Machine Learning (2021) 110:3015–3035
https://doi.org/10.1007/s10994-021-06054-y

Tensor decision trees for continual learning from drifting data streams

Bartosz Krawczyk[1]

Received: 2 March 2021 / Revised: 11 August 2021 / Accepted: 23 August 2021 /
Published online: 24 September 2021

Abstract
Data stream classification is one of the most vital areas of contemporary machine learning, as many real-life problems generate data continuously and in large volumes. However, most of research in this area focuses on vector-based representations, which are unsuitable for capturing properties of more complex multi-dimensional structures, such as images and video sequences. In this paper, we propose a novel methodology for learning adaptive decision trees from data streams of tensors. We introduce Chordal Kernel Decision Tree for continual learning from tensor data streams. In order to maintain the tensor characteristics, we propose to train and update classifiers in the kernel space designed to work with tensor representation. We use chordal distance to compute similarities between tensors and then apply it as a new feature space in which decision trees are trained. This allows for a direct decision tree induction on tensors. In order to accommodate the streaming and drifting nature of data, we propose a concept drift detection scheme based on tensor representation. It allows us to reconstruct the kernel feature space every time when change is detected. The proposed approach allows for fast and efficient induction of decision trees on streaming data with tensor representation. Experimental study, conducted on 4 real-world and 52 artificial large-scale tensor data streams, shows that using the native tensor feature space leads to more accurate classification than outperforms the vectorized representations.

Keywords Data stream mining · Continual learning · Concept drift · Online learning · Decision trees

1 Introduction

Learning from data streams is one of the most rapidly developing fields in the contemporary machine learning (Sun 2008; Ditzler et al. 2015). This is motivated by a plethora of real-world applications in which data arrives continuously and floods the system. This calls

Editors: João Gama, Alípio Jorge, Salvador García.

✉ Bartosz Krawczyk
bkrawczyk@vcu.edu

[1] Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA

for developing new algorithms that are able to handle the ever-growing data volume and constantly update their structure within time and resource limits. Additionally, data streams may be subject to changes over time, a phenomenon known as concept drift (Gama et al. 2014). Such a change point must be detected as soon as possible in order to handle the drift appropriately and allow for fast recovery of the system. Data streams are strongly connected with the recently emerging paradigm of continual learning, where it is assumed that the machine learning model must be capable of continuous self-improvement and accumulation of new knowledge (Parisi et al. 2019). It is interesting to note that data streams are de facto a task-free continual learning scenario (Aljundi et al. 2019).

Vast majority of data stream mining algorithms are designed only for vector representation of input data. This representation is not a proper one for many real-world problems that generate multi-dimensional data with dependencies between different dimensions, such as computer vision (Yang et al. 2018) or social networks (Nakatsuji et al. 2017). Although one may easily vectorize such information, it will lead to loss of information, as relationships between factors in the input space will not be preserved (Gu et al. 2018). In order to overcome this limitations a tensor representation has been proposed, where input data is stored as multi-dimensional cubes that preserve the dependencies between factors (Lathauwer 2009; Fu et al. 2015). Tensors gained popularity in various areas of machine learning and data mining (Sidiropoulos et al. 2017; Maruhashi et al. 2018), but their application to data streams is very limited. At the same time, many modern data sources generate multi-dimensional data streams (Mardani et al. 2015) and these areas may definitely benefit from dedicated tensor-based streaming algorithms (Shin et al. 2017; Song et al. 2017). Most of existing works focus on tensor factorization (Smith et al. 2018), using stochastic descent approaches (Mardani et al. 2015), Tucker model (Sun et al. 2008), and online canonical polyadic decomposition (Smith et al. 2018). At the same time, best to our knowledge, the field of tensor classification has not been studied in the data stream setup, especially in the context of concept drift. This paper aims at bridging this gap and proposing an efficient framework for data stream classification with tensor input.

1.1 Goal

To propose a novel continual decision tree induction technique that allows for learning from drifting data streams using tensor-based data representation.

1.2 Motivation

Among classifiers dedicated to data streams, decision trees have gained a significant attention, due to their excellent capabilities for incremental learning by creating new splits with arriving instances, high classification accuracy, and low model complexity (Ditzler et al. 2015). However, existing decision trees for data streams work only with vector representation. This limits their applicability to modern data sources, such as texts or images. Vectorization of such data leads to a significant loss of information. Thus it is beneficial to use tensor-based representation of such data that maintains all the properties of such complex data. However, current techniques dedicated for tensor classification are not suitable for data streaming scenarios, nor poses any mechanisms to handle the presence of concept drift. The same holds for modern deep learning architectures that, while being extremely effective for static tensor data, cannot handle velocity and rapidly evolving nature of data streams (Sahoo et al. 2018).

1.3 Overview

In this paper, we propose a novel framework classifying data streams with tensor representation. We introduce a decision tree learning scheme capable of handling tensors directly, without a need for vectorization. at the same time, our proposal maintains all the advantages of decision trees. We achieve this by training classifiers in the similarity space that is defined by a kernel using tensor representation. Chordal distance allows to measure a similarity between two tensors and may be used to construct a kernel feature space, which in turns allow for induction of a decision tree directly from tensors. Additionally, we propose a concept drift detection scheme working with tensor representation. It allows to effectively detect the moment of change and update our model in two ways: (1) by reconstructing the kernel feature space using new instances; and (2) by retraining the decision tree on the current concept and new feature space. Experimental study shows the efficacy of the proposed approach and its wide usability in various data stream classification scenarios, where tensor representation is required.

1.4 Main contributions

This paper offers following insights into learning from drifting data streams with complex data:

- *Chordal Kernel Decision Tree* We propose a novel decision tree classifier (CKDT) for continual learning from drifting data streams with data arriving in tensor form. CKDT is a full adaptive classifier, capable of both continual accumulation of new knowledge from arriving tensors, as well as flexible adaptation to drifts in the stream, when previously learned concepts become outdated. CKDT uses McDiarmid's inequality to control the continual splitting procedure from streaming tensor data.
- *Adaptive tensor kernel similarity space* We introduce a kernel similarity space for continual induction of decision trees from tensor data streams. A subsampled kernel is used to create a new tensor-based representation that allows for continual learning from tensor data streams. We present a mechanism for rebuilding the kernel space whenever concept drift occurs, allowing for adaptive feature crafting from evolving data.
- *First concept drift detector for tensors* We propose a simple, yet effective tool for monitoring properties of tensors incoming from the data stream. This allows us for early detection of any changes in tensor properties, allowing for cost-efficient adaptation of CKDT whenever streams becomes subject to significantly strong changes. The proposed drift detector works directly on tensor representation of data.
- *Experimental study* We provide a detailed experimental benchmark on drifting tensor data streams, comparing the proposed approach with three state-of-the-art method for incremental tensor classification. We use 4 real-world and 52 artificial tensor data stream benchmarks that capture various domains and learning difficulties.

2 Background

2.1 Learning from data streams

We will now provide a short background for data stream setting in the context of machine learning.

Definition 1 (Data stream) Data stream is a sequence $< S_1, S_2, \ldots, S_n, \ldots >$, where each element S_n is a new instance arriving over time. Each instance in the stream is independent and randomly drawn from a stationary probability distribution $\Psi_n(\mathbf{x}, y)$. Data stream is a task-free continual learning problem (Aljundi et al. 2019).

Definition 2 (Concept drift) Concept drift is a phenomenon that influences estimated decision rules or classification boundaries, reducing or voiding their relevance to the new state of the stream. Real concept drift influences the conditional probabilities $p_j(y|\mathbf{x})$ over time.

Concept drift has crucial impact on the learning system and must be handled as soon as it occurs (Gama et al. 2014). There are three main approaches for handling this learning difficulty. The first one relies on an external tool, known as concept drift detector. It monitors the properties of stream and informs when a significant change takes place in order to rebuilt the model. This solution is often combined with decision trees. The second one uses a sliding window that keeps most recent instances in the temporal memory, using them as the current representation of the stream. The third one relies on online classifiers and ensemble models (Krawczyk et al. 2017) that adapt to changes on their own, resulting in an implicit drift detection.

2.2 Tensors in machine learning and classification

We will define now the basic notations for representation and classification of data coming in the form of tensors.

Definition 3 (Tensor) A tensor is a L-dimensional cube of real valued data, where each individual dimension represents a different factor in the input data space:

$$\mathcal{A} \in \mathfrak{R}^{N_1 \times N_2 \times \ldots N_L} \tag{1}$$

The j-mode of the K-th order tensor (tensor order standing for its number of directions/dimensionality) is a vector that is calculated from $\mathcal{A}$ by manipulating selected dimension $k \in \{1, 2, \cdots, N_j\}$, while remaining dimensions are intact.

Definition 4 (Tensor flattening) A j-mode tensor flattening (known also as tensor matricization) is matrix $\mathbf{A}_{(j)}$ for which its columns are j-mode vectors of $\mathcal{A}$:

$$\mathbf{A}_{(j)} \in \mathfrak{R}^{N_j \times (N_1 N_2 \ldots N_{j-1} N_{j+1} \ldots N_L)} \tag{2}$$

The j-th index is a row index of $\mathbf{A}_{(j)}$, while a product of all remaining L-1 indices is its column index.

Definition 5 (Tensor product) A p-mode product of a tensor $\mathcal{A} \in \mathfrak{R}^{N_1 \times N_2 \times \ldots N_L}$ with matrix $\mathbf{M} \in \mathfrak{R}^{Q \times N_p}$ creates a tensor $\mathcal{B} \in \mathfrak{R}^{N_1 \times N_2 \times \ldots N_{p-1} \times Q \times N_{p+1} \times \ldots N_L}$ with elements:

$$\begin{aligned} \mathcal{B}_{n_1 n_2 \ldots n_{p-1} q n_{p+1} \ldots n_L} &= \left(\mathcal{A} \times_p \mathbf{M}\right)_{n_1 n_2 \ldots n_{p-1} q n_{p+1} \ldots n_L} \\ &= \sum_{n_p=1}^{N_p} a_{n_1 n_2 \ldots n_{p-1} n_p n_{p+1} \ldots n_L} m_{q n_p}. \end{aligned} \tag{3}$$

where $a_{n_1 n_2 \ldots n_L}$ is an element of $\mathcal{A}$ at index $(n_1, n_2, ..., n_L)$ and analogously $m_{q n_p}$ is an element of $\mathbf{M}$ at index (q, n_p).

The *p-mode* product can be represented in an equivalent manner as flattened tensors $\mathbf{A}_{(p)}$ and $\mathbf{B}_{(p)}$. Assuming the following holds:

$$\mathcal{B} = \mathcal{A} \times_p \mathbf{M} \tag{4}$$

then

$$\mathbf{B}_{(p)} = \mathbf{M}\, \mathbf{A}_{(p)} \tag{5}$$

Each distinctive tensor flattening creates an unique matrix with specific properties. Therefore, by analyzing each flattening $\mathbf{A}_{(j)}$ we obtain an unique perspective on $\mathcal{A}$ from j-th dimension. We will use this property to construct a tensor-based kernel for data stream representation, which will be discussed in details in the next subsection.

Definition 6 (Singular Value Decomposition) SVD is a procedure for analyzing the properties of each flattening as follows:

$$\begin{aligned} \mathbf{A}_{(j)} &= \mathbf{S}^{(j)} \mathbf{V}^{(j)} \mathbf{D}^{(j)^T} = \sum_{i=1}^{R_{A_{(j)}}} v_i^{(j)} \mathbf{s}_i^{(j)} \mathbf{d}_i^{(j)^T} \\ &= \begin{bmatrix} \mathbf{S}_{\mathbf{A},1}^{(j)} & \mathbf{S}_{\mathbf{A},2}^{(j)} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{\mathbf{A},1}^{(j)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{D}_{\mathbf{A},1}^{(j)^T} \\ \mathbf{D}_{\mathbf{A},2}^{(j)^T} \end{bmatrix}. \end{aligned} \tag{6}$$

where $_{\mathbf{A},1}$ and $_{\mathbf{A},2}$ denote respectively indices of block matrices related to the kernel and null spaces of $\mathbf{A}_{(j)}$. $\mathbf{S}_{\mathbf{A},1}^{(j)}$ and $\mathbf{D}_{\mathbf{A},1}^{T(j)}$ are unitary matrices of the kernel of $\mathbf{A}_{(j)}$. $\mathbf{V}_{\mathbf{A},1}^{(j)}$ is a diagonal matrix with $R_{\,A}$ non-zero elements.

By assuming this definition of SVD, it follows that:

$$\mathbf{D}_{\mathbf{A},1}^{(j)^T} \mathbf{D}_{\mathbf{A},1}^{(j)} = \mathbf{I}_{R_\mathbf{A} \times R_\mathbf{A}} \tag{7}$$

Analogous properties are preserved for j-th mode flattening of the tensor $\mathcal{B}$. However, its rank may be different and thus we will denote it as $R_{\,B}$.

In this work, we focus the task of tensor classification, i.e., assigning a class label to an input tensor (Li and Schonfeld 2014).

Definition 7 (Tensor classification) This task aims at creating a classifier defined as a function Ψ with domain $\mathcal{A}$ and codomain $\mathcal{M}$:

$$\Psi : \mathcal{A} \rightarrow \mathcal{M}, \tag{8}$$

where $\mathcal{M} = \{1, \cdots, M\}$ stands for a set of class labels.

2.3 Related works for streaming tensor analysis

Streaming tensors have been considered in the literature mainly from the perspective of a singular tensor that evolves over time (Yang et al. 2021). This may include changes in existing dimensions / factors (Chhaya et al. 2020), or emergence of new ones over time (Letourneau et al. 2018). CP decomposition has been successfully used for streaming tensors, either based on simultaneous diagonalization, or weighted least squares that track the online third-order decomposition. (Rambhatla et al. 2020) Other approaches use grid division for large streaming tensors and using local factorization independently for each subtensor (Gujral et al. 2020). OnlineCP (Zhou et al. 2016) incrementally tracks CP decomposition of streaming tensor with arbitrary modes. There exist also Tucker decomposition methods for online tensor analysis that can be effectively used under streaming conditions (Sun et al. 2020). From the data stream mining perspective, there exist a plethora of effective classification and drift detection methods, but all of them are dedicated to shallow vector representations and therefore cannot properly capture multi-dimensional relationships in tensor data(Pinage et al. 2020)(Zyblewski et al. 2021).

3 Decision tree learning for tensor data streams

3.1 Decision trees in the era of deep learning

Deep learning have dominated the world of learning from complex and high-dimensional data, offering unparalleled predictive and generative capabilities power. However, research in traditional (shallow) machine learning algorithms is still as vibrant as ever, due to a number of limitations of current deep learning architectures in specific learning scenarios. This is especially visible for learning from data streams, where existing deep architectures have difficulties with handling the presence of concept drift (Sahoo et al. 2018), or their adaptation mechanisms, while well-designed, are too slow for high-speed data streams (Ashfahani and Pratama 2019). Decision trees are well-known and attractive learning algorithms for data streams, offering low computational cost with excellent adaptation capabilities to concept drift (Gomes et al. 2019). Furthermore, they are explainable and interpretable models, offering white-box approach for streaming data. While their predictive power is weak on their own, they can be efficiently combined in ensemble architectures, leading to significant increase in their accuracy (Krawczyk et al. 2017). All this factors motivate us to develop novel decision tree model that is capable of learning from tensor data streams under concept drift.

3.2 Proposed algorithm overview

We present the details of Chordal Kernel Decision Tree (CKDT) for continual learning from tensor data streams. We discuss the used decision tree model for unbounded data

streams, the usage of kernel feature space designed for working with tensor representation, as well as concept drift detection from tensor data. Overview of the proposed CKDT algorithm is presented in pseudo-code form in Fig. 1.

3.3 Decision tree induction from streaming data

Decision tree induction algorithms for data streams are based on Hoeffding inequality in order to determine what number of new instances is sufficient to conduct a new split. Recent study highlighted the existing flaws in the Hoeffding bound (Rutkowski et al. 2013), showing its potential for incorrect calculations. Therefore, in this work we use McDiarmid's inequality for decision tree induction from streaming data. It can be seen as a generalized version of Hoeffding's inequality, more capable of handling various types of input data and measuring the split quality.

Theorem 1 (McDiarmid's Theorem) *We define $X_1, \cdots, X_n$ as a set of independent random variables and define a function $f(x_1, \cdots, x_n)$ that fulfills inequality :*

Algorithm 1 Chordal Kernel Decision Tree Induction

procedure CKDT(S) ▷ Build CKDT on a data stream
  Let CKDT be a decision tree with leaf l_1
  for $\mathcal{A} \leftarrow 1$ to $|S|$ **do** ▷ For each tensor coming from the stream
    $K_j(\mathcal{A}, \mathcal{B})$ ▷ Transform the new tensor to Chordal space
    for $\mathcal{B}_i \in W$ **do** ▷ For each tensor in the sliding window of CKDT
      $\mathbf{F}_i = D^2_{ch}(\mathcal{A}, \mathcal{B}_i)$ ▷ Calculate Chordal distance
      span new feature space with each $\mathbf{F}_i$
      l $sort(\mathcal{A}, y)$ ▷ Sort tensor $\mathcal{A}$ to leaf l
      $k \leftarrow y$ ▷ get class label
      for j F_j to $|\mathbf{F}|$ **do** ▷ For each feature
        label l with majority class of tensors at l
        if all tensors at l do not belong to the same class **then**
          $\Delta g^G_{i_1}(S)$ null ▷ best feature
          $\Delta g^G_{i_2}(S) \leftarrow$ null ▷ second best feature
          for j F_j to $|\mathbf{F}|$ **do** ▷ For each feature
            Gini getGini(F_j) ▷ Eq. 3.11
            $\Delta g^G_{i_1}(S)$ getBest($\Delta g^G_{i_1}(S)$,Gini)
            $\Delta g^G_{i_2}(S)$ getBest($\Delta g^G_{i_2}(S)$,Gini)
            $\epsilon = \sqrt{\frac{8 \ln(1/\delta)}{n(S)}}$ ▷ McDiarmid's bound
            if $\Delta g^G_{i_1}(S) - \Delta g^G_{i_2}(S) > \epsilon$ **then**
              replace l with a new leaf that splits on F_j
              for each branch of split **do**
                add a new branch l_m

Fig. 1 Pseudocode of the proposed Chordal Kernel Decision Tree

$$\begin{aligned} &\sup_{x_1,\cdots,\hat{x_i}} |f(x_1,\cdots,x_i,\cdots,x_n) - f(x_1,\cdots,\hat{x_i},\cdots,x_n)| \\ &\leq c_i, \forall_{i=1,\cdots,n}. \end{aligned} \tag{9}$$

For any $\epsilon > 0$ the following is true:

$$\begin{aligned} &Pr\big(f(X_1,\cdots,X_n) - E\big[f(X_1,\cdots,X_n)\big] \geq \epsilon\big) \\ &\leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n} c_i^2}\right) = \delta. \end{aligned} \tag{10}$$

McDiarmid's inequality can be used in combination with any splitting measure to estimate the lowest number of instances n sufficient to conduct a split when new data arrives. It has been shown to work well with Gini gain (Rutkowski et al. 2013), thus we use this metric. Gini gain is defined as:

$$\Delta g_i^G(S) = g^G(S) - \sum_{q\in\{L,R\}} \frac{n_{q,i}(S)}{n(S)}\left(1 - \sum_{k}^{K}\left(\frac{n_{q,i}^k(S)}{n_{q,i}(S)}\right)^2\right), \tag{11}$$

where S is a set of instances in the current tree node, L and R are left and right child nodes, $n_{q,i}(S)$ is the number of instances in the current node that will go to q-th child node if the split will be conducted on i-th feature, and $n_{q,i}^k(S)$ is the number of instances from k-th class that will be passed to q-th child node if the split will be conducted on i-th feature.With this we may formulate McDiarmid's inequality for computing and comparing Gini gains for any two selected features.

Theorem 2 (McDiarmid's Inequality for Gini Gain) *Let $\Delta g_h^G(S)$ and $\Delta g_i^G(S)$ be the Gini gain values (see Eq. 11) for h-th and i-th considered feature. If the condition is satisfied:*

$$\Delta g_h^G(S) - \Delta g_i^G(S) > \sqrt{\frac{8\ln(1/\delta)}{n(S)}}, \tag{12}$$

then the inequality holds with probability of $1-\delta$ or higher:

$$E[\Delta g_i^G(S)] > E[\Delta g_h^G(S)]. \tag{13}$$

Theorem 3 (McDiarmid's Split Criterion for Gini Gain) *We assume that $\Delta g_{i_1}^G(S)$ and $\Delta g_{i_2}^G(S)$ are the metric values for features with respectively highest and second highest Gini gain. If the condition is satisfied:*

$$\Delta g_{i_1}^G(S) - \Delta g_{i_2}^G(S) > \sqrt{\frac{8\ln(1/\delta)}{n(S)}}, \tag{14}$$

then following Theorem 2, with the probability equal to $(1-\delta)^{d-1}$ the following statement is true:

$$i_1 = \arg\max_{i=1,\cdots,d} \big\{E[g_i^G(S)]\big\}, \tag{15}$$

where d is the number of features and i_1-th feature is selected to split the current node.

3.4 Induction of decision trees in kernel feature space

Existing decision tree induction algorithms, including the presented one using McDiarmid's Inequality, work only with vector inputs. Therefore, it is not possible to apply them directly on tensor data without conducting vectorization. In order to alleviate this drawback and extend the applicability of decision trees to tensor data streams, we propose to conduct the tree induction procedure in an alternative feature space. We need a simple, yet efficient representation of tensors that will maintain their multi-dimensional properties and relationships among different factors. In this paper, we propose to construct the new feature space using kernels.

A kernel $\mathcal{K}$ can be used to transform the original feature space into a projected space $\varphi_{\mathcal{K}}(\mathcal{X})$ such that $\mathcal{K}(x,y) = \langle \varphi_{\mathcal{K}}(x), \varphi_{\mathcal{K}}(y) \rangle$. Kernels are tricky to use in data stream scenarios, as they require a computation of the whole Gram matrix, which is of size $\mathcal{O}(N^2)$. In order to speed up the computations, one may use a random sampling of the input instances to create a new projected feature space. By sampling s instances from the stream, one may create a subsampled kernel:

$$\varphi_{\mathcal{K}}^{\text{rand}}(x) = [\mathcal{K}(x, x^1), ..., \mathcal{K}(x, x^s)]^T. \tag{16}$$

One must note that this is fundamentally different from sampling the incoming instances from the stream, as all of them will be used for decision tree induction and incremental update - the subsample is only used for a faster computation of the new feature space. This allows for a significant reduction of feature space projection computational complexity.

Now one required a proper kernel that is capable of working directly with tensor representation. For this, we propose to use chordal distance kernel, capable of returning pure tensor-based similarities that will allow us to span a new feature space for the decision tree induction.

3.5 Chordal distance kernel for tensor similarity

Definition 8 (Chordal distance) A chordal distance (Signoretto et al. 2011) is defined as a similarity between two tensors represented by their *j-th* flattened mode matrices $\mathbf{A}_{(j)}$ and $\mathbf{B}_{(j)}$:

$$D_{ch}^2\left(\mathbf{A}_{(j)}, \mathbf{B}_{(j)}\right) = D_F^2\left(\Pi_{\mathbf{A}_{(j)}}, \Pi_{\mathbf{B}_{(j)}}\right) = \left\| \Pi_{\mathbf{A}_{(j)}} - \Pi_{\mathbf{B}_{(j)}} \right\|_F^2 \tag{17}$$

where $\Pi_{\mathbf{A}_{(j)}}$ stands for a projection matrix of $\mathbf{A}_{(j)}$:

$$\Pi_{\mathbf{A}_{(j)}} = \mathbf{D}_{\mathbf{A},1}^{(j)} \mathbf{D}_{\mathbf{A},1}^{T(j)} \tag{18}$$

Then one may insert Eq. 17 into Eq. 18, obtaining:

$$D_{ch}^2\left(\mathbf{A}_{(j)}, \mathbf{B}_{(j)}\right) = \left\| \mathbf{D}_{\mathbf{A},1}^{(j)} \mathbf{D}_{\mathbf{A},1}^{T(j)} - \mathbf{D}_{\mathbf{B},1}^{(j)} \mathbf{D}_{\mathbf{B},1}^{T(j)} \right\|_F^2 \tag{19}$$

Definition 9 (Chordal kernel) A chordal distance-based kernel (Signoretto et al. 2011) can be formulated as follows:

$$
\begin{aligned}
K_j(\mathcal{A},\mathcal{B}) &= \exp\left(-\frac{1}{2\sigma^2}D^2_{ch}\left(\mathbf{A}_{(j)},\mathbf{B}_{(j)}\right)\right)\\
&= \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{D}^{(j)}_{\mathbf{A},1}\mathbf{D}^{T(j)}_{\mathbf{A},1}-\mathbf{D}^{(j)}_{\mathbf{B},1}\mathbf{D}^{T(j)}_{\mathbf{B},1}\right\|^2_F\right).
\end{aligned}
\tag{20}
$$

which allows to formulate a kernel for a L-dimensional tensor product (Cyganek et al. 2015):

$$
\begin{aligned}
K(\mathcal{A},\mathcal{B}) &= \prod_{j=1}^{L} K_j(\mathcal{A},\mathcal{B})\\
&= \prod_{j=1}^{L} \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{D}^{(j)}_{\mathbf{A},1}\mathbf{D}^{T(j)}_{\mathbf{A},1}-\mathbf{D}^{(j)}_{\mathbf{B},1}\mathbf{D}^{T(j)}_{\mathbf{B},1}\right\|^2_F\right)
\end{aligned}
\tag{21}
$$

Computation of Eq. 21 requires computation of $2{\cdot}L$ SVD decompositions. This makes it prohibitive to be used in the considered scenario of learning from data streams, as new tensors will continuously arrive and latency in their processing must be avoided. However, one may simplify this computation as follows. We start by denoting the squared norm in Eq. 19 as:

$$
\|\mathbf{P}-\mathbf{Q}\|^2 = Tr\left(\mathbf{P}^T\mathbf{P}\right) - 2Tr\left(\mathbf{P}^T\mathbf{Q}\right) + Tr\left(\mathbf{Q}^T\mathbf{Q}\right)
\tag{22}
$$

where $Tr(.)$ stands for matrix trace, and $\mathbf{P}$ and $\mathbf{Q}$ are defined as:

$$
\mathbf{P} = \mathbf{D}^{(j)}_{\mathbf{A},1}\mathbf{D}^{T(j)}_{\mathbf{A},1}, \quad \mathbf{Q} = \mathbf{D}^{(j)}_{\mathbf{B},1}\mathbf{D}^{T(j)}_{\mathbf{B},1}
\tag{23}
$$

Matrices $\mathbf{P}$ and $\mathbf{Q}$ are of the same size. We may supply this to the first term in Eq. 22:

$$
\begin{aligned}
Tr\left(\mathbf{P}^T\mathbf{P}\right) &= Tr\left(\left(\mathbf{D}_\mathbf{A}\mathbf{D}^T_\mathbf{A}\right)^T\left(\mathbf{D}_\mathbf{A}\mathbf{D}^T_\mathbf{A}\right)\right)\\
&= Tr\left(\mathbf{D}_\mathbf{A}\underbrace{\mathbf{D}^T_\mathbf{A}\mathbf{D}_\mathbf{A}}_{\mathbf{I}}\mathbf{D}^T_\mathbf{A}\right) = Tr\left(\mathbf{D}_\mathbf{A}\mathbf{D}^T_\mathbf{A}\right)\\
&= Tr\left(\mathbf{D}^T_\mathbf{A}\mathbf{D}_\mathbf{A}\right) = R_\mathbf{A}.
\end{aligned}
\tag{24}
$$

Analogously, this holds for the third term in Eq. 22:

$$
Tr\left(\mathbf{Q}^T\mathbf{Q}\right) = R_\mathbf{B}
\tag{25}
$$

The second term in Eq. 22 can be expanded accordingly:

$$
\begin{aligned}
Tr\left(\mathbf{P}^T\mathbf{Q}\right) &= Tr\left(\left(\mathbf{D}_\mathbf{A}\mathbf{D}^T_\mathbf{A}\right)^T\mathbf{D}_\mathbf{B}\mathbf{D}^T_\mathbf{B}\right)\\
&= Tr\left(\mathbf{D}_\mathbf{A}\mathbf{D}^T_\mathbf{A}\mathbf{D}_\mathbf{B}\mathbf{D}^T_\mathbf{B}\right) = Tr\left(\mathbf{D}^T_\mathbf{A}\mathbf{D}_\mathbf{B}\mathbf{D}^T_\mathbf{B}\mathbf{D}_\mathbf{A}\right)\\
&= Tr\left(\left(\underbrace{\mathbf{D}^T_\mathbf{B}\mathbf{D}_\mathbf{A}}_{\mathbf{G}}\right)^T\underbrace{\mathbf{D}^T_\mathbf{B}\mathbf{D}_\mathbf{A}}_{\mathbf{G}}\right) = Tr\left(\mathbf{G}^T\mathbf{G}\right).
\end{aligned}
\tag{26}
$$

These transformations of three terms allows us to write Eq. 22 as:

$$D^2_{ch}\left(\mathbf{A}_{(j)}, \mathbf{B}_{(j)}\right) = R_{\mathbf{A}} + R_{\mathbf{B}} - 2\,Tr\left(\mathbf{G}^T_{(j)}\mathbf{G}_{(j)}\right) \tag{27}$$

where

$$\mathbf{G}_{(j)} = \mathbf{D}^{T(j)}_{\mathbf{B},1}\mathbf{D}^{(j)}_{\mathbf{A},1} \tag{28}$$

Obtaining this allows us to significantly speed-up computations of chordal distance and related kernel, as Eq. 27 and Eq. 28 have significantly lower computational complexity than Eq. 22. This is because only $\mathbf{G}_{(j)}$ needs to be computed after carrying out SVD decompositions of *j-th* mode flattening of tensors $\mathcal{A}$ and $\mathcal{B}$, respectively. In order to obtain the chordal kernel, we need to repeat these computations L times (to account for the dimensionality of tensors).

This, combined with the input subsampling (subsampled kernel) presented in Eq. 16 makes the computational cost of tensor-based feature space spanning via kernel projections suitable for data stream scenarios.

4 Concept drift detection in tensor data streams

Having defined the used decision tree induction method, creation of a tensor-based feature space for it, and a proper kernel for computing similarities between tensors, we need a concept drift detector fitting this framework. Most of existing drift detectors are based on either statistical properties of new vectors arriving from the stream, or on the error of classifiers (Gama et al. 2014). The former ones cannot be directly used for tensor data, while the latter ones are criticized as in real-world scenario we do not have an instant oracle access to classifier error. Therefore, we propose a concept drift detection method based on tensor properties.

For drift detection, we need to keep a window of w most recent tensors and use them for comparison with the newest incoming tensor, to check if it still falls into the current concept. We propose to conduct drift detection using j-mode tensor flattening (see Eq. 2), computing it for all tensors in the window and the recently arrived one. Mean ($\overline{\mathbf{A}}_{(j)} = \frac{1}{w}\sum_{i=1}^{w}\mathbf{A}_{(j)}{}^{(i)}$) and standard deviation ($\sigma^2 = \frac{1}{w-1}\sum_{i=1}^{w}(\mathbf{A}_{(j)}{}^{(i)} - \overline{\mathbf{A}}_{(j)})^2$) of these flattening matrices can be used to describe the current concept. When a new tensor arrives, its j-mode tensor flattening can be used to check how well it fits the current concept using a popular 3σ rule, allowing for a two-level signal output with alarm level and drift detection level:

$$\Theta_{alarm} : \|\mathbf{A}_{(j)}{}^{(new)} - \overline{\mathbf{A}}_{(j)}\| \geq \sigma \tag{29}$$

$$\Theta_{drift} : \|\mathbf{A}_{(j)}{}^{(new)} - \overline{\mathbf{A}}_{(j)}\| \geq 3\sigma \tag{30}$$

We use these two-level signals for two important actions in our tensor data stream classification framework.

4.1 Spanning new feature space after drift

As we use a subsample of tensors to span the kernel similarity-based feature space for decision tree induction, we must take into consideration that after the concept drift the

current projection may no longer be representative. Therefore, we need to span a new projection, which imposes additional computational cost on the system. As it is not feasible to do this after every new tensor arrives, we propose to combine this with the drift detector. Whenever an alarm signal is being raised, we start collecting the incoming tensors in a temporal buffer. Then, when changes become significant and a drift alarm is being raised, we randomly subsample this buffer (see Eq. 16) and use it to span a new feature space with chordal kernel. This approach significantly reduces the number of times when we need to recompute similarity-based feature projections. The entire buffer (all stored tensors) will be then used to train a new decision tree in the newly spanned kernel feature space.

4.2 Updating the decision tree classifier after drift

As decision trees do not have in-built mechanisms for handling concept drift, they are combined with drift detector to guide their update. When concept drift occurs, it is less computationally expensive to discard the old classifier and build a new one on the current concept than to try to adapt the pre-existing tree structure. As after the alarm signal has been raised we already collect incoming tensors for calculating new feature space, we may use them as well for training a new decision tree. While the new feature space is created using only a subsample of tensors from the buffer, the new decision tree is trained using all of the stored tensors. When a drift alarm is being raised, we train a new decision tree using newly created kernel feature space, discard the old model, and replace it with the new decision tree. Then we may discard all the tensors stored in the temporal buffer, as they will not be needed.

5 Experimental study

In this section, we present the experimental evaluation of the proposed framework for tensor data stream classification with decision trees. We conduct two independent experiments: (i) on large-scale real-world tensor benchmarks that have streaming characteristics in order to examine the usability of the kernel feature space for training decision trees; (ii) on artificial datasets with injected specific type of concept drift in order to evaluate the scalability of the proposed framework to growing number of tensor factor dimensionality.

The experimental study was designed to answer the following research questions:

RQ1: Does the proposed chordal kernel-based decision tree is capable of more accurate classification of tensor data streams than the state-of-the-art reference methods?
RQ2: Does the proposed online kernel transformation of tensors and training decision trees in the kernel feature space do not impose prohibitive computational costs on the classification system?
RQ3: How does the proposed kernel-based decision tree handle increasing tensor dimensionality (number of factors)?
RQ4: Does the proposed method can efficiently handle various types of concept drifts present in tensor data streams?

5.1 Tensor benchmarks for data stream classification

We have selected four real-world tensor datasets that display streaming characteristics, as well as generated 52 artificial tensor datasets with injected specific types of concept drift. Their details are presented below and in Table 1.

5.1.1 Chicago Crime (CC)

A collection of crime reports in the city of Chicago, ranging from January 1st, 2001 to December 11th, 2017. We split the original tensor into 5 000 000 separate small tensors, each representing a single crime. The classification task is to predict the crime based on remaining information from the report.

5.1.2 Yahoo Music (YM)

A collection of user ratings of music items in Yahoo! services. Concept drift is strongly embedded, as data reflects the changes in music distribution platforms and market needs. We subsampled the dimensionality each of individual factors to make it feasible for a single machine computation. Original task was to predict the user rating of an item. We discretized this task into class labels via average ranking values for items.

5.1.3 Street View House Numbers (SVHN)

A collection of 640 420 images representing house numbers, each digit displayed individually in a form of 32x32x3 RGB color image tensor. The classification task is digit recognition.

5.1.4 CIFAR-100 (C100)

A collection of 60 000 images, each stored as 32x32x3 RGB color image tensor. The task is to predict to which group target image belongs to.

Table 1 Details of used real-world and artificial tensor benchmarks

Name	# Tensors	Dimensionality	# Classes
Real-world tensor data streams			
CC	5,000,000	6186 x 24 x 77 x 32	31
YM	1,710,000	625 x 844 x 101	5
SVHN	640,420	32 x 32 x 3	10
C100	60,000	32 x 32 x 3	100
Artificial tensor data stream generators			
ST_{nd}	2,000,000	$100^{[3;15]}$	2
ST_{gd}	2,000,000	$100^{[3;15]}$	2
ST_{id}	2,000,000	$100^{[3;15]}$	2
ST_{sd}	2,000,000	$100^{[3;15]}$	2

5.1.5 SimTensor (ST)

An artificial tensor generator (Fanaee-T and Gama 2016) that we are using to evaluate the impact of different factor dimensionality in tensor data streams on decision tree induction. Each artificial benchmark consists of 2 000 000 tensors and each tensor factor has 100 values. We investigate tensor factor dimensionality $\in$ [3;15]. By combining it with MOA (Bifet et al. 2010) functionality, we are able to create four datasets with distinctive types of concept drift (none, incremental, gradual, sudden). SimTensor allows for streaming data generation with defined change points that served as drift injection moments. Each artificial tensor data stream is a two-class problem, with each tensor class generated from a distinct Gamma distributions. Class labels are assigned to each distribution generator, leading to a supervised learning problem and allowing for creation of 52 unique tensor data stream benchmarks.

5.2 Experimental set-up

5.2.1 Reference methods

As mentioned, up to our best knowledge this is the first work proposing usage of decision trees for classification of tensor data streams. As we propose a native tensor representation via chordal kernel (named Chordal Kernel Decision Tree, CKDT), as reference method we selected three state-of-the-art approaches for tensor vectorization that are able to work in an incremental fashion. We adapted them to this particular learning scenario. Online Robust Low-Rank Tensor Modeling for Streaming Data Analysis (LRTCR) (Li et al. 2019) uses the bilinear formulation of tensor nuclear norms and a stochastic optimization algorithm to learn the tensor low-rank structure alternatively for online updating. Online PCA with Optimal Regret (OPOR) (Nie et al. 2016) was proposed for low-dimensional data representation in online scenarios, thus can be used for tensors. Low-rank tensor decomposition (LRTD) (Guo et al. 2017) was developed for motion detection from videos using deep learning. In order to ensure a fair comparison, we train identical McDiarmid's decision tree on these tensors representations, as we use for our kernel-based feature space spanning. Additionally, as none of these methods were developed for concept drift, we enhance them with our tensor-based drift detector. Furthermore, we present results for a standard McDiarmid's decision tree (MDDT) that uses vector-based representation. This allows us to evaluate if operating in tensor space holds advantages over vector space for drifting data streams.

5.2.2 Parameters

For drift detection, we use a window $w = 100$ tensors. For subsampling procedure during kernel feature space spanning, we use 20% of tensors stored in the window.

5.2.3 Evaluation metrics

We evaluate examined tensor classifiers according to their prequential classification accuracy (accumulative metric used in data streams) and prequential multi-class

AUC (Wang and Minku 2020), model update time (in seconds) and memory usage (in RAM-hours).

5.3 Experiment 1: real-world tensor streams

In this experiment, we compare our proposed CKDT with three recent approaches for incremental tensor vectorization on four diverse real-world benchmarks that display streaming characteristics. We are interested in evaluating, if the proposed kernel feature space is more information-rich than vectorized spaces, which will translate into improved classification rates. Additionally, we wanted to evaluate the speed and memory consumption of analyzed approaches, in order to evaluate their usefulness for data stream scenarios.

Prequential accuracies and prequential multi-class AUC results are presented in Table 2, while Fig. 2 depicts streaming dependencies between prequential accuracy and number of processed tensors. Table 3 presents update time and memory consumption of analyzed models, while Table 4 presents the outcome of Shaffer multiple comparison statistical test of significance with $\alpha = 0.05$.

Obtained results show that vector-based adaptive decision tree (MDDT) cannot handle drifting data streams with tensor representation. On the other hand, experiments highlight the high efficacy of the proposed CKDT framework. Our approach is able to achieve significantly better classification accuracies than the same decision tree model trained using state-of-the-art incremental tensor vectorization. This shows how information-rich is the tensor representation in the context of data stream classification and that it is highly beneficial to maintain it. Our kernel-based feature space is capable of capturing these valuable

Table 2 Prequential accuracy (%) and prequential multi-class AUC (%) metrics for analyzed streaming-based tensor classification methods

Dataset	Metric	MDDT	LRTCR	OPOR	LRTD	CKDT
CC	pACC	7.28 ± 6.02	43.21 ± 9.23	47.33 ± 7.45	47.21 ± 6.29	**52.45** ± 5.88
	pmAUC	10.83 ± 5.81	48.34 ± 9.18	50.22 ± 7.29	51.67 ± 6.14	**58.63** ± 5.72
YM	pACC	13.83 ± 4.07	37.82 ± 5.29	39.30 ± 5.91	40.02 ± 4.82	**44.83** ± 4.19
	pmAUC	12.75 ± 5.22	42.65 ± 5.18	43.17 ± 5.83	41.12 ± 4.55	**46.99** ± 4.03
SVHN	pACC	33.52 ± 8.62	78.44 ± 10.06	82.19 ± 9.18	79.58 ± 8.28	**87.92** ± 7.81
	pmAUC	27.48 ± 7.59	72.04 ± 10.75	75.98 ± 10.04	73.19 ± 8.59	**83.02** ± 7.53
C100	pACC	4.91 ± 0.60	11.23 ± 1.45	10.04 ± 1.84	11.49 ± 2.10	**14.92** ± 0.94
	pmAUC	3.28 ± 0.52	10.43 ± 1.19	10.01 ± 1.37	10.72 ± 1.89	**13.98** ± 0.81
ST_{nd}	pACC	22.75 ± 9.82	82.19 ± 14.65	77.93 ± 12.34	82.83 ± 14.99	**83.86** ± 11.02
	pmAUC	19.86 ± 7.49	81.49 ± 13.98	76.54 ± 11.96	81.04 ± 13.84	**83.22** ± 10.68
ST_{gd}	pACC	18.43 ± 10.03	72.38 ± 16.81	70.03 ± 10.76	72.84 ± 17.45	**82.19** ± 11.99
	pmAUC	16.92 ± 8.42	70.18 ± 14.99	67.95 ± 11.38	70.03 ± 12.58	**81.96** ± 11.63
ST_{id}	pACC	13.73 ± 5.91	67.93 ± 17.98	62.78 ± 16.89	68.45 ± 18.92	**80.86** ± 12.87
	pmAUC	11.88 ± 4.83	65.84 ± 17.26	59.73 ± 16.03	65.27 ± 18.48	**80.11** ± 12.56
ST_{sd}	pACC	7.04 ± 2.66	57.44 ± 20.05	54.89 ± 18.97	58.93 ± 21.02	**79.28** ± 13.94
	pmAUC	5.82 ± 1.98	56.11 ± 18.56	52.07 ± 17.82	56.01 ± 19.77	**78.80** ± 12.90

Best values are presented in bold

Results for artificial tensor benchmarks averaged over 13 different tensor dimensionality sizes

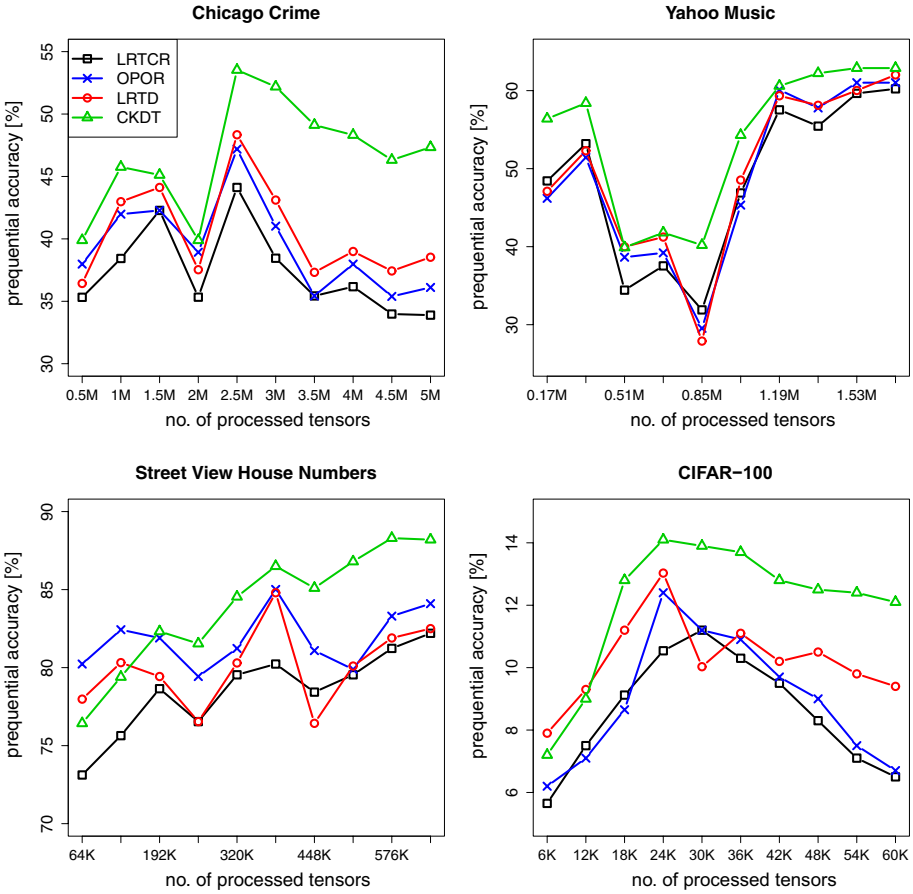

Fig. 2 Prequential accuracy over progressing real-world tensor data streams

properties and translating them into a more effective decision tree induction. In analyzed real-world datasets (especially in CC and YM) we can observe significant drops in performance of each analyzed classifier. These moments stand for a severe drift presence that renders the entire system outdated and needs to be handled by a drift detector that will replace the decision tree. While all methods suffer from the presence of drift, we should notice that CKDT achieves faster recovery rates after changes and its performance does not drop as significantly as in reference approaches. This can be contributed to efficient spanning of the similarity-based feature space that leads to better handling of new concepts and quicker adaptation after the occurrence of concept drift (**RQ1 answered**).

When analyzing the computational performance of CKDT, one should notice its shorter update time and lower memory consumption than these displayed by reference methods. This can be contributed to the computational tricks discussed in chordal kernel section, as well as to computing the new feature space only when concept drift took place. Additionally, we may observe that CKDT scales much better to bigger tensor representations (as seen with Chicago Crime and Yahoo Music datasets), outperforming significantly other algorithms (**RQ2 answered**).

Table 3 Average update time [s.] and memory consumption [RAM-hours] (calculated over 1000 tensors each) of analyzed decision tree approaches for tensor data stream classification

Dataset	Metric	MDDT	LRTCR	OPOR	LRTD	CKDT
CC	Time	72.39 ± 11.04	236.43 ± 21.43	345.12 ± 68.28	183.21 ± 44.19	**84.83** ± 11.28
	Memory	0.72 ± 0.13	2.38 ± 0.48	5.21 ± 0.66	1.87 ± 0.71	**1.21** ± 0.29
YM	Time	18.42 ± 5.93	73.34 ± 19.35	139.31 ± 41.18	70.95 ± 20.09	**56.72** ± 11.28
	Memory	0.38 ± 0.07	1.87 ± 0.44	2.89 ± 0.90	1.72 ± 0.52	**0.98** ± 0.40
SVHN	Time	48.47 ± 13.59	99.23 ± 28.03	109.31 ± 33.92	**94.02**v14.29	95.01 ± 15.22
	Memory	0.11 ± 0.03	0.45 ± 0.10	1.02 ± 0.31	**0.36** ± 0.08	0.37 ± 0.05
C100	Time	38.46 ± 11.99	97.13 ± 18.29	111.47 ± 22.89	90.76 ± 16.32	**68.09** ± 17.04
	Memory	0.10 ± 0.03	0.44 ± 0.07	1.03 ± 0.18	0.33 ± 0.08	**0.31** ± 0.07
ST_{nd}	Time	62.49 ± 17.88	174.34 ± 31.56	204.98 ± 44.92	136.86 ± 21.47	**74.82** ± 11.59
	Memory	0.51 ± 0.11	1.93 ± 0.19	4.41 ± 1.03	1.38 ± 0.16	**1.07** ± 0.11
ST_{gd}	Time	78.81 ± 15.93	208.41 ± 22.98	229.72 ± 33.71	161.06 ± 31.02	**104.82** ± 19.93
	Memory	0.55 ± 0.22	2.19 ± 0.38	4.82 ± 1.02	1.58 ± 0.33	**1.39** ± 0.34
ST_{id}	Time	88.91 ± 20.74	221.09 ± 37.85	258.03 ± 39.99	142.16 ± 22.19	**88.98** ± 13.69
	Memory	0.79 ± 0.44	2.28 ± 0.39	4.96 ± 0.83	1.70 ± 0.32	**1.50** ± 0.22
ST_{sd}	Time	91.38 ± 27.54	241.64 ± 38.44	201.77 ± 39.91	122.58 ± 19.27	**79.94** ± 12.86
	Memory	0.80 ± 0.31	2.62 ± 0.23	5.19 ± 1.38	2.01 ± 0.14	**1.69** ± 0.12

Best values are presented in bold

Table 4 Outcome of Shaffer post-hoc statistical test for comparison among CKDT and reference methods over multiple datasets (4 real-world and 52 benchmarks)

CKDT vs.	MDDT	LRTCR	OPOR	LRTD
p-value	0.000000	0.000183	0.000149	0.000199

5.4 Experiment 2: evaluating the impact of tensor dimensionality

In this experiment, we wanted to evaluate the scalability of our framework to high-dimensional input tensors (i.e., tensors containing a high number of factors). Most existing tensor datasets have between 3 to 5 factors (as they are either images, link relationships, or reviews), but we can predict that soon more complex tensor domains will become increasingly popular. Therefore, we used SimTensor generator to create a number of tensor data streams with varying number of associated factors. This also allowed us to inject concept drift in a controlled manner. Prequential accuracies are presented in Fig. 3.

Obtained results confirm our assumption that increasing number of factors will pose progressively increasing difficulty for all classifiers. We can see that CKDT shows much better scalability to high number of factors than other methods, as chordal distance allows for maintaining the tensor properties regardless of the input. The quality of feature spaces obtained by vectorization methods suffer in a much more significant manner, making their usage prohibitive in such cases (**RQ3 answered**).

It is interesting to analyze the interplay between the type of concept drift and the increasing number of factors. One can see that more complex tensors make proper drift

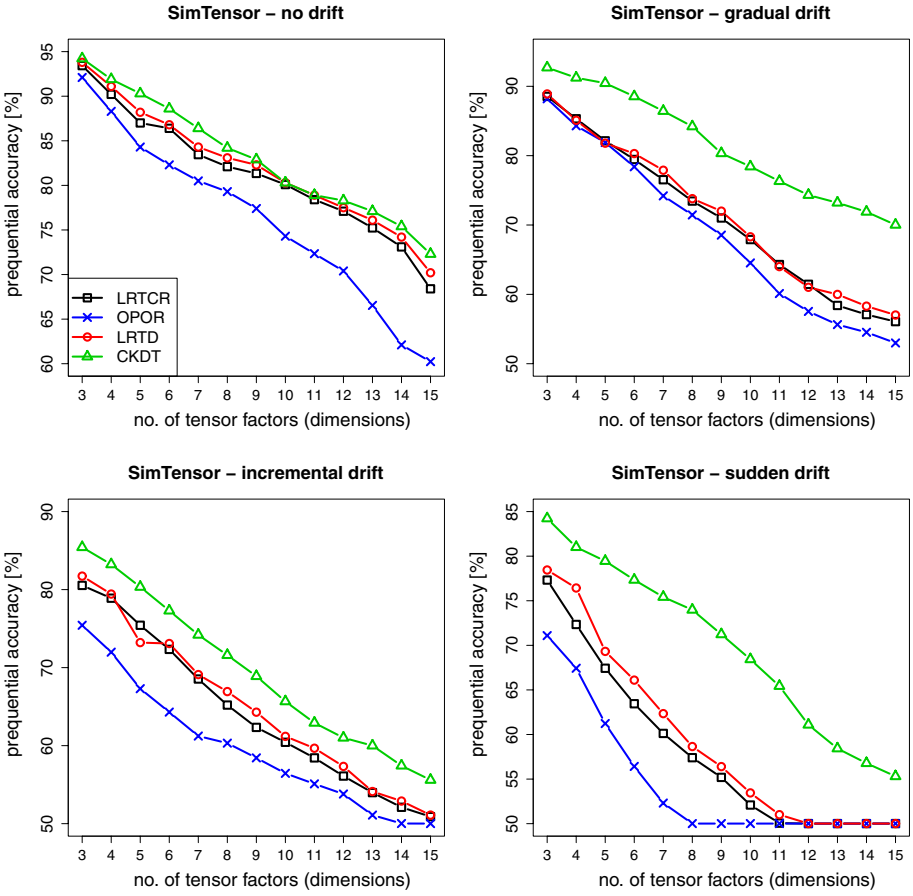

Fig. 3 Averaged prequential accuracy calculated over the artificial data streams with respect to increasing tensor dimensionality and different types of concept drift

detection much more difficult, leading to overall drops in accuracy. Most challenging type of drift is sudden one, which come to no surprise, as system has no time to react to it. Second most difficult drift is much more surprising, as incremental changes are usually easy to handle (Ditzler et al. 2015). In this case, we may attribute this learning difficulty to the way we designed our drift detector. If the change is small enough, the detection signal (3σ rule) will never be triggered, thus never reconstructing the feature space. We will continue our work in this direction, to propose more advanced tensor-based drift detector that is robust to such situations.

Overall, the proposed CKDT offers superior performance to all reference methods, even when they are enhanced with the proposed tensor-based drift detector. This can be contributed to the combination of the drift detection with kernel feature space that is more sensitive to changes in data distributions (**RQ4 answered**).

6 Conclusions and future works

6.1 Summary

In this paper, we have presented a first framework for tensor data stream classification with decision trees under concept drift. We have identified the drawback of existing data stream classification approaches, namely their limitation to vector representation of input data. We argued that as many real-world data sources generate multi-dimensional data that cannot be vectorized without a loss of information, there is a need for tensor-based classifiers for data streams. As a base classifier we selected McDiarmid's incremental decision tree. In order to alleviate its limitations, we proposed to create a new feature space that operates on tensors and use it for decision tree induction. To this aim we employed kernel feature mapping, where a dedicated similarity measure using chordal distance was used. It allowed for calculating direct similarity between two tensors, without a need for vectorization. We showed how to speed-up the creation of the new feature space using random subsampling. We also proposed a concept drift detector based on tensor data representation that was used to control when to create a new feature space and when to update the classifier. Experimental study carried out on large-scale real-world and artificial tensor data streams showed that our framework preserves the information within tensors, leading to an excellent classification accuracy. Additionally, it scales-up to high-dimensional tensors and is much less computationally expensive than online vectorization.

6.2 Future works

In our future works, we plan to continue developing a holistic framework for tensor data stream classification that will encompass the following research directions:

- **Ensembles of CKDTs.** A natural step forward will be to propose adaptive and online ensembles of Chordal Kernel Decision Trees to boost their predictive accuracy and make them competitive to modern deep learning algorithms (González et al. 2020).
- **Explainable learning from tensor streams.** Decision tree structure offers a natural explainable and interpretable format (Sagi and Rokach 2020). This can be leveraged towards understanding the nature of changes in drifting tensor streams.
- **Speeding-up CKDTs.** Current implementation of CKDT is efficient and faster than state-of-the-art methods, but can be further improved by using approximate decomposition approaches (Cyganek and Wozniak 2016).
- **Evolving tensor dimensionality.** A fully robust framework for tensor data stream mining must offer the capability of adapting to evolving dimensionality and factors of tensors (da Silva Fernandes et al. 2019).

Declarations

Conflicts of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Aljundi, R., Kelchtermans, K., & Tuytelaars, T. (2019). Task-free continual learning. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, Computer Vision Foundation, IEEE, (pp. 11254–11263).

Ashfahani, A., & Pratama, M. (2019). Autonomous deep learning: Continual learning approach for dynamic environments. In: Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019, Calgary, Alberta, Canada, May 2–4, 2019, SIAM, (pp. 666–674).

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. *Journal of Machine Learning Research, 11,* 1601–1604.

Chhaya, R., Choudhari, J., Dasgupta, A., & Shit, S. (2020). Streaming coresets for symmetric tensor factorization. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event, PMLR, Proceedings of Machine Learning Research, vol 119, (pp. 1855–1865).

Cyganek, B., & Wozniak, M. (2016). Efficient computation of the tensor chordal kernels. In: International Conference on Computational Science 2016, ICCS 2016, 6–8 June 2016, San Diego, California, USA, Elsevier, Procedia Computer Science, vol 80, (pp. 1702–1711).

Cyganek, B., Krawczyk, B., & Wozniak, M. (2015). Multidimensional data classification with chordal distance based kernel and support vector machines. *Engineering Application of Artificial Intelligence, 46,* 10–22.

Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. *Computational Intelligence Magazine, 10*(4), 12–25.

Fanaee-T, H., & Gama, J. (2016). Simtensor: A synthetic tensor data generator. CoRR abs/1612.03772.

Fu, X., Huang, K., Ma, W., Sidiropoulos, N. D., & Bro, R. (2015). Joint tensor factorization and outlying slab suppression with applications. *IEEE Transaction on Signal Processing, 63*(23), 6315–6328.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Survey, 46*(4), 44:1-44:37.

Gomes, H. M., Read, J., Bifet, A., Barddal, J. P., & Gama, J. (2019). Machine learning for streaming data: State of the art, challenges, and opportunities. *SIGKDD Explore, 21*(2), 6–22.

González, S., García, S., Ser, J. D., Rokach, L., & Herrera, F. (2020). A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities. *Information Fusion, 64,* 205–237.

Gu, L., Zhou, N., & Zhao, Y. (2018). An euclidean distance based on tensor product graph diffusion related attribute value embedding for nominal data clustering. In: AAAI, AAAI Press.

Gujral, E., Theocharous, G., & Papalexakis, E.E. (2020). SPADE: streaming PARAFAC2 decomposition for large datasets. In: Demeniconi C, Chawla NV (eds) Proceedings of the 2020 SIAM International Conference on Data Mining, SDM 2020, Cincinnati, Ohio, USA, May 7-9, 2020, SIAM, (pp. 577–585).

Guo, H., Wu, X., & Feng, W. (2017). Multi-stream deep networks for human action classification with sequential tensor decomposition. *Signal Processing, 140*, 198–206.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Wozniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion, 37*, 132–156.

Lathauwer, L.D. (2009). A survey of tensor methods. In: ISCAS, IEEE, (pp. 2773–2776).

Letourneau, P., Baskaran, M.M., Henretty, T., Ezick, J.R., & Lethin, R. (2018). Computationally efficient CP tensor decomposition update framework for emerging component discovery in streaming data. In: 2018 IEEE High Performance Extreme Computing Conference, HPEC 2018, Waltham, MA, USA, September 25–27, 2018, IEEE, (pp. 1–8).

Li, P., Feng, J., Jin, X., Zhang, L., Xu, X., & Yan, S. (2019). Online robust low-rank tensor modeling for streaming data analysis. *IEEE Transactions on Neural Networks and Learning Systems, 30*(4), 1061–1075.

Li, Q., & Schonfeld, D. (2014). Multilinear discriminant analysis for higher-order tensor data classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36*(12), 2524–2537.

Mardani, M., Mateos, G., & Giannakis, G. B. (2015). Subspace learning and imputation for streaming big data matrices and tensors. *IEEE Trans Signal Processing, 63*(10), 2663–2677.

Maruhashi, K., Todoriki, M., Ohwa, T., Goto, K., Hasegawa, Y., Inakoshi, H., & Anai, H. (2018). Learning multi-way relations via tensor decomposition with neural networks. In: AAAI, AAAI Press.

Nakatsuji, M., Zhang, Q., Lu, X., Makni, B., & Hendler, J. A. (2017). Semantic social network analysis by cross-domain tensor factorization. *IEEE Transactions on Computational Social Systems, 4*(4), 207–217.

Nie, J., Kotlowski, W., & Warmuth, M. K. (2016). Online PCA with optimal regret. *Journal of Machine Learning Research, 17*, 173:1-173:49.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks, 113*, 54–71.

Pinage, F. A., dos Santos, E. M., & Gama, J. (2020). A drift detection method based on dynamic classifier selection. *Data Mining and Knowledge Discovery, 34*(1), 50–74.

Rambhatla, S., Li, X., & Haupt, J.D. (2020). Provable online CP/PARAFAC decomposition of a structured tensor via dictionary learning. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual.

Rutkowski, L., Pietruczuk, L., Duda, P., & Jaworski, M. (2013). Decision trees for mining data streams based on the mcdiarmids bound. *IEEE Transactions on Knowledge and Data Engineering, 25*(6), 1272–1279.

Sagi, O., & Rokach, L. (2020). Explainable decision forest: Transforming a decision forest into an interpretable tree. *Information Fusion, 61,* 124–138.

Sahoo, D., Pham, Q., Lu, J., & Hoi, S.C.H. (2018). Online deep learning: Learning deep neural networks on the fly. In: Lang J (ed) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden, ijcai.org. (pp 2660–2666).

Shin, K., Hooi, B., Kim, J., & Faloutsos, C. (2017). Densealert: Incremental dense-subtensor detection in tensor streams. In: KDD, ACM, (pp. 1057–1066).

Sidiropoulos, N. D., Lathauwer, L. D., Fu, X., Huang, K., Papalexakis, E. E., & Faloutsos, C. (2017). Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing, 65*(13), 3551–3582.

Signoretto, M., Lathauwer, L. D., & Suykens, J. A. K. (2011). A kernel-based framework to tensorial data analysis. *Neural Networks, 24*(8), 861–874.

da Silva Fernandes, S., Fanaee-T, H., & Gama, J. (2019). Evolving social networks analysis via tensor decompositions: From global event detection towards local pattern discovery and specification. In: Discovery Science - 22nd International Conference, DS 2019, Split, Croatia, October 28-30, 2019, Proceedings, Springer, Lecture Notes in Computer Science, vol 11828, (pp. 385–395).

Smith, S., Huang, K., Sidiropoulos, N.D., & Karypis, G. (2018). Streaming tensor factorization for infinite data sources. In: SDM, SIAM, (pp. 81–89).

Song, Q., Huang, X., Ge, H., Caverlee, J., & Hu, X. (2017). Multi-aspect streaming tensor completion. In: KDD, ACM, (pp. 435–443).

Sun, J. (2008). Incremental pattern discovery on streams, graphs and tensors. *SIGKDD Explorations, 10*(2), 28–29.

Sun, J., Tao, D., Papadimitriou, S., Yu, P. S., & Faloutsos, C. (2008). Incremental tensor analysis: Theory and applications. *TKDD, 2*(3), 11:1-11:37.

Sun, Y., Guo, Y., Luo, C., Tropp, J. A., & Udell, M. (2020). Low-rank tucker approximation of a tensor from streaming data. *SIAM Journal on Mathematics of Data Science, 2*(4), 1123–1150.

Wang, S., & Minku, L.L. (2020). AUC estimation and concept drift detection for imbalanced data streams with multiple classes. In: 2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19–24, 2020, IEEE, (pp. 1–8).

Yang, K., Gao, Y., Shen, Y., Zheng, B., & Chen, L. (2021). Dismastd: An efficient distributed multi-aspect streaming tensor decomposition. In: 37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19–22, 2021 (pp. 1080–1091) IEEE.

Yang, S., Wang, M., Feng, Z., Liu, Z., & Li, R. (2018). Deep sparse tensor filtering network for synthetic aperture radar images classification. *IEEE Transactions on Neural Networks and Learning Systems, 29*(8), 3919–3924.

Zhou, S., Nguyen, X.V., Bailey, J., Jia, Y., & Davidson, I. (2016). Accelerating online CP decompositions for higher order tensors. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, ACM, (pp. 1375–1384).

Zyblewski, P., Sabourin, R., & Wozniak, M. (2021). Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. *Information Fusion, 66,* 138–154.