Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Stochastic gradient based extreme learning machines for stable online learning of advanced combustion engines

Vijay Manikandan Janakiraman [a,*], XuanLong Nguyen [b], Dennis Assanis [c]

[a] *Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA*
[b] *Department of Statistics, University of Michigan, Ann Arbor, MI, USA*
[c] *Stony Brook University, NY, USA*

ARTICLE INFO

Article history:
Received 6 May 2015
Received in revised form
1 November 2015
Accepted 7 November 2015

Keywords:
Online learning
Extreme learning machine
System identification
Lyapunov stability
Engine control
Operating envelope

ABSTRACT

We propose and develop SG-ELM, a stable online learning algorithm based on stochastic gradients and Extreme Learning Machines (ELM). We propose SG-ELM particularly for systems that are required to be stable during learning; i.e., the estimated model parameters remain bounded during learning. We use a Lyapunov approach to prove both asymptotic stability of estimation error and boundedness in the model parameters suitable for identification of nonlinear dynamic systems. Using the Lyapunov approach, we determine an upper bound for the learning rate of SG-ELM. The SG-ELM algorithm not only guarantees a stable learning but also reduces the computational demand compared to the recursive least squares based OS-ELM algorithm (Liang et al., 2006). In order to demonstrate the working of SG-ELM on a real-world problem, an advanced combustion engine identification is considered. The algorithm is applied to two case studies: An online regression learning for system identification of a Homogeneous Charge Compression Ignition (HCCI) Engine and an online classification learning (with class imbalance) for identifying the dynamic operating envelope. The case studies demonstrate that the accuracy of the proposed SG-ELM is comparable to that of the OS-ELM approach but adds stability and a reduction in computational effort.

1. Introduction

Homogeneous Charge Compression Ignition (HCCI) Engines are of significant interest to the automotive industry owing to their ability to reduce emissions and fuel consumption significantly compared to existing methods such as spark ignition (SI) and compression ignition (CI) engines [1–3]. Although HCCI engines tend to do well in laboratory controlled tests, practical implementation is quite challenging because HCCI engines do not have a direct trigger for ignition (such as spark in SI or fuel injection in CI). Further, HCCI requires some special engine designs such as exhaust gas recirculation (EGR) [4], variable valve timings (VVT) [5], intake charge heating [6] among others. Such advanced designs also increase the complexity of the engine operation making it unstable and extremely sensitive to operational disturbances [7,8]. A model based control is typically opted to address the challenges involved in controlling HCCI [9,5,10]. For model development, both physics based approaches [9,5,10] and data based approaches [11–14] were shown to be effective. A key requirement for a model based control is the ability of the models to accurately predict the engine state variables for several operating cycles ahead of time, so that a control action with a known consequence can be applied to the engine. Further, in order to be vigilant against the engine drifting towards instabilities such as misfire, ringing, knock, etc. [15,16], the operating limits of the engine particularly in transients, is required to be known. In order to develop controllers and operate the engine in a stable manner, both models of the engine state variables as well as the operating envelope are necessary.

Data based modeling approaches for the HCCI engine state variables and dynamic operating envelope were demonstrated using neural networks [11], support vector machines [12], extreme learning machines [13,14] by the authors. However, previous research considered an offline approach where the data collected from engine experiments were taken offline and models were developed using computer workstations that had high processing and memory. However, a key requirement in advancing HCCI modeling is to perform online learning for the following reasons. The models developed offline are valid only in the controlled experimental conditions. For instance, the experiments are performed at a controlled ambient temperature, pressure and

* Corresponding author. Presently at: NASA Ames Research Center, MS 269-1, Moffett Field, CA 94035-1000, USA. Tel.: +1 734 358 6633.
E-mail address: vijai@umich.edu (V.M. Janakiraman).

http://dx.doi.org/10.1016/j.neucom.2015.11.024
0925-2312/© 2015 Elsevier B.V. All rights reserved.

humidity conditions. As a result, the models developed are valid for the specified conditions and when the models are implemented on a vehicle, the expectation is that the model works on a wide range of climatic conditions that the vehicle is exposed to, possibly on conditions that were not experimented. Hence, an online adaptation to learn the behavior of the system at new/unfamiliar situations is required. Also, since the offline models are developed directly from experimental data, they may perform poorly in certain operating regions where the density of experimental data is low. As more data becomes available in such regions, an online mechanism can be used to adapt to such data. In addition, the engine produces high velocity streaming data; operating at about 2500 revolutions per minute, an in-cylinder pressure sensor can produce about 1.8 million data observations per day. It becomes infeasible to store this volume of data for offline model development. Thus, an online learning framework that processes every data observation, updates the model and discards the data is required for advanced engines like HCCI.

Online learning, as the name suggests, refers to obtaining a model online; i.e., learning happens when the system is in operation and as data is streaming in. Typically, the learning is sequential; i.e., the data from the system is processed one-by-one or batch-by-batch and the model parameters are updated. A data processor on-board a combustion engine usually is low on computation power and memory. Thus, simple linear models used to be the natural choice for combustion engines. However, for a system like the HCCI engine, linear models may be insufficient to capture the complex dynamics, particularly for predicting several steps ahead in time [11]. While numerous nonlinear methods for online learning do exist in machine learning literature, a complete survey is beyond the scope of this paper. The recent paper on online sequential extreme learning machines (OS-ELM) [17] surveys popular online learning algorithms in the context of classification and regression and develops an efficient algorithm based on recursive least squares. The OS-ELM algorithm appears to be the present state of the art (although some variants have been proposed such as [18–20]) for classification/regression problems achieving a global optimal solution, high generalization accuracies and most importantly, in quick time. Also, based on observations from our previous work [21], we choose extreme learning machines (ELM) over other popular methods such as neural networks and support vector machines for the HCCI engine problem. It has been shown that both polynomial and linear methods were inferior in terms of prediction accuracy [12,11] although they have simple algorithms suitable for online applications. The online variants of SVM usually work by approximating the batch (offline) loss function so that data can be processed sequentially [22,23] and achieve accuracies similar to that of the offline learning counterparts. However, SVMs Come with a high computation and memory requirement to be used efficiently on a memory limited system such as the engine control unit [13]. Thus we prefer ELM over SVM and other state of the art nonlinear models.

In spite of its known advantages, an over-parameterized ELM may suffer from ill-conditioning problem when a recursive least squares type update is performed (as in OS-ELM). This sometimes results in poor regularization behavior as reported in [24,25,20,26,27], which leads to an unbounded growth of the model parameters and unbounded model predictions. This may not be a serious problem for many applications as the model usually improves as more data becomes available. However, for control problems in particular, if decisions are made simultaneously based on the online learned model (as in adaptive control [28]), it is critical that the parameter estimation algorithm behaves in a stable manner so that control actions can be trusted at all times. Hence a guarantee of stability and parameter boundedness is of extreme importance. To address this issue, we propose the SG-ELM, a stable online learning algorithm based on stochastic gradient descent and extreme learning machines. By extending ELM to include a notion of stable learning, we hope that the simplicity and generalization power of ELM can be retained along with stability of identification, suitable for real-time control applications. We use a Lyapunov approach to prove both asymptotic stability of estimation error and boundedness in the estimated parameters suitable for identification of nonlinear dynamic systems. Using the Lyapunov approach, we determine an upper bound for the learning rate of SG-ELM that seems to avoid bad regularization that may arise during online learning. These are the main contributions of this paper. Further, we also apply the SG-ELM algorithm to two real-world HCCI identification problems including online state estimation and online operating boundary estimation which is a novel application of online extreme learning machines.

The remainder of the article is organized as follows. The ELM modeling approach is described in Section 2 along with algorithm details on batch (offline) learning as well as the present state of the art; the OS-ELM algorithm. In Section 3, the stochastic gradient based ELM algorithm is derived along with a stability proof. In Section 4, the background on HCCI engine and experimentation are discussed. Sections 5 and 6 cover the discussions on the application of the SG-ELM algorithm on the two applications, followed by conclusions in Section 7.

2. Extreme learning machines

Extreme Learning Machine (ELM) is an emerging learning paradigm for multi-class classification and regression problems [29,30]. An advantage of the ELM method is that the training speed is extremely fast, thanks to the random assignment of input layer parameters which do not require adaptation to the data. In such a setup, the output layer parameters can be analytically determined using a linear least squares approach. Some of the attractive features of ELM [29] include the universal approximation capability of ELM, the convex optimization problem of ELM resulting in the smallest training error without getting trapped in local minima, closed form solution of ELM eliminating iterative training and better generalization capability of ELM [30]. In comparison, a backpropagation neural network has the same objective function as that of ELM but they often get trapped in local minima whereas ELM do not. Support vector machines on the other hand, solves a convex optimization problem but the computation involved is quite high and running times are slow for large datasets. Thus, ELM appears to be very efficient both in terms of accuracy and running times compared to several state-of-the-art algorithms.

Consider the following data set

$$\{(x_1, y_1), \ldots, (x_N, y_N)\} \in (\mathcal{X}, \mathcal{Y}), \tag{1}$$

where N denotes the number of training samples, $\mathcal{X}$ denotes the space of the input features and $\mathcal{Y}$ denotes labels whose nature differentiate the learning problem in hand. For instance, if $\mathcal{Y}$ takes integer values $\{1, 2, 3, ..\}$ then the problem is referred to as classification and if $\mathcal{Y}$ takes real values, it becomes a regression problem. ELMs are well suited for solving both regression and classification problems faster than state of the art algorithms [30]. A further distinction could be made depending on the availability of training data during the learning process, as offline learning (or batch learning) and online learning (or sequential learning). Offline learning could make use of all training data simultaneously as all data is available to the algorithm and time is generally not a limiting factor. So it is possible to have the model see the data several times (iterations) so that the best accuracy can be achieved. On the other hand, there may be situations where offline learning becomes infeasible and one has to resort to online learning, such as those involving high velocity steaming data where time taken for learning becomes a bottleneck. In an online learning setting, data is available one-by-one or batch-by-batch and needs to be processed with limited computational effort and storage. Further, inference is required to be made with each new available data along with the ones recorded in the past.

2.1. Batch (Offline) ELM

When the entire data is available and a model is required to be trained using all the available data, batch learning is adopted. In this case, the ELM algorithm involves solving the following optimization problem similar to that of a ridge regression

$$\min_W \left\{ \|HW - Y\|^2 + \lambda \|W\|^2 \right\} \tag{2}$$

$$H^T = \psi(W_r^T x(k) + b_r) \in \mathbb{R}^{n_h \times 1}, \tag{3}$$

where λ represents the regularization coefficient determined using cross-validation, Y represents the vector of labels, ψ represents the hidden layer activation function (sigmoidal, sinusoidal, radial basis, etc. [30]) and $W_r \in \mathbb{R}^{n \times n_h}$, $W \in \mathbb{R}^{n_h \times y_d}$ represents the input and output layer parameters respectively. Here, n represents the dimension of inputs $x(k)$, n_h represents the number of hidden neurons of the ELM model, H represents the hidden layer output matrix and y_d represents the dimension of outputs Y. The matrix W_r consists of randomly assigned values that map the input vector to a high dimensional feature space while $b_r \in \mathbb{R}^{n_h}$ is a bias component assigned in a random manner similar to W_r. The number of hidden neurons determines the expressive power of the transformed feature space. The values of W_r and b_r can be assigned based on any continuous random distribution [30] and remains fixed during the learning process. Hence the training reduces to a single step calculation given by Eq. (4). The ELM decision hypothesis can be expressed as in Eq. (5) for classification and as in Eq. (6) for regression. It should be noted that the hidden layer and the corresponding activation functions give a nonlinear mapping of the data, which if eliminated, the ELM model becomes a linear least squares (Linear LS) model and is considered as one of the baseline models in this study.

$$W^* = \left(H^T H + \lambda I\right)^{-1} H^T Y \tag{4}$$

$$f(x) = sgn\left(W^T[\psi(W_r^T x + b_r)]\right). \tag{5}$$

$$f(x) = W^T[\psi(W_r^T x + b_r)] \tag{6}$$

Since training involves solving a linear least squares with a convex objective function, the solution obtained by ELM is extremely fast and is a global optimum for the chosen n_h, W_r and b_r. The above formulation for classification (5) is not designed to handle imbalanced or skewed data sets [13]. As a modification to weigh the minority class data more, a simple weighting method can be incorporated in the ELM objective function (2) as

$$\min_W \left\{ (HW - Y)^T \Gamma (HW - Y) + \lambda W^T W \right\} \tag{7}$$

$$\Gamma = \begin{bmatrix} \gamma_1 & 0 & . & . & 0 \\ 0 & \gamma_2 & . & . & 0 \\ . & . & . & . & 0 \\ 0 & 0 & . & . & \gamma_N \end{bmatrix}$$

$$\gamma_i = \begin{cases} 1 & \text{majority class data} \\ r \times f_s & \text{minority class data} \end{cases} \tag{8}$$

where Γ represents the weight matrix, r represents the ratio of number of majority class data to number minority class data and f_s represents a scaling factor to be tuned for a given data set [13]. This results in the training step given by Eq. (9) and the decision hypothesis takes the same form as in Eq. (5):

$$W^* = \left(H^T \Gamma H + \lambda I\right)^{-1} H^T \Gamma Y. \tag{9}$$

2.2. Online Sequential ELM (OS-ELM)

The OS-ELM [17] is a recursive version of the batch ELM algorithm. This version of the algorithm is used for online learning purposes where data is processed one-by-one or batch-by-batch and the model parameters are updated after which the data is not required to be stored. In this process, training involves two steps – an initialization step and a sequential learning step. During the initialization step, a set of data observations (N_0) are required to initialize H_0 and W_0 by solving the batch ELM optimization problem as follows

$$\min_{W_0} \left\{ \|H_0 W_0 - Y_0\|^2 + \lambda \|W_0\|^2 \right\} \tag{10}$$

$$H_0 = [g(W_r^T x_0 + b_r)]^T \in \mathbb{R}^{N_0 \times n_h}. \tag{11}$$

The solution W_0 is given by

$$W_0 = K_0^{-1} H_0^T Y_0 \tag{12}$$

where $K_0 = H_0^T H_0 + \lambda I$. Suppose given another new data x_1, the problem becomes

$$\min_{W_1} \left\| \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} W_1 - \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \right\|^2. \tag{13}$$

The solution can be derived as

$$W_1 = W_0 + K_1^{-1} H_1^T (Y_1 - H_1 W_0)$$
$$K_1 = K_0 + H_1^T H_1.$$

Based on the above, a generalized recursive algorithm for updating the least-squares solution can be computed as follows [17]

$$M_{k+1} = M_k - M_k H_{k+1}^T (I + H_{k+1} M_k H_{K+1}^T)^{-1} H_{k+1} M_k \tag{14}$$

$$W_{k+1} = W_k + M_{k+1} H_{k+1}^T (Y_{k+1} - H_{k+1} W_k) \tag{15}$$

where M represents the covariance of the parameter estimate.

3. Stochastic gradient based ELM algorithm

In this section, we propose an extension of the ELM algorithm for online learning using stochastic gradient descent (SGD). Stochastic gradient descent methods have been popular for several decades for online learning but practically limited because of poor optimization characteristics (failure to converge to an absolute minimum, for instance) and slow convergence rates. However, only recently, the asymptotic behavior of SGD methods has been analyzed indicating that SGD methods can be very powerful for learning large data sets [31,32]. SGD based algorithms have been developed successfully for perceptron models, K-means, SVM and Lasso [31]. In this work, we propose to use the simple and scalable SGD algorithm to extreme learning machines and derive stability properties, so that an online learning algorithm useful for control purposes can be developed.

The justification of SGD based algorithms in machine learning can be briefly discussed as follows. In any learning problem, three types of errors are encountered, namely the approximation error, the estimation error and the optimization error [31], and the expected risk $E_{exp}(f)$ and the empirical risk $E_{emp}(f)$ for a supervised learning problem can be given by

$$E_{exp}(f) = \int l(f(x), y) dP(x, y)$$

$$E_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} l(f(x_i), y_i)$$

where $l(f(x), y)$ denotes the loss function between the prediction $f(x)$ and label y, $P(x, y)$ denote the joint probability density of x and y. Let $f^* = \operatorname{argmin}_f E_{exp}(f)$ be the best possible prediction function. In practice, the prediction function is chosen from a family of parametric functions denoted by $\mathcal{F}$. Let $f^*_{\mathcal{F}} = \operatorname{argmin}_{f \in \mathcal{F}} E_{exp}(f)$ be the best prediction function chosen from a parameterized family of functions $\mathcal{F}$. When a finite training data set becomes available, the empirical risk becomes a proxy for the expected risk for the learning problem [33]. Let $\overline{f}^*_{\mathcal{F}} = \operatorname{argmin}_{f \in \mathcal{F}} E_{emp}(f)$ be the solution that minimizes the empirical risk. However, the global solution is not typically obtained because of computational limitations and hence the solution of the learning problem is reduced to finding $\overline{f}_{\mathcal{F}} = \operatorname{argmin}_{f \in \mathcal{F}} E_{emp}(f)$.

Using the above setup, the approximation error (E_{app}) is the error introduced in approximating the true function space with a family of functions $\mathcal{F}$, the estimation error (E_{est}) is the error introduced in optimizing over $E_{emp}(f)$ instead of $E_{exp}(f)$, the optimization error (E_{opt}) is the error induced as a result of stopping the optimization to $\overline{f}_{\mathcal{F}}$. The total error E_{tot} can be expressed as

$$E_{app} = E_{exp}(f^*) - E_{exp}(f^*_{\mathcal{F}})$$

$$E_{est} = E_{exp}(f^*_{\mathcal{F}}) - E_{emp}(\overline{f}^*_{\mathcal{F}})$$

$$E_{opt} = E_{emp}(\overline{f}^*_{\mathcal{F}}) - E_{emp}(\overline{f}_{\mathcal{F}})$$

$$E_{tot} = E_{app} + E_{est} + E_{opt}$$

The following observations are taken from the asymptotic analysis of SGD algorithms [31,34].

1. The empirical risk $E_{emp}(f)$ is only a surrogate for the expected risk $E_{exp}(f)$ and hence an increased effort to minimize E_{opt} may not translate to better learning. In fact, if E_{opt} is very low, there is a good chance that the prediction function will over-fit the training data.
2. SGD are worst optimization algorithms (in terms of reducing E_{opt}) but they minimize the expected risk relatively quickly. Therefore, in the large scale setup, when the limiting factor is computational time rather than the number of examples, SGD algorithms perform asymptotically better.
3. SGD results in a faster convergence when the loss function has strong convexity properties.

The last observation is key in developing our algorithm based on ELM models. The ELM models have a squared loss function and when the hidden neurons are randomly assigned and fixed, the training translates to solving a convex optimization problem. This motivates the use of ELM models for performing SGD based learning. The SGD based algorithm can be derived for the ELM models as follows.

3.1. SG-ELM parameter update

Let (x_i, y_i) where $i = 1, 2, .. N$ be the streaming data in consideration. The data can be considered to be available to the algorithm from a one-by-one continuous stream or artificially sampled one-by-one from a very large data set. Let the ELM empirical risk be defined as follows

$$\begin{aligned} J(W) &= \min_W \frac{1}{2}\sum_{i=1}^{N} \| y_i - \phi_i^T W \|^2 \\ &= \min_W \left\{ \frac{1}{2}\| y_1 - \phi_1^T W \|^2 + \cdots + \frac{1}{2}\| y_N - \phi_N^T W \|^2 \right\} \\ &= \min_W \{ J_1(W) + J_2(W) + \cdots + J_N(W) \}. \end{aligned} \tag{16}$$

where $W \in \mathbb{R}^{n_h \times y_d}$, $y_i \in \mathbb{R}^{1 \times y_d}$ $\phi \in \mathbb{R}^{n_h \times y_d}$ is the hidden layer output (see H^T in Eq. (3)). If an error $e_i \in \mathbb{R}^{1 \times y_d}$ can be defined as $(y_i - \phi_i^T W)$, the learning objective for a data observation i can be given by

$$\begin{aligned} J_i(W) &= \frac{1}{2} e_i^T e_i \\ &= \frac{1}{2}(y_i - \phi_i^T W)^T (y_i - \phi_i^T W) \\ &= \frac{1}{2} y_i^T y_i + \frac{1}{2} W^T \phi_i \phi_i^T W - y_i^T \phi_i^T W \\ \frac{\partial J_i}{\partial W} &= \phi_i \phi_i^T W - \phi_i y_i = \phi_i(\phi_i^T W - y_i) \\ &= -\phi_i e_i. \end{aligned} \tag{17}$$

In a regular gradient descent (GD) algorithm, the gradient of $J(W)$ is used to update the model parameters as follows.

$$\begin{aligned} \frac{\partial J}{\partial W} &= \frac{\partial J_1}{\partial W} + \frac{\partial J_2}{\partial W} + \cdots + \frac{\partial J_N}{\partial W} \\ \Rightarrow \frac{\partial J}{\partial W} &= -\phi_1 e_1 - \phi_2 e_2 - \cdots - \phi_N e_N \\ W_{k+1} &= W_k - \Gamma_{SG} \frac{\partial J}{\partial W} \\ &= W_k + \Gamma_{SG}(\phi_1 e_1) + \cdots + \Gamma_{SG}(\phi_N e_N) \end{aligned} \tag{18}$$

where k is the iteration count, $\Gamma_{SG} \in \mathbb{R}^{n_h \times n_h}$ represents the step size or update gain matrix for the GD algorithm.

It can be seen from Eq. (18) that the parameter matrix W is updated based on gradients calculated from all the available examples. If the number of data observations is large, the gradient calculation can take enormous computational effort. The stochastic gradient descent algorithm considers one example at a time and updates W based on gradients calculated from (x_i, y_i) as shown in

$$W_{i+1} = W_i + \Gamma_{SG}(\phi_i e_i). \tag{19}$$

From Eq. (18), it is clear that the optimal W is a function of gradients calculated from all the examples. As a result, as more data becomes available, W converges close to its optimal value in SGD algorithm. Processing data one-by-one significantly reduces the computational requirement making the algorithm scale well to large data sets.

In order to handle class imbalance, the algorithm in (19) can be modified by weighting the minority class data more. The modified algorithm can be expressed as

$$W_{i+1} = W_i + \Gamma_{imb} \Gamma_{SG}(\phi_i e_i) \tag{20}$$

where $\Gamma_{imb} = r \times f_s$, r and f_s represent the imbalance ratio (a running count of majority class data to minority class data until that instant) and the scaling factor that needs to be tuned to obtain tradeoffs between high false positives and missed detections for a given application.

3.2. Stability analysis

The stability analysis of the SG-ELM algorithm can be derived as follows. The ELM structure makes the analysis simple and similar to that of a linear gradient based algorithm [35].

The instantaneous prediction error e_i (Here the error e and output y are transposed as opposed to their previous definition in

Table 1
Specifications of the experimental HCCI engine.

Engine type	4-stroke In-line
Fuel	Gasoline
Displacement	2.0 L
Bore/stroke	86/86 mm
compression ratio	11.25:1
Injection type	Direct injection
Valvetrain	Variable valve timing with hydraulic cam phaser having 119° constant duration, defined at 0.25 mm lift, 3.5 mm peak. lift and 50° crank angle, and phasing authority.
HCCI strategy	Exhaust recompression using negative valve overlap

Section 3.1 for ease of derivations) can be expressed in terms of the parametric error ($\tilde{W} = W_* - W$) as

$$\begin{aligned} e_i &= y_i - W^T\phi_i \\ &= W_*^T\phi_i - W^T\phi_i \\ &= \tilde{W}^T\phi_i \end{aligned} \tag{21}$$

where W_* represents true model parameters. Further, the parametric error dynamics can be obtained as follows.

$$\begin{aligned} \tilde{W}_{i+1} &= W_* - W_{i+1} \\ &= W_* - W_i - \Gamma_{SG}\phi_i e_i^T \\ &= \tilde{W}_i - \Gamma_{SG}\phi_i e_i^T \end{aligned} \tag{22}$$

Consider the following positive definite, decrescent and radially unbounded [35] Lyapunov function V

$$V(\tilde{W}) = tr(\tilde{W}^T \Gamma_{SG}^{-1} \tilde{W}) \tag{23}$$

where tr represents the trace of a matrix.

$$\begin{aligned} \Delta V(\tilde{W}_i) &= V(\tilde{W}_{i+1}) - V(\tilde{W}_i) \\ &= tr(\tilde{W}_{i+1}^T \Gamma_{SG}^{-1} \tilde{W}_{i+1}) - tr(\tilde{W}_i^T \Gamma_{SG}^{-1} \tilde{W}_i) \\ &= tr((\tilde{W}_i - \Gamma_{SG}\phi_i e_i^T)^T \Gamma_{SG}^{-1} (\tilde{W}_i - \Gamma_{SG}\phi_i e_i^T)) \\ &\quad - tr(\tilde{W}_i^T \Gamma_{SG}^{-1} \tilde{W}_i) \\ &= tr(-2\tilde{W}_i^T \phi_i e_i^T + e_i \phi_i^T \Gamma_{SG} \phi_i e_i^T) \\ &= tr(-2e_i e_i^T + e_i \phi_i^T \Gamma_{SG} \phi_i e_i^T) \\ &= -2e_i^T e_i + e_i^T e_i \phi_i^T \Gamma_{SG} \phi_i \\ &= -2e_i^T e_i + e_i^T \phi_i^T \Gamma_{SG} \phi_i e_i \\ &= -e_i^T M_{SG} e_i \end{aligned} \tag{24}$$

where $M_{SG} = 2 - \phi_i^T \Gamma_{SG} \phi_i$. It can be seen that $V_{i+1} - V_i \le 0$ if $M_{SG} > 0$ or $2 - \phi_i^T \Gamma_{SG} \phi_i > 0$ or

$$0 < \lambda_{max}(\Gamma_{SG}) < 2 \tag{25}$$

When (25) is satisfied, $V(\tilde{W}) \ge 0$ is non-increasing in i and the limit

$$\lim_{k\to\infty} V(\tilde{W}) = V_\infty \tag{26}$$

exists. From (24),

$$\begin{aligned} V_{i+1} - V_i &= -e_i^T M_{SG} e_i \\ \sum_{i=0}^{\infty}(V_{i+1} - V_i) &= -\sum_{i=0}^{\infty} e_i^T M_{SG} e_i \end{aligned} \tag{27}$$

$$\Rightarrow \sum_{i=0}^{\infty} e_i^T M_{SG} e_i = V(0) - V_\infty < \infty \tag{28}$$

Also,

$$\sum_{i=0}^{\infty} e_i^T I e_i \le \sum_{i=0}^{\infty} e_i^T M_{SG} e_i < \infty \tag{29}$$

when $M_{SG} > I$ or when

$$\lambda_{max}(\Gamma_{SG}) < 1. \tag{30}$$

Hence, when (30) is satisfied, $e_i \in \mathbf{L}_2$. From (19), $(W_{i+1} - W_i) \in \mathbf{L}_2 \cap \mathbf{L}_\infty$. Using discrete time Barbalat's lemma [36],

$$\lim_{i\to\infty} e_i = 0 \tag{31}$$

$$\lim_{i\to\infty} W_{i+1} = W_i \tag{32}$$

Hence, the SGD learning law in (19) guarantees that the estimated output $\hat{y}_i$ converges to the actual output y_i and the model parameters W converge to some constant values. The parameters converge to the true parameters W_* only under conditions of persistence of excitation [35] in input signals of the system (amplitude and frequency richness of x). Further, using boundedness of V_i, $e_i \in \mathbf{L}_\infty$ which guarantees that the online model predictions are bounded as long as the system output is bounded. As the error between the true model and the estimation model converges to zero, the estimation model becomes a one-step ahead predictive model of the nonlinear system. In the next few sections, we evaluate our SG-ELM algorithm on a HCCI engine identification problem.

4. Homogeneous charge compression ignition engine

This section gives an overview of the homogeneous charge compression ignition engine system and experimentation. The engine specifications are listed in Table 1 [11]. HCCI is achieved by auto-ignition of the gas mixture in the cylinder. The fuel is injected early in the intake stroke and given sufficient time to mix with air forming a homogeneous mixture. A large fraction of exhaust gas from the previous cycle is retained to elevate the temperature and reaction rates of the fuel and air mixture. The variable valve timing capability of the engine enables trapping suitable quantities of exhaust gas in the cylinder.

The engine control knobs include injected fuel mass (FM in mg/cyc), crank angle at intake valve opening (IVO), crank angle at exhaust valve closing (EVC), crank angle at start of fuel injection (SOI). The valve events are measured in degrees after exhaust top dead center (deg eTDC) while SOI is measured in degrees after combustion top dead center (deg cTDC). Other important physical variables that influence the performance of HCCI combustion include intake manifold temperature T_{in}, intake manifold pressure P_{in}, mass flow rate of air at intake $\dot{m}_{in}$, exhaust gas temperature T_{ex}, exhaust manifold pressure P_{ex}, coolant temperature T_c, fuel to air ratio (FA) etc. The engine performance metrics are given by combustion phasing indicated by the crank angle at 50% mass fraction burned (CA50), combustion work output given by net indicated mean effective pressure (IMEP[1]). For further reading on HCCI combustion and related variables, please refer [37].

4.1. Experiment design

In order to perform HCCI engine identification, suitable experiments to obtain dynamic data from the engine needs to be designed. The modeled variables such as engine states and operating envelope are dynamic variables. In order to capture both transient and steady state behavior, a set of dynamic experiments

[1] IMEP and NMEP has been interchangeably used in this paper although both refers to the net quantity.

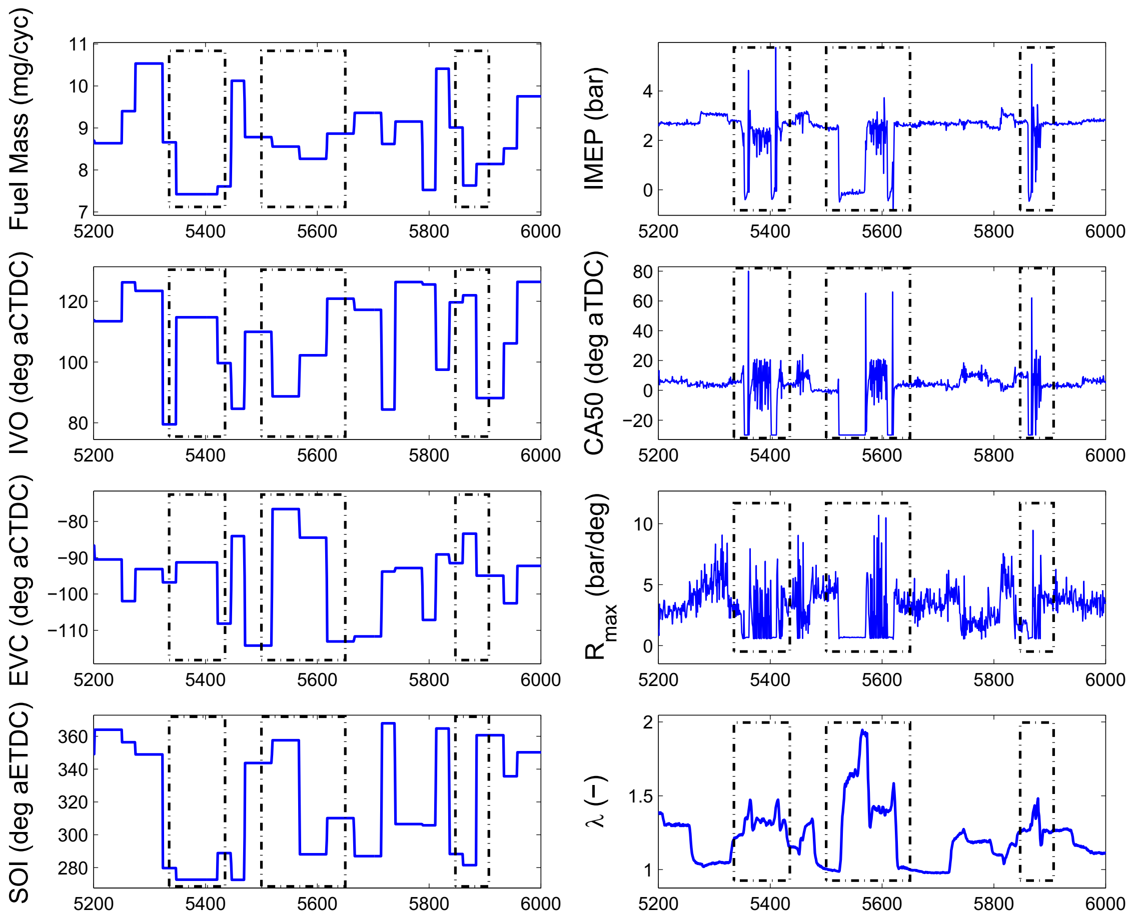

Fig. 1. A subset of the HCCI engine experimental data showing A-PRBS inputs and engine outputs. The misfire regions are shown in dotted rectangles. The data is indexed by combustion cycles.

is conducted at constant rotational speeds and naturally aspirated conditions (no supercharging/turbocharging) by varying FM, IVO, EVC and SOI in a uniformly random manner. At every step change, the engine makes a transition between two set points and the transition is recorded as temporal data. In order to capture several such transients, an amplitude modulated pseudo-random binary sequence (A-PRBS) has been used to design the excitation signals for FM, IVO, EVC and SOI. A-PRBS excites the engine at different amplitudes and frequencies exploring the operating space of the engine for the identification problem considered in this work.

4.2. HCCI instabilities

A subset of the data collected from the engine is shown in Fig. 1 [12] where it can be observed that for some combinations of the inputs (left figures), the HCCI engine misfires (shown by dotted rectangles). HCCI operation is limited by several phenomena that lead to undesirable engine behavior. As described in [38], the HCCI operating range is constrained to a small region of permissible unburned (pre-combustion) and burned (post-combustion) charge temperature states. A sufficiently high unburned gas temperatures are required to achieve ignition in the HCCI operating range without which a complete misfire will occur. If the resulting combustion cannot achieve sufficiently high burned gas temperatures, commonly occurring in conditions with low fuel to diluent ratios or late combustion phasing, various degrees of quenching can occur resulting in reduced work output and increased hydrocarbon and carbon monoxide emissions. Under some conditions, this may lead to high cyclic variation due to the positive feedback loop existing through the trapped residual gas [15,16]. Operation with high burned gas temperature, although stable and commonly reached at higher fueling rates where the fuel to diluent ratio is also high, yields high heat release and thus pressure rise rates that may pose challenges for engine noise and durability constraints. A discussion of the temperatures at which these phenomena occur may be found in [38].

HCCI operation is limited by a combination of the above instabilities and during transients, it may be challenging to reactively respond to instabilities. Thus, a proactive means by which such instabilities can be determined is by developing a predictive model for the operating envelope of the engine discussed in Section 6.

4.3. Learning the HCCI engine data

In the HCCI modeling problem, both the inputs and the outputs of the engine are available as sensor measurements. The HCCI engine is a nonlinear dynamic system and sensor measurements represent discrete time sequences. The input–output mapping can be modeled using a nonlinear auto regressive model with exogenous input (NARX) [39] as follows:

$$\begin{aligned} y(k) = f_{NARX}[&u(k-1),..., u(k-n_u), \\ &y(k-1),..., y(k-n_y)] \end{aligned} \qquad (33)$$

where $u(k) \in \mathbb{R}^{u_d}$ and $y(k) \in \mathbb{R}^{y_d}$ represent the inputs and outputs of the system respectively, k represents the discrete time index, $f_{NARX}(.)$ represents the nonlinear function mapping specified by the model, n_u, n_y represent the number of past input and output samples required (order of the system) while u_d and y_d represent the dimension of inputs and outputs respectively. Let x represent the augmented input vector obtained by time-embedding the input and output measurements from the system.

$$x = [u(k-1)\ ,..,\ u(k-n_u), y(k-1)\ ,..,\ y(k-n_y)]^T \tag{34}$$

The measurement sequence can be converted to the form of training data

$$\{(x_1, y_1), \ldots, (x_N, y_N)\} \in (\mathcal{X}, \mathcal{Y}) \tag{35}$$

where N denotes the number of training samples, $\mathcal{X}$ denotes the space of the input features (Here $\mathcal{X} = \mathbb{R}^{u_d n_u + y_d n_y}$ and $\mathcal{Y} = \mathbb{R}$ for regression and $\mathcal{Y} = [+1, -1]$ for a binary classification). The above conversion of system measurements to training data is natural for a series-parallel model architecture, that can be used to perform a one-step ahead prediction (OSAP) i.e., given a set of measurements until time index k, the model predicts the output at time $k+1$ (see Eq. (36)). A parallel architecture on the other hand is used to perform multiple step ahead predictions (MSAP)[2] by feeding back the predictions of the OSAP model in a recurrent manner (see Eq. (37)). The series-parallel and parallel architectures are well explained in [40].

$$\hat{y}(k+1) = \hat{f}_{NARX}[u(k)\ ,..,\ u(k-n_u+1), y(k)\ ,..,\ y(k-n_y+1)] \tag{36}$$

$$\begin{aligned}\hat{y}(k+n_{pred}) = \hat{f}_{NARX}\,[&u(k+n_{pred}-1)\ ,..,\ u(k-n_u+n_{pred}),\\ &\hat{y}(k+n_{pred}-1)\ ,..,\ \hat{y}(k-n_y+n_{pred})]\end{aligned} \tag{37}$$

The OSAP model is used for training as existing simple training algorithms can be used and once the model becomes accurate for OSAP, it can be converted to a MSAP model in a straightforward manner. The MSAP model can be used for making long term predictions useful for predictive control [5,41].

5. Application case study 1: online regression learning for system identification of an hcci engine.

The problem considered in this case study is to develop a predictive model of the state variables of the HCCI engine using online learning. The state variables of an engine are the fundamental quantities that represent the engine's state of operation. As a consequence, these variables also influence the performance of the engine such as fuel efficiency, emissions and stability, and are required to be monitored/regulated carefully. The significance of the state variables for control and a data based modeling approach were recently analyzed [11,14] where an offline model was developed using archived data. In this paper, an online learning framework for modeling the state variables of HCCI engine such as the IMEP and CA50, is developed and is shown to be comparable to that of the offline models.

The proposed SG-ELM is applied and is compared against OS-ELM for regression performance evaluated using one-step ahead and multi-step ahead predictions. A linear model and an offline trained nonlinear ELM model similar to the one in [11] are included as baselines. The linear baseline model is included to justify the benefits of adopting a nonlinear model while the offline trained model is included to show the effectiveness of online algorithms in capturing the underlying behavior fully in spite of processing data sequentially. The offline ELM model is expected to produce an accurate model as it has sufficient time, computation and utilization of all training data simultaneously to learn the HCCI behavior sufficiently well.

[2] MSAP predictions are necessary for planning trajectories for a given engine operation, such as in the case of optimal control.

5.1. Model setup and evaluation metric

For the purpose of demonstration, the variables IMEP and CA50 are considered as outputs whereas the control variables such as fueling (FM), exhaust valve closing (EVC) and fuel injection timing (SOI) are considered inputs. Transient data from the HCCI engine at a constant speed of 1800 RPM and naturally aspirated conditions is used. A NARX model as shown in Section 4.3 is considered where $u = [FM \quad EVC \quad SOI]^T$ and $y = [IMEP \quad CA50]^T$, n_u and n_y chosen as 1 (tuned by trial and error). The nonlinear model approximating f_{NARX} is initialized to an extreme learning machine model with random input layer weights and random values for the covariance matrices and output layer weights.

All the nonlinear models consist of 100 hidden units with fixed randomized input layer parameters. About 11,000 cycles of data is considered one-by-one as it is sampled by the engine data acquisition and model parameters updated in a sequential manner. After the training phase, the parameter update is switched off and the models are evaluated for the next 5100 cycles of data for one step ahead predictions. Further, to evaluate if the learned models represent the actual HCCI dynamics, the multi-step ahead prediction of the models are compared using about 600 cycles of data. It should be noted that both the one-step ahead and multi-step ahead evaluations are done using data unseen during the training phase.

The parameters of each of the models are tuned for the given dataset. As recommended by OS-ELM [17], the model is initialized prior to online learning using about 800 cycles of data (see Eqs. (14) and (15)). The initialization was performed using the batch ELM algorithm [30]. In order to have a fair comparison, the W_0 is used as an initial condition for both OS-ELM and SG-ELM. The only parameter of SG-ELM, namely the gradient step size was tuned to be $\Gamma_{SG} = 0.0008 \quad I_{100}$ for best accuracy.

The performance of the models are measured using normalized root mean squared error (RMSE) given by

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{y_d}(y_j^i - \hat{y}_j^i)^2} \tag{38}$$

where both y_j^i and $\hat{y}_j^i$ are normalized to lie between -1 and +1.

5.2. Results and discussion

On performing online learning, it can be observed from Fig. 2 that the parameters of OS-ELM grow more aggressively as compared to the SG-ELM. In spite of both models having the same initial conditions, the step size parameter Γ_{SG} for SG-ELM gives additional control over the parameter growth and keeps them bounded as shown in Section 3.2. On the other hand, OS-ELM does not have any control over the parameter evolution. It is governed by the evolution of the co-variance matrix M (see Eq. (14)). It is expected that the co-variance matrix M would add stability to the parameter evolution but in practice, it tends to be more aggressive especially when having correlated data and over-parameterized models, leading to potential instabilities as reported in [24,25,20,26,27]. As a consequence, the parameter values for SG-ELM remain small compared to the OS-ELM (the norm of estimated parameters for OS-ELM is 16.64 and SG-ELM is 3.71). This has a significant implication in the statistical learning theory [33]. A small norm of model parameters implies a simpler model which results in good generalization. Although this effect is slightly reflected in the results summarized in prediction results summarized in Table 2 (see SG-ELM having the lowest MSAP RMSE), it is not significantly better for this problem possibly because of insufficient data for convergence. The value of Γ_{SG} has to be tuned correctly along with sufficient training data in order to ensure

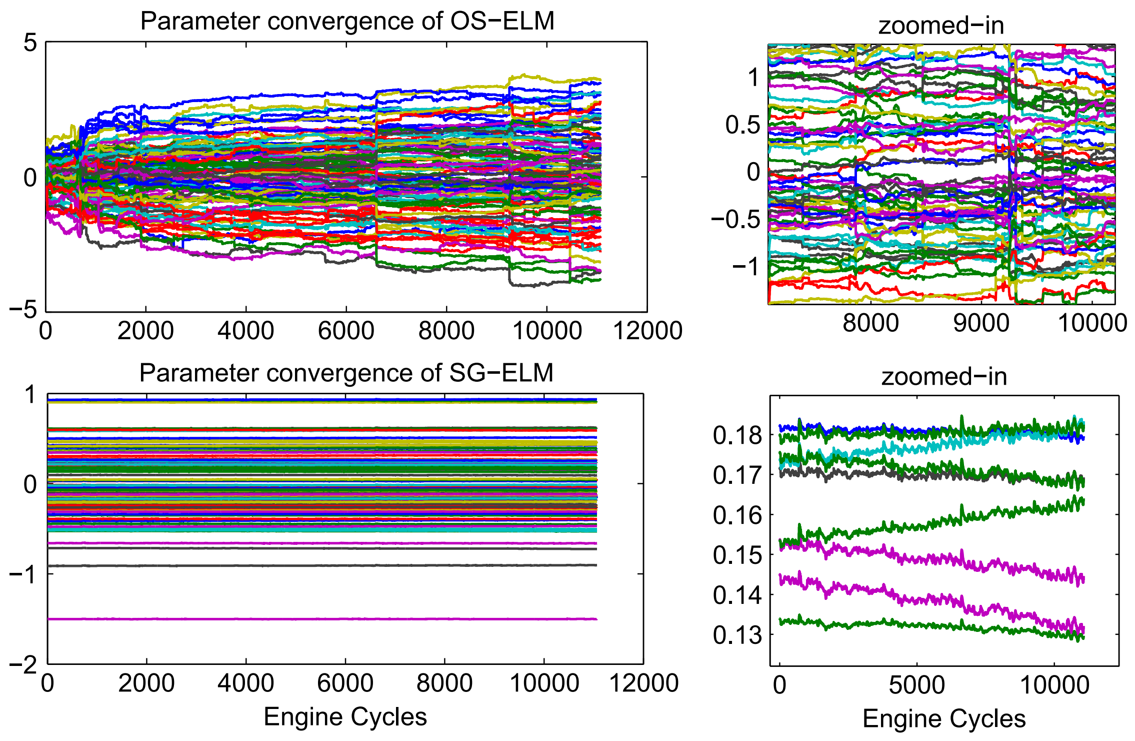

Fig. 2. Comparison of parameter evolution for the OS-ELM and SG-ELM algorithms during online learning (each engine cycle corresponds to a sample of engine data processed by the models). A zoomed-in plot (figures to the right) shows that the parameter update for OS-ELM is more aggressive compared to SG-ELM. Although both OS-ELM and SG-ELM parameters are initialized to the same values, the parameters of OS-ELM continue to grow aggressively compared to the SG-ELM. Note that small parameter values indicate good regularization. This plot gives a qualitative visualization of this behavior.

Table 2
Performance comparison of OS-ELM and SG-ELM for the HCCI online regression learning problem. A baseline linear model and an offline trained ELM model (O-ELM) are also included for comparison. The offline O-ELM algorithm has access to all the data and can use the available memory and computational power, so its training time is not compared with the online algorithms. The RMSE values are averaged over 100 different trials.

	training Time[3] in s	OSAP RMSE	MSAP RMSE
Linear	1.0111	0.1277	0.1859
OS-ELM	9.6563	**0.0952**	0.1018
SG-ELM	**1.5624**	0.1050	**0.0955**
O-ELM	–	0.1018	0.1027

parameter convergence. Ultimately, the online learning mechanism is aimed to run along with the engine and hence the slow convergence may not be an issue in a real application.

The prediction results as well as training time[3] for the online models are compared in Table 2 where each algorithm is run for 100 trials and the average RMSE is reported. It can be observed that the computational time for SG-ELM is significantly less (about 6.2 times) compared to OS-ELM showing the time gain in eliminating the covariance estimation step. The reduction in computation is expected to be more pronounced as the dimension and complexity of the data increase. It could be seen from Table 2 that the one-step ahead prediction accuracies (OSAP RMSE) of the nonlinear models are similar, and OS-ELM winning marginally. On the other hand, the multi-step prediction accuracies (MSAP RMSE) are similar for the nonlinear models with SG-ELM performing marginally better. The MSAP accuracy reflect the generalization performance of the model and is more crucial for the modeling problem as the models ultimately feed its prediction to a predictive control framework that requires accurate and robust predictions of the engine several steps ahead of time. From our understanding on model complexity and generalization error, a model that is less complex (indicated by minimum norm of parameters [30,33]) tend to generalize better, which is again demonstrated by SG-ELM. The performance of the linear baseline model is significantly low compared to the nonlinear models justifying the need for nonlinear models for the HCCI system. In order to show that the results of SG-ELM are statistically significant with respect to OS-ELM, a pairwise t-test is performed [42,43] using 100 bootstrapped sub-sample instances[4]. The p-values of the pairwise t-test for the OSAP RMSE and MSAP RMSE are 2.9632E − 96 and 3.8491E − 76, indicating that the results of the SG-ELM is statistically significant from that of the OS-ELM with a very low significance level (high probability that the two algorithms are statistically significant).

The MSAP predictions of the models are summarized in Fig. 3 (a)–(d) where model predictions for NMEP and CA50 are compared against real experimental data. Here the model is initialized using the experimental data at the first instant and allowed to make predictions recursively for several steps ahead. It can be seen that the nonlinear models outperform the linear model and at the same time the online learning models perform similar to the offline trained models indicating that online learning can fully identify the engine behavior at the operating condition where the data is collected. It should be noted that this task is a case of multi-

[3] The training time is the time taken for training without considering tuning of hyper-parameters such as number of hidden neurons, etc.

[4] For the pairwise t-test, the proposed SG-ELM is compared with the competing OS-ELM algorithm and a statistical test is performed with the null hypothesis being that the two algorithms are not statistically significant. About 100 trials were performed with a subset of data being sampled with replacement from the training data set for both algorithms and the OSAP RMSE and MSAP RMSE were measured. Using the 100 values of OSAP and MSAP RMSE for both SG-ELM and OS-ELM, the pairwise t-test was carried out.

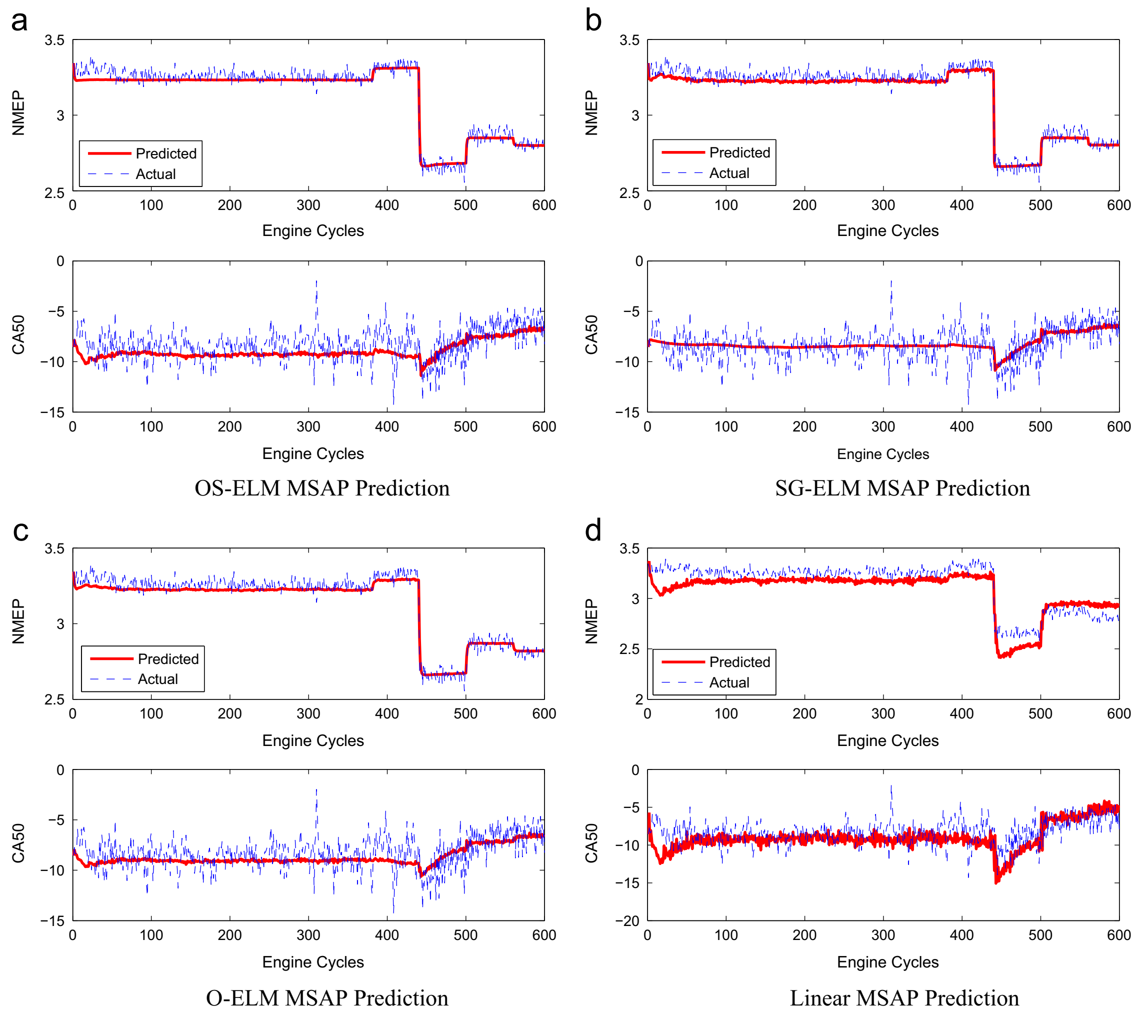

Fig. 3. (a) OS-ELM MSAP prediction, (b) O-ELM MSAP prediction, (c) Linear MSAP prediction. 600-step ahead prediction results of OS-ELM, SG-ELM, O-ELM and the linear model are compared. The OS-ELM, SG-ELM and linear models are learned online using 11,000 cycles of data while the O-ELM model is trained using the same data but in an offline manner. The 600-step ahead prediction is performed on an unseen dataset. For each of the models, NMEP and CA50 predictions are compared to the experimentally recorded values. It has to be noted that the control inputs at the 600 cycles are used while NMEP and CA50 are recurrently fed back to the model to perform multi-step ahead predictions.

input multi-output modeling which adds some limitations to the SG-ELM methods. When the model complexity increases, the SG-ELM may require more excitations for convergence, as opposed to OS-ELM which converges more aggressively (although possibly losing stability). Further, tuning the learning rate Γ_{SG} may be time-consuming for models predicting multiple outputs with different noise characteristics and stability requirements.

6. Application case study 2: online classification learning (with class imbalance) for identifying the dynamic operating envelope of an HCCI engine

The problem considered in this case study is to model the dynamic operating envelope of the HCCI engine using online learning. The dynamic operating envelope of an engine can be defined as the stable operating space of the engine. The significance of the operating envelope and data based modeling approaches were recently introduced [13] where an offline model was developed using archived data. In this paper, an online learning framework for modeling the operating envelope of HCCI engine is developed is shown to be as good as the offline models in determining the HCCI operating envelope.

We consider the operating envelope defined by two common HCCI unstable modes – a complete misfire and a high variability combustion (a more detailed description is given in Section 4.2) is studied. The problem of identifying the HCCI operating envelope using experimental data can be posed as a binary classification problem. The engine sensor data can be labeled as being stable or unstable depending on some engine based heuristics [13]. Further, the engine dynamic data consists of a large number of stable class data compared to unstable class data, which introduces a class imbalance. A cost-sensitive approach that modifies the objective function of the learning system to weigh the minority class data more heavily, is preferred over under-sampling and over-sampling approaches [13] and is used in this study.

The proposed SG-ELM is applied and is compared against OS-ELM for classification performance. A linear classification model and an offline trained nonlinear ELM model similar to the one in [13] are included as baselines to make similar justifications as in the previous case study. The linear baseline model is included to justify the benefits of adopting a nonlinear model while the offline trained model is included to show the effectiveness of online algorithms in capturing the underlying behavior fully in spite of processing data sequentially.

Table 3
Performance comparison of the nonlinear models (OS-ELM and SG-ELM) for the online class imbalance learning problem. A baseline linear model and an offline trained ELM model (O-ELM) are also used for comparison. The O-ELM results are included for comparing classification accuracies. The offline O-ELM algorithm has access to all the data and can use the available memory and computational power, so its training time is not compared with the online algorithms. All values reported are averaged over 100 different trials.

Algorithms	Training time[6] in s	TPR	TNR	Total accuracy	GM accuracy
Linear	1.8648	0.9996	0.5665	0.7830	0.7524
OS-ELM	1.8443	0.7000	0.8713	0.7857	0.7791
SG-ELM	1.6011	0.8605	0.6923	0.7764	0.7714
O-ELM	–	0.7530	0.8426	0.7978	0.7947

Table 4
Comparison of best performance of the nonlinear models after fine tuning the parameters and with a good set of initial conditions. These models are used for final prediction of the operating envelope of the HCCI engine.

Algorithms	Training Time[6] in s	TPR	TNR	Total Accuracy	GM Accuracy
OS-ELM	1.8374	0.8328	0.8341	0.8335	0.8335
SG-ELM	**0.9822**	**0.9876**	0.7707	**0.8792**	**0.8725**
O-ELM	-	0.8265	**0.8569**	0.8417	0.8416

6.1. Model setup and evaluation metric

The HCCI operating envelope is a function of the engine control inputs and engine physical variables such as temperature, pressure, flow rate, etc. Also, the envelope is a dynamic system and so a predictive model requires the measurement history up to an order of N_h. The dynamic classifier model can be given by

$$\hat{y}_{k+1} = sgn(f(x_k)) \tag{39}$$

where $sgn(.)$ represents the sign function, $\hat{y}_{k+1}$ indicates model prediction for the future cycle $k+1$, $f(.)$ can take any structure depending on the learning algorithm and x_k is given by

$$x_k = [IVO, EVC, FM, SOI, T_{in}, P_{in}, \dot{m}_{in}, \\ T_{ex}, P_{ex}, T_c, FA, IMEP, CA50]^T \tag{40}$$

at cycle k up to cycle $k-N_h+1$. In the following sections, the function $f(.)$ is learned using the available engine experimental data using OS-ELM and SG-ELM algorithms. The engine measurements and their time histories (defined by x_k) are considered inputs to the model while the stability labels are considered outputs. The feature vector is of dimension $n=39$ includes sensor measurements such as FM, IVO, EVC, SOI, T_c, T_{in}, P_{in}, $\dot{m}_{in}$, T_{ex}, P_{ex}, IMEP, CA50 and FA along with $N_h=1$ cycles of history (see (40)). The engine experimental data is split into training and testing sets. The training set consists of about 14300 cycles of data processed one-by-one as sampled by the engine data acquisition. After the training phase, the parameter update is switched off and the models are evaluated for the next 6200 cycles of data for one step ahead classification. The ratio of number of majority class data to number minority class data (r) for the training set is about 4.5 and for the testing set is 9. The nonlinear model approximating $f(.)$ is initialized to an extreme learning machine model with random input layer weights and random values for the covariance matrices and output layer weights. All the nonlinear models consist of 10 hidden units with fixed randomized input layer parameters. Similar to the previous case study, a small portion of the training data is used to initialize the ELM model parameters as well as the covariance matrix. The SG-ELM parameter Γ_{SG} is tuned to be 0.001 I_{10} using trial and error. A weighted classification version of the algorithms is developed to handle the class imbalance problem. The minority class data is weighted higher by r times f_s where r is the imbalance ratio of the training data and is computed online as the ratio of the number of majority class to number of minority class data until that instant.

For the class imbalance problem considered here, a conventional classifier metric like the overall misclassification rate cannot be used as it would find a biased classifier, i.e., it would find a classifier that ignores the minority class data. For instance, a data set that has 95% of majority class data (with label $+1$) would achieve 95% classification accuracy by predicting all the labels to be $+1$ which is obviously undesirable. Hence the following evaluation metric used for skewed data sets is considered. Let TP and TN represent the total number of positive (stable operation) and negative class (unstable modes) data classified correctly by the classifier. If N^+ and N^- represent the total number of positive and negative class data respectively, the true positive rate (TPR) and true negative rate (TNR), geometric mean (GM) of TPR and TNR, and the total accuracy (TA) of the classifier can be defined as follows [44]. It should be noted that the total accuracy and geometric mean weights the accuracy of majority and minority classes equally, i.e., they have high values only when both classes of data are classified correctly.

$$TPR = \frac{TP}{N^+} \\ TNR = \frac{TN}{N^-} \\ GM = \sqrt{TPR \times TNR} \\ TA = 0.5(TPR+TNR). \tag{41}$$

6.2. Results and discussion

The results of online imbalance classification can be summarized in Table 3 where computational time as well as classification performance can be compared based on 100 different trials. It can be observed that for the HCCI classification problem, all the models perform with similar average accuracies. Both OS-ELM and SG-ELM achieve results similar to an offline model indicating completeness of learning. The SG-ELM has a slight advantage in terms of training time because of the simplicity of SG-ELM compared to OS-ELM.

In the experiments above, it should be noted that for different initializations of the 100 trials, the model's hyper-parameters are not fine-tuned and so it may be possible to achieve better performance by fine-tuning. A further experiment is conducted where the hyper-parameters of each algorithm are fine-tuned for one particular initialization so that the best model is identified for further engine controls development. Ignoring the results of the linear model (that had a large imbalance in TPR and TNR similar to the average results in Table 3), the results of the fine-tuned nonlinear models are reported in Table 4. It can be seen that the accuracies of all the algorithms improved significantly with SG-ELM slightly better and with a stability guarantee, indicates the suitability of SGD based online learning for the HCCI problem. A subtle advantage observed for the OS-ELM is that, although the combined accuracy is slightly inferior to that of the SG-ELM, the accuracies of the positive examples and negative examples are very close to each other indicating that the model is well balanced to predict both majority class as well as minority class data well. The SG-ELM on the other hand, in spite of fine-tuning the parameters, fails to achieve this. A further tuning can be done to improve the accuracy of a particular class of data, typically sacrificing some accuracy predicting the other. In order to show that the results of SG-ELM are statistically significant with respect to

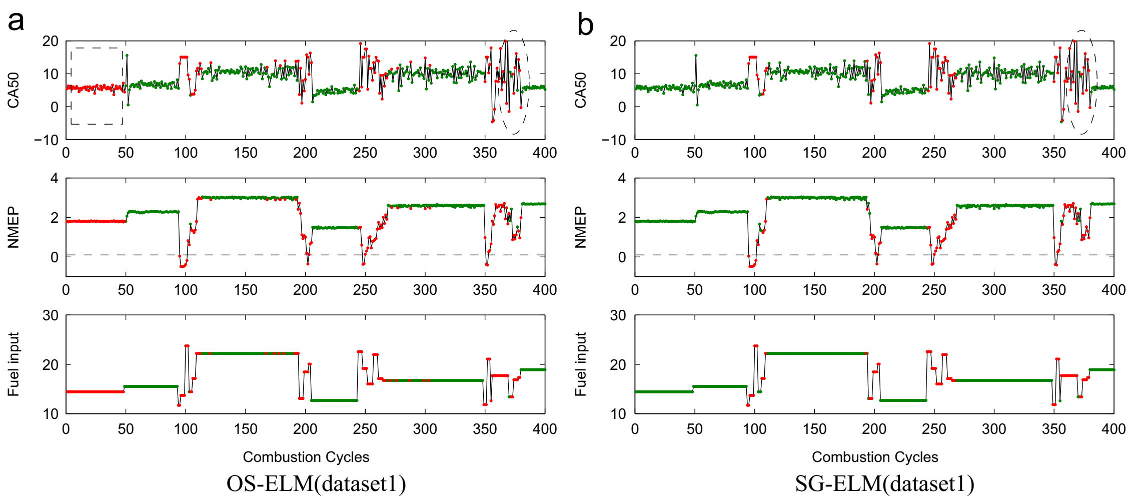

Fig. 4. (a) OS-ELM (dataset 1) and (b) SG-ELM (dataset 1). Online classification results of OS-ELM and SG-ELM models showing CA50, NMEP and one input variable (fueling) for 2 different unseen data sets. The color code indicates model prediction - green (and red) indicate stable (and unstable) prediction by the model. The horizontal dotted line in the NMEP plot indicates misfire limit, dotted ellipse in CA50 plot indicates high variability instability mode while dotted rectangle shows false alarms by model.(For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

OS-ELM, a pairwise t-test is performed [42,43] using 100 bootstrapped sub-sample instances[5]. The p-value of the pairwise t-test is 0.0208 which indicates that the results of the SG-ELM is statistically significant from that of the OS-ELM with a significance level of 5%.

The models developed using OS-ELM and SG-ELM algorithms are used to make predictions on unseen engine inputs and class predictions are summarized in Fig. 4, while quantitative results are included in Table 3. As mentioned earlier, the operating envelope is a decision boundary in the input space within which any input operates the HCCI in a stable manner and any input outside the envelope might operate the engine in an unstable manner. The HCCI state variables such as IMEP, CA50 and engine sensor observations such as $T_{in}, P_{in}, \dot{m}_{in}, T_{ex}, P_{ex}, T_c$ at time instant k, along with engine control inputs such as FM, EVC, SOI at time instant $k+1$, are given as input to the models (see (40)). The model predictions at time $k+1$ are obtained. The engine's actual response at time $k+1$ is also recorded. A data point is marked in red if the model predicts the engine operation to be unstable (labeled as -1) while it is marked in green if the model predicts the data point to be stable (labeled as $+1$). In the figures, a dotted line in the NMEP plot indicates the misfire limit, a dotted ellipse in CA50 plot indicates high variability instability mode while a dotted rectangle indicates misclassified predictions by model. To understand the variation of NMEP and CA50 with changes in control inputs, the fueling input (abbreviated as FM) is also included in the plots. It should be noted that FM is not the only input for prediction and the signals are defined as in Eq. (40) but only the fueling input is shown in the plots owing to space constraints.

It can be seen from the above plots that as a whole, both OS-ELM and SG-ELM models classify the HCCI engine data fairly well in spite of the high amplitude noise inherent in the HCCI experimental data. The data consists of step changes in FM, EVC and SOI and whenever a 'bad' combination of inputs is chosen, the engine either misfires completely (see NMEP fall below misfire limit) or exhibits high variability combustion (see dotted ellipses). The goal of this work as stated previously, is to predict if a future HCCI combustion event is stable or unstable based on available measurements. The results summarized in Table 3 indicates that the developed models indeed accomplished the goal with a reasonable accuracy. From Fig. 4, it is observed that the OS-ELM has some clear false alarms in predicting stable class data (see dotted rectangles in the plots) while this is not observed for SG-ELM. This is not surprising as the false alarm rate of SG-ELM (see Table 3) is very low[6]. On the other hand, the SG-ELM has an inferior TNR. By adjusting the weighting factor Γ_{imb} in Eq. (20), one can achieve a required tradeoff between TPR and TNR as desired in the application.

7. Conclusion

A stochastic gradient descent based online learning algorithm for ELM has been developed, that guarantees stability in parameter estimation suitable for control purposes. Further, the SG-ELM demands less computation compared to the OS-ELM algorithm, as the covariance estimation step is eliminated. A stability proof is developed based on Lyapunov approach. However, the SG-ELM algorithm might involve tedious tuning of step-size parameter as well as suffer from slow convergence. The tuning of step-size parameter and convergence properties of SG-ELM will be considered for future work.

The SG-ELM and OS-ELM algorithms are applied to develop online models for state variables and dynamic operating envelope of a HCCI engine to assist in model based control. The results from this paper suggest that good generalization performance can be achieved using both OS-ELM and SG-ELM methods but the SG-ELM might have an advantage in terms of stability, crucial for designing robust control systems.

Although the SG-ELM appears to perform well in the HCCI identification problem, a comprehensive analysis and evaluation on several benchmark data sets is required and will be considered for future. From an application perspective, interesting areas for exploration include implementing the algorithm in real-time

[5] For the pairwise t-test, the proposed SG-ELM is compared with the competing OS-ELM algorithm and a statistical test is performed with the null hypothesis being that the two algorithms are not statistically significant. About 100 trials were performed with a subset of data being sampled with replacement from the full data set for both algorithms and the total accuracy was measured. Using the 100 values of total accuracy for both SG-ELM and OS-ELM, the pairwise t-test was carried out.

[6] The false alarm rate of SG-ELM is about 1.24%. In this study, the label of -1 corresponds to a 'bad' data and so false alarm rate corresponds to false negative rate which is 1-TPR

hardware, exploring a wide operating range of HCCI operation and development of controllers.

Disclaimer

Acknowledgements

This material is based upon work supported by the Department of Energy [National Energy Technology Laboratory] under Award number(s) DE-EE0003533. This work is performed as a part of the ACCESS project consortium (Robert Bosch LLC, AVL Inc., Emitec Inc.) under the direction of PI Hakan Yilmaz, Robert Bosch, LLC. Prof. X. Nguyen is supported in part by NSF Grants CCF-1115769 and ACI-1047871.

References

[1] R. Thring, Homogeneous-charge compression-ignition engines, SAE (1989) 892068.
[2] M. Christensen, P. Einewall, B. Johansson, Homogeneous charge compression ignition using iso-octane, ethanol and natural gas – a comparison to spark ignition operation, in: International Fuels & Lubricants Meeting & Exposition, Tulsa, OK, USA, 1997, (SAE paper 972874).
[3] T. Aoyama, Y. Hattori, J. Mizuta, Y. Sato, An experimental study on premixed-charge compression ignition gasoline engine, in: International Congress & Exposition, Detroit, MI, USA, 1996, (SAE paper 960081).
[4] K. Chang, A. Babajimopoulos, G. A. Lavoie, Z. S. Filipi, D. N. Assanis, Analysis of load and speed transitions in an HCCI engine using 1-D cycle simulation and thermal networks, SAE international, 2006.
[5] J. Bengtsson, P. Strandh, R. Johansson, P. Tunestal, B. Johansson, Model predictive Control of Homogeneous Charge Compression Ignition (HCCI) engine dynamics, in: 2006 IEEE International Conference on Control Applications, 2006.
[6] Y. Wang, S. Makkapati, M. Jankovic, M. Zubeck, D. Lee, control oriented model and dynamometer testing for a single-cylinder, heated-air HCCI engine, SAE international, 2009.
[7] Y. Urata, M. Awasaka, J. Takanashi, T. Kakinuma, T. Hakozaki, A. Umemoto, A study of gasoline-fuelled HCCI engine equipped with an electromagnetic valve train, SAE International, 2004.
[8] R. Scaringe, C. Wildman, W.K. Cheng, On the high load limit of boosted gasoline HCCI engine operating in NVO mode, SAE, Int. J. Engines 3 (2010) 35–45.
[9] C. Chiang, C. Chen, Constrained control of homogeneous charge compression ignition (HCCI) engines, in: 5th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2010.
[10] N. Ravi, M. J. Roelle, H. H. Liao, A. F. Jungkunz, C. F. Chang, S. Park, J. C. Gerdes, Model-based control of HCCI engines using exhaust recompression, in: IEEE Transactions on Control Systems Technology, 2009.
[11] V.M. Janakiraman, X. Nguyen, D. Assanis, Nonlinear identification of a gasoline HCCI engine using neural networks coupled with principal component analysis, Appl. Soft Comput. 13 (5) (2013) 2375–2389.
[12] V. M. Janakiraman, X. Nguyen, J. Sterniak, D. Assanis, A system identification framework for modeling complex combustion dynamics using support vector machines, in: Informatics in Control, Automation and Robotics, Vol. 283 of Lecture Notes in Electrical Engineering, Springer International Publishing, 2014, pp. 297–313.
[13] V.M. Janakiraman, X. Nguyen, J. Sterniak, D. Assanis, Identification of the dynamic operating envelope of HCCI engines using class imbalance learning, IEEE Trans. Neural Netw. Learn. Syst. 26 (1) (2015) 98–112.
[14] V.M. Janakiraman, X. Nguyen, D. Assanis, An ELM based predictive control method for HCCI engines, Eng. Appl. Artifi. Intell. 48 (2016) 106–118, http://dx.doi.org/10.1016/j.engappai.2015.10.007.
[15] G.T. Kalghatgi, R.A. Head, Combustion limits and efficiency in a homogeneous charge compression ignition engine, Int. J. Engine Res. 7 (2006) 215–236.
[16] M. Shahbakhti, C.R. Koch, Characterizing the cyclic variability of ignition timing in a homogenous charge compression ignition engine fueled with N-heptane/iso-octane blend fuels, Int. J. Engine Res. 9 (2008) 361–397.
[17] N. Liang, G. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (6) (2006) 1411–1423.
[18] Y. Jun, M. J. Er, An enhanced online sequential extreme learning machine algorithm, in: Control and Decision Conference, 2008.CDC 2008. Chinese, 2008, pp. 2902–2907.
[19] B. Mirza, Z. Lin, K.-A. Toh, Weighted online sequential extreme learning machine for class imbalance learning, Neural Process. Lett. 38 (3) (2013) 465–486.
[20] M. T. Hoang, H. Huynh, N. Vo, Y. Won, A robust online sequential extreme learning machine, in: Advances in Neural Networks, Vol. 4491 of Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2007, pp. 1077–1086.
[21] V. M. Janakiraman, Machine learning for identification and optimal control of advanced automotive engines (Dissertation) University of Michigan, Ann Arbor, ⟨http://hdl.handle.net/2027.42/102392⟩.
[22] J. Kivinen, A. Smola, R. Williamson, Online learning with kernels, IEEE Trans. Signal Process. 52 (8) (2004) 2165–2176.
[23] A. Bordes, L. Bottou, The huller: a simple and efficient online SVM, in: In Machine Learning: ECML 2005, Lecture Notes in Artificial Intelligence, LNAI 3720, Springer Verlag, 2005, pp. 505–512.
[24] G. Zhao, Z. Shen, C. Miao, Z. Man, On improving the conditioning of extreme learning machine: a linear case, in: Proceedings of the 7th International Conference on Information, Communications and Signal Processing, 2009, ICICS 2009., 2009, pp. 1–5.
[25] F. Han, H.-F. Yao, Q.-H. Ling, An improved extreme learning machine based on particle swarm optimization, in: Bio-Inspired Computing and Applications, Lecture Notes in Computer Science.
[26] H.T. Huynh, Y. Won, Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks, Pattern Recognit. Lett. 32 (14) (2011) 1930–1935.
[27] V. M. Janakiraman, X. Nguyen, D. Assanis, A lyapunov based stable online learning algorithm for nonlinear dynamical systems using extreme learning machines, in: Neural Networks (IJCNN), The 2013 International Joint Conference on, 2013, pp. 1–8. http://dx.doi.org/10.1109/IJCNN.2013.7090813.
[28] V. Akpan, G. Hassapis, Adaptive predictive control using recurrent neural network identification, in: Proceedings of the 17th Mediterranean Conference on Control and Automation, 2009. MED '09., 2009, pp. 61–66.
[29] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[30] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B 42 (2) (2012) 513–529.
[31] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Y. Lechevallier, G. Saporta (Eds.), Proceedings of COMPSTAT'2010, Physica-Verlag HD, 2010, pp. 177–186.
[32] N. Le Roux, M. Schmidt, F. Bach, A Stochastic Gradient Method with an Exponential Convergence Rate for Strongly-Convex Optimization with Finite Training Sets, Tech. Rep.arXiv:1202.6258v1, INRIA 2012.
[33] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[34] S. Shalev-Shwartz, N. Srebro, SVM optimization: inverse dependence on training set size, in: Proceedings of the 25th International Conference on Machine learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 928–935.
[35] P. Ioannou, J. Sun, Robust Adaptive Control, PTR Prentice-Hall, 1996.
[36] J. Spooner, M. Maggiore, R. Ordóñez, K. Passino, Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques, Adaptive and Learning Systems for Signal Processing, Communications and Control Series, Wiley, 2004.
[37] F. Zhao, T.N. Asmus, D.N. Assanis, J.E. Dec, J.A. Eng, P.M. Najt, Homogeneous charge compression ignition (HCCI) engines, SAE International (March) .
[38] G.A. Lavoie, J. Martz, M. Wooldridge, D. Assanis, A multi-mode combustion diagram for spark assisted compression ignition, Combust. Flame 157 (6) (2010) 1106–1110.
[39] O. Nelles, Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models, Springer, 2001.
[40] K.S. Narendra, K. Parthasarathy, Identification and Control of Dynamical Systems Using Neural Networks 1 (1) (1990) 4–27.
[41] L. Re, F. Allgöwer, L. Glielmo, C. Guardiola, I. Kolmanovsky, Automotive Model Predictive Control: Models, Methods and Applications, Lecture Notes in Control and Information Sciences, Springer, 2010.
[42] J.-B. Yang, C.-J. Ong, Feature selection using probabilistic prediction of support vector regression, IEEE Trans. Neural Netw. 22 (6) (2011) 954–962.
[43] X. Liu, L. Wang, G.-B. Huang, J. Zhang, J. Yin, Multiple kernel extreme learning machine, Neurocomputing 149, Part A 0 (2015) 253–264.
[44] K.-A. Toh, Deterministic neural classification, Neural Comput. 20 (6) (2008) 1565–1595.

Vijay Manikandan Janakiraman received the bachelor's degree in Mechanical Engineering (2007) from Sri Venkateswara College of Engineering, Chennai, India. He received Master's degrees in Mechanical Engineering (2008) and Electrical Engineering – Systems (2013) and the Ph.D. degree in Mechanical Engineering (2013) from the University of Michigan at Ann Arbor, MI, USA. Since August 2013, he has been working as a research scientist with the Data Sciences Group, Intelligent Systems Division at the NASA Ames Research Center, Moffett Field, CA, USA. His current research interests include machine learning, data mining in high dimensional time series and decision making under uncertainty.

XuanLong Nguyen received the Ph.D. degree in computer science and the M.S. degree in statistics, both from the University of California, Berkeley. He is currently an Assistant Professor of Statistics at the University of Michigan. His research interests lie in distributed and variational inference, nonparametric Bayesian statistics, and applications to detection/estimation problems in distributed and adaptive systems. Dr. Nguyen is a recipient of the CAREER award from the NSF Division of Mathematical Sciences, the Leon O. Chua Award from the UC Berkeley, the IEEE Signal Processing Society's Young Author Best Paper award, and an Outstanding Paper award from the International Conference on Machine Learning.

Dennis Assanis received the Ph.D. degree in Power and Propulsion and the M.S. degrees in Naval Architecture and Marine Engineering and Mechanical Engineering from the Massachusetts Institute of Technology. Dr. Assanis is a Professor in the Department of Mechanical Engineering and is also the Provost, Senior Vice President for Academic Affairs, and Vice President for Brookhaven Affairs at the Stonybrook University, NY, USA. Assanis served as the Jon R. and Beverly S. Holt Professor of Engineering and Arthur F. Thurnau Professor at the University of Michigan, as well as Director of the Michigan Memorial Phoenix Energy Institute, Founding Director of the US–China Clean Energy Research Center for Clean Vehicles and Director of the Walter E. Lay Automotive Laboratory. Dr. Assanis' research interests lie in the thermal sciences and their applications to energy conversion, power and propulsion, and automotive systems design. His research focuses on analytical and experimental studies of the thermal, fluid and chemical phenomena that occur in internal combustion engines, after-treatment systems, and fuel processors. His efforts to gain new understanding of the basic energy conversion processes have made significant impact in the development of energy and power systems with significantly improved fuel economy and dramatically reduced emissions. His group's research accomplishments have been published in over 250 articles in journals and international conference proceedings. Dr. Assanis is a Member of the National Academy of Engineering and is an ASME and SAE Fellow.