ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-1129

Implementing an Intelligent Moving Average with a Neural Network

Nuno C. Marques and **Carlos Gomes** [1]

Abstract. Recent results in hybrid neural networks using extended versions of the core method have shown that we can use background knowledge to guide back-propagation learning. This paper further explores this ideas by adding numeric functions to the encoded knowledge and using the traditional recursive Elman neural network model. An illustration of the properties of these neural networks will be used to calculate a simple moving average. Simulations on generated data and on the $Eurostoxx50$ financial index will illustrate the potential of such a strategy.

1 Introduction

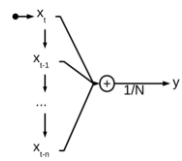
Artificial neural networks (ANN) are best known for their capability of learning how to classify unseen information from classified examples. They are one of the most efficient techniques for pattern recognition and data classification in highly complex and nosy supervised domains. However, ANN are also general learning devices where it is hard to understand how learning is taking place or where (and how) knowledge is represented. This paper explores recent results in hybrid neural networks that use extended versions of the core method ([4], [5]) to propose an implementation of a dynamic moving average. This dynamic moving average seems of particular interest as an advisor for stock market investment.

Recently, [4] has proposed the use of hybrid neuro-symbolic methods for encoding logical programs in a feed-forward neural network core (the model is usually called *the core method*). Moreover, in [2] the authors present experimental evidence that backpropagation learning can be guided by encoding logical rules. Unfortunately [5] shows that logical knowledge may not be enough for different classes of problems. This proposal goes one step further, by generalising the represented knowledge from $true - false$ values to real values and by explicitly modelling the time series inside the network.

2 Representing knowledge for training Elman Neural Networks

Please consider a discrete input time signal x_t (for $t = 1...m$), we can represent a moving average of the last N values, by equation in figure 1 (a). This calculation can also be easily represented by a graph like the one in figure 1 (b). In the graph each node can be associated with a state or variable. Each arrow represents the memorisation of a given (previous state) value by other state. When a transition has a number associated to it, this memorisation should be multiplied by previous state value. In the graph the addition corresponds to the sum of all connected variables.

$$y(t) = \frac{\sum_{i=0}^{N-1} x_{t-i}}{N}$$



(a) (b)

Figure 1. Equantion and derived graph for calculating a moving average.

The graph in figure 1 (b) can be directly translated into a semi-recursive Elman network [3]. Our goal will be to support the computation of function 1 (a) while maintaining a method that could be compatible with core method encoding for logical programs. The node specifying the addition will be seen as a logical and, while an additional layer could implement the logical or layer[2]. To implement the logic operators a step function is needed. We will use the non-linear effect of the hyperbolic tangent ($htan$): real value encodings can be achieved while keeping backpropagation learning by following the ideas in [5]. For this work we will assume that all values for x_t are in range $[-0.1, 0.1]$.

Elman Neural Network Encoding The first layer in the graph will be represented by a hidden unit layer in the neural network and a contextual layer (in an Elman neural network [3]). The initial state in the graph will represent the input value for the neural network (i.e. the value of the time series x_t), and will be connected (without further information we will assume weights of 1.0) to a hidden unit x_t. Then each transition among $x_t \rightarrow x_{t-1}$ will represent a recursive connection from x_t to a memory contextual unit x_{t-1}, followed by a feed forward connection to the hidden layer unit x_{t-1}.

Implemented neural network can be seen as a non-parametric statistical model of the proposed graph. To study the convergence of model while using a learning algorithm (namely, Elman's backpropagation through time [3]) a small amount of noise (± 0.01 or 2% of neural weights) was applied to connection weights. The output neuron implements the addition for calculating y, receiving connections from the hidden layer using $1/N$ as a multiplying factor.

Testing the Moving Average Elman Network We have compared the Elman network with the proposed encoding (ini) with the Elman

[1] CENTRIA — Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa and GoBusiness, email: maximus-ai@gobusiness.pt.

[2] For convenience we will use the disjunctive normal form instead of the more usual —but equivalent— Horn clause formalism.

network with all non recursive weighs randomly initialised (denoted rnd). Then we have considered an input time series generated by equation $input_t = A \cdot sin(kt) + (B \cdot sin(kt))^2$. This equation has two (overlapping) periodic components. The quadratic sinusoid wave reinforces positive values and reduces the amplitude of negative values in the linear sinusoid (the acquired behaviour has some resemblance to stock markets and is simple enough to this study). A family of related equations is randomly generated by selecting A and B amplitudes as free random factors (values between 0 and 1). Angular frequency k was always set to $\frac{1}{10}$. Random noise was added over this basic *signal* component (values between -0.5 and 0.5). The output is a moving average over the previous two values of the input signal. Values are then normalized to the $[-0.1, 0.1]$ range. An example is represented in figure 2 – A, for $A = 0.48636$, $B = 0.11116$ after adding the *noise* component: when learning this moving average, the *signal* component in the input time series will be easily modeled by the Elman network based on previous input [3], however the *noise* component will be reflected on the moving average (and, can not be modelled based on previous values).

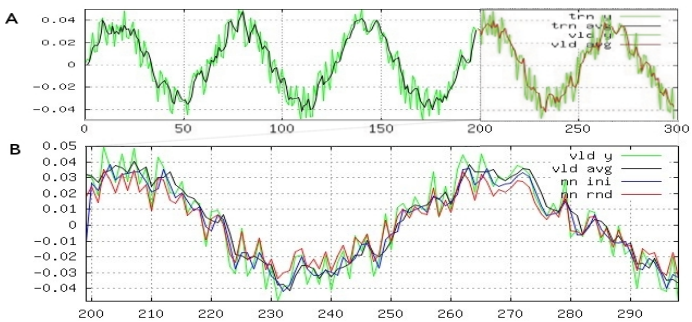


Figure 2. Output of the proposed (nn_{ini}) and standard (nn_{rnd}) models when compared with series input (trn/vld_y) and corresponding moving average ($trn/vld_y\ avg$). Subplot A presents train and test set, while subplot B zooms on test set.

Comparing the output of both trained networks regarding the target moving average output (figure 2 — B), we notice a much smoother output of the average made by the initialised neural network (please recall we have a 2% noise so weights are not 100% correct). Contrasting with this, the non-initialised network is much more sensible to noise, and presents a tendency to be attracted to the signal global average (i.e. the output of the starting random model initialised with small weights). This pattern was observed in several thousand independent runs. The bar graph of figure 3 quantifies the difference to the target moving average among the standard and the proposed model for validation data. Positive values represent the improvement made by proposed method.

A total of 500 runs were made (each with distinct randomly selected values and generating a distinct time series). Both networks were fully trained with backpropagation though time for each run ($\eta = 0.2$). Full convergence was achieved in less than 2000 iterations by setting $error_{max} = 0.01$ in all runs and networks (initialised networks usually achieve convergence after only 100 iterations, but rnd networks need the 2000 iterations for good convergence). The average error (as calculated from the distances plotted in figure 3) was taken. After 250000 such observations, the total reported error was 57.295, i.e. initialised neural network was, in average around 25% better (and less sensible to noise) than the standard Elman network.

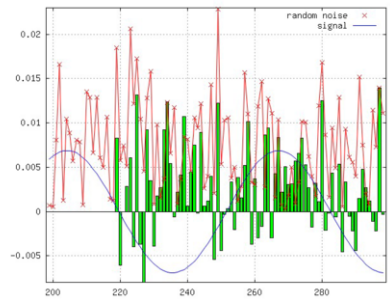


Figure 3. Distance of errors among outputs of the proposed and standard Elman on validation data (green bars) printed against $input_t$ signal and noise components of input on one of the runs.

Experimental results on $SX5E$ An adapted version of described method was tested over the European *DJ EURO STOXX 50* financial index ($SX5E$: a weighted index of 50 European stocks). A financial simulator was developed. The value calculated by implemented neural network (Ima) was used as an early warning for selling stocks (i.e. whenever the value of the $SX5E$ index dropped bellow neural Ima index) or as a cautious advice for buying stocks (if neural Ima is above the $SX5E$ index). Global return of this Ima index was 58% a quite surprising and unexpected value during a financial crisis period (the index lost 17% of its value). Under the same conditions a simple 40 day moving average has a (negative) return of -37%.

3 Conclusions

With the proposed method we have illustrated how we can encode a moving average calculation inside a traditional neural network having learning algorithms and properties quite well studied: the Elman Neural Network [3]. Backpropagation though time algorithm was still used to discover patterns in the dataset. However that search was guided, in the sense that we always want some kind of index or moving average. Proposed neural network can conjoin the encoding of both statistical and logic models, and then learns to adjust (i.e. fits) to experimental data. This way a more formal and general formalisation of this theory is now being pursued. Future work addresses the inclusion of economic models for reasoning (based on fundamental and technical features) and on what is the most probable short term economic scenario (so that the best moving average can be selected). Results are so promising that we are now starting the process for the creation of a financial fund based on presented ideas. This financial fund should make the described strategy available to all interested investors [1].

REFERENCES

[1] C. Gomes, 'Maximus investment fund', Tech. report, GoBusiness, (2010).
[2] S. Bader, S. Hölldobler, and N. Marques, 'Guiding backprop by inserting rules', in *ECAI08 Workshop on Neural-Symbolic Learning and Reasoning, Patras, Greece*, vol. 366, CEUR, (2008). ISSN: 1613-0073.
[3] Jeff L. Elman, 'Finding structure in time', *Cognitive Science*, **14**, 179–211, (1990).
[4] S. Hölldobler and Y. Kalinke, 'Towards a massively parallel computational model for logic programming', in *ECAI94 Workshop on Combining Symbolic and Connectionist Processing*, pp. 68–77. ECCAI, (1994).
[5] N. Marques, 'An extension of the core method for continuous values: Learning with probabilities', in *EPIA09: New Trends in Artificial Intelligence*, pp. 319–328. (2009).