Formal Determination of Context in Contextual Recursive Cascade Correlation Networks

Alessio Micheli[1], Diego Sona[2], and Alessandro Sperduti[3]

[1] Dipartimento di Informatica, Università di Pisa, Italy
[2] SRA division, ITC-IRST, Trento, Italy
[3] Dipartimento di Matematica Pura ed Applicata, Università di Padova, Italy

Abstract. We consider the Contextual Recursive Cascade Correlation model (CRCC), a model able to learn contextual mappings in structured domains. We propose a formal characterization of the "context window", i.e., given a state variable, the "context window" is the set of state variables that contribute, directly or indirectly, to its determination. On the basis of this definition, a formal and compact expression describing the "context windows" for the CRCC, and RCC model, are derived.

1 Introduction

In recent years, neural network models for processing of structured data have been developed and studied (see for example [6, 3, 7, 2]). Almost all these models assume causality and stationarity. Unfortunately, in several real-world applications, such as language understanding and DNA and proteins analysis, these assumptions do not hold. For this reason some authors are starting to propose models that try to exploit contextual information within recurrent and recursive models. Examples of these models for the processing of sequences are described in [1, 4, 8], while in [5] the *Contextual Recursive Cascade Correlation* model (CRCC), able to process structured data, i.e., directed positional acyclic graphs, in an incremental and contextual fashion has been proposed.

In CRCC, which combines both structured data and contextual information, the adoption of an incremental learning strategy, such as Cascade Correlation, makes it hard to relate the insertion of new state variables in the model (i.e., hidden units) with the "shape" of the contextual information taken into account during learning and computation.

The aim of this paper is to formally elucidate how the "shape" of the contextual information is determined by the insertion of new state variables. Specifically, here we abstract from any specific neural realization of the CRCC model, and propose to define the "context window" of a given state variable in the model as the set of (already defined) state variables that contribute, directly or indirectly, to the determination of the state variable in consideration. Starting from this definition, we are able to define a compact expression which describes the context window as a function of the state variables in the model, and the sets of parents and children belonging to the vertex taken in consideration in the current input structure. The "context window" for the non-contextual Recursive Cascade Correlation (RCC) [6] is computed as well. These context windows can be exploited to characterize the computational power of CRCC versus RCC.

2 Structured Domains

Given a directed acyclic graph $\mathcal{D}$ we denote the vertex set with $\mathrm{vert}(\mathcal{D})$ and the edge set with $\mathrm{egd}(\mathcal{D})$. Given a vertex $v \in \mathrm{vert}(\mathcal{D})$ we denote the set of edges entering and leaving from v with $\mathrm{egd}(v)$. Moreover, we define: $\mathrm{out\_set}(v) = \{u | (v,u) \in \mathrm{egd}(v)\}$, and $\mathrm{in\_set}(v) = \{u | (u,v) \in \mathrm{egd}(v)\}$.

We assume that instances in the learning domain are DPAGs (directed positional acyclic graphs) with bounded outdegree *out* and indegree *in*. An instance of DPAG is a DAG where we assume that for each vertex $v \in \mathrm{vert}(\mathcal{D})$, two injective functions $P_v : \mathrm{egd}(v) \to [1, 2, ..., in]$ and $S_v : \mathrm{egd}(v) \to [1, 2, ..., out]$ are defined on the edges entering and leaving from v. In this way, a positional index is assigned to each entering and leaving edge from a vertex v. Moreover, given $(v,u) \in \mathrm{egd}(\mathcal{D})$, $\forall u \in \mathrm{vert}(\mathcal{D})$, let define $\forall j \in [1, \ldots, in] \quad \mathrm{in\_set}_j(u) = v \;$ if $\exists v \in \mathrm{vert}(\mathcal{D})$ such that $P_u((v,u)) = j$; nil otherwise, and similarly, $\forall j \in [1, \ldots, out] \quad \mathrm{out\_set}_j(u) = v \;$ if $\exists v \in \mathrm{vert}(\mathcal{D})$ such that $S_u((u,v)) = j$; nil otherwise. Vertexes are labeled by vectors of real numbers. With $\boldsymbol{l}(v)$ we denote the label associated to v. We shall require the DPAGs to possess a supersource, i.e. a vertex $s \in \mathrm{vert}(\mathcal{D})$ such that every vertex in $\mathrm{vert}(\mathcal{D})$ can be reached by a directed path starting from s. With $\mathrm{dist}(u,v)$, where $u, v \in \mathcal{D}$, we denote the shortest (directed) path in $\mathcal{D}$ from the vertex u to the vertex v.

3 Recursive Models

Recursive Neural Networks [3, 7] possess, in principle, the ability to memorize "past" information to perform structural mappings. The state transition function $\tau()$ and the output function $g()$, in this case, prescribe how the state variable, or better the state vector $\boldsymbol{x}(v)$ associated to a vertex v is used to obtain the state and output vectors corresponding to other vertexes, respectively. Specifically, given a state vector $\boldsymbol{x}(v) \equiv [x_1(v), \ldots, x_m(v)]^t$, we define the extended shift operators $q_j^{-1} x_k(v) \equiv x_k(\mathrm{out\_set}_j(v))$ and $q_j^{+1} x_k(v) \equiv x_k(\mathrm{in\_set}_j(v))$. If $\mathrm{out\_set}_j(v) = nil$ then $q_j^{-1} x_k(v) \equiv x_k(nil) \equiv x_0$, the null state. Similarly, if $\mathrm{in\_set}_j(v) = nil$ then $q_j^{+1} x_k(v) \equiv x_0$.

Moreover, we define $\boldsymbol{q}^{-1} x_k(v) = [q_1^{-1} x_k(v), \ldots, q_{out}^{-1} x_k(v)]^t$, and $\boldsymbol{q}^{+1} x_k(v) = [q_1^{+1} x_k(v), \ldots, q_{in}^{+1} x_k(v)]^t$ and, given $e \in \{-1, +1\}$, $\boldsymbol{q}^e \boldsymbol{x}(v) = [\boldsymbol{q}^e x_1(v), \ldots, \boldsymbol{q}^e x_m(v)]$.

On the basis of these definitions, the mapping implemented by a Recursive Neural Network can be described by the following equations:

$$\begin{cases} \boldsymbol{x}(v) = \tau(\boldsymbol{l}(v), \boldsymbol{q}^{-1}\boldsymbol{x}(v)) \\ \boldsymbol{y}(v) = g(\boldsymbol{l}(v), \boldsymbol{x}(v)) \end{cases} \tag{1}$$

where $\boldsymbol{x}(v)$ and $\boldsymbol{y}(v)$ are respectively the network state and the network output associated to vertex v,. This formulation is based on a structural version of the *causality* assumption, i.e., the output $\boldsymbol{y}(v)$ of the network at vertex v only depends on v and on its descendants. Specifically, RCC equations, where we disregard direct connections between hidden units, can be written as ($j = 1, \ldots, m$)

$$x_j(v) = \tau_j(\boldsymbol{l}(v), \boldsymbol{q}^{-1}[x_1(v), \ldots, x_j(v)]), \tag{2}$$

where $x_i(v)$ is the output of the i-th hidden unit in the network when processing vertex v. Since RCC is a constructive algorithm, training of a new hidden unit is based on already frozen units. Thus, when training hidden unit k, the state variables $x_1, \ldots, x_{k-1}$ for *all* the vertexes of all the DPAGs in the training set are already available, and can be used in the definition of x_k. This observation is very important since it yields to the realization that *contextual* information is already available in RCC, but it is not exploited.

The *Contextual Recursive Cascade Correlation* network [5] exploits this information. Specifically, eqs. (2) can be expanded in a contextual fashion by using, where possible, the shift operator $\boldsymbol{q}^{+1}$:

$$x_1(v) = \tau_1(\boldsymbol{l}(v), \boldsymbol{q}^{-1}x_1(v)), \tag{3}$$

$$x_j(v) = \tau_j(\boldsymbol{l}(v), \boldsymbol{q}^{-1}[x_1(v), .., x_j(v)], \boldsymbol{q}^{+1}[x_1(v), .., x_{j-1}(v)]), \; j = 2, \ldots, m. \tag{4}$$

4 Formal Determination of Context

In this section, we characterize the "shape" of the context exploited by a state variable in terms of the set of state variables which contribute directly or indirectly to its determination. Specifically, we propose a definition of "contextual window" and elucidate how the "shape" of the "contextual window" evolves with the addition of hidden units, both in the CRCC model and in the RCC model.

In order to formalize the above concepts, given a subset $V \subseteq \text{vert}(\mathcal{D})$ let us define the "descendants" operator $\downarrow$ as $\downarrow V = V \cup \downarrow \text{out\_set}(V) = V \cup \{u | v \in V \wedge \exists \, path(v, u)\}$, where the descendants set of an empty set is still an empty set, i.e., $\downarrow\emptyset = \emptyset$.

Moreover, let the set functions in_set$(\cdot)$ and out_set$(\cdot)$ be defined also for sets of vertexes as argument (e.g., $\text{in\_set}(V) = \bigcup_{v \in V} \text{in\_set}(v)$) and let denote with $(\downarrow\text{in\_set})^p(V)$ the repeated application (p times) of the in_set function composed with the "descendants" operator $\downarrow$. Moreover, given a subset of vertexes V, with the notation $x_j.V$ we refer to the set of state variables $\{x_j(v) | v \in V\}$, where for an empty subset of vertexes we define $x_j.\emptyset = \{x_0\}$. Note that for any couple of vertexes sets $V, Z \subseteq \text{vert}(\mathcal{D})$, $\downarrow V \cup \downarrow Z = \downarrow (V \cup Z)$ and $x_j.V \cup x_j.Z = x_j.(V \cup Z)$. Finally, let formally define the concept of *context window*, i.e. the set of internal states which may contribute to the computation of a new internal state

Definition 1 (Context Window). *The context window of a state variable $x_k(v)$, denoted by $\mathcal{C}(x_k(v))$, is defined as the set of* all *state variables (directly or indirectly) contributing to its determination.*

Note that, by definition, $\mathcal{C}(x_0) = \emptyset$. In the following we will use the above definition to characterize the context window of the CRCC and RCC models.

First of all, let us state some basic results that will be used to derive the main theorems.

All the results reported in the following will concern the CRCC model. Referring to eq. (3) it is now possible to derive the following

Lemma 1. *Given a DPAG $\mathcal{D}$, for any vertex $v \in \mathrm{vert}(\mathcal{D})$*

$$\mathcal{C}(x_1(v)) = x_1. \downarrow \mathrm{out\_set}(v).$$

Proof. It can be proved by induction on the partial order of vertexes.
By definition of context window, and by eq. (3) for any v in $\mathrm{vert}(\mathcal{D})$, the context window for the state variable $x_1(v)$ can be expressed as

$$\mathcal{C}(x_1(v)) = \underbrace{\bigcup_{i=1}^{out} q_i^{-1} x_1(v)}_{\text{direct}} \ \cup \ \underbrace{\bigcup_{i=1}^{out} \mathcal{C}(q_i^{-1} x_1(v))}_{\text{indirect}}$$

Base Case: v is a leaf of $\mathcal{D}$.

By definition of out_set, $\forall i \in [1, \ldots, \mathrm{out}]$ $\mathrm{out\_set}_i(v) = \emptyset$, thus by definition of "descendants" operator $x_1. \downarrow \mathrm{out\_set}(v) = x_1.\emptyset = \{x_0\}$.
Moreover, by definition of extended shift operator, $\forall i \in [1, \ldots, out]$ $q_i^{-1} x_1(v) = x_0$, and since $\mathcal{C}(x_0) = \emptyset$, $\mathcal{C}(q_i^{-1} x_1(v)) = \mathcal{C}(x_0) = \emptyset$. This proves that $\mathcal{C}(x_1(v)) = x_1. \downarrow \mathrm{out\_set}(v) = \{x_0\}$ for any leaf v of the data structure $\mathcal{D}$.
Inductive Step: v is an internal vertex of $\mathcal{D}$.

Assume that: $\forall u \in \mathrm{out\_set}(v) \quad \mathcal{C}(x_1(u)) = x_1. \downarrow \mathrm{out\_set}(u)$ (Inductive Hyp.)

By definition of generalized shift operator, and because of the inductive hypothesis,

$$\mathcal{C}(x_1(v)) = \bigcup_{u \in \mathrm{out\_set}(v)} [x_1(u) \ \cup \ x_1. \downarrow \mathrm{out\_set}(u)] = \bigcup_{u \in \mathrm{out\_set}(v)} x_1. \downarrow u = x_1. \downarrow \mathrm{out\_set}(v)$$

□

Referring to eq. (4) it is now possible to derive the following results, which are then used to prove Theorem 1.

Lemma 2. *Given a DPAG $\mathcal{D}$, for any vertex $v \in \mathrm{vert}(\mathcal{D})$ the following equation holds*

$$\mathcal{C}(x_2(v)) = x_1. \downarrow \mathrm{in\_set}(\downarrow v) \ \cup \ x_2. \downarrow \mathrm{out\_set}(v). \tag{5}$$

Proof. It can be proved by induction on the partial order of vertexes.
By definition of context window, and by eq. (4) for any v in $\mathrm{vert}(\mathcal{D})$ the context window for the state variable $x_2(v)$ can be expressed by

$$\begin{aligned}\mathcal{C}(x_2(v)) = & \bigcup_{i=1}^{out} [q_i^{-1} x_2(v) \ \cup \ \mathcal{C}(q_i^{-1} x_2(v))] \ \cup \bigcup_{i=1}^{out} [q_i^{-1} x_1(v) \ \cup \ \mathcal{C}(q_i^{-1} x_1(v))] \\ & \cup \bigcup_{i=1}^{in} [q_i^{+1} x_1(v) \ \cup \ \mathcal{C}(q_i^{+1} x_1(v))]\end{aligned} \tag{6}$$

Base Case: v is a leaf of $\mathcal{D}$.

By the definitions of out_set, in_set, and "descendants" operator it follows that $x_2.\downarrow\text{out\_set}(v) = x_2.\emptyset = \{x_0\}$, and $x_1.\downarrow\text{in\_set}(\downarrow v) = x_1.\downarrow\text{in\_set}(v)$. Thus eq. (5) becomes $\mathcal{C}(x_2(v)) = x_1.\downarrow\text{in\_set}(v) \cup \{x_0\}$.

By definition of generalized shift operators, $q_i^{-1}x_2(v) = x_2.\text{out\_set}_i(v) = \{x_0\}$, and $q_i^{+1}x_1(v) = x_1.\text{in\_set}_i(v)$. Moreover, since $\mathcal{C}(x_0) = \emptyset$, $\mathcal{C}(q_i^{-1}x_2(v)) = \mathcal{C}(x_2.\text{out\_set}_i(v)) = \mathcal{C}(\{x_0\}) = \emptyset$, and by Lemma 1, $\mathcal{C}(q_i^{+1}x_1(v)) = \mathcal{C}(x_1(\text{in\_set}_i(v))) = x_1.\downarrow\text{out\_set}(\text{in\_set}_i(v))$. Thus, by the associative rule on "dot" and by definition of "descendants" operator, eq. (6) becomes $\mathcal{C}(x_2(v)) = \{x_0\} \cup x_1.\text{in\_set}(v) \cup x_1.\downarrow\text{out\_set}(\text{in\_set}(u)) = x_1.\downarrow\text{in\_set}(v)$. The thesis follows for all leaves of $\mathcal{D}$.

Inductive Step: v is an internal vertex of $\mathcal{D}$. Assume that $\forall u \in \text{out\_set}(v)$ $\mathcal{C}(x_2(u)) = x_1.\downarrow\text{in\_set}(\downarrow u) \cup\ x_2.\downarrow\text{out\_set}(u)$ (Inductive Hyp.).

By definition of generalized shift operator, associative rule for "dot" definition of "descendants" operator, and by Lemma 1, eq. (6) becomes

$$\mathcal{C}(x_2(v)) = \bigcup_{u\in\text{out\_set}(v)} [x_2(u)\ \cup\ \mathcal{C}(x_2(u))]\ \cup\ x_1.\downarrow\text{out\_set}(v) \cup\ x_1.\downarrow\text{in\_set}(v)$$

By the inductive hypothesis and by the definition of "descendants" operator

$$\mathcal{C}(x_2(v)) = x_2.\downarrow\text{out\_set}(v) \cup\ x_1.\downarrow\text{in\_set}(\downarrow\text{out\_set}(v)) \cup x_1.\downarrow\text{out\_set}(v) \cup\ x_1.\downarrow\text{in\_set}(v)$$

By the associative rules and by definition of in_set

$$\mathcal{C}(x_2(v)) = x_2.\downarrow\text{out\_set}(v) \cup\ x_1.\downarrow\text{in\_set}(\downarrow v) \cup\ x_1.\downarrow\text{out\_set}(v).$$

Since for any $v \in \text{vert}(\mathcal{D})$ it holds $\downarrow\text{out\_set}(v) \subset \downarrow v \subseteq \downarrow\text{in\_set}(\downarrow v)$ the thesis follows. □

Theorem 1. *Given a DPAG $\mathcal{D}$, $\forall v \in \text{vert}(\mathcal{D})$, and $k \geq 2$*

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k-1} x_i.(\downarrow\text{in\_set})^{k-i}(\downarrow v)\ \cup\ x_k.\downarrow\text{out\_set}(v). \tag{7}$$

Proof. The theorem can be proved by induction on the partial order of vertexes, and on the order of indexes of variables.

By definition of context window, by eq. (4), and by definition of generalized shift operator, for any v in $\text{vert}(\mathcal{D})$ the context window for the state variable $x_k(v)$, using the fact that for any vertex $v \in \text{vert}(\mathcal{D})$ and for any $k \geq 2$, $\mathcal{C}(x_k(v)) \supseteq \mathcal{C}(x_{k-1}(v))$ (which is not hard to prove), can be expressed as

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k} x_i.\text{out\_set}(v)\ \cup\ \mathcal{C}(x_k.\text{out\_set}(v))\ \cup\ \bigcup_{i=1}^{k-1} x_i.\text{in\_set}(v)\ \cup\ \mathcal{C}(x_{k-1}.\text{in\_set}(v))$$

Base Case: v is a leaf of $\mathcal{D}$. For $k = 2$, eq. (7) reduces to eq. (5). Now, let assume that $\forall j,\ 2 \leq j \leq k-1$

$$\mathcal{C}(x_j(v)) = \bigcup_{i=1}^{j-1} x_i.(\downarrow\text{in\_set})^{j-i}(\downarrow v)\ \cup\ x_j.\downarrow\text{out\_set}(v) \quad \text{(Inductive Hyp.)} \tag{8}$$

Since v is a leaf, eq. (8) can be reduced to $\mathcal{C}(x_k(v)) = \{x_0\} \cup \emptyset \cup \bigcup_{i=1}^{k-1} x_i.\text{in\_set}(v) \cup \mathcal{C}(x_{k-1}.\text{in\_set}(v))$ and, applying the inductive hypothesis 8, it becomes

$$\mathcal{C}(x_k(v)) = x_{k-1}.\text{in\_set}(v) \;\cup\; x_{k-1}.\downarrow\text{out\_set}(\text{in\_set}(v)) \;\cup\; \bigcup_{i=1}^{k-2}[x_i.\text{in\_set}(v) \;\cup\; x_i.(\downarrow\text{in\_set})^{k-i}(v)]$$

By definition of "descendants" operator, and because of the fact that for any $v \in \text{vert}(\mathcal{D})$ and for any $j \geq 1$ it trivially holds $\text{in\_set}(v) \subseteq\downarrow\text{in\_set}(v) \subseteq (\downarrow\text{in\_set})^j(v)$, it follows that

$$\mathcal{C}(x_k(v)) = x_{k-1}.\downarrow\text{in\_set}(v) \;\cup\; \bigcup_{i=1}^{k-2} x_i.(\downarrow\text{in\_set})^{k-i}(v) = \bigcup_{i=1}^{k-1} x_i.(\downarrow\text{in\_set})^{k-i}(v)$$

which is equivalent to eq. (7) since $v =\downarrow v$ and $x_k.\downarrow\text{out\_set}(v) = \{x_0\}$. Which proves the correctness of the assertion for all leaves of $\mathcal{D}$ and for any $k \geq 2$.
Inductive Step: v is an internal vertex of $\mathcal{D}$. Assume that

$$\mathcal{C}(x_j(u)) = \bigcup_{i=1}^{j-1} x_i.(\downarrow\text{in\_set})^{j-i}(\downarrow u) \;\cup\; x_j.\downarrow\text{out\_set}(u) \quad \text{(Inductive Hyp.)} \quad (9)$$

holds $\forall u \in \text{vert}(\mathcal{D})$ if $2 \leq j \leq k-1$, and $\forall u \in \text{out\_set}(v)$ if $j = k$, then, eq. (8) becomes

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k} x_i.\text{out\_set}(v) \;\cup\; \mathcal{C}(x_k.\text{out\_set}(v)) \;\cup\; \bigcup_{i=1}^{k-1} x_i.\text{in\_set}(v) \;\cup\; \left[\bigcup_{i=1}^{k-2} x_i.(\downarrow\text{in\_set})^{k-i}(v) \;\cup\; x_{k-1}.\downarrow\text{out\_set}(\text{in\_set}(v))\right]$$

By the associative rule of "dot" and "descendants" operators

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k} x_i.\text{out\_set}(v) \;\cup\; \mathcal{C}(x_k.\text{out\_set}(v)) \;\cup\; \bigcup_{i=1}^{k-2} x_i.\text{in\_set}(v) \;\cup \bigcup_{i=1}^{k-1} x_i.(\downarrow\text{in\_set})^{k-i}(v)$$

Applying again the induction hypothesis 9 on $\mathcal{C}(x_k.\text{out\_set}(v))$

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k} x_i.\text{out\_set}(v) \;\cup \bigcup_{i=1}^{k-1} x_i.(\downarrow\text{in\_set})^{k-i}(\downarrow\text{out\_set}(v)) \;\cup\; x_k.\downarrow\text{out\_set}(\text{out\_set}(v)) \;\cup\; \bigcup_{i=1}^{k-2} x_i.\text{in\_set}(v) \;\cup\; \bigcup_{i=1}^{k-1} x_i.(\downarrow\text{in\_set})^{k-i}(v)$$

By the associative properties of "dot" and "descendants" operators applied to $x_i.(\downarrow\text{in\_set})^{k-i}$ and $x_k.\text{out\_set}$

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k-1} x_i.\text{out\_set}(v) \;\cup\; \bigcup_{i=1}^{k-2} x_i.\text{in\_set}(v) \;\cup \bigcup_{i=1}^{k-1} x_i.(\downarrow\text{in\_set})^{k-i}(\downarrow v) \;\cup\; x_k.\downarrow\text{out\_set}(v)$$

Since for any $v \in \mathrm{vert}(\mathcal{D})$ and for any $j \geq 1$ it trivially holds that $\mathrm{out\_set}(v) \subseteq \downarrow\!\mathrm{in\_set}(v) \subseteq (\downarrow\!\mathrm{in\_set})^j(v) \subseteq (\downarrow\!\mathrm{in\_set})^j(\downarrow v)$, and similarly $\mathrm{in\_set}(v) \subseteq \downarrow\!\mathrm{in\_set}(v) \subseteq (\downarrow\!\mathrm{in\_set})^j(v) \subseteq (\downarrow\!\mathrm{in\_set})^j(\downarrow v)$ the thesis follows. □

In general, for CRCC, the evolution of the context with respect to the addition of hidden units is characterized by the following property.

Proposition 1. *Given a vertex v in a DPAG $\mathcal{D}$ with supersource s, such that* $\mathrm{dist}(s, v) = d$, *the contexts $\mathcal{C}(x_h(v))$, with $h > d$ involve all the vertexes of $\mathcal{D}$.*

Proof. According to eq. (7), when computing $\mathcal{C}(x_{d+1}(v))$, in_set is recursively applied d times. Thus the shortest path from s to v is fully followed in a backward fashion starting from v, so that $x_1(s)$ is included in $\mathcal{C}(x_{d+1}(v))$. Moreover, since $x_1(s)$ is included in $\mathcal{C}(x_{d+1}(v))$, by definition of eq. (7), also the state variables $x_1(u)$ for each $u \in \mathrm{vert}(\mathcal{D})$ are included in $\mathcal{C}(x_{d+1}(v))$. The statement follows from the fact that $\mathcal{C}(x_{d+1}(v)) \subset \mathcal{C}(x_h(v))$. □

Moreover, when considering a transduction whose output for each vertex v depends on the whole structure, the following proposition suggests that such information is available to a CRCC model with "enough" hidden units.

Proposition 2. *Given a DPAG $\mathcal{D}$ there exists a finite number h of hidden units such that for each $v \in \mathrm{vert}(\mathcal{D})$ the context $\mathcal{C}(x_h(v))$ involves all the vertexes of $\mathcal{D}$. In particular, given $r = \max_{v \in \mathrm{vert}(\mathcal{D})} \mathrm{dist}(s, v)$, any $h > r$ satisfies the proposition.*

Proof. Let consider $h = r + 1$. The proposition follows immediately by the application of Proposition 1 for each v since $h > \mathrm{dist}(s, v)$. □

For the sake of comparison, let us now consider the "context window" for the RCC model (eq. 2). The following theorem holds:

Theorem 2. *Given a DPAG $\mathcal{D}$, $\forall v \in \mathrm{vert}(\mathcal{D})$, and $k \geq 1$*

$$\mathcal{C}_{RCC}(x_k(v)) = \bigcup_{i=1}^{k} x_i.\downarrow\!\mathrm{out\_set}(v). \tag{10}$$

Proof. For $k = 1$ the proof is given by Lemma 1, since eq. 2 is equivalent to eq. 3. For $k > 1$ the proof is similar to the proof given for Theorem 1. □

The above result formally show that RCC is unable to perform contextual transductions, since only information on the descendants of a vertex is available. Finally, it is worth to stress out that the above results allow to prove

Lemma 3. *There exist two distinct DPAGs, $\mathcal{D}_1$ with supersource s_1 and $\mathcal{D}_2$ with supersource s_2, such that the state vectors $\boldsymbol{x}^{RCC}(s_1)$ and $\boldsymbol{x}^{RCC}(s_2)$ computed by RCC are identical, while the state vectors $\boldsymbol{x}(s_1)$ and $\boldsymbol{x}(s_2)$ computed by CRCC are different.*

This lemma can be used to show that RCC can compute only supersource transductions (thus, non-contextual transductions) involving *tree structures*, while CRCC allows for the computation of (many, not all!) transductions (both supersource transductions and contextual transductions) that involve DPAGs.

5 Conclusion

We have considered the problem to give a formal characterization of the functional dependencies between state variables in recursive models based on Cascade Correlation for the processing of structured data. Specifically, we have defined the "context window" for a state variable as the set of state variables that contribute, directly or indirectly, to the determination of the state variable in consideration. Then, on the basis of this definition, we have been able to define a compact expression which describes the "context window" as a function of the state variables in the model associated to specific sets of ancestors and descendants belonging to the vertex taken in consideration. This expression has been devised both for the *Recursive Cascade Correlation* model and the *Contextual Recursive Cascade Correlation* model.

The relevance of the results obtained in this paper is due to the possibility to use them in order to characterize the computational power of the above models. Moreover, the approach used in this paper, which is independent from the specific neural implementation, can easily be applied to other recurrent and recursive models in order to compare them from a computational point of view.

References

1. P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, and G. Soda. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.
2. P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9(5):768–786, 1998.
3. C. Goller and A. Küchler. Learning task-dependent distributed structure-representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*, pages 347–352, 1996.
4. A. Micheli, D. Sona, and A. Sperduti. Bi-causal recurrent cascade correlation. In *Proc. of the Int. Joint Conf. on Neural Networks - IJCNN'2000*, volume 3, pages 3–8, 2000.
5. A. Micheli, D. Sona, and A. Sperduti. Recursive cascade correlation for contextual processing of structured data. In *Proc. of the Int. Joint Conf. on Neural Networks - WCCI-IJCNN'2002*, volume 1, pages 268–273, 2002.
6. A. Sperduti, D. Majidi, and A. Starita. Extended cascade-correlation for syntactic and structural pattern recognition. In P. Perner, P. Wang, and A. Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition*, volume 1121 of *Lecture notes in Computer Science*, pages 90–99. Springer-Verlag, 1996.
7. A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.
8. H. Wakuya and J. Zurada. Bi-directional computing architectures for time series prediction. *Neural Network*, 14:1307–1321, 2001.