Dimensionality reduction in the context of dynamic social media data streams

Moritz Heusinger¹ · Christoph Raab¹ · Frank-Michael Schleif¹

Received: 13 January 2021 / Accepted: 22 July 2021 / Published online: 13 August 2021
© The Author(s) 2021

Abstract

In recent years social media became an important part of everyday life for many people. A big challenge of social media is, to find posts, that are interesting for the user. Many social networks like Twitter handle this problem with so-called hashtags. A user can label his own Tweet (post) with a hashtag, while other users can search for posts containing a specified hashtag. But what about finding posts which are not labeled by the creator? We provide a way of completing hashtags for unlabeled posts using classification on a novel real-world Twitter data stream. New posts will be created every second, thus this context fits perfectly for non-stationary data analysis. Our goal is to show, how labels (hashtags) of social media posts can be predicted by stream classifiers. In particular, we employ random projection (RP) as a preprocessing step in calculating streaming models. Also, we provide a novel real-world data set for streaming analysis called NSDQ with a comprehensive data description. We show that this dataset is a real challenge for state-of-the-art stream classifiers. While RP has been widely used and evaluated in stationary data analysis scenarios, non-stationary environments are not well analyzed. In this paper, we provide a use case of RP on real-world streaming data, especially on NSDQ dataset. We discuss why RP can be used in this scenario and how it can handle stream-specific situations like concept drift. We also provide experiments with RP on streaming data, using state-of-the-art stream classifiers like adaptive random forest and concept drift detectors. Additionally, we experimentally evaluate an online principal component analysis (PCA) approach in the same fashion as we do for RP. To obtain higher dimensional synthetic streams, we use random Fourier features (RFF) in an online manner which allows us, to increase the number of dimensions of low dimensional streams.

Keywords Data streams · Random projection · Social media · Stream classification · Random fourier features

1 Introduction

Textual data are very common and a key communication format in social media. To analyze these data by machine learning models embedding approaches are used, leading to high dimensional representation (Mikolov et al. 2013). This becomes even more challenging in non-stationary environments and is a severe problem. There is ongoing research

on the analysis of high dimensional data, in particular in stationary environments (Lee and Verleysen 2007; Kaban 2015). Only recently dimensionality reduction in non-stationary environments has been addressed (Carraher et al. 2016). The high dimensionality of the data can lead to the phenomenon, that the distance to the nearest data point is very close to the distance to the farthest data point. Given the data is intrinsically low dimensional, the dimensionality can be reduced with low error (Kaban 2015). To address this problem, there exist different projection and embedding methods. The most common projection and embedding techniques are the principal component analysis (PCA), locally linear embedding, ISOMAP, Multidimensional Scaling and t-distributed Stochastic Neighbor Embedding (t-SNE) (Sacha et al. 2017).

These methods have one problem in common. They only work on a moderate number of dimensions, but not on thousands of input dimensions. In Maaten and Hinton

✉ Moritz Heusinger
moritz.heusinger@fhws.de

Christoph Raab
christoph.raab@fhws.de

Frank-Michael Schleif
frank-michael.schleif@fhws.de

¹ Department of Computer Science, University of Applied Sciences Würzburg-Schweinfurt, Sanderheinrichsleitenweg 20, Würzburg, Germany

(2008) it is recommended to preprocess very high dimensional data to a dimensionality of 30 via PCA before applying t-SNE. However, PCA is costly taking $\mathcal{O}(d^{2n} + d^3)$, where n is the number of samples and d the number of features (Wold et al. 1987). Hence, less expensive dimensionality reduction techniques are desirable. This points to Random Projection, which is a dimensionality reduction method to reduce the dimensionality of data from very high to high, e.g., from 10,000 to 1000 dimensions (Achlioptas 2001; Kaban 2015; Kaski 1998). The guarantees given by RP are based on the Johnson Lindenstrauss (JL) lemma Johnson and Lindenstrauss (1984).

In stationary environments as well as in streaming data, dimensionality reduction is a topic of interest. On one hand, it enables the visualization of data, and on the other hand, it reduces the complexity of data. In non-stationary environments, algorithms face additional problems. E.g., there are often time-dependent changes of the data distribution, called concept drift (CD) as detailed later on. Additionally we do not have access to the whole dataset at the beginning of a learning task. Thus, the model adaption has to be done in an online manner. The time to predict a label is also strictly limited, due to the real-time characteristics of data streams. Hence, we want to avoid too high dimensional data because of the time it will take to predict the corresponding label y for an incoming data point x (Bifet and Gavaldà 2009).

Due to the various challenges caused by high dimensional data representations in streaming analysis the field is not yet very much in the focus of the research community. In particular, there are not many data sets available to analyze new methods. Here, we provide a huge and novel dataset consisting of textual social media posts and their high dimensional encoding. Analyzing such streams is challenging, due to the high dimensional wordspace. E.g. encoding these words with bag of words term frequency-inverse document frequency (TF-IDF) can lead to tens of thousands of dimensions easily. But also word2vec based techniques provide high dimensional representations (Mikolov et al. 2013.) Our contributions included in this paper are:

- A We evaluate sparse matrix creation of RP for streaming data
- B We present a valid parameterization of RP in non-stationary environments
- C We discuss concept drift detection via RP in high dimensional dataspace
- D We present a scenario, where the dimensionality over a stream changes and how RP can handle it, using concept drift detection algorithms
- E We provide a complex novel real-world streaming data set called NSDQ which consists of real tweets using hashtags as labels and show the challenges of this data-

set in our experiments, which represents the major focus of this paper

- F We show how low dimensional data streams can be transformed into a higher dimensional and linear separable space, using RFF and use the obtained data to evaluate drift detection on data transformed by online PCA

The paper is organized as follows. Section 2 analyzes previously published work addressing the focus of stream analysis, RP and social media analysis. In Sect. 3 we review RP and stream analysis and its characteristics in detail as well as Concept Drift. Section 4 explains how dimensionality reduction techniques like RP and PCA can be used in non-stationary environments. In Sect. 5 we explain how our novel NSDQ dataset is structured and the topic of Hashtag prediction is described, which can be performed on this dataset. Section 6 provides extensive experiments using RP and PCA combined with stream classifiers to predict the labels of the NSDQ dataset. Finally, we summarize our research results and give an outlook on future research in Sect. 7.

2 Related work

Stream analysis is an active field of research (Gama et al. 2014; Bifet et al. 2019, 2018), widely focusing on sensor data. Additionally, social media data by means of Twitter feeds, customer touchpoints, and other user-generated data streams are becoming more and more important (Edosomwan et al. 2011).

Social media is not only a leisure-time activity but also an important research field with high economic impact (Edosomwan et al. 2011). There are several fields of interest e.g. extracting information about the user's personality (Golbeck et al. 2011) and finding new trends. Furthermore, sentiment analysis is an active field of research on social media data, not only for companies but also for investors (Habernal et al. 2013). In particular, hashtags are typically available for social streaming data as meta information, but not yet widely used in streaming analysis. However, hashtag prediction based on image data has already been applied (Park et al. 2016) in an offline manner.

Popular streaming benchmark datasets are common to be low dimensional containing mostly binary class labels (Gomes et al. 2017). Especially stream generators are often designed to evaluate a particular classification algorithm. While there are some high dimensional datasets of other domains, they are often missing stream characteristics like time-dependent concept drift and a higher number of instances. Most classifiers are also not analyzed on higher dimensional data due to slow runtime and high memory usage.

Preprocessing is another active field of research in streaming analysis (Bifet et al. 2019), but still quite limited. Some dimensionality reduction algorithms have been adapted to work with streaming data Ross et al. (2008); Grabowska and Kotłowski (2018). In Grabowska and Kotłowski (2018) an online PCA for evolving data streams has been published, which only keeps track of a subspace of small dimensions that capture most of the variance in the data.

Another preprocessing technique that aims to make data linear separable is the Random Fourier Features technique (Rahimi and Recht 2008). RFF provides an explicit, high dimensional mapping of a kernel function. RFF has already been used in non-stationary environments (Ullah et al. 2018). In Francis and Raimond (2018) RFF is used to detect anomalies in streaming data, compared to previous approaches which are based on Low-Rank Approximation, the proposed approach attains significant empirical performance improvements in datasets with a large number of samples. RFF has also been used for a streaming kernel PCA algorithm based on Oja's PCA, the algorithm is memory efficient compared to previous approaches (Ullah et al. 2018). Another recent application of RFF is a kernel conjugate gradient algorithm (Xiong and Wang 2019), which is based on mapping the original input data to a fixed-dimensional RFF feature space. The algorithm is competitive to other state-of-the-art conjugate gradient algorithms.

Furthermore, there are existing stream manifold learning algorithms (Schoeneman et al. 2017). In Schoeneman et al. (2017) error metrics for learning manifolds on streaming data are provided. The idea is to learn the manifold only on a fraction of the stream until the embedding is stable and then do the embedding by a nearest neighbor approach, eventually implemented in a stream ISOMAP algorithm (Schoeneman et al. 2017). Random Projection has already been applied in the fields of non-stationary data (Carraher et al. 2016; Pham et al. 2017). A stream clustering algorithm called streaming-RPHas, which uses RP and locality-sensitivity hashing with reasonable results was published in Carraher et al. (2016). Another work uses a Hoeffding Tree Ensemble to classify streaming data, but instead of working on high dimensions, it works on a lower dimensional space, the dimensionality reduction is done by RP Pham et al. (2017). All the previously mentioned approaches are limited in a way, that the particularities of streaming data are not sufficiently addressed or techniques are used only without theoretical foundations. For example, both former approaches use RP in streaming context without taking the effect of CD on RP into account, which is a major challenge in supervised streaming analysis. While the authors use different datasets, e.g. one for non-binary classification and one for unbalanced data, the techniques have not been evaluated on a complex dataset that combines most of the stream challenges in one dataset. RP was also not analyzed or proven for

non-stationary environments w.r.t. Johnson–Lindenstrauss lemma.

Despite the effort in prior work on stream analysis, the available methods are still not well aligned with real-life problems. Accordingly, we focus on typical real-life problems of streaming analysis and provide a real-world stream dataset consisting of textual social media posts with real challenges for streaming analysis.

3 Preliminaries

3.1 Random projection

First, we review some material about the JL lemma and particularities of RP for stationary data. We consider a data set $\mathbf{X}$ with datapoints $\mathbf{x} \in \mathbb{R}^d$ in a d -dimensional space. The data are subject to a random projection to obtain a lower dimensional representation of the data. In RP the d -dimensional input data is projected on a k -dimensional ($k \ll d$) subspace using a random matrix $\mathbf{R} \in \mathbb{R}^{k \times d}$.

The distortion introduced by RP π is asserted by the fact that π defines an ϵ -embedding with high probability defined by:

$$(1 - \epsilon)\|\mathbf{u} - \mathbf{v}\|^2 < \|\pi(\mathbf{u}) - \pi(\mathbf{v})\|^2 < (1 + \epsilon)\|\mathbf{u} - \mathbf{v}\|^2 \quad (1)$$

where $\mathbf{u}$ and $\mathbf{v}$ are any rows taken from data with n samples and d dimensions and π is a projection by a random Gaussian $\mathcal{N}(0, 1)$ matrix $\mathbf{R}$ (Dasgupta and Gupta 2003) or a sparse Achlioptas matrix (Achlioptas 2003).

To determine a minimum number of dimensions in the projected space, the lemma is given by:

$$k \geq \frac{4 \log n}{(\epsilon^2/2 - \epsilon^3/3)} \quad (2)$$

where the error made by the projection is less than ϵ (Achlioptas 2003). There exist different approaches for generating a sparse random projection matrix $\mathbf{R}$. One approach creates $\mathbf{R}$ by drawing samples from $\mathcal{N}(0, 1)$ and is called Gaussian RP. The other approaches create $\mathbf{R}$ with entries $r_{i,j}$ where i denotes the i -th row and j the j -th column of $\mathbf{R}$, mainly differing in the density:

$$r_{i,j} = \sqrt{s} \begin{cases} -1 & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{s} \\ +1 & \text{with probability } \frac{1}{2s} \end{cases} \quad (3)$$

In Achlioptas (2001) it is recommended to generate the density as shown in Eq. (3) with $s = 3$ to get better results than by Gaussian RP. In Li et al. (2006) it is recommended to use $s = \sqrt{d}$ instead of $s = 3$ because this speeds up the computation of the projection in comparison to the previous approach

by having a more sparse matrix $\mathbf{R}$. The first method is often referred to as sparse RP while the latter is called RP-VS.

3.2 Stream analysis

Many preprocessing algorithms can not be applied to streaming data, because there are some particularities Aggarwal (2014) when analyzing data in non-stationary environments:

- process an instance at a time t and inspect it (at most) once
- use a limited amount of time to process each instance
- use a limited amount of memory
- be ready to give an answer (prediction, clustering, patterns) at any time
- adapt to temporal changes

Some algorithms have been adapted to work with streaming data with a comprehensive review given in Aggarwal (2014). Often, restrictions of these adapted algorithms are stronger than restrictions of corresponding offline versions. E.g. statistics cannot be calculated over the whole data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t\}$, because the data become available once at a time. Thus, whenever a data point $\mathbf{x}_t$ arrives at time step t at the model h , the model takes this data point into account by learning an adapted model $h_t = \text{train}(h_{t-1}, \mathbf{s}_t)$, where $\mathbf{s}_t$ is the tuple consisting of $\mathbf{x}_t$ and its corresponding label y_t . Thus, when applying RP we will project one data point $\mathbf{x}_t$ at time t by our initially created projection matrix $\mathbf{R}$ to provide the projected data set to an online learning algorithm.

Another problem in non-stationary environments is CD which means that joint distributions of a set of samples $\mathbf{X}$ and corresponding labels $\mathbf{y}$ changes between two timesteps:

$$\exists \mathbf{X} : p(\mathbf{X}, \mathbf{y})_t \neq p(\mathbf{X}, \mathbf{y})_{t-1} \quad (4)$$

Note, that we can rewrite Eq. (4) to

$$\exists \mathbf{X} : p(\mathbf{X})_t p(\mathbf{y} | \mathbf{X})_t \neq p(\mathbf{X})_{t-1} p(\mathbf{y} | \mathbf{X})_{t-1}. \quad (5)$$

If there is only a change in the prior distribution $p(\mathbf{X})$ then it is called virtual drift.

There are five types of drift categorized by the change over time as shown in Fig. 1. For a comprehensive study of various CD types see Gama et al. (2014) and Raab et al. (2019). For this case, online classification algorithms use statistical tests to detect CD between two distributions. These tests are adapted and used as so-called CD detectors (CDD) [(e.g. Adaptive Windowing (Bifet and Gavalda 2007) and Kolmogorov-Smirnov Windowing Raab et al. (2019)) by testing Eq. (5).

3.3 Concept drift

As stated in Sect. 3.2, CD is a key challenge in streaming environments. Modern CDDs detect changes in the distribution by performing statistical tests (Bifet and Gavalda 2007; Raab et al. 2019). These tests are performed on every single dimension of the data. Hence, on d dimensional data, KSWIN Raab et al. (2019) performs d Kolmogorov-Smirnov tests at every t . A solution to address the slow performance is to perform the statistical test on a performance indicator of a classifier, e.g. accuracy, like Drift Detection Method (DDM) and Early DDM Baena-Garcia et al. (2006) do. However, this has the disadvantage that CD can only be detected after the change in the distribution is present for some iterations, leading to statistical relevant misclassifications. Hence, monitoring a performance indicator is a suboptimal heuristic. In summary, it might be desirable to work with not too high dimensional data, to be fast enough to address the other particularities of streaming data, especially time constraints. Thus, RP seems to be a suitable method, which can reduce the detection time of CDDs by allowing to perform the tests only k instead of d times. To be reliable the CDD has to detect drifts that exist in d dimensions in a k dimensional space obtained by RP. There exist many low dimensional synthetic stream generators which introduce specific forms of CD as described in Fig. 1. However, we are using our real-world NSDQ dataset and thus do not have introduced a specific kind of drift. Furthermore, different types of drift can be present in all real-world data.

3.4 Random Fourier features

The RFF algorithm efficiently converts the training and evaluation of any kernel machine into the corresponding operations of a linear machine by mapping data into a relatively low-dimensional randomized feature space. This is done by using a set of random features consisting of random Fourier bases $\cos(\mathbf{w}'\mathbf{x} + b)$ where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are random variables. These mappings project datapoints on a randomly chosen line and then pass the resulting scalar through a sinusoidal function. Drawing the direction of these lines from an appropriate distribution guarantees that the product of two transformed points will approximate a desired shift-invariant kernel (Rahimi and Recht 2008; Sriperumbudur and Szabo 2015).

The random Fourier features algorithm takes an shift-invariant kernel (e.g. Gaussian) $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ and then ensures a randomized feature map $z(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^D$ so that $z(\mathbf{x})'z(\mathbf{y}) \approx k(\mathbf{x} - \mathbf{y})$. Then it computes the Fourier transformation p of the kernel by $p(\mathbf{w}) = \frac{1}{2\pi} \int e^{-j\mathbf{w}'\delta} k(\delta) d\Delta$. In the next step, we draw D i.i.d. samples $\mathbf{w}_1, \dots, \mathbf{w}_D \in \mathbb{R}^d$ from p and D i.i.d. samples $b_1, \dots, b_D \in \mathbb{R}$ from the uniform distribution on $[0, 2\pi]$. In the last step, we calculate our mapping

Fig. 1 Different types of drifts, one per subfigure and illustrated as data mean. The different styles mark the dominate concept at given time step. The vertical axis shows the data mean and the transition from one to another concept. Given the time axis the speed of the transition is given. The figures are inspired by Gama et al. (2014)

by $z(\mathbf{x}) = \sqrt{\frac{2}{D}} [\cos \mathbf{w}'_1 \mathbf{x} + b_1 \cdots \cos \mathbf{w}'_D \mathbf{x} + b_D]'$. For a more detailed description of RFF, we refer to Rahimi and Recht (2008).

In the stream setting, we can map each arriving batch of datapoints $\mathbf{X} \in \mathbb{R}^d$ at a timestep t by our mapping $z(\mathbf{x})$ to obtain a higher dimensional data representation $\phi(\mathbf{X}) \in \mathbb{R}^D$. This higher dimensional representation can then be used by a linear online classifier. Having non-linear data it may be desirable to use RFF with a complexity of $\mathcal{O}(D + d)$, where D is the target space dimensionality and d the original dimensionality, instead of calculating a kernel which takes

$\mathcal{O}(Nd)$ operations due to the time constraints in streaming environments. Note, that once the random matrix $\mathbf{W}$ has been calculated, the transformation of every batch of data is performed by a simple matrix multiplication $\mathbf{W}\mathbf{X}$ like in RP.

4 Dimensionality reduction in non-stationary environments

In this section, we provide information on the usage of PCA and RP in the streaming context.

4.1 Non stationary random projection

4.1.1 Matrix creation and projection

Assume we have a stream with a fixed number of dimensions, which will not change over time, we can generate the projection matrix similar to the offline setting. Depending on a fast projection in non-stationary settings, RP-VS (Li et al. 2006) seems to be most suitable. The matrix creation of $\mathbf{R}$ and projection of $\mathbf{X}$ is of order $\mathcal{O}(dkn)$, however, if $\mathbf{X}$ is sparse, we only need $\mathcal{O}(ckn)$ with $\mathbf{X}$ having c non-zero elements per column.

After the matrix creation of $\mathbf{R}$ is done at the beginning of a stream in $\mathcal{O}(ck)$, we can project single datapoints $\mathbf{x} \in \mathbf{X}$ one after each other, performing $\mathbf{R} \times \mathbf{x}$. Note, that in the streaming scenarios, considered so far, d does not change over time. Thus, $\mathbf{R}$ does not need to be updated during the stream.

4.1.2 Valid parameterization of random projection

As shown in Eq. (2), a suitable number of k dimensions making an error less than ϵ can be calculated when n is known. Furthermore, there exist bounds that do not rely on n , but need other statistics over the data which are not known prior in streaming settings (Klartag and Mendelson 2005). Hence, we will focus on the JL lemma.

In streaming contexts, n is unknown and potentially infinite. However, stream classifiers store a fixed number of samples, in a window $\mathbf{W}$ (Losing et al. 2017a; Bifet and Gavaldà 2009) or prototypes θ which represent statistics of the data (Heusinger et al. 2020). We have to keep in mind, that the goal of RP is to preserve the distances in our data. A window-based stream classifier only calculates distances between stored samples in $\mathbf{W}$ and newly arriving datapoints. The window-size or number of prototypes is known prior to deploying a classifier. Hence, calculation of k by Eq. (2) is possible.

For clarification, we assume a window $\mathbf{W}$ of size $w = 1000$ which is a common window size (Bifet and Gavaldà 2009). Preserving distances between this window and a batch of 10 datapoints arriving at a time t leads us to $n = 1010$. Now, we calculate k from Eq. (2) by filling in our n and a suitable ϵ . Setting $\epsilon = 0.2$ leads us to $k \geq 1596$. Note, that the window has to be filled at the beginning of a stream, thus $|\mathbf{W}| < w$. However, this is not a problem, because $n < w$ leads to $k \leq k_w$.

4.1.3 Alternating dimensionality

We are introducing a scenario, where features of a data stream will be removed or new features are added. This can happen in real-world scenarios when a sensor gets replaced

with another sensor or the user of an application, which is based on the model, decides to take more or fewer dimensions into account. RP and the underlying classifier needs to handle this change. Assume d as number of dimensions, then the described scenario happens when $d_{t-1} \neq d_t$ at a given timestep t . Hence, the random matrix $\mathbf{R}$ needs to be adapted.

Here we assume a sparse matrix generated by Eq. (3) with $s = \sqrt{k}$. When a dimension is removed in the high dimensional space, we remove the corresponding column of $\mathbf{R}$. Thus, removing a dimension $\mathbf{X}_{:,j}$, where $:$ denotes all rows and j the j -th column of $\mathbf{X}$, leading to a new random matrix $\mathbf{R}_t = \mathbf{R}_{t-1} \setminus \mathbf{R}_{:,j}$. A classifier would still receive k dimensions and does not have to handle altering dimensions in forms of features. However, due to the possible change in one of the k dimensions, it introduces CD which can be handled by a CDD in advance. It is also possible, that a new feature gets added during the stream, thus we assume that this feature is appended to $\mathbf{X}$. Assuming $\mathbf{X}_{old}$ has a column length of j , the newly added dimension will be in $\mathbf{X}_{:,j+1}$. Thus, we also need a new column in $\mathbf{R}$, which should be generated by Eq. (3) and $s = \sqrt{d}$. The possible introduced CD should be handled in the same fashion as described for removing dimensions. Furthermore, only d changes, thus the number of rows k of $\mathbf{R}$ remain static because they can be addressed by our proposed fixed-length window. In summary RP allows the conversion of altering the number of dimensions into CD, which can be handled by CDDs instead of training a new classifier. This can be very handful because there are various CDDs and adaptive algorithms for CD handling (Raab et al. 2019; Bifet and Gavaldà 2007; Heusinger et al. 2020; Gomes et al. 2017), while there are no existing techniques for altering dimensions in non-stationary environments. Thus, a new model needs to be trained from scratch, every time the dimensionality of a datapoint $\mathbf{x}_t$ changes.

4.2 Online principal component analysis

Among many incremental PCA versions, we choose the version by Ross (Ross et al. 2008), which is an extension of the Sequential Karhunen-Loeve algorithm Levey and Lindenbaum (2000). The algorithm needs some hyperparameters which need to be defined in advance, the number of dimensions d , number of samples n processed, $m \geq 2$ the number of accumulated datapoints for the next updates, and k the number of principal components to include. The reason why we have chosen this algorithm is its low time and space complexity. The classic PCA has a time and space complexity of $\mathcal{O}(d^2(n + m) + d^3)$ and $\mathcal{O}(d^2)$, while the model of Ross only requires $\mathcal{O}(dm^2)$ and $\mathcal{O}(d(k + m))$. The reduced complexity is caused by updating the Singular Value Decomposition (SVD) of all datapoints incrementally with the partial SVD of only m new datapoints. In streaming environments, the

chosen m will be small and thus the cost can be reduced significantly.

Besides, there are other advantages of the method provided by Ross over other online PCA approaches. The model constantly updates its sample mean which is used for calculating the eigenbasis of the PCA. This removes the need for an initial learning phase and allows to use the model in the first iterations of a data stream. The algorithm also allows setting a forgetting factor f , which reduces the contribution of past datapoints to the latest update. The parameter $f \in [0, 1]$ can be set to 1 if no forgetting is desired, for $f < 1$ the contributions of previous observations are gradually decreased. The size of observations which affect the current update are $\frac{m}{1-f}$ (Ross et al. 2008). By adapting f we can support the incremental addition of new observations as well as incremental forgetting of past observations.

5 Twitter stream analysis

This section contains the description and analysis of the characteristics of the Twitter Stream Analysis. For readability, the results obtained from the stream classifier are shown in Sect. 6.

5.1 NSDQ dataset

The current real-world datasets (Heusinger et al. 2020) in the field do not present exceptional challenges for state-of-the-art stream analysis algorithms. In this work, we present a new dataset called NSDQ (NASDAQ)¹, which presents new challenges even for state-of-the-art algorithms (Losing et al. 2018; Gomes et al. 2017; Heusinger et al. 2020; Bifet and Gavaldà 2009; Raab et al. 2019). Regarding the name of the data set, please note that we follow the naming convention of previously published streaming data sets, see for example (Gomes et al. 2017). The dataset is based on Twitter feeds that have been crawled over a period from February 10, 2019, to December 3, 2019, and includes tweets of 15 NASDAQ companies. The tweets are time ordered to employ the natural occurrence of the feed. Due to the Twitter API usage restrictions, a subset of 30278 tweets was downloaded. The task is to predict the company label of a tweet. The ground truth label is the hashtag of their tweets.

The assumption in the dataset is, that the high volatility of the stock market is also reflected in the Twitter streams. The volatility of the stock market is known to be chaotic, therefore it is reasonable that all types of concept drift given in Fig. 1, but not limited to it, are present. The challenge

Fig. 2 Class distribution (Hashtags) of NSDQ dataset. The given hashtags are equivalent to the NASDAQ share codes

for stream algorithms is hereby to adapt to new concepts of NASDAQ Twitter feeds. The high amount of change is also shown in Fig. 5, where the popularity of the amazon hashtag is shown. Every local maxima indicates a new concept due to a new topic that is discussed on Twitter, as evaluated by a human expert. The concept drift changes the underlying language context of the tweets, which can be considered as a distribution change. Further, the dataset has large imbalances, naturally due to the varying popularity of the companies. The dataset is given via a skip-gram embedding (Mikolov et al. 2013) and encoding the 30,278 tweets as 1000-dimensional vectors. The high dimensionality proposes further challenges for the stream algorithms. In summary, the NSDQ dataset poses the following challenges:

- High feature dimension compared to existing datasets.
- High number of classes with large imbalances compared to existing datasets.
- High volatile dataset with many non-specified concept drifts.

Figure 4 shows a 2-dimensional t-SNE plot of the complete skip-gram data embeddings and a plot of the corresponding eigenvalues of the dataset is given in Fig. 3. The eigenvalue plot indicates that despite the challenges, the intrinsic dimensionality of the data is low. Therefore, it can be assumed that the data set is still sufficiently descriptive to reasonably predict the hashtags. The distribution of the labels is given in Fig. 2, showing the imbalanced classes as a bar diagram. Before training the embeddings, special characters are removed, smileys and emojis are encoded as phrases. Words are stemmed and stop words are removed. Further, we also propose a *tf-idf* feature encoding with a minimal document frequency of 10 and a maximal document

¹ Dataset available at <https://github.com/ChristophRaab/NASDAQ-Dataset>.

Fig. 3 Spectrum of the 100 largest eigenvalues of NSDQ embeddings

frequency of 100, which results in 3442 data dimensions. Preprocessing was done in offline mode after all tweets were crawled because for the embedding and *tf-idf* measure, all occurring words must be present at the time of processing. This scenario is similar for other real-world datasets such as the Airlines.² However, the repository also contains the raw tweets so that online pre-processing can be used as desired.

5.2 Hashtag prediction

The hashtag (the # sign followed by a phrase to a tweet, for example, #superbowl) is probably the most important function of Twitter search. The hashtag enables Twitter users to create topic-specific searches and enables the user to navigate the hypertext structures of the whole site. The power of the hashtag is that it creates very specific sets of content. If you want to know what other people think of the Super Bowl, you can find it easier by searching for the hashtag than by searching for something similar in a normal search engine. Analyzing the use of hashtags can also give companies and investors information about companies, e.g. combined with sentiment analysis and the hashtag corresponding to a companies name, can give us a clue about the current sentiment of a company (Kouloumpis et al. 2011).

Figure 5 shows the popularity of the #amzn hashtag over the last two months. We can see large time depending changes in popularity for this hashtag. Furthermore, the problem can arise, that a user tweets about a company but does not use the corresponding hashtag. Now, we would miss these tweets. Thus, we came up with the idea of predicting whether a given tweet includes a companies hashtag and thus affects a company or not.

² http://kt.ijs.si/elena_ikonovska/data.html.

6 Experiments

This section contains the details of the experiments and is split into four parts. Immediately afterward we detail the experimental setup, used algorithms, parameterization, and datasets. Further, the results are presented and discussed in three subsequent sections:

1. Section 6.1 details the results of classifiers given high dimensional random features and low-dimensional representations projected via RP.
2. Section 6.2 analyzes the results of classifiers on high dimensional features space obtained by RFF and a projected low-dimensional space via Incremental-PCA.
3. Finally, Sect. 6.3 details the influence of concept drift detection sensitivity give the four preprocessing steps as just detailed in (1) and (2).

Setup: In our experiments, we analyze the performance of the various strategies with respect to prediction accuracy and runtime of the stream classifiers using common stream generators (Gama et al. 2014) and real high dimensional data. On our imbalanced NSDQ dataset (Sect. 5 we also report Kappa statistics (Bifet et al. 2018), which measures classifier's performance w.r.t. class imbalance:

$$\kappa = \frac{p_0 - p_c}{1 - p_c} \quad (6)$$

where p_0 is the classifiers prequential accuracy and p_c is the probability, that a chance classifier makes a correct prediction. If the classifier is always correct, then $\kappa = 1$. If the predictions coincide with the correct predictions as often as those of the chance classifier, then $\kappa = 0$. All experiments are implemented in Python supported by the scikit-multiflow framework (Montiel et al. 2018). Code is available at <https://github.com/foxriver76/eais-si>.

Datasets: As stream generators we use SEA and LED, as detailed in Gama et al. (2014). Further, we are analyzing our NSDQ dataset described in Sect. 5 in two variations. In one version the words are encoded via *bag of words tf-idf* (NSDQ_{tf-idf}) and the second version (NSDQ_{embedd}) uses a *skip-gram* (Mikolov et al. 2013) embedding as encoding.

Algorithms: As state-of-the-art streaming benchmark classifiers we use Adaptive Robust Soft Learning Vector Quantization (RSLVQ_{ADA}) Heusinger et al. (2020), Reactive Robust Soft Learning Vector Quantization (RRSLVQ) Raab et al. (2020), Adaptive Random Forest (ARF) Gomes et al. (2017) and self-adjusting memory KNN (SAMKNN) (Losing et al. 2017b).

Parametrization: SAMKNN is parameterized with 5 neighbors and a window size of 1000 to match our scenario of Sect. 4.1.2, RSLVQ_{ADA} is parameterized with $\gamma = 0.9$ and

Fig. 4 t-SNE plot of NSDQ embeddings. Best viewed in color

Fig. 5 Popularity of #amzn hashtag on Twitter. The popularity rating is relative to the most popular hashtag on Twitter: The most popular hashtag will get 100, while a hashtag that is never used would get 0. Week from -1 to 0 is showing a forecast of the ongoing week. Graph created via <https://hashtagify.me>

two prototypes per class. RRSLVQ also uses two prototypes per class and a $\alpha = 10^{-10}$ with a window size of 100 and a statistics window of 30. ARF is applied with default settings as proposed in Gomes et al. (2017). We use the immediate (train-then-test) setting (Bifet et al. 2018) by a 5-fold cross-validation.

6.1 Experiments via random-projection dimensionality reduction

Preprocessing: The generators are low dimensional, thus we enrich every sample with meaningful (multiplied existing features) dimensions to get up to 10,000 features.

Furthermore, we evaluate the performance of the classifiers on high dimensional space compared to low k_w dimensional space, obtained by RP. As projection method, RP-VS is used. All datasets except $NSDQ_{tf-idf}$ are projected to a k_w dimensional subspace. However, $NSDQ_{tf-idf}$ only consists of 1000 dimensions, thus we project it to a 200 dimensional space as a rule of thumb.

Results: Table 1 contains the comparison of achieved accuracy scores by the different classifiers in the original and projected space. The performance stays approximately on the same level, no matter if operating in the projected or the original dataspace. In some cases, the accuracy is even better when operating in the projected space. The synthetic

Table 1 Comparison of concept drift streams w.r.t. accuracy

Algorithm	ARF	ARF	RSLVQ _{ADA}	RSLVQ _{ADA}	SAMKNN	SAMKNN	RRSLVQ	RRSLVQ
Dataspace	Original	Projected	Original	Projected	Original	Projected	Original	Projected
SEA _G	99.50 ± 0.50	99.64 ± 0.38	98.95 ± 1.70	96.75 ± 2.77	97.54 ± 1.59	98.25 ± 1.87	90.92 ± 1.37	76.91 ± 5.71
SEA _A	99.46 ± 0.41	99.77 ± 0.14	99.14 ± 1.39	98.47 ± 1.56	99.39 ± 1.21	98.97 ± 1.97	88.28 ± 0.65	68.28 ± 1.87
LED _G	99.99 ± 0.00	99.99 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	99.93 ± 0.00	99.91 ± 0.02	99.34 ± 3.57	99.18 ± 0.28
LED _A	100.0 ± 0.00	99.99 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	99.91 ± 0.02	99.94 ± 0.02	98.61 ± 1.22	98.97 ± 3.45
NSDQ _{tf-idf}	29.82 ± 0.19	29.66 ± 0.20	17.76 ± 0.10	19.03 ± 0.33	37.80 ± 0.0	36.36 ± 0.55	20.31 ± 0.09	18.41 ± 0.00
NSDQ _{embedd}	32.78 ± 0.57	34.05 ± 0.54	21.66 ± 1.10	22.77 ± 0.02	48.84 ± 0.00	46.60 ± 0.77	30.07 ± 0.58	30.79 ± 3.55

Data is projected via random projection

Table 2 Comparison of Kappa score on imbalanced NSDQ dataset

Algorithm	ARF	ARF	RSLVQ _{ADA}	RSLVQ _{ADA}	SAMKNN	SAMKNN	RRSLVQ	RRSLVQ
Dataspace	Original	Projected	Original	Projected	Original	Projected	Original	Projected
NSDQ _{tf-idf}	2.02 ± 0.23	2.87 ± 0.34	−0.94 ± 0.14	−0.86 ± 0.44	20.31 ± 0.00	17.19 ± 1.02	−0.95 ± 0.06	0.00 ± 0.00
NSDQ _{embedd}	8.17 ± 0.96	9.93 ± 0.89	5.32 ± 0.01	7.08 ± 2.59	36.33 ± 0.00	33.49 ± 0.91	8.91 ± 0.74	10.67 ± 4.60

Data is projected via random projection

Table 3 Comparison of concept drift streams w.r.t. runtime in seconds

Algorithm	ARF	ARF	RSLVQ _{ADA}	RSLVQ _{ADA}	SAMKNN	SAMKNN	RRSLVQ	RRSLVQ
Dataspace	Original	Projected	Original	Projected	Original	Projected	Original	Projected
SEA _G	9591 ± 1007	3547 ± 139	1291 ± 0	1265 ± 1	5473 ± 6986	1286 ± 1301	181717 ± 3804	30431 ± 465
SEA _A	10055 ± 925	3567 ± 75	1269 ± 1	1260 ± 1	5971 ± 9009	1409 ± 1734	184298 ± 3392	30022 ± 432
LED _G	6890 ± 726	2660 ± 211	1247 ± 0	871 ± 1	8999 ± 2313	3511 ± 1037	183374 ± 3708	29939 ± 417
LED _A	6986 ± 708	2656 ± 217	1229 ± 1	860 ± 0	7448 ± 2275	3668 ± 571	185256 ± 4037	29934 ± 482
NSDQ _{tf-idf}	12213 ± 374	5705 ± 221	488 ± 1	400 ± 2	5187 ± 34	864 ± 10	19361 ± 84	8449 ± 34
NSDQ _{embedd}	14988 ± 701	5470 ± 201	353 ± 54	311 ± 1	1669 ± 82	311 ± 1	19911 ± 1600	18335 ± 570

Data is projected via random projection. The projection costs are included

data streams do not seem to be a big challenge for our chosen classifiers. Also on the Reuters dataset, they achieve results near 100%. However, looking at our NSDQ datasets, the accuracy is very low. Notice, that NSDQ has 15 labels which makes it a lot harder compared to two classes of the other streams. By random choice we would get Accuracy around 6.66%, thus all classifier's perform clearly above random choice, but SAMKNN achieves around 37% on NSDQ_{tf-idf}, which is more than double the accuracy of RSLVQ_{ADA} and ~ 7% above ARF. On the *skip-gram* embedding the accuracy of SAMKNN is again superior with approximately 50%, while ARF is around 34% and RSLVQ_{ADA} is only around 22%. However, also our leading classifier is not able to predict more than 50% correct labels in a stream setting. Comparing the RSLVQ-variants the experiments point out that RSLVQ_{ADA} achieves better performance on stream generators while RRSLVQ is better at NASDAQ.

Due to the class imbalance in NSDQ, Table 2 shows the Cohens Kappa score of the classifiers on the two NSDQ datasets. We can see, that RSLVQ_{ADA} predicted the same label every time on the *tf-idf* dataset, and was able to distinguish minimal between classes on NSDQ_{embedd}. The same is valid for the RRSLVQ with a slightly better score at NSDQ_{embedd}. ARF was also nearly predicting only one label on NSDQ_{tf-idf} and achieved a Kappa score around 10% on the *skip-gram* version. However, SAMKNN shows, that also stream classifiers are able to distinguish between those classes, achieving a Kappa around 20% on NSDQ_{tf-idf} and up to 36% on NSDQ_{embedd} which is pretty acceptable for such an unbalanced dataset in a streaming scenario.

In Table 3 the classifiers runtime is shown. Here we are interested in time-savings between operating in the original and projected dataspace. As we would expect, the time improvements can be seen on every data stream and every classifier. However, the prototype-based RSLVQ only needs

Table 4 Comparison of concept drift streams w.r.t. accuracy

Algorithm	ARF	ARF	RSLVQ _{ADA}	RSLVQ _{ADA}	SAMKNN	SAMKNN	RRSLVQ	RRSLVQ
Dataspace	Original	Projected	Original	Projected	Original	Projected	Original	Projected
SEA _G	99.25 ± 0.51	73.45 ± 2.04	98.15 ± 1.23	74.30 ± 1.82	25.69 ± 1.40	25.82 ± 1.8	97.73 ± 1.23	74.30 ± 1.83
SEA _A	99.04 ± 0.46	57.30 ± 2.68	98.34 ± 1.23	57.34 ± 2.04	39.06 ± 1.67	39.03 ± 2.15	98.19 ± 1.21	73.35 ± 2.19
LED _G	1.00 ± 0.0	14.00 ± 0.21	1.00 ± 0.0	12.46 ± 0.66	10.0 ± 0.13	10.20 ± 0.16	1.0 ± 0.0	3.97 ± 1.0
LED _A	1.00 ± 0.0	15.19 ± 0.25	1.00 ± 0.0	12.46 ± 0.66	9.86 ± 0.58	10.11 ± 0.12	1.0 ± 0.0	4.97 ± 2.83
NSDQ _{tf-idf}	30.08 ± 0.33	24.65 ± 0.21	19.57 ± 0.12	25.22 ± 0.22	37.55 ± 1.00	21.23 ± 0.15	20.27 ± 0.07	19.06 ± 0.0
NSDQ _{embedd}	32.53 ± 0.40	28.87 ± 0.13	21.87 ± 0.98	17.11 ± 0.42	48.76 ± 0.0	24.69 ± 0.0	29.86 ± 0.48	21.02 ± 1.50

Data is projected via Incremental-PCA Ross et al. (2008)

Table 5 Comparison of Kappa score on imbalanced NSDQ dataset

Algorithm	ARF	ARF	RSLVQ _{ADA}	RSLVQ _{ADA}	SAMKNN	SAMKNN	RRSLVQ	RRSLVQ
Dataspace	Original	Projected	Original	Projected	Original	Projected	Original	Projected
NSDQ _{tf-idf}	30.07 ± 0.33	24.64 ± 0.21	−0.62 ± 0.1	−0.32 ± 0.13	19.41 ± 0.21	0.263 ± 0.0	−0.97 ± 0.0	0.27 ± 0
NSDQ _{embedd}	7.74 ± 0.65	0.35 ± 0.17	5.61 ± 1.0	0.40 ± 0.24	36.24 ± 0.0	2.16 ± 0.0	8.89 ± 0.83	0.28 ± 0.7

Data is projected via Incremental-PCA

to calculate distances between its few prototypes and new datapoints, which makes it also fast in high dimensions. Hence, time improvement is not that important here. Furthermore, more complex classifiers like SAMKNN and ARF can save lots of time. When we look at our best classifier on the two NSDQ datasets, we can save most of the time (around 80%) on NSDQ_{embedd} by operating in the projected space and do not lose significant accuracy. This can make the difference between being able to use a classifier in a real time scenario or not. The RRSLVQ time requirements are significantly larger than compared approaches because of the dimension-wise testing of the concept drift detector, making it only suitable for low dimensional streams.

6.2 Experiments via incremental-PCA dimensionality reduction

Preprocessing: Again, to evaluate Incremental-PCA on high dimensional data streams, we face the problem, that most stream generators produce low dimensional data (Gomes et al. 2017; Street and Kim 2001; Aggarwal 2014). To get a high dimensional representation of the data, we use the idea of RFF as described in Sect. 3.4.

Given the stream generators, we evaluate the performance of the baseline classifiers described above on data that has been projected via online PCA. As a comparison, we also evaluate the classifiers on the streams in the higher dimensional space, before PCA has been applied. To generate the higher dimensional data, we use RFF as described above. This time we only transform the data via RFF to

a 500 dimensional space. Due to the high dimensionality of NSDQ_{tf-idf} and NSDQ_{embedd}, the original high dimensional representation is evaluated and only synthetic data is enriched. PCA then projects real and synthetic data from a $\mathbb{R}^{500}$ to a $\mathbb{R}^{50}$ space.

Results: Table 4 shows all tested cases in the original (high dimensional space) and projected (low-dimensional space). Note, that the NSDQ streams are a lot more difficult due to more classes compared to stream generators. Overall, the classifiers perform a lot worse, when projecting the data via incremental PCA. This does not depend on the stream or even drift type. It seems that the calculated 50 eigenvectors corresponding to the largest eigenvalues do not contain enough information, to make a good classification decision for the used classifiers. Further, it seems that handling the RFF features is difficult for SAMKNN, in comparison to random features, as a drastic performance drop is also observable in the high dimensional space.

At the NASDAQ datasets, the drop in performance for ARF and SAMKNN is by far not that dramatic as with the stream generators, because the original high dimensionality is not artificially created. Further, the performance of RSLVQ_{ADA} improves by 6% and RRSLVQ remains stable for NSDQ_{tf-idf}, while at NSDQ_{embedd} the performance gap between original and projected is smaller as for non-RSLVQ methods. Given the experiments, we assume that combining the Incremental-PCA with the RSLVQ variants behaves more in the sense of a traditional application of PCA at real-world scenarios. Overall, RFF feature enrichment does not contribute to a better performance in this stream setting.

Table 6 Comparison of concept drift streams w.r.t. runtime in seconds

Algorithm	ARF	ARF	RSLVQ _{ADA}	RSLVQ _{ADA}	SAMKNN	SAMKNN	RRSLVQ	RRSLVQ
Dataspace	Original	Projected	Original	Projected	Original	Projected	Original	Projected
SEA _G	974 ± 91	658 ± 130	39 ± 1	17 ± 0.1	9537 ± 5118	793 ± 508	18748 ± 584	186 ± 5
SEA _A	1094 ± 60	590 ± 95	40 ± 1	195 ± 2	9613 ± 5098	792 ± 515	18748 ± 584	195 ± 2
LED _G	719 ± 12	1573 ± 28	109 ± 1	70 ± 1	9513 ± 5135	793 ± 516	17005 ± 27	597 ± 1
LED _A	675 ± 45	1583 ± 148	109 ± 1	70 ± 1	9434 ± 5075	786 ± 509	17005 ± 27	597 ± 1
NSDQ _{tf-idf}	5749 ± 400	690 ± 44	455 ± 13	238 ± 1	2565 ± 90	167 ± 5	22863 ± 310	402 ± 2
NSDQ _{embedd}	6546 ± 241	22 ± 1	326 ± 1	225 ± 1	883 ± 13	22 ± 1	23491 ± 924	1290 ± 3

Data is projected via Incremental-PCA. The projection costs are included

Fig. 6 RP drifts comparison of high dimensional and projected space

Table 5 shows the Cohens Kappa score of the classifiers on the two NSDQ datasets, which is required due to the imbalanced nature of the dataset. The Kappa score at ARF and SAMKNN is best at the high dimensional datasets while again the performance drops in the projected space. The Kappa scores of these two methods suggest that they are better than random chance, which is not the case for the RSLVQ variants which are around zero indicating that random guessing is comparable in performance. However, this behavior is the same with RP as shown in Table 2.

Table 6 shows the runtime of the methods in seconds. Note, that due to the high time requirements, the hardware differs from Table 3 to 6 and is therefore discussed independently. Overall, the experiments show a drastic runtime reduction of classifiers when applied to the low-dimensional space even though the projection costs are included. However, an increase in runtime is detected at ARF on LED_G and LED_A. We assume that the optimization of the ensembling process takes longer in the projected space because in the high dimensional space the data is easier which leads to faster convergence.

6.3 Comparison of concept drift sensitivity

In our next experiment, we analyze the drift detection ability in low and high dimensional spaces. For this scenario we use sparse RP and an incremental PCA (Ross et al. 2008).

The high dimensional spaces are obtained from the synthetic SEA stream (Street and Kim 2001), which consists of three numerical attributes. Due to the fact, that we should not have too high dimensional data when using PCA, we use RFF, which is suitable to obtain a relatively low dimensional data representation, while still aims to make data linearly separable. Using RFF in an online fashion (Rahimi and Recht 2008) we obtain a 500 dimensional space for the PCA setup and a 10,000 dimensional space for the RP setup at every iteration of the stream. Accordingly, PCA will project the space down to 50 dimensions, while RP will reduce the dimensionality to 1000 dimensions.

Figure 6 shows 2000 iterations of a total of 20,000 iterations. On the y-axis, we distinguish between the higher dimensional and the projected space. The filled triangle marks a true positive and an empty triangle marks a false positive. As detection algorithm, KSWIN (Raab et al. 2019) is used. We can see that in the RP setup we have a lot of false positives in the higher dimensional space. This gets better, when projecting the data down, while we are still able to detect most of the drifts. The high false positive rate in the 10,000 dimensional space occurs because the statistical test is performed 10,000 times instead of just 1,000 times, which makes it more likely to find significant differences in the data. In Fig. 7, we can see that in the PCA setting we also have more false positives in the 500 dimensional space, than

Fig. 7 PCA drifts comparison of high dimensional and projected space

in the 50 dimensional space for the same reasons. However, here we also miss a drift in the low dimensional space and vice versa. Missing a drift in the low dimensional space can happen due to the loss of information due to the projection. However, the detected drift at step 3000 in the lower dimensional space seems to be due to coincidence.

7 Conclusion

In this paper, our goal was to show, that RP can be applied in non-stationary environments. We have shown, that this can be done by using a sliding window technique, allowing the Johnson-Lindenstrauss Lemma to be also applied in non-stationary environments. Another goal was to present a new streaming dataset, which presents multiple challenges for stream classifiers, like multiclass, imbalanced classes, many non-specified concept drifts, and high dimensional data. We used this dataset to demonstrate, that RP can be used in non-stationary environments as shown in Section 4. After showing, that the usage of RP can be advantageous, and drift in high dimensional spaces can still be detected in the low dimensional space, we also wanted to check if this also counts for PCA. For this, we used common data stream generators which are low dimensional and transformed them into a medium high dimensional space of 500 dimensions by RFF.

In summary, our experiments have shown, that using RP in non-stationary environments can save lots of time on more complex classifiers. While the $RSLVQ_{ADA}$ shows a moderate improvement, the effect is strong in the case of other state-of-the-art methods like ARF and SAMKNN. Our results show, that applying RP has in general no negative impact w.r.t. accuracy on CD streams. Thus, the usage of RP on streaming data leads to enormous runtime savings and the error is bounded on window- and prototype-based classifiers. Experiments with PCA have shown, that it is not appropriate to try to detect drift in the transformed data, because many drifts that are detected in the original data are not detected in the projection. It is more appropriate to check the angle

of the principal components as in Shao et al. (2014). However, in future work, the influence of the performance of Incremental-PCA without RFF and with random features should be inspected to re-validate the results.

Compared to common stream generators, an high dimensional dataset of other domains, like our proposed NSDQ dataset, is a new challenge for stream classifiers as shown in our experiments. The dataset is useful to evaluate and compare stream classifiers for more complex real-time scenarios. In further research, it will be interesting to create a multi-labeled and multi-class Twitter dataset in NSDQ fashion, to challenge the boundaries of stream classifiers even more.

Acknowledgements We are thankful for support in the FuE program Informations- und Kommunikationstechnik of the StMWi, project OBerA, Grant No. IUK-1709-0011// IUK530/010 and ESF program WiT-HuB/2014-2020.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Achlioptas D (2001) Database-friendly random projections. In: Proc. of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, pp 274–281
- Achlioptas D (2003) Database-friendly random projections: Johnson-Lindenstrauss with binary coins. J Comput Syst Sci 66:671–687

- Aggarwal CC (2014) A survey of stream classification algorithms. In: Aggarwal CC (ed) *Data classification: algorithms and applications*. CRC Press, Boca Raton, pp 245–274
- Baena-Garcia M, del Campo-Ávila J, Fidalgo R, Bifet A, Gavalda R, Morales-Bueno R (2006) Early drift detection method. In: *Fourth international workshop on knowledge discovery from data streams*, vol 6, pp 77–86
- Bifet A, Gavalda R, Holmes G, Pfahringer B (2018) *Machine learning for data streams with practical examples in MOA*. MIT Press, Cambridge
- Bifet A, Gavalda R (2007) Learning from time-changing data with adaptive windowing. In: *Proc. of the Seventh SIAM International Conference on Data Mining*, April 26–28, 2007, Minneapolis, Minnesota, USA, pp 443–448
- Bifet A, Gavalda R (2009) Adaptive learning from evolving data streams. In: *Advances in Intelligent Data Analysis VIII, IDA 2009. Proceedings. Lecture Notes in Computer Science*, vol. 5772. Springer, pp 249–260
- Bifet A, Hammer B, Schleif FM (2019) Recent trends in streaming data analysis, concept drift and analysis of dynamic data sets. In: *27th European Symposium on Artificial Neural Networks, ESANN 2019, Bruges, Belgium, April 24–26, 2019*, pp 421–430
- Carraher LA, Wilsey PA, Moitra A, Dey S (2016) Random projection clustering on streaming data. In: *2016 IEEE 16th ICDMW*, pp 708–715
- Dasgupta S, Gupta A (2003) An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct Algorithms* 22:60–65
- Edosomwan S, Prakasan S, Kouame D, Watson J, Seymour T (2011) The history of social media and its impact on business. *J Appl Manag Entrep* 16:79–91
- Francis DP, Raimond K (2018) A random Fourier features based streaming algorithm for anomaly detection in large datasets. In: *Rajasingh EB, Veerasamy J, Alavi AH, Peter JD (eds) Advances in big data and cloud computing*. Springer Singapore, Singapore, pp 209–217
- Gama J, Zliobaite I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. *ACM Comput Surv* 46(4):1–37
- Golbeck J, Robles C, Turner K (2011) Predicting personality with social media. In: *CHI'11 extended abstracts on human factors in computing systems*. ACM, pp 253–262
- Gomes HM, Bifet A, Read J, Barddal JP, Enembreck F, Pfahringer B, Holmes G, Abdesslem T (2017) Adaptive random forests for evolving data stream classification. *Mach Learn* 106(9–10):1469–1495
- Grabowska M, Kotłowski W (2018) Online principal component analysis for evolving data streams. *Computer and Information Sciences*. Springer, Berlin, pp 130–137
- Habernal I, Ptáček T, Steinberger J (2013) Sentiment analysis in Czech social media using supervised machine learning. In: *WASSA@ NAACL-HLT*, pp 65–74
- Heusinger M, Raab C, Schleif FM (2020) Passive concept drift handling via momentum based robust soft learning vector quantization. *Advances in SOM, LVQ, clustering and data visualization*. Springer, Berlin, pp 200–209
- Johnson WB, Lindenstrauss J (1984) Extensions of Lipschitz mappings into a Hilbert space. *Contemp Math* 26:189–206
- Kaban A (2015) Improved bounds on the dot product under random projection and random sign projection. In: *Proc. of the 21th ACM SIGKDD, KDD '15*, pp. 487–496. ACM, New York, NY, USA
- Kaski S (1998) Dimensionality reduction by random mapping: fast similarity computation for clustering. In: *IJCNN98*, vol. 1, pp. 413–418. Piscataway, NJ, IEEE
- Klartag B, Mendelson S (2005) Empirical processes and random projections. *J Funct Anal* 225(1):229–245
- Kouloumpis E, Wilson T, Moore J (2011) Twitter sentiment analysis: the good the bad and the omg! In: *Fifth International AAAI conference on weblogs and social media*
- Lee JA, Verleysen M (2007) *Nonlinear dimensionality reduction*, 1st edn. Springer Publishing Company, Incorporated
- Levey A, Lindenbaum M (2000) Sequential Karhunen–Loeve basis extraction and its application to images. *IEEE Trans Image Process* 9(8):1371–1374. <https://doi.org/10.1109/83.855432>
- Li P, Hastie TJ, Church KW (2006) Very sparse random projections. In: *Proc. of the 12th ACM SIGKDD*. ACM, pp 287–296
- Losing V, Hammer B, Wersing H (2018) Tackling heterogeneous concept drift with the self-adjusting memory (SAM). *Knowl Inform Syst* 54(1):171–201
- Losing V, Hammer B, Wersing H (2017) KNN classifier with self adjusting memory for heterogeneous concept drift. in: *Proc. IEEE, ICDM*, pp 291–300
- Losing V, Hammer B, Wersing H (2017) Self-adjusting memory: How to deal with diverse drift types. In: *Proc. IJCAI 2017*, pp 4899–4903
- Lvd Maaten, Hinton G (2008) Visualizing data using t-SNE. *J Mach Learn Res* 9:2579–2605
- Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13*. Curran Associates Inc., Red Hook, NY, USA, pp 3111–3119
- Montiel J, Read J, Bifet A, Abdesslem T (2018) Scikit-multiflow: a multi-output streaming framework. *J Mach Learn Res* 19(72):1–5
- Park M, Li, H, Kim J (2016) Harrison: A benchmark on hashtag recommendation for real-world images in social networks. *CoRR arXiv: abs/1605.05054*
- Pham XC, Dang MT, Dinh SV, Hoang S, Nguyen TT, Liew AW (2017) Learning from data stream based on random projection and Hoeffding tree classifier. *DICTA 2017*:1–8
- Raab C, Heusinger M, Schleif FM (2019) Reactive soft prototype computing for frequent reoccurring concept drift. In: *Proc. of the 27. ESANN*, pp 437–442
- Raab C, Heusinger M, Schleif FM (2020) Reactive soft prototype computing for concept drift streams. *Neurocomputing*
- Rahimi A, Recht B (2008) Random features for large-scale kernel machines. In: *Platt JC, Koller D, Singer Y, Roweis ST (eds) Advances in neural information processing systems 20*, pp 1177–1184. Curran Associates, Inc. <http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf>
- Ross DA, Lim J, Lin RS, Yang MH (2008) Incremental learning for robust visual tracking. *Int J Comput Vis* 77(1–3):125–141
- Sacha D, Zhang L, Sedlmair M, Lee JA, Peltonen J, Weiskopf D, North SC, Keim DA (2017) Visual interaction with dimensionality reduction: a structured literature analysis. *IEEE Trans Vis Comput Graph* 23(1):241–250
- Schoeneman F, Mahapatra S, Chandola V, Napp N, Zola J (2017) Error metrics for learning reliable manifolds from streaming data. In: *Proc. of the 2017 SIAM International Conference on Data Mining, SIAM (2017)*, pp 750–758
- Shao J, Ahmadi Z, Kramer S (2014) Prototype-based learning on concept-drifting data streams. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pp 412–421. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/2623330.2623609>
- Sriperumbudur B, Szabo Z (2015) Optimal rates for random Fourier features. In: *Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) Advances in neural information processing systems*, vol 28. Curran Associates Inc, Red Hook, pp 1144–1152

- Street WN, Kim Y (2001) A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, pp 377–382. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/502512.502568>
- Ullah E, Mianjy P, Marinov TV, Arora R (2018) Streaming kernel PCA with $\tilde{O}(\sqrt{n})$ random features. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, pp 7322–7332. Curran Associates Inc., Red Hook, NY, USA
- Wold S, Esbensen K, Geladi P (1987) Principal component analysis. *Chemom Intell Lab Syst* 2:37–52
- Xiong K, Wang S (2019) The online random Fourier features conjugate gradient algorithm. *IEEE Signal Process Lett* 26(5):740–744. <https://doi.org/10.1109/LSP.2019.2907480>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.