COMPUTATIONALLY FEASIBLE ONLINE SECOND ORDER TIME SERIES PREDICTOR

Burak C. Civek and Suleyman S. Kozat, Senior Member, IEEE

Department of Electrical and Electronics Engineering, Bilkent University

ABSTRACT

We investigate the problem of sequential linear prediction for real life big data applications. The second order algorithms, i.e., Newton-Raphson Methods, asymptotically achieve the performance of the "best" possible linear predictor much faster compared to the first order algorithms, e.g., Online Gradient Descent. However, implementation of these methods is not usually feasible in big data applications because of the extremely high computational needs. To this end, we introduce a highly efficient implementation reducing the computational complexity of the second order methods from quadratic to linear scale. We do not rely on any statistical assumptions, hence, lose no information. We demonstrate the computational efficiency of our algorithm on a real life sequential big dataset.

Index Terms— Second order, efficient, time series prediction.

1. INTRODUCTION

We investigate the widely studied sequential prediction problem for high dimensional data streams. Unfortunately, conventional methods in machine learning and signal processing literatures are inadequate to efficiently and effectively process high dimensional data sequences [1, 2, 3]. Even though today's computers have powerful processing units, traditional algorithms creates a bottleneck even for that processing power when the data is acquired at high speeds and too large in size [1, 2]. These problems bring an essential requirement for an algorithm that is both computationally feasible and highly effective in terms of performance.

In order to mitigate the problem of excessive computational cost, we introduce sequential, i.e., online, processing, where the data is neither stored nor reused, and avoid "batch" processing [3, 4]. The first order methods, e.g., Online Gradient Descent, are one of the well known online learning algorithms in the signal processing and the machine learning literatures [5, 6]. These methods offer a computational complexity of $O(M)$ for a sequence of M-dimensional feature vectors $\{\boldsymbol{x}_t\}_{t\geq 0}$, where $\boldsymbol{x}_t \in \mathbb{R}^M$. However, their convergence rate remains significantly slow when achieving an optimal solution [3, 6, 7]. Another well known family of algorithms is the second order methods, e.g., Online Newton Step [5]. They asymptotically achieve the performance of the "best" possible predictor much faster and outperforms the first order methods in terms of convergence rate and steady state error performances [3, 4, 6, 8]. However, the second order methods offer a quadratic increase in the computational complexity, i.e., $O(M^2)$. Hence, it is not usually feasible for real-life big data applications to utilize the merits of the second order algorithms [9].

Overall, in this paper, we introduce an online sequential prediction algorithm that $i)$ process only the currently available data without any storage, $ii)$ efficiently implements the Newton-Raphson methods, i.e., the second order methods $iii)$ outperforms the gradient based methods in terms of performance, $iv)$ has $O(M)$ computational complexity same as the first order methods and $v)$ requires no statistical assumptions on the data sequence. We illustrate the outstanding gains of our algorithm in terms of computational efficiency using two sequential real life big datasets and compare the resulting error performances with the regular Newton-Raphson methods.

2. PROBLEM DESCRIPTION

In this paper, all vectors are real valued and column-vectors. We use lower case (upper case) boldface letters to represent vectors (matrices). The ordinary transpose is denoted as $\boldsymbol{x}^T$ for the vector $\boldsymbol{x}$. The identity matrix is represented by $\boldsymbol{I}_M$, where the subscript is used to indicate that the dimension is $M \times M$. We denote the M-dimensional zero vector as $\mathbf{0}_M$.

We study sequential prediction, where we sequentially observe a real valued data sequence $\{x_t\}_{t\geq 0}$, $x_t \in \mathbb{R}$. At each time t, after observing $\{x_t, x_{t-1}, ..., x_{t-M+1}\}$, we generate an estimate of the desired data, $\hat{x}_{t+1} \in \mathbb{R}$, using a linear model as

$$\hat{x}_{t+1} = \boldsymbol{w}_t^T \boldsymbol{x}_t + c_t, \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^M$ represents the feature vector of previous M samples, i.e., $\boldsymbol{x}_t = [x_t, x_{t-1}, ..., x_{t-M+1}]^T$. Here, $\boldsymbol{w}_t \in \mathbb{R}^M$ and $c_t \in \mathbb{R}$ are the corresponding weight vector and the offset variable respectively at time t. With an abuse of notation, we combine the weight vector $\boldsymbol{w}_t$ with the offset variable c_t, and denote it by $\boldsymbol{w}_t = [\boldsymbol{w}_t; c_t]$, yielding $\hat{x}_{t+1} = \boldsymbol{w}_t^T \boldsymbol{x}_t$, where $\boldsymbol{x}_t = [\boldsymbol{x}_t; 1]$. The prediction error at each time instant is given by $e_t = x_t - \hat{x}_t$.

We use the second order Online Newton Step (ONS) algorithm to adaptively learn the weight vector coefficients, i.e.,

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \frac{1}{\mu}\boldsymbol{A}_t^{-1}\nabla_t, \tag{2}$$

where $\mu \in \mathbb{R}$ is the step size and $\nabla_t \in \mathbb{R}^M$ corresponds to the gradient of the cost function $\ell_t(\boldsymbol{w}_t)$ at time t w.r.t. $\boldsymbol{w}_t$, i.e., $\nabla_t \triangleq \nabla\ell_t(\boldsymbol{w}_t)$. Here, the $M \times M$ dimensional matrix $\boldsymbol{A}_t$ is given by

$$\boldsymbol{A}_t = \sum_{i=0}^{t} \nabla_i \nabla_i^T + \alpha \boldsymbol{I}_M, \tag{3}$$

where $\alpha > 0$ is chosen to guarantee that $\boldsymbol{A}_t$ is positive definite, i.e., $\boldsymbol{A}_t > 0$, and hence, invertible. Selection of the parameters μ and α is crucial for good performance [5]. Note that (3) has a recursive structure, i.e.,

$$\boldsymbol{A}_t = \boldsymbol{A}_{t-1} + \nabla_t \nabla_t^T, \tag{4}$$

with an initial value of $\boldsymbol{A}_{-1} = \alpha \boldsymbol{I}_M$. Hence, instead of performing an inverse operation at each iteration, we get a straight update from $\boldsymbol{A}_{t-1}^{-1}$ to $\boldsymbol{A}_t^{-1}$ using the matrix inversion lemma [10]

$$\boldsymbol{A}_t^{-1} = \boldsymbol{A}_{t-1}^{-1} - \frac{\boldsymbol{A}_{t-1}^{-1}\nabla_t\nabla_t^T\boldsymbol{A}_{t-1}^{-1}}{1 + \nabla_t^T\boldsymbol{A}_{t-1}^{-1}\nabla_t}. \tag{5}$$

Then, the computational complexity for computing the matrix $\boldsymbol{A}_t^{-1}$ is reduced to the order of $O(M^2)$. Multiplying both sides of (5) with ∇_t and using in (2) yields

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \frac{1}{\mu}\left[\frac{\boldsymbol{A}_{t-1}^{-1}\nabla_t}{1 + \nabla_t^T\boldsymbol{A}_{t-1}^{-1}\nabla_t}\right]. \tag{6}$$

Although the second order update algorithms provide faster convergence rates and better steady state performances, computational complexity issue prohibits their usage in most real life applications [6, 8, 10]. Since each update in (5) requires the multiplication of an $M \times M$ dimensional matrix with an M dimensional vector for $\boldsymbol{x}_t \in \mathbb{R}^M$, the computational complexity is in the order of $O(M^2)$, while the first order algorithms just need $O(M)$ multiplication and addition. In the next section, we introduce a sequential prediction algorithm, which achieves the performance of the Newton-Raphson methods, while offering $O(M)$ computational complexity same as the first order methods.

3. HIGHLY EFFICIENT ONLINE SEQUENTIAL PREDICTOR

We work on time series data sequences, which directly implies that the feature vectors $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$ are highly related. More precisely, we have the following relation between these two consecutive vectors as

$$[x_{t+1}, \boldsymbol{x}_t^T] = [\boldsymbol{x}_{t+1}^T, x_{t-M+1}]. \tag{7}$$

This relation shows that consecutive data vectors carry quite the same information, which is the basis of our algorithm. We use the instantaneous absolute loss, which is defined as

$$\ell_t(\boldsymbol{w}_t) = \|\hat{x}_{t+1} - \boldsymbol{w}_t^T\boldsymbol{x}_t\|. \tag{8}$$

We resolve the differentiability issue of absolute loss by setting a threshold ϵ close to zero and not updating the weight vector when the absolute error is below this threshold, $\|e_t\| < \epsilon$. From (5) and (6), the absolute loss results in the following update rules for $\boldsymbol{w}_t$ and $\boldsymbol{A}_t^{-1}$,

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} \pm \frac{1}{\mu}\left[\frac{\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t}{1 + \boldsymbol{x}_t^T\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t}\right], \tag{9}$$

$$\boldsymbol{A}_t^{-1} = \boldsymbol{A}_{t-1}^{-1} - \frac{\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t\boldsymbol{x}_t^T\boldsymbol{A}_{t-1}^{-1}}{1 + \boldsymbol{x}_t^T\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t}, \tag{10}$$

since $\nabla_t = \pm\boldsymbol{x}_t$ depending on the sign of the error.

It is clear that the complexity of the second order algorithms essentially results from the matrix-vector multiplication, $\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t$ as in (9). Rather than getting matrix $\boldsymbol{A}_{t-1}^{-1}$ from $\boldsymbol{A}_{t-2}^{-1}$ and then calculating the multiplication $\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t$ individually at each iteration, we develop a direct and compact update rule, which calculates $\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t$ from $\boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}$ without any explicit knowledge of the $M \times M$ dimensional matrix $\boldsymbol{A}_{t-1}^{-1}$.

Similar to [10], we first define the normalization term of the update rule given in (9) as

$$\eta_t = 1 + \boldsymbol{x}_t^T\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t. \tag{11}$$

Then, the difference between the consecutive terms η_t and η_{t-1} is given by

$$\eta_t - \eta_{t-1} = \boldsymbol{x}_t^T\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t - \boldsymbol{x}_{t-1}^T\boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}. \tag{12}$$

We define the $(M+1) \times 1$ dimensional extended vector $\tilde{\boldsymbol{x}}_t = [x_t, \boldsymbol{x}_{t-1}^T]^T$ and get the following two equalities using the relation given in (7),

$$\eta_t = 1 + \tilde{\boldsymbol{x}}_t^T \begin{bmatrix} \boldsymbol{A}_{t-1}^{-1} & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & 0 \end{bmatrix} \tilde{\boldsymbol{x}}_t, \tag{13}$$

$$\eta_{t-1} = 1 + \tilde{\boldsymbol{x}}_t^T \begin{bmatrix} 0 & \boldsymbol{0}_M^T \\ \boldsymbol{0}_M & \boldsymbol{A}_{t-2}^{-1} \end{bmatrix} \tilde{\boldsymbol{x}}_t. \tag{14}$$

Therefore, (12) becomes

$$\eta_t - \eta_{t-1} = \tilde{\boldsymbol{x}}_t^T\Delta_{t-1}\tilde{\boldsymbol{x}}_t, \tag{15}$$

where the update term Δ_{t-1} is defined as

$$\Delta_{t-1} \triangleq \begin{bmatrix} \boldsymbol{A}_{t-1}^{-1} & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \boldsymbol{0}_M^T \\ \boldsymbol{0}_M & \boldsymbol{A}_{t-2}^{-1} \end{bmatrix}. \tag{16}$$

This equation implies that we do not need the exact values of $\boldsymbol{A}_{t-1}^{-1}$ and $\boldsymbol{A}_{t-2}^{-1}$ individually and it is sufficient to know the value of the defined difference Δ_{t-1} for the calculation of η_t. Moreover, we observe that the update term can be expressed in terms of rank 2 matrices, which is the key point for the reduction of complexity.

Initially, we assume that $x_t = 0$ for $t < 0$, which directly implies $\boldsymbol{A}_{-1}^{-1} = \boldsymbol{A}_{-2}^{-1} = \frac{1}{\alpha}\boldsymbol{I}_M$ using (3) and (4). Therefore, Δ_{-1} is found as

$$\Delta_{-1} = \frac{1}{\alpha}\,\mathrm{diag}\{1, 0, \ldots, 0, -1\}. \tag{17}$$

At this point, we define the $(M+1)\times 2$ dimensional matrix Λ_{-1} and the 2×2 dimensional matrix Π_{-1} as

$$\Lambda_{-1} = \sqrt{\frac{1}{\alpha}}\begin{bmatrix}1 & 0 & \ldots & 0 & 0\\ 0 & 0 & \ldots & 0 & 1\end{bmatrix}^T, \Pi_{-1} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}, \tag{18}$$

to achieve the equality given by

$$\Delta_{-1} = \Lambda_{-1}\Pi_{-1}\Lambda_{-1}^T. \tag{19}$$

We show, at the end of the discussion, that once the rank 2 property is achieved, it holds for all $t \geq 0$. By using the reformulation of the difference term, we restate the η_t term given in (15) as

$$\eta_t = \eta_{t-1} + \tilde{\boldsymbol{x}}_t^T\Lambda_{t-1}\Pi_{t-1}\Lambda_{t-1}^T\tilde{\boldsymbol{x}}_t. \tag{20}$$

For the further discussion, we prefer matrix notation and represent (20) as

$$\begin{bmatrix}\sqrt{\eta_t} & \boldsymbol{0}_2^T\end{bmatrix}\begin{bmatrix}\sqrt{\eta_t}\\ \boldsymbol{0}_2\end{bmatrix} = \begin{bmatrix}\sqrt{\eta_{t-1}} & \tilde{\boldsymbol{x}}_t^T\Lambda_{t-1}\end{bmatrix}\Theta_{t-1}\begin{bmatrix}\sqrt{\eta_{t-1}}\\ \Lambda_{t-1}^T\tilde{\boldsymbol{x}}_t\end{bmatrix}, \tag{21}$$

where Θ_{t-1} is defined as

$$\Theta_{t-1} \triangleq \begin{bmatrix}1 & \boldsymbol{0}_2^T\\ \boldsymbol{0}_2 & \Pi_{t-1}\end{bmatrix}. \tag{22}$$

We first employ a unitary Givens transformation $\boldsymbol{H}_{G,t}$ in order to zero out the second element of the vector $[\sqrt{\eta_{t-1}}, \tilde{\boldsymbol{x}}_t^T\Lambda_{t-1}]$ and then use a Θ_{t-1}-unitary Hyperbolic rotation $\boldsymbol{H}_{HB}$, i.e., $\boldsymbol{H}_{HB,t}\Theta_{t-1}\boldsymbol{H}_{HB,t}^T = \Theta_{t-1}$, to eliminate the last term. Consequently, we achieve the following update rule

$$\begin{bmatrix}\sqrt{\eta_t} & \boldsymbol{0}_2^T\end{bmatrix} = \begin{bmatrix}\sqrt{\eta_{t-1}} & \tilde{\boldsymbol{x}}_t^T\Lambda_{t-1}\end{bmatrix}\boldsymbol{H}_t, \tag{23}$$

where $\boldsymbol{H}_t$ represents the overall transformation process. Existence of these transformation matrices is guaranteed [10]. This update gives the next normalization term η_t, however, for the $(t+1)^{th}$ update, we also need the updated value of Λ_{t-1}, i.e., Λ_t, explicitly. Moreover, even calculating the Λ_t term is not sufficient, since we also need the individual value of the vector $\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t$ to update the weight vector coefficients.

We achieve the following equalities based on the same argument that we used to get (13) and (14)

$$\begin{bmatrix}\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t\\ 0\end{bmatrix} = \begin{bmatrix}\boldsymbol{A}_{t-1}^{-1} & \boldsymbol{0}_M\\ \boldsymbol{0}_M^T & 0\end{bmatrix}\tilde{\boldsymbol{x}}_t, \tag{24}$$

$$\begin{bmatrix}0\\ \boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}\end{bmatrix} = \begin{bmatrix}0 & \boldsymbol{0}_M^T\\ \boldsymbol{0}_M & \boldsymbol{A}_{t-2}^{-1}\end{bmatrix}\tilde{\boldsymbol{x}}_t. \tag{25}$$

Here, by subtracting these two equations, we get

$$\begin{bmatrix}\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t\\ 0\end{bmatrix} = \begin{bmatrix}0\\ \boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}\end{bmatrix} + \Delta_{t-1}\tilde{\boldsymbol{x}}_t. \tag{26}$$

We emphasize that the same transformation $\boldsymbol{H}_t$, which we used to get $\sqrt{\eta_t}$, also transforms Λ_{t-1} to Λ_t and $\boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}$ to $\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t$, if we extend the transformed vector as follows

$$\begin{bmatrix}\sqrt{\eta_{t-1}} & \tilde{\boldsymbol{x}}_t^T\Lambda_{t-1}\\ \frac{1}{\sqrt{\eta_{t-1}}}\begin{bmatrix}0\\ \boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}\end{bmatrix} & \Lambda_{t-1}\end{bmatrix}\boldsymbol{H}_t = \begin{bmatrix}\sqrt{\eta_t} & \boldsymbol{0}_2^T\\ \boldsymbol{q} & \boldsymbol{Q}\end{bmatrix}, \tag{27}$$

where we show that $\boldsymbol{q} = \frac{1}{\sqrt{\eta_t}}[\boldsymbol{x}_t^T\boldsymbol{A}_{t-1}^{-1}, 0]^T$ and $\boldsymbol{Q} = \Lambda_t$. We denote (27) as $\boldsymbol{B}\boldsymbol{H}_t = \tilde{\boldsymbol{B}}$, where $\boldsymbol{B}$ represents the input matrix and $\tilde{\boldsymbol{B}}$ states the output matrix of the transformation. Then, the following equality is achieved

$$\tilde{\boldsymbol{B}}\Theta_{t-1}\tilde{\boldsymbol{B}}^T = \boldsymbol{B}\Theta_{t-1}\boldsymbol{B}^T \tag{28}$$

since $\boldsymbol{H}_t$ is Θ_{t-1} unitary, i.e., $\boldsymbol{B}\boldsymbol{H}_t\Theta_{t-1}\boldsymbol{H}_t^T\boldsymbol{B}^T = \boldsymbol{B}\Theta_{t-1}\boldsymbol{B}^T$. Equating the elements of matrices in both sides of (28) yields

$$\begin{aligned}\boldsymbol{q}\sqrt{\eta_t} &= \begin{bmatrix}0\\ \boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}\end{bmatrix} + \Delta_{t-1}\tilde{\boldsymbol{x}}_t,\\ \boldsymbol{q}\boldsymbol{q}^T + \boldsymbol{Q}\Pi_{t-1}\boldsymbol{Q}^T &= \frac{1}{\eta_{t-1}}\begin{bmatrix}0\\ \boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}\end{bmatrix}\begin{bmatrix}0\\ \boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}\end{bmatrix}^T + \Delta_{t-1}.\end{aligned} \tag{29}$$

We know from (26) that the left hand side of the first term in (29) equals to $[\boldsymbol{x}_t^T\boldsymbol{A}_{t-1}^{-1}, 0]^T$ and $\boldsymbol{q}$ is given by

$$\boldsymbol{q} = \frac{1}{\sqrt{\eta_t}}\begin{bmatrix}\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t\\ 0\end{bmatrix}. \tag{30}$$

Hence, we identify the value of $\boldsymbol{Q}$ matrix using the second term in (29) as

$$\begin{aligned}\boldsymbol{Q}\Pi_{t-1}\boldsymbol{Q}^T = &\begin{bmatrix}0 & \boldsymbol{0}_M^T\\ \boldsymbol{0}_M & \dfrac{\boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}\boldsymbol{x}_{t-1}^T\boldsymbol{A}_{t-2}^{-1}}{\eta_{t-1}}\end{bmatrix}\\ &+ \left(\begin{bmatrix}\boldsymbol{A}_{t-1}^{-1} & \boldsymbol{0}_M\\ \boldsymbol{0}_M^T & 0\end{bmatrix} - \begin{bmatrix}0 & \boldsymbol{0}_M\\ \boldsymbol{0}_M^T & \boldsymbol{A}_{t-2}^{-1}\end{bmatrix}\right)\\ &- \boldsymbol{q}\boldsymbol{q}^T,\end{aligned} \tag{31}$$

where we expand the Δ_{t-1} term using its definition given in (16). We know that the term $\frac{1}{\eta_{t-1}}\boldsymbol{A}_{t-2}^{-1}\boldsymbol{x}_{t-1}\boldsymbol{x}_{t-1}^T\boldsymbol{A}_{t-2}^{-1}$ equals

to the difference $\boldsymbol{A}_{t-2}^{-1} - \boldsymbol{A}_{t-1}^{-1}$ using the update relation (10). Therefore, substituting this equality and inserting the value of $\boldsymbol{q}$ yields

$$\begin{aligned}\boldsymbol{Q}\Pi_{t-1}\boldsymbol{Q}^T &= \left(\begin{bmatrix} 0 & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & \boldsymbol{A}_{t-2}^{-1} \end{bmatrix} - \begin{bmatrix} 0 & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & \boldsymbol{A}_{t-1}^{-1} \end{bmatrix}\right) \\ &+ \left(\begin{bmatrix} \boldsymbol{A}_{t-1}^{-1} & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & \boldsymbol{A}_{t-2}^{-1} \end{bmatrix}\right) \\ &- \left(\begin{bmatrix} \boldsymbol{A}_{t-1}^{-1} & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & 0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{A}_{t}^{-1} & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & 0 \end{bmatrix}\right) \\ &= \begin{bmatrix} \boldsymbol{A}_{t}^{-1} & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \boldsymbol{0}_M \\ \boldsymbol{0}_M^T & \boldsymbol{A}_{t-1}^{-1} \end{bmatrix} \\ &= \Delta_t \\ &= \Lambda_t \Pi_t \Lambda_t^T. \end{aligned} \tag{32}$$

This equality implies that Π is time invariant, i.e., $\Pi_{t-1} = \Pi_t$ and $\boldsymbol{Q}$ is given as

$$\boldsymbol{Q} = \Lambda_t. \tag{33}$$

Hence, we show that when the low rank property of the difference term Δ_t is achieved for $t = i - 1$, it is preserved for the iteration $t = i$, for $i \geq 0$. Therefore, the transformation in (27) gives all the necessary information and provides a complete update rule. As a result, the weight vector is updated as

$$\boldsymbol{w}_t = \begin{cases} \boldsymbol{w}_{t-1} + \dfrac{1}{\mu}\left[\dfrac{\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t}{\eta_t}\right], & \text{if } e_t > \epsilon \\ \boldsymbol{w}_{t-1} - \dfrac{1}{\mu}\left[\dfrac{\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t}{\eta_t}\right], & \text{if } e_t < -\epsilon \\ \boldsymbol{w}_{t-1}, & \text{otherwise} \end{cases}, \tag{34}$$

where the individual value of $\boldsymbol{A}_{t-1}^{-1}\boldsymbol{x}_t$ is found by multiplying (30) by $\sqrt{\eta_t}$, which is the left upper most entry of the transformed matrix $\tilde{\boldsymbol{B}}$, and taking the first M elements. The processed matrix $\boldsymbol{B}$ in each iteration has the dimensions $(M + 2) \times 3$, which results in the computational complexity in the order of $O(M)$.

4. EXPERIMENTS

In this section, we illustrate the efficiency of our algorithm on a real life sequential big dataset, i.e., CMU ARCTIC speech dataset where a professional US English male speaker reads 1132 distinct utterances[11]. We implement both the regular and the fast ONS algorithms and compare the total computation times. The recording is performed at 16 KHz and there exist more than 50 million samples.

We work on the two partitions of CMU ARCTIC speech dataset with lengths $n = 5{\cdot}10^7$ and $n = 2.5{\cdot}10^7$, and measure the corresponding total processing times. Sequences of different lengths are chosen to illustrate the effect of increasing data length. For both sequences, we choose feature vectors, $\boldsymbol{x}_t \in \mathbb{R}^M$, with several dimensions ranging from $M = 16$ to $M = 128$. In Fig. 1, we demonstrate the computation time comparisons of the regular and the fast implementations of ONS algorithm. As expected from our results, complexity of the regularly implemented ONS algorithm shows a quadratic increase with respect to the dimension of the feature vectors, M. However, the proposed efficient implementation provides a linear growth on the computational complexity of the ONS algorithm. A substantial observation from Fig. 1 is that, with an increasing dimensionality of the space of feature vectors, the reduction in the complexity becomes outstanding. We also observe that the growth in the dataset length causes the same linear effect on both algorithms, i.e., doubling the total length n results in the doubled computation times.

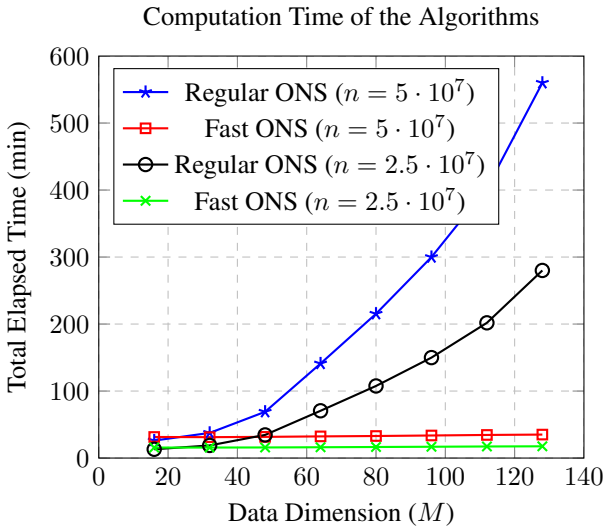

Fig. 1. Processing times for both Regular and Fast ONS algorithm with different data dimensions M.

5. CONCLUSION

In this paper, we investigate online time series prediction for high dimensional data streams and introduce a highly efficient implementation that reduces the computational complexity of the second order methods from $O(M^2)$ to $O(M)$. The presented algorithm does not require any statistical assumption on the data sequence since we only use the similarity between the consecutive feature vectors. Hence, our algorithm offers the outstanding performance of the second order methods with the low computational cost of the first order methods. We illustrate that the efficient implementation of second order methods attains significant computational gains, as the data dimension grows.

6. REFERENCES

[1] L. Bottou and Y. Le Cun, "On-line learning for very large data sets," *Applied Stochastic Models in Business and Industry*, vol. 21, no. 2, pp. 137–151, 2005. [Online]. Available: http://dx.doi.org/10.1002/asmb.538

[2] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems*, 2008, pp. 161–168. [Online]. Available: http://leon.bottou.org/publications/pdf/nips-2007.pdf

[3] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge: Cambridge University Press, 2006.

[4] A. C. Singer, S. S. Kozat, and M. Feder, "Universal linear least squares prediction: upper and lower bounds," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2354–2362, Aug 2002.

[5] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2-3, pp. 169–192, 2007.

[6] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, ser. Athena scientific series in optimization and neural computation. Belmont (Mass.): Athena Scientific, 1997. [Online]. Available: http://opac.inria.fr/record=b1094316

[7] R. D'Ambrosio, W. Belhajali, and M. Barlaud, "Boosting stochastic newton descent for bigdata large scale classification," in *2014 IEEE International Conference on Big Data*, Oct 2014, pp. 36–41.

[8] K. P. Murphy, *Machine learning : A probabilistic perspective*, ser. Adaptive computation and machine learning series. Cambridge (Mass.): MIT Press, 2012. [Online]. Available: http://opac.inria.fr/record=b1134263

[9] E. K. P. Chong and S. H. Zak, *An introduction to optimization*, ser. Wiley-Interscience series in discrete mathematics and optimization. New York: Wiley, 2008. [Online]. Available: http://opac.inria.fr/record=b1128546

[10] A. H. Sayed, *Fundamentals of Adaptive Filtering*. NJ: John Wiley & Sons, 2003.

[11] J. Kominek and A. W. Black, "Cmu arctic databases." [Online]. Available: http://www.festvox.org/cmu_arctic/index.html