An Adaptive k-NN Classifier for Medical Treatment Recommendation Under Concept Drift

Nengjun Zhu, Jian Cao(✉), and Yan Zhang

Department of Computer Science and Engineering,
Shanghai Institute for Advanced Communication and Data Science,
Shanghai Jiao Tong University, Shanghai, China
{zhu_nj,cao-jian,zy1992}@sjtu.edu.cn

Abstract. In the real world, concept drift happens in various scenarios including medical treatment planing. Traditional approaches simply eliminate/dilute the effect of outdated samples on the prediction, leading to a less confident (based on fewer samples) prediction and a waste of undiscovered information contained in past samples. With the knowledge of how concepts change, outdated samples can be adapted for up-to-date prediction, which improves the confidence of prediction, especially for medical data sets of which the scale is relatively small. In this paper we present an adaptive k-NN classifier which can detect the occurrence of target concept drift and update past samples according to the knowledge of the drift for better prediction, and assess the performance over simulated and real-world categorical medical data sets. The experiment results show our classifier achieves better performance under concept drift.

Keywords: k-NN classifier · Recommendation · Concept drift
Medical treatment

1 Introduction

Concept drift was originally observed in streaming data, where the relation between the target class and features changes over time in unforeseen ways. The phenomenon also exists in other fields such as medical diagnosis. For example, Fig. 1 shows that the chemotherapies being used in breast cancer diagnosis are changing as time goes on, over different molecular subtypes [8,9] (e.g., Luminal-A), and the patients of some molecular subtypes receive different chemotherapies in different periods of time.

However, most state-of-art concept-aware learning algorithms focus on streaming data but medical diagnosis data. They achieve good performance using techniques such as ensemble learning, transfer learning, etc. For example, a novel algorithm named DETL [12] utilizes each preserved historical model as an initial model which is further trained with the new data via transfer learning. Even

Y. Sun et al. (Eds.): ChineseCSCW 2018, CCIS 917, pp. 546–556, 2019.
https://doi.org/10.1007/978-981-13-3044-5_42

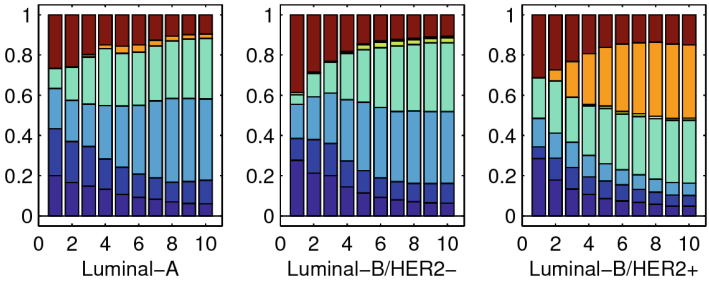


Fig. 1. Distribution of samples for different molecular subtypes, where the x-axes represent samples in temporal order, and the y-axes represent a proportion of the seven chemotherapies in each period of time.

though DETL can effectively handle concept drift in both synthetic data (e.g. SEA [11], STAGGER [7]) and real-world data streams. However, it is not easy to port this kind of algorithms directly to process such medical diagnosis data without transforming categorical data into numerical.

The medical diagnosis data is very limited compared with other fields, so that it is natural to process it with case-based approaches, i.e., the k Nearest Neighbors algorithm (k-NN). The k Nearest Neighbors algorithm is a non-parametric method used for classification and regression, and it has been applied to various areas [1]. To deal with concept drift, based on the idea of 'forgetting' old samples, previous work uses sliding windows to filter the historical samples for reference [13], or applies an age-related weight to each sample to dilute the influence of older samples [4,5]. However, simply forgetting old samples is not suitable in medical treatment recommendation. For small-scale data sets such as medical diagnosis data, each sample is precious and contains invaluable information due to the individual variation in patients. During the process of prediction, we should take as many samples into account as we can. And the prediction result based on more samples is more confident and more resistant to noises.

To tackle the problems introduced by concept drift, and to make better use of old samples, we propose a new k-NN based classifier using the paired learner technique [2]. In this approach, we discover knowledge of how an old concept changes to a new one. Instead of discarding outdated samples, we create a replica of them to preserve the original information, and then revise the labels according to new concepts for future predictions. To study the effectiveness of the approach, we compare it with two baseline methods over both simulated and real-world data, and analyze the influence of the parameters in the proposed approach. Focusing on medical treatment recommendation, the contribution of the paper is list as follows:

1. We improve the concept-drift detection method based on the paired learner technique.
2. We propose an adaptive k-NN classifier for concept-drifting data.

3. Extension experiments over both simulated and real data sets prove our approach has better performance against baselines.

The rest of the paper is organized as follows: Sect. 2 details our approach. Section 3 presents the results of our experiments using different approaches over simulated and real-world data sets. Section 4 concludes the paper.

2 Learning Under Concept Drift

In this part, we first put forward a modified *overlap* similarity, and then illustrate in detail a k-NN based adaptive classifier which behaves well under concept drift.

2.1 Notations

For a data set D containing N samples, we define A as the set of d attributes where A_k denotes the k^{th} attribute, and define C as the set of all types of target class. Temporally, let t^{th} sample in data set D be represented as follows:

$$D_t = < \boldsymbol{x}^{(t)}, y^{(t)} > \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, x_3, ..., x_d)^T$ denotes an attribute vector of a sample, and $y \in C$ is the labeled class of D_t. For convenience, we use $\mathbb{N}_a^b$ to represent $\mathbb{N} \cap [a, b]$, which is the set of integers from a to b inclusive.

Similarity measure is a key issue of classifiers such as k-NN. For numerical attributes, the Mahalanobis distance [6] is as follows:

$$Dist(\boldsymbol{x}^{(a)}, \boldsymbol{x}^{(b)}) = \sqrt{(\boldsymbol{x}^{(a)} - \boldsymbol{x}^{(b)})^T S^{-1} (\boldsymbol{x}^{(a)} - \boldsymbol{x}^{(b)})} \tag{2}$$

where S is the covariance matrix. We then convert this distance measure to a similarity measure with the scale from 0 to 1 using the following formula:

$$Sim(\boldsymbol{x}^{(a)}, \boldsymbol{x}^{(b)}) = \frac{1}{1 + Dist(\boldsymbol{x}^{(a)}, \boldsymbol{x}^{(b)})} \tag{3}$$

Further, during classification, the sample will be labeled by a class according to the labeled classes of its neighbors as well as its similarity with its neighbors, i.e., for a sample $\boldsymbol{x}$, the similarity is used as a weight for each neighbor, and then, the sample will be labeled by a class with the largest linear combination of weight among the neighbors, rather than simply assigning the majority of class labels in its k nearest neighbors to the target case.

$$\hat{y} = \arg\max_{y_c \in Y} \sum_{D_i \in S(\boldsymbol{x}, k)} [\![y^{(i)} = y_c]\!] \cdot Sim(\boldsymbol{x}, \boldsymbol{x}^{(i)}) \tag{4}$$

where $[\![\cdot]\!]$ is an indicator function, where $[\![true]\!] = 1$ and $[\![false]\!] = 0$, and $S(\boldsymbol{x}, k)$ represents the set of k-nearest neighbors of a sample $\boldsymbol{x}$.

2.2 Similarity Definition for Categorical Features

Since the value of categorical features is discrete, it prevents application of the Mahalanobis technique. The similarity value between two samples $D_a = < \boldsymbol{x}^{(a)}, y^{(a)} >$ and $D_b = < \boldsymbol{x}^{(b)}, y^{(b)} >$ in this context is usually synthesized by similarities in all attributes as follows:

$$Sim(\boldsymbol{x}^{(a)}, \boldsymbol{x}^{(b)}) = \Sigma_{k=1}^{d} \omega_k Sim_k(x_k^{(a)}, x_k^{(b)}) \tag{5}$$

where $Sim_k(x_k^{(a)}, x_k^{(b)})$ denotes the similarity between single attribute A_k of sample D_a and sample D_b, and the weight ω_k indicates the importance of each attribute A_k.

Overlap [10] is a method to define the similarity between two categorical attributes. It calculates the similarity by counting the number of matched attributes, e.g., 0 represents not matching, and 1 represents matching, during the measurement of the similarity of a pair of samples on a single attribute. Thus, the proportion of the number of attributes in which they match is considered as the similarity between two multivariate categorical samples. However, in real data sets, missing the value in some attributes is very common. Consider three samples D_a, D_b and D_c having the following properties:

$$x_k^{(a)} \neq x_k^{(b)} \textbf{ and } x_k^{(c)} \text{ is missing}, \forall k \in \mathbb{N}_1^d \tag{6}$$

Notice that D_a and D_b are completely different because they do not match in any attribute, while the similarity between D_a and D_c should be neutral since we do not know any of the attributes of D_c. Hence, it follows that the heuristic, $Sim(x^{(a)}, x^{(b)})$ should be the minimum while $Sim(x^{(a)}, x^{(c)})$ should be defined as a compromised value. To deal with cases where the data set has missing values, the modified overlap measure is defined as follows:

$$Sim_k(x_k^{(a)}, x_k^{(b)}) = \begin{cases} 1 & \text{if } x_k^{(a)} = x_k^{(b)} \\ 0.5 & \text{if } x_k^{(a)} or x_k^{(b)} \text{ is missing} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\omega_k = \frac{1}{d} \tag{8}$$

Equation 8 shows that the weights of all attributes are the same. However, the experimental results in a past study [3] suggest that there is no one best performing similarity measure, hence we choose *overlap* to measure similarity for the sake of convenience.

2.3 Adaptive Classifier and Concept-Sensitive Detector

Similar to the paired learner, we equip the k-NN classifier with a concept-sensitive detector with a fixed-size sliding window to enable it to adapt to concept drift. Denote the adaptive classifier as AC and the concept-sensitive detector as CSD_{sw} with a sliding window of size sw. AC classifies new samples based on

Algorithm 1. Adaptive k-NN Classifier

Input: D, N_{init}, sw, k
Output: predicted class for each object
$D = \{< \boldsymbol{x}^{(i)}, y^{(i)} >\}_{i=1}^{N}$: training data
N_{init}: initial training sample number
sw: sliding window size of concept-sensitive detector
1: $AC \leftarrow$ an adaptive k-NN classifier
2: $CSD_{sw} \leftarrow$ a concept-sensitive detector of window size sw
3: **for** $i \leftarrow 1$ **to** N_{init} **do**
4: AC.train(D_i) & CSD_{sw}.train(D_i)
5: **end for**
6: **for** $i \leftarrow N_{init} + 1$ **to** N **do**
7: $\hat{y}_{AC} \leftarrow AC$.classify($\boldsymbol{x}^{(i)}$)
8: **output** $\hat{y}_{AC}$
9: $\hat{y}_{CSD} \leftarrow CSD_{sw}$.classify($\boldsymbol{x}^{(i)}$)
10: **if** concept drift detected **then**
11: revisePastSamples(AC)
12: **end if**
13: AC.train(D_i) & CSD_{sw}.train(D_i)
14: **end for**

all historical samples while CSD_{sw} makes use of the most recent samples within sw. So the effectiveness of the paired learner is based on the following heuristics:

1. Based on more knowledge, the result is more reliable.
2. Based on more recent knowledge, the result is more reactive.

As shown in Algorithm 1, the input is a data set D of N samples, the number of initial training samples N_{init}, the sliding windows size sw, and the number of nearest neighbors k. To initialize AC and CSD_{sw}, the algorithm trains them with $\{D_i | i \in \mathbb{N}_1^{N_{init}}\}$ independently (line 3–5). Note that, different from AC, CSD_{sw} keeps at most sw training samples, and will forget the earliest sample if the size of the training samples exceeds sw. After initialization, the classifier predicts new samples via AC (line 7–8) and in the meantime both classifiers learn from the sample (line 13). During the prediction, if the algorithm has observed the occurrence of concept drift, the main classifier AC will be updated, so as to maintain its performance under the new concept (line 10–12). Note that we always create a replica to preserve the original labels before revising outdated samples.

In common cases, AC should behave no worse than CSD for its abundant knowledge of both recent and past data. At the very beginning of the occurrence of concept drift, new samples are classified according to outdated knowledge, thus performance degradation for both classifiers is observed. But CSD's performance can return to a normal level soon, because the historical data of CSD is replaced with those under the new concept due to its abandoning of past samples. Based on this heuristic, the occurrence of concept drift can be observed by monitoring the performance of the paired classifiers. That is, when CSD performs better

than AC for a period of time, we assert an occurrence of concept drift, and then revise the historical data of AC for better adaptation to the new concept.

2.4 Sample-Based Drift Detection and Adaptation

Instead of detecting concept drift by comparing the accuracy of each classifier in the paired classifiers for a batch of recent samples, we explore a sample-based drift detection and adaption method, since the batch accuracy comparison method cannot handle concept drift in small local part of the original data. Intuitively, we can separate the feature space into many small zones or apply clustering on the original data, and monitor concept drift independently in each zone or cluster. More extremely, we regard each sample as a single zone, in other words, we keep monitoring each sample during the classification and then we can judge for each historical sample whether the labeled class should be changed to ensure better prediction of future samples. Meanwhile, based on information of misclassification for each sample we have logged during classification, the new class to which a drifted sample should be revised can be decided simultaneously along with the detection, as shown in Algorithm 2.

Denote the sample being processed at the moment as D_{now}, if AC incorrectly classifies it but CSD_{sw} correctly classifies it (line 4). Then the training samples in AC, based on which the prediction was made, are marked as conflicting with the current sample D_{now}, for they are considered to have provided wrong implications (line 5). For each training sample in AC, if it satisfies certain conditions (line 6), e.g. being observed frequently conflicting with recent samples, it is regarded that its features fall in the zone where concept drift occurs. In this case, AC revises this sample with a new class according to its *confliction records* (line 8). The confliction records for sample D_t contain tuples of samples in future predictions which select D_t as one of the nearest neighbors for reference but receive a negative effect due to the difference in their actual classes. The definition of *confliction records* $C[t]$ is as follows:

$$\{(i, y^{(i)}) | i > t \wedge \boldsymbol{x}^{(t)} \in AC.\text{nearest}(k, \boldsymbol{x}^{(i)}) \wedge y^{(i)} \neq y^{(t)}\} \tag{9}$$

Note that we have $y^{(i)} \neq y^{(t)}, \forall (i, y^{(i)}) \in C[t]$, once D_t contributes to a correct classification for D_i, we clear the former confliction records of D_t to avoid a false alarm caused by noise (line 11). The concept-drift condition (line 6) is a key point to this algorithm, which directly decides whether to declare an occurrence of concept drift so as to revise the historical samples. We use a score-based indicator in our algorithm.

What we are most concerned about are the target classes for present and future samples, thus for tuples in the confliction records $C[t]$, conflicts with recent samples are apparently more valuable than those with older ones. In addition, responses that are too fast always cause false alarms, and conflicts in stable periods are much more meaningful than those in unstable periods. To differentiate between stable and unstable periods, according to the confliction records, we say it is stable if there is no conflicts in a fixed period of time,

otherwise it is unstable. Based on these heuristics, we assign scores for each conflict which appears in $C[t]$ with a function having a positive correlation with *time*, so that confliction records in stable periods have higher scores. Then we add the scores for each class as its measure.

The function $Score(y)$ can be defined in a variety of ways, as long as it has a positive correlation with *time*. The following is one candidate for the function using the exponential technique:

$$Score(y) = \sum_{(i,y)\in C[t]} 1 - e^{(-(i-t))} \tag{10}$$

If the score of a certain class dominates among the candidates, we assert that D_t is affected by concept drift and we know exactly the trend of its change.

Algorithm 2. Sample-based detection and adaptation method

Input: D, C, k, S, $\hat{y}_{AC}$, $\hat{y}_{CSD}$, $< x^{(now)}, y^{(now)} >$
Output: Whether concept drift detected
$D = \{< \boldsymbol{x}^{(i)}, y^{(i)} >\}_{i=1}^{now-1}$: training data of AC
C: a dict of confliction records for training data
$\hat{y}_{AC}$: class predicted by AC
$\hat{y}_{CSD}$: class predicted by CSD
$< \boldsymbol{x}^{(now)}, y^{(now)} >$: current sample.
nearest$(k, \boldsymbol{x})$: returns a set of $\boldsymbol{x}$'s k nearest neighbors

```
S ← {t|D_t ∈ AC.nearest(k, x^(now)) ∧ y_t = ŷ_AC}
flagOfDrift ← False
for t ∈ S do
    if ŷ_AC ≠ y_now and ŷ_CSD = y_now then
        C[t].append((now, y_now))
        if C[t] satisfies concept-drift conditions then
            flagOfDrift ← True
            revise D_t
        end if
    else
        C[t].clear()
    end if
end for
return flagOfDrift
```

3 Experiment Study

To verify the effectiveness of our adaptive k-NN classifier, we compare it with several baseline approaches over two data sets. Since finding the best k in the k-NN classifier is not within the scope of this paper, we simply use $k = 10$ in all the following experiments. Commonly the value of k should be an odd number

to avoid tied votes, but it does not matter if we set $k = 10$ in our approach, as we choose the final class according to the sum of similarity instead of the plain count for each candidate.

3.1 Evaluation Metrics

We assess the performance of our approach and conduct comparison experiment with following two metrics: **Accumulative accuracy** and **Next-n accuracy**. Accumulative accuracy is defined as follows:

$$\text{Accuracy}_{accum}(t) = \frac{1}{t - N_{init}} \sum_{i=N_{init}+1}^{t} [\![\hat{y}^{(i)} = y^{(i)}]\!] \tag{11}$$

where t and N_{init} denote the index of current sample and that of the last sample in initial training set, respectively. Accumulative accuracy evaluates the ratio of samples which are correctly classified *to the current moment*, while Next-n accuracy evaluates the ratio of samples which are correctly classified *among the next n samples*, and it is defined as follows:

$$\text{Accuracy}_{n}(t) = \frac{1}{n} \sum_{i=t+1}^{t+n} [\![\hat{y}^{(i)} = y^{(i)}]\!] \tag{12}$$

A curve of accumulative accuracy reveals long-term trends in the classifier performance, while a curve of next-k accuracy shows short-term changes in performance. For both metrics, the larger the value of accuracy, the better performance of the classifier.

3.2 Comparison Classifiers

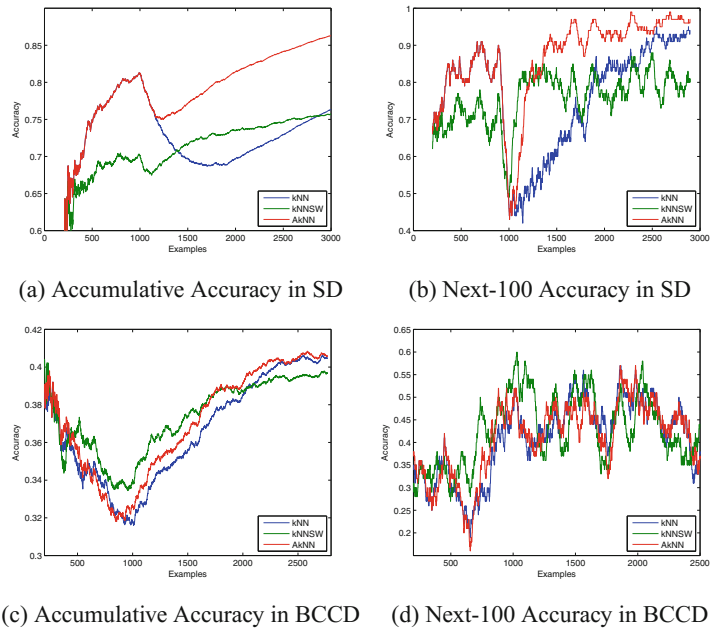
- k-**NN:** This is a basic k-nearest neighbor classifier playing the role of baseline, using the *modified overlap* similarity measure mentioned in Eq. 7.
- k-**NNSW:** This is a window-based extension of k-NN in order to deal with concept drift by learning knowledge from the most recent samples within a sliding window.
- Ak-**NN:** This is an adaptive k-NN classifier mentioned in Algorithm 1. To study the influence of different concept-drift indicators in Algorithm 2, we implement all three indicators respectively and compare them in the experiments.

3.3 Data Sets

To verify the performance of our algorithm, we perform experiments over an artificial data set with concept drift, named **Simulated Data (SD)**. Moreover, we also compare different methods over a real medical data set named **Breast Cancer Chemotherapy Data (BCCD)**, where we use the classifiers to recommend chemotherapy for breast cancer patients according to historical cases.

Table 1. Data sets overview

Name	Entries	Attributes	Classes	Drift position	Name	Entries	Attributes	Classes	Drift position
SD	3000	6	4	1000	**BCCD**	2770	12	8	unknown



(a) Accumulative Accuracy in SD (b) Next-100 Accuracy in SD

(c) Accumulative Accuracy in BCCD (d) Next-100 Accuracy in BCCD

Fig. 2. Results for k-NN, k-NNSW, Ak-NN

Samples in both data sets are arranged in temporal order, where indices of each sample also represent the time. Characteristics of these two data sets are summarized in Table 1. For the simulated data set SD, there is an explicit drift position of 1000 (the first 1000 samples are labeled according to their attributes using a different linear function from the remaining 2000 samples), representing the position where concept drift occurs, while for the real data set BCCD, samples are collected from real cases, thus we cannot declare any explicit drift positions.

3.4 Experimental Results

On the simulated data set, we apply the classifiers and calculate the accumulated accuracy of all predictions, and the average accuracy of the next 100 predictions. The results of the two metrics are shown respectively in Fig. 2a and b. Both k-NN and Ak-NN exploit the whole historical data, and thus they are more accurate on average than k-NNSW as shown in Table 2. For long-term learning, k-NNSW is not a good choice.

We knew in advance that there is an occurrence of concept drift at the 1000^{th} sample in SD. As shown in Fig. 2a, from the 1000^{th} sample onwards, affected

Table 2. Overall accuracy over SD

sw	k-NN	k-NNSW	Ak-NN	**sw**	k-NN	k-NNSW	Ak-NN	**sw**	k-NN	k-NNSW	Ak-NN
100	0.7598	0.7212	**0.8634**	150	0.7637	0.7451	**0.8662**	200	0.7634	0.7573	**0.8631**

by the new concept, all the classifiers experience degradation in accuracy. The basic k-NN classifier takes a long time to get its accuracy back to the former level, while k-NNSW and Ak-NN both adapted to the new concept reactively. The effect on the accuracy of all the classifiers is shown more clearly in Fig. 2b, where Ak-NN and k-NNSW rebound rapidly soon after concept drift. However, to return to the former accuracy, the plain k-NN classifier learns as many as 1000 new samples. If we have more samples before concept drift, the k-NN classifier has to learn even more new samples to dilute the influence of outdated historical samples.

The results over BCCD look slightly complicated and the concept shifts slowly over time instead of changing suddenly. Rapid drops in accuracy for both k-NN and Ak-NN are observed in Fig. 2d around the 500^{th} sample, while k-NNSW reacts well and maintains its accuracy at the former level. Due to the concept-drift detection mechanism, Ak-NN's accuracy rebounds slightly faster than k-NN's. Note that k-NNSW's accuracy exceeds the other two at the very beginning, but Fig. 2c shows the downward trend of using only the most recent samples for reference and that k-NNSW will be defeated as more samples are learned. The defect of k-NNSW is shown more clearly in Fig. 2d. At around the 1800^{th} and 2300^{th} samples, k-NNSW suffers from its narrow window and has an obvious degradation of accuracy while the other two classifiers remain stable. Even though it can recover very soon, its over reactivity harms the overall accumulated accuracy.

4 Conclusion

We have proposed an adaptive k-NN classifier (Ak-NN) to tackle the concept drift in medical treatment recommendation, which performs better than baseline methods on both simulated data set and real data set. Derived from k-NN, our approach is also non-parametric as it does not rely on the knowledge of distribution of the input data. Instead of discarding outdated samples, the approach discover the knowledge of concept drift, based on which it revises the old samples for better prediction under the new concept.

Acknowledgments. China National Science Foundation (Granted Number 61272438,61472253), Cross Research Fund of Biomedical Engineering of Shanghai Jiao Tong University (YG2015MS61), and Research Funds of Science and Technology Commission of Shanghai Municipality (Granted Number 15411952502, 14511107702) support this work.

References

1. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**(3), 175–185 (1992)
2. Bach, S.H., Maloof, M.A.: Paired learners for concept drift. In: 2008 Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 23–32. IEEE (2008)
3. Boriah, S., Chandola, V., Kumar, V.: Similarity measures for categorical data: a comparative evaluation. Red **30**(2), 3 (2008)
4. Klinkenberg, R.: Learning drifting concepts: example selection vs. example weighting. Intell. Data Anal. **8**(3), 281–300 (2004)
5. Koychev, I.: Tracking changing user interests through prior-learning of context. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, pp. 223–232. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-47952-X_24
6. Mahalanobis, P.C.: On the generalized distance in statistics. Proc. Natl. Inst. Sci. (Calcutta) **2**, 49–55 (1936)
7. Minku, L.L., White, A.P., Yao, X.: The impact of diversity on online ensemble learning in the presence of concept drift. IEEE Trans. Knowl. Data Eng. **22**(5), 730–742 (2010)
8. Perou, C.M.: Molecular stratification of triple-negative breast cancers. The Oncol. **16**(Suppl. 1), 61–70 (2011)
9. Prat, A., Perou, C.M.: Deconstructing the molecular portraits of breast cancer. Mol. Oncol. **5**(1), 5–23 (2011)
10. Stanfill, C., Waltz, D.: Toward memory-based reasoning. Commun. ACM **29**(12), 1213–1228 (1986)
11. Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 377–382. ACM (2001)
12. Sun, Y., Tang, K., Zhu, Z., Yao, X.: Concept drift adaptation by exploiting historical knowledge. arXiv preprint arXiv:1702.03500 (2017)
13. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. **23**(1), 69–101 (1996)