Change Points Detection Algorithms and Eye Gaze Detection

INTRODUCTION

Time series analysis has become increasingly important in various diverse fields including medicine, aerospace, finance, business, meteorology, and entertainment. Time series data are sequences of measurements over time describing the behavior of systems. Time series data is a collection of historic data of systems throughout the interval. These behaviors can change over time due to many reasons from the internal or external surroundings of systems, various phase changes, unexpected outside interruptions, or any other causes that lead to abrupt changes in systems. These abrupt changes are called change points.

Changepoint detection (CPD) is the problem of finding abrupt changes in data when a property of the time series changes. Segmentation, edge detection, event detection, and anomaly detection are similar concepts that are occasionally applied as well as change point detection.

Changepoint detection is an important part of time-series analysis and several algorithms exist to find change points in time-series data.

There could be abrupt variations in a data sequence that could be characterized by any combination of parameters — For eg., Mean or variance or periodicity of the data may abruptly change after a particular time. We might be interested in catching the earliest time at which this variation occurs and this is referred to as

“Changepoint” detection. The applications of changepoint detection could be in varieties of domains — financial data, sensor data, biometrics, etc.,
There are two types of change-point detection 1) Online Change Point Detection 2) Offline Change Point Detection.
Online change point detection continuously data is added in time-series and online change point detection algorithms detect an abrupt change in point simultaneously while new data is being added. Offline change point detection algorithms work on historical time-series data to detect change points, no more data can be added when the algorithm is running on a dataset.

Eye Gaze Dataset

Eye gaze means how the human eye is looking at a timestamp. Eye gaze dataset that is being used in processing the following algorithms. Change-point detection algorithms are applied to eye gaze dataset and results, a set of analysis from algorithm application is used for adding further aspects to developed algorithms.
The eye gaze dataset contains X and Y coordinates of the eye gaze point at continuous timestamps. Thus it is time-series data of coordinates of the eye-gaze point at various timestamps.

Earlier eye-tracking and similar studies were only accessible through high-end hardware and costly sensors. But methods like time-series analysis and change point detection has helped in democratizing the process where data can be collected through simply web-cameras and processed with algorithms to get results.

LITERATURE REVIEW

Time-series analysis and subsequently change point detection has attracted a lot of researchers over the world. Many algorithms are developed to detect change points in both offline and online methods. These methods and algorithms are proven with their outcomes.
Following are some of the pioneered methods of online and offline change points detections:
Generalized Likelihood Ratio
Bayesian Online Changepoint Detection
The Subspace Methods for Online Changepoint Detection
Online Kernel Change Detection Algorithm
Changepoint Detection by Direct Density Ratio Estimation
Non-parametric maximum likelihood
Rank-based detection

Various evaluation metrics are used to validate the results obtained by the above algorithms and methods. Bayesian-based approach to change-point detection if far widely used in the various use cases. Being a continuous process with no fixed pre-known outcomes of change points there is always a scope of improving and innovating existing methods and also developing new ways to detect change points in time-series data. There can be different approaches to the problem of time-series analysis and change point detection and various available survey methods and survey research papers emphasize a comparison of different algorithms and can help in figuring out the method which suits best to your use case.

PRELIMINARY

In our approach to time-series analysis and change point detection, the approach consists of pre-analysis of time-series data to gather more insights into it and pre-process it to make it ready to use for applying algorithms. Time-series data is visualized to get a visual picture of it, in the further process it helps in statistical analysis for setting threshold values used in algorithms.

Algorithms are developed to detect change point detection in eye gaze dataset and compatibility of algorithms is set to extend their use for all other time-series dataset to detect change point detection. Developed algorithms are applied to eye gaze datasets to check their outcomes and verify results.

For evaluation of results given by algorithms, actual change points are needed for getting accuracy, precision, and other metrics. However, as an actual change point, data may not be available so the performance of the algorithm can be checked by splitting into a test dataset where you can check if detected points are actual change points and also comparing with outcomes given by other algorithms can be done.

Definitions of commonly used terms ahead:

Change points: abrupt changes in time-series data

Differences: Absolute differences between predicted and actual values of time-series data

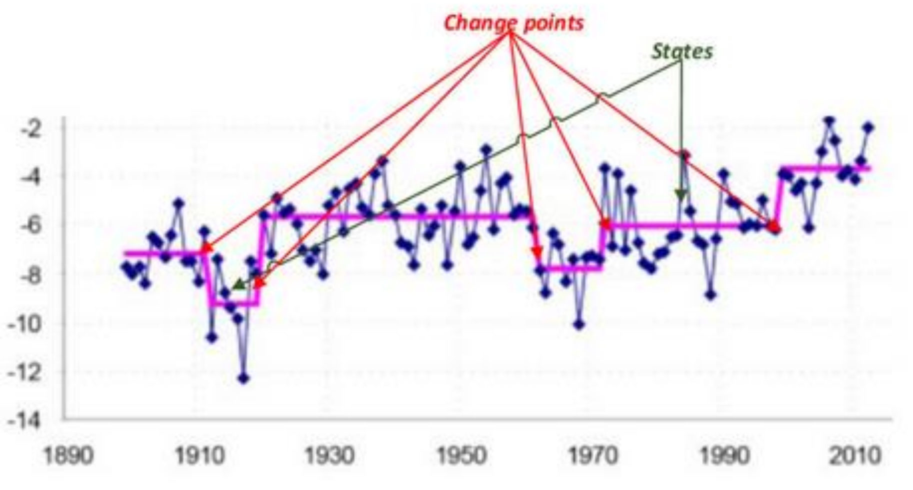
Threshold: Set threshold value to be used in the algorithm

Methodology

Here, we are focused on offline change point detection. In offline change point detection, we have all data with us before working and no other data is added while running the algorithm. Historical time-series data may have some abrupt variations in it, like at some point in time-span values in the dataset may have sudden changes in mean and variations and our algorithm aims at detecting those changes.

Algorithm 1:

Time-series data when plotted against its timestamps in the x-axis and values in the y-axis, it will come up as a continuous smooth curve with ups and downs and abrupt points in it. We have to convert these data into vertical and horizontal lines.



Sample time-series data with change points

X_{axis} →*Time in years*

Y_{axis} →*Temperature*

(It’s sample plot to show to change points in time-series data)

Steps:

1. Plot time-series data with time-stamps in the x-axis and values in the y-axis.
2. Set up Decision Tree Regressor Model to fit on time-series data
3. Predict values on the same time-series data by trained Decision Tree Regressor model.
4. Plot both predicted and original values on the same graph against timestamps on the x-axis.
5. Tune in hyperparameters of regressor to a good fit of predicted values against actual values.
6. Loop over stored predicted values, when predicted value changes mark that timestamp as a potential change point and store all timestamps where change occurs.

The first step is to split the data.
Starting from the root, the data is split on the feature that results in the largest **Information Gain** (**IG**)

Largest Information Gain :

To split the nodes at the most informative features, we need to define an objective function that we want to optimize via the tree learning algorithm. Here, our objective function is to maximize the information gain at each split :

$$IG(D_P, f) = I\left(D_p\right) - \left(\frac{N_{left}}{N_p} I\left(D_{left}\right) + \frac{N_{right}}{N_p} I(D_{right})\right)$$

f= the feature to perform the split
$D_p, D_{left}\ and\ D_{right}$ = datasets of the parent and child nodes

I = the impurity measure
N_{left} and N_{right}= number of samples in the child nodes and
N_p = total number of samples at the parent node

To use a decision tree for regression, we need an impurity metric that is suitable for continuous variables, so we define the impurity measure using the **weighted mean squared error (MSE)** of the children nodes :

$$MSE(T) = \frac{1}{N_t} \sum_{iD_t} \left(y^{(i)} - \widehat{y_t}\right)^2$$

Here, *N*$_t$ is the number of training samples at node *t*, *D*$_t$ is the training subset at node *t*,*y*$^{(i)}$ is the true target value, and$\hat{y}_t$ is the predicted target value :

$$\widehat{y_t} = \frac{1}{N_t} \sum_{iD_t} y^{(i)}$$

The above description depicts the setting of a decision tree model and applying it to time-series data and predicting using a trained model on the same data.

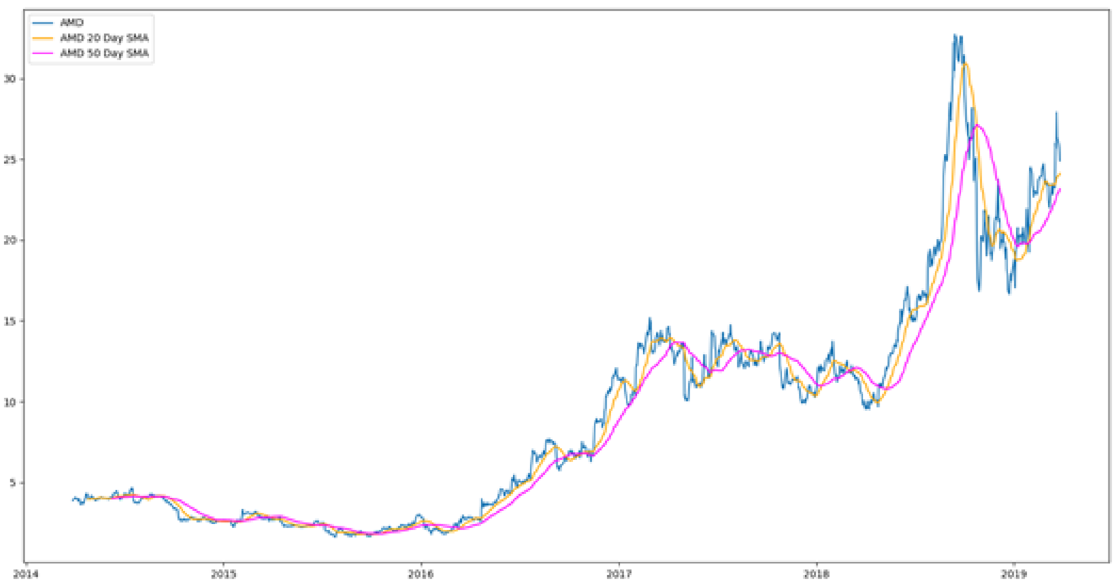
Predicted_Data(t) = Value predicted by Decision Tree Regressor at t timestamp
Data(t) = Original value at timestamp t
Count = no of change points

Predicted_Data(t+1) $\neq$ Predicted_Data(t):
t = timestamp of potential change point
Count = No of occurrences of such timestamps t

Algorithm 2:

As said above in the description of change points that these points are where mean and variance or periodicity of our time-series data changes abruptly. This algorithm hits at the point of mean changing abruptly and detects such abrupt changes in the mean of time-series data. These points are further classified as potential change points.

We continuously calculate the moving average of time-series data and store it. Further, we perform statistical analysis on calculated moving average data in terms of variance and other parameters to get allowed a change in average values and fix a threshold value of allowed change in average. If at some timestamp, change in average in time-series data is more than the set threshold value, then we classify that point and timestamp as a potential change points.



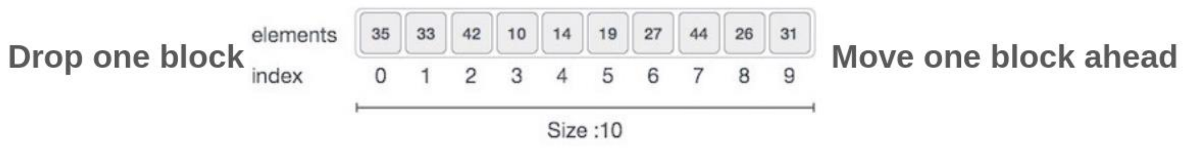
Sample moving average plotting of time-series data

X_{axis} $\rightarrow$ Company's turnover

Y_{axis} $\rightarrow$ Time in years

Steps:

1. Process time-series data and clean it out.
2. Fix ‘n’ following the time-series dataset where n is the size of the batch used to calculate average.
3. Get the average of first n entries in data.
4. Loop over data to calculate the average of n values leaving the first entry in batch and adding the next entry to batch. Store the average.
5. Get the description of calculated average values and get its mean, mode, 5%, 10%, etc.
6. Set a threshold value of the difference following average data so that difference above the threshold will be marked as a change point.
7. Loop over the average values and get the difference between consecutive average values, if the difference is above the threshold mark point as a potential change point.

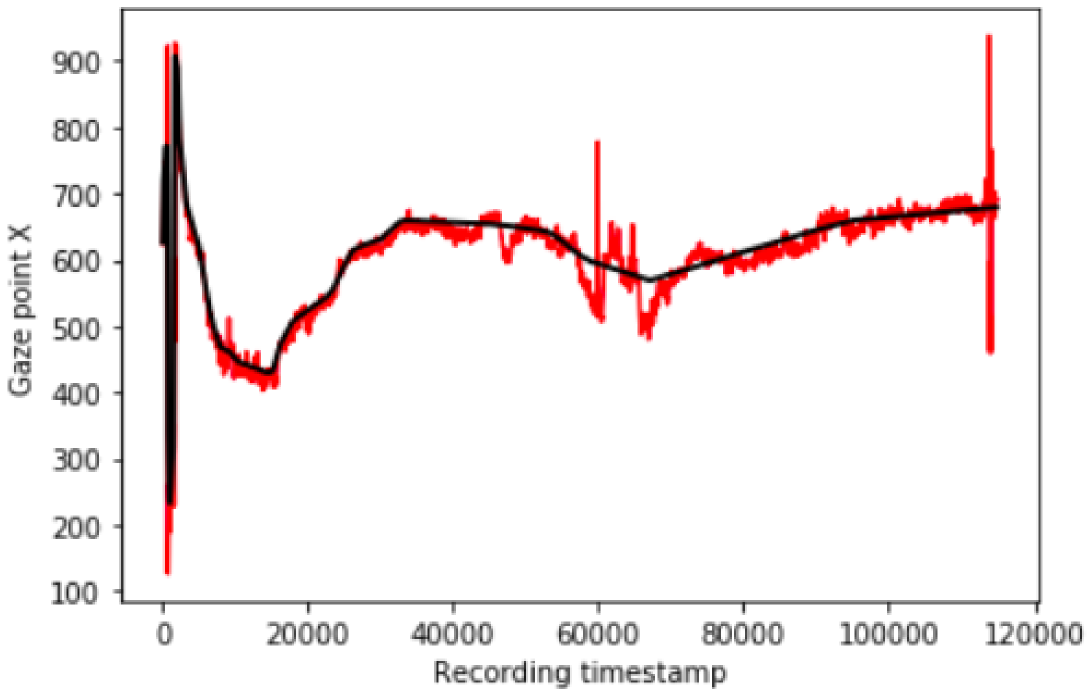


Representation of moving average method

Moving average method based algorithm for change point detection is supported by the idea that while taking the mean of a continuous set of values, we get balanced value(average value), and while taking the difference of moving average values if that balance gets disturbs leading to greater difference value, we can mark that point as a potential change point.

Algorithm 3:

When we train on some machine learning or deep learning algorithm on a dataset and predict using the trained model, it is usual practice to avoid overfitting on data, thus maintaining a balance between bias and variance. To achieve this, models are regularized and algorithms and designed to reduce such over fittings.
In this algorithm, we are above overfitting ideology in time-series data and abrupt changes are to be treated as points of high variance similar to overfitted points.
A deep learning model is trained on time-series data with properly tuned hyperparameters. While training this model, we want it fit on our time-series data such that it smoothens the abrupt point and does not overfit them. So deep learning model is regularized such that it avoids fitting on abrupt changes. The trained model is used to predict value on the same time-series data. In this way, given data is successfully mimicked using the deep learning model with no abrupt changes in it.
Later, we calculate the difference between predicted values and original values at their timestamps and store them. A certain threshold value for the difference is to be calculated, and the timestamps where the difference is above the threshold value are marked as potential change points.



$$X_{axis} \rightarrow Recording\ Timestamp$$
$$Y_{axis} \rightarrow Gaze\ point\ X$$

Steps:

1. Process the data and remove irregularities from the data.
2. Plot time-series data with time-stamps in the x-axis and values in the y-axis.
3. Created Artificial Neural Network model having 7 hidden layers to fit the data
4. Predict values on the same time-series data by trained Neural Network model.
5. Plot both predicted and original values on the same graph against timestamps on the x-axis. (above Image)
6. Calculate the differences between predicted values by deep learning model and actual values.

7. In accordance with the data, get the threshold value such that differences above the threshold value can be classified as potential change points.

Equation :

$$h(x,w) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots \ldots + w_k x^k,$$

we use a quadratic loss to measure how well our network performs:

$$L(w) = \sum_i \quad (h(x_i, w) - y_i)^2$$

This is the sum squared error of our network's predictions over our entire training set.

We will then use gradient descent on the loss's gradient $\nabla_w L(w)$ to minimize the overall error on the training data. We first derive the gradient of the loss with respect to a particular weight $w_{j\to k}$(which is just the weight of the edge connecting node j to node k [note that we treat inputs as "nodes," so there is a weight $w_{j\to k}$for each connection from the input to a first-layer node]) in the general case:

$$\frac{\partial}{\partial w_{j\to k}} L(w) = \frac{\partial}{\partial w_{j\to k}} \Sigma_i \quad (h(x_i, w) - y_i)^2$$

$$= \Sigma_i \quad \frac{\partial}{\partial w_{j\to k}} (h(x_i, w) - y_i)^2$$

$$= \Sigma_i \quad 2(h(x_i, w) - y_i) \frac{\partial}{\partial w_{j\to k}} h(x_i, w)$$

At this point, we must compute the gradient of our network function with respect to the weight in question $\frac{\partial}{\partial w_{j \to k}} h(x_i, w)$. In the case of a single layer network, this turns out to be simple.

Recall our simple two input network above. The network function is $h(x, w) = w_1 x_i^{(1)} + w_2 x_i^{(2)}$. The gradient with respect to w_1 is just x_1, and the gradient with respect to w_2 is just x_2, .We usually store all the weights of our network in a vector or a matrix, so the full gradient is:

$$\square_w L(w) = \left(\frac{\partial L(w)}{\partial w_1}, \frac{\partial L(w)}{\partial w_2} \right)$$
$$= \left(\sum_i \quad 2x_i^{(1)} h(x_i, w), \sum_i \quad 2x_i^{(2)} h(x_i, w) \right)$$

Using this, we then update our weights using standard gradient descent:

$$w = w - \eta \nabla_w L(w)$$

As with all gradient descent methods, care must be taken to select the "right" step size η, with "right" being application-dependent. After a set amount of epochs, the weights we end up with define a line of best-fit.

activation_function :

The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

there are many types of activation function but we have used only two activation functions :

1. **Linear Function :**
   - **Equation :** $y = ax$
   - **Range:** $-\infty\ to + \infty$
   - **Uses:** Linear activation function is used at just one place i.e. output layer

2. **RELU:** Stands for the *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of the Neural network.

   - **Equation :** $A(x) = max(0, x)$. It gives an output x if x is positive and 0 otherwise.
   - **Range :** $[0, \infty)$
   - **Uses:** ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

$Predicted_{Data} - Actual_{Data}$> Threshold Difference Value
then,
Datapoint is classified as a potential change point and
the timestamp is stored in change points timestamps data

Steps to use change point detection algorithms on time-series data

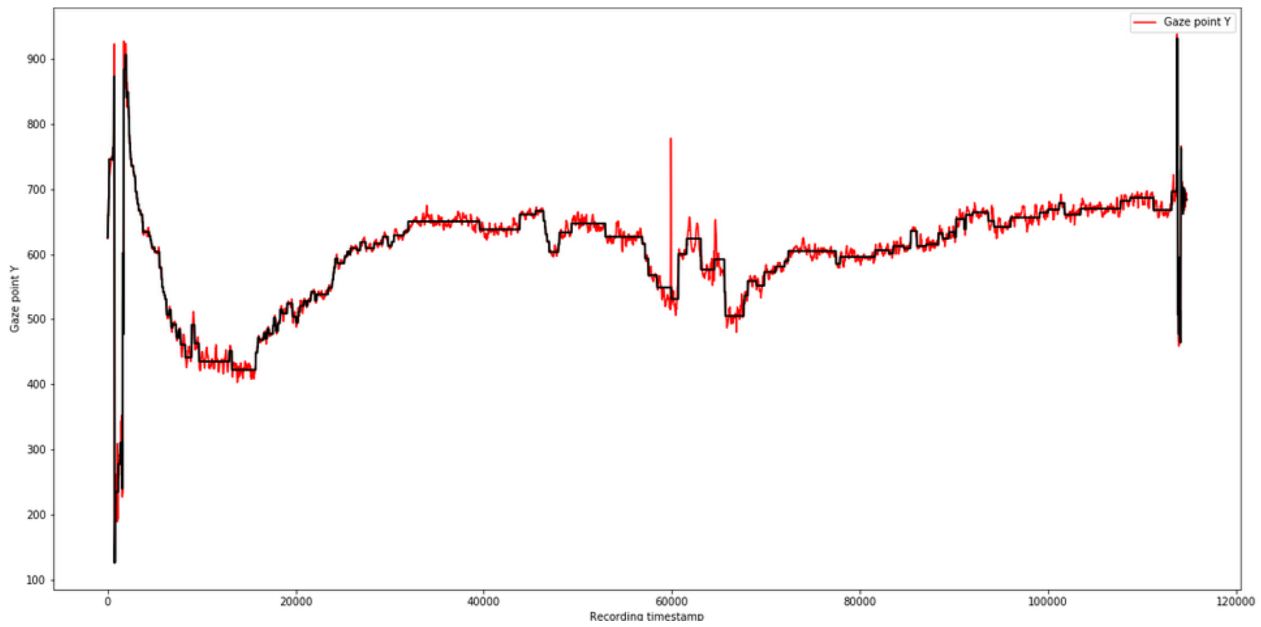
We have used these algorithms on the eye gaze dataset with x and y coordinates of eye gaze point at a given timestamp. These algorithms can be used on any time-series data to detect change points using the steps mentioned below.

- Pre-process the time-series data by removing all the anomaly values in the dataset to avoid any discrepancies while applying algorithms.
- Plot data to get an idea of it about states, abrupt changes, etc. this will help in hyperparameter tuning required in further steps of algorithms.
- Create the model needed for the algorithm and train it on the time-series data.
- Use the trained model to predict the values on the same time-series data and record it.
  -For algorithm 1, you have to train
- Plot values are given by trained models and actual values on the same plot to get an understanding of hyperparameter tuning.
- Implement the final step in the algorithm to get change points as it is a loop-based step to get a count of potential change points in data.

Result

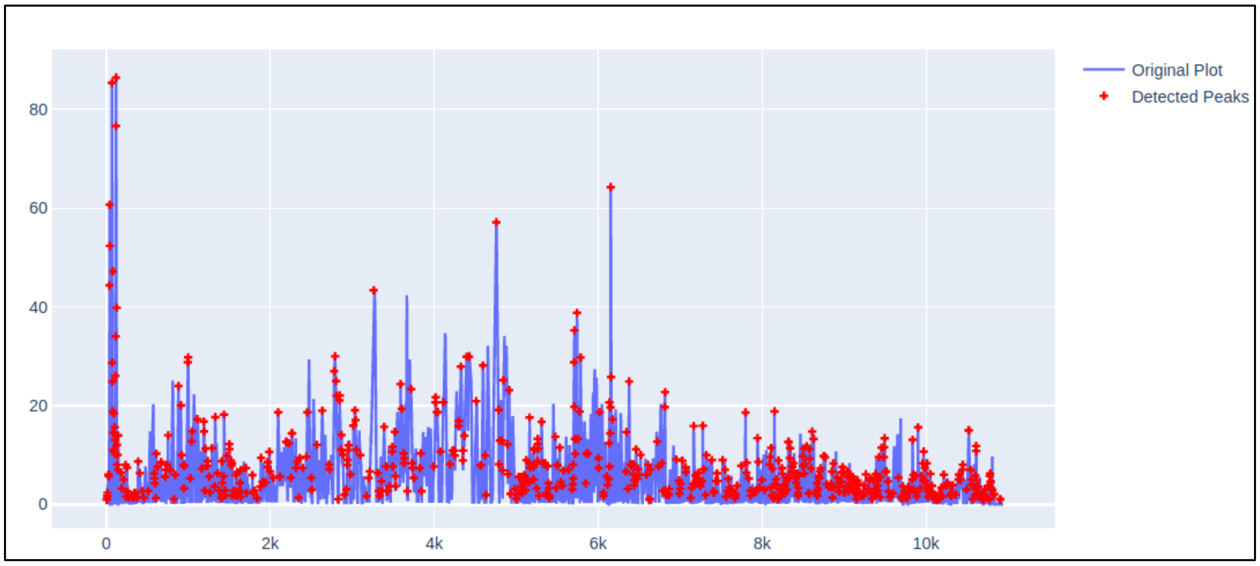
While studying eye-tracking or similar fields, you must have a clear idea of change points in your dataset. Change points information gives a clear picture of time intervals where abrupt changes are occurring
and change points are forming.
All three algorithms above are applied to eye gaze time-series data. Commendable results arrived for change points using these algorithms.
Although to check the actual accuracy of the algorithm, actual change points data is required. But in such cases where actual change points data is not available or a significant amount of original change points data is not available, comparing of change points can be done to get the accuracies of algorithms.

<u>Results from application algorithm 1 on eye gaze data</u>



$$X_{axis} \rightarrow Recording\ Timestamp$$
$$Y_{axis} \rightarrow Gaze\ point\ Y$$

Predicted values by trained Decision Tree Regressor and original time-series data plotted on the same graph to get a visual idea about data and trained model.



Peaks in differences plot

$$X_{axis} \rightarrow Recording\ timestamps$$
$$Y_{axis} \rightarrow Differences\ in\ values\ of\ data$$

Calculating no of change points:

Predicted_Data(t) = Value predicted by Decision Tree Regressor at t timestamp
Data(t) = Original value at timestamp t
Count = no of change points

Predicted_Data(t+1) $\neq$ Predicted_Data(t):

t = timestamp of potential change point

Count = No of occurrences of such timestamps t

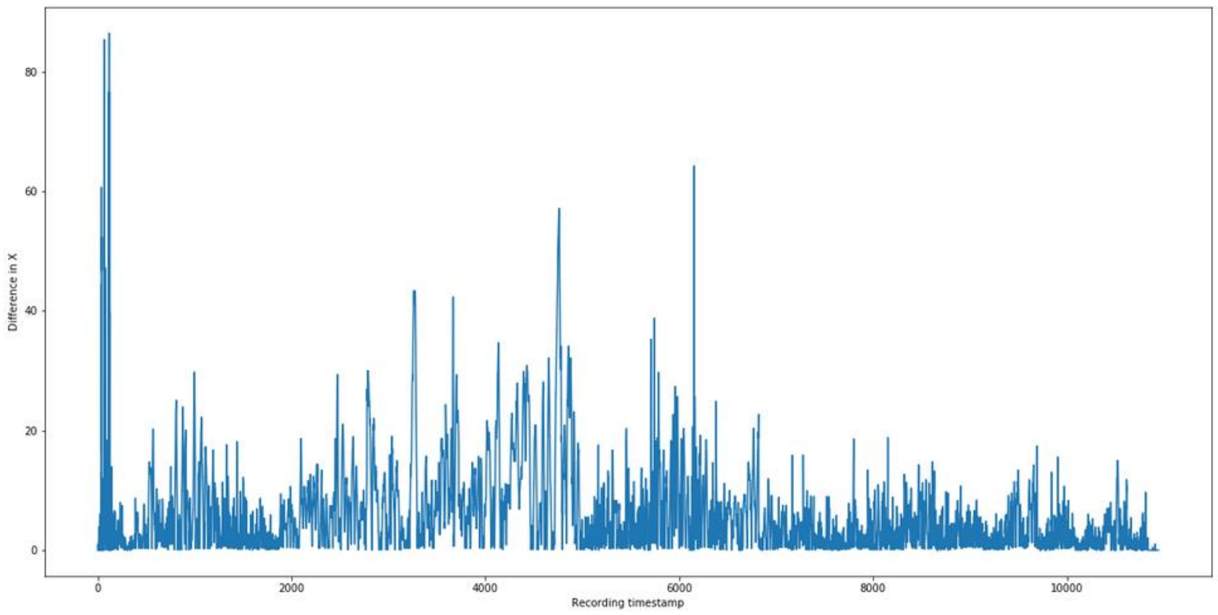
```
count1   #No of change points in gaze point Y
```

```
227
```

```
count  #No of change points in Gaze point X
```

```
523
```

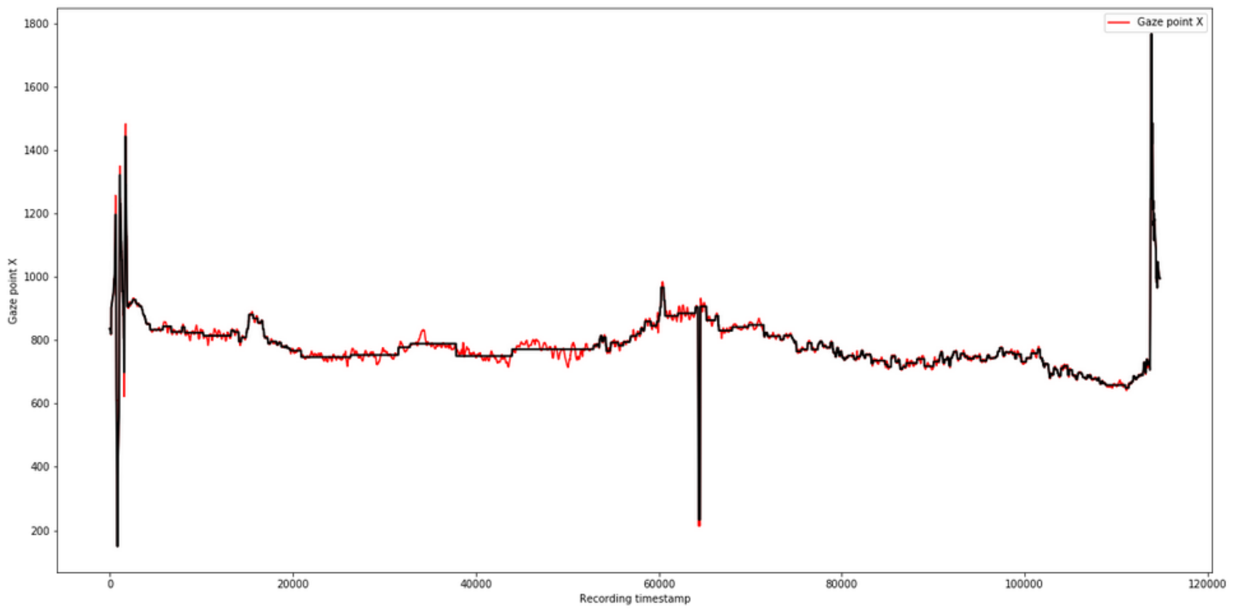
Results from application algorithm 2 on eye gaze data



Plotting differences against timestamps

$$X_{axis} \rightarrow Recording\ timestamps$$

$$Y_{axis} \rightarrow Differences\ in\ values\ of\ data$$



$$X_{axis} \rightarrow Recording\ Timestamp$$
$$Y_{axis} \rightarrow Gaze\ point\ X$$

Predicted values by trained model and actual values on the same plot

Description of differences calculated between actual data and predicted data by deep learning algorithms

count	**10943.0**
mean	**6.084**
std	**7.124**
min	**0.00**
25%	**1.374**
50%	**3.731**
75%	**8.0**

max	**86.40**

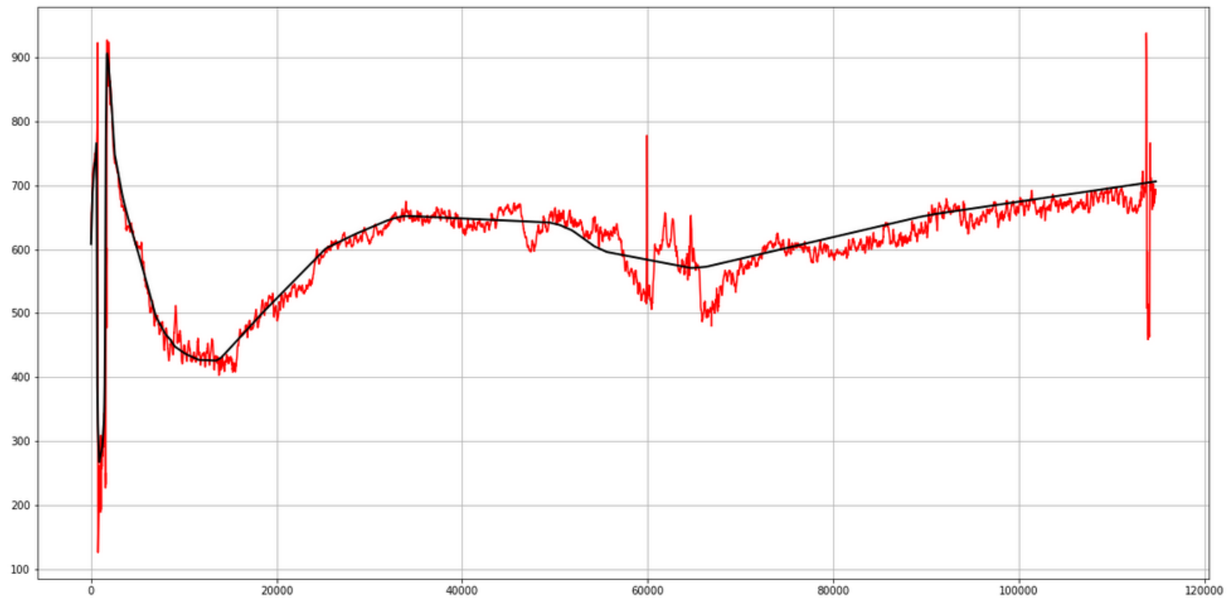
count	**10943.0**
mean	**5.507**
std	**6.851**
min	**0.00**
25%	**1.700**
50%	**3.980**
75%	**7.362**
max	**229.097**

Table 1: Descriptions of differences in gaze point X data
Table 2: Description of differences in gaze point Y data

count = Total no. of data points in the dataset
mean = mean value of differences between predicted and actual values
std = Standard Deviation of differences
25% = 25% of of all the differences
max = Maximum value of all the differences

This description is used for statistical analysis for getting threshold values to be used in algorithms.

<u>Results from application algorithm 3 on eye gaze data</u>



$X_{axis} \rightarrow Recording\ Timestamp$

$Y_{axis} \rightarrow Gaze\ point\ Y$

Deep Learning curve-fitting on the original dataset

For applying Algorithm 2 on Eye Gaze Dataset, deep learning is used to fit on the eye-gaze time series dataset. The above black curve represents the curve of predicted values with *multilayer Keras sequential model* with 7 layers, loss='mse', optimizer="adam", 100 epochs.

Discussion

Eye Gaze Dataset has coordinates of the gaze point of the eye at certain timestamps. To use this dataset further for studying the behavior of that particular eye, disturbances caused to the eye, details of abnormal behavior of eyes have to be known beforehand. Thus it is necessary to locate all the change points in the dataset and have their timestamps.
All 3 algorithms are used upon the eye gaze dataset to get change points in it.
Eye-tracking and eye gaze studies are used in various scenarios like visually impaired person vision problems, focus of eyes while working or studying, onfield jobs which require high attention, driver vehicle control, etc. all this field depends highly on human eye vision. Also, the results of the analysis in these scenarios depends highly on abrupt change points in human eye vision. For visually impaired people, he may face some difficulties while looking that leads to abrupt changes in the way he is looking. Here change point detection is important to find when there were abrupt changes and how many changes were there at a certain time all these things to get the cause and tackle it. Similarly, in all other situations like a person studying or working cases, all the change points are to be known for finding the causes for disturbances and to solve them.
Thus change point detection and algorithms needed to detect them come in the picture in cases like this. These algorithms are applied to time-series data to find change points and are used for further research and analysis.
Similar to human eye vision where eye-gaze data with timestamps serves as time-series data, various other areas require similar kinds of analysis. Historical temperature or climatic data, entertainment industry user data, historical financial data, etc. all such scenarios where data is primarily time-series data, change points come into the picture. For climatic or temperature

data it is necessary to know of abrupt changes occurring in temperature, user fluctuations are necessary to be known for better growth of product/service. It is bound to get phase changes knowledge, abrupt changes, and anomal behaviors in data to work upon those areas and any of the change point detection algorithms can be used to get them.

CONCLUSION

Time-series data contains various sudden changes, phase changes, abrupt changes that result in behavioral traits that are not present all over the dataset. Such points are referred to as change points and finding them out is a task performed by change point detection algorithms.

Change-point detection algorithms can predict abrupt changes in the time-series dataset independent of its cause or magnitude. These algorithms have defined criteria used in the statistical analysis part of the algorithmic process where threshold values are set, these highly control the nature of detected points that will be labeled as potential change points by algorithms.

This gives limitations to change point detection algorithms where the extent or nature of detected change points depends on statistical parameters used to set threshold values while applying algorithms. Due to this, some of the change points that are actual change points in the context of the scenario of the dataset but they might not be detected by the algorithm. Such problems can be tackled by choosing threshold values according to the scenario.