Anomaly scoing of univariate temporal processes

Abhishek.Mukherjee

January 2021

1 Introduction

There are plethora of techniques to detect anomalous points in a temporal process. One such technique is Density-based spatial clustering(DBSCAN) which tries to find regions having minimum density being separated by areas of lower density and since DBSCAN is one of the most common clustering methods when it comes to finding clusters in temporal processes I decided to use this algorithm for outlier classification or Noise classification and anomaly scoring. Other off the shelf algorithms such as the K-means or the K-means ++ are more suited to clustering algorithm where clusters are assumed to be more or less convex and moreover these techniques require us to specify the number of clusters before hand which might not be always possible.

For a temporal stochastic process, the *Outlier scores* or *Anomaly scores* (A_s) for every data-point evolves over time and at each time step the magnitude of the data-point for that time-step might affect A_s of the previous data-points.A higher A_s(greater than 1) would indicate that the point can be classified as an Outlier,Anomaly or Noise whereas any values less than that would represent the abnormality score or the Anomaly scores(A_s) for the *inliers* i.e points which cannot be classified as *outliers*. To get numeric values representative of the abnormality of every data-point a cost function can be derived using the *cardinality* of the different class labels and the distance of every data-point from the non-anomalous mean i.e the numeric mean or *True Mean* of all the points except the ones classified as noise or anomalous.The class labels referred here are the final membership labels assigned by the DBSCAN algorithm.It might occur to the reader that there would be only two distinct classes or class labels but the *inliers* can also have different class labels depending on their distance from the *True Mean*. These class labels or the class membership labels for every data point can be thought of cluster labels for the different clusters formed by the *inlying* data-points. Since this is a uni-variate process all the samples or data-points can be pictured as distributed on a number line.

2 Algorithm Description

Primary Assumption: *The download activity of the users has been considered as independent to one another.This is done keeping in mind that in general a certain user might have a general tendency to have high download activity hence for that person a lower download measure would indicate an anomaly. So, Anomality is probably better defined with respect to a specific users general activity.*

As previously mentioned the original algorithm identifies points within a certain radius(generally denoted by ϵ) to be considered part of the same cluster as the considered data point and if the number of such points in the ϵ neighbourhood of the *core point* are less than a minimum threshold(*minPts*) then that cluster and all the points within it can be labeled as *Noise* or *Outlier* otherwise the point is an *Inlier* and gets a certain cluster index [1]. This process is carried out for every incoming data point and if some data point at a certain time step having been just labeled as an *Outlier* by the above described process comes within ϵ neighbourhood of any point which was previously labeled as *Noise* that point along its ϵ neighbourhood re-labeled as an *Inlier* and gets the same cluster index as the point within the ϵ neighbourhood of the considered data point.

The above process just assigns cluster index to the data-points but the problem considered here requires us to assign score or measure of abnormality($\mathrm{A_s}$) for each data-point at every time step. This requires us to come up with some sort of cost function that would gives us the measure of how much abnormal a point was. To come up with that we have to take into account the factors that affects its *abnormality* like its distance(euclidean norm) or likelihood from the *True Mean* and the *cardinality* of the cluster the point belongs to. At this juncture, the reader might argue about the need for including the *cardinality* of the class (*cardinality* of the cluster) in the cost function. To justify this I would encourage the reader to think of circumstances where the number of *high-magnitude* data points are large in number considered to *small-magnitude* points.

Having ironed out the factors that affect the *Anomaly score* of every data-point we need to think of the relation they might have with the score ($\mathrm{A_s}$). It is quite clear that the *distance function* or the *euclidean norm* of a data point from the *True Mean* would have a direct relation since it gives us a *sort of direct measure* of the abnormality and the *cardinality* of the class or cluster would have an inverse effect on the *abnormality score* since higher the number of points in that cluster lower are the chances of it being a *Anomaly*.

As a side note, the reader is encouraged to derive a cost-function following *square law* by modifying the hyper-parameters in the cost function albeit for a small data-set this might not make any significant difference.

2.1 Parameter and Hyper-parameter tuning

The cost function has two hyper-parameters α and β. The need for hyper-parameter α comes from the fact that we need to assign a *Base score* or minimum

score to every point. This is mostly to avoid cases where a point might come very close or even be equal to the *True Mean* of all the points. In such cases the value(s) would become very small and could have precision issues.To avoid such cases a base score is assigned to every point so that they are bootstrapped. A base score of 0.1 has been found viable experimentally for the data-set provided.

β can be thought of as the *penalty parameter* this penalizes the ratio of the *distance function* by *cluster cardinality* for the considered point. Depending on the value of β the *importance* or weight of the two factors affecting the cost function can be varied. The reader can think of β performing the *Balancing act* between the numerator and the denominator for various use-cases. A penalty value of 0.01 has been found to give *better results* for the provided data-set.

The term *better results* can cause a bit of conundrum, since the results(scores) from the experiment are rather subjective and on top of that we don't have any so-called *ground truth* to measure our result against. So, the interpretation of the term *better results* narrows down to the fact that scores should be easily distinguishable from one another i.e having some minimum precision difference among themselves(both within a class and among classes) and also be in a range which doesn't cause any error in the output instrument. This fact turns out to be more important in cases where a large outlying value can cause the *Anomaly score* to go out of bounds.To negate such cases tuning of the hyper-parameter turns out to be quite important. The chosen hyper-parameter helps gives a score between 0.1 and 1 to all the *Inlying* points and assigns a score of more than 1 to all *Anomaly points*.Intra-cluster scores as well as the inter-cluster scores also turn out to be more *prominent* from one another using this hyper-parameter set.

For tuning the parameters, ϵ and *minPts*, a grid search over the parameter space can be performed or one can go with the *heuristic approach* detailed by the author and some revision papers on it[2][1]. A value of (3,2) for the parameter set (ϵ,*minPts*) has been used in the experiments.

Algorithm 1: Modified DBSCAN algorithm

Input: DB : Database
Input: ϵ : Radius
Input: *minPts* : Density Threshold
Input: *dist* : Distance function
Data: *label* : Point labels,initially undefined
Parameter α : Base score, experimentally determined
Parameter β : Penalizing Hyper-parameter, experimentally determined

```
for each point p in Database DB do
    if label(p) ≠ undefined then continue
    Neighbours N = RangeQuery(DB,dist,p,ϵ)
    if |N| <minPts then
        label(p) = Noise
        continue
    end
    c = next cluster label
    label(p) = c
    Seed set S = N/p
    for each q in S do
        if label(q) = Noise then label(q)=c
        if label(q) ≠ Undefined then continue
        Neighbours N = RangeQuery(DB,dist,p,ϵ)
        label(q) = c
        if | N | <minPts then
            continue
        end
        S = S ∪ N
    end
end
```

Addendum to the original algorithm for Anomaly scoring

$$\textbf{True Mean } \mu = \frac{\sum_{i=1}^{|S|} S_i \times \mathbb{1}_{[label(S_i) \neq' Noise']}}{\mid S \mid}$$

```
/* |S| defines the cardinality of set S                    */
for each s in S do
    LabelCardinality(s) = |label(s)|
    /* Number of points with same label/class/cluster
       membership as element s                              */
```

$$A_s = \alpha + \frac{\beta \times \mid s - \mu \mid}{LabelCardinality(s)}$$

```
end
```

3 Conclusion

The modified DBSCAN algorithm seems to have done well in assigning *Anomaly score* to the given data-points and given the high usage of DBSCAN for detecting *Anomalies* in temporal processes the modified algorithm can quickly become a useful addendum to the existing literature of algorithms used for anomaly detection in a temporal processes.

References

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[2] Erich Schubert, Jörg Sander, Martin Ester, Peter Hans Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 2017.