Computer Science • 14 (2) 2013 http://dx.doi.org/10.7494/csci.2013.14.2.191

Che Ngufor
Janusz Wojtusiak

UNSUPERVISED LABELING OF DATA FOR SUPERVISED LEARNING AND ITS APPLICATION TO MEDICAL CLAIMS PREDICTION

Abstract *The task identifying changes and irregularities in medical insurance claim payments is a difficult process of which the traditional practice involves querying historical claims databases and flagging potential claims as normal or abnormal. Because what is considered as normal payment is usually unknown and may change over time, abnormal payments often pass undetected; only to be discovered when the payment period has passed.*

This paper presents the problem of on-line unsupervised learning from data streams when the distribution that generates the data changes or drifts over time. Automated algorithms for detecting drifting concepts in a probability distribution of the data are presented. The idea behind the presented drift detection methods is to transform the distribution of the data within a sliding window into a more convenient distribution. Then, a test statistics p-value at a given significance level can be used to infer the drift rate, adjust the window size and decide on the status of the drift. The detected concepts drifts are used to label the data, for subsequent learning of classification models by a supervised learner. The algorithms were tested on several synthetic and real medical claims data sets.

Keywords unsupervised learning, concept drift, medical claims

1. Introduction

Labeled examples are the most common form of data for concept learning. Given a set of such examples, concept learning algorithms are able to create classification models that can be used in decision support systems. However, most databases do not always have labeled examples. Assigning labels to examples in data is a labor-intensive, expensive and error-prone task; especially when it is performed manually and access to domain experts are limited. This difficulty increases with the amount and complexity of data to be labeled.

This problem can be further complicated when the assignment of labels changes over time. This situation, known as *concept drift*, occurs when the same example receives different labels at two different times. This situation is not uncommon, for example what is considered to be a normal payment for a provided service at a given time may not be normal after a few years when prices have risen.

The problem considered in this paper is how to automatically label data that can be later used for concept learning. Specifically, it considers labels to be normal, indicating that some values are within specified ranges (that may change overtime), and abnormal, indicating that the values are outside the specified ranges. The presented unsupervised approach is capable of handling univariate or multivariate data streams with concept drift.

The presented work is part of a larger project, briefly outlined in Section 2, whose goal is to create a decision support system capable of predicting payments for medical claims. The system is designed to predict if a specific medical claim will be paid a normal or abnormal amount, and what the abnormal amount is. The core of the system consists of a classifier constructed using a concept learning method applied to a training dataset with historical claims. The dataset, however, did not include normal/abnormal labels, only amounts that were received for specific claims. The method presented in this paper was used to assign the labels.

2. Medical claims payment prediction

Healthcare providers seek reimbursement for the services they provide. Claims are prepared and submitted to payers. While in the majority of cases the claims receive accurate payment, sometimes claims are not paid in full or not paid at all. The latter may be due to improper claim preparation by the provider, improper claim processing by the payer, miss-interpretation of a contract agreement, or deliberate fraudulent actions. An important task is to predict when a claim will be paid or not prior to submission. Moreover, it is crucial to present management with patterns describing situations in which the inappropriate processing occurs, in order to improve claim preparation processes and therefore reduce denials. Thus, the benefits of the presented method are two-fold (1) denied or underpaid claims can be predicted in advance allowing providers to modify them to increase chances of receiving correct

payment, and (2) the process of providing and documenting services, as well as claims preparation can be changed in order to reduce errors.

In order to achieve the above tasks, a machine learning-based decision support system has been designed. The system works in two main phases: model construction from historical data, and payment prediction and model update phase. These two phases are depicted in Figure 1. The model construction phase consists of the following steps:

1. Historical data retrieval from billing systems.
2. Historical data labeling (the main focus of this paper).
3. Classification model construction.
4. Regression model construction.
5. Model testing.

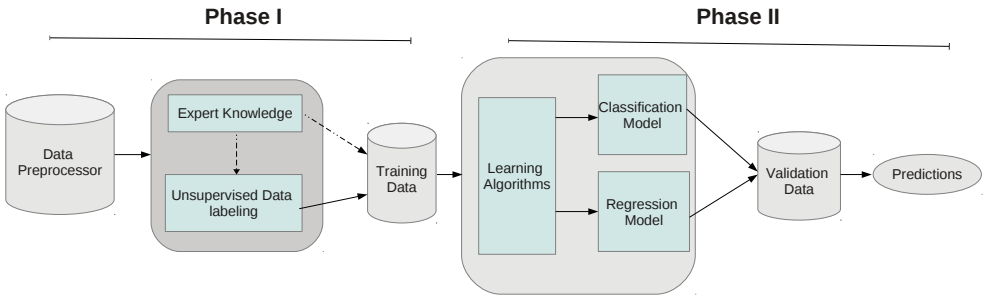


Figure 1. Model phases.

The work presented in this paper is part of the Phase I in Figure 1. Unsupervised learning techniques coupled with prior knowledge from domain experts are used to generate labels for data characterized by drifting distributions. The generated training data is then used in the second, supervised, phase for the learning of classification and regression methods. Because the distribution changes over time and there is little or no knowledge about the time or change points, *batch* learning algorithms are incapable of learning the data since they require all examples to be available in one batch before learning can commence.

The focus of Phase I, therefore, is to design efficient on-line unsupervised learning algorithms for the detection of changes in the distribution of data streams that may be contaminated with noise and ultimately labeling examples. Because of the focus of this paper, other details of the larger project concerning the complete process of predicting medical claim payments are out of scope. The broader project has been previously reported by Wojtusiak et al. [16, 17]

3. Related work

The standard approach for detecting changes in medical insurance claim payments consist of expert knowledge if available and ad-hoc querying of claims database and manually flagging potential claims as abnormal. The non-availability of automated methods for detecting changes is one of the major challenges for most healthcare providers.

In the statistical literature, a considerable amount of work has been done on the detection problem i.e the so-called *change-point problem* and various approaches have been proposed for its solution. These approaches usually involve detecting changes in a parameter such as the mean or variance under various distributional assumptions. One of the earliest and most popular works on the change-point problem included those of Page [9] and Hinkley [4]. Page computes a Cumulative Sum statistic (CUSUM) for a data stream to test if a change in the parameter has occurred. The cumulative sums S_n for the first n'th observations are recorded and an action is taken to rectify a possible change in the parameter when $S_n - \min_{0 \leq i < n} S_i \geq h$ i.e the sample path rises a height h above its previous minimum value. This very simple but effective algorithm has one obvious drawback: some prior knowledge will be required to select an optimal value for the height h. In Hinkley's method, estimates and inference about the change-point is obtained through a likelihood ratio test statistic. Other methods that have equally been investigated include, Bayesian techniques [12], wavelet footprints [11], nonparametric regression [7], least square regression trees [1], and Fisher information methods [5]. A review of these and other popular techniques can be found in [10, 2, 8].

These methods are designed to study swift changes in the underlying distribution. They are unable to cope with gradual or more complex changes. Further, they are non-learning methods; a learning algorithm is required to adapt pre-learned changes (or concepts) to environmental changes. Environmental factors may result in swift, gradual or complex changes such that samples previously representing a given concept can later represent a different concept. An algorithm capable of recognizing and adapting to such changes is therefore required. One learning approach capable of dealing with gradual concept changes is described in [3]. The approach consists of using an unsupervised incremental learning algorithm to learn the gradual concept drift initially learned by a supervised learner. This learning approach is however limited to gradual concept drifts and requires a supervised learner to initialize the algorithm.

This paper presents a statistical and machine learning technique to accurately and automatically detect changes in unlabeled data streams with concept drift using a novel unsupervised approach that works well for uni-variate and multivariate data. The learned knowledge is then used to label the data stream for subsequent classification algorithms.

4. Problem statement

The aim of data labeling is to assign labels or tags to data points on the basis of values of its features i.e data points with similar patterns are assigned the same labels. Thus the goal of a machine learning data labeling algorithm is to build a representations of the input patterns that can be used for decision making, predicting future inputs, communicating the inputs to another machine, etc.

A typical unsupervised machine learning algorithm is given a sequence of unlabeled inputs $x_1, x_2, \ldots, x_t, \ldots$ where $x_t \in \mathcal{X}$ is the input pattern at time t and $\mathcal{X}$ is the feature space. An unsupervised classifier can be defined as a mapping

$$\psi \colon x_t \mapsto \theta \tag{1}$$

assigning x_t a unique label θ drawn from a finite set of say K mutually exclusive labels $\Theta = \{\theta_1, \theta_2, \ldots \theta_K\}$ based on some similarity measure. For data known to be of two possible classes, θ can be say "normal" or "abnormal". The unsupervised classifier receives no loss or reward in assigning x_t a specific label. Note that this is different from a supervised classifier in which case ψ is a function that receives in addition to the input, a corresponding output pattern and a loss or reward may be incurred for an incorrect or correct classification.

In changing environments, a different problem description has to be dealt with. The description of the labels or target concepts changes over time. This type of drifting distribution with the presence of a changing target concept is known as *concept drift*. The underlying distribution which generates the data may change gradually over time or suddenly at some unknown point. Hence, the unsupervised classifier ψ for learning concept drift depends on the order in which the data stream arrives. Given a stream of input data points $x_1, x_2, \ldots, x_t, \ldots$ with a possible concept drift at some unknown point, the unsupervised classifier must be able to learn stable concepts as in conventional concept learning and adapt when it meets a new concept. Thus, in general a label is assigned to a new input $x_{t+1} \in \mathcal{X}_{t+1}$ as:

$$\psi_t \colon x_{t+1} \mapsto \begin{cases} \theta_{old} & \text{if } x_{t+1} \equiv x_{old} \colon \text{ a previously learned concept} \\ \theta_{new} & \text{a new concept} \end{cases} \tag{2}$$

where $x_{t+1} \equiv x_{old}$ means x_{t+1} has similar characteristics to some previously learned points x_{old} labeled as θ_{old}. In other words, when a new data point x_t becomes available, and if it represents a new concept, then a new classifier ψ_t is generated by updating the previously learned classifier ψ_{t-1}. Otherwise the old classifier is maintained. Only the parameters of the newly learned distribution need to be stored. The major problem then is how to design the classifier ψ_t to track and analyze concept changes in the streaming data.

One basic approach to track concept changes in streaming data is presented in Algorithm 1. The algorithm starts by initializing the first concept with the first input data point. Each incoming data point is checked if it is in the proximity of the previous

observed data. All samples in proximity of the current learned concept are considered as belonging to that concept. Samples not in proximity to the current concept are checked for proximity to a possible new concept by evaluating the proximity to the mean or median of future data points within a window of size k. Samples not in proximity to any concept are considered as noise.

One obvious drawback of this algorithm (henceforth referred to as **Basic**) is that it is very sensitive to noise. It is not hard to see that **Basic** will perform well on noiseless data with step-like or sudden drifts, but worse in the presence of noise. The proximity measure ε will have to be adjusted for some data points and on each new data set to be able to capture changing concepts. Setting an optimal value for this parameter becomes a difficult task. As with **Basic**, many concept drift algorithms can handle sudden changes well but perform poorly with gradual changes. Most real streaming data are characterized by sudden and gradual changes, this therefore highlights the importance of examining both types of changes. Another drawback of **Basic** that will be discoursed in the next section, is the use of a single window.

Algorithm 1: Basic Unsupervised Concept Drift Learning: **Basic**.

Input : Training data set $\mathcal{X}$, window width k, distance threshold between samples ε

$t \leftarrow 1$
$P_1 \leftarrow \mathcal{X}[t]$ `// first data point initialized to first concept`
repeat
  $P_2 \leftarrow \mathcal{X}[t]$; `// new sample to learn new concepts`
  if $\left\|P_1 - P_2\right\| < \varepsilon$ **then**
    Add P_2 to concept represented by P_1 ;
  else
    Compute M the mean/median of the next k feature points;
    if $\left\|M - P_2\right\| < \varepsilon$ **then**
      P_2 marks the start of a new concept ;
      $P_1 \leftarrow P_2$; `// update` P_1 `towards` P_2
    else
      P_2 represent noise; `// replace with` M
    end
  end
  $t \leftarrow t + 1$
until $t >$ *samplesize*;

5. Proposed approach

Most proposed strategies to handle concept drift are based on the usage of one or more sliding windows [6, 14]. In this approach, a window is maintained that keeps only the most recent data points and older data points are dropped according to how relevant

they are in detecting new concepts. The use of a single window of fixed length for tracking concept drift is very common, however, as addressed in [6] this approach has three major drawbacks. First, it is difficult to determine the optimal window size. No single window size can deal with all types of concept drifts; larger windows perform better on data streams with a slow drift while smaller windows are better suited for rapid drift. The standard solution to this problem is to make the window dynamic, so that the size can be adjusted manually or automatically to better track the concept drift. When the drift *rate* is rapid, the size of the window can be reduced by removing some points from the window and when the rate is slow the size is increased by adding more points. Determining the concept drift rate is therefore important in adjusting the window size. The drift rate simply represents the probability that two successive data points disagree on the concept they represent i.e if x_t represents the concept θ_1 and x_{t+1} represents the concept θ_2, then the drift rate is the probability $\Pr(\theta_1 \neq \theta_2)$.

The second drawback of using a single window (dynamic or fixed) is that a single window cannot optimally handle a continuous change i.e. if the change occurs gradually over a certain time frame.

Finally, concept drift algorithms based on a single window can also suffer from the inability to learn multiple concepts simultaneously. Since a drift can occur at any time frame within the window, multiple drift points can occur within the same window and not all of them can be optimally learned with a single window regardless if they occur gradually or suddenly.

The proposed method in this study uses three different sliding windows. Data points in the first two windows are used to update known concepts or build new concepts. The third window serves as a control window i.e. it is used to reconfirm potential drifts in the concept, thus serving as a control for noise in the data. The algorithm uses a test statistic whose p-value at a specified significant level serves as an indicator for potential concept drifts. The p-value can also be used to infer the drift rate and adjust the window size. Another advantage of using this statistic is that, it can also be used to track gradual and sudden concept drifts. Precisely, for each new data point, a test is performed to verify the null hypothesis H_0 of no concept change at a given level of significance α. If a drift occurs, the null hypothesis is rejected in favor of the alternative H_a. However, before rejecting H_0, another test is carried out on points in the third window (these are points after the proposed concept drift) to reconfirm the rejection of H_0. A sudden concept change will be reflected by a very small p-value (such as far smaller than 0.05) and hence a very high drift rate, while a gradual change will be seen by a not so small p-value, but small enough to be significant. Thus, the confidence bound for the p-value provides a range for the best guess to the true theoretical p-value of the test.

5.1. Learning concept drift

Assume a stream of unlabeled data points $x_1, x_2, \ldots, x_t, \ldots$ are observed. Given a window width k, the algorithm starts by assigning the first k data points (first window)

to the first concept θ_1. The second window comprises all but one data point from the previous window and one new data point from the incoming stream i.e if at time step t a new data point x_t is observed, then the previous window contains the k data points $x_{t-k}, \ldots, x_{t-1}$ and the current window contains $x_{t-k+1}, \ldots, x_t$. The data points in all windows are transformed into the distributions P_1 and P_2 by taking the mutual distances of all points in the windows. Thus, P_1 and P_2 consist of $\frac{k(k+1)}{2}$ points. This means that, a window containing only $k = 10$ points gives a new distribution of size 55 which is sufficient to carry out most statistical tests. By construction, the similarity of x_t with respect to its k-earliest neighbors is determined by the similarity of the distributions P_1 and P_2. For the concept learning to be meaningful, it must provide an estimate of the significance of the detected differences. This can be archived by determining whether the two distributions P_1 and P_2 are the same or not i.e by testing the hypothesis

$$H_0 \colon P_1 = P_2 \text{ versus } H_a \colon P_1 \neq P_2 \tag{3}$$

The statistical significance of the test as expressed by the p-value is the probability of obtaining a test statistic at least as extreme as the one actually observed assuming the null hypothesis is true. If the p-value is less than the significance level (the significance level α is set by the user and usually equal to 0.001, 0.01, 0.05, or 0.1) tested, then the null hypothesis H_0 is rejected in favor of the alternative H_a.

There are two basic types of hypothesis testing: parametric and non-parametric test. Hypothesis tests are parametric when the chosen test statistics are assumed to follow some specific distribution (such as normal) with a set of parameters. Non-parametric tests, on the other hand, do not make any (or minimal) assumption on the distribution of the test statistics. They are referred to as distribution free tests. The non-parametric approach is pursued in this study.

5.1.1. Permutation test

In order to perform the hypothesis test, the probability distribution of the chosen test statistic under the null hypothesis needs to be known. However, the distribution of a particular test statistic cannot be computed without some assumption on the data generating process. Non-parametric techniques do not require such distributional assumptions. Permutation tests have become the standard tool for assessing the statistical significance of a hypothesis test without making any distributional assumption of the underlying test statistics unlike in the case of parametric test such as the student t-test.

Let $\{X_{i1}, \ldots, X_{in_i},\ i = 1, 2\}$ be two random samples from a population with distribution functions P_i, $i = 1, 2$ respectively. Consider the problem of testing the hypothesis given in equation 3 without any assumption on the particular form of P_1 and P_2. In this setting, the problem can be reduced to testing if the two populations differ in location or scale by an unknown amount ϑ. Then the test becomes

$$H_0 : \vartheta = 0 \text{ versus } H_a : \vartheta \neq 0 \tag{4}$$

The steps for a two sample permutation test, is as follows:

1. Choose a test statistic T, such as the sample mean or median when testing for difference in location or the deviance when testing for variability in the observations from the two samples.
2. Select an acceptable significance level $\alpha \in (0, 1)$.
3. Let X^* be the set of $(n_1+n_2)!$ points obtained from $X = (X_{ij},\ j = 1, \cdots, n_i,\ i = 1, 2)$ and M be an integer.
   (a) Repeat M times $(m = 1, \ldots M)$
      i. Sample X_m permutations from X^*.
      ii. Compute the test statistic value for this permutation: $t_m = T(X_m)$.
   (b) Construct the empirical cumulative distribution of the test statistic

$$\hat{P}(t) = \frac{1}{M} \sum_{m=1}^{M} I(t_m \leq t)$$

   where $I(.)$ is the indicator function.
   (c) Compute the value of the statistic for the observe distribution $t_0 = T(X)$ and its corresponding p-value p under the empirical distribution.
   (d) If $p < \alpha$ reject the null hypotheses in favor of the alternative.

The null hypothesis assumes that the two distributions are indistinguishable and exchangeable with respect to the chosen statistic, so all the data points generated through permutations are equally likely to be observed under the null hypothesis. Thus, the permutation test is an exact test only if the assumption of exchangeability of data points under the null hypotheses holds. The distributions P_1 and P_2 as constructed above satisfies the assumption of the null hypothesis.

The p-value of the observed statistic computed from the empirical distribution is an exact p-value and its simply the fraction of the permutation values of the test statistic that are at least as extreme as the observed statistic t_0 (derived from the non-permuted data)

$$p = \frac{\sum_{m=1}^{M} I(t_m \geq t_0)}{M} \tag{5}$$

Note that the p-value above corresponds to a right-tail test. A two-tail test can be obtained by simply replacing t_m and t_0 by their absolute values. A confidence bound for p can be obtained as follows; let N be the number of permutation values of the test statistics that exceeds the observed statistic, i.e

$$N = \sum_{m=1}^{M} I(t_m \geq t_0) \Longrightarrow N = Mp \tag{6}$$

Thus, N follows a binomial distribution. If $Mp > 5$ and $M(1 - p) > 5$ then by the central limit theorem the normal distribution can be used to approximate the

sampling distribution of $\bar{p} = \frac{N}{M}$; the fraction of the number of permutation values that exceeds the observed statistics i.e

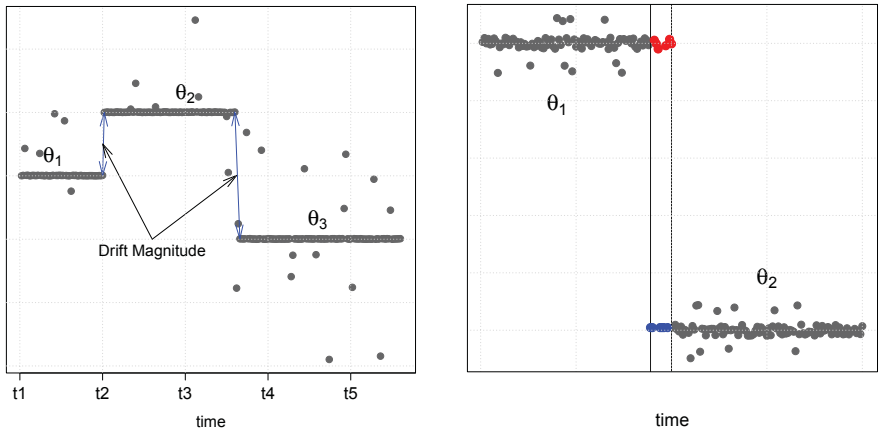
$$\bar{p} \sim \mathcal{N}\Big(p,\ p(1-p)/M\Big) \tag{7}$$

a normal distribution with mean p and standard deviation $\sqrt{p(1-p)/M}$. A known confidence bound for the p-value for a gradual or sudden concept drift can be used to determine the nature of feature changing concepts.

5.1.2. Learning new concepts

A new concept is learned by the application of the permutation test to determine if the data points from the two sliding windows represent different concepts. Three different sliding window algorithms are proposed. The first algorithm **OneFixed** has one window W_1 fixed and the second window W_2 moves and detects new concepts. When a possible new concept is detected, a third window W_3 is setup to confirm the change. The second algorithm **TwoMoving** has both W_1 and W_2 moving and is designed to eliminated some of the limitations of **OneFixed** (to be discoursed). Finally, the third algorithm **SynTwoMoving** is similar to the second but replaces the points in W_3 with artificially generated points.

The first two algorithms are described in this Section while the third is described in Section 5.2.



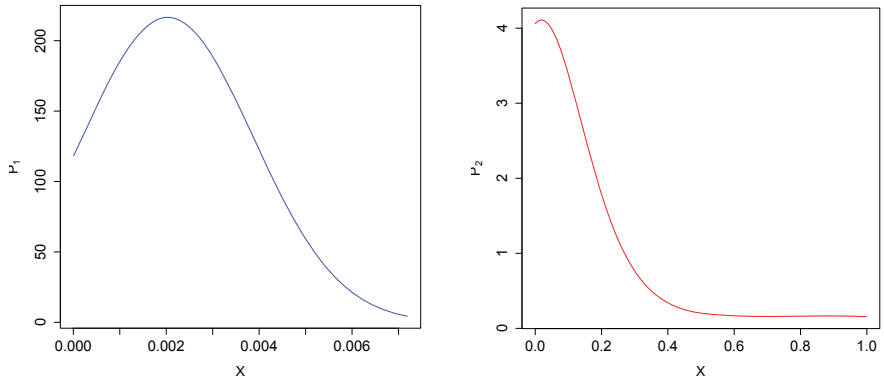
(a) Sudden Drift (b) Gradual Drift

Figure 2. Sudden and Gradual Concept Drift.

A. OneFixed algorithm

Algorithm 2 presents the pseudocode of the **OneFixed** algorithm. The first window W_1 is held fixed with its points representing a learned concept. The second window W_2 moves along by adding any new data point x_t that comes in while the first point x_{t-k}

is discarded. The distribution of points in the two windows P_1 and P_2 is constructed as described in Section 5.1, and the permutation test is performed to test for a change in the concept. Using the difference in medians as test statistics generally produce robust tests than a difference in means. However, for the experiments performed in this study, the difference in mean was found to perform better than the median. This can be explained by the fact that the distribution P_2 near or at a concept drift is highly asymmetrical and heavily right-tailed. The median is not as strongly influenced by the skewness a distribution as the mean, thus the mean quickly captures this difference. Figure 2(a) illustrates sudden concept drift where for example at time t_2 the concept θ_1 is suddenly replaced by a new concept θ_2. The skewness of P_1 and P_2 is displayed in Figure 3. P_1 is the distribution of points to the left of the concept drift at t_2 while P_2 is the distribution of those same points including the drift point. The plots clearly shows the heavily right tailed P_2 while P_1 is only slightly right tailed.



(a) P_1 (b) P_2

Figure 3. Local kernel density estimates of P_1 and P_2 at a concept drift.

Given a significance level α, the p-value p and its $100(1-\alpha)\%$ confidence interval for equal means between P_1 and P_2 is computed by application of the permutation test. If $p < \alpha$ then point x_t is declared a possible concept drift or outlier. If more points are available after x_t, then these points can be used to determine the true nature of x_t. When this is the case, a third window W_3 called the *verification window* is constructed for points to the right of x_t. A new window width k_3 can be chosen for W_3. The distribution P_3 of points in W_3 is similarly constructed as for P_1 and P_2. A second test is carried out to distinguish between P_1 and P_3 and a new p-value p_1 computed. Note that the influence of the potential drift point x_t is excluded from this second test by not involving P_2 in the test. The outcome of the second test categorizes two different states for x_t:

1. if $p_1 > \alpha$ then x_t is declared an outlier,
2. if $p_1 \leq \alpha$ then x_t is declared a concept drift. A second level of significance can be selected for this second test.

The above verification test has one major drawback. Since the distributions are obtained by taking the mutual distances between points in each window, it is possible for P_1 and P_3 to represent quite different concepts, but the test shows no significant difference. This is particularly true for sudden concept drifts. This problem can be resolved by creating two new distribution for points in the verification window: P_3 the distances of all points in this window including the proposed drift point x_t and P_4 excluding it. The test is now between P_3 and P_4 and interpreted as follows:

1. if $p_1 > \alpha$ then x_t is declared a concept drift,
2. if $p_1 \leq \alpha$ then x_t is declared an outlier.

After x_t has been tested and confirmed, the old concept represented by W_1 is stored in memory along with some parameters of its representative distribution P_1. Then, P_1 is updated to P_3 and the next test starts at the point x_{t+k_3+1}. Thus, the new W_2 now contains the second point of W_3 up to x_{t+k_3+1}.

The **OneFixed** algorithm as described has one major limitation: its performance quickly deteriorates in the presence of complex concept drift such as gradual drifts and on data contaminated by noise. Figure 2(b) shows an illustration of a data set with gradual concept drift. At the transition phase between old and new concept, data points with mixed concepts are present with a variable drift rate that declines gradually. Since one window is always held fixed, the variability of its constituent points stay constant, and so it may fail to detect the gradual change. Therefore, the performance of the algorithm drops drastically at each learning phase transition.

One possible approach to address this problem is to update the fixed window to the current window once the p-value of the test is outside a predefined range. This range can be determined by a prior computation of the confidence interval of the p-value for a gradual change. However, a similar problem is encountered here as in the case of determining the optimal proximity measure for the **Basic** algorithm.

B. TwoMoving algorithm

The **TwoMoving** algorithm is designed to overcome some of the limitations of the **OneFixed** algorithm. Instead of holding one window fixed, both widows are allowed to move simultaneously. Thus, in the presence of a gradual change in concept, the algorithm will remain relatively stable and capture the change. At any given time step t, the first window W_1 always contains the points $\{x_{t-k}, \ldots, x_{t-1}\}$ while the second W_2 contains $\{x_{t-k+1}, \ldots, x_t\}$. Thus, W_2 contains all but the first point of W_1, while W_1 does not contain the last point of W_2.

The testing procedure is the same as for **OneFixed**. However, if outliers or multiple drifts are present in the verification window, the testing procedure may fail or produce unpredictable results. The next section describes the third sliding window technique which significantly improves the performance of the **TwoMoving** algorithm in handling outliers.

Algorithm 2: OneFixed.

Input : Training data set $\mathcal{X}$, window width k, significance level α

$t \leftarrow 1$;

$P_1 \leftarrow \left\| \mathcal{X}[1:k] \right\|$; `// first window initialized to first concept`

repeat

  $P_2 \leftarrow \left\| \mathcal{X}[t:k+t] \right\|$; `// distribution used to learn new concepts`

  $\text{test}_1 \leftarrow$ `PermutationTest(`P_1, P_2`)` ;

  if $test_1 < \alpha$ **then**

    $T \leftarrow k + t$; `// possible new concept starts here`

    $P_3 \leftarrow \left\| \mathcal{X}[T:k+T] \right\|$; `// start a possible new concept`

    $P_2 \leftarrow \left\| \mathcal{X}[T+1:k+T] \right\|$;

    $\text{test}_2 \leftarrow$ `PermutationTest(`P_1, P_2`)` ;

    if $test_2 \geq \alpha$ **then**

      $\mathcal{X}[T]$ represents start of a new concept;

      $P_1 \leftarrow P_3$ Update the learned concept ;

      $t \leftarrow T + k + 1$;

    else

      $\mathcal{X}[T]$ represent noise or the look ahead window contain noise. Replace point with mean/median ;

      $t \leftarrow t + 1$;

    end

  else

    $t \leftarrow t + 1$;

  end

until $t > samplesize$;

5.2. Handling noisy data

In many data analysis tasks, outliers are often considered as errors or noise, however, they may carry important information. This is particularly true for the real medical claims data studied in this paper. Denied or underpaid claims occurred as outlaying observations. An important task of the healthcare provider is to be able to predict when these aberrant cases may occur and if possible their values. Thus, it is not only important to detect when a change in payment occurs but also when an unusual payment is made.

Detecting changes in concepts when the data is corrupted by noise is an important problem in concept drift learning that is not often investigated. Most concept drift algorithms will perform optimal on noise free data sets, but in the presence of noise it is quite common for some algorithms to overreact to the noise, erroneously interpreting them as a concept drift. On the other hand, some algorithms may be too robust to

noise and fail to detect or react too slowly to actual changes [13] in the data. Thus, a good concept drift learning algorithm should be able to accurately detect the various types of drifts and to distinguish them from noise.

The **OneFixed** and **TwoMoving** algorithms as presented in Section 5.1.2 can accurately detect outlaying observations if they occur before a drifting concept. However, if outliers are present immediately after a drift, then these algorithms may fail to detect a drift in concept. Moreover, they will perform quite poorly on very noisy data. A straightforward approach to improve their performance is to run them on a filtered data set. The disadvantage of doing so lies in the fact that most outlier detection algorithms are not designed to detect concept drifts, so may flag out some observations representing a change in concept as outliers. An alternative approach that was implemented for **OneFixed** and **TwoMoving** is to only check for outlaying observations in the verification window W_3. For instance, data points in W_3 that deviate significantly from the mean or median are flagged as outliers before the test is performed. The effectiveness of this approach depends greatly on the size of verification window k_3 and the outlier detection method. A large window size may detect all outliers but run the risk of flagging out extra drifts that may have been included. Small sizes may fail to detect any outlier.

A third approach that completely eliminates almost all of the problems described above, leading to a remarkable improvement in the performance of the **TwoMoving** algorithm is described in Algorithm 3. This algorithm which will be called **SynTwoMoving** simply replaces the distribution of points in the verification window by a synthetic distribution. For example, the points in W_3 can be replaced by random variates from a normal distribution with mean and standard deviation equal to the *median* and *median absolute deviation* of W_3 respectively. This synthetic distribution completely eliminates any potential outliers and accurately reflects the true distribution of these points. So any test performed with reference to this distribution is free of any aberrant observations. The permutation test proceeds as before, where now P_3 is the distances of all points in the synthetic distribution including the potential drift point x_t, while P_4 is simply the distances of all points in the synthetic distribution. Optionally, after the test has confirmed a change in concept, each point in the verification can be checked against the synthetic distribution using the permutation test for outliers. That is, a new distribution P_5 can be created by taking the distance of each point in the original W_3 to all points in the synthetic distribution and tested against P_4. This extra check is important so as to flag out outliers that could interfere with subsequent tests. This additional check also has the advantage of revealing the possible presence of multiple drifts within the window.

6. Experiments

This section evaluates the proposed algorithms presented in this study using synthetic as well as real-world data. The performance of the algorithms will be compared against the CUSUM [9] detection method in terms of the number of "change-

Algorithm 3: SynTwoMoving.

Input : Training data set $\mathcal{X}$, window width k, significance level α

$t \leftarrow 1$;
repeat
  $P_1 \leftarrow \left\| \mathcal{X}[t : (k+t)] \right\|$;
  $P_2 \leftarrow \left\| \mathcal{X}[(t+1) : (k+t+1)] \right\|$;
  $p_1 \leftarrow$ `PermutationTest`(P_1, P_2) ;
  if $p_1 < \alpha$ **then**
    $T \leftarrow k + t + 1$; `// potential drift or outlier`
    $y \leftarrow \mathcal{X}[(T+1) : (T+k)]$; `// points in verification window`
    $MED \leftarrow$ `Median`(y) ;
    $MAD \leftarrow$ `Median Absolute Deviation`(y) ;
    `// generate synthetic distribution`
    $x \leftarrow$`Normal`$(size = k, \mu = MED, \sigma = MAD)$;
    $P_3 \leftarrow \left\| x \cup \mathcal{X}[T] \right\|$; `// include potential drift or outlier`
    $P_4 \leftarrow \left\| x \right\|$;
    $p_2 \leftarrow$`PermutationTest`(P_3, P_4) ;
    if $p_2 < \alpha$ **then**
      `// point is outlier, replace with median`
      $\mathcal{X}[T] \leftarrow MED$;
      $t \leftarrow t + 1$; `// move to next point`
    else
      `// point marks start of a new concept`
      $t \leftarrow T + 1$;
      `// might be more outliers in verification window`
      for $i \leftarrow T + 1$ **to** $T + k$ **do**
        $P_5 \leftarrow \left\| x \cup \mathcal{X}[i] \right\|$;
        $p_3 \leftarrow$`PermutationTest`(P_4, P_5) ;
        if $p_3 < \alpha$ **then**
          `// point is outlier, replace with median`
          $\mathcal{X}[i] \leftarrow MED$;
        else
          `// point might be a new concept`
          `// in verification window`
          $t \leftarrow i$;
          `Break`;
        end
      end
    end
  else
    `// point is normal`
    $t \leftarrow t + 1$;
  end
until $t > samplesize$;

points" detected under conditions of noisy and non-noisy data streams with or without change-points. A Monte Carlo simulation was carried out for the CUSUM method on synthetic data sets with different percentages of noise and change-points to determine the optimal value of h to be used in all the experiments.

6.1. Performance measure

The precision, recall and F_1 score measures where used to assess the performance of all algorithms in detecting drifts and outlaying observations. The precision is defined as the number of correctly detected concept drifts/outliers divided by the total number of detected concept drifts/outliers i.e

$$\text{Precision} = \frac{\#\{\text{correctly detected}\}}{\#\{\text{detected}\}} \tag{8}$$

The precision is simply the probability that a detected concept drift/outlier is actually a concept drift/outlier. Recall, on the other hand, is the probability that the drift algorithm detects a true concept drift/outlier, i.e it is the number of correctly detected concept drifts/outliers divided by the total number of true concept drifts/outliers i.e

$$\text{Recall} = \frac{\#\{\text{correctly detected}\}}{\#\{\text{true concept drifts/outliers}\}} \tag{9}$$

F_1 score is a weighted average of the precision and recall rates, where a high value of the F_1 score ensures that the precision and recall rates are reasonably high. The harmonic mean of the precision and recall was used to compute the F_1 score:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

6.2. Synthetic data

To explore the advantage of the proposed algorithms, numerous experiments were conducted on a series of synthetic data sets. Five types of data sets were generated and for each type 100 different copies each of sample size 500 were produced containing a random number of drift points. Specifically, each data may contain up to 10 drift points. Noisy data sets contain between 5% and 20% noise. The complete description of the five data sets is as follows:

1. $\mathcal{D}_0$: The first synthetic data contains no drift. For a relatively small percentage of noise, the expectation is for all algorithms to perform close to optimal.
2. $\mathcal{D}_1$: The second data set is noiseless and contains a random number of step-like concept drifts.
3. $\mathcal{D}_2$: Contains a random number of step-like concept drifts with noise.
4. $\mathcal{D}_3$: Contains a random number of gradual and step-like concept drifts and noise free.
5. $\mathcal{D}_4$: Contains a random number of gradual and step-like concept drifts with noise.

The real medical insurance claim data used in this study is characterized by sudden and gradual changes in payments at unknown times within a specific contract period. Underpaid or denied payments are the abnormal payments and were considered as noise. Thus, most of the analysis will be based on the data set $\mathcal{D}_4$.

Concept drift detection was conducted on these data sets with a fixed window width of $k = 20$ and a significance level $\alpha \in [0.001, 0.05]$. Table 1 shows that the **SynTwoMoving** algorithm outperforms all other algorithms on almost all data sets in detecting the various types of concept drift. As expected, on noise free data sets with step-like drifts, the **Basic** and CUSUM algorithms perform best and poor under more complex drift type data sets.

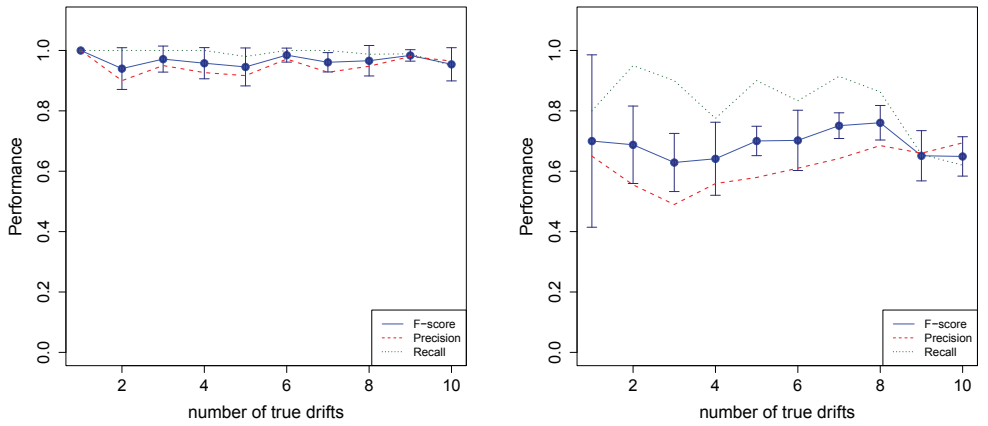
Table 1

Comparing drift detection for **Basic, OneFixed, TwoMoving, SynTwoMoving** and **CUSUM** for 100 copies of data.

Data	Performance	Algorithm				
		Basic	**OneFixed**	**TwoMoving**	**SynTwoMoving**	**CUSUM**
$\mathcal{D}_0$ with 20% noise	No of Drifts Detected	4	10	3	0	15
$\mathcal{D}_1$	Precision	1.0	0.66	1.0	1.0	0.99
	Recall	1.0	0.89	1.0	0.93	1.0
	Fscore	1.0	0.74	1.0	0.96	0.99
$\mathcal{D}_2$	Precision	0.30	0.21	0.65	0.95	0.34
	Recall	0.30	0.55	1.0	0.94	0.99
	Fscore	0.30	0.29	0.77	0.94	0.48
$\mathcal{D}_3$	Precision	0.12	0.63	1.0	0.98	0.38
	Recall	0.12	0.92	1.0	0.99	0.86
	Fscore	0.12	0.74	1.0	0.99	0.51
$\mathcal{D}_4$	Precision	0.03	0.27	0.78	0.99	0.31
	Recall	0.03	0.58	0.86	0.97	0.81
	Fscore	0.03	0.36	0.82	0.97	0.43

Figures 4 and 5 show the various performance measures of the algorithms plotted against the number of random true drifts points in the fifth data set $\mathcal{D}_4$, i.e a noisy data set with gradual and step-like concept drifts. The number of drift points has no apparent impact on the performance of the **SynTwoMoving** algorithm.

The detection of outlaying observations is an important factor prior to any modeling and analysis. It is therefore required for a concept drift algorithm to have high discriminative power, distinguishing changing concepts from outliers. This is very important especially when there is a cost associated with a false positive. Table 2 shows the performance of the algorithms in detecting outlaying observations. No results for CUSUM is shown because the algorithm was not designed to detect outliers.



(a) SynTwoMoving (b) TwoMoving

Figure 4. SynTwoMoving and TwoMoving performance measures of the algorithms depending on the number of random true drift points in the fifth data set $\mathcal{D}_4$.

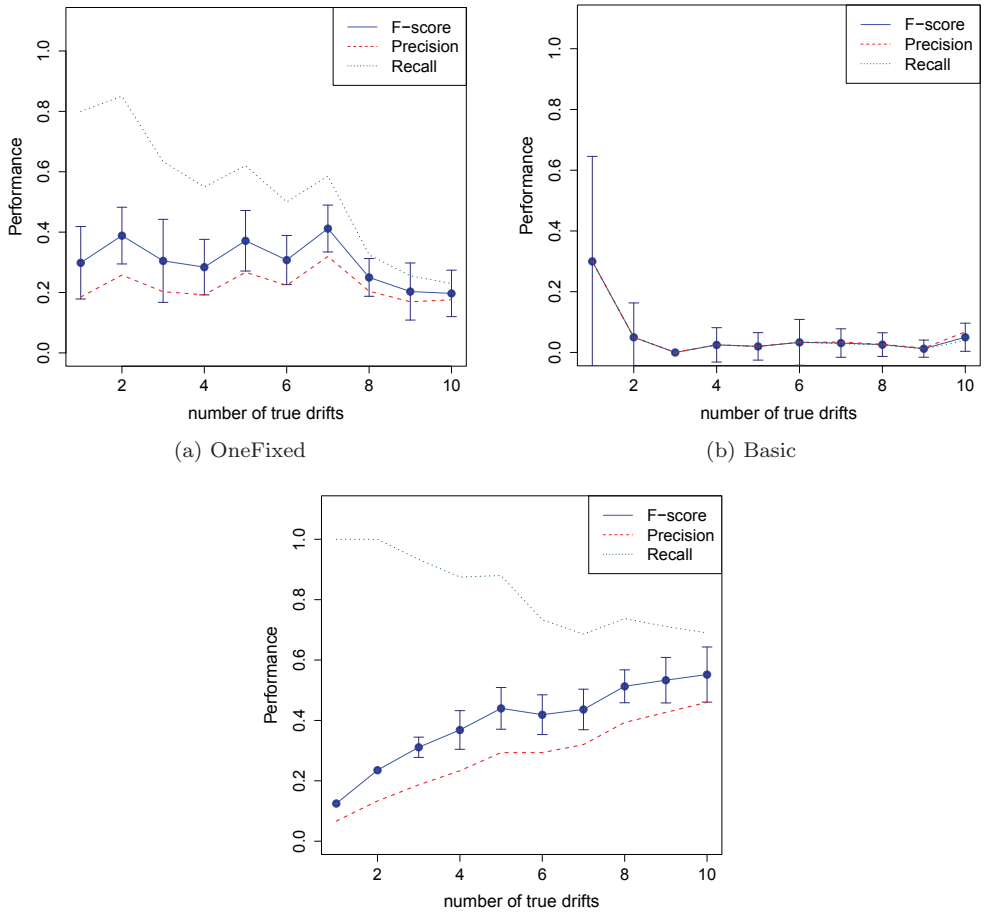
Once more the superior performances of the **SynTwoMoving** and **TwoMoving** algorithms can be clearly seen.

Table 2

Comparing outlier detection for **Basic**, **OneFixed**, **TwoMoving** and **SynTwoMoving** for 100 copies of data.

Data	Performance	Algorithm			
		Basic	OneFixed	TwoMoving	SynTwoMoving
$\mathcal{D}_2$	Precision	0.47	0.76	0.85	0.94
	Recall	1.00	0.72	0.72	0.97
	Fscore	0.59	0.73	0.77	0.95
$\mathcal{D}_3$	Precision	0.0	0.69	0.84	0.67
	Recall	0.0	0.93	0.97	0.96
	Fscore	0.0	0.78	0.90	0.78
$\mathcal{D}_4$	Precision	0.37	0.81	0.86	0.92
	Recall	0.82	0.71	0.70	0.99
	Fscore	0.48	0.74	0.75	0.96

For a fixed number of concept drifts, it is interesting to investigate the performance of the algorithms for different levels of noise in the data. Figures 6 and 7 show how the performance of the algorithms varies with the percentage of outliers in data set $\mathcal{D}_4$ with 5 random concept drifts (sudden and/gradual). The **SynTwoMoving** remains very accurate up to a percentage outlier level of 25%. Notice how the performance of **OneFixed** drops rapidly on noisy data sets with gradual concept drift.



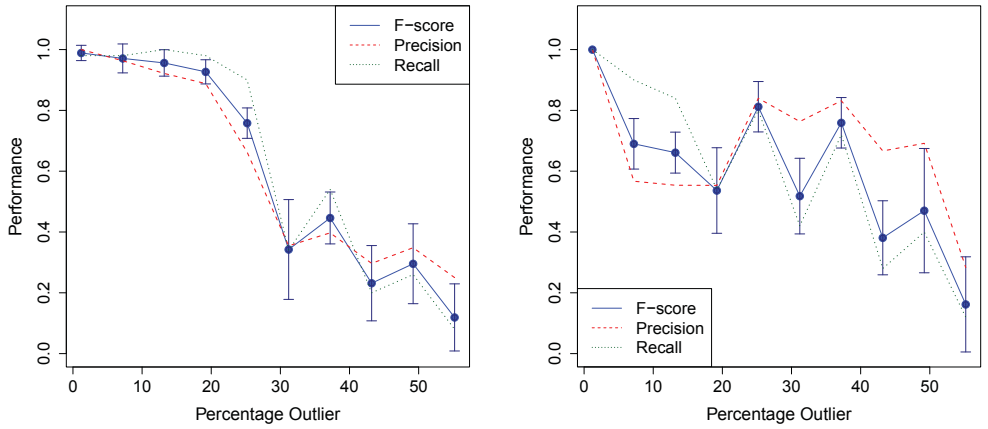
(a) OneFixed (b) Basic

(c) CUSUM

Figure 5. OneFixed, Basic and CUSUM performance measures of the algorithms depending on the number of random true drift points in the fifth data set $\mathcal{D}_4$.

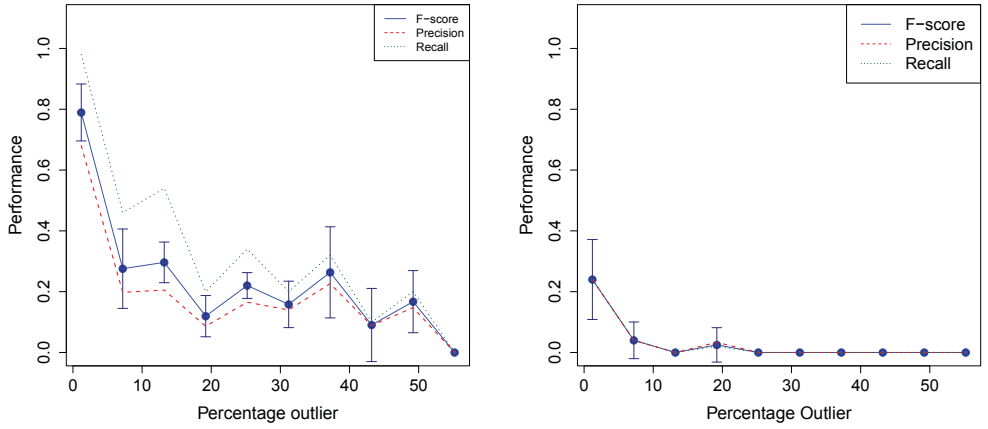
Finally, the performance of the algorithms was evaluated with respect to different sample sizes. Because the **SynTwoMoving** algorithm shows superior performance over all the others, only its results will be given. A sequence of $\mathcal{D}_4$ data sets of sample sizes ranging from 50 to 2000 each having 5 random concept drifts and 15% outliers was generated and the performance of the algorithms evaluated on each set.

Figure 8 shows that the performance of **SynTwoMoving** remains relatively constant as the sample size increases.



(a) SynTwoMoving (b) TwoMoving

Figure 6. The performance of the algorithms depending on the percentage of outliers in data set $\mathcal{D}_4$ for SynTwoMoving and TwoMoving performance measures.



(a) OneFixed (b) Basic

Figure 7. The performance of the algorithms depending on the percentage of outliers in data set $\mathcal{D}_4$ for OneFixed and Basic performance measures.

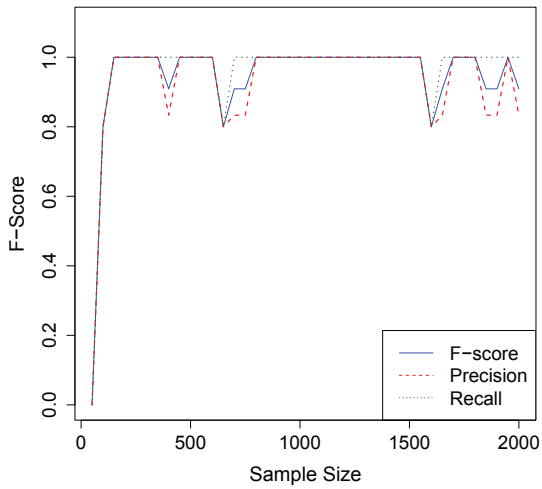


Figure 8. Sample size by performance measure.

6.3. Real data

The performance of the algorithms was also evaluated using real medical insurance claims data sets obtained from two hospitals in the INOVA Health System of Northern Virginia. The data consists of claims payments for obstetric patients based on a specific Diagnosis Related Groups (DRG) code for the years 2008 and 2009. This data was previously used in [16] where a simple off-line labeling approach was introduced to label the data for subsequent classification by the AQ21 machine learning system [15]. Since this is a real data set, the location of concept drifts and outliers are unknown, thus it is not possible to provide the same performance measures as in the synthetic case. However, based on the experiments performed in [16] and some domain expert knowledge, information about changes in payments and percentage of outliers in the data sets have been converted into rules that indicate normal and abnormal payments. Calling the data sets from the two hospitals $\mathbf{H}_1$ and $\mathbf{H}_2$ respectively, an analysis of the data sets indicated that both contained two concept drifts occurring approximately in the month of June of each year. Data set $\mathbf{H}_1$ contained 3045 records with approximately 2.5% outliers while $\mathbf{H}_2$ contained 748 records with about 4.5% outliers.

The **SynTwoMoving** and **TwoMoving** algorithms were used to study concept drifts on both data sets. The number of concepts drifts and percentage outliers detected by both algorithms are given in Table 3. Based on domain experts (i.e the health care providers) knowledge about the possible times insurance claims payments change and the percentage outliers in the data, it can be seen that the **SynTwoMoving** results matches very closely to this prior knowledge.

Table 3
Performance of **SynTwoMoving** and **TwoMoving** on real medical insurance claims.

Data	Performance	Algorithm		
		SynTwoMoving	**TwoMoving**	Expert
$\mathbf{H}_1$	# of drifts	2 (06-29-08 and 06-30-09)	3 (06-29-08, 08-16-08 and 06-30-09)	2
	% outliers	2.3%	2.2%	2.5%
$\mathbf{H}_2$	# of drifts	2 (06-29-08 and 06-29-09)	4	2
	% outliers	4.3%	3.9%	4.5%

7. Conclusion

This paper introduces unsupervised learning algorithms for labeling noisy data streams characterized by drifting concepts. Three presented unsupervised on-line learning algorithms (**SynTwoMoving, TwoMoving** and **OneFixed**) were developed based on the permutation test statistics.

Experiments on synthetic and real datasets showed that the **SynTwoMoving** and **TwoMoving** algorithms are well capable of coping with sudden and gradual concept drifts. Moreover, the **SynTwoMoving** algorithm dramatically outperforms the other algorithms with respect to accuracy, stability and robustness to the number of concept drifts and outliers in the data.

While the methods have been developed for the specific application in medical claims processing before concept learning, it is as well applicable to other domains in which there is a need for unsupervised labeling of data streams for the purpose of supervised learning, anomaly detection etc.

Future extensions of the methods are possible in several potential directions including: the use of adaptive window sizes, using different hypothesis and test statistics, optimizing the methods' parameters to archive the desired false positive and true positive rates required by different applications. In terms of the larger project concerning medical claims payment prediction, the method can be converted into a real-time processing system and integrated with billing systems.

Acknowledgements

The authors thank Michael Smith, Jeff Weinberg, and Mary Palinski for their help in deriving data from the Inova's financial management system, and other project team members Ron Ewald, Jay Shiver, and Kat Irvin. The authors also thank Chris Jose for her edits to the paper. The presented work has been supported by the Mason-Inova fund grant.

References

[1] Cappelli C., Penny R., Rea W., Reale M.: Detecting multiple mean breaks at unknown points in official time series. *Mathematics and Computers in Simulation*, 78(2):351–356, 2008.

[2] Easterling D., Peterson T.: A new method for detecting undocumented discontinuities in climatological time series. *International Journal of Climatology*, 15(4):369–377, 1995.

[3] Hadas D., Yovel G., Intrator N.: Using unsupervised incremental learning to cope with gradual concept drift. *Connection Science*, 23(1):65–83, 2011.

[4] Hinkley D.: Inference about the change-point in a sequence of random variables. *Biometrika*, 57(1):1–17, 1970.

[5] Karunanithi A., Cabezas H., Frieden B., Pawlowski C.: Detection and assessment of ecosystem regime shifts from fisher information. *Ecology and Society*, 13(1):22, 2008.

[6] Lazarescu M., Venkatesh S., Bui H.: Using multiple windows to track concept drift. *Intelligent Data Analysis*, 8(1):29–60, 2004.

[7] Loader C.: Change point estimation using nonparametric regression. *The Annals of Statistics*, 24(4):1667–1678, 1996.

[8] Mantua N.: Methods for detecting regime shifts in large marine ecosystems: a review with approaches applied to north pacific data. *Progress in Oceanography*, 60(2):165–182, 2004.

[9] Page E.: Continuous inspection schemes. *Biometrika*, pp. 100–115, 1954.

[10] Rodionov S.: A brief overview of the regime shift detection methods. In *Large-scale disturbances (regime shifts) and recovery in aquatic ecosystems: challenges for management toward sustainability. UNESCO-ROSTE/BAS Workshop on Regime Shifts, Varna, Bulgaria*, pp. 17–24, 2005.

[11] Sharifzadeh M., Azmoodeh F., Shahabi C.: Change detection in time series data using wavelet footprints. *Advances in Spatial and Temporal Databases*, pp. 923–923, 2005.

[12] Son Y., Kim S.: Bayesian single change point detection in a sequence of multivariate normal observations. *Statistics*, 39(5):373–387, 2005.

[13] Tsymbal A.: *The problem of concept drift: definitions and related work*. Computer Science Department, Trinity College Dublin, 2004.

[14] Widmer G., Kubat M.: Effective learning in dynamic environments by explicit context tracking. In *Machine Learning: ECML-93*, pp. 227–243. Springer, 1993.

[15] Wojtusiak J., Michalski R., Kaufman K., Pietrzykowski J.: The aq21 natural induction program for pattern discovery: initial version and its novel features. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pp. 523–526. IEEE, 2006.

[16] Wojtusiak J., Ngufor C., Shiver J., Ewald R.: Rule-based prediction of medical claims' payments: A method and initial application to medicaid data. In *Machine*

Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on, vol. 2, pp. 162–167. IEEE, 2011.

[17] Wojtusiak J., Ngufor C., Shiver J., Ewald R.: *Development and testing of artificial intelligence application for healthcare financial management: Methods and initial results.* Technical Report, Reports of the Machine Learning and Inference Laboratory, 2013.

Affiliations

Che Ngufor
George Mason University, VA USA, cngufor@masonlive.gmu.edu

Janusz Wojtusiak
George Mason University, VA USA, jwojt@mli.gmu.edu

Received: 16.10.2012
Revised: 22.01.2013
Accepted: 11.02.2013