*International Journal of Recent Engineering Science (IJRES),*
ISSN: 2349-7157, *Volume 2 Issue 5 September to October 2015*

# Core Set Sequential Feed-Forward Neural Networks

Shuxia Lu[1], Chenxu Zhu[2], Yangfan Zhou[1]
[1](*Key Lab. of Machine Learning and Computational Intelligence, College of Mathematics and Information Science, Hebei University, China*)
[2](*College of Science, Northwest Agriculture & Forestry University, China)*

***Abstract:***
*A core set sequential feed-forward neural networks (CS-SFFN) approach is proposed in order to deal with large datasets classification problem. In the first stage, the core set can be obtained efficiently by using the generalized core vector machine (GCVM) algorithm. For the second stage, the sequential feed-forward neural networks (SFFN) can be used to implement classification. A strategy proposed within CS-SFFN is to take hidden-layer input weights as a subset of the core set (input strategy). Experiments show that the CS-SFFN has comparable performance with SV-SFFNs and EM-ELM.*

**Keywords -** *Core vector machine; Core set; Error minimized extreme learning machine; Support vector sequential feed-forward neural networks; Sequential approximations*

## I. INTRODUCTION

How to effectively deal with large-scale data is a hot issue in current research. In recent years, a variety of approaches have been proposed in large datasets problem. These methods include: the extreme learning machine (ELM) [1, 2], ELM is a single hidden layer feed forward network where the input weights and the biases are chosen randomly and the output weights are calculated analytically; a generalized perceptron with margin [3], which can deal with the large datasets problem; a general piecewise linear classifier [4], which can solve the nonlinear separable problem without kernel; the geometric algorithms to large margin classifier based on affine hulls [5]; by chunking or decomposition methods, for example, the well-known sequential minimal optimization (SMO) algorithm [6]; sampling techniques for kernel methods [7]; the core vector machine (CVM) [8, 9], Tsang et al. proposed the core vector machine (CVM) by utilizing an approximation algorithm for the minimum enclosing ball (MEB) problem in computational geometry, the CVM algorithm achieves an asymptotictime complexity that is linear in $N$ and a space complexity that is independent of $N$, where $N$ is the size of the training patterns; maximum vector-angular margin core vector machine (MAMCVM) [10], by connecting the CVM method with MAMC such that the corresponding fast training on large datasets can be effectively achieved.

Methods that construct SLFNs sequentially had been reported [11-13]. Error minimized extreme learning machines has been proposed as a simple and efficient approach to build single hidden layer feed forward networks (SLFNs) sequentially. EM-ELM is an incremental extension of the extreme learning machines. They add random hidden nodes one by one (or group by group) and update the output weights incrementally to minimize the sum-of-squares error in the training set. Support vector sequential feed-forward neural networks (SV-SFNNs) are a particular case of the sequential approximation with optimal coefficients and interacting frequencies (SAOCIF) method. Their hidden-layer weights are a subset of the data instead of being random.

In this paper, we focus on the large datasets effective classification problem, a core set sequential feed-forward neural networks (CS-SFFN) approach is proposed. It consists of two stages. The first stage is to obtain the core set of the large training dataset by using the GCVM algorithm. In the second stage, the sequential feed-forward neural networks (SFFN) can be used to implement classification. A strategy proposed within CS-SFFN is to take hidden-layer input weights as a subset of the core set (input strategy). This work shows a comparison between with random strategy and with input strategy. The goal is finding out whether there is any difference in generalization performance among CS-SFFN, SV-SFNNs and EM-ELM. An experimental study on 10 benchmark data sets for classification problems is presented. Experiments also demonstrated that the CS-SFFN has comparable performance with SV-SFNNs and EM-ELM implementations.

The rest of this paper is organized as follows. Section 2 reviews the GCVM and SV-SFNNs. Section 3 presents the CS-SFFNS approach. In Section 4, the experimental results on several datasets are reported. Some conclusions are finally given in Section 5.

## II. BACKGROUND

### *A. The Generalized Core Vector Machine (GCVM）*

In this section, we first review the generalized core vector machine (The generalized CVM, GCVM) algorithm as proposed in [9]. The GCVM algorithm is much faster and can handle much larger datasets than

existing SVM implementations. The generalized CVM algorithm can be used with any linear/nonlinear kernel and can also be applied to kernel methods such as SVR and the ranking SVM. Moreover, like the original CVM, its asymptotic time complexity is again linear in $N$ and its space complexity is independent of $N$, where $N$ is the size of the training patterns.

The GCVM utilizes an approximation algorithm for the center constrain minimum enclosing ball (CC-MEB) problem, which will be briefly introduced as follows:

Suppose the training set is denoted by $S=\{x_i \mid x_i \in \mathbb{R}^n, \quad i=1,\cdots,N\}$, the minimum enclosing ball of $S$ (denoted $MEB(S)$) is the smallest ball that contains all the points in $S$. In this paper, we denote the ball with center $\mathbf{c}$ and radius $R$ by $B(\mathbf{c},R)$. Also, the center and radius of a ball $B(\mathbf{c},R)$ are denoted by $\mathbf{c}_B$ and $r_B$, respectively. Given an $\varepsilon>0$, a ball $B(\mathbf{c},(1+\varepsilon)R)$ is an $(1+\varepsilon)$-approximation of $MEB(S)$ if $R\le r_{MEB(S)}$ and $S\subset B(\mathbf{c},(1+\varepsilon)R)$. $\varphi: x_i \to \varphi(x_i)$ denotes the feature map associated with a given kernel $k$, and $B(\mathbf{c},R)$ is the desired MEB in the kernel-induced feature space $\Gamma$.

The MEB problem finds the smallest ball containing all $\varphi(x_i)\in S$ in the feature space. In this section, we first augment an extra $\delta_i \in R$ to each $\varphi(x_i)$, forming $\begin{bmatrix}\varphi(x_i)\\ \delta_i\end{bmatrix}$. Then, we find the MEB for these augmented points, while at the same time constraining the last coordinate of the ball's center to be zero (i.e., of the form $\begin{bmatrix}\mathbf{c}\\ 0\end{bmatrix}$). The primal form of the center constrain minimum enclosing ball (CC-MEB) problem can be formulated as

$$\begin{aligned}&\min \quad R^2\\ &s.t. \quad \|\varphi(x_i)-\mathbf{c}\|^2+\delta_i^2 \le R^2, \quad i=1,\cdots,N.\end{aligned} \tag{1}$$

The corresponding dual of (1) is the following QP problem

$$\begin{aligned}&\max \quad \boldsymbol{\alpha}^T(diag(\mathbf{K})+\boldsymbol{\Delta})-\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}\\ &s.t. \quad \boldsymbol{\alpha}^T\mathbf{1}=1, \quad \boldsymbol{\alpha}\ge\mathbf{0}.\end{aligned} \tag{2}$$

where $K=[k(x_i,x_j)]=[\varphi(x_i)^T\varphi(x_j)]$ is the corresponding kernel matrix, and

$$\boldsymbol{\Delta}=[\delta_1^2,\cdots,\delta_N^2]^T\ge\mathbf{0}. \tag{3}$$

From the optimal $\boldsymbol{\alpha}$ solution of (2), we can recover $R$ and $\mathbf{c}$ as

$$R=\sqrt{\boldsymbol{\alpha}^T(diag(\mathbf{K})+\boldsymbol{\Delta})-\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}} \tag{4}$$

$$\mathbf{c}=\sum_{i=1}^{N}\alpha_i\varphi(x_i). \tag{5}$$

The squared distance between the center $\begin{bmatrix}\mathbf{c}\\ 0\end{bmatrix}$ and any point $\begin{bmatrix}\varphi(x_l)\\ \delta_l\end{bmatrix}$

$$\|\varphi(x_l)-\mathbf{c}\|^2+\delta_l^2=\|\mathbf{c}\|^2-2(\mathbf{K}\boldsymbol{\alpha})_l+k_{ll}+\delta_l^2. \tag{6}$$

which does not depend explicitly on the feature map $\varphi$.

Because of the constraint $\boldsymbol{\alpha}^T\mathbf{1}=1$ in (2), an arbitrary multiple of $\boldsymbol{\alpha}^T\mathbf{1}$ can be added to the objective without affecting its solution. In other words, for an arbitrary $\eta\in\mathbb{R}$, (2) yields the same optimal as

$$\begin{aligned}&\max \quad \boldsymbol{\alpha}^T(diag(\mathbf{K})+\boldsymbol{\Delta}-\eta\mathbf{1})-\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}\\ &s.t. \quad \boldsymbol{\alpha}^T\mathbf{1}=1, \quad \boldsymbol{\alpha}\ge\mathbf{0}.\end{aligned} \tag{7}$$

Hence, any QP problem of the form (7), with the condition (3), can also be regarded as a special MEB problem, called center constrained MEB, i.e. CC-MEB. As pointed out by Tsang et al., CC-MEB can be approximately solved with the asymptotic linear time complexity O($N$) and its space complexity independent of $N$ for large datasets by using the generalized core vector machine.

The GCVM algorithm is introduced as follows:

The GCVM algorithm is shown in Algorithm 1. Here, the core set, the ball's center, and radius at the $t$th iteration are denoted by $S_t, \mathbf{c}_t$, and $R_t$ respectively. The GCVM algorithm requires the input of a termination parameter $\varepsilon$.

**Algorithm 1. GCVM**

1) Initialize $\varepsilon$, $t=0, S_t, \mathbf{c}_t, R_t$
2) Update the core set: if there is no training pattern that falls outside the ball $B(\mathbf{c}_t,(1+\varepsilon)R_t)$ in the corresponding feature space, $S=S_t$.
3) Find $\mathbf{z}$ such that it is the farthest away from $\mathbf{c}_t$ in the corresponding feature space and set $S_{t+1}=S_t\cup\{\mathbf{z}\}$
4) Find the new MEB: $B(\mathbf{c}_{t+1},R_{t+1})$
5) Set $t=t+1$, and go to step 2.

### *B. Support Vector Sequential Feed-forward Neural Networks (SV- SFNNs)*

In order to avoid the need of setting in advance the number of hidden units and to reduce the training computational time, a fast sequential algorithm called SV-SFNNs has been reported. The candidates are generated using the input strategy; the resulting method selects the best of the input examples as the hidden-layer weights of the new hidden unit.

The *input strategy*: the one in which the candidates are only selected among the input examples in the training set; more precisely $\omega=\mathbf{x}_j$, for some $j$ not

already used, and $b$ is a constant depending on the activation function.

**Algorithm 2. SV-SFNNs**

Let $\{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, ..., N\}$ be a set of training examples, the maximum number of hidden units $L_{\max}$ , a strategy to generate candidates, the maximum number of candidates for any hidden unit $C_{\max}$ , and the expected learning accuracy $\varepsilon$ .

1) *Initialization phase*:
Let $L$:=0, $H_0 = [\,]$ .
2) *Recursively growing phase*:
Repeat
let $L$:=$L$+1, c=0
while $c < C_{\max}$ do
Generate a candidate $(\omega, b)$ for the $L^{th}$ hidden unit with the given strategy, and store in a temporary matrix $\mathbf{H}$ the corresponding hidden-layer output matrix

$$\mathbf{H} = [\mathbf{H}_{L-1}, \delta\mathbf{H}] \tag{8}$$

where $\delta\mathbf{H}$ contains the new column computed.
If $\mathbf{H}^T\mathbf{H}$ is well conditioned then
Let $c$: =$c$+1
Find the optimal output-layer weights

$$\boldsymbol{\lambda} = \mathbf{H}^{\dagger}\mathbf{T}. \tag{9}$$

Calculate the corresponding output error

$$E(\mathbf{H}) = \tfrac{1}{2}\left\|\mathbf{H}\boldsymbol{\lambda} - \mathbf{T}\right\|^2. \tag{10}$$

If $E(\mathbf{H})$ is minimum in the current loop then
Let $(\omega_L, b_L) = (\omega, b); \boldsymbol{\lambda}_L = \boldsymbol{\lambda}; \mathbf{H}_L = \mathbf{H}$ .
End if
End if
End while
Until $L = L_{\max}$ or $E(\mathbf{H}_L) \le \varepsilon$ .

## III. CORE SET SEQUENTIAL FEED-FORWARD NEURAL NETWORKS (CS-SFFN)

We can now give a fast training algorithm for large datasets which is called a core set sequential feed-forward neural networks (CS-SFFN). It consists of two stages. The first stage is to obtain the core set of the large training dataset by using GCVM. In the second stage, the sequential feed-forward neural networks (SFFN) can be used to implement classification.

The *input strategy*: A strategy proposed within CS-SFFN is to take hidden-layer input weights as a subset of the core set (input strategy). CS-SFFN can be summarized as follows:

**Algorithm 3. CS-SFFN**

Let $\{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, ..., N\}$ be a set of training examples, the maximum number of hidden units $L$, select hidden-layer weight from the *core set*, the maximum number of candidates for any hidden unit $C_{\max}$, and the expected learning accuracy $\varepsilon$ .

*Stage 1*: Using GCVM to obtain the core set $S_t$ .

1) Initialize $\varepsilon$ , $t = 0, S_t, \mathbf{c}_t, R_t$
2) Update the core set: if there is no training pattern that falls outside the ball $B(\mathbf{c}_t, (1+\varepsilon)R_t)$ in the corresponding feature space, go to stage 2.
3) Find $\mathbf{z}$ such that it is the farthest away from $\mathbf{c}_t$ in the corresponding feature space and set $S_{t+1} = S_t \bigcup \{\mathbf{z}\}$
4) Find the new MEB: $B(\mathbf{c}_{t+1}, R_{t+1})$
5) Set $t = t+1$, and go to step 2.

*Stage 2*: Using SFFN to implement classification.

1) *Initialization phase*:
Let $L$:=0, $H_0 = [\,]$ .
2) *Recursively growing phase*:
Repeat
let $L$:=$L$+1, c=0
while $c < C_{\max}$ do
Generate a candidate $(\omega, b)$ for the $L^{th}$ hidden unit by selecting hidden-layer weight from the *core set*, and store in a temporary matrix $\mathbf{H}$ the corresponding hidden-layer output matrix

$$\mathbf{H} = [\mathbf{H}_{L-1}, \delta\mathbf{H}]$$

where $\delta\mathbf{H}$ contains the new column computed.
If $\mathbf{H}^T\mathbf{H}$ is well conditioned then
Let $c$: =$c$+1
Find the optimal output-layer weights

$$\boldsymbol{\lambda} = \mathbf{H}^{\dagger}\mathbf{T}.$$

Calculate the corresponding output error

$$E(\mathbf{H}) = \tfrac{1}{2}\left\|\mathbf{H}\boldsymbol{\lambda} - \mathbf{T}\right\|^2.$$

If $E(\mathbf{H})$ is minimum in the current loop then
Let $(\omega_L, b_L) = (\omega, b); \boldsymbol{\lambda}_L = \boldsymbol{\lambda}; \mathbf{H}_L = \mathbf{H}$ .
End if
End if
End while
Until $L = L_{\max}$ or $E(\mathbf{H}_L) \le \varepsilon$ .

## IV. EXPERIMENTAL RESULTS

In this section, we conduct the performance comparison of the three methods for ten real problems: Spambase, SkinNonSkin, Shuttle, Sat, MiniBooNE, Digit, Artifical, Letter, Page, and Contra. Most of the datasets are taken from the UCI machine learning repository [14]. All the simulations are carried out in MATLAB7.1 environment running in Intel Core(TM) i5-2400, 3.10GHz, 8GBRAM. The numbers of attributes, class, samples for training and testing are shown in Table 1.

In all experiments, the naive QP solver is adopted to solve the QP problem and the Gaussian function is taken as the kernel function where $h$ is the kernel parameter of the Gaussian kernel. The width

parameter *h* is selected to the mean squared norm of the training data. Setting an appropriate the approximation parameter $\varepsilon$ is important in CS-SFFN. The smaller $\varepsilon$ will result in more core vectors and the classification speed becomes slower. In our experiment, the *sigmoid* and *RBF* functions were used as an activation function for all models.

**Table 1. The experimental data sets**

| Data set | Attributes | Training | Testing | Class |
|---|---|---|---|---|
| Spambase | 57 | 2300 | 2300 | 2 |
| Skin_NonSkin | 3 | 147034 | 147034 | 2 |
| Shuttle | 8 | 29000 | 29000 | 7 |
| Sat | 36 | 3217 | 3217 | 7 |
| MiniBooNE | 50 | 78038 | 78038 | 2 |
| Digit, | 64 | 2810 | 2810 | 9 |
| Artifical | 10 | 166666 | 166666 | 2 |
| Letter | 16 | 10000 | 10000 | 26 |
| Page | 10 | 2736 | 2736 | 5 |
| Contra | 9 | 736 | 736 | 3 |

**Table 2. The average test accuracy of the data sets with RBF activation function**

| Data set | EM-ELM | SV-SFFNs | CS-SFFN |
|---|---|---|---|
| Spambase | 0.8865 | 0.9011 | 0.9011 |
| SkinNonSkin | 0.9333 | 0.9231 | 0.8920 |
| Shuttle | 0.9943 | 0.9867 | 0.9971 |
| Sat | 0.8776 | 0.8612 | 0.8812 |
| MiniBooNE | 0.8533 | 0.8623 | 0.8628 |
| Digit, | 0.7913 | 0.7766 | 0.8011 |
| Artifical | 0.8278 | 0.8522 | 0.8612 |
| Letter | 0.8166 | 0.8322 | 0.8326 |
| Page | 0.9551 | 0.9621 | 0.9713 |
| Contra | 0.8865 | 0.8901 | 0.9025 |

Ten trials were conducted for the three algorithms and the average results are shown in Tables 2 and 3. Tables 2 and 3 respectively show the performance comparison of testing accuracy of the three methods with *sigmoid* and *RBF* activation function. As observed from the Tables 2 and 3, general speaking, testing accuracy of CS-SFFN is fully comparable to EM-ELM and SV-SFFNS or even better than them.

**Table 3. The average test accuracy of the data sets with sigmoid activation function**

| Data set | EM-ELM | SV-SFFNs | CS-SFFN |
|---|---|---|---|
| Spambase | 0.9100 | 0.9231 | 0.9563 |
| SkinNonSkin | 0.9971 | 0.9983 | 0.9881 |
| Shuttle | 0.9776 | 0.9673 | 0.9863 |
| Sat | 0.8633 | 0.8735 | 0.8672 |
| MiniBooNE | 0.8338 | 0.7917 | 0.8816 |
| Digit, | 0.9459 | 0.9671 | 0.9629 |
| Artifical | 0.7186 | 0.7512 | 0.8376 |
| Letter | 0.7243 | 0.7513 | 0.7965 |
| Page | 0.9576 | 0.9777 | 0.9819 |
| Contra | 0.7123 | 0.7356 | 0.7562 |

**Table 4. The average hidden nodes numbers of the data sets with RBF activation function**

| Data set | EM-ELM | SV-SFFNs | CS-SFFN |
|---|---|---|---|
| Spambase | 70.12 | 80.11 | 86.21 |
| SkinNonSkin | 69.76 | 73.66 | 79.65 |
| Shuttle | 23.98 | 50.97 | 70.01 |
| Sat | 12.88 | 31.76 | 30.87 |
| MiniBooNE | 60.97 | 70.00 | 95.22 |
| Digit, | 55.90 | 67.08 | 79.19 |
| Artifical | 33.99 | 20.16 | 50.10 |
| Letter | 90.99 | 85.14 | 68.21 |
| Page | 49.08 | 33.77 | 77.05 |
| Contra | 77.77 | 67.15 | 89.02 |

Table 4 and Table 5 respectively show the average number of hidden nodes of data sets with the sigmoid and RBF activation functions for the three methods (EM-ELM, SV-SFFNs and CS-SFFN). From Table 4 and Table 5, although no clear trend is observed about the number of hidden units selected by both strategies, the input strategy seems to need more units than the random strategy. The low number of hidden units is a clear indication of a strong tendency to overfitting in this data set.

**Table 5. The average hidden nodes numbers of the data sets with sigmoid activation function**

| Data set | EM-ELM | SV-SFFNs | CS-SFFN |
|---|---|---|---|
| Spambase | 44.12 | 51.11 | 60.11 |
| SkinNonSkin | 33.13 | 60.02 | 87.36 |
| Shuttle | 10.86 | 13.09 | 20.11 |
| Sat | 60.06 | 44.14 | 61.98 |
| MiniBooNE | 88.03 | 80.17 | 66.91 |
| Digit, | 31.90 | 20.99 | 88.90 |
| Artifical | 87.69 | 60.66 | 50.12 |
| Letter | 66.60 | 75.13 | 90.22 |
| Page | 21.09 | 10.00 | 18.19 |
| Contra | 31.19 | 56.00 | 70.15 |

## V. CONCLUSION

The GCVM utilizes an approximation algorithm for the center constrain minimum enclosing ball (CC-MEB) problem. We proposed the core set sequential feed-forward neural networks (CS-SFFN) approach. It consists of two stages. In the first stage, the core set can be obtained efficiently by using the GCVM algorithm. The GCVM algorithm asymptotic time complexity is again linear in *N* and its space complexity is independent of *N*, where *N* is the size of the training patterns. Thus, we can obtain the core set quickly. In the second stage, the sequential feed-forward neural networks (SFFN) can be used to implement

classification. A strategy proposed within CS-SFFN is to take hidden-layer input weights as a subset of the core set (input strategy). An experimental study on 10 benchmark data sets for classification problems is presented. Experiments also demonstrated that the CS-SFFN has comparable performance with SV-SFNNs and EM-ELM implementations. Another conclusion of the experimental study is that, the input strategy should perform better than the random one, since the input strategy samples the weights from the underlying distribution of the data.

## ACKNOWLEDGEMENTS

This research is supported by the National Natural Science Foundation of China (61170040), by the Natural Science Foundation of Hebei Province (F2015201185, F2013201220).

## REFERENCES

[1] G. B. Huang, Q. Y. Zhu, C. K. Siew, Extreme learning machine: theory and applications, Neurocomputing, 70, 2006, 489-501.

[2] R. Zhang, Y. Lan, G. B. Huang, Z. B. Xu, Universal approximation of extreme learning machine with adaptive growth of hidden node, IEEE Transactions on Neural Networks and Learning Systems, 23(2), 2012, 365-371.

[3] C. Panagiotakopoulos, P. Tsampouka, The Margitron: A Generalized Perceptron with Margin, IEEE Trans. Neural Netw, 22 (3), 2011, 395–407.

[4] L. Yujian, L. Bo, Y. Xinwu, F. Yaozong, L. Houjun, Multiconlitron: a general piecewise linear classifier, IEEE Trans. Neural Netw. 22 (2), 2011, 276–289.

[5] X. Peng, Y. Wang, Geometric algorithms to large margin classifier based on affine hulls, IEEE Trans. Neural Netw. 23 (2), 2012, 236–246.

[6] J. Platt, B. Schölkopf, C. Burges, and A. Smola, Eds., Fast training of support vector machines using sequential minimal optimization. In Advances in Kernel Methods – Support Vector Learning. Cambridge, MA: MIT Press, 1999, 185–208.

[7] D. Achlioptas, F. McSherry, and B. SchÖlkopf, Sampling techniques for kernel methods, Advances in neural information processing systems, 14, 2002, 335-342.

[8] I. W. Tsang, J. T. Kwok, and P. M. Cheung, Core vector machines: fast SVM training on very large data sets, Journal of Machine Learning Research, 6, 2005, 363-392.

[9] I. W. Tsang, J. T. Kwok, and J. M. Zurada, Generalized core vector machines, IEEE Transactions on Neural Networks, 17(5), 2006, 1126-1140.

[10] W. J. Hu, F. L. Chung, S. T. Wang, The Maximum Vector Angular Margin Classifier and its fast training on large datasets using a core vector machine, Neural Networks, 27, 2012, 60-73.

[11] G. Feng, G. B. Huang, Q. Lin, & R Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Transactions on Neural Networks, 20, (2009)1352–1357.

[12] E. Romero, & D Toppo, Comparing support vector machines and feedforward neural networks with similar hidden-layer weights, IEEE Transactions on Neural Networks, 18, 2007, 959-963.

[13] L Guo, J H Hao, M Liu. An incremental extreme learning machine for online sequential learning problems, Neurocomputing, 128, 2014, 50-58.

[14] A. Frank, A. Asuncion, UCI machine learning repository, 2010. URL http: //archive.ics.uci.edu/ml.