Hessian-based Bounds on Learning Rate for Gradient Descent Algorithms

Prayag Gowgi and Shayan Srinivasa Garani
Department of Electronic Systems Engineering,
Indian Institute of Science, Bengaluru 560012, India.
Email: {prayag, shayangs}@iisc.ac.in

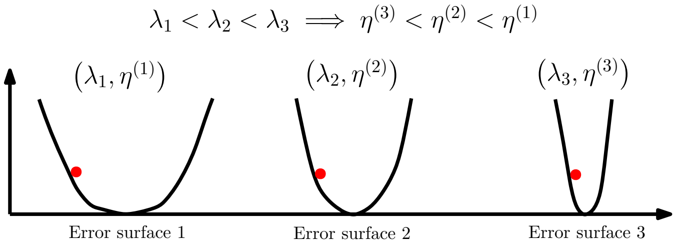
***Abstract*—Learning rate is a crucial parameter governing the convergence rate of any learning algorithm. Most of the learning algorithms based on stochastic gradient descent (SGD) method depend on heuristic choice of learning rate. In this paper, we derive bounds on the learning rate of SGD based adaptive learning algorithms by analyzing the largest eigenvalue of the Hessian matrix from first principles. The proposed approach is analytical. To illustrate the efficacy of the analytical approach, we considered several high-dimensional data sets and compared the rate of convergence of error for the neural gas algorithm and showed that the proposed bounds on learning rate result in a faster rate of convergence than AdaDec, Adam, and AdaDelta approaches which require hyper-parameter tuning.**

I. Introduction

Mathematically well-motivated adaptive learning algorithms use non-linear processing elements (PEs) for learning data statistics based on some optimization criterion. The gradient descent approach is one of the most widely used optimization techniques. In this scheme, the weight vectors associated with PEs are updated based on the gradient of a potential function. The PEs are optimized such that the potential function is minimized at every iteration. Since the update rule is iterative in nature, the analysis of convergence rate is crucial in most of the applications. The rate of convergence is decided by the learning rate.

Variety of techniques have been used for setting learning rates during optimization, such as 1) adaptive procedures [1]–[3] 2) scheduling procedure [4], [5] 3) line search procedures [6] 4) cross validation procedure [7]. One can even fix the learning rate static throughout the learning procedure. However, this would result in slower convergence rate and even oscillations during learning. A large learning rate would result in faster convergence of error rate during first few epochs of adaptation. However this is at the expense of oscillations during later stages of adaptations during learning.

In most unsupervised learning algorithms [4], [5], the learning rate is annealed slowly from a higher value (η_{init}) to a lower value (η_{final}) heuristically. This is essential for proving convergence of the learning algorithm. Now the question is how to choose η_{init} and η_{final}? There have been some analyses in this direction [8]. In [8] it is shown that η is bounded by $1/|\lambda(R)_{\max}|$, where $|\lambda(R)_{\max}|$ is the maximum eigenvalue of the data correlation matrix R. This bound is completely dependent on the data correlation matrix and does not consider the instantaneous error or the nature of the underlying algorithm. A more appropriate bound on the learning rate should be proportional to the eigenvalue of the Hessian matrix of potential function with respect to PEs as it captures the curvature of the potential function surface as shown in Figure 1. Computing eigenvalues of a Hessian matrix is computationally expensive. However, there is a principled approach [9] to compute the largest eigenvalue of a Hessian matrix. Recently, various approaches have been proposed towards adapting learning rates [1]–[3], [10]. Senior et al. [1] proposed an approach called AdaDec which is a variant of AdaGrad [2]. The idea in [2] is to vary each variable of a PE considering cumulated values of sum of squares of gradient function with respect to each variable of PE. This cumulated value increases with time, resulting in unbounded learning rate. Therefore, by considering only the recent past of squares of gradient of potential function with respect to each variable of the PE, the variation of learning rate is bounded and this approach is AdaDec.


$\lambda_1 < \lambda_2 < \lambda_3 \implies \eta^{(3)} < \eta^{(2)} < \eta^{(1)}$
$(\lambda_1, \eta^{(1)})$
$(\lambda_2, \eta^{(2)})$
$(\lambda_3, \eta^{(3)})$
Error surface 1
Error surface 2
Error surface 3
Fig. 1: Error surface and the eigenvalues of the Hessian matrix. The eigenvalues of Hessian matrix of the error surface at a point indicate the curvature of the surface at that point. Larger or smaller eigenvalues indicate higher or lower curvature of the surface at that point. This should dictate the learning rate.

Zeiler [3] proposed an approach called AdaDelta and it is an improved version over AdaGrad. AdaDelta is based on two drawbacks of AdaGrad: 1) The denominator term in the learning update equation (refer to equation (5) in [3]) accumulates the sum of squares of gradients, which in theory can grow to a very large value, reducing the learning rate to an infinitesimally small value. This problem is eliminated by accumulating the sum of squares of gradient over a finite window. 2) The physical units of the parameters do not match in AdaGrad. This mismatch is corrected by modifying the update equation of the learning rate using the inverse of a Hessian matrix. The idea is intuitive and heuristic in nature

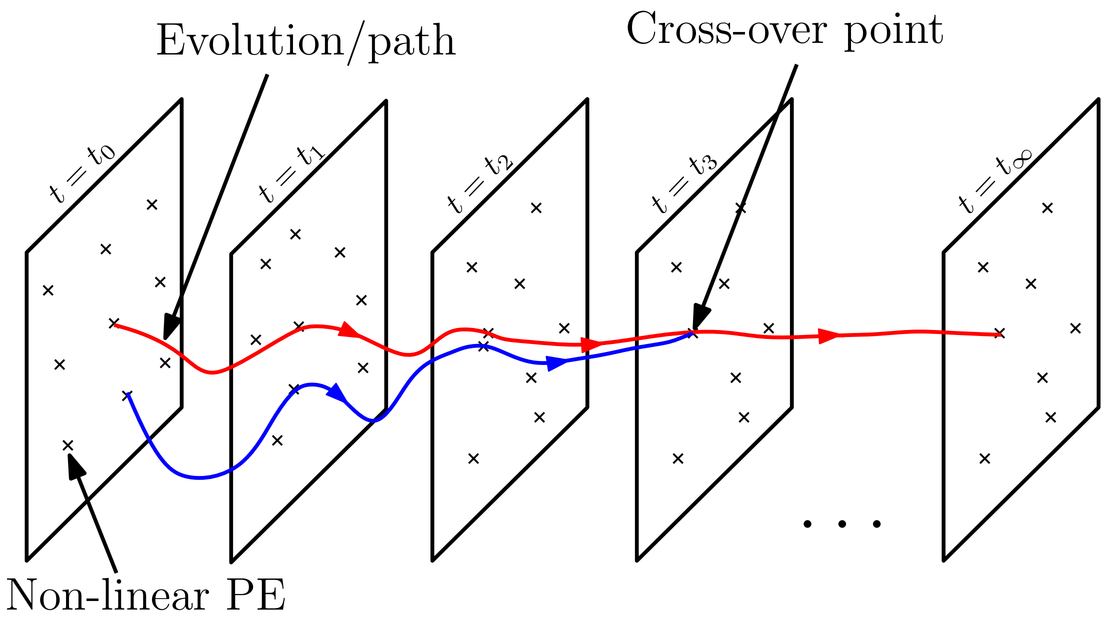


Fig. 2: Evolution of weight vectors associated with PEs in a d-dimensional space. Imagine a thread passing through each PE in a d-dimensional space. For two PEs to merge, the threads should cross over each other.

TABLE I: Various symbols (notations) and their meanings.

Variable	**Description**
t	Time
$\overline{W}_i$	i^{th} weight vector
$\overline{V}_i$	i^{th} input vector
$G(\overline{W}_i, \overline{V}_i)$	Learning algorithm
d	Dimension of input space
σ	Neighborhood radius
$F\left(\overline{W}_i^{(t+1)}, \overline{W}_i^{(t)}\right)$	Function of weight vectors at time t and $t+1$
$\mathbf{J_i}$	Jacobian of $F(\overline{W}_i^{(t+1)}, \overline{W}_i^{(t)})$
E	Potential function
$\mathbf{H_i}$	Hessian matrix of E
η_{init} and η_{final}	Initial and final learning rate
Det(.)	Determinant function
$\|.\|$	Euclidean norm in L2 sense
$\text{P}(\overline{\alpha}'(\overline{u}_i))$	Negative half space of $\overline{\alpha}'(\overline{u}_i)$
P_s	Hyper-surface of singularity
$\mu(.)$	Area measure
$D(g_{ii}, R_i)$	Disc with center g_{ii} and radius R_i
$\mathbf{C_i}$	Rank one matrix
$\mathbf{G_i}$	Symmetric matrix with real eigenvalues $b_{ij} - 1$ for all i, j
$(b_{ij} - 1, \overline{\phi}_j)$	j^{th} eigenvalue pair of $\mathbf{G_i}$
$\lambda_j(H_i)$	j^{th} eigenvalue of $\mathbf{H_i}$

and lacks an analytic approach. Adam [10] is another approach for varying the learning rate based on the first-order gradient, requiring hyper-parameter tuning. Our method is completely data driven and does not require any hyper-parameter tuning, and it is based on maximum eigenvalue of inverse Hessian matrix. In this work, we have compared our method with the AdaDec, Adam and AdaDelta approaches.

Our contributions in this paper are the following: We would like to seek solutions to the following questions: 1) Is it possible to make the choice of η dependent on data and the instantaneous error? (Q1) 2) Does η_{final} always guarantee that the error has reached its minimum? (Q2) 3) Is it possible that two PEs merge to form a single PE during learning? (Q3) In this work, we answer questions Q1 and Q2 by analyzing question Q3 and derive improved bounds on the learning rate.

This paper is organized as follows: In Section II we show that the event of two PEs merging (we call merging PEs as dead PEs) happen with measure zero. In Section III we derive both lower and upper bounds on the learning rate analytically, and show that it is inversely proportional to the largest eigenvalue of the Hessian matrix. In Section IV we compare the computational complexity of our method with Adam, AdaDec, and AdaDelta. In Section V we compare the rate of convergence of error of Adam, AdaDec, and AdaDelta with our method by using the derived bounds, followed by discussion and conclusions in Section V-C and Section VI. To ease the readability of the further sections, a list of variables (symbols) and their descriptions are given in Table I.

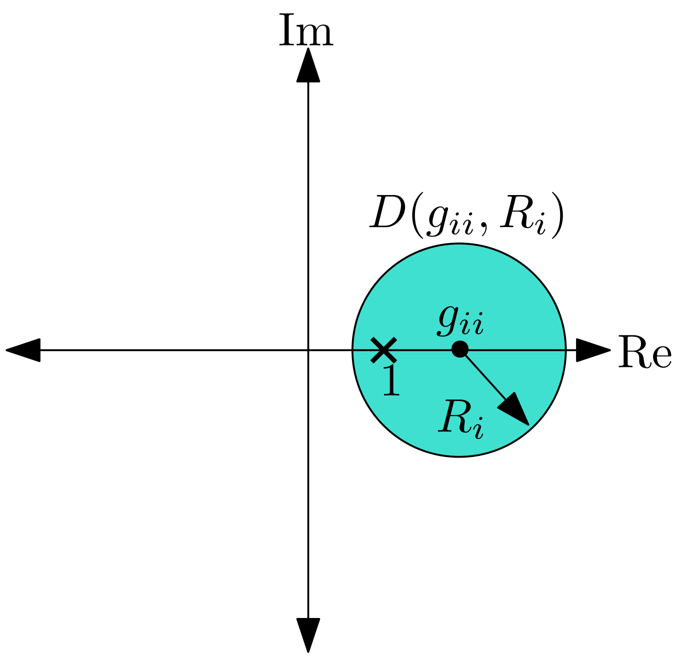


Fig. 3: Intersection of the disc $D(g_{ii}, R_i)$ with the point 1 on real line. The area measure on a point is zero.

II. Hyper-surface of singularity

The general idea of this discussion is as follows: Let $\{\overline{W}_i\}_{i=1}^{N_\mathrm{w}} \in \mathbb{R}^\mathrm{d}$ be the set of weight vectors associated with N_w PEs, $\{\overline{V}_j\}_{j=1}^{N_\mathrm{d}}$ be the set of N_d input vectors, and $G(\overline{V}, \overline{W})$ be some adaptive learning algorithm based on the gradient descent approach. We imagine the path traced by the PEs in a d-dimensional space as a set of threads passing through the d-dimension lattice as shown in Figure 2. For two PEs to merge, the threads have to cross-over each other at some point in time, making the PEs indistinguishable. One of the two PEs is a "dead PE" as a consequence of a non-invertible mapping $f(\overline{W}^{(t)}) = \overline{W}^{(t+1)}$. Now, the question is under what conditions does the inverse mapping exist?
Define a function $F(\overline{W}_i^{(t+1)}, \overline{W}_i^{(t)})$ as follows:

$$F(\overline{W}_i^{(t+1)}, \overline{W}_i^{(t)}) = \overline{W}_i^{(t+1)} - \overline{W}_i^{(t)} + \eta h_i \nabla E_{\overline{W}_i(t)} \quad (1)$$

where t is the time. The reader must note that every variable is a function of time, except the dimension d, the number of PEs N_w, and the number of input data points N_d. h_i is the neighborhood function given by

$$h_i = c(\sigma^2) \exp\left(-\frac{\|\overline{V} - \overline{W}_i\|^2}{2\sigma^2}\right), \quad (2)$$

and $\nabla E_{\overline{W}_i(t)}$ is the gradient of the potential function E with respect to $\overline{W}_i(t)$ and σ is the neighborhood radius.

We would like to express $\overline{W}_i(t)$ in terms of $\overline{W}_i(t+1)$. This calls for the inverse function theorem (IFT) [11]. The requirements for IFT are as follows: $F(\overline{W}_i^{(t+1)}, \overline{W}_i^{(t)}) \in \mathcal{C}^1$ and $\mathrm{Det}\,(\mathbf{J_i}) \neq 0$, where $\mathbf{J_i}$ is the Jacobian of $F(\overline{W}_i^{(t+1)}, \overline{W}_i^{(t)})$ with respect to $\overline{W}_i^{(t)}$, and $\mathrm{Det}(.)$ is the determinant function. Differentiating (1) with respect to $\overline{W}_i(t)$, we get

$$\mathbf{J_i} = \mathbf{0} - \mathbf{I} + \eta\,(h_i \mathbf{H_i} + \mathbf{C_i}) \quad (3)$$

where $\mathbf{I}$ is a $d \times d$ identity matrix, $\mathbf{H_i} := \left[\frac{\partial^2 E}{\partial \overline{W}_i \partial \overline{W}_j}\right]_{i,j=1}^{N_\mathrm{w}}$ is the Hessian matrix and $\mathbf{C_i}$ is given by

$$\mathbf{C_i} = \underbrace{\begin{bmatrix} \nabla E_{W_{\mathrm{i}1}}(t) \\ \nabla E_{W_{\mathrm{i}2}}(t) \\ \vdots \\ \nabla E_{W_{\mathrm{id}}}(t) \end{bmatrix}}_{\overline{u}_i} \underbrace{\begin{bmatrix} \frac{\partial h_i}{\partial W_{\mathrm{i}1}(t)} & \frac{\partial h_i}{\partial W_{\mathrm{i}2}(t)} & \cdots & \frac{\partial h_i}{\partial W_{\mathrm{id}}(t)} \end{bmatrix}}_{\overline{v}_i^\mathrm{T}}. \quad (4)$$

Let $\mathbf{I}_{\eta h_i} = \mathrm{diag}\,(\eta h_i, \ldots, \eta h_i)$ of size $d \times d$. The matrix

$$\begin{aligned} \mathbf{G_i}^\mathrm{T} &= (\mathbf{I}_{\eta h_i}\mathbf{H_i} - \mathbf{I})^\mathrm{T} \\ &= (\mathbf{I}_{\eta h_i}\mathbf{H_i})^\mathrm{T} - \mathbf{I} \\ &= \mathbf{I}_{\eta h_i}\mathbf{H_i} - \mathbf{I} \end{aligned} \quad (5)$$

is a symmetric matrix with real eigenvalues equal to $b_{ij} - 1$ for all i, j and b_{ij} is the j^{th} eigenvalue of $\mathbf{I}_{\eta h_i}\mathbf{H_i}$.

We know that[1] $\mathrm{Det}(\mathbf{J_i}) = \mathrm{Det}(\mathbf{G_i} + \mathbf{C_i})$ and

$$\mathrm{Det}(\mathbf{G_i} + \overline{u}_i\overline{v}_i^\mathrm{T}) = \mathrm{Det}(\mathbf{G_i})\left(1 + \overline{v}_i^\mathrm{T}\mathbf{G_i}^{-1}\overline{u}_i\right). \quad (6)$$

Now we would like to find the conditions under which $\mathrm{Det}(\mathbf{J_i}) = 0$. From Gershgorin circle theorem [12], we see that the eigenvalues of $\mathbf{G_i}$ lie within at least one of the discs $D(g_{ii}, R_i)$, where g_{ii} is the i^{th} diagonal element of $\mathbf{G_i}$ and $R_i = \sum_{i \neq j} |g_{ij}|$. Let $\mu(.)$ be the area measure. We see that area measure of intersection of the disc $D\,(g_{ii}, R_i)$ with a singleton on the real line is zero for all i, and hence $\mathbf{G_i}$ is full rank as illustrated in Figure 3. From (4) we see that $\mathbf{C_i}$ is of the form xy^T for some $x, y \in \mathbb{R}^d$ and hence it is a rank one matrix. From (6), $\mathrm{Det}(\mathbf{J_i}) = 0$ if and only if $\overline{v}_i^\mathrm{T}\mathbf{G_i}^{-1}\overline{u}_i = -1$, written as:

$$\overline{v}_i^\mathrm{T}\overline{\alpha}(\overline{u}_i) = -1, \quad (7)$$

where $\overline{\alpha}(\overline{u}_i) = \mathbf{G_i}^{-1}\overline{u}_i$. With $\|\overline{u}_i\| \neq 0$, normalizing $\overline{\alpha}(\overline{u}_i)$ and $\overline{v}_i$ in (7) in the L_2 sense (Through out the paper $\|.\|$ stands for L_2 norm.), we get

$$\overline{v}_i'^\mathrm{T}\overline{\alpha}'(\overline{u}_i) = \frac{-1}{\|\overline{\alpha}(\overline{u}_i)\|\|\overline{v}_i\|}, \quad (8)$$

where $\|\overline{\alpha}'(\overline{u}_i)\| = 1$ and $\|\overline{v}_i'\| = 1$. Let $\mathrm{P}(\overline{\alpha}'(\overline{u}_i)) = \{\overline{x} \in \mathbb{R}^\mathrm{d} : \overline{x}^\mathrm{T}\overline{\alpha}'(\overline{u}_i) < 0, \|\overline{x}\| = 1\}$ be the negative half-space.

$$\mathrm{P_s} = \{\overline{y} \in \mathbb{R}^\mathrm{d} : \overline{y} \in \mathrm{P}(\overline{\alpha}'(\overline{u}_i)), \overline{y}^\mathrm{T}\overline{\alpha}'(\overline{u}_i) = -1/\left(\|\overline{y}\|\|\overline{\alpha}'(\overline{u}_i)\|\right)\}$$

will be the hyper-surface of singularity i.e., every element in $\mathrm{P_s}$ results in singular $\mathbf{J_i}$. Figure 4 illustrates the hyper-surface of singularity with $d = 3$. The probability measure of $\mathrm{P_s}$ is zero and hence there exists an inverse mapping $\mathcal{F} : \overline{W}_i^{(t+1)} \to \overline{W}_i^{(t)}$ with probability one.

[1]Let $\mathbf{A} = xy^\mathrm{T}$ be a rank one matrix with $x, y \in \mathbb{R}^\mathrm{d}$. We know that 0 is an eigenvalue with algebraic multiplicity $\mathrm{d} - 1$. Also the sum of eigenvalues of $\mathbf{A}$ is equal to $\mathrm{Tr}(\mathbf{A}) = x^\mathrm{T}y$. Therefore, $\mathrm{Det}(\mathbf{A} - \lambda\mathbf{I}) = (-1)^\mathrm{d}\lambda^{\mathrm{d}-1}\left(\lambda - x^\mathrm{T}y\right) = 0$. Now, put $\lambda = -1$ to get the desired result.

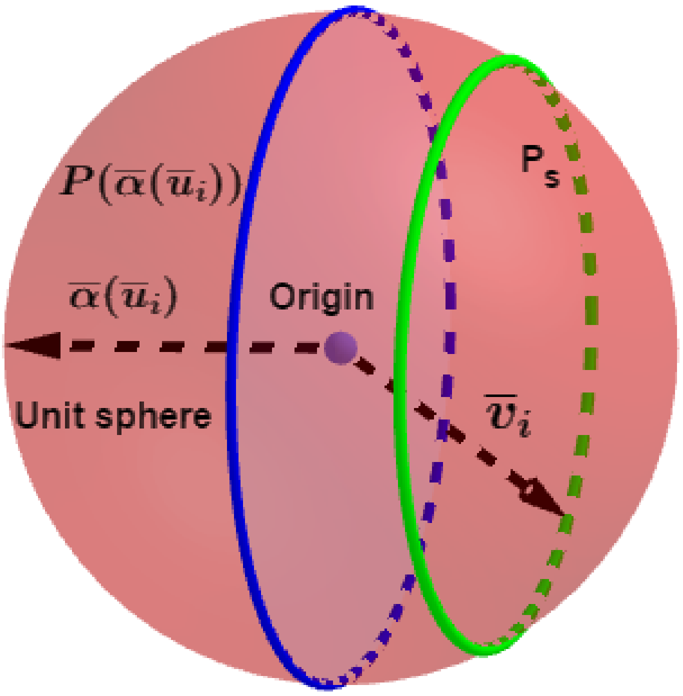

$P(\overline{\alpha}(\overline{u}_i))$
$\overline{\alpha}(\overline{u}_i)$
Origin
Unit sphere
$\overline{v}_i$
P_s

Fig. 4: Hyper-surface of singularity. The right side of the blue colored ring is the negative half space of $\overline{\alpha}(\overline{u}_i)$. Every point on the green colored ring results in singular $\mathbf{J_i}$.

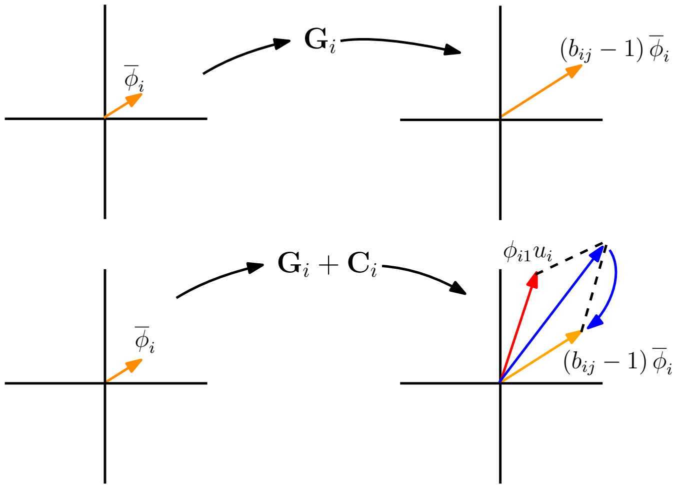

$\overline{\phi}_i$
$\mathbf{G}_i$
$(b_{ij}-1)\,\overline{\phi}_i$
$\overline{\phi}_i$
$\mathbf{G}_i+\mathbf{C}_i$
$\phi_{i1}u_i$
$(b_{ij}-1)\,\overline{\phi}_i$

Fig. 5: Geometrical interpretation of action of the matrix $\mathbf{G_i}$ and $\mathbf{G_i}+\mathbf{C_i}$ on the eigenvector $\overline{\phi}_i$ and the vector $(b_{ij}-1)\overline{\phi}_i$ acts as an attractor.

III. $\mathbf{J_i}$ OPERATING ON ANY VECTOR: A GEOMETRIC INTERPRETATION

Let $(b_{ij}-1,\overline{\phi}_j)$ be the j^{th} eigenvalue pair of $\mathbf{G_i}$. Since $\mathbf{C_i}$ is a rank one matrix, we see that $\mathbf{C_i}\overline{\phi}_j=\phi_{j1}\overline{u}_i$ and $\overline{u}_i$ is nothing but the first column of $\mathbf{C_i}$ i.e.,

$$\overline{u}_i=\left[\nabla E_{\overline{W}_{i1}}(t)\quad \nabla E_{\overline{W}i2}(t)\quad \ldots\quad \nabla E_{\overline{W}id}(t)\right]^{\mathrm{T}}. \tag{9}$$

Consider the transformation

$$\left(\mathbf{G_i}+\mathbf{C_i}\right)\overline{\phi}_j=\left(b_{ij}-1\right)\overline{\phi}_j+\phi_{j1}\overline{u}_i. \tag{10}$$

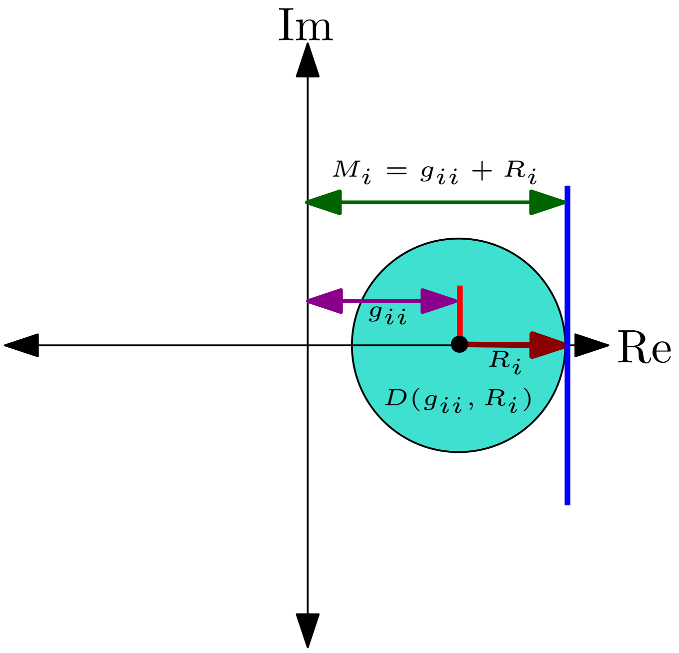
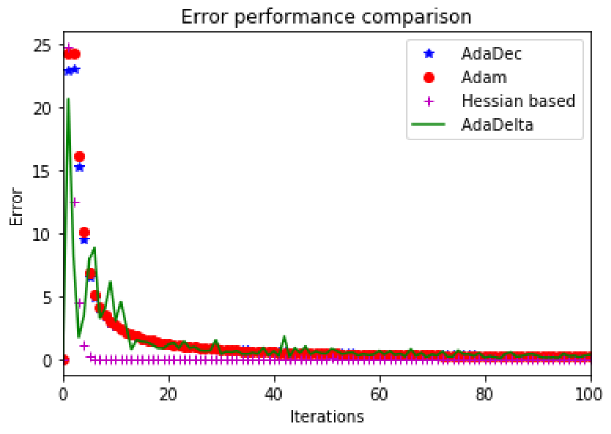


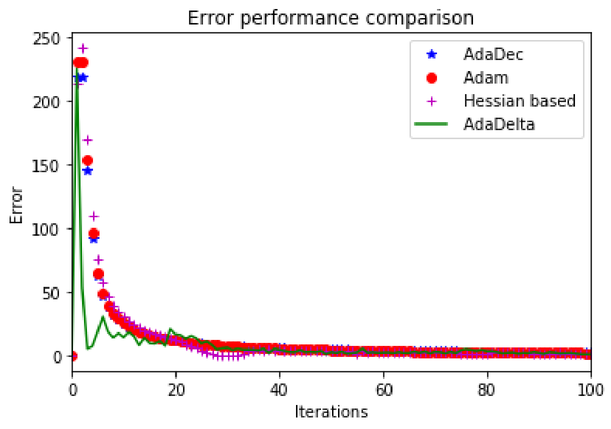
Fig. 6: The eigenvalue of G_i lies within the disc $D\left(g_{ii},R_i\right)$ and it is bounded by $M_i=g_{ii}+R_i$.

Using (10) we can derive bounds on the initial and the final learning rates. We assume the following:

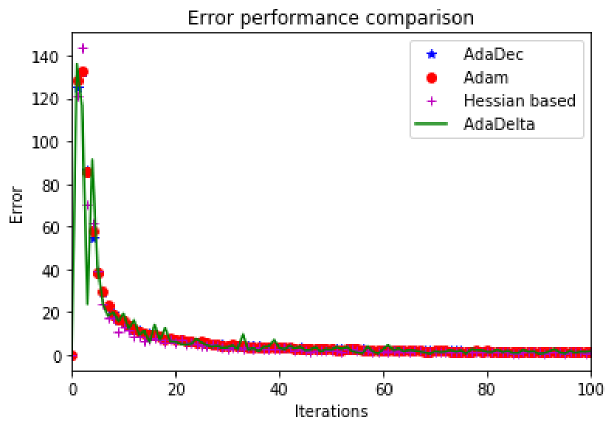
1) $\|\overline{u}_i(t)\|\leq l$ for some $l>0$ for all i, and for $t\in\mathbb{R}^+$,
2) $\lim\limits_{t\to\infty}\|\overline{u}_i(t)\|\to 0$,
3) $\|\overline{\phi}_j(t)\| = 1$ for $t\in\mathbb{R}^+$ (this can be done by normalizing $\overline{\phi}_j(t)$ for all t),
4) $\phi_{j1}(t)|_{t=0}=\phi_{j1}(0)$,



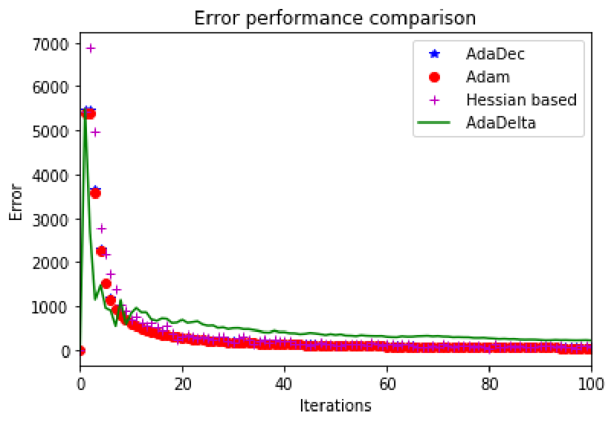
(a) Iris dataset ($\mathbb{R}^4$).



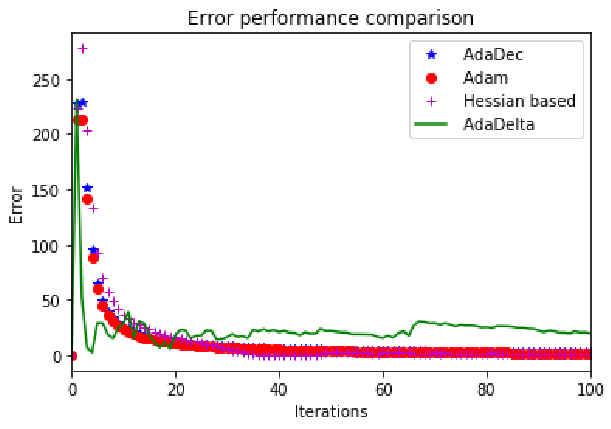
(b) Boston dataset ($\mathbb{R}^{13}$).



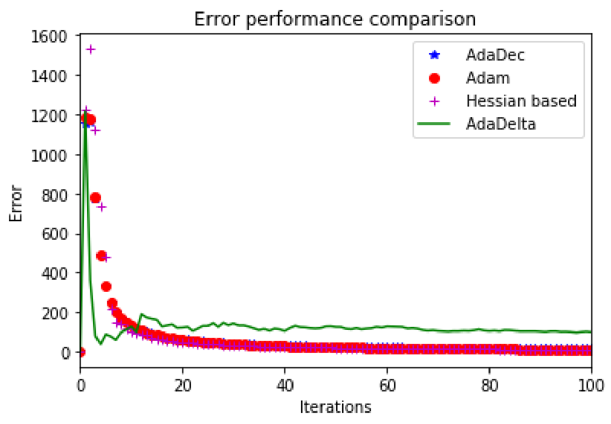
(c) Diabetes dataset ($\mathbb{R}^{10}$).



(d) Digits dataset ($\mathbb{R}^{64}$).



(e) Wine dataset ($\mathbb{R}^{13}$).



(f) Breast cancer dataset ($\mathbb{R}^{30}$).

Fig. 7: Convergence of error. The performance of Hessian based method is compared with Adam [10], AdaDec [1], and AdaDelta [3]. Hessian based approach converges faster compared to other methods with a clear advantage that there is no hyper-parameter selection and it is completely data driven.

5) $\eta_i(t)|_{t=0} = \eta_{i,\text{init}}$, and $\lim_{t\to\infty} \eta_i(t) = \eta_{i,\text{final}}$.

With the assumption 2, we see that the operation $(\mathbf{G_i} + \mathbf{C_i})$ on $\overline{\phi}_j$ is a contraction mapping resulting in $(b_{ij} - 1)\overline{\phi}_j$ and this is illustrated in Figure 5. Applying triangular inequality in (10), we get

$$\begin{aligned} \|\left(\mathbf{G_i} + \mathbf{C_i}\right)\overline{\phi}_j\| \le& |b_{ij} - 1|\|\overline{\phi}_j\| + |\phi_{j1}|\|\overline{u}_i\| \\ \le& |b_{ij} - 1|\|\overline{\phi}_j\| + |\phi_{j1}|l \\ \le& |\eta_i(t)h_i(t)\lambda_j(H_i) - 1|\|\overline{\phi}_j\| + |\phi_{j1}|l \quad (11) \end{aligned}$$

where $\lambda_j(H_i)$ is the j^{th} eigenvalue of $\mathbf{H_i}$. At $t = 0$ we get

$$\|\left(\mathbf{G_i} + \mathbf{C_i}\right)\overline{\phi}_j\| \le |\eta_{i,\text{init}}h_i(0)\lambda_j^{(t=0)}(H_i) - 1|\|\overline{\phi}_j\| + |\phi_{j1}(0)|l. \quad (12)$$

Suppose at $t = 0$ we have $\sigma_i^2(0) \approx \|\overline{V} - \overline{W}_i(0)\|^2$, then we can write[2]

$$h_i(0) \approx c(\sigma_i^2(0))\frac{1}{\sqrt{e}}. \quad (13)$$

Now (12) can be written as

$$\|\left(\mathbf{G_i} + \mathbf{C_i}\right)\overline{\phi}_j\| \le \left|\frac{\eta_{i,\text{init}}c(\sigma_i^2(0))\lambda_j^{(t=0)}(H_i)}{\sqrt{e}} - 1\right| \|\overline{\phi}_j\| + |\phi_{j1}(0)|l. \quad (14)$$

From (14), we see two cases:

Case 1 When

$$\frac{\eta_{i,\text{init}}c(\sigma_i^2(0))\lambda_j^{(t=0)}(H_i)}{\sqrt{e}} \ge 1, \quad (15)$$

(14) becomes

$$\|\left(\mathbf{G_i} + \mathbf{C_i}\right)\overline{\phi}_j\| \le \overbrace{\left(\frac{\eta_{i,\text{init}}c(\sigma_i^2(0))\lambda_j^{(t=0)}(H_i)}{\sqrt{e}} - 1\right)}^{\text{Term A}} \|\overline{\phi}_j\| + |\phi_{j1}(0)|l. \quad (16)$$

We know from the Gershgorin theorem [12] that the eigenvalues of G_i (Term A in (16)) are bounded by $M_i = R_i + g_{ii}$ as shown in Figure 6. With this bound, (16) becomes

$$\left(\frac{\eta_{i,\text{init}}c(\sigma_i^2(0))\lambda_j^{(t=0)}(H_i)}{\sqrt{e}} - 1\right)\|\overline{\phi}_j\| + |\phi_{j1}(0)|l \le M_i\|\overline{\phi}_j\| + |\phi_{j1}(0)|l. \quad (17)$$

Simplifying (17) we get

$$\eta_{\text{init}} \le \min_{i,j}\left(\frac{(1 + M_i)\sqrt{e}}{c\left(\sigma^2(0)\right)\lambda_j^{(t=0)}(H_i)}\right). \quad (18)$$

Case 2 When

$$\frac{\eta_{i,\text{init}}c(\sigma_i^2(0))\lambda_j^{(t=0)}(H_i)}{\sqrt{e}} < 1, \quad (19)$$

following similar steps from (16)-(18) on (19), we get

$$\eta_{\text{init}} \ge \max_{i,j}\left(\frac{(1 - M_i)\sqrt{e}}{c\left(\sigma^2(0)\right)\lambda_j^{(t=0)}(H_i)}\right), \quad (20)$$

[2]One can even consider that $\|\overline{V} - \overline{W}_i\|^2 \le D$ for some $D > 0$ and for all i and proceed with the analysis.

Algorithm 1 Algorithm to choose η_{init} and η_{final}

```
Require: Choose the initial set of PEs $\{\overline{W}_i(0)\}_{i=1}^{N_w}$ from
    a uniform distribution in $\mathbb{R}^d$ and $\eta_{\text{start}}$ such that (18)
    and (20) are satisfied, initial and final neighborhood radii
    $\sigma_{\text{init}} = 10$ and $\sigma_{\text{final}} = 0.1$, number of epochs $N_e$,
    number of data points $N_d$, and $T = 50$. Let $G(\overline{V}, \overline{W})$
    be some adaptive learning algorithm based on gradient
    descent technique.
1: Compute the largest eigenvalue of the Hessian matrix
   (refer to [9]).
2: Compute the mean of
       $\eta_{\text{init}} \le \min_{i,j}\left(\frac{(1 + M_i)\sqrt{e}}{c(\sigma^2(0))\lambda_j^{(t=0)}(H_i)}\right)$.
   and
       $\eta_{\text{init}} \ge \max_{i,j}\left(\frac{(1 - M_i)\sqrt{e}}{c(\sigma^2(0))\lambda_j^{(t=0)}(H_i)}\right)$,
   and assign it to $\eta_{\text{init}}$ .
3: for $j$ =1 to $N_e$ do
4:     Run $G(\overline{V}, \overline{W})$.
5:     Compute the mean of
           $m_i(\epsilon)^- := \frac{1 - \epsilon}{|\lambda(H_i)^{(\max)}|}$
       and
           $m_i(\epsilon)^+ := \frac{1 + \epsilon}{|\lambda(H_i)^{(\max)}|}$.
       after T iterations of the algorithm and assign it to $\eta_{\text{final}}$.
6: end for
```

requiring $R_i + h_{ii} < 1$ for all i. To obtain η_{final}, consider a finite $T > 0$, an $\epsilon > 0$, and define

$$|\lambda(H_i)^{(\max)}| = \max_{t > T}\{|\lambda_1(H_i)|, \ldots, |\lambda_d(H_i)|\} \quad (21)$$

$$m_i(\epsilon)^- := \frac{1 - \epsilon}{|\lambda(H_i)^{(\max)}|} \quad (22)$$

$$m_i(\epsilon)^+ := \frac{1 + \epsilon}{|\lambda(H_i)^{(\max)}|}. \quad (23)$$

The reason for selecting a finite $T > 0$ is to compute the eigenvalue of the Hessian matrix after a finite number of epochs as it would result in a stable value. Using the assumption $\lim_{t\to\infty}\|\overline{u}_i\| \to 0$ in (11), we get

$$\min_i\left(m_i(\epsilon)^+\right) \ge \eta_{\text{final}} \ge \max_i\left(m_i(\epsilon)^-\right). \quad (24)$$

Note that in (24) η_{final} is strictly greater than $\frac{1}{|\lambda(H_i)^{(\max)}|}$. The reason for this is that the eigenvector $\overline{\phi}_j \neq \overline{0}$ for all $j = 1, \ldots, d$. We know that the larger the magnitude of eigenvalue of a Hessian matrix, larger the curvature at that point. If the error surface has a tight curvature at a point, the learning rate has to be very small to avoid oscillations in the error. Similarly, if the error surface is nearly flat or the curvature is not tight, then the learning rate could be increased. This is captured by the bounds given by (18), (20) and (24).

IV. Computational complexity

We compare the computational complexity per epoch of our method with the Adam, AdaDec and AdaGrad approaches. The computationally expensive step in Algorithm 1 is the computation of the largest eigenvalue of the Hessian matrix with computational complexity $\mathcal{O}(N_{\mathrm{w}})$. Let $\mathcal{C}(G)$ be the computational complexity per epoch of the learning algorithm $G(\overline{V}, \overline{W})$. The resulting computational complexity per epoch of our algorithm is $\mathcal{C}(G) + \mathcal{O}(N_{\mathrm{w}})$ i.e., linear in N_{w}. For example, if the learning algorithm is the neural gas algorithm [5], $\mathcal{C}(G) = N_{\mathrm{w}} \log N_{\mathrm{w}}$. Table II gives a description of variables and their meaning and the computational complexity is computed in terms of these variables in Table III. From Table III, we see that the computational complexity of other methods per epoch is a function of number of data points N_{d} and the dimension of the data increasing the complexity of the algorithm and our method is better in terms of computational complexity.

TABLE II: Various symbols and their meanings. The computational complexity is expressed in terms of these variables.

Variables	**Meaning**
N_{w}	Number of PE's
N_{d}	Number of data points
d	Dimension of the input data
τ [1]	Delay constant
$G(\overline{V}, \overline{W})$	Learning algorithm
$\mathcal{C}(G)$	Computational complexity per epoch of $G(\overline{V}, \overline{W})$

TABLE III: Comparison between our method and AdaDec [1], AdaDelta [3], and Adam [10] approach in terms of computational complexity.

Algorithm	**Computational complexity**
Our method	$\mathcal{C}(G) + \mathcal{O}(N_{\mathrm{w}})$
AdaDec [1]	$\mathcal{C}(G) + \mathcal{O}(N_{\mathrm{d}} N_{\mathrm{w}} \mathrm{d}\tau)$
AdaDelta [3]	$\mathcal{C}(G) + \mathcal{O}(N_{\mathrm{w}} N_d d)$
Adam [10]	$\mathcal{C}(G) + \mathcal{O}(N_{\mathrm{w}} N_d d)$

V. Simulation set-up and results

In this section, we compare the convergence of the neural gas algorithm [5] with Adam [10], AdaDec [1], and AdaDelta [3] by choosing initial and final learning rates according to (18), (20), and (24).

A. Datasets

We consider six different datasets [13] with varying dimensions as the input for our experiments. They are 1) Diabetes: The dataset belongs to $\mathbb{R}^{10}$. Each data point is a diabetic patient record which consists of patients insulin level at different time instants, physical activity of a patient etc. 2) Digits: The dataset belongs to $\mathbb{R}^{64}$. The set consists of hand written digits from 43 people. Each 32×32 image is converted into 8×8 matrix by using 4×4 non-overlapping blocks resulting in vector in $\mathbb{R}^{64}$. 3) Wine: The dataset belongs to $\mathbb{R}^{13}$. There are three different classes meaning three different wine samples. The attributes represent the quantity of alcohol, magnesium, malic acid etc. from each wine sample. 4) Breast cancer: The dataset belongs to $\mathbb{R}^{30}$. The attributes are computed using digitized images of a fine needle aspirate of a breast mass. The attributes consists of patient ID number, result of the diagnosis, and characteristics of the cell nuclei such as radius, area, and perimeter etc. 5) Iris: The dataset belongs to $\mathbb{R}^{4}$. There are total 150 data samples, 3 different classes and 50 samples in each class. 6) Boston: The dataset belongs to $\mathbb{R}^{13}$. The dataset consists of housing values in the suburb of Boston. Some of the attributes consist of crime rate by town, average number of rooms per house, distance from Boston employment centers etc. We have normalized each dataset as follows:

$$\overline{V}' = \frac{\overline{V} - \min\left(\{\overline{V}\}_{i=1}^{N_d}\right)}{\max\left(\{\overline{V}\}_{i=1}^{N_d}\right) - \min\left(\{\overline{V}\}_{i=1}^{N_d}\right)} \tag{25}$$

where the $\min\left(\{\overline{V}\}_{i=1}^{N_d}\right)$ or $\max\left(\{\overline{V}\}_{i=1}^{N_d}\right)$ is a vector in which each coordinate is minimum or the maximum along that coordinate. The reason for selecting these datasets are 1) varying high-dimension 2) sampled from different data distributions from various applications. We have fixed the neural gas algorithm as the benchmark.

B. Network initialization and results

We have used $N_{\mathrm{w}} = 15$ PEs, initial and final neighborhood radii $\sigma_{\mathrm{init}} = 10$ and $\sigma_{\mathrm{final}} = 0.1$ respectively, and number of epochs $N_{\mathrm{e}} = 10$. In order to compare with our method, we have compared our method with AdaDec 1, AdaDelta [3], Adam [10]. Various parameters of Adam, AdaDec, and AdaDelta are set as follows:

- Adam: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\alpha = 0.002$, and $\epsilon = 10^{-8}$.
- AdaDec: $c = 1000$, $r = 25$, $\tau = 4$, and $\gamma = 0.4$.
- AdaDelta: $\rho = 0.95$.

Figures 7(a)—7(f) show the comparison of the error convergence rates for the neural gas algorithm for each of the six data sets. In Figure 7(a), we clearly see that our method outperforms all the other methods. From 7(b)-7(f), we see that our method is comparable to the other methods with a clear advantage that one need not worry about tuning any hyper-parameter.

C. Discussion

We have addressed the three questions Q1-Q3 as mentioned in the introduction. In particular, question Q3 only guarantees the existence of the inverse mapping of weight vector update equation from time $t+1$ to t with probability one. However, finding the inverse mapping itself is a challenging problem. As a consequence of this analysis, questions Q1 and Q2 were addressed, making the learning rate data dependent through the maximum eigenvalue of the Hessian matrix from first principles. We know that instantaneous error value should drive the update of weight vectors (because this is a feedback path from the error surface to the weight vector update) as opposed to learning rate driving the weight vector updates (feed forward) and the feedback is in terms of eigenvalues of the Hessian matrix capturing the curvature of the error surface at a point.

VI. Conclusions

We analyzed the situation of dead PEs and showed that the occurrence of dead PE is a measure zero event. We derived Hessian-based lower and upper bounds on the learning rate for gradient descent algorithms, making it data dependent. The learning rate is driven by the curvature of the error surface at a point, captured by the largest eigenvalue of the Hessian matrix evaluated at that point. This guarantees a smaller final learning rate as opposed to other algorithms that reduce learning rate to smaller values, independent of the data. We compared the performance of the neural gas algorithm using the derived bounds on learning rates against Adam, AdaDec, and AdaDelta. Using simulations, we showed that the rate of convergence using our analytic approach outperforms other methods, motivating the use of analytical approaches in deciding hyper-parameters in learning algorithms.

References

[1] A. Senior, G. Heigold, M. Ranzato, and K. Yang, "An empirical study of learning rates in deep neural networks for speech recognition," in *2013 IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, May 2013, pp. 6724–6728.

[2] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[3] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[4] T. Kohonen, "Self-organizing map," *Proc. of the IEEE*, vol. 78, pp. 1464–1480, 1990.

[5] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "Neural-gas network for vector quantization and its application to time-series prediction," *IEEE Trans. on Neural Netw.*, vol. 4, no. 4, pp. 558–569, 1993.

[6] S. Wright and J. Nocedal, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[7] D. Johnson, D. Ellis, C. Oei, C. Wooters, P. Faerber, N. Morgan, and K. Asanovic, "ICSI quicknet software package," 2004.

[8] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[9] Y. LeCun, P. Y. Simard, and B. Pearlmutter, "Automatic learning rate maximization by on-line estimation of the Hessian's eigenvectors," in *Advances in Neural Inf. Process. Syst. 5*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds. Morgan-Kaufmann, 1993, pp. 156–163.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] M. Spivak, *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. CRC Press, 2018.

[12] S. A. Gershgorin, "Uber die abgrenzung der eigenwerte einer matrix," *Proc. of the Russian Academy of Sciences. Mathematical Series*, no. 6, pp. 749–754, 1931.

[13] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

Acknowledgment

The authors acknowledge the Ministry of Human Resource Development, Govt. of India for this work. The financial expenses towards the conference is supported by Ericsson.