A Study on Metrics for Concept Drift Detection Based on Predictions and Parameters of Ensemble Model

Kei Yonekawa(✉), Shuichiro Haruta, Tatsuya Konishi, Kazuhiro Saito, Hideki Asoh, and Mori Kurokawa

KDDI Research, Inc., Saitama, Japan
{ke-yonekawa,sh-haruta,tt-konishi,ku-saitou,hi-aso,
mo-kurokawa}@kddi-research.jp

Abstract. The performance of machine learning models deteriorates when the distribution of test data changes, which is called concept drift. One way to deal with concept drift is to continuously rebuild the model. If we want to minimize the frequency of rebuilding due to some constraints, however, it is important to detect concept drift as the timing when rebuilding is truly necessary. Taking advantage of ensemble models for concept drift detection may improve the detection accuracy. However, the behavior of ensemble model's predictions and parameters in the presence of concept drift has not been fully investigated. In this study, we investigated how the ensemble models constructed by two different methods behave in the presence of concept drift. In the experiments, we monitored some metrics including the metrics that can be calculated only by the ensemble model and the metrics based on the model parameters. As a result, we found that the metrics show some behaviors that seem to be influenced by concept drift, suggesting that the detection accuracy of concept drift may be improved by using these metrics.

Keywords: Concept drift detection · Neural networks · Ensemble learning

1 Introduction

Before a machine learning model can be integrated into a real service, the model alone or the machine learning pipeline[1] is built and verified in the development environment, and then deployed to the production environment [1]. The statistical properties of the data handled in a real service may differ in the distribution of the data between the time when the model is trained and the time when the model performs inference. For example, in a news service, user preferences change over time, so when recommending articles or products, a category with a high response rate in the past may have a low response rate in the recent past [2]. This phenomenon is called concept drift, and it is one of the causes in model performance degradation [2].

[1] A series of processes including preprocessing, model training, data and model validation, and inference using the model.

T. Hara and H. Yamaguchi (Eds.): MobiQuitous 2021, LNICST 419, pp. 568–581, 2022.
https://doi.org/10.1007/978-3-030-94822-1_37

There are two approaches to deal with concept drift: one is to retrain the model continuously, and the other is to retrain the model only when the statistical properties of the data change significantly [3]. In order to implement the former approach, the following conditions must be met: the labels necessary for training must be constantly available, and the possibility of production deployment of the retrained model must be automatically determined. On the other hand, in the latter approach, labels need to be obtained only when they are needed. Besides, in the latter approach, the retrained model can be manually audited, so that the validity of the inference basis can be verified in terms of fairness and causality before the decision to deploy is made. In this paper, we will deal with the latter case, assuming that the conditions for implementing the former are not met. In this case, it is important to detect concept drift as the timing when retraining is truly necessary.

One approach to concept drift detection is based on the change in model parameters when the model is updated with new data [13], which has potential for development in terms of explainability of concept drift and other aspects.

As a related field to concept drift detection, in the field of outlier detection, it has been proposed to increase the reliability of detection by using an ensemble of multiple methods [4]. As for ensembles of neural networks, it has been reported that predictions tend to be agreed upon among ensemble members depending on how the members are created [5]. Thus, it is possible that concept drift may be missed by simply monitoring the output of the model.

Taking advantage of ensemble models for concept drift detection may improve the detection accuracy. However, ensemble models may not be able to fully exploit the mutual agreement rate among ensemble members [12] or the uncertainty in the model output [20] because the predictions may be similar depending on how the members are composed [5]. Thus, it can be said that there are conflicting possibilities. On the other hand, model parameters retain sufficient diversity among members depending on how they are organized [5]. Thus, it can be said that ensemble models may be useful for concept drift detection if they are based on model parameters. In order to investigate these possibilities, this paper investigates how ensemble models constructed by two different methods behave in terms of predictions and model parameters in the presence of concept drift. To the best of our knowledge, this is the first work to monitor some metrics based on ensemble models under concept drift, such as cohesiveness of internal representations in the ensemble, cohesiveness of gradients in the ensemble, and so on.

2 Related Works

2.1 The Sources of Concept Drift

Let X be the input of the model and Y be the label. In the case of labeled data, the distribution of the data at a certain time t can be expressed by the joint probability $P_t(X, Y)$. This can be decomposed as $P_t(X, Y) = P_t(Y|X)P_t(X)$. Thus, there are three patterns in which the distribution of the data changes in comparison with time t and $t+1$ as follows [6]. (1) When only the input distribution changes, i.e., $P_t(X) \neq P_{t+1}(X)$ but $P_t(Y|X) = P_{t+1}(Y|X)$. This is called a covariate shift or virtual drift. (2) When only the conditional distribution changes, i.e., $P_t(Y|X) \neq P_{t+1}(Y|X)$ but $P_t(X) = P_{t+1}(X)$.

This is called the actual drift or real drift. (3) When both the input distribution and the conditional distribution change, i.e., $P_t(X) \neq P_{t+1}(X)$ and at the same time $P_t(Y|X) \neq P_{t+1}(Y|X)$.

2.2 The Types of Concept Drift

The following four patterns of data distribution change over time have been pointed out [6]. (1) A case where the old data distribution is switched to a new one in a short period of time. This case is called *sudden drift*. (2) A case where the old data distribution is switched to the new data distribution over a period of time. Both distributions are mixed in a certain ratio in the middle of the switch, and the ratio of the new data distribution gradually becomes larger. This case is called *gradual drift*. (3) A case where the old data distribution is switched to the new data distribution over time passing through an intermediate distribution. In this case, the distribution itself change continuously during the switching process. This case is called *incremental drift*. (4) A case where the distribution switches from the old data distribution to the new data distribution and then switches back to the old data distribution after a certain period of time has elapsed. This case is called *reoccurring drift*.

2.3 Two Major Approaches to Deal with Concept Drift

There are two main ways to deal with the degradation of model performance due to concept drift [3]. The first approach is to adapt to changes in the data distribution by repeatedly retraining the model using the most recently obtained data. This is called continuous rebuild. The second approach is to retrain the model only when concept drift is detected based on some indicator. This is called triggered rebuild.

The advantage of continuous rebuild is that it adapts quickly to concept drift and does not require additional computation. The disadvantages are that labels are always needed, deployment frequency is high, data and model validation need to be automated, and computational resources are always consumed for retraining.

The advantages of triggered rebuild are that labels are obtained only when retraining is required, the number of deployments can be reduced, validation of data and models does not necessarily have to be automated, and computational resources for retraining can be reduced. The disadvantages are that the adaptation to concept drift is delayed by the detection delay, and that the detection algorithm requires additional computational burden in terms of memory and CPU power to run.

2.4 Concept Drift Detection Methods

The methods of concept drift detection can be classified into three categories according to the target to be monitored [7]. The first is a method that monitors the input of the model. There are methods based on the Hellinger distance between the feature distributions during training and inference [8], and methods based on principal component analysis [9]. While these methods do not require labels and are model-agnostic, they may react to changes that are unrelated to the degradation of model performance, and require a

feature extractor when dealing with unstructured data. The second method is to monitor the output of the model. One method that requires labels is based on prediction error [10]. Methods that do not require labels include a method based on the number of cases that fall into the margin of the classifier [11], a method based on the mutual agreement rate among ensemble members [12], the method based on the distribution of the confidence in the model's output [3], and the method based on the uncertainty of the model's output [20]. While there is always a loss of accuracy or confidence in the model when these concept drift detection methods are reactive, labels are required or the model type is constrained. The third method is to monitor the parameters of the model. There are methods based on changes in the prior distribution of model parameters [13]. While the detection index is theoretically valid, it requires labels and is limited to models that can be handled by Bayesian learning.

2.5 Ensemble

In the field of outlier detection, it has been proposed to use multiple methods or ensembles of models, rather than relying on a single method or model, in order to reduce the possibility of missing outliers [4]. Compared to using a single model, uncertainty can be well quantified when using an ensemble of models [14]. Besides, the uncertainty measure calculated based on an ensemble of models can well reflect distributional shift [24]. Thus, it is expected that the accuracy of concept drift detection will be improved if we employ the uncertainty measure based on ensemble models.

For ensembles of neural networks, it has been reported that randomizing the initialization of model parameters is effective in improving performance [15]. On the other hand, there are cases where it is not practical to prepare multiple production models with different initializations of parameters to monitor the performance. In such cases, the subspace sampling technique can be applied to obtain multiple models that can be members of an ensemble from a single trained model. Subspace sampling is a method of obtaining multiple models with diversity by adding some noise to the models. The simplest method is to add random values to the model parameters. According to [5], random initialization has a larger performance gain than subspace sampling. It has also been reported that models obtained by subspace sampling tend to have similar predictions even if the weights are different [5].

3 Methods

Since we are interested in ensembles of neural networks, we use a multilayer perceptron as the base model. The number of units in each hidden layer was set to 7. For the activation function of the hidden layer, we used the Leaky ReLU [17], which has a slope of 0.01 when the input value is negative. For the activation function of the output layer, we used a sigmoid function. A bias term was provided for all units. All these choices are determined empirically.

Two methods for obtaining the ensemble members are investigated: random initialization and random subspace sampling [5]. In random initialization, the ensemble members are obtained by initializing the weights of the base model with random values and training it. In random subspace sampling, the ensemble member is obtained by adding random values to the learned main model. The random values to be added are calculated by multiplying a random unit vector by 10% or 20% of the L2 norm of the weights of the model. As the random unit vector, we obtained a random orthogonal matrix with the dimensionality of the model parameters as the number of rows and columns by the method of [18], and used its row vector. If the number of ensemble members to be created was larger than the number of dimensions of the model parameters, the random orthogonal matrix was obtained multiple times.

4 Experiments

4.1 Datasets

The dataset used in the experiment consists of five synthetic data using the scikit-multiflow package [21] and one real world data. The data used as synthetic data are MIXED, STAGGER, SINE, SEA, and AGRAWAL. In each data set, concept drift was introduced by changing the classification function (labeling rule). The number of classification functions varies depending on the dataset, but in all datasets, concept drift was introduced so that the number of classification functions cycled in ascending order (e.g., $0 \rightarrow 1 \rightarrow 2 \rightarrow 0$ in STAGGER). In terms of noise, two irrelevant features were introduced in SINE, the label flip rate was set to 10% in SEA, and the feature disturbance rate was set to 10% in AGRAWAL.

The data used as real-world data is Spambase [16]. The Spambase dataset is a dataset for a binary classification task. The target variable is the binary value of whether an e-mail is judged to be spam or not. The features include the percentage of occurrence of specific words (48 types), the percentage of occurrence of specific letters (6 types), and the mean, maximum, and sum of the lengths of consecutive capital letters. The number of samples is 4601.

Since Spambase is not stream data, concept drift needs to be artificially introduced. To artificially introduce sudden drift into Spambase, we refer to the procedure in [11]. Specifically, the features (continuous values) were normalized to the range [0, 1], and the absolute values of the correlation coefficients with the target variable (in [11], information gain was used instead) were calculated for each feature dimension and ranked. The samples are then arranged in a random order to simulate stream data. When a sudden concept drift is to be introduced, the order of the feature dimensions with the top 50% absolute values of the correlation coefficients is randomly reordered to simulate a sudden concept drift. This method makes it easier to observe the influence of concept drift on the prediction accuracy and the optimal solution of the model.

4.2 Evaluation

The evaluation was conducted in two phases: a warm-up phase and a prequential phase. The warm-up phase corresponds to a phase in which models were built in a development environment. The prequential phase corresponds to a phase in which the trained model is deployed in a production environment and trying to detect concept drift while performing inferences.

First, in the warm up phase, the training data and validation data are obtained from the data described above, and the trained main model is obtained by early stopping based on the monitoring of the validation loss. The number of samples for the training and validation data is 10000 and 2000 for the synthetic data, and 2760 and 460 for the real-world data. The base model has one hidden layer, the learning algorithm is Adam [19], the learning rate is 0.1, the batch size is 32, the number of steps per epoch is 10, the maximum number of epochs is 300, and the early stopping patience is 30 epochs.

Next, in the prequential phase, an ensemble is formed, and one batch of test data obtained from the data described above is acquired for each time step to perform prediction, performance evaluation, and model updating. During the process, a sudden concept drift was generated using the method described above. The number of samples for the test data is 25600 for the synthetic data and 1376 for the real-world data. The size of the batch to be acquired was set to 32. In the model update, the learning algorithm is Adam, and the learning rate is 0.1. The time steps when the concept drifts were introduced were 200, 400, and 600 for the synthetic data and 21 for the real-world data.

The following two methods were used to construct the ensemble: random initialization (RI) and random subspace sampling (RSS). When updating the model, all the ensemble members are updated. For the main model, two cases were investigated: updating (main model, MM) and not updating (Frozen). The number of members in an ensemble M is set to 10. Because of the randomness in the generation of data and the composition of the ensemble, the evaluation was performed 10 times using the same procedure.

4.3 Metrics

Several metrics were monitored in the prequential phase. To see how the performance changes with the concept drift introduced in each dataset, the batch-average of the loss and the accuracy per batch were calculated for each time step for Frozen, MM, RI, and RSS. These metrics were ensemble-averaged for RI and RSS (LOS and ACC, respectively).

To examine the diversity of the predictions in the ensemble, the ensemble standard deviation (SD) of the predictions was calculated and batch-averaged (YSD).

As predicted values are related to uncertainty of prediction, we also calculated the batch-average of the entropy of the ensemble-average of the predictions (ENT). Namely, $\mathrm{ENT} = -\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} p_i \log p_i$, $p_i = \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}} f_m(x_i;\theta_m)$. Here, $\mathcal{B}$ is a batch, $\mathcal{M}$ is the ensemble, and p_i is the ensemble-average of prediction by m-th model f_m in $\mathcal{M}$ parameterized by θ_m for i-th sample, whose feature is x_i.

To examine the diversity of model parameters in the ensemble, we calculated the dimension-average of the ensemble standard deviation of the model parameters (PSD). Namely, PSD $= \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} v_d$, $v_d = \sqrt{\frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}}\left(\theta_{md} - \overline{\theta_d}\right)}$, $\overline{\theta_d} = \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}}\theta_{md}$. Here, $\mathcal{D}$ is a set of dimensions in θ_m, v_d is the ensemble standard deviation of θ_{md}, which is d-th dimension of θ_m, and $\overline{\theta_d}$ is the ensemble-average of θ_{md}.

Since we suppose the diversity of model parameters is related to the internal representations, the ensemble-average of the centered kernel alignment (CKA) [23] between the internal representations of each ensemble member and those of the ensemble-average were also calculated (CKA). Namely, CKA $= \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}} linear\_cka(z_{mB}, \overline{z_B})$, $\overline{z_B} = \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}} z_{mB}$. Here, *linear_cka* is the CKA function with a linear kernel, z_{mB} is internal representations of the batch B obtained from model m, and $\overline{z_B}$ is the ensemble-average of z_{mB}. This metric indicates cohesiveness of internal representations in the ensemble. Here, CKA is considered to be a reasonable similarity measure for comparing the internal representations of neural networks [22].

We also focused on the dynamics of the model parameters. We calculated the ensemble-average of the cosine similarity between the gradient of each ensemble member and that of the ensemble-average (GCO). Namely, GCO $= \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}}\cos(g_m, \overline{g})$, $\overline{g} = \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}} g_m$. Here, g_m is the batch-average of gradient of for model m, and $\overline{g}$ is the ensemble-average of g_m. This metric indicates cohesiveness of gradients in the ensemble. We also calculated the ensemble-average of the L2 norm of the gradient (GNO). The above values were calculated for each evaluation trial, and then the mean and standard deviation were calculated for each time step.

5 Results

In the following figures, the solid line is the inter-trial mean of each metric for each time step, and the upper and lower bound of each error band are the inter-trial standard deviations. To make the graphs easier to read, the values are smoothed using moving averages. The window length of the moving average was set to 10 for the synthetic data and 3 for the real-world data.

5.1 Changes in Prediction Performance

We show how the performance changes with concept drift. Figure 1 shows LOS and Fig. 2 shows ACC. For all datasets except SEA, there is a significant increase in LOS and decrease in ACC at each time point of all drifts for Frozen, MM, RI, and RSS, indicating performance degradation due to concept drift. The exception is SEA, where the drift effects at t = 400 for Frozen and t = 200, 400 for the other models appear to be difficult to recognize.

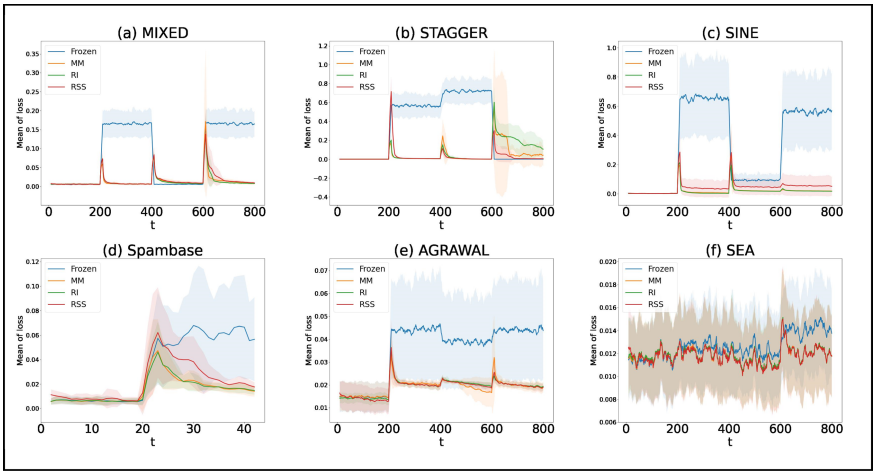

Fig. 1. LOS of each model in each dataset.

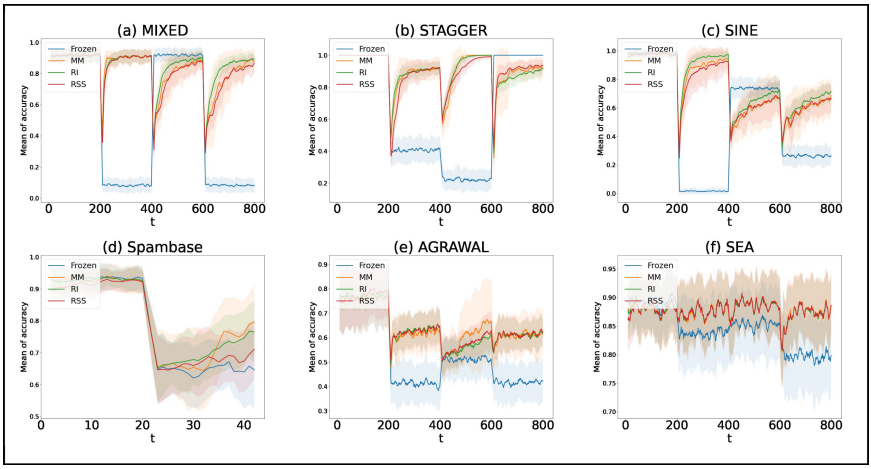

Fig. 2. ACC of each model in each dataset.

Table 1. Summary of changes of each metrics in each dataset in the case of RI.

	MIXED	STAGGER	Spambase	SINE	AGRAWAL	SEA
YSD	✓	✓	✓	✓	✓	
ENT	✓	✓	✓	✓		✗
PSD	✓	✓		✓		✗
CKA	✓	✓	✗			
GCO	✓		✓			✓
GNO	✓	✓	✓	✓	✓	

Table 2. Summary of changes of each metrics in each dataset in the case of RSS.

	MIXED	STAGGER	Spambase	SINE	AGRAWAL	SEA
YSD	✓	✓	✓	✓	✓	
ENT	✓	✓	✓	✓		✗
PSD	✓	✓	✗	✓		✗
CKA	✓	✓	✓			
GCO	✓		✓			
GNO	✓	✓	✓	✓	✓	

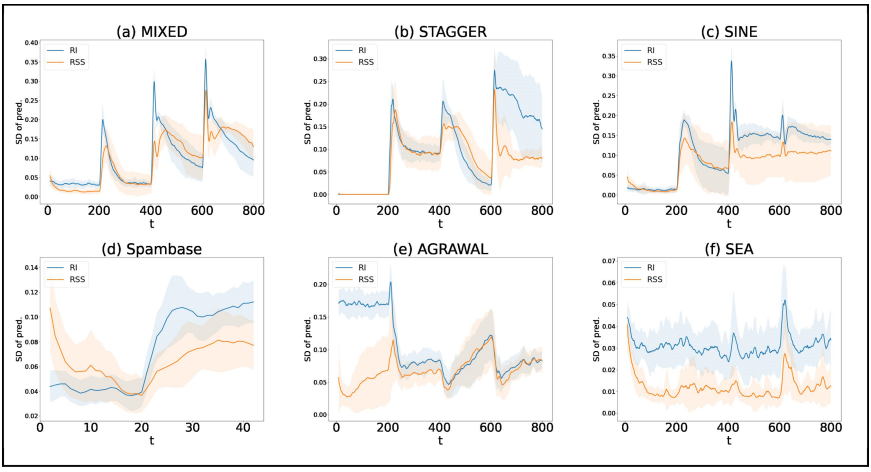

Fig. 3. YSD of each model in each dataset.

5.2 Changes in the Diversity of Predictions

We show how the concept drift changed the diversity of predictions in the ensemble. Table 1 for RI and Table 2 for RSS show how each metric changed at each drift point for each dataset. "✓"means that there was a significant change at all drift points, blank means that there was at least a minor change at some drift points, and "✗" means that the change was difficult to recognize at all drift points. The order of the datasets in the table is such that the dataset with the most ✓ is on the left. Figure 3 shows YSD. The value increases in most cases where the value changes at the drift point, indicating that the diversity of predictions has increased. The exception is AGRAWAL, which falls at t = 400, 600 for RI and RSS. In relation to the predictions, Fig. 4 shows ENT as an uncertainty measure. The value increases when the value changes at the drift time, indicating that the uncertainty of the prediction has increased.

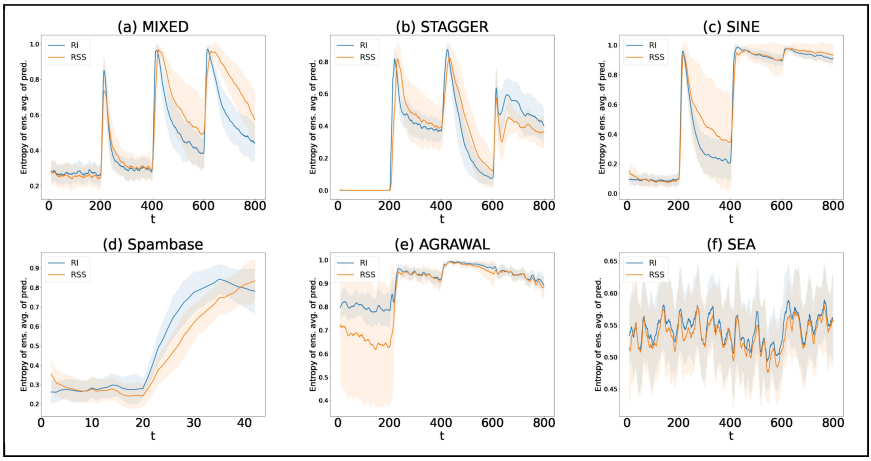

Fig. 4. ENT of prediction of each model in each dataset.

5.3 Changes in the Diversity of Model Parameters

We show how the concept drift changes the diversity of the model parameters in the ensemble. Figure 5 shows PSD. In most of the cases where the values change at the time of drift, they go down and then up, which can be interpreted as the model parameters becoming similar for a while and then diversifying. The exceptions are t = 200 for SINE, STAGGER, and MIXED, and Spambase, where the values rise without falling. In relation to the model parameters, Fig. 6 shows CKA. In most cases where the value changes at the drift point, it falls, which can be interpreted as a diversification of the internal representation. The exceptions are t = 200 for the RI of SINE and t = 600 for the RI and RSS of SEA, which show an increase. In relation to the dynamics of the model parameters, Fig. 7 shows GCO. When the values change at the drift point, there are almost the same number of upward and downward cases, and thus no consistency is observed. Figure 8 shows GNO. When the value changes at the point of drift, it is elevated and is considered to be linked to the increase in loss.

5.4 Comparisons

Prediction Error and the Other Metrics. One case where drift effects are difficult to recognize with prediction error-based metrics such as LOS and ACC is t = 400 for SEA. In this case, the metrics that show drift effects are YSD (RI), CKA (RSS), GCO (RI), and GNO (RI and RSS). It is suggested that the use of ensemble-specific and model-parameter-based metrics may improve the accuracy of concept drift detection based on prediction error.

Predictions and Model Parameters. The case where there is no significant change in the prediction-based metrics such as YSD and ENT is SEA t = 200. In this case, the metric that shows drift effects is GCO. It is desirable to clarify the mechanism of the behaviors of these metrics to create better metrics.

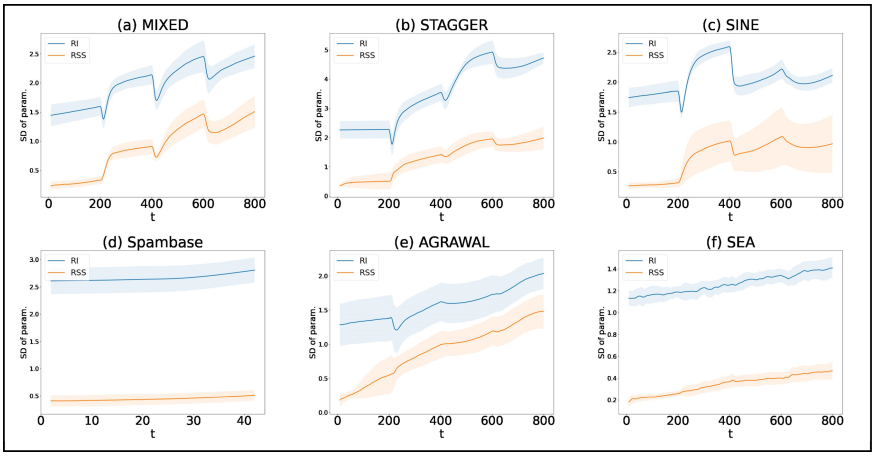

Fig. 5. PSD of each model in each dataset.

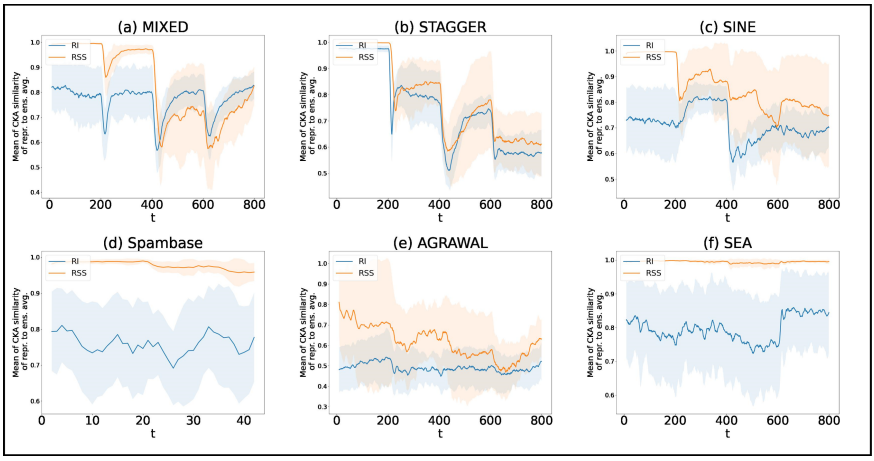

Fig. 6. CKA of each model in each dataset.

RI and RSS. RI may be more convenient because it has a higher diversity of ensemble members and shows more ensemble-specific characteristics, while RSS is more practical because its computational cost is lower than that of RI. In this paper, we discuss whether RSS can replace RI for each metric. The metrics based on predicted values are considered to be substitutable because the patterns in the Tables 1 and 2 are the same. The metrics based on model parameters are also considered to be substitutable because the overall pattern of the Tables 1 and 2 is similar. As an exception, in Spambase, it is difficult to recognize the effect of drift on PSD in RSS. On the other hand, it is difficult to recognize the effect of drift on CKA in RI. Besides, it was assumed that there would be a trade-off between the amount of computation and the prominence of the movement of the metrics. However, depending on the combination of data set and indicator, the movement of RSS

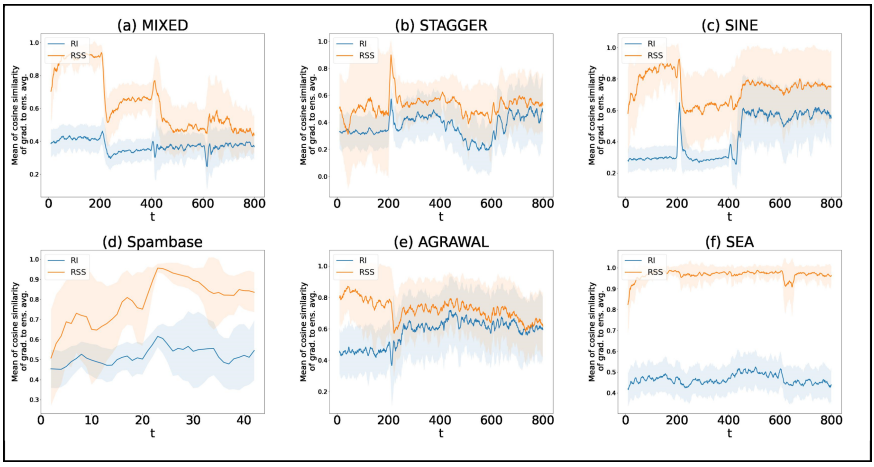

Fig. 7. GCO of each model in each dataset.

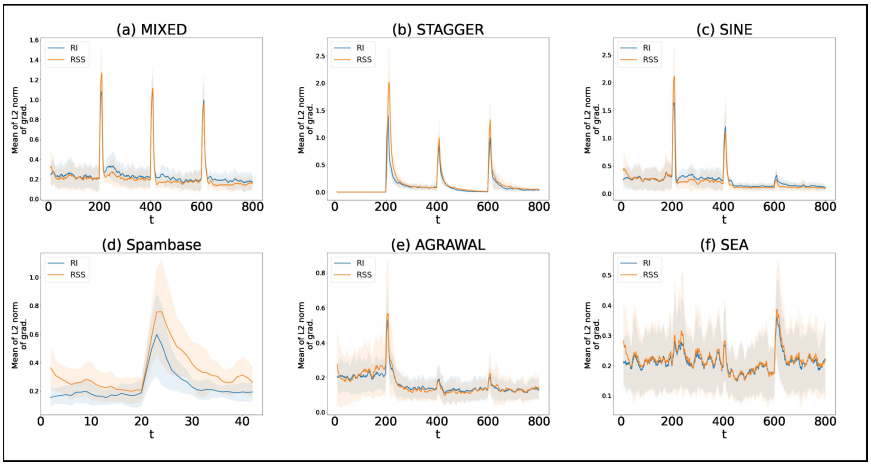

Fig. 8. GNO of each model in each dataset.

was larger than that of RI (e.g., CKA in AGRAWAL and SINE). This suggests that there may not necessarily be a trade-off. It is desirable that the mechanism of the behaviors of these metrics will be clarified to create better metrics.

6 Conclusion

In this study, we investigated the behavior of the predictions and parameters of ensemble models constructed by two different methods in the presence of concept drifts. The results showed that the metrics that can be calculated only by the ensemble model and the metrics based on the model parameters showed behaviors suggestive of concept drift.

Thus, it can be said that the detection accuracy of concept drift may be improved by using the parameters of the ensemble model. There were some metrics based on model parameters that complemented the metrics based on predicted values in terms of concept drift detection, and some indices behaved differently between RI and RSS. It is desirable to clarify the mechanism of the behaviors of these metrics to create better metrics.

Acknowledgments. This research is partially supported by JST CREST Grant Number JPMJCR21F2.

References

1. MLOps: Continuous delivery and automation pipelines in machine learning, https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning. Accessed 29 July 2021
2. Korycki, Ł., Krawczyk, B.: Class-incremental experience replay for continual learning under concept drift. In: CVPR Workshops (2021)
3. Lindstrom, P., Mac Namee, B., Delany, S.J.: Drift detection using uncertainty distribution divergence. In: ICDM Workshops (2011)
4. Zhao, Y., et al.: SUOD: accelerating large-scale unsupervised heterogeneous outlier detection. In: MLSys (2021)
5. Fort, S., Hu, H., Lakshminarayanan, B.: Deep ensembles: a loss landscape perspective. arXiv preprint arXiv:1912.02757v2 (2020)
6. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: a review. IEEE Trans. Knowl. Data Eng. **31**(12), 2346–2363 (2020)
7. Lu, N., Zhang, G., Lu, J.: Concept drift detection via competence models. Artif. Intell. **209**, 11–28 (2014)
8. Ditzler, G., Polikar, R.: Hellinger distance based drift detection for nonstationary environments. In: IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (2011)
9. Qahtan, A., Wang, S.: A pca-based change detection framework for multidimensional data streams categories and subject descriptors. In: KDD (2015)
10. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28645-5_29
11. Sethi, T.S., Kantardzic, M.: On the reliable detection of concept drift from streaming unlabeled data. Expert Syst. Appl. **82**, 77–99 (2017)
12. Smutz, C., Stavrou, A.: When a tree falls: using diversity in ensemble classifiers to identify evasion in malware detectors. In: Proceedings 2016 Network and Distributed System Security (NDSS) Symposium, pp. 21–24. Internet Society, Reston (2016)
13. Haug, J., Kasneci, G.: Learning parameter distributions to detect concept drift in data streams. In: Proceedings of the 25th International Conference on Pattern Recognition (ICPR) (2020)
14. Tran, L., et al.: Hydra: preserving ensemble diversity for model distillation. In: ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning (UDL) (2020)
15. Lakshminarayanan, B., Pritzel, A., Blundel, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: NeurIPS (2017)
16. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. Accessed 29 July 2021
17. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the International Conference on Machine Learning (ICML) (2013)

18. Stewart, G.W.: The efficient generation of random orthogonal matrices with an application to condition estimators. SIAM J. Numer. Anal. **17**, 403–409 (1980)
19. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR) (2015)
20. Baier, L., Schlör, T., Schöffer, J., Kühl, N.: Detecting concept drift with neural network model uncertainty. arXiv preprint arXiv:2107.01873 (2021)
21. Montiel, J., Read, J., Bifet, A., Abdessalem, T.: Scikit-multiflow: a multi-output streaming framework. J. Mach. Learn. Res. **19**(72), 1–5 (2018)
22. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: Proceedings of the 36th International Conference on Machine Learning (ICML) (2019)
23. Cortes, C., Mohri, M., Rostamizadeh, A.: Algorithms for learning kernels based on centered alignment. J. Mach. Learn. Res. **13**, 795–828 (2012)
24. Ovadia, Y., et al.: Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In: NeurIPS (2019)