Anomaly Detection of time-series data using k-means algorithm

DATA PREPARATION

Given time-series data is divided into segments of constant length at every sliding interval of length = 2 units .

Example:

For time-series data of size = 1100 , and segment length = 10 , sliding length = 2 . Our resulting segments will be ( [0:10] , [2:12] , [4:14] , … , [1000:1100] ) .

Total segments = 1000/2 = 501 , approximately = 500

Data is prepared by storing these 500 segments as a dataset and each segment in the dataset is referred to as a data point in further discussion .

K-MEANS ALGORITHM

K-means is an unsupervised machine learning algorithm used for classification .

K-means algorithm is modified to improve time-complexity and optimization as shown :-

1 . Given data of n elements , the data is traversed using iteration .

2 . Distance of each element is calculated with the centers using certain error/cost function .

3 . If distance to any center is less than a threshold value , the element is classified under the cluster in which center belongs to .

4 . If the element is not classified to any of the cluster , then a separate cluster is formed with center as current element .

So , the first data point( = segment ) inserted in the cluster becomes the center of cluster .

SORTING

Clusters are arranged to decreasing order of size using this sorting algorithm as shown .

As new data points are added to clusters , the cluster size increases . Then if this cluster size is greater than previous cluster size , then their positions are swapped .

Sort function is called by passing the position of cluster whose size is increased as sort( position) .

Example : If dataset to be clustered consists of set of characters as [ a,a,b,a,c,d,c,d ].

Then traversing the data by each character , the k-means and sorting algorithm work as shown in the table .

|  |  |  |
| --- | --- | --- |
| Iteration | Number of clusters | clusters |
| 1 | 1 | [a] |
| 2 | 1 | [a,a] |
| 3 | 2 | [a,a] , [b] |
| 4 | 2 | [a,a,a] , [b] |
| 5 | 3 | [a,a,a] , [b] , [c] |
| 6 | 4 | [a,a,a] , [b] , [c] , [d] |
| 7 | 4 | [a,a,a] , [c,c] , [b] , [d] |
| 8 | 4 | [a,a,a] , [c,c] , [d,d] , [b] |

At iteration 7 , the size of cluster c is more than b , so they are swapped .

At iteration 8 , the size of cluster d is more than b , so they are swapped .

ERROR/COST

The error function or cost function is defined as : cost( s1 , s2 ) , s1 and s2 are two segments .

Where cost( s1,s2 ) , is sum of absolute value of difference of each corresponding points of s1 from s2 .

So , cost( s1,s2 ) = Σ | s1(k) - s2(k) | , for k=0 to k=segment length .

If n clusters are formed using this algorithm , then anomalies are identified as data points which are not classified to any of the clusters or classified as a separate cluster of minimum size .

Example : So clusters of size less than average cluster size divided by 100 , can be considered anomalies .

THRESHOLD

If the error/cost of a data point form a center is less than this threshold limit , then it is classified in that cluster .

So , less number of clusters are formed if threshold limit is more .

More number of clusters are formed if threshold limit is less .

Since the total number of data points is constant .

So , less number of clusters with greater cluster size are formed if threshold limit is more .

More number of clusters with less cluster size are formed if threshold limit is less .

Example : Given a dataset of size=500 , to be clustered .

|  |  |  |  |
| --- | --- | --- | --- |
| Threshold | Clusters Sizes | Average Cluster Size | Total Clusters |
| 10 | 200,150,70,20,15,15,10,5,5,5,3,1,1 | 38 | 13 |
| 100 | 300,199,1 | 167 | 3 |
| 1000 | 500 | 500 | 1 |

From the table , 'Cluster sizes' denote – clusters of size 200, 150 , .. , 5,3,1,1 are formed .

This set of cluster sizes is referred to as a cluster distribution in further discussion .

If the data is prepared by segmentation of our time-series data into 500 segments .

1 . If threshold = 10 is used , clusters are: [ 200,150,70,20,15,15,10,5,5,5,3,1,1 ] . Anomalies are difficult to identify just by observing cluster sizes .

2 . If threshold = 1000 is used , clusters are:[ 500 ] . Again no anomalies are found .

3 . If threshold = 100 similarity is used , clusters are:[ 300,199,1 ] . Observing the cluster sizes there are 300 segments of similar pattern , 199 segments of similar pattern , but the remaining 1 segment can be considered outlier or anomaly which is not similar to any of the segments .

So , anomalies cannot be identified using very less or very large threshold limits ,100 is the optimal value to find anomaly in this case .

NOTE : That is why we sort the clusters by size , to easily identify anomalies .

But , this optimal threshold varies for different datasets , which is accounted by different data sizes , different percentage of similarities .

The optimal value can be found using binary search algorithm , which is discussed further .

Example : threshold limit = 130 , means that the data point is classified with the center if the error/cost of data point with center is less than 130 .

Where cost is calculated as cost( s1,s2 ) = Σ abs( s1[k] - s2[k] ) .

MAXIMUM THRESHOLD LIMIT

We calculate the maximum value of threshold limit that can be used to cluster the data , which is approximately calculated by finding maximum value of cost function between 1st data point with any other data point , as shown :

Maximum threshold limit = Maximize : cost( data[0] , data[k] ) ; for k =0 to k = 500

|  |  |  |
| --- | --- | --- |
| Threshold value | Number of Clusters | Size of clusters |
| Maximum threshold limit | 1 | [500] |
| 0 | 500 | [1],[1], … ,[1] |

Actually , Maximum threshold limit is the minimum value of threshold at which the whole data is clustered as a single cluster .

BINARY SEARCH ALGORITHM

Binary Search is an algorithm to search element in a sorted array .

Refer: <https://www.hackerearth.com/practice/algorithms/searching/binary-search/tutorial/>

We define two variables : max = Maximum threshold limit , min = 0

Then find the optimal value of threshold limit in the range of Integers [ min : max ] . So , we are finding the value of threshold to get the best distribution of clusters to identify anomalies .

Algorithm :

Upper bound : max

Lower bound : min

We define another variable : optimalThreshold , in which we store the optimal threshold value found by binary search . Firstly it is initialized to zero .

1 . Current value of threshold limit is = ( min + max )/2 .

2 . K-means is applied on this threshold value , to get a cluster distribution .

3. If we get a very large number of clusters , this cluster distribution is not desirable . To find a better distribution we must reduce the number of clusters . To reduce number of clusters we require a threshold value greater than current value , which is done by increasing the lower bound of binary search to current value , and repeating with step 1 .

4 Similarly , If we get very less number of clusters , we decrease the upper bound and repeat with step 1.

5 In both the cases of step 3 and 4 , our cluster distribution was undesirable , but in case if we get a desirable cluster distribution . Modify the value of optimalThreshold to current value of threshold limit .

The cluster distribution is categorized as desired or undesired , according to a criteria which will be discussed further .

This desired value is not unique in our range of min to max .

For example : d1 = 150 , d2 = 120 ; are two desired values in the range of [ min : max ] . Clusters are created using d1 = 150 and d2 = 120 as shown :

|  |  |
| --- | --- |
| desired threshold value | Cluster distribution |
| 150 | 300,199,1 |
| 120 | 300,100,97,2,1 |

For d1 = 150 ; total anomaly points = 1 , since last cluster size is very less .

For d2 = 120 ; total anomaly points = 2+1 = 3 , since last two cluster sizes are very less .

If actual number of anomalies is 1 , it will be included in both cases . If actual number of anomalies is 3 , it is included only for d2 = 120 .

So , out of two desired values , the value d2 = 120 , is more optimal . Desired value which is less is more optimal .

Therefore , whenever we find a desired value in step 5 , we modify the value of optimalThreshold and decrease the upper bound , so that next time if we find a desired value again , it will be more optimal and optimalThreshold will be modified to more optimal value .

6 . This is a recursive algorithm which stops when min value becomes equal or greater than max value .

CRITERIA TO EVALUATE A CLUSTER DISTRIBUTION

Suppose we apply k-means to some current value as threshold in binary search algorithm . The clusters so produced are passed to this function .

This criteria is used to assist our binary search algorithm by analyzing the cluster distribution of current value in binary search , by determining a cluster distribution as desired or undesired and weather to increase the lower bound or decrease the upper bound .

This function is called within binary search function by passing the cluster distribution of current value This function returns an integer value which may be 0 , 1 or 2 .

1 . If return value is 0 , means cluster is undesirable and increase the lower bound .

2 . If return value is 1 , means cluster is undesirable and decrease the upper bound .

3 . If return value is 2, means cluster is desirable and decrease the upper bound . ( to find more optimal values )

Inside the function , three variables are defined as :

Mean = Total number of data points / Total number of clusters ; mean is the average cluster size .

Large = Largest cluster size

Small = smallest cluster size

As our cluster distribution is sorted by size , so Large is the first element and small is the last element . Example: cluster distribution = [ 400 , 50 , 40 , 10 ] , large = 400 , small = 10 , mean = 500/4 = 125

Then the return value is decided by a set of nested if-else statements with conditions as shown:

if( mean/small < anomalyRatio )

{

if( center.size() > data.size()/20 || large/small > anomalyRatio )

return 0 ;

else

return 1 ;

}

else

return 2 ;

ANOMALY DETECTION TILL HERE

1 . We prepare the dataset of segments from given time-series data with constant segment length and slide length .

2 . Find the value of maximum threshold limit .

3 . Apply binary search to find optimal threshold limit in the range of : min = 0 ; max = maximum threshold limit . The binary search is converged using the normal distribution criteria .

4 . Before applying binary search , value of optimalThreshold = 0 .

So , after binary search

if optimalThreshold > 0 , then anomalies are found .

If optimalThreshold = 0 , then no anomalies found .

FINDING POINTS OF ANOMALIES

When we find that the data contains anomalies , we need to find the points of anomalies .

So , we apply k-means , this time with threshold value =optimalThreshold , which is optimal threshold value found by using binary search .

Since optimal threshold value is used , we get desired cluster distribution , so anomalies can be easily identified as the clusters with exceptionally small size .

For Example :

If time series data size = 1100 , so we had 500 segments , with segment length = 10 , slide length = 2 . Segments are : ( [0:10] , [2:12 ] , [4:14 ] , … , [1000:1100] )

These segments are clustered by k-means .

Another variable is defined called Index , which stores the index as first point of individual segments of each cluster .

For segment s: [ x : x+10 ] , the index value is x .

For a cluster distribution : C[0] , C[1] , …, C[M]

If a cluster C[0] contains a set of segments s : [x1 : x1+10 ] , [x2 : x2+10 ] , .. , [xn : xn+10 ] . The index for the cluster is stored as Index[0] : [ x1,x2,x3, … , xn ] .

Similarly C[1] has index as Index[1] and so on .

So , If after applying k-means on optimal threshold value the cluster distribution is :[ 300] ,[ 198] , [2 ] , segment index of each element of each cluster is stored in variable : Index , as Index[0] , Index[1] and Index[2] .

Algorithm :

1 .Apply k-means on optimal threshold value – optimalthreshold , and get desired cluster distribution .

2 . Clusters with cluster sizes less than average cluster size divided by 100( for example) , have exceptionally small cluster size .

Algorithm is explained using this example .

Example : cluster distribution : [300 , 198 , 2 ] , avg cluster size = 167 .

3 . These clusters contain anomaly segments , last cluster in the example above .

4 . Index of anomaly segments are in Index[2] = [ Index[2][0] , Index[2][1] ] .

5 . So , two anomaly points are calculated as : point 1 = Index[2][0] + segment length

Point 2 = Index[2][1] + segment length

Why we add the segment length is discussed in section : FILTERING THE RESULTS

6 . Point 1 and Point 2 are anomaly points found in the above example .

FILTERING THE RESULTS

If the actual anomaly in time-series data is at point= x .

Using this algorithm , instead of getting a single point anomaly , we will get more number of anomaly points because this point is included in 10 segments as : [ x-10 : x ] , [ x-9 : x+1 ] , .. , [ x : x+10 ] .

So we do filtering by removing redundant anomaly points which are not far by distance = 10 (segment length ) .

So , [ x-10 : x ] , [ x-9 : x+1 ] , .. , [ x : x+10 ] is filtered as - > [ x-10 : x ]

Then anomaly point is -> Index of [x-10 : x ] + 10 -> x-10+10 -> x . , which is actual anomaly point .

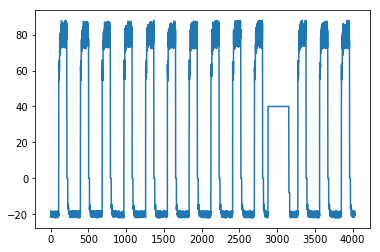
TYPES OF ANOMALY

There are two types of anomalies

1 . Large Scale anomaly : When the time-series data follows a large scale pattern ( For example : sine wave ) , anomalies are the points where pattern is disturbed .

Example :

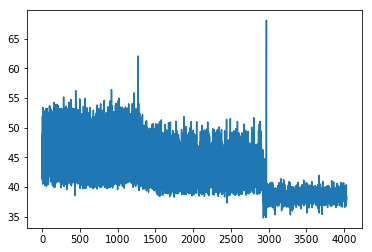
Segment Length = 100 : gives anomaly points [ 3084 ]



2 . Small Scale anomaly : On a small scale , anomalies are points of sudden spikes .

Example :

Segment Length = 12 : gives anomaly points [ 1262 , 2960 ]



Finding the Large Scale or Small Scale anomaly entirely depends on choice of segment length , which we took as 10 in many examples .

If segment length is large = 100 : the dataset prepared by segments will contain information about the pattern of waves .

If segment length is small = 12 : the dataset prepared by segments will contain information about magnitude of points .

RESULTS

We are interested to find both type of anomalies , irrespective of the size of data , which can be done by iterating over different values of segment length .

Algorithm :

Variable is defined allPts .

1 . Initialize segment length = data size/10 .

2 . Find the anomaly points using whole ( k-means for anomaly detection ) algorithm .

3 . Store all the points in variable allPts .

4 . Decrease the segment length by half , and repeat from step 2 .

5 . Stop the loop when the segment length is less than 10 .

6 . Anomaly points of both type large Scale and Small Scale are stored in allPts .

JAVA CODE

STATIC VARIABLES DEFINED ARE:

1 . upperBound : stores the upper bound of threshold value in binary search function .

2 . lowerBound : stores the lower bound of threshold value in binary search function .

3 .optimalThreshold : stores the optimal value found by binary search .

4 . data : stores the whole time series data .

5 . index : stores the position of first point of segments , of each cluster . So for each cluster of segments a separate index array is maintained .

6 .center: stores the center of each cluster as an individual segments in the array .

7 .segment : stores new generated segments to be processed .

8 .clusters : stores the clusters as a hashmap , with key as center and value as the array of segments in cluster .

9 .anomalyPoints : stores the anomaly points found in each iteration of segment length .

10 .segmentLength : stores length of segment by which segment data is to be prepared .

11 .anomalyRatio : used to find anomaly clusters by identifying the cluster sizes which are exceptionally small , its value is = datasize/40 , so clusters with size less than Mean cluster size / anomalyRatio are anomaly clusters .

12 .allPoints : collects the anomly points found in each iteration of segment length by storing the values of anomalyPoints .

FUNCTIONS

1 . readFile() : Reads the csv file , and stores the time series data in variable – data . Code has been tested on csv file , with time series data on second column .

2 . cost() : calculates cost/error between two segments .

3 . sort() : whenever a new element is added in a cluster , its position is passed to this function , to sort the cluster position in decreasing order .

4 . MaxminCost() : calculates maximum cost between first segment and every other segments , this maximum cost is used as an upper bound in binary search function .

5 . kmean() : forms clusters of segments , appends new segments passed to the function to existing clusters .

6 . normalDis() : determines the convergence criteria of binary search function .

7 . binarySearch() : finds the optimal threshold value by using binary search .

STATIC VOID MAIN FUNCTION

