IDIAP RESEARCH REPORT

IDIAP

Martigny - Valais - Suisse

On the Complexity of Recognizing Iterated Differences of Polyhedra

Eddy N. Mayoraz †

IDIAP–RR 97-10

October 1997

PUBLISHED IN

Proceedings of ICANN'97, Lausanne, October 1997. W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (Eds) Lecture Notes in Computer Science 1327, 475-480

Dalle Molle Institute for Perceptive Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

† IDIAP—Dalle Molle Institute for Perceptive Artificial Intelligence, P.O.Box 592, CH-1920 Martigny, Valais, Switzerland Eddy.Mayoraz@idiap.ch

IDIAP Research Report 97-10

On the Complexity of Recognizing Iterated Differences of Polyhedra

Eddy N. Mayoraz

October 1997

PUBLISHED IN

Proceedings of ICANN'97, Lausanne, October 1997. W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (Eds) Lecture Notes in Computer Science 1327, 475-480

Abstract. The iterated difference of polyhedra $V = P_1 \backslash (P_2 \backslash (\ldots P_k) \ldots)$ has been proposed independently in [11] and [7] as a sufficient condition for V to be exactly computable by a two-layered neural network. An algorithm checking whether $V \subset I\!R^d$ is an iterated difference of polyhedra is proposed in [11]. However, this algorithm is not practically usable because it has a high computational complexity and it was only conjectured to stop with a negative answer when applied to a region which is not an iterated difference of polyhedra. This paper sheds some light on the nature of iterated difference of polyhedra. The outcomes are : (i) an algorithm which always stops after a small number of iterations, (ii) sufficient conditions for this algorithm to be polynomial and (iii) the proof that an iterated difference of polyhedra can be exactly computed by a two-layered neural network using only essential hyperplanes.

1 Introduction

Several papers have been lately devoted to the problem of characterizing the regions of the Euclidian space $I\!R^d$ that can be computed by a depth-2 multilayer perceptron (MLP), *i.e.* an MLP with d real inputs, one hidden layer of linear threshold processing units and a single output with a linear threshold processing unit [4, 2, 10, 3, 8, 1]. Different variations of the problem are considered : the function of the MLP and the characteristic function of the region are required to match either (i) *exactly* [10], *i.e.* for any $\boldsymbol{x} \in I\!R^d$, or (ii) *almost everywhere* [3, 1], *i.e.* everywhere but on a set of measure 0; or even (iii) *up to* ϵ [8], *i.e.* for any $\boldsymbol{x}$ at distance more than ϵ with the border of V.

In what follows we will denote by $\mathcal{LP}_2$ the set of regions V which are computable by depth-2 MLPs and if nothing is specified, *exact* computation will be intended.

Simultaneously to the characterization of these regions V in $\mathcal{LP}_2$, another important issue is the complexity of the MLP computing $V \in \mathcal{LP}_2$, which is essentially expressed by its number of hidden units. This question did not get as much as attention as the characterization, although it is crucial for a practical usage of any other results. It turns out that even very simple regions $V \in \mathcal{LP}_2$ seem to require a tremendous amount of hidden units. If we denote by $H_{\boldsymbol{h}}^{h_0}$ the closed halfspace $\{\boldsymbol{x} \mid \boldsymbol{x}^\top \boldsymbol{h} \geq h_0\}$ and if $\Delta > 1$ is a positive integer, consider the region

$$V = (\ H^1_{(1,1)} \cap H^1_{(-1,1)} \cap H^{-\Delta}_{(0,-1)}\ ) \ \cup\ (\ H^1_{(1,-1)} \cap H^1_{(-1,-1)} \cap H^{-\Delta}_{(0,1)}\ ) \tag{1}$$

which is known to be in $\mathcal{LP}_2$ [9, 3]. To the best of our knowledge, any solution known for the computation of V with a depth-2 MLP requires a number of hidden units growing linearly with Δ, *i.e.* exponentially with the size of the instance V which is in $O(\log(\Delta))$. This simple example gives us some faith in the following conjecture :

> **Conjecture 1** There exists a region $V \subset I\!R^2$ in $\mathcal{LP}_2$ such that any depth-2 MLP computing V almost everywhere has a number of hidden units exponential in the size of a compact encoding of V.

Let us assume that a region V — the instance of the problem — is specified by a finite list of closed halfspaces, called *basis* of V, and an expression of V as a union of intersections of some of these halfspaces or their complements (*e.g.* equation (1)).

A region V can have, in general, different minimal bases (in the sense of inclusion). A halfspace is called *essential* to V if it belongs to any basis of V. If $V \subset I\!R^d$ is a union of intersections of finitely many halfspaces and each of these intersections is fully dimensioned (*i.e.* containing one open ball of dimension d), it can be easily verified that V has a unique minimal basis, denoted $\mathcal{H}_V$, which is the set of essential halfspaces. Thus in what follows, if no particular basis is specified for a region V, full dimension of any component of V is implicitly assumed, and the basis of reference is $\mathcal{H}_V$.

The complexity problem raised in Conjecture 1 incites us to focus on a subclass of $\mathcal{LP}_2$, denoted $\overline{\mathcal{LP}}_2$, defined as the set of regions V computable by a depth-2 MLP where the hidden units are computing only essential halfspaces. Two major issues should be addressed :

Q1 find a geometrical characterization of $\overline{\mathcal{LP}}_2$,

Q2 given a basis $\mathcal{H}$ and a region V defined as a union of intersections of some halfspaces and complements of halfspaces in $\mathcal{H}$, what is the complexity of deciding whether $V \in \overline{\mathcal{LP}}_2$.

In [6] we identified Q2 as co-*NP*-Complete. In the present work, we study the class of iterated differences of polyhedra, proposed simultaneously in [11, 7] as a subclass of $\mathcal{LP}_2$. In the rest of this paper, we first recall what has been done in this field, present an efficient algorithm for recognizing the iterated difference of polyhedra and discuss its consequences.

2 Iterated difference of polyhedra

Definition 2.1 *polyhedron* (resp. *pseudo-polyhedron*) is an intersection of finitely many closed (resp. open or closed) halfspaces. A region $V \subset \mathbb{R}^d$ is an *iterated difference of polyhedra* (resp. pseudo-polyhedra) if it can be expressed as $V = P_1 \backslash (P_2 \backslash (\ldots P_k) \ldots)$, where each P_i, $i = 1, \ldots, k$, is a polyhedron (resp. pseudo-polyhedron). The class of iterated differences of polyhedra (resp. pseudo-polyhedra) is denoted $\mathcal{D}$ (resp. $\widetilde{\mathcal{D}}$).

Proposition 2 $\mathcal{D} \subset \widetilde{\mathcal{D}} \subsetneq \mathcal{LP}_2$.

Proof: The first inclusion is obvious. The proof of the second inclusion is based on the fact that $P \backslash V \in \mathcal{LP}_2$ for any pseudo-polyhedron P and any $V \in \mathcal{LP}_2$ (see [11, 7]). $\triangle$

In [11], the authors propose the following algorithm for the recognition of $\mathcal{D}$:

```
input:          V ⊂ ℝ^d;
initialization: V_0 := V; l := 0;
main loop:      while V_l ≠ ∅ and ( l < 2 or else P_l ≠ P_{l-1} ) loop
                    l := l + 1;
                    P_l := op(V_{l-1});
                    V_l := P_l \ V_{l-1};
                end loop
output:         P_1\(P_2\(... P_{l-1}\(P_l\V_l)) ...)  =:  V
```

Algo(op) : Recognition of iterated differences of polyhedra.

The operator "op" stands for the closure of the convex hull, denoted $\overline{\text{conv}}$. The authors proved that $V \in \mathcal{D}$ *iff* Algo($\overline{\text{conv}}$) stops with $V_l = \emptyset$. However, they only conjectured that Algo($\overline{\text{conv}}$) could not cycle, or in other words, that if $V \notin \mathcal{D}$, it would stop with $P_l = P_{l-1} \neq \emptyset$.

At a first glance, one might believe that choosing "op" simply as the convex hull would lead to an algorithm Algo(conv) for the recognition of $\widetilde{\mathcal{D}}$, but as mentioned by the authors, the convex hull of the difference between two pseudo-polyhedra is not necessarily a pseudo-polyhedron (see Figure 2 in [11]). Moreover, with Algo($\overline{\text{conv}}$) in mind we cannot conclude that $\mathcal{D} \subset \overline{\mathcal{LP}}_2$, since the computation of the convex hull will add non essential halfspaces. Finally, the main weakness of Algo($\overline{\text{conv}}$) is its complexity, given that

- there is no proof that it always stops,
- even if $V \in \mathcal{D}$, there is no bound on the number of iterations,
- the computation of the convex hull is exponential in d.

Starting from this basis, the only contribution of this paper is the suggestion of a more appropriate operator "op" which will solve very simply each of the problems mentioned above.

3 The hull operator

Definition 3.1 iven a collection $\mathcal{E}$ of regions of $\mathbb{R}^d$, the operator $\text{hull}_{\mathcal{E}}$ is defined as follows :

$$\forall X \subset \mathbb{R}^d, \quad \text{hull}_{\mathcal{E}}(X) = \bigcap_{E \in \mathcal{E},\ E \supset X} E\ .$$

In order to illsutrate the relation between "hull" and "conv", let $\mathcal{C}$ denote the set of all closed halfspaces, $\widetilde{\mathcal{C}}$ the set of all halfspaces (closed and open), and X^{int} the interior of a set X (according to the usual topology of $\mathbb{R}^d$). In [5] we have established that for any $X \subset \mathbb{R}^d$,

$$\text{conv}^{\text{int}}(X) = \text{hull}_{\mathcal{C}}^{\text{int}}(X) \subset \text{conv}(X) \subset \text{hull}_{\widetilde{\mathcal{C}}}(X) \subset \overline{\text{conv}}(X) = \text{hull}_{\mathcal{C}}(X).$$

Consequently, Algo($\text{hull}_{\mathcal{C}}$) is identical to Algo($\overline{\text{conv}}$). Moreover, $\text{hull}_{\widetilde{\mathcal{C}}}$ does not suffer from the same drawback as "conv" towards pseudo-polyhedra in the sense that $\text{hull}_{\widetilde{\mathcal{C}}}(P_i \backslash P_j)$ is a pseudo-polyhedron for any pseudo-polyhedra P_i and P_j. Therefore, the whole work in [11] can be restated using $\text{hull}_{\widetilde{\mathcal{C}}}$ instead of $\overline{\text{conv}}$ and Proposition 3 will follow.

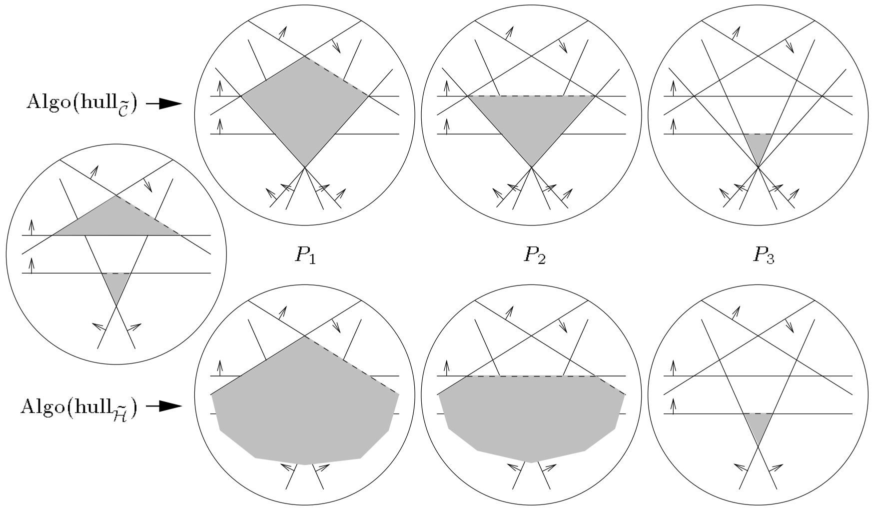

Figure 1: Comparison of Algo(hull$_{\widetilde{\mathcal{C}}}$) and Algo(hull$_{\widetilde{\mathcal{H}}}$).
Each halfplane is indicated by a line (border) and an arrow (pointing toward the halfplane). The halfplanes shown in Figure V constitute the basis of V. Dashed lines denote open faces of gray regions. Algo(hull$_{\widetilde{\mathcal{C}}}$) adds two halfplanes to solve the problem, while Algo(hull$_{\widetilde{\mathcal{H}}}$) uses only the basis.

Proposition 3 Algo(hull$_{\widetilde{\mathcal{C}}}$) recognizes exactly $\widetilde{\mathcal{D}}$.

However, by exploiting the hull operator a bit further, we will get a much simpler algorithm for the recognition of $\widetilde{\mathcal{D}}$.

4 Main result

Let V be an arbitrary region of $I\!R^d$ and $\mathcal{H}$ a basis of V. Let $\widetilde{\mathcal{H}}$ be defined as $\{H \mid H \in \mathcal{H} \text{ or } I\!R^d \backslash H \in \mathcal{H}\}$.

Theorem 4 Algo(hull$_{\widetilde{\mathcal{H}}}$) recognizes exactly $\widetilde{\mathcal{D}}$.

The proof of this theorem is too long to be presented here and can be found in [6]. Instead, we will try to give an idea of why this is true and we will enumerate the consequences of this result.

For a simple region $V \in I\!R^2$, Figure 1 illustrates the two different sequences of pseudo-polyhedra produced by Algo(hull$_{\widetilde{\mathcal{C}}}$) and by Algo(hull$_{\widetilde{\mathcal{H}}}$), where $\mathcal{H}$ is just $\mathcal{H}_V$.

Corollary 5 Any region $V \subset I\!R^d$ that can be expressed as an arbitrary iterated difference of pseudo-polyhedra can also be expressed as an iterated difference of pseudo-polyhedra $P_1 \backslash (P_2 \backslash (\ldots P_l) \ldots)$ where each $P_i, i = 1, \ldots, l$ is an intersection of halfspaces and/or complement of halfspaces, all taken from a basis of V fixed *a priori*.

Proof: For the desired basis $\mathcal{H}$ of V, simply run Algo(hull$_{\widetilde{\mathcal{H}}}$) on the input V to get the P_is. $\triangle$

Proposition 2 can be improved as follows:

Corollary 6 $\mathcal{D} \subsetneq \widetilde{\mathcal{D}} \subsetneq \overline{\mathcal{LP}_2} \subsetneq \mathcal{LP}_2$.

Proof: Let V by a 2-dimensional square with two opposite edges closed, the other two edges open, and without its corners. V is a pseudo-polyhedron but it is not in $\mathcal{D}$ since Algo($\overline{\text{conv}}$) when run on V stops with $V_2 = V_0 \neq \emptyset$. Thus $\mathcal{D}$ is a proper subset of $\widetilde{\mathcal{D}}$. The last inclusion is obvious and the region V given in (1) with $\Delta > 2$ shows that it is a proper inclusion.

The inclusion $\widetilde{\mathcal{D}} \subsetneq \overline{\mathcal{LP}_2}$ follows from the fact that if $\mathcal{H}$ is a basis of V and if P is a pseudo-polyhedron whose basis is a subset of $\mathcal{H}$, then $V \in \overline{\mathcal{LP}_2}$ implies $P \backslash V \in \overline{\mathcal{LP}_2}$. The proof of the latter result follows easily when the geometrical problem is transposed into a Boolean problem (see [6]). Finally, the Swiss flag provides a region which is in $\overline{\mathcal{LP}_2} \backslash \widetilde{\mathcal{D}}$. △

Proposition 7 Algo(hull$_{\widetilde{\mathcal{H}}}$) stops after at most $|\widetilde{\mathcal{H}}|$ steps.

Proof: At iteration l of Algo(hull$_{\widetilde{\mathcal{H}}}$), let $\widetilde{\mathcal{H}}_l$ denote the set of halfspaces $H \in \widetilde{\mathcal{H}}$ such that H is either essential to P_l or its supporting hyperplane intersects P_l^{int}. The proposition follows from the observation that $\widetilde{\mathcal{H}} = \widetilde{\mathcal{H}}_1 \supset \ldots \supset \widetilde{\mathcal{H}}_l$ and that all these inclusions are proper. △

Finally, let us consider the complexity of Algo(hull$_{\widetilde{\mathcal{H}}}$). For V given as a union of s pseudo-polyhedra, the computation of hull$_{\widetilde{\mathcal{H}}}(V)$ requires that for each halfspace $H \in \widetilde{\mathcal{H}}$ and each of the s components of V, we check whether this component P is contained in H or in $\mathbb{R}^d \backslash H$. This is done by testing whether $P \cap (\mathbb{R}^d \backslash H)$ or $P \cap H$ is empty. It requires to check the non feasibility of a system of inequalities, which can be done by linear programming in a time polynomial in the number of inequalities (at most $|\widetilde{\mathcal{H}}|$) and the number d of variables. Thus the overall computation of hull$_{\widetilde{\mathcal{H}}}(V)$ is polynomial in d, s and $|\widetilde{\mathcal{H}}|$.

Even though we replaced the costly convex hull operator by hull$_{\widetilde{\mathcal{H}}}$ working in polynomial time, and we have a linear bound on the number of steps of the algorithm, the recognition of $\widetilde{\mathcal{D}}$ is a *NP*-Complete problem [6]. The complexity is in the computation of the difference of two sets $(P_l \backslash V_{l-1})$. If V is given as a union of pseudo-polyhedra (this expression corresponds to a Disjunctive Normal Form, in Boolean terminology), to get $P \backslash V$ we need to complement V, which is hard in general (dualization of an arbitrary DNF). If both V and $\mathbb{R}^d \backslash V$ are available as unions of intersections of pseudo-polyhedra, Algo(hull$_{\widetilde{\mathcal{H}}}$) can be slightly modified so that it avoids any calculation of complements.

Proposition 8 If expressions as unions of pseudo-polyhedra are available for both V and $\mathbb{R}^d \backslash V$, the recognition of $\widetilde{\mathcal{D}}$ can be solved in polynomial time.

Acknowledgments

The author is thankful to the Swiss National Science Foundation for its support of this research and to Dr Motakuri Ramana for very helpful discussions and his active participation in the proof of Theorem 4.

References

[1] G. Brightwell, C. Kenyon, and H. Paugam-Moisy. Multilayer neural networks : one or two hidden layers ? Technical Report NC-TR-97-001, NeuroCOLT Technical Report Series, 1997.

[2] Gavin J. Gibson and Colin F. N. Cowan. On the decision regions of multilayer perceptrons. *Proceedings of the IEEE*, 78(10):1590–1594, 1990.

[3] Gavin J. Gibson. Exact classification with 2-layered nets. *J. Computer and System Sciences*, 1996.

[4] R. P. Lippmann. An introduction to computing with neural nets. *IEEE, Acoustic, Speech and Signal Processing*, 4:4–22, 1987.

[5] Eddy Mayoraz. On variations of the convex hull operator. IDIAP-RR 96-06, IDIAP, 1996. `ftp://ftp.idiap.ch/pub/mayoraz/Publications/RRR-06-96.ps.gz`.

[6] Eddy Mayoraz. On the complexity of recognizing regions of $\mathbb{R}^d$ computable by a two-layered perceptron. IDIAP-RR 97-05, IDIAP, 1997. `ftp://ftp.idiap.ch/pub/mayoraz/Publications/RRR-05-97.ps.gz`.

[7] Ron Shonkwiler. Separating the vertices of n-cubes by hyperplanes and its application to artificial neural networks. *IEEE Transactions on Neural Networks*, 4(2):343–347, March 1993.

[8] Catherine Z. W. Hassell Sweatman, Gavin J. Gibson, and Bernard Mulgrew. Exact classification with two-layer neural nets in n dimensions. submitted, 1996.

[9] Patrick J. Zwietering. *The Complexity of Multi-Layered Perceptrons*. PhD thesis, University of Eindhoven, 1994.

[10] P. J. Zwietering, E. H. L. Aarts, and J. Wessels. The design and complexity of exact multilayered perceptrons. *International Journal of Neural Systems*, 2:185–199, 1991.

[11] P. J. Zwietering, E. H. L. Aarts, and J. Wessels. Exact classification with two-layered perceptrons. *International Journal of Neural Systems*, 3(2):143–156, 1992.