

---

# Selective Sampling with Drift

---

**Edward Moroshko**

Department of Electrical Engineering,  
The Technion, Haifa, Israel  
edward.moroshko@gmail.com

**Koby Crammer**

Department of Electrical Engineering,  
The Technion, Haifa, Israel  
koby@ee.technion.ac.il

## Abstract

Recently there has been much work on selective sampling, an online active learning setting, in which algorithms work in rounds. On each round an algorithm receives an input and makes a prediction. Then, it can decide whether to query a label, and if so to update its model, otherwise the input is discarded. Most of this work is focused on the stationary case, where it is assumed that there is a fixed target model, and the performance of the algorithm is compared to a fixed model. However, in many real-world applications, such as spam prediction, the best target function may drift over time, or have shifts from time to time. We develop a novel selective sampling algorithm for the drifting setting, analyze it under no assumptions on the mechanism generating the sequence of instances, and derive new mistake bounds that depend on the amount of drift in the problem. Simulations on synthetic and real-world datasets demonstrate the superiority of our algorithms as a selective sampling algorithm in the drifting setting.

## 1 Introduction

We consider the online binary classification task, in which a learning algorithm predicts a binary label given inputs in a sequence of rounds. An example of such task is classification of emails based on their content as spam or not spam. Traditionally, the purpose of a learning algorithm is to make the number of mistakes as small as possible compared to predictions of some *single* function from some class. We call this setting the *stationary* setting.

Following the pioneering work of Rosenblatt [18] many al-

gorithms were proposed for this setting. Some of them are able to employ second-order information. For example, the second-order perceptron algorithm [5] extends the original perceptron algorithm and uses the spectral properties of the data to improve performance. Another example is the AROW algorithm [8] which uses confidence as a second-order information. All these second-order algorithms can be seen as RLS (Regularized Least Squares) based, as their update equations are similar to those of RLS, updating a weight vector and a covariance-like matrix. Under the stationary setting, RLS-based second-order algorithms have been successfully applied to the regression and classification tasks, as shown in Table 1.

Despite the extensive and impressive guarantees that can be made for algorithms in such setting [5, 8], competing with the best *fixed* function is not always good enough. In many real-world applications, the true target function is not fixed, but is slowly changing over time, or switching from time to time. These reasons led to the development of algorithms and accompanying analysis for drifting and shifting settings, which we collectively call the *non-stationary* setting. For online regression, few algorithms were developed for this setting [13, 22, 16]. Yet, for online classification, the Shifting Perceptron algorithm [4] is a first-order algorithm that shrinks the weight vector each iteration, and in this way weakens dependence on the past. The Modified Perceptron algorithm [3] is another first-order algorithm that had been shown to work well in the drifting setting [9]. In this paper we derive a new RLS-based *second-order* algorithm for classification, designed to work with target drift, and thus we fill the missing configuration in Table 1. Our algorithm extends the second-order perceptron algorithm [5], and we provide a performance bound in the mistake bound model.

A practical variant of the fully supervised online classification setting, is where, at each prediction step, the learner can abstain from observing the current label. This setting is called selective sampling [12]. In this setting a learning algorithm actively decides when to query for a label. If the label is queried, then the label value can be used to improve future predictions, and otherwise the algorithm never knows whether his prediction was correct. Roughly

---

Appearing in Proceedings of the 17<sup>th</sup> International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33. Copyright 2014 by the authors.

	Stationary	Non-stationary
<b>Regression</b>	[23, 2, 11]	[16, 22]
<b>Classification</b>	[5, 8]	This work

Table 1: Fully supervised online RLS-based second-order algorithms.

speaking, selective sampling algorithms can be divided in two groups. In the first group, a simple randomized rule is used to turn fully supervised algorithm to selective sampling algorithm. The rule uses the margin of the estimate. This group includes the selective sampling versions of the perceptron and the second-order perceptron algorithms [7]. In the second group, selective sampling algorithms are derived based on comparing the variance of the RLS estimate to some threshold. This group includes the BBQ algorithm [6], where the threshold decays polynomially with  $t$  as  $t^{-\kappa}$ , and more involved variants where the threshold depends on the margin of the RLS estimate [10, 17].

In all previous work on selective sampling the performance of an algorithm is compared to the performance of a *single* linear comparator. To the best of our knowledge, our work is the first instance of learning online in the context of drifting in the selective sampling setting. We build on the work of Cesa-Bianchi et al [7] that combined a randomized rule into the Perceptron algorithm, yielding a selective sampling algorithm. We analyze the resulting algorithm in the drifting setting, and derive a bound on the expected number of mistakes of the algorithm. Thus, we fill the non-stationary cell in Table 2. Simulations on synthetic and real-world datasets show the advantages of our algorithm, in a fully supervised and selective sampling settings.

## 2 Problem setting

We consider the standard online learning model [1, 15] for binary classification, in which learning proceeds in a sequence of rounds  $t = 1, 2, \dots, T$ . In round  $t$  the algorithm observes an instance  $x_t \in \mathbb{R}^d$  and outputs a prediction  $\hat{y}_t \in \{-1, +1\}$  for the label  $y_t$  associated with  $x_t$ . We say that the algorithm has made a prediction mistake if  $\hat{y}_t \neq y_t$ , and denote by  $M_t$  the indicator function of the event  $\hat{y}_t \neq y_t$ . After observing the correct label  $y_t$  the algorithm may update its prediction rule, and then proceeds to the next round. We denote by  $m$  the total number of mistakes over a sequence of  $T$  examples.

The performance of an algorithm is measured by the total number of mistakes it makes on an arbitrary sequence of examples. In the standard performance model, the goal is to bound this total number of mistakes in terms of the performance of the best *fixed* linear classifier  $u \in \mathbb{R}^d$  in hindsight. Since finding  $u \in \mathbb{R}^d$  that minimizes the number of mistakes on a known sequence is a computationally hard problem, the performance of the best predictor in hindsight is often measured using the cumulative hinge loss  $L_{\gamma, T}(u) = \sum_{t=1}^T \ell_{\gamma, t}(u)$ , where  $\ell_{\gamma, t}(u) = \max\{0, \gamma - y_t u^\top x_t\}$  is

Stationary	Non-stationary
[7]	This work

Table 2: Second-order randomized *selective sampling* algorithms for classification.

the hinge loss of the competitor  $u$  on round  $t$  for some margin threshold  $\gamma > 0$ .

In the drifting setting that we consider in this work, the learning algorithm faces the harder goal of bounding its total number of mistakes in terms of the cumulative hinge loss achieved by an arbitrary sequence  $u_1, u_2, \dots, u_T \in \mathbb{R}^d$  of comparison vectors. The cumulative hinge loss of such sequence is  $L_{\gamma, T}(\{u_t\}) = \sum_{t=1}^T \ell_{\gamma, t}(u_t)$ . To make this goal feasible, the bound is allowed to scale also with the norm of  $u_1$  and the total amount of drift defined to be  $V = V(\{u_t\}) = \sum_{t=2}^T \|u_t - u_{t-1}\|^2$ .

We consider two settings: (a) standard supervised online binary classification (described above), and (b) selective sampling. In the later setting, after each prediction the learner may observe the correct label  $y_t$  only by issuing a query. If no query is issued at time  $t$ , then  $y_t$  remains unknown. We represent the algorithm's decision of querying the label at time  $t$  through the value of a Bernoulli random variable  $Z_t$ , and the event of a mistake with the indicator variable  $M_t = 1$ . Note that we measure the performance of the algorithm by the total number of mistakes it makes on a sequence of examples, including the rounds where the true label  $y_t$  remains unknown.

Finally,  $\bar{L}_{\gamma, T}(\{u_t\}) = \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t \ell_{\gamma, t}(u_t) \right]$  is the expected total hinge loss of a competitor on mistaken and queried rounds, and trivially  $\bar{L}_{\gamma, T}(\{u_t\}) \leq L_{\gamma, T}(\{u_t\})$ .

## 3 Algorithms

Online algorithms work in rounds. On round  $t$  the algorithm receives an input  $x_t$  and makes a prediction. We follow Moroshko and Crammer [16] and design the prediction as a last-step min-max problem in the context of drifting. Yet, unlike all previous work, we design algorithms for classification. Specifically, prediction is the solution of the following optimization problem,

$$\hat{y}_T = \arg \min_{\hat{y}_T \in \{-1, +1\}} \max_{y_T \in \{-1, +1\}} \left[ \sum_{t=1}^T (y_t - \hat{y}_t)^2 - \min_{u_1, \dots, u_T} Q_T(u_1, \dots, u_T) \right], \quad (1)$$

where

$$Q_t(u_1, \dots, u_t) = b \|u_1\|^2 + c \sum_{s=1}^{t-1} \|u_{s+1} - u_s\|^2 + \sum_{s=1}^t (y_s - u_s^\top x_s)^2$$

for some positive constants  $b, c^1$ .

This optimization problem can also be seen as a game where the algorithm chooses a prediction label  $\hat{y}_t \in \{-1, +1\}$  to minimize the last-step regret, while an adversary chooses a target label  $y_t \in \{-1, +1\}$  to maximize it. The first term of (1) is the loss suffered by the algorithm while  $Q_t(u_1, \dots, u_t)$  is a sum of the loss suffered by some sequence of linear functions  $\{u_s\}_{s=1}^t$ , a penalty for consecutive pairs that are far from each other, and for the norm of the first to be far from zero.

The following lemma enables to solve (1) by specifying means to solve the inner optimization problem in (1).

**Lemma 1** ([16], Lemma 2). *Denote*

$P_t(u_t) = \min_{u_1, \dots, u_{t-1}} Q_t(u_1, \dots, u_t)$ . Then

$P_t(u_t) = u_t^\top D_t u_t - 2u_t^\top e_t + f_t$  where,

$$\begin{aligned} D_1 &= bI + x_1 x_1^\top & D_t &= (D_{t-1}^{-1} + c^{-1}I)^{-1} + x_t x_t^\top & (2) \\ e_1 &= y_1 x_1 & e_t &= (I + c^{-1}D_{t-1})^{-1} e_{t-1} + y_t x_t & (3) \\ f_1 &= y_1^2 & f_t &= f_{t-1} - e_{t-1}^\top (cI + D_{t-1})^{-1} e_{t-1} + y_t^2, \end{aligned}$$

where,  $D_t \in \mathbb{R}^{d \times d}$  is a PSD matrix,  $e_t \in \mathbb{R}^{d \times 1}$  and  $f_t \in \mathbb{R}$ .

From the lemma we solve,  $\min_{u_1, \dots, u_t} Q_t(u_1, \dots, u_t)$ , by,

$$\min_{u_1, \dots, u_t} Q_t(u_1, \dots, u_t) = \min_{u_t} P_t(u_t) = -e_t^\top D_t^{-1} e_t + f_t. \quad (4)$$

Next, we substitute the value of  $e_T$  from (3) (as a function of  $y_T$ ) in (4), and then substitute (4) in (1). Omitting terms not depending explicitly on  $y_T$  and  $\hat{y}_T$  we get from (1) that,

$$\begin{aligned} \hat{y}_T &= \arg \min_{\hat{y}_T \in \{-1, +1\}} \max_{y_T \in \{-1, +1\}} \left[ (x_T^\top D_T^{-1} x_T) y_T^2 \right. \\ &\quad \left. + 2y_T \left( x_T^\top D_T^{-1} (I + c^{-1}D_{T-1})^{-1} e_{T-1} - \hat{y}_T \right) + \hat{y}_T^2 \right] \\ &= \arg \min_{\hat{y}_T \in \{-1, +1\}} \left[ x_T^\top D_T^{-1} x_T \right. \\ &\quad \left. + 2 \left| x_T^\top D_T^{-1} (I + c^{-1}D_{T-1})^{-1} e_{T-1} - \hat{y}_T \right| + \hat{y}_T^2 \right] \\ &= \text{sign}(\hat{p}_t)|_{t=T}, \end{aligned}$$

where

$$\hat{p}_t = x_t^\top D_t^{-1} (I + c^{-1}D_{t-1})^{-1} e_{t-1}. \quad (5)$$

To the best of our knowledge, this is the first application of the last-step min-max approach directly for classification, and not as a reduction from regression, which is possible by employing the square loss, as in least-squares

<sup>1</sup>We still use the squared loss in (1), as done for least-squares SVMs [21, 20], which allows us to compute all quantities analytically.

SVMs [21, 20]. Indeed, we showed that the optimal prediction for classification is the sign of the optimal prediction for regression [16].

Our algorithm includes the second-order perceptron [5] algorithm as a special case when  $c = \infty$ . The second-order perceptron algorithm is indeed using the sign of the optimal min-max prediction for regression [11], which is in fact the prediction of the AAR algorithm [23] (aka "forward algorithm" [2]). Additionally, similar to other algorithms [18, 5], we update the algorithm only on mistaken rounds. We call the algorithm LASEC for last-step adaptive classifier. LASEC is a special case of Fig. 1 when setting  $a = \infty$  (see below). Note that in the pseudocode two indices are used, the current time  $t$  and the number of examples used to update the model  $k$ . This makes the presentation simpler as some examples are not used to update the model, the ones for which there was no classification mistake. Note, the update equations of Fig. 1 are essentially (2) and (3). The LASEC algorithm can be seen as an extension to the non-stationary setting of the second-order perceptron algorithm [5]. Indeed, for  $c = \infty$  the LASEC algorithm is reduced to the second-order perceptron algorithm.

Next, we turn LASEC from an algorithm that uses the labels of all inputs to one that queries labels stochastically. Specifically, the algorithm uses the margin  $|\hat{p}_t|$  defined in (5) to randomly choose whether to make a prediction. We interpret large values of the margin  $|\hat{p}_t|$  as being confident in the prediction, which should reduce the probability of querying the label. Specifically, the algorithm is querying a label with probability  $a/(a + |\hat{p}_t|)$  for some  $a > 0$ . If  $a \rightarrow \infty$  the algorithm will always query, and reduce to LASEC, while if  $a \rightarrow 0$  it will never query.

This approach for deriving selective-algorithms from margin-based online algorithms is not new, and was used to design an algorithm for the non-drifting case [7]. Yet, unlike other selective sampling algorithms [7, 6, 10, 17], our algorithm is designed to work in the *drifting setting*. Since the algorithm is based on the LASEC algorithm, we call it LASEC-SS, where SS stands for selective sampling. The algorithm is summarized in Fig. 1 as well. Note that LASEC-SS includes other algorithms as special cases. Specifically, LASEC-SS is reduced for  $c = \infty$  to the selective sampling version of the second-order perceptron algorithm [7], and as mentioned above, for  $a = \infty$  it is reduced to LASEC, and the setting of both  $c = \infty, a = \infty$  reduces the algorithm to the second-order perceptron, which in turn reduces to the perceptron algorithm for  $b \rightarrow \infty$ .

The algorithm is flexible enough to be tuned both to drifting or non-drifting setting (using  $c$ ), between selective sampling or supervised learning (using  $a$ ), and between first-order or second-order modeling (using  $b$ ).

Our algorithm can be combined with Mercer kernels as it employs only sums of inner- and outer-products of the in-

**Parameters:**  $0 < b < c$ ,  $0 < a$

**Initialize:** Set  $D_0 = (bc)/(c-b)I \in \mathbb{R}^{d \times d}$ ,  $e_0 = 0 \in \mathbb{R}^d$  and  $k = 1$

**For**  $t = 1, \dots, T$  **do**

- Receive an instance  $x_t \in \mathbb{R}^d$
- Set

$$S_t = (D_{k-1}^{-1} + c^{-1}I)^{-1} + x_t x_t^\top \quad \text{see (2)}$$

$$\hat{p}_t = x_t^\top S_t^{-1} (I + c^{-1}D_{k-1})^{-1} e_{k-1} \quad \text{see (5)}$$

- Output prediction  $\hat{y}_t = \text{sign}(\hat{p}_t)$
- Draw a Bernoulli random variable  $Z_t \in \{0, 1\}$  of parameter  $\frac{a}{a+|\hat{p}_t|}$
- If  $Z_t = 1$  then query label  $y_t \in \{-1, +1\}$  and if  $\hat{y}_t \neq y_t$  then update:

$$e_k = (I + c^{-1}D_{k-1})^{-1} e_{k-1} + y_t x_t \quad \text{see (3)}$$

$$D_k = S_t$$

$$k \leftarrow k + 1$$

Figure 1: LASEC for selective sampling. Set  $a = \infty$  over the supervised setting.

puts. This allows it to build non-linear models (e.g. [19]).

## 4 Analysis

We now prove bounds for the number of mistakes of our algorithm. We provide a mistake bound for the fully supervised version (LASEC) and for the selective sampling version (LASEC-SS). Our bounds depend on the total drift of the reference sequence  $V_m = \sum_{k=2}^m \|u_k - u_{k-1}\|^2$  which is calculated on rounds when the algorithm makes updates. We denote by  $\mathcal{M} \subseteq \{1, 2, \dots\}$  the set of indices when the algorithm updates. For the supervised setting it is the set of mistaken trials ( $M_t = 1$ ). For the selective sampling setting it is the set of indices when  $Z_t = 1$  and  $M_t = 1$ .

**Theorem 2.** *Assume the LASEC algorithm (of Fig. 1 with  $a = \infty$ ) is run on a finite sequence of examples. Then for any reference sequence  $\{u_t\}$  and  $\gamma > 0$  the number  $m = |\mathcal{M}|$  of mistakes satisfies*

$$m \leq \frac{1}{\gamma} L_{\gamma, T}(\{u_t\}) + \frac{1}{\gamma} \sqrt{\left( b \|u_1\|^2 + cV_m + \sum_{k=1}^m (u_k^\top x_k)^2 \right) \sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t} \quad (6)$$

**Remark 3.** *For the stationary case, when  $u_k = u \forall k$  ( $V_m = 0$ ) and we set  $c = \infty$  for the LASEC algorithm we recover the second-order perceptron bound [5].*

**Theorem 4.** *Assume the LASEC-SS algorithm of Fig. 1 is run on a sequence of  $T$  examples with parameter  $a > 0$ . Then for any reference sequence  $\{u_t\}$  and  $\gamma > 0$  the expected number of mistakes satisfies*

$$\mathbb{E} \left[ \sum_{t=1}^T M_t \right] \leq \frac{1}{\gamma} \bar{L}_{\gamma, T}(\{u_t\}) + \frac{a}{2\gamma^2} \left( b \|u_1\|^2 + cV_m + \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right] \right) + \frac{1}{2a} \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t x_t^\top D_t^{-1} x_t \right]. \quad (7)$$

Moreover, the expected number of labels queried by the algorithm equals  $\sum_{t=1}^T \mathbb{E} \left[ \frac{a}{a+|\hat{p}_t|} \right]$ .

**Remark 5.** *As in other context [7]: Thm. 2 is not a special case of Thm. 4. Indeed, setting  $a = \infty$  makes the bound of Thm. 4 unbounded, as opposed to Thm. 2. Also, from the last part of Thm. 4 we observe that more labels would be queried for larger values of  $a$ . However, the tradeoff between number of queries and mistakes is not clear.*

**Remark 6.** *For the stationary case, when  $u_k = u \forall k$  ( $V_m = 0$ ) and we set  $c = \infty$  for the LASEC-SS algorithm we recover the bound of the selective sampling version of the second-order perceptron algorithm [7].*

The bound (7) depends on the parameter  $a$ . If we would know the future, by setting,

$$a = \gamma \sqrt{\frac{\mathbb{E} \left[ \sum_{t=1}^T M_t Z_t x_t^\top D_t^{-1} x_t \right]}{b \|u_1\|^2 + cV_m + \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right]}}$$

we would minimize the bound and get

$$\mathbb{E} \left[ \sum_{t=1}^T M_t \right] \leq \frac{1}{\gamma} \bar{L}_{\gamma, T}(\{u_t\}) + \frac{1}{\gamma} \sqrt{\left( b \|u_1\|^2 + cV_m + \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right] \right) \times \left( \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t x_t^\top D_t^{-1} x_t \right] \right)}.$$

The last bound is an expectation version of the mistake bound for the (deterministic) LASEC algorithm of Thm. 2, and it might be even sharper than the LASEC bound, since the magnitude of the three quantities  $\bar{L}_{\gamma, T}(\{u_t\})$ ,  $\mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right]$  and  $\mathbb{E} \left[ \sum_{t=1}^T M_t Z_t x_t^\top D_t^{-1} x_t \right]$  is ruled by the size of the random set of updates  $\{t : Z_t M_t = 1\}$ , which is typically smaller than the set of mistaken trials of the deterministic algorithm.

We now prove the bounds in Thm. 2 and Thm. 4, in the following unified proof.

*Proof.* Consider only the rounds  $t$  when the algorithm makes an update, that is  $t \in \mathcal{M}$ . Noting that our choice  $\hat{p}_t = x_t^\top S_t^{-1} (I + c^{-1} D_{k-1})^{-1} e_{k-1}$  (in Fig. 1) is the same as the prediction of the LASER algorithm for regression with drift, we can use the result proven by Moroshko and Crammer [16] (Theorem 4 therein), from where we have that for any sequence  $u_1, \dots, u_m$

$$\sum_{t \in \mathcal{M}} (\hat{p}_t - y_t)^2 \leq b \|u_1\|^2 + c \sum_{k=2}^m \|u_k - u_{k-1}\|^2 + \sum_{k=1}^m (y_k - u_k^\top x_k)^2 + \sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t. \quad (8)$$

Note that in (8) the sums are over rounds when the algorithm makes an update (for simplicity, we write  $y_k$  as shorthand for  $y_{t_k}$  where  $t_k \in \mathcal{M}$ ). Expanding the squares in (8), lower bound  $\hat{p}_t^2 \geq 0$  and substituting  $y_t \hat{p}_t = -|\hat{p}_t|$  when  $t \in \mathcal{M}$  we obtain

$$\sum_{t \in \mathcal{M}} |\hat{p}_t| \leq \frac{b}{2} \|u_1\|^2 + \frac{c}{2} V_m - \sum_{k=1}^m y_k u_k^\top x_k + \frac{1}{2} \sum_{k=1}^m (u_k^\top x_k)^2 + \frac{1}{2} \sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t.$$

The last bound is correct for any sequence  $u_k$ . We replace  $u_k$  with  $\frac{a}{\gamma} u_k$  (for some  $a > 0$ ) and get

$$\sum_{t \in \mathcal{M}} |\hat{p}_t| \leq b \frac{a^2}{2\gamma^2} \|u_1\|^2 + c \frac{a^2}{2\gamma^2} V_m - \frac{a}{\gamma} \sum_{k=1}^m y_k u_k^\top x_k + \frac{a^2}{2\gamma^2} \sum_{k=1}^m (u_k^\top x_k)^2 + \frac{1}{2} \sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t.$$

Using  $\gamma - \ell_{\gamma,t}(u_t) \leq y_t u_t^\top x_t$ , which follows the definition of the hinge loss, we get

$$\sum_{t \in \mathcal{M}} (|\hat{p}_t| + a) \leq \frac{a}{\gamma} \sum_{t \in \mathcal{M}} \ell_{\gamma,t}(u_t) + \frac{1}{2} \sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t + \frac{a^2}{2\gamma^2} \left( b \|u_1\|^2 + c V_m + \sum_{k=1}^m (u_k^\top x_k)^2 \right). \quad (9)$$

Next, to prove the bound for the LASEC algorithm in Thm. 2, we further bound  $|\hat{p}_t| \geq 0$  in (9) and then divide it by  $a$ . We obtain

$$m \leq \frac{1}{\gamma} \sum_{t \in \mathcal{M}} \ell_{\gamma,t}(u_t) + \frac{1}{2a} \sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t + \frac{a}{2\gamma^2} \left( b \|u_1\|^2 + c V_m + \sum_{k=1}^m (u_k^\top x_k)^2 \right).$$

The last bound is minimized by setting

$$a = \gamma \sqrt{\frac{\sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t}{b \|u_1\|^2 + c V_m + \sum_{k=1}^m (u_k^\top x_k)^2}},$$

and by using  $\sum_{t \in \mathcal{M}} \ell_{\gamma,t}(u_t) \leq L_{\gamma,T}(\{u_t\})$  we get the desired bound of Thm. 2,

$$m \leq \frac{1}{\gamma} L_{\gamma,T}(\{u_t\}) + \frac{1}{\gamma} \sqrt{\left( b \|u_1\|^2 + c V_m + \sum_{k=1}^m (u_k^\top x_k)^2 \right) \sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t}.$$

To prove the mistake bound for the LASEC-SS algorithm in Thm. 4, we note that the sum  $\sum_{t \in \mathcal{M}} (|\hat{p}_t| + a)$  on the LHS of (9) can be written as  $\sum_t M_t Z_t (|\hat{p}_t| + a)$ . Taking expectation on both sides of (9) and using  $\mathbb{E} Z_t = a / (a + |\hat{p}_t|)$  we bound the expected number of mistakes of the algorithm,

$$\mathbb{E} \left[ \sum_{t=1}^T M_t \right] \leq \frac{1}{\gamma} \bar{L}_{\gamma,T}(\{u_t\}) + \frac{a}{2\gamma^2} \left( b \|u_1\|^2 + c V_m + \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right] \right) + \frac{1}{2a} \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t x_t^\top D_t^{-1} x_t \right].$$

The value of the expected number of queried labels trivially follows,  $\mathbb{E} \left[ \sum_{t=1}^T Z_t \right] = \sum_{t=1}^T \mathbb{E} \left[ \frac{a}{a + |\hat{p}_t|} \right]$ . ■

Next, we further bound the term  $\sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t$  in Thm. 2. Using Lemma 5 and Lemma 7 of Moroshko and Crammer [16] we have

$$\begin{aligned} \sum_{t \in \mathcal{M}} x_t^\top D_k^{-1} x_t &\leq \ln \left| \frac{1}{b} D_m \right| + c^{-1} \sum_{k=1}^m \text{Tr}(D_{k-1}) \\ &\leq \ln \left| \frac{1}{b} D_m \right| + c^{-1} \text{Tr}(D_0) \\ &+ \frac{m}{c} d \max \left\{ \frac{3X^2 + \sqrt{X^4 + 4X^2 c}}{2}, b + X^2 \right\}, \quad (10) \end{aligned}$$

where  $\|x_t\|^2 \leq X^2$ . Substituting (10) in (6) we get a bound of the form  $m \leq \frac{1}{\gamma} D + \frac{1}{\gamma} \sqrt{A(B + mC)}$  for the LASEC algorithm, solved for  $m$  with the following technical lemma.

**Lemma 7.** *Let  $A, B, C, D, \gamma, m > 0$  satisfy  $m \leq \frac{1}{\gamma} D + \frac{1}{\gamma} \sqrt{A(B + mC)}$ . Then*

$$m \leq \frac{1}{\gamma} D + \frac{1}{2\gamma^2} AC + \frac{1}{\gamma} \sqrt{\frac{1}{\gamma} DAC + \frac{1}{4\gamma^2} (AC)^2 + AB}. \quad (11)$$

The proof appears in the supplementary material. Using Lem. 7 we have the bound (11) for the LASEC algorithm,

where

$$A = b \|u_1\|^2 + cV_m + \sum_{k=1}^m (u_k^\top x_k)^2,$$

$$B = \ln \left| \frac{1}{b} D_m \right| + c^{-1} \text{Tr}(D_0),$$

$$C = c^{-1} d \max \left\{ (3X^2 + \sqrt{X^4 + 4X^2c}) / 2, b + X^2 \right\},$$

$$D = L_{\gamma, T}(\{u_t\}).$$

Next, we use corollary 8 from Moroshko and Crammer [16] to get the final bound for LASEC.

**Corollary 8.** Assume  $\|x_t\|^2 \leq X^2$  and set  $b = \varepsilon c$  for some  $0 < \varepsilon < 1$ . Denote  $\mu = \max \left\{ 9/8X^2, \frac{(b+X^2)^2}{8X^2} \right\}$ . Assume the LASEC algorithm is run on  $T$  examples. If  $V_m \leq T \frac{\sqrt{2dX}}{\mu^{3/2}}$  then by setting  $c = \left( \frac{\sqrt{2TdX}}{V_m} \right)^{2/3}$  we have the bound (11) for the number of mistakes of the LASEC algorithm, where

$$D = L_{\gamma, T}(\{u_t\}),$$

$$A = b \|u_1\|^2 + \left( \sqrt{2dX} \right)^{2/3} T^{2/3} V_m^{1/3} + \sum_{k=1}^m (u_k^\top x_k)^2,$$

$$B = \ln \left| \frac{1}{b} D_m \right| + \frac{\varepsilon}{1-\varepsilon} d,$$

$$C = (4dX)^{2/3} T^{-1/3} V_m^{1/3}.$$

*Proof.* As was shown [16] we have

$$\max \left\{ \frac{3X^2 + \sqrt{X^4 + 4X^2c}}{2}, b + X^2 \right\} = 2X\sqrt{2c},$$

and thus

$$\begin{aligned} A &= b \|u_1\|^2 + cV_m + \sum_{k=1}^m (u_k^\top x_k)^2 \\ &= b \|u_1\|^2 + \left( \sqrt{2dX} \right)^{2/3} T^{2/3} V_m^{1/3} + \sum_{k=1}^m (u_k^\top x_k)^2, \end{aligned}$$

$$B = \ln \left| \frac{1}{b} D_m \right| + c^{-1} \text{Tr}(D_0) = \ln \left| \frac{1}{b} D_m \right| + \frac{\varepsilon}{1-\varepsilon} d,$$

$$C = \frac{2\sqrt{2dX}}{\sqrt{c}} = \frac{2\sqrt{2dX}}{\left( \frac{\sqrt{2TdX}}{V_m} \right)^{1/3}} = (4dX)^{2/3} T^{-1/3} V_m^{1/3}. \quad \blacksquare$$

The last bound for the LASEC algorithm is difficult to interpret. Roughly speaking, the number of mistakes grows with the amount of drift as  $\sim T^{1/3} V_m^{2/3}$ , because  $A \sim T^{2/3} V_m^{1/3}$ ,  $C \sim T^{-1/3} V_m^{1/3}$  and the bound is  $\sim AC$ . Another bound for the drifting setting was shown by Cavallanti et al for the Shifting Perceptron [4]. However, they

used other notation of drift, which uses the norm rather than the square norm of the difference of comparison vectors, as we do. Thus, the two bounds are not comparable in general.

Next, we move to get explicit mistake bound for the LASEC-SS algorithm, by bounding the right term in Thm. 4. Again, using Lemma 5 and Lemma 7 from [16] we get,

$$\begin{aligned} \sum_{t=1}^T M_t Z_t x_t^\top D_t^{-1} x_t &\leq \ln \left| \frac{1}{b} D_T \right| + c^{-1} \sum_{t=1}^T \text{Tr}(D_{t-1}) \\ &\leq \ln \left| \frac{1}{b} D_T \right| + c^{-1} \text{Tr}(D_0) \\ &\quad + Tc^{-1} d \max \left\{ (3X^2 + \sqrt{X^4 + 4X^2c}) / 2, b + X^2 \right\}. \end{aligned}$$

Combining this bound with Thm. 4 we get,

$$\begin{aligned} \mathbb{E} \left[ \sum_{t=1}^T M_t \right] &\leq \frac{1}{\gamma} \bar{L}_{\gamma, T}(\{u_t\}) \\ &\quad + \frac{a}{2\gamma^2} \left( b \|u_1\|^2 + cV_m + \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right] \right) \\ &\quad + \frac{1}{2a} \left( \mathbb{E} \ln \left| \frac{1}{b} D_T \right| + c^{-1} \text{Tr}(D_0) \right) \\ &\quad + Tc^{-1} d \max \left\{ (3X^2 + \sqrt{X^4 + 4X^2c}) / 2, b + X^2 \right\}. \end{aligned}$$

We now state the main result of this section, bounding the exact number of mistakes of the LASEC-SS algorithm. This is an immediate application of corollary 8 from [16].

**Corollary 9.** Assume  $\|x_t\|^2 \leq X^2$  and set  $b = \varepsilon c$  for some  $0 < \varepsilon < 1$ . Denote  $\mu = \max \left\{ 9/8X^2, \frac{(b+X^2)^2}{8X^2} \right\}$ . Assume the LASEC-SS algorithm is run on  $T$  examples. If  $V_m \leq T \frac{\sqrt{2dX}}{\mu^{3/2}}$  then by setting  $c = \left( \frac{\sqrt{2TdX}}{V_m} \right)^{2/3}$  we get

$$\begin{aligned} \mathbb{E} \left[ \sum_{t=1}^T M_t \right] &\leq \frac{1}{\gamma} \bar{L}_{\gamma, T}(\{u_t\}) + b \frac{a}{2\gamma^2} \|u_1\|^2 \\ &\quad + \frac{a}{2\gamma^2} \left( \sqrt{2dX} \right)^{2/3} T^{2/3} V_m^{1/3} \\ &\quad + \frac{a}{2\gamma^2} \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right] \\ &\quad + \frac{1}{2a} \left( \mathbb{E} \ln \left| \frac{1}{b} D_T \right| + \frac{\varepsilon}{1-\varepsilon} d + (4dX)^{2/3} T^{-1/3} V_m^{1/3} \right). \end{aligned}$$

Again, we can optimize the last bound for the algorithm's parameter  $a$ . Setting

$$a = \gamma \sqrt{\frac{\mathbb{E} \ln \left| \frac{1}{b} D_T \right| + \frac{\varepsilon}{1-\varepsilon} d + (4dX)^{2/3} T^{-1/3} V_m^{1/3}}{\left( b \|u_1\|^2 + \left( \sqrt{2dX} \right)^{2/3} T^{2/3} V_m^{1/3} + \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right] \right)}}$$

we obtain

$$\begin{aligned} \mathbb{E} \left[ \sum_{t=1}^T M_t \right] &\leq \frac{1}{\gamma} \bar{L}_{\gamma, T}(\{u_t\}) \\ &+ \frac{1}{\gamma} \left[ \left( b \|u_1\|^2 + (\sqrt{2}dX)^{2/3} T^{2/3} V_m^{1/3} \right. \right. \\ &\left. \left. + \mathbb{E} \left[ \sum_{t=1}^T M_t Z_t (u_t^\top x_t)^2 \right] \right) \left( \mathbb{E} \ln \left| \frac{1}{b} D_T \right| + \frac{\varepsilon}{1-\varepsilon} d \right. \right. \\ &\left. \left. + (4dX)^{2/3} T^{2/3} V_m^{1/3} \right) \right]^{1/2}. \end{aligned}$$

## 5 Experimental Study

We evaluated our algorithm with both synthetic and real-world data with shifts, by comparing the average accuracy (total number of correct online classifications divided by the number of examples) of LASEC and LASEC-SS.

**Data:** In our first experiment we use a synthetic dataset with 10,000 examples of dimension  $d = 50$ . The inputs  $x_t \in \mathbb{R}^{50}$  were drawn from a zero-mean unit-covariance Gaussian distribution. The target  $u_t \in \mathbb{R}^{50}$  is a zero-mean unit-covariance Gaussian vector, which is switched every 500 examples to some other random vector. That is  $u_1 = \dots = u_{500}$ ,  $u_{501} = \dots = u_{1000}$ , .... The labels are set according to  $y_t = \text{sign}(x_t^\top u_t)$ . Our second experiment uses the US Postal Service handwritten digits recognition corpus (USPS) [14]. It contains normalized grey scale images of size  $16 \times 16$ , divided into a training (test) set of 7,291 (2,007) images. We combined the sets to get 9,298 examples. Based on the USPS multiclass data we generated binary data with shifts. We chose at random some digits to be positive class (the other digits are negative class). The partition to positive and negative classes is changed every 500 examples at random, that is every 500 samples we changed the subgroup of labels (out of 10) that are labeled as +1 (labels in the complementary subgroup are labeled as -1). Each set of experiments was repeated 50 times and the error bars in the plots correspond to the 95% confidence interval over the 50 runs.

**Supervised Online Learning with Drift:** In the supervised-online classification task we compared the performance of LASEC (setting  $a = \infty$  in Fig. 1) to five other algorithms: the second-order perceptron algorithm (SOP) [5], the Perceptron algorithm [18], the Shifting Perceptron algorithm [4], the Modified Perceptron algorithm [3] and the Randomized Budget Perceptron algorithm [4].

Both the Shifting Perceptron and the Randomized Budget Perceptron are tuned using a single parameter (denoted by  $\lambda$  and  $B$  respectively). Since the optimal values of these

parameters simply reduced these algorithms to the original Perceptron, we set  $\lambda = 0.01$  and  $B = 500$  for synthetic data, and  $\lambda = 0.0001$  and  $B = 1,000$  for real-world data. The setting of  $B = 500$  is actually the switching window, while for real-world data we alleviated the Randomized Budget Perceptron and used twice the switching window as the budget. For the LASEC and SOP algorithms the parameters were tuned using a random draw of the data.

The results comparing supervised classification algorithms on synthetic data are shown in Fig. 2(a). While for  $t < 500$  (before the first shift) the SOP algorithm is the best as expected, we see that the LASEC algorithm deals better with the shifts and outperforms other algorithms. For the real-world USPS dataset (see Fig. 2(d)) LASEC slightly outperforms SOP, and both outperform other perceptron-like algorithms, due to the usage of second-order information.

**Selective Sampling Online Learning with Drift:** For the selective sampling task we compared the LASEC-SS algorithm from Fig. 1 to several selective sampling algorithms: the selective sampling version of second-order perceptron algorithm (SOP-SS) [7], the selective sampling version of perceptron algorithm (Perceptron-SS) [7] and the BBQ algorithm [6]. In addition to the BBQ algorithm [6], we also consider a variant of the BBQ algorithm, which we call BBQ-I. This algorithm is similar to the original BBQ algorithm but it performs updates only when the queried label is different from the predicted label. Each algorithm has one parameter that controls the tradeoff between the query rate (fraction of queried labels) and the accuracy of the algorithm. For fairness, this parameter was set to get about the same query rate for all algorithms.

Fig. 2(b) and Fig. 2(e) summarize the accuracy of the algorithms on synthetic data for query rate  $\sim 0.1$  and  $\sim 0.4$ . In both cases LASEC-SS outperforms other algorithms. Before the first shift at round 500 the BBQ is the best as expected from previous results [17], but its performance significantly degrades after the first shift. This is because this algorithm performs query when the quantity  $r_t = x_t^\top A_t^{-1} x_t$  is large enough (see [6]), while the matrix  $A_t$  grows each time a label is queried. Large  $A_t$  makes  $r_t$  small, the algorithm converges and stops query labels. If after that a switch occurs, the algorithm fails in predictions but cannot query correct labels because  $r_t$  is small. This causes a significant degradation in the prediction accuracy. On the other hand, the BBQ-I algorithm performs less updates (as it performs updates only when a mistake occurs), and thus the algorithm converges much slower. This makes it simpler to adapt to changing environment, after a switch occurs. We note that for stationary environment (when  $u_t = u \forall t$ ), BBQ outperforms BBQ-I, as well as other selective sampling algorithms (see [17]). For low query rate as in Fig. 2(b), all algorithms hardly deal with the shifts, as expected. However, our algorithm still converges on a bet-

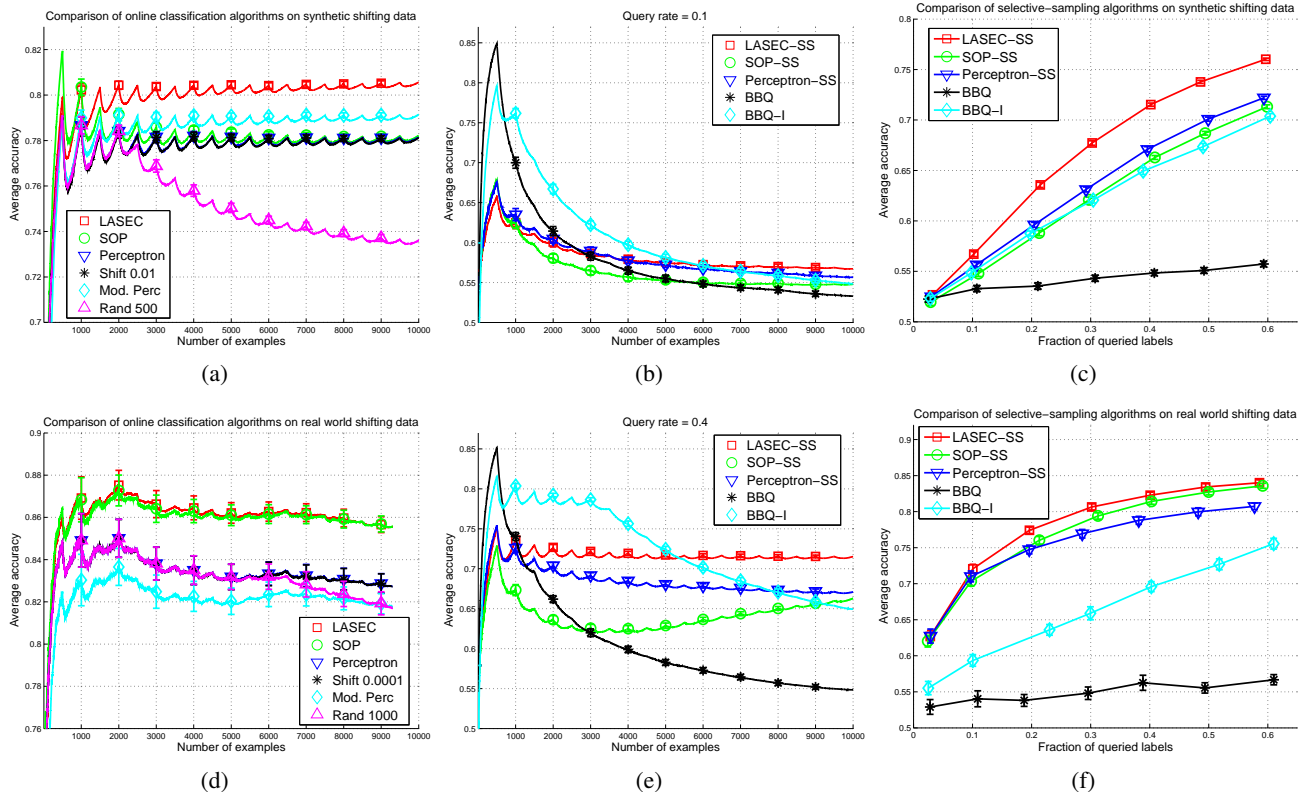


Figure 2: Left: accuracy against number of examples for binary fully supervised classification algorithms on (a) synthetic shifting dataset, and (d) USPS shifting dataset. Middle: accuracy against number of examples with query rate (b)  $\sim 0.1$ , and (e)  $\sim 0.4$ . Right: accuracy against fraction of queried labels for selective sampling algorithms on (c) synthetic shifting dataset, and (f) USPS shifting dataset.

ter average accuracy. For higher rate as in Fig. 2(e), our algorithm deals well with the shifts and the average accuracy does not decrease. This is in contrary to other algorithms. For the SOP-SS algorithm we see in Fig. 2(e) that the performance increase over time after an initial drop. This is because the SOP algorithm tends to converge fast and then it acts as no labels are needed, because the margin is large. After a data shift, the algorithm experiences a drop because no labels are sampled. After some time the algorithm detects that labels are needed (because the margin is small) and performance increase.

Fig. 2(c) and Fig. 2(f) show the tradeoff between average accuracy and fraction of queried labels on synthetic and real-world (USPS) data accordingly. Evidently, LASEC-SS is the best selective sampling algorithm in the drifting setting. In addition, we can see that unlike the stationary setting where it was shown [7] that a small fraction of labels are enough to get the accuracy of a fully supervised setting, in the drifting case much more labels are needed. This is because old queried labels cannot contribute to form a good predictor due to a drift in the model, and the algorithm must query more labels to have a good prediction accuracy. Yet, our algorithm can employ half of the labels to get performance not too far from the full information case.

## 6 Conclusions

We proposed a novel second-order algorithm for binary classification designed to work in non-stationary (drifting) selective sampling setting. Our algorithm is based on the last-step min-max approach, and we showed how to solve the last-step min-max optimization problem directly for classification using the square loss. To the best of our knowledge, this is the first algorithm designed to work in the selective sampling setting when there is a drift. We proved mistake bound for the algorithm in the fully supervised setting, and a bound for the expected number of mistakes for the selective sampling version of the algorithm. Experimental study shows that our algorithm outperforms other algorithms, in the supervised and selecting sampling settings. For the algorithm to perform well, the amount of drift  $V$  or a bound over it should be known in advance. An interesting direction is to design algorithms that automatically detect the level of drift, or are invariant to it.

**Acknowledgements:** This research was funded in part by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI) and in part by an Israeli Science Foundation grant ISF- 1567/10.



## References

- [1] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.
- [2] K.S. Azoury and M.W. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.
- [3] Avrim Blum, Alan M. Frieze, Ravi Kannan, and Santosh Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1/2):35–52, 1998.
- [4] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.
- [5] Nicolás Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *Siam Journal of Computation*, 34(3):640–668, 2005.
- [6] Nicolò Cesa-Bianchi, Claudio Gentile, and Francesco Orabona. Robust bounds for classification via selective sampling. In *ICML*, pages 121–128, 2009.
- [7] Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research*, 7:1205–1230, 2006.
- [8] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weighted vectors. In *Advances in Neural Information Processing Systems 23*, 2009.
- [9] Koby Crammer, Yishay Mansour, Eyal Even-Dar, and Jennifer Wortman Vaughan. Regret minimization with concept drift. In *COLT*, pages 168–180, 2010.
- [10] Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Robust selective sampling from single and multiple teachers. In *COLT*, pages 346–358, 2010.
- [11] Jurgen Forster. On relative loss bounds in generalized linear regression. In *Fundamentals of Computation Theory (FCT)*, 1999.
- [12] Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Selective sampling using the Query By Committee algorithm. *Machine Learning*, 28:133–168, 1997.
- [13] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.
- [14] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):550–554, May 1994.
- [15] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1987.
- [16] Edward Moroshko and Koby Crammer. A last-step regression algorithm for non-stationary online learning. In *AISTATS*, 2013.
- [17] Francesco Orabona and Nicolò Cesa-Bianchi. Better algorithms for selective sampling. In *ICML*, pages 433–440, 2011.
- [18] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.
- [19] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.
- [20] J.A.K. Suykens, T. van Gestel, and J. de Brabanter. *Least Squares Support Vector Machines*. World Scientific Publishing Company Incorporated, 2002.
- [21] Johan A. K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
- [22] Nina Vaits and Koby Crammer. Re-adapting the regularization of weights for non-stationary regression. In *The 22nd International Conference on Algorithmic Learning Theory, ALT '11*, 2011.
- [23] Volodya Vovk. Competitive on-line statistics. *International Statistical Review*, 69, 2001.