Unsupervised Concept Drift Detection with a Discriminative Classifier

Ömer Gözüaçık
Bilkent Information Retrieval Group
Bilkent University
omer.gozuacik@bilkent.edu.tr

Alican Büyükçakır
Bilkent Information Retrieval Group
Bilkent University
alicanbuyukcakir@bilkent.edu.tr

Hamed Bonab
Center for Intelligent Information Retrieval
UMass Amherst
bonab@cs.umass.edu

Fazli Can
Bilkent Information Retrieval Group
Bilkent University
canf@cs.bilkent.edu.tr

ABSTRACT

In data stream mining, one of the biggest challenges is to develop algorithms that deal with the changing data. As data evolve over time, static models become outdated. This phenomenon is called concept drift, and it is investigated extensively in the literature. Detecting and subsequently adapting to concept drifts yield more robust and better performing models. In this study, we present an unsupervised method called D3 which uses a discriminative classifier with a sliding window to detect concept drift by monitoring changes in the feature space. It is a simple method that can be used along with any existing classifier that does not intrinsically have a drift adaptation mechanism. We experiment on the most prevalent concept drift detectors using 8 datasets. The results demonstrate that D3 outperforms the baselines, yielding models with higher performances on both real-world and synthetic datasets.

KEYWORDS

Data stream; concept drift; drift detection

ACM Reference Format:
Ömer Gözüaçık, Alican Büyükçakır, Hamed Bonab, and Fazli Can. 2019. Unsupervised Concept Drift Detection with a Discriminative Classifier. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358144

1 INTRODUCTION

Data stream mining has became a major research topic due to the increasing amount of data generated by various sources such as social networks, online businesses, and military-financial applications [8]. These data are mostly wasted because of computational and memory-related limitations. They need to be processed immediately otherwise they are lost due to the volatile nature of the streaming environment [13].

In traditional classification tasks, the main assumption is that data is static which indicates both train and test environments are from the same distribution. However, this assumption is often violated as real-world applications are dynamic and evolve over time. This is known as concept drift [11]. In such cases, the classification model becomes obsolete as the data distribution changes and the predictive function can no longer correctly map features to labels, hence deteriorating the predictive performance of models.

Concept drift detection is studied under two main categories: explicit and implicit [20]. Explicit (supervised) detection techniques require labels to be available just after data arrives, whereas implicit (unsupervised) methods do not. For most cases, explicit methods track the performance of the classifier over time. When there is a significant drop, a drift is flagged. On the other hand, implicit methods investigate the properties of the features. In many data streaming environments, labels are not available all the time. For some cases, only a percentage of them are present, or they arrive with delay [20].

In this paper, we propose an unsupervised method using a discriminative classifier over a sliding window to monitor the change in the distribution of data. A classifier is trained and tested periodically aiming to distinguish whether the new samples are from a similar distribution as the old. In a batch setting, a related problem is *covariate shift adaptation* [19] where training and test distributions differ. A method that uses a discriminative classifier similar to the proposed approach is introduced to detect and correct covariate shift [3]. In a stream setting, to the best of our knowledge, this is the first method that utilizes a discriminative classifier for the *concept drift detection* task.

The main contributions of this paper are as follows. We **(1)** introduce a simple yet effective method for unsupervised concept drift detection; **(2)** empirically test the proposed method on 8 datasets against widely used concept drift detection methods; **(3)** evaluate the performance of our approach and demonstrate that it outperforms the baselines, achieving the highest average rank.

2 PROBLEM DEFINITION

Stream classification is a supervised learning problem that takes place in data streams, under time and memory constraints [1]. A data stream consists of data instances that arrive in time order, i.e. $\mathcal{D} = \{(X_0, y_0), (X_1, y_1), \dots (X_t, y_t), \dots\}$ where X_t represents features, and y_t classes associated with the instance at time t. Class

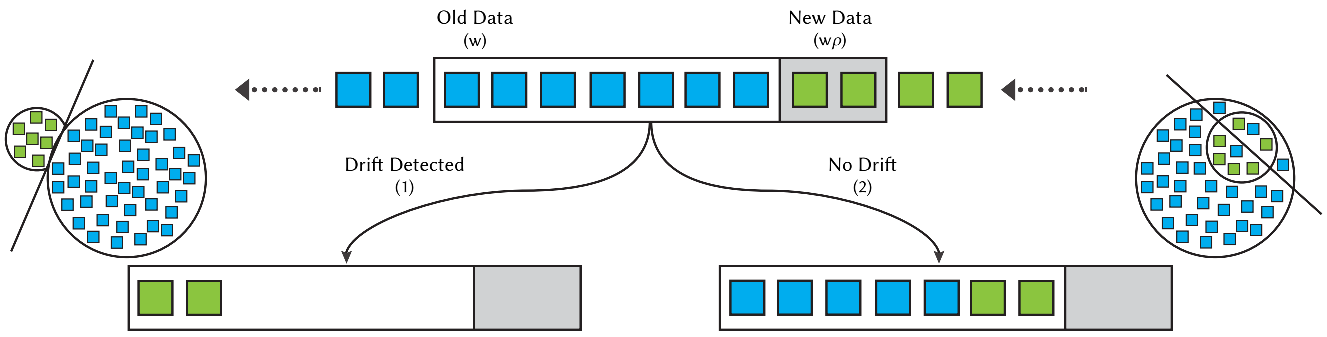


Figure 1: Drift detection workflow: (1): Drift detected. The old and the new data are separable. Samples from the old portion are discarded and partially filled with the samples from the new. (2): No drift. These sets are nested. The oldest $w\rho$ samples are removed and the window is shifted to left where the samples from the new fill the space that becomes empty.

information for a data instance, y_t, is available only after testing, i.e. predicting from X_t. A concept at a given time t is denoted as $p_t(X, y)$. In such an environment, concept drift detection is a task of determining whether the joint distribution of inputs X and targets y differ between times t_0 and t_1 [11].

$$p_{t_0}(X, y) \neq p_{t_1}(X, y) \quad (1)$$

The primary objective for concept drift detection is to design a method that runs jointly with a classification model, modifying the model in accordance with the detected drift to provide robustness against changes in the data distribution, thus improving the classification performance of the model.

3 RELATED WORK

Concept drifts happening on both $p(X)$ and $p(y|X)$ simultaneously are identified as Rigorous Concept Drift [21]. Most implicit detection techniques focus on concept drift assuming that the changes in $p(X)$ lead to changes in $P(y|X)$, deteriorating the predictive performance. There are two approaches for detecting these type of drifts: novelty detection and multivariate distribution monitoring [20]. Methods based on novelty detection are referred to as clustering or outlier detection which use distance or density measures to detect unknown patterns [16].

Approaches based on multivariate distribution monitoring track the distribution of features individually by identifying each as a stream. They make tests batch by batch: storing the older batch's information as a reference, and inspect the changes in the newer batch. KL-divergence, Hellinger distance, and correlation are commonly used for measuring the differences [6, 14]. A drift is signalled if there is a significant change in the average.

Page-Hinckley Test [18], DDM [10], EDDM [2] and ADWIN [4] are the most prevalent methods for concept drift detection. They are explicit methods relying on labeled data. The Page-Hinckley Test tracks the cumulative difference between the observed values and their mean of the chosen metric. It detects a change when the difference is more than a given threshold. DDM monitors the probability of error rate and its standard deviation. It signals a drift if the error rate within a period increases substantially. EDDM tracks the distance in between two back-to-back errors. It assumes this distance to be larger if there is no drift and vice versa otherwise. DDM and EDDM follow the ideas based on control charts and work one sample at a time. Unlike previous methods, ADWIN operates with a sliding window whose size is adaptive with respect to the changes in data. There are two sub-windows storing the performance metrics for older and new data. A drift is detected if the mean of these sub-windows differ by more than a certain threshold.

4 PROPOSED APPROACH: D3

We propose D3 (**D**iscriminative **D**rift **D**etector), an unsupervised drift detection method which uses a discriminative classifier that can be used with any online algorithm without a built-in drift detector. We hold a fixed size sliding window of the latest data having two sets: the old and the new. A simple classifier is trained to distinguish these sets. We detect a drift with respect to classifier performance. This process is done repeatedly as long as there is new data. It is a simple and practical method.

Ideally, the shift between two sets can be observed by estimating their distributions and measuring the change between them (e.g. using KL Divergence). However, it is costly in streaming environments as the estimations need to be done repeatedly and we want instant results. What we want is to observe whether two sets differ continuously, not to estimate their distributions. In our intuition, it may be sufficient to learn the divergence between distributions, to be able to detect concept drifts implicitly.

We hold a sliding window: W of the latest data with size $w(1 + \rho)$ (Alg. line 2). w is the size of the old data. ρ is the percentage of new data with respect to old. The size for the new data is calculated as $w\rho$. We store the samples without breaking their time order. The leftmost side (tail) has the older samples whereas the other side (head) has the newer, which can be seen in Figure 1. In the initialization phase, when the whole window is empty, we wait until it gets full (Alg. line 5). Afterwards, we start with the first check. A new slack variable s is introduced. The oldest members of size w are labeled as old, and given value 0 (Alg. line 9). The remaining are labeled new, and given value 1 (Alg. line 10). Later, a logistic regression model is trained as a discriminative classifier to distinguish old and new with s as labels (Alg. line 11).

We use AUC as a measure of separability. It expresses to what degree a model can distinguish two classes [9]. The AUC of a perfect model is near 1 indicating that the model can discriminate the classes successfully. A poor model has an AUC near 0.5 when the distribution of the classes overlap. Depending on the divergence of the classes, we detect a drift. We expect the class distributions to

Algorithm 1 D3: Discriminative Drift Detector

```
1:  procedure D3($\mathcal{D}, w, \rho, \tau$):
2:      Initialize window $W$ where $|W| = w(1+\rho)$
3:      Discriminative classifier $C$            ▹e.g: Logistic Regression
4:      for $(X, y)$ in $\mathcal{D}$ do                  ▹class label $(y)$ is not used
5:          if $W$ is not full then
6:              $W \leftarrow W \cup X$               ▹i.e., add $X$ to the head of $W$
7:          else
8:              $S$ is vector of $s$, $|W| = |S|$        ▹$s$ is a slack variable
9:              $s = 0$ for $W[1, w]$            ▹label for old (0) and new (1)
10:             $s = 1$ for $W[w+1, end]$
11:             Train $C(W, S)$               ▹train $C$ with $S$ as labels of $W$
12:             if $AUC(C, S) \geq \tau$ then           ▹measure AUC score
13:                 drift = True                       ▹drift detected
14:                 Drop $w$ elements from the tail of $W$
15:             else
16:                 drift = False                          ▹no drift
17:                 Drop $w\rho$ elements from the tail of $W$
```

overlap or have slight differences when there is no drift. We set a threshold (τ) for AUC to measure how much the classes (old and new) are separable. If it is over the threshold, we signal a drift (Alg. line 12-17).

There are two possible outcomes of the discriminative classifier, as it is illustrated in Figure 1. **(1)**: AUC is greater than or equal to τ. A drift is detected, the classifier's performance is high, and the old and the new data are separable in the feature space. In that case, we discard the samples from the old data part and replace them with the ones from the new data. **(2)**: AUC is less than τ. The classifier's performance is poor. This indicates that the predictive function fails to separate the two intertwined distributions. Then, we remove the oldest $w\rho$ samples and shift the window to left where the samples from the new fill the recently freed space. For both cases, we wait for new samples to come and check again for drift when the window is filled. This work-flow can go on, as long as the stream generates data.

Even though the sizes of the old (w) and new data ($w\rho$) are hyper parameters of our model, we set the old to be always larger. We need the old portion of the data to be descriptive enough for the current distribution and span as much area in the feature space as possible. However, it should not be very large, otherwise it may contain multiple concepts. Since the old is compared with a relatively smaller portion (new data) to detect changes, using a metric that works well in the presence of class imbalance is highly important. For this reason, we use AUC to measure the performance of the discriminative classifier, as it works well in such cases [9].

5 EMPIRICAL EVALUATION

5.1 Datasets

We test our approach both on 8 well-known real-world and synthetic datasets. The summary of datasets is shown in Table 1.

5.2 Experimental Setup

Experiments are implemented in scikit-multiflow [17]. A Hoeffding Tree is used with default parameters for stream classification. For this purpose, any classifier that supports incremental updates without having a built-in concept drift detector can be used as well. The Hoeffding Tree is chosen due to its effectiveness and prevalence in the literature. D3 is compared with 3 baseline methods: ADWIN [4], DDM [10] and EDDM [2]. They are all operated with default parameters.

Table 1: Datasets with drift

	Name	|X|	|y|	$\|\mathcal{D}\|$
Real	ELEC [12]	6	2	45,312
	COVTYPE [5]	54	7	581,012
	Poker Hand [5]	10	10	829,201
	Rialto [15]	27	10	82,250
Synthetic	Rotating Hyperplane [15]	10	2	200,000
	Moving Squares [15]	2	4	200,000
	Moving RBF [15]	10	5	200,000
	Interchanging RBF [15]	2	15	200,000

Interleaved-Test-Then-Train (prequential) is used for evaluation. A new sample is tested by the classifier first, then the evaluation metric is recorded and the classifier is trained by the instance. Whenever a drift is detected, the Hoeffding Tree is reset and retrained with the latest $w\rho$ samples (new data portion) for all methods.

As a discriminative classifier, logistic regression is used in D3 with default parameters. D3 is tuned with multiple parameters of $w = [100, 250, 500, 1000, 2500]$, $\rho = [0.1, 0.2, 0.3, 0.4, 0.5]$ and $\tau = [0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90]$. We make D3's implementation public[1].

6 RESULTS AND DISCUSSION

The hyper parameter choice is important for D3's performance. After evaluating it with multiple parameters, we found that it works best overall when $w = 100$, $\rho = 0.1$ and $\tau = 0.70$. They are set as the default parameters of our model. Parameters may be optimized for a selected dataset to achieve better individual scores.

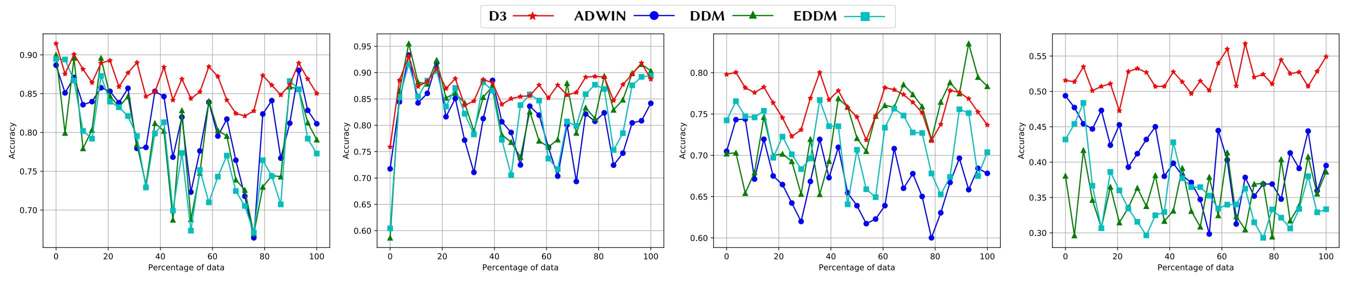
We analyze the results of D3 with default parameters. The overall accuracy for every method on each dataset is given in Table 2. The best scores are highlighted in bold. D3 does not utilize class labels unlike the other concept drift detectors. With less information, it outperforms them in most cases. When the accuracies for each dataset are ranked from best to worst, D3 comes first with an average rank of 1.38.

Table 2: Averaged prequential accuracy and the rankings of the methods for each dataset

Datasets	Accuracy (%)			
	D3	ADWIN	DDM	EDDM
ELEC	**86.69**	81.33	79.30	78.25
COVTYPE	**87.17**	80.48	83.36	82.76
Poker Hand	**75.59**	66.96	73.42	71.31
Rialto	**52.39**	46.64	38.44	51.01
Rotating Hyperplane	85.29	**87.28**	84.01	80.27
Moving Squares	66.28	**67.89**	45.13	33.68
Moving RBF	**51.59**	40.21	35.04	35.33
Interchanging RBF	82.81	**88.65**	41.23	60.06
Average Rank	**1.38**	2.25	3.13	3.25

The other methods track the performance of the classifier and signal a drift when there is a significant drop. They wait for the new

[1]The source code is available on:
https://github.com/ogozuacik/d3-discriminative-drift-detector-concept-drift



(a) ELEC (b) COVTYPE (c) Poker Hand (d) Moving RBF

Figure 2: Prequential accuracy of the methods for the selected datasets. Each dataset is divided into 30 chunks and the results are the averaged prequential accuracies within them.

concept to affect the performance of the classifier adversely and act afterwards. According to the results, we see that D3 acts quicker by signaling a drift when there is a change in $P(X)$. In Figure 2, we give the prequential accuracy scores for 4 different datasets. It shows that the locations of the drifts (declines in accuracy) are in similar areas, but the classifier adapts to it faster with D3.

However, this does not apply for all cases. D3 cannot detect concept drifts caused by changes only in $P(y|X)$ which are referred to as *real concept drift*. In such cases, $P(X)$ stays the same; therefore, our method cannot detect the change. Furthermore, it detects drifts unnecessarily when the change in $P(X)$ does not affect $P(y|X)$ *(virtual drift)*. D3's performance on real-world datasets is better compared to synthetics. This can be caused by the properties of the datasets. In synthetic datasets, real and virtual concept drifts can be more dominant. Even under these drawbacks, the results presented in Table 2 and Figure 2 confirm that D3 works well compared to the other methods.

We used the *Friedman Test with Nemenyi post-hoc analysis* to check the statistical significance of the used methods. The critical distance for Nemenyi Significance is calculated by plugging in values specific for our setting from the *Critical Values Table for Two Tailed Nemenyi Test* [7]. We find $CD = 1.658$ resulting D3 to be statistically significantly better than DDM and EDDM.

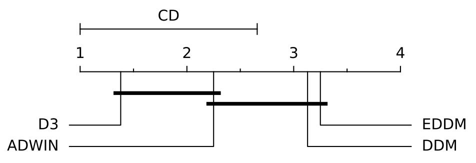


Figure 3: Critical distance diagram for the overall accuracy.

7 CONCLUSION

In this paper, we show an unsupervised method for concept drift detection, D3, using a discriminative classifier over a sliding window of latest elements to monitor whether the old and the new samples vary in their distributions. The experimental results show that D3 performs better overall, compared to the most extensively used methods for drift detection.

As a future work, we plan to investigate the kernelized version of our model. Throughout the experiments, we use logistic regression in D3, which limits us to detect drifts that show a linear pattern on $P(X)$. We also want to test our approach with different types of classifiers, including non-linear ones, such as: SVMs and Decision Trees. Even though we use default parameters for D3, there exist better individual scores when using different parameters on some of the datasets. Therefore, we also plan to work on the auto-parameterization of our model, depending on the properties of the stream.

ACKNOWLEDGMENTS

This study is partially supported by TÜBİTAK grant no. 117E870.

REFERENCES

[1] C. C. Aggarwal. 2007. *Data streams: models and algorithms.* Vol. 31. Springer Science & Business Media.
[2] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno. 2006. Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*, Vol. 6. 77–86.
[3] S. Bickel, M. Brückner, and T. Scheffer. 2007. Discriminative learning for differing training and test distributions. In *Proc. of the 24th ICML*. ACM, 81–88.
[4] A. Bifet and R. Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proc. of the 2007 SIAM SDM.* SIAM, 443–448.
[5] A. Bifet, B. Pfahringer, J. Read, and G. Holmes. 2013. Efficient data stream classification via probabilistic adaptive windows. In *Proc. of the 28th Annual ACM Symposium on Applied Computing*. ACM, 801–806.
[6] S.-H. Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *City* 1, 2 (2007), 1.
[7] J. Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7, Jan (2006), 1–30.
[8] W. Fan and A. Bifet. 2013. Mining big data: current status, and forecast to the future. *ACM SIGKDD Explorations Newsletter* 14, 2 (2013), 1–5.
[9] T. Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.
[10] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. 2004. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*. Springer, 286–295.
[11] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 44.
[12] M. Harries and N. S. Wales. 1999. Splice-2 comparative evaluation: Electricity pricing. (1999).
[13] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, et al. 2014. Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter* 16, 1 (2014), 1–10.
[14] J. Lee and F. Magoules. 2012. Detection of concept drift for learning from stream data. In *2012 IEEE 14th HPCC & 2012 IEEE 9th ICESS*. IEEE, 241–245.
[15] V. Losing, B. Hammer, and H. Wersing. 2016. KNN classifier with self adjusting memory for heterogeneous concept drift. In *2016 IEEE 16th ICDM*. IEEE, 291–300.
[16] M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham. 2011. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering* 23, 6 (2011), 859–874.
[17] J. Montiel, J. Read, A. Bifet, and T. Abdessalem. 2018. Scikit-multiflow: a multi-output streaming framework. *The Journal of Machine Learning Research* 19, 1 (2018), 2915–2914.
[18] E. S. Page. 1954. Continuous inspection schemes. *Biometrika* 41, 1/2 (1954), 100–115.
[19] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. 2009. *Dataset shift in machine learning.* The MIT Press.
[20] T. S. Sethi and M. Kantardzic. 2017. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications* 82 (2017), 77–99.
[21] P. Zhang, X. Zhu, and Y. Shi. 2008. Categorizing and mining concept drifting data streams. In *Proc. of the 14th ACM SIGKDD.* ACM, 812–820.