I. Troch, F. Breitenecker, eds. ISBN 978-3-901608-35-3

LEARNING ORDER INVARIANT AGGREGATION FUNCTIONS WITH RECURRENT NEURAL NETWORKS

W. Heidl[1], C. Eitzinger[1], M. Gyimesi[2]
[1]Profactor GmbH, Steyr, Austria; [2]Vienna University of Technology, Vienna, Austria

Corresponding Author: W. Heidl, Profactor Gmbh
Im Stadtgut A2, 4407 Steyr-Gleink, Austria; wolfgang.heidl@profactor.at

Abstract. Numerous applications benefit from parts-based representations resulting in sets of feature vectors. To apply standard machine learning methods, these sets of varying cardinality need to be aggregated into a single fixed-length vector. Since, per-se, no order is imposed on the set elements the aggregation function has to be order invariant. We have evaluated three common recurrent neural network (RNN) architectures, Elman, Williams & Zipser and Long Short Term Memory networks, on approximating eight aggregation functions of varying complexity. The most difficult to approximate function computes the minimum distance (`MinDist`) between any two elements when interpreted as points in Euclidean space. On this function only the LSTM network can be trained to perform order invariant. However, further investigation into the state-space behaviour reveals that the LSTM network is actually only approximating a function dependent on the element count of the sets. Due to the distribution of the test data this simplification still yields 40% correlation of the network's output with the target value. This behaviour has been verified by using a modified version of `MinDist` where any correlation between target value and input length has been removed.

1 Introduction

Numerous applications benefit from parts-based representations resulting in sets of feature vectors. As an example, the good/bad decision in surface inspection tasks often depends on the whole set of extracted fault features, such as their spatial distribution, their total area and similar quantities. This information cannot be utilized if faults are processed one-by-one. Learning the decision process in this context thus requires methods for classification of sets of feature vectors extracted from the faults [8].

To apply standard machine learning methods, these sets of varying cardinality need to be aggregated into a single fixed-length vector. We define the classification task over sets $X_i \coloneqq \{x_{i,1}, x_{i,2}, \dots, x_{i,n_i}\}$ of feature vectors $x_{i,j} \in \mathbb{R}^m$

$$f \circ g \colon \mathbb{R}^{m \cdot n_i} \mapsto \{-1,1\} \tag{1}$$

$$X_i \mapsto c_i \coloneqq (f \circ g)(X_i) \tag{2}$$

as the composition of the aggregation function $g \colon \mathbb{R}^{m \cdot n_i} \mapsto \mathbb{R}^k, X_i \mapsto a_i$ computing the fixed length k-aggregate a_i and the classification function $f \colon \mathbb{R}^k \mapsto \{-1,1\}, a_i \mapsto c_i$. Since, per-se, no order is imposed on the set elements, the aggregation function has to be order invariant.

Related Work. Some work has been done on classification of structures containing an unknown number of elements [2,5], where the authors propose to use *generalized recursive neurons* to represent the structure of a graph. In order to achieve this, they make use of the linked structured of the graph and directly represent the graph topology in the network. The problem is similar to ours in that the number of input elements can vary, however their approach uses the structure present in the input to deduce the order in which to present the elements to the network. Recently kernel functions have been proposed operating on sets of features [3]. These approaches either assume that direct correspondences exist between the elements of sets or require sets of equal cardinality.

We investigate the capability of common RNN architectures to learn order invariant aggregation functions relevant the context of surface inspection. Section 2 describes the architectures and special mode of operation of RNNs. Section 3 presents the experimental setup and detailed results on one of the aggregation problems. Concluding remarks are given in Section 4.

2 Recurrent Neural Networks for Feature Aggregation

Due to their recurrent structure, RNNs can deal with a variable number of input vectors without growing with the number of objects that we want to process. Furthermore aggregation and succeeding processing steps such as classification can be integrated into a homogenous structure, which allows for training of both components in a uniform manner. The principle structure is shown in Figure 1.

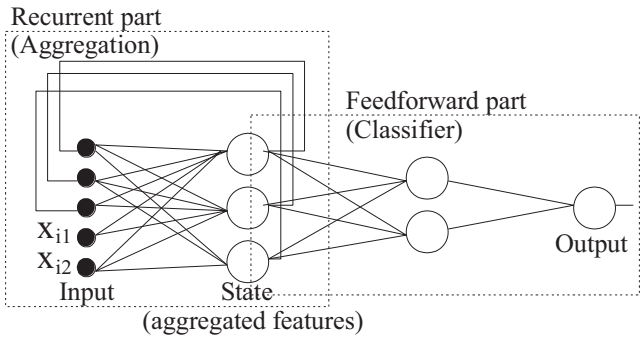

Figure 1: Recurrent Neural Network architecture for classification of sets of feature vectors.

The n input vectors $x_1, x_2, \ldots, x_n$ will be processed by the network as if they were a time sequence. The classification result will be available at the output after n "time steps". However, this sequence of vectors lacks an important property of time series, as it has no inherent ordering. Order invariance of RNNs is realized by reshuffling of the data during training: The sequence of the feature vectors in the set will be randomly permuted after each training epoch of the network. The neural network should thus learn to be order-invariant. Another option is sorting as a pre-processing step that establishes order invariance among the set elements. For dimensions greater than 1 sorting is non-trivial and requires problem-specific order relations (e.g. [4]).

2.1 Network Architectures

We consider three common RNN architectures for learning aggregation functions $g: \mathbb{R}^{m \cdot n_i} \mapsto \mathbb{R}^k$ producing fixed length aggregates a_i out of sets of varying cardinality X_i. The Elman network [1] and the architecture proposed by Williams & Zipser (WZ) [9] date back to the late 1980s and introduce feedback loops to the then well known Multilayer Perceptron (MLP) feed forward neural networks. While Elman limited the feedback to within the hidden layer of the network, Williams & Zipser added feedback loops within the output layer, from the output to input layer and feedforward connections bypassing the hidden layer to their so-called fully connected network.

Long short-term memory (LSTM) networks have been introduced by Hochreiter [6] as a solution to the *vanishing gradient* problem. Standard RNNs can hardly learn long-term dependencies between the data, since the error gradient with respect to the weights vanishes for larger time-lags between output and input. The error signal used to adapt the weights during training thus decreases exponentially when going back through time. Long short-term memory (LSTM) networks solve this problem, by incorporating *constant error carrousels* (CEC), neurons that propagate the error unchanged, thus avoiding the decaying gradient. Access to the CEC is controlled by multipliers which are driven by dedicated neurons called input and output gates. The inputs of the gates are connected the network inputs and the outputs of all cells in the network.

3 Experiments

We have evaluated Elman, WZ and LSTM RNN architectures, on approximating eight aggregation functions g of varying complexity (Table 1). The data sets have been generated by applying aggregation functions to sets of scalar entries $x_{i,j} \in \mathbb{R}$. We have trained each network on both sorted and shuffled representations of the data sets. The goal is to establish baseline results showing whether these existing RNNs can in principle learn order invariant aggregation functions.

3.1 Data Set

The aggregation functions used for the experiments have been designed with visual quality inspection in mind. For example, the `Sum` function could describe the total area of detected faults. Another example is the minimum geometric distance between faults (`MinDist`), which is an important measure to decide if local build-ups of otherwise uncritical faults can be tolerated.

$\texttt{Max} := \max_j x_j$	$\texttt{MaxAbs} := \max_j \lvert x_j \rvert$
$\texttt{Norm} := \sqrt{\sum_j {x_j}^2}$	$\texttt{MeanNorm} := \frac{1}{n_i}\sqrt{\sum_j {x_j}^2}$
$\texttt{SumMax2} := x_u + x_v,$ $x_u \geq x_v \geq x_j, j \in \{1, \ldots, n\} \backslash \{u, v\}$	$\texttt{SumMax2Abs} := \lvert x_u \rvert + \lvert x_v \rvert$ $\lvert x_u \rvert \geq \lvert x_v \rvert \geq \lvert x_j \rvert, j \in \{1, \ldots, n\} \backslash \{u, v\}$
$\texttt{Sum} := \sum_j x_j$	$\texttt{MinDist} := \min_{j,k,j \neq k} \lvert x_j - x_k \rvert$

Table 1: Definition of the aggregation functions underlying the data sets used in the experiments.

Data sets have been produced by applying aggregation functions to sets of scalar entries $x_{i,j} \in \mathbb{R}$. The number of elements in each set is uniformly distributed with a minimum of 15 and a maximum of 30 elements. The range has been chosen such that the networks cannot memorize all set elements. The entries itself are also generated randomly, $x_{i,j} \sim \mathrm{U}(-0.5, 0.5)$. Target values for the data sets are generated by scaling

$$a_i = \frac{g(X_i) - \mu}{6\sigma} + 0.5 \tag{3}$$

with $\mu = \frac{1}{n}\sum_{i=1}^{n} g(X_i)$ and $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(g(X_i) - \mu)^2}$, such that almost all values are in the range [0,1], compatible with the logistic sigmoid activation function at the network's output.

3.2 Training

During training the set elements are fed into the networks as a sequence, target values are presented to the network with the last set element only. Adaptation of the network weights has been performed using Backpropagation through time (BPTT) algorithm [1] for the Elman and Williams & Zipser network. The central idea is to unfold the recurrent neural network into a multilayer feedforward neural network. The feedforward network thus has a layer for each time step in the sequence. Then the standard Backpropagation algorithm is applied to the unfolded networks. The LSTM networks are trained by a combination of truncated BPTT and Real-time Recurrent Learning (RTRL) [9], an algorithm which is specifically tailored to the LSTM architecture [6].

The weights and biases have been randomly initialized in the range of $[-0.5, 0.5]$ for the Elman and WZ networks and $[-0.2, 0.2]$ for the LSTM networks, except for the biases to the input and output gates. These biases are initialized starting with -0.5 for the first, -1.0 for the second, -1.5 for the third and so forth. This way, cell activations are initially close to zero, and during training the cells will be sequentially activated to contribute to the network function [6].

3.3 Results

We have trained each network using both sorted and shuffled representations of the data sets. The accuracy of the approximation is assessed by the Pearson correlation coefficient R of the network's output with the target value on fresh test data not used during training. For each combination of aggregation function and network architecture we have performed 10 independent training runs to reduce the probability of reporting results where training converged to a bad local minimum of the cost function.

The LSTM network is inferior to the other network types for most data sets except for the most difficult-to-approximate `MinDist` aggregation function. On `MinDist` the Elman and WZ network's correlations deteriorate to below 10% on shuffled data, unlike the LSTM network's 40%, which is equal to the easier-to-handle sorted case. Although correlation values below 40% are too low for practical applications, the performance of the LSTM on the shuffled `MinDist` data is significantly better than random.

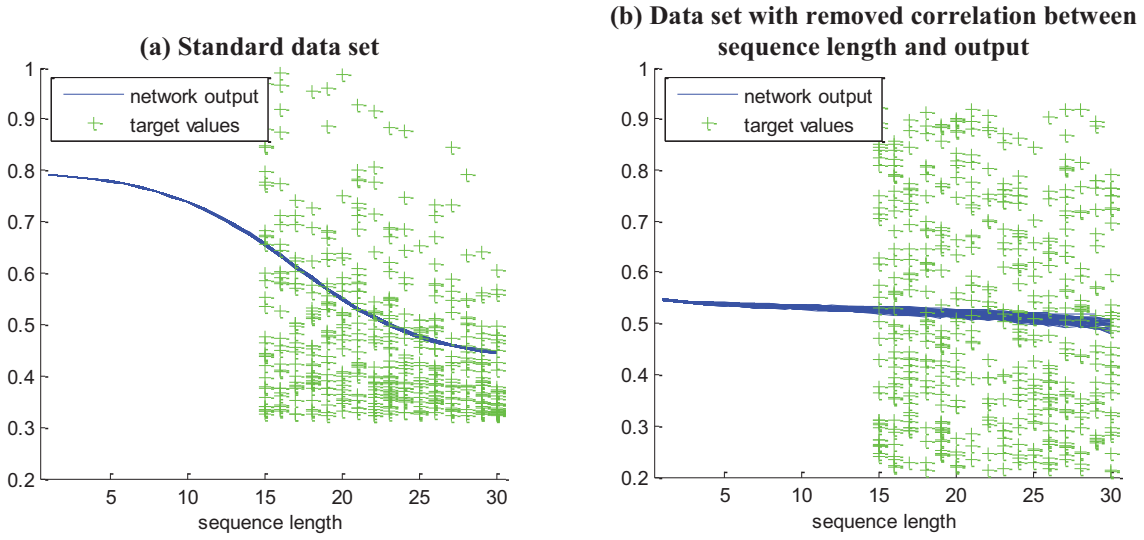

Figure 2: LSTM network output vs. sequence length. The LSTM network is approximating a function dependent on the sequence length only. (a) Due to the distribution of the data set this simplification still yields a 40% correlation between network output and target value. (b) When correlations between sequence length and target value are removed from the data, the accuracy of the network drops to below 10%.

However, after further investigation into the state-space behaviour, we have found that the LSTM network is actually approximating a function dependent only on the element count of the sets. Due to the distribution of the test data this simplification still yields 40% correlation of the network's output with the target value. The `MinDist` data set as described in Section 3.1 shows significant correlation between target value and the cardinality of the sets $|X_i|$ (Figure 2a). Indeed, with increasing number of identically distributed set elements, the average minimum distance between the elements decreases.

To remove this correlation we have implemented a sampling scheme, equalizing the distribution of the target values. Out of the original data set the central 60% quantile of the target values has been determined as target range. Then for each distinct set cardinality $|X_i| \in [15,30]$ an equal number of samples within that range has been generated by the original procedure but rejecting samples falling outside. The resulting target values have then been rescaled according to Equation (3). Results for the modified data set verify the previously observed behaviour of the LSTM network (Figure 2b), which the approximation accuracy now dropping below 10%.

4 Conclusions

Initial experiments indicated that LSTM networks are superior to earlier RNN architectures on the most difficult-to-approximate `MinDist` data set, reaching equal performance for both sorted and shuffled representations of the data. This was interpreted as a special capability of LSTM networks to learn order invariant functions. However, further experiments revealed that the target value distribution in the data set allowed for the network to approximate a function dependent on the number of set elements only and still produce significant accuracy.

None of the tested networks can learn to approximate an equalized version of the `MinDist`. Additional experiments will have to be carried out to check results unbiased by the element count for the remaining aggregation functions. We assume that successful learning of many of these functions will require changes in the high level architecture of RNNs:

Achieving order invariance is a major challenge in using recurrent neural networks for approximating aggregation functions. While the latter requires application-specific knowledge for multi-dimensional data, training on shuffled data creates an unnecessarily large input space which complicates learning. We will therefore consider networks where order-invariance is an inherent property of the structure. For the basic building blocks of such networks we will consider symmetric functions [7], which are defined as being invariant under permutations of their arguments. We assume that by using these basic building blocks, an order-invariant neural network could be created.

5 Acknowledgements

This work has been supported by Austrian FWF grant P19376-N13.

6 References

[1] J. L. Elman, "Finding structure in time," Cognitive Science, vol. 14, iss. 2, pp. 179–211, 1990.

[2] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," IEEE Transactions on Neural Networks, vol. 9, iss. 5, pp. 768–786, 1998.

[3] K. Grauman and T. Darrell, "The Pyramid Match Kernel: Efficient Learning with Sets of Features," J. Mach. Learn. Res. vol. 8, pp. 725–760, 2007.

[4] A. Graves, S. Fernandez, and J. Schmidhuber, "Multi-Dimensional Recurrent Neural Networks," in Proc. Artificial Neural Networks ICANN, 2007, pp. 549–558.Casti, J. L.: *Reality Rules – Picturing the World in Mathematics: I, II*. Wiley, New York, 1992.

[5] B. Hammer, "Recurrent networks for structured data - A unifying approach and its properties," Cognitive Systems Research, vol. 3, iss. 2, pp. 145–165, 2002.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. vol. 9, iss. 9, pp. 1735–1780, 1997.

[7] I. G. Macdonald, "Symmetric Functions and Hall Polynomials (Oxford Mathematical Monographs)," 2nd ed., Oxford University Press, USA, 1999.

[8] M. Tahir, J. Smith, and P. Caleb-Solly, "A Novel Feature Selection Based Semi-supervised Method for Image Classification," in Proc. Computer Vision Systems - ICVS, 2008, pp. 484–493.

[9] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," Neural Computation, vol. 1, iss. 2, pp. 270–280, 1989.

[10] L. Wolf and A. Shashua, "Learning over sets using kernel principal angles," J. Mach. Learn. Res. vol. 4, pp. 913–931, 2003.