The 34th Canadian Conference on Artificial Intelligence
Canadian AI 2021
(Vancouver, May 25-28, 2021)

Incremental Learning with Self-labeling of Incoming High-dimensional Data

Farzana Anowar[†,*], Samira Sadaoui[‡],
[†] anowar@uregina.ca, University of Regina, Regina, Canada
[‡] sadaouis@uregina.ca, University of Regina, Regina, Canada

Abstract

Many incoming data chunks are being produced each day continuously at high speed with soaring dimensionality, and in most cases, these chunks are unlabeled. Our study combines incremental learning with self-labeling to deal with these incoming data chunks. We first search for the best data dimensionality reduction algorithm, leading to the optimal low-dimensional space for all the incoming chunks. The incremental classifier is then adapted gradually with chunks that are optimally reduced and self-labeled. Using a highly-dimensional and multi-class dataset, we conduct several experiments to demonstrate our incremental learning approach's efficacy and compare it with incremental learning using human-annotated labels.

Keywords: incremental learning, self-labeling, incoming data, high dimensional data, dimensionality reduction, feature extraction

1. Introduction

For monitoring risks and specific events of individuals, their physiological information is crawled continuously or periodically from sensor devices. Incremental learning is vital for this particular context in order to respond to incoming data in real-time and learn from the new data efficiently. Numerous studies showed the dominance of incremental learning to static learning in terms of time-efficiency and predictive accuracy [1–4]. Incremental classifiers extend their knowledge without re-training from the scratch, i.e., without accessing the previous training data [2], which saves a great deal of time. The most crucial contrast between incremental and static learning is that the former does not necessitate the accessibility of sufficient training data; rather, the incremental classifiers receive data over the time and learn gradually [3].

Even though many observations can be gathered, only a segment is helpful for particular decision-making and prediction tasks. In fact, when the dimensionality of datasets increases, the classification performance decreases, which is due to the participation of similar and insignificant features [5]. Dimensionality Reduction Algorithms (DRAs) focus on the curse of dimensionality by reducing data complexity and improving data quality. DRAs are primarily classified into feature selection (FS) and feature extraction (FE). FS retains only the most significant and informative features, whereas, FE combines similar features into unique features via algebraic transformations [6]. Our study investigates FE methods since they can better tackle the complexity, noise, sparsity, and non-linearity of real-life datasets [7]. These methods are in turn classified into supervised and unsupervised. We explore only the unsupervised DRAs because incoming data are unlabeled.

Nevertheless, incremental learning assumes that data is already labeled, which is not the case in the setting of new data that arrive over time. To solve the issue of unlabeled data, the study [8] suggested adopting semi-supervised learning with self-training. Hence,

---

[*] anowar@uregina.ca

our study combines self-labeling with chunk-based incremental learning, which is essential for classifying and learning efficiently from incoming unlabeled data. In the situation of highly dimensional chunks, we search for the best DRA method, leading to the optimal low-dimensional feature space. To determine the optimal DRA, we evaluate multiple representative FE methods and assess the transformed datasets' quality using two correlation metrics, the statistical significance and power analysis, and run-time as well. Based on the metric values, we select the optimal feature set that we utilize for all the incoming chunks. The classifier is then adjusted gradually using chunks that are first optimally reduced and then self-labeled. We assess our approach's performance through a highly-dimensional (561 features) and multi-class (6 classes) dataset. We prepare the dataset for the incremental learning process by organizing it into several chunks: one initial training chunk and several unlabeled chunks. To guarantee the reduced chunks' quality, we compare the new and original chunks using the correlation metrics mentioned above. We further validate our incremental learning approach by comparing it with incremental learning using human-annotated labels.

We structure the paper as follows. Section 2 discusses recent research on incremental learning. Section 3 summarizes the characteristics of several DRAs. Section 4 presents two algorithms, the FE process and the incremental learning process based on self-labeling. Section 5 prepares the selected dataset to evaluate and validate both algorithms. Section 6 performs several experiments to assess the two algorithms and compares incremental learning using pseudo-labels with incremental learning using actual labels. Section 7 concludes with some findings and future work.

2. Related Work

For the product prediction of buyers, the study [9] developed an incremental matrix factorization method based on the SGD algorithm but considered only the positive feedback. The method (called ISGD) adjusts the factor matrices gradually for the current instance, and for each rating, it measures the corresponding error and then conducts the update operations. Through four datasets, the experiments reveal that ISGD is competitive, but signicantly faster than other known algorithms, such as "Bayesian Personalized Ranking MF (BPRMF)", its weighted variant (WBPRMF), and the incremental version of the "User-based Nearest Neighbors algorithm (UKNN)". ISGD is a clear winner for two cases, and for the remaining cases, UKNN returned better Recall, but ISGD was faster. However, for training, ISGD employs only the true values (positively rated products) and considers the false values (either a user disliked the product or did not interact with it) missing. Still, it is not clear how these missing values were handled.

In real-life scenario, the streaming data that are generated incessantly over time is unstructured, and it is a tedious task to classify these data as they lack the target class labels. Therefore, the study [10] presented the "Classification of Unstructured data using Incremental Learning approach (CUIL)". The proposed method uses the uCLUST algorithm to cluster meta-data to assign the class-labels to each cluster, and later these labeled data are fed to a feed-forward neural network named "Extreme Learning Machine (ELM)" to incrementally assess the new data batches. CUIL randomly selects the input weights with one hidden layer. Nevertheless, to reduce the training time and memory storage significantly, CUIL trains every new data batch separately. The experimental results demonstrate better efficiency and accuracy over the existing well-known Learn++ algorithm. Nevertheless, the authors did not provide detailed information about hyper-parameters and architecture for the ELM network.

The authors in [11] suggested a chunk-based incremental learning algorithm (named CICSHL-SVM) based on the "Cost-Sensitive Hinge Loss" function. The method adjusts an entire data chunk at once and supports the model stability using the cost-sensitive Bayes risk. It also handles the dual problem of SVM, where the objective function is minimized under some constraints using the convex optimization theory. The authors validated the new algorithm with eleven benchmark datasets. The experimental outcome confirms the efficacy of CICSHL-SVM and demonstrates that it outperforms the static version and the single incremental version (learns instance by instance). Nevertheless, the authors did not mention anything about the penalties of the cost-sensitive classification.

Recently, the research [12] introduced an incremental unsupervised neural network, called SOINN+. The latter differs from its past version SOINN that is capable of learning from the non-stationary data based on the forgetting method. The deletion of nodes and edges is determined using two hyper-parameters in SOINN and those hyper-parameters are optimized utilizing the cross-validation. However, SOINN+ tackles this inadequacy by rejecting the nodes and edges that are not relevant for the learning process, and thus leads to an improved forgetting mechanism. The authors also defined three concepts to obtain a more graceful forgetting: a) idle time, b) (un-)utility of a node, and c) trustworthiness. In the experiments, SOINN+ and five other stream clustering techniques were used to identify the clusters in the noisy data streams with sudden and recurring concept drifts. The authors utilized real-world and synthetic datasets for the experiments and the results revealed that SOINN+ could distinguish the actual clusters and is able to preserve the determined shape even under the sudden and recurring concept drifts than the other methods.

The work [13] defined a chunk-based incremental classification algorithm to highlight the critical issue of shill bidding data labeling in commercial auctions. For addressing the issue of scarcity of labeled data, the authors used SGD algorithm to develop the incremental learning process as SGD is independent of the training chunks' size, which is essential in real-life auction fraud detection scenarios. The authors also demonstrated how to adapt the SGD classifier progressively with new training chunks. Based on a real auction fraud dataset, the experiments showed that the sequential trained classifiers' performance increased for each incremental training phase. In contrast, the misclassification rate, run-time, and loss decreased steadily.

Lastly, to detect malicious bidders in e-auctions, the recent study [14] developed a self-adaptive chunk-based incremental learning framework. The self-adaptive framework can function in real-world auction settings, which first classifies incoming bidder chunks for countering the fraud bidders in every auction and takes required measures. The fraud classifier is adapted with the confident bidders' labels that are validated through two verification models: (1) bidder verification and (2) one-class classification. Besides, the authors presented an extensive experimental study utilizing a real fraud dataset built from eBay commercial auctions where the classifier adapts incrementally by utilizing only the relevant bidding data. Later, the adjusted models are evaluated using misclassification and detection rates. Furthermore, the authors considered the fact that the adaptation chunks can be imbalanced and different class ratios may be present in real-life scenarios, and tackled this concern by retaining only the trusted fraud data and under-sampling the trusted normal data strategically. A comparison between the proposed framework with static learning and learning without data relevancy is also presented to show the superiority of the proposed framework.

3. Dimensionality Reduction

In general, FEs downsize the column (features) number or converts a sphere to a circle in two-dimension [15]. FE methods merge correlated features to new features by maintaining the original features' inherent properties [16, 17]. These DRAs look for a manifold representation to project high dimensional input data and then regulate a lower-dimensional space embedded in the input space [18]. DRAs have several benefits, such as improving the model generalization capability and lowering overfitting through less misleading features, and reducing processing time and data storage [16]. Since DRAs depend on the data characteristics and quality, we explore diverse unsupervised algorithms to non-linearly and linearly transform the high-dimensional space to the reduced feature space. After reviewing the literature thoroughly, we choose the most successful DRAs. The work [18] found out that the following methods are more efficient as they can adapt the local data structure in a better way, such as Locally Linear Embedding (LLE), Laplacian Eigenmap (LE), Kernel Principal Component Analysis (KPCA), Isometric Mapping (ISOMAP) and Multi-Dimensional Scaling (MDS). In Table 1, we summarize the essential characteristics of the selected DRAs.

Table 1. Unsupervised Feature Extraction Methods

	Goal	**Iterative**	**Tuning Parameters**	**Computational Complexity**	**Weakness**	**Transformation**
MDS	Preserve Euclidean pairwise distances	Yes	Iterations, number of components	$O(n^3)$	Require large memory to calculate dissimilarity matrix and high computation time	Dissimilarity matrix by Euclidean distance, Eigen decomposition
KPCA	Maximize variance	Yes	Kernel function, number of components,	$O(n^3)$	Computationally expensive and high memory consuming	Eigen decomposition, Kernel mapping
LLE	Preserve local properties	Yes	iterations, number of components, nearest neighbors	$O(dlog(c)nlog(n)$ $+O(dnc^3)+$ $O(kn^2)$	Less accurate in global structure	Neighborhood graph, Reconstruction of weights, Eigen vector based optimization
ISOMAP	Preserve geodesic pairwise distances	No	Number of components, nearest neighbors	$O[dlog(c)nlog(n)+$ $O[n^2(c+log(n))]+$ $O(kn^2)$	Suffer from topological instability	Geodesic distance, Neighborhood graph, Extension of MDS
LE	Preserve local distances	No	Nearest neighbors, number of components	$O(dlog(c)nlog(n)$ $+O(dnc^3)+$ $O(kn^2)$	Can generate disconnected neighborhood graph	Weight update of edges, Neighborhood graph, Cost function optimization

For instance, MDS converts data into lower dimensional space by calculating the dissimilarity in a way that less similar data are far apart and similar data are together [19]. MDS has two tuning parameters: number of features (called principal components for all the methods) and number of iterations. KPCA adopts the Eigen decomposition to determine the new feature space [20]. KPCA has two meta-parameters: the number of fused features and the Kernel trick, such as Linear, Polynomial, RBF, and Sigmoid functions, to project the data into higher feature space, hence data become linearly separable [20]. However, KPCA has high memory consumption [21]. LLE uses a neighboring graph to transform the input data space and preserve its geometric structure (local properties) [22]. LLE first finds the NNs of every data points employing the Euclidean distance, then computes the local weights to represent data optimally as the linear combination of NNs by minimizing the reconstruction error, and finally defines a new and unique vector space using the Eigen vector-based optimization [23]. LLE has three tuning parameters: number of iterations, number of PCs, and number of NNs. ISOMAP, which is also known as an extension of MDS, initiates a neighborhood graph first by calculating the geodesic distances between every pairs of data [24]. Then, it obtains the low dimensional embedding of the data through Eigenvalue decomposition of the geodesic distance matrix. ISOMAP has two tuning parameters: number of NNs and number of PCs. LE finds the low-dimension data by preserving the "local properties" of a manifold, that is close to LLE and produces a neighborhood graph where all data points are connected to their NNs by an edge. Then, "Gaussian kernel function" is employed to evaluate the weights of the edges [25] and LE minimizes the cost function to reduce the dimensionality. It has two tuning parameters: number of NNs and number of PCs.

A crucial query that rises is how to find out the optimal feature set that yields to the highest data quality and predictive accuracy. For this purpose, we estimate the quality of the reduced datasets using two correlation metrics: p-value and power analysis [15]. The first metric measures the "statistical significance" of a test case and is evaluated using a fixed significance level; the outcome of the p-value is statistically significant if it is less than the fixed level [26]. In our context, a smaller p-value means an elevated possibility of possessing better data [26]. The second metric denotes the likelihood of making a specific decision to nullify the null hypothesis if it is false in a test case scenario. Henceforth, a higher power value means the data is more robust [27].

4. Incremental Learning with Self-Labeling

Our incremental approach learns in real-time from incoming data that we organize in the form of mini-datasets, called chunks. Each chunk represents the accumulated data for a specific time period. However, in real-life scenarios, incoming data are unlabeled, and obtaining the labels is very costly due to the human expertise access and time delay. Hence, this work merges self-labeling and chunk-based incremental learning to instantly assign labels to the new chunks and adjust the classifier progressively with pseudo-labeled chunks.

Incremental learning targets to improve an existing model gradually with new data, without retraining from the beginning, and without immediately forgetting the learned knowledge [3, 4]. For this purpose, we should guarantee the stability and plasticity of the classifier [14]. Indeed, our incremental classifier should be stable enough to retain the information of the current and previous chunks a little longer in the memory and should forget past chunks gradually. Otherwise, the classifier will no longer be able to adapt to new data appropriately [28]. We implement our incremental learning algorithm using the Scikit-learn toolkit to leverage the available incremental mechanisms, such as the incremental API for conducting incremental learning (instance by instance), the ‘PartialFit’ for receiving data sequentially in the form of mini-datasets, and the incremental memory model for implementing the stability and plasticity strategies. All these mechanisms are described in detail in [4].

As the base classifier for the incremental learning process, we choose the SGD algorithm since: a) it supports the ‘PartialFit’ approach to process the data in chunks, b) it is independent of the size of the training chunks, which is important in practice since chunks come in different sizes, and c) has fewer tuning parameters, such as the learning rate, loss function,

Algorithm 1: Incremental Learning with Self-labeling of Incoming Highly-dimensional Chunks

```
Inputs: initialChunk (balanced), incomingChunks
Output: improved classifier

*Make SGD Incremental*
1: connect(IncrementalAPI, PartialFit)
2: wrap(SGD, IncrementalAPI)
*Determine best DRA and best number of PCs for initial chunk*
3: Algorithm2(initialChunk, out optimalReducedChunk, out bestDRA, out
   bestNumberPCs)
*Build Initial Classifier*
4: classifier = train(SGD, optimalReducedChunk, 10-fold CV)
*Incremental Learning phases*
5: for each incomingChunk do
    *Transform incoming chunk*
    5.1: optimalReducedChunk = extractFeature1(incomingChunk, bestDRA,
         bestNumberPCs)
    *Self-label transformed chunk*
    5.2: ClassifiedChunk = selfLabel(optimalReducedChunk, classifier)
    *Re-balance labeled chunk*
    5.3: if highClassRatio(ClassifiedChunk) then
        5.3.1: resample(ClassifiedChunk, newRatio)
    *Adapt classifier with incremental chunk*
    5.4: adapt(classifier, ClassifiedChunk, modelMemory)
```

Algorithm 2: Dimensionality Reduction of Data Chunk

```
Input: dataChunk
Outputs: optimalReducedChunck, bestDRA, bestNumberPCs

1: allReducedChunks = emptyset
2: for each DRA do
    2.1: do
        2.1.1: bestReducedChunk_DRA = extractFeature2(dataChunk, DRA)
        2.1.2: evaluateQuality(bestReducedChunck_DRA)
    while data quality not satisfactory;
    2.2: allReducedChunks = allReducedChunks ∪ bestReducedChunck_DRA
3: selectBest(allReducedChunks, out optimalReducedChunk, out bestDRA, out
   bestNumberPCs)
```

and penalty, when compared to neural networks [13]. Algorithm 1 exposes the incremental learning approach using self-labeling of incoming chunks with highly dimensionality. First, based on the 10-fold Cross Validation, we develop the preliminary SGD classifier using the optimally reduced initial training chunk. The latter is obtained by testing several DRAs to determine the optimal low-dimensional feature space (see Algorithm 2). The data quality is measured with the two correlation metrics presented in Section 3. Subsequently, we transform each incoming unlabeled chunk using the best DRA and the optimal number of Principal Components (PCs) obtained in the initial stage, and then adapt the classifier gradually with incremental chunks: transformed, self-labeled and re-balanced. In case the self-labeled chunk is highly imbalanced, we adopt a hybrid data sampling technique, such as SMOTE-ENN, to re-balance the class distribution with the appropriate ratio [29, 30].

5. Data Chunk Organization

We choose a multi-class dataset called "Human Activity Recognition (HAR) Using Smartphones", made public in the UCI repository in 2012 [31]. The labeled dataset has six classes indicating six activities (Walking, Sitting, Standing, Laying, Walking Downstairs, and Walking Upstairs). HAR has a tally of 10,299 records and 561 features that are already scaled to the range of [-1, 1]. For conducting the initial and incremental learning tasks, in Table 2, using the stratified splitting method, first, we divide the HAR dataset into the initial training chunk (30% of data) and a subset (70% of data). This subset is in turn stratified split into four chunks. However, we consider the four chunks without their labels to simulate the arrival of incoming data. The initial chunk should be adequately representative to build a robust initial classifier. As seen in Table 2, the initial chunk is balanced because the six classes are well distributed.

Table 2. Initial Training Chunk and Incoming Chunks

Initial Training Chunk (3089)					
Class#1	**Class#2**	**Class#3**	**Class#4**	**Class#5**	**Class#6**
516	463	422	533	572	583

Incoming Unlabeled Data (7210)			
Chunk#1	**Chunk#2**	**Chunk#3**	**Chunk#4**
1802	1803	1802	1803

6. Experiments

6.1. Feature Extraction

Initial Chunk: We first evaluate the selected DRAs on the initial chunk to identify the best FE method and the optimal number of principal components (PCs) using three quality metrics: p-value, power analysis, and processing time (in seconds). For KPCA, we use the Gaussian function RBF. We optimize the other DRA parameters, such as the number of iterations and number of NNs. Table 3 exposes the DRA results for the initial chunk: the original and reduced chunks. We can see that LLE is the best method as it returned the lowest p-value and a high power analysis.

LLE is ranked second in terms of run-time, with a gap of 10.3 seconds with the fastest method (KPCA). Time values denote the processing time of a chunk of data (here 3089 records with 6 classes). From the DRA summary in Table 1, we can see that both MDS and KPCA are computationally expensive. In Table 3, KPCA takes much less time than MDS since MDS requires three times more iterations than KPCA. On the other hand, although KPCA and LLE have the same number of iterations, LLE requires more time since it calculates the Eigen vector-based optimization on top of computing the neighborhood graph and reconstruction of weights. We may notice that for p-value and power, we obtain 0 and 1 respectively for all the DRAs except LLE. These values are due to the non-Gaussian distribution of the HAR dataset and the presence of six class targets. LLE efficiently reduced the dimensionality from 561 to 225 (ranked second), saving the processing time for the incremental training and data storage as well. MDS generated the highest number of PCs and is the slowest method. Another important observation is that all the DRAs provided much better data quality than the original chunk. Consequently, we select LLE as the optimal DRA for our 6-class dataset.

Table 3. Initial Chunk Quality: Original vs. Reduced

Initial Chunk	**Principal Components**	**Nearest Neighbors**	**Iterations**	**P-value**	**Power**	**Time**
Original	561	NA	NA	0.437	0.121	NA
KPCA	300	NA	100	0	1	5.2
MDS	478	NA	300	0	1	893.2
LLE	225	5	100	$\mathbf{1.04}e^{-11}$	**1**	15.5
ISOMAP	323	9	NA	0	1	20.7
LE	205	7	NA	0	1	26.5

Incoming Chunks: Next, we perform the LLE method with the number of PCs equal to 225 on the four incoming chunks separately. Each transformed chunk has a feature-space of 225 and the same size as its original version (see Table 2). Table 4 assesses the data quality of the reduced incoming chunks and also the quality of the original (non-reduced) chunks. Across the four chunks, we can see that p-value decreased and power analysis increased after performing LLE. For instance, we obtain a very low p-value of $7.52e^{-46}$ and a high power of 1 for the first reduced chunk, with a substantial p-value and power gaps when compared to the original first chunk. Moreover, the second chunk returned the lowest data quality after the transformation, but it is still very satisfactory. Also, we notice a very low run-time when executing LLE, which is significant for the fast processing of incoming data in real-life contexts. For example, LLE required only 3.42 seconds to transform the first incoming chunk (1802 records and 561 features) to a new one.

Table 4. Quality of Reduced and Original Incoming Chunks

Incoming Chunk	**Principal Components**	**P-value**	**Power**	**Time**
Reduced#1	225	$7.52e^{-46}$	1	3.42
Reduced#2	225	0.332268	0.162	3.74
Reduced#3	225	$1.46e^{-15}$	1	3.40
Reduced#4	225	$3.63e^{-16}$	1	3.82
Original#1	561	0.249	0.210	NA
Original#2	561	0.416	0.128	NA
Original#3	561	0.975	0.050	NA
Original#4	561	0.114	0.114	NA

6.2. Multi-class Incremental Learning

Initial Classifier: For building the initial classifier, we first train the SGD algorithm on the transformed initial chunk of 225 new features (presented in Table 3) using 10-fold cross-validation. Table 5 presents the performance of the initial classifier in terms of Precision, Recall, F1-score, False Negative Rate (FNR) and Run-time. The initial classifier provided high performance where the F1-score and FNR are 92% and 8% respectively and took only 1.995 seconds to learn from the whole chunk.

Incremental Learning: Next, we adjust the classifier over time with newly introduced chunks that are first optimally reduced in Table 4 and then self-labeled using the same classifier. However, before conducting any incremental training, we first verify the class distribution ratio of each self-labeled chunk. The 6 classes are balanced in each chunk.

In Table 5, we build four different models incrementally and show their performance using the reduced and self-labeled chunks (incremental chunks). Across all the models, the classifier performance is high. There is a slight increment between the initial model and the last model because SGD classifier is learning from small-sized chunks. So, we believe the accuracy will increase over time with more incoming chunks. The difference in terms of the misclassification rate between the initial and the last incremental chunks is 1.1%, which means that the classifier is predicting more accurately over time. As observed, the incremental learning does not take much time; the highest time is only 0.370 seconds, which is essential in the context of speedy data flow. Moreover, the classifier requires less time for the last chunk among all other chunks, whereas the initial chunk necessitates the maximum time because this is the first time the classifier is learning from the data. Regarding the incremental chunks, the classifier improves its knowledge without re-training from the beginning, which saves a lot of time.

Table 5. Incremental Learning Performance with Pseudo-labels

Chunk	**Improved Classifier**	**Precision**	**Recall**	**F1-score**	**FNR**	**Time**
Human-Annotated Initial Chunk	Initial Model	0.920	0.920	0.920	0.080	1.995
Incremental Chunk#1	Model#1	0.924	0.926	0.925	0.074	0.370
Incremental Chunk#2	Model#2	0.909	0.909	0.909	0.091	0.220
Incremental Chunk#3	Model#3	0.905	0.904	0.905	0.096	0.200
Incremental Chunk#4	Model#4	0.930	0.931	0.931	0.069	0.170

6.3. Pseudo Labels vs. Human Labels

Table 6 compares the performance of the incremental learning approach between pseudo-labels and human-annotated labels for all the incremental chunks. Note that, here, learning with human-annotated labels is still incremental. The latter returned slightly better accuracy for all the chunks because those labels have been labeled with ground truth by one or more human experts. In contrast, some pseudo-labels may have a weak annotation. Still, our approach is competitive to the approach with human-annotated labels. For incremental chunk#1, the pseudo-labeled data provided an F1-score of 92.5% and the human-annotated data F1-score of 93%, with a difference of only 0.5%. The gaps are 0.3%, 2.7% and 2.3% for chunk#2, chunk#3 and chunk#4, respectively. As new chunks arrive over time, our approach's accuracy will improve as the classifier learns from new data.

Table 6. Performance: Pseudo labels vs. Human-annotated Labels

Incremental Chunk	**F1-score**	
	Pseudo Labels	**Human-annotated Labels**
#1	0.925	0.930
#2	0.909	0.932
#3	0.905	0.932
#4	0.931	0.934

7. Conclusion

Our study jointly addressed three significant problems: incoming unlabeled data chunks, high data dimensionality, and incremental learning. For this purpose, we combined the chunk-based incremental learning approach with self-labeling and FE. We assessed several unsupervised FE techniques using a high-dimensional dataset to determine the chunks' optimal low-dimensional feature spaces. The classifier is then gradually trained with incremental chunks: first optimally reduced and then self-labeled. The learned classifier maintained high detection rates and low misclassification rates over the incremental learning process. Moreover, since speed is crucial in risk and event detection, the incremental classifier could process the incoming chunks in real-time. Lastly, we showed that our incremental learning approach with pseudo labels is competitive to incremental learning with human-annotated labels.

This current work provides several future research directions:

(1) We will investigate DRA for complex data, such as multi-dimensional time-series, because the literature is minimal.

(2) We will also explore incremental learning using deep neural networks.

References

[1] A. Bouchachia, B. Gabrys, and Z. Sahel. "Overview of some incremental learning algorithms". In: IEEE International Fuzzy Systems Conference. 2007, pp. 1–6.

[2] W. Zang, P. Zhang, C. Zhou, and L. Guo. "Comparative study between incremental and ensemble learning on data streams: Case study". In: *Journal Of Big Data, SpringerOpen* 1.5 (2014), pp. 1–16.

[3] V. Losing, B. Hammer, and H. Wersing. "Incremental on-line learning: A review and comparison of state of the art algorithms". In: *Neurocomputing, Elsevier* 275.1 (2018), pp. 1261–1274.

[4] F. Anowar and S. Sadaoui. "Incremental Neural-Network Learning for Big Fraud Data". In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2020, pp. 3551–3557. DOI: 10.1109/SMC42975.2020.9283136.

[5] P. Jindal and D. Kumar. "A review on dimensionality reduction techniques". In: *International journal of computer applications* 173.2 (2017), pp. 42–46.

[6] S. Abe. "Feature selection and extraction". In: *Support vector machines for pattern classification*. Advances in Pattern Recognition, Springer, 2010, pp. 331–341.

[7] J. Yan, B. Zhang, N. Liu, S. Yan, Q. Cheng, W. Fan, Q. Yang, W. Xi, and Z. Chen. "Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing". In: *IEEE transactions on Knowledge and Data Engineering* 18.3 (2006), pp. 320–333.

[8] L. H. Ru, T. Andromeda, and M. Marsono. "Online data stream learning and classification with limited labels". In: *Proceeding of the Electrical Engineering Computer Science and Informatics* 1.1 (2014), pp. 161–164.

[9] J. Vinagre, A. M. Jorge, and J. Gama. "Fast incremental matrix factorization for recommendation with positive-only feedback". In: *22nd Conference on User Modeling, Adaptation, and Personalization*. 2014, pp. 459–470.

[10] S. Madhusudhanan, S. Jaganathan, and J. LS. "Incremental learning for classification of unstructured data using extreme learning machine". In: *Algorithms* 11.10 (2018), pp. 1–19.

[11] B. Gu, X. Quan, Y. Gu, V. S. Sheng, and G. Zheng. "Chunk incremental learning for cost-sensitive hinge loss support vector machine". In: *Pattern Recognition* 83 (2018), pp. 196–208.

[12] C. Wiwatcharakoses and D. Berrar. "SOINN+, a Self-Organizing Incremental Neural Network for Unsupervised Learning from Noisy Data Streams". In: *Expert Systems with Applications* 143 (2020), pp. 1–33.

[13] F. Anowar and S. Sadaoui. "Chunk-Based Incremental Classification of Fraud Data". In: *The Thirty-Third International Florida Artificial Intelligence Research Society Conference (FLAIRS)*. AAAI Press, 2020, pp. 176–180.

[14] F. Anowar and S. Sadaoui. "Incremental learning framework for real-world fraud detection environment". In: *Computational Intelligence* 37.1 (2021), pp. 635–656. DOI: https://doi.org/10.1111/coin.12434. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12434.

[15] F. Anowar, S. Sadaoui, and B. Selim. "Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)". In: *Computer Science Review* 40 (2021), pp. 1–13. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2021.100378. URL: https://www.sciencedirect.com/science/article/pii/S1574013721000186.

[16] S. Khalid, T. Khalil, and S. Nasreen. "A survey of feature selection and feature extraction techniques in machine learning". In: *2014 Science and Information Conference*. IEEE. 2014, pp. 372–378.

[17] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, R. Kaluri, D. S. Rajput, G. Srivastava, and T. Baker. "Analysis of dimensionality reduction techniques on big data". In: *IEEE Access* 8 (2020), pp. 54776–54788. DOI: 10.1109/ACCESS.2020.2980942.

[18] C. O. S. Sorzano, J. Vargas, and A. P. Montano. "A survey of dimensionality reduction techniques". In: *arXiv preprint arXiv:1403.2877* (2014).

[19] M. A. Cox and T. F. Cox. "Multidimensional scaling". In: *Handbook of data visualization*. Springer, 2008, pp. 315–347.

[20] H. Hoffmann. "Kernel PCA for novelty detection". In: *Pattern recognition* 40.3 (2007), pp. 863–874.

[21] F. Zhao, I. Rekik, S.-W. Lee, J. Liu, J. Zhang, and D. Shen. "Two-phase incremental kernel PCA for learning massive or online datasets". In: *Complexity* 2 (2019).

[22] S. T. Roweis and L. K. Saul. "Nonlinear dimensionality reduction by locally linear embedding". In: *science* 290.5500 (2000), pp. 2323–2326.

[23] L. K. Saul and S. T. Roweis. "An introduction to locally linear embedding". In: (2000). URL: http://www.cs.toronto.edu/\~{}roweis/lle/publications.html.

[24] M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford. “The isomap algorithm and topological stability”. In: *Science* 295.5552 (2002), pp. 7–7.

[25] M. Belkin and P. Niyogi. “Laplacian eigenmaps for dimensionality reduction and data representation”. In: *Neural computation* 15.6 (2003), pp. 1373–1396.

[26] B. Vidgen and T. Yasseri. *P-Values: Misunderstood and Misused*. 2016. URL: https://www.frontiersin.org/articles/10.3389/fphy.2016.00006/full#:~:text=The\%20p\%2Dvalue\%20quantifies\%20the,result\%20is\%20considered\%20statistically\%20significant..

[27] L. Theme. *What Is Power?* 2017. URL: https://www.statisticsteacher.org/2017/09/15/what-is-power/..

[28] X. Geng and K. Smith-Miles. “Incremental Learning”. In: *Encyclopedia of Biometrics*. Boston, MA: Springer US, 2009, pp. 731–735. ISBN: 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_304. URL: https://doi.org/10.1007/978-0-387-73003-5_304.

[29] F. Anowar, S. Sadaoui, and M. Mouhoub. “Auction fraud classification based on clustering and sampling techniques”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 366–371.

[30] F. Anowar and S. Sadaoui. “Detection of Auction Fraud in Commercial Sites”. In: *Journal of Theoretical and Applied Electronic Commerce Research, Universidad de Talca* 15.1 (2020), pp. 81–98.

[31] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. “A public domain dataset for human activity recognition using smartphones.” In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Vol. 3. 2013, pp. 437–442.