Learning Concept Drift Using Adaptive Training Set Formation Strategy

Nabil M. Hewahi, Computer Science Department, Faculty of Information Technology, Islamic University of Gaza, Gaza, Palestine

Sarah N. Kohail, Computer Science Department, Faculty of Information Technology, Islamic University of Gaza, Gaza, Palestine

ABSTRACT

We live in a dynamic world, where changes are a part of everyday life. When there is a shift in data, the classification or prediction models need to be adaptive to the changes. In data mining the phenomenon of change in data distribution over time is known as concept drift. In this research, the authors propose an adaptive supervised learning with delayed labeling methodology. As a part of this methodology, the atuhors introduce Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL), which is based on selective training set formation. Our proposed solution is considered as the first systematic training set formation approach which takes into account delayed labeling problem. It can be used with any base classifier without the need to change the implementation or setting of this classifier. The authors test their algorithm implementation using synthetic and real dataset from various domains which might have different drift types (sudden, gradual, incremental recurrences) with different speed of change. The experimental results confirm improvement in classification accuracy as compared to ordinary classifier for all drift types. The authors' approach is able to increase the classifications accuracy with 20% in average and 56% in the best cases of our experimentations and it has not been worse than the ordinary classifiers in any case. Finally a comparison with other four related methods to deal with changing in user interest over time and handle recurrence drift is performed. These methods are simple incremental method, time window approach with different window size, instance weighting method and conceptual clustering and prediction framework (CCP). Results indicate the effectiveness of the proposed method over other methods in terms of classification accuracy.

Keywords: *Adaptive Learning, Concept Drift, Delayed Labeling, Machine Learning, Training Set Formation*

1. INTRODUCTION

A key assumption in supervised learning is that the training and the testing data (or operational data) used to train the classifier come from the same distribution. This means that training data is representative and the classifier will perform well on all future unseen data instances. However, if the statistical properties of the target variable, which the model is trying to predict, change over time while the same classifier is still applicable, the prediction will be no longer

DOI: 10.4018/jtd.2013010103

accurate. In machine learning this phenomenon of change in data distribution over time is known as concept drift (Tsymbal, 2004). Concept drift problem have been stated as the tenth challenging problem facing researchers in data mining and machine learning fields (Yang & Wu, 2006).

To show the importance of this problem, assume a data mining application for spam filtering that is developed using the latest generated spam dataset. As this filter adapted to deal with today's types of spam emails, the spammers will try to bypass the spam filters by disguising their emails to look more like legitimate. So new spam will be generated and the current application will go toward approximation to classify this strange pattern. As time goes by, this will lead to less accurate, poor performance and incorrect knowledge. This dynamic nature of spam email raises a requirement for update in any filter that is to be successful over time in identifying spam (Delany, Cunningham, Tsymbal, & Coyle, 2005).

The main difficulty in mining non-stationary data like spam, intrusion, stock marketing, weather and customer preferences is to cope with the changing of data concept. The fundamental processes generating most real-time data may change over years, months and even seconds, at times drastically. Effective learning in environments with hidden contexts and concept drift requires a learning algorithm that can detect context changes without being explicitly informed about them, recover quickly from a context change and adjust itself to the new context, and can make use of previous experience in situations where old contexts and corresponding concepts reappear (Nishida & Yamauchi, 2009).

In our research, we try to add a contribution to scientific research in solving the problem of concept drift in supervised learning when true labels become known with certain delay. The work presented in this paper is based on training set formation strategy which is reforming the training sets when concept drift is detected. Training set formation methods have an advantages over other adaptivity methods since they do not require complicated parameterization and they can be used for online learning plugging in different types of base classifiers. We can summarize our contribution as:

- We introduce Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL), which is based on selective training set formation. Our proposed solution is considered as the first systematic training set formation approach that take into account delayed labeling problem. Our proposed algorithm can be used with any base classifier without the need to change the implementation or setting of the classifier;
- We test our algorithm implementation using synthetic and real dataset from various domains which might have different drift types (sudden, gradual, incremental recurrences) with different speed of change. Experimental evaluation confirms improvement in classification accuracy as compared to ordinary classifier for all drift types.

The rest of the paper is organized as follows: Section 2 presents related work and gives an introductory background to the main topic of this research, namely concept drift problem and detectability of concept drift when labeled is delayed. Section 3 defines training set formation strategy and summarize the main contributions of our research. Section 4 describes our methodology and proposed algorithms. Experimental results discussed in Section 5. Finally Section 6 concludes the paper.

2. RELATED WORK

2.1. Learning under Concept Drift

In supervised learning, each example is a pair of objects input vectors x and output labels y. The task is to interfere a function F that is able to predict the output labels y', having input vectors of a testing data x'. First present of concept drift causes was by Kelly et al. (1999). They claim that change in outcome distribution

(concept drift) may occur in three ways: Firstly, and most simply, the prior probability for the class, *p(y)* may change over time. Secondly, the distributions of the classes may change; that is, the *p(xly)*, may alter over time. Thirdly, the posterior distributions of class memberships, the *p(y|x)* may alter. Where x is an instance in q-dimensional feature space and $y \epsilon \{ c_1, ..., c_m \}$, the set of class labels.

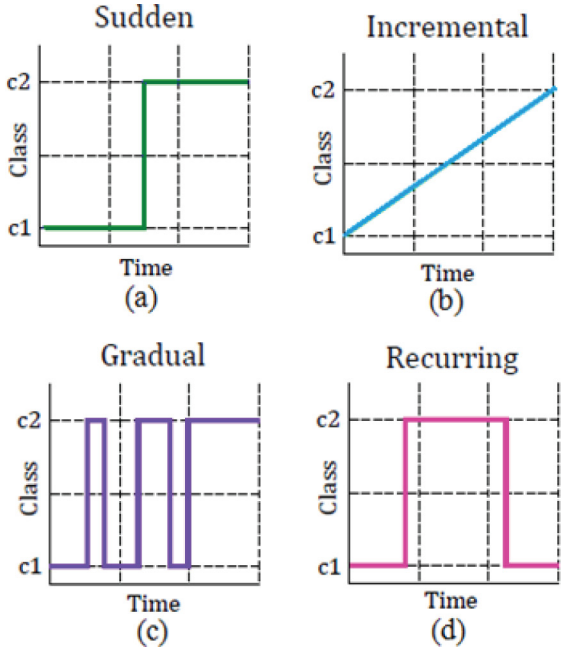
Brzezinski (2010) identifies four main types of drift which may occur in a single variable along time assuming one dimensional data. By drift types we mean the patterns the data sources take over time. The types of change in context/concept are defined based on those patterns.

The simplest pattern of a change is sudden drift illustrated in Figure 1a. Sudden drift shows abrupt changes that instantly and irreversibly change the variables class assignment. Real life examples of such changes include change in e-commerce environment and stock prices.

The next two plots Figure 1b and Figure 1c illustrate changes that happen slowly over time thus the drift is noticed only when looking at a long time period. Incremental drift occurs when variables slowly change their values over time, we can see it as a sequence of small sudden drifts. Gradual drift occurs when the change involves the class distribution of variables. Some researchers do not distinguish these two types of drift and use the terms gradual and incremental interchangeably. A typical example of incremental drift is price growth due to inflation, whilst gradual changes are represented by slowly changing definitions like spam or user-interesting news feeds (Brzezinski, 2010).

The fourth type of drift illustrated in Figure 1d is referred as reoccurring concepts. It happens when several data generating sources are expected to switch over time at irregular time intervals. Thus previously active concepts reappear after some time. This drift is not certainly periodic, it is not clear when the source might reappear, that is the main difference from seasonality concept used in statics. An example of reoccurring drift is changing in food sales.

Figure 1. Illustration of the four structural types of concept drift (Brzezinski, 2010)



2.2. Concept Drift under Delayed Labeling

Most of the work to date on drift detection assumes that the true class of all instances in the data stream will be known shortly after classification (Delany, Cunningham, Tsymbal, & Coyle, 2005; Ludmila, Kuncheva, & Salvador, 2008; Wang, Fan, Yu, & Han, 2003). Under such assumption, the incoming new data can be regularly used to periodically examine the model and compute the real error. In real time, this scenario is not realistic because decisions should be made at real time and in many domains collecting new labeled training objects may be costly (e.g. require sensors and hardware systems) or time-consuming (e.g. require human experts to manually label the new data). While it is relatively easy to obtain unlabelled objects, it still challenging to detect changes using these objects, especially when the prior probability for the class changed. Examples of tasks where delayed labeling exist are sales prediction, bankruptcy prediction, outcome of patient treatment, intrusion or fraud detection and spam categorization tasks.

Dealing with delayed labeling problem will allow the learner to benefit from unlabelled data (i.e. early change detection) until the true labels become available.

3. TRAINING SET FORMATION ADAPTIVITY STRATEGY

Training set formation strategy can be achieved by using one or more of the following methods:

1. **Training set selection:** Used to select the most relevant examples to current concept. The relevancy here related to how representative or important older examples are for predicting new instances of the possibly changed concept. For example, instead of taking all the training history, a number of the instances that is strongly related to the current distribution are considered. Training set selection can be applied in two ways (Tsymbal, 2004; Žliobaitė, 2010): (a) Sequential instance selection (training windows strategies) which select the nearest neighbors according to example arrival time, so the latest examples are more trusted than oldest ones. (b) Selective sampling (instance selection) which pick the closest instances to the target instance according feature space. Selective sampling in space is particularly beneficial when reoccurring or gradual concepts are expected;
2. **Training set weighting:** In this case instances can be weighted according to their age, and their competence with regard to the current concept. Klinkenberg (Klinkenberg, 2004) claimed that instance weighting techniques handle concept drift worse than analogous instance selection techniques, which is probably due to overfitting;
3. **Training set manipulation:** When drifting happened, features or even combinations of attribute values that were relevant in the past may no longer be useful, some labels may disappear and new labels may occur. Training set manipulation is used for feature reselection, adding new labels that appear with time and delete labels that disappear with time.

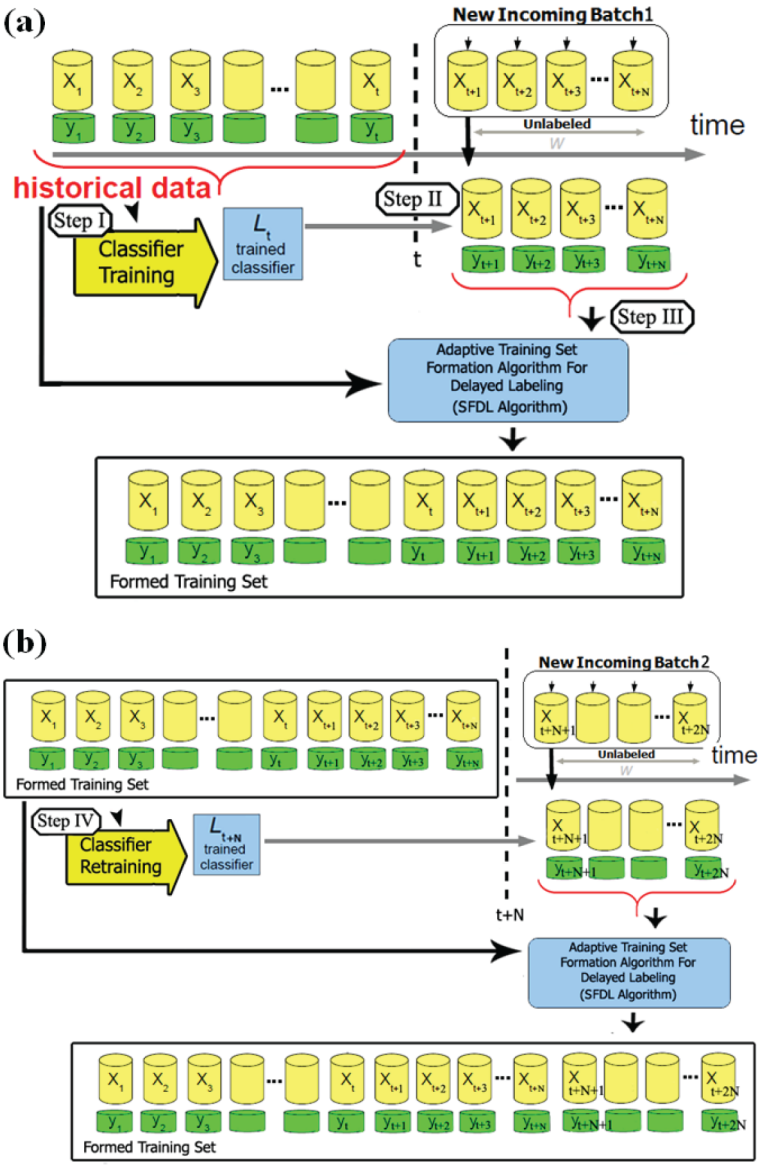
4. METHODOLOGY AND PROPOSED APPROACH

We organize this section as follows: Section 4.1 provides a general idea about the methodology flow. Section 4.2 and section 4.3 explain two important algorithms which have been created and used in our main approach. Finally, section 4.4 discusses our proposed Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL).

4.1. Overview of Our Solution

Figure 2 provides a global view for concept drift learning scenario that we build. To make the flow clear and complete, we illustrate a

Figure 2. Global view for concept drift learning scenario using the pro-posed approach



scenario for the arrival of two new consequent data batches in Figure 2a and Figure 2b.

Figure 2 summarizes our methodology in five general steps:

Step 1: Like most of previously proposed drift learning methods, we used supervised learning as initial training method. After training and testing a classifier, L_t is produced. Classifier L_t is considered as the best and accurate classifier at time t;

Step 2: When the system receives a new instances (a batch from a drifting concept), the new instances will be classified using L_t classifier. This process will continue until a set of row instances of window size *w*

arrived [x_{t+1} to x_{t+N}]. Window size value is fixed for a single system and it depends on the system designer knowledge of context;

Step 3: Apply our proposed algorithm named Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL) to the old historical data which have been used to build the L_t classifier, and the new incoming batch. The work of this algorithm is summarized as follows:

- Select the most relevant instances to current concept (Instance Selection);
- Reclassifying the new arrived batch using the selected instances;
- Reform the old set according to the changes detected;

Step 4: The output of the previous step is a new formed training set which reflect the changes occurred during the period [*t+1* to *t+N*]. This set will be used to retrain the model and produce L_{t+N} classifier as illustrated at Figure 2b;

Step 5: When receiving another new batch, the process will be repeated from step II and so on.

4.2. Modified *k*-Nearest Neighbor Algorithm

The *k*-Nearest Neighbors (*k*-NN) algorithm is the most common instance-based method (Ludmila, Kuncheva, & Salvador, 2008; Nishida, 2008). It classifies objects based on closest training examples in the feature space. The training phase consists of simply storing every training example with its label. To make a classification for a new example, first compute its distance to every training example. For numeric attributes, the distance is usually defined in terms of the standard Euclidean distance. Euclidean distance between two points x_z and x_l where each point is a q-dimensional real feature vector is computed as follows:

$$d\left(x_z, x_l\right) = \sqrt{\sum_{i=1}^{q} \left| x_z^{(i)} - x_l^{(i)} \right|^2} \quad (1)$$

where $x_z^{(i)}$ is the ith feature of the instance x_z and q is the dimensionality.

For Boolean and discrete attributes, the distance is usually defined in terms of the number of attributes that two instances do not have in common. *k*-NN then keep the k closest training examples in distance, where $k \geq 1$ is a fixed integer. The new example is classified by a majority vote of its neighbors. Figure 3 shows the pseudocode of *k*-NN algorithm.

In addition to the class label outputted by k-NN classifier, we modified the *k*-NN so it can output two additional class labels y'' and y''' for the same example. The basic idea of the algorithm does not change, but we add two more computations, one for y'' and the other y'''. The purpose of doing these computations is to decide later which class label should be assigned to the given drifted example. The

Figure 3. k-NN algorithm

#	**Pseudocode for the basic *k*-NN classifier**
1	**Input:** Training set D = {($\boldsymbol{x_1}$, $\boldsymbol{y_1}$), . . . , ($\boldsymbol{x_n}$, $\boldsymbol{y_n}$)}
2	$\boldsymbol{x'}$ new instance to be classified
3	**Output:** predicted class label $\boldsymbol{y'}$ for $\boldsymbol{x'}$
4	**ALGORITHM**
5	FOR each labeled instance ($\boldsymbol{x_i}$, $\boldsymbol{y_i}$) calculate d($\boldsymbol{x_i}$, $\boldsymbol{x'}$) from equation 1
6	Order d($\boldsymbol{x_i}$, $\boldsymbol{x'}$) from lowest to highest, (i = 1, . . . , n)
7	Select the *k* nearest instances to $\boldsymbol{x'}$: $D_{\boldsymbol{x'}}$
8	Output $\boldsymbol{y'}$ that is the most frequent class in $D_{\boldsymbol{x'}}$

details of this process and how the values of y″ and y‴ are used will be explained in the following sections. Modified k-NN algorithm is illustrated in Figure 4:

- **Computing y″:** After ordering the examples according to its distance from the new instance to be classified x' (line 6), we select the nearest k instances from each available class j. We denote the set by $D^{(j)}$, where j = 1,…., η and η is the number of available classes. Then y'' is assigned to the class which have the minimum summation of its distances from x' $Summ_j$;
- **Computing y‴:** After selecting the k nearest instances (line 9) we add the distances of each group of instances that belong to one class and then divide it by the number of nearest neighbor instances belong to that class label from the total k.

4.3. Closest Class Algorithm

We develop this algorithm as a heuristic to get the nearest class to each existing classes. Many other methods calculate the distance between centers directly to get how much one class is far from the others. These methods may not work well when the distribution of the instance points belong to a certain class label are scattered and non-intensive. This heuristic guide the algorithm and identify the changes in classes distribution. It also decide how to change the class label when there is a drift especially when the drift is gradual.

Figure 4. Modified k-NN algorithm

#	**Pseudocode for the modified k-NN classifier**
1	**Input:** Training set D = $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$
2	$\boldsymbol{x}'$ new instance to be classified
3	**Output:** predicted three different class labels y', y'', y''' for $\boldsymbol{x}'$
4	**ALGORITHM**
5	FOR each labeled instance $(\boldsymbol{x}_i, y_i)$ calculate d($\boldsymbol{x}_i$, $\boldsymbol{x}'$) from equation 1
6	Order d($\boldsymbol{x}_i$, $\boldsymbol{x}'$) from lowest to highest, (i = 1, . . . , n).
7	Select k nearest instances to $\boldsymbol{x}'$ that belong to class j, (j = 1,…., η):$D^{(j)}$,
8	η is the number of classes.
9	Select the k nearest instances to $\boldsymbol{x}'$: $D_{\boldsymbol{x}'}$
10	Output y' that is the most frequent class in $D_{\boldsymbol{x}'}$
11	
12	FOR each class j
13	$Summ_j=\sum_{z=1}^{\{D^{(j)}\}} d(\boldsymbol{x}_z, \boldsymbol{x}')$, (where $\{D^{(j)}\}$ is the number of instances in $D^{(j)}$)
14	END FOR
15	y'' = class with minimum $Summ_j$
16	
17	**FOR** each class label j
18	Get all instances from $D_{\boldsymbol{x}'}$ that belong to class j : $D_{\boldsymbol{x}'}{}^{(j)}$
19	**IF** $\{D_{\boldsymbol{x}'}{}^{(j)}\}\neq 0$ ($\{D_{\boldsymbol{x}'}{}^{(j)}\}$ is the number of instances belong to class j from the whole set $D_{\boldsymbol{x}'}$)
20	$S_j = \sum_{z=1}^{\{D_{\boldsymbol{x}'}{}^{(j)}\}} d(\boldsymbol{x}_z, \boldsymbol{x}') / \{D_{\boldsymbol{x}'}{}^{(j)}\}$
21	**END IF**
22	**END FOR**
23	y''' = class with minimum s_j

Figure 5 illustrates the pseudocode for computing the closest class for each existing class. The input for this algorithm is the whole training set and the output is the closest class label for each class available in the training set. It is to be mentioned that if class X is the closest class to class O it is not necessary that class O is the closest class to X. To compute the closest class for a given class c_i, (i= 1,...., η), first the algorithm compute the average between every two classes. The average is computed as follows:

1. The algorithm will group all the instances according to their class label;
2. For each two different classes i and j;
3. For each instance belong to first class i:
   a. Random instance will be picked from the second class j;
   b. Euclidean distance between the two instances will be computed and added to summation S;
4. Summation S will be divided on the number of instances of the class which have the minimum number of instances (either i or j);
5. Now we have a single average for each pair of classes. The number of averages is equal to Binomial Coefficient $\binom{\eta}{2}$ where order is not important. This means, we have η classes, and we want to pick two (pair) of them each time with no repetition;
6. The closest class for a given class c will be the class which have the minimum average of distances from class c.

4.4. Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL)

The idea of training set formation strategy is to continually update the training data and form it according to detected changes from the new

Figure 5. Computing the closest class to each available class

#	**Pseudocode for computing the closest class to each available class**
1	**Input:** Training set D
2	**Output:** closest class label to each available class $y^{closest}$
3	**ALGORITHM**
4	Separate instances that belong to each class label in different set:
5	$(D^{(1)},..., D^{(\eta)})$, η is the number of classes
6	**FOR** j=1 to η-1
7	**FOR** i=j+1 to η where $(j \neq i)$
8	**FOR** z=1 to $\{D^{(j)}\}$ (where$\{D^{(j)}\}$ is the number of instances belong to class j)
9	pick one instance from $D^{(j)}$: $\boldsymbol{x}_z$
10	pick random instance from $D^{(i)}$: $\boldsymbol{x}_r$
11	$\boldsymbol{S}$ +=Euclidean distance d($\boldsymbol{x}_z$, $\boldsymbol{x}_r$) (equation 1)
12	**END FOR**
13	$Average_{j,i} = \boldsymbol{S} / \min(\{D^{(j)}\}, \{D^{(i)}\})$
14	**END FOR**
15	**END FOR**
16	FOR each class c , (c=1,…., η)
17	$y_c^{closest} = min\ [\ Average_{j,i}]$, (find minimum $Average_{j,i}$) where
18	(c=j or c=i).

arriving data. Before explaining our algorithm we should present some important equations.

Equation 2 explains how threshold value alpha (α) is computed for each class label. Alpha (α) parameter indicates the number of closest instances to a given instance example. For the ith class c_i with center v_i, alpha (α) is computed by the following equation:

$$\propto_i = \frac{\max_{z=1}^{\{c_i\}}\left[d\left(v_i, x_z\right)\right] - \min_{z=1}^{\{c_i\}}\left[d\left(v_i, x_z\right)\right]}{2} \quad (2)$$

where $\{c_i\}$ is the number of instances belong to c_i. i=1,……,m; m is the number of classes:

$$\max_{z=1}^{\{c_i\}}[\, d\left(v_i,\ x_z\right)]$$

is the maximum distance between v_i and any instance belong to c_i:

$$min_{z=1}^{\{c_i\}}[\, d\left(v_i,\ x_z\right)]$$

is the minimum distance between v_i and any instance belong to c_i.

Note: Class center is computed as follows:

$$v_i = \frac{\sum_{z=1}^{\{c_i\}} x_z}{\{c_i\}} \quad (3)$$

SFDL Algorithm is illustrated at Figure 6. The algorithm consists of three sub-algorithms (1) Instance selection algorithm, (2) Reclassifying the new incoming batch and (3) Set formation.

4.4.1. Instance Selection Algorithm

Instance selection algorithm is presented in Figure 7. The algorithm is used to select the most relevant examples to current concept. The relevancy here is related to how importance older examples are for predicting new instances in term of time similarity and feature space similarity. The algorithm accepts five inputs:

- Historical data D^H which have been used to build the existing classifier L_t;
- New batch D^B which arrived during the period [t to t+N] and labeled using classifier L_t;
- Computed centers υ and alpha α values for each class in the new batch (equations 2 and 3). Instances in the new arrived batch D^B may not be always classified to all the existing classes, so the number of classes at the new batch could be less than the possible classes ($m \leq \eta$);
- The integer w_{recent} represents how many respective recent instances will be selected before time t. In some application where the drift is sudden, the time factor is not important, therefore selecting instances according to its age is ineffective. So the designer of the application can set w_{recent} to zero.

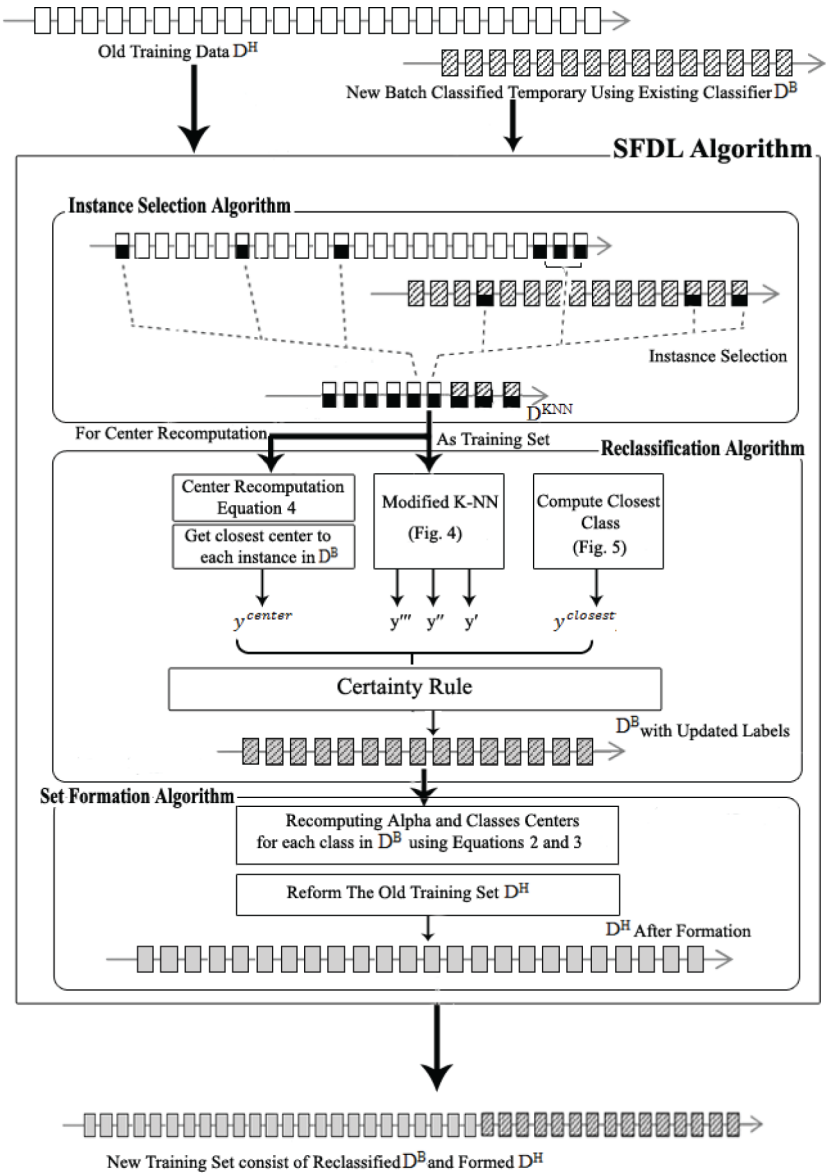
The algorithm output is a set of relevant instances to current concept called DKNN and a set of far instances DFAR (DKNN complement). To select instances according to distance similarity, for each existing class label in the new batch, the algorithm will go through all instances (old and new one) from x1 to xt+N and select instances in which the Euclidean distance between the center of this class and the instance is less than its computed α.

In term of time similarity, relevant instances are selected according to w_{recent} value. The value of w_{recent} depends on the domain at hand, as well as the expectations of the system designer regarding the drift type.

4.4.2. Reclassifying the New Incoming Batch Algorithm

Based on the selected instances in D^{KNN}, the algorithm of reclassifying the new incoming

Figure 6. SFDL algorithm



batch will reclassify the new instances, which were initially classified using the available classifier. The reclassification process is important because the current classifier is assumed to be outdated and useless for classifying the new instances. The algorithm is illustrated in Figure 8.

The main inputs for the reclassification algorithm are D^H, D^B, D^{KNN} and the size of neighborhood k (number of nearest neighbor). The following points summarize the algorithm working flow:

- Applying modified k-NN (Figure 4) with k as a size of neighborhood and D^{KNN} as a training set for the modified k-NN algorithm. Modified k-NN algorithm will

Figure 7. Instance selection algorithm

#	**SFDL – Instance Selection**
1	**Input:**
2	List of historical data D^H =($x_1, \ldots, x_t$) with labels ($y_1, \ldots, y_t$).
3	New unlabeled batch D^B = ($x_{t+1}, \ldots, x_{t+N}$) is labeled using classifier
4	L_t: ($y_{t+1}, \ldots, y_{t+N}$).
5	Computed $\propto$ for each class in D^B (equation 2): ($\propto_1 \ldots\ldots \propto_m$),
6	m is the number of classes in new batch D^B.
7	Computed centers for each class in D^B (equation 3): ($v_1^B \ldots\ldots v_m^B$).
8	Integer window w_{resent} .
9	**Output:**
10	Selected training set D^{KNN} that is very close to current concept and
11	set of far instances D^{FAR}
12	**ALGORITHM**
13	**FOR** i= 1 to m
14	**FOR** j=1 to t+N
15	**IF** $d(v_i^B, x_j) \leq \propto_i$
16	Add x_j to D^{KNN}
17	**ELSE**
18	Add x_j to D^{FAR}
19	**END IF**
20	**END FOR**
21	**END FOR**
22	**IF** $w_{resent} > 0$
23	select most recent w_{recent} instances and add them to D^{KNN}
24	**END IF**

return three different classes as explained in section 4.2, the original k-NN label y′ and two additional labels y″ and y‴.

- The next step is to update the position of the existing class centers. To do that we compute a new center v_x using the formula shown in Box 1.

From this, we can determine that:

$v_\varkappa^B, v_\varkappa^H, v_\varkappa^{KNN}$

are the centers of class $\varkappa$ in D^B, D^H and D^{KNN} datasets respectively, $\varkappa$ = 1….. η, where η is the number of available classes.

In some cases D^B and D^{KNN} do not include all possible classes available in D^H, in this case the associated centers $\left(v_\varkappa^B \text{ or } v_\varkappa^{KNN}\right)$ for miss-

Box 1.

$$v_\varkappa = \frac{v_\varkappa^B + v_\varkappa^H + v_\varkappa^{KNN} + \left(max_{z=1}^{\{c_\varkappa\}}\left[d\left(v_\varkappa^B, x_z\right)\right] - min_{z=1}^{\{c_\varkappa\}}\left[d\left(v_\varkappa^B, x_z\right)\right]\right)}{4} \quad (4)$$

Figure 8. Reclassifying new batch instances algorithm

#	**SFDL – Reclassifying New Batch Instances**
1	**Input:**
2	List of historical data instances $D^H = (x_1, \ldots, x_t)$ with labels $(y_1, \ldots, y_t)$
3	New batch $D^B = (x_{t+1}, \ldots, x_{t+N})$ labeled using classifier L_t $(y_{t+1}, \ldots, y_{t+N})$
4	Selected instances D^{KNN} and far instances D^{FAR} (Fig. 7)
5	Nearest neighbor value : k
6	**Output:**
7	New batch D^B with new class labels
8	**ALGORITHM**
9	Apply modified k-NN (Fig. 4) to reclassify each instance in D^B
10	using D^{KNN} as training set. (three labels y''', y'', y' Returned).
11	Recompute classes center using equation 4.
12	$\varkappa = 1 \ldots\ldots \eta$, η is the number of possible available classes.
13	For each instance in D^B, get the closest center (using Euclidean dis-
14	tance) from all new computed centers in line 11 and assign it to y^{center}
15	For each class in D^B, compute closest class (Fig. 5) ($y^{closest}$)
16	Now there is Five different class label (y^{center} , $y^{closest}$, y''', y'', y') for each
17	instance in D^B where y^{center} The class with closest center to the instance.
18	**FOR** each instance in D^B:
19	**IF** (***Certainty Rule*** return true)
20	reclassify this instance to most frequent class label
21	**ELSE**
22	classify the instance to y^{center}
23	**END IF**
24	**END FOR**

ing classes will be unknown and will be set to zero. Therefore, classes centers which are not included at new batch will not be affected by this formula.

Combining the new computed centers with the previous ones is important to move the centers smoothly and gradually forget the old concept and switch to the new one.

After updating the classes' centers, the algorithm will compute the Euclidean distance between the new centers and every instance in the new batch D^B. Each instance then will have a fourth label y^{center} (in addition to y''', y'' and y') which represent the class with closest center to this instance.

Another class label $y^{closest}$ will be computed using the algorithm illustrated in Figure 5. Unlike y^{center} which represent the closest class to a specific instance, $y^{closest}$ represent the closest class distribution to other class distribution as a whole:

- Now each instance in D^B has five different class labels (y^{center}, $y^{closest}$, y''', y'', y'). The five labels are used to decide if some instance will stay with its current distribution or it must be assigned to other possible closest class. Reclassification of any instance in D^B depends on a heuristic certainty rule. If certainty rule is satisfied, the instance

will be reclassified to the most frequent class label of all the five computed labels. Otherwise, it will be reclassified to y^{center}.

Certainty Rule dictates the following:

1. The instance is not included at D^{FAR};
2. There is no uncertainty in classification (i.e. between the five labels). This means that classification majority must be clear. For example if two of five classes have been classified to label X and two for class O and one for class Z (2:2:1) in this case we say that there is no certainty, because the voting is very close. The same example if three of five classes have been classified to label X and two for class O (3: 2). Cases like (3:1:1) and (4:1) reflects a good majority.

By certainty rule we want to determine those instances that are not classified well by the existing classifier and have a fuzzy membership to their current classes.

We choose y^{center} to be a label for those instances that do not satisfy certainty rule.

4.4.3. Set Formation Algorithm

Reclassifying the new data is not enough. We still need to benefit from the old historical data. Based on the reclassification step, this algorithm reform the old. The algorithm is shown in Figure 9.

The main functions of this algorithm are:

- Recomputing the centers and $\propto$ values for each class in D^B (after reclassification) using equations 2 and 3.
- Reform the old set. For each instance in D^H if the distance to any class center v_i^B is less than $\propto_i$, then this instance will be reclassified to the class with closest center.

The output from SFDL Algorithm is a new training set consists of reclassified new batch

Figure 9. Set formation algorithm

#	**SFDL – Training Formation**
1	**Input:**
2	List of historical data instances $D^H = (x_1, \ldots, x_t)$ with labels $(y_1, \ldots, y_t)$
3	New batch D^B with its new labels (after applying Fig. 8)
4	**Output:**
5	Re-formed old training set of D^H
6	**ALGORITHM**
7	Recompute $\propto$ for each class in D^B (equation 2): $(\propto_1 \ldots\ldots \propto_m)$, m is
8	the number of classes in new batch D^B.
9	Recompute centers for each class in D^B (equation 3.3): $(v_1^B \ldots\ldots v_m^B)$.
10	Recompute $\propto$ for each class in D^B (equation 3): $(\propto_1 \ldots\ldots \propto_m)$, m is
11	**FOR** j=1 to t
12	**FOR** i= 1 to m
13	**IF** $d(x_j, v_i^B) \leq \propto_i$
14	y_j = the class with closest center to x_j .
15	**END IF**
16	**END FOR**
17	**END FOR**

(output of algorithm in Figure 8) and reformed old set (output of algorithm in Figure 9).

The problem of the training set continuous increasing can be solved by using a sampling method which keep the number of instances from exceeding a predefined value.

5. EXPERIMENTAL RESULTS AND EVALUATION

This section discusses the experimental evaluation for the proposed model. Three sub-sections are presented: Section 5.1 presents a brief description for datasets used in our experimentation. Section 5.2 describes the experimental setup and procedure. Finally, Section 5.3 discusses the experimental results.

5.1. Datasets

For the purposes of research related to concept drift learning there is no standard concept drift benchmark dataset. In-stead there are popularly datasets that were used by most of the existing researches. Unfortunately most of the existing real word datasets are not suitable for evaluating drift learning because there is a little concept drift in them. So re-searchers turn to introduce artificial drift in real datasets or create synthetic (fabricated) datasets with artificial drift.

In our experiments we used six datasets, all of which are publicly available. The datasets are chosen from various domains that might have different drift types with different speed of change. They include no missing or noise. Table 1 illustrates the characteristics of each set: number of instances, attributes, classes and the type of dataset. A short description of each data set is given below.

For STAGGER dataset the instance space is defined by three attributes, size = {small, medium, large}, color = {red, green, blue}, and shape = {square, circle, triangular}. The target concepts have changed as follows: (1) size = small and color = red, (2)color = green or shape = circular and (3)size = medium or large. 120 training instances have been generated randomly and the concept has been changed every 40 instances. There are two sources of drift in STAGGER dataset: (1) the changing in posterior probabilities and (2) the changing in class balance (Narasimhamurthy & Kuncheva, 2007).

In SEA data, each instance is described by three features, $x=[x1, x2, x3]^T$, where values of x are uniformly randomly generated between [0 - 10]. Only the first two features are relevant. An instance belongs to class 1 if $x1 + x2 \leq \theta$ and belongs to class 2 otherwise, where θ is a threshold value changed to create a concept. There are four concepts θ = 7, 8, 8.5, 9. We generate 200 instances for each concept (100 instances for each class label). There is no label noise was added and the two classes are perfectly separated (Street & Kim, 2001).

Table 1. Characteristics of the used datasets

Name		Ins.	Dim.	Classes	Type of data
STAGGER		120	9	2	Artificial
SEA		800	3	2	Artificial
Elec		2973	6	2	Real
Chess		533	6	3	Real
Credit		1000	23	2	Real
Usenet	Usenet1	1500	99	2	Real
	Usenet2	1500			

Electricity dataset includes 2973 instances collected along a period of 3 months from May 11 to July 11, 1997 from the Australian New South Wales Electricity Market. Class Label has two values 'up' or 'down' indicating the change of the price. In our experimentation each month represents one concept (Harries, 1999).

Chess data includes a gaming records for one player over a period from 2007 December to 2010 March. A player is developing his skills over time and he can engage into different types of competitions (personal, tournament or champtionship). A player rating and the selected game type are crucial to the system to select an opponent. The task is to predict if the player will win or lose based on the setting. There is a natural problem of delayed labeling, the winner is known only after the game is finished.

Credit dataset classifies customers as having good or bad credit risks. Following Žliobaitė (2010), a gradual concept change was introduced artificially by sorting the data using feature 'age' and eliminate this feature from the dataset. Delayed labeling is relevant for this task, since the true label (whether a person fails to repay the credit) is known after some time. However, the decision makers needs a real time indication of changes in risk (UCI machine learning repository: Data sets, n.d.).

Usenet dataset includes two sets usenet1 and usenet2. The difference between the two sets is illustrated in Figure 10. We have five batches each contains 300 instances as indicated by the first row in Figure 10. The figure also shows the change in news interest among the batches and which newsgroups articles are considered interesting (+) or uninteresting (-) in each time period[1]. This dataset was used to build news recommender systems, document categorization and spam filtering applications (Katakis, Tsouakas, Banos, Bassiliades, & Vlahavas, 2009).

5.2. Experiments Setup

The experiments took place on a machine equipped with an Intel Pentium Core 2 Duo T8300 @ 2.40 GHz processor and 2.00 GB of RAM. To implement our algorithm we used Java programming language. The goal of experiments is to observe the system performance as the target concepts are changing from time to time. Our approach is expected to enhance the classification accuracy which might drop down over time if we use an ordinary classier (a classifier that does not consider concept drift in its approach). To achieve our goal we follow this experiment procedure:

1. We start by dividing the dataset into smaller sub sets, each is called a "batch". We benefit from previous researches in the way they partition the dataset and insure that every "batch" represent a change (Katakis, Tsoumakas, & Vlahavas, 2008; Žliobaitė, 2010);

Figure 10. Usenet1 and Usenet2 datasets

	0-300	301-600	600-900	900-1200	1200-1500
Usenet 1					
medicine	+	-	+	-	+
space	-	+	-	+	-
baseball	-	+	-	+	-
Usenet 2					
medicine	+	-	-	-	+
space	-	+	-	+	-
baseball	-	-	+	-	-

2. We use one batch as a training set for the initial learner. In datasets where instances are ordered according to time, we use oldest batch to be the initial training set. Otherwise we pick a random batch as initial training set, because the drift type in such sets is mostly sudden. Table 2 shows dataset partitioning details. The table presents the following information: type of drift represented, number of instances included in the initial dataset with class balance distribution (in percentage %), number of batches (#B) and if the dataset is time ordered (Y) or not (N);
3. As in traditional machine learning process, we build the initial learner using initial training set. We use 10-cross validation with stratified sampling in order to estimate the performance of the classifier and ensure we get the best model in current time according to its classification accuracy. The accuracy of the model is calculated using the following equation:

$$Accuracy = \frac{number\ of\ instances\ correctly\ classified}{n} \times 100 \quad (5)$$

where n is the number of instances:

4. Next, we pass the first batch, classify it using current learner (ordinary classifier), apply SFDL training set formation algorithm and retrain the model using formed data. As we mention before SFDL algorithm needs two parameters: (1) number of neighborhood k (space similarity) and (2) number of the most recent instances w_{recent} (time similarity). The choice of these two parameters values is directly related to the observed change types and the future expectations as well as designer knowledge of domain. Parameters setting is fixed for one application run;
5. We measure the accuracy at two points after passing the batch: (1) after its been classified by Reclassifying the New Incoming Batch (Figure 8). (2) after training set formation and model retraining;
6. We pass the next batch, classify it using the most recent trained classifier after set formation and so on. The procedure explained previously in section 4.1. After set formation, we compare our results with ordinary classifier results.

5.3. Experimental Results and Discussion

This section discusses the results of numerous experiments that have been conducted.

5.3.1. Sudden Drift Experiments (STAGGER and SEA datasets)

Table 3 illustrates experimental results for both STAGGER and SEA datasets. For STAGGER dataset, the best model for classifying the first concept was Naïve Bayes with training accuracy = 100%. We use the same model to predict batch1 and batch2. The accuracy of

Table 2. Dataset partitioning details

Name	Drift Type	Training Ins. (%)	#B	TO
Stagger	Sudden	40 (58: 42)	2	N
SEA	Sudden	200 (50:50)	3	N
Elec	Gradual	1006 (64: 36)	2	Y
Chess	Incremental	233(42:54: 4)	3	Y
Credit	Gradual	400 (63: 37)	6	Y

Table 3. Results of STAGGER and SEA datasets. Accuracy after the batch is underlined. Accuracy after training set formation and model retraining is in bold.

				B1	B2	
STAGGER	Naïve Bayes Acc = 100% k=2 w_{recent} = 0	Ordinary		57.50%	53.66%	
		Arrival of	B1	65.00% **90.00%**	46.34%	
			B2	-	43.90% **65.85%**	
				B1	B2	B3
SEA	Decision Tree Acc = 100% k=10 w_{recent} = 0	Ordinary		59.00%	50.00%	50.00%
		Arrival of	B1	50.00% **86.00%**	53.00%	-
			B2	-	50.00% **89.50%**	73.50%
			B3	-	-	86.00% **89.00%**

classification dropped to 57.50% and 53.66% for batch1 and batch2 respectively. This results confirm the existence of drift where the current Naïve Bayes model could not classify the other concepts correctly.

Two accuracy observations have been recorded after passing batch1. The underlined observation with value = 65% represents the accuracy after reclassifying the batch using Figure 8 algorithm. The bold observation with value = 90% represents the accuracy after reformation and model retraining. SFDL algorithm was applied with $k = 2$ and $w_{recent} = 0$. The size of batches is very small (40 instances) for this reason we chose a small number of neighborhood k. Including most recent instances from historical data (w_{recent} > zero) is meaningless because we are dealing with sudden drift where source of drift is not related to time ordering and the previous concept is not much trusted to classify the current batch. SFDL algorithm enhance the accuracy by 32.5% (from 57.50% to 90.00%).

After the arrival of batch2 we classify it's instances using the most recent Naïve Bayes model (after retraining). Note that the accuracy decreased to 46.34%. This happened because the last retraining have been done according to the drifting of batch1 which is different from batch2. Additionally, the problem of class imbalance at batch1 makes the updated model biased toward dominate label "2" and decreases accuracy to 43.90%. With perfect parameters setting (of k and w_{recent}) and the role of certainty rule, accuracy increased to 65.85% after applying SFDL algorithm.

Changes in user interests over time are the main cause of concept drift in usenet dataset. It is obvious from Figure 10 that usenet datasets represent recurrence drift type. In fact this dataset is much more complicated in reality due to unpredictable user interests.

For SEA dataset the most accurate classifier for classifying the first concept was Decision Tree with accuracy = 100%. The same model was used to classify the three incoming concepts. The accuracy of classification was 59.00%, 50.00%, 50.00% for batch1, batch2 and batch3 respectively. Classification accuracy of incoming concepts decreased compare to initial concept classification accuracy. Table 3 presents the accuracy after batch reclassification and model retraining after passing the three batches. SFDL algorithm was applied with k =

10 and $w_{recent} = 0$ because we are dealing with sudden and medium-sized dataset. After model retraining, SFDL algorithm increases the accuracy of classification with at least 27%. Unlike the first two batches, reclassification accuracy of batch3 is higher than classification accuracy by initial model. The reason is that the concept sequencing ($\theta = 7, 8, 8.5, 9$) allows the classifier to gain more knowledge after passing the two previous batches.

Figure 11 presents the curves of accuracy over time using SFDL algorithm and ordinary classifier for STAGGER and SEA datasets. It is notable that SFDL algorithm achieves better performance than ordinary classifiers for both sets.

5.3.2. Gradual Drift Experiments (Electricity and Credit Datasets)

Electricity and credit datasets are real world datasets with gradual drift. For both datasets we use Multilayer Perceptron Neural Network (MLP-NN) as a training model. Because we are dealing with time-related gradual drift, time similarity was taken into consideration ($w_{recent} > 0$). In other words, to predict electricity price or credit card approval, most recent examples are more reliable to be used to classify new incoming instances than old historical data.

Table 4 illustrates experimental results for both Electricity and Credit datasets. The average error of classifying the new batches by ordinary classifier for gradual drift experiments is mostly less than it for sudden drift experiments. The reason is that the two datasets used here and most real word datasets include little concept drift.

Figure 12 presents the curves of accuracy over time using SFDL algorithm and ordinary classifier for Electricity and credit datasets. The figure shows that our algorithm achieves higher classification accuracy in comparison to ordinary classifier for both datasets.

5.3.3. Incremental Drift Experiments (Chess Dataset)

Incremental drift is a sequence of small sudden drifts. For this reason it is very difficult to predict and learn. The main difference between incremental and sudden drift is that incremental drift is related to time where sudden drift is not.

The results of chess experiments are presented in Table 5. Chess dataset includes a real incremental drift. The best model for predicting the first concept was Rule-Based Classifier with accuracy = 92.06%. We choose a very small neighborhood k and w_{recent} values where it is suitable to the nature of data and change speed. From the table it is clear that our approach have

Figure 11. Accuracy over time for SFDL algorithm and ordinary classifier, (a) STAGGER dataset, (b) SEA dataset

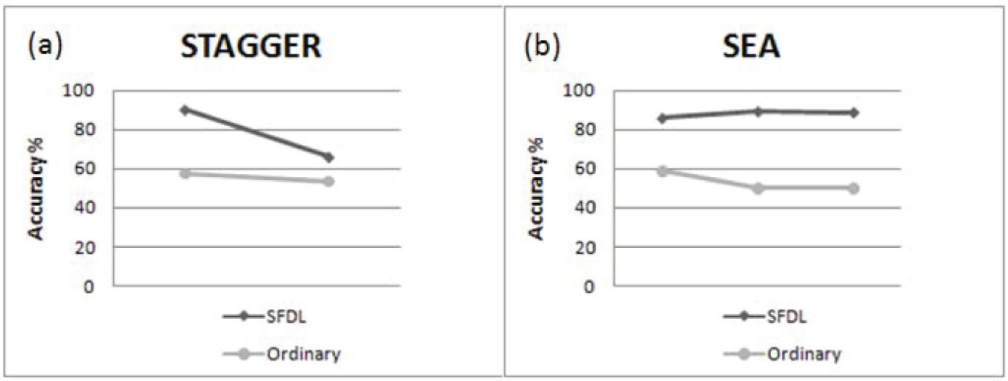
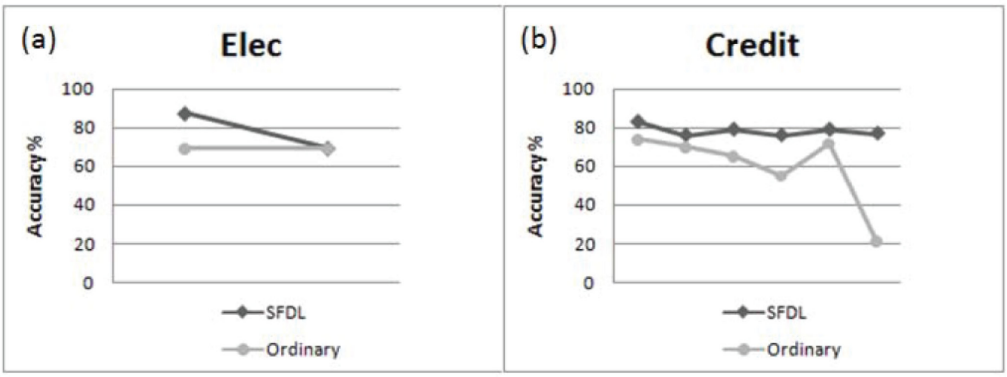


Table 4. Results of electricity and credit databases. Accuracy after the batch reclassification is underlined. Accuracy after training set formation and model retraining is in bold.

				B1	B2	B3	B4	B5	B6
Elec	MLP-NN Acc=89.5% k=60 w_{recent} =200	Ordinary		69.44%	69.26%				
		Arrival of	B1	72.50% **87.43%**	66.66%				
			B2	-	67.17% **69.27%**				
Credit	MLP-NN Acc =100% k=13 w_{recent} =100	Ordinary		74%	70%	65%	55%	72%	21%
		Arrival of	B1	64% **83%**	72%	-	-	-	-
			B2	-	64% **76%**	69%	-	-	-
			B3	-	-	69% **79%**	61%	-	-
			B4	-	-	-	66% **76%**	72%	-
			B5	-	-	-	-	73% **79%**	73%
			B6	-	-	-	-	-	75% **77%**

Figure 12. Accuracy over time for SFDL algorithm and ordinary classifier, (a) Electricity dataset, (b) Credit dataset



better predictive performance than the classical ordinary classifier.

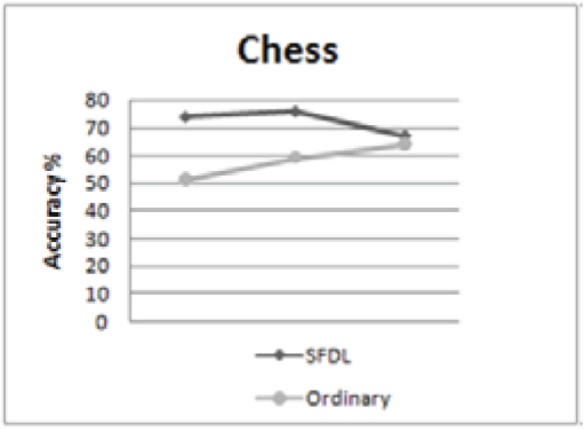
By the arrival of first and second batch, SFDL enhance the accuracy by at least 17% but not more than 3% for the last batch. We think the reason is the extensive sudden drifts during this period. Also in this period the user turns to play personal competitions (70% of total instances in batch3). This may add another hidden causes of drift related to player-opponent relationship.

Figure 13 presents the curves of accuracy for SFDL algorithm and ordinary classifier for chess dataset. SFDL shows superior accuracy over ordinary classifier.

Table 5. Results of chess dataset. Accuracy after the batch reclassification is underlined. Accuracy after training set formation and model retraining is in bold.

				B1	B2	B3
Chess	Rule-based Classifier Acc=92.06% k=13 w_{recent} = 20	Ordinary		51.00%	59.00%	64.00%
		Arrival of	B1	75.00% **74.00%**	74.00%	-
			B2	-	69.00% **76.00%**	67.00%
			B3	-	-	67.00% **67.00%**

Figure 13. Accuracy over time for SFDL algorithm and ordinary classifier for chess dataset



5.3.4. Reoccurring Concept Experiments (Usenet Datasets)

Changes in user interests over time are the main cause of concept drift in usenet dataset. It is obvious from Figure 10 that usenet datasets represent recurrence drift type. In fact this dataset is much more complicated in reality due to unpredictable user interests.

We use the same portioning method as Katakis, Tsoumakas, and Vlahavas (2008) as illustrated in Figure 10. The first user interest (medicine articles) was used to build initial learner and other parts of interest to represent incoming batches. It is to be mentioned that batches with the same interests are not identical.

Table 6 illustrates experimental results for usenet datasets. The best model for predicting first concept for both usenet datasets was MLP-NN with accuracy = 95.83% and 93.33% for usenet1 and usenet2 respectively. We benefit from Katakis et al. (2008) experiments to choose the best w_{recent} value while extensive experiments have been done to choose k neighborhood value.

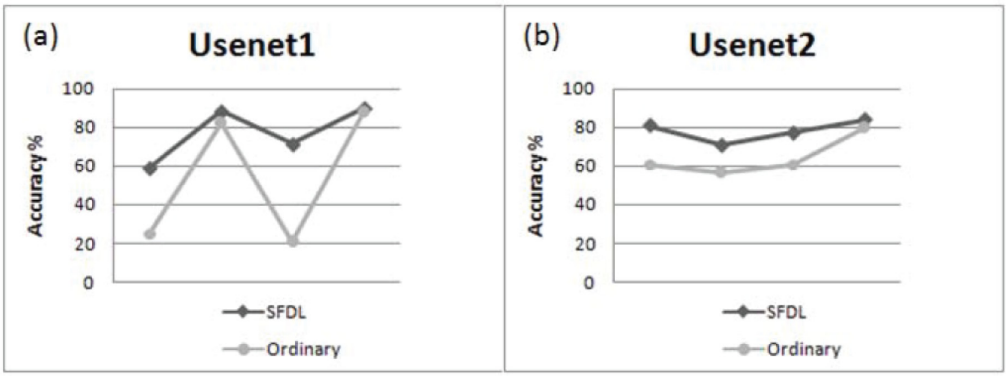
The results proves the SFDL algorithm ability in switching between concepts as user interests change. Figure 14 also shows the advantages of SFDL algorithm over ordinary classifier.

Table 7 presents a comparative study between SFDL algorithm and four other stream classification methods for usenet datasets:

Table 6. Results of Usenet datasets. Accuracy after the batch reclassification is underlined. Accuracy after training set formation and model retraining is in bold.

				B1	B2	B3	B4
Usenet1	MLP-NN Acc =95.83% k=30 w_{recent} =100		Ordinary	24.33%	82.33%	20.67%	88.33%
		Arrival of	B1	52.66% **59.00%**	54.00%	-	-
			B2	-	69.33% **88.33%**	60.00%	-
			B3	-	-	54.33% **71.33%**	55.33%
			B4	-	-	-	66.33% **90.00%**
				B1	B2	B3	B4
Usenet2	MLP-NN Acc =93.33% k= 30 w_{recent} = 100		Ordinary	60.67%	57.00%	60.67%	80.00%
		Arrival of	B1	49.00% **81.00%**	44.67%	-	-
			B2	-	42.00% **70.67%**	72.00%	-
			B3	-	-	64.00% **77.00%**	75.00%
			B4	-	-	-	72.00% **84.00%**

Figure 14. Accuracy over time for SFDL algorithm and ordinary classifier, (a) Usenet1 dataset, (b) Usenet2 dataset



simple incremental method, time window method with different window size, weighted examples and conceptual clustering and prediction framework (CCP). These methods have considered the concept drift problem in their approaches.

It is notable that average classification accuracies observations for usenet2 dataset are higher than of that for usenet1. Usenet1 includes more complicated drift where the same batch includes another drift and the user switch between two different interests. It is clear

Table 7. Average accuracy of the four methods in the usenet datasets

Method	Usenet1	Usenet2
SFDL Algorithm	81.00%	81.20%
Simple Incremental [3]	59.00%	73.00%
TimeWindow (N=100) [13]	56.00%	60.00%
TimeWindow (N=150) [13]	59.00%	62.00%
TimeWindow (N=300) [13]	58.00%	70.00%
Weighted Examples [6]	67.00%	75.00%
CCP (batch size = 50) [3]	81.00%	80.00%

that SFDL approach outperforms all the other methods and Time Window approach (N=100) act the worst.

6. CONCLUSION

In this paper, we addressed the problem of supervised learning over time when the data is changing and label of new instances is delayed. We introduce an adaptive training set formation algorithm called SFDL, which is based on selective training set formation. Our proposed algorithm is considered as the first systematic training set formation approach that take into account delayed labeling problem.

SFDL algorithm includes three sub algorithms: instance selection algorithm that is used to select the most relevant examples to current concept in terms of time similarity and space similarity, reclassification algorithm to reclassify the new instances which were initially classified using the available classifier and the third algorithm is training set formation algorithm which work to reform the old set according to the changes made on reclassification step.

We tested our approach using synthetic and real datasets. The datasets were chosen from various domains which might have different drift types (sudden, gradual, incremental reoccurrences) and different speed of change. Experimental evaluation confirms improvement in classification accuracy as compared to ordinary classifier for all drift types. Our approach is able to increase the classifications accuracy with 20% in average and 56% in the best cases and it has not been worse than the ordinary classifiers in any case.

Finally, we conducted a comparative study with another four methods to identify recurrence drift and predict changes in user interest over time. The results show the superiority of our solution over other methods in handling recurrence drift.

Future research will be directed in the following direction: For input setting parameters like number of neighborhood k and number of most recent instances w_{recent}, these parameters have been determined by application designer. It is better to automatically determine these parameters to preserve self-adaption. Exploring some ideas to enhance the proposed strategy and improve the results accuracy. A very high classification accuracy can be provided if we build a customized version to deal with each drift individually. The algorithm should also be extended so it can add or remove classes. This is important for domains where some classes may disappear by time and must be removed or vice versa. It is also very useful to provide the algorithm with a dynamic feature space formation ability.

Finally we can say that future research on adaptivity to concept drift has to be more specializing in application groups.

REFERENCES

Brzezinski, D. (2010). *Mining data streams with concept drift.* Master thesis, Poznan University of Technology.

Delany, S. J., Cunningham, P., Tsymbal, A., & Coyle, L. (2005). A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, 18.

Harries, M. (1999). *Splice-2 comparative evaluation: Electricity pricing*. Technical report, The University of South Wales. Retrieved October 8, 2011, from http://www.liaad.up.pt/~jgama/ales/ales_5.html

Katakis, I., Tsoumakas, G., Banos, E., Bassiliades, N., & Vlahavas, I. (2009). An adaptive personalized news dissemination system. *Journal of Intelligent Information Systems*. doi:10.1007/s10844-008-0053-8.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2008). An ensemble of classifiers for coping with recurring contexts in data streams. In *Proceeding of 18th European Conference on Artificial Intelligence,* Patras, Greece.

Kelly, M., Hand, D., & Adams, N. (1999, August 15-18). The impact of changing populations on classifier performance. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 367-371).

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, *8*(3), 281–300.

Ludmila, I., Kuncheva, J., & Salvador, S. (2008). Nearest neighbour classifiers for streaming data with delayed labeling. In *Proceedings of the Eighth IEEE International Conference on Data Mining ICDM* (pp.869-874).

Narasimhamurthy, A., & Kuncheva, L. (2007). A framework for generating data to simulate changing environments. In *Proceeding of the 25th IASTED int. Multi-Conference, Artificial Intelligence and Applications (AIAP''07)* (pp. 384–389). ACTA Press.

Nishida, K. (2008). *Learning and detecting concept drift*. PhD thesis, Hokkaido University, Japan, 2008.

Nishida, K., & Yamauchi, K. (2009). Learning, detecting, understanding, and predicting concept changes. *International Joint Conference on Neural Networks (IJCNN)* (pp. 2280-2287).

Street, N., & Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Retrieved October 8, 2011, from http://www.liaad.up.pt/~kdus/kdus_5.html

Tsymbal, A. (2004). *The problem of concept drift: Definitions and related work. Technical Report, Department of Computer Science*. Dublin, Ireland: Trinity College.

UCI machine learning repository: Data sets. (n.d.). Retrieve October 8, 2011, from http://archive.ics.uci.edu/ml/datasets.html

Wang, H., Fan, W., Yu, P., & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)* (pp.226-235). New York, NY: ACM.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*(1), 69–101. doi:10.1007/BF00116900.

Yang, Q., & Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, *5*(4), 597–604. doi:10.1142/S0219622006002258.

Žliobaitė, I. (2010). *Adaptive training set formation*. PhD thesis, Vilnius University.

ENDNOTES

1 http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html