On Linear Separability of Sequences and Structures

Alessandro Sperduti

Dipartimento di Matematica Pura ed Applicata, Università di Padova,
Via Belzoni 7, 35131 Padova, Italy

Abstract. Linear separability of sequences and structured data is studied. On the basis of a theoretical model, necessary and sufficient conditions for nonlinear separability are derived by a well known result for vectors. Examples of sufficient conditions for linear separability of both sequences and structured data are given.

1 Introduction

Historically the concept of linear separability for a set of labeled vectors has been very important since it allowed to characterize the computational limits of the Perceptron model and, consequently, it stimulated the development of more powerful neural network models, such has feed-forward networks. The introduction of neural network models for the processing of sequences (see for example, [9, 2, 3]) and structured data [8, 6, 7, 4] increased the complexity of the problems which can be faced using a neural approach. The increase in complexity of data, however, makes it more difficult to identify classification problems which are easy to solve. In fact, the standard notion of linear separability cannot be applied directly to the case of sequences and structures.

The aim of this paper is to try to redefine for these cases the concept of linear separability and to extend some of the results already known for vectors. Even if here we are able to give necessary and sufficient conditions on linear separability of sequences and structures, as well as some examples of sufficient conditions for linear separability in these contexts, much more work is still needed in order to define an efficient algorithm for learning these classes of problems.

2 Preliminaries on Data Structures

In this paper we assume that instances in the learning domain are DOAGs (directed ordered acyclic graphs) or DPAGs (directed positional acyclic graphs). A DOAG is a DAG $\mathcal{D}$ with vertex set $\mathrm{vert}(\mathcal{D})$ and edge set $\mathrm{egd}(\mathcal{D})$, where for each vertex $v \in \mathrm{vert}(\mathcal{D})$ a total order on the edges leaving from v is defined. DPAGs are a superclass of DOAGs in which it is assumed that for each vertex v, a bijection $P : \mathrm{egd}(\mathcal{D}) \rightarrow I\!N$ is defined on the edges leaving from v. The *indegree* of node v is the number of incoming edges to v, whereas the *outdegree* of v is the number of outgoing edges from v.

We shall require the DAG (either DOAG or DPAG) to possess a supersource[1], i.e. a vertex $s \in \text{vert}(\mathcal{D})$ such that every vertex in $\text{vert}(\mathcal{D})$ can be reached by a directed path starting from s. Given a DAG $\mathcal{D}$ and $v \in \text{vert}(\mathcal{D})$, we denote by $\text{ch}[v]$ the set of children of v, and by $\text{ch}_k[v]$ the k-th child of v.

We shall use lowercase bold letters to denote vectors, and calligraphic letters for representing graphs. A data structure $\mathcal{Y}$ is a DAG whose vertices are labeled by vectors of real-valued numbers which either represent numerical or categorical variables. Subscript notation will be used when referencing the labels attached to vertices in a data structure. Hence $\boldsymbol{y}_v$ denotes the vector of variables labeling vertex $v \in \text{vert}(\mathcal{Y})$.

In the following, we shall denote by $\#^{(i,c)}$ the class of DAGs with maximum indegree i and maximum outdegree c. The class of all data structures defined over the label universe domain $\boldsymbol{\mathcal{Y}}$ and skeleton in $\#^{(i,c)}$ will be denoted as $\boldsymbol{\mathcal{Y}}^{\#^{(i,c)}}$.

Here, we consider binary classification tasks where the training set T is defined as a set of couples $(\mathcal{U}, \xi(\mathcal{U}))$, where $\mathcal{U} \in \boldsymbol{\mathcal{U}}^{\#^{(i,c)}}$ and $\xi()$ is the target function $\xi : \boldsymbol{\mathcal{U}}^{\#^{(i,c)}} \rightarrow \{-1, 1\}$.

3 Theoretical Model

In accordance with several popular neural network models for processing of sequences and structures, we define the net input of a linear recursive neuron for a node v of a structure $\mathcal{U} \in \boldsymbol{\mathcal{U}}^{\#^{(i,c)}}$ as:

$$net(v) = \boldsymbol{W}^t \boldsymbol{u}_v + \sum_{j=1}^{c} \hat{w}_j net(ch_j[v]), \qquad (1)$$

where $\boldsymbol{u}_v \in \boldsymbol{\mathcal{U}}$, $\boldsymbol{W}$ are the weights associated to the input label, and $\hat{w}_j$ are the weights on the recursive connections. Note that, the net input of the neuron for a node v is computed recursively on the net computed for all the nodes pointed by it. The output of the linear recursive neuron for $\mathcal{U}$ is then computed as

$$o(\mathcal{U}) = sgn(net(s)) = \begin{cases} 1 & \text{if } net(s) \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

where s is the *supersource* of $\mathcal{U}$.

4 Linear Separability for Sequences

When the structure at hand is a sequence $\mathcal{S} = (s_0, \ldots, s_{n-1})$ of length n, equation (1) reduces to $net(s_i) = \boldsymbol{W}^t \boldsymbol{u}_{s_i} + \hat{w}_1 net(s_{i-1}), \quad i = 0, \ldots, n-1$ where $\boldsymbol{u}_{s_i}$

[1] If no supersource is present, a new node connected with all the nodes of the graph with null *indegree* can be added.

is the label attached to s_i, and by definition, $net(s_{-1}) = 0$. Thus the output of the neuron for $\mathcal{S}$ can be written as

$$o(\mathcal{S}) = sgn(\boldsymbol{W}^t \underbrace{\sum_{i=0}^{n-1} \boldsymbol{u}_{s_i} (\hat{w}_1)^i}_{\boldsymbol{X}(\mathcal{S},\hat{w}_1)}) \ .$$

The interpretation of the above equation is straightforward: the sequence is encoded as a polynomial (in a vector field) of the recursive connection $\hat{w}_1$. The set of labels attached to the elements of the sequence are mapped, through the parameter $\hat{w}_1$, into a vector of the same dimension. Specifically, a sequence of length n is transformed by a polynomial of order $(n-1)$. The standard concept of *linear separability* of a set of vectors, can thus be restated very naturally

Definition 1 (Linear Separability for Sequences) *A set of sequences* $\boldsymbol{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ *is linearly separable (l.s.) if there exists a real number* $\bar{w}$ *such that the set* $\mathcal{X}(\boldsymbol{S}, \bar{w}) = \{\boldsymbol{X}(\mathcal{S}_1, \bar{w}), \ldots, \boldsymbol{X}(\mathcal{S}_m, \bar{w})\}$ *is linearly separable.*

Using this definition, it is not hard to reformulate some well known results for set of vectors, like the theorem

Theorem 1 (Linearly Nonseparable Vectors [5]). *A set of vectors* $\{Y_1, \ldots, Y_k\}$, $Y_i \in I\!R^d$, *is not l.s. if and only if there exists a positive linear combination of the vectors that equals* $\mathbf{0}$, *i.e., there exists* $b_i \geq 0,\ 1 \leq i \leq k$, *such that* $\sum_{i=1}^{k} b_i Y_i = \mathbf{0}$, *and* $b_j > 0$ *for some* $j,\ 1 \leq j \leq k$.

Note that, in the above theorem, the information about the class of the vectors is retained into the set of vectors by substituting negative examples $(Y_i, -1)$ with corresponding positive examples $(-Y_i, 1)$. In our framework, assuming a similar transformation for (all labels of) negative sequences, we can state

Theorem 2 (Linearly Nonseparable Sequences). *A set of sequences* $\boldsymbol{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ *is not l.s. if and only if* $\forall w \in I\!R$ *there exists a set of nonnegative numbers* $b_i,\ i = 1, \ldots, m$, *not all null, such that*

$$\sum_{i=1}^{m} b_i \boldsymbol{X}(\mathcal{S}_i, w) = \mathbf{0}, \ \ \textit{where} \ \ \boldsymbol{X}(\mathcal{S}_i, w) = \sum_{j=0}^{l_i - 1} \boldsymbol{u}_j^{\mathcal{S}_i} (w)^j \ .$$

Proof: The proof follows immediately by Definition 1 and Theorem 1. □

Equation (2) can be written in a more compact form as $\sum_{j=0}^{M-1} (\boldsymbol{L}_j \boldsymbol{b})(w)^j = \mathbf{0}$, where M is the maximum length of the sequences in $\mathcal{S}$, $\boldsymbol{b}^t = [b_1, \ldots, b_m]$, and the matrices $\boldsymbol{L}_j$, $0 \leq j < M$, are defined as

$$\boldsymbol{L}_j = \left[\boldsymbol{\lambda}_j^{\mathcal{S}_1}, \ldots, \boldsymbol{\lambda}_j^{\mathcal{S}_m} \right], \quad \text{where for } q = 1, \ldots, m, \quad \boldsymbol{\lambda}_j^{\mathcal{S}_q} = \begin{cases} \boldsymbol{u}_j^{\mathcal{S}_q} & \text{if } j < l_q \\ \mathbf{0} & \text{if } j \geq l_q \end{cases}$$

Then we have

Theorem 3. *If $\exists \boldsymbol{b} \in I\!R^m$ not null, and with the non null components all of the same sign, such that $\forall j$, $\boldsymbol{b} \in kernel(\boldsymbol{L}_j)$, then $\boldsymbol{S}$ is not l.s.*

Proof: If every $kernel(\boldsymbol{L}_j)$ contains the vector $\boldsymbol{b}$, then either $\boldsymbol{b}$ or $-\boldsymbol{b}$ will satisfy the conditions of Theorem 2 regardless of the value of w. □

In the following, we give some examples of sufficient conditions for l.s.

Theorem 4. *$\boldsymbol{S}$ is l.s. if one of the following conditions are satisfied:*

i) the columns of $\boldsymbol{L}_0$ are linearly separable;
ii) the set of vectors $\sigma_{\mathcal{S}_i} = \sum_{j=0}^{l_i-1} \boldsymbol{u}_j^{\mathcal{S}_i}$ are linearly separable;
iii) the set $Tails = \{\boldsymbol{u}_{l_1-1}^{\mathcal{S}_1}, \boldsymbol{u}_{l_2-1}^{\mathcal{S}_2}, \ldots, \boldsymbol{u}_{l_m-1}^{\mathcal{S}_m}\}$ is linearly separable;

Proof: If condition *i)* is true, then choosing $\bar{w} = 0$, for $q = 1, \ldots, m$, we have that $\boldsymbol{X}(\mathcal{S}_q, \bar{w}) = \boldsymbol{u}_0^{\mathcal{S}_q}$. Since the vectors $\boldsymbol{u}_0^{\mathcal{S}_q}$ are linearly separable, then also $\boldsymbol{S}$ is linearly separable. If condition *ii)* is satisfied, then trivially $\boldsymbol{S}$ is linearly separable by setting $w = 1$. If condition *iii)* is true, then, by definition of linear separability, there will be a vector $\bar{\boldsymbol{W}}$ such that, given $\xi_{j,i} = cos(\bar{\varphi}_{j,i})$ where $\bar{\varphi}_{j,i}$ is the angle between $\bar{\boldsymbol{W}}$ and $\boldsymbol{u}_i^{\mathcal{S}_j}$, for each i, $\bar{\boldsymbol{W}}\boldsymbol{u}_{l_i-1}^{\mathcal{S}_i} = \|\bar{\boldsymbol{W}}\| \, \|\boldsymbol{u}_{l_i-1}^{\mathcal{S}_i}\| \, \xi_{i,l_i-1} > 0$. Moreover, for each j, assuming $w > 0$

$$\left| \sum_{i=0}^{l_j-2} \xi_{j,i} \, \|\boldsymbol{u}_i^{\mathcal{S}_j}\| \, (w_1)^i \; \right| < \|\boldsymbol{u}_{i_j^*}^{\mathcal{S}_j}\| \sum_{i=0}^{l_j-2} (w_1)^i = \|\boldsymbol{u}_{i_j^*}^{\mathcal{S}_j}\| \, \frac{w_1^{l_j-1} - 1}{w_1 - 1} \; ,$$

where i_j^* is such that for each i, $\|\boldsymbol{u}_{i_j^*}^{\mathcal{S}_j}\| \geq \|\boldsymbol{u}_i^{\mathcal{S}_j}\|$. In order for $\boldsymbol{S}$ to be linearly separable, we have to find a $w_1 = \tilde{w}$ such that for each j

$$\|\boldsymbol{u}_{l_j-1}^{\mathcal{S}_j}\| \, (\tilde{w})^{l_j-1} \, \xi_{j,l_j-1} - \|\boldsymbol{u}_{i_j^*}^{\mathcal{S}_j}\| \, \frac{\tilde{w}^{l_j-1} - 1}{\tilde{w} - 1} \; > 0 \; . \tag{2}$$

Equation (2) is satisfied for each j if we choose $\tilde{w}$ to be the maximum over j of the quantities

$$w_j = \frac{\|\boldsymbol{u}_{l_j-1}^{\mathcal{S}_j}\| \, \xi_{j,l_j-1} + \|\boldsymbol{u}_{i_j^*}^{\mathcal{S}_j}\|}{\|\boldsymbol{u}_{l_j-1}^{\mathcal{S}_j}\| \, \xi_{j,l_j-1}} > 1 \; .$$

□

5 Linear Separability for Structures

Let consider a structure $\mathcal{U} \in \boldsymbol{\mathcal{U}}^{\#^{(i,c)}}$, then

$$o(\mathcal{U}) = o(s) = sgn(\boldsymbol{W}^t \underbrace{\sum_{v \in \mathrm{vert}(\mathcal{U})} \boldsymbol{u}_v \sum_{p \in Path_{\mathcal{U}}(v)} \prod_{j=1}^{n} (\hat{w}_j)^{\rho_j(p)}}_{\boldsymbol{\Xi}(\mathcal{U}, \hat{w}_1, \ldots, \hat{w}_n)}),$$

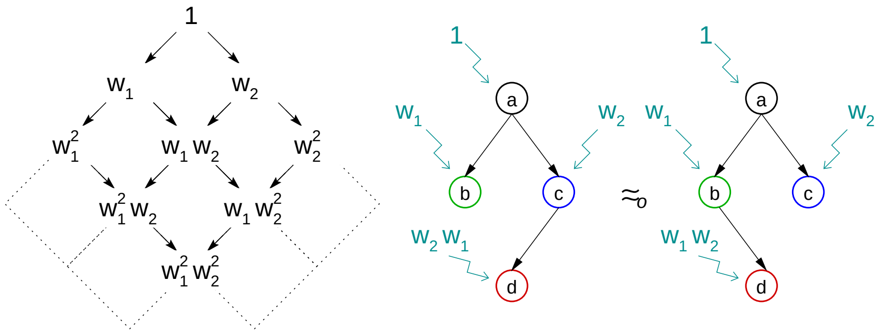

Fig. 1. Example of DAGs which are output-equivalent.

where s is the supersource of $\mathcal{U}$, $Path_{\mathcal{U}}(v)$ is the set of paths within $\mathcal{U}$ from s to v, and $\rho_j(p)$ is the number of occurrences within path p of the jth pointer.

From the above equation, it is clear that the structure $\mathcal{U}$ is encoded as a polynomial, in n variables, of degree $(P-1)$, where P is the maximum length of the paths from s to any node of $\mathcal{U}$. In [4] it was observed that in recursive neural networks, when assuming causality and stationarity, there exist distinct DAGs that get the same output (a *collision* occur), regardless of the value associated to the weights of the network. When considering linear recursive networks collisions are even more frequent. This concept has been formalized in [1] with the introduction of a binary relation $\approx_o$ called *output-equivalence*: two DAGs are output-equivalent if they collide (see Fig. 1). From the above considerations, it is clear that structures that are *output-equivalent* cannot be linearly separable. In the following, we assume that structures belonging to different classes cannot be *output-equivalent*.

Even in this case it is not difficult to define l.s.: it suffices to extend the existential quantifier in Def. 1 to the whole set of recursive weights. Results similar to the ones reported in Theorems 2 and 3 can trivially be obtained, as well as Theorem 4 conditions *i)* and *ii)*. Moreover, condition *iii)* of Theorem 4 can be modified in the following way

Theorem 5. *Let* $\boldsymbol{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_m\}$ *be a set of DAGs, and* $\mathcal{B}_{\boldsymbol{U}} = \{\boldsymbol{\nu}^{\mathcal{U}_1}(d_j), \ldots, \boldsymbol{\nu}^{\mathcal{U}_m}(d_j)\}$*, with*

$$\boldsymbol{\nu}^{\mathcal{U}_j}(d_j) = \sum_{v \in \mathcal{L}_{\mathcal{U}_j}(d_j)} \boldsymbol{u}_v,$$

where d_j *is the maximum depth of* $\mathcal{U}_j$*, and* $\mathcal{L}_{\mathcal{U}_j}(d_j)$ *is the set of nodes in* $\mathrm{vert}(\mathcal{U}_j)$ *at level* d_j*, then* $\boldsymbol{U}$ *is l.s. if* $\mathcal{B}_{\boldsymbol{U}}$ *is l.s.*

Proof: The proof is obtained by observing that considering $\hat{w}_1 = \hat{w}_2 = \cdots = \hat{w}_c = w$, each DAG $\mathcal{U}_j$ returns an output that is equal to the one generated by a sequence with labels $\boldsymbol{\nu}^{\mathcal{U}_j}(0), \ldots, \boldsymbol{\nu}^{\mathcal{U}_j}(d_j)$ and processed by a neuron for sequences with weight w. By applying Theorem 4, condition *iii)*, the Theorem is proved. $\square$

6 Conclusion

We proposed a definition of linear separability for sequences and structures with the aim to characterize classification tasks in structured domains which are "easy" to solve. We have shown how results for vectors can be extended to structured domains. However, much work need to be done in defining efficient algorithms for solving problems which are linearly separable in structured domains. For example, when considering sequences with label domain equal to $\mathbb{R}$, a solution to a classification problem can be found by computing the roots of the polynomials used to encode each sequence and checking whether there is at least one value for which all the polynomials are positive (provided that all the occurrences of labels in negative examples are changed of sign). A nice property of this algorithm is that it can decide about the linear separability of the training set and that the accuracy used in computing the roots can be adapted to the "amount" of separability of the sequences.

When considering a label domain equal to $\mathbb{R}^r$, $r > 1$, however, the problem becomes more difficult and no obvious algorithm seems to exist. Even more difficult is the situation in which structures are considered; in fact, each structure, in the proposed framework, is encoded by a multi-dimensional multi-varied polynomial.

References

1. M. Bianchini, M. Gori, and F. Scarselli. Theoretical properties of recursive neural networks with linear neurons. *IEEE Transactions on Neural Networks*, 12(5):953–967, 2001.
2. J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
3. S. E. Fahlman. The recurrent cascade-correlation architecture. Technical Report CMU-CS-91-100, Carnegie Mellon, 1991.
4. P. Frasconi, M. Gori, and A. Sperduti. A framework for adaptive data structures processing. *IEEE Transactions on Neural Networks*, 9(5):768–786, 1998.
5. K.-Y. Siu, V. Roychowdhury, and T. Kailath. *Discrete Neural Computation*. Englewood Cliffs, New Jersey: Prentice Hall, 1995.
6. A. Sperduti, D. Majidi, and A. Starita. Extended cascade-correlation for syntactic and structural pattern recognition. In Petra Perner, Patrick Wang, and Azriel Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition*, volume 1121 of *Lecture notes in Computer Science*, pages 90–99. Springer-Verlag, Berlin, 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. A. Sperduti, A. Starita, and C. Goller. Learning distributed representations for the classification of terms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 509–515, 1995.
9. R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.