The Impact of Latency on Online Classification Learning with Concept Drift

Gary R. Marrs, Ray J. Hickey, Michaela M. Black

School of Computing and Engineering, University of Ulster, Coleraine, County Londonderry, N. Ireland
marrs-g@email.ulster.ac.uk,
{mm.black, rj.hickey}@ulster.ac.uk

Abstract. Online classification learners operating under concept drift can be subject to latency in examples arriving at the training base. A discussion of latency and the related notion of example filtering leads to the development of an example life cycle for online learning (OLLC). Latency in a data stream is modelled in a new Example Life-cycle Integrated Simulation Environment (ELISE). In a series of experiments, the online learner algorithm CD3 is evaluated under several drift and latency scenarios. Results show that systems subject to large random latencies can, when drift occurs, suffer substantial deterioration in classification rate with slow recovery.

Keywords: Online Learning, Classification, Concept Drift, Data stream, Example life-cycle, Latency, ELISE, CD3

1 Introduction

Online learning for classification involves inducing an initial classifier and updating this at intervals from a data stream of time-stamped training examples. Current approaches deploy a variety of machine learning algorithms either individually or in ensembles. They use a variety of means of handling concept and /or population drift and of maintaining a set of valid training examples. See [1], [2] and [3] for an introduction to and review of existing work and underlying issues; the potential for lack of representativeness of examples in a data stream is discussed in [4].

There has been very little discussion in the literature on the time-stamp itself and what it represents. Implicitly it is taken to be the time at which an example becomes available to the learning algorithm for use at some point in the future.

A training example is supposed to be representative of some underlying true rule operating at a particular time. Yet an example may only become available *after* that time. This leads to a time discrepancy or *latency*. Such latencies may impact on learning updates when drift occurs and are the subject of this work.

In section 2, latency is discussed. This leads to an extended model of the life cycle of an example within an online learning system. An experimental test-bed for investigating the impact of latency on learning is described in section 3 and results are presented and analysed in section 4.

2 Latency and the Online Learning Life-Cycle

In on-line classification learning, the learning algorithm receives new training examples on a periodic basis and adopts a learning update regime to maintain currency of the classifier. These new examples are time-stamped and placed in a training base. The time-stamps will influence the learner.

The issue as to what the time-stamp of a new training example should be has received little attention in the literature. When a classifier makes a classification at time *t*, it is attempting to replicate the classification that would be made by an oracle (possibly a human expert), were such available. This oracle is indexed by time and can change its rules over time, i.e. concept drift. It is the role of the learning regime to try to capture and maintain the oracle's rules over time.

In due course, the true class for this example may be revealed. Consider a credit card fraud prevention system which aims to identify fraud at the time of transaction. Suppose the current classifier accepts the transaction as legitimate. At a later date it may become apparent that the transaction was fraudulent. This information may be returned to the online learning system as a new training example.

When fed to the learner at the next update of learning, what should the time-stamp of this example be? At first sight it might appear that the time-stamp should be that at which the fraud was verified. The training example, however, is to be regarded as a window into the behaviour of the oracle that was current at the time of classification. The time-lapse until verification is seen as purely administrative. Although a decision is made by the company, at the time of verification, to designate the transaction as fraudulent, perhaps after extensive investigation, this decision-making process is distinct from that of the oracle operating at the time of classification. Thus the appropriate time-stamp for the training example, when presented to the learner, is the time of classification.

The time-interval from classification until the verified class for the example is discovered is called its *verification* latency.

In contrast, consider an online medical diagnostic system which receives, from time to time, cases that have been classified by a medical expert. The aim of this system is to emulate the expert. Here the verification latency is zero: there is no gap between the application of the oracle, i.e. the expert, and the appearance of the verified class.

It may transpire in the future that the classification made by the medical expert was wrong, e.g. the patient did not have the disease as diagnosed. Given, however that the purpose of the learner is to emulate the expert, this is not relevant. Were the disease that the patient actually had to be regarded as the verified class and fed back to the learner, this would amount to using nature as the oracle and not the human expert. In this latter situation, the system becomes identical to the fraud system and verification latency should be regarded as the time difference between that when the patient's symptoms were noted and when the correct diagnosis was obtained.

Following verification, there may be a further delay, possibly administrative, before a training example becomes available as a new example to the learner. Such delays can affect both types of systems discussed above.

Clearly latency only becomes an issue should concept drift occur. In a new episode of learning to update the classifier, some or all of the examples received since the last episode will be input to the learner. If drift has occurred from one episode of learning to the next, then some examples will reflect the old oracle and so may be out of date. The extent to which this is the case depends on the magnitude of latency and the time between learning episodes.

Depending on the application, latency can be fixed or random. In a system which predicts rise or fall in a share price from the close of business in a stock market from the end of one day until the end of the next day, verification latency is fixed at one day and there is no administrative delay. In the fraud application, verification latency and administrative delay are likely to be random. With random latency, training examples will become available out of chronological order.

Random latency may be different for different classes or may depend on the example description. In a loan approval system, where the classes are *applicant will/will not default on loan*, the class *will default on loan* can become known before the end of the loan period, whereas the class *will not default on loan* cannot be determined until the loan is completed.

In addition to latency, examples can be subject to filtering. Trackability filtering refers to the rejection of examples so that they never return to the example base. This is referred to as one-sided feedback [5]. In the fraud domain, transactions classified as fraudulent will be blocked thus no verified class will ever be obtained An example filtered in this way can also be regarded as having infinite verification latency.

Selection filtering refers to the sampling of examples for verification and, therefore, their subsequent availability to return as training examples. For instance, in an online spam detection system it would be highly unlikely that every email sent would be verified as spam or non-spam.

In general, filtering can be class dependent and may also be dependent on attribute values within the description of an example.

2.1 The Online Learning Life Cycle

Table 1. OLLC Stages

Stage	Description
Initial example collection	Initial supervised example collection takes place using external sources and is placed into the training base.
Initial classifier induction	Upon receiving data from the training base, the learning algorithm is applied and generates the first classifier.
Classification	A new case, <description>, arrives at time t_c for classification, This is given its predicted class, *pclass*, by the current classifier and stored as (t_c, <*description*>, *pclass*).
Verification and return to the example base	At time, $t_v > t_c$, the true class, *vclass*, may be obtained. Verification latency is t_1^{lat}. After further delay, t_2^{lat}, this may be fed back, at t_{ab}, to the example training base as *(<description>, vclass)*.
Learning update regime	The learner is applied periodically to examples returned to the training example base since the last episode of learning.

The online learning life cycle (OLLC) can be modeled to take account of latency and filtering. The key stages are summarised in Table 1 and illustrated in Figure 1.

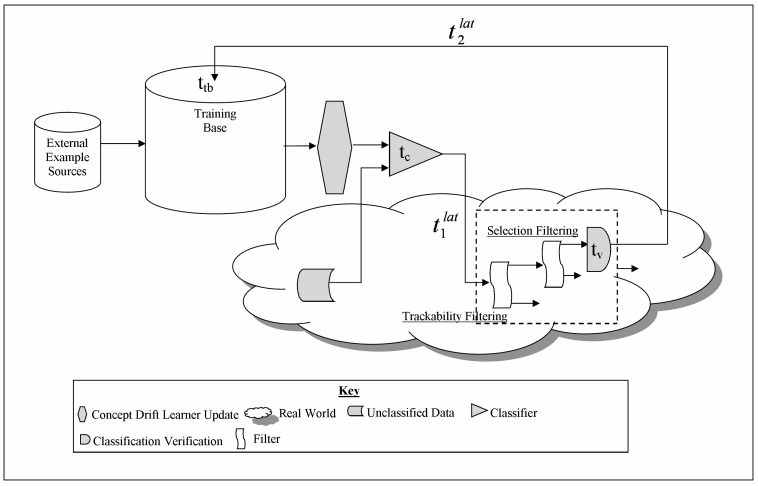

Figure 1. The OLLC Model

3 Experiments on Latency

To investigate the impact of latency on learning under concept drift, a series of experiments was performed using a variety of latency and drift scenarios. In these, latency was modeled as both fixed and random and was assumed to be independent of description attribute values of the examples and of class. No filtering was applied.

3.1 Data Sets

Training and test examples were generated using AutoUniv [6]. AutoUniv creates an artificial universe (U), a complete probabilistic model of the joint distribution of the description attribute and the class, comprising four components: attribute definitions; attribute factorization (into independent factors); attribute factor distributions; and, a rule set. Noise is modelled as the degree of uncertainty in the class distributions of each rule.

Drifted models can be obtained by retaining attribute and class definitions and altering some of the other components of the universe. Drift can be in rules only (concept drift) or in attribute distributions only (population drift) or both.

For the experiments, three universes were generated all with the same attributes and classes, details of which are summarised in table 2. The second universe was obtained by applying concept drift to the first; the third was obtained by applying concept drift to the second. Details are given in tables 3 (a) and 3 (b).

Table 2. Universe attributes (common to all drift variations).

Relevant Attributes	Noise Attributes	Attribute Factors	Minimum Number of Attribute Values
8	2	3	2
Maximum Number of Attribute Values	**Number of Classes**	**Minimum Rule Length**	**Maximum Rules Length**
5	4	2	5

Table 3. (a) Universe drift variations.

Universe	Rules	Average Rule Length	Noise %	Bayes Rate %
1	60	4.2	21.4	78.6
2	88	4.3	19.5	80.5
3	82	4.3	22.1	77.9

(b) Cross-classification rates (XCR) between universes.

Old	New	XCR (%)
Universe 1	Universe 2	32.1
Universe 2	Universe 3	26.8
Universe 1	Universe 3	33.6

From table 3 (a) all three universes have similar Bayes rate, i.e. the maximum classification accuracy possible. The complement of the Bayes rate is the noise level.

The cross-classification rate (XCR), [7], provides a simple measure of the extent of drift. This is defined as the classification rate that would be obtained if the universe rules operating before a drift point were applied to classify examples after drift had occurred. Intuitively this provides a base-line for the consequences of failing to detect drift. The XCR is bounded above by the Bayes rate for the drifted universe, that is, no rule set can outperform the true rules. From table 3(b) it is seen that XCR is very low, in relation to the corresponding Bayes rate in all cases. If an online learner correctly induces rules for universe 1 but fails to detect the drift to universe 2 and then to universe 3, the classification rate will drop first to about 32% and then rise to about 34%. In practice, the learner will usually not induce a completely correct set of rules for universe 1 and this imperfect rule set could achieve classification rates after drift that are higher than that of the XCR, but still far short of the Bayes rate.

3.2 The Learning Algorithm

The CD3 algorithm [7] was selected as the online learner. CD3 uses the decision tree algorithm, ID3, along with post pruning as a base learner. In each episode of learning, CD3 receives in a batch, training examples that have become available since the last episode. These are time-stamped as *new* and added to examples retained from previous episodes, time-stamped as *current*. The time-stamp is added to an example's

description prior to induction by ID3. In effect, CD3 is assessing the relevance of the time-stamp attribute to classification. This is the time-stamp attribute relevance (TSAR) principle [7]. Following induction, rules are extracted from the tree and those which specify the time-stamp value as *current* are deemed to be out of date and purged. Current examples which are covered by a purged rule are purged. Finally, the new examples have their time-stamp changed to *current* for the next learning episode. By this means, CD3 aims to dynamically maintain a base of training examples considered to be valid, that is, they reflect the current oracle. It is an important feature of CD3 that it does remove training examples simply on the basis of age, the argument being that, under concept drift, typically not all rules are subject to drift and so examples covered by un-drifted rules retain their relevance to the learner.

3.3 Example Life-cycle Integration Simulator Environment (ELISE)

Designed to complement examples generated from AutoUniv, ELISE has been developed to load example files into a database and to generate initial time point and additional latency values to include with each example in accordance with the online learner life cycle model in figure 1. Test examples are also loaded into the database.

The system allows for the selection of either constant or random latency types. Random latency offers a further breakdown into latency models generated according to a Normal distribution or negative exponential: a Normal distribution representing latency scenarios where examples may return early or late but will more likely be closer to an expected time, and, negative exponential for the scenario where examples will most likely return soon after classification but still allow for very late example return. When selecting a Normal distribution the user has control over a number of preset variance levels. Overall, random latency is determined from an inputted average latency value.

In addition to the latency periods, the user can specify a regime for examples arriving to be classified. These arrivals can be fixed or random and modeled as above.

ELISE then proceeds to simulate an online environment by handing over batch files of training examples at an inputted time interval, e.g. every 1000 time points (t) to a learner, currently CD3. The learner interface, as well as allowing for learning, also has testing and single classification. Upon every learning cycle, the system will call on a learner to test its most recent classifier with a batch of test examples. The learner then writes out its test results to a file.

ELISE requires all initial training examples to be provided as a representation of the first learner supervised example stage. These first examples are given the time point of zero since they represent historic example collection with unknown times. Additional training examples are added through a domain example file. These examples are given chronological initial ID times starting from one and in intervals of one unit. Additional times are then generated and stored to represent the various latency periods the example incurs through the life cycle.

The user inputs the drift points, i.e. the time points at which the first example of each new drifted universe is experienced. They also enter the time point at the end of each test batch and the total number of test examples that make up a batch.

In all, 21 experiments were scripted to run in ELISE. For each of the drift scenarios in table 4, each latency scenario in table 5 was conducted for both a medium and high latency value, as presented in table 6. The first experiment in each drift scenario, i.e. zero latency experiment, was performed to determine CD3 baseline performance under each drift scenario prior to the addition of latency. This made possible an assessment of the nature of and extent to which latency impacted upon the various drift scenarios. Ten iterations of each of these experiments were performed using different sets of data taken from each drifted universe.

Table 4. Drift scenarios

Drifts	Drift points
0	0
1	4501
2	3001, 6001

Table 5. Latency scenarios

Latency Type	Latency Model
Zero	n/a
Constant	n/a
Random	Normal Distribution
Random	Negative Exponential

Table 6. Standard deviations for random latency

Average Latency	Normal Distribution Standard Deviation	Negative Exponential Standard Deviation
500	96	500
2000	516	2000

Data was supplied as described in table 7. It should be noted that the combined total of examples, 10000, used for learning never changes in the experiments. The effect of increasing the number of drifts reduces the overall time for recovery and therefore represents an increasingly drift-active domain. Also, these initial experiments only consider the impact of latency on domains susceptible to revolutionary drift, i.e. a sudden and immediate change in the rules. This gives a clearer interpretation of the impact of latency. However, it is intended that evolutionary drift, i.e. gradual change in rules, will also be explored in later latency experiments.

Table 7. Example data artificial universe(U) breakdown for each drift scenario.

Category	Training Examples	Domain Examples	Test Examples
No drift	U1 (1000)	U1(9000)	U1(10000)
One drift	U1 (1000)	U1(4500), U2(4500)	U1(10000), Universe 2(10000)
Two drift	U1 (1000)	U1(3000), U2(3000), U3(3000)	U1(10000), U2(10000), U3(10000)

Finally, learning and testing cycles were performed in the ELISE simulation every 500 time points.

4 Analysis of Results

The initial experiments involving no drift performed as expected. With the learner commencing at time point zero, having already conducted an initial learning run of 1000 training examples, it quickly achieved a high classification accuracy bordering on the Bayes rate for the universe, as in figure 2. Under latency, the only difference was in the return of examples after the last true time point of 9001, indicating that the examples were returning later than their true time, i.e. lagging as the result of latency.

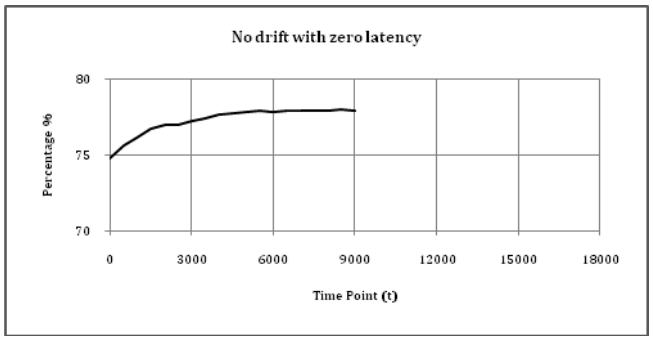

Figure 2 Test result for no drift with zero latency experiment.

The subsequent one drift experiments demonstrate interesting latency rate impacts upon the learner's ability to provide accurate classifiers in time for drift. With zero latency (see figures 3 and 4), the classification rate (CR) crashes at the drift point and then immediately recovers. As seen in comparison to table 8, the lowest accuracy in each of the latency and drift scenarios is comparable to the cross-classification rate (XCR). However, the pattern of recovery is entirely different under latency conditions.

Under the medium latency experiments shown in figure 3, it can be seen that for each of the three latencies a delay is incurred prior to recovery being made. This impact is even more pronounced under high latency.

For constant latency, a low classification plateau occurs for the duration of the example latency, i.e. the nature of constant latency presents as being an overall constant lag behind the current domain. In the high latency scenario, the example latency was for 2000 time points and this matches the duration of the lag prior to recovery. This highlights issues in selecting the time stamp to represent an example in online learners.

While recovery under constant latency appears to be quicker, the Normal distribution performs similarly, although slightly behind. It would appear that the benefit of earlier examples from the new universe is counteracted by the possibility of

old examples from the previous. However, a near Bayes rate classification accuracy is achieved by the end of the last true domain time point of 9001.

Negative exponential latency provides the most interesting results. While initially beginning its recovery quicker than under constant latency, the overall rate of recovery is much slower than the other latency models with a failure to achieve anywhere near the Bayes rate. The overall impact of these patterns becomes even more pronounced and obvious when examined in the two drift experiment results as in figure 4.

In this instance it is seen that even without latency the learner is beginning to struggle by the second drift: failing to achieve a near Bayes rate classification by the last example's true domain time point of 9001. The reduction in time between each drift sees a compounding realisation in the classification rate crash and recovery with the second drift point occurring prior to full recovery.

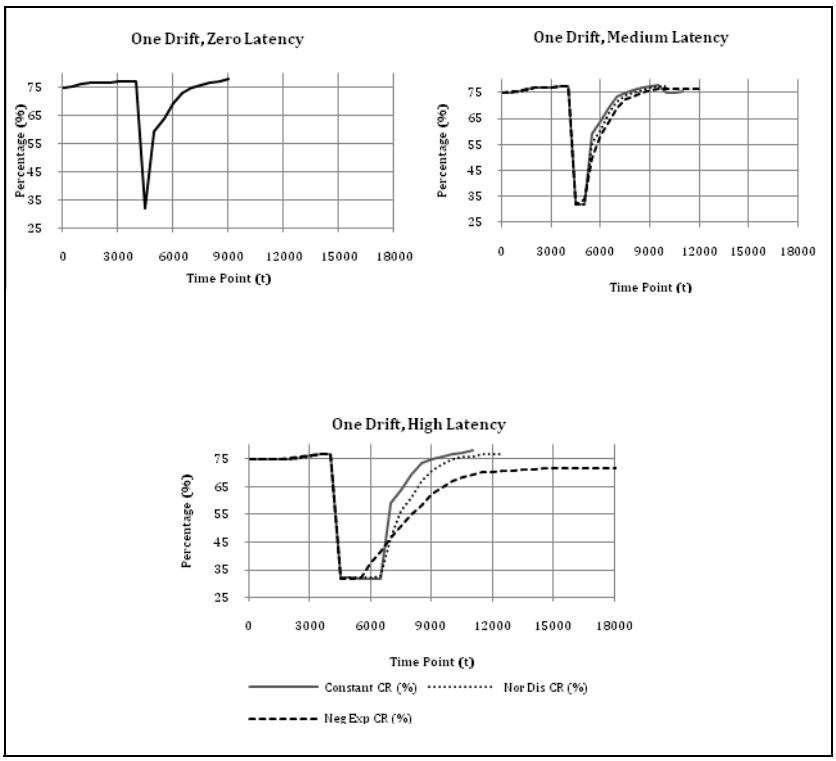

Figure 3 Test results for one drift scenario experiments.

The latency models provide further revelations as to their impact on an online learner under the two drift scenario. While it can be seen that the effect observed under the one drift experiments for both constant and Normal distribution latency is exaggerated further, the extent of damage that a negative exponential latency has upon a learner's classification accuracy becomes more apparent.

The first drift under negative exponential fails to recover in any useful way before the second drift occurs. This results in the second rate of recovery attempt being even lower and stabilising at only 54% classification accuracy around time point 17000 therefore never recovering: 23.9% less than the Bayes rate. In fact, it is only after time point 22001 that the final examples return.

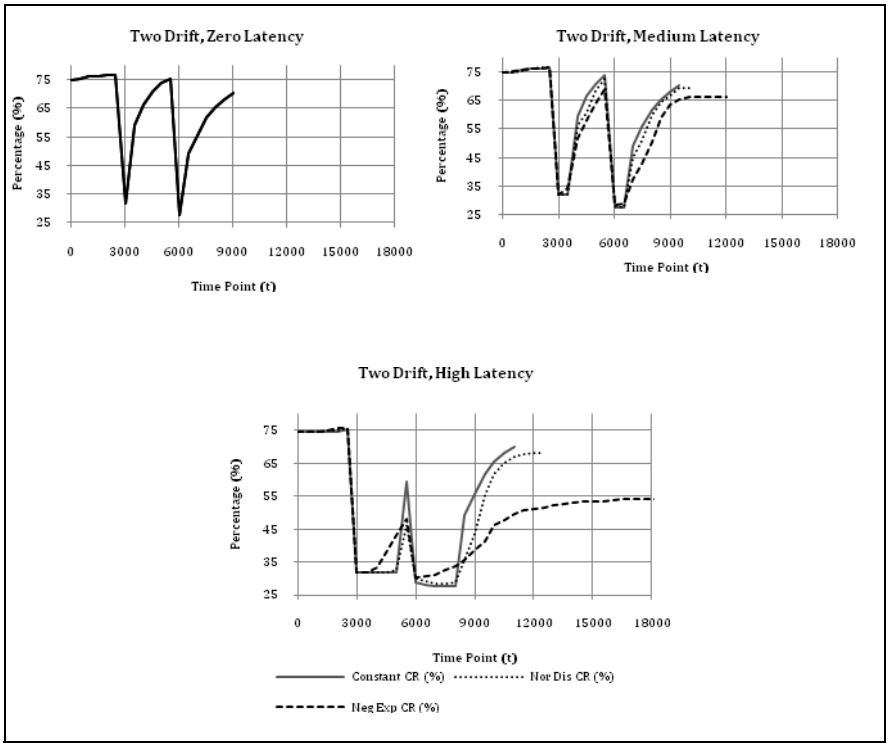

Figure 4 Test results for two drift scenario experiments.

Upon further consideration, the cause of the negative exponential latency's severe impact upon the learner is clear. At the first drift point the learner is still receiving late examples from universe one delaying its recovery. By the second drift point, the learner is still receiving examples from both universe 1 and universe 2 in addition to the new universe. See table 8 where the composition of the first two batches after the drift point are displayed. The contamination caused from the mixing of the universes presents itself to the learner as a new compounded pseudo-universe.

Negative exponential latency in this experiment gives a mixture of examples that are not only contaminated but that also have a majority representation from a previous universe. As a result, it is not possible for the learner to achieve a successful classifier.

Table 8. Example batch composition for two drift, high latency experiment .

	Average Percentage %		
Batch Times (t)	**Universe 1**	**Universe 2**	**Universe 3**
6001 - 6501	16	72	12
6501 - 7001	12.7	55.7	31.6

5 Conclusion and Future Work

Latency in training examples, as defined here, has been shown to have a marked impact on the ability of the online learner CD3 to recover from concept drift. It is reasonable to suppose that other online learners are likely to be similarly affected. Systems in which there is capacity for large latencies such as modelled here by the negative exponential distribution are especially vulnerable. It can therefore be argued that any online learner subject to latency should be tested for accuracy and recovery under these various latency models prior to being deployed.

Further experiments are planned to investigate the performance of an online learner under latency involving different drift scenarios including evolutionary drift as well as under class / attribute - specific example filtering.

Meta attributes such as classification and verification times and latency itself were defined here in an example life cycle for online learning. The resulting meta data can be retained in the training base. The statistical information it provides will enable profiling of the data stream and the training base in addition to directly supporting learning.

As a first step towards equipping an online learner to handle latency, CD3 will be augmented to make use of latency as a meta or context attribute in learning.

References

1. Kolter, J. Z., and Maloof, M. A.: Dynamic Weighted Majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8 (2007) 2755–2790.
2. Minku, L. L., White, A. P. and Yao, X.: The Impact of Diversity on On-line Ensemble Learning in the Presence of Concept Drift., IEEE Transactions on Knowledge and Data Engineering (2009) 730-742.
3. Gao, J., Fan, W. and Han, J.: On appropriate assumptions to mine data streams: Analysis and practice. In: Proc. ICDM' 07, (2007) 143-152.
4. Wang, H., Yin, J., Pei, J., Yu, P. and Yu, J.: Suppressing model over-fitting in mining concept-drifting data streams. In: Proc. KDD 2006, Philadelphia, August 20–23, pp.736–741
5. Sculley, D.: Practical learning from one-sided feed-back. In KDD ’07: Proc. of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (2007).
6. Hickey R. J.: Structure and Majority Classes in Decision Tree Learning, Journal of Machine Learning 8 (2007) 1747-1768.
7. Black, M. and Hickey, R. J.: Maintaining the performance of a learned classifier under concept drift. Intelligent Data Analysis 3 (1999) 453-474.