Theoretical Computer Science 558 (2014) 107–124

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Weighted last-step min–max algorithm with improved sub-logarithmic regret

Edward Moroshko *, Koby Crammer

Department of Electrical Engineering, Technion, Israel

ARTICLE INFO

Available online 21 September 2014

Keywords:
Online learning
Regression
Min–max learning

ABSTRACT

In online learning the performance of an algorithm is typically compared to the performance of a fixed function from some class, with a quantity called regret. Forster [12] proposed a last-step min–max algorithm which was somewhat simpler than the algorithm of Vovk [26], yet with the same regret. In fact the algorithm he analyzed assumed that the choices of the adversary are bounded, yielding artificially only the two extreme cases. We fix this problem by weighing the examples in such a way that the min–max problem will be well defined, and provide analysis with logarithmic regret that may have better multiplicative factor than both bounds of Forster [12] and Vovk [26]. We also derive a new bound that may be sub-logarithmic, as a recent bound of Orabona et al. [21], but may have better multiplicative factor. Finally, we analyze the algorithm in a weak-type of non-stationary setting, and show a bound that is sublinear if the non-stationarity is sub-linear as well.

1. Introduction

We consider the online learning regression problem, in which a learning algorithm tries to predict real numbers in a sequence of rounds given some side-information or inputs $\mathbf{x}_t \in \mathbb{R}^d$. Real-world example applications for these algorithms are weather or stockmarket predictions. The goal of the algorithm is to have a small discrepancy between its predictions and the associated outcomes $y_t \in \mathbb{R}$. This discrepancy is measured with a loss function, such as the square loss. It is common to evaluate algorithms by their regret, the difference between the cumulative loss of an algorithm with the cumulative loss of any function taken from some class.

Forster [12] proposed a last-step min–max algorithm for online regression that makes a prediction assuming it is the last example to be observed, and the goal of the algorithm is indeed to minimize the regret with respect to linear functions. The resulting optimization problem he obtained was convex in the choice of the algorithm and the choice of the adversary, yielding an unbounded optimization problem. Forster circumvented this problem by assuming a bound Y over the choices of the adversary that should be known to the algorithm, yet his analysis is for the version with no bound.

We propose a modified last-step min–max algorithm with weights over examples, that are controlled in a way to obtain a problem that is concave over the choices of the adversary and convex over the choices of the algorithm. We analyze our algorithm and show a logarithmic-regret that may have a better multiplicative factor than the analysis of Forster. We derive additional analysis that is logarithmic in the loss of the reference function, rather than the number of rounds T. This behavior was recently given by Orabona et al. [21] for a certain online gradient descent algorithm. Yet, their bound [21] has

* Corresponding author.
E-mail addresses: edward.moroshko@gmail.com (E. Moroshko), koby@ee.technion.ac.il (K. Crammer).

http://dx.doi.org/10.1016/j.tcs.2014.09.028
0304-3975/

a similar multiplicative factor to that of Forster [12], while our bound has a potentially better multiplicative factor and it has the same dependence on the cumulative loss of the reference function as Orabona et al. [21]. Additionally, our algorithm and analysis are totally free of assuming the bound Y or knowing its value.

Competing with the best *single* function might not suffice for some problems. In many real-world applications, the true target function is not fixed, but may change from time to time. We bound the performance of our algorithm also in non-stationary environment, where we measure the complexity of the non-stationary environment by the total deviation of a collection of linear functions from some fixed reference point. We show that our algorithm maintains an average loss close to that of the best sequence of functions, as long as the total of this deviation is sublinear in the number of rounds T.

A short version appeared in The 23rd International Conference on Algorithmic Learning Theory (ALT 2012). This journal version of the paper includes additionally: (1) Recursive form of the algorithm and comparison to other algorithms of the same form (Section 3.1). (2) Kernel version of the algorithm (Section 3.2). (3) MAP (maximum a posteriori) interpretation of the minimization problems (Remark 1 and Remark 2). (4) All proofs and extended related-work section.

2. Problem setting

We work in the online setting for regression evaluated with the squared loss. Online algorithms work in rounds or iterations. On each iteration an online algorithm receives an instance $\mathbf{x}_t \in \mathbb{R}^d$ and predicts a real value $\hat{y}_t \in \mathbb{R}$, it then receives a label $y_t \in \mathbb{R}$, possibly chosen by an adversary, suffers loss $\ell_t(\text{alg}) = \ell(y_t, \hat{y}_t) = (\hat{y}_t - y_t)^2$, updates its prediction rule, and proceeds to the next round. We assume later that the inputs $\mathbf{x}_t$ are bounded, but not the labels y_t (see also the end of Section 4 for comparison of various settings). The cumulative loss suffered by the algorithm over T iterations is,

$$L_T(\text{alg}) = \sum_{t=1}^{T} \ell_t(\text{alg}). \tag{1}$$

The goal of the algorithm is to perform well compared to any predictor from some function class.

A common choice is to compare the performance of an algorithm with respect to *a single* function, or specifically a single linear function, $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{u}$, parameterized by a vector $\mathbf{u} \in \mathbb{R}^d$. Denote by $\ell_t(\mathbf{u}) = (\mathbf{x}_t^\top \mathbf{u} - y_t)^2$ the instantaneous loss of a vector $\mathbf{u}$, and by $L_T(\mathbf{u}) = \sum_t^T \ell_t(\mathbf{u})$. The regret with respect to $\mathbf{u}$ is defined to be,

$$R_T(\mathbf{u}) = \sum_{t}^{T} (y_t - \hat{y}_t)^2 - L_T(\mathbf{u}).$$

A desired goal of the algorithm is to have $R_T(\mathbf{u}) = o(T)$, that is, the average loss suffered by the algorithm will converge to the average loss of the best linear function $\mathbf{u}$.

Below in Section 5 we will also consider an extension of this form of regret, and evaluate the performance of an algorithm against some T-tuple of functions, $(\mathbf{u}_1, \ldots, \mathbf{u}_T) \in \mathbb{R}^d \times \cdots \times \mathbb{R}^d$,

$$R_T(\mathbf{u}_1, \ldots, \mathbf{u}_T) = \sum_{t}^{T} (y_t - \hat{y}_t)^2 - L_T(\mathbf{u}_1, \ldots, \mathbf{u}_T),$$

where $L_T(\mathbf{u}_1, \ldots, \mathbf{u}_T) = \sum_t^T \ell_t(\mathbf{u}_t)$. Clearly, with no restriction of the T-tuple, any algorithm may suffer a regret linear in T, as one can set $\mathbf{u}_t = \mathbf{x}_t(y_t/\|\mathbf{x}_t\|^2)$, and suffer zero quadratic loss in all rounds. Thus, we restrict below the possible choices of T-tuple either explicitly, or implicitly via some regularization.

Throughout the paper we use $\|\cdot\|$ for the ℓ_2-norm and $|\cdot|$ for the determinant.

3. A last step min–max algorithm

Our algorithm is derived based on a last-step min–max prediction, proposed by Forster [12] and Takimoto and Warmuth [24]. See also the work of Azoury and Warmuth [1]. An algorithm following this approach outputs the min–max prediction assuming the current iteration is the last one. The algorithm we describe below is based on an extension of this notion. For this purpose we introduce a weighted cumulative loss using positive input-dependent weights $\{a_t\}_{t=1}^T$,

$$L_T^{\boldsymbol{a}}(\mathbf{u}) = \sum_{t=1}^{T} a_t \left(y_t - \mathbf{u}^\top \mathbf{x}_t\right)^2, \qquad L_T^{\boldsymbol{a}}(\mathbf{u}_1, \ldots, \mathbf{u}_T) = \sum_{t=1}^{T} a_t \left(y_t - \mathbf{u}_t^\top \mathbf{x}_t\right)^2.$$

The exact values of the weights a_t will be defined below to suit a convexity property. We will see that a_t depends on $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$, but can be computed with a fixed computational cost.

Our variant of the last step min–max algorithm predicts,[1]

[1] y_T and $\hat{y}_T$ serves both as quantifiers (over the max and min operators, respectively), and as the optimal values over this optimization problem.