Detecting Concept Drift in Deployed Machine Learning Models

<How well do Margin Density-based concept drift detectors identify concept drift in case of synthetic/real-world data?>

<Baptiste Andre[1]>

Supervisors: Jan Rellermeyer[1], Lorena Poenaru-Olaru[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 29, 2023

Name of the student: <Baptiste Andre>
Final project course: CSE3000 Research Project
Thesis committee: Jan Rellermeyer, Lorena Poenaru-Olaru, Jesse Krijthe

Abstract

When deployed in production, machine learning models sometimes lose accuracy over time due to a change in the distribution of the incoming data, which results in the model not reflecting reality any longer. A concept drift is this loss of accuracy over time. Drift detectors are algorithms used to detect such drifts. Drift detectors are important as they allow us to detect when a classification model becomes inaccurate. Some possible uses of drift detectors can even go as far as detecting adversarial attacks on machine learning algorithms.[10] The detectors discussed in this paper are Margin Density drift detectors. Their evaluation is made within an unsupervised context, where we assume no testing labels are available. In real world applications of machine learning models, this might often be the case, as finding labels is costly. Experiments in this paper have found that margin density detectors can be useful tools in detecting the first drift for synthetic data, even though parameter tuning must be done to achieve high accuracy for some datasets. In an unsupervised environment with more than one drift, the drift detectors are unreliable as was seen in experiments involving real world data. With this paper comes an implementation of margin density detectors for future endeavors.

1 Introduction

Concept Drift

A limitation of machine learning models deployed in production is that their classification accuracy sometimes decreases over time. This is because the data they are classifying changes over time, and stops reflecting the data they were training on. This naturally occurring change in the data over time is called a concept drift. A drift detector is what is used to detect such a shift in data[7]. The inability to detect drift in data could lead to relying on an inaccurate classification machine learning model. Being unable to detect concept drift can also be seen as a security threat, as concept drifts can be attacks on the machine learning model. [10] This paper will focus on a type of unsupervised drift detector.

Unsupervised drift detectors are drift detectors that only rely on the labels for the training sets. Unsupervised drift detectors are especially important because it is expensive to get labels for data in real world situations. In this paper, a margin density detectors will be evaluated in an unsupervised environment.

Research Question

This paper will be a general as well as a comparative study of multiple drift detectors within the margin density detector category. Being able to apply the best drift detection techniques to real world data is important to have good accuracy and performance. Therefore, the overarching question that this paper partially attempts to answer is this one: How well do unsupervised concept drift detectors identify concept drift in case of synthetic and real-world data? My sub question will focus on Margin Density detectors, and will therefore be: How well do Margin Density-based concept drift detectors identify concept drift in case of synthetic/real-world data?

The paper will be divided into these following sections:

1. Related works, where this paper will describe the relevant literature pertaining to unsupervised drift detection and margin density detectors.
2. The Methodology section where this paper will describe how drifts were found in the real world data. It will also describe implementation details of the experiment.
3. The Result section where relevant results are presented.
4. The Discussion section, where relevant results will extensively be analyzed.
5. A Responsible research section dedicated to ethical matters and considerations.
6. A Conclusion which will summarize the findings of this paper.

The contributions of this paper are summarized below:

1. It describes a methodology and with if finds real world drift in certain real world datasets.
2. It evaluates margin density detectors under the same experimental setup and gives insight into their strengths and limitations.
3. It offers an implementation of multiple margin drift detectors found in multiple papers. This will allow further research using margin density detectors which to the best of our knowledge do not yet have publicly available implemented code. [7]
4. It experiments with a K nearest neighbors classifier based margin drift detector which has not been done before, to the best of our knowledge.

This paper presents a replicative and comparative study of margin density detectors and its contributions are therefore mostly experimental. It describes concretely how different drift detectors from the same category(margin density) compare to each other. The setup of this project is suitable due to the fact that it assembles multiple different classifiers and compares them under the same experimental setup. It also evaluates the detectors on real world data in an unsupervised environment, which to the best of our knowledge, has never been done before. To find how margin density detectors compare to mixed, clustering similarities, and Data-distribution based detectors, a meta analysis might be done outside the scope of this paper.

2 Related work

2.1 Unsupervised Drift Detectors

Unsupervised drift detection is a relatively unexplored subfield of drift detection [4]. Most existing work describes novel techniques or algorithms, and sometimes how they relate to specific industry practices (which is outside of our scope). Some examples are the MD3 algorithm, [8], or the NN-DVI algorithm.[3]. There is also overview literature [1] that assembles multiple different approaches to detecting concept drift with unsupervised algorithms. Even with overviews, there are no comprehensive analysis of how multiple types of unsupervised concept drift detectors compare to each other when it comes to their performance on the same data sets. Our broad project seeks to bridge that gap in knowledge by regrouping many detectors and directly comparing them to each other. Once again, this paper is only a part of the project, and will focus on margin density detectors specifically. Other papers will describe mixed, clustering similarities, and Data-distribution based detectors.

2.2 Margin Density Detectors

Margin density detectors were first introduced as detectors using the margins of soft margin support vector machines to estimate drift. Margin density is defined as $Margindensity = Dm \div Dt$ where Dm is the number of data points inside the boundaries and Dt is the total number of data points. A diagram of margin density with soft margin support vector machines can be seen in figure 1. The reason such a metric is used is that for support vector machines, significant movement of data within the margin should reflect a change in accuracy, as the data is close to the decision boundary. This assumes a coupling of the training model with the drift detector.[8]. We could however also make an arguement that detectors are similar and will reflect drifts in the same places, which is found to be true in this paper's experimental section. The detectors described below are the ones implemented for this paper, and are based, if somewhat loosely on these principles. The margin density detectors mentioned in this paper all use the chunk based algorithm (uses batches of data instead of a stream) approach without the forgetting factor λ,[8] as none of the experiments are made with sliding windows.

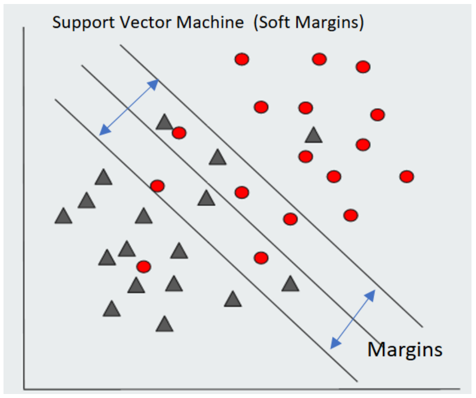

Figure 1: Margin Density of a soft margin SVM

MD3V1

The first algorithm implemented for this paper is based on the first margin density detector invented [8] (at least by name). The algorithm starts by training a support vector machine with the training data and labels. The margin density (as described in section 2.2)of the training data set is recorded, and is set to be the maximum, and minimum seen. Once the detector is trained, it can detect whether a batch of data contains a drift. It does this by comparing the maximum - minimum density seen to the newly found batch density. If it is greater than a parameter ρ (to be chosen) away, we define it as a drift. Having such a parameter unfortunately makes this drift detector very dataset dependent, as will be seen later.

MD3V2

The second detector was implemented as an improved version of the first one. It takes into account the standard deviation of the margin density of the training data [9]. This allows for a statistical approach in choosing a threshold for defining the drift, which reduces dataset dependence. This algorithm uses K cross validation on support vector machines to estimate the expected value and the standard-deviation of the accuracy and margin density. Once those values are found, a support vector machine is trained on the labeled training data to be able to estimate the margin density, and consequently drift of the data it will be analyzing. When deciding whether there is a drift in a given dataset; the detector compares the expected margin density to the margin density of said dataset, and if the difference of those is larger a threshold θ standard deviation away(depending on how sensitive to change in margin density), a drift is suspected. Since the definition of drift relates to a drop in accuracy, the algorithm then needs the labels to calculate how far from the expected accuracy the calculated accuracy is. A drift is confirmed when the accuracy is farther than an arbitrary amount of standard deviations away. Something that one might notice is that we can only use this drift detector with support vector machines, which is not ideal, especially if conceptually, the coupling of the detector and classifier is somewhat important. What if the data classification is more accurate with other classifiers?

MD3_X

The third algorithm[9] used in this paper seeks expand the definition of margin density to allow margin density drift detectors to be made with ensemble classifiers as well as support vector machines. The margin density defined for ensemble classifiers is an uncertainty metric. The probability for each class is calculated as the amount of classifiers in the ensemble that have voted for a certain class given a data point divided by the total number of classifiers in the ensemble. If the probability the data is in class 1 - the probability that it is in class 2 is bigger than a tunable parameter θm(margin of uncertainty),then the data point is within the margin. The margin density or "blindspot density", is the total number of points within this margin divided by the total number of points. Replacing the measure of margin, one can now use algorithm for MD3_V2, with bagging classifiers instead of support vector machines.

Fuzzy margin Density(FMD)

There are other variants to calculate the margin density. The algorithm used to detect drift stays identical, but the function to calculate the margin density is different. One of those papers describes the fuzzy margin density, for example[11]. The fuzzy density algorithm uses the third algorithm defined in this paper, but changes how one calculates the blindspot density. It seeks to improve the previous detector. The new equation is described in Figure 2, where x is the data point, y is the class, and Prs is the probability that x is in class y.

$$\varphi = P_{RS}(y = -1|x) - P_{RS}(y = +1|x)$$

$$\mu_E(x_i) = \begin{cases} \frac{1}{2}(\cos\pi|\varphi| + 1), & |\varphi| \in [0, 0.5] \\ 0, & |\varphi| \in (0.5, 1] \end{cases}$$

Figure 2: Fuzzy Margin Density [11]

3 Methodology

3.1 Implementation of concept drift detectors

The implementation of the algorithms was made to be consistent with the way they might have been implemented in the papers. This lead to the choice of the scikit-learn [6] library, which was also used by the authors of the algorithms this paper will implement. The programming language used was python, version 3.9. This version was used because it is the highest version able to run the environment required for the scikit-learn-intelex library at the time. This library allows a substantially faster runtime for many classifiers of the classic sklearn library. The popular library numpy was used for data handling outside of the classifiers. This includes mostly calculation of margin density, and an implementation of cross validation[9].

All algorithms implemented in this paper use the classifiers provided by the sklearn library. MD3V1,and MD3V2 use the SVM classifier with a linear kernel. The $MD3_X$ and FMD algorithms were implemented using the class BaggingClassifier with a feature sampling of .5. The experiments were then done with a CART tree classifier, and a KNN classifier. The Tree classifier using a CART algorithms was used even though the described tree algorithm in the paper is a C4.5[9]. No C4.5 tree algorithms could be found with implementations that had appropriate run times for our experiments. The choice of the sklearn CART algorithm was made due to its runtime speed[5] and consistency with the rest of our code. This is a limitation of this paper; when it comes to replicating implementations of referenced algorithms fully for experimentation. This trade-off was also done with the airplanes dataset. The class LinearSVC was used with a Nystroem kernel approximation instead of a simple linear SVM. This was done to greatly increase processing speed, at the cost of some classifying accuracy.

To tune how accurately a drift detector can detect drift on a given dataset, many hyperparameters have to be considered. For brevity's sake, only the experimental results of the recommended parameters[8][9] [11]will be displayed for each dataset, with some clear exceptions. Separate data and experiments regarding tuning parameters of the detectors will only be mentioned. For MD3V1, the parameter $\rho = .075$, and $C = 1$ for the trained SVM. [9]. For the MD3V2 algorithm, $\theta = 2$ (standard deviation multiplier),$K = 5$ cross validations, with SVM hyperparameter $C = 1$ for consistency with MD3V1. FDM and MD3_X implemented ensemble classifiers with feature bagging with half of the features present, and 20 classifiers each. [11][9]. The KNN classifier was made with an optimized K (from 4-40) maximizing the accuracy using cross validation. All tables shown will be from implemented algorithms that use the aforementioned tune-able parameters.

3.2 Description of datasets

Synthetic Data

There were three distinct synthetic sets of data sets and distribution used. SEA, AGRAW 1, AGRAW 2 as descried in another paper [7]. All of them represent a distribution that changes over time in an attempt to make classifiers lose accuracy. An important feature of those datasets is that they only contain one drift. SEA has purely numerical data, AGRAW 1 and 2 have both categorical and numerical data. Each of those sets had one abrupt drift version of themselves with many gradual drift versions of themselves each having a different drift width. The definition of the drift width is: the moment between the start and end of a drift [7]. The drift widths considered for each data sets are 500, 1000,5000,10000,20000. In each of those sub sets are 100000 data points. They are to be seen as continuous in time with data point zero being the first data point and data point 100000 being the last received data point. The first 30000 points are to be used as labeled training data, and the subsequent data should be used as unlabeled testing data.

3.3 Data setup

Pre-processing:

Before the data was able to create and be processed by the the drift detectors, it had to be pre-processed to be usable. The libraries used were pandas; for the data frames, and numpy, for the encoders and scalers. Since, the sklearn implementation for SVMs does not support categorical data, it data had to be changed to numerical form. To do this, we used three different encoding methods and compared their performances. ”Target”, ”One Hot”, and ”Ordinal” encodings were used. The results displayed for all data tables shown in the next section are derived from ordinally encoded experiments. All three of these encoding methods will be discussed in the next section. Before being used by the classifiers, the data had to be scaled, as the SVM classifier had a poor runtime and classification performance without scaling. The loss of performance was due to the fact that the scale of the features affects how much a feature is weighted, since the SVM algorithm maximizes distance between separating hyperplanes.[9] The dependency on the scale of the features was negated by normalizing them with a minmax scaler. This scaler was taken from the sklearn library, and was used for all experiments described later.

Synthetic Data

All synthetic data had similar characteristics and had to be processed in similar ways. The data was split into 10 testing batches of 10 thousand data points. The first 30 thousand data points were used to train the drift detectors. Once they were trained, the drift detectors were tested successively on the remaining batches. In each of the data sets, the drift occurred in batch 3 (after the training data). This setup can be visualized in figure 3

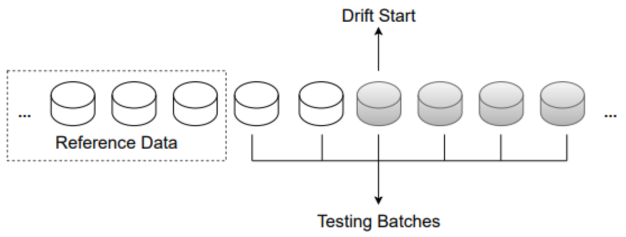

Figure 3: Data Setup Synthetic Data [7]

Real World Data

The Weather dataset was divided into a training set of 6053 samples and a set to be detected of 12106 samples. Two experiments were done with the batches to be detected. Those 12106 samples were divided monthly, with 30 samples per patch, and the other experiment had them divided yearly, with 365 samples per batch. It describes meteorological data from Nebraska, with the purpose of the classifier being to predict rain.

The ELECT2 dataset was divided into a training set of 15104 samples and a set to be detected of 30208 samples. Those 30208 samples were then further divided into yearly predictions with 365 samples per batch. This dataset represents data about energy prices and relevant information, with the purpose of the classifier being to predict increasing or decreasing electricity prices.

The Airplanes dataset was divided into a training set of 179794 samples and a set to be detected of 359589 samples. Those 359589 samples were then further divided to create batches of 17000. This dataset contains flight related data and information, with the purpose of the classifier being to predict if a flight is delayed or not.

Unlike in the synthetic data, the batches where the drift occurred was unknown in the real world data, which meant that the drift had to be defined from the characteristics the data exhibited. Since a drift is defined as the loss in accuracy over time in the context of a classifier, the approach taken was to find such loss in accuracy. Ideally, the most accurate and least biased classifier would have been used to find such a loss in accuracy, and our method tried to replicate such conditions. Practically it would still be possible for a non ideal classifier classifier to detect an accuracy loss, so we assumed some error margin. To estimate accuracy, a cross validation approach was used on a few classifiers to maximize performance. The classifier with the highest performance was then used to estimate the drifts. The classifiers used were:

- K nearest Neighbour(KNN)
- Support Vector Machine (SVM)
- CART Random Forest
- Gaussian Process Classifier
- Ada Boost Classifier
- GaussianNB
- QuadraticDiscriminantAnalysis

Method used to find real world drifts:

F irstly, the chosen real world dataset was divided into subsets according to their characteristics mentioned above to allow a good estimate of classification accuracy. An example would be the ELECT2 dataset.This dataset was divided into a training set of 15104 samples and a set to be detected of 30208 samples. Those 30208 samples were then further divided into yearly predictions with 365 data points per batch.

Secondly, a classifier was trained on the training set. This was done by dividing the training set into batches that are the same size as the ones for the testing set. In this case, the training set was divided into batches of 365 data points. Randomly, a batch within the training set was chosen to serve as a testing batch. The rest of the data was used to train the classifier. The classifier was then run on the randomly chosen batch to find the accuracy at which the classifier predicts that batch. This procedure was repeated with 39 more random batches. Those results were then used to create a normal distribution with expected classifier accuracy μ, and standard deviation σ. The decision to use 40 random samples was made to have a sufficient sample size for the central limit theorem to hold. This procedure was repeated until each classifier had a normal distribution.

Lastly, The classifier with the highest expected accuracy was chosen to predict the the classes of the data in the sequential batches, in the set to be detected. The accuracy at which the classifier predicted the classes of the batches was then compared to μ, with a batch being labeled as a drift if the accuracy of prediction of the classifier on that batch was lower than a standard deviation away. One standard deviation was chosen as only 16 percent of the data should be that far below the mean, meaning that the chosen batch is most likely drifting. Figure 4 shows the batch classification of the classifier with the highest expected accuracy for ELECT2. The classifier is an ADA Boost Classifier with an expected accuracy of .856, and a standard deviation of about .054. The bar going through the graph represent the accuracy threshold determining the classification of the batch into a drifting or non drifting batch. This procedure was done multiple times for ELECT2 for confirmation.

3.4 Evaluation metrics

Synthetic Data

The evaluation metrics used are ”False positive rate, and ”Latency” [7]. False positive rate calculates the amount of times a drift detector detects a drift before a drift happens, and Latency describes how late it detects the drift, compared to when a drift does happen. The exact equations are shown in figure 5

Real World Data

For real world data, the metrics used are the drift accuracy, and the false positive rate (FPR). The drift accuracy is a measure of how many drifts the detector is able to find. The exact equation is $Driftscorrectlyfound \div TotalDrifts$. The FPR is calculated as $NM \div NT$ where NM is the amount of non drifting batches that were classified as drifting, and NT is the total non drifting batches.

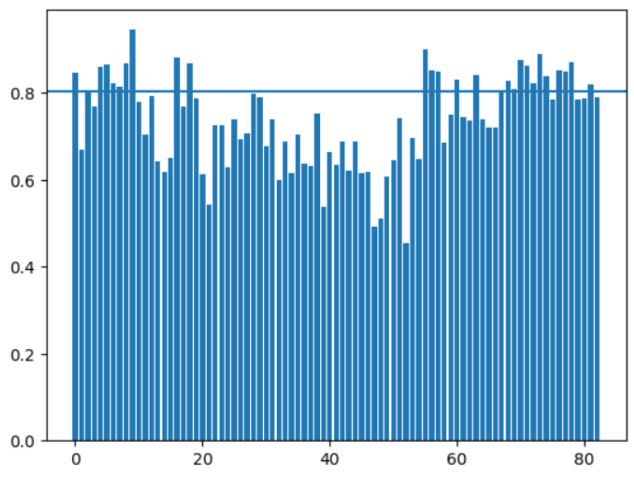

Figure 4: Accuracy of ADA Boost Classifier on Elec2 dataset

$$FPR = \frac{k^F}{|B_{ND}|}; b_k^F \in B, \qquad L = \frac{k-j}{|B|}; b_j, b_k \in B,$$

Figure 5: Data Setup Synthetic Data [7]

4 Results

4.1 Synthetic data

Should contain all the results we obtained from our detectors with different parameter sets (e.g. different encodings):

Abrupt drift

Before describing the experimental results, some context needs to be given to its interpretation. Margin density detectors use the blindspot or margin density of their detectors to try and detect a change in accuracy. Below are some graphs showing the shape of margin density graphs for the synthetic datasets. What are the properties of margin density when a drift is detected?

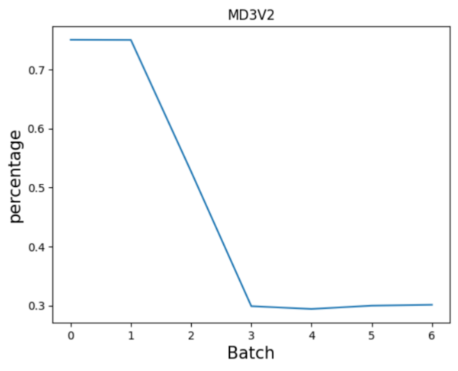

Figure 6: Accuracy of Abrupt Agraw1

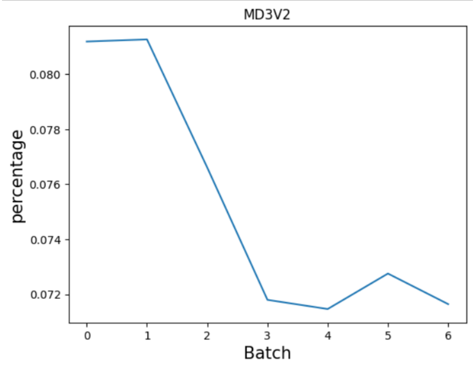

Figure 7: Margin Density of Abrupt Agraw1

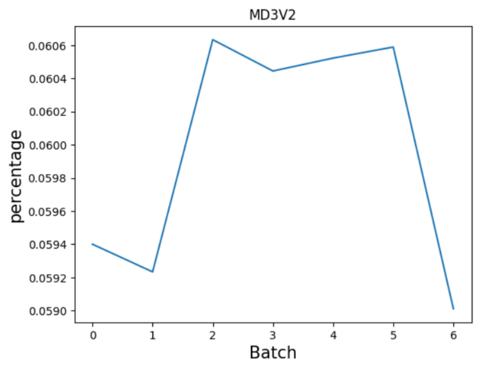

Figure 8: Margin Density of Abrupt Agraw2

These figures exemplify well the shapes of both accuracies and margin densities. All datasets broadly have that shape, with gradual drift accuracies having a bit more slope. From these figures one can find three important properties, which can be generalized to all other datasets present in this paper. Firstly, margin density will often have a rapid change when a drift occurs. Secondly, the margin density can either have an increase or decrease when a drift occurs. Thirdly, after the first drift, a change in margin density might not represent a change in accuracy as well as for the first drift, as can be seen in figure 8.

For the experiments displayed in the next section, Ordinal encoding was used. Target Encoding and One hot encoding were also tried on the abrupt drift datasets, as well as the gradual drift datasets with width of 20 thousand data points. The drift detecting capabilities of the detectors with one hot encoding and target encoding were almost identical. The only difference being that the latency for the agraw2_20 became 0.2 for MD3V2, meaning the detector performed worse. Experments with the other encoders were discontinued after those experiments, in favor of the ordinal encoder for the rest of the synthetic data.

Table 1: FPR Synthetic

	sea A	agraw1 A	agraw2 A
MD3_V1	0	0	0
MD3_V2	1	0	0
MD3_Tree	1	1	1
MD3_KN	0	0	0
Fuzzy_Tree	1	1	1
Fuzzy_KN	0	0	1

Table 2: Latency Synthetic

	sea A	agraw1 A	agraw2 A
MD3_V1	0.25	0.25	1.0
MD3_V2	0.0	0.0	0.0
MD3_Tree	0.0	0.0	0.0
MD3_KN	1.0	1.0	1.0
Fuzzy_Tree	0.0	0.0	0.0
Fuzzy_KN	1.0	1.0	0.0

As we can see, on recommended parameters, the MD3V2 algorithm performs the best on the agraw datasets while the MD3V1 algorithm performs best on the Sea datasets. After performing some experiments, it was found that by changing the parameter ρ for MD3V1, it could be perfectly tuned to detect a drift drifts for Agraw datasets with no false positives or latency. Changing θ for MD3V2, achieved the same effect. Increasing both of those parameters would increase Latency, and reduce the FPR. Drifts detection of both SEA and Agraw datasets could not be made simultaneously with one drift detector. Different values of K up to 20 for cross validation were used to detect drift more accurately. This did not change the results.

The MD3_Tree algorithm consistently had a high FPR, and a low latency. Upon closer inspection, it was found that the standard deviation and mean density of the detector were both 0. The fuzzy counterpart had the same problem. The blind spot density graphs represented drifts in accuracy well, but the detector was unable to take advantage of it.

The MD3_KN either had high Latency or high FPR. Upon further investigation, it was found that the expected value for the detector was either significantly higher than the batch blind spot density, or significantly lower, and its standard deviation was abnormally high. We attempted to find parameters that would allow accurate drift detection. Experiments were made increasing K neighbors to 200, then 1000, and the cross validation amount was increased to 50, but the results did not change significantly.

Gradual drift

Table 3: FPR Gradual

	M_1	M_2	M_T	M_KN	F_T	F_KN
sea A	0	1	1	0	1	0
agraw1 A	0	0	1	0	1	0
agraw2 A	0	0	1	0	1	1
agraw1_05	0	0	1	0	1	0
agraw1_1	0	0	1	0	1	1
agraw1_5	0	0	1	1	1	0
agraw1_10	0	0	1	0	1	0
agraw1_20	0	0	1	0	1	0
agraw2_05	0	0	1	0	1	0
agraw2_1	0	0	1	0	1	0
agraw2_5	0	0	1	0	1	0
agraw2_10	0	0	1	0	1	0
agraw2_20	0	0	1	1	1	1
sea_05	0	1	1	0	1	1
sea_1	0	1	1	0	1	0
sea_5	0	1	1	0	1	0
sea_10	0	1	1	0	1	0
sea_20	0	1	1	0	1	1

Table 4: Latency Gradual

	M_1	M_2	M_T	M_KN	F_T	F_KN
sea A	0.2	0.0	0.0	1.0	0.0	1.0
agraw1 A	0.2	0.0	0.0	1.0	0.0	1.0
agraw2 A	1.0	0.0	0.0	1.0	0.0	0.0
agraw1_05	0.2	0.0	0.0	1.0	0.0	1.0
agraw1_1	0.2	0.0	0.0	1.0	0.0	0.0
agraw1_5	0.2	0.0	0.0	0.0	0.0	1.0
agraw1_10	0.2	0.0	0.0	1.0	0.0	1.0
agraw1_20	0.4	0.0	0.2	1.0	0.0	1.0
agraw2_05	1.0	0.0	0.0	1.0	0.0	1.0
agraw2_1	1.0	0.0	0.0	1.0	0.0	1.0
agraw2_5	1.0	0.0	0.0	1.0	0.0	1.0
agraw2_10	1.0	0.0	0.0	1.0	0.0	1.0
agraw2_20	1.0	0.0	0.0	0.0	0.0	0.0
sea_05	0.2	0.0	0.0	1.0	0.0	0.0
sea_1	0.2	0.0	0.0	1.0	0.0	1.0
sea_5	0.2	0.0	0.0	1.0	0.0	1.0
sea_10	0.2	0.2	0.0	0.2	0.0	1.0
sea_20	0.4	0.2	0.0	1.0	0.0	0.0

The results for gradual drifts show that an increase drift width will increase the latency for the SVM based detectors.

4.2 Real-world data

Weather:

Table 5: FPR Weather

	MV1	MV2	M_T	M_KN	F_Tree	F_KN
Monthly	0.996	1.0	1.0	0.864	1.0	0.783
Yearly	0.933	1.0	1.0	0.800	1.0	0.933

Table 6: Accuracy Weather

	MV1	MV2	M_T	M_KN	F_Tree	F_KN
Monthly	1.0	1.000	1.0	0.842	1.0	0.828
Yearly	1.0	0.888	1.0	0.833	1.0	1.000

The drift detection capabilities of margin density drift detectors is relatively low for the weather data set, with a high false positive rate detected. Experiments with the values of the parameters $\theta = 4$ were tested in an attempt to tune the detector. The experiments with the tuned detectors had a lower false positive rate, but also a lower drift detection accuracy.

Electricity:

Table 7: Electric Dataset

	Elec Acc	Elec FPR
MD3_V1	1.000000	0.962963
MD3_V2	0.892857	0.962963
MD3_Tree	1.000000	1
MD3_KN	1.000000	1
Fuzzy_Tree	0.982143	1
Fuzzy_KN	1.000000	1

The drift detection capabilities of margin density drift detectors is relatively low for the electricity data set, with a high false positive rate detected. The same experiments with $\theta = 4$ were made, with the same results as with the weather dataset. Changing the amount of cross validations yielded similar results.

Airplanes:

Table 8: Airlines

	Air Accuracy	Air FPR
MD3V1	0.0	0
MD3V2	0.0	0
MD3_Tree	1.0	1
MD3KNN	1.0	1
Fuzzy_Tree	1.0	1
FuzzyKNN	0.0	0

The drift detection capabilities of the margin density detectors is relatively low for the airplanes dataset, with a low accuracy, or a high false positive rate.

5 Discussion

5.1 Synthetic Data

Performance under abrupt drift

As shown in the result section, the MD3_V1 algorithm accurately detected drift in only some of the dataset. The dataset which was accurately predicted, however, depended on the tuning parameters used by the algorithm. This shows that this detector can have high accuracy on detecting drift when there is only one drift, and only in some circumstances. The main issue one faces when using this detector is that it needs to be tuned in a data dependent way. Further researh could be done to tune the detectors without requiring knowledge about the incoming drift. To our knowledge, this is not something that has been done yet[8],[9],[11] [10]. Currently, the only way to tune the detector for an accurate drift detection is to experiment with a known drift, which defeats the purpose of the detector. This is also true for the MD3_V2 algorithm

The MD3_Tree and its fuzzy counterpart were unable to deliver any useful results. The margin density graphs resembled the ones from MD3_V2, but the standard deviation and expectation for the blindspot density is equal to 0. Our hypothesis is that the CART trees are over-fitted on the training data when it comes to blindspot density, since they use feature bagging. This happens in the datasets used for this paper because those datasets only have a restricted amount of non linearly dependent features, and they might therefore not create much "disagreement" in the ensemble classifier. In future research, one could investigate the effects of increased number of linearly independent features and bootstrap aggregation on the drift detection accuracy of Tree based blindspot density algorithms.

The MD3_KN algorithm algorithm and its fuzzy counterparts also have expressive margin density graphs which show drifts. The standard deviation and accuracy calculated by the cross validation for each dataset is however very different from what it should be to give meaningful results. The cause of this is unknown. This detector has been tested with multiple tunings, as was described in the result section. Presently, this drift detector is highly unreliable.

The difference in drift detection accuracy between the fuzzy and original MD3 algorithms were not made clear with the experimental results. This is partly due to the inability of blindspot density based detectors to detect drift accurately.

Performance under gradual drift

The results from experiments on gradual drifts show that overall, the performance of the drift detectors doesn't change much when compared to abrupt drifts. The latency of the drift detection does increase, but experiments have also shown that increasing the sensitivity of the detectors by decreasing the ρ and θ value of the MD3 algorithms would make the detectors more sensitive to change, therefore allowing them to detect gradual drifts earlier. No significant change is to be reported for MD_X and FMD detectors.

5.2 Real world data

As shown in the experiments, the drift detecting performance of all the detectors is poor on the given real world data. Even though the accuracy of the detectors is high, the false positive rate is also high. Tuning the values ρ and θ showed a decrease in FPR, but they also showed a decrease of drift detection accuracy. A viable hypothesis about the low accuracy of drift detection under this setup is that the measure for margin density is not suitable to predict accuracy, but only to predict an accuracy change for the first drift. As mentioned or shown in the experimental results, margin density graphs showed change when there was an accuracy change, a margin density change could be found in the same batch. The experimental results also demonstrated that the margin density change could be either positive or negative when there was an accuracy drop. Furthermore, they showed that after the first drift, a large margin density change could happen without being reflected in the accuracy classifying accuracy. This last observation can be explained by the fact that margin density is reliant on the original distribution of the data. If a drift were to happen, the subsequent changes of margin density would not reflect the changes in data points around the decision boundary any longer. They would instead reflect the density of random space in the new data distribution, which might not have any relation to the change in accuracy of the classifier. This makes the margin density metric mostly unreliable after the first drift if one were to attempt accuracy change predictions. This is why in the original papers[8],[9], the detector is re-trained after a drift is found, so that it can reflect the new distribution. The algorithm therefore works best as a semi-supervised drift detector.

6 Responsible Research

6.1 Ethical Concerns

This study did not use any methods that would place it under ethical scrutiny. All sources used are published papers, which are cited below in my reference page. The synthetic data-sets have been created by our supervisor, from whom we have the authorisation of use. The real world data is taken from open source data sets, and has the authorization to be used in research. Additionally, it does not violate anyone's privacy under the GDPR rules. All the libraries used are open source. The impact of this research is also not in question, as it is simply a replicative and comparative study of already existing drift detectors. One may debate on the ethics of how drift detectors might be used, but that is out of the scope of responsibility of this paper.

All principles from the Netherlands code of conduct for research are also followed [2]. All results were reported and explained accurately in their context. The methodology was made as close as possible to the papers it originated from, and any difference was explained. All code and data are publicly available, with a simple API for further research or validation. This research was made in an academic context and independently, and was therefore not influenced by any non-scholarly consideration.

6.2 Methodology and Replication

A main limitation of the study from our supervisor's paper was that it lacked implementations for drift detectors.[7] Our project's papers not only make a comparative analysis, but they seek to solve this problem by implementing those drift detectors. This means that it will not only allow the reproducibility of our papers, but will help with replicating results from other papers that use unsupervised drift detectors. All code will be made available, with parameters used, and all libraries are well known, reliable, easy to access, and do not require any special hardware, except maybe for some values, which will anyway have been made public in this paper. Apart from results achieved from classifiers involving randomness like KNN, the results given in this paper are extremely reproducible, as all instructions are also given in both the related works and the methodology. Not only that, but the code and functions are available in a user-friendly way, which allows for further research building on our implemented drift detectors. The datasets are described and will also be made available.

7 Conclusion

7.1 How well do Margin Density-based concept drift detectors identify concept drift in case of synthetic/real-world data?

As fully unsupervised training detectors, The MD3V1, and MD3V2 detectors detect drifts relatively well on synthetic data for abrupt and gradual drift given that only the first drift must be detected. The out of the box implementation works for some datasets only. For other datasets, the margin density parameters must be tuned, which can thus far not be done without knowing information about the drift ahead of time. This is true for abrupt and gradual drifts. This is a limitation of the Margin density detectors which could be addressed in further research. The MD3_X and FMD detectors performed quite poorly on the synthetic data. The exact cause is not known. For the CART ensemble tree classifier version, it is hypothesized that over-fitting and the dimensionality of the data might play a role. Other limitations of margin density detectors when it comes to detecting drift can be found in their respective authors' papers.[8][9] [11] As fully unsupervised training detectors, margin drift detectors do not accurately predict accuracy drop over time for real world data. This is most likely due to the fact that margin density detector were made to be re-trained every time a drift is detected, and that margin density is ultimately not a good metric to predict accuracy. Given experimental results, margin density seems to only predict a change in accuracy of the distribution associated with its classifier. Further research could be done to find how well margin density detectors predict the first drift in real world data. All code for the implementations of the drift detectors will be made available, which resolves the current limitation of unsupervised drift detectors being unimplemented.[7]

8 Acknowledgements

This paper is focused on margin density drift detectors. It is part of a meta comparative project focusing on regrouping most known unsupervised drift detectors with the goal of comparing them. This project is coordinated by PhD student Lorena Poenaru-Olaru and is under the supervision of Professor Jan Rellermeyer.

Other members are mentioned below

Jindrich Pohl, Toma Zamifirescu, Konsta Kanniainen

References

[1] Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda dos Santos. An overview of unsupervised drift detection methods. *WIREs Data Mining and Knowledge Discovery*, 10(6):e1381, 2020.

[2] Knaw, Nfu, Nwo, T. O. federatie, Vereniging Hogescholen, and Vsnu. Nederlandse gedragscode wetenschappelijke integriteit, 2018/9/30 2018.

[3] Anjin Liu, Jie Lu, Feng Liu, and Guangquan Zhang. Accumulating regional density dissimilarity for concept drift detection in data streams. *Pattern Recognition*, 76:256–272, 2018.

[4] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019.

[5] Khin Myint and Dr Hlaing Htake Khaung Tin. *Analyzing the Comparison of C4.5, CART and C5.0 Algorithms on Heart Disease Dataset using Decision Tree Method*. 2021.

[6] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830, nov 2011.

[7] Lorena Poenaru-Olaru, Luis Cruz, Arie van Deursen, and Jan S. Rellermeyer. Are concept drift detectors reliable alarming systems? - a comparative study. In *7th Workshop on Real-time Stream Analytics, Stream Mining, CER/CEP Stream Data Management in Big Data*, 2022.

[8] Tegjyot Singh Sethi and Mehmed Kantardzic. Don't pay for validation: Detecting drifts from unlabeled data using margin density. *Procedia Computer Science*, 53:103–112, 2015.

[9] Tegjyot Singh Sethi and Mehmed Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99, 2017.

[10] Tegjyot Singh Sethi and Mehmed Kantardzic. Handling adversarial concept drift in streaming data. *Expert Systems with Applications*, 97:18–40, 2018.

[11] Jing Yang, Jie Zhang, and Sujuan Qin. A concept drift detection algorithm based on fuzzy marginal density, 2020.