SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

Highly Dimensional Anomaly Detection

AUTOMATIC VERIFICATION OF INTELLIGENT SYSTEMS

Professor:
Enrico Tronci

Students:
Andrea Di Marco
Jemuel Espiritu Fuentes
{dimarco.1835169, fuentes.1803530}@studenti.uniroma1.it

Academic Year 2022/2023

Contents

1 Introduction

Anomaly detection is a key element of many real world applications like power networks or financial transactions and it is a vital aspect in both the security and safety of critical systems. [1]

Many applications are relying on model based algorithms but in case of highly dimensional data streams, especially in real time systems, this is not a practical solution. An alternative is based on the extraction of useful univariate summary statistics from the observed data in order to detect anomalies.

For our Automatic Verification of Intelligent Systems project we implemented an online anomaly detection system as proposed by [2] using `C++` and `Python` which works by computing summary statistics in an offline setting to be used as a comparison with real time data. The high dimensionality is mitigated by the use of two different algorithms depending on the use case. We subsequently performed optimization on the models in order to infer the most suitable values for the two main hyper-parameters α and h on a number of different situations. Ulterior tests comparing the two models were also a part of our activities in order to observe the sensitivity of various statistics. Lastly we discuss additional improvements and show some key weaknesses in the algorithms.

2 Implementation

We decided to implement the proposed anomaly detection algorithm in `C++` and interface it with `Python` during the optimization and testing phase. We used the `Eigen` [3] library to facilitate linear algebra operations.

The models are divided in an offline training component and an online detection phase are shown in 1. In the offline phase the summary statistics are computed based on nominal data. Then in the online phase the summary statistics are compared to new data point to detect if an anomaly is present. The online phase makes use of a CUSUM-like algorithm to decide if the amount of observed differences between baseline and new data point are enough to raise an alarm. The threshold for the alarm is represented by the hyper-parameter h.

The computation of summary statistics, in both offline and online phase, is done by using a GEM or a PCA based approach depending on the use case, specifically the intrinsic dimensionality of the datasets.

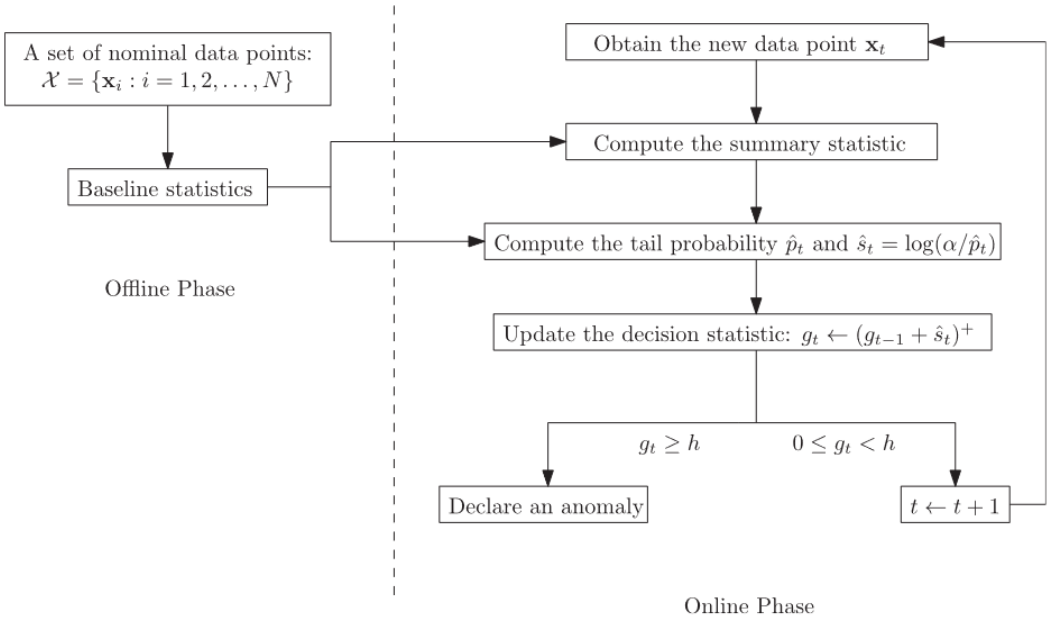

Figure 1: System diagram from [2]

2.1 Geometric Entropy Minimization

The first summary statistic is based on Geometric Entropy Minimization in which an acceptance region A is determined such that only outliers sit outside of A with respect to α. The figure 2 shows the pseudocode of how the GEM based summary statistics is obtained. In this method first the dataset X is uniformly randomly partitioned in S_1 and S_2 with sizes N_1 and $N_2 = |\mathcal{X}| - N_1$. Then $\forall x_j \in S_2$ the kNNs of x_j among each element of the set S_1 is computed.

Let $e_j(i)$ be the euclidean distance of x_j to its ith NN in S_1, we can represent the

sum of the distances from x_j to its k nearest neighbours as:

$$d_j \triangleq \sum_{i=1}^{k} e_j(i)$$

After computing $\{d_j : x_j \in S_2\}$, this set is sorted in ascending order and the acceptance region A is the $(1-\alpha)$ fraction of $x_j \in S_2$ corresponding to the smallest $(1-\alpha)$ fraction of d_j's. Then a new data point $\mathbf{x}_t$ is considered an outlier if its sum of distances to its kNNs among S_1, denoted with d_t, is greater than the smallest $(1-\alpha)$ fraction of d_j's which can be expressed as

$$\frac{\sum_{x_j \in S_2} \mathbf{1}\{d_t > d_j\}}{N_2} > (1-\alpha)$$

where $\mathbf{1}$ is the characteristic function.

In short, $\mathbf{x}_t$ is considered an outlier if it falls out of the acceptance region A which means that the NN statistics d_t takes an higher value compared to non outliers. Moreover if the observed data stream persistently falls outside the acceptance region it will be considered an anomaly.

Algorithm 1. GEM-Based Real-Time Nonparametric Anomaly Detection

Offline Phase

1: Uniformly randomly partition the nominal dataset $\mathcal{X}$ into two subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ with sizes N_1 and N_2, respectively.
2: **for** $j : \mathbf{x}_j \in \mathcal{S}_2$ **do**
3: Search for the kNNs of $\mathbf{x}_j$ among the set $\mathcal{S}_1$.
4: Compute d_j using (6).
5: **end for**
6: Sort $\{d_j : \mathbf{x}_j \in \mathcal{S}_2\}$ in ascending order.

Online Detection Phase

1: Initialization: $t \leftarrow 0$, $g_0 \leftarrow 0$.
2: **while** $g_t < h$ **do**
3: $t \leftarrow t + 1$.
4: Obtain the new data point $\mathbf{x}_t$.
5: Search for the kNNs of $\mathbf{x}_t$ among the set $\mathcal{S}_1$ and compute d_t using (6).
6: $\hat{p}_t = \frac{1}{N_2} \sum_{\mathbf{x}_j \in \mathcal{S}_2} \mathbb{1}\{d_j > d_t\}$.
7: $\hat{s}_t = \log(\alpha / \hat{p}_t)$.
8: $g_t \leftarrow \max\{0, g_{t-1} + \hat{s}_t\}$.
9: **end while**
10: Declare an anomaly and stop the procedure.

Figure 2: Pseudocode of GEM based method from [2]

2.2 Principal Component Analysis

The second summary statistic is computed by using Principal Component Analysis. This particular algorithm was proposed by considering the curse of dimensionality which frequently affects real case datasets. In fact high dimensional dataset frequently exhibit a sparse structure and a lower dimension representation is feasible. In this case the model is effectively:

$$\mathbf{x}_t = \mathbf{y}_t + \mathbf{r}_t$$

where $\mathbf{y}_t$ is the submanifold representation and $\mathbf{r}_t$ is the residual term, mostly noise. The magnitude of the residual term can be used to characterize anomalous data because $\|\mathbf{r}_t\|$ is higher for anomalies compared to nominal data. The figure 3 shows the steps in order to compute the summary statistics using a PCA based approach.

Given $\mathbf{X}$ we can take two subsets S_1 and S_2 with sizes N_1 and N_2 both less than or equal to $|\mathbf{X}|$. Using S_1 we can determine a submanifold where the data is embedded and using S_2 we can compute the residual term $R = \{\|\mathbf{r}_j\| : \mathbf{x}_j \in S_2\}$.

First the sample mean $\bar{\mathbf{x}}$ of S_1 and the sample data covariance matrix $\mathbf{Q}$ are computed.

Then the eigenvalues $\{\lambda_j : j = 1, 2, .., p\}$ and the eigenvectors $\{\mathbf{v}_j : j = 1, 2, ..., p\}$ of $\mathbf{Q}$ are obtained.

We can determine the dimensionality of the submanifold r based on the desired fraction of data variance retained given by

$$\gamma \triangleq \frac{\sum_{j=1}^{r} \lambda_j}{\sum_{j=1}^{p} \lambda_j} \leq 1$$

where the r-dimensional principal subspace is spanned by the orthonormal eigenvectors $\mathbf{V} \triangleq \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_j\}$ corresponding to the r largest eigenvalues of $\mathbf{Q}$.

The representation of $\mathbf{x}_t$ in the linear submanifold can be determined as

$$\mathbf{y}_t = \bar{\mathbf{x}} + \sum_{j=1}^{r} \mathbf{v}_j \mathbf{v}_j^T (\mathbf{x}_t - \bar{\mathbf{x}}) = \bar{\mathbf{x}} + \mathbf{V}\mathbf{V}^T (\mathbf{x}_t - \bar{\mathbf{x}})$$

Then the residual term is easily computed by

$$\mathbf{r}_t = \mathbf{x}_t - \mathbf{y}_t = (\mathbf{I}_p - \mathbf{V}\mathbf{V}^T)(\mathbf{x}_t - \bar{\mathbf{x}})$$

. The summary statistic is $\|\mathbf{r}_t\|_2$.

Algorithm 2. PCA-Based Real-Time Nonparametric Anomaly Detection

Offline Phase

1: Choose subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ of $\mathcal{X}$ with sizes N_1 and N_2, respectively.
2: Compute $\bar{\mathbf{x}}$ and $\mathbf{Q}$ over $\mathcal{S}_1$ using (8) and (9), respectively.
3: Compute the eigenvalues $\{\lambda_j : j = 1, 2, \ldots, p\}$ and the eigenvectors $\{\mathbf{v}_j : j = 1, 2, \ldots, p\}$ of $\mathbf{Q}$.
4: Based on a desired level of γ (see (10)), determine r and form the matrix $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_r]$.
5: **for** $j : \mathbf{x}_j \in \mathcal{S}_2$ **do**
6: $\mathbf{r}_j = (\mathbf{I}_p - \mathbf{V}\mathbf{V}^{\mathrm{T}})(\mathbf{x}_j - \bar{\mathbf{x}})$.
7: Compute $\|\mathbf{r}_j\|_2$.
8: **end for**
9: Sort $\{\|\mathbf{r}_j\|_2 : \mathbf{x}_j \in \mathcal{S}_2\}$ in ascending order.

Online Detection Phase

1: Initialization: $t \leftarrow 0$, $g_0 \leftarrow 0$.
2: **while** $g_t < h$ **do**
3: $t \leftarrow t + 1$.
4: Obtain the new data point $\mathbf{x}_t$.
5: $\mathbf{r}_t = (\mathbf{I}_p - \mathbf{V}\mathbf{V}^{\mathrm{T}})(\mathbf{x}_t - \bar{\mathbf{x}})$ and compute $\|\mathbf{r}_t\|_2$.
6: $\hat{p}_t = \frac{1}{N_2} \sum_{\mathbf{x}_j \in \mathcal{S}_2} \mathbb{1}\{\|\mathbf{r}_j\|_2 > \|\mathbf{r}_t\|_2\}$.
7: $\hat{s}_t = \log(\alpha / \hat{p}_t)$.
8: $g_t \leftarrow \max\{0, g_{t-1} + \hat{s}_t\}$.
9: **end while**
10: Declare an anomaly and stop the procedure.

Figure 3: Pseudocode of PCA based method from [2]

2.3 Online Phase

The summary statistics that are considered are $\mathbf{d}_t$ for the GEM based technique and $\|\mathbf{r}_t\|_2$ for the PCA technique. Outliers have statistics that are higher value compared to nominal data points. This means that outliers correspond to the right tail events considering the nominal pdfs of the summary statistics.

If $\hat{p}_t < \alpha$ then we can claim that $\mathbf{x}_t$ is an outlier with respect to the level of α. Let:

$$s_t \triangleq \log\left(\frac{\alpha}{\hat{p}_t}\right)$$

Then we have $s_t > 0$ if $\mathbf{x}_t$ is an outlier and $s_t \leq 0$ if it is a non outlier. The model free algorithm based on CUSUM therefore is:

$$\Gamma = \inf\{t : g_t \geq h\}$$

$$g_t = \max\{0, g_{t-1} + \hat{s}_t\}$$

$$g_0 = 0$$

3 Hyper-Parameter Optimization

We decided to optimize the two hyper-parameters *alpha* and h of our models with *Nevergrad*[4]. We utilized *NGOpt* [4] as the Black Box Optimization algorithm with the objective function (1) where a is the number of samples identified as anomalies in the training set, N_1 is the amount of samples in the training set, n is the amount of samples identified as normal in the anomalous set, N_2 the amount of samples in the anomalous set, $w_1 = 0.4$, $w_2 = 0.6$.

$$l = w_1 \frac{a}{N_1} + w_2 \frac{n}{N_2} \tag{1}$$

We chose this specific function as a way to limit the possibility of the optimizer to simply pick very extreme parameters. The tests we run are the following.

Mean Change We trained the models on a 100 dimensional $\mathcal{N} \sim (0, 1)$ distribution of 1.000 samples and tested it against a 100 dimensional $\mathcal{N} \sim (1, 1)$ distribution of 10.000 samples.

Variance Change We trained the models on a 100 dimensional $\mathcal{N} \sim (0, 1)$ distribution of 1.000 samples and tested it against a 100 dimensional $\mathcal{N} \sim (0, 2)$ distribution of 10.000 samples.

Subset We trained the models on a 100 dimensional $\mathcal{N} \sim (0, 1)$ distribution of 1.000 samples and tested it against a 100 dimensional $\mathcal{N} \sim (0.5, 0.5)$ distribution of 10.000 samples.

Human Activity We trained the models on a dataset representing embeddings of human activities [5] and tested it against anomalous data from the same dataset.

3.1 Geometric Entropy Minimization

In the table 1 there can be seen the proposed hyper-parameters and the resulting best achieved loss for the three test cases. In the images 4, 5, 6 and 7 it can be observed the graph with the values assumed through time by α, h and the *loss*. It is interesting to see how in 6 with the mean shift test case the algorithm reaches a plateau that it is unable to pierce.

It is worth noting how the GEM-based algorithm generally took significantly longer than the PCA-based algorithm, we believe it is due to GEM's computation of the k best neighbors amongst a partition of the training dataset. Said partition is currently set to be 15% of the original dataset, by reducing it we believe the algorithm's speed will increase at the cost of its accuracy.

Test Case	α	h	Loss	Time
Subset	0.036	5.290	0.6	1937s
Mean Change	0.019	8.124	0.6	1134s
Variance Change	0.535	1.001	0.206	1200s
Human Activity	0.0407	1.150	0.007	63931s

Table 1: Optimization results for GEM

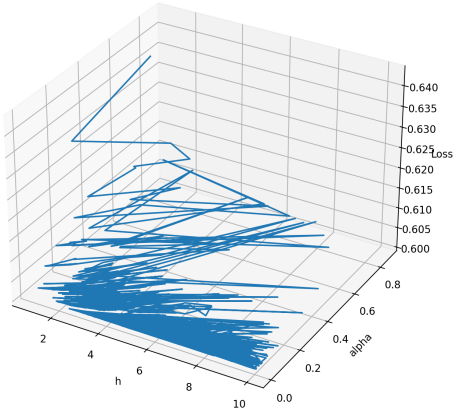

Figure 4: BBO of the GEM model with the subset test

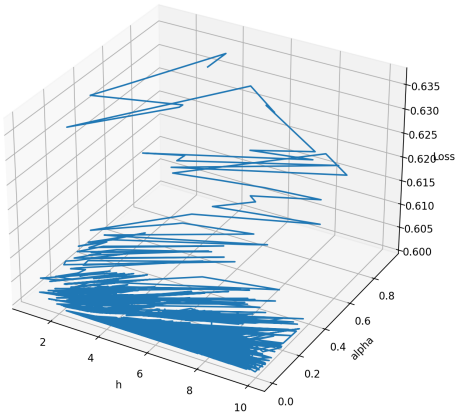

Figure 6: BBO of the GEM model with shifted mean test

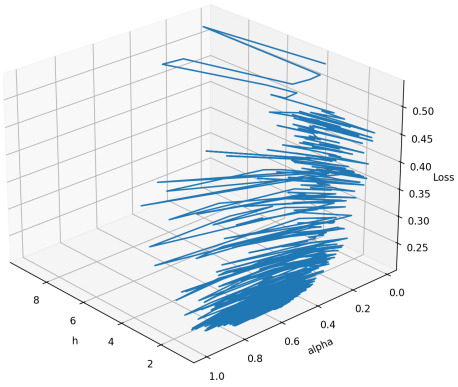

Figure 5: BBO of the GEM model with changed variance test

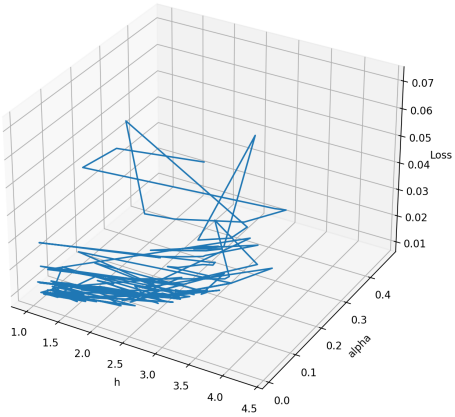

Figure 7: BBO of the GEM model with the human activity test

3.2 Principal Component Analysis

In the table 2 there can be seen the suggested hyper-parameters and the resulting best achieved loss for the three test cases. The resulting values are promising since the behaviour is fairly similar to the GEM-based model. We can see how the objective function returns a higher loss in the *Human Activity* test case compared to the other model. The following graphs clearly represent the descent of the optimizer as the results get closer to the local minimum and the behaviour is again analogous to the optimizing activity for the GEM-based model.

It is worth noting how for these test cases the PCA-based algorithm is significantly faster then the GEM-based one, at the cost of some accuracy with respect of the loss function we defined. More comparisons between the models will be discussed in 4. An important observation is that the loss function results in a plateau when either algorithm is tested against the *Mean Change* test and the *Subset* test.

Test Case	$\boldsymbol{\alpha}$	$\boldsymbol{h}$	**Loss**	**Time**
Subset	0.024	6.934	0.6	221s
Mean Change	0.08	8.758	0.6	266s
Variance Change	0.611	1.004	0.297	265s
Human Activity	0.887	1.127	0.515	18259s

Table 2: Optimization results for PCA

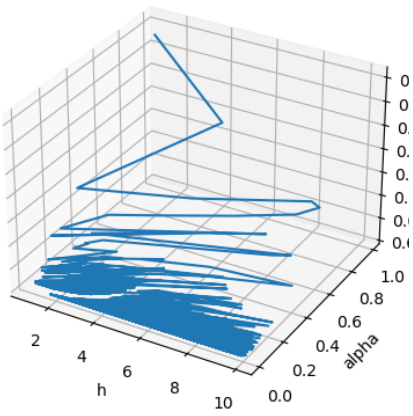

Figure 8: BBO of the PCA model with the subset test

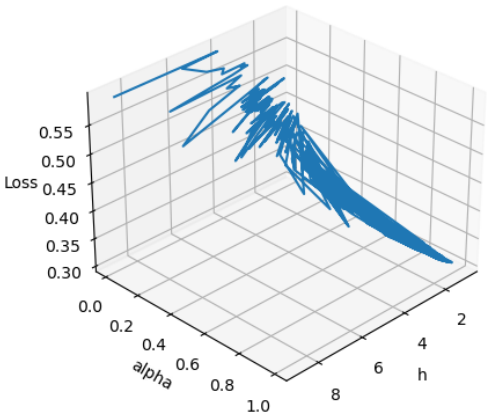

Figure 9: BBO of the PCA model with changed variance test

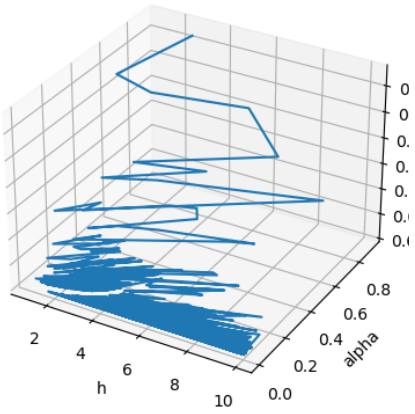

Figure 10: BBO of the PCA model with shifted mean test

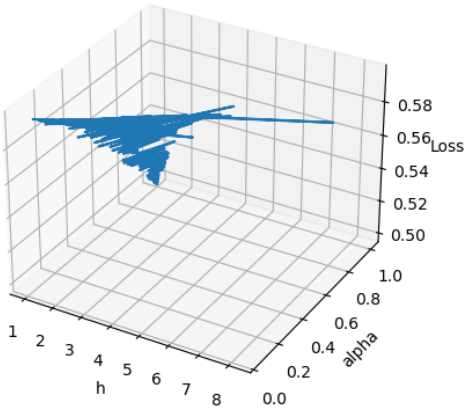

Figure 11: BBO of the PCA model with the human activity test

4 Testing

After the optimization of α and h we performed a number of tests testing to compare the performance of the GEM-based algorithm and the PCA-based algorithm.

We gave both models the following hyper-parameters

- $\alpha = 0.4$
- $h = 5.0$

and tested them on a number of different cases by recording and comparing the respective anomaly find rates

$$r = \frac{a}{|\mathcal{T}|}$$

Where a is the number of anomalies found and $|\mathcal{T}|$ is the cardinality of the testing dataset.

4.1 Increasing Dimension

In this test we plot both models' accuracy as the data dimension (p) increases. We tested the models in two ways each

i.i.d. the training set as a multivariate of independent identically distributed entries sampled from $\mathcal{N} \sim (0, 1)$ and the test set as a multivariate sampled from $\mathcal{N} \sim (0, 2)$

Random covariance matrix the training set was a multivariate with mean 0 and a covariance matrix resulted from randomly perturbing the identity matrix and the test set consisted of data points from $\mathcal{N} \sim (0, 1)$

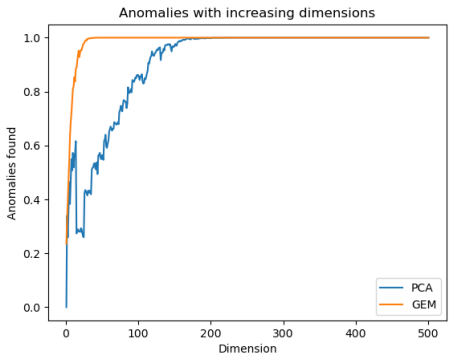

Figure 12: i.i.d. experiment

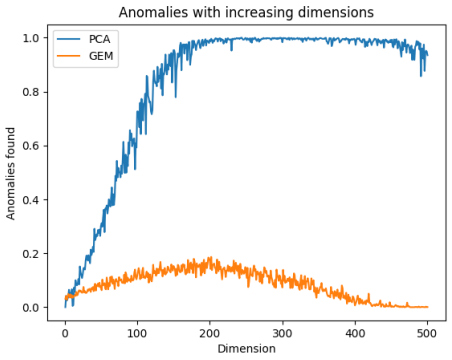

Figure 13: Random covariance experiment

As expected with the *random covariance* test the PCA-based model performed exponentially better than the GEM-based one since it manages to learn the relationships between the dimensions during the training phase while GEM only saw the samples

as i.i.d. so when tested against truly i.i.d. samples with the same mean and variance GEM was not able to recognize the anomaly.

Using a multivariate with i.i.d. dimensions PCA was unable to extract meaningful correlations between them resulting in a performance similar yet slightly worse than the GEM-based algorithm.

4.2 Increasing Offline Samples

In this test we plot the model's accuracy as the training dataset grows against a fixed test dataset. We tested the models in two ways:

i.i.d. the training set as a multivariate of independent identically distributed entries sampled from $\mathcal{N} \sim (0, 1)$ and the test set as a multivariate sampled from $\mathcal{N} \sim (0, 2)$

Random covariance matrix the training set was a multivariate with mean 0 and a covariance matrix resulted from randomly perturbing the identity matrix and the test set consisted of data points from $\mathcal{N} \sim (0, 1)$

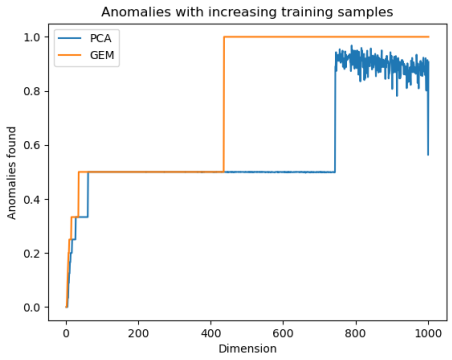

Figure 14: i.i.d. experiment

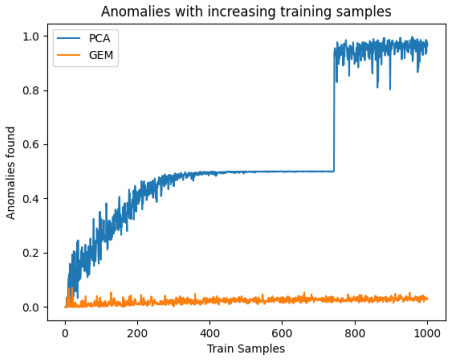

Figure 15: Random covariance experiment

Again, PCA performed better in the *random covariance* test and worse in the *i.i.d.* test. It is interesting to see how in the random covariance test the PCA-based algorithm reaches a plateau at 50% accuracy for some time and then after the training samples reaches a certain number it skyrockets to almost 100%. Also interesting to note how, in the *i.i.d.* test, the increase in training samples resulted in a series of constant values for the accuracy, with sudden steps, this is due to the h hyper-parameter as we discuss later on 4.4.

4.3 Increasing Variance and Perturbation

In the first test we compare the accuracy of both algorithms with respect to a dataset in which the variance gradually increases. Let the training samples be from a normal multivariate random distribution with null mean and identity covariance matrix $G_1 =$

I_p, we used as a testing dataset a normal multivariate distribution with $\mu = 0$ and covariance matrix $G_2 = kG_1$ with an increasing $k = 1, \ldots, 10$.

In the latter test we aim to show both models' accuracy as the test dataset's covariance matrix is gradually perturbed from the one they saw in the offline phase. We again used a multivariate normal random distribution with mean 0 and covariance matrix $G_1 = I_p$ for our training while the testing used samples generated from a multivariate normal random distribution with null mean but covariance matrix $G_2 = I + r$ where r is noise. r was gradually increased by a factor of 0.1 for each iteration.

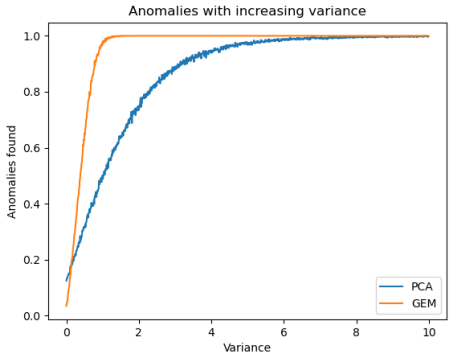

Figure 16: PCA and GEM performance with linear increase

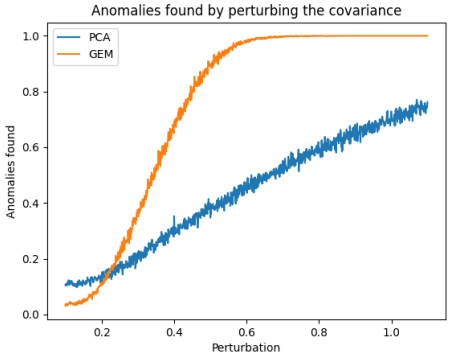

Figure 17: PCA and GEM performance with increasing noise in the covariance matrix

These test showed how a training set made of i.i.d. random variable always results in a worse performance for the PCA-based algorithm, as expected. The GEM-based algorithm, as shown in figure 16, is more sensible to changes in the variance of the samples. In both cases the PCA approach is slower to converge compared to GEM. It is very visible in figure 17 how the linear behaviour doesn't reach convergence by the end of the testing cycle.

4.4 Increasing Threshold Value

In these tests we wanted to see how the anomaly finding rate changes based as the threshold hyper-parameter h increases. The initial threshold was set at 0.1 and the increase is by 0.1 per iteration with the total number of iterations set to 1000. The two experiments differ by the datasets used just as in the previous chapters.

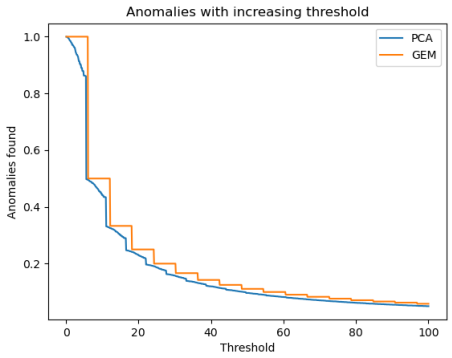

Figure 18: i.i.d. experiment

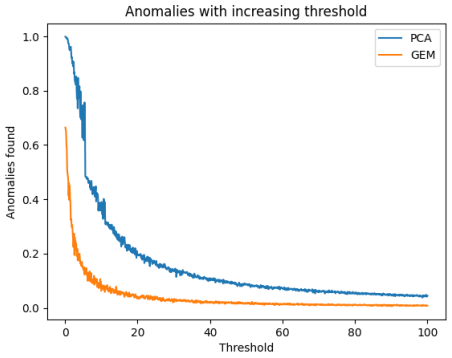

Figure 19: Random covariance experiment

As seen by the results the models behaved as anticipated. The anomaly rate rightfully decreases towards a zero value. It's interesting to see that GEM is far less sensitive to threshold changes compared to PCA in the *i.i.d.* experiment, as shown in figure 18.

4.5 Time and Delay comparison

The last tests were related to the time metrics of the methodologies with a direct comparison of the performances of the two models.

Time Comparison with Increasing Dimensionality The average sample analysis time of the two algorithms were also compared as the dimensionality of the sample increases from 1 up to 500. The dataset used for the training phase used a simple dataset of 1000 samples of a $\mathcal{N} \sim (0, 1)$ while the anomalous data consisted of 2000 samples taken from a multivariate $\mathcal{N} \sim (0, 2)$.

Alarm delay with Increasing Variance In this test we compare the two algorithms in terms of the delay until the first anomaly in the dataset is found, with increasing variance of the test dataset.

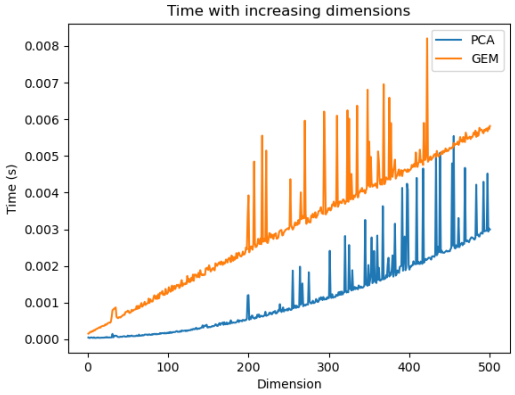

Figure 20: Time Comparison

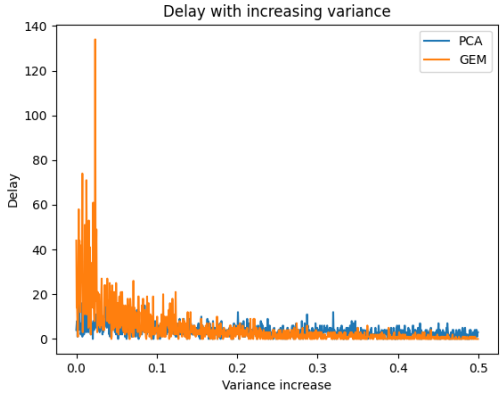

Figure 21: Alarm delay

In figure 20 we show the change in the average time to analyze a sample with both methods and we note how the two algorithm exhibit different behaviours. The GEM-based algorithm seems to be increasing linearly with the dimension while the PCA-based one is slowly growing exponentially. Although PCA is noticeably faster than GEM, both algorithms show very erratic average time with higher dimensions and frequent peaks can be seen in both cases.

With respect to the delay required to launch an alarm as it can be seen in figure 21, as anticipated the delay is greatly reduced after the change in the variance reaches 0.1. After that value the delay stays lower than 10 samples, probably due to the high values added during the CUSUM phase since the samples would greatly differ from the ones seen during the offline phase.

4.6 Other Considerations

During our tests we noticed that none of the models take into account the history of past samples. This results in a much lower anomaly detection rate when the algorithms receive samples taken from *subsets* of the distributions seen during the offline phase *e.g.* Training the models on a multivariate sampled from a $\mathcal{N} \sim (\mu, \sigma^2)$ distribution and testing them against a multivariate sampled from a $\mathcal{N} \sim (\mu/2, \sigma^2/2)$ distribution.

We hypothesize that the origin of this behaviour is due to both models looking at every observed sample individually and the way the tail probability is asserted. If the probability is below α the model will consider the sample a possible anomaly and after observing a number of possible anomalies (or a big one) it will launch the alarm, depending on the h parameter.

With this in mind an attacker could simply return a series of samples all equal to the most commonly observed sample and none of the models will see the sequence as anomalous *e.g.* Training the models on samples from a multivariate $\mathcal{N} \sim (\mu, \sigma^2)$ distribution and testing against a series of μ.

5 Conclusions

We implemented an online anomaly detection system as proposed by [2] using `C++` and `Python`. We subsequently optimized the models' two main parameters α and h in order to infer the most suitable values on a number of different situations. Ulterior tests comparing the two models allowed us to observe the sensitivity and usefulness of the models in different test cases.

6 Future Developments

An important aspect to be considered in the algorithm is the unrealistic assumption that the data is comprised of independent and identically distributed data points (i.i.d.), to overcome this limitation it might be necessary to perform a detrending operation on the data in order to obtain a series of i.i.d. as assumed by the model.

Another improvement to be made is to fix the lack of reasoning on the history of the samples saw during training, as shown in 4.6.

There are also more hyper-parameters than α and h to test and optimize, GEM also requires a k for the kNN phase and a fraction for the partition of the training set into the subsets S_1 and S_2, finally PCA requires a γ for the desired fraction of variance to be maintained in the algorithm.

References

[1] Srikanth Thudumu, Philip Branch, Jiong Jin, and Jugdutt Singh. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7, 07 2020.

[2] Mehmet Necip Kurt, Yasin Ylmaz, and Xiaodong Wang. Real-time nonparametric anomaly detection in high-dimensional settings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2463–2479, July 2021.

[3] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[4] J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. `https://GitHub.com/FacebookResearch/Nevergrad`, 2018.

[5] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà Monsonís, Francesc Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171, 08 2015.