ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Machine Learning Techniques based on Random Projections

Yoan Miche[1,3], Benjamin Schrauwen[2] and Amaury Lendasse[3]

1 – Institut National Polytechnique de Grenoble – Gipsa-Lab
961 rue de la Houille Blanche, BP46, 38402 Grenoble – France

2 – Ghent University – Electronics and Information Systems department
Sint Pietersnieuwstraat 41, 9000 Ghent – Belgium

3 – Aalto University School of Science and Technology –
Dept. of Information and Computer Science
P.O. Box 15400, FI-00076 Aalto – Finland

Abstract. This paper presents a short introduction to the Reservoir Computing and Extreme Learning Machine main ideas and developments. While both methods make use of Neural Networks and Random Projections, Reservoir Computing allows the network to have a recurrent structure, while the Extreme Learning Machine is a Feedforward neural network only. Some state of the art techniques are briefly presented and this special session papers are finally briefly described, in the terms of this introductory paper.

1 Introduction

"Neural Networks and Machine Learning techniques are too slow and complicated to build, for our application".While this sentence might be controversial, it is nevertheless heard quite often when talking with people using large data sets on a daily basis.

Indeed, thanks to improvements in acquisition processes it is possible to obtain large quantities of information about a specific phenomenon, making the data to analyze more abundant, both in terms of number of variables and samples. And therefore, it becomes possible to train much more precise and finely tuned models. This unfortunately has a cost, which lies in the computational time required for such training, often related (at least) to the square of the number of variables and/or samples.

A seemingly easy solution to this situation is to decrease the number of variables to only the most useful and required ones for the considered problem. Two general ideas can be used: pruning the variables, or "combining" them, for example by projection. The problem remains however similar, for one has to either select which variables are relevant using a pruning strategy and a criterion, or find a way of combining which can be very time taking (optimizing a projection matrix for example).

For the matter of projecting to a lower dimensional space, Johnson and Lindenstrauss [13] have shown that for a set of N points in d-dimensional space (using an Euclidean norm), there exists a linear transformation of the data toward a d_f-dimensional space, with $d_f \geq O(\varepsilon^{-2}\log(N))$ which preserves the distances (and hopefully the "topology" of the data) to a $1 \pm \varepsilon$ factor. Achlioptas [1] has recently extended this result and proposed a very simple projection matrix that preserves the distances to the same factor than the J-L theorem mentions, at the expense of a probability on the distance conservation.

While such simple projections are attractive for they are very fast to build, their performances remain below the ones of state of the art techniques [6].

With this double requirement in mind (performance and speed), Machine Learning techniques based on Random Projections have been developing at an increasing rate recently, leading to two main frameworks:

- Reservoir Computing (RC) methods;
- Extreme Learning Machine (ELM) methods.

While this list is not exhaustive, most of the current and past machine learning techniques using random projections can be sorted into these categories. This paper aims at giving a short survey on the most promising techniques emerging from these categories and presenting the papers of this special session.

Section 2 discusses the Reservoir Computing framework, which comprises various specific methods such as Echo-State Networks (ESN) or Liquid-State Machines (LSM) for example. Section 3 proposes a description of the main ELM results and some recent techniques derived from it, such as OP-ELM, EM-ELM, *etc.*

Finally, Section 4 introduces the papers presented in this special session.

2 Reservoir Computing

The main difference between Reservoir Computing techniques and Extreme Learning Machine ones, lie in the recurrence of the underlying neural network. Indeed, Reservoir Computing makes use of a "pool" of neurons, randomly interconnected with each other, which can be described as a Recurrent Neural Network (RNN).

Theoretically, RNNs are powerful tools for solving complex spatio-temporal machine learning tasks. The application of RNNs to real-world machine learning problems is however not always feasible due to the high training costs and poor training convergence. The principal problem for RNN training is the so-called fading gradient: the error gradient vanishes or gets distorted by bifurcations, making training very slow, even with convergence improvements as in [2].

Reservoir Computing globally proposes a solution to finding the weights: do not adapt the internal connection weights (i.e. initialize them randomly) and train the output directly, using a specific classifier or regression method.

In the case of input data $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T, \mathbf{x}_i \in \mathbb{R}^d$, with output to model $\mathbf{y} = (y_1, \ldots, y_N)^T$ the output of the reservoir $\hat{\mathbf{y}}$ at step $k+1$ can be seen as

$$\begin{aligned} \hat{\mathbf{y}}(k+1) \quad = \quad & \mathbf{f}^{\text{out}}(\mathbf{W}^{\text{out}} \cdot [\mathbf{x}(k+1), \mathbf{f}^{\text{in}}(\mathbf{W}^{\text{in}} \cdot \mathbf{x}(k+1) \\ & +\mathbf{W}\mathbf{s}(k) + \mathbf{W}^{\text{back}}\mathbf{y}(k)), \mathbf{y}(k)]), \end{aligned} \tag{1}$$

with $\mathbf{x}(k)$ denoting the input data $\mathbf{x}$ fed to the network at step k, $\mathbf{W}^{\text{out}}$,$\mathbf{W}^{\text{in}}$, $\mathbf{W}$ and $\mathbf{W}^{\text{back}}$ the output weight matrix, input weight matrix, internal weight matrix and back-projection of output to internal network weight matrix respectively (random), $\mathbf{f}^{\text{in}}$ and $\mathbf{f}^{\text{out}}$ the internal network activation function (usually sigmoid) and readout function respectively and finally $\mathbf{s}$ denoting the internal network state.

Note that in the case where $\mathbf{f}^{\text{out}}$ is a linear classifier, this Reservoir Computing framework is described as Echo State Networks [11, 14, 12].

Liquid State Machines (LSM) [16] are rather similar to the ESN, except for the fact that the function $\mathbf{f}^{\text{in}}$ is usually a spiking neural network [15] or a network of threshold logic gates.

3 Extreme Learning Machine

The Extreme Learning Machine (ELM) algorithm is proposed by Huang *et al.* in [10] and uses Single-Layer Feedforward Neural Networks (SLFN). Hence, contrary to the Reservoir Computing, there is no recurrence in the neural network of ELM-based techniques. The key idea of ELM is the random initialization of the SLFN weights. Consider a SLFN with M hidden neurons; in the case it perfectly approximates the data (meaning the error between the estimated output $\hat{y_i}$ and the actual output y_i is zero), satisfies

$$\sum_{i=1}^{M} \beta_i f(\mathbf{w}_i \mathbf{x}_j + b_i) = y_j, 1 \leq j \leq N, \tag{2}$$

with f the activation function, $\mathbf{w}_i$ the input weights and β_i the output weights. This writes compactly as $\mathbf{H}\beta = \mathbf{y}$, with

$$\mathbf{H} = \begin{pmatrix} f(\mathbf{w}_1\mathbf{x}_1 + b_1) & \dots & f(\mathbf{w}_M\mathbf{x}_1 + b_M) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1\mathbf{x}_N + b_1) & \dots & f(\mathbf{w}_M\mathbf{x}_N + b_M) \end{pmatrix} \tag{3}$$

and $\beta = (\beta_1^T \dots \beta_N^T)^T$.

With these notations, the theorem presented in [10] states that with randomly initialized input weights and biases for the SLFN, and under the condition that the activation function f is infinitely differentiable, then the hidden layer output matrix can be determined and will provide an approximation of the target values with arbitrary precision (non-zero).

The output weights β can be computed from the hidden layer output matrix $\mathbf{H}$ and target values by using a Moore-Penrose generalized inverse of $\mathbf{H}$, $\mathbf{H}^\dagger$. Overall, the ELM algorithm is then:

Algorithm 1 ELM.

```
Given a training set $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, an activation function $f$ and the number of hidden nodes $M$,
    • Randomly assign input weights $\mathbf{w}_i$, $1 \leq i \leq M$;
    • Calculate the hidden layer output matrix $\mathbf{H}$;
    • Calculate output weights matrix $\beta = \mathbf{H}^\dagger \mathbf{Y}$.
```

The only parameter of the ELM algorithm is therefore the number of neurons to use in the SLFN. This is determined using an information criterion through model structure selection.

The proposed solution to the equation $\mathbf{H}\beta = \mathbf{y}$ in the ELM algorithm, as $\beta = \mathbf{H}^\dagger \mathbf{y}$ has three main properties making it a rather appealing solution:

- It is one of the least-squares solutions to the mentioned equation, hence the minimum training error can be reached with this solution;
- It is the solution with smallest norm among the least-squares solutions;
- The smallest norm solution among the least-squares solutions is unique and is $\beta = \mathbf{H}^\dagger \mathbf{y}$.

More recently, some possible improvements to the ELM have been proposed, in the form of wrappers, mostly.

Feng presents in [5] the Error Minimized Extreme Learning Machine, which proposes to add random neurons (one-by-one or group-by-group) to the ELM once the original training has been performed. The authors prove that this EM-ELM actually converges. The main interest in this development is in the fact that the computational time is minimal while adding new random neurons to the ELM. That is, the whole pseudo-inverse matrix does not need to be recomputed, but weights only have to be updated using fast update rules derived in the original paper. The main interest lies in the fact that one does not have to know beforehand the exact best number of neurons required for ELM to perform well. Neurons can anyway be added to the ELM without long computations afterwards.

Another example is the Optimally-Pruned Extreme Learning Machine (OP-ELM) [18, 17, 19, 21], which is designed to overcome an early problem encountered with ELM for irrelevant variables in the data set. The OP-ELM uses a large ELM (in the sense of a large number of neurons) of which it first ranks the neurons exactly (i.e. the ranking is exact) thanks to the Multiresponse Sparse Regression algorithm [20, 4]. The neurons are then pruned out using a Leave-One-Out criterion. The main goal of OP-ELM is neither to be the most accurate algorithm nor the fastest. It provides on the other hand, one of the best performance/speed ratio existing, even when compared with state of the art methods, such as Gaussian Processes or SVM.

4 Presented papers

In this special session, six different papers are presented, of which three deal with Reservoir approaches and three with ELM.

Gallichio and Micheli in **A Markovian Characterization of Redundancy in Echo State Networks by PCA** [8] make use of PCA applied to the matrix holding all the ESN internal network states $\mathbf{S} = (\mathbf{s}(1), \ldots, \mathbf{s}(k), \ldots, \mathbf{s}(K))^T$, if K represents the total number of steps considered. Interestingly, they show on a practical example that most of the variance of the internal network states $\mathbf{S}$ lies in only very few principal components; proving that the internal structure is highly redundant. Also, the Markovianity of the reservoir seems to be linked to the redundancy, as can be read in the paper.

The same authors in **TreeESN: a Preliminary Experimental Analysis** [9] propose an efficient approach to Recursive Neural Networks (RecNN, which are a generalization of Recurrent Neural Networks), of which the training can be even more time-consuming than RNN. The ESN approach is here extended to trees, with various possible state mappings. From the conducted experiments, it would seem that this approach while promising for some applications, depends currently highly on the Markovianity of the tasks at hand.

Butcher *et al.* make a smooth transition between RC and ELM methods, by presenting **Extending reservoir computing with random static projections: a hybrid between OP-ELM and RC** [3]. The idea is similar in concepts to the OP-ELM methodology, but applied on a Reservoir, instead of an ELM. Indeed, the authors present a technique named Reservoir with Random Static Projections making use of a standard reservoir and two static non-recurrent hidden layers with no layers interconnections. The first static layer is connected to both the input layer and the output layer, while the second static layer is connected to the reservoir and the output layer. The goal being to have a direct influence of the input data on the output, as well as a reservoir influence. The conducted experiments lead to significantly better results than a standard reservoir technique.

In **Using SVMs with randomised feature spaces: and extreme learning approach** [7], Frénay and Verleysen introduce a new kernel for Support Vector Machines: the ELM. By computing only the first layer of the ELM (that is, the random projection) and applying a SVM on it, the authors make a new kernel which is both much faster and as accurate (from their experiments) as the classical RBF. It is also shown experimentally that the chosen number of neurons for the ELM part (which represents the projecting dimensionality here) highly influences the performance of the model. It is then advised to use a large number of neurons in the first place to ensure a good accuracy through the SVM.

The mentioned EM-ELM gets an improvement in the form of the EEM-ELM proposed by Lan *et al.* in **Random Search Enhancement of Error Minimized Extreme Learning Machine** [23]. The enhancement of EM-ELM lies in the idea of pruning the group of randomly generated neurons before they are added to the ELM. From the newly generated group of neurons, only the node leading to lowest residual error will be finally added to the ELM. This leads to a more compact structure of the network while actually increasing its performance, as the authors demonstrate through numerous experiments in their paper.

Finally, a practical implementation of the ELM on graphics computing units (GPU) is developed in **Solving Large Regression Problems using an Ensemble of GPU-accelerated ELMs** [22] by Heeswijk *et al.* The paper not only deals with the implementation of ELM on GPU, but also about making an ensemble of many ELMs created on the GPU, to finally create linear combination of them with positive weights. The weights are obtained by solving the linear system composed of the validation outputs of all the created ELMs and the actual output, in a Non-Negative Least Squares sense (NNLS). The use of this methodology enables the authors to perform well while decreasing computational times (compared to a CPU version of the algorithm) in a 7 to 10-fold fashion.

References

[1] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

[2] A. F. Atiya and A. G. Parlos. New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Trans. Neural Networks*, 11:697–709, 2000.

[3] J. Butcher, D. Verstraeten, B. Schrauwen, C. Day, and P. Haycock. Extending reservoir computing with random static projections: a hybrid between op-elm and rc. In *European Symposium on Artificial Neural Networks (ESANN) 2010*, Bruges, Belgium, April 28-30th 2010.

[4] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. In *Annals of Statistics*, volume 32, pages 407–499. 2004.

[5] G. Feng, G.-B. Huang, Q. Lin, and R. Gay. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *Trans. Neur. Netw.*, 20(8):1352–1357, 2009.

[6] D. Fradkin and D. Madigan. Experiments with random projections for machine learning. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–522, New York, NY, USA, 2003. ACM.

[7] B. Frenay and M. Verleysen. Using svms with randomised feature spaces: an extreme learning approach. In *European Symposium on Artificial Neural Networks (ESANN) 2010*, Bruges, Belgium, April 28-30th 2010.

[8] C. Gallichio and A. Micheli. A markovian characterization of redundancy in echo state networks by pca. In *European Symposium on Artificial Neural Networks (ESANN) 2010*, Bruges, Belgium, April 28-30th 2010.

[9] C. Gallichio and A. Micheli. Treeesn: a preliminary experimental analysis. In *European Symposium on Artificial Neural Networks (ESANN) 2010*, Bruges, Belgium, April 28-30th 2010.

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

[11] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks, gmd report 148. Technical report, German National Research Institute for Computer Science, 2001.

[12] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 593–600. MIT Press, Cambridge, MA, 2003.

[13] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In *Conference in Modern Analysis and Probability*, pages 189–206, New Haven, USA, 1982.

[14] M. Buehner M. and P. Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, 2006.

[15] W. Maass and C. M. Bishop. *Pulsed Neural Networks.* MIT Press, Nov. 1998.

[16] W. Maass, R. A. Legenstein, and H. Markram. A new approach towards vision suggested by biologically realistic neural microcircuit models. In *BMCV '02: Proceedings of the Second International Workshop on Biologically Motivated Computer Vision*, pages 282–293, London, UK, 2002. Springer-Verlag.

[17] Y. Miche, P. Bas, C. Jutten, O. Simula, and A. Lendasse. A methodology for building regression models using extreme learning machine: OP-ELM. In M. Verleysen, editor, *ESANN 2008, European Symposium on Artificial Neural Networks, Bruges, Belgium*, pages 247–252. d-side publ. (Evere, Belgium), April 23-25 2008.

[18] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse. Op-elm: Optimally pruned extreme learning machine. *Neural Networks, IEEE Transactions on*, 21(1):158–162, December 2009.

[19] Y. Miche, A. Sorjamaa, and A. Lendasse. OP-ELM: Theory, experiments and a toolbox. In Roman Neruda Vera Kurkova and Jan Koutnik, editors, *LNCS - Artificial Neural Networks - ICANN 2008 - Part I*, volume 5163/2008 of *Lecture Notes in Computer Science*, pages 145–154. Springer Berlin / Heidelberg, September 2008.

[20] T. Similä and J. Tikka. Multiresponse sparse regression with application to multidimensional scaling. In *International Conference on Artificial Neural Networks (ICANN)*, pages 97–102, Warsaw, Poland, September 11-15 2005.

[21] D. Sovilj, A. Sorjamaa, Q. Yu, Y. Miche, and E. Séverin. Op-elm and op-knn in long-term prediction of time series using projected input data. *to appear in Neurocomputing*, 2010.

[22] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse. Solving large regression problems using an ensemble of gpu-accelerated elm. In *European Symposium on Artificial Neural Networks (ESANN) 2010*, Bruges, Belgium, April 28-30th 2010.

[23] L. Yuan, S. Yeng Chai, and H. Guang-Bin. Random search enhancement of error minimized extreme learning machine. In *European Symposium on Artificial Neural Networks (ESANN) 2010*, Bruges, Belgium, April 28-30th 2010.