Annotated Bibliography of Drift Detection Methods

Last Updated in December 2021

(Adbelkader 2020) Towards Robust Production Machine Learning Systems: Managing Dataset Shift

The paper was presented at the Doctoral Symposium at ASE 2020. It proposes a methodology that evaluates the robustness of ML components integrated in software system. Robustness in this paper is defined as a model's ability to maintain similar performance when tested under different conditions. The proposed methodology will eventually address dataset shift and it variants: covariate drift, concept drift, and probability shift. However, the scope of this paper is only covariate drift. Covariate drift refers to differences in distribution between training data and production data, caused by things such as adversarial examples, training data that is not representative of the operational domain, changes in the data acquisition modules, or non-stationary environments.

The paper also distinguishes between active and passive shift detection and adaptation. The active approach monitors the existence of drift and triggers a model update if the shift is significant; it is effective against abrupt and gradual distribution changes. The passive approach assumes that there is drift and continuously updates the model. Consistent with the motivation for our work, it states that the passive approach may perform undesirable model updates with abrupt drift.

Study was performed using a recurrent neural network (RNN) model. The Hellinger distance measure was used an as early detector of covariate drift. Results show low Hellinger distance for data sets that follow the same distribution as the training data, and high values for the data sets that were generated to show drift.

H. Abdelkader, "Towards Robust Production Machine Learning Systems: Managing Dataset Shift," 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, VIC, Australia, 2020, pp. 1164-1166.

(Alippi 2016) Hierarchical Change-Detection Tests

The paper presents hierarchical change-detection tests (HCDTs) for detecting changes in data streams. HCDTs are characterized by a hierarchical architecture composed of a detection layer and a validation layer. The detection layer analyzes the input data stream by means of an online, sequential CDT, which operates as a low-complexity trigger that detects changes in the data, which might be related to changes in the process generating the data (concept drift) (e.g., faulty sensor, anomalous event, unforeseen evolution of the surrounding environment). The validation layer is activated when the detection layer reveals a change, and performs an offline, more sophisticated analysis on the data to reduce false alarms.

Examples of HCDTs are Hierarchical CUSUM Test, Hierarchical ICI-Based CDT and Hierarchical NP-CUSUM Test.

Experiments show an advantageous tradeoff between false-positive rate and detection delay than their single-layered, more traditional counterpart. In addition, the interplay between the two layers permits HCDTs to automatically reconfigure after having detected and validated a change. Thus, HCDTs are able to reveal further departures from the post-change state of the data-generating process.

Alippi, C., Boracchi, G. and Roveri, M., 2016. Hierarchical Change-Detection tests. IEEE Transactions on Neural Networks and Learning Systems, 28(2), pp.246-258.

(Baburoglu 2021) Novel Hybrid Pair Recommendations Based on a Large-Scale Comparative Study of Concept Drift Detection

The paper presents pf a pair-wise comparison of 15 detectors with 6 different classifiers to determine and recommend the best pairs. The six classifiers studied were six Naïve Bayes (NB), Hoeffding tree (HT), Hoeffding option tree (HOT), Perceptron (P), decision stump (DS), and k-nearest neighbor (KNN). The 15 detectors were drift detection method (DDM), early drift detection method (EDDM), EWMA for concept drift detection (ECDD), adaptive sliding window (ADWIN), geometrical moving average (GMA), drift detection methods based on Hoeffding's bound (HDDMA and HDDMW), Fisher exact test drift detector (FTDD), Fast Hoeffding drift detection method (FHDDM), Page–Hinkley test (PH), reactive drift detection method (RDDM), SEED, statistical test of equal proportions (STEPD), SeqDrift2, and Wilcoxon rank-sum test drift detector (WSTD). Experiments were run using the MOA framework, specifically using Agrawal, Sine, and Waveform as dataset generators. The concept drift detectors were compared in terms of both their final accuracies per CPU time and the quality of their concept drift detections in a fully supervised setting. The quality of the drift detectors was measured using the minimum FP and FN rates, as well as the shortest delay of detection.

Babüroğlu, E.S., Durmuşoğlu, A. and Dereli, T., 2021. Novel Hybrid Pair Recommendations Based on a Large-Scale Comparative Study of Concept Drift Detection. Expert Systems with Applications, 163.

(Baena-Garcia 2006) Early Drift Detection Method

The paper proposes EDDM, which is a method referenced by several of the other papers. EDDM shows improvement over DDM, even in cases of slow gradual change. It is based on the estimated distribution of the distances between classification errors.

Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R. and Morales-Bueno, R., 2006, September. Early Drift Detection Method. In Fourth International Workshop on Knowledge Discovery from Data Streams (Vol. 6, pp. 77-86).

(Barros 2017) RDDM: Reactive Drift Detection Method

Presents RDDM as an improvement over DDM. It addresses the lack of sensitivity problem of DDM when concepts are very large. Was tested against DDM, ECDD and STEPD using Naive Bayes as base learners. RDDM outperformed the other three methods in accuracy and presented the best balance of false negative and false positive detection rates.

DDM detects changes in a distribution by analyzing the error rate and the corresponding standard deviation in a sequence. An increase in the error rate suggests there was a change in the data distribution and the current base learner is thus likely to have become inefficient.

The main idea behind RDDM is to periodically shorten the number of instances of very long stable concepts to tackle a known performance loss problem of DDM. It is assumed that such a drop is caused by decreased sensitivity to concept drifts as a result of very large number of instances belonging to a given concept. This occurs because, in concepts with many thousands of instances, it takes a fairly large number of prediction errors to sufficiently affect the mean error rate and trigger the drifts.

RDDM implements a softer type of concept drift that does not affect the base learner and is performed after long periods within a stable concept state, periodically recalculating the DDM statistics responsible for detecting the warning and drift levels, using a parametrized smaller number of instances to deal with the aforementioned loss of sensitivity problem of DDM. Moreover, to prevent the failure of this strategy, which would probably occur if DDM stayed at the warning level for too many instances, RDDM also forces the traditional DDM drifts when the length of the warning period (in number of instances) reaches another parametrized threshold.

Barros, R.S., Cabral, D.R., Gonçalves Jr, P.M. and Santos, S.G., 2017. RDDM: Reactive Drift Detection Method. Expert Systems with Applications, 90, pp. 344-355.

(Barros 2018) A Large-Scale Comparison of Concept Drift Detectors

Paper is a more recent comparison of concept drift detectors. Experiments also based on the MOA dataset extensions. Talks about several dataset generators used: Agrawal, LED, Mixed, Random RBF, Sine, and Waveform. Findings are consistent with our hypothesis, that some methods are better at detecting different types of drift. Paper has all the results from a very large set of experiments.

Barros, R.S.M. and Santos, S.G.T.C., 2018. A Large-Scale Comparison of Concept Drift Detectors. Information Sciences, 451, pp.348-370.

(Berend 2020) Cats Are Not Fish: Deep Learning Testing Calls for Out-Of-Distribution Awareness

This is a paper from ASE 2020. The motivation for the paper is that because state-of-the art testing techniques for deep learning (DL) components do not consider data distribution, it is difficult to know if the errors are really meaningful or just outliers that cannot be handled by the model. The authors therefore conducted a large-scale empirical study on state-of-the-art out-of-distribution (OOD) detection techniques towards understanding data distribution and its impact on DL testing activities.

Very simple definition of deep learning: "Different from traditional software whose decision logic is mostly programmed by the developer, DL adopts a data-driven programming paradigm. In particular, the major tasks of a DL developer are preparing the training data, labeling the data, programming the architecture of the deep neural network (DNN), and specifying the training configuration. All the decision logic is automatically learned during the runtime training phase and encoded in the obtained DNN (e.g., by weights, bias, and their combinations). Due to the differences of programming paradigm, the logic encoding format, and the tasks that a DNN is often developed for (e.g., image recognition), testing techniques for traditional software cannot be directly applied and new testing techniques are needed for DNNs."

The fundamental assumption of deep learning is that the training data follows some distribution, based on which the learning algorithms train the DNN to obtain its decision logic and are able to handle future data that follow a similar distribution. If the new unseen input data has a similar distribution as the training data, deep learning provides some statistical guarantee on its prediction correctness in terms of accuracy. However, if the new input data does not follow the training data (i.e., OOD), deep learning does not provide statistical guarantees on its prediction.

Intuitively, erroneous inputs that follow the distribution of training data may reveal the real weakness of the DNN since the DNN is expected to handle this data. On the other hand, input errors that are considered out-of-distribution may either inherit new information benefitting generalization as well as domain shift or are simply irrelevant to the DL application. Thereby, the root cause of an error may be identified through analyzing its distribution behavior, which makes us rethink how to define errors and how to test the DNN by considering the effect on its trained distribution.

The challenge of OOD detection is that there is no perfect ground truth for validating whether one sample is in-distribution (ID) or out-of-distribution. The common approach of existing techniques to overcome this problem is utilizing significantly different datasets to approximate the ground truth.

Coverage-guided testing is a representative and widely used technique, which usually contains three main components: the mutation operator, the coverage criteria, and the oracle. The mutation operator is used to generate diverse test cases such that more behaviors can be tested. For example, in image classification, mutation operators such as image brightness, blur, or contrast are applied under consideration of realistic settings. The coverage criteria measure the degree of how much the DNN is tested. The newly generated test cases are kept when they have achieved new coverage of the DNN. At last, the oracle is used to judge whether a new test case is a benign test case, (i.e., correctly predicted), or an error test case, (i.e., incorrectly predicted). The assumption is that the test cases are generated by adding minor perturbations on the original input, so they should have the similar prediction result. However, the existing testing tools do not consider the distribution of the training data, which determines what data can or cannot be handled by the target DNN. The errors may be caused by the defects of the DNN model itself (e.g., inappropriate model architecture, learning process) or the lack of the training data. Hence, it is important to distinguish the different types of errors (e.g., the ID and OOD errors), which provides more feedback for the developers. In practice, detecting the out-of-distribution data is a challenging problem, especially for the high-dimensional data.

Studied five state-of-the-art OOD detection techniques Baseline, ODIN, Mahalanobis, Outlier Exposure, and Likelihood-Ratio. Mapped them against mutation operators (contrast, blur, brightness, noise, translation, scale, shear, rotation) used by tools such as DeepTest, DeepHunter and TensorFuzz to determine which detection techniques were best at detecting OOD data. Results showed that Outlier Exposure is the best at detecting OOD data and that Image Blur and Image Scale are the best at generating OOD data. Also studied which coverage criteria (NC, KMNC, SNAC, TKNC, NBC, FANN) are more likely to generate ID data or OOD data. Results show that different criteria result in different additions or removals of both error and benign OOD test cases.

They also proved that ID error test cases tend to be a result of defects in the DNN model while OOD error test cases tend to be a result of missing data in the training set.

All datasets and results are available here: https://sites.google.com/view/oodtesting/home

Berend, David, et al. "Cats are not Fish: Deep Learning Testing Calls for Out-of-Distribution Awareness." 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2020.

(Bifet 2009) Learning from Time-Changing Data with Adaptive Windowing

Paper that proposes ADWIN and a newer version ADWIN2.

Present ADWIN (Adaptive Windowing) as an approach for dealing with distribution change and concept drift when learning from data sequences that may vary with time. Use sliding windows whose size, instead of being fixed a priori, is recomputed online according to the rate of change observed from the data in the window itself. The algorithm automatically grows the window when no change is apparent and shrinks it when data changes.

They also present a time- and memory-efficient version of this algorithm, called ADWIN2. Show how to combine ADWIN2 with the Naïve Bayes (NB) predictor, in two ways: one, using it to monitor the error rate of the current model and declare when revision is necessary and, two, putting it inside the NB predictor to maintain up-to-date estimations of conditional probabilities in the data.

Premise is that dealing with data whose nature changes over time is one of the core problems in data mining and machine learning. To mine or learn such data, one needs strategies for the following three tasks, at least: 1) detecting when change occurs, 2) deciding which examples to keep and which ones to forget (or, more in general, keeping updated sufficient statistics), and 3) revising the current model(s) when significant change has been detected.

Bifet, A. and Gavalda, R., 2007, April. Learning from Time-Changing Data with Adaptive Windowing. In Proceedings of the 2007 SIAM International Conference on Data Mining (pp. 443-448). Society for Industrial and Applied Mathematics.

(Budiman 2016) Adaptive Convolutional ELM For Concept Drift Handling in Online Stream Data

This paper outlines a technique using extreme learning machines (ELMs) for building more robust CNNs. We might consider investigating whether there are useful comparisons to be made between this method and our drift detection methods.

It proposes a method called Adaptive CNN-ELM (ACNNELM) that integrates CNNs with ELMs to better handle concept drift in online streaming big data. "ELMs use set and fixed random values in hidden node parameters and a non-iterative generalized pseudo-inverse optimization process, while NNs use iterative a gradient descent optimization process to smooth the weight parameters."

In CNN-ELM the last convolutional layer output from a CNN is used as the hidden nodes weight for an ELM. in ACCELM they use an ensemble of CNN-ELMs to deal with drift.

Introduces the concept of hybrid drift, which is when both real and virtual drift occur (i.e., change in classes plus change in output data distribution). The claim is that their proposed approach can deal better with hybrid drift.

Budiman, Arif, Mohamad Ivan Fanany, and Chan Basaruddin. "Adaptive Convolutional ELM for Concept Drift Handling in Online Stream Data." arXiv preprint arXiv:1610.02348 (2016).

(Cieslak 2009) A Framework for Monitoring Classifiers' Performance: When and Why Failure Occurs?

This paper presents a two-stage drift detection method. In stage one it detects statistically-significant data distribution changes using the Kruskal-Wallis (KW) test. If there is a change in distribution, it then uses the Kolmogorov-Smirnov (KS) test (or the Chi-Squared test) plus

Hellinger distance to indicate the presence or absence of change in the feature space. They use three scenarios for evaluation: sample selection bias (MCAR, MNAR, and MAR), covariate shift, and shifting class priors.

Paper also explains how the different drift scenarios were introduced into the datasets used for evaluation. Includes pointers to the different datasets that they used.

Cieslak, D.A. and Chawla, N.V., 2009. A Framework for Monitoring Classifiers' Performance: When and Why Failure Occurs?. Knowledge and Information Systems, 18(1), pp. 83-108.

(Elwell 2011) Incremental Learning of Concept Drift in Nonstationary Environments

The paper presents Learn++ .NSE, an ensemble-of-classifiers-based approach for incremental learning of concept drift, characterized by nonstationary environments (NSEs), where the underlying data distributions change over time. It learns from consecutive batches of data without making any assumptions on the nature or rate of drift; it can learn from environments that experience constant or variable rate of drift, addition or deletion of concept classes, as well as cyclical drift. Learn ++ .NSE trains one new classifier for each batch of data it receives, and combines these classifiers using a dynamically weighted majority voting. The novelty of the approach is in determining the voting weights, based on each classifier's time-adjusted accuracy on current and past environments. This approach allows the algorithm to recognize, and act accordingly, to the changes in underlying data distributions, as well as to a possible reoccurrence of an earlier distribution.

Makes an interesting point about virtual drift vs. real drift: "Virtual drift is the result of an incomplete representation of the true distribution in the current data. The key difference is that real drift requires replacement learning (where old knowledge becomes irrelevant [restructuring]), whereas virtual drift requires supplemental learning (adding to the current knowledge [tuning])"

Paper falls in the category of using classifier ensembles to adapt to drift.

Elwell, R. and Polikar, R., 2011. Incremental Learning of Concept Drift in Nonstationary Environments. IEEE Transactions on Neural Networks, 22(10), pp. 1517-1531.

(Frias-Blanco 2015) Online and Non-Parametric Drift Detection Methods based on Hoeffding's Bounds

Drift Detection Method based on Hoeffding's Bounds (HDDM) proposes a family of methods to monitor the mean of the classifier's performance over time in order to detect significant changes. Differently from DDM, which assumes that classification values are generated based on a Bernoulli distribution, HDDM does not assume any distribution for the incoming values.

Two main approaches are proposed, the first one involves moving averages and is more suitable to detect abrupt changes. The second one uses using weighted moving averages to deal with gradual changes. To avoid high false positive rates, it uses a WARNING state before going into a DRIFT state.

Frias-Blanco, I., del Campo-Ávila, J., Ramos-Jimenez, G., Morales-Bueno, R., Ortiz-Diaz, A. and Caballero-Mota, Y., 2014. Online and Non-Parametric Drift Detection Methods based on Hoeffding's Bounds. IEEE Transactions on Knowledge and Data Engineering, 27(3), pp.810-823.

(Hofer 2013) Drift Mining in Data: A Framework for Addressing Drift in Classification

Drift denotes changes in the joint distribution of explanatory variables and class labels over time. It entails the deterioration of a classifier's performance and requires the optimal decision boundary to be adapted after some time. However, in the presence of verification latency a re-estimation of the classification model is impossible, because in such a situation only recent unlabeled data is available, and the true corresponding labels only become known after some lapse in time. The drift mining methodology presented in this paper aims at detecting changes over time. It allows to understand evolution in the data from an ex-post perspective or, ex-ante, to anticipate changes in the joint distribution.

The paper first proposes a model for global drift that affects all attributes homogeneously. It then proposes a drift mining methodology that models and evaluates these changes over time. Because of verification latency and lack of recent labeled data, drift mining is used to anticipate changes in the joint distribution and to decompose changes in distributions into global and local changes. Their method was applied to a dataset on default rates in Danish companies, which is a typical example of verification latency use case.

Hofer, V. and Krempl, G., 2013. Drift Mining in Data: A Framework for Addressing Drift in Classification. Computational Statistics & Data Analysis, 57(1), pp.377-391.

(Gama 2004) Learning with Drift Detection

Early paper by the same authors of (Gama 2014) in which they present a drift detection method for online learning models.

The premise of the work is that in online learning models, learning takes place in a sequence of trials. For each trial, the learner makes some kind of prediction and then receives some kind of feedback. They define context as a set of examples in which the function generating examples is stationary. They also assume that a data stream is composed of a set of contexts. Changes between contexts can be gradual - when there is a smooth transition between distributions; or abrupt - when the distribution changes quickly. The goal of the work is to present a straightforward and direct method to detect the moments in which there is a change of context. If they can identify contexts, then they can identify which information is outdated and re-learn the model only with information that is relevant to the present context.

Background: "In general, approaches to cope with concept drift can be classified into two categories: i) approaches that adapt a learner at regular intervals without considering whether changes have really occurred; and ii) approaches that first detect concept changes, and then, the learner is adapted to these changes ... Examples of the former approaches are weighted examples and time windows of fixed size. Weighted examples are based on the simple idea that the importance of an example should decrease with time ... When a time window is used, at each time step the learner is induced only from the examples that are included in the window. Here, the key difficulty is how to select the appropriate window size: a small window can assure a fast adaptability in phases with concept changes but in more stable phases it can affect the learner performance, while a large window would produce good and stable learning results in stable phases but cannot react quickly to concept changes. In the latter approaches, with the aim of detecting concept changes, some indicators are monitored over time ... If during the monitoring process a concept drift is detected, some actions to adapt the learner to these changes can be taken. When a time window of adaptive size is used these actions usually lead to adjusting the window size according to the extent of concept drift. As a general rule, if concept drift is detected the window size decreases, otherwise the window size increases ... Other methods include monitoring the values of performance indicators (e.g., accuracy, recall and precision over time), and then, comparing it to a confidence interval of standard sample errors for a moving average value (using the last M batches) of each particular indicator."

The proposed method monitors the error rate of the learning algorithm and generates a warning when it reaches a set threshold. When a drift level is reached, it basically uses the data between the warning alert and the drift alert as a new training set. This also addresses false alarms.

Gama, J., Medas, P., Castillo, G. and Rodrigues, P., 2004, September. Learning with Drift Detection. In Brazilian Symposium on Artificial Intelligence (pp. 286-295). Springer, Berlin, Heidelberg.

(Gama 2014) A Survey on Concept Drift Adaptation

Similar to Webb (2015) in outlining a taxonomy of drift types but also proposes adaptations to models to account for them.

Adaptive learning refers to updating predictive models online during their operation to react to concept drifts. Adaptive learning algorithms can be seen as advanced incremental learning algorithms that are able to adapt to evolution of the data-generating process over time.

For background: Two learning modes: offline learning and online learning. In offline learning, the whole training data must be available at the time of model training. Only when training is completed can the model be used for predicting. In contrast, online algorithms process data sequentially. They produce a model and put it in operation without having the complete training dataset available at the beginning. The model is continuously updated during operation as more training data arrives.

Classifies drift as sudden/abrupt, incremental, gradual, and reoccurring (Figure 2 has a nice visual distinction).

Adaptation requirements

- detect concept drift (and adapt if needed) as soon as possible;
- distinguish drifts from noise and be adaptive to changes, but robust to noise; and
- operate in less than example arrival time and use not more than a fixed amount of memory for any storage

Four modules of adaptive learning algorithms (Figure 4-8)

- Memory: includes methods for updating the model with new information (i.e., deciding what data to use for training), but methods for forgetting old information (abrupt or gradual)
- Change detection: Four types of detectors
    - Detectors based on sequential analysis (e.g., CUSUM, PH)
    - Detectors based on Statistical Process Control
    - Methods monitoring distributions of two different time windows (e.g., ADWIN)
    - Contextual approaches (e.g., Splice)
- Learning: includes (i) learning mode: updating the model when new data points are available (retraining or incremental); (ii) model adaptation: analyzes the behavior of

predictive models on time-evolving data (blind or informed); and (iii) model management: techniques for maintaining active predictive models (single model or ensemble)
- Loss estimation: can be model-specific or model-independent

Change detection methods are evaluated with respect to probability of true change detection, probability of false alarms, and delay of detection

Gama, João, et al. "A Survey on Concept Drift Adaptation." ACM Computing Surveys (CSUR) 46.4 (2014): 1-37.

(Goldenberg 2019) Survey of Distance Measures for Quantifying Concept Drift and Shift in Numeric Data

The paper presents the results of a survey of distance measures with respect to their suitability for estimating drift and shift magnitude between samples of numeric data. The focus is on high-dimensional numeric data is because they consider it more challenging to extract useful estimates of quantitative measures of drift from sample data. The paper addresses this task of estimating drift and shift from sample data. It provides a summary and analysis of the mathematical tools and distance measures that can quantify the dissimilarity between these samples and/or the magnitude of data drift.

The paper differentiates between concept drift, which is changes in statistical properties of data over time, and concept shift, when models are trained in one context and then applied to another context. It then proposes to quantify drift or shift (mainly covariate drift or shift) by using distance measures between distributions. As is well known, failure to account for concept drift and/or shift leads to models that lose their predictive power when drift occurs.

Concept Drift is defined as a change of distribution between two points in time $P_{t0}(X,Y) \neq P_{t1}(X,Y)$, where X is the covariate variable and Y is the response variable. It also distinguishes between *Real Concept Drift*, which is change in the concept itself ($P_{t0}(Y|X) \neq P_{t1}(Y|X)$ && $P_{t0}(X) = P_{t1}(X)$), and *Virtual Concept Drift* which is changes in the data distribution over time ($P_{t0}(Y|X) = P_{t1}(Y|X)$ && $P_{t0}(X) \neq P_{t1}(X)$). They note that real concept drift under supervised learning can be detected and, to some degree, quantified by monitoring prediction error of the model. This might not be available in case of virtual drift, unsupervised learning, or when the true values of response variable take a long time to arrive

Surveyed measures include (paper provides pros and cons for all)

- Low-dimensional (and univariate) distance measures
    - Measures of dissimilarity in distribution
        - Hellinger distance
        - Kullback-Leibler (KL) divergence
        - Total variation distance (TVD)
        - The Kolmogorov-Smirnov statistic (KS distance)
        - Wasserstein distance (also known as earth mover's distance (EMD) or the transportation metric)
        - Energy distance
    - Measure of distance between objects, T-ratio, T-statistics and T-test
- Multivariate distance measures
    - Measures of dissimilarity in distribution
        - Principal component analysis (PCA)
        - Distribution of Mahalanobis distance

- Measures of distance between objects, Hotelling's T^2 distance

Recommendations

- Hellinger distance can be used as a measure of dissimilarity between distributions for univariate or low-dimensional data.
- T-statistics and Hotelling T^2 can be used as an approximation for univariate and multivariate data, respectively.
- For high-dimensional numeric data, the PCA-based approach can be used, but they caution that there are issues that require further investigation.

Goldenberg, I., Webb, G.I. Survey of Distance Measures for Quantifying Concept Drift and Shift in Numeric Data. Knowledge and Information Systems 60, 591–615 (2019). https://doi.org/10.1007/s10115-018-1257-z

(Goldenberg 2020) PCA-Based Drift and Shift Quantification Framework for Multidimensional Data

This paper builds on findings from (Goldenberg 2019) that stated that there were not many methods to measure drift in multi-dimensional data and that the use of Principal Component Analysis (PCA) to measure drift in multi-dimensional data required more research.

Motivation is that to measure the magnitude of drift, a statistical distance between the distributions that generate the data used to train the model and the data used to test the model must be measured. However, most existing distance measures for quantitative data do not scale up to high dimensions. In addition, these existing methods are good at detecting drift, but not at measuring it.

After explaining and comparing Hellinger distance and PCA, in addition to presenting experiment results, the paper proposes a framework that combines them.

Goldenberg, I., Webb, G.I. PCA-Based Drift and Shift Quantification Framework for Multidimensional Data. Knowledge and Information Systems 62, 2835–2854 (2020). https://doi.org/10.1007/s10115-020-01438-3

(Goncalves 2014) A Comparative Study on Concept Drift Detectors

The paper evaluated eight different concept drift detectors (DDM, EDDM, PHT, STEPD, DOF, ADWIN, Paired Learners, and ECDD). A 2^k factorial design was used to indicate the best parameters for each method (i.e., the parameters that most influence performance). Tests compared accuracy, evaluation time, false alarm and miss detection rates. A Mahalanobis distance (i.e., distance to the drift point) is also proposed as a metric to compare drift methods. DDM was the method that presented the best average results in all tested datasets.

- Drift Detection Method (DDM): Uses a base learner to classify incoming instances and the classification result is used to compute the online error-rate of the base learner. The classification result indicates whether the base learner classified the arriving instance correctly or not. If the base learner correctly classifies the actual instance, the error-rate decreases. DDM considers that, when the concept changes, the base learner will incorrectly classify the arriving instances that are created based on a different data distribution. Thus, if the error-rate increases, it is an indication of a concept drift.

- Early Drift Detection Method (EDDM): Similar to DDM but, instead of using the error rate, it uses the distance-error-rate of the base learner to identify whether a drift has occurred.
- Page-Hinkley Test (PHT): Sequential analysis technique that computes the observed values (e.g., accuracy of the classifier) and their mean up to the current moment. When a concept drift occurs, the base learner will fail to correctly classify incoming instances, making the actual accuracy decrease.
- Adaptive Windowing (ADWIN): Uses sliding windows of variable size, which are recomputed online according to the rate of change observed from the data in these windows. The algorithm dynamically enlarges the window (W) when there is no apparent change in the context and shrinks it when a change is detected. The algorithm attempts to find two sub-windows of W that exhibit distinct averages. If that occurs, it concludes that the corresponding expected values are different, meaning that the older portion of the window is based on a data distribution different than that of the present one, and is therefore dropped.
- Paired Learners (PL): Uses two learners: a stable and a reactive one. The stable learner predicts based on all of its experience, while the reactive one predicts based on a window of recent examples. A circular list of bits with length w (the same length as the window) stores the value one if an instance was incorrectly classified by the stable learner and correctly classified by the reactive learner, and zero otherwise. If the number of bits of this circular list set to one exceeds a parameterized value θ, it indicates that the reactive learner, trained on recent instances, has a better predictive accuracy than the stable learner. Thus, the stable learner is replaced by the reactive one, and all the bits in the list are set to zero.
- Exponentially-Weighted-Moving-Average Concept Drift Detection (ECDD): The classification accuracy of the base learner is calculated together with an estimator of the expected time between false positive detections. Two estimations are compared: one with more weight on recent examples and another with similar emphasis on both recent and old data. When the difference between these two estimations exceeds a certain parameterized threshold, a concept drift is identified.
- Statistical Test of Equal Proportions (STEPD): Computes the accuracy of the base learner in the W most recent instances and compares it to its overall accuracy from the beginning of the learning process.
- Degree of Drift (DOF): Detects drift by processing data chunk by chunk, computing the nearest neighbor in the previous batch for each instance in the current batch and comparing their corresponding labels. A distance map is created, associating the index of the instance in the previous batch and the label computed by its nearest neighbor. A metric known as degree of drift is computed based on the distance map. The average and standard deviation of all degrees of drift are computed and, if the current value is away from the average more than s standard deviations, a concept drift is raised.

Used the Sine and Stagger datasets that contain abrupt drift; the Mixed and Hyperplane datasets that contain gradual drift. Real-world datasets were the ones available at http://moa.cms.waikato.ac.nz/datasets/ (Covertype, Electricity and Poker Hand) and the one available at http://users.rowan.edu/polikar/research/nse/ (Nebraska Weather Prediction). Mixed data sets that they created are available at http://sites.google.com/site/moaextensions/.

Contains details of the experiment setup that we can leverage.

Results

- PL was the method that presented the lowest average rank in datasets with abrupt concept drifts, while DDM presented the highest average accuracies. DOF was the worst in both metrics.
- DDM was the best method in datasets affected by gradual concept drifts, considering both the average rank and the accuracies, while PL was the worst in both metrics.
- In the real-world datasets, PL presented the lowest average rank and PHT the highest average accuracies, while DOF was the worst in both metrics.
- Taking all tested datasets into account, DDM was the best method considering both the average rank and the accuracies.
- Regarding evaluation time, even though ADWIN was the fastest drift detection method in 6 out of 12 datasets tested, being even faster than the base learner in some real-world ones, PHT and DDM presented the lowest average ranks. On the other hand, PL was the slowest in 11 out of 12 datasets and DOF was the second slowest.
- In the abrupt drift datasets, PL was the best method for identifying the drift position close to the actual drift points. STEPD presented the lowest standard deviation, identifying drifts practically always at the same time step, while DOF presented a highly unstable behavior, almost always indicating a context change at each time step. The methods that returned the lowest number of false alarms were STEPD, DDM, and ADWIN, whereas ECDD, PL, and STEPD presented the lowest miss detection rates.
- Finally, we also computed the Mahalanobis distance of the methods and the best possible drift identification pattern. We proposed a weighting function that yields zero if the method matches the benchmark and higher values as the drift identification deviates from the benchmark. STEPD yielded the lowest distance to the expected behavior of the drift methods.

Suggest that an ensemble of methods could be used. Would require testing different values of warning and drift levels. Could also be helpful in determining types of drift.

Gonçalves Jr, P.M., de Carvalho Santos, S.G., Barros, R.S. and Vieira, D.C., 2014. A Comparative Study on Concept Drift Detectors. Expert Systems with Applications, 41(18), pp.8144-8156.

(Haque 2016) Efficient Handling of Concept Drift and Concept Evolution over Stream Data

The paper builds on previous work on SAND, a semi-supervised framework that uses change detection on classifier confidence to detect concept drift. Even though it required only a limited amount of labeled data to detect chunk boundaries (i.e., sliding windows) and to update the classifier when change was detected, it was very expensive in terms of execution time. ECHO (Efficient Concept Drift and Concept Evolution Handling over Stream Data) is the new framework that uses dynamic programming and executes the change detection module selectively, reducing execution time while still providing high accuracy.

ECHO maintains the ensemble approach of SAND and estimates classifier confidence in classifying each test instance, stores it in the sliding window, and tracks any significant change in confidence estimates using a change detection technique. A change in the classifier confidence indicates occurrence of a concept drift. It selects data instances for labeling based on the classifier confidence score calculated during testing. If the confidence in classifying an instance is low, a true label for that instance is requested; if not it labels using majority voting. The new classifier is trained on the new training data and replaces the oldest classifier in the ensemble.

Haque, A., Khan, L., Baron, M., Thuraisingham, B. and Aggarwal, C., 2016, May. Efficient Handling of Concept Drift and Concept Evolution over Stream Data. In 2016 IEEE 32nd International Conference on Data Engineering (ICDE) (pp. 481-492). IEEE.

(Huang 2014) Detecting Volatility Shift in Data Streams

The paper defines stream volatility as the rate of distribution change in a stream. It then proposes SEED, which uses two windows and a statistical test that compares the means of both windows to identify concept drifts. Similar to EDDM, it uses the distance between concept drifts to compute the volatility shift of the stream. Approach shows low false positive rate and much lower execution overhead than ADWIN2.

The premise is that drift detection itself discovers one set of knowledge from the data and with volatility detection, which runs together with drift detection, the user is able to retrieve and discover further knowledge beyond the original set

Huang, D.T.J., Koh, Y.S., Dobbie, G. and Pears, R., 2014, December. Detecting Volatility Shift in Data Streams. In 2014 IEEE International Conference on Data Mining (pp. 863-868). IEEE.

(Kolter 2007) Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts

Ensemble methods maintain a collection of learners and combine their decisions to make an overall decision. The authors propose an ensemble method for concept drift called Dynamic Weighted Majority (DWM). It maintains a weighted pool of experts or base learners. It then adds and removes experts based on the global algorithm's performance. If the global algorithm makes a mistake, then DWM adds an expert. If an expert makes a mistake, then DWM reduces its weight. If, in spite of multiple training episodes, an expert performs poorly, as indicated by a sufficiently low weight, then DWM removes it from the ensemble.

They implemented two versions of DWM based on the algorithm used as the base learner: DWM-NB (Naive Bayes) and DWM-ITI (Incremental Tree Inducer). Used several evaluation benchmarks, including STAGGER.

Kolter, J.Z. and Maloof, M.A., 2007. Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. The Journal of Machine Learning Research, 8, pp. 2755-2790.

(Krawczyk 2017) Ensemble Learning for Data Stream Analysis: A Survey

The motivation for this paper is that data streams pose challenges for machine learning and data mining because of the need to process fast growing amounts of data: limited resources, timely drift detection, rapid arrival rate, non-stationary (i.e., data evolves over time). This paper surveys research on ensembles for data stream classification as well as regression tasks, as a way to deal with these challenges.

An ensemble, also called a multiple classifier or committee, is defined as a set of individual component classifiers whose predictions are combined to predict new incoming instances. Classifier ensembles are an attractive approach to construct data stream classifiers, because they facilitate adaptation to changes in the data distribution. Their adaptation could be done by changing the line-up of the ensemble, e.g., by adding components classifiers trained on the most recent data and/or removing the outdated classifiers, or by retraining the ensemble components.

Related to our work, the paper states that techniques for handling real drift can work for certain types of virtual drift (i.e., changes in data distribution), which is common in data streams. It presents four types of drift: incremental, gradual, sudden, and recurrent (Figure 3 has a nice visual explanation of the differences). Figure 4 presents the concept of model restoration time as time between drift appearance and deployment of a retrained model with acceptable performance. The goal of drift detectors is to reduce maximum performance deterioration and minimize restoration time. Figure 5 presents the idea of drift detection based on tracking classifier errors, differentiating between the real drift point, the warning detection point and the drift detection point. It proposed to gather data for model retraining/update between the warning detection and drift detection points.

Recommend the following metrics for assessing drift detector performance:

- number of true positive drift detections
- number of false alarms, i.e., false positive drift detections
- drift detection delay, i.e., time between real drift appearance and its detection

Some drift detection methods described include:

- DDM (Drift Detection Method): It is the most well-known representative of methods based on statistical process control. It estimates classifier error (and its standard deviation), which (assuming the convergence of the classifier training method) has to decrease as more training examples are received. If the classifier error is increasing with the number of training examples, then this suggests a concept drift.
- EDDM (Early Drift Detection Method) is a modification of DDM to improve the detection of gradual drifts. The same idea of warning and drift levels is realized with a new proposal of comparing distances of error rates.
- Sequential probability ratio tests
    - Wald test
    - Cumulative Sum Approach (CUSUM): detects a change of a given parameter value of a probability distribution and indicates when the change is significant.
    - PageHinkley: Modification of the CUSUM algorithm, where the cumulative difference between observed classifier error and its average is taken into consideration.
- Non-parametric estimation of classifier error
    - Hoeffding’s inequality
    - McDiarmid’s inequality
- ADWIN: Best known representative of methods comparing two sliding windows. In this algorithm a window of incoming examples grows until identifying a change in the average value inside the window. When the algorithm succeeds at finding two distinct sub-windows, their split point is considered as an indication of concept drift.
- Non-parametric tests
    - Computational intelligence cumulative sum test
    - Intersection of confidence intervals-based change detection test
    - CNF density estimation test
    - Multivariate version of the Wald–Wolfowitz test
- Non-parametric univariate statistical tests
    - Two-sample Kolmogorov–Smirnov test
    - Wilcoxon rank sum test

- Two-sample t -test

Nice primer on evaluation methods and how they apply (or not) to data streams. Also include a summary of ensembles for non-stationary data streams that use both active and passive approaches to drift detection and adaptation, for both classification and regression models.

Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J. and Woźniak, M., 2017. Ensemble Learning for Data Stream Analysis: A Survey. Information Fusion, 37, pp.132-156.

(Kumar 2015) Empirical Comparison of Active Learning Strategies for Handling Temporal Drift

The paper presents an empirical study that evaluates the effectiveness of using active learning strategies to train statistical models in the presence of various temporal drift scenarios.

Active learning algorithms attempt to learn an accurate statistical model by selecting the most informative data to be used for training. The paper basically evaluates existing active learning techniques under various temporal drift scenarios to assess if it's worth the additional effort to implement intelligent sample selection strategies over using simple random sampling, when obtaining labels for training is expensive and the domain is susceptible to temporal/concept drift.

Experiments were done using different combinations of the following five parameters: type of drift, amount of drift, target class distribution, evaluation metric of interest, and cost of labeled data.

The performance of the best performing active learning strategies, were found to be at least comparable, if not significantly better than a random sampling strategy across the various types of temporal drifts in 99% of the scenarios tested. In approximately 50% of those instances, active learning strategies were significantly better than random sampling. However, the further away the temporal drift, less is the advantage of using active learning strategies over random sampling.

Kumar, M., Shah, M., Ghani, R. and Abraham, Z., 2015. Empirical Comparison of Active Learning Strategies for Handling Temporal Drift. In ACM SIGKDD 2015 Workshop on Interactive Data Exploration and Analytics (IDEA'15) (pp. 63-71).

(Lipton 2018) Detecting and Correcting for Label Shift with Black Box Predictors

The paper proposes the use of Black Box Shift Estimation (BBSE) for detecting distribution shifts between labeled training data and unlabeled test (production) data. Their approach also attempts to quantify the label shift (i.e., target shift) and correct the model to perform well on the new data. BBSE estimates the ratios $w_l = q(y_l)/p(y_l)$ for each label l, requiring only that the expected confusion matrix is invertible.

Lipton, Z., Wang, Y.X. and Smola, A., 2018, July. Detecting and Correcting for Label Shift with Black Box Predictors. In International Conference on Machine Learning (pp. 3122-3130). PMLR.

(Moreno-Torres 2012) A Unifying View on Dataset Shift in Classification

The goal of the paper is to provide a taxonomy of methods and terminology related to dataset shift.

Background: "Dataset shift occurs when the testing (unseen) data experience a phenomenon that leads to a change in the distribution of a single feature, a combination of features, or the class boundaries. As a result, the common assumption that the training and testing data follow the same distributions is often violated in real-world applications and scenarios." Note that by testing what they really mean is production.

Some of this might be repetitive but included here for completeness. Paper includes formal definitions for all.

Types of dataset shift

- Covariate shift: Changes in the distribution of the input variables.
- Prior probability shift: Changes in distribution of the target variable (class variable).
- Concept shift (aka concept drift): Changes in context (e.g. user behaviors) that lead to changes in target concepts. Relationship between input and class variables changes.

Causes of dataset shift

- Sample selection bias: discrepancy in distribution is due to the fact that the training examples have been obtained through a biased method, and thus do not represent reliably the operating environment where the classifier is to be deployed. Different types of sample selection bias: Missing completely at random (MCAR), Missing at random (MAR), Missing not at random (MNAR), and Missing at random-class (MARC).
- Non-stationary environments: appears when the training environment is different from the production one, whether it is due to a temporal or a spatial change. It commonly appears, among others, in adversarial classification problems such as spam detection and fraud detection.

Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla and Francisco Herrera, "A Unifying View on Dataset Shift in Classification", Pattern Recognition, vol. 45, no. 1, pp. 521-530, 2012.

(Pears 2014) Detecting Concept Change in Dynamic Data Streams

The paper proposes SEQDRIFT2, an approach that uses reservoir sampling to build a sequential change detection model that offers statistically sound guarantees on false positive and false negative rates but has much smaller computational complexity and a smaller false detection rate than ADWIN. Improvement over SEQDRIFT1 is that it uses a more sensitive detection threshold with a reservoir sampling approach to managing data in the detection window that makes it much more sensitive to detecting changes.

Nice definition of evaluation measures

- Detection delay: Detection delay can be expressed as the distance between c and m, where c is the instance at which the change occurred, and m is the instance at which change is detected.
- False positive rate: The false positive rate is the probability of falsely rejecting the null hypothesis for a given test.
- False negative rate: The false negative rate is the probability of falsely accepting the null hypothesis when it is in fact true.

- Processing time: Processing time is the time taken by the change detector in performing hypothesis testing to detect possible concept changes in the given stream segment.

Includes full pseudocode. Compared against SEQDRIFT1, ADWIN, PHT and EWMA.

Pears, R., Sakthithasan, S. and Koh, Y.S., 2014. Detecting Concept Change in Dynamic Data Streams. Machine Learning, 97(3), pp.259-293.

(Pesaraghader 2016) Fast Hoeffding Drift Detection Method for Evolving Data Streams

The paper introduces the Fast Hoeffding Drift Detection Method (FHDDM) which uses a sliding window and Hoeffding's inequality. FHDDM detects a drift when a significant difference between the maximum probability of correct predictions and the most recent probability of correct predictions is observed. Experimental results confirm that FHDDM detects drifts with less detection delay, less false positives and less false negatives, when compared to the state-of-the-art.

False positive, false negative and detection delay are used as evaluation measures for drift detection methods.

The method works based on the fact that the accuracy of a classifier should increase or stay steady as more instances arrive; otherwise it implies the existence of drift points in the stream.

Datasets used are available at http://moa.cms.waikato.ac.nz/datasets/.

Pesaranghader, A. and Viktor, H.L., 2016, September. Fast Hoeffding Drift Detection Method for Evolving Data Streams. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 96-111). Springer, Cham.

(Rabanser 2019) Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift

The paper investigates methods for detecting and characterizing distribution shift. In particular, it investigates shift detection through the lens of statistical two-sample testing: test the equivalence of the source distribution (from which training data is sampled) and target distribution (from which real-world data is sampled). While this is very easy for simple univariate distributions, it is an open question for high-dimensional data. In the method they propose they first apply a dimensionality reduction (DR) technique which yields a representation, either uni- or multi-dimensional, and either continuous or discrete, depending on the method. They then apply a statistical hypothesis test that is appropriate for each representation.

States that "best practices for detecting shift in high-dimensional real-world data have not yet been established," citing TensorFlow as an example: TensorFlow's data validation tools compare only summary statistics of source vs target data: https://tensorflow.org/tfx/data_validation/get_started#checking_data_skew_and_drift

Shift detection techniques

- Dimensionality reduction
    - Principal Components Analysis (PCA)
    - Sparse Random Projection (SRP)
    - Autoencoders — Trained and Untrained (TAE and UAE)

    - Label Classifiers — Black Box Shift Detection (BBSD) — using softmax outputs (BBSDs) or hard-thresholded predictions (BBSDh) for subsequent two-sample testing
    - Domain Classifier
- Statistical hypothesis testing
    - Multivariate Kernel Two-Sample Tests: Maximum Mean Discrepancy (MMD)
    - Multiple Univariate Testing: Kolmogorov-Smirnov (KS) Test + Bonferroni Correction
    - Categorical Testing: Chi-Squared Test
    - Binomial Testing

Run a considerable number of experiments for combinations of dimensionality reduction and statistical hypothesis testing.

Inserted shift in datasets (MNIST and CIFAR-10). For each shift type they explored three levels of shift intensity (magnitude of added noirs) and various percentages of affected data. They explored the following types of shift: adversarial, knock-out, gaussian noise, image, image + knock-out, only-zero + image, original splits, domain adaptation datasets.

Findings

- Multiple univariate testing seems to offer comparable performance to multivariate testing
- In the multiple-univariate testing case (and overall), BBSDs was the best-performing DR method
- In the multivariate-testing case, UAE performed best

Experiment pipeline code available at https://github.com/steverab/failing-loudly

Stephan Rabanser, Stephan Günnemann and Zachary Lipton, "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift", Advances in Neural Information Processing Systems (NIPS), pp. 1396-1408, 2019.

(Read 2018) Concept-Drifting Data Streams are Time Series; The Case for Continuous Adaptation

A general assumption in most literature is that instances are independently distributed in a data stream. The paper shows that in the context of concept drift this assumption is contradictory, and that the presence of concept drift necessarily implies temporal dependence; and thus some form of time series.

Main mechanisms to deal with concept drift are (1) forgetting mechanisms (e.g., kNN and batch-incremental ensembles) and (2) detect and reset mechanisms (e.g., Hoeffding Adaptive Tree (HATs) and Hoeffding Trees (HT)-ensembles with ADWIN). The paper proposes a third approach of *continuous adaptation* to deal with concept drift, where knowledge (e.g., a set of parameters determining a decision boundary) is transferred as best as possible to a newer/updated concept rather than discarded or reset as is currently the case with the popular detect and reset approach. They propose to enact the approach using SGD (stochastic gradient descent), more specifically something called PBF-SGD. The goal of this approach is not to detect drift, but rather track it over time. A problem that they point out with detection is that detectors will fire when the error signal has already shown a significant change, by which time many (possibly very biased) predictions may have been made. This is very consistent with the motivation for this project. Because concept drift can be seen as a time series the goal of the approach is to attempt to forecast and track the drift and adapt continuously.

Similar to other paper, it distinguishes between sudden/abrupt, incremental and gradual drift (i.e., like incremental but in a stochastic way), additionally noting the possibility of reoccurring drift which may involve any of these types.

The paper makes an argument that streaming data for machine learning applications should be treated as time series data, which is an argument for continuous retraining. It might be worthwhile to look at how they're modeling the input data as time series, since this is a pretty similar idea to the idea of building regression models for predicting how a distribution changes with respect to new observations.

In case it is relevant, they used the *Electricity* and *CoverType* datasets. Electricity contains 45,312 instances, with 6 attributes describing an electricity market (the goal is to predict the demand). CoverType contains 581,012 instances of 54 attributes, as input to predict one of seven classes representing forest cover type. Data is available at https://moa.cms.waikato.ac.nz/datasets/

Read, J., 2018. Concept-Drifting Data Streams are Time Series; the Case for Continuous Adaptation. arXiv preprint arXiv:1810.02266.

(Ross 2012) Exponentially Weighted Moving Average Charts for Detecting Concept Drift

The paper proposes ECDD (EWMA for Concept Drift Detection), a single-pass (points from the data stream processed only once and discarded rather than stored in memory) method for detecting concept drift which uses an exponentially weighted moving average (EWMA) chart to monitor the misclassification rate of a streaming classifier. Approach is modular and can be run in parallel with any underlying classifier to provide an additional layer of concept drift detection. A control limit parameter used in EWMA is used to control the rate of false positive detections.

Experiments mostly targeted at detection of abrupt drift using the GAUSS and SINE synthetic datasets, as well as an Electricity Market (MOA) and a colonoscopic imaging data set as the real-world data sets. Created a second variant, ECDD-WT, which uses a warning threshold (WT) in addition to the alert threshold to provide more control over the number of false positive detections.

Ross, G.J., Adams, N.M., Tasoulis, D.K. and Hand, D.J., 2012. Exponentially Weighted Moving Average Charts for Detecting Concept Drift. Pattern Recognition Letters, 33(2), pp.191-198.

(Sakthithasan 2013) One Pass Concept Change Detection for Data Streams

The paper presents OnePassSampler, a concept drift detection method based on a sequential hypothesis testing strategy based on the use of the Bernstein bound as a test statistic. It shows improvements over ADWIN2 by reducing its false positive rate and computational overhead. The low computational complexity comes from avoiding multiple scans on its memory buffer by sequentially processing data. However, experiments show that it has a higher detection delay time for certain cases.

In summary, the way it works is that it compares the distribution of two consecutive blocks. If the distributions are the same, then the two blocks are concatenated, and compared to the next one. This continues until the distribution are different, at which point drift is detected. "At all testing points equal sized samples are used to compare the sample means from the two sides of the window. The use of random sampling accelerates the process of the computation of the sample mean while maintaining robustness. The use of the averaging function helps to smooth variation in the data and makes OnePassSampler more robust to noise than ADWIN2 ... While the use of random sampling ensures that sample means can be computed efficiently, a memory management strategy is required to ensure efficient use of memory as the left sub-window has the potential to grow indefinitely during periods of long stability in the stream."

Sakthithasan, S., Pears, R. and Koh, Y.S., 2013, April. One Pass Concept Change Detection for Data Streams. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2013) (pp. 461-472). Springer, Berlin, Heidelberg.

(Sethi 2017) On the Reliable Detection of Concept Drift from Streaming Unlabeled Data

The paper presents Margin Density Drift Detection (MD3), an algorithm that tracks the number of samples in the uncertainty region (margin) of a classifier, as a metric to detect drift from unlabeled data. It is an unsupervised approach for detecting concept drift that reduces number of false alarms. Paper shows results that are comparable to supervised approaches. In general, the approach generalizes the notion of margin density, as a signal to detect drifts. Method requires an oracle to confirm drift. Once drift is confirmed, model is retrained with samples labeled by the oracle. Experiments were run several using several drift-induced datasets. Results show a high detection rate and prediction performance, coupled with a low false alarm rate.

The paper differentiates between explicit/supervised and implicit/unsupervised detectors: "Explicit drift detectors rely on labeled data to compute performance metrics such as Accuracy or F-measure, which they can monitor online over time. They detect drop in performance and as such are efficient in signaling change when it matters. Implicit drift detectors rely on properties of the unlabeled data's feature values, to signal deviations. They are prone to false alarms, but their ability to function without labeling makes them useful in applications where labeling is expensive, time consuming or not available." Table 1 has a good list of both types of detectors.

Sethi, T.S. and Kantardzic, M., 2017. On the Reliable Detection of Concept Drift from Streaming Unlabeled Data. Expert Systems with Applications, 82, pp. 77-99.

(Shafaei 2019) A Less Biased Evaluation of Out-of-Distribution Sample Detectors

The paper introduces OD-test, a three-dataset evaluation scheme for detecting out-of-distribution samples for deep neural networks (i.e., whether any given input belongs to the population distribution of the training/evaluation data). It applies a reject function to each sample that makes a binary decision on OOD or not before passing on the sample to the model. The three datasets (and their distributions) are the training, outlier, and a third target dataset to measure whether a method can actually detect outliers that are outside of both of the previous two. Evaluated a large set of existing OOD detections methods using these three data sets.

Replication package available at https://github.com/ashafaei/OD-test.

Shafaei, A., Schmidt, M. and Little, J.J., 2019. A Less Biased Evaluation of Out-of-Distribution Sample Detectors. In British Machine Vision Conference (BMVC 2019). https://arxiv.org/abs/1809.04729

(Storkey 2013) When Training and Test Sets Are Different: Characterizing Learning Transfer

This paper provides a more comprehensive set of types of drift, although there is overlap with many of the papers. Defines a model prediction as P(y|x).

- Simple Covariate Shift is when only the distributions of covariates x change and everything else is the same. A typical example of covariate shift occurs in assessing the risk of future events given current scenarios.
- Prior Probability Shift is when only the distribution over y changes and everything else stays the same. A popular example is filtering of spam email.
- Sample Selection Bias is when the distributions differ as a result of an unknown sample rejection process.
- Imbalanced Data is a form of deliberate dataset shift for computational or modeling convenience. Happens when one or more classes are very rare compared with the others.
- Domain Shift involves changes in measurement.
- Source Component Shift involves changes in strength of contributing components.

In this classification it seems to me that the author is mixing types of drift with causes for drift. That said, the paper makes an interesting connection between dataset shift and transfer learning because the latter is basically adapting models (and data) to drift conditions which represent different scenarios.

Storkey, A., 2009. When Training and Test Sets are Different: Characterizing Learning Transfer. Dataset Shift in Machine Learning, 30, pp. 3-28.

(Wang 2018) A Systematic Study of Online Class Imbalance Learning with Concept Drift

The paper provides a review of current research progress in understanding and addressing concept drift in class-imbalanced data streams.

Compares six approaches

- Drift Detection Method for Online Class Imbalance (DDM-OCI)
- Linear Four-Rate (LFR)
- Prequential Area Under the ROC Curve Page–Hinkley (PAUC-PH)
- OOB
- RLSACP
- ESOS-ELM

Used SINE1 and SEA to produce imbalanced data streams containing three types of concept drift: 1) a change in prior probability P(y); 2) change in class-conditional pdf p(x|y); and 3) change in posterior probability P(y|x).

Not surprisingly, experiments show that different approached are better at detecting different types of drift. OOB appears to be the one that better detects all three types of drift.

Wang, S., Minku, L.L. and Yao, X., 2018. A Systematic Study of Online Class Imbalance Learning with Concept Drift. IEEE Transactions on Neural Networks and Learning Systems, 29(10), pp. 4802-4821.

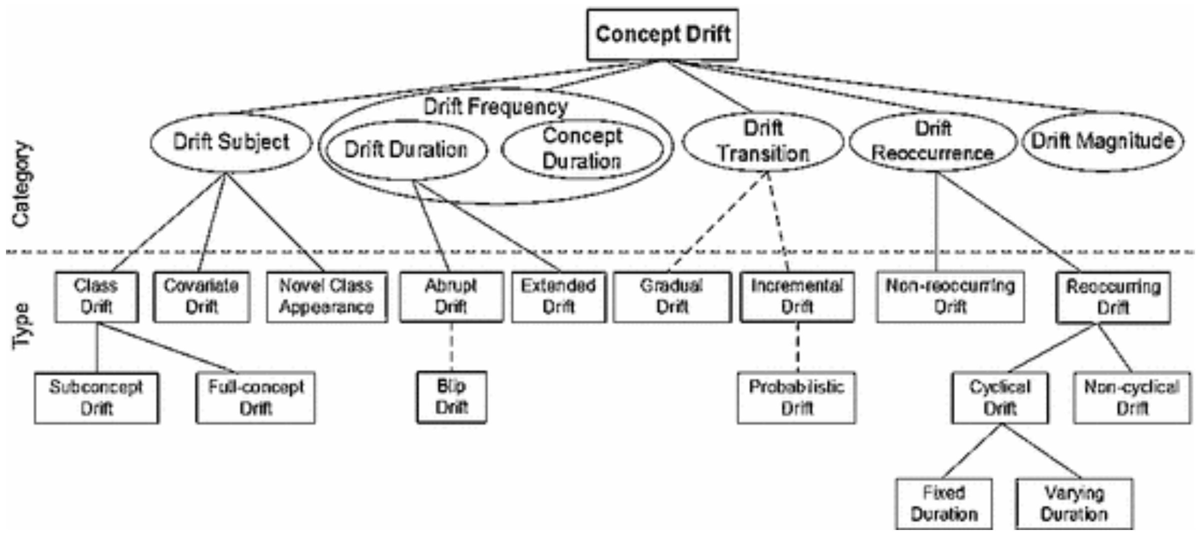
(Webb 2016) Characterizing Concept Drift

This was one of the key papers for putting together the proposal.

Similar to others, they define *concept drift* as a change in distribution over time. Concept drift is the stream learning counterpart of dataset shift from batch learning. They define several measures of concept drift

- Drift Magnitude: Degree of difference between two points of time, further defined as the distance between concepts at the start and end of the period of drift.
- Drift Duration: Elapsed time over which a period of drift occurs.
- Path Length: Length of the path that the drift traverses during a period of drift, further defined as the cumulative deviation observed during the period of drift. Provides another means of quantifying differences between drifts.
- Drift Rate: How fast the distribution is changing at time t.

Key to this paper is the taxonomy of drift categories (Figure 2). Solid lines indicate that the child concepts form a complete set of mutually exclusive alternatives. For example, Drift Duration must be either Extended or Abrupt. Dashed lines mean that the child concepts are not mutually exclusive.



- Drift Subject
    - Class Drift: Also called real concept drift or prior probability shift, occurs when the posterior class probabilities P(Y|X) change over time. For example, tax law may change over time and hence the set of tax-payer attributes that are associated with the class *compliant* will change.
        - Subconcept Drift: Also called intersected drift, it is where the drift scope is limited to a subspace of dom(X). For example, a data stream that deals with financial records may have a class called *fraud*, among others. If a new form of fraud is developed, the conditional probabilities of the fraud class occurring will change, but only for those contexts that relate to the new form of fraud. Whilst this is happening, cases that involve other forms of fraud could remain the same and may continue in the same way as before.

        - Full Concept Drift: Also referred to as severe drift, involves the posterior class distribution changing for all types of objects.
    - Covariate Drift: Also called virtual concept drift, occurs when the distribution of non-class attributes P(X) changes over time. For example, in a business that uses socio-economic factors to make predictions about customers, over time the demographics of the customer base may change, leading to a change in the probability of each demographic factor.
    - Novel Class Appearance: A new class comes into existence. For example, in a business that predicts which option a user will select on a web page, if a new option is added to the web page then a new class is introduced.
- Drift Frequency: How often concept drifts occur over a defined period of time. A high frequency indicates that new concept drifts start within a short amount of time from each other, whereas a low frequency indicates that there are long intervals between drifts.
    - Drift Duration
        - Abrupt Drift: Abrupt drift, or sudden drift occurs when a stream with concept a suddenly changes to concept a+1. A real-world example of abrupt drift could be a market crash. In a stock market stream, almost instantly, stock values will change and follow a pattern different to previously.
            - Blip Drift: Blip drift is a special case of abrupt drift coupled with very short concept duration. In blip drift, the blip concept replaces the dominant concept for a very short period of time
        - Extended Drift: Opposite of abrupt drift. A real-world example of extended drift could be a recession. Unlike a market crash, stock values at the beginning of a recession will slowly change over an extended period of time. Eventually, the changes in all of the stock values will follow a different pattern to before the recession started.
    - Concept Duration
- Drift Transition:
    - Gradual Drift: Gradual progression of small changes
    - Incremental Drift: Steady progression from one concept to the next where at time step the distance between them increases. An example is where credit card fraud patterns change over time. Consider the introduction of RFID chips in credit cards. The new technology changes the types of fraud that can be committed. As more credit card customers get new cards with RFID chips, the new types of fraud become more common until everyone has new cards, where the concept drift in the transaction stream would stop.
        - Probabilistic Drift: Occurs when there are two alternating concepts such that one initially predominates and over time the other comes to predominate. An example is sensor network node. Probabilistic drift would occur in a stream when a sensor network node is replaced. A node cannot be immediately swapped; the new node must be tested to ensure it is working correctly. As such, the two nodes will be turned on and off while the tests are conducted. In terms of the sensor network stream, samples from two different concepts are flowing from that node, one from the faulty node and another from the new node. The data from the new node will become more likely to occur in the stream, until only the concept from the new node is present.

- Drift Reoccurrence: When drift occurs, the new concept may either be one that has not previously appeared in the data stream or may be a recurrence of a pre-existing concept. The latter is known as drift recurrence. A real-world example of recurring drift could be in the use of a phone app. A user using a particular app could use it in a certain way when they are at home compared to how they use it when they are at work. The recurring concepts would be the use at home and the use at work. These concepts of app use would recur whenever a user arrives at home or at work.
    - Non-Reoccurring Drift:
    - Reoccurring Drift:
        - Cyclical Drift: Form of recurring drift that occurs when two or more concepts recur in a specific order. An example is the weather patterns in a city. Assuming no climate change is present, meteorologists can expect the patterns in the weather to recur at particular times of the year when compared to previous years. The concepts could be defined as the four seasons, with incremental drift occurring between each season. This cycle in the drift would renew itself at the start/end of each year.
            - Fixed Duration: Every period of drift occurs for a fixed amount of time. For example, an athlete repeating a set of exercises with rest breaks between each set. The transition of their vital signs between resting and exercising and vice versa will have a fixed transition time.
            - Varying Duration: Periods of drift occur for a variable amount of time.
        - Non-Cyclical Drift: Concepts do not occur in a specific order.
- Drift Magnitude: The reason why this is important is because it affects how drift should be handled. If there is abrupt minor drift then it is likely to be appropriate to retain a model that is accurate for concept *a* and to just refine it as evidence is gathered about concept a+*1*. In contrast, if there is abrupt major drift then it might be best to simply abandon the previous model and start afresh with the evidence about the nature of the new concept as it becomes available.

Drift predictability influences the selection of algorithms; affects the performance of algorithms that seek to anticipate drift; and informs the design of algorithms that will operate in environments with predictable drift, such as cyclical drift governed by seasonality. The only types of drift that are inherently predictable are fixed frequency, concept duration, drift duration, concept onset and drift onset concept drift. These are predictable because there is a constant involved in each that might be inferred. For example, drift due to time of day has a regular cycle and hence many aspects of it are relatively predictable. In contrast, the onset, duration and magnitude of drift due to a stock market crash is relatively difficult to predict.

Code for the generation of perturbed data sets for the study is available at http://dx.doi.org/10.5281/zenodo.35005.

Webb, G.I., Hyde, R., Cao, H., Nguyen, H.L. and Petitjean, F., 2016. Characterizing Concept Drift. Data Mining and Knowledge Discovery, 30(4), pp.964-994.

(Xie 2017) A Selective Transfer Learning Method for Concept Drift Adaptation

The paper proposes a method called Selective Transfer Incremental Learning (STIL) to deal with concept drift adaptation.

Concept drift is typically handled via sliding windows, drift detection, or ensemble methods. Sliding window maintains the most recent examples to assist learning from current data, while drift detection will trigger to rebuild the current model once concept drift is detected in the data stream. Ensemble methods preserve some of the historical knowledge learned from the past data and use it to boost the learning of new data.

In ensemble methods, there is a popular algorithm called chunk-based ensemble. Chunk-based ensemble means that every model in the ensemble is trained from one chunk of data in the data stream. Once the number of models in the ensemble exceeds the limit, one of the existing models is replaced by the new model. Transfer-based ensemble learning (TransferIL) transfers the preserved historical models using the current data chunk first, instead of using them directly in the ensemble, to adapt the models to the new circumstance. A problem with this approach is that he transfer operation is time consuming.

STIL will not transfer all the preserved historical models as TransferIL does. Instead, it uses a selection policy that selects which models do not make sense to transfer: the less relevant model with the new incoming data and the less transfer-effective historical model.

Xie, G., Sun, Y., Lin, M. and Tang, K., 2017, June. A Selective Transfer Learning Method for Concept Drift Adaptation. In International Symposium on Neural Networks (pp. 353-361). Springer, Cham

(Yu 2019) Concept Drift Detection and Adaptation with Hierarchical Hypothesis Testing

The paper presents a hierarchical hypothesis testing (HHT) framework that can detect and also adapt to various concept drift types (e.g., recurrent or irregular, gradual or abrupt), even in the presence of imbalanced data labels. The Hierarchical Linear Four Rates (HLFR) drift detector is implemented as part of the HHT framework.

HHT features two layers of hypothesis tests. Layer 1 is executed online and indicates potential drift. Once potential drift is detected, the Layer 2 test is activated to confirm (or deny) the validity of the suspected drift (see Figure 2 for a clear visual explanation).

Table 1 one shows the different datasets used (SEA, Checkerboard, Hyperplane, USENST1) and their properties, which are whether or not they include gradual drift, abrupt drift, recurrent drift, imbalance, and high-dimensional.

Paper also uses and reports on detection delay which is also one of our evaluation metrics.

Yu, S., Abraham, Z., Wang, H., Shah, M., Wei, Y. and Príncipe, J.C., 2019. Concept Drift Detection and Adaptation with Hierarchical Hypothesis Testing. Journal of the Franklin Institute, 356(5), pp.3187-3215.

(Zhou 2019) A Framework to Monitor Machine Learning Systems Using Concept Drift Detection

The paper presents a system deployed at Mastercard to monitor concept drift in production. Figure 2 shows a very nice depiction of types of drift.

The framework applies anomaly detection techniques to monitor model inputs and outputs independently, to ultimately detect concept drift. Figure 3 shows the "architecture" of the system. There is an Input Tracer that monitors the inputs and an Output Tracer that monitors the outputs. Both tracers, even though they use different anomaly techniques, were designed in the same way: (1) select baseline for comparison, (2) select comparison measure, design and (3) develop flagging method.

Implements some of the drift detection methods that we have identified, such as KL divergence, Jeffrey's divergence, and squared Hellinger distance.

Zhou, X., Faro, W.L., Zhang, X. and Arvapally, R.S., 2019, June. A Framework to Monitor Machine Learning Systems Using Concept Drift Detection. In International Conference on Business Information Systems (pp. 218-231). Springer, Cham