ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Exploiting Similarity in System Identification Tasks with Recurrent Neural Networks

Sigurd Spieckermann[1,2], Siegmund Düll[1,3], Steffen Udluft[1],
Alexander Hentschel[1], Thomas Runkler[1,2]

1- Siemens Corporate Technology – Learning Systems
Otto-Hahn-Ring 6 – 81739 Munich, Germany

2- Technical University of Munich – Department of Informatics
Boltzmannstr. 3 – 85748 Garching, Germany

3- Berlin University of Technology – Machine Learning
Franklinstr. 28-29 – 10587 Berlin, Germany

Abstract. A new dual-task learning approach based on recurrent neural networks with factored tensor components for system identification tasks is presented. The overall goal is to identify the underlying dynamics of a system given few observations which are augmented by auxiliary data from similar systems. The resulting system identification is motivated by various real-world industrial use cases, e.g. gas or wind turbine modeling for optimization and monitoring. The problem is formalized and the effectiveness of the proposed method is assessed on the cart-pole benchmark.

1 Introduction

The dynamics of complex technical systems such as gas or wind turbines can be approximated by data driven models, e.g. recurrent neural networks [1]. Such methods have proven successful to be powerful alternatives to analytical models which are not always available or inaccurate [2]. Optimizing the parameters of these models often requires large amounts of operational data. However, data is a scarce resource in many applications, hence, data efficient procedures utilizing all available data are preferred. The following real-world scenario is one among many that motivated the research presented in this paper.

Consider an industrial plant that is subject to modifications over time. During normal operation, the system behavior is observed and a simulation model is trained from the collected data. In consequence of the modifications made to the plant, its dynamical properties change invalidating the available model. However, an accurate model is needed as soon as possible after recommissioning the plant. The fact that the overall plant remains the same, and thus no fundamental change of the general structure and complexity of the dynamics is expected, suggests to exploit information collected prior to modifications.

The contributions of this paper comprise the formal definition of the considered problem class (section 2), the presentation of a variety of methods, in particular the Factored Tensor Recurrent Neural Network, that allow to share information between similar dynamical systems (section 3) and the assessment of their effectiveness based on experiments with the frictionless cart-pole benchmark [3, 4] (section 4). The choice of using the cart-pole benchmark for the

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

experiments is due to its intuitive dynamics which eased the process of instantiating appropriate similar systems. Section 5 concludes the results and outlines future work.

2 Problem Definition

Let $I := \{1, 2\}$ denote a set of identifiers for fully observable deterministic similar dynamical systems, which are observed in fixed time intervals τ, defined by the tuple (S, A, f) with a state space S, an action space A, and an unknown state transition function $f\colon I \times S \times A \to S$ describing the temporal evolution of S.

A data set $D = \bigcup_{i \in I} D_i$ of size $|D|$, consisting of observations $(i, s, a, s') \in D_i$, is drawn i.i.d. from a probability distribution $\mathcal{D}$. Each observation contains information about system $i \in I$ of a single state transition from state $s \in S$ to state $s' \in S$ caused by the momentum and the effect of action $a \in A$.

Let $H \subseteq \{h \,|\, h\colon I \times S \times A \to S\}$ denote a hypothesis space, i.e. a set of functions that are assumed to approximate the state transition function f. Further, let $h^* \in H$ be the optimal hypothesis of f within H, i.e. the best approximation of f within the considered space of hypotheses. Let $\mathcal{L}\colon S \times S \to \mathbb{R}_{\geq 0}$ be an error metric between a predicted successor state $\hat{s}' = h(i, s, a)$ and the true successor state s'. The optimal hypothesis h^* minimizes the expected error $\varepsilon(h) := \mathbb{E}_{(i,s,a,s') \sim \mathcal{D}}[\mathcal{L}(h(i, s, a), s')]$ where $\mathbb{E}$ denotes the expectation operator, hence, $h^* = \arg\min_h \varepsilon(h)$. Since $\mathcal{D}$ is generally unknown, an approximately optimal hypothesis $\hat{h}$ is determined by minimizing the empirical error $\hat{\varepsilon}_D(h) := \frac{1}{|D|} \sum_{(i,s,a,s') \in D} \mathcal{L}(h(i, s, a), s')$ induced by a hypothesis h on a data set D.

Let h_i^* be the optimal hypothesis of the dynamics of system i within H. Given sufficient data D_1, $\hat{h}_1$ is close to h_1^*. In contrast, assuming the amount of data D_2 is insufficient, $\hat{h}_2$ is expected to differ significantly from h_2^*. The problem addressed in this paper is to develop and assess methods that yield a better hypothesis $\hat{h}_2$ by exploiting auxiliary information from D_1 given the considered systems are sufficiently similar.

3 System Identification with RNNs

In general, the dynamics of a fully observable deterministic dynamical system can be described by some function $s_{t+1} = f(s_t, a_t)$. However, in practice the learning process of this function often benefits from predicting the sequence of successor states $s_{t+1}, ..., s_{t+T}$ given a trajectory of actions $a_t, ..., a_{t+T-1}$ for $T \in \mathbb{N}_{\geq 2}$ time steps instead of predicting only a single step.

Let n_l denote the dimensionality of layer l in a neural network. Recurrent neural networks (RNNs) are powerful models for sequence modeling tasks. In contrast to feedforward neural networks, RNNs process their input vectors $x_1, ..., x_T$, $x_t \in \mathbb{R}^{n_x}$, sequentially along the time axis thereby taking its sequential structure directly into account. The input sequence is mapped to a hidden state sequence $h_1, ..., h_T$, $h_t \in \mathbb{R}^{n_h}$, from which the output sequence $\hat{y}_1, ..., \hat{y}_T$,

$y_t \in \mathbb{R}^{n_y}$, is computed. Notation is slightly abused by overloading the variable h to describe the hidden state of an RNN as well as a hypothesis. A simple RNN is defined for $t = 1, ..., T$ in the following recursive manner

$$h_0 = h_{\text{init}} \tag{1a}$$
$$h_t = \sigma_h(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{1b}$$
$$\hat{y}_t = \sigma_y(W_{yh}h_t + b_y) \tag{1c}$$

where $W_{vu} \in \mathbb{R}^{n_v \times n_u}$ is the weight matrix from layer u to layer v, $b_v \in \mathbb{R}^{n_v}$ is the bias vector of layer v and $\sigma(\cdot)$ is an elemenwise nonlinear function, e.g. $\tanh(\cdot)$.

3.1 RNN and Naïve RNN

In the context of modeling the dynamics of an open system, a recurrent neural network may be defined by the following equations.

$$h_1 = \sigma_h(W_{hs}s_1 + b_1) \tag{2a}$$
$$h_{t+1} = \sigma_h(W_{ha}a_t + W_{hh}h_t + b_h) \tag{2b}$$
$$\hat{s}_{t+1} = \sigma_s(W_{sh}h_{t+1} + b_s) \tag{2c}$$

The initial state s_1 is mapped into the hidden state space of the RNN by a linear transformation followed by the nonlinear function $\sigma_h(\cdot)$. The state space of a dynamical system is often real-valued and unbounded, hence, $\sigma_s(\cdot)$ becomes the identity function.

The most naïve approach to exploit information from one dynamical system and share it with another one is to mix the data of both systems and train a joint model. This way, the model is forced to generalize over the different properties of the systems. However, since the training examples of both systems are not distinguished, the model can only learn the average dynamics which may be vastly suboptimal for rather different systems.

3.2 RNN+ID

One way to incorporate information that allows the model to distinguish between the systems is to tag each training example with an identifier $i \in I$ corresponding to the system which generated the data. This tag is provided as an extra input z to the model at each time step encoded as the i-th Euclidean basis vector e_i with $\dim(e_i) = |I|$, hence, $z \in \{e_1, ..., e_{|I|}\}$. The resulting RNN is described by the following equations.

$$h_1 = \sigma_h(W_{hs}s_1 + b_1) \tag{3a}$$
$$h_{t+1} = \sigma_h(W_{ha}a_t + W_{hh}h_t + W_{hz}z + b_h) \tag{3b}$$
$$\hat{s}_{t+1} = W_{sh}h_{t+1} + b_s \tag{3c}$$

In fact, computing $W_{hz}z$ is equivalent to selecting the i-th column of W_{hz} so the cart-pole identifier input introduces a separate bias for each cart-pole. The shared bias b_h and the specific biases accessed through $W_{hz}z$ can be combined.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

3.3 Factored Tensor RNN (FTRNN)

The Factored Tensor Recurrent Neural Network (FTRNN) is a modification of the Tensor Recurrent Neural Network (TRNN) as denoted in [5, 6]. The TRNN implies the idea of replacing some or all weight matrices by tensors where each slice of a tensor is associated with a particular system. Since using full tensors introduces many additional parameters and allows for merely limited or no information sharing, a factored tensor of the form

$$W_{vuz} z \approx W_{vf} \operatorname{diag}(W_{fz} z) W_{fu} \tag{4}$$

with $W_{vuz} \in \mathbb{R}^{n_v \times n_u \times |I|}$, $W_{fu} \in \mathbb{R}^{n_f \times n_u}$, $W_{vf} \in \mathbb{R}^{n_v \times n_f}$, and $W_{fz} \in \mathbb{R}^{n_f \times |I|}$ was suggested. Thus, the FTRNN is defined by the equations

$$\begin{aligned} h_1 &= \sigma_h(W_{hs} s_1 + b_1) && \text{(5a)} \\ h_{t+1} &= \sigma_h(\, W_{hf_a} \operatorname{diag}(W_{f_a z} z) W_{f_a a} a_t + && \text{(5b)} \\ &\qquad W_{hf_h} \operatorname{diag}(W_{f_h z} z) W_{f_h h} h_t + b_h) && \\ \hat{s}_{t+1} &= W_{sh} h_{t+1} + b_s. && \text{(5c)} \end{aligned}$$

Only relatively few parameters are specific to each system forcing the model to disentangle properties of the dynamics that are common to all systems from those that are distinct. In some sense, every system has its own set of functions describing the contributions of the previous hidden state and the current external force to yield the current hidden state. However, these functions are not independent for each systems since they are composed of two shared components and one distinct component.

4 Experiments

The evaluation of the methods described in section 3 was performed on the frictionless cart-pole simulation [4] observed every $\tau = 0.02\,\mathrm{s}$. Contrary to settings used in common reinforcement learning tasks, no constraints were enforced on the cart position or the pole angle. In order to avoid discontinuities in the representation of the pole angle when the pole swings beyond 180°, its decomposition into the sine and cosine components was used. Two cart-pole instances (CP1 and CP2) were configured to have the pole lengths 0.5 and 1.0, and the pole masses 0.05 and 0.1, respectively. The data sets D_1 and D_2 were obtained by observing the state transitions (i, s, a, s') along a trajectory of 1000 actions drawn i.i.d. from the uniform distribution $\mathcal{U}(-1, 1)$. After every 1000 steps, the simulation was reset to its initial state. D_1 consisted of 15 000 examples $(i = 1, s_1, a_1, ..., s_T, a_T, s_{T+1})$ and was split into a training set $D_{\mathrm{T},1}$ containing 10 000 examples and a validation set $D_{\mathrm{V},1}$ sized 5000. The data set D_2 was obtained the same way, however, training and validation set sizes were reduced and upsampled to their original sizes in the various experiments. The loss function for training was chosen to be the mean squared error (MSE) between the predicted and the true successor state sequence, i.e. $\mathcal{L}(\hat{s}_2, ..., \hat{s}_{T+1}, s_2, ..., s_{T+1}) = \frac{1}{2}\sum_{t=1}^{T} \|s_{t+1} - \hat{s}_{t+1}\|_2^2$.

All experiments were implemented using Theano [7]. The parameters of the models were optimized using Hessian-Free optimization with structural damping [8]. The maximum number of parameter updates was set to 10 000, the number of CG iterations per update was limited to 150. Due to the relatively small amount of overall training data, the gradient was computed using the full training set and the curvature was estimated using 5000 examples drawn i.i.d. from the training set. An early stopping procedure with a patience time of 100 updates was used to reduce overfitting.

Figure 1 depicts the median performance of five runs with $T = 10$, i.e. the MSE per state component per time step yielded by the different models on a representative generalization set $D_{\mathrm{G},2}$ sized 30 000. In each run, the initial model parameters were sampled at random. For all networks, the hidden state dimension was set to $n_h = 10$. The FTRNN used $n_{f_h} = n_h$ and $n_{f_a} = 2$. The simple RNN model was only trained on data from CP2. The other models, i.e. the Naïve RNN, RNN+ID and FTRNN, were trained on the combined data set $D_1 \cup D_2$. The experimental results show that the simple RNN performed well for 10 000 training examples but degraded rapidly for smaller data set sizes. Surprisingly, a considerable improvement could be achieved by the Naïve RNN, which simply combines the data from both cart-poles without any awareness of the two systems, even for small amounts of data from CP2. The RNN+ID model yielded an expected improvement over the Naïve RNN since it was provided information to actually distinguish the two cart-poles and hence was able to adjust to the different dynamical properties. However, as shown in section 3.2, the system identifier merely acts as a specific bias in the hidden state of the RNN being limited in its ability to properly account for the similar but non-identical state transition function of the two cart-poles. For a training data ratio of $10\,000 : 625 \cdot 16$, the RNN+ID was inferior to the Naïve RNN which might be a result of overfitting the bias parameters. The FTRNN outperformed the Naïve RNN and the RNN+ID models for larger amounts of examples from CP2, but struggled with significantly reduced amounts of data thereof. Again, this might have been caused by overfitting the parameters specific to CP2. While the FTRNN approach seems conceptually promising, further research is required to investigate this issue more closely.

5 Conclusion

The problem class of modeling a set of dynamical systems by sharing information among the systems was introduced, motivated and formalized. The Factored Tensor Recurrent Neural Network (FTRNN) architecture was presented and its effectiveness and potential was assessed in a series of experiments using the cart-pole simulation. The FTRNN was compared to three variants of recurrent neural networks using different approaches of information sharing. It showed superior performance compared to the Naïve methods for reasonably unbalanced data

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

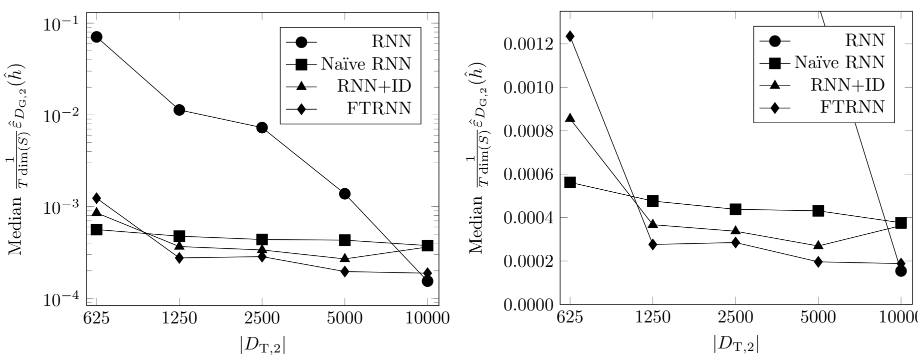

Fig. 1: Median error of five training runs per model. The error is the MSE per state component per time step between the predicted and the true successor state sequence plotted against the varying training set sizes of CP2 for a fixed number of $|D_{T,1}| = 10\,000$ training examples of CP1.

distributions but degraded for significant imbalances which might be attributed to overfitting.

Current and future work includes a more detailed analysis of the degradation of the FTRNN for extreme data imbalances, other simulations than the cart-pole, information sharing among more than two systems, the effect of noisy observation functions and adapting the method for partially observable systems.

References

[1] Coryn A. L. Bailer-Jones, David J. C. MacKay, and Philip J. Withers. A recurrent neural network for modelling dynamical systems. *Network: Computation in Neural Systems*, 9(4):531–547, 1998.

[2] Anton M. Schäfer, Daniel Schneegass, Volkmar Sterzing, and Steffen Udluft. A neural reinforcement learning approach to gas turbine control. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1691–1696, 2007.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*, volume 1. Cambridge University Press, 1998.

[4] Razvan V. Florian. Correct equations for the dynamics of the cart-pole system. *Center for Cognitive and Neural Studies (Coneural), Romania*, 2007.

[5] Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1017–1024, 2011.

[6] Graham W. Taylor and Geoffrey E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th International Conference on Machine Learning*, pages 1025–1032. ACM, 2009.

[7] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[8] James Martens and Ilya Sutskever. Training deep and recurrent networks with Hessian-Free optimization. In *Neural Networks: Tricks of the Trade*, pages 479–535. Springer, 2012.