Dynamic Integration of Classifiers for Handling Concept Drift

Alexey Tsymbal[a*], Mykola Pechenizkiy[b,d], Pádraig Cunningham[c],
Seppo Puuronen[d]

[a]Siemens AG, Günther-Scharowsky-Str. 1, 91058 Erlangen, Germany;
alexey.tsymbal@siemens.com
[*] tel.: +49 - 9131 - 728796; fax: +49 - 9131 - 733190
[b]Technical University of Eindhoven, The Netherlands; m.pechenizkiy@tue.nl
[c]University College Dublin, Ireland; padraig.cunningham@ucd.ie
[d]University of Jyväskylä, Finland; {mpechen, sepi}@cs.jyu.fi

Dynamic Integration of Classifiers for Handling Concept Drift

Abstract. In the real world concepts are often not stable but change with time. A typical example of this in the biomedical context is antibiotic resistance, where pathogen sensitivity may change over time as new pathogen strains develop resistance to antibiotics that were previously effective. This problem, known as concept drift, complicates the task of learning a model from data and requires special approaches, different from commonly used techniques that treat arriving instances as equally important contributors to the final concept. The underlying data distribution may change as well, making previously built models useless. This is known as virtual concept drift. Both types of concept drifts make regular updates of the model necessary. Among the most popular and effective approaches to handle concept drift is ensemble learning, where a set of models built over different time periods is maintained and the best model is selected or the predictions of models are combined, usually according to their expertise level regarding the current concept. In this paper we propose the use of an ensemble integration technique that would help to better handle concept drift at an instance level. In dynamic integration of classifiers, each base classifier is given a weight proportional to its local accuracy with regard to the instance tested, and the best base classifier is selected, or the classifiers are integrated using weighted voting. Our experiments with synthetic data sets simulating abrupt and gradual concept drifts and with a real-world antibiotic resistance data set demonstrate that dynamic integration of classifiers built over small time intervals or fixed-sized data blocks can be significantly better than majority voting and weighted voting, which are currently the most commonly used integration techniques for handling concept drift with ensembles.

Keywords: Machine Learning, Changing Environment, Concept Drift, Ensemble Learning, Dynamic Integration of Classifiers

1 Introduction

The problem of concept drift is of increasing importance to machine learning and data mining as more and more data is organized in the form of data streams rather than static databases, and it is rather unusual that concepts and data distributions stay stable over a long period of time [23, 30].

Ensemble learning is among the most popular and effective approaches to handle concept drift, in which a set of concept descriptions built over different time intervals is maintained, predictions of which are combined using a form of voting, or the most relevant description is selected [13, 20, 21, 28]. However, there is a problem with current ensemble approaches; they are not able to deal with local concept drift, which is a common case with real-world data. For example, only particular bacteria may develop their resistance to certain antibiotics, while resistance to the others can remain the same; or the data distribution can change for particular bacteria depending on the season.

At present, the most common integration approach with ensembles for handling concept drift is weighted voting, where each base classifier receives a weight proportional to its relevance to the current concept [13, 20, 21, 28]. With weighted voting, lower weights can be assigned to predictions from base classifiers simply because their global accuracy on the current block of data falls, even if they are still good experts in the stable parts of the data.

In this paper, we consider one solution to this problem; namely replacing the integration (combination) function of the ensemble. To improve the treatment of local concept drifts, dynamic integration of classifiers can be used, which integrates base classifiers at an instance level. In dynamic integration, each base classifier receives a weight proportional to its local accuracy in the neighbourhood of the current test instance, instead of using global classification accuracy as in normal weighted voting.

We consider ensemble learning with dynamic integration for handling concept drifts in the *rotating hyperplane* and *SEA concepts* data sets, representing simulated gradual and abrupt concept drifts respectively [6, 21]. Besides, we apply dynamic integration to ensembles of classifiers built in the domain of antibiotic resistance in nosocomial infections in order to better handle concept drift.

Our experiments demonstrate that dynamic integration often achieves better classification accuracy than commonly used integration techniques, such as majority voting and weighted voting, on the synthetic data sets and on the problem of antibiotic resistance prediction, supporting our hypothesis that it can be a better technique for handling concept drift.

The idea of the use of dynamic integration for handling concept drift was introduced by us in [26] with experiments focusing on the problem of antibiotic resistance in nosocomial infections, and in this paper we present the dynamic integration approach in the level of detail necessary for possible implementation. We introduce the notion of local concept drift with a simple example, we consider more extensive experiments with data sets representing different types of concept drift, and discuss possible improvements to the dynamic integration techniques considered by comparing them to other related dynamic integration techniques.

This paper is organized as follows: in Section 2 we consider the general problem of concept drift, in Section 3 we introduce the notion of local concept drift, and in Section 4 we review approaches to ensemble integration with a focus on dynamic integration. In Section 5 we consider the basic characteristics of the data sets used for analysis, in Section 6 we present the results of our experiments with the use of different ensemble integration techniques with synthetic and real-world data, in Section 7 we discuss the three dynamic integration techniques considered and possible alternative techniques, and in Section 8 we conclude with a brief summary and a consideration further research directions.

2 The Problem of Concept Drift

A difficult problem with learning in many real-world domains is that the concept of interest may depend on some hidden context, not given explicitly in the form of predictive features. Changes in the hidden context can induce more or less radical changes in the target concept, which is generally known as *concept drift* [30]. A typical example of this is antibiotic resistance, where pathogen sensitivity may change over time as new pathogen strains develop resistance to antibiotics that were previously effective. An effective learner should be able to track such changes and to quickly adapt to them. Kukar [14] states that even in most strictly controlled environments some unexpected changes may happen due to failure and replacement of some medical equipment, or due to changes in personnel, causing the necessity to change the model.

Changes in hidden context may not only be a cause of a change of target concept, but may also cause a change of the underlying data distribution. Even if the target concept remains the same, and it is only the data distribution that changes, this may often lead to the necessity of revising the current model, as the model's error may no longer be acceptable with the new data distribution. The need to the change of current model due to the change of data distribution is called *virtual concept drift* [29]. Virtual concept drift and real concept drift often occur together. From the practical point of view it is not important, what kind of concept drift occurs, real or virtual, or both. In all cases the current model needs to be changed.

Three approaches to handling concept drift can be distinguished: (1) instance selection; (2)

instance weighting; and (3) ensemble learning [23]. In instance selection, the goal is to select instances relevant to the current concept. The most common concept drift handling technique is based on instance selection and consists in generalizing from a window that moves over recently arrived instances and uses the learnt concepts for prediction only in the immediate future [30]. Many case-base editing strategies in case-based reasoning that delete noisy, irrelevant and redundant cases are also a form of instance selection [5].

Instance weighting uses the ability of some learning algorithms such as Support Vector Machines (SVMs) to process weighted instances [12]. Instances can be weighted according to their "age", and their competence with regard to the current concept. Klinkenberg [12] demonstrates in his experiments that instance weighting techniques handle concept drift worse than analogous instance selection techniques, which is probably due to overfitting the data.

Ensemble learning maintains a set of concept descriptions, predictions of which are combined using a form of voting, or the most relevant description is selected. Street and Kim [21] and Wang *et al.* [28] suggest that simply dividing the data into sequential blocks of fixed size and building an ensemble on them may be effective for handling concept drift. Stanley [20] and Kolter and Maloof [13] build ensembles of incremental learners in an online setting, starting to learn new base classifiers after fixed intervals, while continuing to update the existing ones. All incremental ensemble approaches use some criteria to dynamically delete, reactivate, or create new ensemble members, which are normally based on the base models' consistency with the current data.

In the real world, concept drift may often be local, for example only particular bacteria may develop their resistance to certain antibiotics, while resistance to the others could remain the same. In the case of local concept drift, models from an ensemble should not be discarded (or lower weights assigned to them) simply because their global accuracy on the current block of data falls, even if they are still good experts in the stable parts of the data. This is an important problem with most of the existing ensemble approaches for handling concept drift. One solution to this problem is the use of dynamic integration of classifiers, in which the models are integrated at an instance level according to their local accuracies.

3 Local Concept Drift

The most popular approach for handling concept drift is windowing. However, there is one important drawback with window-based techniques. Often changes in the concept or data distribution occur in some regions of instance space only, and besides, the type and severity of changes may depend on the location in the instance space. We call this phenomenon *local concept drift*.

We may define local concept drift as changes in concept and data distribution occurring at an instance rather than data set level. Formally speaking, it can be said that local concept drift occurs between two consecutive time points t_1 and t_2, if there is a sub-space X' of the whole instance space X such that it has different changes of concept and/or data distribution in comparison with the rest of the data. This is reflected by a different change in (local) predictive performance of currently used model *h* in this sub-space.

We believe that this type of local concept drift is quite common and will occur when concepts are complex or diverse. For instance it occurs in spam filtering [5] as new categories of spam appear and disappear. Things like "papal memorabilia" spam come and go but prescription medicine spam is always with us. By contrast a classification system for routing or filing text messages might be faced with global rather than local concept drift.

Window-based approaches implicitly assume that concept drift is ubiquitous and global by selecting one window for representing the current concept. This is assumed even if the size of the window is adaptive and selected to be optimal in terms of amount of relevant instances within it. However, in fact, some sub-areas in the instance space may be rather stable and may benefit from selecting larger windows for them, and, vice versa, some sub-areas might need significantly smaller windows due to recent concept drift.

Imagine a slowly rotating hyperplane, dividing the instance space into two classes (Figure 1). In this classification task, instances lying close to the hyperplane would need smaller windows for better prediction, while instances, lying far enough from the hyperplane could benefit from larger windows. Thus, even if semantically the concept is continuously changing for the whole data set, there are regions of the instance space that will remain stable significantly longer than the other regions. The older models of stable regions could still be useful for prediction within these regions (for example, in an ensemble scenario), or instances from these regions could be used for building a better updated current model. Different stability of different regions of the instance space is illustrated in Figure 1 with the simple *rotating hyperplane* problem. The thickness of the instances represents stability of the corresponding area for four consecutive time points $t_1 – t_4$. Notice that there are some instances (the thickest ones in t_4) that could be used for representing all of the four stages of the concept.

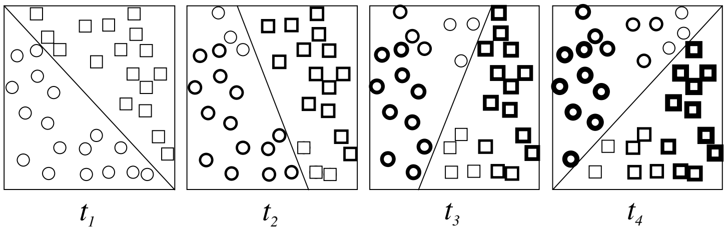

Figure 1. Stability of regions in the rotating hyperplane problem

Thus, the main drawback of window-based approaches with local concept drift is that instances with different relevance with regard to the current concept are often included in the window. Based on this observation, combination of window-based and instance weighting techniques might be effective for handling local concept drift, and forms an interesting novel direction for further research.

In fact, most *gradual* concept drifts may be considered local, if the velocity of changes is small relative to the amount of instances arriving in the data stream, and most regions of the data remain stable. Most *abrupt* concept drifts are not local, unless substantial sub-areas remain stable between the two changing concepts. Abrupt concept drift can also be local, if it relates to a subgroup of the whole population. Besides, concept drift may also be *complex*, with different population subgroups or clusters of the data being subject to different concept or data distribution changes. For example, changes in antibiotic resistance and data distribution are usually different for different bacteria in the antibiotic resistance problem. Clearly, complex concept drift is local as well according to our definition.

In conclusion, as local concept drift occurs at an instant level – its treatment should appropriately be at that level as well. A few techniques have been suggested in the literature that work at an instance level and thus would potentially be able to handle local concept drift well. In Case-Based Reasoning a case base is updated at an instance level, which makes CBR a well-suited approach for handling local concept drift [5]. In [6] a hybrid of ensemble learning and instance selection was proposed, which might also be effective with local concept drift. Ensemble integration based on local accuracies as considered in this paper is another alternative for handling concept drift at an instance level.

4 Ensemble Learning and Dynamic Integration of Classifiers

Brodley and Lane [3] have shown that simply increasing diversity of an ensemble is not enough to ensure increased prediction accuracy. If an integration method does not utilize diversity, then no benefit arises from the integration. The task of integration is to decide which of the classifiers to select or how to combine the results produced by the base classifiers. A number of *selection* and *combination* approaches have been proposed.

One of the most popular and simplest techniques used to combine the results of base classifiers is simple voting (also called majority voting) [2]. In voting, the output of each base classifier is considered as a vote for the corresponding class value. The class value that receives the biggest number of votes is selected as the final classification. Weighted Voting (WV), where each vote has a weight proportional to the estimated generalization performance of the corresponding classifier, usually has better predictive performance than simple voting [2].

A number of selection techniques have also been proposed to address the task of integration. One of the most popular and simplest selection techniques is Cross-Validation Majority (CVM) [19]. In CVM, cross-validation accuracy for each base classifier is estimated, and the classifier with the highest accuracy is selected.

The approaches described above are *static*. They select one model for the whole instance space or combine the models uniformly. In *dynamic* integration information about each new instance is taken into account. Experimental studies in many application domains demonstrate that better results can be achieved if integration is dynamic [9, 10, 17, 25, 27, 33].

We consider in our experiments three dynamic techniques based on the same local performance estimates: Dynamic Selection (DS), Dynamic Voting (DV), and Dynamic Voting with Selection (DVS), which were previously used for stationary problems [27]. These techniques have the same training phase, when the local performance of each base classifier for each instance of the validation set is estimated. The application phase begins with determining *k* nearest neighbours for a new instance from the validation set. Then, weighted nearest neighbour learning is used to predict the local performance of each base classifier.

Then, DS simply selects a classifier with the best local predictive performance. In DV, each base classifier receives a weight that is proportional to its estimated local performance, and the final classification is produced using weighted voting. In DVS, the base classifiers with the worst local performances are discarded (the classifiers with the performances that fall into the lower half of the performance interval) and locally weighted voting (DV) is applied to the remaining classifiers.

In fact, the basic idea in the dynamic integration approach suggested consists in learning (meta-level learning) the predictive performance of base models. Instance-based learning is used in order to predict the local performance of the base models in an ensemble.

Different distance functions can be used for determining the neighbourhood of the current test instance in dynamic integration. The simplest and most common way is to use the heterogeneous Euclidean/overlap metric (HEOM)[31] in the instance space as in [27]. In HEOM, the Euclidean distance is used with numeric features, and the overlap distance with categorical features in order to find a distance between two instances $\mathbf{x}_1$ and $\mathbf{x}_2$ as shown in (1) and (2), where m is the number of features. For a numeric feature a the distance is normalised by the width of the range of values of the corresponding feature on the training set $range_a$:

$$d_{heom}(\mathbf{x}_1,\mathbf{x}_2) = \sqrt{\sum_{a=1}^{m} heom_a^2(\mathbf{x}_1,\mathbf{x}_2)} \tag{1}$$

$$heom_a(\mathbf{x}_1,\mathbf{x}_2) = \begin{cases} \text{if } a \text{ is categorical, } \begin{cases} 0, \text{if } x_{1a} = x_{2a} \\ 1, \text{otherwise} \end{cases} \\ \text{else, } \dfrac{|x_{1a} - x_{2a}|}{range_a} \end{cases} \tag{2}$$

The Euclidean distance for numeric features and the overlap distance for categorical were demonstrated to be robust and difficult to compete with in many applications, see for example [31]. In order to calculate the weight for model i in dynamic integration for a new instance $\mathbf{x}$, we suggest to use:

$$w_i(\mathbf{x}) = \sum_{j=1}^{k} \left(\sigma(\mathbf{x},\mathbf{x}_j) \cdot mr_i(\mathbf{x}_j)\right) \Big/ \sum_{j=1}^{k} \sigma(\mathbf{x},\mathbf{x}_j) \tag{3}$$

where k is the size of the neighbourhood, $\sigma(\mathbf{x},\mathbf{x}_j)$ is a distance-based relevance coefficient and $mr_i(\mathbf{x}_\mathrm{j})$ is the margin (4) of model i on jth nearest neighbour of $\mathbf{x}$. Margin is defined as usual for a classifier with crisp outputs (1 for a correct prediction, and –1 for a wrong one):

$$mr_i(\mathbf{x}) = \begin{cases} 1, & h_i(\mathbf{x}) = y(\mathbf{x}) \\ -1, & h_i(\mathbf{x}) \neq y(\mathbf{x}) \end{cases} \tag{4}$$

In fact, weight (3) represents the expected margin of model i at instance $\mathbf{x}$. We normalize the weights (3) to be non-negative and to sum to one in order to apply them in (locally) weighted voting in dynamic integration. After the weights (3) are calculated, the three dynamic integration functions (DS, DV and DVS) are applied as described above.

The distance-based weight coefficient $\sigma(\mathbf{x},\mathbf{x}_j)$ should reflect similarity between the two instances. In our experiments we use the inverse HEOM distance as the corresponding distance-based weight coefficients:

$$\sigma(\mathbf{x},\mathbf{x}_j) = 1/d_{heom}(\mathbf{x},\mathbf{x}_j) \tag{5}$$

A simple weighting function that just raises the distance to a negative power is perhaps the most commonly used weighting function in locally weighted learning, and is often used for the Euclidean and overlap metrics [1]. It is also used in most previous experiments with dynamic integration [27, 25].

Dynamic integration was successfully applied in a number of contexts with stationary problems, outperforming other integration methods. In [27] the three dynamic integration techniques were used to combine the base classifiers generated with bagging and boosting, improving the predictive performance of the two most popular ensemble techniques. In [25] dynamic integration was considered in the context of ensembles with base classifiers generated on different feature subsets (using so-called ensemble feature selection). In [18] an adaptation of the three dynamic integration techniques to regression is considered and applied for ensembles generated using the random subspace method.

In the context of ensembles for handling concept drift the most commonly used integration techniques are voting and weighted voting [13, 20, 21, 28], although as it is demonstrated in experiments in this paper they are not always the most appropriate techniques.

5 Data Sets Used

In this section we consider the data sets used in our experiments. These include two synthetically generated problems (the *SEA concepts* problem, and the *rotating hyperplane* problem[1]), and a real-world data set from the domain of antibiotic resistance in nosocomial infections. It is important to note that the underlying concepts in all these data sets, as it is often so with real-world data, while changing with time, are still relatively stable in substantial parts of data, so that more base classifiers of the ensemble are reliable in these parts, and it is natural to expect that dynamic integration – if it is able to differentiate between stable and unstable data – may be effective with these data sets.

5.1 *SEA concepts* with abrupt concept drift

In order to test our algorithm on data with *abrupt* concept drift, we generated the SEA (from Streaming Ensemble Algorithm) concepts first introduced in [21]. We followed the same procedure to generate the data. First, 60,000 random points were generated in a three-

[1] The data sets representing the synthetically generated problems are available in the *arff* format at: http://www.win.tue.nl/~mpechen/data/DriftSets/ .

dimensional feature space. The three features had values in the range $[0;10)$, and only the first two features were relevant. Those points were then divided into 4 blocks with different concepts. In each block, a data point belongs to class 1, if $f_1 + f_2 \leq \theta$, where f_1 and f_2 represent the first two features, and θ is a threshold value for the two classes. Threshold values of 8, 9, 7, and 9.5 were used for the four data blocks. 10% class noise was then introduced into each block of data by randomly changing the class value of 10% of instances. Finally, 2,500 points from each block were reserved as a fixed test set for the corresponding concept.

5.2 *Rotating hyperplane* problem with gradual concept drift

To evaluate the ability of dynamic integration to handle *gradual* concept drift, we apply it to the commonly used synthetic benchmark data based on a rotating hyperplane. The hyperplane problem has been actively used to simulate time-changing concepts because it allows controlling the type and rate of concept drift, context recurrence, presence of noise, and irrelevant attributes [6, 13, 28].

A hyperplane in *d*-dimensional data is denoted by equation $\sum_{i=1}^{d} a_i x_i = a_0$. We use the same hyperplane settings as in [6]. We label examples satisfying $\sum_{i=1}^{d} a_i x_i \geq a_0$ as positive, and $\sum_{i=1}^{d} a_i x_i < a_0$ as negative. We generate random examples uniformly distributed in space $[0,1]^d$. Weights a_i $(1 \leq i \leq d)$ are initialized randomly in the range of [0,1]. We choose the value of a_0 so that the hyperplane divides the space into two parts of the same volume, that is, $a_0 = \frac{1}{2}\sum_{i=1}^{d} a_i$. Noise is introduced by randomly switching the labels of *p*% of the examples (5% in our experiments).

Concept drift is simulated by a series of parameters. *K* specifies the total number of dimensions whose weights are changing. *T* specifies the magnitude of change (every *N* examples) for weights $a_1,\ldots,a_k$, and $s_i \in \{-1,1\}$, $1 \leq i \leq K$ specifies the direction of change for each weight. Weights change continuously, i.e. a_i is adjusted by $s_i \cdot T / N$ after each example is generated. Furthermore, there is a possibility of 10% that the change would reverse its direction after every *N* examples; that is s_i is replaced by $-s_i$ with probability 10%. Also, each time the weights are updated, we recompute $a_0 = \frac{1}{2}\sum_{i=1}^{d} a_i$ so that the class distribution is not changed.

We generated 9 data sets of 10 000 instances and 10 dimensions (*d*=10) each, using different combinations of the number of changing dimensions *K* (2, 5, and 8), and the magnitude of change *T* (0.1, 0.5, and 1; every 1000 examples).

5.3 Antibiotic resistance data

An important problem with most real-world data sets in known experimental investigations with concept drift is that there is little concept drift in them, or the drift is introduced artificially [23]. In contrast to that, in our experiments we also use a real-world data set with a significant amount of inherent concept drift. The data set represents the problem of antibiotic resistance in nosocomial infections. Antibiotic resistance is an especially difficult problem with nosocomial infections in hospitals because pathogens attack critically ill patients who are more vulnerable to infections than the general population and therefore require more antibiotics [8, 22].

The data for our analysis were collected in the N.N. Burdenko Institute of Neurosurgery, Russia, over the years 2002-2004, using a bacterial analyzer *Vitek-60* (developed by *bioMérieux, www.biomerieux-vitek.com*).

Each instance of the data used in the analysis represents one sensitivity test and contains the following features: pathogen that is isolated during the bacterial identification analysis, antibiotic that is used in the sensitivity test and the result of the sensitivity test itself (sensitive *S*, resistant *R* or intermediate *I*), obtained according to the guidelines of National Committee for Clinical Laboratory Standards (NCCLS) [7]. The information about sensitivity analysis is related to a patient, his/her demographical data (sex, age) and hospitalization in the institute (main department, days spent in ICU, days spent in the hospital before the test, etc.).

Each instance of microbiological test in the database corresponds to a single specimen that may be blood, cerebrospinal fluid (liquor), urine, etc. In this study we focus on the analysis of meningitis cases only, and the specimen is liquor. For the purposes of our analysis we picked up all 4430 instances of sensitivity tests including the meningitis cases of the years 2002-2004.

After a discussion with medical experts, we have formed new binary features for antibiotics and pathogens corresponding to a tree-like categorization of them. For antibiotics the 35 different classes of them were grouped into 4 major categories, and for pathogens the 16 different classes were categorized into 7 major groups.

Each instance included 34 features that contained information corresponding to a single sensitivity test augmented with data concerning the antibiotic used, the isolated pathogen, clinical characteristics of the patient and his/her demographics.

This data set was considered earlier in [16] where it was experimented with using different techniques including various inductive learning and dimensionality reduction algorithms and the principle of natural clustering. Besides, it was shown that there is a significant level of concept drift pertinent to this domain. The data set was divided into sets of sequential blocks corresponding to different time intervals, and these blocks were used as train and test sets sequentially using a few suggested evaluation schemes in order to analyze underlying concept and data distribution changes. A few interesting findings were discovered, including seasonal context recurring with winter and spring models, which corresponds to yearly spring and winter infection outbreaks.

6 Experimental Studies

In our experimental studies we used an implementation based on the machine learning library WEKA 3.4.2 (available at *http://www.cs.waikato.ac.nz/~ml/weka/*), which is currently perhaps the most popular library of machine learning algorithms [32]. Default settings were used in the WEKA learning algorithms used in our experiments. In Naïve Bayes, a normal distribution was assumed for numeric features, and the Laplace correction with a multiplicative factor of 1 was used in probability estimation for categorical features. C4.5 decision trees were built using 0.25 as the confidence factor for pruning and 2 as the minimum number of instances per leaf.

With all ensembles considered here we use the simple so-called *replace the loser* ensemble pruning strategy [15]. With this strategy, if the ensemble size is greater than or equal to 25, the worst classifier, according to the current validation estimates, is replaced with a new one trained on the most recent data.

An important question with any lazy learning technique is the size of neighbourhood used. We experimented with 5 different sizes of neighbourhood *k*; 7, 15, 31, 63, and 127. Naturally, accuracy normally decreases with the increase in the size of neighbourhood, becoming closer to static voting. Our experiments demonstrated that dynamic integration was not very sensitive to the size of neighbourhood. A reason for that is the locally weighted learning scheme used, with which the more distant an instance is from the current test instance, the less influence it will have on the prediction of local performance. However, the smaller neighbourhoods (7 and 15) sometimes result in noisy performance estimates and inferior accuracies (especially with DS). We continue our analysis of experimental results focusing on the size of neighbourhood equal to 31, as usually it gives the best improvement due to dynamic integration in the problems considered.

We use five integration strategies evaluated in our experiments: Voting (V), Weighted Voting (WV), Dynamic Selection (DS), Dynamic Voting (DV), and Dynamic Voting with Selection (DVS). In V, each classifier from the ensemble is considered as an equal contributor to the final decision. In WV, each base classifier receives a weight, proportional to its estimated global accuracy on $\mathbf{D}_v^t$. V and WV work in the same manner as was described in [21 , 28] for progressive evaluation with "chunk"-based data streams. The dynamic integration techniques DS, DV, and DVS are implemented as it was described in Section 4. In dynamic integration, for each instance x from $\mathbf{D}_{te}^t$, a local accuracy is predicted for each ensemble member, based on the neighbourhood of x in $\mathbf{D}_v^t$.

We used the so-called *progressive evaluation* in order to evaluate ensembles for the three problem domains in our experiments. This name was suggested in [11] and it was argued that when dealing with temporal data and prediction, the only way to get a fair estimate of classifier accuracy is to select a test set that occurs after the training set. In progressive evaluation with a real data stream, the principle of a sliding window is used to form the test set, and all the past instances preceding the current test set are considered as the training data. After a new test set is formed, the current model is updated with a new block of instances excluded from the test data (a new ensemble member is added in our case), and the model is evaluated on the test set. Usually a new test set is formed and the model is updated after a certain time interval, or after a group of instances of a certain size is accumulated.

In more detail, the progressive evaluation principle works as follows, in our case. First, we take the first two consecutive blocks (chunks) of instances as the training set $\mathbf{D}_{tr}^1$ and the test set $\mathbf{D}_{te}^1$, correspondingly. For example, the first 100 instances are used as the original training set, and the next 100 instances are used as the test set. Or, alternatively, for example, instances corresponding to January are used as the training set and instances corresponding to February are used as the test set (the division into chunks could be according to the number of instances or to a certain time interval, depending on the problem domain). Now, we are able to construct the first ensemble which contains one member only, and to evaluate it on the current test set. Then, for an ensemble containing t members, the sliding window principle is used to form a new training set $\mathbf{D}_{tr}^{t+1}$ and to add a new classifier to the ensemble and to form a new test set $\mathbf{D}_{te}^{t+1}$. For this, we use a shift ΔT corresponding to a certain number of instances (e.g., 50), or a certain time interval (e.g., half a month). Weighted Voting and all dynamic techniques need a validation set in order to evaluate the ensemble. We use the data set corresponding to the last classifier in the ensemble as the validation set: $\mathbf{D}_v^t = \mathbf{D}_{tr}^t$. In order to avoid overly optimistic estimation because of the overlap of the validation and training data, cross validation is used for the classifiers affected by the overlap.

6.1 Experimental results with the SEA concepts

In the case with the SEA concepts (where we know a priori the true concept for each moment in time), instead of a sliding test set as was described above, we are able to use a fixed test set for each current concept $\mathbf{D}_{te}^1$, $\mathbf{D}_{te}^2$, $\mathbf{D}_{te}^3$, and $\mathbf{D}_{te}^4$. The ensemble is still formed according to the sliding window principle. In Figures 2-5 classification accuracy over sequential data blocks for the synthetic SEA data is shown with Naïve Bayes and C4.5 as the learning algorithms for the base classifiers in ensembles, and with non-overlapping data blocks including 500 and 1000 instances. The figures report accuracy for the five integration strategies evaluated in our experiments.

As can be seen from the figures, dynamic integration, as expected, often recovers from concept changes faster than the voting techniques. In most cases DS has the best performance on average and adapts to concept changes faster than the other techniques (see Figure 2 for example). However, with the data blocks of 500 instances (Figures 2 and 4) DS is rather unstable and the corresponding line is considerably noisy (especially with C4.5 as the base classifier, Figure 4). We believe, the reason for that is that with smaller data blocks DS is not able to learn the patterns of base classifier errors, overfitting the noise in the validation data. This is especially true for C4.5 which has more complex classification error's patterns than Naïve Bayes, which has a linear decision boundary. In general, as in the case with stationary data, DS seems to be more susceptible to noise than the other dynamic techniques also with data with inherent concept drift. In contrast, the best performance of DS (it almost always dominates the other techniques) is in the case with Naïve Bayes and 1000 instances (Figure 3).

DVS is the second best strategy on average in these experiments, and, in line with previously reported experiments with DVS on stationary data, its behaviour is usually more stable than that of DS, combining the advantages of DS and DV. In the case with Naïve Bayes and 500 instances (Figure 4) it is the best integration strategy, improving considerably on DS. As can be seen from the figures, V and WV have comparable classification accuracy and similar behaviour, WV being usually somewhat better than V.

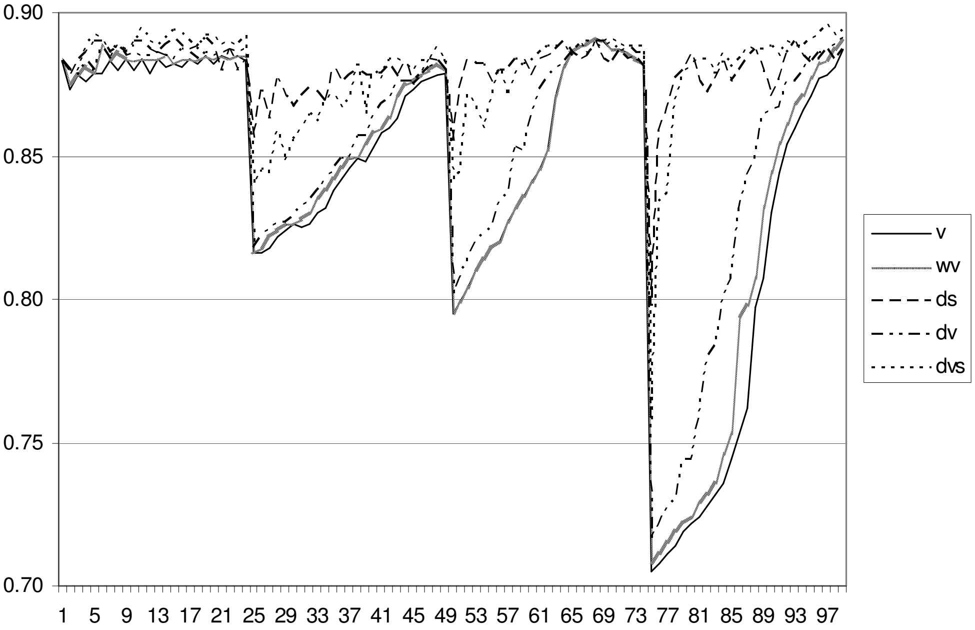

Figure 2. Classification accuracy over sequential data blocks for the synthetic SEA data (Naïve Bayes over blocks of 500 instances)

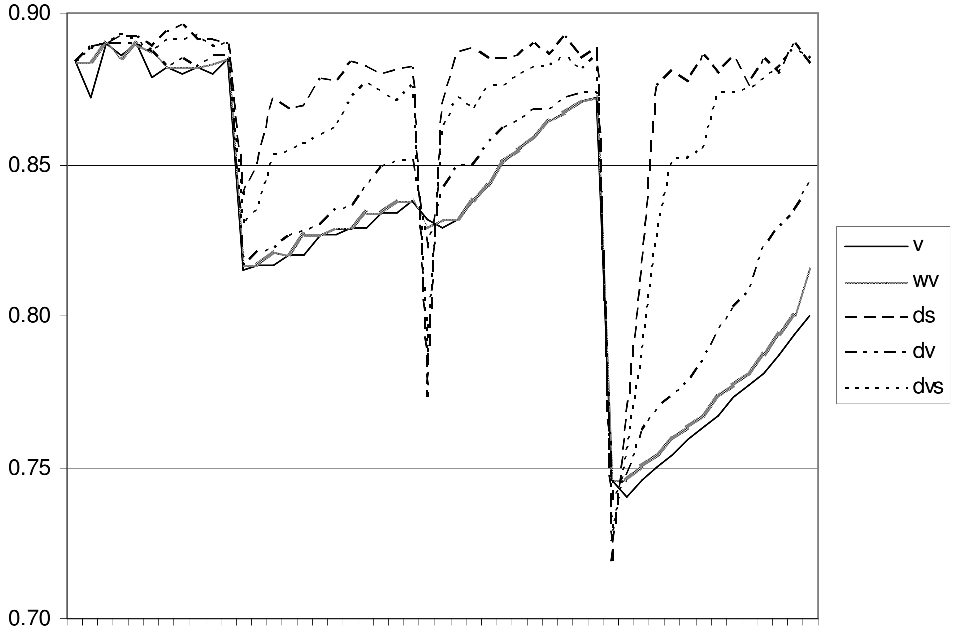

Figure 3. Classification accuracy over sequential data blocks for the synthetic SEA data (Naïve Bayes over blocks of 1000 instances)

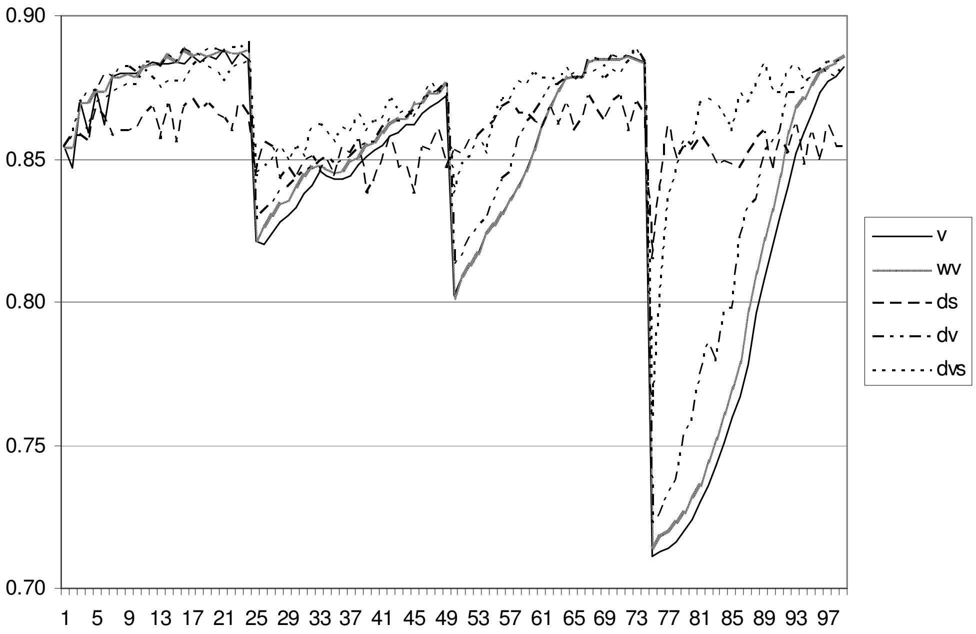

Figure 4. Classification accuracy over sequential data blocks for the synthetic SEA data (C4.5 over blocks of 500 instances)

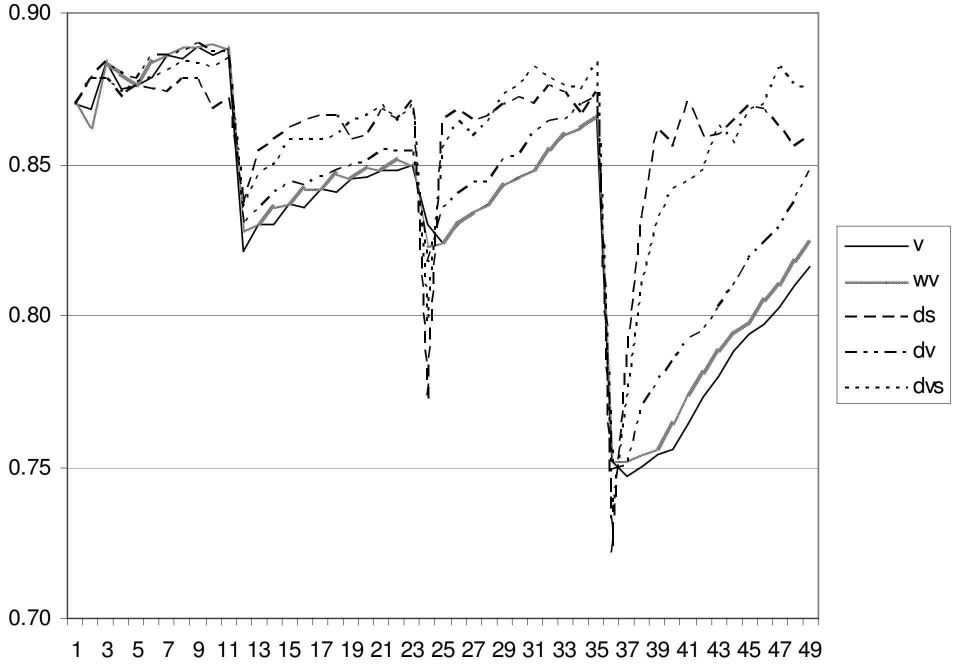

Figure 5. Classification accuracy over sequential data blocks for the synthetic SEA data (C4.5 over blocks of 1000 instances)

The patterns of adaptation to concept drift with Naïve Bayes and C4.5 are similar for the integration strategies considered, except the already mentioned difference in the behaviour of DS with the data blocks of 500 instances. The classification accuracy of Naïve Bayes and C4.5 is comparable in this domain (with the static techniques performing slightly better with C4.5, and the dynamic techniques being slightly better with Naïve Bayes).

6.2 Experimental results with the rotating hyperplane

To generate ensembles with the rotating hyperplane data, we considered four different window sizes for the sequential blocks: 250, 500, 750, and 1000; they had no overlap. The same progressive evaluation principle was used as in the previously considered domain, but the test set was not fixed as in the previous experiments; it was formed from the data stream itself as a sliding window. The most recent window before the current test block was used in order to get the validation estimates for dynamic integration and weighted voting. In order to avoid overly optimistic validation estimates for the last classifier in the ensemble, which is built on data from the current validation block, 10-fold cross validation was used.

The results for the 9 hyperplane data sets are given in Table 1. Each cell corresponds to a data set with a certain combination of *K* and *T*. Each number represents improvement in accuracy, in percents, of the best (on average) dynamic integration technique (DVS) over the best (on average) plain technique (WV). Each cell contains accuracy improvements for two learners (Naïve Bayes/C4.5 decision tree) and for the four window sizes (250, 500, 750, and 1000, from top to bottom).

Table 1.

Accuracy improvement (percents) due to dynamic integration for the synthetic data.

K/T	*T*=0.1	*T*=0.5	*T*=1
K=2	0.9/0.5	1.1/0.7	1.1/0.7
	1.4/0.3	1.9/0.9	1.8/1.1
	1.7/0.2	1.4/0.8	2.0/1.5
	1.6/0.3	1.7/1.0	1.9/2.0
K=5	0.8/0.4	2.9/1.3	5.1/3.9
	1.1/0.6	4.1/1.4	6.1/3.3
	0.7/0.8	3.9/1.0	6.9/2.8
	1.3/0.2	4.0/1.8	6.4/3.7
K=8	1.7/0.7	5.2/2.4	4.9/2.3
	2.4/0.3	6.5/2.9	6.9/1.8
	3.0/0.7	6.7/3.0	7.3/2.5
	2.7/0.1	6.9/3.2	7.3/3.1

We could see from the experimental results that Naïve Bayes is a more appropriate learner for this domain than C4.5 (accuracy with it is at least 5% better). As can be seen from Table 1, accuracy improvement due to dynamic integration, as a rule, is greater with Naïve Bayes as well (which is in line with the experiments on the SEA concepts data). For all the data sets, block sizes and the two learners there is some improvement due to dynamic integration (it is never negative).

Beside that, one can see that the accuracy improvement is, as expected, much greater for data sets with a higher level of concept drift (e.g. it reaches 7.3 for *K*=8 and *T*=1).

In Figure 6, example classification accuracy of a Naïve Bayes ensemble over the sequential blocks of synthetic data is shown for window size of 500, *T*=1, and *K*=8. The same notation is used in this figure as in Figures 2-5.

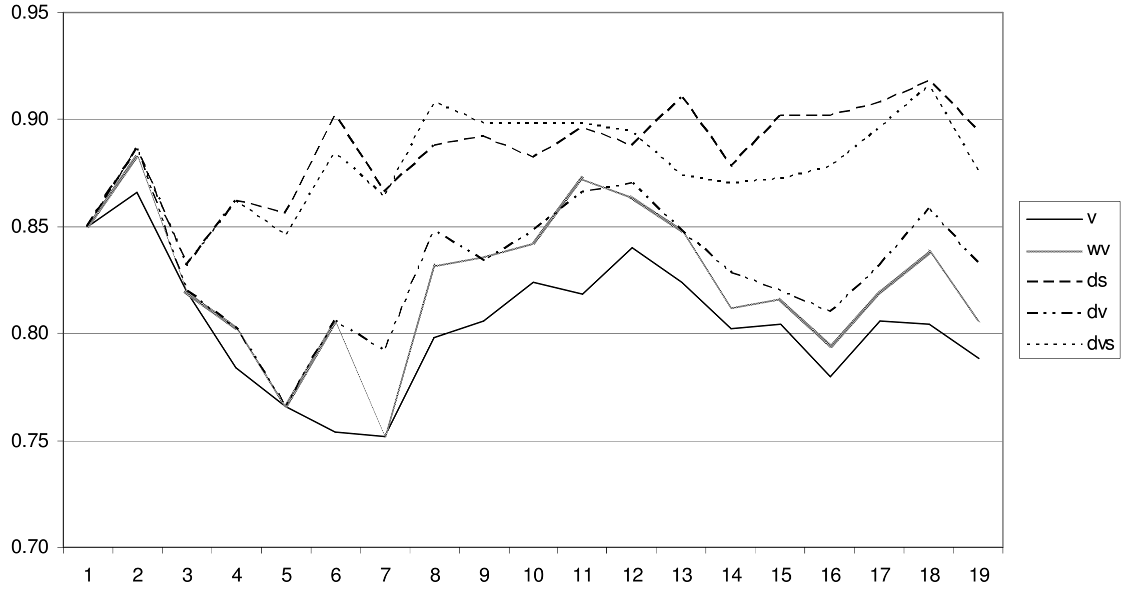

Figure 6. Classification accuracy over sequential data blocks for the synthetic *rotating hyperplane* data.

On this data set, dynamic integration achieves the best improvement over WV (with data blocks of size 500 it is 6.9%, see Table 1). One can see from this figure that DVS and DS are the best dynamic strategies on this particular data set significantly outperforming all the other integration strategies. WV is a little better than V in this domain as well. As in the case with the SEA concepts data, DV is better that the static voting techniques, but still significantly worse than

DVS and DS. In general, the same patterns appear with all the 9 data sets, but with different improvement due to dynamic integration, which is greater for data sets with greater *K* and *T*.

6.3 Experimental results with the antibiotic resistance data

As the basis of our analysis with the antibiotic resistance data we consider the classification problem aimed at predicting the sensitivity of a pathogen to an antibiotic based on data about the antibiotic, the isolated pathogen, and the demographic and clinical features of the patient.

To build ensembles, we divide the data into blocks corresponding to a certain time interval. We use the sliding window approach, and thus, when the window shift is less than the size of the window, the data blocks are not mutually exclusive (our preliminary experiments demonstrated that the use of overlapping blocks helps to improve accuracy considerably in this domain). We use the same progressive evaluation procedure as in the case with the hyperplane data. We use the last (current) data block as the test set, and the current ensemble includes only those base classifiers that are built on preceding data blocks that include different instances only with regard to the test set in order to avoid overly optimistic error estimate for the ensemble.

We have tried a number of learning algorithms available in WEKA (including Naïve Bayes, C4.5 decision tree, *k*-NN etc.) for the base classifiers in this domain. The accuracy of base models is usually radically different for different learning algorithms with this data set. For example, for one set of data blocks, the maximum base classifier accuracy with Naïve Bayes can be 0.6 only, but 0.75 with decision trees and instance-based learning. However, the final ensemble accuracy was not so different with different learning algorithms, and it always achieved 0.80 on average with the dynamic approaches (weighted average accuracy according to the number of instances in each test block).

Weighted voting is always a little better than plain majority voting, supporting our previous experiments and experiments in [2]. Dynamic integration is always better than the best base classifier and weighted voting regardless of the learning algorithm, window size and window shift in this domain. The maximum gain by dynamic integration was achieved with Naïve Bayes: for some blocks where the maximum base classifier accuracy was 0.60 only, the accuracy of dynamic integration was 0.85. On average with Naïve Bayes, the accuracy of WV was 0.65, best base classifier accuracy 0.69, and the accuracy of DVS (best dynamic strategy) 0.81. The accuracies of three dynamic strategies (DS, DV, and DVS) are always close to each other, and the relative efficacy of them depends on the selected learning algorithm, window size and window shift, however DVS and DS seem to be a little superior to DV.

In Figures 7 and 8 examples of experimental results are shown for C4.5 decision tree, window size of 3 months, and window shift of 1 month. Figure 8 reports also the minimum, average and

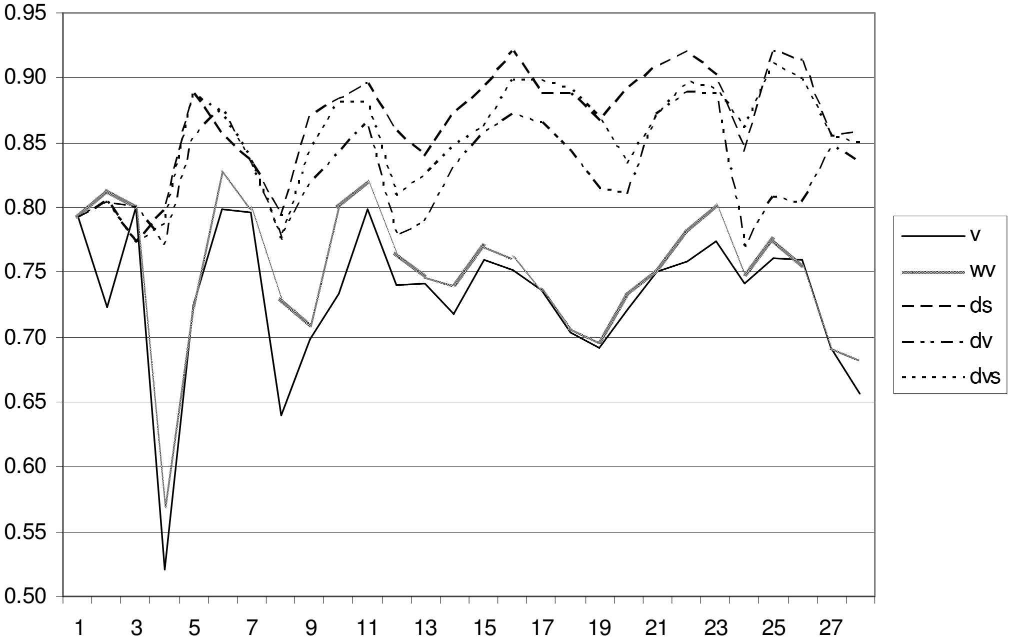

Figure 7. Classification accuracy over sequential data blocks (C4.5 ensembles) in the antibiotic resistance domain.

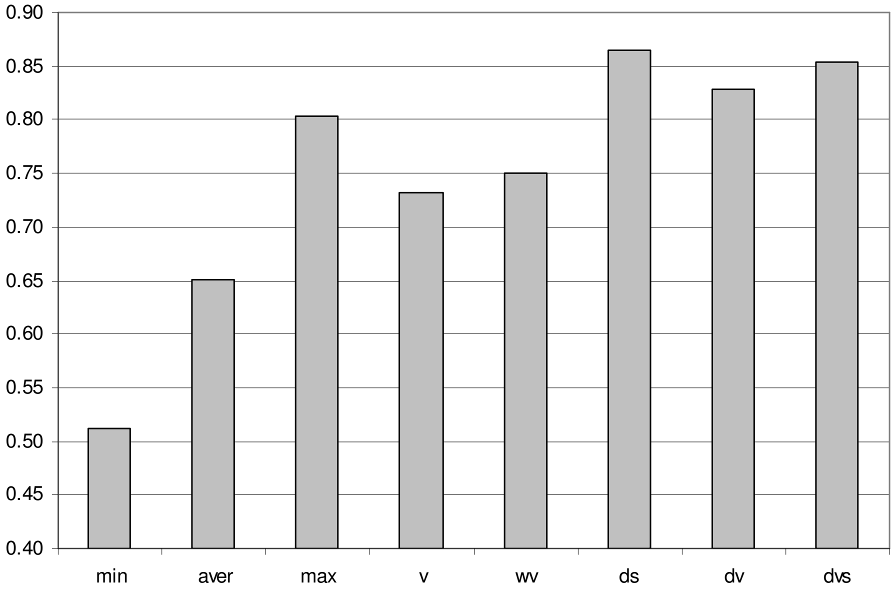

Figure 8. Weighted average of classification accuracy (C4.5 ensembles) in the antibiotic resistance domain.

maximum accuracies of the base classifiers in the ensemble (min, aver, max) averaged over the test data blocks. With this configuration, the best weighted average ensemble accuracy was achieved with this data set (0.86 with DS, Figure 8).

With our ensemble construction, the first ensembles including a few members only, naturally, are very weak. If we consider ensembles including 7 classifiers and more (omit the first 6 points in Figure 7), the average accuracy with dynamic integration is more than 0.88. From Figure 8 one can see that the dynamic integration techniques improve ensemble accuracy by more than 10% on average in this domain in comparison with the voting techniques.

7 Discussion

In this paper we considered 3 alternative dynamic integration techniques; DS, DV, and DVS. Our DS integration procedure is similar to DCS_LA (Dynamic Classifier Selection based on Local Accuracies) presented in [33], and to DCS presented in [9,10]. The main differences between DS and the others are: (1) DCS_LA does not use distances for weighting the nearest instances. (2) Both DCS_LA and DCS can be based on the local accuracy estimate that takes into account also the classification produced by the corresponding base classifier. The local accuracy is calculated in this case only for those instances from the neighbourhood that are classified into the same class as the test instance by the corresponding classifier. This local accuracy is called the *local class accuracy* in [33], and *a posteriori accuracy* in [9,10], and it was demonstrated to be slightly superior w.r.t. the simple local accuracy (which is used in DS, DV, and DVS).

As regarding these differences, it was discussed in [10] that distance-weighted local accuracies are better as they allow us to handle more effectively the problem related to the choice of the size of the neighbourhood, and provide more robust estimates of local accuracies. The use of the a posteriori local accuracy within DS, DV, and DVS is an interesting issue for further research. We presume that the a posteriori local accuracy might be superior to the simple local accuracy also for our DS, DV, and DVS integration procedures in the context of handling concept drift.

In [9] it was proposed to use the information about multiple classifier behaviour (MCB) in dynamic classifier selection (DCS). For each instance, a vector whose elements are the decisions taken by the base classifiers represents the behaviour of the ensemble for that instance [9]. The basic idea of MCB-based DCS is that the neighbourhoods for the calculation of the local accuracies should be defined not only by the proximity in the feature space, but also by the proximity in MCB. First, close instances are defined in the feature space, as usual, and then, this neighbourhood is reduced to instances that exhibit close multiple classifier behaviour with that of the test instance. This idea can be used also in our dynamic integration strategies, as it influences only the selection of the neighbourhood, which is a common phase for DS, DV, and DVS.

Besides the straightforward use of local accuracies for dynamic classifier selection (DS) as in [10] and [33], we also consider two other integration strategies, which are based on dynamic voting (namely DV and DVS). The reliability-based weighted voting (RBWV) introduced in [4] is another example of dynamic voting. It uses a model-dependent estimation of the reliability (confidence) of predictions for each particular instance instead of local accuracy as the weights. The use of confidences in predictions for the base models is another interesting alternative to consider in the future research of dynamic voting for handling concept drift.

Among the three dynamic integration techniques considered, dynamic selection (DS) has often the best performance in the present context. However, it is important to notice that DS is a technique that is very sensitive to the validation data and to the models integrated. If the validation set is not representative enough of the problem, in order to reliably predict local performance, then DS may even decrease the accuracy of static integration. This happens, for example, in the case illustrated in Figure 4. The bad behaviour of DS, as reported in the context of stationary data, can be also caused by the low accuracy of base models. For example, on weak and highly diverse decision trees generated by the Random Forests algorithm, DS was proven to always have very poor performance in comparison with other integration techniques [17, 24]. DVS, which combines the advantages of DS and DV, is a more stable technique. DVS is always close to the best, if not the best, and never degrades the performance of static integration. The same behaviour of DVS was reported with stationary problems [25, 27]. Interestingly, in the experiments considered, DV was never the best, although it is often better than DS and is the best in the stationary context. Presumably, the reason for such behaviour is that, in the experiments considered, there are too many irrelevant base classifiers in many cases, and DV fails to discard them from consideration, still assigning them some positive weight.

8 Conclusions

In the real world concepts are often not stable but change with time, which is known as the problem of concept drift. Concept drifts complicate the task of learning and require unusual solutions, different from commonly used batch learning techniques. In this paper we consider synthetically generated concept drifts and an example of concept drift from the area of antibiotic resistance. Among the most popular and effective approaches to handling concept drift is ensemble learning, where a set of concept descriptions built on data blocks corresponding to different time intervals is maintained, and the final prediction is the aggregated prediction of ensemble members.

In this paper we suggest a dynamic integration approach for ensembles to handle concept drift. Our approach integrates the base classifiers at an instance level, assigning them weights proportional to their local accuracy on each instance considered.

Our experiments have demonstrated that dynamic integration often results in better accuracy with the considered data sets than the more commonly used weighted voting. This supports our hypothesis that dynamic integration can be a more appropriate integration technique for handling concept drift, and that it may be especially useful in the presence of local concept drift.

As a direction for future research, it is interesting to compare dynamic integration with other techniques that might be effective in handling local concept drift, including case-base maintenance [5] and the systematic data selection of Fan [6].

We believe that dynamic integration of classifiers can become a reliable tool to fight concept drift for data streams in many subject areas. As another direction for future research we plan to consider its application to other data sets with inherent concept drift, including biomedical and financial data streams, where concept drift is a common phenomenon.

Acknowledgements. This material is based upon work supported by the Science Foundation Ireland under Grant No. S.F.I.-02/N.1/111. This research was also partly supported by the Academy of Finland. We are thankful to Prof. Michael Shifrin and Dr. Irina Alexandrova from N.N.Burdenko Institute of Neurosurgery, Moscow, Russia for the database used in our study. We are grateful to the anonymous reviewers and to the guest editors of this special issue for their valuable comments and constructive criticism.

References

1. Atkeson C., A. Moore, S. Schaal, Locally weighted learning. AI Review, 11, 1997, 11-73.
2. Bauer E., Kohavi R. An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Machine Learning, 36 (1,2), 1999, 105-139.
3. Brodley C., Lane T. Creating and exploiting coverage and diversity. In: Proc. AAAI-96 Workshop on Integrating Multiple Learned Models, Portland, OR, 1996, 8-14.

4. Cordella L.P., P. Foggia, C. Sansone, F. Tortorella, M. Vento, Reliability parameters to improve combination strategies in multi-expert systems, Pattern Analysis and Applications, 2(3), 1999, 205-214.
5. Delaney S.J., Cunningham P., Tsymbal A., Coyle L. A case-based technique for tracking concept drift in spam filtering, Knowledge-Based Systems, Elsevier, 18 (2-3), 2005, 187-195.
6. Fan W. Systematic data selection to mine concept-drifting data streams. In: Proc. 10th Int. Conf. on Knowledge Discovery and Data Mining KDD'04, 2004.
7. Ferraro M.J., et al. Methods for dilution antimicrobial susceptibility tests for bacteria that grow aerobically: approved standard; Performance standards for antimicrobial susceptibility testing. Wayne, PA: National Committee for Clinical Laboratory Standarts, NCCLS, 2005. (Documents M7-A6 and M100-S14, *www.nccls.org*).
8. Gaynes R.P. Surveillance of nosocomial infections: a fundamental ingredient for quality. Infect Control Hosp Epidemiol, 18(7), 1997, 475– 478.
9. Giacinto G., F. Roli, Dynamic classifier selection based on multiple classifier behaviour, Pattern Recognition, 34 (9), 2001, 1879-1881.
10. Giacinto G., F. Roli, Methods for dynamic classifier selection. In: Proc. ICIAP '99, 10th International Conference on Image Analysis and Processing, IEEE CS Press, 1999, 659-664.
11. Harries M., Horn K. Detecting concept drift in financial time series prediction using symbolic machine learning. In: 8th Australian Joint Conf. on Artificial Intelligence, World Scientific, 1995, 91-98.
12. Klinkenberg R. Learning drifting concepts: example selection vs. example weighting, Intelligent Data Analysis, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, 8 (3), 2004.
13. Kolter J.Z., Maloof M.A. Dynamic weighted majority: a new ensemble method for tracking concept drift. In: 3rd Int. Conf. on Data Mining ICDM'03, IEEE CS Press, 2003, 123-130.
14. Kukar M. Drifting concepts as hidden factors in clinical studies. In: Proc. 9th Conf. on Artificial Intelligence in Medicine in Europe, AIME 2003, Springer, LNCS, 2003, 355-364.
15. Kuncheva L.I., Classifier ensembles for changing environments. In: Proc. 5th Int. Workshop on Multiple Classifier Systems, MCS'2004, Springer, LNCS, 3077, 1-15.
16. Pechenizkiy M., Tsymbal A., Puuronen S., Shifrin M., Alexandrova I. Knowledge discovery from microbiology data: many-sided analysis of antibiotic resistance in nosocomial infections. In: 3rd Int. Conf. on Professional Knowledge Management: Experience and Visions (WM05), Kaiserslautern, Germany, Springer, LNAI 3782, 2005, 360-372.
17. Robnik-Sikonja M. Improving random forests. In: J.F. Boulicaut at al. (eds.), Proc. 15th

European Conf. on Machine Learning ECML 2004, Springer, LNCS 3201, 2004, 359-370.

18. Rooney N., D. Patterson, S. Anand, A. Tsymbal, Dynamic integration of regression models. In: Proc. 5th Int. Workshop on Multiple Classifier Systems MCS'2004, LNCS 3181, Springer, 2004, 164-173.
19. Schaffer C. Selecting a classification method by cross-validation, Machine Learning, 13, 1993, 135-143.
20. Stanley K.O. Learning concept drift with a committee of decision trees, Tech. Report UT-AI-TR-03-302, Dept of Computer Science, University of Texas at Austin, USA, 2003.
21. Street W., Kim Y. A streaming ensemble algorithm (SEA) for large-scale classification. In: Proc. 7th Int. Conf. on Knowledge Discovery and Data Mining KDD-2001, 2001, 377-382.
22. The Problem of Antibiotic Resistance, NIAID Fact Sheet. National Institute of Allergy and Infectious Diseases (NIAID), National Institutes of Health, U.S. Dept of Health and Human Services, USA, April 2004 (available at *www.niaid.nih.gov/factsheets/antimicro.htm*).
23. Tsymbal A. The problem of concept drift: definitions and related work, Technical Report TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Ireland, 2004.
24. Tsymbal A., M. Pechenizkiy, P. Cunningham, Dynamic integration with random forests. Tech Report TCD-CS-2006-23, Dept of Computer Science, Trinity College Dublin, Ireland, 2006 (available online at *http://www.cs.tcd.ie/publications/tech-reports/reports.06/*).
25. Tsymbal A., M. Pechenizkiy, P. Cunningham, Sequential genetic search for ensemble feature selection. In: Proc. 19th Int. Joint Conf. on Artificial Intelligence IJCAI'2005, Morgan Kaufmann, 2005, 877-882.
26. Tsymbal A., M. Pechenizkiy, P. Cunningham, S. Puuronen, Handling local concept drift with dynamic integration of classifiers: domain of antibiotic resistance in nosocomial infections. In: Proc. 19th IEEE Int. Symposium on Computer-Based Medical Systems CBMS'2006, IEEE CS Press, 2006 (to appear).
27. Tsymbal A., Puuronen S. Bagging and boosting with dynamic integration of classifiers. In: D.A. Zighed, J. Komorowski, J. Żytkow (eds.), Principles of Data Mining and Knowledge Discovery, Proceedings of PKDD 2000, Springer, LNAI 1910, 2000, 116-125.
28. Wang H., Fan W., Yu P.S., Han J. Mining concept-drifting data streams using ensemble classifiers. In: Proc. 9th Int. Conf. on Knowledge Discovery and Data Mining KDD'03, 2003, 226-235.
29. Widmer G., Kubat M. Effective learning in dynamic environments by explicit context tracking. In: 6th European Conf. on Machine Learning ECML'93, Springer, LNCS 667, 1993, 227-243.
30. Widmer G., Kubat M. Learning in the presence of concept drift and hidden contexts,

Machine Learning, 23 (1), 1996, 69-101.

31. Wilson D.R., T.R. Martinez, Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, 6(1), 1997, 1-34.
32. Witten I., Frank E. Data Mining: Practical Machine Learning Tools With Java Implementations, San Francisco: Morgan Kaufmann, 2000.
33. Woods K., W.P. Kegelmeyer, K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Trans. PAMI, 19 (4), 1997, 405-410.