Training Recurrent Connectionist Models on Symbolic Time Series *

Michal Čerňanský[1] and Ľubica Beňušková[2]

[1] Faculty of Informatics and Information Technologies, STU Bratislava, Slovakia
[2] Department of Computer Science, University of Otago, Dunedin, New Zealand
cernansky@fiit.stuba.sk, lubica@cs.otago.ac.nz

Abstract. This work provide a short study of training algorithms useful for adaptation of recurrent connectionist models for symbolic time series modeling tasks. We show that approaches based on Kalman filtration outperform standard gradinet based training algorithms. We propose simple approximation to the Kalman filtration with favorable computational requirements and on several linguistic time series taken from recently published papers we demonstrate superior ability of the proposed method.

1 Introduction

To process data with spatio-temporal structure recurrent neural networks (RNNs) were suggested. RNNs were successfully applied in many real-life applications where processing time-dependent information was necessary. Unlike feedforward neural networks, units in RNNs are fed by activities from previous time steps through recurrent connections. In this way contextual information can be kept in units' activities, enabling RNNs to process time series.

Many commonly used real-world data with time structure can be expressed as a sequence of symbols from finite alphabet - symbolic time series. Since their emergence neural networks were applied to symbolic time series analysis. Especially popular is to use connectionist models for processing of complex language structures. One of the main driving forces behind such studies has been formulating the models of human performance in processing linguistic patterns of various complexity [1, 2]. Other works study what kind of dynamical behavior has to be acquired by RNNs to solve particular tasks such as processing strings of context-free languages, where counting mechanism is needed [3, 4].

Gradient descent approaches such as backpropagation through time or real-time recurrent learning algorithms are widely used by researchers working with symbolic sequences. The aim of this paper is to show how KF-based techniques used for training RNNs can deal with symbolic time series. We also compare standard means square error cost function with cost function based on the entropy. Simple approximation significantly reducing the complexity of the Kalman filtration is proposed and results on the linguistic datasets taken from recently published paper are shown.

* This work was supported by the grants VG-1/0848/08 and VG-1/0822/08

2 Recurrent Neural Network Training

Elman's simple recurrent network (SRN) proposed in [1] is probably the most widely used RNN architecture.

It was trained using well-known error backpropagation algorithm. Although simple backpropagation is not appropriate algorithm for training recurrent networks it can be used for simple tasks or as a reference method for other approaches.

Backpropagation Through Time

Backpropagation through time algorithm (BPTT) provides precise error gradient computation for RNNs training. The trick consists of unfolding an RNN in time into regular feedforward network and then apply standard backpropagation directly. Usually approximation to the precise calculation is performed and recurrent network is unfolded into feedforward network only for several time steps called time window. Later theoretical results revealed that when acquiring long-term dependencies the error information of gradient-based training algorithms tends to vanish or blow up, therefore unfolding the recurrent network backwards for long time interval is not necessary [5]. Probably the most contributive article regarding BPTT is [6].

Adaptation of Extended Kalman Filter to Neural Network Training

Training of Elman's SRN, and generally any other multilayer perceptron network (recurrent or not), can be regarded as an optimal filtering problem [7]. Multilayer perceptron network can be described as nonlinear system by

$$\mathbf{x}(t) = \mathbf{x}(t-1), \tag{1}$$

$$\mathbf{z}(t) = h(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{v}(t), \tag{2}$$

where state $\mathbf{x}(t)$ is vector of network weights. Weights of trained network do not change and so state transition matrix $\mathbf{F} = \mathbf{I}$ where $\mathbf{I}$ is identity matrix. Measurement $\mathbf{z}(t)$ stands for desired values and measurement function $\mathbf{H}(t)$ is nonlinear function of network weights $\mathbf{x}(t)$ and input $\mathbf{u}(t)$. Jacobian matrix $\mathbf{H}(t)$ is calculated in every time step k as

$$\mathbf{H}(t) = \frac{\partial \mathbf{H}(\hat{\mathbf{x}}(t), \mathbf{u}(t))}{\partial \mathbf{x}}. \tag{3}$$

The set of EKF equations for the network training can be formulated as follows:

$$\mathbf{K}(t) = \mathbf{P}(t-1)\mathbf{H}^T(t)[\mathbf{H}(t)\mathbf{P}(t-1)\mathbf{H}^T(t) + \mathbf{R}(t)]^{-1}, \tag{4}$$

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \mathbf{K}(t)\mathbf{H}(t)\mathbf{P}(t-1) + \mathbf{Q}(t), \tag{5}$$

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mathbf{K}(t)[\mathbf{D}(t) - \mathbf{O}(t)]. \tag{6}$$

Let n_w and n_o denote the number of all network weights and number of output units, respectively. $\mathbf{W}$ is a vector of all weights (concatenated from matrices $\mathbf{W}^{RI}$, $\mathbf{W}^{RC}$, $\mathbf{W}^{OR}$) of the length n_w. $\mathbf{H}$ is the Jacobian matrix, $n_o \times n_w$, calculated in every

time step and containing in rows the derivatives of corresponding output activity with respect to all weights. These derivatives can be calculated by routines similar to BPTT [6]. $\mathbf{P}$ is the $n_w \times n_w$ error covariance matrix, it holds error covariances corresponding to each pair of network weights. The $n_w \times n_o$ matrix $\mathbf{K}$ called the Kalman gain is used in updating the weights $\mathbf{W}$ according to the difference between the desired output vector $\mathbf{D}$ and actual network output $\mathbf{O}$. The $n_o \times n_o$ matrix $\mathbf{R}$ stands for the measurement noise covariance matrix and similarly to the learning rate in RTRL or BPTT can control the training speed of EKF. Higher values represent higher amount of uncertainty attributed to the difference $\mathbf{D}(t) - \mathbf{O}(t)$ leading to slower training. Note, that small process noise is still considered: the $n_w \times n_w$ matrix $\mathbf{Q}$ stands for the process noise covariance matrix. Nonzero process noise improves convergence of the filter.

Entropy Cost Function Training Algorithms

Output units of RNNs are often equiped with logistic sigmoid or linear activation function and quadratic-error cost function was minimized in order to derive equations for weight adaptation. Alternative approach better suited to the symbolic time series processing scenario is to use output units with linear activation function and soft-max combining function. Predictive performance is evaluated using the probabilities $p(t)$ calculated using soft-max combining function:

$$p(t) = \frac{\exp\left(o_i(t)\right)}{\sum_j \exp\left(o_j(t)\right)}, \tag{7}$$

where o_i is the activation of output unit corresponding to the observed target symbol. Training algorithms are modified in order to minimize entropy cost function $E(t)$ in each weight update step t:

$$E(t) = -\log_{|A|} p(t). \tag{8}$$

Entropy Cost Function KF Approximation

Kalman filter is derived to minimize quadratic error cost function. Approximation technique is proposed to minimize the entropy. Neural network can be considered as a system with a single output of $-\log p(t)$ where the probability $p(t)$ is calculated from output unit activities by soft-max combining function (Eq. 7). Advantage of the single output system is that no matrix inversion need to be calculated what can significantly reduce the time complexity of the algorithm. The price for reducing the computational complexity may be poorer training performance. In this work this approximation to the Kalman filtration for RNN training will be called SM-EKF (soft-max EKF). Similar KF approximation as SM-EKF called "scalar error training" was proposed in [7].

3 Experiments

Standard gradient descent training techniques represented by simple backpropagation and backpropagation through time algorithms are first compared with standard extended

Kalman filter adopted for RNN training with derivatives calculated by BPTT-like algorithm. Standard versions of algorithms derived using qudratic-error cost function and also versions using soft-max combining function minimizing entropy cost function were compared on simple symbolic dataset: sequence of quantized activations of real laser and the sequence generated by context free grammar. Then proposed approximation to the Kalman filtration is used for processing linguistic datasets and resulting performance is compared to published results.

Simple Symbolic Datasets

The Laser dataset was obtained by quantizing activity changes of laser in chaotic regime, where relatively predictable subsequences are followed by hardly predictable events. The original real-valued time series was composed of 10000 differences between the successive activations of a real laser. The series was quantized into a symbolic sequence over four symbols corresponding to low and high positive/negative laser activity change. The first 8000 symbols are used as the training set and the remaining 2000 symbols form the test data set [8].

Deep Recursion data set is composed of strings of context-free language L_G. The set of production rules is composed of three simple rules: $R \rightarrow aRb | R \rightarrow ARB | R \rightarrow e$ where e is the empty string. This language is in [3] called palindrome language. The training and testing data sets consist of 1000 randomly generated concatenated strings. No end-of-string symbol was used. Shorter strings were more frequent in the training set than the longer ones. The total length of the training set was 6156 symbols and the length of the testing set was 6190 symbols.

Linguistic Datasets

For our experiments we have chosen two languages from recently published papers. The first language (labeled EG) was generated by Elman's grammar [9] and was also used in [10]. Dataset alphabet consists of 24 words including end-of-sentence marker. The second dataset (labeled CG) was generated using the grammar from [11]. Language was composed of 72 words including end-of-sentence marker. For both tasks the training and the test set were composed of 10000 words. Ground true probabilities were recorded during datasets creation.

Language entropy H can be estimated using generated samples from the probabilistic grammars:

$$\mathrm{H} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{a\in A} G_t(a)\log_{|A|} G_t(a), \tag{9}$$

where $G_t(a)$ is the ground true probability of generating symbol a in time step t and $|A|$ is the size of the language alphabet A. The longer the sample, the more accurate the entropy estimation, nevertheless we have used test sets to provide entropy estimations: 0.535 for EG dataset and 0.398 for CG dataset.

Method

First BP, BPTT and standard EKF with both quadratic error cost function and entropy cost function (prefix SM: SM-BP, SM-BPTT and SM-EKF-BPTT, where SM is an abbreviation for soft-max) were used to train Elman's SRN on Laser and Deep Recursion datasets. 10 training epochs for EKF and up to 100 epochs for BP and BPTT were done. Symbols were encoded using one-hot-encoding, i.e. all input or target activities were set to 0, except the one corresponding to given symbol, which was set to 1. Unipolar 0-1 sigmoidal activation function was used.

Different parameters for BP and BPTT were used for each data set to obtain the best results. We improved training by using scheduled learning rate. We used linearly decreasing learning rate in predefined intervals. But no improvements made the training as stable and fast as the EKF training (taking into account the number of epochs). For EKF training, error covariance matrix $\mathbf{P}$ was set to $\mathbf{P} = 1000 * \mathbf{I}$, the measurement noise covariance matrix $\mathbf{R}$ was set to $\mathbf{R} = 100 * \mathbf{I}$, and the process noise covariance matrix $\mathbf{Q}$ was set to $\mathbf{Q} = 0.0001 * \mathbf{I}$, where $\mathbf{I}$ stands for the identity matrix. These values were used throughout all experiments. Matrices $\mathbf{Q}$ and $\mathbf{R}$ remained fixed during training. Measurement matrix $\mathbf{H}$ was calculated in every time step using algorithm almost identical to BPTT with window size set to 10 [7].

Predictive performance was evaluated using a normalized negative log-likelihood (NNL) calculated over symbolic sequence $s_1 s_2 \ldots s_t \ldots s_T$ used for testing as

$$\mathrm{NNL} = -\frac{1}{T}\sum_{t=1}^{T} \log_{|A|} P_t(s_{t+1}) \approx -\frac{1}{T}\sum_{t=1}^{T}\sum_{a\in A} G_t(a) \log_{|A|} P_t(a), \qquad (10)$$

where the base of the logarithm $|A|$ is the number of symbols in the alphabet A and the $P_t(a)$ is the probability of predicting symbol a in the time step t. $\mathrm{NNL} = 0$ corresponds to the 100% next-symbol prediction.

We also trained SRN on linguistic dataset using SM-BP, SM-BPTT and SM-EKF-BPTT. 10 training epochs were performed for all algorithms. The same parameters as in the previous experiments were used for SM-EKF-BPTT. For SM-BP and SM-BPTT exponencially decaying learning rate was used. We also provide results for alternative performance criteria used by the cognitive science community, such as cosine scores defined as:

$$\mathrm{COS} = \frac{1}{T}\sum_{t=1}^{T} \cos\left(G_t(s_t), P_t(s_t)\right), \qquad (11)$$

where $\cos$ is the cosine between two (in our case normalized) vectors $\mathbf{x}$ and $\mathbf{y}$:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{|\mathbf{x}||\mathbf{y}|}. \qquad (12)$$

Another alternative measure is averaged Kullback-Leibler divergence defined as:

$$\mathrm{KLD} = \frac{1}{T}\sum_{t=1}^{T}\sum_{a\in A} G_t(a) \log_{|A|} \frac{G_t(a)}{P_t(a)}. \qquad (13)$$

There is a strong correspondence between measures KLD and NNL: $\mathrm{KLD} \approx \mathrm{NNL} - \mathrm{H}$, where H is the language entropy.

Results

For simple symbolic datasets we present mean and standard deviations of 10 simulations. Results for both Laser dataset and Deep Recursion are shown in Fig. 1. Unsatisfactory simulations with significantly low performance were thrown away for BP and BPTT algorithms, which seems to be sensitive to get stuck in local minima or to diverge. EKF approach to training RNNs on symbolic sequences shows higher robustness and better resulting performance. BP algorithm is too week to give satisfactory

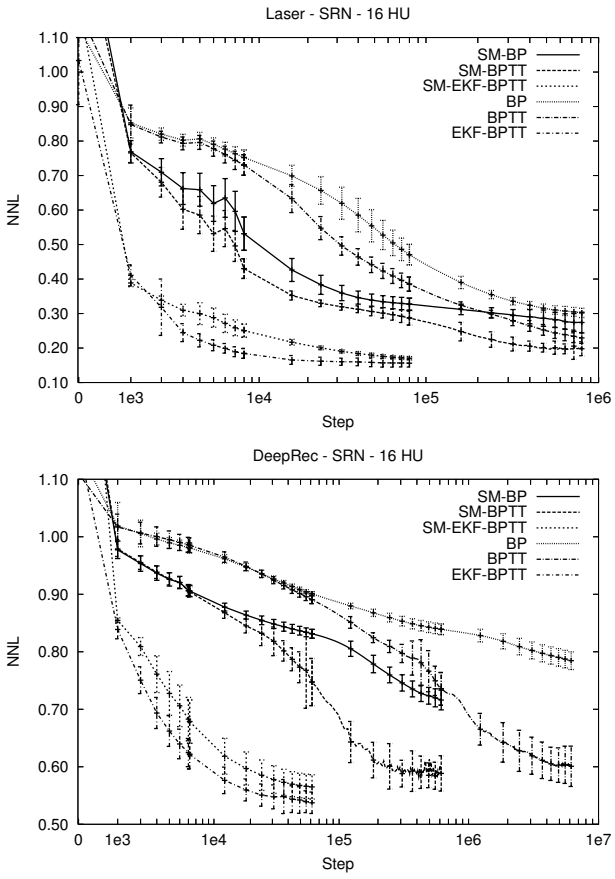

Fig. 1. Performance of Elman's SRN with 16 hidden units trained on Laser and DeepRec dataset.

results. NNL performances are significantly worse in comparing with algorithms that take into account the recurrent nature of network architectures. SMNNL version of BP and BPTT algorithms converge faster and finally better NNL error is usually achieved in comparing with standard BP pr BPTT. On the other hand SMNNL version of EKF shows slightly inferior NNL results in comparing with EKF-BPTT.

For linguistic datasets we provide results of SRN trained by the proposed approximation to the Kalman filtration - SM-EKF-BPTT. Computational requirements of standard EKF RNN training are very high because of high number of output units. Proposed approximation SM-EKF-BPTT helped us to obtain results in reasonable time.

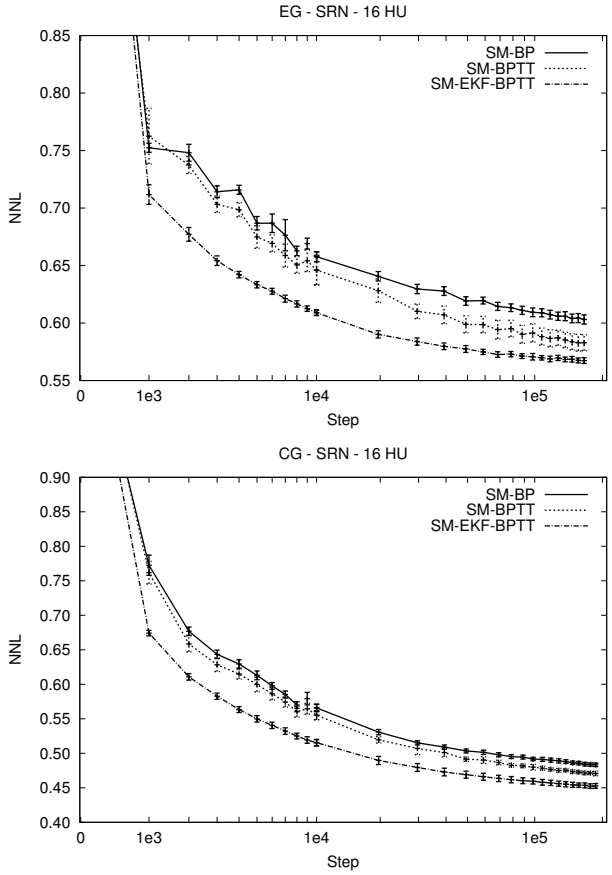

Fig. 2. Performance of Elman's SRN with 16 hidden units trained on EG dataset and CG dataset.

The Tab. 1 summarizes results obtained by training SRNs using SM-BP, SM-BPTT and SM-EKF-BPTT on linguistic datasets. We also provide results using variable length Markov models (VLMMs), since RNNs and VLMMs are similar [12]. Results reported here are not directly comparable to [10, 11], nevertheless we have also conducted experiments with modifications such as limiting maximal sequence length to 11 words for EG dataset, using 10 datasets and removing duplicated sentences from test sets for CG dataset and better results than reported were achieved using SM-EKF-BPTT algorithm.

	EG Dataset			CG Dataset		
	NNL	**KLD**	**COS**	**NNL**	**KLD**	**COS**
VLMM	0.63	0.097	0.91	0.51	0.114	0.86
SM-BP	0.60	0.070	0.93	0.48	0.084	0.90
SM-BPTT	0.58	0.046	0.96	0.47	0.072	0.92
SM-EKF-BPTT	0.56	0.031	0.96	0.45	0.053	0.93

Table 1. Summary of achieved results on linguistic datasets.

4 Conclusion

Significantly better results can be achieved by algorithms based on the Kalman filtration. Extended Kalman filter shows much faster convergence in terms of number of epochs and resulting performance is better. Results obtained with architectures with soft-max combining function and gradient-based algorithms minimizing the entropy proved to be better alternative than their standard counterparts based on quadratic error cost function. Faster convergence and better resulting performance was obtained. SM-EKF-BPTT did not achieve the same performance as standard EKF, the resulting NNL errors were slightly higher. On the other side the computational requirements of this algorithm are very favorable, comparable to the standard BPTT algorithm.

References

1. Elman, J.L.: Finding structure in time. Cognitive Science **14**(2) (1990) 179–211
2. Christiansen, M., Chater, N.: Toward a connectionist model of recursion in human linguistic performance. Cognitive Science **23** (1999) 417–437
3. Rodriguez, P.: Simple recurrent networks learn contex-free and contex-sensitive languages by counting. Neural Computation **13** (2001) 2093–2118
4. Bodén, M., Wiles, J.: On learning context free and context sensitive languages. IEEE Transactions on Neural Networks **13**(2) (2002) 491–493
5. Hochreiter, J., Schmidhuber, J.: Long short term memory. Neural Computation **9**(8) (1997) 1735–1780
6. Werbos, P.: Backpropagation through time; what it does and how to do it. Proceedings of the IEEE **78** (1990) 1550–1560
7. Prokhorov, D.V.: Kalman filter training of neural networks: Methodology and applications. Tutorial on IJCNN 2004, Budapest, Hungary (2004)
8. Tiňo, P., Dorffner, G.: Predicting the future of discrete sequences from fractal representations of the past. Machine Learning **45**(2) (2001) 187–218
9. Elman, J.: Distributed representations, simple recurrent networks, and grammatical structure. Machine Learning **7** (1991) 195–225
10. Tong, M.H., Bickett, A.D., Christiansen, E.M., Cottrell, G.W.: Learning grammatical structure with Echo State Networks. **20** (2007) 424–432
11. Farkaš, I., Crocker, M.: Recurrent networks and natural language: exploiting self-organization. In: Proceedings of the 28th Cognitive Science Conference, Vancouver, Canada. (2006) 1275–1280
12. Tiňo, P., Čerňanský, M., Beňušková, Ľ.: Markovian architectural bias of recurrent neural networks. IEEE Transactions on Neural Networks **15**(1) (2004) 6–15