Concept drift detection for distributed multi-model machine learning systems

Beverly Abadines Quon
Electrical Engineering and Computer Science Department
University of California, Irvine
Irvine, USA
babadine@uci.edu

Jean-Luc Gaudiot
Electrical Engineering and Computer Science Department
University of California, Irvine
Irvine, USA
gaudiot@uci.edu

2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC) | 978-1-6654-8810-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/COMPSAC54236.2022.00168

***Abstract*—Many works focus on optimizing machine learning models during their training phase, but fail to account how these models adapt into their model-serving phase once they are deployed into real world applications. In this phase models must process through streams of data that can evolve over time and distort the relationship between incoming data, causing concept drift. This paper proposes leveraging the advantages of emerging features stores in order to improve concept drift detection on unlabeled, dynamic data streams across multiple models. Firstly, we introduce Drift Detection on Distributed Datasets (QuaD), which combines classical drift detectors to make use of labeled and unlabeled data, and create local context (i.e. per live model) and global context (i.e. across multiple models). Secondly, we propose using feature store entities, SHAP values, and Collaborative Filtering (CF) to augment unlabeled data across multiple models. To the best of our knowledge, QuaD is the first work that examines the collective behavior of concept drift across multiple models and discerns associations between models that may share a susceptibility in a dynamic setting. QuaD uses a combination of performance-based and data distribution-based drift detectors and CF to capture varying types of concept drifts for labeled and unlabeled data streams and is modeled around the data abstraction provided by emerging feature stores.**

I. Introduction

Many works focus on optimizing machine learning (ML) models during their training phase, but fail to account how these models adapt into their model-serving phase once they are deployed into real world applications (e.g. online sentiment analysis, intrusion detection, fraud detection, etc). In this phase models must process through streams of data that can evolve over time and distort the relationship between incoming data, $\boldsymbol{X}$, and target variables (e.g. class labels for classification, regression, or unsupervised problems), $\boldsymbol{y}$. If left unaccounted, models that performed optimally prior to this change ceases to be optimal, despite the fact that the model itself is unmodified, and results in the phenomena known as *concept drift*. Concept drift is defined as an unexpected change in the *context* or joint distribution of $P(\boldsymbol{X},\boldsymbol{y})$, such that $P_t(\boldsymbol{X},\boldsymbol{y}) \neq P_{t+1}(\boldsymbol{X},\boldsymbol{y})$ for time t.

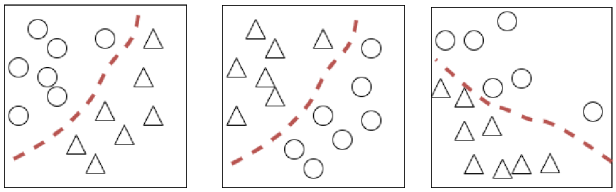
Many forms of drift detection and recovery models have been proposed to mitigate concept drifts for online data streams and can be categorized as performance-based and data distribution-based approaches. Designing such approaches is non-trivial as there is a trade-off between performance and cost-efficiency [1]. Performance-based approaches monitor a model's performance measurements, such as accuracy, *F*-measure, precision, and recall. They have the advantage of detecting all types of drift (e.g. gradual, incremental, abrupt, fixed space, or non-fixed space), but can only process labeled data, which is cost inefficient. Expecting most of the data to be labeled is impractical and expensive in terms of the scale of ML applications. Rather than measuring classifier performance metrics, data distribution-based approaches track changes in location, density, and range of the data itself. These approaches have the advantage of being able to process both labeled and unlabeled data, but are limited in the types of drift they can detect. For example, they cannot detect fixed space drift for unlabeled data without combining multiple approaches together. Hence, the trade off is that the ability to detect all types of drift is dependent on whether unlabeled data or labeled data are used.

Interestingly, these drift detectors only demonstrate how to react to drift acting on a single model and/or single pair of source and target stream. They do not take into account the possibility of multiple live models acting on different streams. In the real world setting, multiple models can run simultaneously across streams and share subsets of training data. This idea of managing offline training data and online streams for multiple models and creating logically centralized features based on physically distributed data has been gaining popularity, so much so that it has given rise to several feature stores [2]–[4]. The purpose of these feature stores are to create an abstraction layer between the offline and online data and promote data reuse by removing data silos among models, reproducibility in training data, and mitigate training-serving skews. Additionally, if feature stores are for maintaining and deploying ML models in production, then SHAP values are a means to promote explainable ML. SHAP values apply cooperative game theory to distinguish each feature's contribution to a model.

This paper proposes leveraging the advantages of emerging features stores in order to improve drift detection on unlabeled, dynamic data streams across multiple ML models. The purpose

This work was supported by Graduate Assistance in Areas of National Need (GAANN) under award P200A10052 and partially supported by the National Science Foundation under award CCF-176379.

978-1-6654-8810-5/22/$31.00 ©2022 IEEE
DOI 10.1109/COMPSAC54236.2022.00168



(a) Original distribution (b) Class distribution changed (c) Input sample distribution changed

Fig. 1: Examples of concept drift compared to (1(a)) either due to (1(b)) changes in class distribution (i.e. fixed space) or (1(c)) changes in $P(\boldsymbol{X})$, (i.e. non-fixed space). Shapes indicate different classes of data.

of this work is two-fold.

Firstly, we introduce Drift Detection on Distributed Datasets (QuaD), which combines classical drift detectors to make use of labeled and unlabeled data, and create local context (i.e. per live model) and global context (i.e. across multiple models). Secondly, we propose using feature store entities, SHAP values, and Collaborative Filtering (CF) to augment unlabeled data across multiple models.

This paper is organized as follows. Section II reviews background information related to drift detection methods, feature store, SHAP values, and CF. Section III describes our framework, QuaD. Section IV discusses metrics to evaluate our framework and Section V details future work.

II. Related Work

A. Concept Drift

The goal of ML models is that given a set of input features $\boldsymbol{X} \in \mathcal{R}$, predict a target variable $\boldsymbol{y} \in \mathcal{R}$ for regression tasks (or classes for classification tasks) [5]. The prediction of y is dependent on the prior probability of p($\boldsymbol{y}$) and of $P(\boldsymbol{X}|\boldsymbol{y})$. Using Bayesian Decision Theory, the prediction of $\boldsymbol{y}$ given $\boldsymbol{X}$ can be represented as

$$p(\boldsymbol{y}|\boldsymbol{X}) = \frac{p(y)p(X|y)}{p(X)}, \tag{1}$$

$$p(\boldsymbol{X}) = \sum_{i=1}^{c} p(y)P(\boldsymbol{X}|\boldsymbol{y}), \tag{2}$$

Concept drift occurs when there is a statistically significant difference in (1) as the model consumes streams of online data in its model serving phase, such that $P_t(\boldsymbol{X},\boldsymbol{y}) \neq P_{t+1}(\boldsymbol{X},\boldsymbol{y})$ for time t. [1], [6]–[8]. Note that for our definition, p($\boldsymbol{X}$) may have changed or the class label has changed as shown in Fig. 1(a). Other factors can contribute to the type of drift such as the rate of drift (gradual, incremental, sudden) and change in distribution (fixed space or non-fixed space).

B. Drift detection methods

The most common forms of drift detection for single concept drift can be broken down into performance-based and data distribution-based detectors [1]. Performance-based detectors are supervised approaches that require labeled data to monitor performance metrics, such as accuracy, precision, and recall. Performance-based techniques such as DDM [9] and STEPD [10] use error rate as their performance metric. These methods utilize thresholds to indicate when drift has occurred and which samples should be used to update the model. They act on the premise that the model's error rate will decrease as samples increase so long as the stream is stationary and therefore capturing instances when data is non-stationary as is the case for concept drift.

Data distribution techniques on the other hand can operate in a semi-supervised or unsupervised approach, but their disadvantage is that they are unable to capture all forms of drift and cannot identify for concept drifts due to class distribution changes in unlabeled data as is the case in Fig (1(b)). Rather than monitoring model metrics, these techniques monitor the distribution of the data itself. They rely on clustering and density estimations to detect whether distributions are significantly statistically different [11]–[14].

C. Feature store

There is a distinction between a model's offline training phase and online serving phase. Offline training ensures the availability of a finite training set during the process. The data is often retrieved in batches and possibly used by different models or analyzed by multiple parties of data scientists and engineers. The online serving phase must work along windows of non-finite, real time data that are processed in streams. Many problems can arise between the two phases while the data layers are disconnected. Problems such as silos of redundant data and susceptibility to training-serving skews can occur. Feature stores serve as an abstraction layer between the model training data from offline store and the model serving data from online store. They can provide the benefits of removing siloed data, promoting feature resuse over rebuild, and decreasing occurrences of training-serving skews using point in time consistency.

Feature stores provide a logically centralized registry physically distributed data by creating a catalog of feature data and their metadata [3]. It is made of a hierarchy of project, feature view and the triplet (feature, entity, and data source). Data source is the raw underlying data that can be located anywhere. The entity is a collection of semantically related features. For example, a ride sharing service can have entities of values customer or driver, while both entities have a shared feature of trips taken. Feature views are made of the triplet and represents a logical grouping or *context*. Finally, updates of features can be done easily using the registry.

D. SHAP

Shapley values employ a coalition game theory to fairly distribute the contribution of features for a prediction. It takes the average marginal contribution of a feature across all coalitions aka all possible permutations. SHAP uses this notion of Shapley values to get a fair, order agnostic payout of the features. The disadvantage to SHAP is that they are

appropriate for linear models, but are costly for models with many features because of the complexity of SHAP is on the order O($2^{feature\_size}$) [15], [16].

E. Collaborative Filtering

Collaborative Filtering (CF) is often used for recommender systems. Given a set of relationship scores between users to items, it builds an association between any the relationships among user-to-items, user-to-user, or item-to-items. It works on a labeled set and is dependent on the sparsity of data. CF makes prediction on the empty user to item values based on the similarity scores it generates from performing matrix factorization or support vector machine. Modeling based on the interactions of user-to-user and user-to-items is tricky, since users themselves can change their mind. Hence, these models are susceptible to concept drift and performance-based methods have been used to track whether drift occurs.

III. Proposed Framework

A. Problem statement

Given a system where streams of data are fed into multiple models that share intersections of features, we aim to develop a method to accomplish the following:

1) Detect drift for each model (local context)
2) Augment the labels based on shared features
3) Make the holistic features of the system more resistant to drift (improve global context)

B. Detect Drift for each model

Drift Detection on Distributed Datasets (QuaD) is comprised of two classical drift detection methods, DDM and KS test. It combines both methods in order to detect varying types of concept drifts and switch between labeled and unlabeled data streams.

DDM is a performance-based method for drift detection and tests for the statistical distribution of its model's performance. It is measured by its error rate, p and standard deviation, s (3) and detects drift using thresholds, such as the warning level (4) and the drift level (5). The values, s_{min} and p_{min} are defined in the training phase and are updated if the sample, i at time t achieves (6).

$$s_t = \sqrt{p_t(1-p_t)/i} \tag{3}$$

$$p_t + s_t \geq p_{min} + 2s_{min} \tag{4}$$

$$p_t + s_t \geq p_{min} + 3s_{min} \tag{5}$$

$$p_t + s_t < p_{min} + s_{min} \tag{6}$$

For example, if the warning level is triggered at instance t_m and reaches the drift level at t_n, then the model should be retrained on the samples stored between t_m and t_n.

KS test [12], [13] can be used for data distribution-based technique for concept drift detection in streams of data. It is non-parametric in form and compares the location and shape between probability distributions, $F_{A,n}$and $F_{B,m}$ across samples A with n observations and samples B with m observations. Their empirical distribution functions are computed as:

$$F_n(t) = \frac{1}{n}\sum_{i=1}^{n} 1\{x_i \leq t\} \tag{7}$$

where $(x_1, ...x_n)$ are independent and identically distributed (i.i.d) random variables in the real numbers domain.

Concept drift can be detected when the KS test rejects the null hypothesis at α if:

$$D > c(\alpha)\sqrt{\frac{n+m}{nm}}, \tag{8}$$

D is the KS statistic (i.e. obtained p-value), $c(\alpha)$ is the confidence interval at α, and the product on the right side of the inequality is the obtained target p-value. Lastly, D is defined as:

$$D = \max_{x \subset A \cup B} |F_A(x) - F_B(x)| \tag{9}$$

C. Augment the labels based on shared features

For this scenario, we assume that the models running have shared feature dependencies. Unlike other models, we plan to employ feature views from feature stores to retrieve and update shared features between models with the combination of SHAP and CF. Analagous to the user-item relationship, we translate models as our users and feature views as our items. This will generate connections between models and features within the feature view. To generate the weight of the connection, which is expected from CF, we propose to use SHAP. SHAP will fairly distribute the weight of a feature for a prediction. From here we apply CF onto a recommender system and build predictions of whether certain models should be augmented with other features outside of their original parameters, where the new features may or may not be labeled.

D. Make the holistic features of the system more resistant to drift

We continue with our CF relationships. From here we not only analyze the model-to-feature relationship, but also the model-to-model relationship and try to discern whether some models are more susceptible to concept drift. If drift is detected on one model, the system can either 1) activate local drift detector(s) on associated models or 2) update models without measuring for drift. There is a trade-off between false alarms and delay in detection between the two actions. Updating will rely on the materialization process of feature stores.

IV. Discussion

A. Metrics

The following are characteristics to evaluate the quality of our concept drift detector.

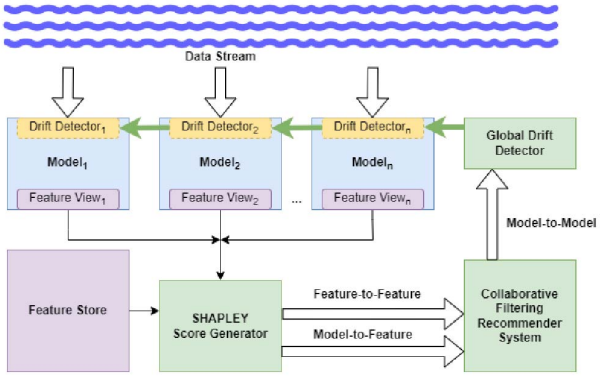


Fig. 2: Overview of QuaD in a single-stream, multiple model system.

1) Probability of true change detection: This requires synthetic data or ground truth and characterizes the capacity to detect drift occurrences.

2) Probability of false alarms: This characterizes resiliency and is equivalent to the inverse of the time to detection, which is the expected time between false-positive detections. This can be used on real data without drifts and the resulting detections are considered as false alarms.

3) Delay of detection: This estimates the number of instances required to detect a change after the actual occurence of drift. Average time to detection is used on synthetic data.

B. Evaluation

Both DDM and KS test are reactive methods towards concept drift, meaning that updates are made upon detection of drift. They do not employ forecasting to prevent concept drift from occurring in the first place. QuaD creates an ensemble of these methods and tests across multiple models that share a significant portion of their training data. The novelty of our method is the consideration of multiple models and features to form a global context and better explain concept drift. Upon detection of drift, QuaD calls for an update on that specific model *and* for models whose intersection with its training sets is significant. Thereby calling for an update before concept drift occurs on these susceptible models.

Additionally, it uses the strength of DDM to detect drifts on non-fixed space and KS to function over unlabeled data.

The constraints, however, is that multiple models must have a significant portion of their training set be shared. Moreover the rate of false positives may be high, since our method calls for an update based on association of similarly trained models. There is an assumption that updates are computationally inexpensive due to the resources (e.g. data parallelism) available in the system. This is especially true for SHAP, which generates the weight for our novel way of using CF. There are methods for model and data parallelism in ML that can be explored.

V. Conclusion and Future Work

To the best of our knowledge, QuaD is the first work that examines the collective behavior of concept drift across multiple models and discerns associations between models that may share a susceptibility in a dynamic setting. QuaD uses a combination of performance-based and data distribution-based drift detectors and CF to capture varying types of concept drifts for labeled and unlabeled data streams and is modeled around the data abstraction provided by emerging feature stores.

Developing QuaD will require frameworks, such as River [17] to enable models to run concurrently and to process data as streams. Metrics mentioned in Section 4 should be used to evaluate QuaD's performance and reliability.

Future work can explore if association of models based on feature store values can benefit from other drift detectors. For example, DDG-DA [18] forecasts drift by generating datasets based on a sampling of historical data instead of the most recent data. The notion of generating synthetic data may relax the constraint of relying on shared datasets between models.

References

[1] H. Hu, M. M. Kantardzic, and T. S. Sethi, "No free lunch theorem for concept drift detection in streaming data classification: A review," *WIREs Data Mining Knowl. Discov.*, vol. 10, no. 2, 2020. [Online]. Available: https://doi.org/10.1002/widm.1327

[2] "Hopsworks: data-intensive ai with a feature store," https://github.com/logicalclocks/hopsworks, accessed: 2021-09-30.

[3] "Feast," https://github.com/gojek/feast, accessed: 2021-09-30.

[4] "Iguazio," https://www.iguazio.com/feature-store/, accessed: 2021-09-30.

[5] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, 2014. [Online]. Available: https://doi.org/10.1145/2523813

[6] A. Liu, J. Lu, F. Liu, and G. Zhang, "Accumulating regional density dissimilarity for concept drift detection in data streams," *Pattern Recognit.*, vol. 76, pp. 256–272, 2018. [Online]. Available: https://doi.org/10.1016/j.patcog.2017.11.009

[7] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, 2013. [Online]. Available: https://doi.org/10.1016/j.neunet.2013.01.012

[8] F. A. Pinage, E. M. dos Santos, and J. M. P. da Gama, "Classification systems in dynamic environments: an overview," *WIREs Data Mining Knowl. Discov.*, vol. 6, no. 5, pp. 156–166, 2016. [Online]. Available: https://doi.org/10.1002/widm.1184

[9] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*, ser. Lecture Notes in Computer Science, A. L. C. Bazzan and S. Labidi, Eds., vol. 3171. Springer, 2004, pp. 286–295. [Online]. Available: https://doi.org/10.1007/978-3-540-28645-5_29

[10] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Discovery Science, 10th International Conference, DS 2007, Sendai, Japan, October 1-4, 2007, Proceedings*, ser. Lecture Notes in Computer Science, V. Corruble, M. Takeda, and E. Suzuki, Eds., vol. 4755. Springer, 2007, pp. 264–269. [Online]. Available: https://doi.org/10.1007/978-3-540-75488-6_27

[11] X. Song, M. Wu, C. M. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, P. Berkhin, R. Caruana, and X. Wu, Eds. ACM, 2007, pp. 667–676. [Online]. Available: https://doi.org/10.1145/1281192.1281264

[12] D. M. dos Reis, P. A. Flach, S. Matwin, and G. E. A. P. A. Batista, "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds. ACM, 2016, pp. 1545–1554. [Online]. Available: https://doi.org/10.1145/2939672.2939836

[13] P. Sobolewski and M. Wozniak, "Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors," *J. Univers. Comput. Sci.*, vol. 19, no. 4, pp. 462–483, 2013. [Online]. Available: https://doi.org/10.3217/jucs-019-04-0462

[14] A. Dries and U. Rückert, "Adaptive concept drift detection," *Stat. Anal. Data Min.*, vol. 2, no. 5-6, pp. 311–327, 2009. [Online]. Available: https://doi.org/10.1002/sam.10054

[15] M. Sundararajan and A. Najmi, "The many shapley values for model explanation," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 9269–9278. [Online]. Available: https://proceedings.mlr.press/v119/sundararajan20b.html

[16] "Interpretable machine learning," https://christophm.github.io/interpretable-ml-book/shap.html, accessed: 2021-09-30.

[17] "Online-ml/river: nline machine learning in python,"," https://github.com/online-ml/river, accessed: 2022-01-24.

[18] W. Li, X. Yang, W. Liu, Y. Xia, and J. Bian, "DDG-DA: data distribution generation for predictable concept drift adaptation," *CoRR*, vol. abs/2201.04038, 2022. [Online]. Available: https://arxiv.org/abs/2201.04038