Modeling the Example Life-Cycle in an Online Classification Learner

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. An online classification system maintained by a learner can be subject to latency and filtering of training examples which can impact on its classification accuracy especially under concept drift. A life-cycle model is developed to provide a framework for studying this problem. Meta data emerges from this model which it is proposed can enhance online learning systems. In particular, the definition of the time-stamp of an example, as currently used in the literature, is shown to be problematic and an alternative is proposed.

Keywords: Classification, Online Learning, Example Life-Cycle, Latency, Filtering, Concept Drift.

1 Introduction

Online learning for classification to date largely involves sourcing data descriptions with classifications for use in creating an initial classifier and for subsequent updates: an excellent introduction to the current state of research has been written by Kolter and Maloof [1]. The current approaches mainly deploy machine learning algorithms for use in determining a concept representation, such as rules, for that environment. In doing so they inherit all the issues and difficulties of machine learning approaches [2]. Even the collecting and preparation of training examples for such algorithms remains an area open to difficulty, both in balancing the data sets [3] and in the actual sources and sourcing of such data [4]. Furthermore, as a participating factor within a concept environment, the classifier may induce a concept drift of its own thus making any classifying success a short-term success, as might be reasonably concluded from Fama's Efficient Market Hypothesis [5], e.g. in email spam filtering the update of a classifier to filter out spam emails immediately results in the spammers altering their behaviour to avoid detection in this latest version.

Concept learning raises so many issues and criticisms; it is desirable that a common understanding and models of the components of concept learning exist to aid in targeting research efforts [6]. In this paper, we explore and present a model of the life cycle such learners might face; a rationale for latency, as previously summarised by Marrs, Hickey and Black [7]; and the impact of example filtering in order that more informed approaches might be developed towards online learning.

2 The Online Learner Life-Cycle (OLLC)

According to Domingos and Hulten [8], the online data-stream learner must incorporate certain design criteria, in particular: “When concept drift is present, the model at any time should be up-to-date but also include all information from the past that has not become outdated.” While this could be viewed as specifically an algorithmic issue, it could also be seen as being deeply influenced by the example life-cycle. The learner may only be as up-to-date as the quality of examples it contains in its training base.

They also raise the question of just how much data is enough to learn an accurate model [9]. Alternatively we might consider how much quality data is required to ensure a good model is produced, i.e. what circumstances could occur during the example life-cycle that may directly impact the usefulness of the next training data and just how might this be improved upon to aid our learner.

2.1 Describing the Online Learner Process

The first stage in the online learning cycle commences with the supervised collection of data to comprise the initial training set. Various sources for example data may be used and measures taken to preserve an accurate representation of the current universe: a concept universe being the collective environment represented by a contemporary true rule-base. For example, for a loan granting system the training samples may consist of historic bank data of customers granted loans and the outcomes of such decisions. The supervising authority selects and pre-processes the training data and forwards it on as the initial learning data. Receiving the training data, the learner induces the first classifier and releases it to make predictions with real world input.

The domain presents a descriptive sample as input for classification, e.g. a description of a loan applicant's current circumstances. The classifier predicts the likely class and releases the now classified sample back into the domain. Dependent upon the nature of the classification, our sample may have its true classification determined at a future date. Some classes may not lend themselves to being track-able, rejection classes [10], e.g. a choice to reject a loan applicant cannot be verified as the loan applicant is no longer with the lending authority. Others will give you a determinable verification of your initial classification, acceptance classes, e.g. at the end of the period given for a successful loan application we should be able to verify the validity of our initial classification and should the applicant fail in paying back the loan prior to the end of the period we will have determined a failed prediction and the true class that the applicant should have been rejected. With the domain sample now verified it might find itself returned at a future date to the training base for use in improving the accuracy of subsequent classifiers. These may be composed of both supervised data and returned verified classification examples.

2.2 Overview of the Stages in the OLLC

The progress of examples through an online learning system can be modeled in a series of stages.

At the first stage, initial supervised example collection takes place using external sources and is placed into the training base (figure 1).

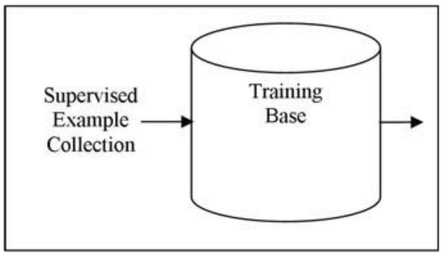

Figure 1: Supervised Example Collection

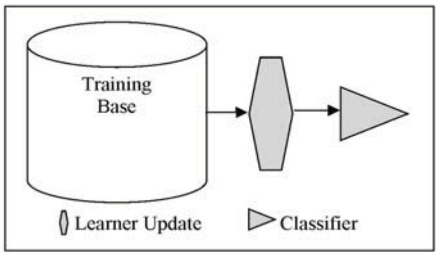

Figure 2: Classifier Induction

Next, the learning algorithm, upon receiving data from the training base, generates the first instance classifier, as in figure 2. This classifier is then placed online for use in the domain. Training data may not necessarily arrive at a constant rate.

At the third stage, a real-world unclassified example, *description*, arrives at time t_c for classification. Once again the rate of arrival is unlikely to be constant. It is given a predicted class , *pclass*, it is sent back out to the domain (figure 3).

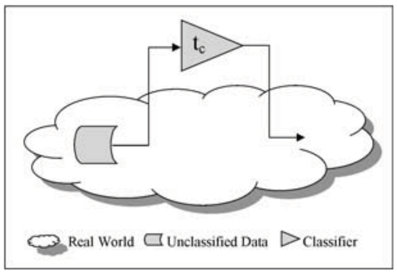

Figure 3: Classification

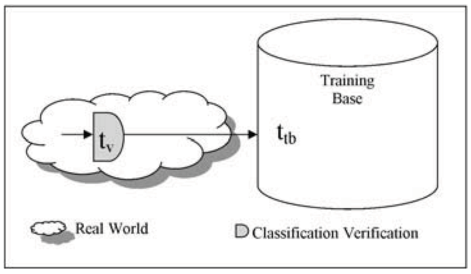

Figure 4: Verification and Feedback

Finally, at time, $t_v > t_c$, the true class, *vclass*, may be discovered from the domain. The lag time, referred to as *verification latency*, between classification and verification can vary from example to example. After a further period of *feedback latency*, the example may be fed back to the training base at time t_{tb} (figure 4).

For some existing online learner approaches a time-stamp may be used [11]. In these learners, the time of arrival at the training base, $t_{tb,}$ is most often used as the timestamp, i.e. examples are time-stamped as they arrive for use, *(t_{tb}, description,* *vclass)* **.** However, we propose that any time-stamp other than that at which the data was first classified, t_c, may give the learner an erroneous representation of the current concept universe. This point is discussed further below.

Upon completion of the verification and feedback stage the cycle repeats with the learner algorithm updating the classifier using newly available training examples. In addition to previously classified examples returning, new examples may be obtained from sources external to the cycle.

2.3 Latency

Concepts can be in a state of drift and change, the act of learning the latest concept universe is a competition against the rate of drift. During any period of latency if the concept universe drifts beyond that represented by our most current data then the data is now in a state of lag. We have determined the following key latency periods:

- External source training example collection
- Training base to learner
- Classification to verification
- Verification to training base

During the external example collecting period, the time taken, latency, might lead to examples being in a state of lagging behind the drift in the current concept universe. By minimizing the latency between the example and its inclusion in the training base the concept drift rate is less likely to render an example irrelevant.

As the example batch is released to the learner, the random latency of the examples risks them being in a state of lag with the current concept universe. The latency of the training base to the learner impacts on the relevance of the learned classifier to the current concept universe.

Within the OLLC, as noted above, an example is subject to verification latency $t_1^{lat} = t_v - t_c$ and feedback latency $t_2^{lat} = t_{tb} - t_v$. In many applications these latencies are random and therefore may bias the availability of training examples in the next round of learning. In a real estate domain, a 25 year mortgage may be granted to several similar clients. For each example granted that fails, we discover the true class at some point prior to the closure of the 25 year period. Sending failed examples back to the training base immediately at the time of failure will result in a disproportionate number of negative cases available for the next round of learning. In this domain it could be argued that examples corresponding to failed mortgages should be withheld from the training base until the end of the 25 year loan period and, at that point, returned with the successful loan repayment cases. Unfortunately, the lag factor for the return of these training examples against the rate of concept drift, could ultimately eliminate any true value.

2.4 Filtering

During the life cycle some examples may be filtered out and so never return to the training base. The incidence of such filtering may depend on the example description and on class. Thus there is the potential for bias in the training base: examples returning may not be a genuine random sample. This would present a problem for the learner. In effect such bias could be perceived as false concept drift. Two types of filtering occur during the OLLC: *trackability* and *selection*.

Trackability filtering is an example of unsupervised filtering. It is first necessary to group classification values under two headings: trackable classes and non-trackable classes: sometimes referred to as positive and negative classes [10].

In a bank loan granting domain the classifier might be required to predict whether to grant a loan application or reject a loan application. If the loan is granted then the example remains trackable and therefore verifiable, i.e. the bank will be able to monitor the application's progress over the loan period. If the prediction proves to be wrong then the bank will be able to determine this fact and use to update the training base for the next learning cycle with a new domain example.

If loan is rejected, however, the applicant returns to the real world again and is no longer able to be tracked by the bank. In this instance the bank will never be able to verify the prediction. In this case, the example is said to be non-trackable.

Therefore, there is a natural tendency for some examples to filter themselves out. Only certain trackable examples are likely to return to the training base for future updates leading to a bias in the next classifier concept representation. In this example, where the trackability factor is dependent upon class value, we could say that an example with a grant loan value has a probability of 1 to remain trackable and 0 to be non-trackable, whereas, a reject loan value has a probability of 0 to be trackable and 1 to be non-trackable.

Other domains may be less exact in the probability values of an example's trackability. Cancer diagnosis may have a trackable probability of 1 for those who are diagnosed as having cancer and 0 probability of being non-trackable. However, those who are diagnosed as not having cancer may not have entirely clear trackability probabilities. If there is a 0.1 probability that a healthy diagnosis is wrong then there remains a 0.1 chance that the example will return and become trackable and a 0.9 probability that it becomes non-trackable.

Selection filtering is a managed filtering process where examples may be selected for verification. The example's true class may not become obvious without some supervised examination of the classifications, e.g. spam filtering. An email may be accepted as being non-spam by an online spam detection software system. It would require managed interrogation of these examples in order to determine whether they were classified correctly. It is highly unlikely that every example would be interrogated in a large system and therefore a selection process is likely to occur. The practices undertaken for selecting examples for further verification could establish a bias leading to an inaccurate concept representation. The rest of the examples are not verified then these are never returned to the training base for subsequent learning.

Examples may be verified incorrectly, e.g. human error, or deliberately bogus, e.g. malicious attempts to break the classifier [12]. There are two potential types of malicious attempts to break the classifier. Firstly, we have gaming the classifier, i.e. disguising the data so as to defeat the current classifier. The second approach is to game the learner, i.e. introducing enough malicious examples into the training base that the learner updates the classifier with a bogus concept representation. Externally sourced examples remain a substantial problem. There may be a lack of control and knowledge of what filtering bias has occurred prior to their arrival.

Ideally, training data is a random sample of the world. However, as a result of the filtering processes this may not be a possibility since certain example types may be less likely to become available. Rare classes may not be represented at all. For many domains it is the rare classes that are the most important target for learning, e.g. credit card fraud detection. In a credit card system, the most important concept class to capture is the fraudulent transaction. However, the ratio of fraudulent to non-fraudulent transactions ensures that our fraudulent class will have a low representation. This may require steps to improve the representation in the training base through a balancing process, e.g. duplicating existing examples. However it may come at the price of altering the class distribution. The value of such balancing measures is still in question [3].

The now verified examples, whether correctly or incorrectly verified, are now available in the training base for possible inclusion in the next learning cycle.

2.5 The Complete OLLC and the Meta Example

The complete OLLC with latencies and filters as developed above is summarised in Figure 5.

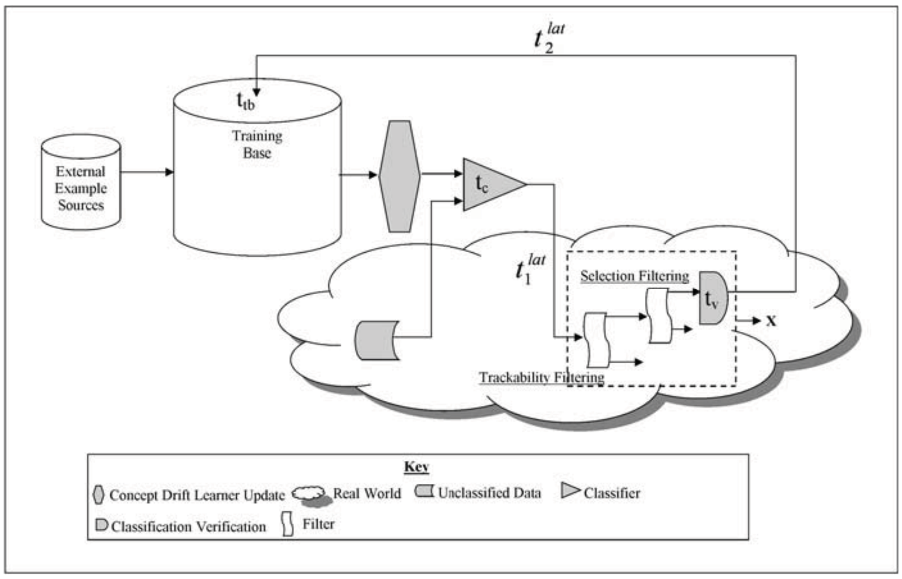

Figure 5: Online Learner Cycle

The traditional training example for a classification learner has, in the notation used here, the form *e = (description, vclass)*. From the OLLC model this can be enhanced with meta attributes to produce the meta example

$$e = (t_c, description, pclass, t_v, vclass, t_{tb})$$

The training base may also receive examples other than those which have been through the life cycle. This can be flagged by an additional meta attribute. It is likely that for external attributes some or all of the meta attribute values will be missing.

4 Discussion

The development of the OLLC was driven by the awareness that latency and filtering as defined above do occur in the real world and can present a serious hazard to an online learner when concept drift occurs.

Wang et al [13] raise issues about online learning. They state that using only historical sources may lead to learning obsolete models whereas using only current data can lead to biased classifiers. The benefit of using internal examples, over external examples, includes the ability to have a greater understanding of the training samples. This is not to say that external examples have no value in the system. Trackability filtering demonstrates how certain classes could end up being poorly represented in the training base causing either up-sampling or down-sampling to occur in the training base. The impact that a highly-skewed class distribution can have upon a learner's performance is clearly demonstrated by Weiss and Provost [14], McCarthy et Al [15], and, Chawla [16].

It has been demonstrated that latency, and the nature of, has a significant impact on the overall performance of an online classifier [7]. The experiments investigated the impact of four latency models, zero, constant, normal distribution-based and negative exponential-based latency, upon an online learner, CD3 [17], using a variety of drift simulations. Proving that latency can hamper both the time taken and overall ability of an online learner to recover accuracy: even to the extent of never recovering.

Hitherto, the only meta attribute routinely used in online learners has been t_{tb}. This has been implicitly assumed to be the appropriate time-stamp of the example. The correct time-stamp, however is t_c . An online learner is attempting to discover the true concept definitions operating a a particular time. A training example is an instance of the true definition and so its time-stamp is that when the rules were applied to produce the example not t_{tb} which is just a result of latencies. Put another way, using t_{tb} as the time-stamp amounts to assuming zero latency.

The value of meta attributes lies in their potential to assist in the maintenance of an online learner when concept drift occurs and there is latency. In the absence of drift, latency will not impact upon a learner.

When latency is random, examples will typically not return in chronological order. Therefore, having an appropriate time-stamp for the example is essential if we are to preserve the grouping of each example within its relevant concept universe.

By including accurate time-stamp, t_c, the internal example gives greater accuracy and understanding of the arrival order of examples and the relationship they have with both current and historic concepts. Current experimentation in progress has already shown that the use of t_c as a meta attribute brings substantial improvements.

Furthermore, by comparing the historic frequency of concept drift sequences against the latency time recorded in the example meta data it is possible to estimate the lag factor occurring in the system, i.e. just how accurate is the classifier produced in representing the current concepts and as such the domain's suitability for host an online learner.

Finally, internal examples and the feedback they provide allow for a post-mortem evaluation of the current classifier performance.

References

1. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8 (2007) 2755-2790
2. Ben-David, A., Frank, E.: Accuracy of Machine Learning Models Versus "hand Crafted" Expert Systems – A Credit Scoring Case Study. Expert Syst. Appl., 36 (2009) 5264-5271
3. Provost, F.: Learning with Imbalanced Data Sets 101. AAAI'2000 Workshop on Imbalanced Data Sets (2000)
4. Gao, J., Fan, W., and Han, J.: On appropriate assumptions to mine data streams: Analysis and practice. In: Proc. ICDM' 07, (2007), 143-152.
5. Fama, E.F. Efficient capital markets: A review of theory and empirical work. Journal of Finance, May (1970), 383–417.
6. Gama, J., Pedersen, R., U.: Predictive Learning in Sensor Networks. In: Learning from Data Streams, pp. 143-164. Springer, Heidelberg (2007)
7. Marrs, G.R., Hickey R.J., Black M.M.: Impact of Latency on Online Classification Learning with Concept Drift, In: Proceedings of 4th International Conference on Knowledge Science, Engineering & Management, LNAI 2010, Springer
8. Domingos, P, Hulten, G.: Catching up with the data: research issues in mining data streams. In: Proc. of Workshop on Research Issues in Data Mining and Knowledge Discovery, (2001)
9. Hickey, R.J. (1996) Noise modelling and evaluating learning from examples, Artificial Intelligence 82: 157–179
10. 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)
11. Hickey, R.J., Black, M.M.: Refined Time Stamps for Concept Drift Detection During Mining for Classification Rules. In: Proceedings of the First International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining, LNCS 2007, Springer-Verlag, London, pp. 20-30. (2001)
12. Sculley D., Cormack G. V.: Filtering spam in the presence of noisy user feedback. Tufts University, 2008.
13. Wang, H., Yin, J., Pei, J., Yu, P., Yu, J.: Suppressing model over-fitting in mining concept-drifting data streams. In: Proc. KDD 2006, Philadelphia, August 20–23, pp.736–741.
14. Weiss, G.M., & Provost, F.: Learning when training data are costly: The effect of class distribution on tree induction. Journal of Artificial Intelligence Research, 19, (2003), 315–354.
15. McCarthy, K., Zabar, B., Weiss G.: Does Cost-sensitive Learning Beat Sampling for Classifying Rare Classes? In: Proceedings of the 1st International Workshop on Utility based Data Mining, ACM Press, New York, NY, USA, pp. 69–77, 2005.
16. Chawla, N.V.: C4.5 and imbalanced data sets: Investigating the effect of sampling method, probabilistic estimate and decision tree structure. In: Workshop on Learning from Imbalanced Datasets II, ICML, Washington DC, 2003.
17. Black, M. and Hickey, R.J.: Maintaining the performance of a learned classifier under concept drift. Intelligent Data Analysis 3 (1999), pp. 453-474.