# Real-time Anomaly Detection for Multivariate Data Streams

Kenneth Odoh

**Abstract**
We present a real-time multivariate anomaly detection algorithm for data streams based on the Probabilistic Exponentially Weighted Moving Average (PEWMA). Our formulation is resilient to (abrupt transient, abrupt distributional, and gradual distributional) shifts in the data. This novel anomaly detection routines utilize an incremental online algorithm to handle streams. Furthermore, our proposed anomaly detection algorithm works in an unsupervised manner, eliminating the need for labelled examples. Our algorithm performs well and is resilient to concept drift.

**Keywords**
signal processing, online learning

## 1. Introduction

Anomaly detection is the task of classifying patterns that depict abnormal behaviour. Outliers can arise due to (human/equipment) errors, faulty systems, and others. Anomaly detection is well-suited to unbalanced data scenarios, where the ideal scenario is to predict minority class behaviour. There are numerous applications for detecting loan default, fraud detection, and network intrusion detection. An anomaly detection algorithm can work in Unsupervised, Supervised, or hybrid modes.

There are different types of anomalies described as follows.

- Point Anomaly: the algorithm identifies a single instance as an anomaly concerning the entire data set.
- Contextual Anomaly: a data instance can be anomalous based on the context (attributes and position in the stream) and proximity of the chosen anomaly. This anomaly type is ideal for multivariate data, e.g. in the snapshot reading of a machine, an attribute of a single data point may seem abnormal but can be normal behaviour based on consideration of the entire data.
- Collective Anomaly: the algorithm decides a set of data points that are anomalies as a group, but individually these data points exhibit normal behaviours.

Anomaly detection algorithms can operate in the following settings:

- Static: These algorithms work with static data. Every item is loaded into memory at the same time in order to perform computations.
- Online: These algorithms work in real-time data streams. Items are incrementally loaded into memory and processed in chunks.
- Static + Online: The model may operate in two stages, as initial parameters get estimated in the static setting. The parameters are incrementally updated as more data arrives. Our work is of this type.

PEWMA was introduced in the work [1] for online anomaly detection on univariate time series. Drawing inspiration from their work, we have provided extensions to support real-time anomaly detection of a multivariate data stream.
[1]

## 2. Background

An anomaly detection algorithm can identify outliers in a variety of signal changes in time-dependent data. Signal changes are common in time-dependent data. They include abrupt transient shift, abrupt distributional shift, and gradual distributional shift [1] labelled as "A", "B", and "C" in Figure 1 respectively.

Online algorithms are useful for real-time applications, as they operate incrementally on data streams. These algorithms incrementally receive input and decide based on an updated parameter that captures the current state of the data stream. This philosophy contrasts with offline algorithms that assume the entire data is available in memory. Offline algorithms require data must fit in

*CIKM'22: Applied Machine Learning Methods for Time Series Forecasting Workshop, October 21, 2022, Georgia, GA*
kenneth.odoh@gmail.com (K. Odoh)

[1]Source code: https://github.com/kenluck2001/anomalyMulti, Blog: https://kenluck2001.github.io/blog_post/real-time_anomaly_detection_for_multivariate_data_stream.html

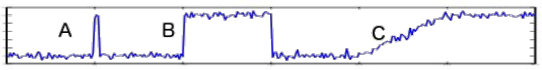

**Figure 1:** Different types of signal changes: abrupt transient, abrupt distributional shift, and gradual distributional shift [1] (from left to right).

memory. The online algorithm should be time and space efficient.

An anomaly detection algorithm can work in modes such as diagnosis and accommodation [2]. Firstly, the diagnosis method finds the outlier in the data and removes it from the data sample to avoid skewing the distribution. This method is applicable when the distribution of expected behaviours is known. The outliers get excluded when the estimation of the parameters of the distribution [2]. Secondly, the accommodation method finds outliers in the data and estimates the statistical model parameters. The accommodation method is suitable for data streams that account for concept drift [3].

Muth [4] laid the foundation for exponential smoothing by showing that it provides optimal estimates using random walks with some noise. Further works were done to provide a statistical framework for exponential smoothing, leading to the development of linear models [5, 6]. Unfortunately, this applies to nonlinear models. Yule postulated a formulation of time series as a realization of a stochastic process [7]. Exponential Weighted Moving Average (EWMA) is ideal for keeping running moments in the data stream. EWMA is a smoothing technique that adds a forgetting parameter to modify the influence of a recent item in the data stream. This is shown in Equation 1. This smoothing causes volatility in abrupt transient changes and is unfit for distribution shifts.

$$\mu_{t+1} = \alpha\mu_{t-1} + (1-\alpha)X_t \tag{1}$$

EWMA limitations motivated the discovery of the Probabilistic Exponentially Weighted Moving Average (PEWMA). PEWMA [1] improves on EMWA by adding a parameter that includes the probability of the data in the model, as shown in Equation 2. PEWMA works for every shift including abrupt transient shift, abrupt distributional shift, and gradual distributional shift respectively. PEWMA and EWMA use a damped window [8].

$$\mu_{t+1} = \alpha(1-\beta P_t)\mu_{t-1} + (1-\alpha(1-\beta P_t))X_t \tag{2}$$

## 3. Method

Our formulation provides an implementation of the online covariance matrix in Subsection 3.2, alongside an online inverse covariance matrix based on Sherman–Morrison formula in Subsection 3.3, and PEWMA in Subsection 3.1.

Our implementation of the online covariance matrix builds on work [9]. We simplify the algorithm by ignoring evolutionary computation details in the paper. Our work adapts evolution as described in the paper [9] as a transition from one generation to the next; it is equivalent to moving from one state to another state. This is analogous to how online algorithms work with dynamic changes as new data enters the stream.

### 3.1. Probabilistic Exponentially Weighted Moving Average (PEWMA)

PEWMA [1] algorithm works in accommodation mode. The routine shown in Algorithm [1] allows for concept drift [3], which occurs in data streams, by updating the set of parameters that convey the stream state.

**Algorithm 1** Probabilistic Exponential Weighted Moving Average [1]

```
Require: $X_t, \hat{X_t}, \hat{\alpha_t}, T, t$
Ensure: $P_t, \hat{X_{t+1}}, \hat{\alpha_{t+1}}$
  //incremental Z score
  $Z_t \leftarrow \frac{X_t - \hat{X_t}}{\hat{\alpha_t}}$
  //probability density function
  $P_t \leftarrow \frac{Z_t}{\sqrt{2\pi}} e^{\frac{Z_t}{2}}$
  if $t < T$ then
  //increment standard deviation (training phase)
    $\alpha_t \leftarrow 1 - 1/t$
  else
  //increment standard deviation
    $\alpha_t \leftarrow (1 - \beta P_t)\alpha$
  end if
  //moving average
  $s_1 \leftarrow \alpha_t s_1 + (1 - \alpha_t)X_t$
  $s_2 \leftarrow \alpha_t s_2 + (1 - \alpha_t)X_t^2$
  //incremental mean
  $\hat{X_{t+1}} \leftarrow s_1$
  //incremental standard deviation
  $\hat{\alpha_{t+1}} \leftarrow \sqrt{s_2 - s_1^2}$
```

The parameters of the anomaly detection algorithm consist of $X_t$ the current data, $\mu_t$ the mean of the data, $\hat{X_t}$ is the mean of the data, $\hat{\alpha_t}$ the current standard deviation, $P_t$ the probability density function, $\hat{X_{t+1}}$ the mean of the next data (incremental aggregate), $\hat{\alpha_{t+1}}$ the next standard deviation (incremental aggregates), $T$ the data size, and $t$ a point in $T$. Initialize the process by setting the initial data for training the model $s_1 = X_1$ and $s_2 = X_1^2$.

Our work used the following parameters $\alpha$, $\beta$, and $\tau$ as seen in Subsection 3.2. These anomaly thresholds are chosen based on the criteria that outliers are $\geq 3$ times the standard deviation in normally distributed data.

## 3.2. Online Covariance matrix

1. Estimate covariance matrix for initial data, $X \in R^{n \times m}$.
   Initial covariance matrix, $C$ where $C \in R^{n \times m}$, $n$ is the number of samples, $m$ is the number of dimensions as shown in Equation 3.
   $$C = X * X^T \tag{3}$$
2. Perform Cholesky factorization on the initial covariance matrix (positive-definite), $C$ as shown in Equation 4.
   $$C_t = A_t * {A_t}^T \tag{4}$$
3. The updated covariance in the presence of new data is equivalent to the weighted average of the past covariance without the updated data and the covariance matrix of the transformed input, as shown in Equation 5.
   $$C_{t+1} = \alpha C_t + \beta v_t * {v_t}^T \tag{5}$$
   Where $v_t = A_t * z_t$ and $z_t \in R^m$ is understood in our implementation as the current data. $\alpha$ and $\beta$ are positive scalar values.
4. Increment the Cholesky factor of the covariance matrix as shown in Equation 6.
   $$A_{t+1} = \sqrt{\alpha} A_t + \frac{\sqrt{\alpha}}{\left\|z_t\right\|^2} \left( \sqrt{1 + \frac{\beta \left\|z_t\right\|^2}{\alpha}} - 1 \right) v_t * z_t \tag{6}$$
5. There are difficulties with setting $/alpha$ and $\beta$ respectively. $\alpha + \beta = 1$ as an explicit form of exponential moving average coefficients. The author chose to set the values of $\alpha$, $\beta$ using the data stream statistics, as shown in Equation 7. The parameters are set as $\alpha = {C_a}^2$, $\beta = 1 - {C_a}^2$ and $n$ is the size of the original data in the static settings.
   Where $C_a = \sqrt{1 - C_{cov}}$ and $C_{cov} = \frac{2}{n^2+6}$.
   $$A_{t+1} = C_a A_t + \frac{C_a}{\left\|z_t\right\|^2} \left( \sqrt{1 + \frac{(1 - {C_a}^2) \left\|z_t\right\|^2}{{C_a}^2}} - 1 \right) v_t * z_t \tag{7}$$

## 3.3. Online Inverse Covariance matrix

1. Estimate covariance matrix for initial data, $X \in R^{n \times m}$.
   The initial covariance matrix, $C$ where $C \in R^{n \times m}$, $n$ is the number of samples, $m$ is the number of dimensions as shown in Equation 8.
   $$C = X * X^T \tag{8}$$
   Inverse the covariance matrix, $C^{-1}$ as shown in Equation 9.
   $$C^{-1} = \left(X * X^T\right)^{-1} \tag{9}$$
2. Perform Cholesky factorization on the initial covariance matrix, $C$ as shown in Equation 10.
   $$C_t = A_t * {A_t}^T \tag{10}$$
3. Increment the Cholesky factor of the covariance matrix
   $$C_{t+1}^{-1} = (\alpha C_t + \beta v_t * {v_t}^T)^{-1} \tag{11}$$
   $$C_{t+1}^{-1} = \alpha^{-1} (C_t + \frac{\beta v_t * {v_t}^T}{\alpha})^{-1} \tag{12}$$
   Let us fix, $\hat{v}_t = \frac{\beta * v_t}{\alpha}$. The resulting simplification using Sherman–Morrison Formula reduces the expression to
   $$C_{t+1}^{-1} = \frac{1}{\alpha} \left( C_t^{-1} - \frac{{C_t}^{-1} \hat{v}_t {v_t}^T {C_t}^{-1}}{1 + (\hat{v}_t {C_t}^{-1} {v_t}^T)} \right) \tag{13}$$

## 3.4. Online Multivariate Anomaly Detection

The probability density function utilizes ideas from hypothesis testing for deciding on a threshold to set the confidence level. This threshold is used for determining the acceptance and rejection regions of the Gaussian distribution curve.

1. Use the covariance matrix, $C_{t+1}$ and inverse covariance matrix, ${C_{t+1}}^{-1}$.
2. We increment the mean vector, $\mu$ as new data arrives. It is possible to simplify the Covariance matrix, $C$, which captures a number of system dynamics. Let $n$ represent the current data count before updated data arrives. Also, $\hat{x}$: is the most recent data, $\mu_{t+1}$: moving average as shown in Equation 14.
   $$\mu_{t+1} = \frac{(n * \mu_t) + \hat{x}}{n + 1} \tag{14}$$

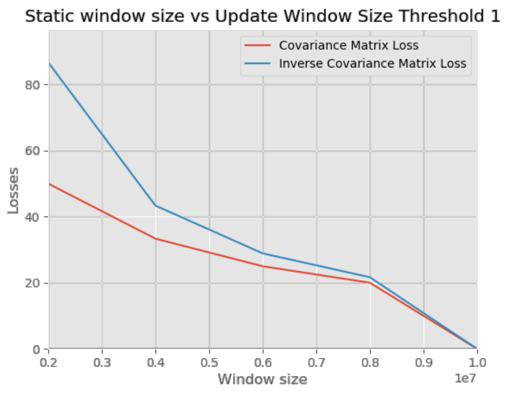

**Figure 2:** Compare the threshold of static vs incremental impact performance of anomaly detection (Version 1)

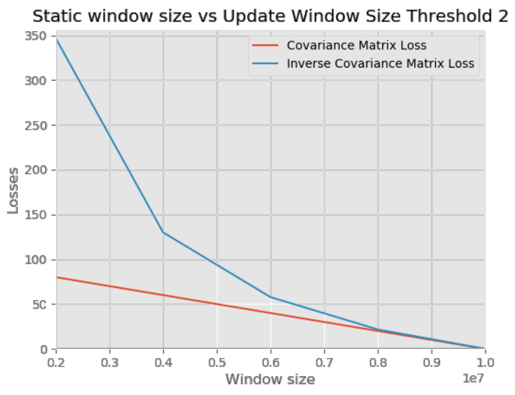

**Figure 3:** Compare the threshold of static vs incremental impact performance of anomaly detection (Version 2)

3. Set a threshold to determine the acceptance and rejection regions. Items in the acceptance region are considered normal behaviour as shown in Equation 15.

$$p(x) = \frac{1}{\sqrt{(2\pi)^m|C|}} \exp\left(-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)\right) \tag{15}$$

Where $\mu$ is mean vector, $C$ is the covariance matrix, $|C|$ is the determinant of $C$ matrix, $x \in R^m$ is data vector, and $m$ is the dimension of $x$ respectively.

## 4. Experiment

Furthermore, we have provided detailed experiments on the proposed algorithms in different realistic scenarios. However, we maintain the statistical framework provided by the work [1] with theoretical guarantees.

We have experimented to determine the usefulness of our algorithm by creating a simulation with 10000000 random vectors with dimensions of 15. The repeated trial shows that our algorithm is not sensitive to initialization seeds and matrix dimensions. This requirement was a deciding factor in the choice of the evaluation metric, as shown in Equation 16. We have provided more information on the metric in Section 5.

Our experiment will check the effect of varying the size of the initial static window versus the update window. This is shown in Subsection 4.1 and Subsection 4.2.

### 4.1. Experiment 1

We evaluated the trade-off between the static window and the update window. The experiment setup is as follows:

- Split the data into 5 segments train on 1st segment(static), update covariance on 2nd (online), compare with static covariance, and calculate the error.
- Train on 1, 2 segments (static), update covariance on 3rd (online), compare with static covariance and calculate the error.
- Train on 1, 2, 3 segments (static), update covariance on 4th (online), compare with static covariance and calculate the error.
- Train on 1, 2, 3, 4 segments (static), update covariance on 5th (online), compare with static covariance and calculate the error.

Figure 2 contains the experimental results.

### 4.2. Experiment 2

The experiment setup is as follows:

- Split the data into 5 segments.
- Train on 1st segment(static), update covariance on remaining segments (2,3,4,5) (online), compare with static covariance and calculate error on segments (2,3,4,5)
- Train on 1, 2 segments (static), update covariance on remaining segments (3,4,5) (online), compare with static covariance and calculate error on segments (3,4,5)
- Train on 1, 2, 3 segments (static), update covariance on remaining segments (4,5) (online), compare with static covariance and calculate error on segments (4,5)
- Train on 1, 2, 3, 4 segment(static), update covariance on remaining segments (5) (online), compare

with static covariance and calculate error on segments (5)

Figure 3 contains the experimental results.

## 5. Result Analysis

Our random matrices get flattened into vectors and used as input. The length of the flattened vector is applied as a normalization factor to make the loss metric agnostic of the matrix dimension. The loss function used in the evaluation is Absolute Average Deviation (AAD) because it gives a tighter bound on the error than mean squared error (MSE) or mean absolute deviation (MAD) as shown in Equation 16. We take the average of the residuals divided by the ground truth for every sample in our evaluation set. If the residual is close to zero, we contribute almost nothing to the measure. On the contrary, if the residual is high, we want to know the difference from the ground truth.

$$AAD = \sum_{i=1}^{n} \left| \frac{\hat{Y}_i - Y_i}{Y_i} \right| \tag{16}$$

Where $\hat{Y}_i$ is the predicted value, $Y_i$ is the ground truth, and $n$ is the length of the flattened matrix.

We can observe that building your model with more data in the init (static) phase leads to lower errors than having fewer data in the init phase and using more data for an update. The observation matches our intuition because when you operate online, you tend to use smaller storage space. However, there is still a performance trade-off compared to batch mode.

The error at the beginning of our training is significant in both charts. This insight shows that rather than performing the expensive operation of converting a covariance matrix to have positive definiteness, it is better to use random matrices that are positive definite. More data would help us achieve convergence as more data arrives.

The success of the experiments has given us confidence that our multivariate anomaly detection algorithm would have similar characteristics to the univariate case described in the work [1].

## 6. Conclusion

There is no generic anomaly detection that works for every task. The underlying assumption in this work is that the features in use capture relevant information about the underlying dynamics of the system. Our proposed implementation is an anomaly detection algorithm for handling multivariate streams, even with challenging shifts. In future work, we will extend support for non-stationary distributions in multivariate data streams.

## References

[1] K. M. Carter, W. W. Streilein, Probabilistic reasoning for streaming anomaly detection, in: Proceedings of the Statistical Signal Processing Workshop, 2012, pp. 377–380.

[2] V. Hodge, J. Austin, A survey of outlier detection methodologies, Artificial Intelligence Review 22 (2004) 85–126.

[3] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Transactions on Knowledge and Data Engineering 25 (2013) 2283–2301.

[4] J. F. Muth, Optimal properties of exponentially weighted forecasts, Journal of the American Statistical Association 55 (1960) 299–306.

[5] G. E. P. Box, G. M. Jenkins, Time Series Analysis: Forecasting and Control, Holden-Day, 1976.

[6] S. A. Roberts, A general class of holt-winters type forecasting models, Journal of Management Science 28 (1982) 808–820.

[7] G. U. Yule, On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers, Philosophical Transactions of the Royal Society of London 226 (1927) 267–298.

[8] M. Salehi, L. Rashidi, A Survey on Anomaly Detection in Evolving Data: With Application to Forest Fire Risk Prediction, SIGKDD Explorations Newsletter 20 (2018) 13–23.

[9] C. Igel, T. Suttorp, N. Hansen, A computational efficient covariance matrix update and a (1+1)-cma for evolution strategies, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Washington, USA, 2006.