VT2024: IL2233 Project
Time-Series Prediction and Anomaly Detection

Zhonghai Lu

April 12, 2024

1 Introduction

Anomaly detection is a common task in many medical, industrial, and financial applications where continuous monitoring of the system signals or data is needed in order to prevent anomalies from causing harmful consequences. In data analysis, it is often a first pre-processing data-cleanup step in order to remove outlier data that are inconsistent with the majority.

Anomaly detection is often nontrivial because it relies heavily on what anomaly is. In many situations, it is difficult to strictly define anomalies, and thus difficult to detect any anomalies. Even though anomaly is difficult to define in a general context, it is a concept easy to understand, as anything deviating from normal is an anomaly. Depending on the types of assumed anomalies, there are many different ways of detecting anomalies. From the time-series signal perspective, we may structure the signal anomaly in the three dimensions, namely, *amplitude*, *frequency*, *phase*. In terms of the scope of anomaly, we may define *local (contextual)* or *global* anomaly. A local anomaly occurs in a local context, for example, two peaks rather than one peak in one period, or in a global context, for example, the value is extremely above all other data points. In terms of continuity, an anomaly may be persistent in multiple cycles known as *sequence* anomaly or just in a data point known as *point* anomaly. For the anomaly detection of an entire sequence, one can use classification or similarity comparison (normal vs abnormal) methods after signal filtering and feature extraction.

Assuming an amplitude-related anomaly, we can devise different approaches to identify such anomalies. First, we can define an anomaly criterion to isolate anomalies from the statistical perspective. For example, use the Gaussian distribution, box plot to set a deviation criterion to identify outliers. Then we can devise a threshold-based anomaly detection using the deviation value as a threshold level to filter anomaly points.

In this project, we will perform anomaly detection using the *decomposition-based* approach, the *prediction based* approach, and the *clustering-based* approach. The prediction models to be used are the ARIMA and the neural network models. We consider a univariate time series for the first approach and the ARIMA model. We consider both uni-variate and bi-variate series for the neural network model. We consider a multivariate (actually bivariate) time series for the clustering-based approach.

To prepare you for the time-series anomaly detection with neural networks, we start with basic training on time-series prediction with neural networks.

2 Purpose and Setup

The project contains multiple lab-like tasks. It intends to train students in time-series prediction with neural networks and three approaches for anomaly detection with different data sets. The time-series prediction with neural networks is the base to conduct the prediction-based anomaly detection in this project. For each anomaly detection task, it includes a chain of sub-tasks from exploratory data analysis, feature extraction, model building, prediction towards anomaly detection.

This project uses the following libraries.

- The pandas library.
- The statistical learning library: statsmodels. `https://www.statsmodels.org/stable/index.html`
- The machine learning library: sklearn.
- The deep learning library: keras, tensorflow.
- Other common libraries such as numpy, matplotlib etc.

This project is an individual work.

3 Methods and Tasks

This section describes first methods and then tasks. The methods are given in concrete implementation steps. You can follow the respective methods to implement the tasks.

Task 1 is time-series prediction with neural networks including Task 1.1 and 1.2. Following that are the three anomaly detection methods and tasks. Task 2 is the decomposition-based anomaly detection. Task 3 is the prediction-based anomaly detection that includes Task 3.1 and Task 3.2. Task 4 is the clustering-based anomaly detection. Task 5 is summary. The final task, Task 6, is the documentation and deliverables.

3.1 Task 1. Time-series prediction with neural networks

3.1.0 Method: Time-series prediction based on neural networks

To build a prediction model using a statistical or machine-learning algorithm, we generally go through the following four steps.

- Exploratory data analysis. To analyze time-series data, the first step is usually a routine work of performing exploratory data analysis. This includes a number of visualization and statistics generation tasks such as visualizing the data sequence and exposing its statistical characteristics. In exploratory data analysis, sometimes data transformation is needed to satisfy the assumptions of the underlying model.
- Feature extraction. Besides visualizing data structure and statistics of time-series sequences, one important aspect in exploratory data analysis is to identify and extract features from the time-series data.

- Model construction and validation. After feature extraction, we go for building a model and then validating it.
- Prediction. After validating the model to be good enough, we can use it for prediction.

Depending on the particular statistical or machine-learning model to be applied to a time series, the above general steps may vary. Some steps may become more important or less important. Some steps may be added. For example, a data pre-processing step is essential for time-series prediction with neural networks.

In the following we detail the following main steps for time-series prediction with neural networks:

1. Data pre-processing. The first step, also a key step, in applying neural network models for time-series prediction is to turn the prediction problem into a supervised learning problem. To this end, one can re-arrange the time series into an input-output pair list. The input and output can be a vector.

   For example, we are going to predict the next two values for a sequence [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. We can re-organize the sequence as a data set with "labels". Assume that the input vector has a size of three, and the output vector a size of 2. Then we can create a training data set as follows.

Training set	input vector	output vector
	[1, 2, 3]	[4, 5]
	[2, 3, 4]	[5, 6]
	[3, 4, 5]	[6, 7]
	[4, 5, 6]	[7, 8]
	[5, 6, 7]	[8, 9]
	[6, 7, 8]	[9, 10]
Testing set	input vector	output vector
	[7, 8, 9]	[10th, 11th]
	[8, 9, 10]	[11th, 12th]

Table 1: Transform a series into a data set for supervised learning

2. Model construction. Then we create a neural network with an input layer of 3 neurons and an output layer of two neurons. The number of hidden layers or recurrent layer is up to the designer. You also need to choose a proper activation function for your neural network layer.

3. Model validation. After the neural network is well trained, we can validate its accuracy using *in-sample prediction.* This means you use data points in the time series to compare with the predicted results. For regression type of problem, this is usually mean squared error (mse), mean absolute percentage error (mape) etc.

4. Forecasting. After validating the model, it is time to use it for forecasting future data points. In the example, we can forecast the future two data points, i.e., the 11th and 12th data. This is called *out-of-sample prediction.* The test input vector would be [7, 8, 9], which predicts the 10th, the 11th output; the test input vector [8, 9, 10], predicting the 11th and 12nd output.

It is possible even to predict the 12th and 13th data by using the input vector [9, 10th, 11th], and so on. Of course, we need to be aware of the limitation of the forecasting with the predicated estimates, which may introduce other errors.

3.1.1 Task 1.1 Prediction with synthetic series using MLP, RNN, and LSTM

Dataset and Task Generate the following uni-variate series, and then use neural networks to do in-sample and out-of-sample predictions.

1. An equal-difference series starting from 0, ending to 1 (excluding 1), with a length of 200 points (step = 0.005).

   Design an MLP for one-step prediction. The output vector has a size of 1. Let the input vector be a size of 4.

2. An equal-difference series starting from 0, ending to 1, with a length of 200 points (step = 0.005), plus white noise i.e., random variable with zero mean and 1 variance. You may need to control the amplitude of the noise series in order to control the signal-noise ratio.

   Design an MLP for one-step prediction. The output vector has a size of 1. Let the input vector be a size of 4.

3. A deterministic series sampled from a sinusoidal wave with period 20 seconds, with a sample rate of 100 Hz. Generate sufficient samples (at least 3 periods of data) as needed to achieve good performance, e.g. MSE (mean squared error) below 0.5.

   Design an RNN and a LSTM for two-step prediction. The output vector has a size of 2. Set the input vector size by yourself.

4. A stochastic series sampled from a sinusoidal wave with period 20 seconds, with a sample rate of 100 Hz, plus random white noise i.e., random variable with zero mean and 1 variance. Control the amplitude of the noise with a fractional number, e.g. 0.1.

   Design an RNN and a LSTM for two-step prediction. The output vector has a size of 2. Set the input vector size by yourself.

Implementation: Following the steps in Section 3.1.0 to visualize the relevant data and realize the task.

A few notes, hints, tips:

- You need to re-shape your data set into a supervised learning data set before feeding data to your neural networks. Write a function for this purpose.
- When you train your neural networks, use 80% of data for training and 20% of data for testing. Write a function for the data splitting.
- Train your neural networks with sufficient epochs to obtain good accuracy.
- Pay attention to which activation function (e.g. sigmoid or ReLU or tanh etc.) to use.

Discussion: Discuss in your technical report:

(1) How have you designed the neural network for each series?

(2) What hyper-parameters do you use in each case?

(3) Can the neural network fit well to the specific series well? What are the accuracy merits?

(4) How is the performance of LSTM in comparison with RNN? Is the LSTM outperforming the RNN in general?

3.1.2 Task 1.2 Predict white noise, random walk, an ARMA process using neural networks

Dataset. We use three synthetic data sets, each with 1000 data points.

1. A pure white-noise signal.

2. A random-walk series.

3. A stationary series generated by an ARMA(2, 2) process. Make sure that the process with right parameters generates a stationary series.

Task Your task is to design and test an neural network to build a prediction model for the three series, ie., white noise, random-walk, and ARMA(2, 2) process.

Implementation: Following the steps in Section 3.1.0 to visualize the relevant data and realize the task.

A few notes, hints, tips:

- Which neural network to use is your choice. You can use MLP, RNN or LSTM to design the neural network. You need to re-shape your data set into a supervised learning data set before feeding data to your neural networks. Write a function for this purpose.

- How to choose the hyper-parameters and how to train are also up to you. The goal is to minimize the prediction error.

- The prediction error series should be a white noise.

Discussion: Discuss in your technical report:

(1) How have you designed the neural network?

(2) What hyper-parameters do you choose? Give short motivation.

(3) Can the neural network fit well to the white noise series?

(4) Can the neural network fit well to the random walk series?

(5) Can the neural network fit well to the ARMA process? Why or Why not?

3.1.3 Task 1.3 Comparison with ARIMA-based modeling and prediction

In this task, we compare neural network-based modeling and prediction with ARIMA-based modeling and prediction.

Dataset. Generate a certain-length (e.g. 50 points) Fibonacci series and add standard Gaussian noise by yourself. Control the signal-noise ratio properly by controlling the amplitude of the signal and the noise, e.g. 95% of signal, 5% of noise.

The Fibonacci series is a series of integer numbers (except first 0) in the following sequence:

{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...}

The sequence F_n of Fibonacci numbers is defined by the recurrence relation:

$$F_n = F_{n-1} + F_{n-2}$$

with seed values $F_0 = 0$ and $F_1 = 1$.

Task Your task is to build four models for the Fibonacci series, use the models to predict future values, and make a comparative evaluation. Specifically, following the sketch below to do the task.

1. Generate a Fibonacci series. You decide the length of the series to generate. Split the data into a training set and a test set. You decide the splitting ratio.
2. Build an MLP model for the series and use it for prediction.
3. Build an RNN model for the series and use it for prediction.
4. Build an LSTM model for the series and use it for prediction.
5. Build an ARIMA model for the series and use it for prediction.
6. Compare the accuracy (MSE, MAE, MAPE of the prediction errors) of the four methods for the training set and the test set.

Implementation: For neural-network based modeling and prediction, follow the steps in Section 3.1.0 to visualize the relevant data and realize the task.

For ARIMA-based modeling and prediction, follow the Box-Jenkins methodology (see Lab 2) to visualize the relevant data, transform the data (in whichever way it is meaningful), draw ACF, PACF graphs etc. to determine and train a proper ARIMA(p, d, q) model for the series and use it for in-sample and out-of-sample predictions.

A few notes, hints, tips:

- When using neural networks (MLP, RNN, LSTM), you need to re-shape your data set into a supervised learning data set before feeding data to your neural networks. Write a function for this purpose. How to re-shape your data (determining input vector length, output vector length) is your choice.
- How to choose the hyper-parameters of the neural networks and how to train (which training parameters to choose) are also up to you. The goal is to ensure training convergence and minimize the prediction error.
- The prediction error (residual) series should be ideally a white noise.

Discussion: Discuss in your technical report:

(1) How have you designed the neural networks? What hyper-parameters do you choose? Give short motivation.

(2) How have you trained your neural networks? Report the training epoch, learning rate, optimization algorithm.

(3) Can your MLP, RNN, LSTM networks fit well to the Fibonacci series? Which one is best?

(4) Can your ARIMA model fit well to the Fibonacci series?

(5) Which modeling approach, neural network based or ARIMA based, gives a better performance? Why? Discuss the pros and cons of different modeling approaches.

(6) Which model, ARIMA or NN, is more tolerant to noise? Increasing the noise ratio and report how the accuracy of the two models will be worsened.

3.2 Task 2. Decomposition-based anomaly detection

3.2.1 Method: Anomaly detection by decomposition

Decomposition is a powerful means to understand the structure of the series by splitting it into multiple components, such as trend, season (fixed period), and remainder. From the remainder series, we can analyze if an outlier exists according to a certain threshold, for example, the z-score, or outliers on the box plot.

One can use the following steps to find and visualize anomalies:

- Use a decomposition method, e.g. the classic decomposition or STL, to split the components of the series.
- Visualize the components.
- Set a threshold, and find the time index and value of the anomalies.
- Mark and visualize the anomalies in the series.

3.2.2 Task 2. Anomaly identification in global land temperature changes

Dataset: The global temperature statistics data set records various temperature statistics for many years. We are going to use data from year 1880 to 2020. It is recorded per month, meaning that each year has 12 data points. As it crosses 141 years including old times, there might be some missing data points.

The file is named GlobalTemperatures.csv, which you can download from the Canvas course room. This data set is from Kaggle. This is a comprehensive data set containing many columns. Besides the date-time index, we use only one column 'LandAverageTemperature'.

Task: Your task is to (1) insert one or two anomaly points as ground truth, and then (2) identify the anomaly points from the above data set for the 'LandAverageTemperature' data using the decomposition method.

Specifically, you change a single data point as anomaly. Assume the data sequence is called 'series' as a pandas data frame. Do the following change:

```
series.loc["1998-12-1"] = 20
```

Implementation: Following the steps in Section 3.2.1 to visualize the relevant data and realize the task.

A few notes, hints, tips:

- Write a function to calculate the z-score.
- Write a function to identify outliers from the box plot.
- You can use the simple decomposition method (seasonal_decompose()) or STL() in statsmodel to realize the decomposition.
- To highlight better the anomaly, you may want to focus only on part of the data, for example, from 1995-01-01 to 2000-01-01 (61 data points).

Discussion: Discuss in your technical report:

(1) Can the decomposition clearly separate the trend, season (constant period), and remainder components?

(2) When decomposing the series, is there a general rule to determine which part belongs to a trend, a season, or a remainder? Or is it embedded in and thus dependent on each individual algorithm?

(3) Is there a growing tendency in the trend series?

3.3 Task 3. Prediction-based anomaly detection

3.3.0 Method: Anomaly detection by prediction

The basic idea of prediction-based anomaly detection is to generate a remainder series using a prediction model, and then identify outlier points from the remainder (error) sequence. Typically, there are four steps:

- Build a model from the underlying series. The prediction model can be designed using any statistical or machine learning algorithm.
- Use the model to do in-sample prediction, i.e., for the data points in the original time series.

  Note that there are a few initial points should be skipped. For an ARIMA model, how many points shall be skipped depends on the lag number. For a neural network, how many points shall be skipped depends on the size of the input vector.
- Find the point-wise difference, i.e., generate a remainder series between the predicted series and the actual series.
- Identify anomalies from the remainder series. If the difference is larger than the deviation threshold, this data point is marked as anomaly.

The idea of predication-based anomaly detection is quite similar to the decomposition-based anomaly detection. However, the decomposition-based anomaly detection is limited to existing series and cannot do any prediction. This makes it inconvenient to deal with new data points (out-of-sample data points). To deal with an additional data point or multiple additional data points would mean to re-run the decomposition with the inclusion of the new data points. This is usually computationally intensive. In contrast, the prediction-based method can predict the next-step or next multi-step data points, and if the actual data points appear, a comparison can be made in real time. Thus it can handle real-time or online anomaly detection more efficiently.

3.3.1 Task 3.1 Anomaly detection for uni-variate series with ARIMA

Dataset: The global land temperature anomalies data (1880-2020). The temperate is recorded per year. It contains 141 data points.

Task: Your task is to identify the anomaly points from the above data set using the prediction-based anomaly detection with ARIMA. Suppose that the anomaly ratio is 2%.

Implementation: Following the steps below (also referring to the steps in Section 3.3.0 to visualize the relevant data and realize the task.

- Exploratory data analysis. Draw seven graphs: line plot, histogram, density plot, heatmap, box plot, lag-1 plot, and lag-2 plot for the series.

  If the series is not stationary, differencing the series, until it is stationary, but not over-differenced.

- Feature extraction. Summarize the statistical features of the series, such as mean, standard deviation. Plot the temporal correlation of the series by drawing its acf and pacf graphs.

- Model construction and selection. Follow the Box-Jenkins methodology step by step to construct an ARIMA(p, d, q) process to model the series.

  You can either try different combinations of (p, d, q) values and select an optimal model based on AIC. Or alternatively, you can first identify the maximum (p, q) values using the ACF, PACF graphs, and then use the auto_arima() function (in the pmdarima library) to automatically search for optimal parameters based on AIC.

- Prediction. Use the model to do in-sample prediction, and generate the prediction error series. Check if the remainder (prediction error) series is random.

- Anomaly definition and detection. Implement the Z-score, Boxplot criterion to identify the outliers and mark the anomaly points in the original series.

Discussion: Discuss in your technical report:

(1) Since this task uses a different approach (prediction-based anomaly detection) from Task 2 which uses decomposition for anomaly detection, describe what the differences of the two methods are?

(2) Do they achieve the same results? Why or Why not?

(3) Given the anomaly ratio of 2%, what is the value of z-score?

3.3.2 Task 3.2 Anomaly detection in ECG signals with LSTM

Dataset: We use the ECG signals from the MIT-BIH Arrhythmia database obtainable from the PhysioNet service [2][3]. This is a well-known database of cardiac arrhythmias. The MIT-DB arrhythmia sub-dataset contains 48 ECG records, each record is about 30 minutes long and the sampling frequency is 360 Hz and each signal consists of two leads.

In this project, we choose one of the records labeled 100 and divided the first 80% signals (17280 points) as the training set and the left 20% of the records as the test set. The data format is in .csv. The file name is 100.csv.

Task and implementation: Your task is to build a prediction model for the ECG series using LSTM. You are supposed to do the following.

- Exploratory data analysis. Draw the line plot, lag-1 plot, and lag-2 plot for the two signals in the data set.
- Model construction. Split the data set into a training set (80%) and a test set (20%). Design and train an LSTM which can predict the next value given a few known values.

  When you re-organize the data structure, let the input vector be a size of 4, 8, 16 (Each time, use one of the 3 options) and the output vector be a size of 1.

  Treat the time-series data in two ways: 2 individual uni-variate series, 1 bi-variate series.
- Validate the quality of the model. Calculate the MSE, MAPE etc. of the prediction model for the input size being 4, 8 and 16.
- Anomaly definition and detection. Calculate the prediction error series. Assume 0.5% of error rate, find the anomaly points.
- Visualize the anomaly points in the prediction error series.

Discussion: Discuss in your technical report:

(1) How have you designed the neural network?

(2) What hyper-parameters do you choose? Why?

(3) Can the neural network fit well to the ECG signals? What is the accuracy of your model?

(4) How much is the influence of the input vector size on the prediction accuracy?

(5) How many epochs do you use for training your LSTMs in order to achieve good accuracy? How much is the learning rate? What is your training optimization algorithm (e.g. SGD, Adam etc.)?

(6) Which way of treating the time series data gives better accuracy: two uni-variate series or one bi-variate series? Why?

3.4 Task 4. Clustering-based Anomaly detection

3.4.1 Method: Anomaly detection by clustering

Anomaly detection can be done by un-supervised learning as well. Clustering is such a straightforward possibility. By clustering, data instances that fall outside of defined clusters or with a small size can potentially be marked as anomalies. If data instances fall in a cluster, we can still use the distance to filter them.

With un-supervised anomaly detection, we actually have no idea if the data are all right or not, because there is no ground truth. We usually need to involve some manual experience or knowledge before the analysis: Assuming that there are some anomalous points in the data statistically or deterministically. By this assumption, we may further assume a meaningful outlier ratio. Then we can set an outlier ratio, e.g., 0.1%, 0.05% in order to find how many data points belong to anomaly and calculate their distance to their respective centroid. This can be largely considered as a hyper-parameter that needs a trial or grid search to set it to be meaningful for the problem at hand. Since clustering is based on distance, the outlier ratio can be translated into a distance threshold.

We can use the following steps to find and visualize anomalies:

1. Visualize the multi-variate series, plotting the line plot and scatter plot.
2. Determine the number of clusters, and do clustering on the data.
3. Calculate the distance between each point and its nearest centroid (the centroid of its belonging cluster).
4. Use the outlier ratio to calculate the total number of anomalous points (outliers). According to the point-wise distance, those points with largest distances are considered to be the outliers. This step leads naturally to set the distance threshold. By this step, the data points are split into a normal subset and an anomalous subset.
5. Visualize anomalies in a cluster view using the scatter plot (for 2D, 3D data), and in a time series view using the line plot.

3.4.2 Task 4. Anomaly detection in a bivariate series

Dataset: Generate a two-variable (X_1, X_2) time series, each variable with 200 data points. X_1, X_2 are stochastic variables subject to Gaussian distribution. X_1 has a probability density function (pdf) with mean $\mu = 0$, variance $\sigma = 2$. X_2 has a probability density function (pdf) with mean $\mu = 1$, variance $\sigma = 2$.

Task: Assume an anomaly ratio of 2%. Employ the K-means clustering algorithm and Self-Organizing Map (SOM) to identify the anomaly points using the clustering-based anomaly detection method.

Implementation: Following the steps in Section 3.4.1 to visualize the relevant data and realize the task.

Discussion: Discuss in your technical report:

(1) How do you set the number of clusters? Why?

(2) Which distance metric do you use? Are there other distance metrics which might be useful for this task?

(3) Do the two different clustering methods (K-menas and SOM) achieve the same results? Discuss why or why not.

3.5 Task 5. Summary

After completing the project tasks, it is time to take a break, and reflect what you have learned.

You have learned two classes of approaches to perform time-series modeling and prediction. One is the statistical learning approach, specifically, ARIMA, and the other is the deep learning approach using neural network (NN) models such as MLP, RNN, LSTM.

Discussion: Answer the following questions in your technical report.

1. Describe the two approaches in your own words.
2. Compare the two approaches and discuss their strength and weakness.
3. List key points of the two approaches and key pros and cons in a table shown below (Table 2). Feel free to extend the table by adding more columns, if necessary.

Approach	Assumption	How modeling & prediction are done?	Strength	Weakness
ARIMA				
NN models				

Table 2: Comparisons of prediction approaches

You have exercised three different anomaly detection methods in this project. This is not a complete set. You can find other methods in the literature as well.

Discussion: Discuss in your technical report:

(1) Describe the three different methods.

(2) Compare and discuss the strength and weakness of the three methods.

(3) List key points of the approaches and key pros and cons in a table below (Table 3).

(4) Find another anomaly detection method from literature, and fill in the table in the row "Another method".

Feel free to extend the table by adding more columns, and more rows for more methods, if necessary.

Method	How is anomaly detected?	Anomalies assumed	Strength	Weakness
Decomposition-based				
Prediction-based				
Clustering-based				
Another method				

Table 3: Comparisons of anomaly detection methods

3.6 Task 6. Project Documentation and Deliverables

After completing the project tasks above, you are not complete yet. You need to do the following to finalize:

1. Result check and approval: Ask a teaching assistant to check your results for approval (More information about this will be posted in the Canvas course room). The lab assistant will also ask questions present in this manual and possibly other questions regarding your implementations.

2. Write a technical report: write a short technical document, where you write down what you have done, how you have done it, what results you have obtained, and discuss the results (often the questions in the project manual contains discussion points), and present any conclusions you may draw in your own words. Try to use figures and tables to assist your explanations and discussions.

   Note that, **when answering questions or presenting discussions in your technical report, you need to copy & paste the questions from the project description into your technical report**. In this way your report is self-contained.

3. Prepare deliverable. Prepare a deliverable which zips all your source code and the result figures.

   Write a short Readme.txt file to describe the zip folder structure and purpose of each file.

4. Prepare PPT or PDF slides about the project. You are going to use the slides for your oral presentation in the Final workshop session of the course.

5. Submit report and slides: After (1) (2) (3) (4), you submit your technical report and slides to the Canvas course page below.

   Your report and slides will be reviewed and, if accepted, your project will be counted as completed. If not, comments will be given to you for revision and re-submission of your technical report.

   `https://canvas.kth.se/courses/46239`

4 Appendix

4.1 Anomaly criterion

By single-point anomaly detection, the actual data point which deviates too far from its predicted value would be considered as an anomaly, because it does not follow the predicted pattern (assuming the model is correct). Reflected in the prediction error series, the error with an absolute value too larger than zero is considered to be anomaly.

To find anomalies in the prediction error series is to find the outliers of the series. The following two are common outlier qualification methods: z-score and Box-plot identification. If the real errors follow the standard normal distribution, both criteria are statistically solid.

1. z-score calculation

   z-score, also known as the standard score, is the commonly used indicator which measures the deviation of a data point from the whole series data. The z-score can be presented as:

   $$z = (x - \mu)/\sigma \tag{1}$$

   where x is the data value, μ is the mean of the series data, and σ is the standard deviation of the data set. The large the data value deviates from the mean, the higher the z-score would be. Usually, the data points with the z-score higher than 3 ($z > 3$) could be considered as outliers.

   For example, assume a series $\{8, 10, 4, 5, 5, 4, 7, 25, 8, 9, 9, 7, 10, 1, 9, 6\}$. Should 25 be regarded as an outlier? The mean μ of the series is 7.94. The standard deviation σ is 5.20. The standard score z of number 25 is 3.29, which is higher than 3. Thus 25 can be considered as an outlier.

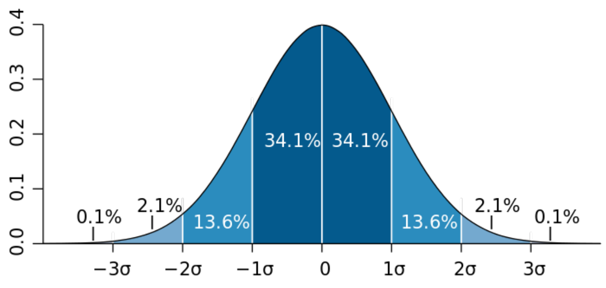

Figure 1: Normal distribution and z-score [Wikipedia]

   The reason for z equal to 3 is due to the 3-sigma (σ) rule. Consider a normal distribution, about 68% of values are within one standard deviation σ away from the mean; about 95% of the values lie within two standard deviations 2σ; and about 99.7% are within three standard deviations 3σ. This fact is known as the empirical 68-95-99.7 rule, or the 3-sigma rule. See Figure 1 from Wikipedia.

   There is a z-score function in scipy. Do "from scipy.stats import zscore", then use the function call zscore().

2. Box-plot identification

Boxplot is another common statistic method to isolate outliers. It uses the first quartile and the third quartile of the series data as the references. Figure 2 shows how it works. Q1 is the first quartile of the series data and Q3 is the third quartile. The "Maximum" and "Minimum" are the thresholds. The data points larger than "Maximum" or smaller than "Minimum" are considered as outliers.

The formulas of the thresholds are:

$$Minimum\_Threshold = Q1 - k \cdot (Q3 - Q1) \tag{2}$$

$$Maximum\_Threshold = Q3 + k \cdot (Q3 - Q1) \tag{3}$$

Smaller k means a more tight threshold and data are more likely to be detected as outliers. Usually, k equals to 1.5.

Take the series {8, 10, 4, 5, 5, 4, 7, 25, 8, 9, 9, 7, 10, 1, 9, 6} as an example. Should 25 be considered as an outlier? The first quartile Q1 is 5, and the third quartile Q3 is 9. The maximum threshold is 15, and the minimum threshold is -1. Value 25 is larger than the maximum threshold, so it is an outlier.

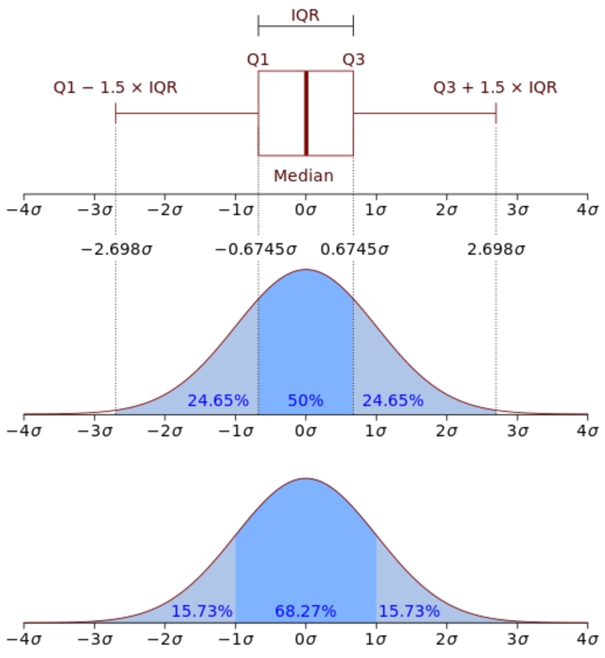

Figure 2: Box plot and its corresponding coverage in normal distribution [Wikipedia]

Again consider a normal distribution, the choice of $k = 1.5$ in the Boxplot is equivalent to the choice of 2.698σ, i.e., the normal data fall within 95% (2σ) to 97.5% (3σ) of the total data. See Figure 2 from Wikipedia.

4.2 Evaluation of anomaly detection accuracy

After anomaly detection, one needs to evaluate the quality of anomaly detection.

Since anomaly detection is similar to anomaly classification, we can use the metrics for classification problems to evaluate its quality. In the case of having ground-truth data (the data set has marked anomaly points), it is often useful to build a confusion matrix to visualize the results. For example, if the anomaly detection problem may be treated as a binary classification problem, then we can build a two-class confusion matrix. For multiple classes, one can also draw a multi-class confusion matrix.

Using the confusion matrix for the binary classification problem, one can calculate multiple evaluation metrics, such as precision, recall, F1 score etc.

The machine learning library sklearn provides functional support for model evaluation. Refer to details in `https://scikit-learn.org/stable/modules/model_evaluation.html`.

References

[1] Andrzejak, Ralph G et al. "Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state". In: Physical Review E 64.6 (2001), p. 061907.
[2] Moody GB, Mark RG. The impact of the MIT-BIH Arrhythmia Database. IEEE Engineering in Medicine and Biology 20(3):45-50 (May-June 2001). (PMID: 11446209)
[3] Physionet. The mit-bih arrhythmias database.
http://www.physionet.org/physiobank/database/mitdb
[4] Plawiak, Pawel. "ECG signals (1000 fragments)". In: Mendeley Data, v3 (2017).