Semialgebraic Optimization for Lipschitz Constants of ReLU Networks

Tong Chen  
LAAS-CNRS  
Université de Toulouse  
31400 Toulouse, France  
tchen@laas.fr

Jean-Bernard Lasserre  
LAAS-CNRS & IMT  
Université de Toulouse  
31400 Toulouse, France  
lasserre@laas.fr

Victor Magron  
LAAS-CNRS  
Université de Toulouse  
31400 Toulouse, France  
vmagron@laas.fr

Edouard Pauwels  
IRIT & IMT  
Université de Toulouse  
31400 Toulouse, France  
edouard.pauwels@irit.fr

Abstract

The Lipschitz constant of a network plays an important role in many applications of deep learning, such as robustness certification and Wasserstein Generative Adversarial Network. We introduce a semidefinite programming hierarchy to estimate the global and local Lipschitz constant of a multiple layer deep neural network. The novelty is to combine a polynomial lifting for ReLU functions derivatives with a weak generalization of Putinar’s positivity certificate. This idea could also apply to other, nearly sparse, polynomial optimization problems in machine learning. We empirically demonstrate that our method provides a trade-off with respect to state of the art linear programming approach, and in some cases we obtain better bounds in less time.

1 Introduction

We focus on the multiple layer networks with ReLU activations. We propose a computationally efficient method to give a valid upper bound on the Lipschitz constant of such networks. Recall that a function \( f \), defined on a convex set \( X \subseteq \mathbb{R}^n \), is \( L \)-Lipschitz with respect to the norm \( \| \cdot \| \) if for all \( x, y \in X \), we have \( |f(x) - f(y)| \leq L \| x - y \| \). The Lipschitz constant of \( f \) with respect to norm \( \| \cdot \| \), denoted by \( L^\| \cdot \| f \), is the infimum of all those valid \( L \):

\[
L^\| \cdot \| f := \inf \{ L : \forall x, y \in X, |f(x) - f(y)| \leq L \| x - y \| \}.
\]  

(1)

For deep networks, they play an important role in many applications related to robustness certification which has emerged as an active topic. See recent works \([9, 31]\) based on semidefinite programming (SDP), \([8, 40]\) based on linear programming (LP), \([34]\) based on mixed integer programming (MIP), and \([6, 38, 39, 41]\) based on outer polytope approximation. \([7]\) based on averaged activation operators. We follow a different route and compute upper bounds on the Lipschitz constant of neural networks \([35]\).

Another important application is the Wasserstein Generative Adversarial Network (WGAN) \([3]\). Wasserstein distance is estimated by using the space of functions encoded by 1-Lipschitz neural networks. This requires a precise estimation of the Lipschitz constants, see recent contributions \([2, 11, 25]\).

Recently there has been a growing interest in polynomial optimization for such problems. In [31], robustness certification is modeled as a quadratically constrained quadratic problem (QCQP) for ReLU networks. Similarly, in [21], an upper bound on the Lipschitz constant of an ELU network is obtained from a polynomial optimization problem (POP). In contrast to optimization problems with more general functions, powerful (global) positivity certificates are available for POP. Such certificates are needed to approximate global optima as closely as desired [20].

Such positivity certificates have been already applied with success in various areas of science and engineering. The first attempt to compute lower bounds of a QCQP by solving an SDP can be traced back to Shor [32], recently applied to certify robustness of neural networks in [31]. Converging LP based hierarchies are based on Krivine-Stengle’s certificates [15] [19] [39]. In [21], sparse versions are used to bound the Lipschitz constant of neural networks. On the other hand, Putinar’s certificate [17] [29] is implemented via an SDP-based hierarchy (a.k.a., “Lasserre’s hierarchy”) and provides converging approximate solutions of a POP. Various applications are described in [16], see also its application to large scale optimal power flow problems in [26] [27] and for roundoff error certification in [24]. The LP hierarchy is cheaper than the SDP hierarchy, but less efficient for combinatorial optimization [22], and cannot converge in finitely many steps for continuous POPs. Finally, weaker positivity certificates can be used, for example DSOS/SDSOS [1] based on second-order cone programming, or hybrid BSOS hierarchy [19].

1.1 Related Works

Upper bound on Lipschitz constants of deep networks can be obtained by a product of the layer-wise Lipschitz constants [13]. This is however extremely loose and has many limitations [13]. Note that [56] propose an improvement via a finer product.

Departing from this approach, [21] propose a QCQP formulation to estimate the Lipschitz constant of neural networks. Shor’s relaxation allows to obtain a valid upper bound. Alternatively, using the LP hierarchy, [21] obtain tighter upper bounds. By another SDP-based method, [10] provide an upper bound of the Lipschitz constant. However this method is restricted to the $L_2$-norm whereas most robustness certification problems in deep learning are rather concerned with the $L_{\infty}$-norm.

1.2 Preliminaries and Notations

Denote by $F$ the multiple layer neural network, $m$ the number of hidden layers, $p_0, p_1, \ldots, p_m$ the number of nodes in the input layer and each hidden layer. For simplicity, $(p_0, p_1, \ldots, p_m)$ will denote the layer structure of network $F$. Let $x_0$ be the initial input, and $x_1, \ldots, x_m$ be the activation vectors in each hidden layer. Each $x_i, i = 1, \ldots, m$, is obtained by a weight $A_i$, a bias $b_i$, and an activation function $\sigma$, i.e., $x_i = \sigma(A_i x_{i-1} + b_i)$. We only consider coordinatewise application of the ReLU activation function, defined as $\text{ReLU}(x) = \max \{0, x\}$ for $x \in \mathbb{R}$. The ReLU function is non-smooth, we define its generalized derivative as the set-valued function $G(x)$ such that $G(x) = 1$ for $x > 0$, $G(x) = 0$ for $x < 0$ and $G(x) = \{0, 1\}$ for $x = 0$.

We assume that the last layer in our neural network is a softmax layer with $K$ entries, that is, the network is a classifier for $K$ labels. For each label $k \in \{1, \ldots, K\}$, the score of label $k$ is obtained by an affine product with the last activation vector, i.e., $c_k^T x_m$ for some $c_k \in \mathbb{R}^m$. The final output is the label with the highest score, i.e., $y = \arg \max_k c_k^T x_m$. The product $xy$ of two vectors $x$ and $y$ is considered as coordinate-wise product.

1.3 Contribution

- We first express both graphs of ReLU and its generalized derivative via basic closed semialgebraic sets, i.e., sets defined with finite conjunctions of polynomial (in)equalities. Indeed, if $y = \text{ReLU}(x)$ and $v \in G(x)$, then equivalently: $y(y - x) = 0, y \geq x, y \geq 0$ and $v(v - 1) = 0, (v - 1/2)x \geq 0$. Note that the explicit semialgebraic expression of ReLU has already been used in related works, see for example [31]. The semialgebraic expression for $G$ is our contribution. Being exact, this semi-algebraic reformulation is a noticeable improvement compared to the model proposed in [21] where the generalized derivative of ReLU is simply replaced with a decision variable lying between 0 and 1 (and so is only approximated).
Table 1: Global Lipschitz constant and solver running time of networks of size (80, 80) and (40, 40, 10) obtained by HR-1, HR-2, LipOpt-3 and LipOpt-4 on various sparsities $s$. The abbreviation “HR-2” (resp. “HR-1”) stands for the second-order (resp. first-order) SDP-based method we propose (see Appendix E), and “LipOpt-3/4” stands for the LP-based method from [21] (which uses Krivine-Stengle’s positivity certificate of degree 3 or 4). LBS is the lower bound computed by random sampling. For networks of more than 2 hidden layers, we use the technique introduced in Appendix E in order to deal with the cubic terms in the objective. OfM means out of memory while building the model. The results are for a single random network, complete results are shown in Appendix F and G.

<table>
<thead>
<tr>
<th>Network Size</th>
<th>HR-2</th>
<th>HR-1</th>
<th>LipOpt-3</th>
<th>LipOpt-4</th>
<th>LBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(80, 80)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s = 20$</td>
<td>1.45</td>
<td>0.50</td>
<td>0.56</td>
<td>0.29</td>
<td>0.20</td>
</tr>
<tr>
<td>$s = 40$</td>
<td>2.05</td>
<td>1.16</td>
<td>1.68</td>
<td>0.85</td>
<td>0.48</td>
</tr>
<tr>
<td>$s = 60$</td>
<td>2.41</td>
<td>1.82</td>
<td>3.01</td>
<td>0.79</td>
<td>0.61</td>
</tr>
<tr>
<td>$s = 80$</td>
<td>2.68</td>
<td>2.05</td>
<td>4.68</td>
<td>3.57</td>
<td>0.62</td>
</tr>
<tr>
<td>(40, 40, 10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s = 20$</td>
<td>271.34</td>
<td>271.34</td>
<td>165.68</td>
<td>174.36</td>
<td>174.32</td>
</tr>
<tr>
<td>$s = 40$</td>
<td>3.14</td>
<td>3.14</td>
<td>10.36</td>
<td>71.49</td>
<td>71.49</td>
</tr>
<tr>
<td>$s = 60$</td>
<td>7.78</td>
<td>7.78</td>
<td>20.99</td>
<td>20.99</td>
<td>20.99</td>
</tr>
<tr>
<td>$s = 80$</td>
<td>8.61</td>
<td>8.61</td>
<td>71.49</td>
<td>71.49</td>
<td>71.49</td>
</tr>
</tbody>
</table>

• Second, we provide a heuristic approach based on an SDP-hierarchy for nearly sparse polynomial optimization problems. In such problems one assumes that only a few affine constraints destroy a sparsity pattern satisfied by the other constraints. This new approach is mainly based on the sparse version of the Lasserre’s hierarchy, popularized in [18, 37]. It provides upper bounds yielding a trade-off between third and fourth degree LP relaxations in [21], and sometimes a strict improvement.

1.4 Main Results

In recent work [21] a certain sparsity structure arising from a neural network is exploited. Consider a neural network $F$ with one single hidden layer, and 4 nodes in each layer. The network $F$ is said to have a sparsity of 4 if its weight matrix $A$ is symmetric with diagonal blocks of size at most $2 \times 2$:

$$
\begin{pmatrix}
* & * & 0 & 0 \\
* & * & * & 0 \\
0 & * & * & * \\
0 & 0 & * & *
\end{pmatrix}
$$

(2)

Larger sparsity values refer to symmetric matrices with band structure of a given size. This sparsity structure (2) of the networks greatly influences the number of variables involved in the LP program to solve in [21]. This is in deep contrast with our method which does not require the weight matrix to be as in (2). Hence when the network is fully-connected, our method is more efficient and provides tighter upper bounds.

Table 1 gives a brief comparison outlook of the results obtained by our method and the method in [21]. For (80, 80) networks, apart from $s = 20$, which is not significative, HR-2 obtains much better bounds and is also much more efficient than LipOpt-3. LipOpt-4 provides tighter bounds than HR-2 but suffers more computational time, and run out of memory when the sparsity increases. For (40, 40, 10) networks, HR-1 is a trade-off between LipOpt-3 and LipOpt-4, it provides tighter (resp. looser) bounds than LipOpt-3 (resp. LipOpt-4), but takes more (resp. less) computational time.

2 Problem Setting

In this section, we recall basic facts about optimization and build the polynomial optimization model for estimating Lipschitz constant of neural networks.

2.1 Polynomial Optimization

In a polynomial optimization problem (POP), one computes the global minimum (or maximum) of a multivariate polynomial function on a basic closed semialgebraic set. If the semialgebraic set is the whole space, the problem is unconstrained, and constrained otherwise. Given a positive integer $n \in \mathbb{N}$, let $x = (x_1, \ldots, x_n)^T$ be a vector of decision variables, and denote by $[n]$ the set $\{1, \ldots, n\}$.
A POP has the canonical form:

$$\inf_{x \in \mathbb{R}^n} \{ f(x) : f_i(x) \geq 0, i \in [p]; g_j(x) = 0, j \in [q] \},$$

where $f, f_i, g_j$ are all polynomials in $n$ variables. With $I \subset \{1, \ldots, n\}$, let $x_I := (x_i)_{i \in I}$ and let $\mathbb{R}[x]$ be the space of real polynomials in the variables $x$ while $\mathbb{R}[x_I]$ is the space of real polynomials in the variables $x_I$.

In particular, if the objective $f$ and constraints $f_i, g_j$ in [POP] are all of degree at most 2, we say that the problem is a quadratically constrained quadratic problem (QCQP). The Shor’s relaxation of a QCQP is a semidefinite program which can be solved efficiently numerically. If all polynomials involved in [POP] are affine then the problem is a linear program (LP).

### 2.2 Lipschitz Constant Estimation Problem (LCEP)

Suppose we train a neural network $F$ for $K$-classifications and denote by $A_i, b_i, c_k$ its parameters already defined in section 1.2. Thus for an input $x_0 \in \mathbb{R}^n$, the targeted score of label $k$ can be expressed as $F_k(x_0) = c_k^T x_m$, where $x_i = \text{ReLU}(A_i x_{i-1} + b_i)$, for $i \in [m]$. Let $z_i = A_i x_{i-1} + b_i$ for $i \in [m]$. By applying the chain rule on the non-smooth function $F_k$, we obtain a set valued map for $F_k$ at point any $x_0$ as $G_{F_k}(x_0) = (\prod_{i=1}^m A_i^T \text{diag}(G(z_i))) c_k$.

We fix a targeted label (label 1 for example) and omit the symbol $k$ for simplicity. We define $L_F^{||\cdot||}$ of $F$ with respect to norm $|| \cdot ||$, is the supremum of the gradient’s dual norm, i.e.:

$$L_F^{||\cdot||} = \sup_{x_0 \in \Omega, \ v \in G_{F_k}(x_0)} ||v|| = \sup_{x_0 \in \Omega} \left\| \prod_{i=1}^m A_i^T \text{diag}(G(z_i)) c \right\|_s,$$

where $\Omega$ is the convex input space, and $|| \cdot ||_s$ is the dual norm of $|| \cdot ||$, which is defined by $||x||_s := \sup_{||t||_1 \leq 1} ||t, x||$ for all $x \in \mathbb{R}^n$. In general, the chain rule cannot be applied to composition of non-smooth functions \[s[14]. Hence the formulation of $G_{F_k}$ and \[3 may lead to incorrect gradients and bounds on the Lipschitz constant of the networks. The following ensures that this is not the case and that the approach is sound, its proof is postponed to Appendix A.

**Lemma 1** If $\Omega$ is convex, then $L_F^{||\cdot||}$ is a Lipschitz constant for $F_k$ on $\Omega$.

When $\Omega = \mathbb{R}^n$, $L_F^{||\cdot||}$ is the global Lipschitz constant of $F$ with respect to norm $|| \cdot ||$. In many cases we are also interested in the local Lipschitz constant of a neural network constrained in a small neighborhood of a fixed input $x_0$. In this situation the input space $\Omega$ is often the ball around $x_0 \in \mathbb{R}$ with radius $\varepsilon$: $\Omega = \{x : ||x - x_0|| \leq \varepsilon\}$. In particular, with the $L_{\infty}$-norm (and using $1 \leq x \leq u \Leftrightarrow (x-1)(x-u) \leq 0$), the input space $\Omega$ is the basic semialgebraic set:

$$\Omega = \{x : ||x - x_0 + \varepsilon|| \leq \varepsilon\}.$$

Combining Lemma 1 and 3, the Lipschitz constant estimation problem (LCEP) for neural networks with respect to norm $|| \cdot ||$, is the following POP:

$$\max_{x_i, u_i, k} \left\{ t^T \left( \prod_{i=1}^m A_i^T \text{diag}(u_i) \right) c : u_i (u_i - 1/2)(A_i x_{i-1} + b_i) \geq 0, i \in [m]; \right.$$

$$x_i - 1(x_{i-1} - A_i x_{i-2} - b_i) = 0, x_{i-1} \geq 0, x_i \geq A_i x_{i-2} + b_{i-1}, 2 \leq i \leq m; \right.$$

$$t^T \leq 1, (x_0 - x_0 + \varepsilon)(x_0 - x_0 - \varepsilon) \leq 0. \}$$

**LCEP** treats an exact formulation of the generalized derivative of the ReLU function by exploiting its semialgebraic character.

### 3 Lasserre’s Hierarchy

In this section we briefly introduce the Lasserre’s hierarchy [17] which has already many successful applications in and outside optimization [20].
3.1 Convergent SDP Relaxations without Exploiting Sparsity [17]

In the Lasserre’s hierarchy for optimization one approximates the global optimum of the POP

\[ f^* = \inf_{x \in \mathbb{R}^n} \{ f(x) : g_i(x) \geq 0, i \in [p] \}, \]

(\text{Opt})

(where \( f, g_i \) are all polynomials in \( \mathbb{R}[x] \)), by solving a hierarchy of SDPs\(^1\) of increasing size. Equality constraints can also be taken into account easily. Each SDP is a semidefinite relaxation of (\text{Opt}) in the form:

\[ \rho_d = \inf_y \{ L_y(f) : L_y(1) = 1, M_d(y) \succeq 0, M_{d-\omega_i}(g_i,y) \succeq 0, i \in [p] \}, \]

(\text{MomOpt-d})

where \( \omega_i = [\deg(g_j)/2], y = (y_{\alpha})_{\alpha \in \mathbb{N}^d}, L_y : \mathbb{R}[x] \to \mathbb{R} \) is the so-called Riesz linear functional:

\[ f = \sum_{\alpha} f_\alpha x^\alpha \mapsto L_y(f) := \sum_{\alpha} f_\alpha y_{\alpha} \text{ with } f \in \mathbb{R}[x], \]

and \( M_d(y), M_{d-\omega_i}(g_i,y) \) are moment matrix and localizing matrix respectively; see [20] for precise definitions and more details. The semidefinite program (\text{MomOpt-d}) is the \( d \)-th order moment relaxation of problem (\text{Opt}). As a result, when \( K := \{ x : g_i(x) \geq 0, i \in [p] \} \) is compact, one obtains a monotone sequence of lower bounds \( \rho_d \) with the property \( \rho_d \uparrow f^* \) as \( d \to \infty \) under a certain technical Archimedean condition; the latter is easily satisfied by including a quadratic redundant constraint \( M - \|x\|^2 \succeq 0 \) in the definition of \( K \) (redundant as \( K \) is compact and \( M \) is large enough). At last but not least and interestingly, generically the latter convergence is finite [28]. Ideally, one expects an optimal solution \( y^* \) of (\text{MomOpt-d}) to be the vector of moments up to order \( 2d \) of the Dirac measure \( \delta_{x^*} \) at a global minimizer \( x^* \) of (\text{Opt}). See Appendix B.1 for an example to illustrate the core idea of Lasserre’s hierarchy.

3.2 Convergent SDP Relaxations Exploiting Sparsity [18, 37]

The hierarchy (\text{MomOpt-d}) is often referred to as dense Lasserre’s hierarchy since we do not exploit any possible sparsity pattern of the POP. Therefore, if one solves (\text{MomOpt-d}) with interior point methods (as all current SDP solvers do), then the dense hierarchy is limited to POPs of modest size. Indeed the \( d \)-th order dense moment relaxation (\text{MomOpt-d}) involves \((n+2d)/2d\) variables and a moment matrix \( M_d(y) \) of size \((n+2d)/2d\) at fixed \( d \). Fortunately, large-scale POPs often exhibit some structured sparsity patterns which can be exploited to yield a sparse version of (\text{MomOpt-d}), as initially demonstrated in [37]. As a result, wider applications of Lasserre’s hierarchy have been possible.

Assume that the set of variables in (\text{Opt}) can be divided into several subsets indexed by \( I_k \), for \( k \in [l] \), i.e., \([n] = \bigcup_{k=1}^l I_k \), and suppose that the following assumptions hold:

A1: The function \( f \) is a sum of polynomials, each involving variables of only one subset, i.e.,

\[ f(x) = \sum_{k=1}^l f_k(x_{I_k}); \]

A2: Each constraint also involves variables of only one subset, i.e., \( g_i \in \mathbb{R}[x_{I_k(i)}] \) for some \( k(i) \in [l] \);

A3: The subsets \( I_k \) satisfy the Running Intersection Property (RIP): for every \( k \in [l-1] \), \( I_{k+1} \cap \bigcup_{j=1}^k I_j \subseteq I_s \), for some \( s \leq k \),

A4: Add redundant constraints \( M_k - \|x_{I_k}\|^2 \succeq 0 \) where \( M_k \) are constants determined beforehand.

A POP with such a sparsity pattern is of the form:

\[ \inf_{x \in \mathbb{R}^n} \{ f(x) : g_i(x_{I_k(i)}) \geq 0, i \in [p] \}, \]

(\text{SpOpt})

and its associated sparse Lasserre’s hierarchy reads:

\[ \theta_d = \inf_y \{ L_y(f) : L_y(1) = 1, M_d(y, I_k) \succeq 0, k \in [l]; M_{d-\omega_i}(g_i,y, I_k(i)) \succeq 0, i \in [p] \}, \]

(\text{MomSpOpt-d})

where \( d, \omega_i, y, L_y \) are defined as in (\text{MomOpt-d}) but with a crucial difference. The matrix \( M_d(y, I_k) \) (resp. \( M_{d-\omega_i}(g_i,y, I_k(i)) \)) is a submatrix of the moment matrix \( M_d(y) \) (resp. localizing matrix

\(^1\)Semidefinite programming (SDP) is a subfield of convex conic optimization concerned with the optimization of a linear objective function over the intersection of the cone of positive semidefinite matrices with an affine subspace.
\(M_d - \omega_i (g, y)\) with respect to the subset \(I_k\), and hence of much smaller size \(\tau_k + d\) if \(|I_k| =: \tau_k \ll n\). See Appendix B.2 for an example to illustrate the core idea of sparse Lasserre’s hierarchy.

If the maximum size \(\tau\) of the subsets is such that \(\tau \ll n\), then solving \((\text{MomSpOpt-}d)\) rather than \((\text{MomOpt-}d)\) results in drastic computational savings. In fact, even with not so large \(n\), \((\text{MomOpt-}d)\) the second relaxation with \(d = 2\) is out of reach for currently available SDP solvers. Finally, \(\theta_d \leq f^*\) for all \(d\) and moreover, if the subsets \(I_k\) satisfy RIP, then we still obtain the convergence \(\theta_d \uparrow f^*\) as \(d \to \infty\), like for the dense relaxation \((\text{MomOpt-}d)\).

There is a primal-dual relation between the moment problem and the sum-of-square (SOS) problem, as shown in Appendix C. The specific MATLAB toolboxes Gloptipoly [12] and YALMIP [23] can solve the hierarchy \((\text{MomOpt-}d)\) and its sparse variant \((\text{MomSpOpt-}d)\), and also their dual SOS problem.

4 Heuristic Approaches

For illustration purpose, consider 1-hidden layer networks. Then in \((\text{LCEP})\) we can define natural subsets \(I_i = \{u^{(i)}_1, x_0\}, i \in [p]\) (w.r.t. constraints \(u_1 - (u_1 - 1/2)(A_4 x_0 + b_1) \geq 0\), and \((x_0 - x_0 + \varepsilon)(x_0 - x_0 - \varepsilon) \leq 0\); and \(I_j = \{t^{(j)}\}, j \in [p]\) (w.r.t. constraints \(t^2 \leq 1\)). Clearly, \(I_i, J_j\) satisfy the RIP condition and are subsets with smallest possible size. Recall that \(x_0 \in \mathbb{R}^{p_0}\). Hence \(|I_i| = 1 + p_0\) and the maximum size of the PSD matrices is \((1+p_0+d)\). Therefore, as in real deep neural networks \(p_0\) can be as large as 1000, the second-order sparse Lasserre’s hierarchy \((\text{MomSpOpt-}d)\) cannot be implemented in practice.

In fact \((\text{LCEP})\) can be considered as a “nearly sparse” POP, i.e., a sparse POP with some additional “bad” constraints that violate the sparsity assumptions. More precisely, suppose that \(f, g_i\) and subsets \(I_k\) satisfy assumptions \(A_1, A_2, A_3\) and \(A_4\). Let \(g\) be a polynomial that violates \(A_2\). Then we call the POP

\[
\inf_{x \in \mathbb{R}^n} \{ f(x) : g(x) \geq 0, g_i(x) \geq 0, i \in [p] \},
\]

\((\text{NlySpOpt})\)
a nearly sparse POP because only one constraint, namely \(g \geq 0\), does not satisfy the sparsity pattern \(A_2\). This single “bad” constraint \(g \geq 0\) precludes us from applying the sparse Lasserre hierarchy \((\text{MomSpOpt-}d)\).

In this situation, we propose a heuristic method which can be applied to problems with arbitrary many constraints that possibly destroy the sparsity. The key idea of our algorithm is: (i) Keep the “nice” sparsity pattern defined without the bad constraints; (ii) Associate only low-order localizing matrix constraints to the “bad” constraints. In brief, the \(d\)-th order heuristic hierarchy \((\text{HR-d})\) reads:

\[
\inf_y \{ L_y(f) : M_1(y) \geq 0, M_d(y, I_k) \geq 0, k \in [\ell]; M_{d-\omega_i}(g_i y, I_{k(i)}) \geq 0, i \in [p];
\]

\[
L_y(g) \geq 0, L_y(1) = 1 \},
\]

\((\text{MomNlySpOpt-d})\)

where \(y, L_y, M_d(y, I_k), M_{d-\omega_i}(g_i y, I_{k(i)})\) have been defined in section 3.2. For more illustration of this heuristic relaxation and how it is applied to estimate the Lipschitz constant of neural networks, see Appendix D.

For simplicity, assume that the neural networks have only one single hidden layer, i.e., \(m = 1\). Denote by \(A, b\) the weight and bias respectively. As in (4), we use the fact that \(l \leq x \leq u\) is equivalent to \((x - l)(x - u) \leq 0\). Then the local Lipschitz constant estimation problem with respect to \(L_{\infty}\)-norm can be written as:

\[
\max_{x, u, z, t} \{ t^T A^T \text{diag}(u)c : (z - Ax - b)^2 = 0, t^2 \leq 1, (x - \bar{x}_0 + \varepsilon)(x - \bar{x}_0 - \varepsilon) \leq 0,\]

\[
u(u - 1) = 0, (u - 1/2)z \geq 0 \}.
\]

\((\text{LCEP-MLP})\)

Define the subsets of \((\text{LCEP-MLP})\) to be \(I^i = \{x^i, t^i\}, J^j = \{u^j, z^j\}\) for \(i \in [p_0], j \in [p_1]\), where \(p_0, p_1\) are the number of nodes in the input layer and hidden layer respectively. Then the second-order
(d = 2) heuristic relaxation of $\text{LCEP-MLP}_1$ is the following SDP:

$$\inf \{ L_y (t^T A^T \text{diag}(u) c) : L_y (1) = 1, M_1 (y) \succeq 0, L_y (z - Ax - b) = 0, L_y ((z - Ax - b)^2) = 0 ;
M_2 (y, I^i) \succeq 0, M_1 ((- (x^{(i)} - y^{(i)}) + \varepsilon) (x^{(i)} - y^{(i)}) - \varepsilon) y, I^i) \succeq 0, M_1 (((1 - I^2) y, I^i) \succeq 0, i \in [p_0] ;
M_2 (y, J^j) \succeq 0, M_1 (u_j (u_j - 1) y, J^j) = 0, M_1 ([u_j (u_j - 1) 2 y, J^j]) \succeq 0, j \in [p_1] \} .$$

(MomLCEP-2)

The $d$-th order heuristic relaxation $(\text{MomNlySpOpt-d})$ also applies to multiple layer neural networks. However, if the neural network has $m$ hidden layers, then the criterion in $(\text{LCEP})$ is of degree $m + 1$. If $m \geq 2$, then the first-order moment matrix $M_1 (y)$ is no longer sufficient, as moments of degree $> 2$ are not encoded in $M_1 (y)$ and some may not be encoded in the moment matrices $M_2 (y, I^i)$, if they include variables of different subsets. See Appendix E for more information to deal with higher-degree polynomial objective.

5 Experiments

In this section, we provide results for the global and local Lipschitz constants of random networks of fixed size (80, 80) and with various sparsities. We also compute bounds of a real trained 1-hidden layer network. The complete results for global/local Lipschitz constants of both 1-hidden layer and 2-hidden layer networks can be found in Appendix F and G. For all experiments we focus on the $L_\infty$-norm, the most interesting case for robustness certification. Moreover, we use the Lipschitz constants computed by various methods to certify robustness of a trained network, and compare the ratio of certified inputs with different methods. Let us first provide an overview of the methods with which we compare our results.

SHOR: Shor’s relaxation applied to $(\text{LCEP})$. Note that this is different from Shor’s relaxation described in [21] since we apply it to a different QCQP.
HR-2: second-order heuristic relaxation applied to $(\text{LCEP})$.
LBS: lower bound obtained by sampling 50000 random points and evaluating the dual norm of the gradient.

The reason why we list LBS here is because LBS is a valid lower bound on the Lipschitz constant. Therefore all methods should provide a result not lower than LBS, a basic necessary condition of consistency.

As discussed in section 2, if we want to estimate the global Lipschitz constant, we need the input space $\Omega$ to be the whole space. In consideration of numerical issues, we set $\Omega$ to be the ball of radius 10 around the origin. For the local Lipschitz constant, we set by default the radius of the input ball as $\varepsilon = 0.1$. In both cases, we compute the Lipschitz constant with respect to the first label. All experiments are run on a personal laptop with a 4-core i5-6300HQ 2.3GHz CPU and 8GB of RAM. We use the (Python) code provided by [21] to execute the experiments for LipOpt with Gurobi solver. For HR-2 and SHOR, we use the YALMIP toolbox (MATLAB) [23] with MOSEK as a backend to calculate the Lipschitz constants for random networks. For trained network, we implement our algorithm on Julia [4] with MOSEK optimizer to accelerate the computation.

Remark: The crossover option [8] in Gurobi solver is activated by default, and it is used to transform the interior solution produced by barrier into a basic solution. We deactivate this option in our experiments since this computation is unnecessary and takes a lot of time. Throughout this paper, running time is referred to the time taken by the LP/SDP solver (Gurobi/Mosek) and $Of/M$ means running out of memory during building up the LP/SDP model.

5.1 Lipschitz Constant Estimation

Random Networks. We first compare the upper bounds for (80, 80) networks, whose weights and biases are randomly generated. We use the codes provided by [21] to generate networks with various sparsities. For each fixed sparsity, we generate 10 different random networks, and apply all the
Local Lipschitz Constant
Running Time
1
2
3
4
100
10
80
40
80
20
40
80
20
40

As a trade-off, the running time of HR-2 work and moreover, with sparsity 10, 20, 30, 40, 50, 60, 70, 80. In the meantime, we display median and quartiles over 10 random networks.

Figure 1 displays a comparison of average upper bounds of global and local Lipschitz constants. For global bounds, we can see from Figure 1a that when the sparsity of the network is small (10, 20, etc.), the LP-based method LipOpt-3 is slightly better than the SDP-based method HR-2. As the sparsity increases, HR-2 provides tighter bounds. Figure 1b shows that LipOpt-3 is more efficient than HR-2 only for sparsity 10. When the networks are dense or nearly dense, our method not only takes much less time, but also gives tighter upper bounds. For global Lipschitz constant estimation, SHOR and HR-2 give nearly the same upper bounds. However, in the local case, HR-2 provides strictly tighter bounds than SHOR. In both global and local cases, SHOR has smaller computational time than HR-2. In Appendix F and G, we present more results of global and local Lipschitz constant bounds for networks of various sizes and sparsities.

Trained Network  We use the MNIST classifier (SDP-NN) described in [30]. The network is of size (784, 500). In Table 2, we see that the LipOpt-3 algorithm runs out of memory when applied to the real network SDP-NN to compute the global Lipschitz bound. In contrast, SHOR and HR-2 still work and moreover, HR-2 provides tighter upper bounds than SHOR in both global and local cases. As a trade-off, the running time of HR-2 is around 5 times longer than that of SHOR.

5.2 Robustness Certification

Binary Classifier  We generate 20000 random points in dimension 80, where half of them are concentrated around the sphere with radius 1 (labeled as 1) and the rest are concentrated around the sphere with radius 2 (labeled as 2). We train a (80, 80) network for this binary classification task, and use sigmoid layer as the output layer. Therefore, the input is labeled as 1 (resp. 2) if the output is negative (resp. positive). For an input \( x_0 \) and a perturbation \( \epsilon \), we compute an upper bound of the Lipschitz constant with respect to \( x_0, \epsilon \) and \( L_\infty \)-norm, denoted by \( L_{x_0,\epsilon}^{x_0,\epsilon} \). Then, if the output \( y_0 \) is negative (resp. positive) and \( y_0 + \epsilon L_{x_0,\epsilon}^{x_0,\epsilon} < 0 \) (resp. \( y_0 - \epsilon L_{x_0,\epsilon}^{x_0,\epsilon} > 0 \)), the input \( x_0 \) is \( \epsilon \)-robust. With this criteria, if we have \( n \) inputs to certify, then we need to execute \( n \) experiments.

https://worksheets.codalab.org/workheets/0xa21e794020bb474d8804ec7bc0543f52/
Table 3: Comparison of ratios of certified examples for (80, 80) network by HR-2 and LipOpt-3.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
<th>0.05</th>
<th>0.06</th>
<th>0.07</th>
<th>0.08</th>
<th>0.09</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR-2</td>
<td>87.51%</td>
<td>75.02%</td>
<td>62.46%</td>
<td>49.89%</td>
<td>37.22%</td>
<td>24.36%</td>
<td>8.15%</td>
<td>2.75%</td>
<td>0.76%</td>
<td>0.12%</td>
</tr>
<tr>
<td>LipOpt-3</td>
<td>69.03%</td>
<td>37.84%</td>
<td>4.78%</td>
<td>1.5%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 4: Ratios of certified test examples for SDP-NN network by HR-2.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
<th>0.05</th>
<th>0.06</th>
<th>0.07</th>
<th>0.08</th>
<th>0.09</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratios</td>
<td>98.80%</td>
<td>97.24%</td>
<td>95.16%</td>
<td>92.84%</td>
<td>90.18%</td>
<td>87.10%</td>
<td>83.01%</td>
<td>78.34%</td>
<td>73.54%</td>
<td>67.63%</td>
</tr>
</tbody>
</table>

However, for large networks such as MNIST networks, this is impractical, even if we certify the network directly without using Lipschitz constants (see [30]). Therefore, we compute the Lipschitz constant with respect to $x_0 = 0$ and $\epsilon = 3$, denoted by $L$, and generate $10^6$ random points in the box $B = \{ x : ||x||_\infty \leq 2.9 \}$. For any $\epsilon \leq 0.1$ and $x_0 \in B$, we have $L x_0,\epsilon \leq L$, then the point $x_0$ is $\epsilon$-robust if $y_0(y_0 - \text{sign}(y_0)\epsilon L) > 0$. Instead of running $10^6$ times the algorithm, we are able to certify robustness of large number of inputs by only doing one experiment. Table 5 shows the ratio of certified points in the box $B$ by HR-2 and LipOpt-3. We see that HR-2 can always certify more examples than LipOpt-3.

Multi-Classifier: As described in Section 5.1 the SDP-NN network is a well-trained (784, 500) network to classify the digit images from 0 to 9. Denote the parameters of this network by $A \in \mathbb{R}^{500 \times 784}$, $b_1 \in \mathbb{R}^{500}$, $C \in \mathbb{R}^{10 \times 500}$, $b_2 \in \mathbb{R}^{10}$. The score of an input $x$ is denoted by $y^x$, i.e., $y^x = C \cdot \text{ReLU}(Ax_0 + b_1) + b_2$. The label of $x$, denoted by $r^x$, is the index with the largest score, i.e., $r^x = \arg \max y^x$. Suppose an input $x_0$ has label $r$. For $\epsilon$ and $x$ such that $||x - x_0||_\infty \leq \epsilon$, if for all $i \neq r$, $y^x_i - y^x_r < 0$, then $x_0$ is $\epsilon$-robust. Alternatively, denote by $L_{i,r}^{x_0,\epsilon}$ the local Lipschitz constant of function $f_{i,r}(x) = y^x_i - y^x_r$ with respect to $L_\infty$-norm in the ball $\{ x : ||x - x_0||_\infty \leq \epsilon \}$. Then the point $x_0$ is $\epsilon$-robust if for all $i \neq r$, $f_{i,r}(x_0) + \epsilon L_{i,r}^{x_0,\epsilon} < 0$. Since the $28 \times 28$ MNIST images are flattened and normalized into vectors taking value in $[0, 1]$, we compute the local Lipschitz constant (by HR-2) with respect to $x_0 = 0$ and $\epsilon = 2$, the complete value is referred to matrix $L$ in Appendix 4. We take different values of $\epsilon$ from 0.01 to 0.1, and compute the ratio of certified examples among the 10000 MNIST test data by the Lipschitz constants we obtain, as shown in Table 4. Note that for $\epsilon = 0.1$, we improve a little bit by 67% compared to Grad-cert (65%) described in [30], as we use an exact formulation of the derivative of ReLU function.

6 Conclusion and Future Work

Optimization Aspect: In this work, we propose a new heuristic moment relaxation based on the dense and sparse Lasserre’s hierarchy. In terms of performance, our method provides bounds no worse than Shor’s relaxation and better bounds in many cases. In terms of computational efficiency, our algorithm also applies to nearly sparse polynomial optimization problems without running into computational issue.

Machine Learning Aspect: The ReLU function and its generalized derivative $G(x)$ are semialgebraic. This semialgebraic character is easy to handle exactly in polynomial optimization (via some lifting) so that one is able to apply moment relaxation techniques to the resulting POP. Moreover, our heuristic moment relaxation provides tighter bounds than Shor’s relaxation and the state-of-the-art LP-based algorithm in [21].

Future research: The heuristic relaxation is designed for QCQP (e.g. problem LCEP) for 1-hidden layer networks. As the number of hidden layer increases, the degree of the objective function also increases and the approach must be combined with lifting or sub-moment techniques described in Appendix 4 in order to deal with higher-degree objective polynomials. Efficient derivation of approximate sparse certificates for high degree polynomials should allow to enlarge the spectrum of applicability of such techniques to larger size networks and broader classes of activation functions. This is an exciting topic of future research.
Broader Impact

Developing optimization methods resulting in numerical certificates is a necessary step toward the verification of systems involving AI trained components. Such systems are expected to be more and more common in the transport industry and constitute a major challenge in terms of certification. We believe that polynomial optimization is one promising tool to address this challenge.

Acknowledgments and Disclosure of Funding

This work has benefited from the AI Interdisciplinary Institute ANITI funding, through the French “Investing for the Future – PIA3” program under the Grant agreement n°ANR-19-PI3A-0004. Edouard Pauwels acknowledges the support of Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant numbers FA9550-19-1-7026 and FA9550-18-1-0226, and ANR MasDol. Victor Magron was supported by the FMJH Program PGMO (EPICS project) and EDF, Thales, Orange et Criteo, the Tremplin ERC Stg Grant ANR-18-ERC2-0004-01 (T-COPS project) as well as the European Union’s Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Actions, grant agreement 813211 (POEMA). A large part of this work was carried out as Tong Chen was a master intern at IRT-Saint-Exupéry, Toulouse, France.

References


