An approach to improve online sequential extreme learning machines using restricted Boltzmann machines

Andre G. C. Pacheco
agcpacheco@inf.ufes.br
Renato A. Krohling
rkrohling@inf.ufes.br

July 10, 2018

Outline

Introduction

- The extreme learning machine (ELM) is a straightforward approach to handle single-hidden layer feedforward neural network (SLFN)
- Although it is a very fast approach and may provide a good generalization, two shortcomings can be pointed:
  - It does not allow sequential learning
  - It assigns the input weights randomly

Introduction

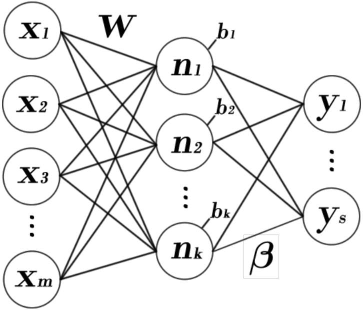
- To tackle the sequential learning issue, the online sequential ELM (OS-ELM) was proposed
  - It is able to learn from a block of data with fixed or varying size
- Different approaches have been proposed to improve the OS-ELM
- However, none of them handle the input weights

Introduction

- Recently, we proposed an approach to determine the ELM input weights using the Restricted Boltzmann Machine (RBM)
  - This approach is called RBM-ELM
  - It achieves good results for different datasets
  - Nonetheless, it does not allow sequential learning
- In this work, we extend the RBM-ELM by combining it with the OS-ELM to create the RBM-OS-ELM
  - It is faster than the RBM-ELM
  - For most datasets, it achieves a better performance than the OS-ELM.

Extreme Learning Machine (ELM)

- The ELM was developed specifically to handle SLFN architecture



Extreme Learning Machine (ELM)

- All network values are model as matrices:

$$\mathbf{x} = [x_1, \cdots, x_m, 1] \quad \mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mk} \\ b_1 & \cdots & b_k \end{bmatrix}$$
$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1s} \\ \vdots & \ddots & \vdots \\ \beta_{k1} & \cdots & \beta_{ks} \end{bmatrix} \quad \mathbf{y} = [y_1, \cdots, y_s] \tag{1}$$

- From $\mathbf{W}$ we compute the feature map $\mathbf{H}$

$$\mathbf{h}^i = [x_1^i, \cdots, x_m^i, 1] \times \begin{bmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mk} \\ b_1 & \cdots & b_k \end{bmatrix} \Rightarrow \mathbf{H} = \begin{bmatrix} f(\mathbf{h}^1) \\ f(\mathbf{h}^2) \\ \vdots \\ f(\mathbf{h}^N) \end{bmatrix}_{N \times k} \tag{2}$$

Extreme Learning Machine (ELM)

- The weight matrix β is obtained by solving the linear system:

$$\mathbf{H}\beta = \mathbf{Y} \rightarrow \beta = \mathbf{H}^{\dagger}\mathbf{Y} \tag{3}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$

Online sequential ELM (OS-ELM)

- The OS-ELM is able to process blocks/batches of data when they become available

- The algorithm has two phases:
  1. Initialization phase
  2. Sequential phase

Online sequential ELM (OS-ELM)

- **Initialization phase:** given a small block of the training data $(\mathbf{X}_0, \mathbf{Y}_0)$
  1. Assign the input weights $\mathbf{W}_0$ randomly and do not change it
  2. Compute $\mathbf{H}_0$ according the Eq. 3 and $\mathbf{X}_0$ and $\mathbf{W}_0$
  3. Compute $\boldsymbol{\beta}_0$ as follows:

$$\begin{gathered}\boldsymbol{\beta}_0 = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{Y}_0, \text{ where} \\ \mathbf{P}_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1}\end{gathered} \tag{4}$$

Online sequential ELM (OS-ELM)

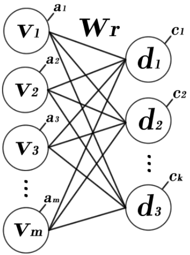
- **Sequential phase:** given an arrived block of data $(\mathbf{X}_j, \mathbf{Y}_j)$
  1. Compute $\mathbf{H}_j$ according the Eq. 3 and $\mathbf{X}_j$ and $\mathbf{W}_0$
  2. Compute $\boldsymbol{\beta}_j$ as follows:

$$\begin{aligned} \boldsymbol{\beta}_j &= \boldsymbol{\beta}_{j-1} + \mathbf{P}_j \mathbf{H}_j^T (\mathbf{X}_j - \mathbf{H}_j \boldsymbol{\beta}_{j-1}), \text{ where} \\ \mathbf{P}_j &= \mathbf{P}_{j-1} - \mathbf{P}_{j-1} \mathbf{H}_j^T (\mathbf{I} + \mathbf{H}_j \mathbf{P}_{j-1} \mathbf{H}_j^T)^{-1} \mathbf{H}_j \mathbf{P}_{j-1} \end{aligned} \tag{5}$$

- Every time a new block arrives, this phase is performed to adjust β

Restricted Boltzmann Machine (RBM)

- The RBM is an energy-based system that
  - It aims to learn the probability distribution
  - Unsupervised learning
  - Visible ($\mathbf{v}$) and hidden ($\mathbf{d}$) layers
  - Conection weights ($\mathbf{W_r}$) and the bias ($\mathbf{a}$ and $\mathbf{b}$)



Restricted Boltzmann Machine (RBM)

- The $(\mathbf{v}, \mathbf{d})$ configuration has an associated energy value defined by:

$$E(\mathbf{v}, \mathbf{d}; \boldsymbol{\theta}) = -\sum_{i=1}^{m} \frac{(v_i - a_i)^2}{2\sigma^2} - \sum_{j=1}^{k} c_j d_j - \sum_{i,j=1}^{m,k} \frac{v_i}{\sigma^2} d_j w_{ij} \quad (6)$$

where $\boldsymbol{\theta} = (\mathbf{W}_r, \mathbf{a}, \mathbf{c})$

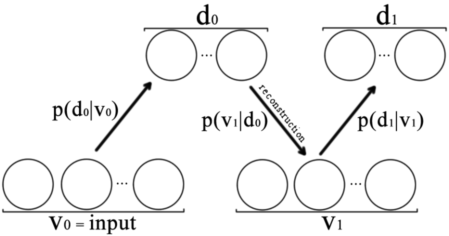
- From the energy, one computes the conditional probabilities:

$$p(d_j = 1 | \mathbf{v}; \boldsymbol{\theta}) = \phi(c_j + \sum_{i=1}^{m} v_i w_{ij}), \quad \text{where } \phi(x) = \frac{1}{1+e^{-x}} \quad (7)$$

$$p(v_i = v | \mathbf{d}; \boldsymbol{\theta}) = G(v | a_i + \sum_{j=1}^{k} d_j w_{ij}, \sigma^2), \qquad \text{where } G \text{ is the normal distribution} \quad (8)$$

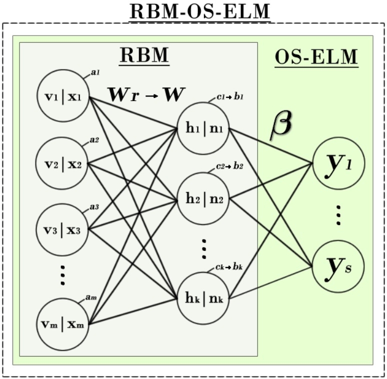
Restricted Boltzmann Machine (RBM)

- The contrastive divergence algorithm:
  - Usupervised algorithm
  - It uses k steps of Gibbs sampling algorithm
  - The Gibbs sampling is initialized with the training data



Restricted Boltzmann machine OS-ELM (RBM-OS-ELM)

- The algorithm's main idea:
  - Updating $\mathbf{W}_0$ for every training data block using the RBM
  - In brief, $\mathbf{W}_\mathbf{r} \rightarrow \mathbf{W}_0$ and $\mathbf{c} \rightarrow \mathbf{b}$



Experimental results

- We used two type of datasets:

Common dataset	Samples	Features	Labels	Permutation
Credit Australia	690	14	2	Yes
Diabetic	1151	19	2	Yes
DNA	3186	180	3	No
Isolet	7797	617	26	No
Madelon	2600	500	2	Yes
MNIST	70000	784	10	No
Spam	4601	57	2	Yes
Urban land cover	675	147	9	Yes
Large dataset	**Samples**	**Features**	**Labels**	**Permutation**
Covertype	581012	54	7	Yes
Higgs	11×10^6	28	2	Yes
Susy	5×10^6	18	2	Yes

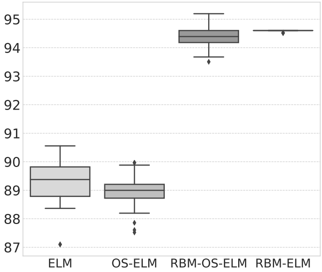
Experimental results

- The algorithms' performance for common datasets:

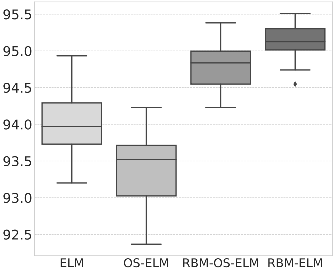
Database	ELM		OS-ELM		RBM-OS-ELM		RBM-ELM	
	Accuracy (%)	*Time (sec)*	*Accuracy (%)*	*Time (sec)*	*Accuracy (%)*	*Time (sec)*	*Accuracy (%)*	*Time (sec)*
Credit Australia	85.732 ± 2.292	0.004	86.731 ± 1.979	0.006	86.199 ± 1.877	0.010	86.070 ± 1.960	0.380
Diabetic	74.415 ± 2.562	0.010	73.478 ± 2.330	0.013	74.241 ± 2.529	0.075	75.323 ± 1.996	0.442
DNA	89.232 ± 0.827	0.146	88.943 ± 0.622	0.104	$\mathbf{94.356 \pm 0.353}$	1.397	$\mathbf{94.592 \pm 0.028}$	3.279
Isolet	94.032 ± 0.385	3.738	93.386 ± 0.503	3.240	94.766 ± 0.310	5.175	$\mathbf{95.135 \pm 0.218}$	23.342
Madelon	55.393 ± 1.732	0.129	55.521 ± 1.529	0.094	65.487 ± 1.441	3.096	$\mathbf{82.286 \pm 1.139}$	9.706
MNIST	91.191 ± 0.251	15.514	91.154 ± 0.221	10.375	93.993 ± 0.450	32.122	$\mathbf{96.155 \pm 0.091}$	101.941
Spam	$\mathbf{91.178 \pm 0.899}$	0.096	90.166 ± 0.921	0.085	90.582 ± 0.692	0.293	$\mathbf{91.137 \pm 0.696}$	1.715
Urban land cover	76.288 ± 2.860	0.044	75.303 ± 3.233	0.035	77.502 ± 2.419	0.082	$\mathbf{80.098 \pm 2.589}$	2.161

Experimental results

- The algorithms' performance for common datasets:



(a) DNA



(b) Isolet

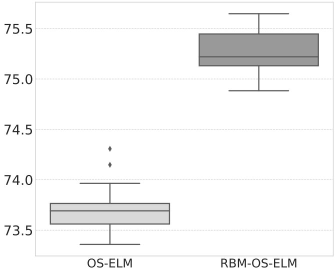
Experimental results

- The algorithms' performance for large datasets:

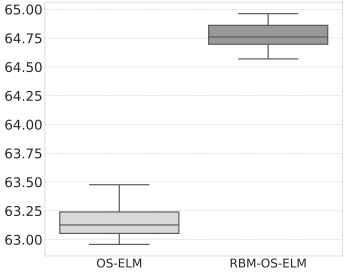
Dataset	OS-ELM		RBM-OS-ELM	
	Accuracy (%)	*Time (sec)*	*Accuracy (%)*	*Time (sec)*
Covertype	73.699 ± 0.200	34.765	$\mathbf{75.271 \pm 0.211}$	56.842
Higgs	63.155 ± 0.136	529.32	$\mathbf{64.975 \pm 0.104}$	1008.35
Susy	78.694 ± 0.098	245.436	$\mathbf{79.709 \pm 0.045}$	540.143

Experimental results

- The algorithms' performance for common datasets:



(c) Covertype



(d) Higgs

Conclusion

- The RBM-OS-ELM uses a straightforward idea to improve the OS-ELM
- Experimental results show that the proposed approach is able to improve the OS-ELM for most datasets
  - On the other hand, the OS-ELM is around two times faster than it
- It is a compromise between accuracy and computational time. If we have:
  - Hardware, time and the data is available at once $\rightarrow$ **RBM-ELM**
  - Time and the data is sequential $\rightarrow$ **RBM-OS-ELM**
  - All factors are very important $\rightarrow$ **OS-ELM**

Acknowledgements

- A. Pacheco would like to thank the financial support of the Brazilian agency CAPES
- R. Krohling thanks the financial support of the Brazilian agency CNPq under grant n. 309161/2015-0 and the local Agency of the state of Espirito Santo FAPES under grant n. 039/2016

Main references

[1] G.B. Huang, Q.Y. Zhu, and C.K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks". IEEE International Joint Conference on Neural Networks, vol. 2, pp. 985-990, 2004.

[2] N.Y. Liang, G.B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks", IEEE Transactions on neural networks, vol. 17, no. 6, pp. 1411–1423, 2006.

[3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks". Science, vol. 313, no. 5786, pp. 504–507, 2006.

[4] A. G. C. Pacheco, R. A. Krohling, and C. A. S. da Silva, "Restricted Boltzmann machine to determine the input weights for extreme learning machines". Expert Systems with Applications, vol. 96, pp. 77–85, 2018.