**Team members:**

*Yunkun Niu, Qi Tang, Yibin Qian, Bilal Waris*

**Coordinator:**

Aino Siippainen

**Guiding teacher:**

*Sari Peltonen*

**Intel:**

*Rauno Tamminen, Kalle Koivisto*

*Intel Corporation && Tampere University of Technology*

*Anomaly Detection*

In digital signature requests

Table of Contents

[Agenda 3](#_Toc422086627)

[Background 3](#_Toc422086628)

[Problem 3](#_Toc422086629)

[Solutions 4](#_Toc422086630)

[Project Skills 4](#_Toc422086631)

[Git Repository 4](#_Toc422086632)

[1 GHSOM (Growing Hierarchical Self-Organizing Map) 4](#_Toc422086633)

[1.1 Why we left GHSOM 5](#_Toc422086634)

[1.2 Materials 5](#_Toc422086635)

[2 GMM (Gaussian Mixture Model) 5](#_Toc422086636)

[2.1 Installation 5](#_Toc422086637)

[2.2 Experiment 6](#_Toc422086638)

[2.3 Why we left GMM 6](#_Toc422086639)

[2.4 Materials 7](#_Toc422086640)

[3 RP (Relation Probability) 7](#_Toc422086641)

[3.1 Algorithm 8](#_Toc422086642)

[3.2 Accuracy 8](#_Toc422086643)

[3.3 Performance 9](#_Toc422086644)

[3.4 installation 9](#_Toc422086645)

[Conclusion 9](#_Toc422086646)

# Agenda

The agenda is to build and demonstrate a system that can learn to detect anomalies in input signing request data. The detection is done by data pattern recognition and preferably happens in real-time. The system is taught by feeding it with big number of different sorts of valid signing request data e.g. from existing logs.

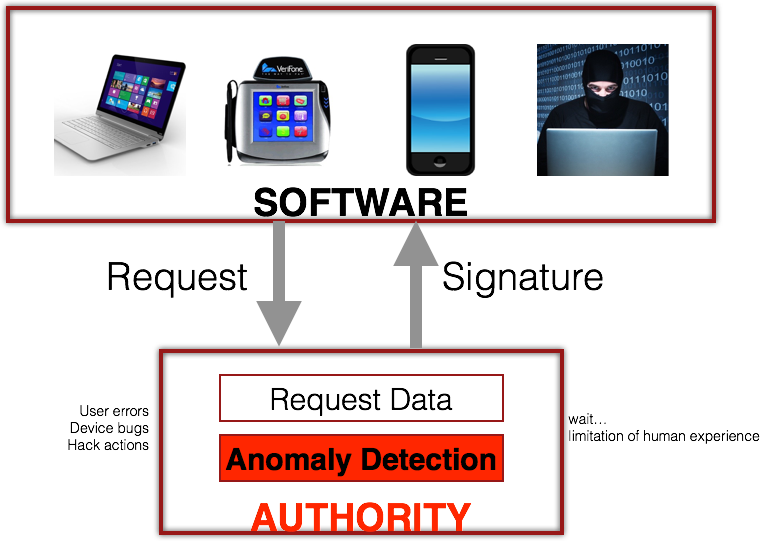
# Background

The logic of the detection system is quite analogue to different recognition methods in signal processing (e.g. face and speech recognition). There are also some studies of anomaly detections in software science e.g. URI screening using N-grams (Mikhail Zolotukhin, Timo Hämäläinen, Antti Juvonen: Online anomaly detection by using N-gram model and growing hierarchical self-organizing maps. IWCMC 2012: 47-52).

Signing systems play big roles in companies’ core security. Therefore controlling and monitoring the input data, especially in more complex infrastructures, is one way to improve security.

# Problem

The root of the problem is that some people have to sign software and hardware images and configuration data on daily basis. Manual review and approval for signing requests cannot be applied as at the same time quick reactions to problems are required in customer oriented environment. A picture below illustrates the problem.



However, we would like to have some controls in place preventing one individual from misusing his or hers signing rights. The red background place in above illustration is where we are going to control the anomalies.

In complex signing systems the input data format can vary greatly and is typically handled as binary data:

* Variety of different data header formats, which contain changing amount of critical data entries.
* Some data types also contain critical data in its payload.

This causes challenges in data screening and security monitoring when we must automatically detect malicious input data.

# Solutions

At this project we want to address the challenge by demonstrating a machine learning system which identifies norm for daily usage, and detects anomalies which deviate too much from the norm. In total, we tried and analyzed three machine learning algorithms, they are SOM, GMM and RP.

Project Skills

* Software development
* Algorithms & data structures
* Machine learning
* Mathematics

# Git Repository

<https://github.com/AnomalyDetection/demola>

# 1 GHSOM (Growing Hierarchical Self-Organizing Map)

The first idea which comes into our minds is GHSOM, because it is introduced in the description of the project.

The growing hierarchical SOM is an artificial neural network model with hierarchical architecture composed of independent growing self-organizing maps. By providing a global orientation of the independently growing maps in the individual layers of the hierarchy, navigation across branches is facilitated. The GHSOM is used as a basis for data organization in both the SOMLib and SOMeJB systems, and forms a core component in the KONTERM project.

## 1.1 Why we left GHSOM

We first tried to use SOM to distinguish good data and anomalies. The accuracy is rather disappointed, only around 70%. We analysed the reason of the result. SOM needs not only good data but also anomaly data. For example, if SOM is trained only with normal human pictures, it will classify a human sculpture as a human, rather than a sculpture. This is a wrong classification and caused by lack of sculpture pictures as training data. For this project, the real status of request data is that only rare anomalies exit. Thus SOM is not a suitable solution.

GHSOM developed from SOM, and we only have good data and don’t have anomaly data. So based on above analysis, we have reasons to believe that GHSOM will not work on the real request data either.

One more reason of leaving it is that the GHSOM Project is only a Research Prototype and not continuous any more. A discontinuous project would be a risk for real commercial application.

So we come up with new algorithm naming GMM which can fulfill our requirements.

## 1.2 Materials

Homepage of GHSOM: <http://www.ifs.tuwien.ac.at/~andi/ghsom/index.html>

GHSOM Project: <http://www.ifs.tuwien.ac.at/~andi/ghsom/download.html>

# 2 GMM (Gaussian Mixture Model)

A Gaussian Mixture Model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

## 2.1 Installation

**System requirements:**

1. Python (>=2.6)
2. Python lib ‘NumPy’ (>=1.6.1): <https://github.com/numpy/numpy>
3. Python lib ‘SciPy’ (>=0.9): <http://www.scipy.org>
4. scikit-learn: <http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GMM.html>

**Manual:**

1. Download source codes from git repository:

<https://github.com/AnomalyDetection/demola>

1. Prepare train data:

* put one good data file and one anomaly data file under path $PROJECT/data/
* data MUST be transformed into numbers.

1. Prepare check data:

* put the data file to be detected under path $PROJECT/data/
* data MUST be transformed into numbers.

1. Configure gmm:

* edit file $PROJECT/gmm/main.py,
* modify the values of the three variables for data path under the instruction of notes.



* modify the value of ‘used\_column’ for choosing the relevant features.



1. Run gmm:

* python main.py

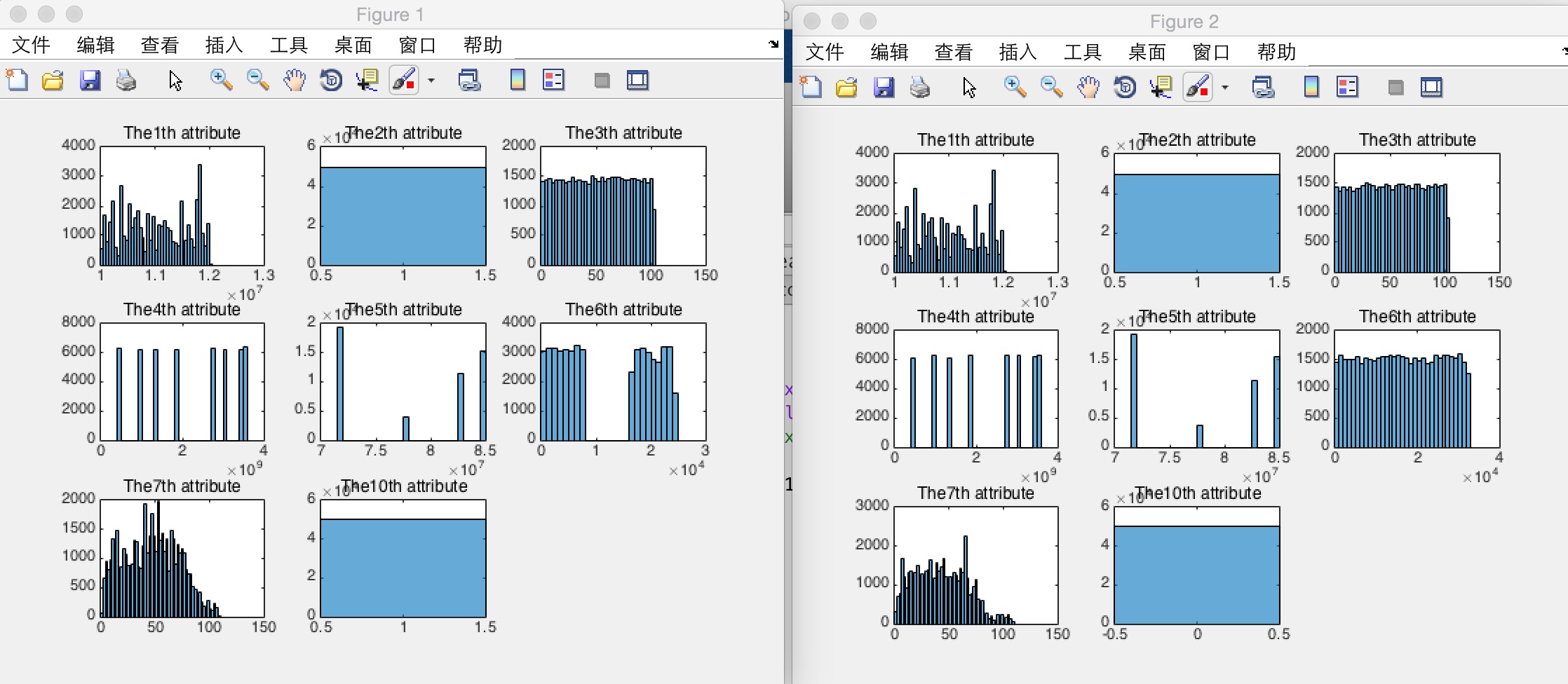
## 2.2 Experiment

We first tried to use all the features to detect anomalies, the accuracy is rather amazing, 100%. But when we remove the timestamp feature, our system can not detect any anomalies out at all. We analysed two reasons for the result.

## 2.3 Why we left GMM

The first, there is a big difference between the timestamp feature of good data and anomaly data. So gmm could use this feature to detect anomalies.

The second, the other feature values are modified manually. It is not correspond with the real distribution. And the manually modified distribution does not differ from that of the good data at all. Here is a comparison of the distributions of good data and anomaly data.



left figure is good data, right figure is anomaly data

From above figures, it is easy to find out that the distributions of the same features have high similarities. It is almost same. That means no feature is suitable for GMM to use for detection.

Although GMM is theoretically designed for anomaly detection, it does not work on our simulated data set. And for the security reason, Intel can not offer us the real data. Thus, we have to move on to find a new algorithm, RP, for this specific artificial data set.

## 2.4 Materials

Coursera online course:

<https://www.coursera.org/learn/machine-learning/home/info>

# 3 RP (Relation Probability)

The idea of RP is to detect anomalies based on the probability of related features. It used only daily valid data for training. We find that the probability of the values of related features of a daily valid request data is very high. So if the probability of a new request is low, then it is detected as anomalies.

## 

## 3.1 Algorithm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Feature 1 | Feature 2 | … | Feature M | … | Feature N |
| Key1 | V1\_1 | V1\_2 |  | V1\_M |  | V1\_N |
| Key2 | V2\_1 | V2\_2 |  | V2\_M |  | V2\_N |
| Key3 | V3\_1 | V3\_2 |  | V3\_M |  | V3\_N |
| … |  |  |  | … |  | … |
| KeyN | VN\_1 | VN\_2 |  | V5\_M |  | VN\_N |

A sample good data as training set

1. Make a key for each request. The key is a combination of the values of related features selected. For example, select Feature 2, 3 and 7 to be used for detection. Then,

Key3 = ‘V3\_2 V3\_3 V3\_7’.

If there are N requests, then N keys will be generated.

1. There are only M (M < N) unique keys. M is far smaller than N, Because the training set made of historical good requests. The values of the selected features repeat a lot.
2. Calculate the Probability of each key.

P(key\_M) = (number of copies of keyM) / N;

Here, the training phase is done. RP generates a hash-table ( key to probability).

|  |  |
| --- | --- |
| KEY | PROBABILITY |
| Key1 | P(Key1) |
| Key2 | P(Key2) |
| … | … |
| KeyM | P(KeyM) |

Hash-table

1. Detect a new request.
2. get the key of the new request as above step 1.
3. Get the probability of the key from hash-table.
4. If the probability is low ( close to 0 ), then this new request can be defined as anomaly.

## 3.2 Accuracy

Now, the accuracy of RP is 99.33%. The reason why it is not 100% is that we do not have enough good data for training. Theoretically, ‘enough data’ means a number of data which can represent the whole data set. Generally, 3 or 6 months historical data are enough.

3.3 Performance

The process of anomaly detection can be divided into two phases, training and detection.

Training phase’s time cost could be long and depended on the size of training data set. But this phase is one-off. It is executed only once for the whole life-span of the system.

Detection phase is real-time. Because of the key-to-probability hash-table, it cost O(1) time to detect a new request. The real time cost for detecting a new request is less than 1ms.

## 3.4 installation

1. Download source codes from git repository:

<https://github.com/AnomalyDetection/demola>

1. Prepare historical good data file for training.
2. Prepare new data file to be detected.
3. Configure data path

Edit conf file ‘$RP/conf.py’

Modify the below two variables to training file and detection file respectively.



1. Select features

Modify the value of below variable



1. Execution

Command: cd $RP/ && python rp.py

# Conclusion

In this project, we tried three machine learning algorithms to solve the anomaly detection problem. First we used SOM, the basis of GHSOM, but the result is not satisfied. Because there are rare anomalies in real system environment. Thus SOM is not suitable for it. Then we used GMM which is designed for anomaly detection. The first attempt of GMM is perfect, 100% accuracy. But if the detecting data are generated by slightly modifying good data manually, GMM failed. Cause the artificial data do not correspond to the distribution of the real data. Even though, GMM is a good algorithm for anomaly detection and deserved to be tested in the real system environment. Finally, we designed and implemented a new algorithm, RP. RP works fairly well on the artificial data. And more it is high efficient and can detect new requests in real-time. Overall, different algorithms match with different data. Each of them has its specific application situation. RP, designed and implemented for the given data by us, performs best for this project.