Probabilistic Hoeffding Trees

Sped-Up Convergence and Adaption of Online Trees on Changing Data Streams

Jonathan Boidol[1,2](✉), Andreas Hapfelmeier[2], and Volker Tresp[1,2]

[1] Institute for Computer Science, Ludwig-Maximilians University, Oettingenstr. 67, 80538 München, Germany
boidol@cip.informatik.uni-muenchen.de

[2] Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 München, Germany
{andreas.hapfelmeier,volker.tresp}@siemens.com

Abstract. Increasingly, data streams are generated from a growing number of small, cheap sensors that monitor, e.g., personal activities, industrial facilities or the natural environment. In these settings, there are often rapid changes in input-to-target relations and we are concerned with tree-structured models that can rapidly adapt to these changes. Based on our new algorithms accuracy and tracking behavior is improved, which we demonstrate for a number of popular tree based-classifiers with over state-of-the-art change detection using five data sets and two different settings. The key novel idea is the representation of record values as distributions rather than point-values in the stream setting, covering a larger part of the instance space early on, and resulting in an often smaller, more flexible classification model.

Keywords: Online decision tree learning · Uncertainty-aware data streams · Classification · Concept change · Regularization

1 Introduction

Recent technological developments have immensely increased the data volume and require new analytic techniques beyond ordinary batch learning. Streaming data analysis is concerned with applications where the records are processed in non stopping streams of information. Examples include the analysis of streams of text, like in twitter, or the analysis of image streams like in flickr or the analysis of video streams. Other applications include large scale remote monitoring of environmental sensors and of industrial sensors where data rates can reach terabytes per day. Also, streams are often be subject to gradual or sudden changes in the relation between attributes (or input) and target variable, and algorithms have to adapt to these changing conditions. A gradual change is termed *concept drift*, a sudden change is called *concept change.* In this work, we will consider the case of concept changes, more specifically changes in the conditional probability $P(y|x)$ of events y given measurements x [14]. Examples are changes caused

P. Perner (Ed.): ICDM 2015, LNAI 9165, pp. 94–108, 2015.
DOI: 10.1007/978-3-319-20910-4_8

by the transition from day to night, by changes in production phases, by the introduction of new features or other sudden changes in the environment.

For these scenarios, online learners for classification have been developed that should meet the following criteria: Learning should operate iteratively, i.e. build a classification model incrementally without needing all the data before training starts. It should use every record in a single pass, i.e. look at every example only once. It should use finite resources, i.e. the algorithm's training time and space requirements should not grow with the data size. It should exhibit any-time readiness, i.e. provide the best possible classification model at any time during execution. Hoeffding Tree-Based classifiers possess most of these desired properties, and remain fairly easy to implement and analyze, and have been shown to be robust and highly scalable. In this work, we aim to improve classification on data streams that undergo concept changes to which the classifier has to react promptly.

The Hoeffding Tree is a classifier that deals with streaming data [7], also known as VFDT (Very Fast Decision Tree), upon which many state-of-the-art Online-Learners build, e.g. FIMTDD [14], CVFDT [13], VFDTc [9], iOVFDT [11], Hoeffding Option Trees [21]. VFDT and its derivatives incrementally build a decision tree and prune parts again as necessary without looking at any record more than once. Splits in the tree are introduced when sufficient examples have been seen to make a confident decision. This decision is guided by statistical bounds, e.g. the eponymous Hoeffding bound, that need only sufficient statistics of fixed size stored in the tree. The nature of these statistics varies but typically allows to calculate the best split on promising attributes. Different pruning criteria have been added to the basic algorithm to detect changes in the underlying data stream and adapt the tree accordingly. If the growth of the tree is suitably checked to avoid unlimited growth – and eventual overfitting –, the whole classifier is therefore in size independent of the size of the data stream.

More recently, methods have been developed that deal with inherent uncertainty in the data that stems e.g. from measurement errors, processing errors, technical limitations or natural fluctuations. The Uncertainty-Aware approach does not assume recorded attribute values as given, but recognizes that attribute values are representative of an underlying probability distribution. Such a situation might also arise if there are multiple measurements, say from redundant sensors in the same environment, without practical means to pick one measurement over the other if they differ. This paper shows how an Uncertainty-Aware handling of the data significantly improves accuracy and any-time-readiness in Online-Classification of changing dynamic streams.

The remainder of the paper is organized as follows. Section 2 reviews related work. Sections 3 and 4 introduce our algorithm in the context of existing decision tree algorithms. Section 5 describes the data sets, we used in our experiments, and shows the success of our algorithm compared to popular Online-Classifiers. Finally, Sect. 6 discusses the conclusions we reached based on these experiments and outlines directions for future research.

2 Related Work

In the last decade, there has been substantial research in the areas of stream processing and uncertain data. Uncertainty-aware research covers topics from clustering uncertain data [6,18,20] or outlier detection [1] to querying probabilistic databases [17]. Classification with tree models has been done by e.g. [25]. The common idea is that the expected distance between two objects is calculated with probability distributions of these objects. This work deals exclusively with static and stationary data but we borrow concepts from the research into uncertainty and apply them to data where uncertainty is not apparent but used as a tool to essentially extract more information from the data.

The earliest stream classification with iterative tree models has been developed by [7] and built upon by [9,11,13,21]. Reference [22] used an uncertainty-aware approach to improve classification models on static data, [19] used a similar approach for online stream-classification. We improve upon their work and extend the analysis to cases with time-changing data streams. To the best of our knowledge, we present the first analysis of an Uncertainty-aware classifier for data streams with concept change.

3 Online Trees for Changing Data

Our algorithm design is based on the basic Hoeffding Tree algorithm, but is in principal adaptable to any tree-like incremental learner. We introduce an elementary notation in the following section and review basic concepts of Online Decision Trees. We then present our approach PHT (Probabilistic Hoeffding Tree) as extension of those trees and outline how these changes can be implemented in an online fashion in the next section.

3.1 Online Decision Trees

Let $A_i = (a_{i,1}, \ldots, a_{i,k})$ be an instance of the data stream with k single-valued attributes where the index i notes the position in the data stream. Like all decision trees, Hoeffding Trees consist of nodes and edges (V, E) where the nodes contain tests to decide which edge to follow towards a leave of the tree. To build a Hoeffding tree, during the training phase leave nodes are recursively replaced with decision nodes. The leave nodes store statistics, decision nodes contain a split attribute. Each instance is assigned to one leaf node v after a series of tests that determine the path from the root down. These tests select the appropriate path based on the relevant split attribute of the instance in each node along the path. Thereby they determine the one branch A_i falls into and the statistics stored in leaf v are updated with the information from A_i. In some versions statistics in the nodes on the path to v are also updated. A decision to grow or prune the tree is then based on these updated statistics. They are also crucial to detect changes in the data stream and adapt the tree via pruning and regrowth

Algorithm 1. Basic Online Tree Induction

```
Input: data stream $s$ yielding records $A_i$
Output: decision tree $t$
1: procedure TreeInduction
2:     $t \leftarrow empty\,leaf$
3:     while $A_i \leftarrow next\_from(s)$ do
4:         $v \leftarrow get\_leaf(A_i)$
5:         $update(v, A_i)$
6:         $test\_and\_split(v)$
7:         $prune(t)$
8:     end while
9: end procedure
```

to the changes [2]. The pseudo code to build an incremental tree is given in Algorithm 1.

Note that we will only explicitly consider two-way splits for numeric attributes. More than two branches are possible, and common for categorical attributes, but the case for multi-way splits and discrete distributions follows easily. The tree can at any time be trained further with more instances from the stream and conversely prediction with the induced tree is possible at any point in the lifetime of the tree. Ordinarily, the prediction for a record A_i is based on whatever model is stored in the leaf to which A_i is assigned. In the simplest case this might be a single class-label or numeric value, more sophisticated versions store specific classification or regression models in the leaves.

4 Probabilistic Hoeffding Trees

The main idea in our approach is to treat records not as sets of exactly measured single values but to treat the attributes as a probability density function (PDF) centered around the recorded value instead. We call the resulting class of Hoeffding-tree algorithms PHT (Probabilistic Hoeffding Trees).

4.1 Probabilistic Records for Decision Trees

The modifications compared to the base algorithm are again given as pseudo code in Algorithm 2.

We replace the single value of a_{ij} with a PDF $p(a_{ij})$ centered around a_{ij}. For numeric attributes a uniform or Gaussian distribution are standard choices, for categorical attributes any discrete distribution specified over the possible values of a_{ij} is acceptable [5,23]. The training process is then adapted in the following way: We assume again an initial weight of 1 for every instance A_i. For every test A_i encounters in a node, e.g. $a_{ij} < t_m$, the integrals $w_l = \int_{-\infty}^{t_m} p(a_{ij})\,\mathrm{d}a_{ij}$ and $w_r = \int_{t_m}^{\infty} p(a_{ij})\,\mathrm{d}a_{ij}$ for the left and right branch are calculated. w_l and w_r simply determine, how much of the probability mass of the attribute falls in the left and right branch respectively. The values w_l and w_r are then interpreted as the weight of the branch. A_i follows every branch where w is larger than 0 simultaneously and may reach more than one leaf of the tree (cf. line 4 of Algorithm 2). The relative weight of a leaf v is then $w^{A_i,v} = \prod_{m \in M} w_{I,m}$, the product of all weights along the path to leaf v branching at nodes m, where $I \in \{l, r\}$ determines the branch taken at node m. The statistics in these leaves are then updated with the information from A_i, as in the original case, but down-weighted by $w^{A_i,m}$ (cf. line 6 of Algorithm 2). The total weight of v still sums to 1 but it promotes growth in more than a single leaf.

Algorithm 2. Incremental Uncertain Tree Induction

```
Input: data stream $s$ yielding records $A_i$
Output: decision tree $t$
 1: procedure ProbabilisticTreeInduction
 2:     $t \leftarrow empty\,leaf$
 3:     while $A_i \leftarrow next\_from(s)$ do
 4:         $L \leftarrow get\_leaves(A_i)$
 5:         for all $v \in L$ do
 6:             $update(v, A_i, rel\_weight(A_i, v))$
 7:             $test\_and\_split(v)$
 8:         end for
 9:         $prune(t)$
10:     end while
11: end procedure
```

We treat instances for prediction the same way as in training, see the modifications to the prediction process in Algorithm 3. We do not need to change the prediction model used in the tree, but we do not limit the prediction to one of those models. Our algorithm filters one record down to several leaves instead, and averages the predictions from every leaf weighted by $w^{A_i,v}$.

The voting (cf. line 9 in Algorithm 3) has the advantage of giving a distribution for the prediction from which a confidence value can be inferred, even if the base algorithm does not provide one.

In the long run in the stream setting, using a symmetric distribution and using point-values will – assuming a stationary stream – in theory converge. The advantages lie in more independence towards the order of the instances, greater flexibility during training and prediction and – as experiments will show – in the speed of the convergence towards the expected optimal tree.

4.2 Online Approximation of Density Functions

The PDFs for the attribute values have always been chosen as uniform distribution with mean equal to the original attribute point value and a standard deviation proportional to $(b - a) \times w$. Here a and b are the minimum and maximum values for the attribute that actually appear in the data set and w controls

Algorithm 3. ProbabilisticTreePrediction

```
Input: tree $t$, instance $A_i$
Output: prediction $\tilde{x}$
 1: procedure ProbabilisticTreePrediction
 2:     $L \leftarrow get\_leaves(A_i)$
 3:     $V = \emptyset$
 4:     for all $v \in L$ do
 5:         $vote \leftarrow predict(A_i, v)$
 6:         $weight \leftarrow rel\_weight(A_i, v)$
 7:         $V = V \cup (vote, weight)$
 8:     end for
 9:     $\tilde{x} \leftarrow average(V)$
10:     return $\tilde{x}$
11: end procedure
```

the width of the distribution and the 'fuzzines' of the attribute value. For the categorical attributes, the PDF has been constructed in such a way that $1 - w$ of the probability mass is placed onto the original value and the rest spread uniformly on the possible attribute values. For the synthetic data set (with numerical attributes only), a and b have been chosen so that $P(x_a \in [a, b]) \geq 0.997$ or approximately within three standard deviations of the mean.

The notation as range of values is closely related to, but here more intuitive, than the standard deviation. If the attribute range is unknown, it can be estimated from the stream for example with a number of algorithms that incrementally calculate the variance of the attribute, e.g. [16]. The ranges follow easily from the variance, for example for uniform distributions $\sigma^2 = \frac{(b-a)^2}{12}$.

Representing the PDF $p(a_{ij})$ is simple if the attribute j is categorical. Then we need only the probability for every possible value of j which has a finite and in practice usually small domain. In principal, numeric attributes could be discretized in a number of bins and treated equivalently [19]. This, however, discards the ordinality of the attribute values, forces multi-way splits and is necessarily low grained. In a simple, non-analytical solution, which has been used for example in [25], the PDF can be represented numerically by storing a set of s sample points drawn from $p(a_{ij})$ which approximates any function with a discrete distribution. Conveniently, this works equally well for numeric and categorical attributes and for all types of distributions. We chose $s = 100$ which provided a balance between approximation quality and performance in our tests.

5 Experiments

To test our algorithm we used 4 large data sets collected from sensor readings or network streams and one synthetic data set. The real data sets are all available at the UCI machine learning repository and range from 5 k to 580 k in size. While these are sufficient to gauge the algorithm behavior, we also use synthetic data to test performance in longer runs. For those experiments we used instance streams of 5 million instances. All test runs have been performed on a PC with an Intel Xeon 1.80 GHz CPU, running Linux with a 2.6.32 x86_64 kernel, and with memory limitations set to 64 MB.

5.1 Implementation

We adapted three different tree induction algorithm to PHT: Adaptive Hoeffding Trees [2], iOVFDT [11] and Hoeffding Option Trees [21] to HoeffdingAdaptiveTreePHT, iOVFDTPHT and HoeffdingOptionTreePHT. The implementation was done in the MOA framework [3], where reference implementations of the aforementioned algorithms exist and the algorithms could easily be extended.

For the evaluation of the experiments, we recorded accuracy, resulting tree size and training time measured in an interleaved test-then-train setting where every instance is first used for blind testing, and then to train the tree [2]. The standard deviation for each measure is computed over 10 repeated experiments with shuffled data sets or different initialization parameters for the synthetic data set. For the accuracy we use a fading average as described in [10] with a fading factor α of 0.99. The fading average $M_\alpha(i)$ is defined as

$$M_\alpha(i) = \frac{S_\alpha(i)}{N_\alpha(i)} \tag{1}$$

$$S_\alpha(i) = I_i + \alpha \times S_\alpha(i-1);\; S_\alpha(1) = I_1 \tag{2}$$

$$N_\alpha(i) = 1 + \alpha \times N_\alpha(i-1);\; N_\alpha(1) = 1, \tag{3}$$

where S_α is the fading sum of observations, N_α the fading increment and $I = 1$ for a correct prediction, 0 otherwise.

We report tree size in number of nodes rather than model size in bytes. The consumed memory depends not only on the implementation but also on the number and types of attributes in a data set. The number of nodes on the other hand allows an easier comparison of different tree models. We test our algorithms first on the static data sets to establish their performance and advance to time changing data streams in the following sections.

5.2 Data Sets

Robot Movement Data (RM). The RM data set is available since 2010. It contains 24 numeric attributes recorded from the a robot's sensors and four distinct classes, which determine the robot's course along a wall. The data set contains 5,456 instances [8].

Person Activity Analysis (PA). The PA data set is available since 2010. It recorded the instances collected from four sensors placed on both ankles, belt and chest of five people. Each instance has five numeric attributes, two categorical attributes and one of eleven classes. The classes distinguish human activities, e.g. walking, standing, falling, etc. The data set contains 164,860 instances [15].

Network Attack Detection (NA). The NA data set has been published for the KDD CUP 1999. It describes network connections and is used to classify normal and abnormal connections, i.e. attacks . It contains 34 numeric and seven categorical attributes like duration, error rate and protocol type. The connection types are distinguished in 23 distinct classes. We use 10 % of the full data with 494,021 instances [24].

Cover Type (CT). The CT data set is available since 1999. It collects surveillance sensor data of forestland. Each instance provides 42 categorical attributes and eleven numeric attributes like soil type, elevation, and hill shade. It distinguishes cover types in seven classes. The data set contains 581,012 instances [4].

Synthetic RBF Stream (RBF). This type of synthetic stream uses a radial basis function to generate arbitrarily large data sets. Using different initialization parameters we can create different streams, each of arbitrary length. The streams for the experiments were initiated with fixed seeds to ensure reproducibility. We set the parameters to use 50 base functions that generate 15 attributes and 4 classes and limited stream size to five million instances.

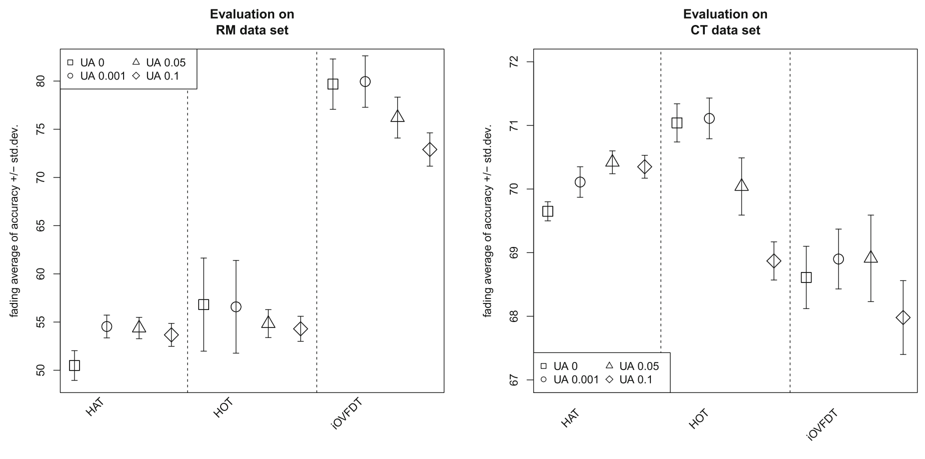
5.3 Results on Static Data

We implemented as PHT variants the following classifiers: HoeffdingAdaptive PHT, iOVFDTPHT and HoeffdingOptionPHT. We tested these on the five large data sets described in Sect. 5.2 and varied the values for the width w of the assumed distribution from 0 to 0.5. $w = 0$ means no uncertainty and is equivalent to the base classifiers our algorithms build upon. In general, we see an improvement for $w \leq 0.1$, with small to moderate (3.3 %) improvement of accuracy. Accuracy drops for larger values of w that would imply major uncertainty and are not reported.

Taking the best performing setting for each classifier and data set, we see an improvement in 10 out if 15 cases (each significant with $p < 0.1$, in a one sided t-test) in Table 1. Figure 1 shows the final accuracy for the smallest (RM) and the largest (CT) UCI data set. The fading accuracy used gives less weight to the earlier test examples, giving an overall accuracy that favors the recent predictions.

Figure 2 shows the accuracy during the lifetime of the data stream of the RBF data set. We used the RBF data set to analyze the behavior of the algorithms on much longer lived data streams and see improvement over the base classifiers, especially for the Hoeffding Adaptive Tree.

The tree size on average stays within two nodes of the base classifier, with a few exceptions on the larger data sets. There is no clear correlation between changes in model size and improved performance, with five of the eight improved models being smaller, three larger than the base classifier. Tree size does, on the other hand, decrease slightly with increasing values of w since a flatter distribution makes splits in the tree less likely.



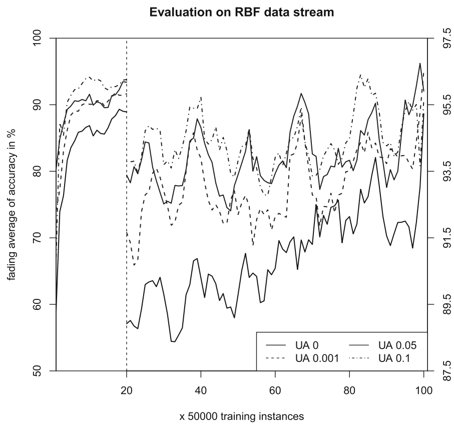
(a) RM data set with 5k instances. (b) CT data set with 580k instances.

Fig. 1. Comparison of different flavors of Hoeffding-Tree based classifiers (iOVFDT-PHT (iOVFDTPHT), HoeffdingOptionTreePHT (HOTPHT), HoeffdingAdaptiveTree (HATPHT)) on large UCI data sets. Shown here are only the smallest and the largest of the used data sets. UA gives the width-parameter of the PDF replacing the attribute values. Accuracy is the accuracy with a fading factor of 99%. The standard deviation is calculated from 10 shuffled runs.

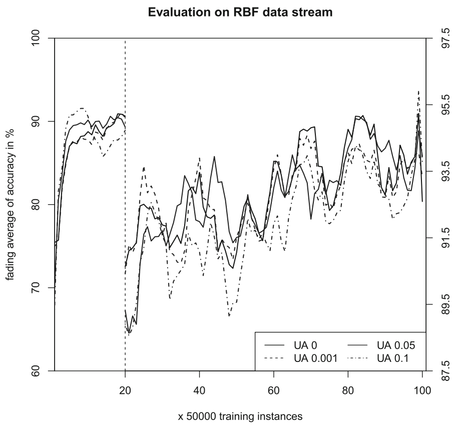
The most interesting observation here is, how even very small values for w can improve the classification without major cost to the model. Running time for the best performing models with $w \neq 0$ stays within a factor of 2 to the run time of the base classifier. Our algorithms (with $w \leq 10\,\%$) hold equal to or considerably outperform the base algorithms.

5.4 Significant Improvements During Concept Change

While stationary streams are much easier to deal with, we expect both gradual and sudden changes in real life streams. We therefore especially studied the effects of concept change, i.e. changes in the conditional probability of the classes given attribute vectors, and the improvements our algorithm achieves in such a setting. This occurs if an observed stream/the underlying system undergoes different phases in its lifetime like seasonal changes, day-night cycles in ecological systems, or different production phases in industrial machinery. Normal behavior might look completely different before and after these changes and the classification algorithm has to adapt accordingly. While HoeffdingAdaptiveTrees and HoeffdingOptionTrees have the capability to detect changes and adapt, iOVFDT does not have a mechanism to adapt to dynamic streams and is not included in this section.



(a) HoeffdingAdaptiveTreePHT on RBF data stream with 5M instances.



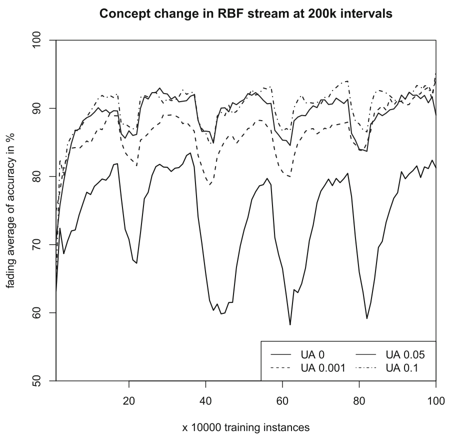
(b) HoeffdingOptionTreePHT on RBF data stream with 5M instances.

Fig. 2. Evaluation on RBF stream. The plots shows the accuracy over a the lifetime of a stream with 5 million instances exemplary for the Hoeffding Adaptive Trees and Hoeffding Option Trees. The vertical axis is enlarged from the 20 %-mark on.

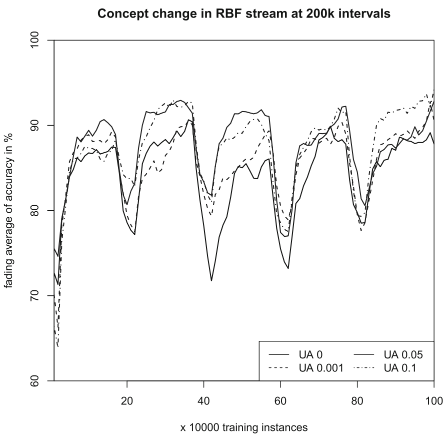
To evaluate the effect of changes for the other classifiers, we streamed each of the original UCI data sets in the stream five times in a row, but at each repetition switched the order of the numeric attributes as suggested by e.g. [12,26,27]. This permutation of attributes induces the desired concept changes and at the same time keeps the integrity of the data set compared to, say, introducing bias or noise into the data. In the RBF data set we simulated such changes by changing the parameters of the generating function. Figure 3 shows the classification accuracy during the lifetime of a RBF-stream with concept changes every 200.000 records. To compare the accuracy over the total lifetime, we averaged the accuracy over 100 sample points during the stream life time and report the results in Table 2. After every concept change, all classifiers fall back in accuracy and recover gradually, but our algorithms recovers at a much faster rate and in this setting of changing streams significantly ($p < 0.1$ in a one-sided t-test for the best-performing setting) outperforms the base classifiers. Figure 3 show how accuracy behaves before and after the concept change. For the smaller data set shown in Fig. 3(c) the break-down between change is less pronounced since the classifier has not reached a stable plateau before the concept change as in the longer-lived streams. Our algorithm improves the results of the base classifiers by up to 16 % with an average improvement of 3.2 %. We improve over HoeffdingOptionTree in 3 out of 5 data sets and over HoeffdingAdaptiveTree in 5 out of five data sets with no significant increase in model size.

Table 1. Tree size of final tree and classification accuracy on 4 UCI data sets with different width-options for the attribute PDFs. UA00 is equivalent to the base algorithm, UAmin to a PDF with 0.1 % width of the attribute range, UA05 to a width of 5 % of the attribute range, etc.

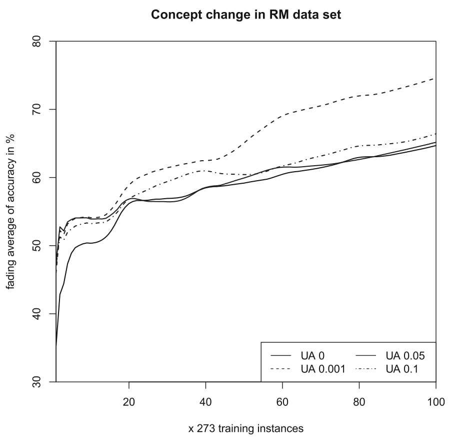
RM data set						
	Accuracy in %			Tree size in #nodes		
	HOTPHT	HATPHT	iOVFDTPHT	HOTPHT	HATPHT	iOVFDTPHT
UA00	**56.81** ± 4.83	50.49 ± 1.54	79.68 ± 2.61	0.60 ± 1.26	0.00 ± 0.00	5.90 ± 0.57
UAmin	56.58 ± 4.81	**54.54** ± 1.18	**79.95** ± 2.67	0.60 ± 1.26	0.00 ± 0.00	6.10 ± 0.74
UA05	54.84 ± 1.45	54.38 ± 1.11	76.21 ± 2.12	0.20 ± 0.63	0.00 ± 0.00	6.90 ± 0.88
UA10	54.30 ± 1.30	53.67 ±1.19	72.90 ± 1.73	0.40 ± 0.84	0.00 ± 0.00	6.80 ± 0.63
PA data set						
	Accuracy in %			Tree size in #nodes		
UA00	50.12 ± 0.29	46.84 ± 0.15	**39.43** ± 0.22	3.30 ± 0.48	4.00 ± 0.00	3.80 ± 0.42
UAmin	**50.21** ± 0.28	**48.10** ± 0.44	39.35 ± 0.34	3.20 ± 0.42	3.20 ± 0.42	3.90 ± 0.32
UA05	49.80 ± 0.36	47.90 ± 0.53	39.22 ± 0.31	3.10 ± 0.32	3.40 ± 0.52	3.90 ± 0.32
UA10	48.75 ± 0.21	47.19 ± 0.66	38.40 ± 0.43	3.00 ± 0.00	3.10 ± 0.57	3.70 ± 0.48
NA data set						
	Accuracy in %			Tree size in #nodes		
UA00	**99.70** ± 0.04	98.34 ± 0.02	**98.89** ± 0.17	4.60 ± 1.26	2.10 ± 0.32	3.80 ± 0.92
UAmin	99.38 ± 0.17	99.08 ± 0.23	98.79 ± 0.15	3.00 ± 0.00	3.70 ± 0.48	4.30 ± 0.67
UA05	99.35 ± 0.17	**99.11** ± 0.20	98.84 ± 0.15	3.20 ± 0.42	4.20 ± 0.42	3.60 ± 0.70
UA10	99.04 ± 0.15	98.75 ± 0.38	98.41 ± 0.22	6.50 ± 1.18	3.70 ± 0.67	5.20 ± 0.42
CT data set						
	Accuracy in %			Tree size in #nodes		
UA00	71.04 ± 0.30	69.65 ± 0.15	68.61 ± 0.49	10.20 ± 1.40	8.33 ± 1.12	6.30 ± 1.16
UAmin	**71.11** ± 0.32	70.11 ± 0.24	68.90 ± 0.47	10.00 ± 1.70	13.00 ± 1.94	7.30 ± 0.82
UA05	70.04 ± 0.45	**70.42** ± 0.18	**68.91** ± 0.68	8.90 ± 0.74	9.56 ± 0.73	7.60 ± 1.07
UA10	68.87 ± 0.30	70.35 ± 0.18	67.98 ± 0.58	7.80 ± 0.63	9.78 ± 1.39	7.60 ± 0.84
RBF data set						
	Accuracy in %			Tree size in #nodes		
UA00	91.73 ± 1.21	91.72 ± 0.04	**88.68** ± 5.21	22.00 ± 7.07	27.00 ± 0.00	9.50 ± 0.71
UAmin	92.14 ± 1.70	**93.33** ± 0.02	83.75 ± 4.55	22.00 ± 2.83	26.00 ± 0.00	10.00 ± 0.00
UA05	**92.89** ± 1.49	92.52 ± 0.03	84.29 ± 0.42	23.50 ± 12.02	32.00 ± 0.00	9.50 ± 0.71
UA10	92.45 ± 1.86	90.85 ± 0.07	79.62 ± 0.08	21.00 ± 7.07	26.00 ± 0.00	8.50 ± 0.71



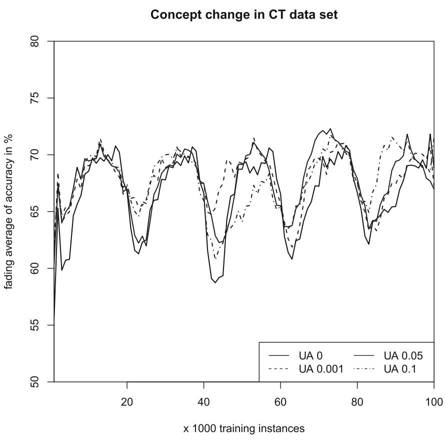
(a) HoeffdingAdaptiveTreePHT



(b) HoeffdingOptionTreePHT



(c) HoeffdingAdaptiveTreePHT



(d) HoeffdingAdaptiveTreePHT

Fig. 3. Effect of concept changes simulated with RBF stream (Figures (a) and (b)) and UCI data sets (figures (c) and (d)). UA gives the width-parameter of the PDF replacing the attribute values. Accuracy is the accuracy with a fading factor of 99 %.

Table 2. Accuracy and model size on the data sets with 4 induced concept changes. UA00 is equivalent to the base algorithm, UAmin to a PDF 0.1 % width of the attribute range, UA05 to a width of 5 % of the attribute range, etc. Accuracy is the average of the accuracies at 100 sample points during the stream existence. Model size is the number of nodes in the final tree structure.

RM data set × 5				
	Accuracy in %		Tree size in #nodes	
	HOTPHT	HATPHT	HOTPHT	HATPHT
UA00	**65.43** ± 4.07	58.65 ± 0.97	8.30 ± 1.06	5.70 ± 0.67
UAmin	65.09 ± 4.23	**67.16** ± 1.73	8.50 ± 0.97	5.40 ± 1.65
UA05	60.42 ± 0.91	62.23 ± 3.78	6.70 ± 0.82	8.70 ± 2.58
UA10	58.31 ± 0.61	63.62 ± 3.55	6.70 ± 0.67	6.50 ± 2.55
PA data set × 5				
	Accuracy in %		Tree size in #nodes	
UA00	**44.66** ± 0.89	43.11 ± 0.48	1.00 ± 0.00	1.10 ± 0.32
UAmin	44.66 ± 0.91	**45.22** ± 0.33	1.00 ± 0.00	1.40 ± 0.52
UA05	44.34 ± 0.94	45.05 ± 0.32	1.10 ± 0.32	1.20 ± 0.42
UA10	43.81 ± 0.94	44.60 ± 0.27	1.10 ± 0.32	1.20 ± 0.42
NA data set × 5				
	Accuracy in %		Tree size in #nodes	
UA00	97.65 ± 0.58	97.61 ± 0.21	3.90 ± 0.57	2.70 ± 0.48
UAmin	97.72 ± 0.56	98.31 ± 0.51	3.00 ± 0.00	1.00 ± 0.00
UA05	97.97 ± 0.46	98.63 ± 0.28	3.60 ± 0.52	1.00 ± 0.00
UA10	**97.99** ± 0.45	**98.77** ± 0.27	3.40 ± 0.52	1.00 ± 0.00
CT data set × 5				
UA00	Accuracy in %		Tree size in #nodes	
UA00	63.81 ± 1.12	65.81 ± 0.50	8.90 ± 0.57	2.50 ± 0.53
UAmin	63.67 ± 1.10	**67.45** ± 0.46	9.00 ± 0.67	3.20 ± 0.42
UA05	63.59 ± 0.95	67.23 ± 0.32	8.60 ± 1.17	3.00 ± 0.00
UA10	**64.41** ± 1.04	67.05 ± 0.52	9.10 ± 1.20	2.90 ± 0.32
RBF data set × 5				
	Accuracy in %		Tree size in #nodes	
UA00	84.70 ± 1.16	73.52 ± 2.40	12.50 ± 0.58	9.40 ± 2.70
UAmin	85.62 ± 0.69	85.99 ± 0.37	13.25 ± 1.26	9.80 ± 0.84
UA05	87.15 ± 0.41	88.61 ± 0.70	13.50 ± 1.00	10.00 ± 0.00
UA10	**87.18** ± 0.98	**89.71** ± 0.70	13.00 ± 0.82	10.20 ± 0.45

6 Conclusion and Further Work

In this paper we have shown a generic approach that extends stream classification models to incorporate the concept of uncertain data. We tested this approach on several classifiers and data sets and achieved significantly improved accuracy with comparable model size and run time across all data sets and classifiers we examined. In the case of data sets with concept change we improve accuracy by up to 16 % with 3.2 % on average. Our approach reacts swiftly to changing data streams which makes it especially suited to environments where the concept generating the streams changes periodically, as is the case in many industrial or ecological applications. Non-synthetic data where the data quality is quantified, i.e. the actual uncertainty of measured values is known appears not to be available at the moment. If such data sets become accessible, we expect much interesting results if we could substitute empirical values for the idealized PDFs. Also, we believe that the approach of uncertainty-aware data can be broadened to other types of algorithms, not limited to classification or tree-like prediction models.

References

1. Aggarwal, C.C., Philip, S.Y.: Outlier detection with uncertain data. In: SDM, pp. 483–493. SIAM (2008)
2. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009)
3. Bifet, A., Holmes, G., Kirkby, R., et al.: Moa: Massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
4. Blackard, J.A., Dean, D.J.: Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Comput. Electron. Agric. **24**(3), 131–151 (1999)
5. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 551–562. ACM (2003)
6. Cormode, G., McGregor, A.: Approximation algorithms for clustering uncertain data. In: Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 191–200. ACM (2008)
7. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80. ACM (2000)
8. Freire, A.L., Barreto, G.A., Veloso, M., et al.: Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In: 2009 6th Latin American Robotics Symposium (LARS), pp. 1–6. IEEE (2009)
9. Gama, J., Medas, P., Rodrigues, P.: Learning decision trees from dynamic data streams. In: Proceedings of the 2005 ACM Symposium on Applied Computing, pp. 573–577. ACM (2005)
10. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Mach. Learn. **90**(3), 317–346 (2013)

11. Hang, Y., Fong, S.: Stream mining dynamic data by using iOVFDT. J. Emerg. Technol. Web Intell. **5**(1), 78–86 (2013)
12. Hashemi, S., Yang, Y., Mirzamomen, Z., et al.: Adapted one-versus-all decision trees for data stream classification. IEEE Trans. Knowl. Data Eng. **21**(5), 624–637 (2009)
13. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97–106. ACM (2001)
14. Ikonomovska, E., Gama, J.: Learning model trees from data streams. In: Boulicaut, J.-F., Berthold, M.R., Horváth, T. (eds.) DS 2008. LNCS (LNAI), vol. 5255, pp. 52–63. Springer, Heidelberg (2008)
15. Kaluža, B., Mirchevska, V., Dovgan, E., Luštrek, M., Gams, M.: An agent-based approach to care in independent living. In: de Ruyter, B., Wichert, R., Keyson, D.V., Markopoulos, P., Streitz, N., Divitini, M., Georgantas, N., Mana Gomez, A. (eds.) AmI 2010. LNCS, vol. 6439, pp. 177–186. Springer, Heidelberg (2010)
16. Knuth, D.E.: The Art of Computer Programming. Seminumerical Algorithms, 3rd edn., vol. 2, p. 232. Addison-Wesley, Boston (1998)
17. Kriegel, H.P., Bernecker, T., Renz, M., et al.: Probabilistic Join Queries in Uncertain Databases (A Survey of Join Methods for uncertain data), vol. 35. Springer (2010)
18. Kriegel, H.P., Pfeifle, M.: Density-based clustering of uncertain data. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 672–677. ACM (2005)
19. Liang, C., Zhang, Y., Song, Q.: Decision tree for dynamic and uncertain data streams. In: ACML, pp. 209–224 (2010)
20. Ngai, W.K., Kao, B., Chui, C.K., et al.: Efficient clustering of uncertain data. In: Sixth International Conference on Data Mining, ICDM 2006, pp. 436–445. IEEE (2006)
21. Pfahringer, B., Holmes, G., Kirkby, R.: New options for hoeffding trees. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 90–99. Springer, Heidelberg (2007)
22. Qin, B., Xia, Y., Li, F.: DTU: a decision tree for uncertain data. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 4–15. Springer, Heidelberg (2009)
23. Singh, S., Mayfield, C., Prabhakar, S., et al.: Indexing uncertain categorical data. In: IEEE 23rd International Conference on Data Engineering, ICDE 2007, pp. 616–625. IEEE (2007)
24. Stolfo, S.J., Fan, W., Lee, W., et al.: Cost-based modeling for fraud and intrusion detection: Results from the JAM project. In: Proceedings of the DARPA Information Survivability Conference and Exposition, DISCEX 2000, vol. 2, pp. 130–144. IEEE (2000)
25. Tsang, S., Kao, B., Yip, K.Y., et al.: Decision trees for uncertain data. IEEE Trans. Knowl. Data Eng. **23**(1), 64–78 (2011)
26. Wang, P., Wang, H., Wu, X., et al.: On reducing classifier granularity in mining concept-drifting data streams. In: Fifth IEEE International Conference on Data Mining, 8-p. IEEE (2005)
27. Yang, Y., Wu, X., Zhu, X.: Combining proactive and reactive predictions for data streams. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 710–715. ACM (2005)