Re-Adapting the Regularization of Weights for Non-Stationary Regression

Nina Vaits and Koby Crammer

Department of Electrical Engneering,
The Technion, Haifa, Israel
nvaits@tx.technion.ac.il,koby@ee.technion.ac.il

Abstract. The goal of a learner in standard online learning is to have the cumulative loss not much larger compared with the best-performing prediction-function from some fixed class. Numerous algorithms were shown to have this gap arbitrarily close to zero compared with the best function that is chosen off-line. Nevertheless, many real-world applications (such as adaptive filtering) are non-stationary in nature and the best prediction function may not be fixed but drift over time. We introduce a new algorithm for regression that uses per-feature-learning rate and provide a regret bound with respect to the best sequence of functions with drift. We show that as long as the cumulative drift is sub-linear in the length of the sequence our algorithm suffers a regret that is sub-linear as well. We also sketch an algorithm that achieves the best of the two worlds: in the stationary settings has $\log(T)$ regret, while in the non-stationary settings has sub-linear regret. Simulations demonstrate the usefulness of our algorithm compared with other state-of-the-art approaches.

1 Introduction

We consider the classical problem of online learning for regression. On each iteration, the algorithm receives a new instance (for example, input from an array of antennas) and outputs a prediction of a real value (for example distance to the source). The correct value is then revealed, and the algorithm suffers a loss based on both its prediction and the correct output value.

In general, the goal of the learner is to achieve an average loss that is not *too big* compared with the loss it would have received if it had chosen to predict according to the single best-performing function from some fixed class. It is well-known that as the number of time steps grows, very simple aggregation algorithms are able to achieve an average loss arbitrarily close to that of the best function in retrospect. Furthermore, such guarantees hold even if the input and output pairs are chosen in a fully adversarial manner with no distributional assumptions [6].

Despite the extensive and impressive guarantees that can be made for algorithms in such settings, 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. Consider a filter designed to cancel echoes in an hall. Over time, people enter and leave the hall, furniture are being moved, microphones are replaced and so on. When this drift occurs, the predictor itself must also change in order to remain relevant.

These reasons led to the development of algorithms and accompanying analysis for drifting or shifting settings (for example [24, 1, 20, 23] and the references therein). In this setting, the performance of an algorithm is compared with a sequence of functions (and not a single function). Often such a sequence is either drifting, where each function is close in some sense to its predecessor, or shifting, where conceptually the sequence can be partitioned into few segments, for each there is a single function that performs well on all examples of that segment.

Recently there is an increased amount of interest in algorithms that exploits second order information. For example the second order perceptron algorithm [5], confidence-weighted learning [10, 8], adaptive regularization of weights (AROW) [9], all designed for classification; and AdaGrad [11] and FTPRL [25] for general loss functions.

In this paper we build on the AROW algorithm and develop an algorithm for regression. Such algorithms are known to work well in practice and converge fast for stationary-problems. However, for non-stationary problems AROW and other similar algorithms gradually stop updating their prediction rule, even though their performance deteriorates. We thus modify the algorithm to overcome these shortcomings. We analyze the algorithm in the worst-case regret framework and show, that as long as the amount of average-drift is sublinear, the average-loss of our algorithm will converge to the average-loss of the best sequence of functions. Specifically, we show that if the cumulative deviation is of order $\mathcal{O}\left(T^{1/p}\right)$ for some known $p > 1$, then the cumulative regret is $\mathcal{O}\left(T^{(p+1)/(2p)} \log(T)\right)$. We also show that for stationary setting the algorithm suffers logarithmic regret, similar to previous results [13].

Additionally, we sketch an algorithm that does not employ such prior knowledge. Specifically, this algorithm runs $C + 1$ copies of the algorithm mentioned above, each copy with a parameter chosen to fit a specific assumption of the amount of non-stationarity in the data. The algorithm then feeds these $C+1$ outputs to an additional copy that computes a linear combination of these $C + 1$ outputs. Thus, the cumulative loss of the later algorithm is bounded by the cumulative loss of the best copy (ie the one with the "best" choice of parameter) with additional regret of $\mathcal{O}(\log T)$. Therefore, this algorithm for the non-stationary setting will suffer a polynomial sub-linear regret.

Notation: For a symmetric matrix Σ we denote its jth eigenvalue by $\lambda_j(\Sigma)$. Similarly we denote its smallest eigenvalue by $\lambda_{min}(\Sigma) = \min_j \lambda_j(\Sigma)$, and its largest eigenvalue by$\lambda_{max}(\Sigma) = \max_j \lambda_j(\Sigma)$.

2 Problem Setting

We work in the online setting for regression evaluated using the square loss. On each iteration our algorithm receives an instance $\boldsymbol{x}_t \in \mathbb{R}^d$ and predicts a real value $\hat{y}_t \in \mathbb{R}$ it then receives the correct label $y_t \in \mathbb{R}$, suffers loss $\ell\left(y_t, \hat{y}_t\right) = \left(\hat{y}_t - y_t\right)^2$. The algorithm then updates its prediction rule, and proceeds to the next round.

Our algorithms employs linear functions (with bounded norm), $\hat{y}_t = \boldsymbol{x}_t^\top \boldsymbol{w}_{t-1}$. The goal of the learning algorithm is to suffer low loss compared with the best linear

function. Formally, we define the regret of an algorithm to be,

$$\mathcal{R}(T) = \sum_t^T \left(\boldsymbol{x}_t^\top \boldsymbol{w}_{t-1} - y_t\right)^2 - \inf_{\boldsymbol{u}} \sum_t^T \left(\boldsymbol{x}_t^\top \boldsymbol{u} - y_t\right)^2 \ .$$

The goal of the algorithm is to have $\mathcal{R}(T) = o(T)$, such that the average loss will converge to the average loss of the best linear function $\boldsymbol{u}$. We use an extended notion of evaluation, comparing the performance of an algorithm to the performance of a sequence of functions,

$$\mathcal{R}(T) = \sum_t^T \left(\boldsymbol{x}_t^\top \boldsymbol{w}_{t-1} - y_t\right)^2 - \inf_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_T} \sum_t^T \left(\boldsymbol{x}_t^\top \boldsymbol{u}_t - y_t\right)^2 \ .$$

It is reasonable to assume that the total deviation of the compared sequence is sub-linear in T, that is, for which $\sum_t \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| = o(T)$. Clearly, if the total deviation is $\Omega(T)$ we can not expect a learning algorithm to achieve a vanishing averaged regret. Yet, it may be the case that the sequence of functions is not converging $\lim_{t\to\infty} \sum_{s=1}^t \|\boldsymbol{u}_{s-1} - \boldsymbol{u}_s\|$ yet the algorithm will have vanishing average regret.

3 Algorithm

As in CW [10, 8] and AROW [9] our algorithm maintains a Gaussian distribution parameterized by a mean $\boldsymbol{w}_t \in \mathbb{R}^d$ and a full covariance matrix $\Sigma_t \in \mathbb{R}^{d\times d}$. Intuitively, the mean $\boldsymbol{w}_t$ represents the current linear function, while the covariance matrix Σ_t captures the uncertainty in the function $\boldsymbol{w}_t$. Given a new example $(\boldsymbol{x}_t, y_t)$ the algorithm uses its current mean to make a prediction $\hat{y}_t = \boldsymbol{x}_t^\top \boldsymbol{w}_{t-1}$. Our algorithm then sets the new distribution to be the solution of the following optimization problem,

$$\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{w}, \Sigma\right) \| \mathcal{N}\left(\boldsymbol{w}_{t-1}, \Sigma_{t-1}\right)\right) + \frac{1}{2r}\ell\left(y_t - \boldsymbol{w}^\top \boldsymbol{x}_t\right) + \frac{1}{2r}\left(\boldsymbol{x}_t^\top \Sigma \boldsymbol{x}_t\right)$$

This optimization problem is similar to the one of AROW [9] for classification, except we use the square loss rather than squared-hinge loss used in AROW. Intuitively, the optimization problem trades off between three requirements. The first term forces the parameters not to change much per example, as the entire learning history is encapsulated within them. The second term requires that the new vector $\boldsymbol{w}_t$ should perform well on the current instance, and finally, the third term reflects the fact that the uncertainty about the parameters reduces as we observe the current example $\boldsymbol{x}_t$. Writing the KL explicitly, we get the following.

$$\frac{1}{2}\log\left(\frac{\det \Sigma_{t-1}}{\det \Sigma}\right) + \frac{1}{2}\mathrm{Tr}\left(\Sigma_{t-1}^{-1}\Sigma\right) + \frac{1}{2}\left(\boldsymbol{w}_{t-1} - \boldsymbol{w}\right)^\top \Sigma_{t-1}^{-1}\left(\boldsymbol{w}_{t-1} - \boldsymbol{w}\right) - \frac{d}{2} + \frac{1}{2r}\ell\left(y_t - \boldsymbol{w}^\top \boldsymbol{x}_t\right) + \frac{1}{2r}\left(\boldsymbol{x}_t^\top \Sigma \boldsymbol{x}_t\right) \ . \quad (1)$$

We now develop the update rule of (1) explicitly. Taking the derivative of (1) with respect to $\boldsymbol{w}$ and setting it to zero, we get $\Sigma_{t-1}^{-1}(\boldsymbol{w}-\boldsymbol{w}_{t-1})-\frac{1}{2r}2(\boldsymbol{w}_t\cdot\boldsymbol{x}_t-y_t)\boldsymbol{x}_t=0$. Therefore, if Σ_{t-1} is non-singular, the update for the mean parameters is given by

$$\boldsymbol{w}_t=\boldsymbol{w}_{t-1}-\frac{1}{r}(\boldsymbol{w}_t\cdot\boldsymbol{x}_t-y_t)\Sigma_{t-1}\boldsymbol{x}_t\ . \tag{2}$$

We solve for $\boldsymbol{w}_t$ by taking the dot product of each side of the equality with $\boldsymbol{x}_t$: $\boldsymbol{w}_t\cdot\boldsymbol{x}_t=\boldsymbol{w}_{t-1}\cdot\boldsymbol{x}_t-\frac{1}{r}(\boldsymbol{w}_t\cdot\boldsymbol{x}_t-y_t)\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t$. Rearranging the terms yields, $(\boldsymbol{w}_t\cdot\boldsymbol{x}_t)\left(r+\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t\right)=(\boldsymbol{w}_{t-1}\cdot\boldsymbol{x}_t)r+\left(y_t\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t\right)$, and substituting back in (2), we get

$$\begin{aligned}\boldsymbol{w}_t=&\boldsymbol{w}_{t-1}-\frac{1}{r}\left(\frac{(\boldsymbol{w}_{t-1}\cdot\boldsymbol{x}_t)r+\left(y_t\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t\right)}{r+\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t}-y_t\right)\Sigma_{t-1}\boldsymbol{x}_t\\ =&\boldsymbol{w}_{t-1}-\left(\frac{(\boldsymbol{w}_{t-1}\cdot\boldsymbol{x}_t)-y_t}{r+\boldsymbol{x}_t^\top\Sigma_{t-1}\boldsymbol{x}_t}\right)\Sigma_{t-1}\boldsymbol{x}_t\ .\end{aligned} \tag{3}$$

We compute the update of the confidence parameters by setting the derivative of (1) with respect to Σ to zero, $-\frac{1}{2}\Sigma^{-1}+\frac{1}{2}\Sigma_{t-1}^{-1}+\frac{1}{2r}\boldsymbol{x}_t\boldsymbol{x}_t^\top=0$ .From this we obtain the following update for the confidence parameters.

$$\Sigma_t^{-1}=\Sigma_{t-1}^{-1}+\frac{1}{r}\boldsymbol{x}_t\boldsymbol{x}_t^\top\ . \tag{4}$$

Our goal is to develop algorithms for non-stationary settings. As observed in the context of CW [10], AROW [9], AdaGrad [11] and FTPRL [25] the matrix Σ can be interpreted as adaptive learning rate. Therefore, due to the update of (4) the eigenvalues of Σ_t goes to zero with t (equivalently, the update (4) forces the eigenvalues of Σ_t^{-1} to increase) and the effective learning rate goes to zero. As a consequence the algorithm will gradually stop updating using instances which lie in the subspace of examples that were previously observed numerous times. This property leads to fast convergence in the stationary case, but at the same time to poor performance in the non-stationary case. As it might happen there is a need to update the prediction rule with using some instance, yet the learning rate for this specific update is too small, and no useful update may be performed.

We propose two modifications to the above algorithm, that combined together overcome the problem that learning rate gradually goes to zero. The modified algorithm operates on segments of the input sequence. In each segment indexed by i, the algorithm checks weather the lowest eigenvalue of Σ_t is greater than a given lower bound Λ_i. Once the lowest eigenvalue of Σ_t is smaller than Λ_i the algorithm resets $\Sigma_t=I$ and updates the value of the lower bound Λ_{i+1}. Formally, the algorithm uses the update (4) to compute an intermediate candidate for Σ_t denote by $\tilde{\Sigma}_t=\left(\Sigma_{t-1}^{-1}+\frac{1}{r}\boldsymbol{x}_t\boldsymbol{x}_t^\top\right)^{-1}$. If indeed $\tilde{\Sigma}_t\succeq\Lambda_i I$ then it sets $\Sigma_t=\tilde{\Sigma}_t$ otherwise it sets $\Sigma_t=I$ and the segment index is increased by 1.

Additionally, before this modification, the norm of the weight vector $\boldsymbol{w}_t$ did not increase much as the "effective" learning rate (the matrix Σ_t) went to zero. After our

update, as the learning rate is effectively bounded from below, the norm of $\boldsymbol{w}_t$ may increase too fast, which in turn will cause a low update-rate in non-stationarity inputs.

We thus employ one more modification exploited later by the analysis. After updating the mean $\boldsymbol{w}_t$ as in (3) we project it into a ball B around the origin with radius R_B using a Mahalanobis distance. Formally, we define the function $\text{proj}(\tilde{\boldsymbol{w}}, \Sigma, R_B)$ to be the solution of the following optimization problem,

$$\arg \min_{\|\boldsymbol{w}\| \leq R_B} \frac{1}{2} (\boldsymbol{w} - \tilde{\boldsymbol{w}})^\top \Sigma^{-1} (\boldsymbol{w} - \tilde{\boldsymbol{w}}) \tag{5}$$

We write the Lagrangian,

$$\mathcal{L} = \frac{1}{2} (\boldsymbol{w} - \tilde{\boldsymbol{w}})^\top \Sigma^{-1} (\boldsymbol{w} - \tilde{\boldsymbol{w}}) + \alpha \left(\frac{1}{2} \|\boldsymbol{w}\|^2 - \frac{1}{2} R_B^2 \right) .$$

Setting to zero the gradient with respect to $\boldsymbol{w}$ we get, $\Sigma^{-1} (\boldsymbol{w} - \tilde{\boldsymbol{w}}) + \alpha \boldsymbol{w} = 0$. Solving for $\boldsymbol{w}$ we get

$$\boldsymbol{w} = \left(\alpha I + \Sigma^{-1}\right)^{-1} \Sigma^{-1} \tilde{\boldsymbol{w}} = \frac{1}{\alpha} \left(I + (\alpha \Sigma)^{-1} \right)^{-1} \Sigma^{-1} \tilde{\boldsymbol{w}} \quad = (I + \alpha \Sigma)^{-1} \tilde{\boldsymbol{w}} .$$

From KKT conditions we get that If $\|\tilde{\boldsymbol{w}}\| \leq R_B$ then $\alpha = 0$ and $\boldsymbol{w} = \tilde{\boldsymbol{w}}$. Otherwise, α is the unique positive scalar that satisfy the equality,

$$\| (I + \alpha \Sigma)^{-1} \tilde{\boldsymbol{w}} \| = R_B .$$

The value of α can be found using binary search and eigen-decomposition of the matrix Σ. We write explicitly $\Sigma = V \Lambda V^\top$ for a diagonal matrix Λ. By denoting $\boldsymbol{u} = V^\top \tilde{\boldsymbol{w}}$ we get that the last equation equals to, $\| (I + \alpha \Lambda)^{-1} \boldsymbol{u} \| = R_B$. We thus wish to find α such that $\sum_j^d \frac{u_j^2}{(1+\alpha \Lambda_{j,j})^2} = R_B^2$. It can be done using a binary search in the range $\alpha \in [0, a]$ where $a = (\|\boldsymbol{u}\|/R_B - 1)/\lambda_{\min}(\Lambda)$. To summarize the projection step can be performed in time cubic in d and logarithmic in R_B and Λ_i. We call the algorithm ARCOR for adaptive regularization with covariance reset. A pseudo-code of the algorithm is summarized in Fig. 1. We note that it can be combined with Mercer kernels, and omit the details due to lack of space.

4 Analysis

We turn to analyze the algorithm in the non-stationary case, computing the regret with respect to more than a single comparison vector. Before we proceed with the bound we define additional notation. We denote by t_i the example index for which a restart was performed for the ith time, that is $\Sigma_{t_i} = I$ for all i. We define by n the total number of restarts, or intervals. We denote by $T_i = t_i - t_{i-1}$ the number of examples between two consecutive restarts. Clearly $T = \sum_{i=1}^n T_i$. Finally, we denote by $\Sigma^{i-1} = \Sigma_{t_i - 1}$ just before the ith restart, and we note that it depends on exactly T_i examples (since the last restart).

In what follows we compare the performance of the algorithm to the performance of a sequence of weight vectors $\boldsymbol{u}_t \in \mathbb{R}^d$ all of which are of bounded norm R_B. In other words, all the vectors $\boldsymbol{u}_t$ belong to $B = \{\boldsymbol{u} \ : \ \|\boldsymbol{u}\|_2 \leq R_B\}$.

Parameters: $0 < r, R_B$, a sequence $1 > \Lambda_1 \geq \Lambda_2 ...$
Initialize: Set $\boldsymbol{w}_0 = 0$, $\Sigma_0 = I$, $i = 1$
For $t = 1, \ldots, T$ do
- Receive an instance $\boldsymbol{x}_t$
- Output prediction $\hat{y}_t = \boldsymbol{x}_t^\top \boldsymbol{w}_{t-1}$
- Receive the correct label y_t
- Update:

$$\tilde{\Sigma}_t^{-1} = \Sigma_{t-1}^{-1} + \frac{1}{r}\boldsymbol{x}_t \boldsymbol{x}_t^\top \tag{6}$$

$$\tilde{\boldsymbol{w}}_t = \boldsymbol{w}_{t-1} + \frac{(y_t - \boldsymbol{x}_t^\top \boldsymbol{w}_{t-1})\Sigma_{t-1}\boldsymbol{x}_t}{r + \boldsymbol{x}_t^\top \Sigma_{t-1}\boldsymbol{x}_t} \tag{7}$$

- Update Σ_t:
  If $\tilde{\Sigma}_t \succeq \Lambda_i I$ set $\Sigma_t = \tilde{\Sigma}_t$ else set $\Sigma_t = I$, $i = i + 1$
- Update $\boldsymbol{w}_t$:
  $\boldsymbol{w}_t = \text{proj}\left(\tilde{\boldsymbol{w}}_t, \Sigma_t, R_B\right)$

Output: $\boldsymbol{w}_T$, Σ_T

Fig. 1. ARCOR: adaptive regularization of weights for regression with covariance reset.

Theorem 1. *Assume the algorithm of Fig. 1 is run with an input sequence* $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$*, that all the inputs are of unit norm* $\|\boldsymbol{x}_t\| = 1$*, and that the outputs are bounded* $Y = \max_t |y_t|$*. Let* $\boldsymbol{u}_t$ *be any sequence of bounded weight vectors* $\|\boldsymbol{u}_t\| \leq R_B$*. The cumulative loss is bounded by,*

$$\sum_t \left(\boldsymbol{x}_t^\top \boldsymbol{w}_{t-1} - y_t\right)^2 \leq \sum_t \left(\boldsymbol{x}_t^\top \boldsymbol{u}_t - y_t\right)^2 + 2\left(R_B^2 + Y^2\right)\sum_{i=1}^{n} \log\det\left(\left(\Sigma^i\right)^{-1}\right) + r\boldsymbol{u}_T^\top \Sigma_T^{-1}\boldsymbol{u}_T + 2R_B r \sum_t \frac{1}{\Lambda_{i(t)}}\|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| , \tag{8}$$

where n *is the number of covariance restarts and* Σ^{i-1} *is the value of the covariance matrix just before the* i*th restart.*

Note that the number of restarts n is not fixed but depends both on the total number of examples T and the scheme used to set the values of the lower bound of the eigenvalues Λ_i. In general, the lower the values of Λ_i are, the smaller the number of covariance-restarts occur, yet the larger the value of the last term of the bound is, which scales inversely proportional to Λ_i. A more precise statement is given in the next corollary.

Corollary 1. *Assume the algorithm of Fig. 1 made* n *restarts. Under the conditions of Theorem 1 we have,*

$$\sum_t \left(\boldsymbol{x}_t^\top \boldsymbol{w}_{t-1} - y_t\right)^2 \leq \sum_t \left(\boldsymbol{x}_t^\top \boldsymbol{u}_t - y_t\right)^2 + 2\left(R_B^2 + Y^2\right) dn \log\left(1 + \frac{T}{nrd}\right) + r\boldsymbol{u}_T^\top \Sigma_T^{-1}\boldsymbol{u}_T + 2R_B r \Lambda_n^{-1} \sum_t \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| , \tag{9}$$

Proof. By definition we have $\left(\Sigma^i\right)^{-1} = I + \frac{1}{r}\sum_{t=t_i}^{T_i+t_i} \boldsymbol{x}_t\boldsymbol{x}_t^\top$. Denote the eigenvalues of $\sum_{t=t_i}^{T_i+t_i} \boldsymbol{x}_t\boldsymbol{x}_t^\top$ by $\lambda_1,\ldots,\lambda_d$. Since $\|\boldsymbol{x}_t\| = 1$ their sum is $\mathrm{Tr}\left(\sum_{t=t_i}^{T_i+t_i} \boldsymbol{x}_t\boldsymbol{x}_t^\top\right) = T_i$. We use the concavity of the $\log$ function to bound $\log\det\left(\left(\Sigma^i\right)^{-1}\right) = \sum_j^d \log\left(1+\frac{\lambda_j}{r}\right)$ $\leq d\log\left(1+\frac{T_i}{rd}\right)$. Applying concavity again we bound the following sum,

$$\sum_i^n \log\det\left(\left(\Sigma^i\right)^{-1}\right) \leq \sum_i^n d\log\left(1+\frac{T_i}{rd}\right) \leq dn\log\left(1+\frac{T}{nrd}\right) ,$$

where we used the fact that $\sum_i^n T_i = T$. Substituting the last inequality in Theorem 1, as well as using the monotinicity of the coefficients, $\Lambda_i \geq \Lambda_n$ for all $i \leq n$, yields the desired bound. ■

Implicitly, the second and fourth terms of the bound have opposite dependence on n. The second term is increasing with $n \ll T$, while the fourth is decreasing with n. If n is small it means that the lower bound Λ_n is very low (otherwise we would make more restarts) and thus Λ_n^{-1} is large. We now make this implicit dependence explicit.

Our goal is to bound the number of restarts n as a function of the number of examples T. This depends on the exact sequence of values Λ_i used. The following lemma provides a bound on n given a specific sequence of Λ_i.

Lemma 1. *Assume the algorithm of Fig. 1 is run with some sequence of Λ_i, then the number of restarts is upper bounded by,*

$$n \leq \min_N \left\{ N \;:\; T \leq r\sum_i^N \left(\Lambda_i^{-1} - 1\right) \right\} .$$

Proof. Since $\sum_{i=1}^n T_i = T$, then the number of restarts is maximized when the number of examples between restrats T_i is minimized. We prove now a lower bound on T_i for all $i = 1\ldots n$. A restart occurs for the ith time when the smallest eigenvalue of Σ_t is larger (for the first time) then Λ_i.

As before, by definition we have, $\left(\Sigma^i\right)^{-1} = I + \frac{1}{r}\sum_{t=t_i}^{T_i+t_i} \boldsymbol{x}_t\boldsymbol{x}_t^\top$. By Theorem 8.1.8 of [15] we have that there exists a matrix $A \in \mathbb{R}^{d\times T_i}$ with each column belongs to the $d-1$-dimensional simplex (that is $a_{k,l} \geq 0$ and $\sum_k a_{k,l} = 1$ for $l = 1,\ldots,T_i$) such that the kth eigenvalue λ_k^i of $\left(\Sigma^i\right)^{-1}$ equals to $\lambda_k^i = 1 + \frac{1}{r}\sum_{l=1}^{T_i} a_{k,l}$. The value of T_i is defined when the largest eigenvalue of $\left(\Sigma^i\right)^{-1}$ hits Λ_i^{-1}. Formally, we get the following lower bound on T_i,

$$\begin{aligned} \arg\min_{\{a_{k,l}\}} \; s \quad \text{s.t.} \qquad & \max_k \left(1 + \frac{1}{r}\sum_{l=1}^{s} a_{k,l}\right) \geq \Lambda_i^{-1} \\ & a_{k,l} \geq 0 \quad \text{for } k = 1,\ldots,d, l = 1,\ldots,s \\ & \sum_k a_{k,l} = 1 \quad \text{for } l = 1,\ldots,s \end{aligned}$$

For a fixed s a maximal value $\max_k \left(1 + \frac{1}{r}\sum_{l=1}^{s} a_{k,l}\right)$ is obtained if all the "mass" is concentrated in one value k. That is we have $a_{k,l} = 1$ for $k = k_0$ and $a_{k,l} = 0$ otherwise. In this case $\max_k \left(1 + \frac{1}{r}\sum_{l=1}^{s} a_{k,l}\right) = \left(1 + \frac{1}{r}s\right)$ and the lower bound is obtained when $\left(1 + \frac{1}{r}s\right) = \Lambda_i^{-1}$. Solving for s we get that the shortest possible length of the ith interval is bounded by, $T_i \leq r\left(\Lambda_i^{-1} - 1\right)$. Summing over the last equation we get, $T = \sum_i^n T_i \leq r\sum_i^n \left(\Lambda_i^{-1} - 1\right)$. Thus, the number of restarts is upper bounded by the minimal value n that satisfy the last inequality. ■

Combining Lemma 1 with Corollary 1 we get,

Corollary 2. *Assume the algorithm of Fig. 1 is ran with a polynomial schema, that is* $\Lambda_i^{-1} = i^{q-1} + 1$ *for some* $q > 1$*. Under the conditions of Theorem 1 we have,*

$$\sum_t \left(\boldsymbol{x}_t^\top \boldsymbol{w}_{t-1} - y_t\right)^2 \leq \sum_t \left(\boldsymbol{x}_t^\top \boldsymbol{u}_t - y_t\right)^2 + r\boldsymbol{u}_T^\top \Sigma_T^{-1}\boldsymbol{u}_T$$

$$+ 2\left(R_B^2 + Y^2\right) d\left(q(T+1)+1\right)^{\frac{1}{q}} \log\left(1 + \frac{T}{nrd}\right) \quad (10)$$

$$+ 2R_B r\left(\left(q(T+1)+1\right)^{\frac{q-1}{q}} + 1\right)\sum_t \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \,. \quad (11)$$

Proof. Substituting $\Lambda_i^{-1} = i^{q-1} + 1$ in Lemma 1 we get,

$$r\sum_i^n \left(\Lambda_i^{-1} - 1\right) = r\sum_{i=1}^n i^{q-1} \geq \int_1^n x^{q-1}dx = \frac{1}{q}\left(n^q - 1\right) \,. \quad (12)$$

Setting the last term to $T + 1$ we get an upper bound on n,

$$n \leq \left(q(T+1)+1\right)^{1/q} \quad \Rightarrow \quad \Lambda_n^{-1} \leq \left(q(T+1)+1\right)^{(q-1)/q} + 1 \,. \quad (13)$$

■

Comparing the last two terms of the bound of Corollary 2 we observe a natural tradeoff in the value of q. The third term of (10) is decreasing with large values of q, while the fourth term of (11) is increasing with q.

Assuming a bound on the deviation $\sum_t \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \leq C\,T^{1/p}$. We set $q = 2p/(p+1)$ and get that the sum of (10) and (11) is of order $\mathcal{O}\left(T^{(p+1)/(2p)}\log(T)\right)$. As a consequence we get that, as long as $p > 1$ the sum of (10) and (11) is $o(T)$ and thus is vanishing. Furthermore, when the noise is very low we have $p \approx -(1+\epsilon)$ in this case $q \approx 2 + (2/\epsilon)$, and we get that the algorithm will not make any restarts and retrieve the bound of $\mathcal{O}(\log T)$ for the stationary case. Intuitively, for this choice of q the algorithm will have only one interval, and there will be no restarts.

Intuitively, this schema to set Λ_i balances between the amount of noise (need for many restarts) and the property that using the covariance matrix for updates achieves fast-convergence. We note that an exponential schema $\Lambda_i = 2^{-i}$ will lead to very few restarts, and very small eigenvalues of the covariance matrix. This is because the last segment will be about half the length of the entire sequence.

5 Proof of Theorem 1

We first state the following lemmas, for which we define

$$d_t(\boldsymbol{z},\boldsymbol{v}) = (\boldsymbol{z}-\boldsymbol{v})^\top \Sigma_t^{-1}(\boldsymbol{z}-\boldsymbol{v})\,,\; d_{\tilde{t}}(\boldsymbol{z},\boldsymbol{v}) = (\boldsymbol{z}-\boldsymbol{v})^\top \tilde{\Sigma}_t^{-1}(\boldsymbol{z}-\boldsymbol{v})\,,\; \chi_t = \boldsymbol{x}_t^\top \Sigma_{t-1}\boldsymbol{x}_t\,.$$

Lemma 2. *Let $\tilde{\boldsymbol{w}}_t$ and $\tilde{\Sigma}_t$ be defined in* (6) *and* (7) *in Fig. 1, then,*

$$d_{t-1}(\boldsymbol{w}_{t-1},\boldsymbol{u}_{t-1}) - d_{\tilde{t}}(\tilde{\boldsymbol{w}}_t,\boldsymbol{u}_{t-1}) = \frac{1}{r}\ell_t - \frac{1}{r}g_t - \frac{\ell_t\chi_t}{r\,(r+\chi_t)}\,. \tag{14}$$

The proof follows a chain of algebraic equalities and is omitted due to lack of space.

Lemma 3. *Denote by $\Delta_t = d_{t-1}(\boldsymbol{w}_{t-1},\boldsymbol{u}_{t-1}) - d_t(\boldsymbol{w}_t,\boldsymbol{u}_t)$ then*

$$\begin{aligned}\Delta_t \geq \frac{1}{r}(\ell_t - g_t) - \ell_t\frac{\chi_t}{r(r+\chi_t)} \\ + \boldsymbol{u}_{t-1}^\top\Sigma_{t-1}^{-1}\boldsymbol{u}_{t-1} - \boldsymbol{u}_t^\top\Sigma_t^{-1}\boldsymbol{u}_t - 2R_B\Lambda_i^{-1}\|\boldsymbol{u}_{t-1}-\boldsymbol{u}_t\|\end{aligned} \tag{15}$$

where $i-1$ is the number of restarts performed before example t.

Proof. We write Δ_t as a telescopic sum of four terms as follows,

$$\begin{aligned}\Delta_{t,1} &= d_{t-1}(\boldsymbol{w}_{t-1},\boldsymbol{u}_{t-1}) - d_{\tilde{t}}(\tilde{\boldsymbol{w}}_t,\boldsymbol{u}_{t-1}) & \Delta_{t,2} &= d_{\tilde{t}}(\tilde{\boldsymbol{w}}_t,\boldsymbol{u}_{t-1}) - d_t(\tilde{\boldsymbol{w}}_t,\boldsymbol{u}_{t-1}) \\ \Delta_{t,3} &= d_t(\tilde{\boldsymbol{w}}_t,\boldsymbol{u}_{t-1}) - d_t(\boldsymbol{w}_t,\boldsymbol{u}_{t-1}) & \Delta_{t,4} &= d_t(\boldsymbol{w}_t,\boldsymbol{u}_{t-1}) - d_t(\boldsymbol{w}_t,\boldsymbol{u}_t)\end{aligned}$$

We lower bound each of the four terms. Since, the value of $\Delta_{t,1}$ was computed in Lemma 2 we start with the second term. If no reset occurs then $\Sigma_t = \tilde{\Sigma}_t$ and $\Delta_{t,2} = 0$. Otherwise, we use the facts that $0 \preceq \tilde{\Sigma}_t \preceq I$, and $\Sigma_t = I$ and get,

$$\begin{aligned}\Delta_{t,2} &= (\tilde{\boldsymbol{w}}_t-\boldsymbol{u}_{t-1})^\top\tilde{\Sigma}_t^{-1}(\tilde{\boldsymbol{w}}_t-\boldsymbol{u}_{t-1}) - (\tilde{\boldsymbol{w}}_t-\boldsymbol{u}_{t-1})^\top\Sigma_t^{-1}(\tilde{\boldsymbol{w}}_t-\boldsymbol{u}_{t-1}) \\ &= \mathrm{Tr}\left((\tilde{\boldsymbol{w}}_t-\boldsymbol{u}_{t-1})(\tilde{\boldsymbol{w}}_t-\boldsymbol{u}_{t-1})^\top\left(\tilde{\Sigma}_t^{-1}-\Sigma_t^{-1}\right)\right) \\ &\geq \mathrm{Tr}\left((\tilde{\boldsymbol{w}}_t-\boldsymbol{u}_{t-1})(\tilde{\boldsymbol{w}}_t-\boldsymbol{u}_{t-1})^\top(I-I)\right) = 0\,.\end{aligned}$$

To summarize, $\Delta_{t,2} \geq 0$. We can lower bound $\Delta_{t,3}$ by using the fact that $\boldsymbol{w}_t$ is a projection of $\tilde{\boldsymbol{w}}_t$ onto a closed set (a ball of radius R_B around the origin), which by our assumption contains $\boldsymbol{u}_t$. Employing Corollary 3 of [20] we get, $d_t(\tilde{\boldsymbol{w}}_t,\boldsymbol{u}_{t-1}) \geq d_t(\boldsymbol{w}_t,\boldsymbol{u}_{t-1})$ and thus $\Delta_{t,3} \geq 0$.

Finally, we lower bound the fourth term $\Delta_{t,4}$,

$$\begin{aligned}\Delta_4 &= (\boldsymbol{w}_t-\boldsymbol{u}_{t-1})^\top\Sigma_t^{-1}(\boldsymbol{w}_t-\boldsymbol{u}_{t-1}) - (\boldsymbol{w}_t-\boldsymbol{u}_t)^\top\Sigma_t^{-1}(\boldsymbol{w}_t-\boldsymbol{u}_t) \\ &= \boldsymbol{u}_{t-1}^\top\Sigma_t^{-1}\boldsymbol{u}_{t-1} - \boldsymbol{u}_t^\top\Sigma_t^{-1}\boldsymbol{u}_t - 2\boldsymbol{w}_t^\top\Sigma_t^{-1}(\boldsymbol{u}_{t-1}-\boldsymbol{u}_t)\end{aligned} \tag{16}$$

We use Hölder inequality and then Cauchy-Schwartz inequality to get the following lower bound,

$$\begin{aligned}-2\boldsymbol{w}_t^\top\Sigma_t^{-1}(\boldsymbol{u}_{t-1}-\boldsymbol{u}_t) &= -2\mathrm{Tr}\left(\Sigma_t^{-1}(\boldsymbol{u}_{t-1}-\boldsymbol{u}_t)\boldsymbol{w}_t^\top\right) \\ &\geq -2\lambda_{max}\left(\Sigma_t^{-1}\right)\boldsymbol{w}_t^\top(\boldsymbol{u}_{t-1}-\boldsymbol{u}_t) \geq -2\lambda_{max}\left(\Sigma_t^{-1}\right)\|\boldsymbol{w}_t\|\|\boldsymbol{u}_{t-1}-\boldsymbol{u}_t\|\,.\end{aligned}$$

We use the facts that $\|\boldsymbol{w}_t\| \leq R_B$ and that $\lambda_{max}\left(\Sigma_t^{-1}\right) = 1/\lambda_{min}\left(\Sigma_t\right) \leq \Lambda_i^{-1}$, where i is the current segment index, and get

$$-2\boldsymbol{w}_t^\top \Sigma_t^{-1}\left(\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\right) \geq -2\Lambda_i^{-1} R_B \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \,. \tag{17}$$

Substituting (17) in (16) and using $\Sigma_t \preceq \Sigma_{t-1}$ we get,

$$\begin{aligned}\Delta_{t,4} &\geq \boldsymbol{u}_{t-1}^\top \Sigma_t^{-1} \boldsymbol{u}_{t-1} - \boldsymbol{u}_t^\top \Sigma_t^{-1} \boldsymbol{u}_t - 2R_B \Lambda_i^{-1} \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \\ &\geq \boldsymbol{u}_{t-1}^\top \Sigma_{t-1}^{-1} \boldsymbol{u}_{t-1} - \boldsymbol{u}_t^\top \Sigma_t^{-1} \boldsymbol{u}_t - 2R_B \Lambda_i^{-1} \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \,. \end{aligned} \tag{18}$$

Combining (18) with Lemma 2 concludes the proof. ■

Lemma 4. *During the runtime of the algorithm of Fig. 1 we have*

$$\sum_{t=t_i}^{t_i+T_i} \frac{\chi_t}{(\chi_t + r)} \leq \log\left(\det\left(\Sigma_{t_{i+1}-1}^{-1}\right)\right) = \log\left(\det\left(\left(\Sigma^i\right)^{-1}\right)\right) . \tag{19}$$

We remind the reader that: (1) t_i is the first example index after the ith restart (2) T_i is the number of examples observed before the next restart (3) the notation $\Sigma^i = \Sigma_{t_{i+1}-1}$ is the covariance matrix just before the next restart.

The proof of the lemma is similar to the proof of Lemma 4 [9] and thus omitted. We now turn to prove Theorem 1,

Proof. We bound the sum $\sum_t \Delta_t$ from above and below, and start with an upper bound using the property of telescopic sum and get,

$$\sum_t \Delta_t = \left(d_0\left(\boldsymbol{w}_0, \boldsymbol{u}_0\right) - d_T\left(\boldsymbol{w}_T, \boldsymbol{u}_T\right)\right) \leq d_0\left(\boldsymbol{w}_0, \boldsymbol{u}_0\right) \tag{20}$$

We compute a lower bound by applying Lemma 3 and get,

$$\begin{aligned}\sum_t \Delta_t \geq \sum_t \Bigg(&\frac{1}{r}\left(\ell_t - g_t\right) - \ell_t \frac{\chi_t}{r(r+\chi_t)} \\ &+ \boldsymbol{u}_{t-1}^\top \Sigma_{t-1}^{-1} \boldsymbol{u}_{t-1} - \boldsymbol{u}_t^\top \Sigma_t^{-1} \boldsymbol{u}_t - 2R_B \Lambda_{i(t)}^{-1} \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \Bigg) \,,\end{aligned}$$

where $i(t)$ is the number of restarts occurred before observing the tth example. We continue to develop the last equation and get,

$$\begin{aligned}\sum_t \Delta_t \geq &\frac{1}{r}\sum_t \ell_t - \frac{1}{r}\sum_t g_t - \sum_t \ell_t \frac{\chi_t}{r(r+\chi_t)} \\ &+ \sum_t \left(\boldsymbol{u}_{t-1}^\top \Sigma_{t-1}^{-1} \boldsymbol{u}_{t-1} - \boldsymbol{u}_t^\top \Sigma_t^{-1} \boldsymbol{u}_t\right) - \sum_t 2R_B \Lambda_{i(t)}^{-1} \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \\ = &\frac{1}{r}\sum_t \ell_t - \frac{1}{r}\sum_t g_t - \sum_t \ell_t \frac{\chi_t}{r(r+\chi_t)} \\ &+ \boldsymbol{u}_0^\top \Sigma_0^{-1} \boldsymbol{u}_0 - \boldsymbol{u}_T^\top \Sigma_T^{-1} \boldsymbol{u}_T - 2R_B \sum_t \Lambda_{i(t)}^{-1} \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \,. \end{aligned} \tag{21}$$

Combining (20) with (21) and using $d_0\left(\boldsymbol{w}_0, \boldsymbol{u}_0\right) = \boldsymbol{u}_0^\top \Sigma_0^{-1} \boldsymbol{u}_0$ ($\boldsymbol{w}_0 = \mathbf{0}$) we get,

$$\frac{1}{r}\sum_t \ell_t - \frac{1}{r}\sum_t g_t - \sum_t \ell_t \frac{\chi_t}{r(r+\chi_t)} - \boldsymbol{u}_T^\top \Sigma_T^{-1} \boldsymbol{u}_T \\ - 2R_B \sum_t \Lambda_{i(t)}^{-1} \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \leq 0 \,.$$

Rearranging the terms we get,

$$\sum_t \ell_t \leq \sum_t g_t + \sum_t \ell_t \frac{\chi_t}{r+\chi_t} + r\boldsymbol{u}_T^\top \Sigma_T^{-1} \boldsymbol{u}_T + 2R_B r \sum_t \frac{1}{\Lambda_{i(t)}} \|\boldsymbol{u}_{t-1} - \boldsymbol{u}_t\| \quad (22)$$

Since $\|\boldsymbol{w}_t\| \leq R_B$, (and we assume that) $\|\boldsymbol{x}_t\| = 1$ and $\sup_t |y_t| = Y$, we get that $\sup_t \ell_t \leq 2(R_B^2 + Y^2)$. The second term in the right-hand-side is bounded using the last inequality and Lemma 4,

$$\begin{aligned} \sum_t \ell_t \frac{\chi_t}{r+\chi_t} &= \sum_i^n \sum_{t=t_i}^{t_i+T_i} \ell_t \frac{\chi_t}{r+\chi_t} \\ &\leq \sum_i^n \left(\sup_t \ell_t\right) \log\det\left(\left(\Sigma^i\right)^{-1}\right) \\ &\leq 2\left(R_B^2 + Y^2\right) \sum_i^n \log\det\left(\left(\Sigma^i\right)^{-1}\right) \,. \end{aligned} \quad (23)$$

■

To conclude, we showed that if the algorithm is given an upper bound on the amount of drift, which is sub-linear in T, it can achieve sub-linear regret. Furthermore, if it is known that there is no non-stationarity in the reference vectors, then running the algorithm with $q = \infty$ will have a regret logarithmic in T. We use this property in the next section, where we describe a version of the algorithm when such a bound is not known, or is very loose.

6 Algorithm for Unknown Amount of Drift

We sketch now an algorithm, which does not assuming knowing the exact drift level, yet achieves $\log(T)$ regret in the stationary case, and sub-linear regret otherwise

In a nutshell, the algorithm runs $C+1$ copies of ARCOR, one with $q = \infty$ (no reset) and the others with $q = 1, 2, 3...C$. On each iteration the input vector $\boldsymbol{x}_t$ is fed into the $C+1$ copies each of which computes a prediction $\hat{y}_{t,c}$. These $C+1$ predictions are collected into a vector in $\mathbb{R}^{C+1}$. This vector is then fed into another copy of the algorithm which is run with $q = \infty$ and $R_B = 1$. Denote its weight vector by $\boldsymbol{v}_t \in \mathbb{R}^{C+1}$. The output of our algorithm is thus $\hat{y}_t = \boldsymbol{v}_{t-1}^\top \hat{\boldsymbol{y}}_t$. Given the feedback, the algorithm updates all $C+1$ copies using ARCOR, as well as the additional copy.

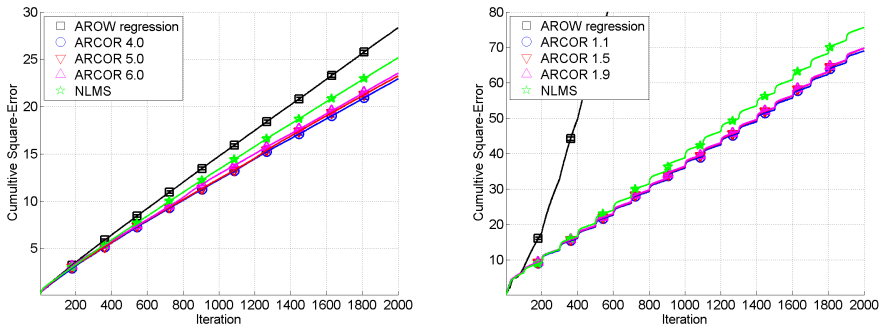

Fig. 2. Cumulative squared loss for AROW for regression, ARCOR with few value of q and NLMS vs iteration. Left panel shows result for dataset with no shifts and the right panel for dataset with shifts.

Conceptually, we position $C + 2$ copies of the algorithm in a network of depth 2. The first layer is composed of $C + 1$ nodes, each runs its own copy of the algorithm under a specific assumption of drift value (as above). The outputs if the first layer are fed into the second layer, that integrates them linearly into a final output.

Intuitively, since the value of $\boldsymbol{v}_t$ can be any member of the standard basis (ie $(0..1..0)$, we get that the loss of the additional copy is bounded with the loss of the best copy with additional $\log(T)$ term (the regret for the stationary case). Thus if the best copy is the one with $q = \infty$ (ARCOR for the stationary case), the total regret is logarithmic in T. Otherwise, in the non-stationary case, the regret of the best copy would be polynomial in T, which is the final regret. We just sketched the proof of the following theorem,

Theorem 2. *Assuming the algorithm just presented is run with $C + 1$ copies of ARCOR and the additional copy. Then, the total loss of the algorithm is bounded by $\sum_t (\hat{y}_t - y_t)^2 \leq \min_{c=1}^{C+1} \sum_t (\hat{y}_{c,t} - y_t)^2 + D \log(T)$,where D is a constant depending on the number of copies C and the parameter used r.*

We note that Theorem 1 or one of its corollaries can be used to bound each of the terms $\sum_t^T (\hat{y}_{c,t} - y_t)^2$. Specifically, if the minima is obtained for the stationary case we get a logarithmic regret all together, otherwise the polynomial term is dominant in the bound.

7 Simulations

We illustrate the algorithms with two synthetic datasets, one with drifting only, and the other also with switching. We generated $2,000$ points in $\mathbb{R}^{20}$ where the first ten coordinates were grouped into five groups of size two. Each such pair was drawn from a 45° rotated Gaussian distribution with standard deviation 10 and 1. The remaining 10 coordinates were drawn from independent Gaussian distributions $\mathcal{N}(0, 2)$. The first dataset was generated using a sequence of vectors $\boldsymbol{u}_t \in \mathbb{R}^{20}$ for which the only non-zero coordinates are the first two. This vector in $\mathbb{R}^2$ is of unit norm $\|\boldsymbol{u}_t\| = 1$ and rotating in a rate of $t^{-0.01}$.

Similarly, the second dataset was generated with a sequence of rate $t^{-0.5}$, but with one additional twist. Every 50 time-steps the two-dimensional vector defined above was "embedded" in different pair of coordinates of the reference vector $\boldsymbol{u}_t$, for the first 50 steps it were coordinates $1, 2$ in the next 50 examples, coordinates $3, 4$ and so on. This change causes a switch in the reference vector $\boldsymbol{u}_t$. Finally, the target label y_t was set to be $\boldsymbol{x}_t^\top \boldsymbol{u}_t + \xi_t$ where $\xi_t \sim \mathcal{N}(0, 2)$.

Three algorithms are evaluated: NLMS (normalized least mean square) [2, 21] which is a state-of-the-art first-order algorithm, AROW for regression, as developed in Sec. 3 with no restarting nor projection and ARCOR for various value of q. All algorithms have one parameter to tune, which was performed using a single random sequence. We repeat each experiment 100 reporting the mean cumulative square-loss and 95% confidence interval. We note that Aggregating Algorithm (AA) and Ridge Regression(RR) algorithm are very close algorithmically and in performance to AROW for regression and thus omitted.

The results are summarized in Fig. 2. In a nutshell, AROW for regression performs worst, NLMS is second and ARCOR is the best. This is surprising, as AROW for classification outperforms many algorithms that are related in spirit to NLMS. Yet, as mentioned above, the algorithm drives its learning rate to zero, not allowing for the ability to track drifting concepts. For both datasets, and mainly for the one with switching (right panel), AROW for regression is sensitive to the non-stationary properties of the data, and thus suffers very large loss, as its tracking ability is very slow. NLMS has nice tracking properties, but its learning rate is relatively slow. ARCOR tracks as fast as AROW, yet it bounds the learning rate and thus allows fast tracking rate. Note that in both datasets the "gap" between the cumulative error of all algorithms increases with time, this means that ARCOR tracks better both on drifting and switching settings.

8 Related Work and Summary

There is a large body of research in online learning for regression problems. Almost half a century ago, Widrow and Hoff [28] developed a variant of the least mean squares (LMS) algorithm for adaptive filetering and noise reduction. The algorithm was further developed and analyzed extensively (see e.g. [12]). The normalized least mean squares filter (NLMS) [2, 21] builds on LMS and performs better to scaling of the input. The recursive least squares (RLS) [19] is the closest to our algorithm in the signal processing literature, it also maintains a weight-vector and a covariance-like positive semi-definite (PSD) matrix used to re-weight the input.

In the machine learning literature the problem of online regression was studied extensively, and clearly we can not cover all the relevant work. Cesa-Bianchi et al. [4] studied gradient descent based algorithms for regression with the squared loss. Kivinen and Warmuth [22] proposed various generalizations for general regularization functions. We refer the reader to a comprehensive book in the subject [6].

Foster [14] studied an online version of the ridge regression algorithm in the worst-case setting. Vovk [18] proposed a related algorithm called the Aggregating Algorithm (AA), and later Forster [13] improved its regret analysis for the square loss. Both algorithms employ second order information. ARCOR for the separable case is very similar

to these algorithms, although has alternative derivation. Recently, few algorithms were proposed either for classification [5, 10, 8, 9] or for general loss functions [11, 25] in the online convex programming framework. Our work shares the same design principles of AROW [9] yet it is aimed for regression. Furthermore, it has two important modifications which makes it work in the drifting or shifting setting. These modifications make the analysis more complex than of AROW.

Two of the approaches used in previous algorithms for non-stationary setting are to bound the weight vector and covariance reset. Bounding the weight vector was performed either by projecting it into a bonded set [20], shrinking it by multiplication [23] or subtracting previously seen examples [3]. These three methods (or at least most of their variants) can be combined with kernel operators, and in fact, the last two approaches were designed and motivated by kernels.

The Covariance Reset RLS algorithm (CR-RLS) [26, 17, 7] was designed for adaptive filtering. CR-RLS makes covariance reset every fixed amount of data points, while ARCOR performs restarts based on the actual properties of the data: the eigenspectrum of the covariance matrix. Furthermore, as far as we know, there is no analysis in the mistake bound model for this algorithm. Both ARCOR and CR-RLS are motivated from the property that the covariance matrix goes to zero and becomes rank deficient.

In both algorithms the information encapsulated in the covariance matrix is lost after restarts. In a rapidly varying environments, like a wireless channel, this loss of memory can be beneficial, as previous contributions to the covariance matrix may have little correlation with the current structure. Recent versions of RLS+CR [16, 27] employ covariance reset to have numerically stable computations.

Our work combines both techniques with online learning with second order algorithm for regression. In this aspect we have the best of all worlds, fast convergence rate due to the usage of second order information, and the ability to adapt in non-stationary environments due to projection and resets. Current work includes extending the algorithm for general loss function, efficient implementation of the algorithm and automatically detecting the level of non-stationarity.

Acknowledgments: This research was supported in part by the Israeli Science Foundation grant ISF-1567/10 any by the European Unions Seventh Framework Programme (FP7/2007-2013) under PASCAL2 (PUMP PRIMING). KC is a Horev Fellow, supported by the Taub Foundations.

References

1. Peter Auer and Manfred K. Warmuth. Tracking the best disjunction. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(70), 2000.
2. N. J. Bershad. Analysis of the normalized lms algorithm with gaussian inputs. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4):793–806, 1986.
3. 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.
4. Nicolo Ceas-Bianchi, Philip M. Long, and Manfred K. Warmuth. Worst case quadratic loss bounds for on-line prediction of linear functions by gradient descent. Technical Report IR-418, University of California, Santa Cruz, CA, USA, 1993.

5. Nicoló Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *Siam Journal of Commutation*, 34(3):640–668, 2005.
6. Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.
7. Min-Shin Chen and Jia-Yush Yen. Application of the least squares algorithm to the observer design for linear time-varying systems. *Automatic Control, IEEE Transactions on*, 44(9):1742 –1745, sep 1999.
8. K. Crammer, M. Dredze, and F. Pereira. Exact confidence-weighted learning. In *NIPS 22*, 2008.
9. K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weighted vectors. In *Advances in Neural Information Processing Systems 23*, 2009.
10. M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *ICML*, 2008.
11. John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT*, pages 257–269, 2010.
12. A. Feuer and E. Weinstein. Convergence analysis of lms filters with uncorrelated gaussian data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(1):222–230, 1985.
13. Jurgen Forster. On relative loss bounds in generalized linear regression. In *Fundamentals of Computation Theory (FCT)*, 1999.
14. Dean P. Foster. Prediction in the worst case. *The Annals of Statistics*, 19(2):1084–1090, 1991.
15. Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
16. S.G. Goodhart, K.J. Burnham, and D.J.G. James". Logical covariance matrix reset in self-tuning control. *Mechatronics*, 1(3):339 – 351, 1991.
17. G.C. Goodwin, E.K. Teoh, and H. Elliott. Deterministic convergence of a self-tuning regulator with covariance resetting. *Control Theory and App., IEE Proc. D*, 130(1):6 –8, 83.
18. Volodimir G.Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.
19. Monson H. Hayes. 9.4: Recursive least squares. In *Statistical Digital Signal Processing and Modeling*, page 541, 1996.
20. Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.
21. R. R. itmead and B. D. O. Anderson. Performance of adaptive estimation algorithms in dependent random environments. *IEEE Transactions on Automatic Control*, 25:788–794, 1980.
22. Jyrki Kivinen and Manfred K.Warmuth. Exponential gradient versus gradient descent for linear predictors. *Information and Computation*, 132:132–163, 1997.
23. Jyrki Kivinen, Alex J. Smola, and Robert C. Williamson. Online learning with kernels. In *NIPS*, pages 785–792, 2001.
24. Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.
25. H. Brendan McMahan and Matthew J. Streeter. Adaptive bound optimization for online convex optimization. In *COLT*, pages 244–256, 2010.
26. Mario E. Salgado, Graham C. Goodwin, and Richard H. Middleton. Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control*, 47(2):477 –491, 1988.
27. Hong-Seok Song, Kwanghee Nam, and P. Mutschler. Very fast phase angle estimation algorithm for a single-phase system having sudden phase angle jumps. In *Industry Applications Conference. 37th IAS Annual Meeting*, volume 2, pages 925 – 931, 2002.
28. B. Widrow and Jr. M.E. Hoff. Adaptive switching circuits. 1960.