Abstract

When monitoring sensory data (e.g., from a wearable device) the context oftentimes changes abruptly: people move from one situation (e.g., working quietly in their office) to another (e.g., being interrupted by one’s manager). These context changes can be treated like concept shifts, since the underlying data generator (the concept) changes while moving from one context situation to another. We present an entropy-based measure for data streams that is suitable to detect concept shifts in a reliable, noise-resistant, fast, and computationally efficient way. We assess the entropy measure under different concept shift conditions (real and virtual concept shifts). To support our claims we illustrate the concept shift behavior of the stream entropy. We also present a simple algorithm control approach to show how useful and reliable the information obtained by the entropy measure is compared to an ensemble learner as well as an experimentally inferred upper limit. Our analysis is based on three large synthetic data sets representing real, virtual, and a combination of both concept drifts under different noise conditions (up to 50%). Last but not least, we demonstrate the usefulness of the entropy-based measure context switch indication in a real-world application in the context-awareness/wearable computing domain.

1 Introduction

In real-world applications the mining of data streams, rather than time-independent data, is increasingly important. In many applications data (e.g., from the financial industry, sensor data, multimedia content) is gathered over time, which raises the problem that the concepts to be learned may drift (i.e., change) over time [21]. Also, the increasing amount of data (e.g., multimedia content, data warehouses) and the limitation of computing power due to miniaturization (e.g., wearable computing) call for faster and more resource-friendly algorithms. The motivation for this paper is a real-world problem which stands exemplary for the problem mentioned above – the analysis of sensor data on wearable devices. In our research on context-awareness [1], where we learned classifiers predicting peoples’ anticipated behavior based on sensory input, we found that contexts (or contextual situations) switch rather than gradually change. We also found, that contextual information could be reused, even for new, not yet encountered situations. Therefore, an ongoing monitoring of the sensor stream is needed. An online pattern matching mechanism comparing the sensor stream to the entire library of already known contexts is, however, computational complex and not yet suitable for today’s wearable devices. One solution is to indicate possible candidates (or hot spots) for context changes limiting the computationally intensive context (re-)determination on those candidates. Thus, a computationally "cheap" technique to find such context-switch candidates would be very helpful. From the machine learning point of view the context generating the sensor data can be viewed as the underlying concept generating the data stream and the context switches can be viewed as "abrupt concept drifts" or better concept shifts (further discussed in the next section).

This paper introduces an entropy-based measure [19] to detect concept shifts. In the following we will show that this measure is very sensitive to concept shifts while remaining noise-tolerant. Additionally, it allows to distinguish between different shift intensities. In order to be able to assess this measure, we introduce a coarse concept shift adapting algorithm, which we show to (1) provide mostly a better prediction quality than conventional approaches, (2) require limited computational power, (3) exhibit quick reaction time, and (4) show good performance under noisy conditions. After the assessment of the algorithm on synthetic data sets we apply our approach to sensor data obtained by a context-aware wearable computing setup [1], where the entropy measure clearly indicates context switches on the basis of audio and accelerometer recordings.

The next section provides a review on the related work with focus on the nature of concept drifts and relates our
switching contexts from sensor data, it focuses solely on Since this paper is motivated by the problem of indicating can occur suddenly (abruptly, instantaneously) or gradually. problem is known as concept drift,...¨

....

make the model built on old data inconsistent with the new data distribution may change as well. Often these changes follow:

-tions to other projects in the field. Section 3 introduces our novel concept shift measure and algorithm. To evaluate our proposed measure and algorithm, section 4 presents the experimental setup, synthetic data sets, and benchmarks including an (experimental) upper limit for the learning algorithms used in this study. The following sections present/discuss the results and are followed by a presentation of our approach’s performance on the real-world data set. We close with the limitations, future work and a final conclusion section.

2 Related Work

In his survey paper Tsymbal defines concept drifts as follows: “In the real world concepts are often not stable but change with time. Typical examples of this are weather prediction rules and customer preferences. The underlying data distribution may change as well. Often these changes make the model built on old data inconsistent with the new data, and regular updating of the model is necessary. This problem is known as concept drift." [21]. Obviously, drifts can occur suddenly (abruptly, instantaneously) or gradually. Since this paper is motivated by the problem of indicating switching contexts from sensor data, it focuses solely on sudden concept drifts, which we call concept shifts1. Widmer and Kubat [23] differentiate between changes in the actual target concept called real concept drifts and changes in the distribution called virtual concept drifts. Tsymbal [21] also states that "...from the practical point of view it is not important, what kind of concept drift occurs, real or virtual, or both. In all cases the current model needs to be changed." We found that our entropy measure is able to distinguish between these two kinds of drift. Thus, we will show, however, that virtual and real drifts are different and should, hence, be treated differently.

Given the frequent occurrence of concept drifts in machine learning applications, a number of concept drift handling approaches have been developed in the literature. These, typically, adapt to drift by (1) instance selection, (2) instance weighting, or (3) ensemble learning, whereas instance selection can be viewed as a special case of instance weighting using binary weights. Most instance selection approaches use a window-based approach, i.e., they consider all instances within a certain time-window for model induction and disregard (or “forget”) all other instances. Algorithms of the FLORA family [23] and FRANN [24] make use of the prediction error (on the last instances in a data stream) to control the window size. [10] uses different metrics (such as the f-measure) to compare a reference window representing the last concept with a window on the very latest instances to select the most relevant instances for model induction. None of the metrics employ entropy as measure in contrast to our approach in section 3. All of the window-based approaches assume that the very latest instances are the most important for the current model. [11], in contrast, considers not only the age, but also the contribution to the current concept for instance weighting.

Rather than relying on a selection of instances ensem-

le learning methods combine a selection of models typically resulting in very robust and precise but computationally complex predictions [18, 20, 22, 12, 9, 2, 15, 13]. Some algorithms can also re-activate previously stored models if a concept recurs such as FLORA3 [23] or PECS [17].

While the literature reports on many comparisons between algorithms of a specific “family” we found no benchmarks against a theoretically or experimentally derived upper limit.

Another major issue in the literature are the data sets for evaluation and/or benchmarking different approaches. The most used synthetic data sets are STAGGER [18], the moving hyperplane in a cube [8, 22], the SEA concept [20], and a moving sphere in a unit cube [2]. All these data sets allow to control the type, rate, and recurrence of the concepts as well as setting the noise level and adding irrelevant attributes. A major disadvantage of these data sets is that the class-distribution cannot be changed independently from the target concept and vice versa – complicating the comparison of virtual and real drifts. The literature also presents some real-world data sets [6, 8, 20]. Unfortunately, these typically show little concept drifts and/or are sometimes adapted for evaluation purposes making it difficult to assess their usefulness as a benchmark.

Our work distinguishes itself from previous studies in the following ways. First, we introduce novel synthetic data sets based on the idea of a rotating hyperplane [8, 22]. This setup allows us to investigate real and virtual drifts independent of each other, which supports a comprehensive assessment of concept drift approaches. Second, we present an experimentally determined upper limit for the used learning algorithms on the presented data sets. Third, we show that real and virtual drifts cannot be lumped together as stated in [21]. Fourth, our main contribution is the introduction of an entropy-based measure as concept shift indicator, which is able to quantify the intensity of the shift as well as differentiate between real and virtual shifts. To be able to assess the power of the measure, we introduce a simple window-based algorithm using the entropy measure. This algorithm shows it’s strength compared to ensemble classifiers both with regards to quality and computational performance on the synthetic data set. Last but not least, we show the usefulness of the measure in the context of a real-world wearable computing data set.

1We have strong indication that our approach also holds for gradual drifts, but such an investigation goes beyond the scope of this paper.
3 Entropy and Concept Shift Adaption

In this section we motivate and introduce the entropy measure applied on data streams. Our approach bases on the following assumptions: 1) As long as the distribution of older instances (features and target values) is similar to the distribution of new instances no concept drift occurred. 2) A distribution difference between old and more recent instances indicates a change in the target concept. Hence, the current model may be outdated and needs to be adjusted.

To measure the distribution inequality we make use of the entropy to compare old and new instances of a data stream. If two distributions are equal, the entropy measure results in a value of 1, if they are absolutely different the measure will result in an value of 0. Although entropy is well known from information theory [19] as a measure for information content - and its application, thus, is self-evident - we make use of it mainly because of its symmetry and additive properties. This section first specifies how to tailor the entropy measure for data-streams and discusses some features of this entropy measure. Based on these features we introduce a simple coarse instance-selection algorithm, which allows us the evaluation of the measure in the next sections.

3.1 Calculating Entropy on Data Streams

Shannon’s entropy [19] is a measure of the amount of information in a set and is defined as follows: $H(x) = \sum_x P(x) \log_2[P(x)]$. To use this measure in the context of data streams we have to adapt it. To that end we chose the sliding window technique, which compares two windows, one representing older and the other representing more recent instances (light and dark gray areas in Figure 1), in the stream. The application of entropy on the comparison of these two windows is not straightforward. Essentially, we compare the two windows by counting and comparing all instances with respect to their class and stream membership. Additionally, we discretize the range of instance values to a fixed number of bins to take the approximate value distribution into account. This subsection explains the calculation of the stream entropy in detail. We define a data stream as a sequence consisting of sequentially ordered tuples $d_i$ in time $t_i$, where $i \in \{1, 2, 3, \ldots\}$. Each tuple $d_i$ consists of $S$ feature streams $\vec{s}$ and one label stream $\vec{l}$, formally $d_i := (\vec{s}_i, \vec{l}_i)$, where $\vec{s}_i$ is the vector of all feature stream instances $s_{ni}$ at time $t_i$. For classification problems, the domain of the label stream $\vec{l}$ is discrete and contains all class values $c \in C$. In the following evaluation on the synthetic data sets, for example, we will limit all experiments to 2 class problems (i.e., $C = 2$) with 3 feature streams ($S = 3$). Thus each $d_i$ has four scalar values (see section 4.1).

Let $H_i$ be the resulting entropy at time $t_i$. $H_i$ is defined as the mean of all data stream entropies $H_{is}$ at time $t_i$.

$$H_{is} = \frac{1}{S} \sum_{s=1}^{S} H_{is}$$

where $S$ is the number of feature-streams and

$$H_{is} = \sum_{c=1}^{C} \sum_{b=1}^{B} H_{iscb}.$$  (2)

Equation 2 shows that $H_{is}$ is calculated from the entropies $H_{iscb}$, that represent the entropy of each class ($c \in C$) and bin ($b \in B$) given the stream $s$ at time $t_i$. We introduced the bins as discrete aggregation of the values of each feature stream $s$. For $B > 1$ this allows to detect changes in the feature value set distribution (if the distribution is uniform enough, which holds for most real-world applications). To simplify the presentations we will use 2 bins for all calculations ($B = 2$) – all calculations generalize to more bins.

$H_{iscb}$ is generated by calculating the entropy of the two sliding windows (shown in Figure 1). Thus,

$$H_{iscb} = -w_{iscb} \log_2 (\varrho_{iscb}) + w_{iscb} \log_2 (\varrho_{iscb})$$

where $\varrho_{iscb}$ represents the probability of the instance-types and $w_{iscb}$ is a weight. We will discuss each of these in turn. $\varrho_{iscb}$ is the probability that an instance occurs in the old window at time $t_i$, belonging to class $c$, with the feature domain of stream $s$ in bin $b$. Specifically (Eq. 4), $v_{iscb}$ is the number (or count) of instances in the "old" sliding window at time $t_i$ belonging to class $c$, with the feature domain of stream $s$ in bin $b$, which we normalize by the total number of instances in this "old" window $\lambda_{iscb}$. The normalization transforms the count $\nu$ to the probability $\varrho$ and ensures the validity of the formula if the two sliding windows would be of different size. Obviously, $\varrho_{iscb}$ is calculated analogously.

$$\varrho_{iscb} = \frac{v_{iscb}}{\lambda_{iscb}}, \varrho_{iscb} = \frac{v_{iscb}}{\lambda_{iscb}}$$

$w_{iscb}$ is the weight assigned to each bin $b$ (e.g., if we like to focus on bins at the edge of the distribution), to different
classes \( c \) (e.g., for cost sensitive classification – see section 4.2), to different streams \( s \), or to different points in time \( t \). The border condition \( \sum_{s=1}^{S} \sum_{c=1}^{C} \sum_{b=1}^{B} w_{iscb} = 1, \forall i \) must be fulfilled in order to keep the entropy in the range \([0,1]\). Again, to simplify the calculations we will choose \( w_{iscb} = 1 \) for \( \forall i, s, c, b \) and keep both sliding windows of all streams at the same length (i.e., \( \lambda_{old} = \lambda_{new} \)).

Since \( H_i \) is dependent on the overall class distribution we normalize it using the normalization factor \( H_{inorm} \). \( H_{inorm} \) is computed similar to \( H_i \), with the difference that only the instance counts per class are concerned without regarding the streams and the bins. Thus, it reflects the pure change of the class distribution. Consequently, \( \tilde{H}_i \) is the entropy normalized on the class distributions at time \( t_i \).

\[
\tilde{H}_i = \frac{H_i}{H_{inorm}}, \text{where } H_{inorm} = \sum_{c=1}^{C} w_{ic} \varrho_{ic} \log_2(\varrho_{ic}) \quad (5)
\]

### 3.2 Properties of the Entropy Measure

Consider a simple data set having a single concept shift (e.g., generated like the “real” shift data set presented in section 4.1), which is symbolized by a concept parameter in the upper part of Figure 2. The lower part of the figure shows how the entropy measure indicates a concept shift under two different constant window sizes \( \lambda \) (\( \lambda_{old} = \lambda_{new} = \lambda \), i.e., when the "old" window and "new" window are of equal length). The bold dotted line stands for the entropy measure calculated on a \( \lambda \) of the length 1000 and the solid line’s \( \lambda \) is of the length 100. Note, that in absence of any shift the measure shows a value of 1 (the noisy behavior of the solid line is a statistical sampling effect due to the small number of considered instances). The Figure also shows that larger sliding windows produce smoother curves than the smaller ones, but the smaller ones show a sharper and more distinctive peak – a typical trade-off problem between a faster response to shifts (small window) and robustness to noise (large window). Also, larger sliding windows show delay of the indication of the full concept shift. Given that we want to optimize on sensitivity and reaction time we decided to conduct all following experiments with a small window size of 100 instance tuples for both the "old" and "new" sliding window for computing the entropy.

### 3.3 Algorithm Control Strategy using Entropy Measure

Encouraged by the indications in the last subsection that the entropy measure is able to detect and measure concept shifts, this subsection focuses on developing a simple, coarse algorithm that automatically adapts to concept shifts based on the entropy measure. This allows us to benchmark our algorithm to other approaches in terms of prediction power. Thus, we can draw conclusions from these comparisons for our entropy-based measure as concept shift indicator. Our approach is an instance selection style algorithm that adapts the window size whenever the entropy measure detects a shift. The window size control strategy is based on the very simple rule depicted in Figure 3. Let us assume that we start before a shift and the entropy measure value is at (or near) 1 and the window of the algorithm is of some given size \( \xi \). When a shift occurs the entropy measure reacts. If it intersects an arbitrary chosen threshold \( \tau \) we collapse the window size \( \xi \) of the algorithm to a minimal size and let it grow again by the newly arriving instances to an upper threshold, resulting in a linear recovery of the window size after the drift. Thus, every time the entropy intersects with the threshold (with a negative slope) the algorithm "forgets" its current model and starts to relearn on the most recent instances.

In the remainder of this study we have chosen a fixed threshold \( \tau = 0.95 \) and set the lower bound window size \( \xi_{\text{lower bound}} \) to 20 and the upper bound window size
\( \xi_{\text{upper bound}} \) to 1000 instances (the only reason to introduce an upper bound was to allow a fair comparison with the benchmark algorithms presented in section 4.2, which have a maximum window size of 1000).

4 Experimental setup

When evaluating algorithms that learn from data streams with concept drifts, researchers are faced with a new set of issues. While the traditional induction algorithm performance measures such as accuracy and (area under the) ROC-curve still apply, they provide no information about how good an algorithm could get assuming it had perfect knowledge about the drifts. Hence, we are missing an absolute benchmark for learning algorithms. Also, it is difficult to find data sets that cleanly differentiate between real and virtual concept drifts. In this section we, therefore, introduce three generated data sets and their corresponding upper bound benchmarks models, which are based on an optimal window size choice approach. Additionally, we present three ensemble based benchmarks to compare our simple approach (see section 3.3) to state of the art concept drift adapting algorithms.

4.1 Data sets

For a comprehensive analysis of concept drift algorithms the first requirement to a benchmark data set is that it needs to differentiate between virtual and real drifts—optimally we would have data sets with either drifts and one combining the two. Furthermore, we need to ensure that the data sets don’t contain any artifacts from their generation such as asymmetrical features or other hidden dependencies. As we did not find any benchmark data set in the literature conforming to these requirements we adapted the method of [22] to generate our own synthetic data set. Our data set domain consists of a sphere containing all instances and a plane intersecting this sphere through its origin. The orientation of the plane is defined by a three dimensional vector \( \vec{n} \) standing perpendicular of the plane’s surface. Instances above the plane belong to class A and instances below the plane belong to class B. Hence, this mechanism defines a two class problem. We obtain the overall data set by combining three random and independent data streams with a fourth data stream generated by the rule above. Formally, let

\[
d_i = (s_i, l_i) \text{ the tuple at time } t_i \text{ and } s_i = (s_{1i}, s_{2i}, s_{3i})^T
\]

where the instances \( s_1, s_2, \) and \( s_3 \) are randomly chosen from the discrete set \{ -0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.4, 0.5 \} and represent a point in the three dimensional space satisfying the following equation in Cartesian coordinates

\[
s_1^2 + s_2^2 + s_3^2 \leq 0.25 \tag{7}
\]

and \( l_i \) is the label calculated by the following rule

\[
\begin{align*}
\text{if } \text{sgn}(\vec{s} \cdot \vec{n}) = \begin{cases} 1 & \rightarrow \text{assign class } A \\ -1 & \rightarrow \text{assign class } B \\ 0 & \rightarrow \text{randomly assign class } A \text{ or } B 
\end{cases}
\end{align*}
\tag{8}
\]

as depicted in Figure 4.

**Real drifts** are produced by rotating the plane around the origin. In our design we have chosen the axis of rotation (of \( \vec{n} \)) to be congruent to the \( s_3 \) axis, such \( s_3 \), will not affect the class assignment (irrelevant attribute). **Virtual drifts** are produced by changing the class distributions defined by \( \psi_A \) and \( \psi_B \) without varying \( \vec{n} \). All data sets discussed below consist of consecutive intervals with 3000 instances each, where the class distribution and rotation settings remain constant, representing a concept. Abrupt concept shifts result from having intervals with different settings (i.e., \( \psi_A, \psi_B, \) and \( \vec{n} \)) following one after another. The result is a data stream time series starting with \( \vec{d}_1 \) and ending with \( \vec{d}_{3000 \kappa} \), where \( \kappa \) is the total number of of concepts. For each concept segment in the data set we created a separate test set consisting of 10000 instances representing the same concept. To analyze real shifts only we created a **real shift data set** with constant equal class distribution \( \psi_A = \psi_B = 0.5 \), but different directions of the \( \vec{n} \) vectors at a concept shift. Specifically, we chose 20 absolute angles (in spherical coordinates: the azimuth \( \Theta \), see Figure 4) representing the direction of \( \vec{n} \) in the following order \( (0 \pi, 1/128 \pi, 3/128 \pi, 7/128 \pi, 15/128 \pi, 31/128 \pi, 55/128 \pi, 87/128 \pi, 127/128 \pi, 175/128 \pi, 231/128 \pi, 39/128 \pi, 111/128 \pi, 191/128 \pi, 23/128 \pi, 119/128 \pi, 223/128 \pi, 79/128 \pi, 199/128 \pi, 199/128 \pi) \). This absolute values correspond to a relative increase of the \( \vec{n} \) rotation (intensity of
The virtual shift data set was created to contain virtual shifts only. We created it with a constant plane orientation $\vec{n} = (1, 0, 0)^T$. We considered all transition possibilities between the class distributions of class $A \psi_A \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ (note $\psi_B = 1 - \psi_A$) resulting in 72 drifts between 73 concepts. The order and $\psi_A$ values of the concepts are $\{0.1, 0.5, 0.9, 0.4, 0.8, 0.3, 0.7, 0.2, 0.6, 0.1, 0.6, 0.2, 0.7, 0.3, 0.8, 0.4, 0.9, 0.5, 0.1, 0.4, 0.7, 0.1, 0.3, 0.6, 0.9, 0.3, 0.5, 0.8, 0.2, 0.5, 0.7, 0.9, 0.2, 0.4, 0.6, 0.8, 0.1, 0.8, 0.6, 0.4, 0.2, 0.9, 0.7, 0.5, 0.2, 0.8, 0.5, 0.3, 0.9, 0.6, 0.3, 0.1, 0.7, 0.4, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$. The length of this data set is 219000 instances. (see Figure 8, top curve).

The mixed shift data set combines both real and virtual concept shifts. To that end we reused and extended the real shift data set by overlaying it with varying class distributions $\psi_A \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ (note $\psi_B = 1 - \psi_A$) resulting in 72 drifts between 73 concepts. The order and $\psi_A$ values of the concepts are $\{0.1, 0.5, 0.9, 0.4, 0.8, 0.3, 0.7, 0.2, 0.6, 0.1, 0.6, 0.2, 0.7, 0.3, 0.8, 0.4, 0.9, 0.5, 0.1, 0.4, 0.7, 0.1, 0.3, 0.6, 0.9, 0.3, 0.5, 0.8, 0.2, 0.5, 0.7, 0.9, 0.2, 0.4, 0.6, 0.8, 0.1, 0.8, 0.6, 0.4, 0.2, 0.9, 0.7, 0.5, 0.2, 0.8, 0.5, 0.3, 0.9, 0.6, 0.3, 0.1, 0.7, 0.4, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$. The length of this dataset is 219000 instances. (see Figure 7, top curve).

Throughout this evaluation we have chosen to use a batch version of the Naïve Bayes algorithm as it is known for its robustness, does not require much computational power, and also produces class probability estimations as required for the ROC curve computation. We adjusted the predictions of the algorithms using the Laplace Estimation [3]. As using a sliding window technique, we induced the model not on all instances available at time $t_i$ but on a window $w$ of size $\xi$. Thus, the window used was $w_{\xi,i} = [d_{i-\xi}, d_{i-\xi+1}, ..., d_i]$.

In section 3.3 we presented a general rule to adapt an algorithm based on the outcome of the entropy measure. To adapt the algorithm to the two different quality measures, one that is class distribution dependent (accuracy) and the other that is class distribution independent (AUC), we also used two different variations of the entropy measure: For the “accuracy scenario” we employed the class distribution dependent entropy measure $H$ as trigger, for the “AUC scenario” we used the class distribution independent (normalized) entropy measure $\tilde{H}$. The algorithm controlling threshold $\tau$ of the entropy measure has been set to 0.95 for all data set and quality measure combinations (as mentioned in Section 3.3).

### 4.3 Benchmarks

To compare our solution against two accepted standards we calculated a representative set of benchmarks on the three data sets presented in Section 4.1. First, a so-called perfect benchmark, which assumes an oracle-given ideal window size $\xi$ for any point in time, and second, a selection of ensemble classifiers, which the literature [12] so far showed to have the highest accuracy and robustness against noise. We limit the training set of all classifiers to a maximal window size of 1000 to keep the range in the order of magnitude of a single concept length as used in the synthetic data sets.

#### 4.3.1 Perfect Benchmark

The perfect benchmark represents an unbeatable benchmark, as we assume that it always learns the model given the best performing (oracle-given) window size $\xi_i$ for any given time $t_i$. To determine the optimal window size we computed 100 Naïve Bayes classifiers for any point of time $t_i$ within the data set, each having a different window size $\xi \in \{10,20,30, ..., 900, 1000\}$, and chose the best performing one as the predictor for our model at that point of time. The result was an experimentally optimal Naïve Bayes model for each point in time akin to having an oracle provide the optimal window size for each point in time. Given that the goal is to provide a mechanism to determine the optimal window size at any point in time in order to adapt to any concept drift based on past data alone this benchmark provides an experimental upper bound for all Naïve Bayes based approaches.
4.3.2 Ensemble Benchmarks

As mentioned in the introduction, the most robust and accurate approach to handle concept drifts so far have been ensemble (or committee) classifiers [12]. To compare against this quasi-standard, we evaluated several ensemble decision policies to get the three best performing ensemble classifiers presented in this evaluation. The ensemble’s basic set consist of 9 members having the window sizes \{10, 25, 50, 75, 100, 250, 500, 750, 1000\}. The decision policy of the first ensemble was based on an equal weighted voting between the three best performing members (10-fold cross-validated on their training set). The decision policy of the second ensemble considered all members but weighed with the quality measure derived on their training set (also 10-fold cross-validated). The third algorithm used a linear weighting of all ensemble members according to their evaluation ranking on the training set.

5 Experimental Results

In this section we discuss first some fundamental observations based on the Figures 7, 8, 10, and 11 (see at the end of the paper). Second, we will have a closer look at the predictions of all algorithms against noisy conditions (Figures 9 and 12).

Figures 7, 8, 10, and 11 provide an overview of the results and an intuitive understanding of the entropy measure. All of these four figures are composed the same way. The curve (A) at the top represents the concept drift parameter. For real shifts this is the vector orientation angle \(\Theta\) (see Section 4.1). For virtual shifts the concept is represented by the class distribution \(\psi_A\). We do not present the curves on the mixed data set because their behavior is consistent with curves presented on the real and the virtual shifts and thus, these curves do not provide any new information. Curve (B) is the derived entropy \(H\) for the "accuracy scenario" and \(H\) for the "AUC scenario" respectively. The curves (C) and (D) represent the perfect benchmark. (C) shows the highest reachable accuracy (or AUC) by a window based forgetting Naïve Bayes algorithm. (D) shows the corresponding window size in order to reach that prediction above. The two last curves (E) and (F) represent the accuracy (or AUC) and the window size of our entropy measure based algorithm. All curves are calculated under noise-absent conditions.

First, we focus on the experiments reporting the results in accuracies (Figures 7 and 8). The entropy based measure (B) reflects the concepts shifts (A) on both data sets and each amplitude corresponds to the shift intensity. The accuracy of the perfect benchmark (C) is a little bit biased by the prior class distribution as it follows the top line (A) in Figure 8 in contrast to Figure 7. The window size behavior of the two algorithms ((D) and (F)) are very similar on real shift data except on the very small shifts that are overlooked by the entropy based algorithm. The perfect benchmark behavior on the virtual data is totally different. It shows vehement window size variations - even in non-drifting sequences. The entropy based approach is, again, synchronous to the virtual shift - except for the very small ones.

The most eye-catching feature in the AUC quality measure results is the algorithm behavior on the virtual shift data set. Since the AUC measure is insensitive to class distribution changes, the model does not need any update (the outliers of the perfect benchmark are caused by a little imbalance in the data set due to the random class assignment of the instances located on the class separation border). Here, a non-adapting algorithm performs nearly as well than the perfect benchmark. Thus, the normalized entropy measure \(H\) is ideal for this problem by ignoring virtual shifts, but indicating real shift in the exact same way than \(H\) (Fig. 7).

The Figures 9 and 12 show the prediction qualities (accuracies and AUC respectively) on the three data sets against increasing noise levels. All six graphs show that our algorithm (dashed line) is as noise resistant as the benchmarks. The graphs in both of the figures shows also that our algorithm outperforms the ensemble classifiers - except for the virtual shift dataset (Fig. 9). The fact that our algorithm cannot keep up with two of the ensemble algorithms for this setting indicates that our simple coarse approach is insufficient for this situation, although the entropy indicates almost every virtual concept shift (Fig. 8). The graphs on the right show the results on the mixed data set. These results are sound regarding the results of the real and the virtual shift results.

Finally, we conducted experiments concerning computational complexity. We compared the ensemble classifiers and the entropy measure based algorithm. We first measured the elapsed time for all three committee classifiers for both of the quality measures. The elapsed time was about the same such we decided to report it as mean and standard deviation. The computation\(^2\) of 10000 tuples taken from a data set presented in section 4.1 required 2031.6 ± 15s. The entropy based algorithm required 148.6s, which is 13.7 times faster than the committee. The entropy calculation without following Naïve Bayes model building requires only 1.1 ± 0.1s. This indicates that the performance difference originates from the number and size of the used Naïve Bayes models. This emphasizes the computational advantage of our approach as expected.

As we found it difficult to get benchmarks, we intend to publish all our data sets, predictions, and resulting quality measures to simplify future comparisons.

\(^2\)Using Matlab on a 3 GHz Pentium 4 machine with 1 GByte RAM.
6 Discussion of the Experiment

It is remarkable that the simple coarse algorithm based on the entropy measure outperforms the ensemble benchmark algorithm on real concept shifts for both quality measures. This confirms that the entropy measure is a very good indicator for detecting and controlling an algorithm adapting to real concept shifts. Also, our algorithm is one order of magnitude faster than the ensemble approach, because our approach calculates the Naïve Bayes algorithm only once, whereas the ensemble requires a Naïve Bayes calculation for each of its members. Hence, our algorithm exhibits a greater predictive power while requiring less computational resources. Note that the calculation of the entropy measure only accounted for less than 1% of the computational requirement of our algorithm. Furthermore, the entropy measure based algorithm showed the nearly the same robustness towards noise as the perfect benchmark and the committee classifiers – both of which are known to be very robust towards noise. Hence, it seems that our simple, coarse entropy based algorithm approaches the goal of an ideal learner set by [24]: "An ideal learner should combine robustness to noise and sensitivity to concept drift." To reach this goal we invested the domain knowledge that the structure of the examined drifts is abrupt; i.e., that the domain exhibited concept shifts rather than concept drifts. But this assumptions holds for our initial real-world problem as we will show in the next section.

Last but not least, we have shown that real and virtual drifts cannot be lumped together as stated in [21]. While lumping the two types of drift might be useful for some real world applications using accuracy as quality measure, we found that this is definitely an artifact of the use of accuracies. Addressing the issue of real versus virtual drifts we showed that the entropy measure is able to cope with both cases. Either by using the class distribution dependent entropy measure \( H \) or the class distribution independent \( \tilde{H} \).

7 Application to a Real-World Problem: Context Switches in Sensor Data

As mentioned in the introduction section, the original motivation for the entropy based measure was the monitoring of sensor data streams for context switches. To demonstrate that functionality we use the exact same data set as presented in a prior study [1]. The data set consists of audio and accelerometer data recorded over a time of 15381 seconds. The wearable data acquisition setup included a microphone and three three-dimensional accelerometers attached on the subject’s shoulder, wrist, and leg. To illustrate the applicability of the measure we focus on the audio stream and one single accelerometer (leg, would correspond to a mobile device’s accelerometer carried in a pocket). The data was preprocessed in a very simple and fast way as it could be performed e.g. on a smart phone resulting in one feature vector for each second. The audio signal was decomposed into 10 features: spectral center of gravity, temporal fluctuations of spectral center of gravity, tonality, mean amplitude onsets, common onsets across frequency bands, histogram width, variance, mean level fluctuations strength, zero crossing rate, and total power. The accelerometer data was merged in one single feature: the absolute value of the amplitude. To calculate the entropy based measure we applied the exact same parameters as used in the evaluation before. As input stream we have chosen the audio features and as target class we picked the accelerometer feature which has been discretized to represent a two class problem (large and small acceleration). We had 2 bins for each input stream, and chose a window size of 100 instances (=100 seconds).

The upper line in Figure 5 shows the entropy \( H \) calculation on the sensor data and the lower line illustrates the subject’s actual context sequence (scenario). The scenario consists of 6 context situations: (A) walking, (B) streetcar, (C) office work, (D) lecture, (E) cafeteria, and (F) meeting. The large peaks in the entropy measure look synchronous to the concept shifts. Based on this observation we can construct an algorithm that indicates a context switch every time the entropy crosses a given threshold (analogously to the algorithm introduced to adapt to the shifts, see Section 3.3). If we arbitrarily choose the threshold to be, e.g., 0.7 the algorithm would indicate 17 of total 18 context switches and six times cause a "false alarm". The one context switch at 8462 seconds is not detected because its signal overlaps with the signal of the context switch just before at 8391 seconds. Going back to the original data we found that the six "false alarms" were actually correct: the subject had, e.g., been interrupted shortly during office work or stood up to get a second coffee in the cafeteria. These situations do not appear in the coarse-grained report above. This shows that we are able to detect even more fine-grained context switches than reported (by the subject). Raising the threshold even further will result in increasingly fine-grained indications of context switches – not only concept switches between "walking" and other context situations. Additionally, the intensity of the peaks indicates the magnitude of the context switch. Hence, one can derive some degree of similarity between the context situations, which might be used to control the granularity of the segmentation. We also calculated the class-distribution independent entropy \( \tilde{H} \), which turned out to be very similar to \( H \). Thus, we are dealing with real (not virtual) shifts in this problem.

---

3We are well aware that there is a huge potential of improving our results by fine tuning the parameter settings, but we only want to show that satisfying results can be achieved - even with the most simple settings.
8 Limitations and Future Work

To support our claims we showed the power of the entropy concept drift indicator by controlling the instance selection of a Naïve Bayes algorithm on discrete attribute data sets containing instantaneous concept drifts (i.e., concept shifts). For some (randomly) chosen settings we conducted similar experiments by controlling a K-Nearest Neighbor KNN (with $k = 21$) algorithm on data sets with continuous attributes, which reconfirmed the findings reported above. Due to the limited computational resources (required to compute the perfect benchmark) we did not further investigate this avenue for this paper. Nevertheless, to further test the generalizability of our approach to continuous attributes we intend to conduct such experiments with a variety of synthetic as well as real-world data sets in future. This project focused on concept shifts. Nonetheless, Figure 6 illustrates that continuous drifts can also be found by our entropy measure. Our coarse algorithm control strategy is, however, insufficient to handle continuous drifts. We are, therefore, planning to investigate more sophisticated control strategies in future work.

The chosen window size of our algorithm was experimentally chosen to cope with the signal-to-noise ratio. Alternatively, one could try to find boundary conditions such as lower and upper bounds for the window size as presented in [7] and [14].

For the sake of simplicity we limited our investigations to a batch-style version of the Naïve Bayes algorithm. We do believe, however, that there is no reason that our approach is limited to using a batch learner. On the contrary, like [11] we are convinced that an on-line learner could be easily included in our approach to improve the computational performance.

Some algorithms recognize recurring concepts and exploit this information [23, 17, 6]. While this has not been the focus of this project, any algorithm based on our entropy measure could be enhanced by comparing stored models with new data as soon as the entropy indicates the appearance of a new concept.

In this paper we illustrated our technique on a 2-class problem, but it is generalizable to n-class problems, since the entropy formula, the classifiers, and the quality measures [5] generalize accordingly.

Last but not least, we provided a real-world example to show the usefulness of this approach. In future we would like to investigate the generalizability both to other subjects (two other data sets of the same applications showed similar results) and different applications. Also the choice of the suitable parameters could be optimized.

9 Conclusion

In this paper we set out to find a measure for detecting and measuring concept shifts as an analogon for context switches. Our experimental findings show that the formulation of entropy on data streams presented in section 3
is indeed capable to detect and measure concept shifts. A simple and coarse algorithm with an entropy based instance selection strategy outperformed ensemble based algorithms on real concept shift data sets. Given our algorithms robustness towards noise, its sensitivity towards concept shifts, its computational efficiency, and predictive power on real concept shift data sets it addresses two central trade offs of current data streams mining approaches: predictive power versus computational complexity and noise versus sensitivity. As such we believe that our entropy based measure is a very promising basis to gain further insight into the problem of concept shifts, ultimately resulting in better induction algorithms for this increasingly important application domain.

10 Acknowledgments

We would like to thank Iwan Stierli and Martin Constand for their substantial support in the initial stage of this project. We also like to thank Haym Hirsh and Patrice Egger.

References

Figure 7. Overview on the real concept shift parameter (orientation angle $\Theta$ of $\vec{n}$) (A), the Entropy $H$ (B), the accuracy of the ultimate benchmark (C), its corresponding window sizes (D), the accuracy of our approach (E), and its corresponding window sizes (F) for the real drift dataset.

Figure 8. Overview on the virtual concept shift parameter (class distribution $\psi_A$) (A), the Entropy $H$ (B), the accuracy of the ultimate benchmark (C), its corresponding window sizes (D), the accuracy of our approach (E), and its corresponding window sizes for the virtual drift dataset (F).

Figure 9. The results for the entropy based and all benchmark algorithm for the real, virtual, and the real and virtual mixed concept drift data sets. The quality measure is the accuracy.
Figure 10. Overview of the real concept shift parameter (orientation angle $\Theta$ of $\vec{n}$) (A), the Entropy $\tilde{H}$ (B), the AUCs of the ultimate benchmark (C), its corresponding window sizes (D), the AUCs of our approach (E), and its corresponding window sizes (F) for the real drift dataset.

Figure 11. Overview on the virtual concept shift parameter (class distribution $\psi_x$) (A), the Entropy $\tilde{H}$ (B), the AUCs of the ultimate benchmark (C), its corresponding window sizes (D), the AUCs of our approach (E), and its corresponding window sizes for the virtual drift dataset.

Figure 12. The results for the entropy based and all benchmark algorithm for the real, virtual, and the real and virtual mixed concept drift data sets. The quality measure is the AUC.