JMLR: Workshop and Conference Proceedings 17 (2011) 48–55 2nd Workshop on Applications of Pattern Analysis

MOA Concept Drift Active Learning Strategies for Streaming Data

Indrė Žliobaitė IZLIOBAITE@BOURNEMOUTH.AC.UK
Bournemouth University, Poole, UK

Albert Bifet ABIFET@CS.WAIKATO.AC.NZ
Geoff Holmes GEOFF@CS.WAIKATO.AC.NZ
Bernhard Pfahringer BERNHARD@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikato, Hamilton, New Zealand

Editor: Tom Diethe, José L. Balcázar, John Shawe-Taylor, and Cristina Tîrnăucă

Abstract

We present a framework for active learning on evolving data streams, as an extension to the MOA system. In learning to classify streaming data, obtaining the true labels may require major effort and may incur excessive cost. Active learning focuses on learning an accurate model with as few labels as possible. Streaming data poses additional challenges for active learning, since the data distribution may change over time (concept drift) and classifiers need to adapt. Conventional active learning strategies concentrate on querying the most uncertain instances, which are typically concentrated around the decision boundary. If changes do not occur close to the boundary, they will be missed and classifiers will fail to adapt. We propose a software system that implements active learning strategies, extending the MOA framework. This software is released under the GNU GPL license.

1. Introduction

Supervised learning models the relationship between the observed variables of an instance and the target variable (label). To build a predictor we need to know the true labels of the training data. Often unlabeled data is abundant but labeling is expensive. Labels can be costly to obtain due to required human input (labor cost). Consider, for example, textual news arriving as a stream. The goal is to predict if a news item will be interesting to a given user at a given time. The interests of the user may change. To obtain training data the historical news needs to be read and labeled as interesting or not interesting. This requires human labor. For instance, Amazon Mechanical Turk[1] provides a marketplace for intelligent human labeling. Labeling can also be costly due to a required expensive, intrusive or destructive laboratory test. Consider a production process in a chemical plant where the goal is to predict the quality of production output. The relationship between input and output quality might change over time due to constant manual tuning, complementary ingredients or replacement of physical sensors. In order to know the quality of the output (the

1. https://www.mturk.com

true label) a laboratory test needs to be performed which is costly. Under such conditions it may be unreasonable to require true labels for all incoming instances.

Active learning studies how to label selectively instead of asking for all true labels. It has been extensively studied in pool-based (Lewis and Gale, 1994) and online settings (Cohn et al., 1994). In pool-based settings the decision concerning which instances to label is made from all historical data. In this paper we explore active learning in data stream settings, where this decision needs to be made immediately for every incoming instance, as there is no re-access to it. The main difference between online active learning and active learning in data streams is in expectations around changes. Online active learning typically fixes a threshold (e.g. an uncertainty threshold) and asks for the true label if the threshold is exceeded. In data streams the relationship between the input data and the label may change (concept drift) and these changes can happen anywhere in the instance space. Thus, existing active learning strategies may never query instances from some regions and thus may never know that changes are happening and therefore never adapt. Moreover, in data streams we cannot keep the decision threshold or a region of uncertainty fixed, as eventually the system would stop learning and fail to react to changes. Finally, active learning with data streams must preserve the incoming data distribution to the extent that changes could be detected as they happen.

In brief, the framework setting is as follows. Data arrives in a stream, and predictions need to be made in real time. Concept drift is expected, thus learning needs to be adaptive. The true label can be requested immediately or never, as the instances are regularly discarded from memory. Our goal is to maximize prediction accuracy over time, while keeping the labeling costs fixed within an allocated budget. After scanning an instance and outputting the prediction for it, we need a strategy to decide, whether or not to query for the true label so that our model could train itself with this new instance. Regular retraining is needed due to changes in data distribution. Active learning strategies in data streams in addition to being able to learn an accurate classifier in stationary situations, need to be able to

- balance the labeling budget over time;
- notice changes happening anywhere in the instance space;
- preserve the distribution of the incoming data for detecting changes;

To the best of our knowledge this framework is the first to address active learning for instance-incremental streaming data (we preclude methods that learn from a stream in batches) where historical data cannot be stored in memory.

2. Strategies

In this Section we present active learning strategies for data streams. We start with two basic techniques and discuss their drawbacks. Then we introduce two new strategies in two steps, where each step aims to overcome a challenge posed by the data stream setting. We start with a formal definition of our setting.

Algorithm 1: Active Learning Framework

Input: labeling budget B and strategy parameters
for each X_t - *incoming instance,* **do**
  if *ActiveLearningStrategy*$(X_t, B, \ldots)$ = **true then**
    request the true label y_t of instance X_t
    train classifier L with (X_t, y_t)
    if L_n *exists* **then**
      train classifier L_n with (X_t, y_t)
    end
    if *change warning is signaled* **then**
      start a new classifier L_n
    end
    if *change is detected* **then**
      replace classifier L with L_n
    end
  end
end

2.1. Setting

Let X_t be an instance, y_t its true label, where t indicates the time when an instance arrives. $X_1, X_2, \ldots, X_t, \ldots$ is then a data stream. The labeling cost is the same for any instance. We impose a budget B to obtain the true labels, which is expressed as a fraction of the number of incoming instances. $B = 1$ means that all arriving instances are labeled, whereas $B = 0.2$ means that 20% of the arriving instances are labeled.

Algorithm 1 shows our framework, that combines active learning strategies with adaptive learning. In our framework we use the change detection technique of (Gama et al., 2004): when the accuracy of the classifier begins to decrease a new classifier is built and trained with new incoming instances. When a change is detected, the old classifier is replaced by the new one.

2.2. Random strategy

The first (baseline) strategy is naive in the sense that it labels the incoming instances at random instead of actively deciding which label would be more relevant. For every incoming instance the true label is requested with a probability B, where B is the budget. See Algorithm 2 for a formal description.

Algorithm 2: Random(X_t, B)

Input: X_t - incoming instance, B -labeling budget.
Output: *label* $\in \{\textbf{true}, \textbf{false}\}$ indicates whether to request the true label y_t.

generate a uniform random variable $\xi_t \in [0, 1]$
return $\xi_t < B$

2.3. Fixed Uncertainty strategy

Uncertainty sampling is perhaps the simplest and the most common active learning strategy (Settles, 2009). The idea is to label the instances for which the current classifier is the least confident. In an online setting it corresponds to labeling the instances for which the certainty is below some fixed threshold. A simple way to measure uncertainty is to use the posterior probability estimates, output by a classifier. The uncertainty strategy with a fixed threshold is presented in Algorithm 3.

Algorithm 3: FixedUncertainty(X_t, θ, L)

Input: X_t - incoming instance , θ - labeling threshold, L - trained classifier
Output: *label* $\in \{\textbf{true}, \textbf{false}\}$ indicates whether to request the true label y_t.

$\hat{y}_t = \arg\max_y P_L(y|X_t)$, where $y \in \{1, \ldots, c\}$ is one of the class labels.
return $P_L(\hat{y}_t|X_t) < \theta$

2.4. Variable uncertainty strategy

One of the challenges with the uncertainty strategy in a streaming data setting is how to distribute the labeling effort over time. If we use a fixed threshold after some time a classifier would either exhaust its budget or reach the threshold certainty. In both cases it will stop learning and thus fail to adapt to changes.

Instead of labeling the instances that are less certain than the threshold we would like to label the least certain instances within a time interval. Thus we introduce *a variable threshold*, which adjusts itself depending on the incoming data to align with the budget. If a classifier becomes more certain (stable situations), the threshold expands to be able to capture the most uncertain instances. If a change happens and suddenly a lot of labeling requests appear, then the threshold is contracted to query the most uncertain instances first.

It may seem counter intuitive that we are asking for more labels at certain situations and fewer labels at changes. In fact, our dynamic threshold assures that we are asking for the same number of labels in all situations. This is how we balance the budget as we do not know when or how often changes will be happening, so we aim to spend the budget uniformly over time.

The uncertainty strategy with a variable threshold is described in Algorithm 4.

2.5. Uncertainty strategy with randomization

The uncertainty strategy always labels the instances that are close to the decision boundary of the classifier. In data streams changes may happen anywhere in the instance space. When concept drift happens in labels the classifier will not notice it without the true labels. In order not to miss concept drift we would like, from time to time, to label the instances about which the classifier is very certain. For that purpose for every instance we randomize the labeling threshold by multiplying by a normally distributed random variable that follows $\mathcal{N}(1, \delta)$. This way we will label the instances that are close to the decision boundary more often, but occasionally we will also label some distant instances.

Algorithm 4: VariableUncertainty(X_t, L, B, s)

Input: X_t - incoming instance , L trained classifier, B - budget, s - adjusting step.
Output: *label* $\in \{\textbf{true}, \textbf{false}\}$ indicates whether to request the true label y_t.
Starting defaults: total labeling cost $u = 0$, initial labeling threshold $\theta = 1$.

if *($u/t < B$)* **then**
    // budget is not exceeded

    $\hat{y}_t = \arg\max_y P_L(y|X_t)$, where $y \in \{1, \ldots, c\}$ is one of the class labels.
    if *($P_L(\hat{y}_t|X_t) < \theta$)* **then**
        // uncertainty below the threshold

        $u = u + 1$ // labeling costs increase

        $\theta = \theta(1 - s)$ // the threshold decreases

        return true
    else
        // certainty is good

        $\theta = \theta(1 + s)$ // make the uncertainty region wider

        return false
    end
else
    // budget is exceeded

    return false
end

This strategy trades off labeling some very uncertain instances for labeling very certain instances, in order not to miss changes. Thus, in stationary situations this strategy is expected to perform worse than the uncertainty strategy, but in changing situations it is expected to adapt faster. The uncertainty strategy with randomization is presented in Algorithm 5.

Algorithm 5: VariableRandomizedUncertainty(X_t, L, B, s, δ)

Input: X_t - incoming instance , L trained classifier, B - budget, s - adjusting step,δ - variance of the threshold randomization.
Output: $label \in \{\textbf{true}, \textbf{false}\}$ indicates whether to request the true label y_t.
Starting defaults: total labeling cost $u = 0$, initial labeling threshold $\theta = 1$.

if $(u/t < B)$ **then**
    // budget is not exceeded

    $\hat{y}_t = \arg\max_y P_L(y|X_t)$, where $y \in \{1, \ldots, c\}$ is one of the class labels.
    $\theta_{randomized} = \theta \times \eta$, where $\eta \in \mathcal{N}(1, \delta)$ is a random multiplier,
    if $(P_L(\hat{y}_t|X_t) < \theta_{randomized})$ **then**
        // uncertainty below the threshold

        $u = u + 1$ // labeling costs increase

        $\theta = \theta(1 - s)$ // the threshold decreases

        return true
    else
        // certainty is good

        $\theta = \theta(1 + s)$ // make the uncertainty region wider

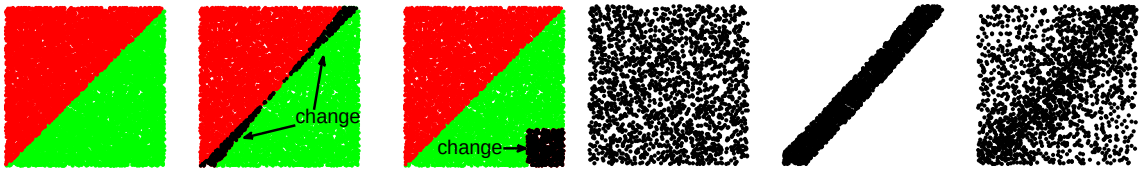
        return false
    end
else
    // budget is exceeded

    return false
end

Table 1 summarizes the four strategies with respect to the requirements indicated in the introduction. The random strategy satisfies all three requirements. Randomized uncertainty satisfies budget and coverage, but it produces biased labeled data. The variable uncertainty satisfies only budget and the fixed uncertainty satisfies none.

Table 1: Summary of strategies.

	Controlling Budget	Instance space Coverage	Labeled Data Distribution
Random	present	full	iid
Fixed uncertainty	no	fragment	biased
Variable uncertainty	handled	fragment	biased
Randomized uncertainty	handled	full	biased



original close change remote change

Figure 1: Data with changes close and far from the decision boundary.

random fixed unc. rand. unc.

Figure 2: 20% of the true labels queried with different labeling strategies.

3. Analysis of how the labeling strategies learn

In this section we explore the main learning aspects of the strategies: the ability to notice changes in dynamic situations and to learn accurate classifiers in stationary situations. In order to demonstrate the behavior of the strategies in controlled settings we employ synthetic data in $2D$. The data is distributed uniformly at random in a square, the distribution $p(X)$ does not change over time, $p(y|X)$ changes. This data represents a binary classification problem. The initial decision boundary is set at $x_1 = x_2$, as illustrated in Figure 1 (left).

Figure 2 shows how the strategies work on the hyperplane problem. The instances that would be labeled by different strategies are visualized. Each strategy labels the same number of instances. The random strategy labels uniformly from the instance space, while the uncertainty strategy concentrates around the decision boundary. The randomized uncertainty infuses randomization into the uncertainty sampling to cover the full instance space.

4. Website, Tutorials, and Documentation

The MOA framework (Bifet et al., 2010) (software available at `http://moa.cs.waikato.ac.nz/`) provides an environment for running experiments in a data stream context.

The website includes a tutorial, an API reference, a user manual, and a manual about mining data streams. Several examples of how the software can be used are available.

An active classifier will encapsulate all the active learning strategies and will allow to have benchmark streaming data experiments through stored, shared, and repeatable settings for synthetic and real data. A proper testing on real or synthetic datasets can be found in (Žliobaitė et al., 2011).

5. Conclusions

We presented an experimental framework for active learning classification on data streams, so that it is easy for researchers to run experimental data stream learning benchmarks on active learning settings. We proposed and implemented active learning strategies for streaming data when changes in the data distribution are expected. Our strategies are equipped with mechanisms to control and distribute the labeling budget over time, to balance the labeling for learning more accurate classifiers and to detect changes. This work can be considered as the first step in active learning in the data stream setting.

Acknowledgements. Part of the research leading to these results has received funding from the EC within the Marie Curie Industry and Academia Partnerships and Pathways (IAPP) programme under grant agreement no. 251617.

References

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering. *JMLR - Proceedings Track*, 11:44–50, 2010.

D. Cohn, l. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, May 1994. ISSN 0885-6125.

J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *SBIA'04*, pages 286–295, 2004.

D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR*, pages 3–12, 1994.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Indrė Žliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. Active learning with evolving streaming data. In *ECML-PKDD '11*, pages 597–612. Springer-Verlag, 2011.