Adaptive Wavelet Density Estimators over Data Streams

Christoph Heinz Bernhard Seeger
Department of Mathematics and Computer Science
University of Marburg
{heinzch, seeger}@mathematik.uni-marburg.de

Abstract

A variety of scientific and commercial applications requires an immediate analysis of transient data streams. Many approaches for analyzing data share the property that an estimation of the underlying data distribution is used as a fundamental building block. To estimate the density of a continuous data distribution, wavelet density estimation, a technique from the area of nonparametric statistics, is very appealing as it is theoretically well-founded and practically approved. For that reason, its application to data streams is highly promising; it provides a convenient way to analyze the characteristics of a stream. However, the heavy computational cost of wavelet density estimators renders their direct application to the streaming scenario impossible. In this work, we tackle this problem and present a novel approach to adaptive wavelet density estimators over data streams. Not only do our estimators meet the rigid processing requirements for data streams, they also adapt to changing system resources in a well-defined manner. A thorough experimental evaluation demonstrates the efficacy of our wavelet density estimators and shows their superiority to competing kernel- and histogram-based estimators.

1 Introduction

The exploration and analysis of data streams is a crucial prerequisite in many application scenarios, e.g., traffic management, monitoring of vital features. In order to keep pace with a data stream, the corresponding analysis techniques have to meet restrictive processing requirements [7], e.g., each element can only be processed once and the amount of available memory is bounded.

In this work, we concentrate on a building block of many data analysis approaches, namely the estimation of continuous densities. Density estimation captures an unknown continuous distribution by estimating its probability density function. The resulting density estimator can be exploited to gain insight into essence and structure of the data; it can be utilized for all kinds of analysis tasks, e.g., computation of summary measures, outlier detection [18]. For that reason, the application of density estimation to real-valued data streams is highly promising. To estimate the density of an unknown continuous distribution, mathematical statistics provides a plethora of estimation techniques among which the nonparametric ones are of particular interest as they solely base on the sample. To cite [11]: *"a well-known benefit of nonparametric methods is their ability to achieve estimation optimality for ANY input distribution as more data are observed"*. Among the most popular nonparametric estimation techniques are kernel density estimators and histograms [17]. In comparison to the latter two techniques, density estimators based on wavelets [8] are superior in terms of a higher convergence rate and a better resolution of irregular densities. *"For data analytic purposes with small to moderate data size a kernel estimate may be preferred for its simplicity and wide distribution. For finer local analysis and good asymptotic properties the wavelet estimator is certainly the method to be chosen."*[13]. However, the computational complexity of wavelet density estimators makes their direct application to data streams impossible.

In this work, we address this problem and develop wavelet density estimators that can be computed in an online fashion over a data stream while allocating only a constant amount of memory. To the best of our knowledge, this is the first adaptation of wavelet density estimators to data streams. Basically, their computation relies on a framework for maintaining nonparametric estimators over data streams [2]. The main idea is to process the stream in a block-wise manner, where each data block is associated with a separate wavelet density estimator. While processing the stream, we successively merge those block estimators, which gives us an overall estimator for the already processed stream. In [14], we already presented an early stage of this work, which only gave a rough sketch.

In order to scrutinize our wavelet density estimators, we conducted a broad experimental study, including the comparison with competitive density estimation techniques.

This study substantiated the good approximation properties of our estimators as well as their superiority to dynamic histograms [12] and M-Kernels [3] in terms of estimation accuracy and processing time.

The paper is organized as follows. Section 2 introduces the problem of density estimation over data streams. We give a brief review of wavelets and wavelet density estimation in Section 3. Section 4 sketches the aforementioned framework and Section 5 discusses its application with wavelet density estimators. Section 6 gives an overview of related work. The results of our experimental evaluation are presented in Section 7. Finally, we conclude with a summary and an outlook on future work in Section 8.

2 The Problem

Let a data stream be an unbounded sequence of real-valued elements $X_1, X_2, ...$ with $X_i \in \mathbb{R}$. Except where otherwise stated, we consider one-dimensional streams. We assume that the elements constitute at each time instant an iid-sample (independent and identically distributed observations) of a random variable X with an unknown density f. The assumption of independence between two stream elements is justifiable as sources that generate data streams typically send their elements autonomously. It is worth mentioning that these are the only assumptions on the stream.

With respect to those relatively weak assumptions, the core question of this work is how to compute wavelet density estimators over data streams in compliance with the rigid processing requirements for streams [7, 2]? Before we go into the details of how we addressed this question, let us first give a brief overview of wavelets and wavelet density estimation.

3 An Introduction to Wavelets

For the sake of simplicity, we concentrate in the following on $L_2(\mathbb{R})$, the space of all square-integrable functions on $\mathbb{R}$. Its inner product is defined as $\langle f, g \rangle = \int fg \, dx$. Assume a **wavelet** function ψ defined on $\mathbb{R}$ is given. For $j, k \in \mathbb{Z}$ let $\psi_{j,k}(x) := 2^{j/2}\psi(2^j x - k)$ with **dilation** (or **resolution**) index j and **translation** index k. The dilation squeezes or expands the wavelet, while the translation shifts it along the x-axis. A remarkable property of wavelets is that the entirety of those versions $\psi_{j,k}, j, k \in \mathbb{Z}$ of ψ constitutes an orthonormal basis of $L_2(\mathbb{R})$. Among the most popular wavelets are Haar wavelets as they are simple and Daubechies wavelets [13] as they combine a compact support with different degrees of smoothness.

A fundamental principle of wavelets is the **multiresolution analysis** (MRA), which provides views at different resolutions on a function. The higher the resolution, the more details of a function are resolved. Formally, the different resolutions correspond to an increasing sequence of subspaces $(V_j)_{j \in \mathbb{N}}$ with $... \subset V_{j-1} \subset V_j \subset V_{j+1} \subset ...$, whose union is $\overline{\bigcup_{j=-\infty}^{\infty} V_j} = L_2(\mathbb{R})$. Each V_j can be represented as V_{j-1} plus its orthogonal complement W_{j-1}, i. e. $V_j = V_{j-1} \oplus W_{j-1}$. While V_j is generated by translations $\{\phi_{j,k} : k \in \mathbb{Z}\}$ of a **scaling function** ϕ, the **detail space** W_j is generated by translations $\{\psi_{j,k} : k \in \mathbb{Z}\}$ of the wavelet ψ_j.

By means of these subspaces, we can decompose $L_2(\mathbb{R})$ as follows

$$L_2(\mathbb{R}) = \bigoplus_{j \in \mathbb{Z}} W_j = V_{j_0} \oplus \bigoplus_{j \geq j_0} W_j. \quad (1)$$

Hence, each function $f \in L_2(\mathbb{R})$ can be represented in its **wavelet series expansion**

$$f(x) = \sum_{j,k \in \mathbb{Z}} d_{j,k}\psi_{j,k}(x) \quad (2)$$

$$= \sum_{k \in \mathbb{Z}} c_{j_0,k}\phi_{j_0,k}(x) + \sum_{j=j_0}^{\infty} \sum_{k \in \mathbb{Z}} d_{j,k}\psi_{j,k}(x) \quad (3)$$

with **scaling coefficients** $c_{j_0,k} = \langle f, \phi_{j_0,k} \rangle$ and **wavelet coefficients** $d_{j,k} = \langle f, \psi_{j,k} \rangle$. Generally, the subspaces V_j represent coarse, global features of f and their complements W_j the local details. As we will see, the wavelet series expansion is of utmost importance for the computation of wavelet density estimators.

Another important concept of wavelets is the **discrete wavelet transform** (DWT), which offers a hierarchical decomposition of a function. Given the scaling coefficients of a function at resolution j_1, the DWT computes scaling and wavelet coefficients of lower resolutions. This offers a simple and efficient method to compress a function in case its number of coefficients exceeds a preset maximum number. We apply the DWT and decompose the scaling coefficients of the function into scaling and wavelet coefficients of lower resolutions. To reduce the coefficient number, we simply discard the least relevant coefficients. Since the wavelet coefficients describe the local details, a convenient approach is to discard the wavelet coefficients with lowest absolute value. This strategy is also theoretically well-founded as it ensures a minimum compression error [5].

3.1 Wavelet Density Estimation

The starting point for a **wavelet density estimator** (WDE) is the previously discussed wavelet series expansion - see equation (2) - of the unknown density f. The basic idea is to estimate the unknown scaling and wavelet coefficients of f by utilizing the sample points $X_1, ..., X_n$. As f is a probability density function, it holds

$$c_{j,k} = \langle f, \phi_{j,k} \rangle = \int f(x)\phi_{j,k}(x)dx = E(\phi_{j,k}(X)), \quad (4)$$

$$d_{j,k} = \langle f, \psi_{j,k} \rangle = \int f(x)\psi_{j,k}(x)dx = E(\psi_{j,k}(X)). \quad (5)$$

The estimation of these expectation values with the help of the sample values delivers an estimate of the coefficients:

$$\hat{c}_{j,k} = \frac{1}{n}\sum_{i=1}^{n} \phi_{j,k}(X_i), \ \hat{d}_{j,k} = \frac{1}{n}\sum_{i=1}^{n} \psi_{j,k}(X_i). \quad (6)$$

The combination of these **empirical coefficients** with the wavelet series expansion of f delivers an initial WDE

$$\hat{f}(x) = \sum_{k\in\mathbb{Z}} \hat{c}_{j_0,k}\phi_{j_0,k}(x) + \sum_{j=j_0}^{\infty}\sum_{k\in\mathbb{Z}} \hat{d}_{j,k}\psi_{j,k}(x). \quad (7)$$

Since the number of coefficients in equation (7) may be infinite, we must truncate the series expansion of f [13]. The crucial step in this context is the choice of the empirical coefficients to keep. If not enough coefficients are kept, the resulting WDE can be oversmoothed and may hide local features. If too many coefficients are kept, the resulting WDE can be undersmoothed and may exhibit spurious artifacts. The different types of WDEs distinguish themselves by their strategy for choosing the empirical coefficients. In the following, we focus on two well-established WDE types [8, 13]: linear WDEs and thresholded WDEs.

A **linear WDE** estimates the projection of f in $V_{j_1^{lin}}$ with j_1^{lin} a suitably chosen resolution. Hence, given the empirical coefficients of this resolution, a linear WDE is computed via

$$\hat{f}(x) = \sum_{k\in\mathbb{Z}} \hat{c}_{j_1^{lin},k}\phi_{j_1^{lin},k}(x). \quad (8)$$

The quality of a linear WDE fundamentally depends on an appropriate setting of j_1^{lin}. As the optimal j_1^{lin} suffers from the drawback that it depends on the unknown regularity properties of f, [58] proposes to use $j_1^{lin} = (\log_2 n)/3 - 2 - \log_2 \sigma$ as an approximate solution. More advanced strategies for the setting of j_1^{lin} exist [13], but their computational complexity renders an application in the data stream scenario difficult. In general, linear WDEs are a convenient technique for the estimation of smooth densities. However, for the case of densities with heterogeneous smoothness properties, thresholded wavelet density estimators are preferable.

The basic idea of **thresholded WDEs** is to perform an adaptive fit to the local smoothness of the density. In their seminal work [8], Donoho et al. introduced a nonlinear WDE which applies a thresholding procedure to its empirical wavelet coefficients:

$$\hat{f}(x) = \sum_{k\in\mathbb{Z}} \hat{c}_{j_0^{thr},k}\phi_{j_0^{thr},k}(x) + \sum_{j=j_0^{thr}}^{j_1^{thr}}\sum_{k\in\mathbb{Z}} \tilde{d}_{j,k}\psi_{j,k}(x) \quad (9)$$

where $\tilde{d}_{j,k}$ denotes a thresholded wavelet coefficient and j_0^{thr}, j_1^{thr} are suitably chosen resolutions. By a careful selection of empirical wavelet coefficients, local features of f like discontinuities or sharp cusps shall be detected without introducing spurious local artifacts. An appropriate setting of the threshold is crucial since it dictates which local details are kept. In [8], the authors proposed two thresholding strategies: **soft thresholding** and **hard thresholding**. Soft thresholding

$$\tilde{d}_{j,k} = \begin{cases} \hat{d}_{j,k} - \lambda_j, & \hat{d}_{j,k} > \lambda_j \\ 0, & |\hat{d}_{j,k}| \le \lambda_j \\ \hat{d}_{j,k} + \lambda_j, & \hat{d}_{j,k} < -\lambda_j \end{cases} \quad (10)$$

shrinks large coefficients. Hard thresholding sets coefficients with absolute value smaller than λ_j to zero:

$$\tilde{d}_{j,k} = \begin{cases} \hat{d}_{j,k}, & |\hat{d}_{j,k}| > \lambda_j \\ 0, & otherwise. \end{cases} \quad (11)$$

While soft thresholding guarantees better mean squared error properties, hard thresholding keeps the visual appearance of jumps and peaks [8]. Overall, a thresholded WDE has the following parameters: the resolutions j_0^{thr}, j_1^{thr} and the thresholds λ_j. According to [20], $j_0^{thr} = \lceil \frac{\log_2 n}{2r-1} \rceil$ (with r the regularity of the wavelet) and $j_1^{thr} = \lfloor \log_2 n - \log_2(\log n) \rfloor$ are reasonable resolution settings. Concerning the level-dependent threshold λ_j, [13] proposes to use a multiple of $\max_{k\in\mathbb{Z}} |\hat{d}_{j,k}|$.

For practical purposes, wavelets with bounded support are advisable as they ensure a finite number of coefficients. Given the corresponding resolution j_1^{lin} (or j_1^{thr}), we can compute the minimum and maximum indexes k_{min}, k_{max} of the non-zero empirical scaling coefficients at this resolution, i.e., the number of empirical scaling coefficients to compute is $k_{max} - k_{min} + 1$. With respect to the resolution settings of linear and thresholded WDEs, k_{min} and k_{max} are linear in the sample size.

Generally, a closer look at the computation of WDEs reveals that we can not directly compute them over data streams. The reason is that their parameters depend on the sample size, which continuously increases in our case. As a result, the resolutions j_1^{lin}, j_1^{thr} will change while processing the stream. A resolution change in turn necessitates the recomputation of all empirical coefficients. A recomputation, however, requires access to all already processed stream elements, a fact that violates the one-pass paradigm postulated in data stream processing.

4 Maintaining Nonparametric Estimators over Data Streams

In [2], we presented a framework for maintaining nonparametric estimators over data streams. In this work, we exploit this framework to compute WDEs over data streams. The basic idea of the framework is to partition the data stream into disjoint data blocks consisting of b elements. Each of those blocks is associated with a separate block estimator $\hat{f}_i$. After we have built a new block estimator, we determine the **overall estimator** $\hat{g}$ for the already processed stream as convex linear combination of the block estimators, i.e., $\hat{g}(x) = \sum_{i=1}^{m} \omega_i \hat{f}_i(x)$ with $0 \leq \omega_i \leq 1, i = 1, ..., m$ and $\sum_{i=1}^{m} \omega_i = 1$. For each weighting sequence $(\omega_i)_{i\geq 1}$, we can determine an equivalent sequence $(\tilde{\omega}_i)_{i\geq 1}$ that allows us to compute the overall estimator online without accessing all previous block estimators:

$$\hat{g}_i(x) = \begin{cases} \hat{f}_1(x), & i = 1 \\ (1-\tilde{\omega}_i)\hat{g}_{i-1}(x) + \tilde{\omega}_i \hat{f}_i(x), & i \geq 2 \end{cases} \tag{12}$$

with $0 \leq \tilde{\omega}_i \leq 1,\ i \geq 1$ and $\tilde{\omega}_1 = 1$. In order to comply with the requirement of a constant amount of allocatable memory, the overall estimator is compressed after each update. For more details about this framework and the interplay of its components, we refer to [2].

The relevant question for this work is how to use this framework. Essentially, we have to define the following steps with respect to the general idea sketched above:

- **Computation of a block estimator:** A separate estimator must be built for each data block.
- **Convex merge step:** With respect to equation (12), the 'sum' of two estimators, which constitutes the new estimator, must be defined.
- **Compression step:** The overall estimator must be compressed so that it fits into the available memory.

5 Adaptive Wavelet Density Estimators over Data Streams

In the following, we investigate the specification of the above processing steps for WDEs. Due to their implementation within the framework, the resulting online WDEs inherently meet the processing requirements for streams. In particular, they allocate only a constant amount of memory, which they even can adapt to changing system resources.

5.1 Computation of a Block WDE

The first step is to choose the underlying wavelet family. Due to their convenient properties, Daubechies wavelets are a suitable choice. The second step is to decide for linear or thresholded WDEs and to compute the corresponding resolution j_1^{lin} or j_1^{thr} for the given data block. The next step is to compute the empirical scaling coefficients for this resolution as described in equation (6). By means of the DWT, they are decomposed into scaling and wavelet coefficients of lower resolutions. For thresholded WDEs, these resolutions are $j_0^{thr}, ..., j_1^{thr} - 1$. We additionally apply in case of thresholded WDEs a level-dependent thresholding procedure to the empirical wavelet coefficients. Eventually, we have a set of non-zero empirical scaling and wavelet coefficients. Their entirety constitutes the block WDE.

5.2 Convex Merge Step

A vital feature of the framework is the convex merge of the current overall estimator and the new block estimator. To define this merge step for the case of WDEs, we exploit their wavelet series expansions.

Without loss of generality, we consider the convex merge of the first two block WDEs, i.e. $\hat{g}_2(x) = (1-\tilde{\omega}_2)\hat{f}_1(x) + \tilde{\omega}_2 \hat{f}_2(x)$. As we will see, a necessary prerequisite for the merge is that both WDEs have the same scaling resolution. For that reason, we use the DWT to transform the scaling coefficients of both WDEs down to the minimum j_0 of their current scaling resolutions. Let $\{\hat{c}_{j_0,k}^{(1)} : k \in \mathbb{Z}\}$ and $\{\hat{c}_{j_0,k}^{(2)} : k \in \mathbb{Z}\}$ be the empirical scaling coefficients of the wavelet series expansions of $\hat{f}_1$ and $\hat{f}_2$ respectively. In the same manner, we denote the empirical wavelet coefficients of $\hat{f}_1$ and $\hat{f}_2$ as $\{\hat{d}_{j,k}^{(1)} : j, k \in \mathbb{Z}, j \geq j_0\}$ and $\{\hat{d}_{j,k}^{(2)} : j, k \in \mathbb{Z}, j \geq j_0\}$. Let those sets also comprise the coefficients with value zero as it facilitates the introduction of the merge step. By means of the wavelet series expansion of $\hat{f}_1$ and $\hat{f}_2$, we can decompose $\hat{g}_2$ as follows

$$\begin{aligned} \hat{g}_2(x) &= (1-\tilde{\omega}_2)\hat{f}_1(x) + \tilde{\omega}_2 \hat{f}_2(x) \\ &= (1-\tilde{\omega}_2)\sum_{k\in\mathbb{Z}} \hat{c}_{j_0,k}^{(1)} \phi_{j_0,k}(x) \\ &\quad +(1-\tilde{\omega}_2)\sum_{j\geq j_0}\sum_{k\in\mathbb{Z}} \hat{d}_{j,k}^{(1)} \psi_{j,k}(x) \\ &\quad +\tilde{\omega}_2 \sum_{k\in\mathbb{Z}} \hat{c}_{j_0,k}^{(2)} \phi_{j_0,k}(x) + \tilde{\omega}_2 \sum_{j\geq j_0}\sum_{k\in\mathbb{Z}} \hat{d}_{j,k}^{(2)} \psi_{j,k}(x) \\ &= \sum_{k\in\mathbb{Z}} \hat{c}_{j_0,k}^{(3)} \phi_{j_0,k}(x) + \sum_{j\geq j_0}\sum_{k\in\mathbb{Z}} \hat{d}_{j,k}^{(3)} \psi_{j,k}(x). \end{aligned}$$

with $\hat{c}_{j_0,k}^{(3)} = (1-\tilde{\omega}_2)\hat{c}_{j_0,k}^{(1)} + \tilde{\omega}_2 \hat{c}_{j_0,k}^{(2)}$ and $\hat{d}_{j,k}^{(3)} = (1-\tilde{\omega}_2)\hat{d}_{j,k}^{(1)} + \tilde{\omega}_2 \hat{d}_{j,k}^{(2)}$. Hence, the convex merge step of two block WDEs is simply the convex merge of their empirical coefficients. This is a remarkable result as we can merge two functions without even evaluating them. An important aspect in this context is that the result of the merge, the new overall WDE, is also given in its wavelet series expansion. Thereby, we can merge the current overall WDE and a new block WDE in the same manner, i.e., this procedure is a valid realization of the merge step for WDEs.

Having this merge procedure in mind, the prerequisite of the same scaling resolution for both WDEs becomes clear. In case they would differ, the resulting estimator would include scaling coefficients of different resolutions, which no more complies with the definition of the wavelet series expansion in equation (2).

A valuable side effect of the merge procedure is that we expect the resulting number of empirical coefficients after the merge to be smaller than the number of the contributing empirical coefficients. As all block WDEs follow the same distribution, the occurrence of similar features is likely. Since these features are described by the same empirical coefficients, we expect a high number of 'merge partners'. The merge of these coefficients in turn reduces the number of coefficients.

Figure 1 depicts an example for the convex merge of two functions given in their wavelet representations. Both are represented with respect to Daubechies2 wavelets. The weight $\tilde{\omega}$ for the merge is set to 0.75.

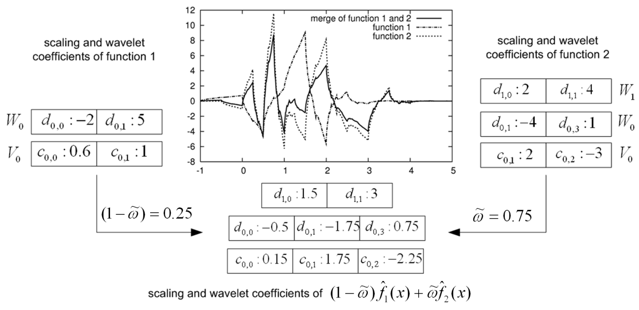

Figure 1. Convex merge of two functions given in their wavelet series expansion

5.3 Compression Step

The compression step is subsequent to the merge step. In case the new overall WDE does not fit into the available amount of memory, we must perform a suitable compression. As the memory allocation of a WDE is basically determined by its empirical coefficients, we compute the maximum number of coefficients M with respect to the available amount of memory. Hence, the task we tackle in the following is how to trim a WDE's coefficient number down to M in a suitable manner.

In Section 3, we already introduced a convenient strategy for compressing a function given in its wavelet series expansion. This strategy discards the least significant local details. In wavelet terminology, these details correspond to the wavelet coefficients with lowest absolute value. Therefore, we must sort the entirety of empirical wavelet coefficients by their absolute values and discard the required number of coefficients. Let us emphasize that this proceeding should not be applied to scaling coefficients. If we would proceed analogously with them, we would delete global features of the function and obscure its shape.

It may occur that all wavelet coefficients were discarded and the remaining scaling coefficients still allocate too much memory, i.e., their number exceeds M. In that case, the DWT gives us a simple solution. We decompose the scaling coefficients into scaling and wavelet coefficients of the next-lower resolution. Important in this context is that this gives us roughly half as many scaling coefficients. Again, we discard the resulting wavelet coefficients with lowest absolute value. The repeated application of this procedure ensures that we eventually get a set of coefficients, whose number is less or equal to M.

However, discarding wavelet coefficients always introduces a compression error because we loose local details. We introduce a measure for the relative error resulting from a compression step. Let $\hat{g}$ be the current overall WDE before and $\hat{g}^c$ the one after the compression. Let $I \subset \mathbb{Z} \times \mathbb{Z}$ be the set of indexes (j,k) of the discarded empirical wavelet coefficients $\hat{d}_{j,k}$. For the L_2-norm of the compression error $e(x) := |\hat{g}(x) - \hat{g}^c(x)|$, it follows

$$\begin{aligned} ||e||_2^2 &= || \sum_{(j,k)\in I} \hat{d}_{j,k}\psi_{j,k}(x)||_2^2 \qquad (13) \\ &= \sum_{l,m\in\mathbb{Z}} |\langle \sum_{(j,k)\in I} \hat{d}_{j,k}\psi_{j,k}(x), \psi_{l,m}\rangle|^2 \\ &= \sum_{(j,k)\in I} \hat{d}_{j,k}^2 . \end{aligned}$$

The first equation relies on Parseval's identity [13] and the second one on the orthonormality of wavelets. With respect to this identity, we define the **relative compression error** as

$$err := \frac{||e||_2^2}{||\hat{g}||_2^2} = \frac{\sum\limits_{(j,k)\in I} \hat{d}_{j,k}^2}{\sum\limits_{j,k\in\mathbb{Z}} \hat{d}_{j,k}^2}. \qquad (14)$$

The squared sum of wavelet coefficients is often defined as energy. Hence, the relative compression error quantifies the percentage of lost energy.

Overall, with the compression step, we completed the discussion of the processing steps required for using the framework. Algorithm 1 summarizes the general steps for computing WDEs over data streams.

Let us now consider the adaptation to changing system resources. We have to distinguish between two cases: an increase or decrease of the available amount of memory. In case the available memory of the overall WDE, i.e., its maximum number of coefficients, is increased, we simply determine a new maximum number. In case it is decreased, we can use the same mechanism as in the compression step to reduce the coefficient number. We successively discard the empirical wavelet coefficients with the lowest absolute values. In this context, the relative compression error can be used to quantify the loss in accuracy for different compression ratios before the actual compression is carried out.

Algorithm 1 Adaptive WDEs over a data stream

```
 1: for each incoming element do
 2:   insert element into current data block;
 3:   if block size $\geq b$ then
 4:     increment block counter $i$;
 5:     compute WDE $\hat{f}_i$ for the block elements;
 6:     if $i = 1$ then
 7:       $\hat{g}_i = \hat{f}_i$;
 8:     else
 9:       compute weight $\tilde{\omega}_i$;
10:       decompose empirical scaling coefficients of $\hat{g}_i$
          and $\hat{f}_i$ down to the minimum of their scaling
          resolutions;
11:       set $\hat{g}_i$ as convex merge of $\hat{g}_{i-1}$ and $\hat{f}_i$ with
          weight $\tilde{\omega}_i$;
12:     end if
13:     set $M_s$ as number of empirical scaling coefficients
        of $\hat{g}_i$;
14:     set $M_w$ as number of empirical wavelet coefficients of $\hat{g}_i$;
15:     while $M_s + M_w > M$ do
16:       if $M_s > M$ then
17:         discard all empirical wavelet coefficients;
18:         decompose empirical scaling coefficients
            down to the next lower resolution;
19:         update $M_s$ and $M_w$;
20:       else
21:         $p = 1 - \frac{M-M_s}{M_w}$;
22:         sort empirical wavelet coefficients by absolute value in descending order;
23:         discard the last $p \cdot M_w$ empirical wavelet coefficients;
24:       end if
25:     end while
26:     transfer $\hat{g}_i$;
27:   end if
28: end for
```

5.4 Algorithm Analysis

Before we go into the details of the algorithm analysis, let us briefly discuss the implementation concept for a data structure storing a set of empirical coefficients. We utilize a combined data structure termed **coefficient set**, which consists of a sorted list. Each list entry corresponds to one resolution and stores all coefficients of this resolution in a height-balanced binary search tree. The list is sorted by the resolution, while the trees are sorted by the translation index. Generally, a WDE has two coefficient sets: one for the scaling and one for the wavelet coefficients.

With respect to this implementation, let us summarize the computational complexity for processing a data block with b elements and updating the overall WDE.

The complexity of computing a block WDE is determined by the computation of the empirical scaling coefficients and the subsequent DWT. Recall from Section 3 that the number of empirical scaling coefficients to compute is linear in the size of the underlying sample. In combination with equation (6) follows a complexity of $O(b^2)$ for the coefficient computation. As the DWT has complexity $O(b \log b)$, the computation of a block WDE has an overall complexity of $O(b^2)$.

The computational complexity of the merge step is as follows. The current overall WDE stores $O(M)$ empirical coefficients; M is their maximum number. The block WDE stores $O(b)$ empirical coefficients. Scaling all coefficients of the overall WDE with the new weight has complexity $O(M)$. The scaling and subsequent insertion of the coefficients of the block WDE into the coefficient sets of the overall WDE has complexity $O(b \log M)$. Hence, the merge step has complexity $O(M + b \log M)$.

The complexity of the compression step depends on the coefficient number of the overall WDE. After the merge step, this number is $O(M + b)$. Sorting the elements by their absolute values and deleting the required number has complexity $O((M + b) \log(M + b))$.

If we combine the complexities of each step, we get an overall complexity of $O(b^2 + (M + b) \log(M + b))$ for processing a block of b elements.

5.5 Multivariate Wavelet Density Estimators

Finally, we sketch the case of WDEs over multidimensional streams. The crucial step is the computation of the block WDEs, i.e., how is a WDE over d-dimensional data defined? The starting point is a multivariate MRA [20] for $L_2(\mathbb{R}^d)$. This MRA can be exploited to define the wavelet series expansion of the unknown d-dimensional density with respect to a set of d-dimensional wavelet basis functions. By estimating the according scaling and wavelet coefficients of this expansion, a WDE can be established. Analogous to the onedimensional case, the possibly infinite number of empirical coefficients must be reduced by suitable strategies. For the case of multivariate linear WDEs, an according strategy has been developed in [19].

Given suitable block WDEs, we can compute overall WDEs in the same manner as in the onedimensional case. While processing the stream, we successively merge the empirical coefficients of the overall WDE with those of the current block WDE and, if necessary, trim their coefficient number by omitting those with lowest absolute value.

6 Related Work

The practical relevance of data stream processing and analysis has initiated the development of a plethora of techniques for mining and querying transient streams. In general, a mining technique must fulfill restrictive processing requirements to be applicable to a data stream [7]. Some mining techniques for static data sets were already successfully adapted to streams, e. g. approximate histograms [12], change detection [1].

The focus of our work is the estimation of continuous densities. In general, density estimation serves as core operation in many probabilistic learning methods [11]. A well-established method for density estimation utilizes kernels [17]. Kernel density estimation was successfully applied in many different scenarios, e.g. range query selectivity estimation [6]. Concerning its application to data streams, [3] provides an online approach that merges kernels located close to each other in order to fulfill preset memory requirements. In [18], an online variant of kernel density estimators is utilized to detect outliers in sensor networks. As already sketched in Section 4, we developed in [2] a framework for maintaining nonparametric estimators over data streams and placed kernel density estimators on top of it. In this work, we used the framework to adapt wavelet density estimators to data streams. We already presented an early stage of this work in [14].

The mathematical theory of wavelets is extensively discussed in [13, 5] and their application for statistical modeling purposes in [8, 13]. Wavelets were also successfully applied in different data mining techniques as the comprehensive overview in [16] illustrates. One of the most common applications of wavelets is the discrete transformation of numerical values, accompanied by a compression. These wavelet synopses aim to minimize the compression error with respect to the given values. Similar to this approach, [9] uses Fourier analysis to approximate an underlying finite signal. Up to this point, several approaches for efficient wavelet synopses were proposed, e.g. probabilistic ones [4]. Wavelet synopses offer to approximate simple aggregate, range, or inner product queries. In [10], a continuously maintained sketch of a data stream provides an approximation of wavelet coefficients during runtime. In contrast to those approaches, we exploit wavelets for estimating an unknown function in a well-defined manner.

7 Experimental Results

In order to assess our WDEs over data streams, we carried out a set of experiments, whose core results we present in the following. In the experiments, we examined the estimation accuracy of our WDEs and compared them with three competitive density estimation techniques for streams. Another aspect we addressed is how our WDEs react to sudden changes of the amount of available memory.

7.1 Test Environment

Data Streams As real-world data streams, we chose a set of time series from the UCR Time Series Data Mining Archive [15], which provides a large collection of time series from diverse application scenarios. Due to space constraints, we present in this work only the results for the following streams: Burstin, Physiological Data B1, Standard and Poor 500, Tide. We also examined other streams from this archive and basically observed the same trends for them in the results.

Besides the real-world streams, whose densities are per se unknown, we also examined Claw and CP2 [14], two synthetic streams whose true densities are known.

Techniques The objective of the experimental evaluation was to scrutinize the online WDEs presented in the previous sections. To convey an impression of their performance, we compared them with three other nonparametric techniques providing density estimates over data streams: a kernel technique based on the previously discussed framework [14], dynamic histograms [12], and M-Kernels [3]. For the sake of a fair comparison, each technique allocated the same amount of memory in the experiments, which we ensured by adequate parameter settings.

The online WDEs and the online kernel-based density estimators (KDE) rely on the framework presented in Section 4. Both processed data blocks consisting of 500 elements and weighted each block estimator equally. The memory requirements of online KDEs, which use cubic splines in the compression step, are determined by the number of spline coefficients. While their maximum number was set to 300, the maximum number of empirical coefficients of online WDEs was set to 100. The other parameter settings of online WDEs are based on the strategies discussed in Section 5. As underlying wavelet family, we decided for Daubechies5 wavelets as they produced the best results throughout the experiments.

Dynamic histograms (DynHist) [12] use a continuously maintained sketch of the stream to derive an approximate histogram with k buckets on demand. However, the computation of the histogram comes along with a heavy computational burden. From the different variants of DynHist presented in [12], we implemented an improved version of GREEDY which computes the optimal buckets in each iteration. As DynHist is a discrete method, we had to discretize the data domain into 100 bins of equal size beforehand. A larger number of bins did not improve the estimation quality significantly. Note that this binning presupposes the knowledge of the data range, a requirement that is not met in most real-world applications. This fact adversely affects the applicability of histograms. Concerning the memory requirements of DynHist, we set the size of the sketch to 100 units.

M-Kernels [3] provide kernel density estimators over a data stream. While processing the stream, a set of so-called M-Kernels is continuously maintained by merging the closest ones if necessary. A prerequisite for this merge is the numerical approximation of a minimum of a function by means of the downhill simplex algorithm. In our implementation, we set the number of iterations of this algorithm to 30. Additionally, a numerical approximation of an integral is required. On account of this necessity, we implemented the compound Simpson's Rule with 50 partitions. We also performed experiments with higher numbers of simplex iterations and integration partitions. This massively increased the processing time, but did not improve the estimation accuracy significantly. The memory requirements of this technique are basically determined by the number of M-Kernels, which we set to 160 in the experiments.

Quality Measure As accuracy measure for the difference between a continuously computed density estimate $\hat{g}$ and the density f that is to be estimated, we utilize the **mean squared error** (MSE):

$$MSE(f, \hat{g}) := \frac{1}{100} \sum_{i=1}^{100} (f(x_i) - \hat{g}(x_i))^2 \qquad (15)$$

where $x_1, ..., x_{100}$ is an equidistant partition of the support of f. While processing the stream, we computed the MSE after each 500 processed elements.

7.2 Quality for Real Data Streams

In the first set of experiments, we assessed the consistency of online WDEs. Hence, the main question was whether their estimation quality improves with the number of processed elements? To answer this question, we computed the best offline WDEs for the real-world streams. The best offline WDEs for a given stream refer to the best linear WDE, the best WDE with soft thresholding, and the best WDE with hard thresholding, each of them computed over the complete stream having unlimited memory at their disposal. While processing a data stream, we continuously compared the current online WDE with the corresponding offline WDE in terms of the MSE. Figure 2 displays the results for the different streams.

For all streams, we observe similar tendencies, namely a decreasing of the MSE the more elements are processed. This is an important result as it substantiates the consistency of our online WDEs. The more stream elements they process, the closer they approach the best offline WDE, even though they have only a very small amount of memory at hand. It is also worth mentioning that the MSE for the different streams is very low, mostly at around 10^{-4} and 10^{-5}. Generally, the MSE decreased very smoothly except for Standard and Poor 500, where it abruptly jumped.

Concerning the performance of the WDE types, linear WDEs achieved the best results; the thresholded variants were inferior. A closer look at the online thresholded WDEs revealed that their bounded number of coefficients did not allow them to capture all local details of their offline counterparts, which featured a very large number of coefficients.

Overall, we can state that our online WDEs only require a small amount of memory to keep pace with the stream and to provide suitable estimates of its density.

7.3 Quality for Synthetic Data Streams

The second set of experiments is similar to the first set, except that we examined synthetic streams, whose densities are known. The computation of the MSE with respect to the known densities allows us a fair comparison of our online WDEs with the competitors discussed above. Figure 3 presents the results for the different techniques. For reasons of clarity, online WDEs with hard thresholding are not displayed in this figure; they performed very similar to the variant based on soft thresholding.

As this figure indicates, M-Kernels were clearly inferior to all other techniques. We traced this effect back to their inadequate setting of the bandwidth, the essential parameter of a kernel density estimator.

For both synthetic streams, online linear WDEs and online KDEs achieved comparable estimation accuracies. While both were inferior to DynHist for the Claw stream, they were superior to DynHist for the CP2 stream.

Online thresholded WDEs have proved to be best; they outperformed all competitors for both streams.

We observe for all techniques only a slight decreasing of the MSE which, however, was very low for online WDEs.

Besides the estimation accuracy, we also examined in this experiment the processing time of each technique as this is a crucial factor in stream applications. More precisely, we measured the absolute processing time each tech-

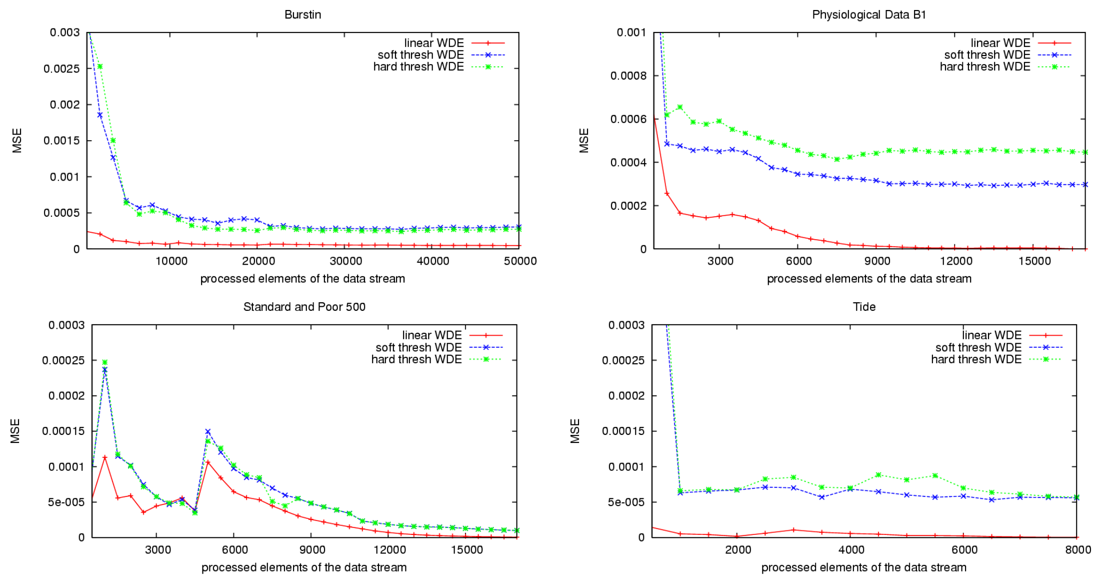

Figure 2. MSE of online WDEs for real-world streams

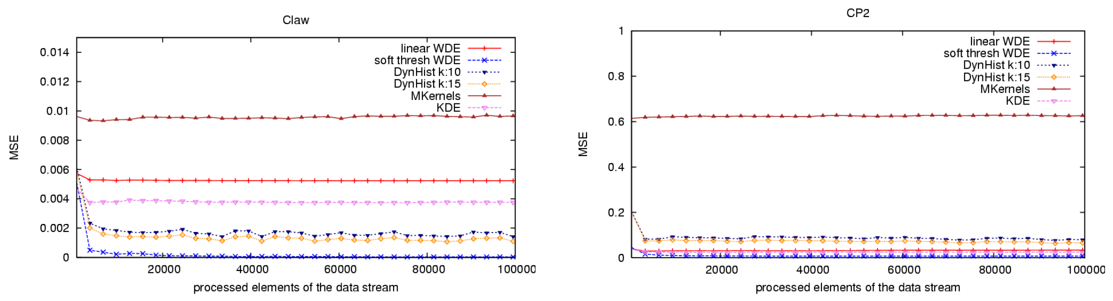

Figure 3. MSE of online WDEs and competitors for synthetic streams

nique required for processing the complete Claw stream. Table 1 summarizes the measurements.

Linear WDE	Soft Thresh WDE	KDE
9	13	11
M-Kernels	DynHist k:10	DynHist k:15
2473	14253	21548

Table 1. absolute processing times (in sec)

In terms of processing time, online WDEs and KDEs outperform DynHist and M-Kernels by orders of magnitude. Concerning DynHist, the expensive computation of an approximate histogram from the current sketch caused the high processing times. We also examined DynHist with more buckets, which resulted in better estimates, but even higher processing times. Concerning M-Kernels, the numerical approximations mentioned above caused a high computational effort.

Overall, if we take the estimation accuracy and the processing time into account, we can state that online thresholded WDEs are clearly superior to DynHist, online KDEs, and M-Kernels.

7.4 Memory-Adaptivity

We emphasized in this work the necessity of memory-adaptivity as it is a fundamental prerequisite for using an online analysis technique in a complex application. For that reason, we examined how online WDEs react to sudden changes of their available amount of memory.

The experiment we conducted had the following setup: Over a stream of CP2 data, we computed online linear WDEs. While processing the stream, we randomly varied the number of empirical coefficients of the online WDEs from minimum 10 to maximum 100 always after 5000 elements had been processed. To measure the effects of these memory modifications on the estimation accuracy, we computed the MSE between current WDE and CP2 density. Figure 4 summarizes the results of this experiment. In the charts, the x-axis displays the number of processed elements and the left y-axis the current MSE. The right y-axis displays the current number of empirical coefficients.

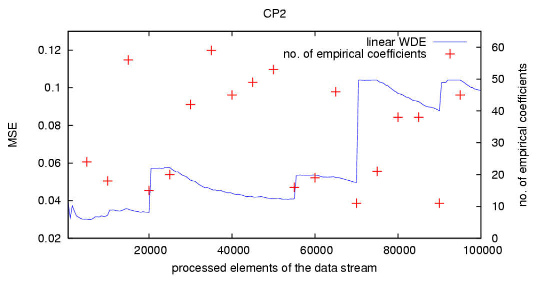

Figure 4. MSE of online WDEs for varying amounts of memory

We observe that online WDEs reacted in a flexible and adaptive manner to the changes. Significant reductions of the available memory adversely affected the estimation accuracy as the suddenly increasing MSE indicates. However, online WDEs quickly 'recovered' in terms of a henceforth decreasing MSE.

Thus, we can state that online WDEs meet the requirement of memory-adaptivity; they quickly adapt to changing amounts of available memory and succeed in providing suitable estimators meanwhile.

8 Conclusions

In this work, we investigated the nonparametric estimation of the probability density function of a data stream. We particularly concentrated on wavelet-based density estimators as they are superior to other nonparametric approaches based on kernels or histograms.

We derived our wavelet density estimators from a framework for maintaining nonparametric estimators over data streams. Basically, we build separate estimators for disjoint data blocks of the stream and successively merge them into an overall estimator. A well-defined compression scheme allows us to adapt the allocated memory of the estimator to the current system resources.

By means of an experimental evaluation, we have shown the feasibility of our approach. In comparison to dynamic histograms and M-Kernels, two competitive density estimation techniques for streams, our estimators were superior in terms of estimation quality and processing time. They also practically approved their ability to adapt to changing system resources in a convenient manner.

In our future work, we aim at coupling our estimators with change point detection methods in order to capture changes in the distribution underlying the stream. We will also delve more deeply into wavelet density estimators over multidimensional streams, an aspect we only briefly sketched in this work.

Acknowledgements

This work has been supported by the German Research Society (DFG) under grant no. SE 553/4-3.

References

[1] S. Ben-David, J. Gehrke, and D. Kifer. Detecting Change in Data Streams. In *VLDB*, 2004.

[2] B. Blohsfeld, C. Heinz, and B. Seeger. Maintaining Nonparametric Estimators over Data Streams. In *BTW*, 2005.

[3] Z. Cai, W. Qian, L. Wei, and A. Zhou. M-Kernel Merging: Towards Density Estimation over Data Streams. In *DASFAA*, 2003.

[4] A. Deligiannakis, M. Garofalakis, and N. Roussopoulos. A Fast Approximation Scheme for Probabilistic Wavelet Synopses. In *SSDBM*, 2005.

[5] R. A. DeVore and B. J. Lucier. Wavelets. In *Acta Numerica 1*. Cambridge University Press, 1992.

[6] C. Domeniconi, D. Gunopulos, G. Kollios, and V. J. Tsotras. Selectivity estimators for multidimensional range queries over real attributes. *VLDB Journal*, 2003.

[7] P. Domingos and G. Hulten. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, 2003.

[8] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Density estimation by wavelet thresholding. *The Annals of Statistics*, 24(2), 1996.

[9] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse fourier representations via sampling. In *STOC*, 2002.

[10] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries. In *VLDB*, 2001.

[11] A. Gray and A. W. Moore. Nonparametric Density Estimation: Toward Computational Tractability. In *ICDM*, 2003.

[12] S. Guha, P. Indyk, N. Koudas, and N. Thaper. Dynamic Multidimensional Histograms. In *SIGMOD*, 2002.

[13] W. Härdle, G. Kerkyacharian, D. Picard, and A.Tsybakov. *Wavelets, Approximation, and Statistical Applications*. Springer Verlag, New York, 1998.

[14] C. Heinz and B. Seeger. Wavelet Density Estimators over Data Streams (short paper). In *SAC*, 2005.

[15] E. Keogh and T. Folias. The UCR Time Series Data Mining Archive. `www.cs.ucr.edu/~eamonn/TSDMA`, 2002.

[16] T. Li, Q. Li, S. Zhu, and M. Ogihara. A Survey on Wavelet Applications in Data Mining. In *SIGKDD Explorations*, 2002.

[17] D. W. Scott. *Multivariate Density Estimation : Theory, Practice, and Visualization*. John Wiley & Sons, 1992.

[18] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data Using Non-Parametric Models. In *VLDB*, 2006.

[19] K. Tribouley. Practical estimation of multivariate densities using wavelet methods. *Statistica Neerlandica*, 49(1), 1995.

[20] B. Vidakovic. *Statistical Modeling by Wavelets*. John Wiley & Sons, 1999.