ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

An Algorithmic Approach to Establish a Lower Bound for the Size of Semiring Neural Networks

Martin Böhm[1] and Thomas Schmid[2,3]

1- Universität Leipzig - Algebraische und logische Grundlagen der Informatik
Augustusplatz 10, Leipzig - Germany

2- Universität Leipzig - Machine Learning Group
Augustusplatz 10, Leipzig - Germany

3- Lancaster University in Leipzig
Nikolaistrasse 10, Leipzig - Germany

Abstract. Semiring neural networks have been introduced as a recurrent neural network-type representation of weighted automata with the potential to learn a recognizable series. Whether a given semiring neural network actually can or cannot compute a recognizable series, however, depends on the size of the network. Therefore, it is desirable to determine whether a proposed size is too small before initiation of the training procedure. Here, we present an algorithm that achieves this in polynomial time. As there is a one-to-one correspondence between semiring neural networks and weighted automata, our algorithm can also be used to derive lower bounds for the size of a recognizing automaton. Our algorithm complements previous work in this area as it works over commutative zero-sum-free semirings.

1 Motivation

Recurrent neural networks (RNNs) and weighted finite automata (WFAs) are two distinct concepts in computer science that are linked by semiring neural networks (SNNs) as introduced in [1]. While RNNs are usually considered over the field of reals and trained via backpropagation through time [2], gradient-based training algorithms fail for SNNs when instantiated over a semiring that cannot be embedded in the reals, e.g. a tropical semiring. Particle-swarm optimization, another popular training method for RNNs and proposed e.g. in [3], also fails as the required update equations cannot be easily solved when the semiring lacks inverse elements. Rank-based genetic algorithms such as Genitor II [4] are more promising in a semiring milieu and their principal use in RNN training has already been confirmed [5], yet convergence to an optimal solution may be slow [6]. For performance reasons as well as to facilitate analysis of the resulting automaton, one seeks to keep the size of the SNN/WFA as small as possible, yet large enough to achieve the task. To this end, we present a polynomial time algorithm that checks if the proposed SNN size is too small for training to succeed. It can also be used to estimate lower bounds for the size of a WFA to recognize a given series, a common problem studied in automata theory, cf. [7] and [8].

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

2 Theoretical Prerequisites

2.1 Commutative, Zero-Sum-Free Semirings

Given a finite **alphabet** Σ, the **free monoid** Σ^* consists of all finite words over Σ with concatenation as monoid operation and the empty word, ϵ, as neutral element. We set $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. The length of a word w is denoted by $|w|$ and for $a \in \Sigma$, $|w|_a$ indicates the number of letters "a" in w. A **commmutative semiring** $\mathbb{S} = (S, \bigoplus, \bigodot, 0, 1)$ is an algebraic structure where $(S, \bigoplus, 0)$ ("addition") and $(S, \bigodot, 1)$ ("multiplication") constitute Abelian monoids with neutral element 0 and 1, respectively, such that multiplication distributes over addition from the left and from the right, and 0 is absorbing for $\bigodot$. Moreover, $\mathbb{S}$ is **zero-sum-free** if $x \bigoplus = 0$ implies $x = y = 0$ for all $x, y \in S$. Examples of commutative zero-sum-free semiring include the set $\mathbb{N}$ equipped with the usual arithmetic as well as the tropical semirings where addition corresponds to taking the maximum (or minimum) and multiplication is the ordinary addition, e.g. $(\mathbb{N} \cup \{-\infty\}, max, +, -\infty, 0)$.

2.2 Semiring Neural Networks

Introduced in [1], SNNs are a special kind of RNN, expressively equivalent to WFAs, where the weights are taken from a commutative semiring $\mathbb{S} = (S, \bigoplus, \bigodot, 0, 1)$. A SNN consists of four layers: input layer, gate layer, state layer, and output layer. Intuitively, the nodes in the state layer correspond to the states of an equivalent WFA; the number of nodes in this layer is called the **size**.

Figure 1 shows a schematic view of a SNN of size 2. Input and gate layer as well as state and output layer are fully connected in a feed forward fashion while there is limited forward and recurrent connectivity between gate and state layer. The precise topology and function of the various node types of a SNN is described in [1].

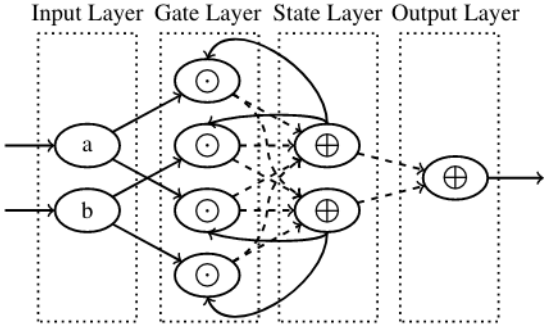

Fig. 1: Schematic view of a SNN of size 2 over $\Sigma = \{a, b\}$. Solid lines indicate weights fixed to 1 while dashed ones are to be trained.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

3 Determining the Network Size

3.1 Polynomial Equations

Due to their close relationship to WFAs, SNNs are fed recognizable series. Whether training is eventually successful therefore depends on the size n of the SNN - if it fails to compute the series, this is due to the number of nodes in the state layer having been chosen too small. It is thus crucial to determine this parameter n. Once n is known, the topology is fully specified as there are $|\Sigma| \cdot n$ nodes in the gate and exactly one node in the output layer.

In the following, let $f : \Sigma^* \to \mathbb{S}$ be a recognisable series over a commutative, zero-sum-free semiring $\mathbb{S}$, $\mathcal{N}$ the SNN to compute f, $Q = \{1, \dots, n\}$ the state layer and $G = \{(a_i, j) \mid a_i \in \Sigma, j \in Q\}$ the gate layer. By $W(a_i, j, k)$ we denote the weight of the connection from node (a_i, j) to node k. The value of node i in the state layer after processing word $v \in \Sigma^*$ is denoted by $q(i, v)$.

We consider the system of polynomial equations of degree at most $|w|$ induced by $\mathcal{N}$.

Observation 1. *The following equations hold for all $a \in \Sigma, v = a_1 \dots a_k \in \Sigma^*$, and $i \in Q$:*

$$\begin{aligned} W(a, 1, n) &= f(a), \\ q(i, a) &= W(a, 1, i), \\ q(n, v) &= f(v), \\ q(i, a_1 \dots a_k) &= \sum_{j=1}^{n} W(a_k, j, i) \cdot q(j, a_1 \dots a_{k-1}), \\ f(a_1 \dots a_k) &= \sum_{i=1}^{n} W(a_k, i, n) \cdot q(i, a_1 \dots a_{k-1}) \end{aligned} \tag{1}$$

Finding the minimal size now reduces to checking the system of equations for solvability as clearly as long as the equations yield a contradiction, the proposed size is not large enough.

3.2 Construction of the Computation Tree

To algorithmically check whether a contradiction in equations of type 1 occurs, we consider the **computation tree** of a word $w \in \Sigma \cdot \Sigma^+$, which is constructed as follows. The root is labelled with w while each other node carries as label two integers j, i. The tree has height $|w| + 1$ and every node at height $h \leq |w| - 2$ has exactly n children; the nodes at height $h = |w| - 1$ each have only one leaf as child node. The node labelled (j, i) at height h represents weight (a_h, j, i), and multiplying the weights represented by the nodes alongside a path from root to leaf in the tree corresponds to computing a summand in Equation 1. Observe that computation trees of different words of the same length differ only in the root as the represented symbol is implicit in a node's height.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Preparation. We define the **support** of a series as $supp(f) = \{w \in \Sigma^* \mid f(w) \neq 0\}$. The following preparatory steps have to be executed just once for a given series. The output from step 1.1, an array of words, can then be treated as a constant for recurring runs of the actual algorithm.

1.1 Find all $w \in \Sigma^+$ of minimal length such that $w \in supp(f)$.

1.2 For all $a \in \Sigma$ such that $f(a) = 0$, set $W(a, 1, n) = 0$.

Algorithm. On input n, the algorithm now proceeds as follows:

2.1 For each $u \in \Sigma^+ \setminus supp(f)$ with $|u| \leq |w|$ (with w from step 1.1), inspect the computation tree of u: every path in the computation tree of u must contain at least one non-empty set of weights **forced** to 0.

2.2 For each w from step 1.1, prune the computation tree of w by deleting each path containing a set of weights forced to 0 as determined in the previous step. If all paths have been deleted, a contradiction has been found and the size is too small.

For $f \neq 0$ (otherwise, the recognizing automaton is trivial), a w as in step 1.1 always exists. Since Σ^* is countable, all qualifying words in step 1.1 can be found by mere enumeration. Our algorithm uses (in step 2.1 above) the fact that the underlying semiring is zero-sum-free and that $1 \neq 0$; thus, when a sum of products evaluates to zero, we may conclude that each summand must have a factor that vanishes. The correctness is shown in the following theorem.

Theorem 1. *Let $(\mathbb{S}, +, \cdot, 0, 1)$ be a commutative, zero-sum-free semiring, let $f : \Sigma^* \to \mathbb{S}$ denote a series, and let $\mathcal{N}$ be a $\mathbb{S}$-neural network of size n. If the computation tree algorithm on input n detects a word $u \in \Sigma^+ \setminus supp(f)$ with $|u| \leq |u_0|$ where $u_0 \in \Sigma^+\Sigma^+$ is a word of minimal length such that $u_0 \in supp(f)$ then $\mathcal{N}$ cannot compute f, that is, there is a word $u_1 \in \Sigma^*$ such that $f(u_1) \neq (\|\mathcal{N}\|, u_1)$.*

Proof. We show that a contradiction inevitably arises in this situation. Let $u_0 = a_1 \dots a_k \in \Sigma^+\Sigma^+$ be minimal with the property $f(u_0) \neq 0$. Since

$$\begin{aligned} 0 \neq f(u_0) &= \sum_{i=1}^{n} W(a_k, i, n) \cdot q(i, a_1 \dots a_{k-1}) \\ &= \sum_{i=1}^{n} W(a_k, i, n) \cdot \sum_{j=1}^{n} W(a_{k-1}, j, i) \cdot q(j, a_1 \dots a_{k-2}), \end{aligned}$$

it suffices to show that for $i \in \{1, \dots, n\}$

$$\sum_{j=1}^{n} W(a_{k-1}, j, i) \cdot q(j, a_1 \dots a_{k-2}) = 0. \tag{2}$$

We prove Equation (2) by induction on n.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

For n = 1, the statement is clear since

$$\begin{aligned}\sum_{j=1}^{1} W(a_{k-1}, j, i) \cdot q(j, a_1 \dots a_{k-2}) &= W(a_{k-1}, 1, i) \cdot q(1, a_1 \dots a_{k-2}) \\ &= W(a_{k-1}, 1, i) \cdot f(a_1 \dots a_{k-2}) \\ &= 0\end{aligned}$$

where $f(a_1 \dots a_{k-2}) = 0$ since $|a_1 \dots a_{k-2}| < |u_0|$ and u_0 was minimal with $f(u_0) \neq 0$.
For the inductive hypothesis, assume that (2) holds for a fixed n. Then:

$$\begin{aligned}&\sum_{j=1}^{n+1} W(a_{k-1}, j, i) \cdot q(j, a_1 \dots a_{k-2}) \\ &= \sum_{j=1}^{n} (W(a_{k-1}, j, i) \cdot q(j, a_1 \dots a_{k-2})) \; + \; W(a_{k-1}, n+1, i) \cdot q(n+1, a_1 \dots a_{k-2}) \\ &\overset{IH}{=} 0 + W(a_{k-1}, n+1, i) \cdot f(a_1 \dots a_{k-2}) \\ &= 0\end{aligned}$$

Note that $f(a_1 \dots a_{k-2}) = 0$ because of the minimality of u_0. □

Figure 2 shows as an example the computation trees for a series f that counts the number of $X's$ after the last o. For $n = 1$, the only path available is closed since $f(o) = 0$ forces the leaf $(1, 1)$ to 0. On the other hand, for $n = 2$, one path remains open, while the path containing weight $(o, 1, 2)$ has been closed as this weight is forced to 0 in step 2.2., leading to the pruned tree as shown. This is consistent with the fact that a 2-state automaton can recognize f.

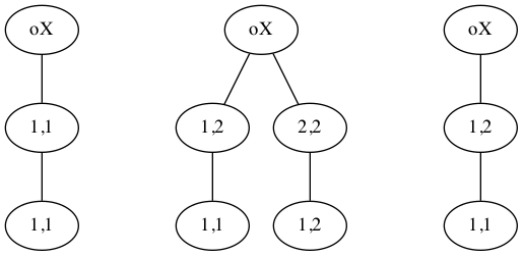

Fig. 2: Computation trees for the word $w = oX$ and a SNN of size $n = 1$ (left) and $n = 2$ (middle) as well as the pruned version (right) for the series f.

3.3 Complexity

We briefly review computational costs for executing the computation tree algorithm. In the first preparatory step, we enumerate all words up to a certain length; thus step 1.1 has cost $\mathcal{O}(|\Sigma|^l)$ where l is the length of the qualifying words.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Step 1.2 clearly has complexity $\mathcal{O}(|\Sigma|)$. While this preparation is expensive, note that it only needs to be performed once. To determine the SNN's required size n, one will usually execute above algorithm several times for different values of the input parameter n.

Theorem 2. *The computation tree algorithm runs in polynomial time on input n.*

Proof. Let k denote the (constant) number of words from step 1.1. The for-loops in steps 2.1 and 2.2 are executed $|\Sigma|^k$ and k times, respectively, and involve in their body traversal of a perfect n-ary tree with an additional row of leaves. (Since only entire paths are deleted in step 2.2, this, too, is essentially a tree traversal as no node rearrangements are necessary.) Tree traversal is linear in the number of nodes, which is $n^{l-1} + \frac{n^l-1}{n-1}$. Thus, the entire algorithm runs in $\mathcal{O}(|\Sigma|^k \cdot (n^{l-1} + \frac{n^l-1}{n-1}))$, which is polynomial in the input parameter n. □

4 Conclusions and Future Work

As training of SNNs is computationally expensive, it is desirable to know a-priori how many nodes are needed for a SNN to compute a given series. We have presented the computation tree to algorithmically check if the proposed size of a semiring neural network is too small to be successfully trained, thus avoiding training runs that are bound to be unsuccessful. An interesting open problem for future research is whether *not* finding a contradiction in the computation tree guarantees that the network can be successfully trained. This solvability of the induced system of polynomial equations depends on the underlying semiring.

References

[1] Sebastian Bader, Steffen Hölldobler, and Alexandre Scalzitti. Semiring Artificial Neural Networks and Weighted Automata. In Susanne Biundo, Thom W Frühwirth, and Günther Palm, editors, *27th Annual German Conference on AI*, pages 281–294, Ulm, Germany, September 2004. Springer.

[2] Paul Werbos. Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*, 78(10):1550–1560, October 1990.

[3] Anna S Rakitianskaia and Andries Petrus Engelbrecht. Training neural networks with PSO in dynamic environments. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pages 667–673, Trondheim, Norway, May 2009. IEEE.

[4] Darrell Whitley and Timothy Starkweather. GENITOR II: a distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:189–214, 1990.

[5] Darrell Whitley and Thomas Hanson. Using genetic recombination to optimize neural networks. In *International Joint Conference on Neural Networks*, page 591 vol.2. IEEE, 1989.

[6] Hiroaki Kitano. Empirical studies on the speed of convergence of neural network training using genetic algorithms. *AAAI'90: Proceedings of the eighth National conference on Artificial intelligence*, 2:789–795, July 1990.

[7] Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. Springer, Berlin, Germany, January 2009.

[8] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.