Detecting Concept Drift With Neural Network Model Uncertainty

Lucas Baier
IBM, Germany
lucas.baier@ibm.com

Tim Schlör
KIT, Germany
tim.schloer@gmail.com

Jakob Schöffer
KIT, Germany
jakob.schoeffer@kit.edu

Niklas Kühl
KIT, Germany
niklas.kuehl@kit.edu

Abstract

Deployed machine learning models are confronted with the problem of changing data over time, a phenomenon also called concept drift. *While existing approaches of concept drift detection already show convincing results, they require true labels as a prerequisite for successful drift detection. Especially in many real-world application scenarios—like the ones covered in this work—true labels are scarce, and their acquisition is expensive. Therefore, we introduce a new algorithm for drift detection,* Uncertainty Drift Detection (UDD), *which is able to detect drifts without access to true labels. Our approach is based on the uncertainty estimates provided by a deep neural network in combination with Monte Carlo Dropout. Structural changes over time are detected by applying the* ADWIN *technique on the uncertainty estimates, and detected drifts trigger a retraining of the prediction model. In contrast to input data-based drift detection, our approach considers the effects of the current input data on the properties of the prediction model rather than detecting change on the input data only (which can lead to unnecessary retrainings). We show that* UDD *outperforms other state-of-the-art strategies on two synthetic as well as ten real-world data sets for both regression and classification tasks.*

Keywords: Concept Drift Detection, Uncertainty, Monte Carlo Dropout, No Labels, Data Stream

1. Introduction

Across most industries, machine learning (ML) models are deployed to capture the benefits of the ever-increasing amounts of available data. When deploying models, most practitioners assume that future incoming data streams are stationary, i.e., the data generating process does not change over time. However, this assumption does not hold true for the majority of real-world applications (Aggarwal et al., 2003). In the literature, this phenomenon is referred to as *concept drift* or *dataset shift*, which usually leads to a decreasing prediction performance (Baier et al., 2019; Gama et al., 2014). Even small changes or perturbations in the distribution can cause large errors—which has been shown through, e.g., adversarial examples (Szegedy et al., 2014).

The concept drift community has developed several learning algorithms that are able to adapt incrementally (Shalev-Shwartz, 2011) or detect concept drift and trigger retrainings of a corresponding learning algorithm (Bifet and Gavalda, 2007; Gama et al., 2004). These techniques usually require full and immediate access to ground-truth labels, which is an unrealistic assumption in most real-world use cases. As an example, let us consider a manufacturing line with a manual end-of-line quality control. By collecting sensor data from all manufacturing stations and combining this information with previously acquired quality assessments (labels) of human experts, a predictive model can be built to replace the manual quality control and thus reduce repetitive and expensive human labour. However, this prediction model is likely exposed to concept drift due to, e.g., modifications in raw materials, machine wear, ageing sensors or changing indoor temperatures due to seasonal changes. A continuous stream of true labels for concept drift detection is not available in this use case—which is why traditional concept drift detection algorithms are not applicable.

To address these shortcomings, we investigate the following research question: *How can we improve concept drift detection in situations with limited* *availability of true labels?* To that end, we propose a novel concept drift detection algorithm which detects drifts based on the prediction uncertainty of a neural network at inference time, and we call this method *Uncertainty Drift Detection (UDD)*. Specifically, we derive uncertainty by applying Monte Carlo Dropout (Gal and Ghahramani, 2016). In case of a detected drift, we assume that true labels are available upon request (e.g., provided by domain experts) for retraining of the prediction model. In contrast to most drift detection algorithms, *UDD* can be used for both regression and classification problems. We evaluate *UDD* on two synthetic as well as ten real-world benchmark data sets and show that it outperforms other state-of-the-art drift detection algorithms.

2. Background and Related Work

2.1. Dataset Shift and Concept Drift

The ML and data mining communities use different terms to describe the phenomenon of changing data distributions over time and its impact on ML models (Moreno-Torres et al., 2012). *Dataset shift* (Quionero-Candela et al., 2009) is described as a change in the common probability distribution of input data x and corresponding labels y between training (tr) and test time (tst): $P_{tr}(x, y) \neq P_{tst}(x, y)$. This is similar to a common definition of *concept drift* (Gama et al., 2014): $P_{t_0}(x, y) \neq P_{t_1}(x, y)$, where t_0 and t_1 are two different points in time with $t_1 > t_0$. Note the difference regarding the indices: Dataset shift focuses on the difference between training and testing environment, whereas concept drift refers to the temporal structure of the data.

Dataset shift and concept drift can be further divided into different subcategories: *Virtual drift* (Gama et al., 2014) refers to changes in the distribution of the input data x, without affecting the distribution of labels: $P_{tr}(x) \neq P_{tst}(x)$ and $P_{tr}(y|x) = P_{tst}(y|x)$. *Real concept drift* refers to any changes in $P(y|x)$, independent of whether this change is triggered by $P(x)$ or not.

2.2. Handling Concept Drift

There are many reasons for changing data. Usually, it is intractable to measure all confounding factors—which is why those factors cannot directly be included in the ML model. Often, those factors are considered as "hidden context" of the ML models' environment (Tsymbal, 2004). Concept drift handling has been applied in a variety of different application domains such as spam detection (Gama et al., 2014) or demand prediction (Baier et al., 2021). In general, three different categories for detecting concept drift can be distinguished (Lu et al., 2018): First, error rate-based drift detection, which is also the largest group of methods (Lu et al., 2018) and aims at tracking changes in the error rate of a ML model. Popular algorithms in this category are the *Drift Detection Method* (DDM) (Gama et al., 2004), *Page-Hinkley* test (Page, 1954), and *ADaptive WINdowing* (ADWIN) (Bifet and Gavalda, 2007). Note that the error rate-based drift detection necessarily requires access to ground-truth labels. Second, data distribution-based drift detection usually applies some distance function to quantify the similarity between the distributions of a reference batch of data and the current data. Algorithms in this category work on the input data x only and do not require true labels for drift detection. Popular approaches are based on tests for distribution similarity, such as Kolomogorov-Smirnov test (Raab et al., 2020). Third, the multiple hypothesis test category detects drift by combining several methods from the previous two categories.

In many real-world applications, the assumption that all true labels are available is unrealistic (Krawczyk et al., 2017). Furthermore, the acquisition of true labels from experts (e.g., in quality control) is likely expensive. Those limitations have inspired research on handling concept drift under limited label availability. In general, methods can be distinguished based on their (non-)requirement of true labels for either drift detection or for retraining of the corresponding model: The first category of algorithms assumes that true labels are available for both drift detection and retraining, but they are only provided in limited portions at specific points in time. In this category, algorithms based on active learning have been developed, where true labels for selected samples are acquired based on a certain decision criterion (Fan et al., 2004; Žliobaitė et al., 2013). The second category requires no true labels for detection of concept drifts, but it uses them for retraining of the model in case of a drift. One approach uses confidence scores produced by support vector machines during prediction time and compares those over time (Lindstrom et al., 2013). If there is a large enough difference, the model is retrained using a limited set of current true labels. Other algorithms monitor the ratio of samples within the decision margin of a support vector machine for change detection (Sethi and Kantardzic, 2015). An incremental version of the Kolmogorov-Smirnov test has also been applied in this category (dos Reis et al., 2016). The third category handles concept drift without any label access, neither for drift detection nor for retraining, e.g., by applying ongoing self-supervised learning to the underlying classifier (Sun et al., 2020).

Note that the first category requires some true labels continuously over time in order to be able to detect a drift and trigger corresponding retraining. In contrast, the second category monitors the data stream for drifts based on the input data only and then requires true labels in case a drift has been detected. This is also the category that *UDD* belongs to. The third category can adapt without any true label knowledge. However, this category of algorithms also has the least adaption capabilities due to its limited knowledge of changes.

2.3. Uncertainty in Neural Networks

In many applications it is desirable to understand the certainty of a model's prediction. Often times, class probabilites (e.g., outputs of a softmax layer) are erroneously interpreted as a model's confidence. In fact, a model can be uncertain in its predictions even with a high softmax output for a particular class (Gal, 2016). Generally, neural networks are not good at extrapolating to unseen data (Haley and Soloway, 1992). Hence, if some unusual data is introduced to the model, the output of a softmax layer can be misleading—e.g., unjustifiably high. This likely happens in the case of concept drifts.

Generally, existing literature distinguishes two types of uncertainty: *aleatory* and *epistemic* (Der Kiureghian and Ditlevsen, 2009). The former (also called *data uncertainty*) can usually be explained by randomness in the data generation process and, e.g., corresponds to the error term in a regression setting. The latter (*statistical* or *model uncertainty*) usually results from insufficient training data. For classification tasks, uncertainty can be for instance quantified through entropy, variation ratios or mutual information (Hemmer et al., 2020).

One state-of-the-art approach to capture model uncertainty for neural networks is *Monte Carlo Dropout* (MCD) (Gal and Ghahramani, 2016). While dropout at training time has been widely used as a regularization technique to avoid overfitting (Srivastava et al., 2014), the idea of MCD is to introduce randomness in the predictions using dropout at inference time. This allows to deduce uncertainty estimates by performing multiple forward passes of a given data instance through the network and analyzing the resulting empirical distribution over the outputs or parameters.

Another family of methods to quantify predictive uncertainty is called *Deep Ensembles* (Lakshminarayanan et al., 2017). In essence, the authors of this paper propose to enhance the final layer of a neural network such that the model's output is not just a single prediction but a set of distributional parameters, e.g., the mean and variance for a Gaussian distribution. The corresponding parameters can then be fitted by using the (negative) log-likelihood as loss function. For previously unseen data, the approach suggests then to train an ensemble of several neural networks with different initializations at random. The average of all variance estimates can eventually be interpreted as model uncertainty.

Other recent approaches for quantifying uncertainty in neural networks include variational inference (Blundell et al., 2015), expectation propagation (Hernández-Lobato and Adams, 2015), evidential deep learning (Sensoy et al., 2018), some of which have been applied to areas like active learning (Hemmer et al., 2020) and others. A good overview of state-of-the-art methods for quantifying uncertainty, including an empirical comparison regarding their performance under dataset shift, is provided by Ovadia et al. (Ovadia et al., 2019).

3. Methodology

When labels are expensive and their availability is limited, popular drift detection algorithms like ADWIN, DDM and Page-Hinkley are not applicable in their original form, as these algorithms detect drifts based on a change in the prediction error rate (and therefore require true labels). As described in Section 2.2, there are different scenarios for concept drift handling with limited label availability. In this paper, we develop a novel approach which detects drifts without access to true labels—yet it requires labels for retraining the model. For detecting drifts, we rely on the uncertainty of a (deep) neural network's predictions. Previously, it has been shown that the uncertainty of a prediction model is correlated with the test error (Kendall and Gal, 2017; Roy et al., 2018). Thus, we argue that model uncertainty can be used as a proxy for the error rate and should therefore be a meaningful indicator of concept drift.

To investigate this hypothesis, we develop the following approach: For each data instance, we measure the uncertainty of the corresponding prediction issued by the neural network. Subsequently, this uncertainty value is used as input for the ADWIN change detection algorithm. We call our approach *Uncertainty Drift Detection (UDD)*. By applying *UDD*, we can detect significant changes in the mean uncertainty values over time. If a drift is detected, we require true labels for retraining of the model. Since there are methods for measuring uncertainty in both regression and classification settings, this approach allows to detect concept drifts for both learning tasks—as opposed to most other concept drift detection algorithms, which handle classification tasks only (Krawczyk et al., 2017). Note that *UDD* cannot detect any label shift where $P_{tr}(x) = P_{tst}(x)$ and $P_{tr}(y|x) \neq P_{tst}(y|x)$. However, we assume that in most real-world settings there is no label shift without any changes in the input distribution.

For drift detection without true label availability, input data-based drift detection, such as Kolmogorov-Smirnov (Raab et al., 2020), is generally also appropriate. However, considering solely input data bears the risk of detecting changes in features that may not be important for the prediction model. Specifically, it may occur that input data-based methods detect drifts where no retraining is required, because this drift will have little or no impact on the predictions of the model (e.g., virtual drift where $P_{t_0}(x) \neq P_{t_1}(x)$ and $P_{t_0}(y|x) = P_{t_1}(y|x)$). However, by using uncertainty of the underlying prediction model, we are investigating the effect of input data on properties of the prediction model rather than considering the input data only. Thus, only changes in the input data distribution relevant to the prediction model are detected. Imagine a feature in a high-dimensional feature space which is irrelevant for a neural network at inference time (e.g., low or zero weights have been assigned to this feature during training). An input data-based method will detect a significant change in this feature, even though this drift will not influence the prediction of the model due to the properties of the corresponding weights. In fact, a detected drift will lead to the acquisition of new labels at a high cost, even though no retraining is required at this point in time. *UDD*, in contrast, considers only changes in the input data that also have an impact (reflected by the uncertainty) on the prediction model.

For measuring uncertainty and computing predictions, we apply Monte Carlo Dropout (MCD) because it showed the best performance during our experiments. Furthermore, MCD has been shown to work well in a variety of different machine learning tasks (Gal, 2016) and the computational requirements are limited, which is an important factor in a stream setting. However, note that the proposed method can be easily extended to use other uncertainty estimates (e.g., Deep Ensembles) as well. In practice, MCD applies dropout at inference time with a different filter for each stochastic forward pass through the network. We denote T the number of stochastic forward passes. Predictions $\widehat{p}(y|x)$ are computed by averaging the predictions for each forward pass T given the samples w_i of model parameters from the dropout distribution and the input data x:

$$\widehat{p}(y|x) = \frac{1}{T}\sum_{i=1}^{T} p_i(y|w_i, x)\,. \tag{1}$$

Regression and classification require different methods for determining predictive uncertainty. We choose to evaluate the uncertainty for *classification* tasks based on Shannon's entropy H over all different label classes K:

$$H\left[\widehat{p}(y|x)\right] = -\sum_{k=1}^{K} \widehat{p}(y=k|x) * \log_2 \widehat{p}(y=k|x)\,. \tag{2}$$

For *regression* tasks, uncertainty estimates can be obtained by computing the variance of the empirical distribution of the T stochastic forward passes through the network (Gal and Ghahramani, 2016):

$$\widehat{\sigma}^2 = \frac{1}{T}\sum_{i=1}^{T} \left(p_i(y|w_i, x) - \widehat{p}(y|x)\right)^2\,. \tag{3}$$

For change detection, we choose ADWIN as it as able to work with any kind of real-valued input and does not require any knowledge regarding the input distribution (Bifet and Gavalda, 2007). Other drift detection algorithms such as DDM (Gama et al., 2004) or EDDM (Baena-García et al., 2006) are designed for inputs with a Binomial distribution and are therefore not applicable to uncertainty measurements (which can have different distributions by nature). Real-world data streams for concept drift handling are heterogeneous, e.g., in their number of class labels and size (Souza et al., 2020). This variability is also reflected by heterogeneous distributions of the respective uncertainty indicator. Furthermore, due to different approaches for computing uncertainty, this indicator varies significantly in scale and fluctuation between regression and classification problems. Therefore, ADWIN has to be adjusted to each data stream, which can be achieved by setting its sensitivity parameter $\alpha \in (0, 1)$: A change is detected when two sub-windows of a recent window of observations exhibit an absolute difference in means larger than α.

New data instances arrive individually and are predicted at the time of arrival. The obtained uncertainty U_t (either expressed as entropy or variance) from the prediction at time t is used as input for an ADWIN change detector. Once a drift is detected, a retraining of the prediction model is performed. For retraining, *UDD* uses the most recent data instances in addition to the original training data. This way, we can ensure that the model (a) can adapt to new concepts and (b) has enough training data for good generalization. Algorithm 1 on page 5 describes the required steps for *UDD* in a regression (U_t equals variance of prediction $\widehat{\sigma}^2$) or classification setting (U_t equals entropy of prediction H_t).

Algorithm 1 Uncertainty Drift Detection

```
1:  Input: Trained model $M$; Data stream $\mathcal{D}$; Training data $\mathcal{D}_{tr}$
2:  Output: Prediction $\widehat{y}_t$ at time $t$
3:  repeat
4:      Receive incoming instance $x_t$
5:      $\widehat{y}_t, U_t \leftarrow M$.predict($x_t$)
6:      Add $U_t$ to $ADWIN$
7:      if $ADWIN$ detects change then
8:          Acquire most recent labels $y_{recent}$
9:          $M$.train($\mathcal{D}_{tr} \cup \mathcal{D}_{recent}$)
10:     end if
11: until $\mathcal{D}$ ends
```

4. Experiments

For evaluation purposes, we conduct extensive experiments to compare *UDD* with several competitive benchmark strategies on two synthetic and ten real-world data sets. This stands in contrast to most concept drift literature, where new methods are mainly evaluated on simulated data sets with artificially induced concept drifts. The code for our experiments can be found under https://github.com/anonymous-account-research/uncertainty-drift-detection.

4.1. Experimental Setup

Throughout the experiments, for MCD, we set the number of stochastic forward passes $T = 100$ for regression tasks and $T = 50$ for classification tasks. Regarding the deep feed forward network, we vary the structure between three to five hidden layers with relu activation functions depending on the data set. Each hidden layer is followed by a dropout layer with dropout rate 0.1 or 0.2, as it is proposed in the original MCD paper (Gal and Ghahramani, 2016).

For initial model training, we use the first five percent of a data stream's instances. We perform a parameter optimization for *UDD* by requiring the associated ADWIN algorithm to detect one drift on a given validation data set—this yields a concrete value for the sensitivity parameter α. If no drifts are detected on the validation data with the initial value for α, we assume that no drifts are present in the validation data and α is set to the `scikit-multiflow` (Montiel et al., 2018) default value of 0.002. We use the ten percent of instances following the initial training data as validation data. Every time we detect a drift, we provide the last data instances as well as corresponding labels equivalent to one percent of the overall data stream's length. The exemplary partitioning of a data stream is depicted in Figure 1.

In order to benchmark *UDD*, we compare it against six different strategies within two groups. The first group of strategies handles concept drift with *Limited Label Availability* whereas the second group of strategies allows for *Unlimited Label Availability*.

4.1.1. Limited Label Availability

The first benchmark is a non-adaptive model, *No Retraining (No Retr.)*. This strategy does not test for drifts and the ML model is only trained once with the initial training set. The performance of this strategy constitutes a lower-bound benchmark.

The second benchmark is an *Uninformed Retraining (Uninf.)* strategy which randomly draws retraining points out of all possible time stamps included in the respective data stream. To ensure comparability, we set the number of retrainings of this strategy to be equal to the *UDD* approach. This also ensures that the uninformed retraining strategy receives access to the same number of true labels. Otherwise, a strategy with access to more true labels will likely perform better due to larger training set sizes. To get a reliable performance estimate for this strategy, we repeat this experiment five times and average the results.

The third benchmark, *Equal Distribution (Equal D.)*, is similar to the previous benchmark but the retraining points are equally distributed over the course of the data stream.

The *Kolmogorov-Smirnov* test-based drift detector (*KSWIN*) belongs to the category of input data-based drift detection and works by individually investigating each input feature for changes. We optimize its sensitivity parameter α with the same procedure as for *UDD*. This detector is known to produce many false positive concept drift signals, due to multiple hypothesis testing (Raab et al., 2020). Again, we restrict the number of retrainings to be equal to the *UDD* approach. If this strategy detects more drifts, detected drifts are sorted by the order of their p-values and only the top drifts are considered for retraining. For this strategy, we use the `scikit-multiflow` (Montiel et al., 2018) implementation *KSWIN* (Kolmogorov-Smirnov WINdowing) with the following parameters: $window\_size = 200$, $stat\_size = 100$.

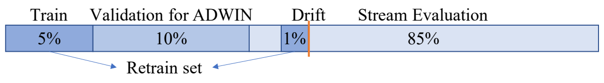


Figure 1: Partitioning of data stream.

4.1.2. Unlimited Label Availability The second group of strategies is not restricted with respect to the amount of allowed retrainings. Therefore, they are *not* an appropriate benchmark in a context where true labels are scarce. We still include these strategies since they serve as an upper-bound performance benchmark. This allows us to estimate the performance loss when confronted with a situation where full label availability is infeasible.

The *Kolmogorov-Smirnov* test with *unlimited* retrainings (*KSWIN(unl.)*) benchmark is similar to the previous *KSWIN* strategy but without restricting the number of retrainings. Therefore, all detected drifts trigger a retraining of the prediction model.

The last benchmark is the *ADWIN* change detection algorithm applied to the prediction error rate. This strategy already requires all true labels for the computation of the error rate and therefore for drift detection. Note that all other strategies manage the drift detection without any true labels and then only require labels for retraining. For this method, we use the `scikit-multiflow` (Montiel et al., 2018) implementation with default parameter settings.

4.2. Data Sets

For evaluation, we consider two synthetic data sets (Friedman and Mixed) and ten real-world data sets. All data sets are widely used in concept drift research and are therefore suitable for evaluating *UDD*. The **Friedman** regression data set (Friedman, 1991) consists of ten features that are each drawn from a uniform distribution from the interval $[0, 1]$. The first five features are relevant for the prediction task, the remaining five are noise. The **Mixed** classification data set is inspired by (Gama et al., 2004) and contains six features where two features are Boolean and the other four features are drawn from a discrete distribution. Two of the features are noise which do not influence the classification function. By modifying the distribution of some features, we can either induce real or virtual concept drifts (see Section 2.1) in both the Friedman and the Mixed data set.

Furthermore, ten real-world data sets—eight classification and two regression tasks—are used for the evaluation of the *UDD* method. The **Air Quality** data set (De Vito et al., 2008) contains measurements from five metal oxide chemical sensors, a temperature, and a humidity sensor. The learning task is to predict the benzene concentration, which is a proxy for air pollution. Concept drift is present due to seasonal weather changes. The **Bike Sharing** data set (Fanaee-T and Gama, 2013) provides hourly rental data for a bike sharing system in Washington, D.C., with the objective to predict the hourly demand for bike rentals. Concept drift is again assumed to be present due to seasonal weather changes.

All following classification data sets are taken from the USP Data Stream Repository (Souza et al., 2020): The various **Insects** data sets were gathered by controlled experiments on the use of optical sensors to classify six types of different flying insects. Concept drift is artificially induced by changes in temperature. The **Abrupt** data set contains five sudden changes in temperature, whereas in the Incremental (**Inc**) data set, temperature is slowly increased over time. The Incremental Abrupt (**IncAbr**) data set has three cycles of incremental changes with additional abrupt drifts included as well. In the Incremental Reoccurring (**IncReo**) data set, the temperature increases incrementally within several cycles. The **KDDCUP99** data set contains TCP connection records from a local area network. The learning task is to recognize whether the connection is normal or relates to one of 22 different types of attacks. The **Gas Sensor** data set contains records where one of six gases is diluted in synthetic dry air, and the objective is to identify the respective gas. Both sensor drift (due to aging) and concept drift (due to external alterations) are included in the data. The **Electricity** data set was gathered at the Australian New South Wales Electricity Market. The learning task is to predict whether the market price will increase or decrease compared to the last 24 hours. The **Rialto Bridge** Timelapse data set contains images taken by a webcam close to the Rialto Bridge in Venice, Italy. The objective is to correctly classify nearby buildings with concept drift occurring due to changing weather and lighting conditions.

4.3. Performance Metrics

Evaluating concept drift detection on real-world data sets is a challenging endeavor as most real-world data sets do not have specified drift points. Specifically, for most real-world data, it is intractable to measure the accuracy of drift detection itself. Therefore, we perform two different analyses regarding the behaviour of *UDD* in this work. First, we apply *UDD* on two synthetic data sets to specifically evaluate its drift detection capabilities. Second, we perform extensive experiments to investigate its performance on ten real-world data sets.

For synthetic data sets, the real drift points are known, which allows to compute metrics regarding the drift detection capabilities of a drift detector (Bifet et al., 2013). In this work, we compute the Mean Time to

Detection (MTD), the False Alarm Count (FAC), and the missed detection count (MDC).

In contrast, an appropriate evaluation for real-world data sets is more difficult. However, one can assume that a drift is present when the prediction performance of a static model decreases over time. Since the real drift points are unknown, we evaluate the different strategies based on their prediction performance, as it is common in the concept drift literature (Elwell and Polikar, 2011). For regression tasks, we apply the Root Mean Squared Error (RMSE), and we use the Matthews Correlation Coefficient (MCC) for classification tasks. MCC is a popular metric for classification settings as it can also handle data sets with class imbalance (Chicco and Jurman, 2020).

4.4. Analysis on Synthetic Data Sets

To test the capabilities of *UDD*, we analyze its behaviour when applied on two synthetic data sets (Friedman and Mixed). Both data sets contain virtual as well as real concept drifts. Virtual drifts refer to changes in the input data with no effect on the resulting label. Hence, *UDD* should *not* raise an alarm for these drifts as a retraining of the ML model in this case is unnecessary. Recall that this kind of analysis is only feasible on synthetic data sets, as we do not have any knowledge regarding the type of concept drift as well as its timing on real-world data sets. On the synthetic data sets, we test *UDD* and *KSWIN(unl.)* as they both do not require true labels for drift detection. The parameters of both approaches are optimized based on a validation set which includes one drift (see Section 4.1).

Figure 2 shows the trajectory of the predictive uncertainty over the course of the Friedman data set. The uncertainty changes significantly each time a real concept drift occurs. Accordingly, this is also detected by *UDD*. As expected, the two virtual drifts (marked by orange vertical lines in the figure) do not trigger a drift detection. In contrast, the input data-based detection (*KSWIN*) detects also these virtual drifts. Furthermore, note the overall large number (20) of detected drifts by *KSWIN* despite a parameter optimization. This illustrates *KSWIN*'s problem of high reactivity leading to several false-positive drift detections.

Table 1: Evaluation on synthetic data sets.

Data Set	Drift Detector	MTD	FAC	MDC
Friedman	*UDD*	132.7	0	0
	KSWIN	65.7	17	0
Mixed	*UDD*	247.3	0	0
	KSWIN	50.3	11	0

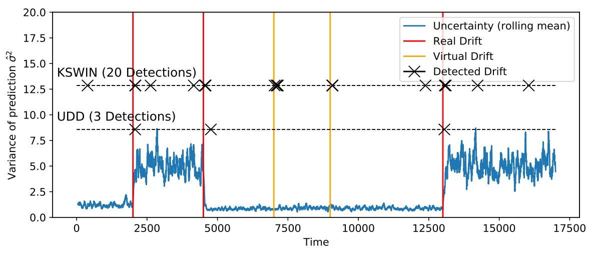


Figure 2: Behaviour of *UDD* and *KSWIN* on synthetic Friedman data set.

Since the real drift points for the synthetic data sets are known, we compute the mean time to detection (MTD), the false alarm count (FAC), and the missed detection count (MDC) for both strategies in Table 1. *UDD* correctly identifies all real concept drifts in both data sets. Furthermore, no false alarms are raised. However, *KSWIN* achieves lower MTD values compared to *UDD* in both data sets, which means that *KSWIN* recognizes concept drifts faster. This can likely be explained by the high sensitivity of *KSWIN* regarding changes. However, this sensitivity also leads to large numbers of false alarms (17 and 11, respectively), as depicted in Table 1. Such a behaviour is especially detrimental in scenarios where the acquisition of true labels is expensive. Each time a false alarm is raised, new true labels must be acquired at a high cost—even though a retraining is not required since no real concept drift has occurred.

4.5. Experimental Results

Both *UDD* and *KSWIN* require as input a suitable value for α, which determines their sensitivity regarding concept drift detection. Since the data sets included in this experiment are fundamentally different from each other (e.g., different number of class labels), individual values of α are required for each data set. As described in Section 4.1, we determine the respective value for both strategies by performing a test on a validation data set.

A summary of the experimental results on all data sets is provided in Table 2 for regression data sets (RMSE) and in Table 3 for classification (MCC). For the evaluation, we primarily focus on the first five columns of the table which as a group can be characterized by only requiring a limited amount of true labels. This is also illustrated by the values in parentheses which describe how often the corresponding ML models are retrained. As explained in Section 4.1, *KSWIN(unl.)* and *ADWIN* serve as an upper-bound benchmark due to their requirement of full label availability.

Table 2: RMSE (the lower the better) on *regression* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark.

		Limited Label Avail.				**Unlimited Label Avail.**	
Data Set	**No Retr.**	**Uninf.**	**Equal D.**	**KSWIN**	**UDD**	***KSWIN(unl.)***	***ADWIN***
Air Quality	1.170 (0)	1.383 (14)	1.231 (14)	1.285 (14)	**1.151** (14)	1.304 (19)	1.387 (12)
Bike Sharing	171.47 (0)	170.00 (5)	144.94 (5)	143.88 (5)	**129.93** (5)	120.69 (27)	127.07 (8)

Table 3: MCC (the higher the better) on *classification* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark.

		Limited Label Avail.				**Unlimited Label Avail.**	
Data Set	**No Retr.**	**Uninf.**	**Equal D.**	**KSWIN**	**UDD**	***KSWIN(unl.)***	***ADWIN***
Insects Abrupt	0.452 (0)	0.468 (9)	0.475 (9)	0.456 (9)	**0.516** (9)	0.521 (192)	0.497 (9)
Insects Inc	0.052 (0)	0.210 (4)	0.211 (4)	0.191 (4)	**0.242** (4)	0.238 (27)	0.251 (3)
Insects IncAbr	0.292 (0)	0.463 (22)	0.483 (22)	0.464 (22)	**0.522** (22)	0.488 (107)	0.516 (23)
Insects IncReo	0.114 (0)	0.190 (10)	0.197 (10)	0.126 (10)	**0.208** (10)	0.218 (149)	0.239 (13)
KDDCUP99	0.663 (0)	0.830 (20)	0.873 (20)	0.772 (20)	**0.964** (20)	0.986 (345)	0.984 (61)
Gas Sensor	0.255 (0)	0.472 (39)	0.469 (39)	0.325 (39)	**0.484** (39)	0.454 (149)	0.480 (49)
Electricity	0.139 (0)	0.372 (13)	0.362 (13)	0.254 (13)	**0.436** (13)	0.511 (269)	0.471 (45)
Rialto Bridge	0.534 (0)	0.558 (14)	0.561 (14)	**0.583** (14)	**0.583** (14)	0.586 (17)	0.600 (116)

The best strategy with limited label availability per data set is marked in bold. For both regression data sets, *UDD* outperforms the other four strategies. Regarding the classification tasks, *UDD* achieves the best prediction performance on seven out of eight data sets and always outperforms the strategies *No Retraining*, *Uninformed* and *Equal Distribution*. Solely for the Rialto data set, the strategy based on *KSWIN* performs equally well, which might be explained with rather significant changes in individual input features that can be detected well with *KSWIN*. As expected, the *No Retraining* strategy usually performs worst. This finding clearly illustrates the presence of concept drift in all of the selected real-world data sets even though the exact drift points are not measurable. Interestingly, the *Uninformed* already achieves good prediction performance and sometimes even outperforms the *KSWIN* strategy, especially for the regression tasks. By design, the number of retrainings is equal for all four strategies—*Uninformed*, *Equal Distribution*, *KSWIN*, and *UDD*.

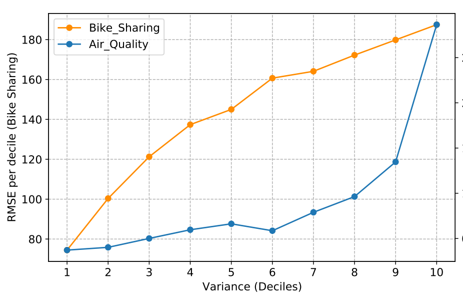
The right two columns in both Table 2 and Table 3 show the prediction performance of the *KSWIN(unl.)* and *ADWIN* strategy. As expected, these strategies usually outperform all other strategies but also require significantly more true labels for retraining. For the KDDCUP99 data set, the difference in amounts of retrainings for *UDD* compared to *KSWIN(unl.)* is most striking: While *UDD* requires 16 retraining, *KSWIN(unl.)* performs 345 retrainings in total. Yet, the difference in predictive performance is rather small. Also, recall that the *ADWIN* strategy requires *all* true labels for drift detection itself. For the Insects Abrupt, Insects IncAbr, and the Gas Sensor data set, the *UDD* strategy performs even better than *ADWIN*.

We also investigate the average prediction performance for *UDD* based on the level of uncertainty in Figure 3. Per data set, we sort instances in deciles, from instances with lowest uncertainty (decile 1) up to instances with highest uncertainty (decile 10) based on entropy H or variance $\widehat{\sigma}^2$, respectively. Subsequently, we compute the average prediction performance per decile. As expected, the RMSE for regression data sets increases with rising uncertainty, as shown in the left plot (a). The right plot (b) shows the classification data sets—decile 1 shows the highest mean accuracy and decile 10 the lowest.[1] Thus, Figure 3 confirms our assumption that uncertainty represents a proxy for the error metric.

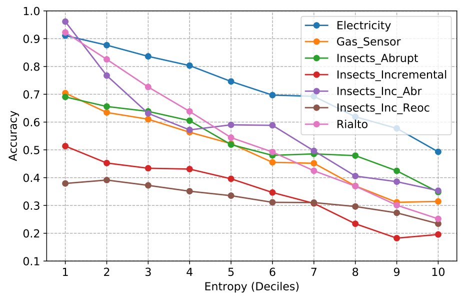
5. Conclusion

In this work, we have introduced the *Uncertainty Drift Detection (UDD)* algorithm for concept drift detection. As stated in the research question, this algorithm is also suitable for situations with limited availability of true labels since it does not depend on true labels for detection of concept drift. Only in case of

[1]KDDCUP99 data set is not included in Figure 3 because deciles cannot be computed due to the skewed entropy distribution.



(a) Regression



(b) Classification

Figure 3: Relationship between deciles of uncertainty and prediction performance.

a detected drift, it requires access to a limited set of true labels for retraining of the prediction model. Therefore, this algorithm is especially suitable for drift handling in deployed ML settings within real-world environments where the acquisition of true labels is expensive (e.g., quality control). Standard drift detection algorithms such as DDM and ADWIN are not applicable in such settings because they require access to the entire set of true labels. Our approach is based on the uncertainties derived from a deep neural network in combination with Monte Carlo Dropout. Drifts are detected by applying the ADWIN change detector on the stream of uncertainty values over time. In contrast to most existing drift detection algorithms, our approach is able to detect drift in both regression and classification settings. We have performed an extensive evaluation on two synthetic as well as ten real-world concept drift data sets to demonstrate the effectiveness of *UDD* for concept drift handling in comparison to other state-of-the-art strategies.

However, more evaluation of *UDD* in various use cases with different data sets is required to prove its overall effectiveness. Additionally, *UDD* can only be applied successfully for use cases where concept drift can be observed in the input data—as opposed to drift in the labels alone. In future work, we aim to improve the *UDD* method by including active learning methods. Including only those instances with high uncertainty in the retraining set rather than all recent instances could further improve the prediction performance. Furthermore, we also want to analyze which type and magnitude of concept drift can best be handled by applying *UDD*.

References

Aggarwal, C. C., Philip, S. Y., Han, J., & Wang, J. (2003). A framework for clustering evolving data streams. *VLDB*.

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method. *Knowledge Discovery from Data Streams*.

Baier, L., Kellner, V., Kühl, N., & Satzger, G. (2021). Switching scheme: A novel approach for handling incremental concept drift in real-world data sets. *HICSS*.

Baier, L., Kühl, N., & Satzger, G. (2019). How to cope with change? Preserving validity of predictive services over time. *HICSS*.

Bifet, A., & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. *SIAM SDM*.

Bifet, A., Read, J., Pfahringer, B., Holmes, G., & Žliobaitė, I. (2013). CD-MOA: Change detection framework for massive online analysis. *IDA*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *ICML*.

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 1–13.

De Vito, S., Massera, E., Piga, M., Martinotto, L., & Di Francia, G. (2008). On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*.

Der Kiureghian, A., & Ditlevsen, O. (2009). Aleatory or epistemic? Does it matter? *Structural Safety*.

dos Reis, D. M., Flach, P., Matwin, S., & Batista, G. (2016). Fast unsupervised online drift detection using incremental Kolmogorov-Smirnov test. *KDD*.

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*.

Fan, W., Huang, Y.-a., Wang, H., & Yu, P. S. (2004). Active mining of data streams. *SIAM SDM*.

Fanaee-T, H., & Gama, J. (2013). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 1–67.

Gal, Y. (2016). Uncertainty in deep learning. *University of Cambridge*.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *ICML*.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. *Brazilian Symposium on Artificial Intelligence*, 286–295.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*.

Haley, P. J., & Soloway, D. (1992). Extrapolation limitations of multilayer feedforward neural networks. *IJCNN*.

Hemmer, P., Kühl, N., & Schöffer, J. (2020). DEAL: Deep evidential active learning for image classification. *arXiv:2007.11344*.

Hernández-Lobato, J. M., & Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. *ICML*.

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? *NeurIPS*.

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

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*.

Lindstrom, P., Mac Namee, B., & Delany, S. J. (2013). Drift detection using uncertainty distribution divergence. *Evolving Systems*.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., et al. (2018). Learning under concept drift: A review. *IEEE Trans. on Knowledge and Data Eng.*

Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *JMLR*.

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., & Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., et al. (2019). Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *NeurIPS*.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2009). *Dataset shift in machine learning*. The MIT Press.

Raab, C., Heusinger, M., & Schleif, F.-M. (2020). Reactive soft prototype computing for concept drift streams. *Neurocomputing*.

Roy, A. G., Conjeti, S., Navab, N., & Wachinger, C. (2018). Inherent brain segmentation quality control from fully convnet Monte Carlo sampling. *MICCAI*.

Sensoy, M., Kaplan, L., & Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. *NeurIPS*.

Sethi, T. S., & Kantardzic, M. M. (2015). Don't pay for validation: Detecting drifts from unlabeled data using margin density. *INNS*.

Shalev-Shwartz, S. (2011). Online learning and online convex optimization. *Foundations and Trends in Machine Learning*.

Souza, V. M. A., dos Reis, D. M., Maletzke, A., & Batista, G. E. A. P. A. (2020). Challenges in benchmarking stream learning algorithms with real-world data. *DMKD*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., et al. (2020). Test-time training with self-supervision for generalization under distribution shifts. *ICML*.

Szegedy, C., Zaremba, W., Sutskever, I., Estrach, J. B., Erhan, D., et al. (2014). Intriguing properties of neural networks. *ICLR*.

Tsymbal, A. (2004). The problem of concept drift: Definitions and related work. *Trinity College Dublin*.

Žliobaitė, I., Bifet, A., Pfahringer, B., & Holmes, G. (2013). Active learning with drifting streaming data. *IEEE Trans. Neural Networks Learn. Syst.*