RAL – Improving Stream-Based Active Learning by Reinforcement Learning

Sarah Wassermann[1], Thibaut Cuvelier[2], and Pedro Casas[1]

[1] AIT Austrian Institute of Technology – Vienna, Austria
sarah@wassermann.lu, pedro.casas@ait.ac.at
[2] CentraleSupélec – Gif-sur-Yvette, France; thibaut.cuvelier@supelec.fr

Abstract. One of the main challenges associated with supervised learning under dynamic scenarios is that of periodically getting access to labels of fresh, previously unseen samples. Labeling new data is usually an expensive and cumbersome process, while not all data points are equally valuable. Active learning aims at labeling only the most informative samples to reduce cost. In this paradigm, a learner can choose from which new samples it wants to learn, and can obtain the ground truth by asking an oracle for the corresponding labels. We introduce RAL – Reinforced stream-based Active Learning –, a new active-learning approach, coupling stream-based active learning with reinforcement-learning concepts. In particular, we model active learning as a contextual-bandit problem, in which rewards are based on the usefulness of the system's querying behavior. Empirical evaluations on multiple datasets confirm that RAL outperforms the state of the art, both by improving learning accuracy and by reducing the number of requested labels. As an additional contribution, we release RAL as an open-source Python package to the machine-learning community.

Keywords: Stream-based active learning · Reinforcement learning · Bandits

1 Introduction

A wide range of popular end-user applications such as Skype or Facebook Messenger request users' feedback about their experience at the end of a session. This user feedback is paramount for such services, as they may use the massively collected information to build prediction models shedding light on service performance, user engagement, preferences, etc. A major challenge when querying users is the fact that asking too often is annoying and might have negative consequences on engagement. In a general supervised-learning scenario, labeled data is extremely important, but labeling data is an expensive and tedious task, especially if based on human effort.

Active learning [9] offers a solution to the data exploration and exploitation trade-off, allowing a learner to interactively query the label of only the most informative samples. Active learning is generally used in an offline setup, to limit the number of labels required from a predefined set of unlabeled samples.

The learning tasks we tackle in this paper have two specific characteristics which are not covered by offline active learning: first, the learning data is time-and-size unbounded; second, it comes in the form of an online, non-periodic, sequential stream of samples. An appropriate learning approach should be able to track the evolution of the oracle feedback and the underlying concept drifts. In addition, it should do so by smartly and dynamically deciding at which specific times it is better to query the oracle. This would constantly improve the model-prediction capabilities and avoid unnecessarily querying the oracle.

Stream-based active-learning schemes have been proposed in the past, notably in [16,17]. However, as in traditional active learning, the decision on whether to request for a label or not is solely based on the the model's prediction uncertainty, and not on the informativeness of such a request. *The question that is raised is: can we improve the performance of stream-based active learning by considering how informative the requested label was?* We propose RAL – a novel Reinforced stream-based Active-Learning approach – combining both traditional stream-based active-learning techniques with reinforcement learning. While reinforcement learning has already been used to solve pool-based active-learning problems, the combination with stream-based active-learning algorithms is a mostly unexplored topic. RAL is specifically designed for stream-based, time-unbounded data, and, as we show next, it manages to improve the accuracy of the underlying supervised-learning model, while also significantly reducing the number of requested labels, compared to the state of the art. RAL is available as an open-source Python package[3].

2 Related Work

Many research efforts have already been undertaken in the field of active learning. For example, [16,17] present three simple approaches for this learning paradigm. Their proposed Randomized Variable Uncertainty approach tackles the problem of stream-based active learning using the model's prediction uncertainty to decide whether to query and trying to detect concept drifts by randomizing the certainty threshold used for querying decisions. [15] develops an active-learning algorithm with two different classifiers: one "reactive" and one "stable". The stable classifier is trained on all available labeled instances, while the reactive one learns on a window of recent instances. In [6], the authors present an active-learning technique based on clustering and prediction uncertainty. [7] conceives an approach relying on a modification of the Naïve Bayes classifier to update the different learners through the queried samples. In particular, the author uses one-versus-one classifiers to tackle multi-class problems and update the weights of the different classifiers by comparing their predictions to the ground truth. Krawczyk's technique behaves similarly to ours. However, the major difference is that he is using information about the classifiers' prediction certainty (without considering the corresponding weights) in order to adapt the minimum threshold

[3] `https://github.com/SAWassermann/RAL`

to query the oracle, while we rely on the usefulness of the decisions taken by RAL to tune the system according to the data stream.

Extending active learning through reinforcement learning is currently a very active research area. Active learning alone can easily converge on a policy of actions that have worked well in the past, but are sub-optimal. Reinforcement learning helps to improve the exploration-exploitation trade-off by letting the learner take risks with an uncertain outcome. However, most proposals do not consider the stream scenario, and operate on the basis of pool-based approaches. In [2,5], authors rely on the multi-armed bandit paradigm. [5] develops ALBL, which uses a modified version of EXP4 [1], a weight-updating rule, to attribute adaptive weights to different learners based on rewards; the learner to use is then determined through these weights; the chosen learner selects the samples in the pool to hand to the oracle based on its uncertainty measure. The approach described in [2] is similar to the one in [5], except for the reward-computation scheme. Some other papers in the field of pool-based active learning are [20,22]. The algorithm presented in [10,11] relies on the same principles as the system we are proposing, but tackles a different problem: Song's goal is to introduce an active-learning component into a contextual-bandit problem, while we are aiming at solving an active-learning problem by using contextual bandits.

Other recent papers dealing with active learning and reinforcement learning include [12,18,21,23]. However, most of them consider only one of the perspectives addressed by RAL, namely the enhancement of pool-based active learning through reinforcement learning [18,21,23], or the application of active learning to the streaming setup [12]. Combining active learning with reinforcement learning in a streaming, adaptive learning context is the most important contribution of RAL, a very timely yet vaguely addressed problem in the literature. [19,24] use the idea of *learning to active learn*, i.e. data-driven active learning. [19] proposes this view on pool-based active learning: the querying decision for a sample is based on an estimation of the accuracy improvement. [24] uses reinforcement learning in stream-based active one-shot learning, but this work is different from RAL on multiple aspects: (*i*) it tackles a different learning task, as it aims at detecting new classes instead of improving overall classification accuracy, (*ii*) their scheme relies on reinforcement learning only during the training phase and not once deployed, while RAL continuously adapts its querying policy during the whole incoming stream, and (*iii*) the system heavily relies on deep recurrent networks, too cumbersome to use in real-time resource-constrained scenarios, unlike RAL.

3 Introducing RAL

RAL relies on prediction uncertainty and reinforcement-learning principles, using rewards and bandit algorithms. The overall idea is summarized in Figure 1.

The intuition behind the different reward values is that we attribute a positive reward in case our system behaves as expected, and a negative one otherwise, to penalize it. RAL obtains rewards/penalties only when it is asking for ground truth. In a nutshell, it earns a positive reward ρ^+ in case it queries the oracle

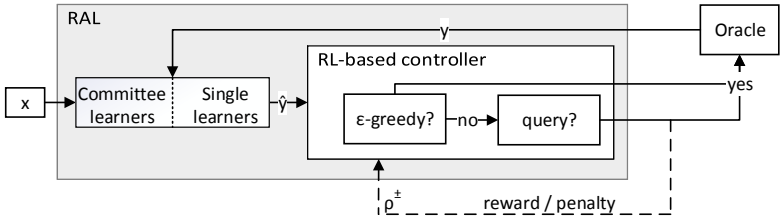


Fig. 1. Overall idea of the RAL system.

and would have predicted the wrong label otherwise (the system made the right decision to ask for the ground truth) and a penalty ρ^- (i.e. a negative reward) when it asks the oracle even though the underlying classification model would have predicted the correct label (querying was unnecessary). The rationale for using reinforcement learning is that RAL learns not only based on the queried samples themselves, but also from the usefulness of its decisions. The objective function to maximize is the total reward $\sum_{i=1}^{n} r_n$, where r_n is the nth reward (either ρ^+ or ρ^-) obtained by RAL.

The conceived system additionally makes use of the prediction certainty of the classification models. It is defined as the highest posterior classification probability among all possible labels for sample x. More formally, the certainty of a model is equal to $\max_{\hat{y}} P(\hat{y}|x)$ with $\hat{y}$ being one of all the possible labels for x.

The underlying assumption for designing RAL is based on the rationale that the model's uncertainty is an appropriate proxy for assessing the usefulness of a data point. Indeed, if the learner is uncertain about its prediction, this sample likely represents a region which has not been explored enough. Adding that sample with its true label to the training set would improve the overall prediction accuracy; the alternative is that the uncertainty is due to noise or to concept drift, these two points being especially challenging in a streaming setting. In RAL, we combine the aforementioned reward mechanism with the model's uncertainty to tune the sample-informativeness heuristic to better guide the query decisions.

Additionally, we implement an ε-greedy policy (also inspired by the bandit literature [14]) for the sake of data-space exploration: with probability ε, the system queries the oracle, even if the committee agreed not to query; we call this the *ε-scenario*. This ensures that we have a good chance of detecting potential concept drifts: without this policy, the system could end up being too confident about its predictions (and thus never ask the oracle again) even though its estimations are erroneous.

In the next two sections, we provide details about the single-classifier and the committee versions of RAL. We first devise the committee version, of which the single-classifier alternative is a simple adaptation.

3.1 Learning with a Committee of Learners

The algorithm behind RAL is summarized in Algorithm 1. Our approach is inspired by contextual bandits [1]. We rely on a set of experts (i.e. different

Algorithm 1 RAL algorithm.

procedure RAL(x, E, α, θ, ε, η)
    x: sample to consider
    E: set of learners, members of the committee
    α: vector of decision powers of learners in E
    θ: certainty/querying threshold
    ε: threshold for ε-greedy
    η: learning rate for updating decision powers and the querying threshold
    decisions $\leftarrow \{\}$ ▷ will contain decisions of learners
    for e $\in$ E **do**
        decisions[e] $\leftarrow$ e.askCertainty(x) $< \theta$ ▷ decisions[e] $\in \{0, 1\}$
    committeeDecision $\leftarrow \text{round}\left(\sum_{e \in E} \alpha[e] \cdot \text{decisions}[e]\right)$
    $p \leftarrow \mathcal{U}_{[0,1]}$ ▷ random number drawn from a uniform distribution
    if $p < \varepsilon$ **or** committeeDecision $= 1$ **then** ▷ ε-scenario or not?
        $y \leftarrow$ acquireLabel(x)
    if committeeDecision $= 1$ **then**
        $r \leftarrow$ getReward(x, y)
        $\alpha \leftarrow$ updateDecisionPowers(r, E, decisions, committeeDecision, α, η)
        $\theta \leftarrow \min\left\{\theta\left(1 + \eta \times \left(1 - 2^{\frac{r}{\rho^-}}\right)\right), \quad 1\right\}$

function updateDecisionPowers(r, E, decisions, committeeDecision, α, η)
    for e $\in$ E **do**
        if decisions[e] $=$ committeeDecision **then**
            $\alpha[e] \leftarrow \alpha[e] \times \exp(\eta \times r)$ ▷ EXP4
    $\alpha \leftarrow \alpha / \sum_{e \in E} \alpha[e]$ ▷ normalize each value of α
    return α

function getReward(x, y)
    return (ρ^- **if** $\hat{y}(x) = y$ **else** ρ^+)

machine-learning models), referred to as a *committee*. Each expert gives its advice for the sample to consider: should the system ask the oracle for feedback or is the expert confident enough about its prediction? To assess a model's prediction certainty, we rely on a certainty threshold θ: if the model's certainty is below θ, the model is too uncertain about the prediction to make and thus it advises that RAL asks the ground truth. The query decision of the committee takes into account the opinions of the experts, but also their decision power: if the weighted majority of the experts votes for not querying, RAL will rely on the label prediction provided by the committee, used in the form of a voting classifier. The decision power of each expert gets updated such that the experts which agree with the entire committee are obtaining more power in case that particular decision is rewarding, i.e. informative (otherwise, these experts get penalized). These weights are updated through the EXP4 rule [1], with a learning rate η. RAL does not update the decision powers of the different learners in the ε-scenario: the committee did not take the querying decision and thus the weights of the models should not be impacted by this querying action.

The computation of the reward is carried out every time the committee decided to query (i.e. not in the ε-scenario). RAL therefore gets rewarded with ρ^+ when it queried the oracle and asking was rewarding (i.e. the voting classifier would have predicted the wrong label). Conversely, RAL is penalized with ρ^- if the system used the oracle because the committee decided to do so, even though the underlying classifier would have predicted the correct class.

As an additional step, to ensure that RAL adapts as best as possible to the data stream, we do not only tune the weights of the committee members based on rewards, but also the uncertainty threshold θ, denoted in the remainder of this section as θ_n to stress that it is influenced by the $n-1$ samples observed so far. Again, as for the decision powers, θ_n is not updated in the ε-scenario.

The update rule of θ_n we implemented for our tool is written as follows:

$$\theta_n \leftarrow \min\left\{\theta_{n-1} \times \left(1 + \eta \times \left(1 - 2^{\frac{r_n}{\rho^-}}\right)\right), \quad 1\right\} \tag{1}$$

Design of update rule. In this subsection, we detail the reasoning behind our choice of the update policy. We are looking for a rule of this form:

$$\theta_n \leftarrow \min\left\{\theta_{n-1}\ (1 + f(r_n)), \quad 1\right\}, \tag{2}$$

where $f(r_n) = 1 - \exp(a \times r_n)$. The design goals of this update policy are that the threshold increases slightly when the reward is positive, conversely when the reward is negative. Our update policy should satisfy the following properties:

1. θ_n **should decrease *rapidly* in case r_n is negative**, as this indicates that the system queries too often and thus is performing poorly. Therefore, θ_n should be adapted fast to improve its performance.
2. θ_n **should *slightly* increase when r_n is positive**, so that the system does not keep decreasing the threshold. The model was right to ask for more samples, and thus the threshold should be increased. Nevertheless, the system is doing well: the threshold should not be too reactive to the queries.
3. f **must have two extrema**: a minimum at $\rho^- < 0$ and a maximum at $\rho^+ > 0$.
4. θ_n **represents a probability**. $\theta_n = 0$ is not acceptable due to the product form of the update policy, thus the values of θ_n must be in the interval $(0, 1]$.
5. $f(r_n)$ **must be in the interval** $(-1, 1]$ to ensure that θ_n takes values corresponding to a probability. We exclude -1 from the allowed range of values to avoid that θ_n drops to 0.

The Properties 1 and 2 lead us to choose the family of functions $f_a : r \mapsto 1 - \exp(a \times r)$ parameterized by a. Property 5 can be translated into an equation to determine this parameter:

$$\lim_{r \to \rho^-} f(r) = 1 - \exp\left(a \times \rho^-\right) = -1 \tag{3}$$

After solving Equation 3, we get $a = \frac{\ln 2}{\rho^-}$. As f is strictly increasing, and because a is nonpositive, f will have a maximum when $r_n = \rho^+$. To satisfy Property 5, ρ^+ must be chosen such that $f(\rho^+) \leq 1$.

As a final step, we introduce an additional hyperparameter to the update rule, namely the learning rate η. This rate aims at smoothing the evolution of the threshold θ_n, i.e. avoiding that θ_n changes too dramatically with a single query. We thus have the following update rule:

$$\theta_n \leftarrow \min\left\{\theta_{n-1}\left(1+\eta\times\left(1-2^{\frac{r_n}{\rho^-}}\right)\right),\quad 1\right\} \tag{4}$$

The values of η are restricted to the range $(0,1)$. Indeed, we still must satisfy Property 5 (a value of 1 would violate this one) and $\eta = 0$ would lead to an nonreactive system, as the threshold would never adapt. Note that this version of RAL uses the same value of η for updating both θ_n and the decision powers in α.

Choice of hyperparameters. We acknowledge that RAL includes a non-negligible number of hyperparameters which should be well chosen in order to obtain the best results. While we do not have any rule of thumb on how to define exact values, the following guidelines help RAL learn from the streaming data:

- the initial value of θ should be high when the number of possible labels is low, to avoid that the model is always too certain about its prediction for the encountered samples
- ε should be higher when dealing with more dynamic datasets, to increase the probability to accurately grasp concept drifts; in general, we would advise using values in the range of 1 to 5%
- η should be small to avoid changing the decision powers of the different learners (α) and θ_n too abruptly; we would advise values below 0.1
- there is no specific range of values for $\rho^{\pm}$ which works better than others and these values should be picked considering the situation in which RAL is used: if unnecessary queries are a major issue, one should set ρ^- such that its absolute value is much higher than the one of ρ^+

3.2 Learning with a Single Classifier

RAL can also be used with a single classifier instead of a committee of learners. In that case, RAL becomes very lightweight and the only element of the system that allows it to efficiently adapt to and learn from the data stream is the variation of the uncertainty threshold θ by relying on the rewards r_n.

4 Expected Reinforcement Reward Analysis

As the main novelty of RAL lies in the introduction of a reinforcement learning loop to improve querying effectiveness and the data exploration-exploitation trade-off, we devote this section to the study of the reward properties in RAL. We rely on concepts from the bandit theory to understand its expected behavior. In the general case of a multi-class classification problem, under the assumptions

that $\rho^+ \pm \rho^- \geq 0$, we prove the following bounds for RAL, $f_n(\alpha, \gamma)$ being a nonnegative function described next:

$$T\left(\rho^+ - \rho^-\right)\left[\alpha\, f_n(\alpha,\gamma) - 1\right] \leq \mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\} \leq T\left(\rho^+ + \rho^-\right) + T\left(\rho^+ - \rho^-\right)\left[(1-\alpha)\, f_n(\alpha,\gamma) + \alpha\right] \quad (5)$$

We show in the experimental results (Section 5) that the cumulative reward is always positive for a very dynamic dataset, pointing to the good performance and added benefits of RAL.

4.1 Expected Reward Analysis – Single Classifier

Let us analyze the expected total reward obtained by using RAL, i.e. $\mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\}$, where T denotes the number of samples in the considered data stream and r_n indicates the reward obtained for the nth sample.

In the following developments, we use these notations:

- $\hat{y}_n$ – nth predicted value
- $\hat{p}_n$ – certainty of the model for the nth prediction
- $\rho^{\pm}$ – reward and penalty obtained by RAL respectively; the reward must be nonnegative and the penalty nonpositive
- $\mathcal{VC}$ – VC dimension [13] of the learner
- θ_n – uncertainty threshold before having observed the nth sample
- err_n – error rate of our classifier before having observed the nth sample
- $\overline{err}_n$ – training error of model before having observed the nth sample

The expected total reward writes $\mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\} = \sum_{n=1}^{T} \mathbb{E}\{r_n\}$.

Based on the classical result of [13], we have the following bound:

$$f_n(\alpha) = \overline{err}_n + \sqrt{\frac{1}{N_n}\left[\mathcal{VC}\left(\log\frac{2N_n}{\mathcal{VC}} + 1\right) - \log\frac{\alpha}{4}\right]} \quad (6)$$

$$\mathbb{P}\left(err_n \leq f_n(\alpha)\right) = 1 - \alpha \quad (7)$$

where N_n denotes the training-set size for the underlying classifier at the nth round and α is a confidence level whose value lies in the interval $[0, 1]$. We therefore can write this probabilistic bound as:

$$\mathbb{P}\Big[\mathbb{P}(\hat{y}_n \neq y_n) \leq f_n(\alpha)\Big] = 1 - \alpha \quad (8)$$

This means that the probability of making a mistake can be written as:

$$\begin{aligned}\mathbb{P}\left[\hat{y}_n \neq y_n\right] &= \mathbb{P}\left[\hat{y}_n \neq y_n | \mathbb{P}(\hat{y}_n \neq y_n) \leq f_n(\alpha)\right] \times \mathbb{P}\left[\mathbb{P}(\hat{y}_n \neq y_n) \leq f_n(\alpha)\right] \\ &+ \mathbb{P}\left[\hat{y}_n \neq y_n | \mathbb{P}(\hat{y}_n \neq y_n) > f_n(\alpha)\right] \times \mathbb{P}\left[\mathbb{P}(\hat{y}_n \neq y_n) > f_n(\alpha)\right] \\ &= \mathbb{P}\left[\hat{y}_n \neq y_n | \mathbb{P}(\hat{y}_n \neq y_n) \leq f_n(\alpha)\right](1-\alpha) + \mathbb{P}\left[\hat{y}_n \neq y_n | \mathbb{P}(\hat{y}_n \neq y_n) > f_n(\alpha)\right]\alpha\end{aligned} \quad (9)$$

Its upper and lower bounds are thus:

$$0 + \alpha\, f_n(\alpha) \leq \mathbb{P}[\hat{y}_n \neq y_n] \leq (1-\alpha)\ f_n(\alpha) + \alpha \times 1 \tag{10}$$

For the next proofs, we will require bounds on the probability of the certainty of the model being less than a threshold. Unfortunately, to the best of our knowledge, no generic result exists for the probability distribution of these certainties, which leads to very lose bounds:

$$0 \leq \mathbb{P}[\hat{p}_n \leq \theta_n] \leq 1 \tag{11}$$

For the following steps, we rely on classical results in probability theory, namely the union bound and Fréchet's inequality. For two probabilistic events A and B, be they independent or not, the following bounds hold:

$$\mathbb{P}(A \wedge B) \leq \mathbb{P}(A) + \mathbb{P}(B) \tag{12}$$

$$\mathbb{P}(A \wedge B) \geq \max\{0, \quad \mathbb{P}(A) + \mathbb{P}(B) - 1\} \tag{13}$$

We have that $\mathbb{E}\{r_n\} = \sum_{r \in \mathcal{R}} r \times \mathbb{P}(r_n = r)$ with $\mathcal{R} = \{\rho^+, \rho^-\}$ being the set of all possible reward values. As RAL does not obtain any reward in the ε-scenario, it can be ignored. Therefore, we have the following decomposition of the expectation and a generic upper bound:

$$\begin{aligned}
\mathbb{E}\{r_n\} &= \underbrace{\rho^+}_{\geq 0}\ \underbrace{\mathbb{P}[\hat{p}_n \leq \theta_n \wedge \hat{y}_n \neq y_n]}_{\leq (\mathbb{P}[\hat{p}_n \leq \theta_n] + \mathbb{P}[\hat{y}_n \neq y_n])} + \underbrace{\rho^-}_{\leq 0}\ \underbrace{\mathbb{P}[\hat{p}_n \leq \theta_n \wedge \hat{y}_n = y_n]}_{\geq (\mathbb{P}[\hat{p}_n \leq \theta_n] + \mathbb{P}[\hat{y}_n = y_n] - 1)} \\
&\leq \mathbb{P}[\hat{p}_n \leq \theta_n]\ \left(\rho^+ + \rho^-\right) + \mathbb{P}[\hat{y}_n \neq y_n]\ \rho^+ + [1 - \mathbb{P}[\hat{y}_n \neq y_n]]\ \rho^- - \rho^- \\
&\qquad \text{by factoring out the probabilities and using the opposite events} \\
&\leq \mathbb{P}[\hat{p}_n \leq \theta_n]\ \left(\rho^+ + \rho^-\right) + \mathbb{P}[\hat{y}_n \neq y_n]\ \left(\rho^+ - \rho^-\right)
\end{aligned} \tag{14}$$

Finally, the upper bound on the expected total reward, under the assumption that both $(\rho^+ + \rho^-)$ and $(\rho^+ - \rho^-)$ are nonnegative, is:

$$\mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\} \leq T\left(\rho^+ + \rho^-\right) + T\left(\rho^+ - \rho^-\right)\ [(1-\alpha)\ f_n(\alpha) + \alpha] \tag{15}$$

If these two assumptions do not hold, a similar bound can still be achieved:

- First, suppose that $\rho^+ + \rho^- \geq 0$ and $\rho^+ - \rho^- \leq 0$. In this case, the only solution is to have $\rho^+ = \rho^- = 0$, thus trivially $\mathbb{E}\{\sum_{n=1}^{T} r_n\} = 0$
- Second, suppose that, conversely, $\rho^+ + \rho^- \leq 0$ and $\rho^+ - \rho^- \geq 0$. These assumptions lead to:

$$\mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\} \leq T\left(\rho^+ - \rho^-\right)\ [(1-\alpha)\ f_n(\alpha) + \alpha] \tag{16}$$

- Third, the case where both $\rho^+ + \rho^- \leq 0$ and $\rho^+ - \rho^- \leq 0$ should not be studied further, because that would imply that $\rho^+ \leq 0$, which violates the defined range of allowed values for ρ^+ (in case $\rho^+ = 0$, we must have $\rho^- = 0$)

As a next step, we derive a lower bound of the expected total reward, with a very similar reasoning. First, the expected reward can be decomposed as:

$$\begin{aligned}\mathbb{E}\{r_n\} &= \underbrace{\rho^+}_{\geq 0}\ \underbrace{\mathbb{P}\left[\hat{p}_n \leq \theta_n \wedge \hat{y}_n \neq y_n\right]}_{\geq(\mathbb{P}[\hat{p}_n\leq\theta_n]+\mathbb{P}[\hat{y}_n\neq y_n]-1)} + \underbrace{\rho^-}_{\leq 0}\ \underbrace{\mathbb{P}\left[\hat{p}_n \leq \theta_n \wedge \hat{y}_n = y_n\right]}_{\leq(\mathbb{P}[\hat{p}_n\leq\theta_n]+\mathbb{P}[\hat{y}_n=y_n])} \\ &\geq \mathbb{P}\left[\hat{p}_n \leq \theta_n\right]\left(\rho^+ + \rho^-\right) + \left(\mathbb{P}\left[\hat{y}_n \neq y_n\right] - 1\right)\left(\rho^+ - \rho^-\right) \\ &\text{by factoring out the probabilities and using the opposite events}\end{aligned} \tag{17}$$

Eventually, if $\rho^+ + \rho^- \geq 0$ and $\rho^+ - \rho^- \geq 0$, the expected total reward is at least $T\left(\rho^+ - \rho^-\right)\left[\alpha\, f_n\left(\alpha\right) - 1\right]$. Conversely, if $\rho^+ + \rho^- \leq 0$ and $\rho^+ - \rho^- \geq 0$, the lower bound is $T\left(\rho^+ - \rho^-\right)\left[\alpha\, f_n\left(\alpha\right) - 2\right]$.

4.2 Generalization to the multi-class case

The VC dimension makes no more sense when the classification problem includes multiple classes. There have been several generalizations thereof, for instance the covering number $\mathcal{N}^{(p)}(\gamma/4, \Delta_\gamma\mathcal{G}, 2\,N_n)$ [4], where $\Delta_\gamma\mathcal{G}$ is the set of classification margins obtained by any classifier of the family $\mathcal{G}$ in the known N_n data points (if a margin is larger than γ, it is clipped to γ). $\overline{err}_{\gamma,n}$ is the number of misclassifications, where an element is misclassified if its margin is less than γ. With a margin $\gamma \in \mathbb{R}_0^+$, a real number $\Gamma \in \mathbb{R}_0^+$ $(\gamma \leq \Gamma)$, and the previously defined notations, the following bound on the generalization error holds:

$$f_n\left(\alpha, \gamma\right) = \overline{err}_{\gamma,n} + \frac{1}{N_n} + \sqrt{\frac{2}{N_n}\left[\log\left(2\,\mathcal{N}^{(p)}\Big(\frac{\gamma}{4}, \Delta_\gamma\mathcal{G}, 2\,N_n\Big)\right) - \log\frac{2\,\Gamma}{\alpha\,\gamma}\right]} \tag{18}$$

$$\mathbb{P}\left(err_n \leq f_n\left(\alpha, \gamma\right)\right) = 1 - \alpha \tag{19}$$

Notation is taken directly as defined in [4].

The upper bound of the expected total reward can be computed as in the binary-classification problem:

$$\mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\} \leq T\left(\rho^+ + \rho^-\right) + T\left(\rho^+ - \rho^-\right)\left[(1-\alpha)\ f_n\left(\alpha, \gamma\right) + \alpha\right] \tag{20}$$

Similarly, the lower bound for a multi-class problem can be expressed as:

$$\mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\} \geq T\left(\rho^+ - \rho^-\right)\left[\alpha\, f_n\left(\alpha, \gamma\right) - 1\right] \tag{21}$$

4.3 Committee version

The mathematical developments for the committee version are very similar to the single classifier ones. First of all, the committee is still a classifier, and thus

the same kind of bound applies on the probability of misclassifying. The only difference is that we have to take the VC dimension of the stacked classifiers instead of the one of the single classifier.

RAL asks the oracle for a label (and obtains the corresponding reward) if the weighted average of the decisions encourages it to query. We denote by $d_{i,n}$ the random variable indicating whether the ith classifier decides to query the oracle or not, i.e. whether its certainty $\hat{p}_{i,n}$ for the nth prediction is below the threshold θ_n (in case of querying, $d_{i,n} = 1$; otherwise, $d_{i,n} = 0$). $\alpha_{i,n}$ is the weight of the ith classifier for the nth sample; we have previously imposed that the sum of the weights must be one ($\sum_{i=1}^{C} \alpha_{i,n} = 1$ for each sample n). Thus, RAL asks when:

$$\sum_{i=1}^{C} \alpha_{i,n}\, d_{i,n} \geq \frac{1}{2} \tag{22}$$

For the upper bound, the previous developments still hold:

$$E\left\{r_n\right\} \leq \mathbb{P}\left[\sum_{i=1}^{C} \alpha_{i,n}\, d_{i,n} \geq \frac{1}{2}\right] \left(\rho^{+} + \rho^{-}\right) + \mathbb{P}\left[\hat{y}_n \neq y_n\right] \left(\rho^{+} - \rho^{-}\right) \tag{23}$$

Again, to the best of our knowledge, no generic result exists for a probability distribution of the querying decisions; we therefore have to resort to a very broad bound:

$$0 \leq \mathbb{P}\left[\sum_{i=1}^{C} \alpha_{i,n}\, d_{i,n} \geq \frac{1}{2}\right] \leq 1. \tag{24}$$

Finally, the expected total reward is, if $\rho^{+} + \rho^{-} \geq 0$ and $\rho^{+} - \rho^{-} \geq 0$, at most:

$$\mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\} \leq T\left(\rho^{+} + \rho^{-}\right) + T\left(\rho^{+} - \rho^{-}\right)\left[(1-\alpha)\; f_n\left(\alpha, \gamma\right) + \alpha\right] \tag{25}$$

Similarly, concerning the lower bound, we obtain, for the same reasons, the same lower bound as in the single classifier case. Specifically, if $\rho^{+} \pm \rho^{-} \geq 0$,

$$\mathbb{E}\left\{\sum_{n=1}^{T} r_n\right\} \geq T\left(\rho^{+} - \rho^{-}\right)\left[\alpha f_n\left(\alpha, \gamma\right) - 1\right] \tag{26}$$

4.4 Summary

This theoretical analysis of RAL yields bounds on the expected total reward which could be tightened by stronger results on the probability distribution of $\hat{p}_n$. Nevertheless, our results show that the expected total reward is *significantly higher* than $T\rho^{-}$, whatever the values of $\rho^{\pm}$: RAL usually takes the appropriate decision, and thus mostly queries when necessary. Conversely, the upper bound is always nonnegative.

	MAWI	**Woodcover**
ε	2.5%	5%
η	0.01	0.02
initial θ	0.9	0.9
ρ^+	1	1
ρ^-	-1	-1

Fig. 2. RAL hyperparameters, selected by grid search.

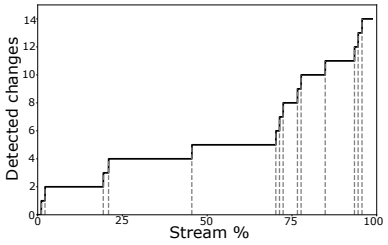


Fig. 3. Concept-drift detection in MAWI. Changes are marked with dashed lines.

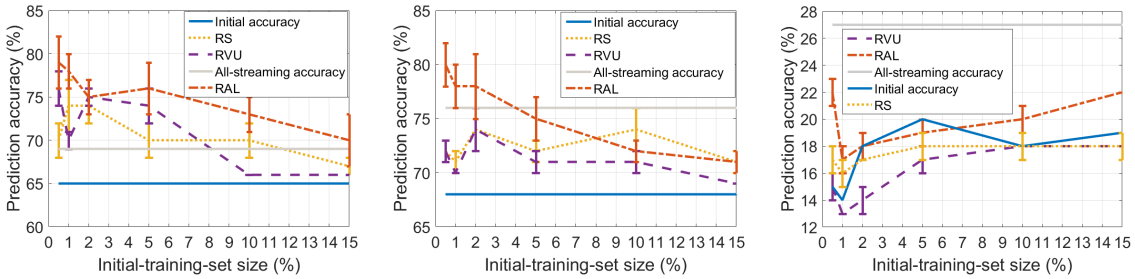
It is more beneficial to choose the rewards such that $\rho^+ + \rho^- \geq 0$. Indeed, in this case, the upper bound is higher (we add a term $T(\rho^+ + \rho^-)$ to the initial bound) as well as the lower bound (this time, we add a term $T(\rho^+ - \rho^-)$). This means that, for a promising behavior of RAL, good decisions should be more rewarded than bad ones are penalized.

By studying special cases of these formulae, we can obtain significant insight into their properties. For instance, if the prediction algorithm is very inaccurate (a situation that is expected for the first samples of the stream) and almost constantly infers the wrong class, i.e. if $\alpha f_n(\alpha, \gamma)$ is close to 1 (equal to $1 - \zeta$), the lower bound becomes $T(\rho^+ - \rho^-)\zeta$. This means that the expected total reward, in this case, is at least zero. This result is significantly stronger than the trivial bound $T\rho^-$: RAL's decisions generate a positive total reward, on average.

5 Evaluation

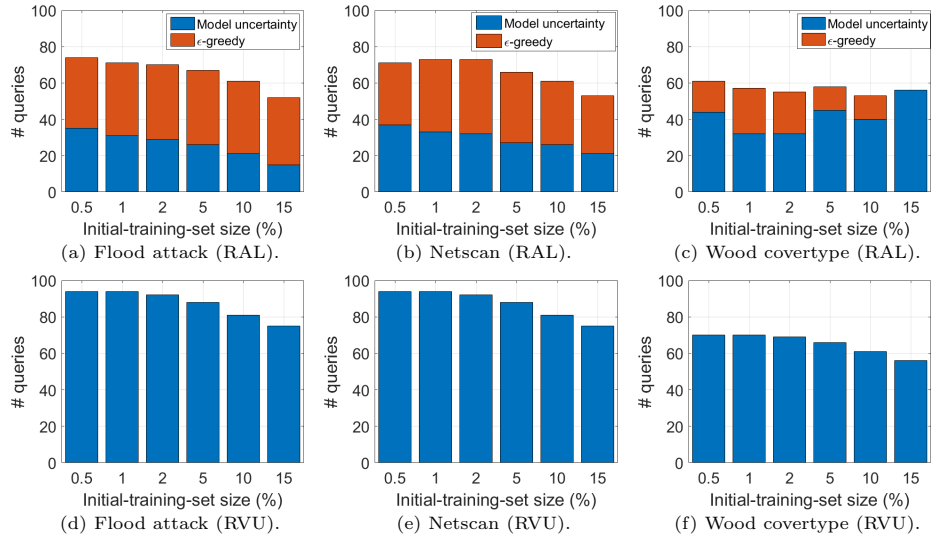
To showcase the performance of RAL, we evaluate our tool and compare it to a state-of-the-art algorithm for stream-based active learning, and to random sampling (RS). In particular, we compare RAL to the Randomized Variable Uncertainty (RVU) technique proposed in [16,17] and mentioned in Section 2, as this approach also heavily relies on the uncertainty of the underlying machine-learning model to take the querying decisions. We use three datasets for the sake of generalization, namely two datasets extracted from MAWILab [3] and a subset of the widely used Forest Covertype data[4]. The covertype dataset contains samples labeled with forest cover type represented by cartographic variables. The dataset provided by MAWILab is publicly available and consists of 15-minute-network-traffic traces captured every day on a backbone link between Japan and the US since 2001 in a stream-based fashion. Traces are annotated with the attack types observed during their corresponding measurement period. In this work, we focus on two attack detections, namely the flood and the UDP-netscan attacks. The MAWI data is subject to concept drifts. We relied on a commonly used statistical test, namely, the Page-Hinkley test [8], for the detection of changes. Figure 3 depicts the cumulative number of drifts observed in the dataset. The test detects 14 abrupt changes during the total measurement time span.

[4] `https://archive.ics.uci.edu/ml/datasets/Covertype`, accessed in April 2019



(a) Flood attack. (b) Netscan. (c) Wood covertype.

Fig. 4. Prediction-accuracy evaluation for RAL, RVU, and RS. For each of the tested datasets, RAL outperforms both techniques.



(a) Flood attack (RAL). (b) Netscan (RAL). (c) Wood covertype (RAL).

(d) Flood attack (RVU). (e) Netscan (RVU). (f) Wood covertype (RVU).

Fig. 5. Number of queries issued by RAL and RVU. RAL asks for significantly fewer samples to reach a better accuracy than RVU.

5.1 Setup

For each benchmarked algorithm, we proceed as follows: first, we subdivide the considered datasets into three consecutive, disjoint parts, i.e. the initial training set, the streaming data, and the validation set. The validation set consists of the last 30% of the dataset, the initial training set is a variable fraction of the first samples (varying between the first 0.5%, 1%, 2%, 5%, 10%, and 15%), and the streaming part includes all the remaining samples not belonging to the other two subsets. We then train a model on the initial training set and check its accuracy (referred to as the *initial accuracy*) on the validation part. Next, we run the benchmarked algorithm on the streaming part and let it pick the samples it wants to learn from. We retrain the models after each queried label. Finally, we

evaluate the final model (i.e. trained on the initial training set and the chosen samples) again on the validation set and report this model's prediction accuracy (referred to as the *final accuracy*). In the context of this evaluation, we implement for both RAL and RVU the budget mechanism presented in [16], based on the ratio between the number of queries and the total number of samples observed so far; the system is allowed to issue queries to the oracle as long as this ratio is below a certain threshold, i.e. the budget. For random sampling, we use a budget indicating the exact number of samples to ask feedback for. For each dataset, we set it to the highest average number of queried samples by either RAL or RVU among all the tests with all the considered initial-training-set sizes. All tests are repeated 10 times, and we report both the average accuracy and its standard error. For RAL, we indicate the average number of queries performed due to the uncertainty of the underlying model and the ones issued through the ε-greedy mechanism. For RVU, we also report the average number of queries. The hyperparameter values of RAL are chosen by grid search for our datasets on the training set within the ranges prescribed in Section 3. The used values are indicated in Figure 2. For RAL and RVU, the budget is set to 0.05 for all the experiments and we test RVU with the hyperparameters recommended in [17].

5.2 Results

The results are shown in Figures 4 and 5. The reported *all-streaming accuracy* refers to the accuracy obtained by the model in case it queries all the samples seen in the stream.

For the evaluations based on the two MAWI datasets, we use the committee version of RAL. More precisely, the committee is a voting classifier composed of a k-NN model with k equal to 5, a decision tree, and a random forest with 10 trees. We also use the same model for RVU and RS. On Figures 4(a) and (b), we can clearly note that RAL outperforms both RVU and RS on average. A striking example are the results for the netscan detection, where RAL obtains final accuracies which are 5 percent points higher than the ones of RVU and RS for the two smallest initial-training-set sizes. It is also worth highlighting that RAL is the only algorithm yielding higher final accuracies than the all-streaming one as long as the initial training set contains less than 5% of the data. To our surprise, RVU is often outperformed by RS. Finally, the flood-attack-detection analysis shows that the three approaches often yield a final accuracy higher than the all-streaming one, underlining that learning from the entire data stream does not necessarily output the best possible accuracy. When it comes to the number of queried samples, we see that RAL queries on average significantly less often than RVU, and a non-negligible part of these queries are due to the model's uncertainty, suggesting that the samples picked by RAL for its learning purposes are wisely chosen.

For the evaluation on the Forest Covertype dataset, we rely on the single-classifier version of RAL, using a 10-tree random forest. As for MAWI, RVU and RS use the same model as RAL. Again, RAL yields better final accuracies than both RVU and RS, even though this prediction task is very challenging.

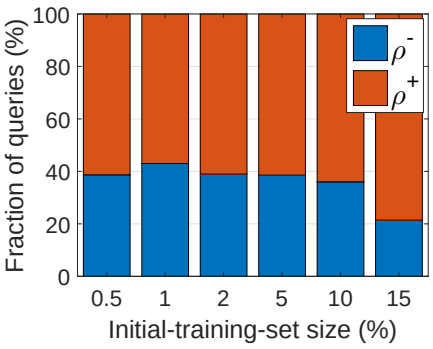


Fig. 6. Proportion of obtained rewards vs. obtained penalties by RAL – wood covertype dataset.

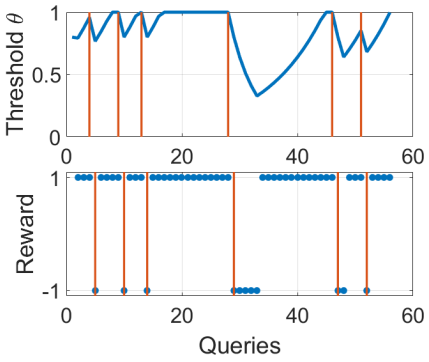


Fig. 7. Evolution of RAL's uncertainty threshold w.r.t. rewards – wood covertype dataset. Red lines indicate penalties.

Moreover, in this study, RVU performs at most as well as RS, but generally worse, showing that the decision-making algorithm of RVU is not always appropriate for complex tasks. Next, we analyze the ratio between the rewards obtained by RAL versus its penalties, i.e. we check whether querying the oracle when uncertain is necessary. Figure 6 shows the outcome of this analysis and it is very encouraging. Indeed, for each test case, the fraction of useful queries is at least 60%. Lastly, we analyze the reactivity of RAL's certainty threshold θ with respect to the obtained rewards and penalties. Figure 7 depicts the evolution of θ with respect to the rewards while RAL is processing the data stream. θ reacts swiftly to penalties, i.e. its value decreases rapidly once RAL gets penalized and, very often, the next query is useful (reward equal to ρ^+), underlining the fact that updating the threshold based on rewards is very relevant.

Note that the initial accuracy is constant for the two different MAWILAB datasets. This is due to the fact that the first 15% of these datasets consist of points with the same label (more precisely, they represent an attack). The first parts of the woodcover dataset are much more dynamic.

6 Conclusions

We have introduced RAL, a novel Reinforced stream-based Active-Learning approach, to tackle challenges of stream-based active learning, i.e. selecting the most valuable sequentially incoming samples, using reinforcement-learning principles. It does not only learn from the data stream, but also from the relevance of its querying decisions. RAL provides a completely different exploration-exploitation trade-off than existing algorithms, as it queries fewer samples of higher relevance. The theoretical analysis underlines the encouraging behavior of RAL. We showed on several datasets that RAL provides promising results, outperforming the state of the art. We make RAL publicly available as an open-source Python package.

References

1. P. Auer et al.: The Nonstochastic Multiarmed Bandit Problem. SIAM Journal on Computing, vol. 32(1), pp. 48–77, 2002
2. Y. Baram et al.: Online Choice of Active Learning Algorithms. Journal of Machine Learning Research (JMLR), vol. 5, pp. 255–291, 2004
3. R. Fontugne et al.: MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. ACM CoNEXT, 2010
4. Y. Guermeur: VC Theory of Large Margin Multi-Category Classifiers. Journal of Machine Learning Research (JMLR), vol. 8, pp. 2551–2594, 2007
5. W.N. Hsu et al.: Active Learning by Learning. AAAI Conference on Artificial Intelligence (AAAI), 2015
6. D. Ienco et al.: Clustering Based Active Learning for Evolving Data Streams. Discovery Science. pp. 79–93, 2013
7. B. Krawczyk: Active and adaptive ensemble learning for online activity recognition from data streams. Knowledge-Based Systems, vol. 138, pp. 69–78, 2017
8. E.S. Page: Continuous Inspection Schemes. Biometrika, vol. 41, pp. 100–115, 1954
9. B. Settles: Active Learning Literature Survey. Tech. rep., University of Wisconsin-Madison, 2010
10. L. Song: Stream-based Online Active Learning in a Contextual Multi-Armed Bandit Framework. arXiv:1607.03182, 2016
11. L. Song et al.: A Contextual Bandit Approach for Stream-Based Active Learning. arXiv:1701.06725, 2017
12. A. Santoro et al.: One-shot Learning with Memory-Augmented Neural Networks. arXiv:1605.06065, 2016
13. V. Vapnik: The Nature of Statistical Learning Theory, 2000
14. C. Watkins: Learning from Delayed Rewards. Ph.D. thesis, King's College, Cambridge, 1989
15. W. Xu et al.: Active Learning Over Evolving Data Streams Using Paired Ensemble Framework. International Conference on Advanced Computational Intelligence (ICACI), 2016
16. I. Žliobaitė et al.: Active Learning with Evolving Streaming Data. Machine Learning and Knowledge Discovery in Databases. pp. 597–612, 2011
17. I. Žliobaitė et al.: Active Learning With Drifting Streaming Data. IEEE Transactions on Neural Networks and Learning Systems, vol. 25(1), pp. 27–39, 2014
18. P. Bachman et al.: Learning Algorithms for Active Learning. International Conference on Machine Learning (ICML), 2017
19. K. Konyushkova et al.: Learning Active Learning from Real and Synthetic Data. Conference on Neural Information Processing Systems (NIPS), 2017
20. G. Contardo et al.: A Meta-Learning Approach to One-Step Active Learning. International Workshop on Automatic Selection, Configuration and Composition of Machine Learning Algorithms, 2017
21. M. Fang et al.: Learning how to Active Learn: A Deep Reinforcement Learning Approach. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2017
22. S. Ravi et al.: Meta-Learning for Batch Mode Active Learning. International Conference on Learning Representations (ICLR), Workshop Track, 2018
23. K. Pang et al.: Meta-Learning Transferable Active Learning Policies by Deep Reinforcement Learning. International Conference on Machine Learning (ICML), AutoML Workshop, 2018
24. M. Woodward et al.: Active One-shot Learning. Conference on Neural Information Processing Systems (NIPS), Deep Reinforcement Learning Workshop, 2016