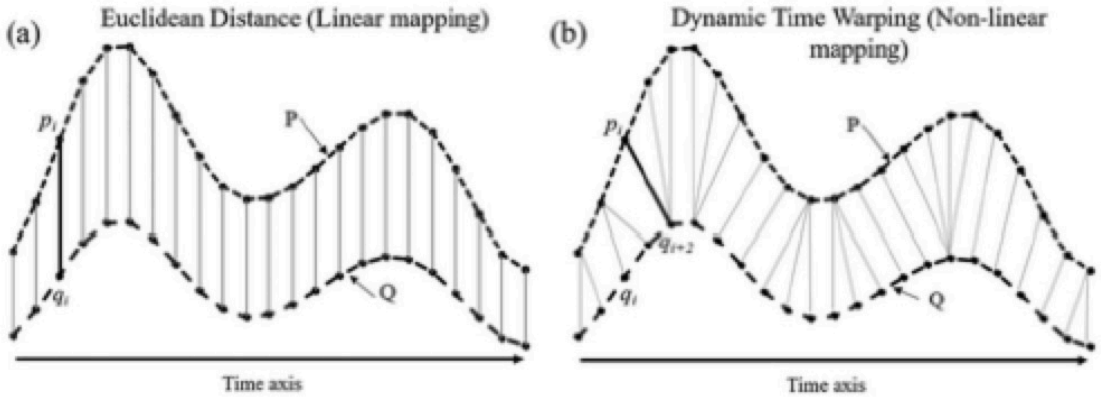
1）Dimensionality reduction

Dimensionality reduction is a trade-off between speed and quality. I tried PCA, T-SNE and combination of PCA and T-SNE for dimensionality reduction. However, the results of the test are not ideal. Therefore, I didn't use dimensionality reduction for time-series clustering.

2）Distance measures

Distance measures is the most important part since clustering algorithms are based on a notion of distance among vectors or points. In this part, Euclidean Distance, Normalized Euclidean Distance, Euclidean Distance through Dynamic Time Warping and Normalized Euclidean Distance through Dynamic Time Warping were tested.

Euclidean Distance through Dynamic Time Warping is using dynamic programming to find an optimal Euclidean Distance match between two given sequences.



$$D(i,j) = Dist(i,j) + \min[D(i-1,j), D(i,j-1), D(i-1,j-1)]$$

The time complexity for Dynamic Time Warping **is O(N*M),** where N and M are the lengths of the two input sequences. I implemented the Fast DTW algorithm to get the distance between two trend logs. **The key point is to reduce the search space by only searching the nearest K points.**

3）Clustering algorithm

DBSCAN(Density-based spatial clustering of applications with noise), K-means and K-medoids had been tested for clustering. There is no need to set **the number of clusters** in advance and the most important parameters in DBSCAN are **eps** and **min_samples**. I tested the by grid search to find the best **eps** and **min_samples**. However, the results are unsatisfactory since most of the trend logs are in the same cluster.

K-Means and **K-Medoids** are quite similar. The algorithm works iteratively to assign each data point to one of the k clusters. The most difference between **K-Means** and **K-Medoids** is that **K-Medoids** chooses data points as centers. Therefore, I can build a distance matrix in advance when using **K-Medoids.** By precomputing distance matrix, K-Medoids is much faster than K-Means since I only need to build distance matrix once. The most important parameter for these

two algorithms is **K** and I choose **K** by **elbow method**. One should choose a number of clusters so that adding another cluster doesn't give much better SSE.

4）Clustering Evaluation Measures
Silhouette Coefficient and Sum of Squares Error are used for evaluating clustering results. The **Silhouette Coefficient** is a measure of how similar an object or vector is to its own cluster and poorly matched to neighboring clusters, which ranges from -1 to +1. The higher is better.

Sum of Squares Error is the sum of the squares of residuals. A small SSE indicates a tight fit of the model to the data.

Therefore I choose **K** based on the **Silhouette Coefficient** and **Sum of Squares Error.**

3) What does the interactive visualization demo look like? Brief introduction to the functions, implementation techniques.

The interactive visualization demo is built to assist find the potential anomaly trend logs. It is a prototype developed on Jupyter Notebook based on the Plotly library, which plans to be a new feature to the next version of Kaizen.

It is based on clustering results. The user specifies a trend log, its time range(like one month) and the similar number, then clicking the show curves button.

For example, a user chose a Room Temperature trend log with time range on 2019 March and a similar number with 3. The interactive visualization demo will show three figures below. The first one is a distance matrix with the three most similar trend logs on March 2019, last year March 2018 and recent time range January and February 2019. This three most similar trend log is based on clustering result for last year March and this year January and February.

The second figure shows the selected trend log and its similar trend logs from January to March.

The last figure shows the selected trend log and its similar trend logs last one year on last year March.

Since we saved the cluster results for each month on the database, this demo can quickly show the three figures to help engineer determine whether this trend log is normal or anomaly.

3. Multi-sensor Anomaly Detection.(Maybe of high interest)

Firstly, let me explain the background. For a single room, Building Automation System collected many trend logs (we called variable air volume system) like room temperature, room humidity, supply air temperature, Heating Coil Valve Output, Cooling Coil Valve output, Damper output and theirs set point trend logs. Usually, we detect anomaly behavior according to a set of rules.

For example, Determine if a damper output is changing direction (Default 1 cycle = 5% change) too many times in one day.

This kind of rules is set by experienced engineers. However, the rules may not be able to detect all the anomalies. We know that all the trend logs for a single room interact with each other. For example, the room temperature will increase if we increase the Heating Coil Valve output and keep Cooling Coil Valve output.

People cannot predict all the interaction among these trend logs. I tried to build an anomaly detection module based on LSTM.

1) More detailed introduction.

Background

An anomaly detector is implemented based on LSTM (Long Short Term Memory Networks), which consists of a two-stage strategy.

The first one is time-series prediction or reconstruction. And the second one is anomaly score calculation.

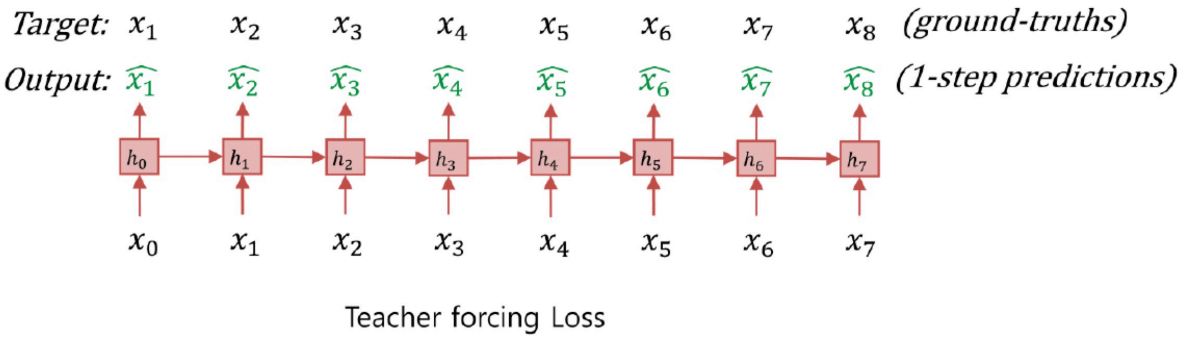
We first train this model with a trainset which contains no anomalies, then we use the trained model to detect anomalies in a test set, where anomalies are included.

This idea is that when the model can predict or reconstruct the normal data well on train dataset well, it can also predict or reconstruct the normal data on test data set well. However, it cannot predict or reconstruct the anomaly sequence well on test data set.

1) Loss Function

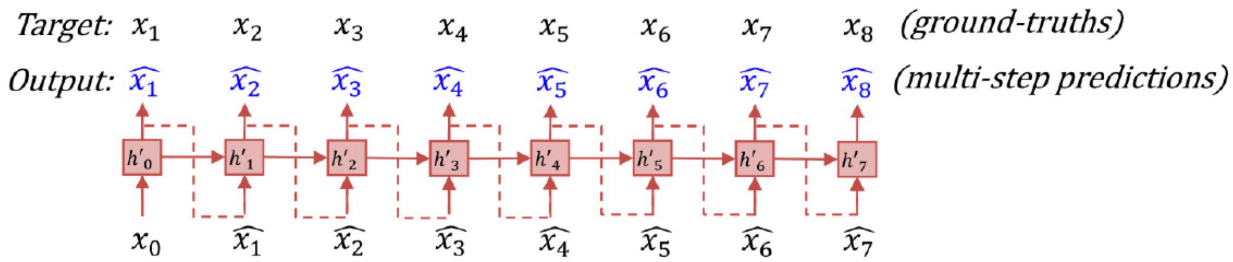
The loss function is a central part of any machine learning model. The calculation of the LSTM model consists of three parts, teacher forcing loss, free running loss and simplified professor forcing loss.

Teacher forcing loss is calculated based on 1-step predictions.



Teacher forcing Loss

Free Running loss is calculated based on multi-step predictions.



Free Running Loss

Simplified Professor Forcing Loss is calculated based on the difference between the teacher forcing and free running hidden states.

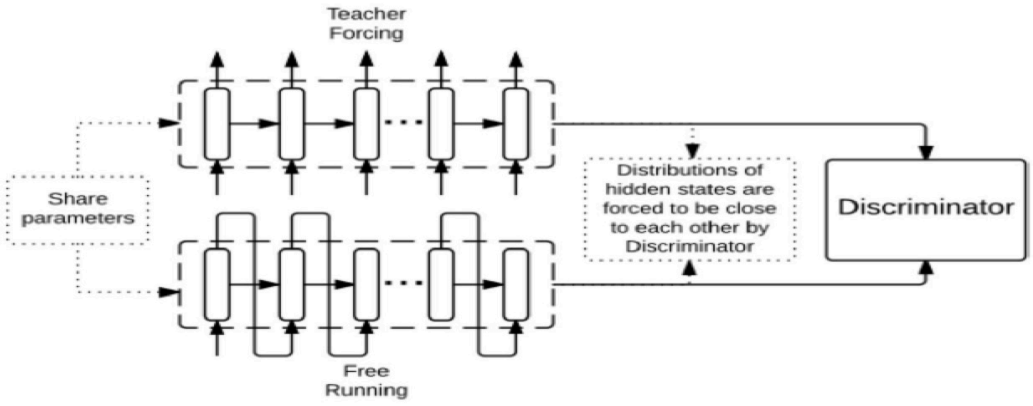
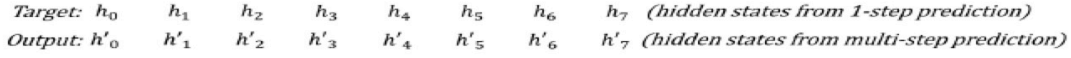


Figure 4: Simplified professor forcing loss structure.



Simplified Professor Forcing Loss

2) How to define anomaly?
In Kaizen, facility department engineer set several hourly, daily or weekly rule insights. I label the trend logs as anomaly when the rule insights generated.

For example: Determine if a damper output is changing direction (Default 1 cycle = 5%
change) too many times in one day.

3) Prediction and Reconstruction
May I draw some pictures to illustrate this?

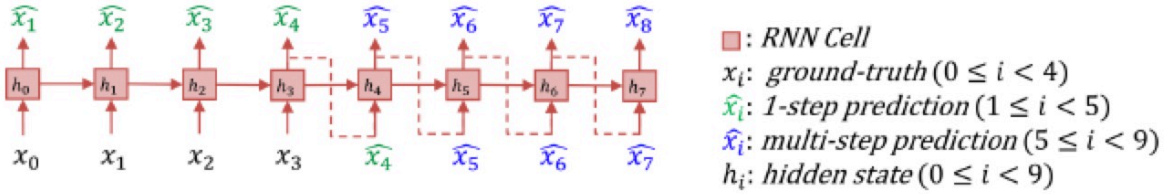
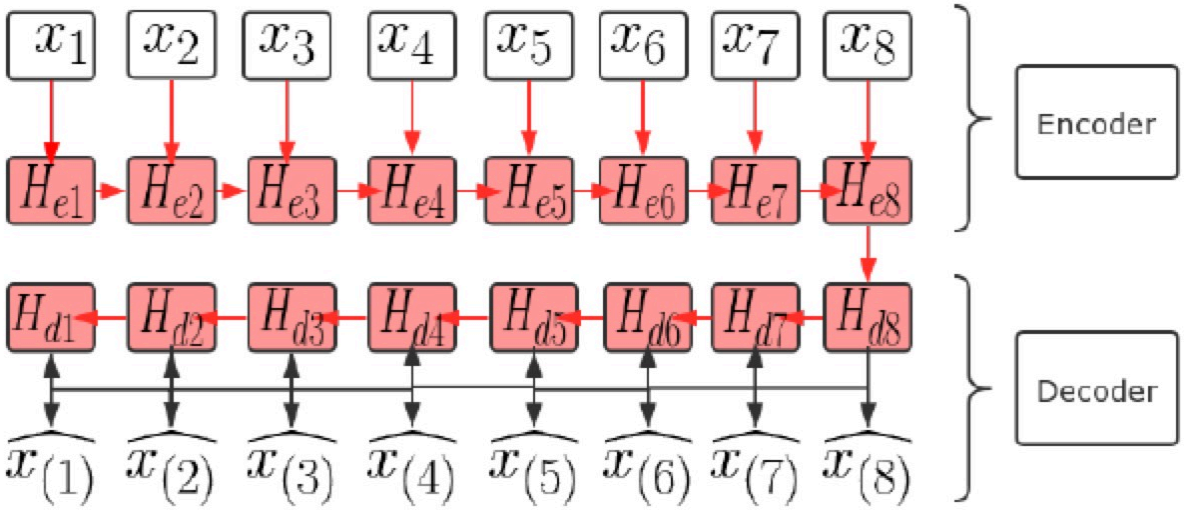


Figure 1: 1-step and multi-step predictions.



The LSTM-based Encoder-Decoder model learns to reconstruct 'normal' time-series behavior, and then uses reconstruction error to detect anomalies. Compared to the current model, it has two separate LSTM layers (encoder and decoder).

4) Anomaly Score calculation method

The anomaly score described in the following is essentially a z-score of prediction differences, normalized over an interval of predictions with window length l, done separately for each of the m varieties.

To simplify, we consider 1-step prediction for a single variate only, $m = 1$, which means the prediction model learns to predict the next 1 time step for one variate, as in Figure 2. With a window length of l, one variate $x^{(t)} \in X$ at time-instant t $(1 \leq t < l)$ has a prediction $\widehat{x^{(t)}}$. We compute an error vector $e^{(t)}$ for variate $x^{(1)}$ to $x^{(l)}$ as $[e_1^{(1)}, e_1^{(2)}, ...e_1^{(l)}]$, where $e_i^{(t)}$ is the difference between its value $x^{(t)}$ and predicted value $\widehat{x^{(t)}}$ at time t.

The error vector for one variate $(m = 1)$ (e.g. $E = [e_1^{(1)}, e_1^{(2)}, ...e_1^{(l)}]$) is modelled to fit a Normal distribution $E \sim N(\mu, \sum)$ with the parameters μ and $\sum$ determined using maximum likelihood estimation. Then, for any point $x_i^{(t)}$, the anomaly score at time t for this 1-dimensional variate is as follows:

$$\text{anomaly\_score}^{(t)} = (e_1^{(t)} - \mu)^T \Sigma^{-1} (e_1^{(t)} - \mu)$$

5) Missing Data

If the missing data is a few data points, we can use the linear interpolation method.

However, if the number of continuously missing data is large, linear interpolation is not a good solution. If we discard the missing data period directly, the anomaly score will be abnormally high in the next time instant and other time instant will be very low.

In order to deal with this problem, I added four features as input in order to let model learn cyclical nature of the trend logs. These four features are ‘sine Of the day’, ‘cosine Of the day’, ‘sine Of the week’, ‘cosine Of the week’ which helps our model to learn cyclical weekly and daily changes.

6) How to choose multi-variate time series data

The SFU facility department provides us the detailed document of the building systems. I tested one variable air volume system, which contains 11 time series trend logs such as room temperature, supply air temperature, Heating Coil Valve output and so on.

---
online test: according to memory.
1. MR or spark implementation. I forget the detailed requirement, but it can be finished in less than 20-30 lines of code.
2. Java or python: implement url parser. You can use the library. But I implement it by myself.
3. About database. Two questions.
One is easier, something like create a table and some SQL statements.
The second one is about distributed database. something about database optimization, like fetch time optimization and concurrency control.
BTW, we don't use oracle.