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.

TreeESN: a Preliminary Experimental Analysis

Claudio Gallicchio and Alessio Micheli

Department of Computer Science - University of Pisa
Largo B. Pontecorvo, 3 56127 - Italy

Abstract.
In this paper we introduce an efficient approach to Recursive Neural Networks (RecNNs) modeling, the Tree Echo State Network (TreeESN), extending the Echo State Network (ESN) model from sequential to tree structured domains processing. For structure-to-element transductions, the state mapping (i.e. the way in which the state values for the whole structure are selected/collected) turns out to have a relevant role and the importance of its choice is pointed out by experimental results.

1 Introduction

Echo State Networks (ESNs) [1, 2] constitute an efficient approach for Recurrent Neural Networks (RNNs) modeling. ESNs typically consist in a large non-linear *reservoir* hidden layer of recurrent units connected to a linear *readout* layer. Only the readout is trained (e.g. by efficient linear regression), while the reservoir implements a fixed contractive state transition function.

Recursive Neural Networks (RecNNs) [3, 4] represent a generalization of RNNs for processing structured domains (SDs). Training RecNNs may be even more computationally expensive than training RNNs. It is therefore interesting to investigate efficient approaches to RecNNs modeling. Preliminary theoretical results on RecNNs implementing contractive state transition mappings and related Markovian effects [5] on network dynamics have been proposed in [6]. Inspired by the ESN approach, in this paper we introduce a neural networks model for processing tree SDs named the *Tree Echo State Network* (TreeESN).

2 TreeESNs for Tree Domain Processing

In the following, a rooted tree structure is represented by $\mathbf{t}$. Children of a node n in $\mathbf{t}$ are represented by $ch_1(n), \ldots, ch_k(n)$, where k is the ariety of $\mathbf{t}$. If the i-th child of n is not present, then $ch_i(n) = nil$. If n is a leaf node $ch_i(n) = nil \quad \forall i = 1, \ldots, k$. The label information associated to node n is denoted by $\mathbf{u}(n)$. The set of trees with node labels in $\mathbb{R}^{N_U}$ and ariety k is indicated by $(\mathbb{R}^{N_U})^{\#k}$, while $(\mathbb{R}^{N_U})^{\#}$ is used if the ariety is not specified.

We are interested in computing transductions on tree SDs. A *tree transduction* is a function mapping an input tree SD into an output tree SD, $\mathcal{T} : (\mathbb{R}^{N_U})^{\#} \rightarrow (\mathbb{R}^{N_O})^{\#}$, where $(\mathbb{R}^{N_U})^{\#}$ denotes the input set of trees with node labels in the (vectorial) real input space $\mathbb{R}^{N_U}$ and $(\mathbb{R}^{N_O})^{\#}$ is the set of trees with node labels in the (vectorial) real output space $\mathbb{R}^{N_O}$.

In a *tree-to-tree* transduction $\mathcal{T}$ the output is always isomorphic to the input structure, i.e. $\mathcal{T}$ associates an output node for each node of the input tree. $\mathcal{T}$ is said a *tree-to-element* (or *structure-to-element*) transduction if a single output element is computed in correspondence of an input tree, in this case the output domain reduces to a real subspace. $\mathcal{T}$ is said a *causal* transduction if the value computed for each node n depends on the node itself, its associated label $\mathbf{u}(n)$ and the descendants of n. In a *stationary* transduction $\mathcal{T}$, the function applied by $\mathcal{T}$ does not depend on the node to which it is applied. An *adaptive* transduction is learned from observed data and not a-priori fixed. A tree transduction $\mathcal{T}$ can be usefully decomposed as $\mathcal{T} = \hat{g} \circ \hat{\tau}$, where $\hat{\tau}$ is the *encoding* function and $\hat{g}$ is the *output* function. The encoding function $\hat{\tau} : (\mathbb{R}^{N_U})^{\#} \rightarrow (\mathbb{R}^{N_R})^{\#}$ maps the tree input domain into a tree structured feature space $(\mathbb{R}^{N_R})^{\#}$, where $\mathbb{R}^{N_R}$ is a real feature space. The output function $\hat{g}$ maps the tree feature representation into the output space: $\hat{g} : (\mathbb{R}^{N_R})^{\#} \rightarrow (\mathbb{R}^{N_O})^{\#}$. Causal functions $\hat{\tau}$ can be computed by using a node-applied recursive function $\tau : \mathbb{R}^{N_U} \times (\mathbb{R}^{N_R})^k \rightarrow \mathbb{R}^{N_R}$, where $(\mathbb{R}^{N_R})^k$ for each node represents the contextual information corresponding to the set of features in $\mathbb{R}^{N_R}$ computed for its children. Analogously, function $\hat{g}$ can be computed by using a node-applied readout function $g : \mathbb{R}^{N_R} \rightarrow \mathbb{R}^{N_O}$.

A TreeESN is a RecNN model implementing causal, stationary and partially adaptive tree transductions: it consists in a large hidden *reservoir* layer of N_R recursive neurons implementing a fixed encoding function plus a linear *readout* layer implementing an adaptive output function. The node-applied function τ is the reservoir *state transition function*, computed as:

$$\mathbf{x}(n) = \tau(\mathbf{u}(n), \mathbf{x}(ch_1(n)), \ldots, \mathbf{x}(ch_k(n)) = f(\mathbf{W}_{in}\mathbf{u}(n) + \sum_{i=1}^{k} \hat{\mathbf{W}}_i \mathbf{x}(ch_i(n))) \quad (1)$$

where $\mathbf{x}(n) \in \mathbb{R}^{N_R}$ is the state (i.e. the feature representation) computed for node n and $\mathbf{x}(ch_i(n))$ is the state computed for the $i-th$ child of n, with $\mathbf{x}(nil) = \mathbf{0} \in \mathbb{R}^{N_R}$. Matrix $\mathbf{W}_{in}$ is the input-to-reservoir weight matrix, while $\hat{\mathbf{W}}_i$ denotes the recurrent reservoir weight matrix for child i. As non-linear activation function f we use *tanh*. The states computed for the children of a node n can be concatenated into a *context state* $\mathbf{x}_c(n) = [\mathbf{x}(ch_1(n)); \ldots ; \mathbf{x}(ch_k(n))] \in \mathbb{R}^{kN_R}$. Analogously, the recurrent weight matrices can be arranged into a global matrix $\hat{\mathbf{W}} = [\hat{\mathbf{W}}_1 \ldots \hat{\mathbf{W}}_k] \in \mathbb{R}^{N_R \times kN_R}$, such that $\mathbf{x}(n) = \tau(\mathbf{u}(n), \mathbf{x}_c(n)) = f(\mathbf{W}_{in}\mathbf{u}(n) + \hat{\mathbf{W}}\mathbf{x}_c(n))$. The readout function g is computed by the linear readout layer. For tree-to-tree transductions the output is computed for every node n as $\mathbf{y}(n) = g(\mathbf{x}(n)) = \mathbf{W}_{out}\mathbf{x}(n)$. For structure-to-element transductions a *state mapping* function $\mathcal{X} : (\mathbb{R}^{N_R})^{\#} \rightarrow \mathbb{R}^{N_R}$ is preliminary used to map the structured feature representation computed for the whole input structure into a fixed-size state representation. In this case the output is computed for the whole input tree $\mathbf{t}$ as $\mathbf{y}(\mathbf{t}) = g(\mathcal{X}(\hat{\tau}(\mathbf{t}))) = \mathbf{W}_{out}\mathcal{X}(\hat{\tau}(\mathbf{t}))$. The standard RecNN choice for $\mathcal{X}$ is a *root state mapping*, consisting in selecting the state of the root node of the input tree. Another possible implementation of $\mathcal{X}$, used in the following and proposed in [7], is a *mean state mapping*, evaluating the mean value among all the states computed for the nodes in the input tree.

As in standard ESNs, the reservoir of a TreeESN is left untrained after initialization (i.e. $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ are fixed) and the linear readout is adapted through linear regression (i.e. $\mathbf{W}_{out}$ is learned). The reservoir transition function is initialized to be a contraction, i.e. $\exists C \in (0,1)$ s.t. for every couple of context states $\mathbf{x}_c, \mathbf{x}'_c$ and input label $\mathbf{u}$, $\|\tau(\mathbf{u},\mathbf{x}_c) - \tau(\mathbf{u},\mathbf{x}'_c)\| \leq C\|\mathbf{x}_c - \mathbf{x}'_c\|$ holds. Using the Euclidean norm and for *tanh* activation function, the contractivity condition is satisfied for $\|\hat{\mathbf{W}}\|_2 < 1$. In fact, in this case $\|\tau(\mathbf{u},\mathbf{x}_c) - \tau(\mathbf{u},\mathbf{x}'_c)\|_2 = \|tanh(\mathbf{W}_{in}\mathbf{u} + \hat{\mathbf{W}}\mathbf{x}_c) - tanh(\mathbf{W}_{in}\mathbf{u} + \hat{\mathbf{W}}\mathbf{x}'_c)\|_2 \leq \|\hat{\mathbf{W}}(\mathbf{x}_c - \mathbf{x}'_c)\|_2 \leq \|\hat{\mathbf{W}}\|_2\|\mathbf{x}_c - \mathbf{x}'_c\|_2$. $\|\hat{\mathbf{W}}\|_2$ is denoted by σ and is called the *contraction coefficient* of the TreeESN as it rules the contractivity (at least in the Euclidean norm) of its state transition function. Note that the value of σ is equal to the maximum singular value of matrix $\hat{\mathbf{W}}$. Matrix $\hat{\mathbf{W}}$ is randomly initialized and then scaled to the desired value of σ as for ESNs. The contractive setting of RecNN state transition functions involves a Markovian organization of the network state dynamics [6] which applies to TreeESNs (with root state mapping) as well. As a consequence, more similar network states correspond to different input structures sharing a larger common suffix (i.e. a common top subtree starting from the root).

Note that if the input domain reduces to a sequential domain, the TreeESN model with root state mapping reduces to a standard ESN.

3 Experiments

We applied the TreeESN model on two benchmark tasks related to structure-to-element transductions. The first one is a Quantitative Structure-Property Relationship (QSPR) analysis of alkanes [4, 7], to which RecNNs and other neural networks models for SDs have been successfully applied (see [7] and references therein). The task consists in predicting the boiling point (measured in Celsius degrees, ^{o}C) of the alkanes on the basis of tree representations composed by the Carbon atoms (with $k = 3$), whereas the input dataset contains 150 molecules. A 10 fold cross-validation method was used for training and testing.

The second task is a predictive task on a sequential symbolic domain, with Markovian/anti-Markovian flavor, to which ESNs have been applied in [8]. In this case input trees reduce to input sequences (i.e. $k = 1$), involving a temporal dimension such that the root node corresponds to the most recent input symbol. Elements of the input sequences are randomly chosen from a uniform distribution over a numerical representation of the symbolic alphabet $\{a, \ldots, j\}$, such that a is represented by 0.1 and so on up to j represented by 1.0. The target value associated to an input sequence $\mathbf{t}$ depends on a parameter $\lambda > 0$, which controls the degree of Markovianity/anti-Markovianity of the task. The target value is computed as: $\hat{\mathbf{y}}(\mathbf{t}) = \sum_i \frac{\mathbf{u}(i)}{\lambda^i}$ and normalized in $[-1, 1]$, where the sum spans over the set of nodes in the input structure for increasing values of i. For Markovian sequences the target $\hat{\mathbf{y}}(\mathbf{t})$ is obtained by scanning input nodes from the root to the leaf, such that more similar target values are associated to input sequences with more similar suffixes. A larger value of λ implies a stronger Markovian characterization of the target. For anti-Markovian sequences, the target is obtained by scanning the input sequence in the opposite direction, leading to an anti-Markovian characterization of the target controlled by the value of λ. For the Markovian and the anti-Markovian tasks, training and test sets contain respectively 500 and 100 sequences, of length between 50 and 100.

We tested TreeESNs with σ varying between 0.1 and 1.4. We used full connected reservoirs, with dimension (N_R) set to 500 for the alkanes task and to 100 for the Markovian/anti-Markovian sequences task. $\mathbf{W}_{in}$ values were randomly chosen from a uniform distribution over $[-1, 1]$. Values in $\hat{\mathbf{W}}$ were initialized in the same way and such that its sub-matrices $\hat{\mathbf{W}}_1, \ldots, \hat{\mathbf{W}}_k$ were all identical. $\hat{\mathbf{W}}$ was then rescaled to the desired value of σ. For the alkanes task a uniform noise of size 10^{-5} was added to the input for the readout before training. Error measures reported were averaged over 30 and 10 independent trials for the alkanes and the Markovian/anti-Markovian sequences tasks, respectively.

Fig. 1 shows the mean absolute test error on the alkanes task obtained by TreeESNs with root state and mean state mappings. For increasing values of σ,

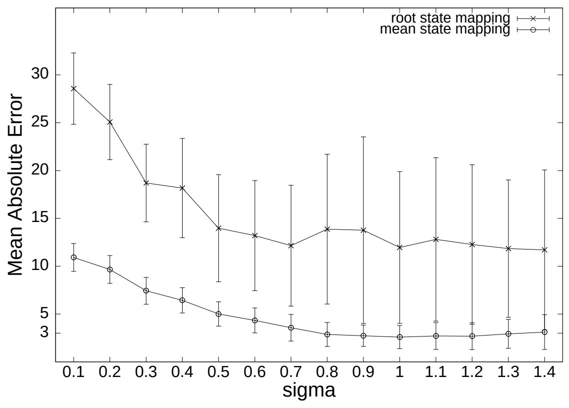

Fig. 1: Mean absolute 10 fold cross validation test error (and std. deviation) on the alkanes task by TreeESNs with root state mapping and mean state mapping. Reservoir dimension equal to 500 units and σ varying between 0.1 and 1.4.

the performance of the model is increased. However, for larger values of σ, such result is of a limited significance, due to the relative short depth of the trees in the dataset. More importantly, it is evident that the performance of TreeESNs resulted to be sensitive to the choice of the state mapping. Results obtained with standard root state mapping were rather worse than those obtained with mean state mapping. The best result achieved by TreeESNs with mean state mapping, in correspondence to $\sigma = 1.0$, is 2.595 ^{o}C. What is really noteworthy is the fact that such result, obtained by a model with fixed encoding, is comparable to those achieved by RecNNs with learning, though inferior to other learning models for SDs [7] (see Table 1). In particular, the largest test errors were found in correspondence of the smallest alkanes, which are in a high non-linear relation with the their boiling point.

Note that the task of predicting the boiling point of alkanes violates the

Model	**RCC**	**CRCC**	**NN4G**	**TreeESN** (mean)	**TreeESN** (root)
Error	2.87	2.56	1.74	2.60	11.71

Table 1: Best mean absolute test errors on the alkanes task for Recursive Cascade Correlation (RCC), Contextual RCC (CRCC), Neural Networks for Graph (NN4G) and TreeESN models with mean and root state mapping.

Markovian assumption, being related to the number of nodes and to the branching of the molecular structure [4]. Therefore RecNNs with contractive state transition functions, such are TreeESNs with root state mapping, being characterized by Markovian (suffix biased) dynamics are unsuitable for it. However, due to the peculiarity of the task, the target is consistent with the mean operator, which equally treats every node in a tree independently of its position and allows the readout to distinguish among trees with identical suffixes but different dimensions. In the following, using a critical approach, we show how the same mean state mapping could be less helpful for other classes of tasks, even restricting to sequence domains.

The mean squared test errors obtained by TreeESNs with mean state mapping on the Markovian/anti-Markovian tasks are reported in Fig. 2.

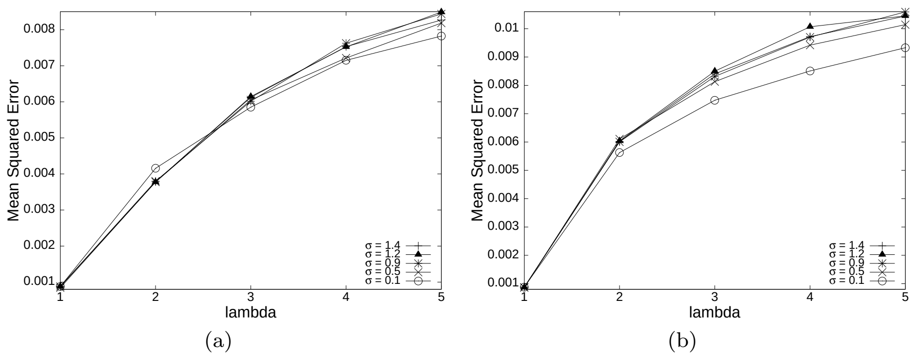

Fig. 2: Mean squared test errors on the Markovian (**a**) and on the anti-Markovian (**b**) sequences tasks for increasing values of λ by TreeESNs with mean state mapping, 100 reservoir units and σ varying between 0.1 and 1.4.

In this case the mean operator has the effect of merging state information from every time step in the input sequence, such that the relevance of prefixes and suffixes is the same. Thus the Markovian/anti-Markovian characterizations of the tasks cannot be caught by the model and poor results are obtained. Such results are almost the same for the Markovian and the anti-Markovian sequences, with increasing errors for increasing Markovian/anti-Markovian characterization of the target (i.e. for increasing λ) and for decreasing contractivity of the state transition function (i.e. for increasing σ). Note that the best results were found for $\lambda = 1$, in which case the target is no more characterized by either Markovian or anti-Markovian properties. ESNs have been applied to the same benchmarks in [8], reporting a test error of 6.1983×10^{-10} for the Markovian sequences and of 0.17249 for the anti-Markovian ones ($\lambda = 2$, $\sigma = 0.9$ and 100 reservoir units for both the cases). Such results correspond to TreeESNs with root state mapping as well. The mean operator was not helpful by itself for solving the task with anti-Markovian sequences (achieving just a slightly better result than ESNs). Even more remarkable is the fact that it led to a worsening of the performance (with respect to ESNs) in the case of Markovian sequences.

4 Conclusions

We have presented a preliminary experimental analysis of a generalization of the ESN approach to trees (TreeESN) using different state mappings. A root state mapping, preserving the Markovian nature of TreeESN dynamics, was found to be inappropriate for a task on a chemical domain violating the Markovian assumption. On such a task, a mean state mapping resulted in better performances, even comparable with those of other RecNNs with adaptive encodings. However, the effectiveness of the mean operator turned out to be related to the peculiarity of the task at hand rather than to provide a solution to the Markovian bias. In fact, TreeESNs with mean state mapping were unable to solve a predictive task with a strict Markovian flavor, neither in the form violating or respecting the Markovian assumption (for which ESNs with root state mapping are naturally suitable). Although adaptive encodings provide more flexible solutions to SD learning, fixed-encoding models like TreeESNs still offer very efficient solutions that can result (less or more) appropriate according to the Markovian characterization of the task and to the effect of the state mapping used to extract the encoded state information. This investigation can help in supporting the critical exploitation of the model in relational learning tasks.

References

[1] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[2] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Tech.Rep. 148, GMD - German National Research Institute for Computer Science, 2001.

[3] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[4] A.M. Bianucci, A. Micheli, A. Sperduti, and A. Starita. Application of cascade correlation networks for structures to chemistry. *Applied Intelligence*, 12:117–146, 2000.

[5] P. Tiño, M. Cernanský, and L. Benusková. Markovian architectural bias of recurrent neural networks. *IEEE Transactions on Neural Networks*, 15(1):6–15, 2004.

[6] B. Hammer, P. Tiño, and A. Micheli. A mathematical characterization of the architectural bias of recursive models. Technical Report 252, Universitat Osnabruck, Germany, 2004.

[7] A. Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[8] C. Gallicchio and A. Micheli. Architectural and markovian factors of echo state networks. Submitted to journal. Technical Report TR-09-22 (November 2009) available at `http://compass2.di.unipi.it/TR/Files/TR-09-22.pdf.gz`.