Online Cluster Drift Detection for Novelty Detection in Data Streams

Shon Mendelson
Department of Industrial Engineering and Management
Ben Gurion University of the Negev
Be'er Sheva, Israel
shonm@post.bgu.ac.il

Boaz Lerner
Department of Industrial Engineering and Management
Ben Gurion University of the Negev
Be'er Sheva, Israel
boaz@bgu.ac.il

***Abstract*—A major challenge in data stream applications is the change in the target variable over time in unexpected ways, a phenomenon called concept drift (CD). Another challenge is the emergence of novel classes, soliciting novelty detection (ND) by, e.g., one-class or semi-supervised classification. But, in online ND, these two challenges interfere with each other although they should be dealt with jointly. We present the cluster drift detection (CDD) algorithm that, using a single hyper-parameter, performs offline clustering to learn the diverse normal profile, and detects online whether a never-seen-before example is novel or normal using a multivariate statistical test. If it is normal, the CDD uses this example to update the normal-profile cluster, enabling continuous CD monitoring. Experimental results using popular real-world and synthetic data sets, as well as a precision agriculture data set of banana plants under water stress and a COVID-19 data set demonstrate that the CDD algorithm: 1) distinguishes between normal and novel concepts more accurately than state-of-the-art algorithms, 2) provides information about why specific novel concepts are misdetected, and 3) is more robust to the complexity, drift, and noise in the problem than other algorithms.**

***Index Terms*—Concept drift, Novelty detection, Streaming data**

I. Introduction

Most machine-learning methods assume examples are generated by a stable process, coming from a stationary distribution. But, this is not true in many real-life applications, especially in data streams. Assuming stationarity to deal with data streams, the ability to identify novel concepts, different from the normal profile, is challenging.

Outlier and novelty detection both aim at detecting anomalies, but the former goal is cleaning deviant examples from the data, and the latter goal is identifying new examples as novel to the data. Some works [1] do not distinguish between the two (calling them both "anomaly detection"): some originally proposed for outlier detection are also used for novelty detection [2], [3], while some can do the opposite [4].

Algorithms for novelty detection (ND) learn first (offline) the normal profile, and second (online) distinguish examples that fit the (normal profile) model from those that are do not (i.e., novel). Often, the normal profile should be represented by more than a single source [e.g., several types of consumers, each with its own distribution, make up the normal (transactions) profile in a fraud detection task]; hence, a single-source assumption can undermine the results. In the offline phase, there are only normal examples, so fewer parameterized or not well-optimized single-source algorithms may wrongly be preferred, and yet, hyper-parameter tuning and feature selection are neither trivial nor accurate. However, the ability to ignore irrelevant dimensions may be found essential. In the online phase, 1) data may be infinite, so the ND algorithm cannot expect to store all examples in memory, and 2) the normal profile distribution changes over time, so the algorithm must continuously update the model to keep its stable accuracy.

To address the above issues, we propose the cluster drift detection (CDD) algorithm that combines 1) clustering for deciphering the normal profile distribution; 2) Hotelling's T^2-based statistical test to detect concepts that are novel to all clusters and, thus, to detect drifts from the normal profile; and 3) online updating of the normal profile, enabling continuous concept-drift monitoring. In its first (offline training) phase, CDD clusters the normal profile data into homogeneous clusters, detecting sub-distributions in the domain, if they exist (if the profile does not contain meaningful clusters, CDD treats all the examples from the normal profile as one cluster from a single distribution). In its second (online) phase, the CDD algorithm detects whether a never-seen-before arriving example is novel based on its Hotelling's T^2 statistic (distance) to the clusters composing the normal model. If based on the statistic, the example cannot be associated to any of the clusters, it is defined as novel to the normal profile, but if it can be associated to any of the clusters, the statistics (mean and covariance matrix) of the nearest cluster are updated online, thereby adopting the normal profile to better monitor future concept drifts.

Our contributions are: 1) while ND clustering-based approaches [5] associate an example to a cluster based on the less-informative (arguable) Euclidean distance, the CDD associates examples statistically by accounting for the covariances and normalizing distances by standard deviations, exploiting variable interrelations to detect novelty; 2) while both modern ND approaches, e.g., deep auto-encoder [6], and traditional ones, e.g., statistical process control (SPC) [7], miss the rich complex distribution of the normal profile by assuming unimodality, the CDD decomposes the distribution into sub-group distributions that enrich domain representation;

and 3) while most ND approaches assume stationarity, the CDD dispenses with this assumption and addresses ND in data streams by updating online the model clusters of the normal profile as more data are seen.

Finally, we evaluate the CDD using 1) four ND applications including two streaming data sets for detecting intrusion in computer networks and fraud detection and two anomaly detection data sets; 2) experiments with synthetic data to demonstrate the CDD robustness to the number of sources in the normal profile, noise in the distribution, and level of concept drift; 3) a precision-agriculture data set describing the development of water stress in banana plants; and 4) a COVID-19 patient database.

Sect. II presents related work, Sect. III describes the CDD algorithm, Sects. IV–VII present its empirical evaluation, and Sect. VIII concludes and suggests future research directions.

II. RELATED WORK

Most ND approaches are static, e.g., kernel-based [8], [9], clustering-based [10], [11], based on density estimation [2], [12], or SPC [7], [13]. Another approach is isolation forest (IForest) [3] that isolates anomalies, rather than learns the normal profile, using decision trees that identify novelties. A recent deep auto-encoder (AE) [6], [14] learns hidden representations of the normal profile, and by the reconstruction error distinguishes it from novel concepts.

Streaming ND can be either multi-class [15]–[17] or one-class. OLINDDA [5] is a one-class streaming ND algorithm that represents the normal profile by clusters in the data distribution, where the boundaries of each are computed by the distance between the centroid and the example farthest from it, and the union of the clusters' boundaries defines the model's boundaries. MINAS [17] extends OLINDDA for the multi-class scenario, learning decision boundaries for each class separately. Another one-class streaming ND algorithm is streaming half-space trees (HST) [1], an ensemble of decision trees with random generated rules. HST is very efficient in terms of time and memory complexity, but it has difficulties to ignore irrelevant dimensions and requires input to be in [0,1], which is hard to guarantee, especially in data streams. Finally, lightweight on-line detector of anomalies (LODA) [18] is an online outlier detection algorithm that uses an ensemble of one-dimensional histograms to detect anomalies.

III. ONLINE CLUSTER DRIFT DETECTION

Figure 1 (top) depicts the CDD algorithm. In the first phase [Figure 1 (top-left)], it learns the normal profile distribution by clustering the normal examples (Sect. III-A). In the second phase (Sect. III-B), any new example arriving online is considered (by Hotelling's T^2 test) for association to any of the clusters of the normal profile [Figure 1 (top-middle)], and is associated with the nearest cluster (top-right) unless declared novel (black points) because it does not statistically fit any of the clusters. Associated examples are then used to update online the parameters of the normal profile [Figure 1 (bottom); see Sect. III-C].

A. *Learning the Normal Model*

To decide if the normal profile distribution is based on one or more sources, the latter case soliciting clustering, CDD uses the Hopkins statistic [19] for randomness. The statistic measures the difference between uniformly distributed data and a random subset of the original data. If the statistic is greater/smaller than an accepted threshold of 0.5, then the null hypothesis of ”no difference” can/cannot be rejected. If rejected, cluster tendency is declared, and clustering is applied to the data, whereas if it cannot be rejected, the normal profile is considered as based on a single cluster. Any clustering method can be used, but we recommend the Gaussian mixture model (GMM) [20] since it considers the covariances, which are needed in the online phase, unlike, e.g., K-means. Cluster validity (i.e., setting the number of clusters) is determined by the maximal value of the silhouette coefficient [21]. For each cluster, the CDD estimates its centroid and covariance matrix, and counts its number of examples, which are used in the online phase.

B. *Online Novelty Detection vs. Example Association*

After establishing the normal profile distribution, the CDD detects novel concepts in the stream through the multivariate Hotelling T^2 statistical test. This statistic measures the Mahalanobis distance of X_i, the i^{th} new example, from the k^{th} of K clusters, having centroid (sample mean) C_k, covariance matrix S_k, and number of examples M_k:

$$T^2 = (X_i - C_k)^t * {S_k}^{-1} * (X_i - C_k). \tag{1}$$

This statistic, multiplied by a constant, is F-distributed with P and $M_k - P$ degrees of freedom [22]:

$$T^2 * \frac{M_k * (M_k - P)}{P * (M_k - 1) * (M_k + 1)} \sim F_{P,M_k-P}, \tag{2}$$

where P is the number of features (identical for all clusters). The upper control limit (UCL) of the k^{th} cluster is:

$$UCL_k = \frac{P * (M_k - 1) * (M_k + 1)}{M_k * (M_k - P)} * F_{\alpha,P,M_k-P}, \tag{3}$$

where F_{α,P,M_k-P} is the $(1-\alpha)^{th}$ quantile of the F_{P,M_k-P} distribution. A statistic T^2 (1) for X_i to be associated with cluster k that is greater than UCL_k (3) indicates that X_i cannot be associated with k, and thus is defined as ”novel” (T^2 is non negative by definition; thus, the lower control limit for all clusters is equal to 0). Since this test for clusters with $M_k \leq P$ has non-positive degrees of freedom, making the F-distribution meaningless, we apply the test only for clusters here $M_k > P$, which are then defined as valid.

While a new example that cannot statistically fit (based on the T^2 test) any of the valid clusters is defined ”novel” for all clusters, a new example that can be associated with at least one of the clusters is included in the normal model and either is: 1) associated with the nearest cluster (”hard association”) or 2) probabilistically associated with the nearest cluster (”soft association”). Associated probabilistically, the probability is one minus that of the new example to be novel (the ”anomaly

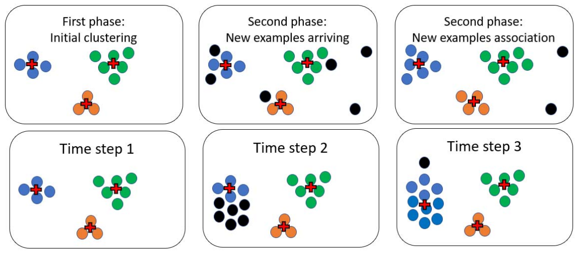


Fig. 1: CDD association (top) and parameter update (bottom).

score"), i.e., the probability to not fit the nearest valid cluster (by the T^2 test), which is $P(F_{st} > F_{P,M_k-P})$, F_{st} is the statistic [left expression in (2)]. We use hard association when evaluating the algorithm detectability between normal and novel examples, and soft association ("anomaly scores" for all examples) when we compare the association probability with different thresholds (to find a balance between Type I and Type II errors), e.g., when computing the area under curve (AUC) in Sect. IV. Then, after associating an example to a cluster, the cluster statistics are updated online (Sect. III-C).

C. *Online Update of the Normal Profile*

The CDD periodically updates the normal profile clusters' statistics to ensure that clusters represent the current normal profile to detect future drifts. In an illustration of the update mechanism [Figure 1 (bottom)], Step 1 represents initial clustering into three clusters. In Step 2, new examples are associated with the blue cluster, updating its centroid to move down. Due to this update, a novel example in Step 3 (the black point) is not erroneously detected as normal, but as novel.

Parameters are updated for cluster k independently of other clusters once it grows by $\pi * M_k$ examples, $0 < \pi < 1$. Hyper-parameter π controls the update timing to maintain stable process, which is especially important as the online association is based on a statistical test and not on ground truth labels. If, on the one hand, we update the cluster statistics with every new example arriving, the chance increases that false positive (identifying normal as novel) and false negative (identifying novel as normal) errors will follow each other, and no real improvement in drift detection will be achieved. If, on the other hand, we update the cluster statistics only after a (large) batch of new examples arrives, then the normal profile will not be updated frequently enough, and once updated, it is more likely that the update will be based mostly on normal profile examples, which are the majority class, undermining novelty detection. Although π could be set separately for each cluster, for simplicity, it is set here as identical for all clusters.

Once needing updating, a cluster centroid and covariance matrix are updated

$$C_k[t] := w * C_k[t] + (1-w) * C_k[t-1] \tag{4}$$

$$S_k[t] := w * S_k[t] + (1-w) * S_k[t-1], \tag{5}$$

where w is the ratio of the number of examples added to a cluster, $\pi * M_k$, to the number of examples in the updated cluster, $(\pi+1) * M_k$. Once used to update cluster parameters, updating examples are deleted from memory. If not valid because $M_k \leq P$ (Sect. III-B), cluster k can become valid once $M_k > P$ following some updates.

1) Memory Complexity: This is $\mathcal{O}(K*P^2 + P*\pi*M_{norm})$, where M_{norm} is the number of examples of the normal profile. The first term stands for the $P \times P$ covariance matrix for each of the K clusters, and the second stands for the $\pi * M_{norm}$ P-dimensional updated examples. Since following update, the number of examples belonging to the normal profile grows with time, we bound this number, saved in memory, by C, enabling the CDD to deal with streaming data. Once C examples have been saved in memory, 10% of the oldest examples in the largest cluster are removed, allowing the CDD to save fresh arriving examples in memory, further updating.

2) Cluster Merging: Once cluster i has been updated, CDD tests whether it has become too similar to cluster $j \neq i$, soliciting cluster merging. Hypothesis

$$H_0 : C_i = C_j \quad H_1 : else$$

is tested using a two-sample statistic [23],

$$T^2 = \frac{n_i * n_j}{n_i + n_j} * (C_i - C_j)^t * S^{-1} * (C_i - C_j), \tag{6}$$

where the estimated pooled covariance matrix is,

$$S = \frac{(n_i - 1) * S_i + (n_j - 1) * S_j}{n_i + n_j - 2}, \tag{7}$$

and n_i and n_j are the sizes of the clusters. When multiplied by a constant, the statistic follows an F-distribution with P and $n_i + n_j - 1 - P$ degrees of freedom [22]:

$$T^2 * \frac{n_i + n_j - 1 - P}{(n_i + n_j - 2) * P} \sim F_{P, n_i + n_j - 1 - P}. \tag{8}$$

CDD applies this test for the i^{th} cluster against all the other clusters. If the null hypothesis is not rejected for the j^{th} cluster, CDD merges the two, and the parameters of the new, merged cluster are estimated with a weighted mean for both the centroid and covariance matrix. For simplicity and conventionally, CDD uses the same α (the test significance level) as for the SPC test (3). This test is taken only when $n_i + n_j - 1 - P > 0$ because, otherwise, the statistic has non-positive degrees of freedom, which makes the F-distribution meaningless.

IV. EXPERIMENTS: REAL-WORLD DATA SETS

In this section, we compare the CDD to other ND algorithms using the AUC and two streaming data sets and two anomaly detection static data sets. The first data set is of detecting intrusion in computer networks, which is a streaming benchmark in evaluating anomaly detection (both supervised and semi-supervised), and the second is a huge fraud detection data set that can be used as streaming data since the data are "infinite".

Duplicate examples, categorical features with too many categories, and zero variance features were removed. Categorical features were transformed using one-hot encoding to insure that the distance between each two categories is equal. Numeric features of training examples were Z-scored, and the corresponding features of the test examples were scaled based on the means and variances computed in the training. That is, mean $\mu_j(train)$ and standard deviation $\sigma_j(train)$, computed for feature j using the training set, scaled this feature in the i^{th} test example $X_{ij}(test) = [X_{ij}(test) - \mu_j(train)]/\sigma_j(train)$. For HST, we used min-max scaling to ensure that the features are bounded in [0,1], as HST requires.

The CDD was implemented in Python. The Scikit-Learn package [24] was used for GMM clustering. $\alpha = 0.05$ (3) and $\pi = 0.1$ (Sect. III-C) were default hyper-parameters.

A. Data Sets and Evaluation Metrics

For detecting intrusion in computer networks, we used the 10% version of the KDD Cup 1999 data set [25] from the *UCI*. This was originally created for supervised classification tasks; thus, its training set consists of both normal traffic and attacks. Focused on ND, we used only the normal traffic examples for training (87,832 examples). The test consists of both normal traffic (47,913 examples) and intrusions of four classes/attacks: denial of service attack (DOS), user to root attack (U2R), remote to local attack (R2L), and probing attack with 21,765, 39, 2,328, and 1,269 examples, respectively.

For fraud detection, we used the *Synthetic Financial Dataset for Fraud Detection* provided by Kaggle and created by the PaySim mobile money simulator [26]. Each example has a time step that maps a unit of time in the real world, e.g., Step 1 represents the first hour. Since the data set was not split into training and test sets, we chose all first time steps of normal transactions as the training set (2,692 examples), and the remaining steps as the test set (6,362,620 examples, 0.128% of them fraudulent).

The two additional data sets (OpenML [27]) are not used in streaming, but are used for anomaly detection. The first is page-blocks, a document analysis task, in which $4,913$ blocks of text, which are the majority class (89.8%), establish the normal profile, whereas 560 graphic areas of classes horizontal lines, pictures, vertical lines, and graphics are labeled as anomalies (10.2%). The second data set is Satellite, where the task is to identify two classes of vegetation as abnormal (with 75 anomalies; 1.49%) to a normal profile of four soil classes ($5,025$ normal examples). For both data sets, we used 200 randomly normal examples for training and the rest for testing.

B. First Phase (offline): Building the Normal Model

To decide if there is more than one distribution in the data set, which solicits clustering, we computed the Hopkins statistic (Sect. III-A), which was about 1, 0.97, 0.94, and 0.81 for the intrusion, fraud detection, page-blocks, and Satellite training sets, respectively – all higher than the conventional 0.5 threshold, indicating the existence of meaningful clusters in these data sets.

To avoid overfitting (reflected in too many unnecessary clusters), the optimal number of clusters must be defined

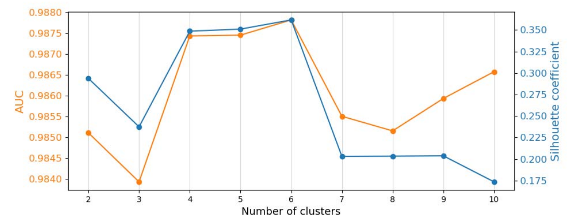


Fig. 2: Intrusion Detection for Different Numbers of Clusters.

according to the training set size (in a positive correlation) and the feature dimension (in a negative correlation), since CDD defines valid clusters as those with more examples than features. Empirically, we observed that the number of clusters which maximizes the silhouette coefficient (i.e., our cluster validity in accordance with Sect. III-A) is much smaller than the maximum number according to this intuition. Figure 2 shows a positive correlation between the silhouette coefficient (blue) and the CDD AUC value (orange) for different numbers of clusters in the initial clustering for the intrusion detection data set (both recommending four to six clusters), where the picture is similar for the fraud detection data set. For automatic model selection (Sect. III-A), in the experiments we used the number of clusters that maximized the silhouette coefficient, which for the the intrusion, fraud detection, page-blocks, and Satellite data sets is six, five, three, and two, respectively.

C. Second Phase (online): Novelty Detection

For the experiments, we used both state-of-the-art static and streaming ND algorithms. One static competitor is Hotelling's T^2, a representative of the SPC approach to ND. Because the CDD and Hotelling's T^2 use the same statistic, but the CDD also uses clustering, this comparison examines the contribution of clustering to ND. Other representatives of the static state-of-the-art ND algorithms are: auto encoder (AE), one-class SVM (OCSVM) [8], local outlier factor in ND mode (LOF) [2], and isolation forest (IForest) [3]. For the last three, we used the Scikit-Learn package, and for the first algorithm PyOD [28]. The streaming ND competitors are HST and LODA. For the first one, we used the Scikit-multiflow package [29] and for the second PyOD [28].

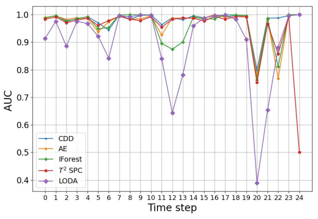
For a fair comparison between the two streaming data sets, fraud and intrusion detection, for each algorithm, we tested several values of hyper-parameters. In almost all cases, the default parameters were the best. The only exception was for the intrusion detection data set and the LOF algorithm that significantly benefited from 200 neighbors more than from the default value of 10. For the fraud detection data set, we could not increase the number of neighbors for LOF or the number of trees in IForest due to memory issues. For HST, in the intrusion detection data set, the optimal setting was 200 trees, a depth of 15, and window size of 20,000. For the fraud detection, the default values of 25 trees, a depth of 10, and window size of 2,000 yielded the best results. For LODA and the intrusion detection data set, increasing the default numbers of bins from 10 to 100, random cuts from 100 to 500, and contamination from 0.1 to 0.3 significantly improved the results, but also significantly extended the run time. Thus, for the huge fraud detection data set, we used the default values for the numbers of bins and random cuts (both affecting the run time). For the smaller two static data sets, we used the default hyper-parameters for all the algorithms. For the AE, the main default parameters were: hidden neurons–64,32,32,64, hidden activation–ReLU, output activation–Sigmoid, optimizer–Adam, and dropout rate (same for all layers)–0.2. For the IForest, the main default parameters were: number of trees–100, maximum examples to draw each tree–the minimum between 256 and the number of training examples, and no limitation on the maximum number of features per tree. To enable nonlinear mapping for the OCSVM, we used a radial basis function kernel. For the Hotelling T^2, we used our CDD implementation. The CDD, IForest, AE, HST, and LODA are randomly initialized; hence, we ran them ten times with different random seeds and reported the means of these ten runs.

TABLE I: AUC for real-world data sets

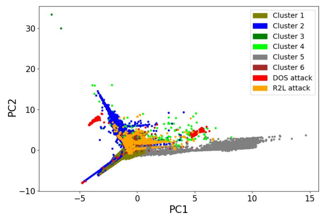
Algorithm	Fraud	Intrusion	Page-blocks	Satellite	Average
CDD	**0.898**	**0.987**	0.933	**0.949**	**0.942**
LODA	0.590	0.957	0.858	0.771	0.794
HST	0.770	0.949	0.860	0.844	0.855
AE	0.830	0.980	0.930	0.898	0.909
IForest	0.769	0.985	0.907	0.948	0.902
OCSVM	0.744	0.970	0.932	0.930	0.894
LOF	0.631	0.977	**0.965**	0.938	0.878
T^2	0.879	0.980	0.939	0.942	0.935

Table I shows that for three of the four data sets, the CDD achieved the best AUC results, and for the fourth, it was third after LOF and T^2. For the intrusion detection data set, all of the algorithms achieved high results, but LODA and HST were the lowest, although both are streaming. The window parameter, which controls the frequency of the update of the HST algorithm, that gave the best results was very large, which might suggest that the HST did not benefit from its update mechanism for the intrusion detection data set. For the fraud detection data set, LODA and LOF achieved the worst results, perhaps because their hyper-parameters could not be optimized due to memory and run time issues. Although all of the randomly initialized algorithms achieved stable results (std values are not shown), the CDD was the least noisy (0% std) for all data sets besides page-blocks (0.11% std).

Further analysis for the intrusion detection data set shows in Figure 3a the AUC results of most of the algorithms over time in steps of $3,000$ examples (we omitted inferior algorithms to make the figure more readable). While the LODA was inferior almost always, most algorithms are similar most of the time with an advantage to CDD. Besides at a single time step (6), CDD is always the best or among the best algorithms, showing stable performance. We relate the drop in performance of almost all algorithms from time step 20 to the increasing proportions of anomalies from this step.



(a) AUC Over Time.



(b) Error Analysis.

Fig. 3: Analyses for the Intrusion Data Set.

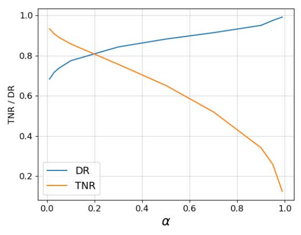
Drilling down the CDD performance for the intrusion detection data set reveals an inferior detection rate (DR) for some of the attacks. For example, while CDD and T^2 had similar DRs for the DOS attacks, CDD was inferior regarding the R2L attacks. Figure 3b shows, using the first two principal components [30], clustering of the normal profile transactions into (the optimal number of) six clusters. The figure also shows a random subset of the R2L (orange points) and DOS (red points) attacks. While most of the DOS (red) points do not overlap any of the normal profile clusters, which explains the high DR for this attack, R2L examples overlap some clusters, indicating that the R2L attack examples are not too statistically different from some of the normal transaction clusters, which explains the errors the CDD makes for this attack.

Run time is crucially important in streaming algorithms in the online phase since novelty detection must be as fast as possible. We tested run time of these algorithms using their default hyper-parameters (which did not necessarily yielded the best run time, but were optimized for best accuracy) and using the intrusion detection data set. The HST was the fastest algorithm with an average processing time per example (including both evaluation and update mechanism) of 0.00189 seconds, the CDD was second with 0.0144 seconds, and LODA was third with 0.0156 seconds.

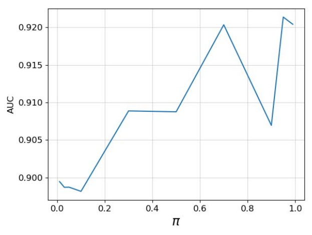
D. Sensitivity Analysis

To examine the effect of its two hyper-parameters, we tested the CDD in two experiments using the fraud detection data set.

Significance Level Effect (α). Detection tasks involve a trade-off between the DR and true negative rate (TNR) performances. The CDD trades off these performances by controlling the significance level (3), and Figure 4a shows this for levels from 0.01 to 0.99. As the figure demonstrates, a high significance level decreases the TNR, but increases the DR, in accordance with the decrease in the UCL [as in (3), the UCL decreases with the decrease of the F value, which decreases with the increase of α)], and vice versa. The choice for α depends on user preferences. For scenarios where the DR and TNR are equally important, we recommend setting α at 0.05-0.1 (as is customary in most statistical tests), and then there is no need to optimize α. However, if the DR is more important than the TNR, we recommend increasing α, and vice versa.



(a) Significance Level Effect.



(b) Update Parameter Effect.

Fig. 4: Hyper-Parameter Effects.

Update Parameter Effect (π). Figure 4b shows the effect of the update parameter (π) on the AUC for π values ranging from 0.01 to 0.99 (note that AUC values for the full range of π values are between 0.9 and 0.92 with changes of an order of 0.01). The AUC increases until a certain point, which is reasonable, since as long as the update mechanism is based on more examples, it will be more reliable (Sect. III-C). On the other hand, setting too high a π value (not shown in the figure) causes the update mechanism to take place only rarely. For example, if we set π to be 10 for a cluster of 50,000 samples, update will occur only when 500,000 examples are associated with this cluster, which can take too long, if at all. Although the best π value may change from task to task, based on our empirical experience, we recommend setting $\pi \geq 0.1$, where $\pi = 0.1$ is the CDD default value (used throughout this paper).

V. EXPERIMENTS: SYNTHETIC DATA SETS

In this section, we evaluated the algorithm sensitivity to the number of sources in a multi-source distribution of the normal profile, level of concept drift, and number of irrelevant dimensions. We used the Scikit-Learn package (with the *make-blobs* function) to generate two-dimensional data of sources with a standard deviation of 0.5.

Number of Sources. We gradually increased the number of sources in the normal profile distribution from one with a centroid of $(2,2)$ to 8 with centroids $(2,2),(-2,-2),(2,-2),(-2,2),(6,6),(-6,-6),(6,-6),$ and $(-6,6)$), and in each case, uniformly sampled 1,000 training examples from the increased number of sources. For the test set, we sampled 10,000 examples, 99% from the normal profile and 1% uniformly in $[-8,8]^2$, the latter used as the ground truth anomalies. Figure 5(a) shows that while the CDD, OCSVM, and LOF experience only a slight drop in their AUC performances, AUC values drastically drop for AE and T^2, already for two sources, and for IForest, HST, and LODA from five sources.

Level of Concept Drift. For training, we sampled 1,000 examples from two clusters with centroids of $(2,2)$ and $(-2,-2)$. For testing, we gradually shifted the centroid of the first cluster to $(2+d, 2+d)$, where d is the drift level, and sampled 10,000 examples, 99% from the (training) normal profile distribution and 1% uniformly in $[-8,8]^2$, the latter used as the ground truth anomalies. Figure 5(b) shows that IForest, AE, and LODA are the most sensitive algorithms to the concept-drift level, and T^2 and HST are almost insensitive to this level, but are less accurate. Figure 5(b) also shows that the CDD, IForest, OCSVM, and LOF achieve similar high results for $d = 0$ (no drift), where as the AE and T^2 show inferiority [this was already seen for this case of two centroids $(2,2)$ and $(-2,-2)$ in the first experiment, i.e., the second point on the graph in Figure 5(a)]. But, while IForest, OCSVM, and LOF present a withdrawal in their AUC values with the drift level (for IForest, AUC was significantly worse from $d = 0.4$, where for OCSVM and LOF, AUC values were significantly worse from $d = 0.6$), the CDD keeps high stable performance regardless of the drift level.

Number of Irrelevant Dimensions. While feature selection is elementary in supervised learning, it is more challenging in ND, since the training set consists of only one-class examples. Therefore, we challenged the ND algorithms with increasing numbers of irrelevant dimensions. The relevant dimensions were the first two, where for the training set, we sampled 1,000 examples from two clusters with centroids of $(2,2)$ and $(-2,-2)$, and for the test set, we sampled 10,000 examples, 99% from the normal profile and 1% uniformly distributed in $[-8,8]^2$, the latter used as the ground truth anomalies. Then we added to each example $d = [1,8]$ irrelevant dimensions sampled uniformly in $[-1,1]^d$. In this way, we get examples in $2+d$ dimensions, where only the first two are relevant to determine if an example is normal or novel. Figure 5(c) shows that IForest and HST are the most sensitive to the irrelevant dimensions, probably since they both generate random rules (i.e., randomly select variables for a tree), which may be devastating as the numbers of irrelevant dimensions (variables) increases. The AE and LODA are less sensitive to these dimensions, but are also less accurate. The CDD, OCSVM, LOF, and T^2 show stable and similar results, where the CDD outperforms the other three.

VI. WATER-STRESS EXPERIMENT

As part of a precision agriculture project, with an agro-biotechnology partner, we tested the ability of the CDD algorithm to detect novelty in the form of water stress in greenhouse banana plants. The experiment was conducted with four treatment schemes of watering, each applied to another 30 plants, that simulated different water stress conditions. In the first three days of the experiment, all 120 plants were equally watered by an agronomist, and in the next 14 days, watering was done according to four schemes: A–100%, B–80%, C–60%, and D–40% of the water quantity of the first three days (also the 100% level was elevated through the experiment to provide the plants with a water level that suited their development stage, and the 80-40% were derived from this 100%). While the first three days were the "in-control" stage (normal profile), the next 14 days for schemes B–D were the "out-of'control" stage.

Measurements taken to characterize the plant condition on each day were the amounts of water entering and leaving the

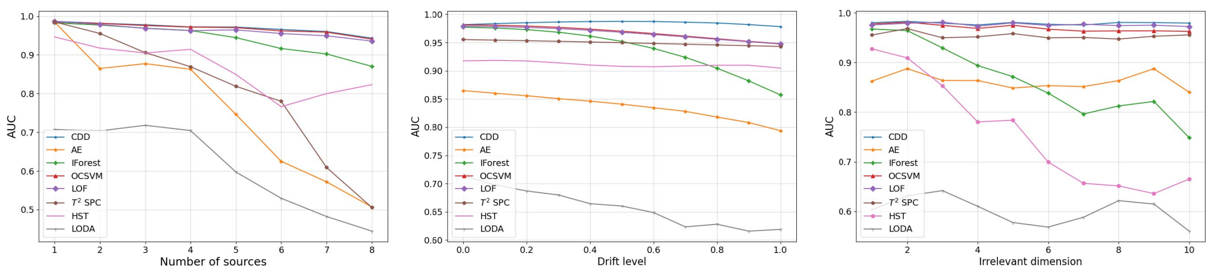


Fig. 5: Evaluation of the effects of the: (a) number of sources, (b) level of concept drift, and (c) number of irrelevant dimensions.

plant and their electrical conductivity levels, (greenhouse) outside temperature, chlorophyll levels, plant size, and maximal and average plant temperatures (the three last measurements were captured from an infra-red image of the plant, aided by an agronomist's annotation of the plant in the image). Feature scaling followed that described in Sect. III.

We used all 360 examples from the first three days as the "in-control" points, as all plants during this period received the same amount of water regardless of the treatment scheme, thereby establishing the training set (normal profile). All 1,680 examples from the following 14 days served as the test set. For these days, and according to the agricultural protocol, we considered the examples of scheme A as normal (i.e., plants watered with the optimal water amount), and all other examples as novel (i.e., water amount was less than optimal) in order to evaluate detection of a novel condition of water stress.

For comparison, we used the algorithms described before (Sect. IV-C) with their default hyper-parameters. The CDD achieved the highest AUC value of 0.906, followed by LOF, and T^2 with AUC values of 0.872, and 0.867, respectively. Therefore, the CDD is most likely the best water-stress detector.

VII. COVID-19

The COVID-19 pandemic started in December 2019 and has been spreading around the world. Many countries were, and still are, in lockdown with the aim to slow down the spread of the disease. One of the biggest problems in many places is that the number of ventilators is limited. Hence, predicting whether a sick person will deteriorate is an important task. One of the major challenges in doing this is that most of the sick people easily recover. Therefore, learning a model of the patient who will recover is a reasonable idea (i.e., one-class classification).

We used a Kaggle open data set which included COVID-19-confirmed patient data such as age, sex, symptoms, and outcome (which may be either discharged/death or empty if a patient does not have an outcome). From the symptoms, we derived for each patient binary variables indicating whether they have fever, cough, and fatigue (one binary variable for each symptom).

Since most of the patients do not have an outcome, learning a one-class classifier (such as the CDD) is not trivial, and for this reason, we designed an unsupervised version of the CDD. As a first stage, we used only examples without an outcome as the training set and learned clusters in the distribution (similar to the CDD first stage). For testing, we used examples with an outcome:discharged or death. We used a validation set, which is a subset of the test set, and for each cluster, computed the average Hotelling's T^2 distance for both outcome groups. Since the training set should depict the normal distribution (and luckily the discharged group is the most common, hence, the normal one), each cluster whose average Hotelling's T^2 distance to the group of people who died was smaller than the discharged group was removed. Finally, we evaluated the CDD performance on the entire test set. In the test set, 67 of the patients were discharged and 52 of them (78%) were correctly identified as normal by the CDD, and eight of the eleven patients who died were correctly detected as anomalies by the CDD (72%). We believe that with a larger richer database, which will probably be available soon, CDD could improve this performance.

VIII. CONCLUSION

Data-stream analysis is not a trivial task, especially in scenarios where novel concepts appear or drift over time. We presented the CDD algorithm—an ND algorithm that harnesses clustering and the Hoteling's T^2 statistic to detect novelty, while continuously updating the normal profile in order to be sensitive to further concept drifts.

Compared with other static and streaming ND algorithms, e.g., those based on trees, one-class SVM, statistical process control, or auto-encoder neural networks, CDD includes only two hyper-parameters to choose, which is an advantage for a semi-supervised algorithm (practically, CDD relies on only one hyper-parameter, π, because the significance level, α, is conventionally, and as supported empirically here, set at 0.05).

As our error analysis for the intrusion detection data set demonstrated, the CDD clustering mechanism enables exploration of the source of errors in novelty detection. Such analysis can promote applying additional rules or detection mechanisms to identify examples previously associated erroneously with the normal profile as novel, an ability that is lacking in other ND algorithms.

The results for the intrusion and fraud detection streaming data sets suggest that the CDD can distinguish well between the normal profile and novel concepts, better than state-of-the-art ND algorithms. Since the former data set is large and the latter is small, this superiority may suggest overall superiority of the CDD. Also for the static data sets, CDD succeeded. Demonstrated using synthetic data sets, the CDD was more robust to the number of sources in the distribution, level of concept drift, and number of irrelevant dimensions in the normal profile than any of the other state-of-the-art ND algorithms. Also, CDD was shown to detect water stress in banana plants more accurately. Finally, applied to a small COVID-19 database, an unsupervised version of the CDD algorithm succeeded in detecting most of the patients who were later either discharged or died, giving some hope to the role ND algorithms may play in combating the COVID-19 pandemic.

Future work could be directed to alleviate the CDD time complexity, as the algorithm consists of multivariate statistical tests. This can be performed, e.g., by not considering the full covariance matrix. Also, better, less extensive update mechanisms can be suggested. In addition, evaluation of different types of concept drift (e.g., abrupt and gradual) is solicited.

Finally, the code for the CDD and data sets used in our experiments are available *online*.

References

[1] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[2] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[3] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[4] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems*, 2000, pp. 582–588.

[5] E. J. Spinosa, A. P. de Leon F de Carvalho, and J. Gama, "Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks," in *Proceedings of the 2008 ACM Symposium on Applied Computing*. ACM, 2008, pp. 976–980.

[6] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 665–674.

[7] M. L. Fugate, H. Sohn, and C. R. Farrar, "Vibration-based damage detection using statistical process control," *Mechanical Systems and Signal Processing*, vol. 15, no. 4, pp. 707–721, 2001.

[8] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, 2001.

[9] H. Hoffmann, "Kernel PCA for novelty detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863–874, 2007.

[10] N. Fanizzi, C. d'Amato, and F. Esposito, "Conceptual clustering and its application to concept drift and novelty detection," in *European Semantic Web Conference*. Springer, 2008, pp. 318–332.

[11] Y.-J. Cha and Z. Wang, "Unsupervised novelty detection-based structural damage localization using a density peaks-based fast clustering algorithm," *Structural Health Monitoring*, vol. 17, no. 2, pp. 313–324, 2018.

[12] M. Desforges, P. Jacob, and J. Cooper, "Applications of probability density estimation to the detection of abnormal conditions in engineering," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 212, no. 8, pp. 687–703, 1998.

[13] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality and Reliability Engineering International*, vol. 17, no. 2, pp. 105–112, 2001.

[14] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, 2015.

[15] M. M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham, "Addressing concept-evolution in concept-drifting data streams," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 929–934.

[16] T. Al-Khateeb, M. M. Masud, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham, "Stream classification with recurring and novel class detection using class-based ensemble," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 31–40.

[17] E. R. de Faria, A. C. P. de Leon Ferreira Carvalho, and J. Gama, "MINAS: multiclass learning algorithm for novelty detection in data streams," *Data Mining and Knowledge Discovery*, vol. 30, no. 3, pp. 640–680, 2016.

[18] T. Pevnỳ, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, no. 2, pp. 275–304, 2016.

[19] B. Hopkins and J. G. Skellam, "A new method for determining the type of distribution of plant individuals," *Annals of Botany*, vol. 18, no. 2, pp. 213–227, 1954.

[20] D. A. Reynolds, "Gaussian mixture models." *Encyclopedia of Biometrics*, vol. 741, 2009.

[21] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[22] S. Bersimis, S. Psarakis, and J. Panaretos, "Multivariate statistical process control charts: an overview," *Quality and Reliability Engineering International*, vol. 23, no. 5, pp. 517–543, 2007.

[23] H. Hotelling, "The generalization of student's ratio," in *Breakthroughs in Statistics*. Springer, 1992, pp. 54–65.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[25] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD Cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 1–6.

[26] E. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection," in *28th European Modeling and Simulation Symposium, EMSS, Larnaca*. Dime University of Genoa, 2016, pp. 249–255.

[27] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 2, pp. 49–60, 2014.

[28] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

[29] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2915–2914, 2018.

[30] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.