Unsupervised learning in evolving environments

Suraj Pattar and Zsolt Pasztori

Abstract— **This project proposal addresses the problem of learning from data streams. Data streams are quickly taking the stage as a major paradigm in data science. A number of relevant cases have arisen as a consequence of the availability of cheap, highly autonomous devices for sensing, computing, and transmitting information. These include sensor-equipped smartphones, the Internet of Things, ambient and wearable sensors. Data streams play a very important role in robotics since the output of proximity sensors, cameras, encoders etc. can be treated as such. All of these provide uninterrupted flows of data that need to be taken care of in real time. In other words, many, if not most, interesting data today come in the form of streams. We are particularly interested in the data analysis process of data clustering, which is very relevant because it is by itself one of the pillars of information mining, and at the same time it is also a building block underlying some successful approximation (regression) and classification (decision-making) models. In this research, we will study methods to evaluate the validity of an existing streaming clustering model when the data is changing. As a consequence of such change, which may be due to several reasons (e.g., an autonomous robot exploring different parts of an environment, a computer vision system working under evolving daylight), the current model will be less reliable and lose relevance, so it will need to be adjusted. Methods for detecting such decrease, and methods for responding to the situation, will be implemented.**

I. INTRODUCTION

In recent years with the spread of information technology in to the every day life, the amount of data generated each day has become almost unimaginable. Just to depict this, there is 300 hours of video uploaded to the worlds biggest video streaming site, youtube.com, every minute. There are millions of mobile phones, computers, cameras and other sensors connected to the Internet, uploading data to it constantly. These flows of data are called data streams. During our group project we have examined two different approaches to process this vast data, concentrating on removing the inherent noise from it. Working with data streams has some very specific challenges, these include:

- The sheer size of the data can be overwhelming. Data might not fit into the memory, or even to any other storage. The processing of the data must be constant and faster then the generation of it. Otherwise some parts of the data might be lost or the algorithm might lag behind. It is important when dealing with data streams to be able to recursively update the model, this way we can make only a single pass over data and reduce the computation time.
- The underlying patterns of the data can change over time. The algorithm must be updated to reflect these changes. This can happened because the robot is situated in a changing environment, or it is exploring a new one.There are two kinds of changes:
  - The underlying distribution's mean and variance can change slowly over time. This change is called shift. It is relatively easy to follow by updating the models regularly as data arrives. During our work we have focused on being able to manage this phenomenon.
  - In the underlying data some distributions might disappear and new ones can appear. This is harder to detect, since the new distributions are uncorrelated with the old ones, at first they can be falsely identified as anomalies. It is a conceptually hard problem to manage this kind of change, since every algorithm's first assumptions is that the there is a correlation between the past and future values.
- For some types of data, for example great amount of discrete attributes, even the representation of a model might get too big. The model representations has to be balanced both in computation effort and storage space against precision. For some data types this representation issues can be crucial but in our case, since we are focusing on floating point data, this effect can be neglected.

II. LITERATURE REVIEW

During our group project we have examined ways to detect anomalies in data streams. For this purpose we need to be able to compare new data points to a representation of previous knowledge, since an anomaly is a point which does not conform to an expected distribution. The first approach we looked at was based on unsupervised machine learning. Unsupervised learning gives an opportunity to learn representations from high amount of data without the help of labeled training data. This way we can reduce the data from potentially millions of entries to a few managable distributions, which can be represented by points. Supervised learning has the advantage of easy interpretability and evaluation, but it needs carefully labeled data, which is usually scarce and expensive to obtain. During k-means clustering we are partitioning n data points into k clusters. The k clusters are represented by k center points, this way we reduce the input data dimension n to k considerably, where $n \gg k$. When a new data point arrives the algorithm tries to assign it to the already existing clusters, if a strong assignment is possible then the point conforms to our expectations. If the point cannot be assigned then it can be considered an anomaly. The cluster centers are later updated according to the new point if it was found not to be an anomaly. The difficulty in k-means clustering comes from the decision of hyperparameters. We need to decide how many clusters to use and what is the measure when we can say that a point is anomaly. The second

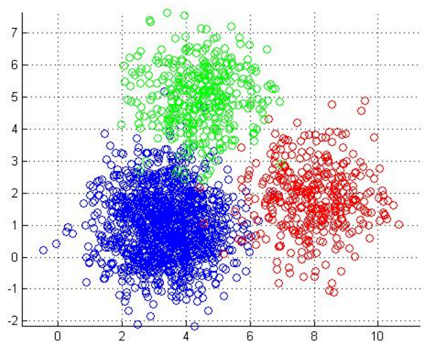

Fig. 1. 3-means algorithm on 3 normal distributions

approach is based on the recently proposed typicality and eccentricity data analysis (TEDA) framework. This approach is conceptually different from the usual statistical or the clustering methods. It does not seek to represent our data by reducing dimensionality to an ”average” distribution, but rather compares new points to all previous points and from there it deduces whether it is an anomaly.

The popular statistical methods are based on three assumptions: there are infinite data points, the underlying distribution is Gaussian and the data samples are independent. The TEDA framework can be considered as a generalization of the popular statistical methods, it does not depend on the previously mentioned assumptions. Its only constraint is that the data points can not be fully independent.

TEDA is based on two measures. The first is the accumulated proximity, which is the distance of each point from every other point in the dataset. The second measure is eccentricity which is the points contribution to the accumulated proximities in the whole dataset. This is obviously a good measure of how strongly the point is correlated to every other point in the dataset. To give a threshold of eccentricity most methods use Chebyshev's inequality. It states that no more then $1/n^2$ of the data can be n* away from the mean (where denotes the standard deviation). Chebyshev's inequality depends on the mean and the standard deviation of the samples, and also the number n has to be defined earlier. TEDA provides the so called ” gap” threshold which is analogous to it, but it looks for a difference between points and not means.

III. METHODOLOGIES

In the next section we will describe the two methods which we have evaluated for anomaly detection.

A. Anomaly detection based on membership

This method uses Fuzzy K-means Clustering. Fuzzy K-means is based on the k-means method. Since k-means method assigns each point from the dataset to a single cluster, it can not be used for finding anomalies. On the other hand fuzzy k-means assigns a probability to each point, the probability shows how much it is likely that the point belongs to one of the k clusters, this means that each point has at maximum k probabilities (membership) assigned. If the point clearly belongs to one of the clusters it will have one dominant probability and the others will be close to zero. On the other hand if the point can not be assigned easily it will have several probabilities but none of them dominant, this feature can be used to asses how well it fits to our previous points.

In fuzzy k-means methods the sum of memberships for a point is equal to 1. However in the algorithm presented here this is not the case, and the sum of memberships will be used to show the strength of a point being an anomaly. To calculate this, we need to calculate the new point's distance from the k cluster centers, the distance metric we used was the euclidean distance. Since we assume that the data points are independent and they are influenced by high amount of random processes, we can assume, according to the central limit theorem, that the underlying clusters will conform to a Gaussian distribution. To take this into account we scale the distances exponentially. At the scaling we also introduce a scaling factor b, which is a predefined coefficient, it controls the level of scaling.

$$d_{scaled} = e^{-\frac{d}{b}}$$

This scaling transforms each distance from the center to a number between 0 and 1. We want however the sum of distances to become less then 1, so another scaling is needed, regarding the sum of the distances. Another model parameter, alpha, must be added, which shows how stable the clusters are. If the value of alpha is close to 1, our clusters are stable and we are getting close to the probabilistic k-means method.

$$d_{rescaled} = \frac{d_{scaled}}{(\sum_{j=1}^{k} d_{scaled})^{a}}$$

Now everything is given to recognize outliers. If we sum up the membership of each cluster we get a number between zero and one. If the sum of the membership is close to one, then our point fits well into the model. On the other hand when the sum is smaller than 1/k, we have found an outlier. If the point is not an outlier we can recalculate the cluster centers with its help, to balance out the drift.

Although the method was created to balance out only the shifting of the clusters, it can give an idea about the drift too. If there are no outliers the cluster centers are slowly updated. If there are only few outliers they can be easily filtered out. But if there are several outliers it means that our data is drifting, new clusters are appearing old ones are disappearing. This change is not handled by the method presented here, but can be assessed looking at the clustering results.

The algorithm can be summed up as:

1) Calculate membership of each data point to each cluster at each iteration.

2) Check if sum of membership to each cluster is greater than $1/K$.(Here K is number of clusters).
3) IF yes, then update the clusters at the next iteration.
4) ELSE ignore the data points with sum of membership less than $1/K$.

bi0 = 0.005 (more or less sensitive to distance) if b is less more sensitive to distance, vice versa (put formula)

eucledian distance taken(squared eucledian distance) square because we are comparing (formula)

eta0: 1 (what is eta0)

alphamin: 0.8 (keeps the membership from dropping to zero)

- mention how the membership is being calculated

The above algorithm is illustrated by using artificial 2D drifting data with anomaly created at a specific point of time. See Figure (a) and (b):

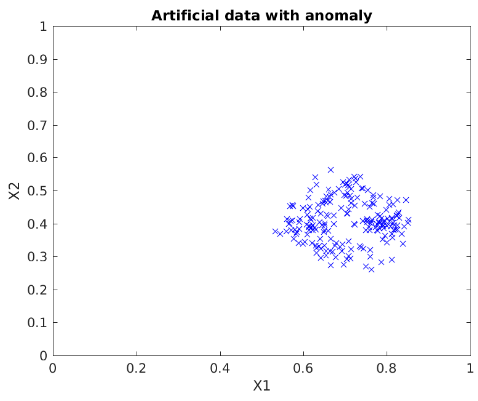

Fig. 2. Fig(a) Data at iteration around 50 with no outliers

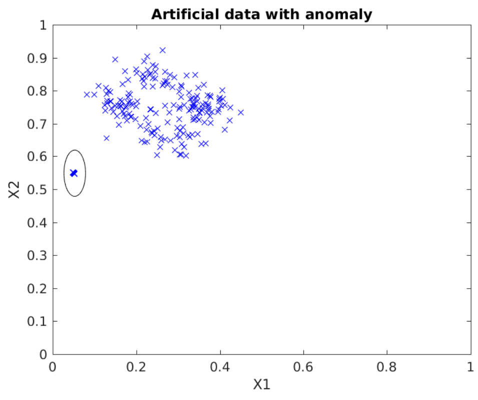

Fig. 3. Fig(b) Data at iteration around 2000 with anomalies

To cluster the above drifting data, we use random initial centroids. If specific points are provided, they work just as well as random points. **K** is the number of clusters, which in our case is 4. This is one of the presumptions required for our algorithm. See Figure (c):

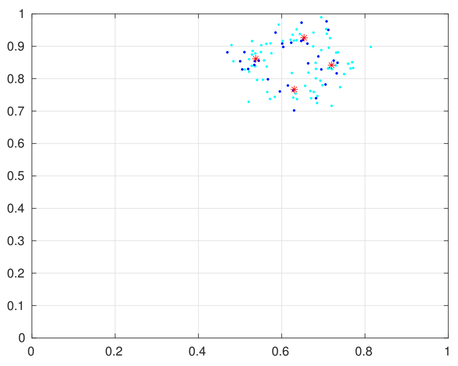

Fig. 4. Fig(c) Drifting data clustered with K = 4

Here the *light colored data points* represent the past values and the *dark colored data points* represent the current values.

Before implementing our algorithm, the anomalies as shown in fig(b) is not detected as its membership still affects the cluster centroid to shift. See Figure (d):

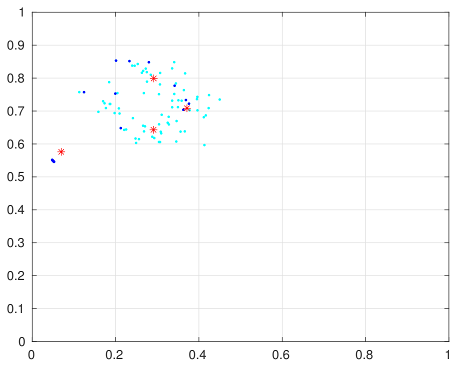

Fig. 5. Fig(d) Anomalies causing centroids to shift

But when the data points whose memberships are below *1/K* (where K is the number of clusters) are ignored, the clustering process considers these points as anomalies and they are ignored from clustering. See Figure (e):

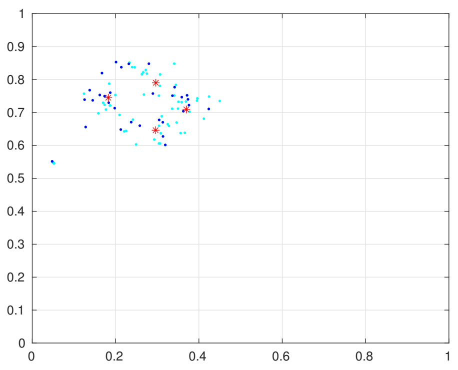

Fig. 6. Fig(e) Anomalies ignored by the algorithm

bi0 = 0.005 (more or less sensitive to distance) if b is less more sensitive to distance, vice versa (put formula)

eucledian distance taken(squared eucledian distance) square because we are comparing (formula)

eta0: 1 (what is eta0)

alphamin: 0.8 (keeps the membership from dropping to zero)

- mention how the membership is being calculated - Steps: - get X by running showdriftdata -

B. The "σ gap" principle

The alternate algorithm uses a different approach. It detects anomalies before clustering. The following characteristics valid for each data sample are introduced:

1) *accumulated proximity,* π from a particular point x $\varepsilon \chi$, to all remaining data samples up to the k^{th} from a given, j^{th} (j¿1) data sample calculated when k (k¿1) data samples are available[2]:

$$\pi_k(x_j) = \pi_{jk} = \sum_{i=1}^{k} d_{ij} \quad k > 1$$

where d_{ij} denotes a distance measure between data samples, x_i and x_j, for example Eucledian, Mahalanobis, cosine, etc.[1]

2) *eccentricity* of a particular j^{th} data sample calculated when k (k ¿ 2) data samples are available (and they are not all the same by value) [2]:

$$\xi_{jk} = \frac{2\pi_{jk}^{k}}{\sum_{i=1}^{k} \pi_{ik}} \quad \sum_{i=1}^{k} \pi_{ik} > 0$$

3) Variance is defined as[1]:

$$\sigma_k^2 = \sum_{i=1}^{k} \frac{(x_i - \mu_k)^T (x_i - \mu_k)}{k}$$

4) Normalized eccentricity[1]:

$$\zeta = \frac{(x_k - \mu_k)^2}{2k\sigma_k^2} + \frac{1}{2k}$$

It can be verified that eccentricity ξ is bounded between *0* and *1* and normalized eccentricity ζ sum upto *1*.

The rationale is as follows. The *outlier* is a data point/sample that stands out and is different from other data samples. In the traditional **nσ** analysis each sample is compared with the mean/average which is representative of all data samples. The "σ gap" principle proposes to analyze the normalized eccentricity of the data samples instead of their values. In this way, the spatial proximity to all other data samples is taken into account by definition and pairs of data points are compared in an accumulated and aggregated form with all other points (through the eccentricity). If for a pair of data points starting from the point with the highest normalized eccentricity, $x^1 = x|\zeta(x^1) = max(\zeta(x_i))$ there is a σ gap in terms of their eccentricities then it is an outlier.

The σ gap condition is very intuitive and is defined as follows:

$$\mathbf{IF}(\Delta\zeta^{1,2} > n/k)\mathbf{THEN}(x^1 is an outlier)$$

The algorithm can be summed up as:

1) Calculate normalized eccentricity of a point.
2) Keep the point with the maximum normalized eccentricity, x^1 second maximum normalized eccentricity, x^2, etc.
3) Check the **"σ gap"** condition.
4) If it is satisfied, declare the point x^1 an outlier.

We made use of the same data[5] to replicate similar results as in the reference paper[1], see Figure 1.

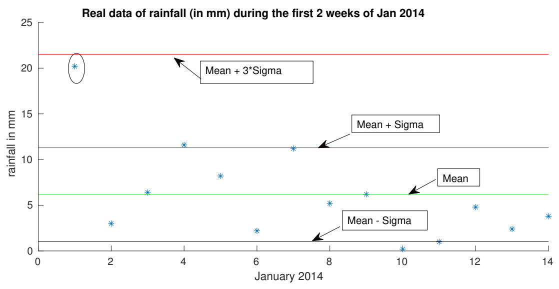

Fig. 7. Fig 1 Real rainfall data from Bristol, UK, first two weeks of January, 2014 [7,14].

As can be seen from the above graph, traditional "$n\sigma$" approach ignores the anomaly. But the "σ *gap*" method correctly identifies the anomaly. See Figure 2.

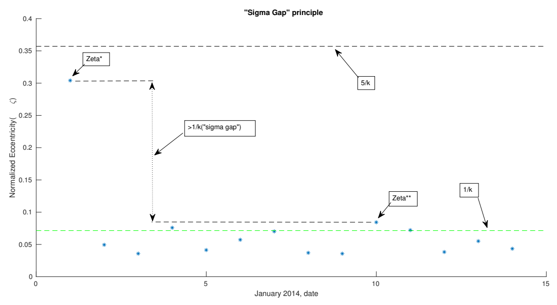

Fig. 8. Fig 2 The σ gap principle is illustrated on the simple 1D rainfall data from the first couple of weeks in South-West UK.

Here,

- k is the size of the data sample.
- ζ^* is the point with the highest normalized eccentricity.
- ζ^{**} is the point with the second highest eccentricity.

We introduced an artificial data on 14th January, 2014 with rainfall changed from 3.8 mm to 19.5 mm. The σ gap principle was still able to detect the two new anomalies. Here, both ζ^* and ζ^{**} are anomalies. See Figure 3:

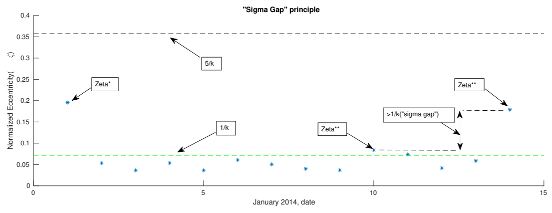

Fig. 9. Fig 3 The σ gap principle illustrated on the simple 1D rainfall data with artificial data on 14th January.

We were also able to detect anomalies in our own data, but for which we had to specify the sliding window size. See Figure 4:

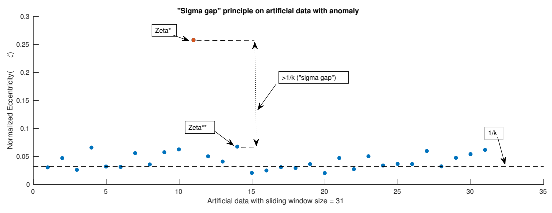

Fig. 10. Fig 4 The σ gap principle illustrated on the artificial data at the time when anomalies are created.

This illustrates that the "σ *gap*" principle can also be used for *data streams* with some minor changes.

IV. Results & Discussions

The two methods of detecting anomalies were evaluated on different data sets and it was found that each method had its own advantages and disadvantages.

The *anomaly detection based on membership* algorithm can detect anomalies during clustering. Thus there is no need to find anomalies during pre-processing of data. This can be advantageous with live streaming data which needs to be clustered online. But it has all the drawbacks which are associated with clustering algorithms such as the need to specify the number of clusters beforehand.

The σ *gap* principle does not need any presumptions on the data such as used in the traditional "$n\sigma$" approaches. It can find anomalies with datasets as small as 3 samples[1]. But to implement in on data streams, the window size needs to be pre-assigned. It also adds an extra step of computation. There could also be difficulties with live data streams where the data needs to clustered or classified online.

V. Application

To be able to showcase the properties of anomaly detection we have decided to make a python implementation of the TEDA framework and connect it to a data stream. For the data stream we have decided to use a recently popular application, object detection. Object detection is a very important problem, since it is one of the cornerstones of automated driving, and can be used also in CCTVs for surveillance. It is also essential in robotics for mapping new environments, identifying obstacles and objects the robot can interact with.

The goal of object detection is to find predefined entities in pictures, and provide bounding boxes for their localization. It is a step harder then object recognition, where there is only one item to be recognized on the picture, and hence there is no need to provide bounding boxes. Usually the base of an object detector is a neural network trained for image recognition. For the object detection we have chosen the recently published YOLOv2 architecture. It is a 14 layer convolutional neural-network, which excels not just in precision, but also is unrivaled in speed. It can compete in precision with the most robust models, and the same time can be run in real time on a low end graphics card. It is written in C++, but there is a third party python wrapper available. We decided to use a single simple object for the detection task, this object is a white sheet of paper held by a person. Later it can be used for augmented reality applications, for example text, images or even videos can be broadcaster on it.

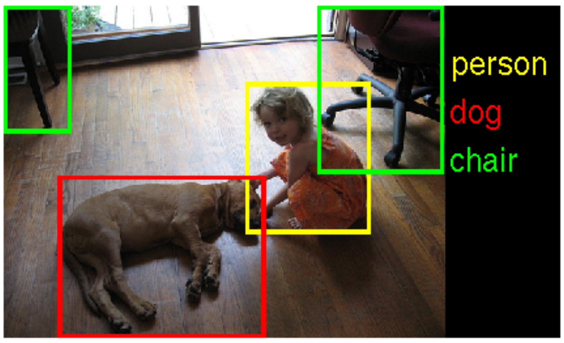

Fig. 11. Object detection in work

For training the neural network we have carefully collected 1000 sample images from the website reddit.com . We labeled the images manually, annotating them and assigning bounding boxes. After careful preprocessing we have trained several models in Google Cloud virtual machines using K80 GPUs. Training one model takes approximately 8 hours, using transfer learning with a model trained on the COCO dataset. Although the initial models were promising we had to discard 600 of the pictures because they were too ambiguous, and retrain the models. The resulting model is precise when the object is standing still, but during movement the bounding box coordinates become noisy and need to be stabilized.

Fig. 12. A sample image from the training batch

For filtering out the noise we have chosen the TEDA framework because of its simplicity and fast computation. On the other hand fuzzy k-means would have several advantages in this use case, for example the bounding box coordinates could be inputed without formatting, and the two resulting clusters would be the upper-left and lower-right corners. Also with fuzzy k-means the cluster centers could be used as final coordinates smoothing out the noise. Unfortunately its computation cost and algorithmic complexity did not allow for its usage.

The resulting object detection is more reliable, and much smoother with anomaly detection. Only one parameter has to be tuned which is n, used in the n/k threshold.

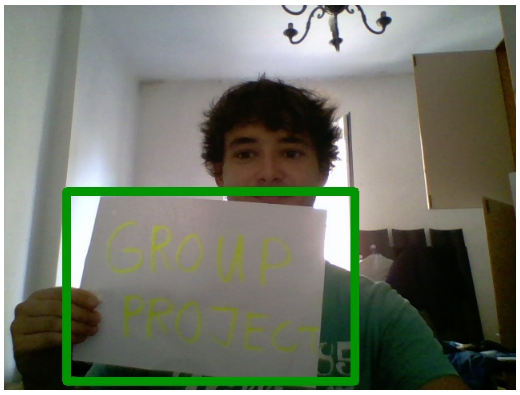

Fig. 13. Correctly detecting a sheet of paper

VI. Conclusion

Compared the two methodologies - replicated the results of Angelov's paper with our data(not done yet) - found anomalies in our data with Angelov's methods - how the methods differ - our method finds anomalies during clustering - good because no data manipulation or extra computation needed before starting clustering - Angelov's method requires extra computation before clustering - both find anomalies at different stages - with data streams it is better to use our method.

Acknowledgment

We would like to express our sincere gratitude to professor Stefano Rovetta for all his help and support. We also want to thank professor Fulvio Mastrogiovanni for the opportunity.

References

[1] Angelov, Plamen. ”Anomaly detection based on eccentricity analysis.” Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on. IEEE, 2014.

[2] Angelov, Plamen. ”Outside the box: an alternative data analytics framework.” Journal of Automation Mobile Robotics and Intelligent Systems 8.2 (2014): 29-35.

[3] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. ”Anomaly detection: survey.” ACM computing surveys (CSUR) 41.3 (2009): 15.

[4] Gama, Joo, et al. ”A survey on concept drift adaptation.” ACM Computing Surveys (CSUR) 46.4 (2014): 44.

[5] http://www.martynhicks.uk/weather/data.php?page=m01y2014

[6] Joseph Redmon, Ali Farhadi : YOLO9000: Better, Faster, Stronger