OnMLM: An Online Formulation for The Minimal Learning Machine

Alan L. S. Matias[1], César L. C. Mattos[1], Tommi Kärkkäinen[2], João P. P. Gomes[1], and Ajalmar R. da Rocha Neto[3]

[1] Federal University of Ceará, Fortaleza-CE, Brazil,
matiasalsm@gmail.com, cesarlincoln@dc.ufc.br, jpaulo@lia.ufc.br
[2] University of Jyvaskyla, Jyvaskyla, Finland
tommi.karkkainen@jyu.fi
[3] Federal Institute of Ceará, Fortaleza-CE, Brazil,
ajalmar@ifce.edu.br

Abstract. Minimal Learning Machine (MLM) is a nonlinear learning algorithm designed to work on both classification and regression tasks. In its original formulation, MLM builds a linear mapping between distance matrices in the input and output spaces using the Ordinary Least Squares (OLS) algorithm. Although the OLS algorithm is a very efficient choice, when it comes to applications in big data and streams of data, online learning is more scalable and thus applicable. In that regard, our objective of this work is to propose an online version of the MLM. The Online Minimal Learning Machine (OnMLM), a new MLM-based formulation capable of online and incremental learning. The achievements of OnMLM in our experiments, in both classification and regression scenarios, indicate its feasibility for applications that require an online learning framework.

Keywords: online learning, incremental learning, stochastic optimization, Minimal Learning Machine.

1 Introduction

The learning algorithm of many popular machine learning methods, such as the Least Squares Support Vector Machine (LSSVM) [14] and the Extreme Learning Machine (ELM) [3], are based on a batch method, such as the Ordinary Least Squares (OLS). Even though batch approaches perform well in some cases, they present some major drawbacks: (i) they lack scalability for larger datasets; (ii) they cannot be used in scenarios where data are sequentially made available [7].

Both aforementioned issues are usually tackled by online learning techniques. For instance, the classical backpropagation algorithm [8] has been fundamental to enable the use of machine learning models in real world applications with large and/or sequential data. Thus, some authors have directed their efforts to propose alternative formulations, both algorithmically and mathematically, for standard batch learning-based methods in order to perform online learning. Santos and Barreto [12] proposed an approach to train the LSSVR following an online method which is able to obtain sparse solutions incrementally from the data. Jian *et al.* [4] proposed the Budget Online LSSVM, which significantly scales LSSVM through an online learning approach. Online variants of the ELM can also be found in the literature, such as the Online Sequential ELM (OS-ELM) [2] and its recent improvement by Matias *et al.* [10]. Efforts torwards online learning can also be found for ensemble methods such as Gradient Boosting [1] and Random Forests [6].

The recently proposed Minimal Learning Machine (MLM) [13] aims to tackle the general supervised learning task also from a batch learning perspective. The MLM considers a linear mapping between input and output distance matrices, computed from the available data points and M reference points randomly chosen from the dataset. In the generalization step, the learned distance map is used to provide an estimate of the distance from the output reference points to the unknown target output value [11]. This mapping assumes that a multi-response linear regression method is applied between the distance matrices. In the original MLM formulation, such solution is obtained via the batch OLS algorithm. Thus, the MLM inherits the lack of scalability and the inability to handle streaming data of other batch learning techniques.

In this context, our objective is to introduce a feasible online and incremental learning formulation for the MLM. Our proposal, named Online MLM (OnMLM), is able to handle both regression and classification tasks in scenarios where batch approaches are not feasible. The OnMLM starts with one input and one output reference points and, as the training is performed, it incrementally increases the multi-response linear system by adding new reference points based on a given criterion. The new mechanisms provided by the OnMLM requires the addition of three new hyperparameters. However, such hyperparameters are very intuitive and can be easily set.

The properties offered by OnMLM makes it suitable and scalable to applications involving large datasets, and since the training is performed online, the OnMLM is also applicable to datastreams scenarios. Also, we must emphasize that OnMLM reaches always sparser solutions in comparison to MLM, as a consequence of its ability to incrementally add new reference points. Furthermore, our formulation for OnMLM can be optimized with any variant of the Stochastic Gradient Descent (SGD) algorithm. Thus, one can use OnMLM with more efficient SGD variants, boosting its generalization performance and speeding the training phase. To summarize, our contribution with this work is an online formulation based on the SGD algorithm for the MLM model which incrementally includes reference points in the online learning step, with the additional benefit of reaching much sparser solutions.

We have organized our paper by presenting a full description of the MLM and the OnMLM models in Section 2 – in the OnMLM formulation, we show a special focus in providing an overview about the newly included three hyperparameters. We then follow with the experiments performed in both regression and classification scenarios in Section 3. Finally, the conclusions we have obtained with this work are presented in Section 4.

2 Methods

2.1 Minimal Learning Machine

Consider a set of input points $\mathcal{X} = \{\boldsymbol{x}_i \in \mathcal{R}^p : i = 1, \ldots, N\}$ and their corresponding outputs $\mathcal{Y} = \{\boldsymbol{y}_i \in \mathcal{R}^q : i = 1, \ldots, N\}$.We can define the following distance matrices $\mathbf{D}$, whose elements are defined by $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{s}_j\|_2$, and $\boldsymbol{\Delta}$ whose elements are given by $\delta_{ij} = \|\boldsymbol{y}_i - \boldsymbol{t}_j\|_2$. The point denoted by $\boldsymbol{r}_j$ and $\boldsymbol{t}_j$, named referrence points, are randomly selected vectors from $\mathcal{X}$ and $\mathcal{Y}$, respectively. Their corresponding sets are defined as $\mathcal{S} = \{\boldsymbol{s}_j \in \mathcal{X} : j = 1, \ldots, M\}$ and $\mathcal{T} = \{\boldsymbol{t}_j \in \mathcal{Y} : j = 1, \ldots, M\}$, where M is the number of reference points. Under the assumption of a linear mapping between $\mathbf{D}$ and $\boldsymbol{\Delta}$, we can obtain a multiresponse regression model

$$\boldsymbol{\Delta} = \mathbf{D}\mathbf{B} + \mathbf{E}, \tag{1}$$

where $\mathbf{E}$ is the residual matrix.

Thus, the objective of the MLM is to obtain the optimal parameters $\hat{\mathbf{B}}$ which minimizes the multivariate residual sum of squares:

$$\arg\min_{\mathbf{B}} \ \mathcal{J}(\mathbf{B}) = \mathrm{tr}[(\boldsymbol{\Delta} - \mathbf{D}\mathbf{B})^T(\boldsymbol{\Delta} - \mathbf{D}\mathbf{B})]. \tag{2}$$

It is noteworthy that the reference points are randomly chosen before optimizing the objective function in Eq.(2). Furthermore, the optimization problem above can be solved in two different ways: (i) with the usual optimal least squares when $M < N$, and (ii) uniquely solvable with the direct inverse of $\mathbf{D}$ when $M = N$. To better understand the training process of the MLM, step by step, we describe its training algorithm in the Algorithm 1.

Algorithm 1 Training procedure for MLM.

```
Input: Training sets $\mathcal{X}$ and $\mathcal{Y}$, and the number of reference points $M$.
Output: The optimal weights $\hat{\mathbf{B}}$, and the sets $\mathcal{S}$ and $\mathcal{T}$.
1: Randomly selects $M$ reference points from $\mathcal{X}$ and their correspond output from $\mathcal{Y}$,
   in order to obtain the sets $\mathcal{S}$ and $\mathcal{T}$, respectively.
2: Compute the input distance matrix $\mathbf{D}$ from $\mathcal{X}$ and $\mathcal{S}$.
3: Compute the output distance matrix $\boldsymbol{\Delta}$ from $\mathcal{Y}$ and $\mathcal{T}$.
4: Calculate $\hat{\mathbf{B}} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\boldsymbol{\Delta}$.
```

Supposing that $\mathbf{B}$ was either estimated ($M < N$) or uniquely solved ($M = N$), for an arbitrary input test point $\boldsymbol{x}_i \in \mathcal{R}^p$, and its representation in the distance space $\boldsymbol{d}_i = [\|\boldsymbol{x}_i - \boldsymbol{s}_1\|_2, \ldots, \|\boldsymbol{x}_i - \boldsymbol{s}_M\|_2]^T$, the MLM will predict the output related to the distance space

$$\hat{\boldsymbol{\delta}}_i = \boldsymbol{d}_i^T\mathbf{B}. \tag{3}$$

The vector $\hat{\boldsymbol{\delta}}_i$ provides an estimate of the geometrical configuration of $\boldsymbol{y}_i$ related to the input test $\boldsymbol{x}_i$ and the reference set $\mathcal{T}$. To recover the estimation of $\boldsymbol{y}_i$ from $\hat{\boldsymbol{\delta}}_i$, MLM solves a multilateration problem. From a geometric view point, locating $\boldsymbol{y}_i \in \mathcal{R}^q$ is equivalent to solve the overdetermined set of M nonlinear equations corresponding to q-dimensional hyper-spheres centered in $\boldsymbol{t}_j$ which passes through $\boldsymbol{y}_i$. Fig. 1 graphically depicts the problem for $q = 2$.

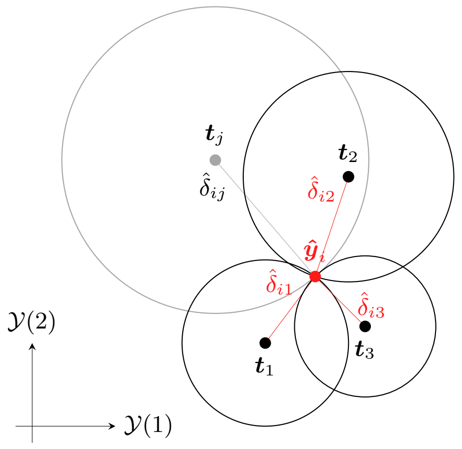

Fig. 1: Output estimation.

Thus, given the set of $j = 1, \ldots, M$ hyper-spheres, each with radius equal to $\hat{\boldsymbol{\delta}}_i$, the location of $\boldsymbol{y}_i$ can be estimated by finding the optimal solution of the following objective function:

$$\underset{\hat{\boldsymbol{y}}_i}{\arg\min}\ \mathcal{J}(\boldsymbol{y}_i) = \sum_{j=1}^{M} [(\boldsymbol{y}_i - \boldsymbol{t}_j)^T (\boldsymbol{y}_i - \boldsymbol{t}_j) - \hat{\delta}_{ij}^2]^2. \tag{4}$$

2.2 Online Minimal Learning Machine (Proposal)

To initiate the OnMLM formulation, we must highlight that, unlike MLM, which have the same number of reference points M in the input set $\mathcal{S}$ and in the output set $\mathcal{T}$, the OnMLM may have different number of reference points M_s and M_t, respectively. The original formulation of MLM adds pairs of points as reference points $(\boldsymbol{x}_i \in \mathcal{X}, \boldsymbol{y}_i \in \mathcal{Y})$ for each i-th indexation chosen to represent a reference point. On the other hand, the OnMLM adds new reference points in $\mathcal{S}$ and $\mathcal{T}$ independently – in fact, the new reference point criterion, which we will be introduced later, determines if an input $\boldsymbol{x}_i$ or its related output $\boldsymbol{y}_i$ will become a new reference input point or a new reference output point, respectively, in a supervised way. To proceed with the OnMLM formulation let us redefine sets $\mathcal{S}$ and $\mathcal{T}$ as $\mathcal{S} = \{\boldsymbol{s}_j \in \mathcal{X} : j = 1, \ldots, M_s\}$ and $\mathcal{T} = \{\boldsymbol{t}_k \in \mathcal{Y} : k = 1, \ldots, M_t\}$.

Let $\mathcal{X}_l$ and $\mathcal{Y}_l$ be the sets of the input patterns and the expected outputs of the l-th mini batch. As in the standard MLM learning algorithm, we must compute the distance input matrix $\mathbf{D}_l = [d_{ij}] = [\|\boldsymbol{x}_i - \boldsymbol{s}_j\|_2]$ and the distance output matrix $\boldsymbol{\Delta}_l = [\delta_{ik}] = [\|\boldsymbol{y}_i - \boldsymbol{t}_k\|_2]$, where $i = 1, \ldots, N_l$, $j = 1, \ldots, M_s$, and $k = 1, \ldots, M_t$. Under the assumption of a linear mapping between $\mathbf{D}_l$ and $\boldsymbol{\Delta}_l$ we have that:

$$\boldsymbol{\Delta}_l = \mathbf{D}_l \mathbf{W} + \mathbf{E}_l, \tag{5}$$

where $\mathbf{W} \in \mathcal{R}^{M_s \times M_t}$ is the weight matrix and $\mathbf{E}_l$ is the l-th residual matrix.

Our objective is to find the optimal $\mathbf{W}$ that minimizes the mean squared sum of residuals for each $\mathbf{D}_l$ and $\boldsymbol{\Delta}_l$, so that

$$\mathcal{J}(\mathbf{W}) = \frac{1}{2N_l} \sum_{i=1}^{N_l} \sum_{k=1}^{M_t} (\delta_{ik} - \boldsymbol{d}_i^T \boldsymbol{w}_{:k})^2, \tag{6}$$

where $\boldsymbol{w}_{:k}$ is the k-th column of $\mathbf{W}$ and $\boldsymbol{d}_i^T$ is the i-th row of $\mathbf{D}_l$.

Considering the SGD algorithm, the matrix $\mathbf{W}$ can be updated by

$$\boldsymbol{w}_{:k}^{new} = \boldsymbol{w}_{:k} - \eta \frac{\partial \mathcal{J}(\mathbf{W})}{\partial \boldsymbol{w}_{:k}}, \ \forall k = 1, \ldots, M_t, \tag{7}$$

where η is the learning rate parameter.

The formulation presented above composes the weight updating rule of OnMLM, however it does not explains how the reference points are included during the training step. We must emphasize that OnMLM initiates the training process with only one input and one output reference points $(\boldsymbol{s}_1, \boldsymbol{t}_1)$ that were randomly selected from $\mathcal{X}$ and $\mathcal{Y}$, respectively. After that, at several iterations, one point is evaluated and included in the set reference points if the error reduction is higher than a pre-defined threshold. The criterion for adding new reference points is based on the hyperparameters $\mathit{\Gamma} > 0$, $\gamma \geq 0$, and $0 \leq \beta \leq 1$. The value $\mathit{\Gamma}$ determines the iterations in which the OnMLM will verify if a new set of reference points can be added, while γ is the error threshold in the new reference point criterion. The β parameter is used to update the value of γ, in order to adjust the error threshold along with the learning process of the OnMLM. Algorithm 2 summarizes how new reference points are included, being triggered after the weight matrix updating.

It is important to highlight that the weight matrix $\mathbf{W}$ increases as the number of input and output reference points increases – remember that $\mathbf{W} \in \mathcal{R}^{M_s \times M_t}$, which implies that for each new input reference point the number of lines in $\mathbf{W}$ increases by one. A similar consequence is observed for each new output reference points, however, in this last case, the number of columns in $\mathbf{W}$ is increased. Furthermore, from Algorithm 2, we can see that the γ value is not constant (step 4), but rather updated each time the new reference point criterion algorithm is executed. In summary, the criterion enables the model to decide if a new input pattern or expected output will become a reference point looking at the related error of each input pattern at the current l-th mini-batch.

Algorithm 2 New reference point criterion.

```
Input: The hyperparameters $\mathit{\Gamma}$ and $\beta$, the initial value for $\gamma$, the current iteration $l$,
    the updated weight matrix $\mathbf{W}$, the distance matrices $\mathbf{D}_l$ and $\mathbf{\Delta}_l$, and the sets $\mathcal{X}_l$
    and $\mathcal{Y}_l$.
Output: The new sets of input and output reference points $\mathcal{S}^{new}$ and $\mathcal{T}^{new}$.
 1: Verify if $l \pmod{\mathit{\Gamma}} = 0$. If yes, then go to step 2; otherwise, stop the algorithm.
 2: Compute the expected output distance matrix: $\hat{\mathbf{\Delta}}_l = \mathbf{D}_l\mathbf{W}$.
 3: Compute the mean squared error obtained by each input vector $\boldsymbol{d}_i$:
    $\boldsymbol{e}^2 = \frac{1}{M_t} diag[(\mathbf{\Delta}_l - \hat{\mathbf{\Delta}}_l)(\mathbf{\Delta}_l - \hat{\mathbf{\Delta}}_l)^T]$.
 4: Update $\gamma$:
    $\gamma = \beta\gamma + (1-\beta)\overline{\gamma}, \ \overline{\gamma} = \max\{e_i^2 : i = 1 \dots, N_l\}$.
 5: Update the sets $\mathcal{S}$ and $\mathcal{T}$:
    $\mathcal{S}^{new} = \mathcal{S} \cup \{\boldsymbol{x}_i \in \mathcal{X}_l : e_i^2 > \gamma, i = 1, \dots, N_l\}$,
    $\mathcal{T}^{new} = \mathcal{T} \cup \{\boldsymbol{y}_i \in \mathcal{Y}_l : e_i^2 > \gamma, i = 1, \dots, N_l\}$.
```

As stated before, the OnMLM model includes 3 new parameters ($\mathit{\Gamma}$, γ and β) and the task of adjusting such parameters may seem challenging . However, as we will argue, the adjustment intuitive. First, consider $\mathit{\Gamma} > 0$. Such parameter controls the frequency in which the model executes Algorithm 2, checking for new input and output reference points in the current l-th mini-batch. To choose a coherent value for $\mathit{\Gamma}$, one must consider the following: (i) each line and column of $\mathbf{W}$ is randomly initialized with positive and negative values; and (ii) the OnMLM training is based on SGD. From (i) and (ii), one can realize that, for each new input or output reference point, a line or a column with random values will be included in $\mathbf{W}$, and the model will require a number of iterations large enough to reach stability before adding new reference points, since the model is trained via SGD. This implies that $\mathit{\Gamma}$ must not be a small value, but large enough to reach some stability in the current matrix $\mathbf{W}$. Also, a large value for $\mathit{\Gamma}$ can lead the training process of the model to be slow. Taking this in consideration, and after some experiments with Adam algorithm [5], we verified that $\mathit{\Gamma} = \lceil \frac{N}{N_l} \rceil \times 50$ is a good value – note that N is the number of training samples and N_l the number of training samples in the l-th mini-batch.

Let us now focus on the $\gamma \geq 0$ parameter, which delimits if a input/output pattern will become a reference point. As shown in Algorithm 2, the parameter γ is updated considering the maximum quadratic error $\overline{\gamma}$ among the errors in the l-th mini-batch. For regression problems we recommend to use as initial value $\gamma = 0$ because γ will be adjusted during the training process. For instance, assume $\beta > 0$, then the γ provides the lower bound for adding new reference points. If $\overline{\gamma} > \gamma$, then the new value of γ increases, but under the convex combination between $\overline{\gamma}$ and γ, it remains lower than $\overline{\gamma}$, so that at least the pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ related to the maximum quadratic error $\overline{\gamma}$ of the current mini-batch will become a reference point. On the other hand, if $\overline{\gamma} < \gamma$, then the new value of γ decreases, but remains greater than $\overline{\gamma}$ and no reference point will be added. This implies that as the error is increasing, new reference points are being added; unlike, when the error is decreasing, no reference points are being added. Also, we must

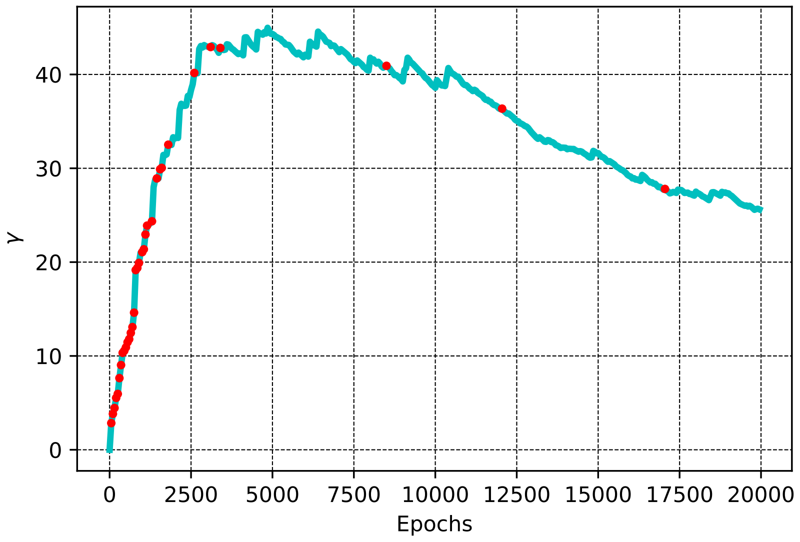

Fig. 2: γ value fluctuation along 20000 epochs with $\beta = 0.99$ applied to the BHG dataset.

emphasize that at some point of the training process, the model can experience a continuous error reduction, which can lead γ to be very small. As a consequence of a small γ, a new maximum quadratic error $\overline{\gamma} > \gamma$ can arise, adding new reference points. This dynamic adds new reference points in a supervised way, so that, when the model is experiencing excessive error values (instability) during the training process, the number of reference points tends to increase. On the other hand, when the model has the necessary information provided by the current reference points and performed a large enough number of iterations to stabilize $\mathbf{W}$, the training error tends to stabilizes, making it difficult to include new reference points. The same dynamic applies for classification scenario, however, in such a scenarios, we recommend the initial value to be $\gamma = 1$ to speed up the training process (this only applies for 1-hot coded expected outputs).

In Fig. 2 we show the γ value fluctuation for 20000 epochs and $\beta = 0.99$ applied to the Boston Housing (BHG) dataset from UCI Machine Learning [9] – some information about such a dataset can be seen in Table 1. The model was trained via Adam algorithm. In this Figure, one can see the blue line, which corresponds to the values of γ through the epochs and the red spots, showing the time at which at least one reference point was added. Taking a closer look on this Figure, one can see that at the beginning of the training process, where the model is experiencing large errors, a bigger number of reference points is added (epochs 0-5000). However, after the epoch 5000, the algorithm stabilizes and a

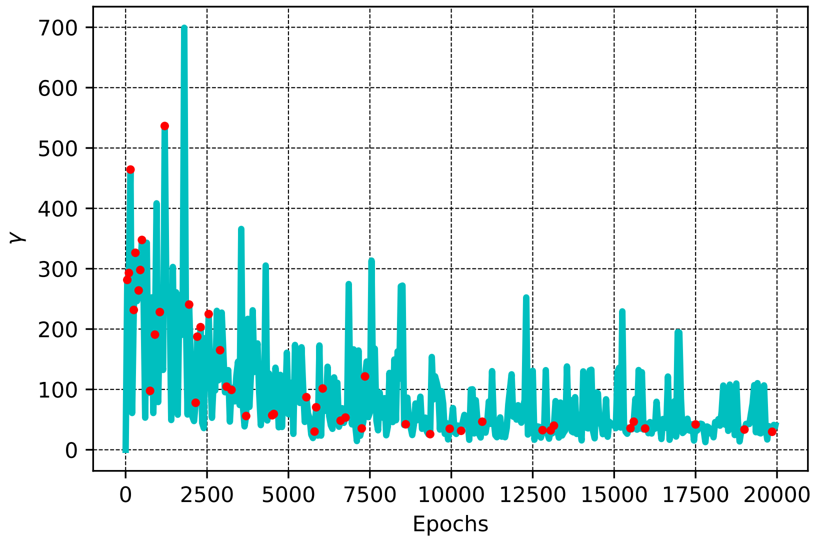

Fig. 3: γ value fluctuation through $2e+4$ epochs with $\beta = 0.01$ applied to BHG dataset.

reduced number of reference points are added. It is also noticeable the occurrence of some peaks in γ between the epochs 5000-12500 where no reference point is being added. This behavior occurs because this peaks of γ are being provided by patterns that are already reference points, and since the operation for adding new reference points is based in the union operation, a pattern that is already a reference point which provides a maximum quadratic error $\overline{\gamma} > \gamma$ will not be added as a reference point.

At last, we can make some considerations about the $\beta \in [0, 1]$ parameter. This parameter is used in the convex combination between $\overline{\gamma}$ and γ. For $\gamma \to 0$, the value of gamma will tend to reach the current value of $\overline{\gamma}$. In such a scenario, fewer reference points will be added, which can lead to poor generalization, and the value of γ will suffer some deep fluctuations. This scenario is depicted in Fig. 3. The only difference concerning that one from Fig. 2 is that $\beta = 0.01$, and the number of input and output reference points reached is 46 and 35, respectively. From figures 2 and 3, one can see that a value for $\beta \to 1$ provides a more stable learning process .

Despite the three new OnMLM parameters, a main difference between OnMLM and MLM relies on the fact that OnMLM do not use duplicated output reference points. For instance, to make it clear, let us consider a binary classification problem with $M > 2$. In this scenario, the MLM will use M reference points, so that there will be $M > 2$ output reference points and some of than will be duplicated. On the other hand, the union operation used by OnMLM in the Algorithm 2 avoid such a duplication, providing much sparse solutions at the output reference point space.

3 Experiments

To provide an overview of OnMLM capabilities, we have performed experiments considering both classification and regression tasks. We compared the OnMLM with the standard MLM and the Multilayer Perceptron (MLP). For each task we have used 3 datasets obtained from UCI Machine Learning repository [9], totalizing 6 datasets: Abalone with 3 classes (ABA3C), Adult (ADU), Bank Marketing (BKM), Boston Housing (BHG), Abalone (ABA), and Wine Quality (WIN). Tab. 1 summarizes the information datasets information: The column *task* identifies the task proposed by each dataset, where we have C for classification and R for regression. The columns p and q specify the dimension of the input and output vectors, respectively – notice that for classification tasks, q represents the number of classes. The column *test size* says the proportion used to evaluate the model in each scenario.

Table 1: Summary of datasets informations.

Dataset	*task*	*#samples*	p	q	*test size* (%)
ABA3C	C	4177	8	3	33%
ADU	C	45222	14	2	33%
BKM	C	45211	16	2	33%
BHG	R	506	13	1	20%
ABA	R	4177	8	1	33%
WIN	R	6497	11	1	33%

The training of both OnMLM and MLP (with one hidden layer) was performed using the Adam algorithm with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$ [5]. Concerning the learning rate, we have used $\eta = 1e-3$ for MLP, while for OnMLM we have set $\eta = 1e-2$ in regression tasks and $\eta = 1e-3$ in classification tasks. We have performed a 5-fold cross validation to optimize the number of reference points of the standard MLM and the size of the MLP hidden layer. For both OnMLM and MLP, we performed the simulations with mini batches of sizes $N_l = 256$ (ABA3C, ABA, and WIN), $N_l = 64$ (BHG), and $N_l = 512$ (BKS, ADU, and BKM). The maximum number of epochs was 2000. The parameters for OnMLM were fixed: we set $\Gamma = \lceil \frac{N}{N_l} \rceil \times 50$ for all datasets, $\gamma = 1.0$ for classification and $\gamma = 0$ for regression. We used $\beta = 0.95$ for all datasets.

The results for classification and regression can be viewed in Tab. 2. We must emphasize that, for ADU and BKM datasets, we have trained MLM with 6000 training samples in order to ensure computational efficiency. The first comment we should make about OnMLM is related to its superiority concerning sparsity degree compared to standard MLM. For all applications, the OnMLM achieves fewer number of input and output reference points, with special attention to BKM and WIN datasets – even when OnMLM performance was inferior to MLM (WIN dataset), the standard MLM required almost the entire training dataset to be used as reference points.

Table 2: Classification accuracy and regression MSE, number of input/output reference points, and hidden layer size (# Neurons) averaged over 10 runs.

Dataset	*Model*	*Accuracy* (σ)	# *Neurons*	M_s	M_t
ABA3C	MLP	**67.5** (0.9)	40.1		
	MLM	65.6 (1.6)	–	238.0	238.0
	OnMLM	65.6 (1.0)		**153.3**	**3.0**
ADU	MLP	80.2 (0.4)	78.1		
	MLM	78.3 (0.5)		540.0	540.0
	OnMLM	**82.5** (0.6)		**317.3**	**2.0**
BKM	MLP	**89.0** (0.1)	119.1		
	MLM	88.3 (0.5)		900.0	900.0
	OnMLM	88.9 (0.3)		**150.7**	**2**
		MSE (σ)			
BHG	MLP	$1.4e+1$ (3.7)	40.1	-	-
	MLM	$\mathbf{1.2e+1}$ (4.4)	-	396.0	396.0
	OnMLM	$1.5e+1$ (4.6)	-	**110.4**	**84.3**
ABA	MLP	4.6 ($1.7e-1$)	5.0	-	-
	MLM	**4.5** ($2.2e-1$)	-	210.0	210.0
	OnMLM	4.6 ($2.4e-1$)	-	**172.6**	**22.6**
WIN	MLP	$5.5e-1$ ($1.1e-2$)	29.3	-	-
	MLM	$\mathbf{3.9e-1}$ ($2.4e-2$)		3764.7	3764.7
	OnMLM	$5.0e-1$ ($2.3e-2$)	-	**352.7**	**7.0**

Concerning the accuracy of the methods, we can say that OnMLM in general outperforms MLM, with an exception for ABA3C dataset. Also, although the OnMLM reached lower accuracy than MLP in the ABA3C dataset and equivalent accuracy in the BKM dataset, the OnMLM performed well when compared to its counterparts in the ADU dataset. Finally, taking a close look into the MSE measures for the regression tasks, one can see that the MLM is always superior, although the OnMLM still obtains much sparser solutions.

4 Conclusions

We have presented the OnMLM, a feasible formulation based on MLM for performing online and incremental learning. Thus, OnMLM greatly increases the applicability of MLM, since it is able to tackle large datasets and data streams. Furthermore, the proposed OnMLM is able to reduce the number of input and output reference points when its hyperparameters are carefully chosen. The new reference points criterion is of central importance because of the incremental nature of the training procedure and the sparseness of the generated model. Those particularities turn OnMLM a promising online learning method. Future works intend to provide more detailed analysis of the OnMLM incremental behavior and investigate alternatives for enabling it to remove reference points in the training step.

Acknowledgements The work was supported by the Academy of Finland from grants 311877 and 315550.

References

1. Beygelzimer, A., Hazan, E., Kale, S., Luo, H.: Online gradient boosting. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2458–2466. Curran Associates, Inc. (2015)
2. Huang, G., Liang, N., Rong, H., Saratchandran, P., Sundararajan, N.: On-line sequential extreme learning machine. In: the IASTED International Conference on Computational Intelligence (CI 2005), Calgary, Canada, July 4-6, 2005
3. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing 70(1), 489 – 501 (2006), neural Networks
4. Jian, L., Shen, S., Li, J., Liang, X., Li, L.: Budget online learning algorithm for least squares svm. IEEE Transactions on Neural Networks and Learning Systems 28(9), 2076–2087 (Sept 2017)
5. Kingma, D.P., Ba, L.J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (ICLR, 2015)
6. Lakshminarayanan, B., Roy, D.M., Teh, Y.W.: Mondrian forests: Efficient online random forests. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. pp. 3140–3148. NIPS'14, MIT Press, Cambridge, MA, USA (2014)
7. Langford, J., Li, L., Zhang, T.: Sparse online learning via truncated gradient. J. Mach. Learn. Res. 10, 777–801 (Jun 2009)
8. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.: Efficient BackProp, pp. 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
9. Lichman, M.: UCI machine learning repository (2013), `http://archive.ics.uci.edu/ml.`
10. Matias, T., Souza, F., Araújo, R., Gonçalves, N., Barreto, J.P.: On-line sequential extreme learning machine based on recursive partial least squares. Journal of Process Control 27, 15 – 21 (2015)

11. Mesquita, D.P.P., Gomes, J.P.P., Souza Junior, A.H.: Ensemble of efficient minimal learning machines for classification and regression. Neural Processing Letters 46(3), 751–766 (Dec 2017), `https://doi.org/10.1007/s11063-017-9587-5`
12. Santos, J.D.A., Barreto, G.A.: A regularized estimation framework for online sparse lssvr models. Neurocomputing 238, 114 – 125 (2017)
13. de Souza Júnior, A.H., Corona, F., Barreto, G.A., Miche, Y., Lendasse, A.: Minimal learning machine: A novel supervised distance-based approach for regression and classification. Neurocomputing 164, 34 – 44 (2015)
14. Suykens, J., Vandewalle, J.: Least squares support vector machine classifiers. Neural Processing Letters 9(3), 293–300 (Jun 1999)