Robustness Verification of Neural Networks using Polynomial Optimization

Tong Chen

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *02/12/2022* **par :**

Tong CHEN

Robustness Verification of Neural Networks using Polynomial Optimization

---

JURY

Alexandre d'ASPREMONT	Directeur de Recherche	Président du jury
Yohann de CASTRO	Professeur d'Université	Rapporteur
Victor MAGRON	Chargé de Recherche	Co-directeur de thèse
Edouard PAUWELS	Maître de Conférences	Directeur de thèse
David STEURER	Associate Professor	Examinateur
Lihong ZHI	Professeur d'Université	Rapporteure

École doctorale et spécialité :
MITT : Domaine Mathématiques : Mathématiques appliquées
Unité de Recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)
Directeur(s) de Thèse :
Edouard PAUWELS et *Victor MAGRON*
Rapporteurs :
Yohann de CASTRO et *Lihong ZHI*

知之为知之，
不知为不知，
是知也。

——《论语·为政》

To my parents, and to my Whisy.

Acknowledgments

How time flies. It has been three years since I started my doctorate. I can never imagine how small and unhelpful I am towards the passage of time. This thesis is the set of works that I accomplished during this unforgettable period, and is specially dedicated to my supervisors, my parents and my wife, in memory with my lost student time.

First of all, I would like to thank all my supervisors: Edouard Pauwels, Victor Magron and Jean-Bernard Lasserre. You are the guiders who introduce me to the world of polynomial optimization and robustness verification. Verifying the robustness of neural networks is a difficult and important problem in machine learning community, and few researchers apply polynomial optimization techniques to it. I am happy that, thanks to our efforts, we succeeded in proposing new approaches, although there are still many limitations. You are not only my supervisors, but also my friends.

I would like also to express my gratitude to all my jury members: Alexandre d'Aspremont, Yohann de Castro, David Steurer and Lihong Zhi. Thank you all for reading my thesis and providing valuable comments. I am looking forward to future collaborations with you.

I would like to thank my parents for supporting me to continue my studies in France. As I grow, you never interfere with my ideas and always encourage me to do what I want to do. Without my parents, I am nothing.

I would like to thank my colleagues at LAAS, which allow me to interact with others and work in a comfortable environment. Ngoc Hoang Anh Mai, I believe that you will make great achievements in your postdoc career! Jie Wang, I always aim to become an excellent researcher like you! Srecko Durasinovic, I hope that everything goes fine with you in Singapore and I am looking forward to speaking Chinese with you in the near future! Corbinian Schlosser, best wishes for your defense next year! Also thanks to all the ping-pong players: Alexey, Florent, Nicola, Manon...

I would like to thank my previous teachers who have greatly influenced my life. Christophe Giraud, I really appreciate your cleverness and kindness. Bin Li, you not only taught me abstract algebra, but also the quality of a good teacher. Xiaofang Zhou, you are my first and favorite teacher in university. Tingting Liu, Lijun Duan, you are the best Chinese and English teacher in my high school! Kenan Liu, your way of teaching Chinese is really like a loving mother. Youshao Pu, you are the first one who show me the beauty of math and made me fall into love with it.

I would like to thank all my friends appearing in my life, from past to now, from China to France. Hao Hu, you gave me a lot of support in my daily life, all the best with your PhD defense! Martin Mugnier, I really enjoy the time together with you at Orsay! Alexandre Werthe, you are the first person who help me with my courses and my life at Besançon, I will always remember you. Junchao Chen, our time at Besançon and Paris is definitely a wonderful memory. Zhuo Liu, Mingyang Ren,

Lirong Gan, Kanghong Jing, we all won't forget the days at Wuhan University. Jianwei An, it's your hardworking and persistence that brings us to the first prize of Chinese modeling competition. Hengtao Bao, thank you for your generous help as a senior student. Junjie Wan, Yudong Wu, Shuai Yang, Haocheng Liu, Leiying Peng, our friendship will last forever. Yi Chen, my best friend, looking forward to seeing you in my wedding!

Finally, I would like to express my appreciation to my wife, Whisy. It's been 10 years since we've known each other and 1 year since we've got married. There are plenty of memory between us in Guiyang, Wuhan, Chongqing, Paris, Barcelona, Toulouse, Salamanca, Madrid, Berlin, Hamburg, Granada... Thank you so much for your encouragement and love. In the period when I am depressed for my research, or in the difficult time because of language problems, you accompany me and face these obstacles with me together. Those gratitude and love can never be expressed with limited words. My parents gave me the birth and you gave me rebirth.

Tong CHEN
Toulouse, 2 December 2022

Abstract

Nowadays, Neural Networks (NNs) are widely and successfully used for many large-scale and complex machine learning tasks, such as image classification, voice recognition recommender system, and have attracted the attention of many community researchers. However, even though neural networks are powerful, they are generally neither robust nor reliable. This means that, for some input examples, neural network classifiers can produce unstable predictions of their labels, i.e. classifiers are sensitive to small input changes, which is problematic in some crucial areas such as auto-driving or aerospace. Therefore, the robustness of neural networks has become a critical issue in the machine learning community in recent years. Robustness for NNs has two main facets: robust training and robust verification. In this thesis, we mainly focus on verifying the robustness of pre-trained NNs. Based on semi-definite programming, we first develop a sublevel hierarchy for polynomial optimization problems and then apply it to robustness verification of NNs. Our approaches show empirical improvements over other related methods.

Keywords: robustness verification, Lipschitz constant, neural network, polynomial optimization, semidefinite programming, moment-sum of squares hierarchy, correlative sparsity.

Résumé

De nos jours, les Réseaux de Neurones (RN) sont largement et avec succès utilisés pour de nombreuses tâches d'apprentissage automatique à grande échelle et complexes, telles que la classification d'images, le système de recommandation de reconnaissance vocale, et ont attiré l'attention de nombreux chercheurs de la communauté. Cependant, même si les réseaux de neurones sont puissants, ils ne sont généralement ni robustes ni fiables. Cela signifie que, pour certains exemples d'entrées, les classificateurs de réseaux de neurones peuvent produire des prédictions instables de leurs étiquettes, c'est-à-dire que les classificateurs sont sensibles aux petits changements d'entrée, ce qui est problématique dans certains domaines cruciaux comme la conduite automobile ou l'aérospatiale. Par conséquent, la robustesse des réseaux de neurones est devenue un problème critique dans la communauté de l'apprentissage automatique ces dernières années. La robustesse a principalement deux facettes : une formation robuste des RN et une vérification robuste des RN. Dans cette thèse, nous nous concentrons principalement sur la vérification de la robustesse des RN pré-formés. Sur la base de la programmation semi-définie, nous développons d'abord une hiérarchie de sous-niveaux pour les problèmes d'optimisation polynomiale, puis nous l'appliquons à la vérification de robustesse des RN. Nos approches montrent des améliorations empiriques par rapport à d'autres méthodes connexes.

Mots-clés : vérification de robustesse, constante de Lipschitz, réseau de neurones, optimisation polynomiale, programmation semi-définie, hiérarchie moment-somme de carrés, parcimonie corrélative.

Contents

List of Figures

List of Tables

List of Notations

- $\mathbb{R}^n$: the n-dimensional real vector space;
- $\mathbb{Z}^n$: the set of n-tuples of integers;
- $x \in \mathbb{R}$: a real number;
- $\mathbf{x} \in \mathbb{R}^n$: an n-dimensional real vector;
- $\mathbf{x} \circ \mathbf{y}$: element-wise product of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$;
- $[n]$: the set $\{1, 2, \ldots, n\}$;
- $x^{(i)}$: the i-th coordinate of vector $\mathbf{x}$ for $i \in [n]$;
- $\mathbf{x}^{(I)}$: the subvector of $\mathbf{x}$ consisting of the coordinates indexed by $I \subseteq [n]$;
- $\mathbf{x}_{i:j}$: the concatenation of vectors $\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_j$ for $i < j$;
- $\mathbf{A} \in \mathbb{R}^{m \times n}$: a real matrix of size $m \times n$;
- $\mathbf{A}^{(i,:)}$: the i-th row vector of matrix $\mathbf{A}$;
- $\mathbb{S}^n$: the cone of real symmetric matrices;
- $\mathbb{S}^n_+$: the cone of real positive semidefinite matrices;
- $\mathbb{R}[\mathbf{x}]$: the vector space of real polynomials in variable $\mathbf{x}$;
- $\mathbb{R}_d[\mathbf{x}]$: the vector space of real polynomials in variable $\mathbf{x}$ with degree $\leq d$;
- $P[\mathbf{x}]$: the cone of nonnegative real polynomials in variable $\mathbf{x}$;
- $P_d[\mathbf{x}]$: the cone of nonnegative real polynomials in variable $\mathbf{x}$ with degree $\leq d$;
- $\Sigma[\mathbf{x}]$: the cone of sum of squares of polynomials in variable $\mathbf{x}$;
- $\Sigma_d[\mathbf{x}]$: the cone of sum of squares of polynomials in variable $\mathbf{x}$ with degree $\leq 2d$;
- $\mathbf{v}_d(\mathbf{x})$: the monomial basis of variable $\mathbf{x}$ of degree $\leq d$;
- $\mathcal{P}(\mathbf{K})$: the space of Borel probability measure with support contained in $\mathbf{K}$;
- $L_{\mathbf{y}}$: Riesz linear functional;
- $\mathbf{M}_d(\mathbf{y})$: d-th order (dense) moment matrix;
- $\mathbf{M}_d(g\mathbf{y})$: d-th order (dense) localizing matrix of polynomial g;

- $\mathbf{M}_d(\mathbf{y}, I)$: d-th order (sparse) moment matrix with respect to subset I;
- $\mathbf{M}_d(g\mathbf{y}, I)$: d-th order (sparse) localizing matrix of polynomial g with respect to subset I.

List of Acronyms

Introduction

Many human activities and inventions are influenced or inspired by other biological behaviors. For example, birds inspired us to fly and invent airplanes, fish inspired the invention of the submarine, burdock plants inspired Velcro, etc. Artificial Neural Network (ANN) is also a great creativity inspired by the architecture of the human brain, which is composed of a large number of highly interconnected neurons working together to solve a specific problem. Nowadays, ANNs play a very important role in deep learning, the most exciting and powerful branch of machine learning. The power and scalability of ANNs make them widely and successfully used for many large-scale, difficult and complicated machine learning tasks, such as image classification, speech recognition, recommender system and training of Artificial Intelligence (AI) to beat humans in many games like Go or StarCraft. When we talk about a neural network, we often refer to network parameters. An ANN is actually an application, or a function, which is nonlinear with respect to its parameters. Usually the parameters are estimated or learned from a set of training samples, and the learning process is indeed to minimize an objective function which is nonconvex and nonlinear, leading to local minimizers and a Non-deterministic Polynomial-time hard (NP-hard) complexity. The basis for the successful application of ANNs is due to the universal approximation theorem by [Cybenko 1989, Hornik *et al.* 1989]. Moreover, [Lecun 1988] has proposed an efficient algorithm, called backpropagation, to calculate the gradient of an ANNs, which allows us to easily obtain a local minimizer of the optimization problem and is probably the supervised learning algorithm most widely used.

Every coin has two sides, and the artificial neural network is no exception. Although ANNs have been very successful in many areas, there are also many drawbacks and open issues. For example, ANNs are highly hardware dependent. In order to satisfy the requirements of accuracy and instantaneousness, deep learning algorithms and models must be executed on good hardware platforms. No matter for training or verifying neural networks, the computing power of the platform is the main obstacle that limits the application of ANNs. Moreover, as neural networks are often used as black box algorithms, two major challenges for ANNs are to provide proper *interpretabilities* and to ensure high *reliability* for the designed models. An interpretable ANN helps people fully understand the internal structure of the network and explain the relationships between input features and output labels, which is crucial in certain fields such as healthcare and finance. Also, it is very important to know if a classifier is reliable or not, in other words if it is robust with respect to inputs. This induces the problem of verifying the robustness of a given classifier, i.e. if we disturb the inputs a little, the output should not be different. In fact, [Molnar 2019] has already discovered many adversarial examples (instances with small intentional feature perturbations that cause a machine learning model to make a false prediction) in neural networks. Historically,

when the contradictory example occurs, [Lu *et al.* 2017] tried to come up with defenses based on test input transformations, but the defense was broken in just five days by [Athalye *et al.* 2018]. Therefore, there is a need for a systematic method to deal with this problem. The main objective of this manuscript is to address these problems of robustness verification in neural networks using polynomial optimization techniques. [Raghunathan *et al.* 2018b] translated this kind of problem into a nonconvex Quadratically Constrained Quadratic Program (QCQP) and used Semidefinite Programming (SDP) relaxation (an SDP with a larger feasible domain that is easier to solve) to obtain an upper bound on the optimal value of the original QCQP. In Section 5.4, we will see that the verification problem is indeed to identify whether the optimal value of an optimization problem is negative or not, so that if we obtain a negative upper bound of the optimal value, we could still verify robustness. What we propose in this manuscript is also a method based on SDP, with finer approximations and guaranteed convergence. We also propose two other approaches, Lipschtz constant estimation and ellipsoidal propagation, both of which can be applied to verify the robustness of neural networks.

The outline of the manuscript is as follows:

- Chapter 1: Deep learning and neural networks. This chapter gives an introduction on the background of deep learning: in which situation deep learning algorithms (especially neural networks) attract the attention of researchers, what are the significant advantages of deep learning compared to traditional machine learning algorithms, why robustness is an important issue for neural networks. Main references include [Lecun 1988] for backpropagation, [Hornik 1991] for universal approximation theorem, , [Bai *et al.* 2019] for deep equilibrium model, [Winston & Kolter 2020] for monotone operator equilibrium network, [Kolter & Madry 2018] for adversarial example.

- Chapter 2: Existing approaches to robustness verification. In a nutshell, robustness verification can be formalized as a highly nonlinear composite constraint satisfaction problem. We give an overview of existing approaches to verify the robustness of neural networks, which can be classified into three types of approaches: exact deterministic approximation, deterministic approximation and statistical approximation. The approaches and algorithms proposed in this manuscript can be classified either in exact deterministic approach (when we consider the convergent hierarchy) or in approximate deterministic approach (when we stick to a fixed order relaxation). Main references include [Huang *et al.* 2020, Li *et al.* 2020] for surveys on robustness verification of neural networks, [Katz *et al.* 2017] for Reluplex, [Ehlers 2017] for Planet, [Lomuscio & Maganti 2017] for NSVerify, [Tjeng *et al.* 2019] for MIPVerify, [Gehr *et al.* 2018] for AI2, [Weng *et al.* 2018c] for FastLin/FastLip, [Boopathy *et al.* 2019] for CNN-cert, [Raghunathan *et al.* 2018b] for SDP-cert, [Zhang *et al.* 2018] for CROWN, [Wang *et al.* 2021d] for β-CROWN.

- Chapter 3: Convex relaxations and moment-SOS hierarchy. This chapter

focuses on the theoretical preliminaries of the convex relaxation techniques that interest us when we want to obtain approximations of non-convex optimization problems. We also introduce the Lasserre hierarchy, a systematic approach to relax a Polynomial Optimization Problem (POP) to a sequence of SDPs, whose associated sequence of optimal values converges to the exact solution. Main references include [Lasserre 2001] for dense moment relaxation, [Lasserre 2006, Waki *et al.* 2006] for sparse moment relaxation.

- Chapter 4: Sublevel hierarchy. This chapter is the methodological contribution in terms of polynomial optimization. We develop a hierarchy based on SDP, called sublevel hierarchy, for general POPs, in order to obtain finer bounds between the standard Lasserre relaxations of order d and $(d+1)$. The advantage of the sublevel hierarchy is that we can handle dense or nearly dense POPs where the standard hierarchy may not be manageable due to memory issues. We provide many benchmarks in polynomial optimization where the sublevel hierarchy shows improvements over the standard Lasserre hierarchy. The materials of this chapter are referred to [Chen *et al.* 2022].

- Chapter 5: Robustness verification and related problems. This chapter is the application of the sublevel moment-Sums of Squares (moment-SOS) hierarchy for two types of neural networks: Deep Neural Network (DNN) and Monotone Operator Equilibrium Network (MON). We first show how to represent the Rectified Linear Unit (ReLU) function and its subdifferential exactly with polynomial systems, which gives a semi-algebraic representation of the input-output relationship of ReLU networks. Then, we present several applications to network verification and obtain SDP models for the following problems: Lipschitz constant estimation, ellipsoidal propagation and robustness verification. We use these models to verify the robustness of DNNs and MONs w.r.t. norm L_2 and L_∞. Related works include [Raghunathan *et al.* 2018b] for robustness verification of DNNs by Shor's relaxation, [Latorre *et al.* 2020] for estimating Lipschitz constant of DNNs by Linear Programming (LP) relaxation, [Dathathri *et al.* 2020] for robustness verification by fast first-order SDP algorithm, [Chen *et al.* 2020, Chen *et al.* 2021] for robustness verification of DNNs and MONs by sublevel relaxation.

The list of our contributions is as follows:

- [Chen *et al.* 2022], dedicated to the sublevel relaxation framework for polynomial optimization problems, which is presented in Chapter 4;

- [Chen *et al.* 2020], dedicated to the application of sublevel relaxations for robustness verification of DNNs, which is presented in Chapter 5;

- [Chen *et al.* 2021], dedicated to the application of sublevel relaxations for robustness verification of MONs, which is presented in Chapter 5.

Introduction

De nombreuses activités et inventions humaines sont influencées ou inspirées par d'autres comportements biologiques. Par exemple, les oiseaux nous ont inspiré le vol et l'invention des avions, les poissons ont inspiré l'invention du sous-marin, les bardanes ont inspiré le Velcro, etc. Le Réseau de Neurone Artificiel (RNA) est également une grande créativité inspirée par l'architecture du cerveau humain, qui est composé d'un grand nombre de neurones hautement interconnectés travaillant ensemble pour résoudre un problème spécifique. De nos jours, les RNAs jouent un rôle très important dans l'apprentissage profond, la branche la plus passionnante et la plus puissante de l'apprentissage machine. La puissance et l'extensibilité des RNAs font qu'ils sont largement utilisés avec succès pour de nombreuses tâches d'apprentissage machine à grande échelle, difficiles et compliquées, comme la classification d'images, la reconnaissance vocale, les systèmes de recommandation et l'entraînement des Intelligence Artificielle (IA) pour battre les humains dans de nombreux jeux comme le Go ou StarCraft. Lorsque nous parlons d'un réseau neurone, nous faisons souvent référence aux paramètres du réseau. Un RNA est en fait une application, ou une fonction, qui est non linéaire par rapport à ses paramètres. Habituellement, les paramètres sont estimés ou appris à partir d'un ensemble d'échantillons d'apprentissage, et le processus d'apprentissage consiste en fait à minimiser une fonction objective non convexe et non linéaire, ce qui conduit à des minimiseurs locaux et à une complexité NP-hard. La base de l'application réussie de RNA est due au théorème d'approximation universelle de [Cybenko 1989, Hornik *et al.* 1989]. De plus, [Lecun 1988] a proposé un algorithme efficace, appelé rétropropagation, pour calculer le gradient d'un RNA, qui nous permet d'obtenir facilement un minimiseur local du problème d'optimisation et qui est probablement l'algorithme d'apprentissage supervisé le plus largement utilisé.

Toute pièce de monnaie a deux côtés, et le réseau neuronal artificiel ne fait pas exception. Bien que les RNA aient connu un grand succès dans de nombreux domaines, il existe également de nombreux inconvénients et problèmes non résolus. Par exemple, les RNA sont très dépendants du matériel informatique. Afin de satisfaire aux exigences de précision et d'instantanéité, les algorithmes et modèles d'apprentissage profond doivent être exécutés sur de bonnes plateformes matérielles. Que ce soit pour l'entraînement ou la vérification des réseaux de neurones, la puissance de calcul de la plateforme est le principal obstacle qui limite l'application des RNA. De plus, les réseaux neurones étant souvent utilisés comme des algorithmes de type boîte noire, deux défis majeurs pour le RNA sont de fournir des *interprétabilités* appropriées et d'assurer une *fiabilité* élevée pour les modèles conçus. Un RNA interprétable aide les gens à comprendre pleinement la structure interne du réseau et à expliquer les relations entre les caractéristiques d'entrée et les étiquettes de sortie, ce qui est crucial dans certains domaines tels que la santé et la finance. En outre, il est très important de savoir si un classificateur est fiable ou non, en d'autres

termes s'il est robuste par rapport aux entrées. Ceci induit le problème de la vérification de la robustesse d'un classifieur donné, c'est-à-dire que si on perturbe un peu les entrées, la sortie ne devrait pas être différente. En fait, [Molnar 2019] a déjà découvert de nombreux exemples contradictoires (instances avec de petites perturbations intentionnelles des caractéristiques qui amènent un modèle d'apprentissage machine à faire une fausse prédiction) dans les réseaux neurones. Historiquement, lorsque l'exemple contradictoire se produit, [Lu *et al.* 2017] a essayé de trouver des défenses basées sur des transformations d'entrée de test, mais la défense a été brisée en seulement cinq jours par [Athalye *et al.* 2018]. Par conséquent, il est nécessaire de trouver une méthode systématique pour traiter ce problème. L'objectif principal de cette thèse est d'aborder ces problèmes de vérification de la robustesse dans les réseaux neurones en utilisant des techniques d'optimisation polynomiale. [Raghunathan *et al.* 2018b] a traduit ce type de problème en un QCQP non convexe et a utilisé la relaxation basée sur la programmation semi-définie (un programme avec un domaine réalisable plus grand et plus facile à résoudre) pour obtenir une borne supérieure sur la valeur optimale du QCQP original. Dans la section 5.4, nous verrons que le problème de vérification est en fait d'identifier si la valeur optimale d'un problème d'optimisation est négative ou non, de sorte que si nous obtenons une borne supérieure négative de la valeur optimale, nous pouvons toujours vérifier la robustesse. Ce que nous proposons dans cette thèse est également une méthode basée sur la programmation semi-définie, avec des approximations plus fines et une convergence garantie. Nous proposons également deux autres approches, l'estimation de la constante de Lipschtz et la propagation ellipsoïdale, qui peuvent toutes deux être appliquées pour vérifier la robustesse des réseaux neurones.

Le plan de la thèse se présente comme suit :

- Chapitre 1 : L'apprentissage profond et les réseaux de neurones. Ce chapitre donne une introduction sur le contexte de l'apprentissage profond : dans quelle situation les algorithmes d'apprentissage profond (en particulier les réseaux de neurones) attirent l'attention des chercheurs, quels sont les avantages significatifs de l'apprentissage profond par rapport aux algorithmes traditionnels d'apprentissage machine, pourquoi la robustesse est une question importante pour les réseaux de neurones. Les principales références comprennent : [Lecun 1988] pour la rétropropagation, [Hornik 1991] pour le théorème d'approximation universelle, , [Bai *et al.* 2019] pour le modèle d'équilibre profond, [Winston & Kolter 2020] pour le réseau d'équilibre à opérateur monotone, [Kolter & Madry 2018] pour l'exemple contradictoire.

- Chapitre 2 : Les approches existantes de la vérification de la robustesse. En un mot, la vérification de la robustesse peut être formalisée comme un problème composite de satisfaction de contraintes hautement non linéaire. Nous donnons un aperçu des approches existantes pour vérifier la robustesse des réseaux neuronaux, qui peuvent être classées en trois types d'approches : l'approximation déterministe exacte, l'approximation déterministe et l'approximation statistique. Les approches et algorithmes proposés dans cette thèse peuvent

être classés soit dans l'approche déterministe exacte (lorsque nous considérons la hiérarchie convergente), soit dans l'approche déterministe approximative (lorsque nous nous en tenons à une relaxation d'ordre fixe). Les principales références comprennent : [Huang *et al.* 2020, Li *et al.* 2020] pour les enquêtes sur la vérification de la robustesse des réseaux neuronaux, [Katz *et al.* 2017] pour Reluplex, [Ehlers 2017] pour Planet, [Lomuscio & Maganti 2017] pour NSVerify, [Tjeng *et al.* 2019] pour MIPVerify, [Gehr *et al.* 2018] pour AI2, [Weng *et al.* 2018c] pour FastLin/FastLip, [Boopathy *et al.* 2019] pour CNN-cert, [Raghunathan *et al.* 2018b] pour SDP-cert, [Zhang *et al.* 2018] pour CROWN, [Wang *et al.* 2021d] pour β-CROWN.

- Chapitre 3 : Les relaxations convexes et la hiérarchie moment-SOS. Ce chapitre se concentre sur les préliminaires théoriques des techniques de relaxation convexe qui nous intéressent lorsque nous voulons obtenir des approximations de problèmes d'optimisation non convexes. Nous introduisons également la hiérarchie de Lasserre, une approche systématique pour relaxer un problème d'optimisation polynomiale en une séquence de la programmation semi-définie, dont la séquence associée de valeurs optimales converge vers la solution exacte. Les principales références incluent : [Lasserre 2001] pour la relaxation des moments sans parsimonie, [Lasserre 2006, Waki *et al.* 2006] pour la relaxation des moments avec parsimonie.

- Chapitre 4 : La hiérarchie des sous-niveaux. Ce chapitre constitue la contribution méthodologique en termes d'optimisation polynomiale. Nous développons une hiérarchie basée sur la programmation semi-définie, appelée hiérarchie de sous-niveaux, pour les problèmes d'optimisation polynomiale générales, afin d'obtenir des bornes plus fines entre les relaxations de Lasserre standard d'ordre d et $(d+1)$. L'avantage de la hiérarchie de sous-niveaux est que nous pouvons traiter des problèmes d'optimisation polynomiale denses ou presque denses où la hiérarchie standard peut ne pas être gérable en raison de problèmes de mémoire. Nous fournissons de nombreux exemples en optimisation polynomiale où la hiérarchie à sous-niveaux montre des améliorations par rapport à la hiérarchie de Lasserre standard. Les matériaux de ce chapitre sont référencés dans [Chen *et al.* 2022].

- Chapitre 5 : La vérification de la robustesse et les problèmes connexes. Ce chapitre porte sur l'application de la hiérarchie des sous-niveaux pour deux types de réseaux neurones : le réseau neurone profond et le réseau d'équilibre à opérateur monotone. Nous montrons d'abord comment représenter la fonction ReLU et son sous-différentiel exactement avec des systèmes polynomiaux, ce qui donne une représentation semi-algébrique de la relation entrée-sortie des réseaux ReLU. Ensuite, nous présentons plusieurs applications à la vérification de réseaux et obtenons des modèles semi-définie pour les problèmes suivants : l'estimation de la constante de Lipschtz, la propagation ellipsoïdale et la vérification de la robustesse. Nous utilisons

ces modèles pour vérifier la robustesse de réseau neurone profond et réseau d'équilibre à opérateur monotone par rapport à la norme L_2 et L_∞. Les travaux connexes comprennent : [Raghunathan *et al.* 2018b] pour la vérification de la robustesse de réseau neurone profond par la relaxation de Shor, [Latorre *et al.* 2020] pour l'estimation de la constante de Lipschitz de réseau neurone profond par la relaxation linéaire, [Dathathri *et al.* 2020] pour la vérification de la robustesse par l'algorithme semi-défini rapide du premier ordre, [Chen *et al.* 2020, Chen *et al.* 2021] pour la vérification de la robustesse de réseau neurone profond et réseau d'équilibre à opérateur monotone par la relaxation de sous-niveau.

La liste de nos contributions est la suivante :

- [Chen *et al.* 2022], consacré au cadre de relaxation de sous-niveau pour les problèmes d'optimisation polynomiale, qui est présenté au chapitre 4;
- [Chen *et al.* 2020], consacré à l'application des relaxations de sous-niveaux pour la vérification de la robustesse de réseau neurone profond, qui est présenté dans le chapitre 5;
- [Chen *et al.* 2021], consacré à l'application des relaxations de sous-niveaux pour la vérification de la robustesse de réseau d'équilibre à opérateur monotone, qui est présenté dans le chapitre 5.

Chapter 1

Deep Learning and Neural Networks

Machine learning is an application of AI that enables systems to learn and improve from experience without being explicitly programmed. Machine learning focuses on developing computer programs that can access data and use it to learn for themselves. It recognizes patterns in the data and make predictions once new data arrives. In general, the learning process of these algorithms can either be supervised or unsupervised, depending on the data being used to feed the algorithms.

Deep learning algorithms can be regarded both as a sophisticated and mathematically complex evolution of machine learning algorithms. The field has been getting lots of attention recently and for good reason: recent developments have led to results that were not thought to be possible before. Deep learning describes algorithms that analyze data with a logic structure similar to how a human would draw conclusions. Note that this can happen both through supervised and unsupervised learning. To achieve this, deep learning applications use layered structures of algorithms, i.e., NNs. The design of such NNs is inspired by the biological neural network of the human brain, leading to a process of learning which is far more capable than that of standard machine learning models. Moreover, [Lecun 1988] proposed an efficient way to compute the gradient of a neural network, called *back-propagation*, which allows to obtain a local minimizer of the quadratic criterion easily.

Theoretically, [Hornik 1991] showed that any bounded and regular function $f : \mathbb{R}^n \to \mathbb{R}$ can be approximated at any given precision by a neural network with one hidden layer containing a finite number of neurons, having the same activation function, and one linear output neuron. This result was earlier proved by [Cybenko 1989] in the particular case of the sigmoid activation function. Precisely, Hornik's theorem can be stated as follows.

Theorem 1.1. *(Universal Approximation Theorem) Let σ be a bounded, continuous and non-decreasing (activation) function. Let $\mathbf{K}$ be a compact set in $\mathbb{R}^n$ and $\mathcal{C}(\mathbf{K})$ the set of continuous functions on $\mathbf{K}$. Let $f \in \mathcal{C}(\mathbf{K})$. Then for all $\varepsilon > 0$, there exists $N \in \mathbb{N}_+, a_i, b_i \in \mathbb{R}, \mathbf{c}_i \in \mathbb{R}^n$, such that, if we define*

$$\varphi(\mathbf{x}) = \sum_{i=1}^{N} a_i \sigma(\mathbf{c}_i^T \mathbf{x} + b_i),$$

then we have

$$\forall \mathbf{x} \in \mathbf{K}, |\varphi(\mathbf{x}) - f(\mathbf{x})| \leq \varepsilon.$$

This theorem is interesting from a theoretical point of view. From a practical point of view, this is not really useful since the number of neurons in the hidden layer can be very large. The strength of deep learning lies in the deep (number of hidden layers) of the networks.

Today, deep learning is used in many fields. In automated driving, for example, deep learning is used to detect objects, such as "STOP" signs or pedestrians. The military uses deep learning to identify objects from satellites, e.g. to discover safe or unsafe zones for its troops. Of course, the consumer electronics industry is also full of deep learning. Home assistance devices like Amazon Alexa, for example, rely on deep learning algorithms to respond to your voice and learn your preferences.

1.1 Structure of Neural Networks

The structure of a neural network is also referred to its *architecture* or *topology*. It consists of the number of layers and number of neurons. It also consists of the way of interconnection between neurons of adjacent layers. The choice of the structures determines the results we are going to obtain. It is the most critical part of the implementation of a neural network.

1.1.1 Deep neural network (DNN)

Deep Neural Network (DNN) is also known as deep feed-forward neural network. It is an artificial neural network in which the neurons do not form a cycle. A DNN consists of an input layer, several hidden layers and one output layer, where the input layer takes inputs, and the output layer generates outputs. The hidden layers have no connection with the outer world, that's why they are called hidden layers. Every neuron in one layer is connected with each node in the next layer. Therefore, all the nodes are fully-connected. Something else to notice is that there is no visible or invisible connection between the nodes in the same layer. There are no back-loops in DNNs. Hence, to minimize the error in prediction, we usually use the backpropagation algorithm to update the weight values.

Consider a fully-connected multi-layer neural network for classification. Suppose we have L hidden layers in the neural network, where there are p_0 neurons in the input layer and p_i neurons in each hidden layer for $i = 1, \ldots, L$. For each hidden layer, we associate a triple of parameters $(\mathbf{A}_i, \mathbf{b}_i, \sigma_i)$, where $\mathbf{A}_i \in \mathbb{R}^{p_i \times p_{i-1}}$ is called the *weight matrix*, $\mathbf{b}_i \in \mathbb{R}^{p_i}$ is called the *bias*. We denote by $\mathbf{x}_i \in \mathbb{R}^{p_i}$ the neurons at layer i. The neurons at layer i is activated by the neurons at layer $i-1$, i.e. $\mathbf{x}_i = \sigma(\mathbf{A}_i \mathbf{x}_{i-1} + \mathbf{b}_i)$ where $\sigma : \mathbb{R}^{p_i} \to \mathbb{R}^{p_i}$ is called the *activation function*. Suppose we have K labels, we denote by $F : \mathbb{R}^{p_0} \to \mathbb{R}^K$ the deep neural network. The output of network F is an affine combination of the last hidden neurons, i.e., $F(\mathbf{x}_0) = \mathbf{C}\mathbf{x}_L$ with $\mathbf{C} \in \mathbb{R}^{K \times p_L}$. The prediction $y_{\mathbf{x}_0}$ of input $\mathbf{x}_0$ is the index of output $F(\mathbf{x}_0)$ with

the largest value, i.e. $y_{\mathbf{x}_0} = \arg\max_{i=1,\dots,K} F(\mathbf{x}_0)_i$. Figure 1.1 is an illustration of a DNN of 3 hidden layers.

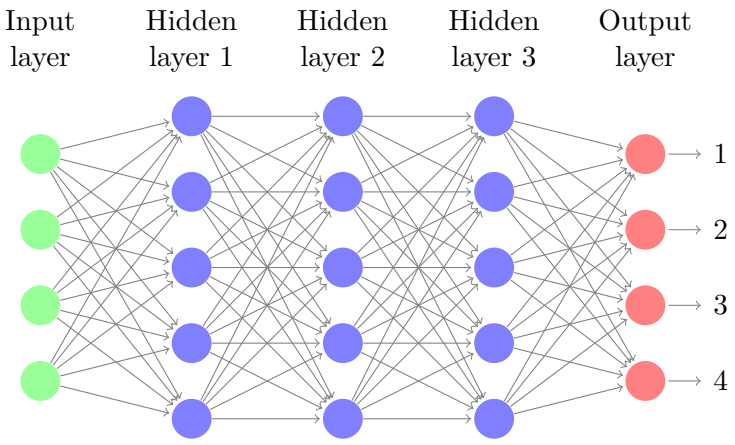


Figure 1.1: Fully-connected deep neural network.

1.1.2 Deep equilibrium model (DEQ)

Deep equilibrium model (Deep Equilibrium Model (DEQ)) is a new variant of deep neural network proposed by [Bai *et al.* 2019]. The difference between a DEQ and a DNN is that there is only one single hidden layer where the hidden neurons are evaluated by a fixed-point equation. Consider a fully-connected deep equilibrium model for classification. Let p_0 be the number of neurons in the input layer and p_1 the number of neurons in the hidden layer. Denote by $\mathbf{x}_0$ and $\mathbf{x}_1$ the neurons at input layer and hidden layer respectively. Then the hidden neurons is computed by $\mathbf{x}_1 = \sigma(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b})$ where $\sigma : \mathbb{R}^{p_1} \to \mathbb{R}^{p_1}$ is the activation function and $\mathbf{A} \in \mathbb{R}^{p_1 \times p_1}, \mathbf{B} \in \mathbb{R}^{p_1 \times p_0}, \mathbf{b} \in \mathbb{R}^{p_1}$ are its parameters. The output of network F is an affine combination of the last hidden neurons, i.e., $F(\mathbf{x}_0) = \mathbf{C}\mathbf{x}_1$ with $\mathbf{C} \in \mathbb{R}^{K \times p_1}$. The prediction $y_{\mathbf{x}_0}$ of input $\mathbf{x}_0$ is the index of output $F(\mathbf{x}_0)$ with the largest value, i.e. $y_{\mathbf{x}_0} = \arg\max_{i=1,\dots,K} F(\mathbf{x}_0)_i$. Figure 1.1 is an illustration of a DEQ.

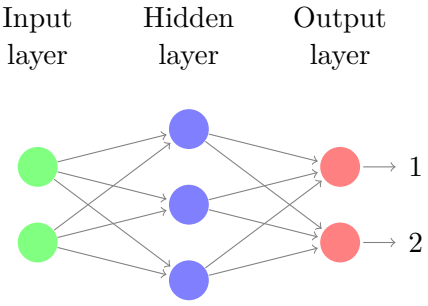


Figure 1.2: Fully-connected deep equilibrium model.

If we add two additional assumptions to a DEQ:

Assumption 1. *(a) Matrix* $\mathbf{I}_{p_1} - \mathbf{A}$ *is* strongly monotone*, i.e.,* $\mathbf{I}_{p_1} - (\mathbf{A} + \mathbf{A}^T)/2 \succeq m\mathbf{I}_{p_1}$ *for some* $m > 0$*,* m *is called the* monotonicity factor*. If* $\mathbf{A}$ *is symmetric, this condition reduces to* $\mathbf{I}_{p_1} - \mathbf{A} \succeq m\mathbf{I}_{p_1}$*;*

(b) The activation function σ *can be represented as proximal operator of some function* f*, i.e.,* $\sigma = prox_f^1$*. For* $\alpha > 0$*, the* proximal operator $prox_f^\alpha$ *is defined as*

$$prox_f^\alpha(\mathbf{x}) := \arg\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2 + \alpha f(\mathbf{z}). \tag{1.1}$$

Remark. ReLU activation function (which will be defined in Section 1.2.3), is equal to the proximal operator of the indicator of the positive orthant $f(x) = \mathbf{1}_{x \geq 0}$. Other familiar activation functions, such as leaky ReLU, tanh, sigmoid, also can be represented by proximal operators [Bibi *et al.* 2019].

A DEQ satisfying Assumption 1 is called a *Monotone Operator Equilibrium Network (MON)* [Winston & Kolter 2020]. The iteration algorithms for the fixed-point equation given by a MON is guaranteed to be convergent. Throughout this thesis, we will always consider MONs instead of DEQs.

1.1.3 Convolutional neural network (CNN)

A Convolutional Neural Network (CNN) uses a variation of the multilayer perceptrons. A CNN contains one or more than one convolutional layers. These layers can either be completely interconnected or pooled. Before passing the result to the next layer, the convolutional layer uses a convolutional operation on the input. Due to this convolutional operation, the network can be much deeper but with much fewer parameters. Due to this ability, convolutional neural networks show very effective results in image and video recognition, signal processing, natural language processing, and recommender systems. Convolutional neural networks also show great success in semantic parsing and paraphrase detection. CNNs are also being used in image analysis and recognition in agriculture where weather features are extracted from satellites like LSAT to predict the growth and yield of a piece of land. Figure 1.3 shows how a typical CNN looks like.

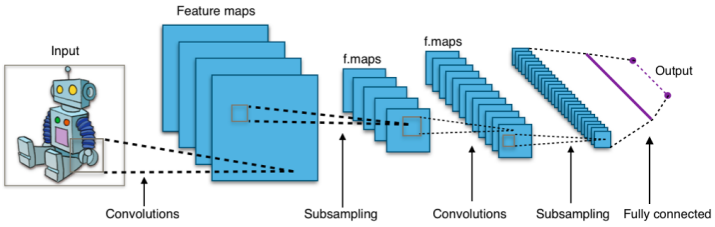


Figure 1.3: A typical CNN architecture.

1.1.4 Recurrent neural network (RNN)

Recurrent Neural Network (RNN) is another variation of deep neural networks. In an RNN, each neuron in hidden layers receives an input with a specific delay in time. We use this type of neural network where we need to access previous information in current iterations. For example, when we are trying to predict the next word in a sentence, we need to know the previously used words first. RNNs can process inputs and share any lengths and weights across time. The model size does not increase with the size of the input, and the computations in this model take into account the historical information. However, the problem with RNNs is the slow computational speed. Moreover, it cannot consider any future input for the current state and it cannot remember information from a long time ago either.

Figure 1.4 gives an illustration of a basic RNN structure. The green circles at the bottom are the input states $\mathbf{x}_t$; the blue boxes in the middle are the hidden states $\mathbf{h}_t$; the red circles on the top are the output states $\mathbf{o}_t$; $\mathbf{U}, \mathbf{V}, \mathbf{W}$ are the weights of the network.

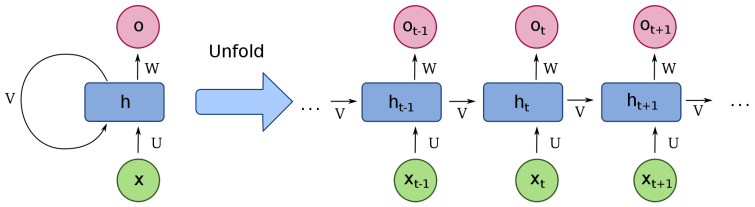


Figure 1.4: A compressed (left) and unfolded (right) basic RNN.

1.1.5 Autoencoder (AE)

An Autoencoder (AE) is an unsupervised machine learning algorithm. In an AE, the number of hidden neurons is smaller than the input neurons. The number of input neurons in an AE equals to the number of output neurons. For autoencoder, we train it to display the output, which is as close as the original input. This forces AEs to find common patterns and regenerate the data. We use autoencoders for smaller representation of the input. We can reconstruct the original data from compressed data. The algorithm is relatively simple as AE requires output to be the same as the input. A Variational Autoencoder (VAE) uses a probabilistic approach for describing observations. It shows the probability distribution for each attribute in a feature set. Figure 1.5 is the basic structure of an AE, where $\mathbf{X}$ is the input neurons, $\mathbf{X}'$ is the output neurons, and $\mathbf{h}$ is the hidden neurons.

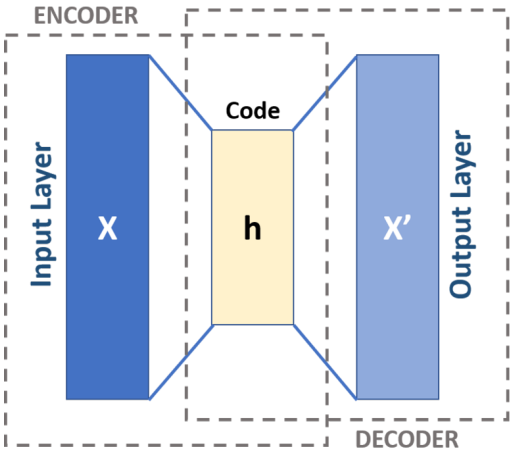


Figure 1.5: Schema of a basic AE.

1.2 Activation Function in Neural Networks

In neural networks, the activation function defines the output of neurons given an input or set of inputs. Only nonlinear activation functions allow neural networks to compute nontrivial problems using only a small number of neurons and hidden layers.

1.2.1 Sigmoid function

Sigmoid function is defined as

$$f(x) = \frac{1}{1 + e^{-x}}, \ \forall x \in \mathbb{R}.$$

This function maps any real number to the open interval $(0, 1)$. It is smooth and monotonically increasing, as shown in Figure 1.6a.

1.2.2 Tanh function

Tanh stands for *hyperbolic tangent* and is defined as

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \ \forall x \in \mathbb{R}.$$

This function is very similar to sigmoid function. It maps any real number to the open interval $(-1, 1)$, and is also smooth and monotonically increasing, as shown in Figure 1.6b.

1.2.3 ReLU function

ReLU stands for *rectified linear unit* and is defined as

$$f(x) = \max\{0, x\}, \ \forall x \in \mathbb{R}.$$

Although it gives an impression of a linear function, ReLU is nonlinear and allows for backpropagation while simultaneously making it computationally efficient. The main catch here is that the ReLU function does not activate all the neurons at the same time. The neurons will only be deactivated if the output of the linear transformation is less than 0. See Figure 1.6c.

1.2.4 Leaky ReLU function

Leaky ReLU function is defined as

$$f(x) = \max\{\alpha x, x\}, \ \forall x \in \mathbb{R}, \ \text{with } \alpha \in (0, 1).$$

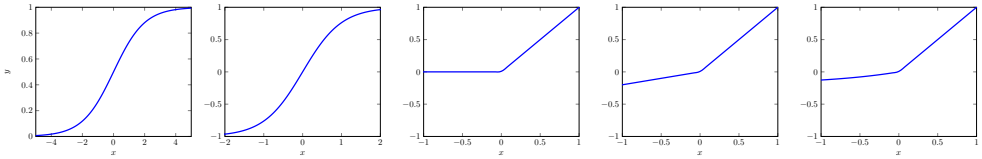
It is an improved version of ReLU function as it has a small positive slope in the negative area. By making this minor modification for negative input values, the gradient of the left side of the graph comes out to be a non-zero value. Therefore, we would no longer encounter dead neurons in that region (which is the main reason of gradient vanishing). See Figure 1.6d.

1.2.5 ELU function

ELU stands for *exponential linear unit* and is defined as

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}, \ \text{with } \alpha \in [0, 1].$$

It is also a variant of ReLU function that modifies the slope of the negative part of the function. ELU uses a log curve to define the negative values unlike the leaky ReLU function with a straight line. See Figure 1.6e.



(a) Sigmoid (b) tanh (c) ReLU (d) Leaky ReLU (e) ELU

Figure 1.6: Familiar activation functions.

1.3 Adversarial Examples

Although artificial neural network is powerful in many areas, the classifiers have been shown in some cases to fail in the presence of small adversarial perturbations of inputs by [Kolter & Madry 2018]. A typical example is displayed in Figure 1.7. The original picture is a panda, we add a tiny perturbation to it and get a new picture which is visually exactly the same (a panda). However, the classifier we trained says that the new picture is a gibbon with high confidence. Such adversarial examples make machine learning models vulnerable to small perturbations. For instance, a self-driving car crashes into another car because it ignores a stop sign. Someone had placed a picture over the sign, which looks like a stop sign with a little dirt for humans, but was designed to look like a parking prohibition sign for the sign recognition software of the car.

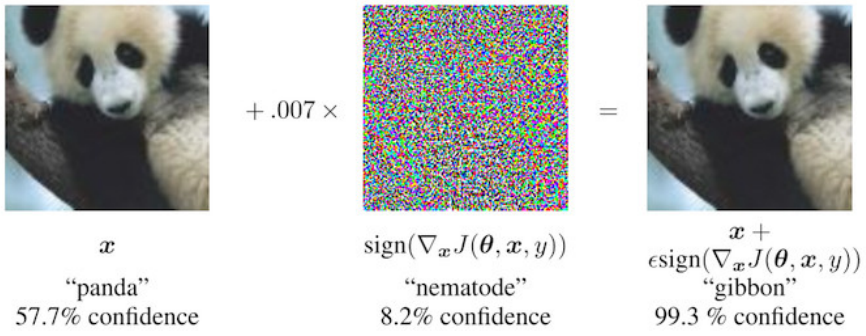


Figure 1.7: An adversarial example [Goodfellow *et al.* 2015].

As we can see, the unsafe or erroneous phenomenon acting on neural networks are essentially caused by the inconsistency of the decision boundaries from deep learning models (that are learned from training datasets) and human oracles. This inevitably raises significant concerns on whether deep learning models can be safely applied in safety-critical domains. This generates difficulties for theoretical analysis and a posteriori performance evaluation, and is problematic for applications where robustness issues are crucial, for example inverse problems (IP) in scientific computing. Indeed such IPs are notoriously ill-posed and is stressed in the March 2021 issue of SIAM News by [Antun *et al.* 2021], *"Yet DL has an Achilles' heel. Current implementations can be highly unstable, meaning that a certain small perturbation to the input of a trained neural network can cause substantial change in its output. This phenomenon is both a nuisance and a major concern for the safety and robustness of DL-based systems in critical applications-like healthcare-where reliable computations are essential"*. Indeed, the Instability Theorem by [Antun *et al.* 2021] predicts unavoidable lower bound on Lipschitz contants, which may explain the lack of stability of some DNNs and over-performing on training sets.

1.3.1 Adversarial training

Numerous defenses have been proposed to deal with the threat of adversarial examples, in particular for those with small perturbations. While many of the existing approaches are heuristic in nature, that is, they do not provide guarantees that tell us when, or if, the model's predictions cannot be manipulated. In some safety critical applications, this may not be good enough. Therefore, researchers try to consider training neural networks that are robust with respect to certain attacks. We briefly introduce two approaches to train robust networks either by controlling the margins along the boundary or estimating the Lipschitz constant.

Constructing globally robust networks: The key idea behind globally robust networks is that we want to construct the network in such a way that a margin will automatically be imposed on the boundary. The output of a neural network classifier is typically what is called a logit vector, containing one dimension (i.e., one logit value) per possible output class. The network makes predictions by selecting the class corresponding to the highest logit value. When we cross a decision boundary, one logit value will surpass the previous highest logit value. Thus, the decision boundary corresponds to the points where there is a tie for the highest logit output. To create a margin along the boundary, we essentially want to thicken the boundary. We can do this by declaring a tie whenever the two highest logits are too close together, i.e., we will consider the decision a draw unless one logit surpasses the rest by at least some δ.

Approximating the Lipschitz Constant: The Lipschitz constant of a function tells us how much the output of the function can change as its input is changed. Intuitively, we can think of it as the maximum slope, or rate of change of the function. Thus, if the Lipschitz constant of a function is L, then the most the function's output can change by, if its input changes by a range of ε, is εL. The Lipschitz constant tells us how much each logit can move when the input is perturbed within a radius of ε, while calculating the exact Lipschitz constant of a neural network is a fundamentally hard problem computationally. Fortunately, we can obtain an upper bound on the Lipschitz constant fairly easily and efficiently by breaking down the computation to consider one layer at a time. This is a naive layer-by-layer multiplication approach to approximate the Lipschitz constant of neural networks. Estimating Lipschitz constant of networks is also a concern of our contribution and we will discuss it more in the following sections.

1.3.2 Robustness verification

Since the discovery of adversarial examples, it becomes critical that we examine not only whether the systems do not simply work "most of the time", but which are truly robust and reliable. Although many notions of robustness and reliability exist, one particular topic in this area that has raised a great deal

of interest in recent years is that of adversarial robustness: can we train classifiers that are certifiably robust to all attacks within a fixed attack model? There are many approaches to verify robustness of neural networks, including the works by [Boopathy *et al.* 2019, Singh *et al.* 2019, Singh *et al.* 2018, Weng *et al.* 2018b, Weng *et al.* 2018a, Weng *et al.* 2018c, Zhang *et al.* 2018], for many of which we construct convex relaxations in order to obtain an upper bound on the worst-case loss (which is the maximum of an optimization problem) over all valid attacks, this upper bound serves as a certificate of robustness.

In general, the robustness problem of a neural network can be formulated as follows: denote the network by F, suppose the input constraint is imposed by a set $\mathcal{X} \subseteq \Omega$ where Ω is the input space, and the corresponding output constraint is imposed by a set $\mathcal{Y} \subseteq \mathcal{O}$ where $\mathcal{O}$ is the output space, then solving the verification problem requires showing that the following assertion holds:

$$\mathbf{x} \in \mathcal{X} \Longrightarrow \mathbf{y} = F(\mathbf{x}) \in \mathcal{Y}.$$

In particular, if the neural network is a classifier, then the output constraint $\mathcal{Y}$ is reduced to a point (label) $\bar{\mathbf{y}}$. The verification problem is then reduced to show that

$$\mathbf{x} \in \mathcal{X} \Longrightarrow \mathbf{y} = F(\mathbf{x}) = \bar{\mathbf{y}}$$

In other words, it requires to show that the predictions of all the inputs in $\mathcal{X}$ are the same.

Generically, a robustness verification problem can be formulated as a Boolean Satisfiability Problem (SAT), which is NP-hard, and no magic method can perfectly solve it with high precision and small computation time. Moreover, we cannot solve the verification problem efficiently for all types of networks. That is to say, in certain cases, we have to design specific algorithms to solve our problems, which might not be applicable in other cases. In the next section, we will categorize the verification approaches into several types, according to the type of guarantees they provide. The trade-off between those approaches is that, if we aim to solve exactly the verification problem (for example, using SAT or Mixed Integer Linear Programming (MILP) solvers), we need to suffer high computational burden; and if we relax the problem to easier ones, we gain lower complexity but we loose the result precision. All those approaches make sense because in some applications we need precision while in other applications we only want high computation speed.

Chapter 2

Existing Approaches to Robustness Verification

In this chapter, we review verification techniques for neural networks. Generally speaking, existing approaches on robustness verification of networks largely fall into the following categories: *deterministic exact approach* (type A), *deterministic approximation approach* (type B), and *statistical approximation approach* (type C). Note that the separation between them may not be strict. The classification of these verification techniques is mainly based on the type of guarantees they provide, they can be formally defined as follows:

Definition 2.1. (1) A *deterministic exact approach* states exactly whether a property holds, e.g., an input example is robust or not robust. Here "exact" means sound and complete (see Section 2.1.1);

(2) A *deterministic approximation approach* provides either lower / upper bounds or over-approximations to the outputs, and thus can serve as a necessary / sufficient condition for a property to hold, e.g., the upper bound of Lipschitz constants of neural networks. A deterministic approximation approach is either sound or complete;

(3) A *statistical approximation approach* quantifies the probability that a property holds, hence is usually neither sound nor complete.

2.1 Characteristics of Verification Methods

When we talk about robustness verification approaches, they certify the lower/upper bound of neural networks performance against any adversarial example under certain constraints, e.g., l_∞-bounded attack. Different approaches have different features and characteristics. We first give the definition of soundness and completeness, which qualitatively determines an approach. Then we briefly discuss about the main features related to the properties of verification methods.

2.1.1 Output-reality relation

If an approach give an output "verified" or "not verified" for some inputs, it is not necessarily guaranteed that the input examples are indeed verified or not verified to be robust. We clarify the relation between the output of the approaches and the ground truth as follows:

Definition 2.2. (1) When a verification approach outputs "verified" for a given input $\mathbf{x}_0$, if it is guaranteed (resp. not guaranteed) that any adversarial example $\mathbf{x}$ around $\mathbf{x}_0$ doesn't exist, then we say this approach is *sound* (resp. *unsound*);

(2) When a verification approach outputs "not verified" for a given input $\mathbf{x}_0$, if it is guaranteed (resp. not guaranteed) that an adversarial example $\mathbf{x}$ around $\mathbf{x}_0$ exists, then we say this approach is *complete* (resp. *incomplete*).

As shown in Table 2.1, a sound verification approach is able to tell us how many test examples are guaranteed to be robust. On the other hand, a complete verification approach is able to tell us how many test examples are guaranteed not to be robust. The remaining examples can be neither verified nor unverified.

Table 2.1: Relation between the output and reality.

	output	reality
sound	verified	verified
unsound	verified	unknown
complete	not verified	not verified
incomplete	unknown	verified

2.1.2 Feature of the approaches

When we talk about robustness verification, we need to specify the norm we consider, the activation functions we used in the neural networks, and the layer structure of the networks. No approaches can be applied to all cases, i.e., they can only be restricted to certain norms or activation functions.

Norm: The norm serves to specify the distance between the input example and the adversarial attack. Principally, the norms we consider are l_1, l_2 and l_∞ norm and some approaches may be constrained to be used for certain norms.

Activation functions: Some verification approaches are only designed for semi-algebraic or piece-wise linear activation functions, such as ReLU or leaky-ReLU, while some can be used for broader set of functions.

Structure of NNs: Some approaches are only designed for single-layer NNs, some are only designed for DNNs. The main interest of the current research is that whether we can adapt our approaches to larger and more complicated NNs such as DEQs, CNNs, or RNNs.

2.1.3 Scalability of the approaches

According to the methodology and solver we use, different approaches can be applied to networks with different sizes, including the number of neurons in each layer and

the total number of layers (depth) of the network. Database determines the size of the outputs, which also influences the scalability of the approaches.

Network: There are some familiar neural networks that researchers often use as benchmarks to compare the performance of different methods:

- *ResNet*, short for Residual Network, a specific type of neural network introduced by [He *et al.* 2016];
- *AlexNet*, a CNN architecture primarily designed by [Krizhevsky *et al.* 2012];
- *LeNet*, a CNN architecture proposed by [Lecun *et al.* 1998];
- *MobileNet* [Howard *et al.* 2017], a simple but efficient and not very computationally intensive CNN structure for mobile vision applications;
- *GoogleNet*, a 22-layer CNN architecture which is a variant of the Inception Network [Szegedy *et al.* 2015];
- *DenseNet*[1] [Huang *et al.* 2017a], a network architecture where each layer is directly connected to every other layer in a feed-forward fashion;
- *VGGNet* proposed by [Simonyan & Zisserman 2014], where "VGG" stands for Visual Geometry Group, a standard deep CNN architecture with multiple layers.

Dataset: The datasets we consider in this chapter mainly includes the follows:

- The *Collision Detection (CD)* dataset [Ehlers 2017], which attempts to predict whether two vehicles with parameterized trajectories are going to collide;
- The *Airborne Collision Avoidance System (ACAS)* dataset [Katz *et al.* 2017], which is a neural network based advisory system recommending horizontal manoeuvres for an aircraft in order to avoid collisions, based on sensor measurements;
- The *Modified National Institute of Standards and Technology (MNIST)* dataset of handwritten digits [Lecun 1988], which is the most commonly used benchmarks for comparing different machine learning approaches;
- The *Canadian Institute For Advanced Research (CIFAR-10)* [2] dataset, which is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research;

[1] https://github.com/liuzhuang13/DenseNet
[2] https://www.cs.toronto.edu/~kriz/cifar.html

- The *ImageNet*[3] project, which is a large visual database designed for use in visual object recognition software research. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided.

In Table 2.2, we show the largest size of neural networks the approaches is applied to, to illustrate the scalability of the approaches. Many approaches presented in this chapter are referred to related surveys [Huang *et al.* 2020, Li *et al.* 2020, Urban & Miné 2021, Liu *et al.* 2021, Meng *et al.* 2022]. We also refer to the book by [Albarghouthi 2021] for a broad view of robustness verification for constraint-based, abstraction-based and optimization-based approaches.

2.2 Deterministic Exact Approaches (type A)

Deterministic guarantees are achieved by transforming a verification problem into a set of constraints (with or without optimization objectives) so that they can be solved with a constraint solver. The name "deterministic" comes from the fact that solvers often return a deterministic answer to a query, i.e., either satisfactory or unsatisfactory. This is based on the current success of various constraint solvers such as SAT solvers, Satisfiability Modulo Theories (SMT) solvers, LP solvers, and MILP solvers.

2.2.1 Satisfiability modulo theories (SMT) problem

The Boolean satisfiability problem (SAT) determines if, given a Boolean formula, there exists an assignment to the Boolean variables such that the formula is satisfiable. Based on SAT, the satisfiability modulo theories (SMT) problem determines the satisfiability of logical formulas with respect to combinations of background theories expressed in classical first-order logic with equality. SMT solvers often have good performance on problems that can be represented as a Boolean combination of constraints over other variable types. Typically, an SMT solver combines a SAT solver with specialised decision procedures for other theories. The theories we consider in the context of neural networks include the theory of real numbers and the theory of integers. For both SAT and SMT problems, there are sophisticated and open-source solvers that can automatically answer the queries about the satisfiability of the formulas.

Two SMT solvers Reluplex [Katz *et al.* 2017] and Planet [Ehlers 2017] were put forward to verify neural networks on properties expressible with SMT constraints. In the verification of networks, they adapt linear arithmetic over real numbers, in which an atom (i.e., the most basic expression) is of the form $\sum_{i=1}^{n} a_i x_i \leq b$, where a_i and b are real numbers. In both Reluplex and Planet, they use the architecture of the Davis-Putnam-Logemann-Loveland (DPLL) algorithm in splitting cases and ruling out conflict clauses, while they differ slightly in dealing with the intersection.

[3] https://www.image-net.org/index.php

For Reluplex, the approach performs tree search in the function space. It extends the simplex algorithm, a standard algorithm for solving LP instances, to support ReLU networks. Through the classical pivot operation, it first looks for a solution for the linear constraints, and then applies the rules for ReLU to satisfy the ReLU relation for every node. Conversely, Planet integrates with a SAT solver for tree search in the function space. The objective of the search is to find an activation pattern of ReLU networks that maps an input in $\mathcal{X}$ to an output not in $\mathcal{Y}$. It combines optimization-based filtering and pruning in the search process, and uses linear approximation to overapproximate the neural networks, and manages the condition of ReLU and max-pooling nodes with a logic formula.

[Bunel *et al.* 2018] uses Branch and Bound (BaB) to compute the output bounds of a network. It has a modularized design that can serve as a unified framework that can support other methods such as Reluplex and Planet.

[Narodytska 2018, Narodytska *et al.* 2018] propose to verify properties of a class of neural networks (i.e., binarised neural networks) in which both weights and activations are binary, by reduction to the well-known Boolean satisfiability. Using this Boolean encoding, they leverage the power of modern SAT solvers, along with a proposed counterexample-guided search procedure, to verify various properties of these networks. A particular focus is on the robustness to adversarial perturbations. The experimental results demonstrate that this approach scales to medium-size DNNs used in image classification tasks.

2.2.2 Mixed integer linear programming (MILP)

Linear programming (LP) is a technique for optimizing a linear objective function, subject to linear equality and inequality constraints. All the variables in an LP are real, if some of the variables are integers, the problem becomes a mixed integer linear programming (MILP) problem. It is noted that, while LP can be solved in polynomial time, MILP is NP-hard.

NSVerify by [Lomuscio & Maganti 2017, Akintunde *et al.* 2018] encodes the network as a set of mixed integer linear constraints. It solves a feasibility problem without an objective function, and tries to find a counterexample for the verification problem. However, it is not efficient to simply use MILP to verify networks, or to compute the output range. [Cheng *et al.* 2017] developed a number of MILP encoding heuristics to speed up the solving process, and moreover, parallelization of MILP-solvers is used, resulting in an almost linear speed-up in the number (up to a certain limit) of computing cores in experiments. MIPVerify by [Tjeng *et al.* 2019] also encodes the network as a set of mixed integer linear constraints. There are two differences between MIPVerify and NSVerify. Firstly, MIPVerify determines the bounds on the nodes to tighten the constraints. Secondly, MIPVerify solves an adversarial problem that tries to estimate the maximum allowable disturbance on the input side.

Sherlock by [Dutta *et al.* 2018] alternately conducts a local and global search to efficiently calculate the output range. In a local search phase, Sherlock uses gradient

descent method to find a local maximum (or minimum). While in a global search phase, it encodes the problem with MILP to check whether the local maximum (or minimum) is the global output range. Additionally, the branch and bound algorithm presented by [Bunel *et al.* 2018] claims that both SAT/SMT-based and MILP-based approaches can be regarded as its special cases.

2.2.3 Layer-by-layer refinement

[Huang *et al.* 2017b] develops an automated verification framework for feed-forward multi-layer neural networks based on SMT solver. The key features of this framework are that it guarantees a misclassification being found if it exists, and that it propagates the analysis layer-by-layer, i.e., from the input layer to, in particular, the hidden layers, and to the output layer. It supports any activation function.

In this work, local robustness is defined as the invariance of a classifier's outcome to perturbations within a small neighborhood of an original input. To be more specific, its verification algorithm uses single-/multi-path search to exhaustively explore a finite region of the vector spaces associated with the input layer or the hidden layers, and a layer-by-layer refinement is implemented using the Z3 solver [de Moura & Bjørner 2008] to ensure that the local robustness of a deeper layer implies the robustness of a shallower layer. The methodology is implemented in the software tool called DLV (Deep Learning Verification)[4], and evaluated on image benchmarks such as MNIST, CIFAR-10, and ImageNet. Though the complexity is high, it scales to work with state-of-the-art networks such as VGGNet [Simonyan & Zisserman 2014].

2.2.4 Reduction to a two-player turn-based game

In DeepGame by [Wu *et al.* 2020], two variants of pointwise robustness are studied:

- the *maximum safe radius (MSR)* problem, which computes the minimum distance to an adversarial example for a given input sample;
- the *feature robustness (FR)* problem, which aims to quantify the robustness of individual features to adversarial perturbations.

It demonstrates that, under the assumption of Lipschitz continuity, both problems can be approximated using finite optimization by discretizing the input space, and the approximation has provable guarantees, i.e., the error is bounded. It subsequently reduces the resulting optimization problems to the solution of a two-player turn-based game, where Player I selects features and Player II perturbs the image within the feature. While Player II aims to minimize the distance to an adversarial example, Player I can be cooperative or competitive depending on the optimization objective. An anytime approach is employed to solve the games, in the sense of approximating the value of a game by monotonically improving its upper and

[4] https://github.com/verideep/dlv

lower bounds. The Monte-Carlo tree search algorithm is applied to compute upper bounds for both games, and the Admissible A^* and Alpha-Beta Pruning algorithms are, respectively, used to compute lower bounds for the MSR and FR games.

2.2.4.1 Global optimization based approaches

[Ruan *et al.* 2018] shows that most known layers of DNNs are Lipschitz continuous, and presents a verification approach based on global optimization. For a single dimension, an algorithm is presented to always compute the lower bounds (by utilizing the Lipschitz constant) and eventually converge to the optimal value. Based on this single-dimensional algorithm, the algorithm for multiple dimensions is to exhaustively search for the best combinations. The algorithm is able to work with state-of-the-art DNNs, but is restricted by the number of dimensions to be perturbed.

[Ruan *et al.* 2019] focuses on the Hamming distance, and study the problem of quantifying the global robustness of a trained DNNs, where global robustness is defined as the expectation of the maximum safe radius over a testing dataset. They propose an approach to iteratively generate lower and upper bounds on the network's robustness. The approach is anytime, i.e., it returns intermediate estimations that are gradually, but strictly, improved as the computation proceeds. It is also tensor-based, i.e., the computation is conducted over a set of inputs simultaneously, instead of one by one, to enable efficient GPU computation. Finally, it has provable guarantees, i.e., the estimations converge to their optimal values.

2.3 Deterministic Approximation Approaches (type B)

The approaches surveyed in this subsection consider the computation of a lower (or by duality, an upper) bound, and are able to claim the sufficiency of achieving properties. Although these approaches can only have a bounded estimation to the value of some variable, they are able to work with larger models, e.g., up to 10000 hidden neurons. Another advantage is their potential to avoid floating point issues in existing constraint solver implementations. Actually, most state-of-the-art constraint solvers implementing floating-point arithmetic only give approximate solutions, which may not be the actual optimal solution or may even lie outside the feasible space, see [Neumaier & Shcherbina 2004]. Indeed, it may happen that a solver wrongly claims the satisfiability or unsatisfiability of a property. For example, [Dutta *et al.* 2018] reports several false positive results in Reluplex, and mentions that this may come from an unsound floating point implementation.

2.3.1 Abstract interpretation

Abstract interpretation is a theory of sound approximation of the semantics of computer programs [Cousot & Cousot 1977]. It has been used in static analysis to verify properties of a program without actually being run. The basic idea of

abstract interpretation is to use abstract domains (represented as boxes, zonotopes, polyhedra, and ellipsoids, etc.) to overapproximate the computation of a set of inputs. Its application has been explored in a few approaches [Gehr *et al.* 2018, Mirman *et al.* 2018, Li *et al.* 2019].

In abstract interpretation, it is important to choose a suitable abstract domain because it determines the efficiency and precision of the abstract interpretation. In practice, a certain type of special shapes is used as the abstraction elements. Formally, an abstract domain consists of shapes expressible as a set of logical constraints. The most popular abstract domains for the Euclidean space abstraction include boxes, zonotopes, polyhedra, and ellipsoids, which are defined as follows:

- *Box*: a box B contains logical constraints of form $a \leq x_i \leq b$. For each variable x_i, B contains at most one constraint with x_i;
- *Zonotope*: a zonotope Z consists of constraints of form $z_i = a_i + \sum_{j=1}^{m} b_{ij}\varepsilon_j$, where a_i, b_{ij} are real constants and $\varepsilon_j \in [l_j, u_j]$. The conjunction of these constraints expresses a center-symmetric polyhedron in the Euclidean space;
- *Polyhedron*: a polyhedron P has constraints in the form of linear inequalities, i.e., $\sum_{i=1}^{n} a_i x_i \leq b$, and it gives a closed convex polyhedron in the Euclidean space;
- *Ellipsoid*: an ellipsoid E has constraints of form $\|\mathbf{Q}\mathbf{x}+\mathbf{b}\|_2 \leq 1$ parametrized by $\mathbf{Q} \in \mathbb{S}^n$ and $\mathbf{b} \in \mathbb{R}^n$, where $\mathbb{S}^n$ is the set of positive semidefinite matrices of size $n \times n$.

Example. The unit box with $0 \leq x_i \leq 1$ is naturally a zonotope ($z_i = x_i \in [0, 1]$) and a polyhedron ($x_i \leq 1, -x_i \leq 0$). The unit ball with $\|\mathbf{x}\|_2 \leq 1$ is an Ellipsoid.

The approaches based on abstract interpretation often perform layer-by-layer reachability analysis to compute the reachable set of the output region. For example, [Xiang *et al.* 2018b] computes the exact reachable set for networks with only ReLU activations. The key insight is that if the input set to a ReLU function is a union of polytopes, then the output reachable set is also a union of polytopes. As there is no overapproximation, this method is sound and complete. However, since the number of polytopes grows exponentially with each layer, this method does not scale.

MaxSens by [Xiang *et al.* 2018a] partitions the input domain into small grid cells, and loosely approximates the reachable set for each grid cell considering the maximum sensitivity of the network at each grid cell. Sensitivity of a function is equivalent to the Lipschitz constant of the function. The union of those reachable sets is the output reachable set. The finer the partition, the tighter the output reachable set. MaxSens works for any activation function and its running time scales well with the number of layers but poorly with the input dimension.

AI2 and ERAN (updated version of AI2) by [Gehr *et al.* 2018] uses representations that overapproximate the reachable set. It trades precision of the reachable

set for scalability of the algorithm. It works for any piecewise linear activation functions, such as ReLU and leaky ReLU. Due to its approximation, the number of geometric objects to be traced during layer-by-layer analysis is greatly reduced. Although AI2 is not complete, it scales well.

2.3.2 Convex optimization based methods

Convex optimization is to minimize convex functions over convex sets. Most neural network functions are not convex (even if they are convex, they can be very complicated), and hence an approximation is needed for the related computation. [Wong & Kolter 2018] proposes a method called ConvDual to learn deep ReLU-based classifiers that are provably robust against norm-bounded adversarial perturbations on the training data. The approach works with interval property, but not with reachability property. It may flag some non-adversarial examples as adversarial examples. The basic idea of ConvDual is to consider a convex outer overapproximation of the set of activations which is reachable through a norm-bounded perturbation, and then develop a robust optimization procedure that minimizes the worst case loss over this outer region via a linear program. Crucially, it is shown that the dual problem to this linear program can be represented itself as a deep network similar to the backpropagation network, leading to very efficient optimization approaches that produce guaranteed bounds on the robust loss. The approach is illustrated on a number of tasks with robust adversarial guarantees. For example, for MNIST dataset, they produce a convolutional classifier that provably has less than 5.8% test error for any adversarial attack with bounded l_∞ norm less than $\varepsilon = 0.1$.

[Dvijotham *et al.* 2018] takes a different formulation of the dual problem, i.e., applying Lagrangian relaxation on the optimization, to solve a Lagrangian relaxation of the optimization problem to obtain bounds on the output. This is to avoid working with constrained nonconvex optimization problem. The dual problem is formulated as an unconstrained convex optimization problem, which can be computed efficiently. This approach is anytime and works for any activation function., i.e., it can be stopped at any time and a valid bound on the maximum violation can be obtained. The authors develop specialized verification algorithms with provable tightness guarantees under special assumptions and demonstrate the practical significance of the general verification approach on a variety of verification tasks.

[Raghunathan *et al.* 2018a] uses a semidefinite relaxation to compute Lipschitz constants of neural networks. Firstly, the authors propose a method based on semidefinite relaxation that outputs a certificate that no attack can force the error to exceed a certain value for a given network and test input. Secondly, as this certificate is differentiable, they jointly optimize it with the network parameters, providing an adaptive regularizer that encourages robustness against all attacks. On MNIST dataset, this approach produces a network and a certificate that no attack that perturbs each pixel by at most $\varepsilon = 0.1$ can cause more than 35% test error. It works for any activation function that is differentiable almost everywhere

but only for neural networks with only one hidden layer.

[Brown *et al.* 2022] develop an exact, convex formulation of verification as a *completely positive program* (CPP), which are linear optimization problems over the cone of completely positive matrices. This gives a clean separation between the two competing desiderata of accuracy and tractability in relaxed verification. The authors also provide analysis explaining how properties of the CPP formulation evolve when the complete positivity constraint is relaxed. They find that many of the favorable properties of the CPP formulation are retained in SDP relaxations, showing that it is a convenient starting point for constructing tight SDP-based verifiers.

2.3.3 Linear approximation of ReLU networks

FastLin/FastLip by [Weng *et al.* 2018a] analyses the ReLU networks on both interval property and Lipschitzian property. For interval property, they consider linear approximation over those ReLU neurons that are uncertain on their status of being activated or deactivated. For Lipschitzian property, they use the gradient computation for the approximations. FastLin computes a certified lower bound on the allowable input disturbance for ReLU networks using a layer-by-layer approach and binary search in the input domain. Based on FastLin, FastLip computes the bounds on the activation functions, and further estimates the local Lipchitz constant of the network. In general, FastLin is more scalable than FastLip, while FastLip provides better solutions for l_1 norm.

CROWN [Zhang *et al.* 2018] is a general framework to certify robustness of neural networks with general activation functions for given input data points. It generalizes the interval property computation algorithm in FastLin/FastLip by allowing the linear expressions for the upper and lower bounds to be different and enabling to work with other activation functions such as sigmoid and tanh. The Lipschitzian property computation is improved by [Zhang *et al.* 2019]. The authors propose a recursive algorithm, called RecurJac, to compute both upper and lower bounds for each element in the Jacobian matrix of a neural network with respect to network inputs, and the network can contain a wide range of activation functions, not limited to ReLU.

CROWN-IBP [Zhang *et al.* 2020] is a certified adversarial training method combining the fast interval bound propagation (IBP) bounds in a forward bounding pass and a tight linear relaxation based bound, CROWN, in a backward bounding pass. CROWN-IBP is computationally efficient and consistently outperforms IBP baselines on training verifiably robust neural networks.

α-CROWN [Xu *et al.* 2021] uses the backward mode linear relaxation based perturbation analysis (LiRPA) to replace LP during the BaB process, which can be efficiently implemented on the typical machine learning accelerators such as GPUs and TPUs. However, unlike LP, LiRPA when applied naively can produce much weaker bounds and even cannot check certain conflicts of subdomains during splitting, making the entire procedure incomplete after BaB.

β-CROWN [Wang *et al.* 2021d] is an improved approach based on CROWN, that can fully encode neuron splits via optimizable parameters β constructed from either primal or dual space, and can be used for both complete and incomplete verification depending on whether we execute early stop. Applied to complete robustness verification benchmarks, β-CROWN with BaB is up to three orders of magnitude faster than LP-based BaB methods, and is notably faster than all existing approaches while producing lower timeout rates. By terminating BaB early, their method can also be used for efficient incomplete verification. Compared to the typically tightest but very costly SDP-based incomplete verifiers, they obtain higher verified accuracy with three orders of magnitudes less verification time.

CNN-cert [Boopathy *et al.* 2019] generalizes FastLin/FastLip and CROWN to general convolutional neural networks. It can handle various architectures including convolutional layers, max-pooling layers, batch normalization layer, residual blocks, as well as general activation functions. The efficiency of the approach is shown by exploiting the special structure of convolutional layers. CNN-cert achieves up to 17 and 11 times of speed-up compared to the state-of-the-art verification algorithms (e.g. FastLin, CROWN) and 366 times of speed-up compared to the dual-LP approach, and in the meantime, it obtains similar or even better verification bounds.

2.3.4 Interval analysis

The interval arithmetic by [Wang *et al.* 2018a] is leveraged to compute rigorous bounds on the outputs of a DNN, i.e., interval property. The key idea is that, given the ranges of operands, an overestimated range of the output can be computed by using only the lower and upper bounds of the operands. Starting from the first hidden layer, this computation can be conducted to the output layer. Beyond this explicit computation, symbolic interval analysis along with several other optimizations are also developed to minimize overestimations of output bounds. These methods are implemented in ReluVal [Wang *et al.* 2018b], a system for formally checking security properties of ReLU-based DNNs. ReluVal uses symbolic interval analysis along with search to minimize overapproximations of the output sets. During the search process, ReluVal iteratively bisects its input range. This process is called *iterative interval refinement*, which is also used in BaB by [Bunel *et al.* 2018]. [Wang *et al.* 2018a] improves ReluVal by two major modifications. Firstly, it introduces symbolic linear relaxation which is tighter than symbolic interval analysis in computing the reachable sets. Secondly, it introduces direct constraint refinement to perform the search more efficiently. An advantage of this approach, comparing to constraint solving based approaches, is that it can be easily parallelizable. In general, interval analysis is close to the interval-based abstract interpretation (see Section 2.3.1).

[Peck *et al.* 2017] takes steps towards a formal characterization of adversarial perturbations by deriving lower bounds on the magnitudes of perturbations necessary to change the classification of neural networks. The proposed bounds can be computed efficiently, requiring time at most linear in the number of parameters

and hyperparameters of the model for any given sample. This makes this approach suitable for use in model selection, when one wishes to find out which of several proposed classifiers is most robust to adversarial perturbations. The approach may also be used as a basis for developing techniques to increase the robustness of classifiers, since it enjoys the theoretical guarantee that no adversarial perturbation could possibly be any smaller than the quantities provided by the bounds. The authors experimentally verify the bounds on the MNIST and CIFAR-10 datasets and find no violations. Additionally, the experimental results suggest that very small adversarial perturbations may occur with non-zero probability on natural samples.

2.4 Statistical Approximation Approaches (type C)

This subsection reviews a few approaches aiming to achieve statistical guarantees on their results, by claiming that the satisfiability of a property, or a value is a lower bound of another value, with certain probability.

2.4.1 Lipschitz constant estimation by extreme value theory

[Weng *et al.* 2018c] proposes an Extreme Value Theory (EVT) based robustness score for large-scale deep neural networks, called CLEVER (Cross-Lipschitz Extreme Value for Network Robustness). It estimates the Lipschitz constant, i.e., it works with Lipschitzian property. And it estimates the robustness lower bound by sampling the norm of gradients and fitting a limit distribution using EVT. Compared with the existing robustness evaluation approaches, their metric is attack-independent and applicable to any neural network classifiers. CLEVER has strong theoretical guarantees and is computationally feasible for large neural networks. However, as argued by [Goodfellow 2018], their evaluation approach can only find statistical approximation of the lower bound, i.e., their approach has a soundness problem. [Weng *et al.* 2018b] proposes two extensions on CLEVER robustness score. Firstly, the authors provide a new formal robustness guarantee for classifier functions that are twice differentiable. They apply EVT on the new formal robustness guarantee and the estimated robustness is called second-order CLEVER score. Secondly, they discuss how to handle gradient masking, a common defensive technique, using CLEVER with Backward Pass Differentiable Approximation (BPDA). With BPDA applied, CLEVER can evaluate the intrinsic robustness of neural networks of a broader class - networks with non-differentiable input transformations. They demonstrate the effectiveness of CLEVER with BPDA in experiments on a 121-layer Densenet [Huang *et al.* 2017a] model trained on the ImageNet dataset.

2.4.2 Robustness estimation

[Bastani *et al.* 2016] proposes an Iterative Linear Programming (ILP), to encode the network as a set of linear constraints by linearizing the network at a reference point. The optimization problem in ILP is an adversarial problem that tries

to estimate the maximum allowable disturbance on the input side. It iteratively solves the optimization. This method is not complete as it only considers one linear segment of the network. The authors propose two statistics of robustness to measure the frequency and the severity of adversarial examples respectively. Both statistics are based on a parameter ε, which is the maximum radius within which no adversarial examples exist. The computation of these statistics is based on the local linearity assumption which holds when ε is small enough. The authors show how these metrics can be used to evaluate the robustness of deep neural networks with experiments on the MNIST and CIFAR-10 datasets. Their algorithm generates more informative estimates of robustness metrics compared to estimates based on existing algorithms. Except for the application of the ReLU activation function which is piece-wise linear, this assumption can be satisfied by the existence of the Lipschitz constant as shown by [Ruan *et al.* 2018].

2.5 Characteristic analysis of each verification method

As introduced in Section 2.1, we summarize in this section the features and characteristics of each method we showed in the previous section.

Table 2.2: Characteristic analysis of various approaches.

Method & Name	Type	Completeness	Soundness	Activations	Norm	Scalability	Dataset	Network
[Katz *et al.* 2017], Reluplex	A	✓	✓	ReLU	L_∞	6×300	ACAS	DNN
[Ehlers 2017], Planet		✓	✓	ReLU	L_∞	1341	CD, MNIST	CNN
[Bunel *et al.* 2018], BaB		✓	✓	ReLU	L_∞	4×25	CD, ACAS	DNN
[Narodytska 2018]		✓	✓	sign	L_∞	4×100	MNIST	Binary
[Lomuscio & Maganti 2017], NSVerify		✓	✓	ReLU	L_∞	$(4608, 128)$	MNIST	CNN
[Tjeng *et al.* 2019], MIPVerify		✓	✓	ReLU	L_∞	3×20, ResNet	MNIST, CIFAR-10	DNN, CNN
[Akintunde *et al.* 2018]		✓	✓	ReLU	L_∞	29×81	TwinStream	DNN
[Cheng *et al.* 2017]		✓	✓	ReLU	L_∞	4×50	MNIST	DNN
[Dutta *et al.* 2018], Sherlock		✓	✓	ReLU	L_1, L_∞	$(200, 100, 50)$	MNIST	DNN
[Huang *et al.* 2017b]		✓	✓	ReLU	L_1, L_2	6e5, 1e6, VGGNet	CIFAR-10, ImageNet	CNN
[Wu *et al.* 2020]		✓	✓	ReLU	L_1, L_2	$256, (3, 3, 128)$	MNIST, CIFAR-10	DNN, CNN
[Ruan *et al.* 2018]		✓	✓	ReLU	L_∞	19×256	ACAS, MNIST	CNN
[Ruan *et al.* 2019]		✓	✓	ReLU	L_0	AlexNet, VGGNet, ResNet	MNIST, ImageNet	DNN
[Xiang *et al.* 2018a]	B	✗	✓	monotonic	L_∞	$2 \times 5 \times 2$	robotic	DNN
[Gehr *et al.* 2018], AI2, ERAN		✗	✓	ReLU	L_∞	9×200, LeNet	MNIST, CIFAR-10	DNN, CNN
[Wong & Kolter 2018]		✗	✓	ReLU	L_∞	4×100, ConvNet	MNIST	DNN, CNN
[Dvijotham *et al.* 2018]		✗	✓	L-differentiable	L_2	1-layer	MNIST, CIFAR-10	DNN
[Raghunathan *et al.* 2018b], SDP-cert		✗	✓	ReLU	L_∞	$500, (200, 100, 50)$	MNIST	DNN
[Weng *et al.* 2018a], FastLin/FastLip		✗	✓	ReLU	L_1, L_2, L_∞	6×2048	MNIST, CIFAR-10	DNN
[Zhang *et al.* 2018], CROWN		✗	✓	ReLU	L_1, L_2, L_∞	6×2048	MNIST, CIFAR-10	DNN
[Zhang *et al.* 2019], RecurJac		✗	✓	ReLU, sigmoid	L_2, L_∞	10×2048	MNIST, CIFAR-10	DNN
[Zhang *et al.* 2020], CROWN-IBP		✗	✓	ReLU	L_∞	10×128	MNIST, CIFAR-10	CNN
[Xu *et al.* 2021], α-CROWN		✗	✓	ReLU	L_∞	ResNet	MNIST, CIFAR-10	DNN, CNN
[Wang *et al.* 2021d], β-CROWN		✗	✓	ReLU	L_∞	6244	CIFAR-10	CNN
[Boopathy *et al.* 2019], CNN-cert		✗	✓	ReLU	L_1, L_2, L_∞	LeNet, ResNet	CIFAR-10, ImageNet	DNN, CNN
[Wang *et al.* 2018a]		✗	✓	ReLU	L_1, L_2, L_∞	2×512, 4804	ACAS, MNIST	DNN, CNN
[Wang *et al.* 2018b], ReluVal		✗	✗	ReLU	L_∞	2×512	ACAS, MNIST	DNN
[Peck *et al.* 2017]		✓	✗	ReLU	L_2	LeNet	MNIST, CIFAR-10	DNN, CNN
[Weng *et al.* 2018c], CLEVER	C	✗	✗	ReLU	L_2, L_∞	AlexNet, ResNet, MobileNet	CIFAR-10, ImageNet	DNN, CNN
[Weng *et al.* 2018b]		✗	✗	tanh	L_2	DenseNet	CIFAR-10, ImageNet	CNN
[Bastani *et al.* 2016]		✗	✗	ReLU	L_∞	LeNet, NiN	MNIST, CIFAR-10	DNN, CNN

Chapter 3

Convex Relaxations and Moment-SOS Hierarchy

Contents

For ReLU networks, many machine learning problems can be formulated as polynomial optimization problems (POP). The advantage of POPs is that, instead of finding local optimizers, there are many approaches to find global optimizers of POPs. However, those POPs are usually nonlinear and non-convex, which are NP-hard to solve. Fortunately, one is able to relax POPs to convex optimization problems such as Linear Programming (LP), Quadratic Programming (QP) and Semidefinite Programming (SDP). For those convex optimization problems, there are already various efficient solvers to calculate accurate optimal values. In this chapter, we first introduce the general primal-dual forms of those convex optimization problems. Then we discuss about the theoretical preliminaries of Lasserre's moment-SOS hierarchy, a systematic approach about convex relaxation of POPs.

3.1 Convex Optimization Problems

For convex problem, any of its local optimizer is a global optimizer, and there are already many polynomial-time algorithm to solve them efficiently.

3.1.1 Linear programming (LP)

Linear programming is a mathematical modeling technique in which a linear function is maximized or minimized when subjected to various linear constraints. This technique has been useful for guiding quantitative decisions in business planning, in industrial engineering, and in the social and physical sciences. An LP has a standard form

$$\begin{aligned} \max_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq 0, \end{aligned} \tag{LP}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n$ are known parameters.

3.1.2 Quadratic programming (QP)

Quadratic programming is the problem of optimizing a quadratic objective function and is one of the simplests form of non-linear programming. The objective function can contain bilinear or up to second order polynomial terms, and the constraints are linear and can be both equalities and inequalities. QP is widely used in image and signal processing, to optimize financial portfolios, to perform the least-squares method of regression, to control scheduling in chemical plants, and in sequential quadratic programming. A QP can be written in the standard form

$$\begin{aligned} \max_{\mathbf{x}\in\mathbb{R}^n} \quad & \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{a},\ \mathbf{B}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq 0, \end{aligned} \tag{QP}$$

where $\mathbf{A} \in \mathbb{R}^{p\times n}, \mathbf{a} \in \mathbb{R}^p, \mathbf{B} \in \mathbb{R}^{q\times n}, \mathbf{b} \in \mathbb{R}^q, \mathbf{Q} \in \mathbb{R}^{n\times n}, \mathbf{c} \in \mathbb{R}^n$ are known parameters.

3.1.3 Semidefinite programming (SDP)

In semidefinite programming we minimize a linear function subject to the constraint that an affine combination of symmetric matrices is positive semidefinite. Such a constraint is nonlinear and nonsmooth, but convex, so positive definite programs are convex optimization problems. Semidefinite programming unifies several standard problems (eg, linear and quadratic programming) and finds many applications in engineering. Although semidefinite programs are much more general than linear programs, they are just as easy to solve. Most interior-point methods for linear programming have been generalized to semidefinite programs. As in linear program-

ming, these methods have polynomial worst-case complexity, and perform very well in practice. An SDP has a standard (primal) form

$$\begin{aligned} \min_{X\in\mathbb{S}^n} \quad & \langle \mathbf{C}, \mathbf{X}\rangle \\ \text{s.t.} \quad & \langle \mathbf{A}_i, \mathbf{X}\rangle = b_i,\ i=[1,m];\ \mathbf{X}\succeq 0, \end{aligned} \tag{SDP-primal}$$

where $\mathbb{S}^n$ denotes the the space of all real symmetric $n\times n$ matrices, and $\langle\cdot,\cdot\rangle$ denotes the Frobenius scalar product in $\mathbb{S}^n$. Sometimes we also use the the dual semidefinite program

$$\begin{aligned} \max_{\mathbf{y}\in\mathbb{R}^m} \quad & \mathbf{b}^T\mathbf{y} \\ \text{s.t.} \quad & \sum_{i=1}^m y_i\mathbf{A}_i \le \mathbf{C}. \end{aligned} \tag{SDP-dual}$$

3.2 Polynomial Optimization and Lasserre's Hierarchy

In this section we introduce the standard form of polynomial optimization problems and Lasserre's hierarchy which can be used to relax any POP to a series of SDPs, and has already many successful applications in and outside optimization.

3.2.1 Polynomial optimization problem (POP)

Let $\mathbb{R}[\mathbf{x}]$ be the vector space of real-valued polynomials in variable $\mathbf{x}$. A general polynomial optimization problem in n variables takes the standard form

$$\begin{aligned} f_1^* = \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \ge 0,\ i\in[p], \end{aligned} \tag{POP}$$

where $f, g_i \in \mathbb{R}[\mathbf{x}]$. This is a special case of nonlinear optimization problem where ojective function and all constraints are polynomials. If there is an equality constraint $g(\mathbf{x})=0$, we can replace $g=0$ by inequality constraints $g\ge 0$ and $-g\ge 0$ and retrieve the standard form (POP).

Definition 3.1. A subset $\mathbf{K}$ of $\mathbb{R}^n$ is called a *semialgebraic set* if $\mathbf{K}$ is defined by a finite sequence of polynomial equalities and inequalities. A function $f:\mathbb{R}^n\to\mathbb{R}$ is called a *semialgebraic function* if its graph is a semialgebraic set, i.e., the set $\{(\mathbf{x}, f(\mathbf{x})) : \mathbf{x}\in\mathbb{R}^n\}\subseteq\mathbb{R}^{n+1}$ is a semialgebraic set.

With the notations in (POP), we define a semialgebraic set $\mathbf{K} := \{\mathbf{x} : g_i(\mathbf{x}) \ge 0, i\in[p]\}$. The standard form (POP) can be written equivalently as

$$f_2^* = \min_{\mu\in\mathcal{P}(\mathbf{K})} \int f(\mathbf{x})\mathrm{d}\mu, \tag{MEAS}$$

where $\mathcal{P}(\mathbf{K})$ is the space of Borel probability measures with support contained in the semialgebraic set $\mathbf{K}$. Indeed, let f_1^* and f_2^* be the optimal solution of (POP) and (MEAS) respectively. Since $f(\mathbf{x}) \geq f_1^*$ for all $\mathbf{x} \in \mathbf{K}$, we have

$$f_2^* = \min_{\mu \in \mathcal{P}(\mathbf{K})} \int f(\mathbf{x}) \mathrm{d}\mu \geq \min_{\mu \in \mathcal{P}(\mathbf{K})} \int f_1^* \mathrm{d}\mu = f_1^*. \tag{3.1}$$

Conversely, suppose that $\mathbf{x}^*$ is the optimizer of (POP), i.e., $f(\mathbf{x}^*) = f_1^*$. Let $\delta_{\mathbf{x}^*}$ be the Dirac measure at point $\mathbf{x}^*$, then we have

$$f_2^* = \min_{\mu \in \mathcal{P}(\mathbf{K})} \int f(\mathbf{x}) \mathrm{d}\mu \leq \int f(\mathbf{x}) \mathrm{d}\delta_{\mathbf{x}^*} = f(\mathbf{x}^*) = f_1^*. \tag{3.2}$$

Hence $f_1^* = f_2^*$ and problem (POP) is equivalent to (MEAS). Denote by $f(\mathbf{x}) = \sum_\alpha f_\alpha \mathbf{x}^\alpha$, where $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$ for $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_n]^T$. Then $\int f(\mathbf{x}) \mathrm{d}\mu = \sum_\alpha f_\alpha y_\alpha$, where $y_\alpha = \int \mathbf{x}^\alpha \mathrm{d}\mu$. By replacing the decision variables in (POP) by probability measure μ in place of $\mathbf{x}$, we transform the polynomial $f(\mathbf{x})$ into a linear combination of the moments of $\mathbf{x}$.

3.2.2 Moment and sum of squares

From the previous section we see that the polynomial optimization problem (POP) is equivalent to a problem whose objective is linear to the moments $\int \mathbf{x}^\alpha \mathrm{d}\mu$. In fact, it has been shown that the theory of moments is strongly related to the representation of nonnegative polynomials.

Definition 3.2. For polynomial $f \in \mathbb{R}[\mathbf{x}]$, the *Riesz linear functional* is the linear application $L_{\mathbf{y}} : f \mapsto \sum_\alpha f_\alpha y_\alpha$, where $\mathbf{y} = \{y_\alpha\}_\alpha$ is the *moment variables*.

Let $\mathbf{v}_d(\mathbf{x}) := [1, x_1, x_2, \dots, x_n, x_1^2, x_1 x_2, \dots, x_1 x_n, x_2 x_3, \dots, x_n^2, \dots, x_1^d, \dots, x_n^d]^T$ be the real-valued monomial basis of degree d.

Definition 3.3. The d-th order *moment matrix*, denoted by $\mathbf{M}_d(\mathbf{y})$, is defined as $\mathbf{M}_d(\mathbf{y}) = L_{\mathbf{y}}(\mathbf{v}_d(\mathbf{x})\mathbf{v}_d(\mathbf{x})^T)$. For a polynomial $f \in \mathbb{R}[\mathbf{x}]$, the d-th order *localizing matrix* of f, denoted by $\mathbf{M}_d(f\mathbf{y})$, is defined as $\mathbf{M}_d(f\mathbf{y}) = L_{\mathbf{y}}(f\mathbf{v}_d(\mathbf{x})\mathbf{v}_d(\mathbf{x})^T) = L_{\mathbf{y}}(f\mathbf{M}_d(\mathbf{y}))$.

Example. Consider $\mathbf{x} = [x_1, x_2]^T$ and polynomial $g(\mathbf{x}) = 1 - x_1^2 - x_2^2$. Then Riesz linear functional of g is $L_{\mathbf{y}}(g) = y_{00} - y_{20} - y_{02}$, the first order moment matrix reads

$$\mathbf{M}_1(\mathbf{y}) = \begin{pmatrix} y_{00} & y_{10} & y_{01} \\ y_{10} & y_{20} & y_{11} \\ y_{01} & y_{11} & y_{02} \end{pmatrix},$$

and the first order localizing matrix of g reads

$$\mathbf{M}_1(g\mathbf{y}) = \begin{pmatrix} y_{00} - y_{20} - y_{02} & y_{10} - y_{30} - y_{12} & y_{01} - y_{21} - y_{03} \\ y_{10} - y_{30} - y_{12} & y_{20} - y_{40} - y_{22} & y_{11} - y_{31} - y_{13} \\ y_{01} - y_{21} - y_{03} & y_{11} - y_{31} - y_{13} & y_{02} - y_{22} - y_{04} \end{pmatrix}.$$

As mentioned before, the moment and localizing matrices are related (and in fact, in duality with) nonnegativity of polynomials. Precisely, the moment matrix is the primal of sum of squares, and the localizing matrix is the primal of weighted sum of squares.

Definition 3.4. A polynomial $f \in \mathbb{R}[\mathbf{x}]$ is a *Sum of Squares (SOS)* if f can be written as $f = \sum_{i=1}^{r} u_i^2$ with $u_i \in \mathbb{R}[\mathbf{x}]$. For $f, g_i \in \mathbb{R}[\mathbf{x}]$, f is a *Weighted Sum of Squares (WSOS)* with respect to g_i if f can be written as $f = u_0^2 + \sum_{i=1}^{r} g_i u_i^2$ with $u_i \in \mathbb{R}[\mathbf{x}]$.

It is convenient for discussing to define some cones of nonnegative polynomials:

Definition 3.5. (1) The *SOS cone*, denoted by $\Sigma[\mathbf{x}]$, is the set of sum of squares in $\mathbb{R}[\mathbf{x}]$; the *SOS cone of order d*, denoted by $\Sigma_d[\mathbf{x}]$, is the set of sum of squares in $\mathbb{R}[\mathbf{x}]$ with degree at most $2d$.

(2) The *nonnegative cone*, denoted by $\mathcal{P}[\mathbf{x}]$, is the set of nonnegative polynomials in $\mathbb{R}[\mathbf{x}]$; the *nonnegative cone of order d*, denoted by $\mathcal{P}_d[\mathbf{x}]$, is the set of nonnegative polynomials in $\mathbb{R}[\mathbf{x}]$ with degree at most d.

Remark. It is obvious that any SOS polynomial is nonnegative. Unfortunately, the converse is false: not all nonnegative polynomials are SOSs. One famous example is *Motzkin polynomial*:

$$p(x, y) = x^4y^2 + x^2y^4 - 3x^2y^2 + 1. \tag{3.3}$$

The nonnegativity of Motzkin polynomial is due to arithmetic / geometric mean inequality, and the proof of why it is not an SOS is referred to [Motzkin 1967, page 205-224]. Hence we have a chain of inclusions

$$\mathbb{R}_+ = \Sigma_0[\mathbf{x}] \subsetneq \Sigma_1[\mathbf{x}] \subsetneq \Sigma_2[\mathbf{x}] \subsetneq \Sigma[\mathbf{x}] \subsetneq \mathcal{P}[\mathbf{x}]. \tag{3.4}$$

Looking back to polynomial optimization problem (POP), it is equivalent to

$$\begin{aligned} f^* = \max_{\lambda, \mathbf{x} \in \mathbf{K}} \quad & \lambda \\ \text{s.t.} \quad & f(\mathbf{x}) - \lambda \geq 0. \end{aligned} \tag{3.5}$$

In other words, we are searching for feasible polynomials $f(\mathbf{x}) - \lambda$ in the nonnegative cone $\mathcal{P}[\mathbf{K}]$. Recall that the semialgebraic set $\mathbf{K}$ is defined by polynomials g_i, i.e., $\mathbf{K} = \{\mathbf{x} : g_i(\mathbf{x}) \geq 0\}$. Instead of dealing with the nonnegative cone $\mathcal{P}[\mathbf{x}]$, it is much less complicated to investigate in the SOS cone $\Sigma[\mathbf{x}]$. In fact, it has been shown that the strictly positive elements in $\mathcal{P}[\mathbf{x}]$ are exactly WSOS under a technical *Archimedean assumption*.

Assumption 2. *For the compact semialgebraic set* $\mathbf{K}$*, there exists a real-valued*

polynomial $u(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ *such that the set* $\{\mathbf{x} \in \mathbb{R}^n : u(\mathbf{x}) \geq 0\}$ *is compact and*

$$u(\mathbf{x}) = \sigma_0(\mathbf{x}) + \sum_{i=1}^{p} g_i(\mathbf{x})\sigma_i(\mathbf{x}),\ \forall \mathbf{x} \in \mathbb{R}^n, \tag{3.6}$$

where the polynomials $\sigma_0, \sigma_i \in \Sigma[\mathbf{x}]$ *for* $i \in [p]$.

Remark. Assumption 2 can be easily satisfied by choosing properly the polynomial $u(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$. For example:

(i) if there is some g_i in (POP) such that the set $\{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \geq 0\}$ is compact;

(ii) if all the polynomials g_i are linear;

(iii) if there are binary inequalities $x_j^2 \geq x_j$ and $x_j \geq x_j^2$ for all $j \in [n]$.

In practice, we usually include a quadratic redundant constraint $g(\mathbf{x}) := M^2 - \|\mathbf{x}\|_2^2 \geq 0$ in the definition of $\mathbf{K}$ since $\mathbf{K}$ is compact and we can always find a large enough M such that $\|\mathbf{x}\|_2 \leq M$ for all $\mathbf{x} \in \mathbf{K}$.

Theorem 3.1. *Assume that Assumption 2 holds. Then the strictly positive polynomials defined in* $\mathbf{K}$ *is represented by WSOS w.r.t. polynomials* g_i*, i.e., for* $f \in \mathcal{P}[\mathbf{x}]$*, if* $f(\mathbf{x}) > 0$ *for any* $\mathbf{x} \in \mathbf{K}$*, then* $f = \sigma_0 + \sum_{i=1}^{p} g_i\sigma_i$ *for some* $\sigma_0, \sigma_i \in \Sigma[\mathbf{x}]$.

It is usually hard to determine polynomials σ_0 and σ_i, since we do not know in advance the degrees of these polynomials. Therefore, it is practical to approximate σ_0 and σ_i by increasing successively their degrees. With this idea in mind, we are able to derive a series of relaxations of problem (3.5):

$$\begin{aligned} \xi_d = \max_{\lambda} \quad & \lambda \\ \text{s.t.} \quad & f - \lambda = \sigma_0 + \sum_{i=1}^{p} g_i\sigma_i, \end{aligned} \tag{DenseSOS-d}$$

where $\sigma_0 \in \Sigma_d[\mathbf{x}], \sigma_i \in \Sigma_{d-\omega_i}[\mathbf{x}], \omega_i = \lceil \deg(g_i)/2 \rceil$ for $i \in [p]$. It is obvious that ξ_d is a monotone increasing sequence upper bounded by f^*, i.e., $\xi_d \leq f^*$, and we will see in the next section that ξ_d is in fact the dual optimal value of the d-th order moment relaxation.

3.2.3 Dense moment relaxation

In the Lasserre's hierarchy for optimization one approximates the *global* optimum of (POP) by solving a series of SDPs of increasing size. Each SDP is a semidefinite relaxation of (POP) in the form:

$$\begin{aligned} \rho_d = \min_{\mathbf{y}} \quad & L_{\mathbf{y}}(f) \\ \text{s.t.} \quad & \begin{cases} L_{\mathbf{y}}(1) = 1,\ \mathbf{M}_d(\mathbf{y}) \succeq 0, \\ \mathbf{M}_{d-\omega_i}(g_i\mathbf{y}) \succeq 0,\ i \in [p], \end{cases} \end{aligned} \tag{DenseMom-d}$$

where $\omega_i = \lceil \deg(g_i)/2 \rceil$.

The semidefinite program (DenseMom-d) is the d-th order *dense moment relaxation* of problem (POP). As a result, when the semialgebraic set $\mathbf{K}$ is compact, one obtains a monotone sequence of lower bounds $(\rho_d)_{d\in\mathbb{N}_+}$ with the property $\rho_d \uparrow f^*$ as $d \to \infty$ under Archimedean condition.

If we consider the dual of problem (DenseMom-d), we retrieve the formulation (DenseSOS-d). By weak duality, we have $\rho_d \leq \xi_d$. And generically, we have strong duality, i.e., $\rho_d = \xi_d$, as the following theorem states.

Theorem 3.2. *[Lasserre 2001] Let Assumption 2 holds. As $d \to \infty$, we have $\rho_d \uparrow f^*$. If $\mathbf{K}$ has a nonempty interior, then we have strong duality $\rho_d = \xi_d$.*

Example. Consider $\mathbf{x} = [x_1, x_2]^T$ and polynomials $f(\mathbf{x}) = x_1x_2, g(\mathbf{x}) = 1 - x_1^2 - x_2^2$. We minimize $f(\mathbf{x})$ over $\mathbb{R}^2$ under constraint $g(\mathbf{x}) \geq 0$. That is:

$$\begin{aligned} f^* = \min_{\mathbf{x}\in\mathbb{R}^2} \quad & x_1x_2 \\ \text{s.t.} \quad & 1 - x_1^2 - x_2^2 \geq 0. \end{aligned} \tag{3.7}$$

For $d = 1$, we have degree 2 moment variables $\mathbf{y} = [y_{00}, y_{01}, y_{10}, y_{20}, y_{11}, y_{02}]^T$ and $L_{\mathbf{y}}(f) = y_{11}$, $\omega = \lceil \deg(g)/2 \rceil = 1$. Then the moment and localizing matrices are of form

$$\mathbf{M}_1(\mathbf{y}) = \begin{pmatrix} y_{00} & y_{10} & y_{01} \\ y_{10} & y_{20} & y_{11} \\ y_{01} & y_{11} & y_{02} \end{pmatrix}, \ \mathbf{M}_0(g\mathbf{y}) = L_{\mathbf{y}}(g) = 1 - y_{20} - y_{02}.$$

The first order ($d = 1$) moment relaxation of (3.7) reads:

$$\begin{aligned} \rho^* = \min_{\mathbf{y}\in\mathbb{R}^6} \quad & y_{11} \\ \text{s.t.} \quad & y_{00} = 1, \ \mathbf{M}_1(\mathbf{y}) \succeq 0, \ 1 - y_{20} - y_{02} \geq 0 \end{aligned} \tag{3.8}$$

with optimal value $\rho^* = -1/2 = f^*$. It turns out that (3.8) is exactly Shor's relaxation applied to (3.7). In fact, for POPs whose polynomials are all of degree at most 2, the first-order moment relaxation is equivalent to Shor's relaxation.

The dual of the moment problem (3.8) can be formulated as the following SOS problem

$$\begin{aligned} \xi^* = \max_{\lambda} \quad & \lambda \\ \text{s.t.} \quad & x_1x_2 - \lambda = \sigma_0 + \sigma_1(1 - x_1^2 - x_2^2), \text{ where } \sigma_0 \in \Sigma_1[\mathbf{x}], \sigma_1 \geq 0, \end{aligned} \tag{3.9}$$

with optimal value $\xi^* = -1/2 = \rho^*$ and $\sigma_0 = (x_1 + x_2)^2/2, \sigma_1 = 1/2$.

3.2.4 Sparse moment relaxation

The relaxation (DenseMom-d) is called *dense* moment relaxation since we do not exploit any possible sparsity pattern in (POP). Therefore, if one solves (DenseMom-d) with interior point methods (as all current SDP solvers do), the dense relaxation is strongly limited to POPs of modest size. Indeed the d-th order dense moment relaxation (DenseMom-d) involves $\binom{n+2d}{2d} = O(n^{2d})$ moment variables and a moment matrix $M_d(\mathbf{y})$ of size $\binom{n+d}{d} = O(n^d)$ at fixed d. Fortunately, large-scale POPs often exhibit some structured sparsity patterns which can be exploited to yield a *sparse version* of (DenseMom-d). As a result, wider applications of Lasserre's hierarchy have been possible.

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

Assumption 3. *(a) The function f is a sum of polynomials, each involving variables of only one subset, i.e., $f(\mathbf{x}) = \sum_{k=1}^m f_k(\mathbf{x}_{I_k})$;*

(b) Each constraint also involves variables of only one subset, i.e., $g_i \in \mathbb{R}[\mathbf{x}_{I_k}]$ for some $k \in [m]$. Define index sets $\{J_k\}_{k=1}^m$ by the subsets of $[p]$ satisfying $J_k := \{i \in [p] : g_i \in \mathbb{R}[\mathbf{x}_{I_k}]\} \subseteq [p]$ and $[p] = \bigsqcup_{k=1}^m J_k$, where $\bigsqcup$ denotes the disjoint union;

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

(d) Add redundant constraints $M_k - \|\mathbf{x}_{I_k}\|_2^2 \geq 0$ where M_k are constants determined beforehand.

By analogy with the dense case, let $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and $I \subseteq [n]$. We denote by $\mathbf{x}_I = [x_i]_{i \in I}$, $\mathbf{v}_d(\mathbf{x}_I)$ the real-valued monomial basis for variable $\mathbf{x}_I$ of degree d, and $\mathbb{R}[\mathbf{x}_I]$ the vector space of polynomials in variable $\mathbf{x}_I$.

Definition 3.6. The d-th order *sparse moment matrix* with respect to subset I, denoted by $\mathbf{M}_d(\mathbf{y}, I)$, is defined as $\mathbf{M}_d(\mathbf{y}, I_k) = L_{\mathbf{y}}(\mathbf{v}_d(\mathbf{x}_I)\mathbf{v}_d(\mathbf{x}_I)^T)$. For a polynomial $f \in \mathbb{R}[\mathbf{x}_I]$, the d-th order *sparse localizing matrix* of f, denoted by $\mathbf{M}_d(f\mathbf{y}, I)$, is defined as $\mathbf{M}_d(f\mathbf{y}, I) = L_{\mathbf{y}}(f\mathbf{v}_d(\mathbf{x}_I)\mathbf{v}_d(\mathbf{x}_I)^T) = L_{\mathbf{y}}(f\mathbf{M}_d(\mathbf{y}, I))$.

By exploiting sparsity patterns, (POP) can be rewritten as:

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}_{I_k}) \geq 0,\ i \in J_k,\ k \in [m], \end{aligned} \tag{SparsePOP}$$

and its associated *sparse Lasserre's hierarchy* reads:

$$\rho_d = \min_{\mathbf{y}} \quad L_{\mathbf{y}}(f) \tag{SparseMom-d}$$
$$\text{s.t.} \quad \begin{cases} L_{\mathbf{y}}(1) = 1,\ \mathbf{M}_d(\mathbf{y}, I_k) \succeq 0, \\ \mathbf{M}_{d-\omega_i}(g_i\mathbf{y}, I_k) \succeq 0,\ i \in J_k,\ k \in [m], \end{cases}$$

where d, ω_i, $\mathbf{y}$, $L_{\mathbf{y}}$ are defined as in (DenseMom-d). The matrix $\mathbf{M}_d(\mathbf{y}, I_k)$ (resp. $\mathbf{M}_{d-\omega_i}(g_i\,\mathbf{y}, I_k)$) is a submatrix of the moment matrix $\mathbf{M}_d(\mathbf{y})$ (resp. localizing matrix $\mathbf{M}_{d-\omega_i}(g_i\mathbf{y})$) with respect to subset I_k, and hence of much smaller size $\binom{\tau_k+d}{\tau_k} = O(\tau_k^d)$ if $\tau_k := |I_k| \ll n$. The dual of the moment problem (SparseMom-d) can be formulated as

$$\begin{aligned} \xi_d = \max_{\lambda} \quad & \lambda \\ \text{s.t.} \quad & f - \lambda = \sum_{k=1}^{m} \Big(\sigma_{0,k} + \sum_{i \in J_k} \sigma_{i,k} g_i \Big), \end{aligned} \tag{SparseSOS-d}$$

where $\sigma_{0,k} \in \Sigma_d[\mathbf{x}_{I_k}], \sigma_{i,k} \in \Sigma_{d-\omega_i}[\mathbf{x}_{I_k}], \omega_i = \lceil \deg(g_i)/2 \rceil$ for $i \in J_k, k \in [m]$.

By analogy, we have similar convergent guarantee for sparse moment relaxations:

Theorem 3.3. *[Lasserre 2006] Let Assumption 3 holds. As $d \to \infty$, we have $\rho_d \uparrow f^*$. If $\mathbf{K}$ has a nonempty interior, then we have strong duality $\rho_d = \xi_d$.*

Example. Let $\mathbf{x} = [x_1, x_2]^T$, consider the following POP:

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^2} \quad & f(\mathbf{x}) = x_1 x_2 + x_2 x_3 \\ \text{s.t.} \quad & \begin{cases} g_1(\mathbf{x}) = 1 - x_1^2 - x_2^2 \geq 0, \\ g_2(\mathbf{x}) = 1 - x_2^2 - x_3^2 \geq 0. \end{cases} \end{aligned} \tag{3.10}$$

Define the subsets $I_1 = \{1, 2\}$, $I_2 = \{2, 3\}$. It is easy to check that Assumption 3 hold. For $d = 1$, we have $\mathbf{y} = [y_{000}, y_{100}, y_{010}, y_{001}, y_{200}, y_{110}, y_{101}, y_{020}, y_{011}, y_{002}]^T$ and $L_{\mathbf{y}}(f) = y_{110} + y_{011}, \omega_1 = \lceil \deg(g_1)/2 \rceil = 1, \omega_2 = \lceil \deg(g_2)/2 \rceil = 1$. Then the first-order dense and sparse moment matrices read:

$$\mathbf{M}_1(\mathbf{y}) = \begin{pmatrix} y_{000} & y_{100} & y_{010} & y_{001} \\ y_{100} & y_{200} & y_{110} & y_{101} \\ y_{010} & y_{110} & y_{020} & y_{011} \\ y_{001} & y_{101} & y_{011} & y_{002} \end{pmatrix},$$

$$\mathbf{M}_1(\mathbf{y}, I_1) = \begin{pmatrix} y_{000} & y_{100} & y_{010} \\ y_{100} & y_{200} & y_{110} \\ y_{010} & y_{110} & y_{020} \end{pmatrix}, \ \mathbf{M}_1(\mathbf{y}, I_2) = \begin{pmatrix} y_{000} & y_{010} & y_{001} \\ y_{010} & y_{020} & y_{011} \\ y_{001} & y_{011} & y_{002} \end{pmatrix}.$$

As we see, the sparse moment matrices $\mathbf{M}_1(\mathbf{y}, I_1)$ and $\mathbf{M}_1(\mathbf{y}, I_2)$ are submatrices of the dense moment matrix $\mathbf{M}_1(\mathbf{y})$, obtained by restricting to rows and columns concerned with subsets I_1 and I_2 respectively. The localizing matrices read:

$$\begin{aligned} \mathbf{M}_0(g_1\mathbf{y}) &= y_{000} - y_{200} - y_{020}, \\ \mathbf{M}_0(g_2\mathbf{y}) &= y_{000} - y_{020} - y_{002}. \end{aligned}$$

The first order ($d = 1$) sparse moment relaxation of problem (3.10) reads:

$$\begin{aligned} \rho^* = \min_{\mathbf{y}\in\mathbb{R}^6} \quad & y_{110} + y_{011} \\ \text{s.t.} \quad & \begin{cases} y_{00} = 1, \\ \mathbf{M}_1(\mathbf{y}, I_1) \succeq 0,\ y_{000} - y_{200} - y_{020} \geq 0, \\ \mathbf{M}_1(\mathbf{y}, I_2) \succeq 0,\ y_{000} - y_{020} - y_{002} \geq 0. \end{cases} \end{aligned} \tag{3.11}$$

If the maximum size τ of the subsets I_k is such that $\tau \ll n$, then solving sparse problems (SparseMom-d) rather than dense one (DenseMom-d) results in drastic computational savings. In fact, even with not so large n, the second relaxation with $d = 2$ is out of reach for currently available SDP solvers. Similarly to the dense case, $\rho_d \leq f^*$ for all d and moreover, if the subsets I_k satisfy RIP, then we still obtain the convergence $\rho_d \uparrow f^*$ as $d \to \infty$.

Chapter 4

Sublevel Hierarchy

Contents

Recall that the standard Lasserre's relaxation is a typical approach based on SDP to approximate the optimal value of polynomial optimization problem (POP), by solving a sequence of SDPs that provide a series of lower bounds and converges to the optimal value of the original problem. Other related frameworks of relaxations, including DSOS [Majumdar *et al.* 2014] based on LP, SDSOS [Majumdar *et al.* 2014] based on SOCP, and the hybrid BSOS [Lasserre *et al.* 2017] combining the features of LP and SDP hierarchies, also provide lower bounds converging to the optimal value of a POP. Generally speaking, when comparing LP and SDP solvers, the former can handle problems of much larger size. On the other hand, the bounds from LP relaxations are significantly weaker than those obtained by SDP relaxations, in particular for combinatorial problems [Laurent 2003]. Based on the standard Lasserre's hierarchy, several further works have explored various types of sparsity

patterns inside POPs to compute lower bounds more efficiently and handle larger-scale POPs. The first such extension can be traced back to [Waki *et al.* 2006] and [Lasserre 2006] where the authors consider the so-called *Correlative Sparsity Pattern (CSP)* with associated CSP graph whose nodes consist of the POP's variables. Two nodes in the CSP graph are connected via an edge if the two corresponding variables appear in the same constraint or in same monomial of the objective. The standard sparse Lasserre's hierarchy splits the full moment and localizing matrices into several smaller submatrices, according to subsets of nodes (maximal cliques) in a chordal extension of the CSP graph associated with the POP. When the size of the largest clique (a crucial parameter of the sparsity pattern) is reasonable the resulting SDP relaxations become tractable. There are many successful applications of the resulting *sparse moment-SOS hierarchy*, including certified roundoff error bounds [Magron *et al.* 2017, Magron 2018], optimal power flow [Josz & Molzahn 2018], volume computation of sparse semialgebraic sets [Tacchi *et al.* 2021], approximating regions of attractions of sparse polynomial systems [Schlosser & Korda 2020, Tacchi *et al.* 2020], noncommutative POPs [Klep *et al.* 2021], sparse positive definite functions [Mai *et al.* 2020]. Similarly, the sparse BSOS hierarchy [Weisser *et al.* 2018] is a sparse version of BSOS for large scale polynomial optimization.

Besides correlative sparsity, [Wang *et al.* 2021c, Wang *et al.* 2021b] exploit the so-called *term sparsity* (TSSOS) or combine correlative sparsity and term sparsity (CS-TSSOS) [Wang *et al.* 2022] to handle large scale polynomial optimization problems. The TSSOS framework relies on a *Term Sparsity Pattern (TSP) graph* whose nodes consist of monomials of some monomial basis. Two nodes in a TSP graph are connected via an edge when the product of the corresponding monomials appears in the supports of polynomials involved in the POP or is a monomial of even degree. Extensions have been provided to compute more efficiently approximations of joint spectral radii [Wang *et al.* 2021a] and minimal traces or eigenvalue of noncommutative polynomials [Wang & Magron 2021]. More variants of the sparse moment-SOS hierarchy have been built for quantum bounds on Bell inequalities [Pál & Vértesi 2009], condensed-matter ground-state problems [Barthel & Hübener 2012], quantum many-body problems [Haim *et al.* 2020], where one selects a certain subset of words (noncommutative monomials) to decrease the number of SDP variables.

Recently, [Campos *et al.* 2022] proposed a *partial* and *augmented partial relaxation* tailored to the Max-Cut problem. It strengthens Shor relaxation by adding some (and not all) constraints from the second-order Lasserre's hierarchy. The same idea was already used in the *multi-order* SDP relaxation of [Josz & Molzahn 2018] for solving large-scale Optimal Power Flow (OPF) problems. The authors set a threshold for the maximal cliques and include the second-order relaxation constraints for the cliques with size under the threshold and the first-order relaxation constraints for the cliques with size over the threshold.

4.1 Framework of Sublevel Hierarchy

Following the previous works concerned with extensions and/or variants of the moment-SOS hierarchy so as to handle large-scale POPs out of reach by the standard hierarchy, we provide a principled way to obtain intermediate alternative SDP relaxations between the d-th and $(d+1)$-th order SDP relaxations of the moment-SOS hierarchy for general POPs. Compared with existing sparse variants of the latter, we propose several possible SDP relaxations to improve lower bounds for general POPs.

Recall that for (POP), the d-th order dense Lasserre's relaxation relates to the Putinar's certificate $f - t = \sigma_0 + \sum_{i=1}^{p} \sigma_i g_i$ where σ_0 is an SOS in $\mathbb{R}[\mathbf{x}]$ of degree at most $2d$ and σ_i are SOS in $\mathbb{R}[\mathbf{x}]$ of degree at most $2(d - \omega_i)$ with $\omega_i = \lceil \deg(g_i)/2 \rceil$. In this section, we are going to choose some subsets of the variable $\mathbf{x}$ to decrease the number of terms involved in the SOS multipliers σ_0 and σ_i, and define the intermediate sublevel hierarchies between the d-th and $(d+1)$-th order relaxations.

Note that for dense Lasserre's relaxations, one approximates the cone of positive polynomials $\mathcal{P}[\mathbf{x}]$ from the inside with the following hierarchy of SOS cones:

$$\mathbb{R}_+ = \Sigma_0[\mathbf{x}] \subseteq \Sigma_1[\mathbf{x}] \subseteq \ldots \subseteq \Sigma[\mathbf{x}] \subseteq \mathcal{P}[\mathbf{x}],$$

with $\bigcup_{d=0}^{+\infty} \Sigma_d[\mathbf{x}] = \Sigma[\mathbf{x}]$. Similarly, in the sparse relaxations, one relies on the following hierarchy of SOS cones: for each subset I,

$$\mathbb{R}_+ = \Sigma_0[\mathbf{x}_I] \subseteq \Sigma_1[\mathbf{x}_I] \subseteq \ldots \subseteq \Sigma[\mathbf{x}_I] \subseteq \mathcal{P}[\mathbf{x}_I],$$

with $\bigcup_{d=0}^{+\infty} \Sigma_d[\mathbf{x}_I] = \Sigma[\mathbf{x}_I]$ and I is a subset of the index set $[n]$.

4.1.1 Sublevel hierarchy of SOS cones

The sublevel hierarchy provides refined SOS cones between $\Sigma_d[\mathbf{x}]$ and $\Sigma_{d+1}[\mathbf{x}]$, we first give the formal definition of the sublevel cones for both dense and sparse case.

Definition 4.1. *(Sublevel cones, dense case)* Let n be the number of variables in (POP). For $d \geq 1$ and $0 \leq l \leq n$, the l-th level SOS cone associated to $\Sigma_d[\mathbf{x}]$, denoted by $\Sigma_d^l[\mathbf{x}]$, is an SOS cone lying between $\Sigma_d[\mathbf{x}]$ and $\Sigma_{d+1}[\mathbf{x}]$, and is defined as

$$\Sigma_d[\mathbf{x}] \subseteq \Sigma_d^l[\mathbf{x}] := \Sigma_d[\mathbf{x}] + \tilde{\Sigma}_{d+1}^l[\mathbf{x}] \subseteq \Sigma_{d+1}[\mathbf{x}],$$

where $\tilde{\Sigma}_{d+1}^l[\mathbf{x}] := \left\{ \sum_{|I|=l} \sigma_I(\mathbf{x}_I) : I \subseteq \{1, \ldots, n\}, \sigma_I(\mathbf{x}_I) \in \Sigma_{d+1}[\mathbf{x}_I] \right\} \subseteq \Sigma_{d+1}[\mathbf{x}]$.

The SOS polynomials in $\tilde{\Sigma}_{d+1}^l[\mathbf{x}]$ are the elements in $\Sigma_{d+1}[\mathbf{x}]$ which can be decomposed into several components where each component is an SOS polynomial in l variables. Let us use the convention $\Sigma_d^0[\mathbf{x}] := \Sigma_d[\mathbf{x}]$. Then, for the dense case, we rely on the sublevel hierarchy of inner approximations of the cone of positive polynomials:

$$\Sigma_d[\mathbf{x}] = \Sigma_d^0[\mathbf{x}] \subseteq \Sigma_d^1[\mathbf{x}] \subseteq \ldots \subseteq \Sigma_d^n[\mathbf{x}] = \Sigma_{d+1}[\mathbf{x}].$$

Similarly, we can define the sparse sublevel cones:

Definition 4.2. *(Sublevel cones, sparse case)* Suppose that $\{I_k\}_{k=1}^m$ are the cliques of the sparse problem (SparsePOP). For $k \in [m]$ and $l \leq \tau_k := |I_k|$, we define the l-th level SOS cone of $\Sigma_d[\mathbf{x}_{I_k}]$, denoted by $\Sigma_d^l[\mathbf{x}_{I_k}]$, as

$$\Sigma_d[\mathbf{x}_{I_k}] \subseteq \Sigma_d^l[\mathbf{x}_{I_k}] := \Sigma_d[\mathbf{x}_{I_k}] + \tilde{\Sigma}_{d+1}^l[\mathbf{x}_{I_k}] \subseteq \Sigma_{d+1}[\mathbf{x}_{I_k}],$$

where $\tilde{\Sigma}_{d+1}^l[\mathbf{x}_{I_k}] := \left\{ \sum_{|I|=l} \sigma_I(\mathbf{x}_I) : I \subseteq I_k, \sigma_I(\mathbf{x}_I) \in \Sigma_{d+1}[\mathbf{x}_I] \right\} \subseteq \Sigma_{d+1}[\mathbf{x}_{I_k}].$

The SOS polynomials in $\tilde{\Sigma}[\mathbf{x}_{I_k}]_{d+1}^l$ are the elements in $\Sigma[\mathbf{x}_{I_k}]_{d+1}$ which can be decomposed into several components where each component is an SOS polynomial in l variables indexed by I_k. Then, for the sparse case, we rely on the sublevel hierarchy of inner approximations of the cone of positive polynomials: for $k \in [m]$,

$$\Sigma_d[\mathbf{x}_{I_k}] = \Sigma_d^0[\mathbf{x}_{I_k}] \subseteq \Sigma_d^1[\mathbf{x}_{I_k}] \subseteq \ldots \subseteq \Sigma_d^{\tau_k}[\mathbf{x}_{I_k}] = \Sigma_{d+1}[\mathbf{x}_{I_k}].$$

Remark. Lasserre's hierarchy relies on a hierarchy of SOS cones, while the sublevel hierarchy relies on a hierarchy of sublevel cones. Take the sparse case for illustration, solving the $(d+1)$-th order relaxation of the standard sparse Lasserre's hierarchy boils down to finding SOS multipliers in the cones $\Sigma[\mathbf{x}_{I_k}]_{d+1}$ and $\Sigma[\mathbf{x}_{I_k}]_{d-\omega_i+1}$ for each clique I_k. Solving the $(d+1)$-th order sublevel hierarchy boils down to finding SOS multipliers in the intermediate cones $\Sigma[\mathbf{x}_{I_k}]_d^{l_k}$ and $\Sigma[\mathbf{x}_{I_k}]_{d-\omega_i}^{l_k}$ for some $0 \leq l_k \leq \tau_k$. These cones approximate the standard cones $\Sigma[\mathbf{x}_{I_k}]_{d+1}$, $\Sigma[\mathbf{x}_{I_k}]_{d-\omega_i+1}$ as l_k gets larger since $\Sigma[\mathbf{x}_{I_k}]_d^{\tau_k} = \Sigma[\mathbf{x}_{I_k}]_{d+1}$ and $\Sigma[\mathbf{x}_{I_k}]_{d-\omega_i}^{\tau_k} = \Sigma[\mathbf{x}_{I_k}]_{d-\omega_i+1}$. We will see in the next definition that this is the so-called sublevel relaxation, and we call the vector $\mathbf{l} = [l_k]$ the vector of levels of the relaxation. Each l_k determines the size of the subsets in the clique I_k and is called a *level*.

4.1.2 Sublevel hierarchy of moment-SOS relaxations

Based on the sublevel cones, we are able to derive the corresponding moment-SOS relaxations for both dense and sparse cases.

Definition 4.3. *(Sublevel relaxations, dense case)* Let n be the number of variables in (POP). For each constraint $g_i \geq 0$ in (POP), we define a *level* $0 \leq l_i \leq n$ and a *depth* $0 \leq q_i \leq n$. Denote by $\mathbf{l} = [l_i]_{i=1}^p$ the vector of levels and $\mathbf{q} = [q_i]_{i=1}^p$ the vector of depths. Then, the $(\mathbf{l}, \mathbf{q})$-*sublevel relaxation* of the d-th order dense SOS problem (DenseMom-d) reads

$$\sup_{t \in \mathbb{R}} \left\{ t : f - t = \theta_0 + \sum_{i=1}^p (\tilde{\theta}_i + (\sigma_i + \tilde{\sigma}_i) g_i) \right\}, \qquad \text{(SubSOS-}[d, \mathbf{l}, \mathbf{q}]\text{)}$$

where θ_0 (resp. σ_i) are SOS polynomials in $\Sigma_d[\mathbf{x}]$ (resp. $\Sigma_{d-\omega_i}[\mathbf{x}]$), and $\tilde{\theta}_i$ (resp. $\tilde{\sigma}_i$) are SOS polynomials in $\tilde{\Sigma}_{d+1}^{l_i}[\mathbf{x}]$ (resp. $\tilde{\Sigma}_{d-\omega_i+1}^{l_i}[\mathbf{x}]$) with $\omega_i = \lceil \deg(g_i)/2 \rceil$.

Moreover, each $\tilde{\sigma}_i$ is a sum of q_i SOS polynomials where each sum term involves variables in a certain subset $\Gamma_{i,j} \subseteq \{1,2,\ldots,n\}$ with $|\Gamma_{i,j}| = l_i$, i.e., $\tilde{\sigma}_i = \sum_{j=1}^{q_i} \tilde{\sigma}_{i,j}$ where $\tilde{\sigma}_{i,j} \in \Sigma_{d-\omega_i+1}[\mathbf{x}_{\Gamma_{i,j}}]$. Each $\tilde{\theta}_i$ is also a sum of q_i SOS polynomials where the sum terms share the same variable sets $\Gamma_{i,j}$ as $\tilde{\sigma}_{i,j}$, i.e., $\tilde{\theta}_i = \sum_{j=1}^{q_i} \tilde{\theta}_{i,j}$ where $\tilde{\theta}_{i,j} \in \Sigma_{d+1}[\mathbf{x}_{\Gamma_{i,j}}]$. The equation (SubSOS-$[d,\mathbf{l},\mathbf{q}]$) can be compressed as an analogical form of the standard dense Lasserre's relaxation:

$$\sup_{t\in\mathbb{R}}\left\{t : f - t = \sum_{i=1}^{p}(\tilde{\theta}_i + \tilde{\sigma}_i g_i)\right\},$$

where $\tilde{\theta}_i$ (resp. $\tilde{\sigma}_i$) are SOS polynomials in $\Sigma^l_{d+1}[\mathbf{x}]$ (resp. $\Sigma^l_{d-\omega_i+1}[\mathbf{x}]$).

Definition 4.4. *(Sublevel relaxations, sparse case)* Suppose that $\{I_k\}_{k=1}^m$ are the cliques of the sparse problem (SparsePOP) with $\tau_k = |I_k|$. For each $k \in [m]$, define $J_k := \{i \in [p] : g_i \in \mathbb{R}[\mathbf{x}_{I_k}]\} \subseteq [p]$ such that $[p] = \bigsqcup_{k=1}^m J_k$. For $i \in J_k$ and $k \in [m]$, define a level $0 \le l_{i,k} \le \tau_k$ and a depth $0 \le q_{i,k} \le \tau_k$. Denote by $\mathbf{l} = [l_{i,k}]_{i\in J_k, k\in[m]}$ the vector of sublevels and $\mathbf{q} = [q_{i,k}]_{i\in J_k,k\in[m]}$ the vector of depths. Then, the $(\mathbf{l},\mathbf{q})$-*sublevel relaxation* of the d-th order sparse SOS problem (SparseMom-d) reads

$$\sup_{t\in\mathbb{R}}\left\{t : f - t = \sum_{k=1}^{m}\left(\theta_{0,k} + \sum_{i\in J_k}(\tilde{\theta}_{i,k} + (\sigma_{i,k} + \tilde{\sigma}_{i,k})g_i)\right)\right\}, \quad \text{(SubSpSOS-}[d,\mathbf{l},\mathbf{q}]\text{)}$$

where $\theta_{0,k}$ (resp. $\sigma_{i,k}$) are SOS polynomials in $\Sigma_d[\mathbf{x}_{I_k}]$ (resp. $\Sigma_{d-\omega_i}[\mathbf{x}_{I_k}]$), and $\tilde{\theta}_{0,k}$ (resp. $\tilde{\sigma}_{i,k}$) are SOS polynomials in $\tilde{\Sigma}^{l_{i,k}}_{d+1}[\mathbf{x}_{I_k}]$ (resp. $\tilde{\Sigma}^{l_{i,k}}_{d-\omega_i+1}[\mathbf{x}_{I_k}]$) with $\omega_i = \lceil \deg(g_i)/2 \rceil$. Moreover, each $\tilde{\sigma}_{i,k}$ with $i \in I_k$ is a sum of $q_{i,k}$ SOS polynomials where each sum term involves variables in a certain subset $\Gamma_{i,k,j} \subseteq I_k$ with $|\Gamma_{i,k,j}| = l_{i,k}$, i.e., $\tilde{\sigma}_{i,k} = \sum_{j=1}^{q_{i,k}} \tilde{\sigma}_{i,k,j}$ where $\tilde{\sigma}_{i,k,j} \in \Sigma_{d-\omega_i+1}[\mathbf{x}_{\Gamma_{i,k,j}}]$. Each $\tilde{\theta}_{i,k}$ is also a sum of $q_{i,k}$ SOS polynomials where the sum terms share the same variable sets $\Gamma_{i,k,j}$ as $\tilde{\sigma}_{i,k,j}$, i.e., $\tilde{\theta}_{i,k} = \sum_{j=1}^{q_{i,k}} \tilde{\theta}_{i,k,j}$ where $\tilde{\theta}_{i,k,j} \in \Sigma_{d+1}[\mathbf{x}_{\Gamma_{i,k,j}}]$. The equation (SubSpSOS-$[d,\mathbf{l},\mathbf{q}]$) can also be compressed as an analogical form of the standard sparse Lasserre's relaxation:

$$\sup_{t\in\mathbb{R}}\left\{t : f - t = \sum_{k=1}^{m}\sum_{i\in J_k}(\tilde{\theta}_{i,k} + \tilde{\sigma}_{i,k}g_i)\right\},$$

where $\tilde{\theta}_{i,k}$ (resp. $\tilde{\sigma}_{i,k}$) are SOS polynomials in $\Sigma^l_{d+1}[\mathbf{x}_{I_k}]$ (resp. $\Sigma^l_{d-\omega_i+1}[\mathbf{x}_{I_k}]$).

Remark. (i). If one of the sublevel l_i (resp. $l_{i,k}$) in the dense (resp. sparse) sublevel relaxation is such that $l_i = n$ (resp. $l_{i,k} = \tau_k$), then the depth q_i (resp. $q_{i,k}$) should automatically be 1.

(ii). The heuristics to determine the subsets ($\Gamma_{i,j}$ for the dense case and $\Gamma_{i,k,j}$ for the sparse case) in the sublevel relaxation will be discussed in the next section.

(iii). The size of the SDP Gram matrix associated to an SOS polynomial in $\Sigma^l_d[\mathbf{x}]$ (resp. $\Sigma^l_d[\mathbf{x}_{I_k}]$) is $\max\{\binom{n+d}{d}, \binom{l+d+1}{d+1}\}$ (resp. $\max\{\binom{|I_k|+d}{d}, \binom{l+d+1}{d+1}\}$). If the lower bound obtained by solving the SOS problem over $\Sigma_{d+1}[\mathbf{x}]$ (resp. $\Sigma_{d+1}[\mathbf{x}_{I_k}]$)

is not satisfactory enough, then we may try to find more accurate solutions in one of the cones of $\Sigma_d^l[\mathbf{x}]$ (resp. $\Sigma_d^l[\mathbf{x}_{I_k}]$).

(iv). In case of ambiguity, we call the sublevel relaxation of a d-th order SOS problem the $(d+1)$-th order sublevel relaxation, i.e., the $(d+1)$-th order sublevel relaxation is an intermediate between the d-th order and $(d+1)$-th order Lasserre's relaxation.

Example. Take the polynomials f, g_k and the cliques I_k as in Example 3.10. Define $\mathbf{l} = [2,2]$ and $\mathbf{q} = [1,1]$. We select subsets w.r.t. g_1 and g_2 respectively as $\Gamma_{1,1} = \{1,2\}$, $\Gamma_{2,1} = \{5,6\}$. Then, the second-order dense $(\mathbf{l},\mathbf{q})$-sublevel relaxation reads

$$\begin{aligned} \max_{t\in\mathbb{R}} \quad & t \\ \text{s.t.} \quad & f(\mathbf{x}) - t = \theta_0(\mathbf{x}) + \tilde{\theta}_1(\mathbf{x}_{\Gamma_{1,1}}) + \tilde{\sigma}_1(\mathbf{x}_{\Gamma_{1,1}})g_1(\mathbf{x}) \\ & \qquad + \tilde{\theta}_2(\mathbf{x}_{\Gamma_{2,1}}) + \tilde{\sigma}_2(\mathbf{x}_{\Gamma_{2,1}})g_2(\mathbf{x}), \end{aligned} \tag{4.1}$$

where θ_0 is a degree-2 SOS polynomial in variable $\mathbf{x}$, $\tilde{\theta}_k$ are degree-4 SOS polynomials in variable $\mathbf{x}_{\Gamma_{k,1}}$, $\tilde{\sigma}_k$ are degree-2 SOS polynomials in variable $\mathbf{x}_{\Gamma_{k,1}}$. In other words, $\theta_0 \in \Sigma_1[\mathbf{x}], \tilde{\theta}_k \in \Sigma_2[\mathbf{x}_{\Gamma_{k,1}}] \subseteq \tilde{\Sigma}_2^2[\mathbf{x}], \tilde{\sigma}_k \in \Sigma_1[\mathbf{x}_{\Gamma_{k,1}}] \subseteq \tilde{\Sigma}_1^2[\mathbf{x}]$.

Similarly, define $\Gamma_{1,1,1} = \{1,2\} \subseteq I_1$ and $\Gamma_{2,2,1} = \{5,6\} \subseteq I_2$, then the second-order sparse $(\mathbf{l},\mathbf{q})$-sublevel relaxation reads

$$\begin{aligned} \max_{t\in\mathbb{R}} \quad & t \\ \text{s.t.} \quad & f(\mathbf{x}) - t = \theta_{0,1}(\mathbf{x}_{I_1}) + \tilde{\theta}_1(\mathbf{x}_{\Gamma_{1,1,1}}) + \tilde{\sigma}_1(\mathbf{x}_{\Gamma_{1,1,1}})g_1(\mathbf{x}) \\ & \qquad + \theta_{0,2}(\mathbf{x}_{I_2}) + \tilde{\theta}_2(\mathbf{x}_{\Gamma_{2,2,1}}) + \tilde{\sigma}_2(\mathbf{x}_{\Gamma_{2,2,1}})g_2(\mathbf{x}), \end{aligned} \tag{4.2}$$

where $\theta_{0,k}$ are degree-2 SOS polynomials in variable $\mathbf{x}_{I_k}$, $\tilde{\theta}_k$ are degree-4 SOS polynomials in variable $\mathbf{x}_{\Gamma_{k,k,1}}$, $\tilde{\sigma}_k$ are degree-2 SOS polynomials in variable $\mathbf{x}_{\Gamma_{k,k,1}}$. In other words, $\theta_{0,k} \in \Sigma_1[\mathbf{x}_{I_k}], \tilde{\theta}_k \in \Sigma_2[\mathbf{x}_{\Gamma_{k,k,1}}] \subseteq \tilde{\Sigma}_2^2[\mathbf{x}_{I_k}], \tilde{\sigma}_k \in \Sigma_1[\mathbf{x}_{\Gamma_{k,k,1}}] \subseteq \tilde{\Sigma}_1^2[\mathbf{x}_{I_k}]$.

Remark. In Definition 4.3 and 4.4, each sublevel relaxation is formulated as a (primal) SOS problem. Similar with the standard Lasserre's relaxation [Lasserre 2006], we can also write the (dual) moment problem, in which the *moment matrices* correspond to SOS polynomial $\theta_0, \tilde{\theta}_i$ for the dense case or $\theta_{0,k}, \tilde{\theta}_{i,k}$ for the sparse case, and the *localizing matrices* correspond to SOS polynomial $\sigma_i, \tilde{\sigma}_i$ for the dense case or $\sigma_{i,k}, \tilde{\sigma}_{i,k}$ for the sparse case. Since the $(d+1)$-th order sublevel relaxation is built based on the d-th order Lasserre's relaxation, i.e., the moment matrices corresponding to θ_0 or $\theta_{0,k}$ are the same as in the standard Lasserre's relaxation. Then if each moment matrix is of rank 1, one is able to verify that the sublevel relaxation reaches the exact optimal value of the original problem and we can follow the extraction procedures of [Henrion & Lasserre 2005] and [Lasserre 2006, Section 3.3] to obtain exact optimal solutions from the moment matrices for the dense and sparse cases, respectively.

In the next example, we show explicitly how the sublevel relaxation performs between order-1 and order-2 Lasserre's relaxation when the order-2 relaxation is

exact.

Example. This example is taken from [Floudas & Pardalos 1990, Problem 2.9], which is a maximum clique problem (see (Max-Cliq)):

$$\begin{aligned} \max_{\mathbf{x}} \quad & f(\mathbf{x}) = \sum_{i=1}^{9} x_i x_{i+1} + \sum_{i=1}^{8} x_i x_{i+2} + x_1 x_9 + x_1 x_{10} + x_2 x_{10} + x_1 x_5 + x_4 x_7 \quad (4.3) \\ \text{s.t.} \quad & \sum_{i=1}^{10} x_i = 1,\ 0 \le x_i \le 1,\ i \in [10]. \end{aligned}$$

The global solution is $f(\mathbf{x}^*) = 0.375$ with maximizer $\mathbf{x}^* = [0, 0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, 0, 0]^T$ which is certified by GloptiPoly [Henrion *et al.* 2009] for second-order dense Lasserre's relaxation.

We now consider the dense second-order sublevel relaxation compared with Shor's relaxation, where we use the *ordered heuristic* **H2** described in Section 4.2 to select the q subsets of size l. Notice that if $l = 0$ or $q = 0$, the sublevel relaxation reduces to Shor's relaxation, and if $l = 10$, the sublevel relaxation is exactly the dense Lasserre's relaxation. Table 4.1 shows how the upper bounds computed by the sublevel relaxations successively converge to the optimal solution of problem (4.3), when we increase the level l and depth q. The upper bounds in bold font indicate that we have reached the exact optimal value and we can extract the (unique) global solution via the dense order-1 moment matrix $\mathbf{M}_1(\mathbf{y})$.

Table 4.1: Summary of the performance of sublevel relaxations applied to problem (4.3) with $l, q = 0, 1, \ldots, 10$.

l \ q	0	1	2	3	4	5	6	7	8	9	10
0	2.226	-	-	-	-	-	-	-	-	-	-
1	-	1.982	1.892	1.813	1.578	1.455	1.280	1.167	1.167	1.167	1.167
2	-	1.772	1.729	1.500	1.400	1.263	1.166	1.166	1.166	1.166	1.166
3	-	1.714	1.500	1.400	1.250	1.039	0.746	0.657	0.616	0.587	0.456
4	-	1.500	1.400	1.250	1.000	0.637	0.483	0.446	0.428	0.406	0.392
5	-	1.400	1.250	1.000	0.577	0.438	0.418	0.411	0.389	0.377	0.377
6	-	1.250	1.000	0.565	0.376	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**
7	-	1.000	0.563	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**
8	-	0.563	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**
9	-	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**	**0.375**
10	-	**0.375**	-	-	-	-	-	-	-	-	-

4.1.3 Relation with other relaxations

The standard Lasserre's hierarchy and many of its variants are contained in the framework of sublevel hierarchy:

Example. (Dense Lasserre's Relaxation [Lasserre 2001]) For (POP), the dense version of the d-th order Lasserre's relaxation is the dense d-th order sublevel relaxation with $\mathbf{l} = [n, n, \ldots, n]$ and $\mathbf{q} = \mathbf{1}_p$, where $\mathbf{1}_p$ denotes the p-dimensional vector with all ones.

Example. (Sparse Lasserre's Relaxation [Lasserre 2006]) For (SparsePOP), the sparse version of the $(d-1)$-th order Lasserre's relaxation is the sparse d-th order sublevel relaxation with $\mathbf{l} = [[\tau_s]_{s\in J_1}; \ldots; [\tau_s]_{s\in J_m}]$ and $\mathbf{q} = \mathbf{1}_{|J_1|+\ldots+|J_m|}$.

Example. (Multi-Order/Partial Relaxation) The multi-order relaxation (which is used to solve the *Optimal Power Flow* problem in [Josz & Molzahn 2018]), also named as partial relaxation (which is specifically encoded to solve the *Max-Cut* problem in [Campos *et al.* 2022]), is a variant of the second-order sparse Lasserre's relaxation. We first preset a value r, then compute the maximal cliques in the chordal extension of the CSP graph of the POP. For those cliques of size larger than r, we consider the first-order moment matrices; for those of size smaller or equal than r, we consider the second-order moment matrices. Denote by S the set of indices such that $\tau_k > r$ for $k \in S$, and T the set of indices such that $\tau_k \leq r$ for $k \in T$. Then the multi-order/partial relaxation is the second-order sublevel relaxation with

$$\mathbf{l} = [[0]_{s\in J_1\cap S}, [\tau_s]_{s\in J_1\cup T}; \ldots; [0]_{s\in J_m\cap S}, [\tau_s]_{s\in J_m\cup T}],$$

and

$$\mathbf{q} = [[0]_{s\in J_1\cap S}, [1]_{s\in J_1\cup T}; \ldots; [0]_{s\in J_m\cap S}, [1]_{s\in J_m\cup T}].$$

Example. (Augmented Partial Relaxation) This relaxation is the strengthened version of the partial relaxation in [Campos *et al.* 2022] to solve Max-Cut problems. It is exactly the second-order sublevel relaxation restricted to Max-Cut problem.

Example. (Heuristic Relaxation) The heuristic relaxation in [Chen *et al.* 2020] for computing the upper bound of the Lipschitz constant of ReLU networks, is a variant of the second-order dense Lasserre's relaxation. The intuition is that some constraints in the POP are sparse, so let us denote by S the set of their indices, while their corresponding cliques are large, thus one cannot solve the second-order relaxation of the standard sparse Lasserre's hierarchy. We then consider the dense first-order relaxation (Shor's relaxation), and choose subsets of moderate sizes (size 2 in [Chen *et al.* 2020]) that contain the variable sets of these sparse constraints. For other constraints with larger variable sets, let us denote by T the set of their indices and let us consider the first-order moment matrices. Then the heuristic relaxation is the second-order sublevel relaxation with $\mathbf{l} = [[0]_{i\in T}, [2]_{i\in S}]$ and $\mathbf{q} = [[0]_{i\in T}, [1]_{i\in S}]$.

Remark. [Nie & Demmel 2009] proposed a moment-SOS relaxation for unconstrained POPs whose objective function can be written as a sum of polynomials involving only small number of variables. Precisely, let the objective function be a polynomial of form $f(\mathbf{x}) = \sum_{i=1}^{r} f_i(\mathbf{x}_{\Delta_i})$, where $\mathbf{x} \in \mathbb{R}^n$ and $\Delta_i \subseteq \{1, \ldots, n\}$ for all i. Note that the subsets Δ_i do not necessarily satisfy the RIP condition. We define subsets $\{I_k\}_{k=1}^{m} \subseteq \{1, \ldots, n\}^m$ satisfying the RIP condition such that for each i, $\Delta_i \subseteq I_k$ for some k. Then the $(d+1)$-th order sublevel relaxation involves the sparse d-th order moment matrices $\mathbf{M}_d(\mathbf{y}, I_k)$ and sparse $(d+1)$-th order moment matrices $\mathbf{M}_{d+1}(\mathbf{y}, \Delta_i)$, which guarantees that the $(d+1)$-th order sublevel relaxation performs no worse than the d-th order Lasserre's relaxation. In contrast, the

$(d+1)$-th order relaxation in [Nie & Demmel 2009] only involves moment matrices $\mathbf{M}_{d+1}(\mathbf{y}, \Delta_i)$ and hence is likely to give looser bounds than the d-th order Lasserre's relaxation. On the other hand, in the case that the size of subsets I_k is much larger than Δ_i, the relaxation in [Nie & Demmel 2009] is faster to solve than the sublevel relaxation.

Summarizing the above discussion, we have the following proposition:

Proposition 4.1. *(1) For dense case, if* $\mathbf{l} = [n, n, \ldots, n]$*, then the d-th order sublevel relaxation is exactly the dense* $(d+1)$*-th order Lasserre's relaxation.*

(2) For sparse case, if $\mathbf{l} = [[\tau_s]_{s\in J_1}; \ldots; [\tau_s]_{s\in J_m}]$*, then the d-th order sublevel relaxation is exactly the sparse* $(d+1)$*-th order Lasserre's relaxation.*

4.2 Determination of the Subsets of Cliques

There are different ways to determine the subsets $\Gamma_{i,j}$ (or $\Gamma_{i,k,j}$) of the sublevel relaxation described in Definition 4.3 and 4.4. Generically, we are not aware of any algorithm that would guarantee that the selected subsets are optimal at a given level of relaxation. In this section, we propose several heuristics to select the subsets. Suppose that $\{I_k\}_{1\le k\le m}$ is the sequence of maximal cliques in the chordal extension of the CSP graph of the sparse problem (SparsePOP) and that the level of relaxation is $l \le |I_k| =: \tau_k$. We need to select the "best" candidate among the $\binom{\tau_k}{l}$ many subsets of size l. However, in practice, the number $\binom{\tau_k}{l}$ might be very large since $\binom{\tau_k}{l} \approx \tau_k^l$ when l is fixed.

In order to make this selection procedure tractable, we reduce the number of sample subsets to τ_k. Precisely, suppose $I_k := \{i_1, i_2, \ldots, i_{\tau_k}\}$, define $I_{k,j} := \{i_j, i_{j+1}, \ldots, i_{j+l}\}$ for $j = 1, 2, \ldots, \tau_k$ and $1 \le l \le \tau_k$. By convention, $i_j = i_k$ if $j \equiv k \mod \tau_k$. Denote by p the depth of the relaxation. Then we use the following heuristics to choose p subsets among the candidates $I_{k,j}$. Without loss of generality, we assume that $l < \tau_k$ (otherwise one has $l \ge \tau_k$, then we only need to select one subset $I = I_k$).

- **H1** *(Random Heuristic)*: For each i and clique I_k, we randomly select p subsets $\Gamma_{i,k,j} \subseteq I_k$ for $j = 1, \ldots, p$, such that $|\Gamma_{i,k,j}| = l$ for all j;
- **H2** *(Ordered Heuristic)*: For each i and clique I_k, we select one after another $\Gamma_{i,k,j} = I_{k,j} \subseteq I_k$ for $j = 1, \ldots, p$. For $p = \tau_k$, we also call this heuristic the *cyclic heuristic*.

The heurisics **H1** and **H2** do not depend on the problem, thus they might not fully explore the specific structure hidden in the POPs. We can also try the heuristic that selects the subsets according to the value of the moments in the first-order moment relaxation (Shor's relaxation).

- **H3** *(Moment Heuristic)*: First of all, we solve the first-order sparse relaxation. For each i and clique I_k, suppose $\mathbf{M}_k$ is the first-order moment matrix

indexed by 1 and the monomials in $\mathbf{x}_{I_k}$. Denote by $\mathbf{M}_k(I_{k,j})$ the submatrix whose rows and columns are indexed by 1 and $\mathbf{x}_{I_{k,j}}$ for $j = 1, 2, \ldots, \tau_k$. We reorder the subsets $I_{k,j}$ w.r.t. the infinity norm (the largest row sum of absolute value) of the submatrices $\mathbf{M}_k(I_{k,j})$, i.e.,

$$\|\mathbf{M}_k(I_{k,1})\|_\infty \geq \|\mathbf{M}_k(I_{k,2})\|_\infty \geq \ldots \geq \|\mathbf{M}_k(I_{k,\tau_k})\|_\infty .$$

Then we pick the first p subsets $\Gamma_{i,k,1} = I_{k,1}, \Gamma_{i,k,2} = I_{k,2}, \ldots, \Gamma_{i,k,p} = I_{k,p}$ after reordering.

In particular, for Max-Cut problem, [Campos *et al.* 2022] proposed the following heuristics that take the weights in the graph or the maximal cliques in the chordal graph into account. We briefly introduce the idea of these heuristics, readers can refer to [Campos *et al.* 2022] for details. For heuristic **H4** to **H6**, denote by $\mathbf{L}$ the *Laplacian* matrix of the graph.

- **H4** *(Laplacian Heuristic)*: For each clique I_k, denote by $\mathbf{L}(I_{k,j})$ the submatrix of the moment matrix $\mathbf{M}_k$ whose rows and columns are indexed by 1 and $\mathbf{x}_{I_{k,j}}$ for $j = 1, 2, \ldots, \tau_k$. We reorder the subsets $\{I_{k,j}\}$ w.r.t. the infinity norm of the submatrices $\{\mathbf{L}(I_{k,j})\}$, i.e.,

  $$\|\mathbf{L}(I_{k,1})\|_\infty \geq \|\mathbf{L}(I_{k,2})\|_\infty \geq \ldots \geq \|\mathbf{L}(I_{k,\tau_k})\|_\infty.$$

  Then we pick the first p subsets $\Gamma_{i,k,1} = I_{k,1}, \Gamma_{i,k,2} = I_{k,2}, \ldots, \Gamma_{i,k,p} = I_{k,p}$ after reordering. The spirit of **H4** and **H5** is to select the subsets involving large value of moments or weights, and we assume a priori that larger value of moments/weights have larger impacts on the subsets;

- **H5** *(Max-Repeated Heuristic)*: We select subsets contained in many maximal cliques;

- **H6** *(Min-Repeated Heuristic)*: We select subsets contained in few maximal cliques;

- **H4-5**: We combine heuristic **H4** and **H5** to select the subsets that are not repeated in other maximal cliques and contain variables with large weights.

In the spirit of the heuristic **H4-5**, we can also combine **H5** with the moment heuristic **H3**:

- **H3-5**: We combine **H3** and **H5** to select the subsets that are not repeated in other maximal cliques and contain variables with large moments.

There is no general guarantee that one of the heuristics always performs better than the others. In Table 4.2, we show the upper bounds obtained by the above heuristics for two Max-Cut instances g_20 and w01_100 (the detail of the numerical settings and the results is referred to Section 4.3.1), for level 4, 6, and depth 1, 2,

Table 4.2: Comparison of different heuristics for Max-Cut instances g_20 and w01_100.

Heuristics	lv=4, p=1		lv=4, p=2		lv=6, p=1		lv=6, p=2		Count
	g20	w01	g20	w01	g20	w01	g20	w01	
H1	548.4	725.6	**539.0**	720.0	**526.6**	709.5	**522.0**	**700.4**	**4**
H2	**546.4**	728.0	539.9	721.1	526.9	**705.7**	523.1	701.7	**2**
H3	550.6	728.8	541.8	723.2	528.5	713.9	524.2	705.6	0
H4	549.7	**723.4**	542.0	**718.6**	526.9	710.5	523.6	701.5	**2**
H5	553.5	731.0	543.1	725.8	529.3	715.6	525.2	708.4	0
H6	553.3	731.2	543.2	726.6	529.3	717.2	525.2	710.3	0
H3-5	550.5	729.5	541.8	726.6	528.5	713.8	524.2	704.8	0
H4-5	549.8	726.6	542.0	719.3	526.9	710.4	523.6	**700.4**	1

respectively. For each heuristic, we count the number of times that the heuristic performs the best. We see that, suprisingly, the random heuristic **H1** performs the best among other heuristics. The ordered heuristic **H2** and Laplacian heuristic **H4** also performs well. For the sake of simplicity, we will only consider the ordered heuristic **H2** and its variants for the forthcoming examples.

4.3 Application to Optimization Problems

This section is about the numerical experiments of sublevel hierarchy in optimization. We apply the sublevel relaxation to different type of POPs in optimization, as discussed in the previous section. Most of the instances are taken from the Biq-Mac library [Rendl *et al.* 2007] and the QPLIB library [Furini *et al.* 2019]. We calculate the *ratio of improvements (RI)* of each sublevel relaxation, compared with Shor's relaxation. We also compute the *relative gap (RG)* between the sublevel relaxation and the optimal solution. The calculations of RI and RG are given by

$$\text{RI} = \frac{\text{Shor} - \text{sublevel}}{\text{Shor} - \text{solution}} \times 100\%, \quad \text{RG} = \frac{\text{sublevel} - \text{solution}}{|\text{solution}|} \times 100\%. \tag{4.4}$$

For each instance, we only show the ratio of improvements and relative gap corresponding to the results of the last sublevel relaxation, and use the bold font if we have improvements larger than 40%. The larger the ratio of improvements or the smaller the relative gap, the better the bounds. If the optimal solution is not known so far, it is replaced by the (best-known) valid upper bounds (UB) or lower bounds (LB). We implement all the programs on Julia, and use Mosek as backend to solve SDP relaxations. The running time (with second as unit) displayed in all tables refers to the time spent by Mosek[1] to solve the SDP relaxation. All experiments are performed with an Intel 8-Core i7-8665U CPU @ 1.90GHz Ubuntu 18.04.5 LTS, 32GB RAM.

[1] https://www.mosek.com/

4.3.1 Maximum cut (Max-Cut) problem

Given an undirected graph $G(V, E)$ where V is a set of vertices and E is a set of edges, a *cut* is a partition of the vertices into two disjoint subsets. The *Max-Cut* problem consists of finding a cut in a graph such that the number of edges between the two subsets is as large as possible. It can be formulated as follows:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \mathbf{x}^T \mathbf{L} \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in \{-1, 1\}^n, \end{aligned} \tag{Max-Cut}$$

where $\mathbf{L}$ is the Laplace matrix of the given graph of n vertices, i.e., $\mathbf{L} := \mathrm{Diag}(\mathbf{W} \cdot \mathbf{1}_n) - \mathbf{W}$ where $\mathbf{W}$ is the weight matrix of the graph. The constraints $\mathbf{x} \in \{-1, 1\}^n$ are equivalent to $(x_i)^2 = 1$ for all i. Suppose that $\{I_k\}_{k=1}^m$ are the maximum cliques in the chordal extension of the given graph. For $i = 1, 2, \ldots, n$, denote by $k(i)$ (one of) the index such that $i \in I_{k(i)}$. Suppose that $I_{k(i)} = \{i_1, \ldots, i_{\tau_{k(i)}}\}$ for some $i_j = i$ with $1 \le j \le \tau_{k(i)}$. Given a depth q and a level l, we select q subsets of size l by order as: for $t \in [q]$ and $i \in [n]$

$$I_{i,t} = \{i_j, i_{j+t}, \ldots, i_{j+t+l-2}\}, \tag{4.5}$$

with the convention that $i_{j_1} = i_{j_2}$ if $j_1 \equiv j_2 \mod \tau_{k(i)}$. If we consider the dense sublevel hierarchy, then we directly select the subsets by order as $I_{i,t} = \{i, i + t, \ldots, i + t + l - 2\}$ for $t \in [q]$.

The following classes of problems and their solutions are from the Biq-Mac library. For each class of problem, we choose the first instance, i.e., $i = 0$, and drop the suffix ".i" in Table 4.4:

- g05_n.i, unweighted graphs with edge probability 0.5, $n = 60, 80, 100$;
- pm1s_n.i, pm1d_n.i, weighted graph with edge weights chosen uniformly from $\{-1, 0, 1\}$ and density 10% and 99% respectively, $n = 80, 100$;
- wd_n.i, pw$d$_n.i, graph with integer edge weights chosen from $[-10, 10]$ and $[0, 10]$ respectively, density $d = 0.1, 0.5, 0.9$, $n = 100$.

The instances g_n and the corresponding upper bounds are from the CS-TSSOS paper [Wang *et al.* 2022]. The instances Gn are from the G-set library by Y.Y. Ye[2], and their best-known solutions are taken from [Kochenberger *et al.* 2013].

In Table 4.3, we give a summary of basic information and the graph structure of each instance: *nVar* denotes the number of variables, *Density* denotes the percentage of non-zero elements in the adjacency matrix, *nCliques* denotes the number of cliques in the chordal extension, *MaxClique* denotes the maximum size of the cliques, *MinClique* denotes the minimum size of the cliques.

In Table 4.4, we display the upper bounds and running times corresponding to the sublevel relaxations of depth 1, and level 0, 4, 6, 8, respectively. Notice that the

[2] http://web.stanford.edu/~yyye/yyye/Gset/

Table 4.3: Summary of the basic information and graph structure of Max-Cut instances.

	nVar	Density	nCliques	MaxClique	MinClique
g05_60	60	50%	11	50	19
g05_80	80	50%	12	69	28
g05_100	100	50%	13	88	37
pm1d_80	80	99%	2	79	76
pm1d_100	100	99%	2	99	95
pm1s_80	80	10%	44	37	4
pm1s_100	100	10%	47	54	4
pw01_100	100	10%	47	54	4
pw05_100	100	50%	12	89	40
pw09_100	100	90%	4	97	83
w01_100	100	10%	47	54	4
w05_100	100	50%	12	89	40
w09_100	100	90%	4	97	83
g_20	505	1.6%	369	15	1
g_40	1005	0.68%	756	15	1
g_60	1505	0.43%	756	15	1
g_80	2005	0.30%	1556	15	1
g_100	2505	0.23%	1930	16	1
g_120	3005	0.19%	2383	15	1
g_140	3505	0.16%	2762	15	1
g_160	4005	0.13%	3131	15	1
g_180	4505	0.12%	3429	15	1
g_200	5005	0.11%	3886	15	1
G11	800	0.25%	598	24	5
G12	800	0.25%	598	48	5
G13	800	0.25%	598	90	5
G32	2000	0.1%	1498	76	5
G33	2000	0.1%	1498	99	5
G34	2000	0.1%	1498	141	5

authors in [Campos *et al.* 2022] use the partial relaxation to compute upper bounds for instances g_20 to g_200. The sublevel relaxation we consider here is actually what they call the augmented partial relaxation, which is a strengthened relaxation based on partial relaxation. From the ratio of improvement, we see that the more sparse structure the graph has, the better the sublevel relaxation performs. Notice that if we obtain better upper/lower bounds than the current best-known bounds, the ratio of improvements will be larger than 100% and the relative gap will become nagative. Particularly, our method provides better bounds for all the instances g_n in the CS-TSSOS paper [Wang *et al.* 2022], and computes upper bounds very close to the best-known solution for the instances Gn in G-set.

In addition to Table 4.4, if the number of variables is of moderate size, the dense sublevel relaxation might performs faster than the sparse one. For example, the instance g05_100 has 13 maximal cliques with maximum size 88 and minimum size 37. The sparse sublevel relaxation consists of 13 first-order moment matrices of size from 37 to 88. However, the dense version only consists of 1 first-order moment matrix of size 100. In fact, the dense sublevel relaxation gives an upper bound of 1463.5 at level 0 in 10 seconds, yielding the same bound as the sparse case at level 0 but with much less computing time, and 1458.1 at level 8 in 178.1 seconds, providing better bounds than the sparse case at level 6, with less computing time.

Table 4.4: Results obtained by sublevel relaxations for Max-Cut problems.

	Sol./UB	nVar	Density	Sublevel relaxation, $l = 0/4/6/8$, $q = 1$ (level 0 = Shor)								
				upper bounds (RI, RG)					solving time (s)			
g05_60	536	60	50%	550.1	548.1	546.0	544.6	(39.0%, 1.6%)	4.5	10.6	17.6	65.7
g05_80	929	80	50%	950.9	949.0	946.6	944.6	(28.8%, 1.7%)	33.8	56.2	61.8	137.4
g05_100	1430	100	50%	1463.5	1462.0	1459.2	1456.8	(20.0%, 1.9%)	138.7	303.7	328.7	460.3
pm1d_80	227	80	99%	270.0	265.9	262.0	258.8	(26.0%, 14.0%)	15.0	29.4	39.2	128.1
pm1d_100	340	100	99%	405.4	402.2	397.9	393.7	(19.0%, 15.8%)	47.6	69.4	110.2	225.1
pm1s_80	79	80	10%	90.3	86.7	83.6	**82.8**	(**66.4%**, 4.8%)	1.4	4.9	13.4	37.7
pm1s_100	127	100	10%	143.2	141.4	137.6	**135.3**	(**48.8%**, 6.5%)	11.1	24.3	28.6	180.3
pw01_100	2019	100	10%	2125.4	2107.8	2088.1	**2075.0**	(**47.4%**, 2.8%)	13.0	20.5	29.7	285.8
pw05_100	8190	100	50%	8427.7	8416.6	8403.6	8388.1	(16.7%, 2.4%)	136.8	223.0	272.9	400.3
pw09_100	13585	100	90%	13806.0	13797.1	13781.1	13766.5	(17.9%, 1.3%)	141.6	218.4	268.7	442.4
w01_100	651	100	10%	740.9	728.3	710.3	**696.2**	(**49.7%**, 6.9%)	10.5	22.4	35.0	224.7
w05_100	1646	100	50%	1918.0	1902.6	1885.5	1869.7	(17.8%, 13.6%)	138.1	265.8	272.2	403.2
w09_100	2121	100	90%	2500.3	2478.2	2447.3	2422.8	(20.4%, 14.2%)	124.3	255.0	280.8	451.7
g_20	537.4	505	1.6%	570.8	547.1	526.7	**513.4**	(**171.9%**, -4.5%)	0.7	15.1	46.1	102.2
g_40	992.2	1005	0.68%	1032.6	982.4	950.8	**927.6**	(**260.0%**, -6.5%)	1.2	18.6	47.9	102.5
g_60	1387.2	1505	0.43%	1439.9	1368.4	1317.8	**1281.9**	(**300.4%**, -7.6%)	2.8	26.0	74.7	431.1
g_80	1838.1	2005	0.3%	1899.2	1803.8	1744.9	**1698.8**	(**328.0%**, -7.6%)	6.0	23.8	76.0	290.7
g_100	2328.3	2505	0.23%	2398.7	2282.9	2205.1	**2149.3**	(**354.3%**, -7.7%)	3.4	30.1	117.4	428.6
g_120	2655.4	3005	0.19%	2731.7	2588.5	2507.3	**2439.8**	(**382.6%**, -8.1%)	3.8	33.3	113.2	434.5
g_140	3027.2	3505	0.16%	3115.8	2947.9	2856.5	**2782.6**	(**376.1%**, -8.1%)	3.8	46.3	138.4	522.1
g_160	3589.0	4005	0.13%	3670.7	3487.1	3380.7	**3310.9**	(**440.4%**, -7.7%)	8.2	56.5	198.2	506.6
g_180	3953.1	4505	0.12%	4054.7	3855.9	3736.9	**3653.5**	(**394.9%**, -7.6%)	8.8	51.5	277.0	693.4
g_200	4472.3	5005	0.11%	4584.6	4353.3	4228.1	**4132.2**	(**402.8%**, -7.6%)	5.4	52.7	203.2	839.2
G11	564	800	0.25%	629.2	581.3	564.6	**564.6**	(**99.1%**, 0.1%)	4.0	15.8	32.6	36.5
G12	556	800	0.25%	623.9	572.5	559.6	**559.6**	(**94.7%**, 0.6%)	17.8	57.8	54.3	51.9
G13	580	800	0.25%	647.1	594.2	585.1	**584.1**	(**93.9%**, 0.7%)	159.2	241.7	340.2	321.6
G32	1398	2000	0.1%	1567.6	1433.4	1415.9	**1415.9**	(**89.4%**, 1.3%)	622.0	736.3	630.8	628.0
G33	1376	2000	0.1%	1544.3	1415.3	1392.7	**1387.4**	(**93.2%**, 0.8%)	1956.6	2115.8	1221.5	1486.8
G34	1372	2000	0.1%	1546.7	1407.9	1388.2	**1388.2**	(**90.7%**, 1.2%)	3613.5	6580.9	6327.9	6147.4

4.3.2 Maximum clique (Max-Cliq) problem

Given an undirected graph $G(V, E)$ where V is a set of vertices and E is a set of edges, a *clique* is defined to be a set of vertices that is completely interconnected. The *Max-Cliq* problem consists of determining a clique of maximum cardinality. It can be stated as a nonconvex quadratic programming problem over the unit simplex [Pardalos & Phillips 1990] and its general formulation is:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \mathbf{x}^T \mathbf{A} \mathbf{x} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1, \ \mathbf{x} \in [0,1]^n, \end{aligned} \tag{Max-Cliq}$$

where $\mathbf{A}$ is the adjacency matrix of the given graph of n vertices. The constraints $\mathbf{x} \in [0,1]^n$ are equivalent to $x_i(x_i - 1) \leq 0$ for $i = 1, 2, \ldots, n$. The Max-Cliq problem is dense since we have a constraint $\sum_{i=1}^{n} x_i = 1$ involving all the variables. Therefore, we apply the dense sublevel hierarchy. To handle the constraint $\sum_{i=1}^{n} x_i = 1$, we select the q subsets of size l by order as $I_t = \{t, t+1, \ldots, t+l-1\}$ for $t = 1, 2, \ldots, q$. For the constraints $x_i(x_i - 1) \leq 0$, we select the subsets by order as $I_{i,t} = \{i, i+t, \ldots, i+t+l-2\}$ for $t \in [q]$.

We take the same graphs as the ones considered in the Max-Cut instances. Some instances share the same adjacency matrix with different weights, in which case we delete these repeated graphs. LB denotes the lower bound of a given instance, computed by 10^6 random samples. Contrast with the strategy used for the Max-Cut instances, we use sublevel relaxations with level 2 and depth 0, 20, 40, 60, respectively. From Table 4.5 we see that the sublevel relaxation yields large improvement compared to Shor's relaxation. The Max-Cliq problem remains hard to solve as emphasized by the large relative gap, ranging from 662.5% to 3660%.

Table 4.5: Results obtained by sublevel relaxations for Max-Cliq problems.

	LB	nVar	Density	Sublevel relaxation, $l = 2$, $q = 0/20/40/60$ (depth 0 = Shor)								
				upper bounds (RI, RG)					solving time (s)			
g05_60	0.8	60	50%	29.9	19.3	8.3	**6.1**	(**81.8%**, 662.5%)	0.6	1.8	3.6	2.4
g05_80	0.9	80	50%	39.9	29.1	20.1	**8.9**	(**79.5%**, 888.9%)	2.8	7.4	7.5	8.3
g05_100	0.8	100	50%	50.0	39.1	28.9	**18.4**	(**64.2%**, 2200.0%)	6.5	30.9	19.1	33.0
pm1d_80	1.0	80	99%	78.2	57.5	37.6	**17.9**	(**78.1%**, 1690.0%)	2.3	5.4	7.6	4.4
pm1d_100	1.0	100	99%	98.0	77.2	57.5	**37.6**	(**62.3%**, 3660%)	5.6	12.5	22.8	17.3
pm1s_80	0.7	80	10%	8.9	6.2	4.6	**4.6**	(**52.1%**, 557.1%)	2.6	6.1	6.4	9.0
pw01_100	0.6	100	10%	10.6	8.2	5.9	**5.4**	(**51.8%**, 800.0%)	7.5	30.7	20.0	29.6
pw05_100	0.8	100	50%	49.8	39.7	28.9	**18.9**	(**63.0%**, 2262.5%)	7.6	21.9	24.0	26.5
pw09_100	1.0	100	90%	89.2	70.2	51.9	**34.0**	(**62.5%**, 3300%)	8.5	15.0	33.2	28.9

4.3.3 Mixed integer quadratically constrained programming (MIQCP)

The MIQCP problem is of the following form:

$$\min_{\mathbf{x}} \quad \mathbf{x}^T\mathbf{Q}_0\mathbf{x} + \mathbf{b}_0^T\mathbf{x} \tag{MIQCP}$$
$$\text{s.t.} \quad \begin{cases} \mathbf{x}^T\mathbf{Q}_i\mathbf{x} + \mathbf{b}_i^T\mathbf{x} \le c_i, \ i \in [p], \\ \mathbf{A}\mathbf{x} = \mathbf{b}, \ \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \ \mathbf{x}_I \in \mathbb{Z}, \end{cases}$$

where each $\mathbf{Q}_i$ is a symmetric matrix of size $n \times n$, $\mathbf{A}$ is a matrix of size $n \times n$, $\mathbf{b}, \mathbf{b}_i, \mathbf{l}, \mathbf{u}$ are n-dimensional vectors, and each c_i is a real number. The constraints $\mathbf{x}^T\mathbf{Q}_i\mathbf{x} + \mathbf{b}_i^T\mathbf{x} \le c_i$ are called *quadratic* constraints, the constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$ are called *linear* constraints. The constraints $\mathbf{l} \le \mathbf{x} \le \mathbf{u}$ and $x_I \in \mathbb{Z}$ bound the variables and restrict some of them to be integers. In our benchmarks, we only consider the case where $\mathbf{x} \in \{0,1\}^n$, which is also equivalent to $x_i(x_i - 1) = 0$ for $i = 1, 2, \ldots, n$. If we only have bound constraints, then we use the same ordered heuristic as for the Max-Cut problem to select the subsets. If in addition we also have quadratic constraints or linear constraints, then the problem is dense and therefore we consider the dense sublevel hierarchy. For quadratic constraints, we don't apply the sublevel relaxation to them, i.e., $l = q = 0$. However, if $\mathbf{Q}_i$ equals the identity matrix, then we use the same heuristic as the linear constraints: we select the subsets by order as $I_t = \{t, t+1, \ldots, t+l-1\}$ for $t \in [q]$.

The following classes of problems and their solutions are from the Biq-Mac library, where there are neither quadratic constraints $\mathbf{x}^T\mathbf{Q}_i\mathbf{x} + \mathbf{b}_i^T\mathbf{x} \le c_i$ nor linear constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$. We only have integer bound constraints $\mathbf{x} \in \{0,1\}^n$.

- bqp$n$-i, with 10% density. All the coefficients have uniformly chosen integer values in $[-100, 100]$, $n = 50, 100, 250, 500$;

- gka$i$a, with dimensions in $[30, 100]$ and densities in $[0.0625, 0.5]$. The diagonal coefficients lie in $[-100, 100]$ and the off-diagonal coefficients belong to $[-100, 100]$;

- gka$i$b, with dimensions in $[20, 125]$ and density 1. The diagonal coefficients lie in $[-63, 0]$ and the off-diagonal coefficients belong to $[0, 100]$;

- gka$i$c, dimensions in $[40, 100]$ and densities in $[0.1, 0.8]$. Diagonal coefficients in $[-100, 100]$, off-diagonal coefficients in $[-50, 50]$;

- gka$i$d, with dimension 100 and densities in $[0.1, 1]$. The diagonal coefficients lie in $[-75, 75]$ and the off-diagonal coefficients belong to $[-50, 50]$.

We also select some instances and their solutions from the QPLIB library with ID 0032, 0067, 0633, 2512, 3762, 5935 and 5944, in which we have additional linear constraints $\mathbf{Ax} = \mathbf{b}$. For the instance 0032, there are 50 continuous variables and 50 integer variables. For the two instances 5935 and 5944, we maximize the objective, the others are minimization problems.

Similarly to the Max-Cut instances, Table 4.6 summarizes the basic information and clique structure of each instance. Table 4.7 is a summary of basic information and the number of quadratic, linear, bound constraints of the instances from the QPLIB library.

In Table 4.8, we show the lower bounds and running time obtained by solving the sublevel relaxations with depth 1 and level 0, 4, 6, 8, respectively. We see that when the problem has a good sparsity structure or is of low dimension, the sublevel relaxation performs very well and provides the exact solution, in particular for the two instances gka2a and gka7a. For dense problems, we are not able to find the exact solution, but still have improvements between 20% and 40% compared to Shor's relaxation. Notice that for the instances gka1b to gka10b, even though we have an improvement ratio ranging from 24.0% to 77.9%, the relative gap is very high, varying from 38.2% to 947.2%. This means that these problems themselves are very hard to solve, so that the gap between the results of Shor's relaxation and the exact optimal solution is very large. Even though the sublevel relaxation yields substantial improvement compared to Shor's relaxation, it's still far away from the true optimum.

Table 4.6: Summary of the basic information and sparse structure of MIQCP instances.

	nVar	Density	nCliques	MaxClique	MinClique	nQuad	nLin	nBound
bqp50-1	50	10%	36	15	3	0	0	50
bqp100-1	100	10%	52	49	4	0	0	100
gka1a	50	10%	36	15	1	0	0	50
gka2a	60	10%	41	20	3	0	0	60
gka3a	70	10%	44	27	3	0	0	70
gka4a	80	10%	48	33	4	0	0	80
gka5a	50	20%	25	26	4	0	0	50
gka6a	30	40%	11	20	7	0	0	30
gka7a	30	50%	10	21	10	0	0	30
gka8a	100	62.5%	64	37	2	0	0	100
gka1b	20	100%	2	19	19	0	0	20
gka2b	30	100%	2	29	29	0	0	30
gka3b	40	100%	2	39	38	0	0	40
gka4b	50	100%	2	49	47	0	0	50
gka5b	60	100%	2	59	56	0	0	60
gka6b	70	100%	2	69	67	0	0	70
gka7b	80	100%	2	79	77	0	0	80
gka8b	90	100%	2	89	87	0	0	90
gka9b	100	100%	2	99	97	0	0	100
gka10b	125	100%	2	124	124	0	0	125
gka1c	40	80%	4	37	25	0	0	40
gka2c	50	60%	6	45	26	0	0	50
gka3c	60	40%	14	47	17	0	0	60
gka4c	70	30%	22	49	12	0	0	70
gka5c	80	20%	27	54	11	0	0	80
gka6c	90	10%	46	45	4	0	0	90
gka7c	100	10%	51	50	3	0	0	100
gka1d	100	10%	50	51	4	0	0	100
gka2d	100	20%	30	71	11	0	0	100
gka3d	100	30%	23	78	18	0	0	100
gka4d	100	40%	15	86	31	0	0	100
gka5d	100	50%	13	88	36	0	0	100
gka6d	100	60%	10	91	47	0	0	100
gka7d	100	70%	7	94	57	0	0	100
gka8d	100	80%	6	95	68	0	0	100
gka9d	100	90%	5	96	79	0	0	100
gka10d	100	100%	2	99	95	0	0	100

4.3.4 Quadratically constrained quadratic problem (QCQP)

A QCQP can be cast as follows:

$$\min_{\mathbf{x}} \quad \mathbf{x}^T\mathbf{Q}_0\mathbf{x} + \mathbf{b}_0^T\mathbf{x} \qquad \text{(QCQP)}$$
$$\text{s.t.} \quad \begin{cases} \mathbf{x}^T\mathbf{Q}_i\mathbf{x} + \mathbf{b}_i^T\mathbf{x} \le c_i, \ i \in [p], \\ \mathbf{A}\mathbf{x} = \mathbf{b}, \ \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \end{cases}$$

where each $\mathbf{Q}_i$ is a symmetric matrix of size $n \times n$, $\mathbf{A}$ is a matrix of size $n \times n$, $\mathbf{b}, \mathbf{b}_i, \mathbf{l}, \mathbf{u}$ are n-dimensional vectors, and each c_i is a real number. This is very similar to the MIQCP except that we drop out the integer constraints. Therefore, we use the same strategy to select the subsets in the sublevel relaxation.

We take the MIQCP instances from the Biq-Mac library whose size are larger

Table 4.7: Summary of the basic information and constraint structure of MIQCP instances from QPLIB library.

	nVar	Density	nQuad	nLin	nBound
qplib0032	100	89%	0	52	100
qplib0067	80	89%	0	1	80
qplib0633	75	99%	0	1	75
qplib2512	100	28%	0	20	100
qplib3762	90	28%	0	480	90
qplib5935	100	28%	0	1237	100
qplib5944	100	28%	0	2475	100

or equal than 50. We add one dense quadratic constraint $||\mathbf{x}||_2^2 = 1$, and relax the integer bound constraints $\mathbf{x} \in \{0,1\}^n$ to linear bound constraints $\mathbf{x} \in [0,1]^n$. UB denotes the upper bound obtained by selecting the minimum value over 10^6 random evaluations.

We also select some instances and their solutions from the QPLIB library with ID 1535, 1661, 1675, 1703 and 1773. These instances have more than one quadratic constraint and involve linear constraints.

Table 4.9 is a summary of basic information as well as the number of quadratic, linear, and bound constraints of the instances from the QPLIB library.

In Table 4.10, we show the lower bounds and running time obtained by the sublevel relaxation with depth 1 for the instances from the QPLIB library, 10 for the instances adapted from the Biq-Mac library, and level 0, 4, 6, 8, respectively. We see that the sublevel relaxation yields a uniform improvement compared to Shor's relaxation. However, for the QCQP problems adapted from the MIQCP instances, it is very hard to find the exact optimal solution as the relative gap varies from 60.5% to 77.0%. This is in deep contrast with the instances from the QPLIB library which are relatively easier to solve as the relative gap varies from 9.4% to 13.8%.

Table 4.8: Results obtained by sublevel relaxations for MIQCP problems.

	Sol.	nVar	Density	Sublevel relaxation, $l = 0/4/6/8$, $q = 1$ (level 0 = Shor)								
				lower/upper bounds (RI, RG)					solving time (s)			
bqp50-1	-2098	50	10%	-2345.5	-2136.3	-2116.3	**-2105.4**	(**97.0%**, 0.4%)	0.1	0.5	1.3	3.4
bqp100-1	-7970	100	10%	-8721.1	-8358.2	-8215.1	**-8101.8**	(**82.5%**, 1.7%)	8.8	16.7	21.8	87.9
gka1a	-3414	50	10%	-3623.3	-3453.2	-3432.6	**-3428.5**	(**93.1%**, 0.4%)	0.1	0.6	0.9	1.8
gka2a	-6063	60	10%	-6204.3	-6076.3	-6063.0	**-6063.0**	(**100%**, 0%)	0.3	1.0	4.6	8.5
gka3a	-6037	70	10%	-6546.2	-6291.5	-6182.6	**-6106.3**	(**86.4%**, 1.1%)	0.7	1.6	6.1	31.0
gka4a	-8598	80	10%	-8935.1	-8767.3	-8713.7	**-8676.0**	(**76.9%**, 0.9%)	2.1	3.4	10.1	30.0
gka5a	-5737	50	20%	-5979.9	-5789.9	-5760.3	**-5750.0**	(**94.6%**, 0.2%)	0.7	1.4	6.2	31.0
gka6a	-3980	30	40%	-4190.2	-4008.9	-3986.0	**-3982.5**	(**98.8%**, 0.1%)	0.2	0.6	3.9	23.6
gka7a	-4541	30	50%	-4696.6	-4566.8	-4541.1	**-4541.1**	(**100%**, 0%)	0.3	0.8	4.9	23.1
gka8a	-11109	100	62.5%	-11283.8	-11148.0	-11124.8	**-11114.0**	(**97.1%**, 0.05%)	2.3	2.7	7.5	19.5
gka1b	-133	20	100%	-362.9	-295.1	-253.6	**-183.8**	(**77.9%**, 38.2%)	0.1	0.5	2.4	25.0
gka2b	-121	30	100%	-505.7	-425.3	-325.4	**-282.5**	(**58.0%**, 133.5%)	0.2	0.7	4.0	29.9
gka3b	-118	40	100%	-718.0	-535.6	-483.4	-437.7	(**46.7%**, 270.9%)	0.7	1.4	6.5	45.9
gka4b	-129	50	100%	-809.8	-670.9	-614.2	-571.5	(35.0%, 343.0%)	1.9	3.3	14.3	65.2
gka5b	-150	60	100%	-1034.8	-820.9	-736.8	-705.5	(37.2%, 370.3%)	3.2	8.4	15.5	76.1
gka6b	-146	70	100%	-1279.0	-972.2	-894.8	-833.5	(39.3%, 470.9%)	9.1	11.6	26.6	86.5
gka7b	-160	80	100%	-1362.5	-1138.1	-1031.0	-982.6	(31.6%, 514.1%)	26.1	31.2	50.8	136.1
gka8b	-145	90	100%	-1479.1	-1269.8	-1190.2	-1120.9	(26.8%, 673.0%)	40.5	60.1	102.3	187.0
gka9b	-137	100	100%	-1663.6	-1385.4	-1298.9	-1212.6	(29.5%, 785.1%)	65.9	92.3	111.2	256.3
gka10b	-154	125	100%	-2073.1	-1782.1	-1707.1	-1612.7	(24.0%, 947.2%)	285.8	413.3	452.2	700.9
gka1c	-5058	40	80%	-5161.1	-5102.9	-5077.9	**-5073.7**	(**84.8%**, 0.3%)	0.8	1.6	5.3	41.9
gka2c	-6213	50	60%	-6392.6	-6291.3	-6263.1	**-6246.2**	(**81.5%**, 0.5%)	1.9	2.8	7.8	50.3
gka3c	-6665	60	40%	-6849.9	-6730.7	-6703.1	**-6688.1**	(**87.5%**, 0.3%)	6.1	9.3	15.9	62.1
gka4c	-7398	70	30%	-7647.1	-7527.7	-7494.9	**-7462.8**	(**74.0%**, 0.9%)	13.1	18.4	24.6	88.1
gka5c	-7362	80	20%	-7684.5	-7543.7	-7474.6	**-7412.8**	(**84.2%**, 0.7%)	15.1	27.7	40.3	112.8
gka6c	-5824	90	10%	-6065.8	-5932.2	-5869.7	**-5847.4**	(**90.3%**, 0.4%)	10.0	11.0	19.0	57.4
gka7c	-7225	100	10%	-7422.7	-7297.8	-7264.3	**-7248.7**	(**88.0%**, 0.3%)	12.4	13.9	22.1	55.6
gka1d	-6333	100	10%	-6592.7	-6475.3	-6403.1	**-6369.6**	(**85.9%**, 0.6%)	11.4	13.4	29.1	71.3
gka2d	-6579	100	20%	-7234.2	-6980.5	-6897.9	**-6811.6**	(**64.5%**, 3.5%)	42.3	70.8	70.6	193.7
gka3d	-9261	100	30%	-9963.0	-9686.2	-9591.7	**-9523.6**	(**62.6%**, 2.8%)	164.8	200.4	262.7	330.0
gka4d	-10727	100	40%	-11592.5	-11303.3	-11175.4	**-11096.5**	(**57.3%**, 3.4%)	302.2	259.1	191.8	387.7
gka5d	-11626	100	50%	-12632.1	-12381.6	-12274.7	-12185.0	(**44.4%**, 4.8%)	324.3	256.3	294.3	380.2
gka6d	-14207	100	60%	-15235.3	-14938.2	-14834.9	**-14720.2**	(**50.1%**, 3.6%)	236.6	239.7	221.9	437.9
gka7d	-14476	100	70%	-15672.0	-15413.2	-15267.6	-15173.6	(**41.7%**, 4.8%)	138.8	225.9	150.0	314.6
gka8d	-16352	100	80%	-17353.3	-17011.5	-16887.6	**-16794.3**	(**55.8%**, 2.7%)	271.5	277.9	291.6	408.6
gka9d	-15656	100	90%	-17010.9	-16652.0	-16513.3	-16409.6	(**44.4%**, 4.8%)	390.5	419.8	367.0	513.5
gka10d	-19102	100	100%	-20421.4	-20121.7	-19974.1	-19863.8	(**44.3%**, 4.0%)	77.8	83.4	130.2	244.8
qplib0032	10.1	100	99%	-19751	-16491	-15962	-15440	(21.8%, 152971.3%)	18.1	19.4	37.0	94.7
qplib0067	-110942	80	89%	-116480	-112923	-112615	**-112478**	(**72.3%**, 1.4%)	6.2	11.1	21.9	158.3
qplib0633	79.6	75	99%	70.9	74.0	75.1	**75.7**	(**55.2%**, 4.9%)	2.9	10.1	27.1	140.0
qplib2512	135028	100	77%	-441284	-125060	27898	**82909**	(**91.0%**, 38.6%)	18.6	19.9	53.4	278.6
qplib3762	-296	90	28%	-345.6	-330.8	-319.9	**-309.5**	(**72.8%**, 4.6%)	6.3	18.1	50.7	183.4
qplib5935	4758	100	99%	67494	40148	36842	**28812**	(**61.7%**, 505.5%)	12.8	39.4	259.0	1745.3
qplib5944	1829	100	99%	66934	27437	23142	**19784**	(**72.4%**, 981.7%)	15.6	182.1	2304.3	13204.6

Table 4.9: Summary of the basic information and constraint structure of QCQP instances from QPLIB library.

	nVar	Density	nQuad	nLin	nBound
qplib1535	60	94%	60	6	60
qplib1661	60	95%	1	12	60
qplib1675	60	49%	1	12	60
qplib1703	60	98%	30	6	60
qplib1773	60	95%	1	6	60

Table 4.10: Results obtained by sublevel relaxations for QCQP problems.

	Sol./UB	nVar	Density	Sublevel relaxation, $l = 0, 4, 6, 8$, $q = 1, 10$ (level 0 = Shor)								
				lower bounds (RI, RG)					solving time (s)			
bqp50-1	-99	50	10%	-215.7	-195.4	-180.6	-172.5	(37.0%, 74.2%)	0.8	2.4	12.3	88.5
bqp100-1	-67.2	100	10%	-323.1	-304.7	-296.1	-290.0	(12.9%, 331.5%)	21.4	22.7	56.9	249.6
gka1a	-109.5	50	10%	-241.8	-224.1	-219.8	-213.8	(21.2%, 95.3%)	0.8	1.9	10.0	65.6
gka2a	-140.7	60	10%	-275.3	-260.9	-258.7	-251.6	(17.6%, 78.8%)	1.7	3.8	16.8	126.9
gka3a	-143.2	70	10%	-300.0	-284.6	-278.8	-275.0	(15.9%, 92.0%)	3.6	9.1	23.1	121.1
gka4a	-126.2	80	10%	-311.0	-288.3	-282.9	-280.0	(16.8%, 121.9%)	6.8	7.9	25.4	160.0
gka5a	-180.2	50	20%	-351.8	-319.3	-306.4	-299.1	(30.7%, 66.0%)	0.7	2.8	15.1	75.6
gka8a	-122.5	100	62.5%	-320.1	-306.8	-302.1	-299.5	(10.4%, 144.5%)	21.3	23.5	72.5	232.0
gka4b	-63	50	100%	-381.4	-326.2	-302.4	-280.8	(31.6%, 345.7%)	0.7	1.8	17.4	79.2
gka5b	-63	60	100%	-446.8	-377.2	-348.9	-327.4	(31.1%, 419.7%)	1.1	4.2	19.5	117.2
gka6b	-63	70	100%	-496.6	-409.9	-385.9	-366.8	(29.9%, 482.2%)	3.2	5.3	17.4	118.1
gka7b	-63	80	100%	-518.3	-447.1	-421.6	-404.3	(25.0%, 541.7%)	5.9	9.5	21.6	170.3
gka8b	-63	90	100%	-534.5	-472.7	-449.0	-430.1	(22.1%, 582.7%)	10.9	16.0	36.0	148.9
gka9b	-63	100	100%	-573.0	-501.3	-477.0	-455.8	(23.0%, 623.5%)	19.7	36.6	42.5	191.4
gka10b	-63	125	100%	-639.4	-569.7	-553.7	-533.6	(18.4%, 747.0%)	80.1	82.1	110.9	410.1
gka2c	-159.1	50	60%	-290.0	-269.3	-261.6	-255.4	(26.4%, 60.5%)	0.8	2.5	11.2	80.8
gka3c	-126.3	60	40%	-271.2	-240.2	-235.4	-231.3	(27.5%, 83.1%)	1.7	4.4	16.0	103.1
gka4c	-123.0	70	30%	-292.7	-263.7	-254.4	-247.9	(26.4%, 101.5%)	3.0	6.5	19.7	155.6
gka5c	-114.0	80	20%	-239.1	-225.9	-223.2	-220.4	(14.9%, 93.3%)	10.6	9.6	32.3	166.5
gka6c	-100	90	10%	-198.8	-190.8	-186.7	-182.4	(16.6%, 82.4%)	12.3	15.9	33.1	216.5
gka7c	-100	100	10%	-225.8	-213.7	-210.5	-208.5	(13.8%, 108.5%)	21.4	28.2	63.4	323.1
gka1d	-75	100	10%	-197.9	-182.5	-177.0	-174.5	(19.0%, 132.7%)	19.3	25.9	64.8	243.9
gka2d	-87.2	100	20%	-259.6	-242.2	-233.8	-229.5	(17.5%, 163.2%)	23.5	28.2	61.1	254.3
gka3d	-88.1	100	30%	-304.0	-281.6	-274.1	-267.5	(16.9%, 203.6%)	26.2	28.9	49.2	278.4
gka4d	-105.5	100	40%	-375.2	-340.1	-326.0	-317.5	(21.4%, 201.0%)	21.6	21.9	53.2	270.7
gka5d	-131.9	100	50%	-383.6	-351.5	-341.5	-332.3	(20.4%, 152.0%)	20.4	22.7	41.3	257.1
gka6d	-137.7	100	60%	-443.1	-400.0	-391.0	-378.9	(21.0%, 175.2%)	23.8	23.3	48.6	254.2
gka7d	-156.3	100	70%	-453.9	-421.4	-406.6	-397.4	(19.0%, 154.3%)	21.4	22.5	71.7	217.8
gka8d	-147.6	100	80%	-488.0	-441.1	-423.3	-414.2	(21.7%, 180.6%)	21.7	25.0	47.0	232.8
gka9d	-179.6	100	90%	-539.7	-487.7	-469.2	-456.8	(23.0%, 154.3%)	20.6	21.5	45.1	222.2
gka10d	-187.0	100	100%	-552.4	-505.7	-491.8	-478.4	(20.3%, 155.8%)	23.2	24.6	56.9	196.5
qplib1535	-11.6	60	94%	-13.9	-13.5	-13.3	-13.2	(30.4%, 13.8%)	1.4	4.3	13.7	99.2
qplib1661	-16.0	60	95%	-18.4	-18.1	-17.8	-17.5	(37.5%, 9.4%)	1.4	3.0	14.7	96.4
qplib1675	-75.7	60	49%	-93.1	-87.0	-85.2	**-83.8**	(**53.4%**, 10.7%)	1.0	4.2	19.2	147.8
qplib1703	-132.8	60	98%	-152.8	-147.0	-145.2	**-143.5**	(**46.5%**, 8.06%)	1.2	4.2	20.8	109.3
qplib1773	-14.6	60	95%	-17.3	-16.8	-16.6	-16.4	(33.3%, 12.3%)	1.1	4.0	-14.0	89.3

Chapter 5

Robustness Verification and Related Problems

Contents

As described in Section 1.3.2, to verify the robustness of a neural network F is indeed to verify whether a property holds, e.g. $F(\mathbf{x}) \in \mathcal{Y}$ for all $\mathbf{x} \in \mathcal{X}$. In this chapter, we formulate this problem into polynomial optimization problems, and apply sublevel relaxation described in Chapter 4 to obtain valid upper bounds of the original problem. We also propose two related problems: Lipschitz constant estimation and ellipsoid propagation, both of which can be applied to certify robustness of neural networks.

5.1 Semialgebraicity of ReLU, ∂ReLU, and L_p-norms

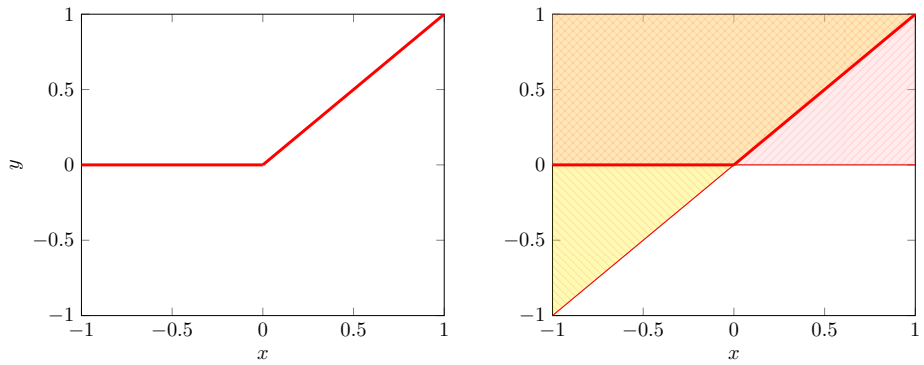
The key reason why neural networks with ReLU activation function can be tackled using polynomial optimization techniques is semialgebraicity of the ReLU function, i.e., it can be expressed with a system of polynomial (in)equalities.

5.1.1 ReLU function

Recall that $\mathrm{ReLU}(x) = \max\{0, x\}$. For $x, y \in \mathbb{R}$, we have

$$y = \mathrm{ReLU}(x) \iff y(y - x) = 0,\ y \geq x,\ y \geq 0. \tag{5.1}$$

By Definition 3.1, the graph of ReLU function is a semialgebraic set, as shown in Figure 5.1. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we denote by $\mathrm{ReLU}(\mathbf{x})$ the coordinate-wise evaluation of ReLU function.



(a) $y = \max\{x, 0\}$. (b) $y(y - x) = 0,\ y \geq x,\ y \geq 0$.

Figure 5.1: ReLU function (left) and its semialgebraicity (right).

Remark. There are other semialgebraic formulations of ReLU function, for example:

$$\begin{aligned} y = \max\{x, 0\} &\iff y(y - x) = 0,\ y \geq \alpha x \text{ with } \alpha \in (0, 1) \\ &\iff y = (x + z)/2,\ x^2 = z^2,\ z \geq 0. \end{aligned}$$

All these formulations are equivalent in mathematics. However, it shows different numerical behavior when using different formulation in the optimization solvers. The one we chose in this thesis has the best empirical performance.

5.1.2 ∂ReLU function

Similar with ReLU function, the subdifferential of ReLU function is also a semialgebraic function. We first give the formal definition of subdifferential.

Definition 5.1. Let $\Omega \in \mathbb{R}^n$ and $f : \Omega \to \mathbb{R}$ be a convex function. The *subdifferential* of f at $\mathbf{x} \in \Omega$, denoted by $\partial f(\mathbf{x})$, is defined as the nonempty set

$$\partial f(\mathbf{x}) := \{\mathbf{z} \in \mathbb{R}^n : f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{z}^T(\mathbf{y} - \mathbf{x})\}, \tag{5.2}$$

the vector $\mathbf{z}$ satisfying (5.2) is called *subgradient* of f at $\mathbf{x}$.

Remark. If f is differentiable at $\mathbf{x}$, then $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$.

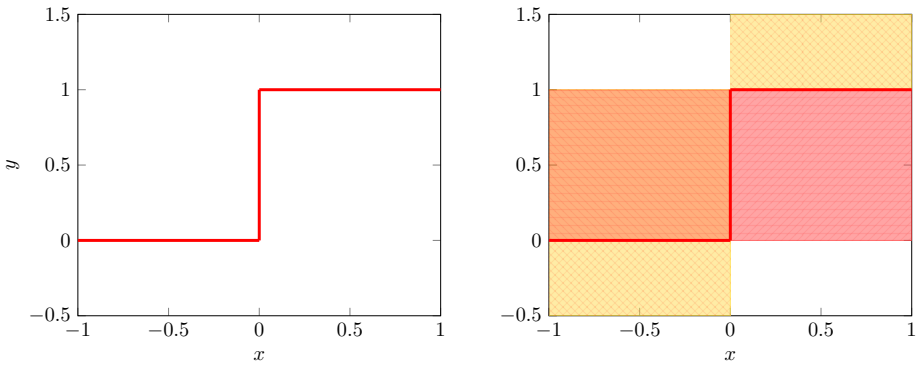
Applying Definition 5.1 to ReLU function, we obtain

$$\partial\mathrm{ReLU}(x) = \begin{cases} \{0\}, & x < 0; \\ [0, 1], & x = 0; \\ \{1\}, & x > 0. \end{cases}$$

Therefore, for $x, y \in \mathbb{R}$, we have

$$y \in \partial\mathrm{ReLU}(x) \iff y(y-1) \leq 0, xy \geq 0, x(y-1) \geq 0. \tag{5.3}$$

By Definition 3.1, the graph of ∂ReLU is also a semiaglebraic set, as shown in Figure 5.2. If $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we denote by $\partial\mathrm{ReLU}(\mathbf{x})$ the coordinate-wise evaluation of ∂ReLU.



(a) $y \in \partial\mathrm{ReLU}(x)$. (b) $y(y-1) \leq 0,\ xy \geq 0,\ x(y-1) \geq 0$.

Figure 5.2: Subdifferential of ReLU function (left) and its semialgebraicity (right).

5.1.3 L_p norm

Recall that for $p \in \mathbb{Z}_+ \cup \{\infty\}, \mathbf{x} \in \mathbb{R}^n$, the L_p norm of $\mathbf{x}$ is defined as

$$\|\mathbf{x}\|_p = \begin{cases} \left(\sum_{i=1}^n |x_i|^p\right)^{\frac{1}{p}}, & \text{if } p < \infty; \\ \max_{i \in [n]} |x_i|, & \text{if } p = \infty. \end{cases}$$

Hence the ball constraint $\|\mathbf{x}\|_p \leq 1$ is equivalent to polynomial inequalities

$$\|\mathbf{x}\|_p \leq 1 \iff \begin{cases} \sum_{i=1}^n x_i^p \leq 1, & \text{if } p = 2k \text{ for some } k \in \mathbb{Z}_+; \\ \sum_{i=1}^n y_i^p \leq 1,\ y_i^2 = x_i^2,\ y_i \geq 0, & \text{if } p = 2k+1 \text{ for some } k \in \mathbb{Z}_+; \\ x_i^2 \leq 1,\ \forall i \in [n], & \text{if } p = \infty. \end{cases}$$

By Definition 3.1, the unit ball $\{\mathbf{x} : \|\mathbf{x}\|_p \leq 1\}$ is a semialgebraic set for any L_p norm with $p \in \mathbb{Z}_+ \cup \{\infty\}$. Similarly, the ball at center $\bar{\mathbf{x}} \in \mathbb{R}^n$ with radius ε, defined by $\{\mathbf{x} : \|\mathbf{x} - \bar{\mathbf{x}}\|_p \leq \varepsilon\}$ is a semialgebraic set for any L_p norm with $p \in \mathbb{Z}_+ \cup \{\infty\}$.

5.2 Lipschitz Constant Estimation

Recall that an application $f : \mathcal{X} \to \mathbb{R}^m$, defined on a convex open set $\mathcal{X} \subseteq \mathbb{R}^n$, is L-Lipschitz with respect to norm $\|\cdot\|$ if for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, we have $\|f(\mathbf{x}) - f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$. In particular, if f is a function from $\mathcal{X}$ to $\mathbb{R}$, then $\|f(\mathbf{x}) - f(\mathbf{y})\| = |f(\mathbf{x}) - f(\mathbf{y})|$. The Lipschitz constant of f with respect to norm $\|\cdot\|$, denoted by $L_{f,\|\cdot\|}$, is the infimum of all those valid Ls:

$$L_{f,\|\cdot\|} := \inf\{L : \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}, \|f(\mathbf{x}) - f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|\}\,. \tag{5.4}$$

For deep neural networks, Lipschitz constants play an important role in many applications related to robustness certification which has emerged as an active topic. Related works include [Raghunathan *et al.* 2018b, Fazlyab *et al.* 2020] based on Shor's relaxation, [Dathathri *et al.* 2020] based on a first-order SDP solver encoded in JAX[1] (a Python library designed for high-performance machine learning research), [Xue *et al.* 2022] based on SDP exploring chordal sparsity, [Dvijotham *et al.* 2018, Wong & Kolter 2018] based on LP relaxation, [Tjeng *et al.* 2019] based on MILP, [Combettes & Pesquet 2020] based on compositions of nonexpansive averaged operators and affine operators, and [Boopathy *et al.* 2019, Zhang *et al.* 2018, Weng *et al.* 2018c, Weng *et al.* 2018a] based on outer polytope approximation.

We follow a different route and compute upper bounds on the Lipschitz constant of neural networks. Upper bounds on Lipschitz constants of deep networks can be obtained by a product of the layer-wise Lipschitz constants, see [Huster *et al.* 2018]. This is however extremely loose and has many limitations. An improved upper bound via a finer product is proposed by [Virmaux & Scaman 2018]. Depart from these approaches, [Latorre *et al.* 2020] propose a nonconvex QCQP formulation to estimate the Lipschitz constant of neural networks, for which Shor's relaxation allows to obtain a valid upper bound. Alternatively, using the LP hierarchy, they obtain tighter upper bounds compared to Shor's relaxation. [Fazlyab *et al.* 2019b] also propose an SDP-based method to provide upper bounds of the Lipschitz constant. However this method is restricted to L_2 norm whereas most robustness certification problems in deep learning are rather concerned with the L_∞-norm.

5.2.1 Problem setting

Generally speaking, if an application $f : \mathcal{X} \to \mathbb{R}^m$ is smooth. Then the Lipschitz constant of f w.r.t norm $\|\cdot\|$ is equal to the supremum of the operator norm $|||\cdot|||$ of the Jacobian of f. In particular, if f is a function from $\mathcal{X}$ to $\mathbb{R}$, it reduces to the dual norm $\|\cdot\|_*$ of the gradient of f.

Theorem 5.1. *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex open set and $f : \mathcal{X} \to \mathbb{R}^m$ is a smooth application defined on $\mathcal{X}$. Denote by $L_{f,\|\cdot\|}$ its Lipschitz constant w.r.t. norm $\|\cdot\|$,*

[1] https://github.com/google/jax

then

$$L_{f,\|\cdot\|} = \sup_{\mathbf{x}\in\mathcal{X}} |||\mathcal{J}_f(\mathbf{x})|||, \tag{5.5}$$

where $\mathcal{J}_f(\mathbf{x})$ *is the Jacobian matrix of* f *evaluated at* $\mathbf{x}$*, and* $|||\cdot|||$ *is the operator norm w.r.t norm* $\|\cdot\|$ *defined by* $|||\mathbf{A}||| := \inf\{\lambda \geq 0 : \|\mathbf{A}\mathbf{x}\| \leq \lambda\|\mathbf{x}\|, \forall \mathbf{x} \in \mathbb{R}^n\}$ *for all* $\mathbf{A} \in \mathbb{R}^{m\times n}$*. In particular, if* f *is a function defined on* $\mathcal{X}$*, then*

$$L_{f,\|\cdot\|} = \sup_{\mathbf{x}\in\mathcal{X}} \|\nabla f(\mathbf{x})\|_*, \tag{5.6}$$

where $\nabla f(\mathbf{x})$ *is the gradient of* f *evaluated at* $\mathbf{x}$*, and* $\|\cdot\|_*$ *is the dual norm of* $\|\cdot\|$ *defined by* $\|\mathbf{x}\|_* := \sup_{\|\mathbf{t}\|\leq 1} |\langle \mathbf{t}, \mathbf{x}\rangle|$ *for all* $\mathbf{x} \in \mathbb{R}^n$*.*

Proof : We only prove (5.5) for the case when f is a vector-valued application. By the convexity and smoothness of f, using Leibniz integral rule, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

$$\begin{aligned}
\|f(\mathbf{y}) - f(\mathbf{x})\| &= \left\| \int_0^1 \mathcal{J}_f(\mathbf{x} + t(\mathbf{y}-\mathbf{x}))(\mathbf{y}-\mathbf{x})\mathrm{d}t \right\| \\
&\leq \int_0^1 |||\mathcal{J}_f(\mathbf{x} + t(\mathbf{y}-\mathbf{x}))||| \cdot \|\mathbf{y}-\mathbf{x}\|\mathrm{d}t \\
&\leq \int_0^1 \sup_{\mathbf{x}\in\mathcal{X}} |||\mathcal{J}_f(\mathbf{x})||| \cdot \|\mathbf{y}-\mathbf{x}\|\mathrm{d}t \\
&= \sup_{\mathbf{x}\in\mathcal{X}} |||\mathcal{J}_f(\mathbf{x})||| \cdot \|\mathbf{y}-\mathbf{x}\|,
\end{aligned} \tag{5.7}$$

which proves that $L_{f,\|\cdot\|} \leq \sup_{\mathbf{x}\in\mathcal{X}} |||\mathcal{J}_f(\mathbf{x})|||$.

In order to prove $L_{f,\|\cdot\|} \geq \sup_{\mathbf{x}\in\mathcal{X}} |||\mathcal{J}_f(\mathbf{x})|||$, one only needs to prove that $L_{f,\|\cdot\|} \geq \sup_{\mathbf{x}\in\mathcal{X}} |||\mathcal{J}_f(\mathbf{x})||| - \varepsilon$ for any $\varepsilon > 0$. For a fixed ε, let $\mathbf{z}_\varepsilon \in \mathbb{R}^n$ be the vector such that $|||\mathcal{J}_f(\mathbf{z}_\varepsilon)||| \geq \sup_{\mathbf{x}\in\mathcal{X}} |||\mathcal{J}_f(\mathbf{x})||| - \varepsilon$. Since $\mathcal{X}$ is open, there exists a sequence $\{\mathbf{v}_i\}_{i=1}^\infty$ in $\mathbb{R}^n$ such that $\mathbf{z}_\varepsilon + \mathbf{v}_i \in \mathcal{X}$ for all i and $\lim_{i\to\infty} \|\mathbf{v}_i\| = 0$. By Taylor formula,

$$f(\mathbf{z}_\varepsilon + \mathbf{v}_i) - f(\mathbf{z}_\varepsilon) = \mathcal{J}_f(\mathbf{z}_\varepsilon) \cdot \mathbf{v}_i + o(\|\mathbf{v}_i\|) \cdot \mathbf{v}_i.$$

Hence $\|f(\mathbf{z}_\varepsilon + \mathbf{v}_i) - f(\mathbf{z}_\varepsilon)\| = \|\mathcal{J}_f(\mathbf{z}_\varepsilon) \cdot \mathbf{v}_i + o(\|\mathbf{v}_i\|) \cdot \mathbf{v}_i\|$. By the definition of Lipschitz constant,

$$L_{f,\|\cdot\|} \geq \left\| \frac{f(\mathbf{z}_\varepsilon + \mathbf{v}_i) - f(\mathbf{z}_\varepsilon)}{\|\mathbf{v}_i\|} \right\| = \left\| \frac{\mathcal{J}_f(\mathbf{z}_\varepsilon) \cdot \mathbf{v}_i}{\|\mathbf{v}_i\|} + o(1) \cdot \mathbf{v}_i \right\| \geq |||\mathcal{J}_f(\mathbf{z}_\varepsilon)||| - o(\|\mathbf{v}_i\|). \tag{5.8}$$

Let $i \to \infty$, by equation (5.8) and the fact that $\lim_{i\to\infty} o(\|\mathbf{v}_i\|) = 0$, we have

$$L_{f,\|\cdot\|} \geq |||\mathcal{J}_f(\mathbf{z}_\varepsilon)||| \geq \sup_{\mathbf{x}\in\mathcal{X}} |||\mathcal{J}_f(\mathbf{x})||| - \varepsilon,$$

for all $\varepsilon > 0$, which conclude the proof. □

Theorem 5.1 gives us an explicit formulation of calculating the Lipschitz con-

stant of smooth functions defined on convex open sets. Unfortunately, for ReLU neural network $F : \Omega \to \mathbb{R}^K$ where $\Omega \subseteq \mathbb{R}^{p_0}$ is a convex set (e.g. the unit ball), it is not smooth due to the non-smoothness of the ReLU function. Therefore, we cannot directly apply Theorem 5.1 to neural networks with ReLU activation function. On the other hand, we can still use the same methodology to compute the *generalized Jacobian* or *generalized gradient* and prove that they are valid upper bounds of the Lipschitz constant.

For DNNs: We first consider fully-connected deep neural networks with ReLU activation function. Suppose we have a pre-trained network F with parameters $\mathbf{W} = (\mathbf{A}_1, \mathbf{b}_1; \ldots; \mathbf{A}_L, \mathbf{b}_L; \mathbf{C})$, where $\mathbf{A}_i \in \mathbb{R}^{p_i \times p_{i-1}}, \mathbf{b}_i \in \mathbb{R}^{p_i}, \mathbf{C} \in \mathbb{R}^{K \times p_L}$, L is the number of hidden layers, p_0 is the number of neurons in the input layer, p_i is the number of neurons in the hidden layer, and K is the number of labels. We say that a network F has size $(p_0, p_1, \ldots, p_L)$ if F has L hidden layers, p_0 neurons in the input layer and p_i neurons in the corresponding hidden layers. Denote by $F_k(\mathbf{x}) := (F(\mathbf{x}))_k$ for $k \in [K]$ and $\mathbf{x} \in \Omega$. For an input $\mathbf{x}_0 \in \mathbb{R}^{p_0}$, the targeted score of label k is $F_k(\mathbf{x}_0) = \mathbf{C}^{(k,:)}\mathbf{x}_L =: \mathbf{c}_k^T \mathbf{x}_L$, where $\mathbf{x}_i = \mathrm{ReLU}(\mathbf{A}_i\mathbf{x}_{i-1} + \mathbf{b}_i)$, for $i \in [L]$. By applying the chain rule on the non-smooth function F_k, we obtain a set valued map for F_k at point $\mathbf{x}_0$ as

$$G_{F_k}(\mathbf{x}_0) = \left\{ \left(\prod_{i=1}^{L} \mathbf{A}_i^T \,\mathrm{Diag}(\mathbf{z}_i) \right) \mathbf{c}_k : \mathbf{z}_i \in \partial\mathrm{ReLU}(\mathbf{A}_i\mathbf{x}_{i-1} + \mathbf{b}_i) \right\}, \tag{5.9}$$

where $\mathrm{Diag}(\mathbf{z}_i)$ is the diagonal matrix whose diagonal entries are composed of vector $\mathbf{z}_i$, and ∂ReLU is the subdifferential of ReLU function as defined in Definition 5.2.

Symbolically, the set valued map G_{F_k} looks like the subdifferential of function F_k since we apply the chain rule on F_k. However, in general, the chain rule cannot be applied to composition of non-smooth functions, see [Kakade & Lee 2018, Bolte & Pauwels 2021]. Hence the formulation of G_{F_k} may lead to incorrect subgradients and bounds on the Lipschitz constant of the networks. Nevertheless, we are going to prove that, in the spirit of Theorem 5.1, the map G_{F_k} can be used to compute a valid upper bound of the Lipschitz constant of function F_k.

Lemma 5.2. *Suppose $F : \Omega \to \mathbb{R}^K$ is a fully-connected deep neural network activated by ReLU function, $F_k(\mathbf{x}) := (F(\mathbf{x}))_k$ for $k \in [K]$ and $\mathbf{x} \in \Omega$. Let $L_{F_k,\|\cdot\|}$ be the Lipschitz constant of F_k w.r.t. norm $\|\cdot\|$. Then the optimal value of the following optimization problem*

$$\begin{aligned} \tilde{L}_{F_k,\|\cdot\|} &:= \sup_{\mathbf{x}_0 \in \Omega,\ \mathbf{v} \in G_{F_k}(\mathbf{x}_0)} \|\mathbf{v}\|_* \\ &= \sup_{\mathbf{x}_0 \in \Omega} \left\{ \left\| \left(\prod_{i=1}^{L} \mathbf{A}_i^T \,\mathrm{Diag}(\mathbf{z}_i) \right) \mathbf{c}_k \right\|_* : \mathbf{z}_i \in \partial\mathrm{ReLU}(\mathbf{A}_i\mathbf{x}_{i-1} + \mathbf{b}_i) \right\}, \end{aligned} \tag{5.10}$$

is a valid upper bound of $L_{F_k,\|\cdot\|}$, i.e., $L_{F_k,\|\cdot\|} \le \tilde{L}_{F_k,\|\cdot\|}$.

Proof : Adopting the terminology from [Bolte & Pauwels 2021], ∂ReLU is *conservative*. The formulation $G_{F_k}(\mathbf{x}_0) = (\prod_{i=1}^{L} \mathbf{A}_i^T \operatorname{Diag}(\mathbf{z}_i))\mathbf{c}_k$ is an application of the chain rule of differentiation, where along each chain the conservative set-valued field ∂ReLU is used in place of derivative of ReLU function. By [Bolte & Pauwels 2021, Lemma 2], chain rule preserves conservativity, hence G_{F_k} is a conservative mapping for function F_k. By conservativity and convexity of Ω, we have for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{p_0}$, integrating along the segment,

$$\begin{aligned}
|F_k(\mathbf{y}) - F_k(\mathbf{x})| &= \left| \int_0^1 \max_{\mathbf{v} \in G_{F_k}(\mathbf{x}+t(\mathbf{y}-\mathbf{x}))} \langle \mathbf{y}-\mathbf{x}, \mathbf{v} \rangle \, dt \right| \\
&\leq \int_0^1 \max_{\mathbf{v} \in G_{F_k}(\mathbf{x}+t(\mathbf{y}-\mathbf{x}))} \|\mathbf{y}-\mathbf{x}\| \|\mathbf{v}\|_* dt \\
&\leq \int_0^1 \|\mathbf{y}-\mathbf{x}\| \tilde{L}_{F_k,\|\cdot\|} dt \\
&= \tilde{L}_{F_k,\|\cdot\|} \|\mathbf{y}-\mathbf{x}\|,
\end{aligned}$$

which conclude the proof. □

Remark. Lemma 5.2 may admit a simpler proof. Indeed the Clarke subdifferential [Clarke 1983] is the convex hull of limits of sequences of gradients. For Lipschitz constant, we want the maximum norm element, which necessarily happens at a corner of the convex hull, therefore for our purposes it suffices to consider sequences. Since the ReLU network is almost everywhere differentiable, we can consider a shrinking sequence of balls around any point, and we will have gradients which are arbitrarily close to any corner of the gradients at our given point. Therefore, the norms of the sequence will converge to the norm of the corner, and thus it suffices to optimize over differentiable points, which means what we choose at the nondifferentiability does not matter.

The above argument is essentially valid because ReLU function only contains univariate nondifferentiability which is very specific. However the argument is implicitly based on the idea that the composition of almost everywhere differentiable functions complies with calculus rules, which is not correct in general due to lack of injectivity. For more general networks, what we choose at the nondifferentiability does matter. Indeed, consider the following functions

$$F\colon x \mapsto \begin{pmatrix} x \\ x \end{pmatrix}, \; G\colon \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \mapsto \max\{y_1, y_2\}.$$

The composition $G \circ F$ is the identity on $\mathbb{R}$ and both F and G are differentiable almost everywhere. Consider the mappings

$$J_F\colon x \mapsto \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \; J_G\colon \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \mapsto \begin{cases} (0,0)^T, & \text{if } y_1 = y_2; \\ (1,0)^T, & \text{if } y_1 > y_2; \\ (0,1)^T, & \text{if } y_1 < y_2. \end{cases}$$

It is easy to see that J_F is the Jacobian of F and J_G is the gradient of G almost everywhere. Now the product $J_G(F(x))^T \cdot J_F(x) = 0$ for all $x \in \mathbb{R}$. Hence computing the product of these gradient gives value 0, meaning that the function is constant, which is not the truth. The reason for the failure here is that J_G, despite being gradient almost everywhere, is not conservative for G. For this reason, the product does not provide any information of Lipschitz constant and the choice at nondifferentiability point does matter for G.

When $\Omega = \mathbb{R}^n$, $L_{F_k,\|\cdot\|}$ is called the *global* Lipschitz constant of F w.r.t. 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 $\bar{\mathbf{x}}$. In this situation, the input space Ω is often the ball around $\bar{\mathbf{x}} \in \mathbb{R}^{p_0}$ with radius ε: $\Omega = \{\mathbf{x} \in \mathbb{R}^{p_0} : \|\mathbf{x} - \bar{\mathbf{x}}\| \leq \varepsilon\} = B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)$, and we denote by $L_{F_k,\varepsilon,\|\cdot\|}$ the local Lipschitz constant. Obviously, the global Lipschitz constant is always an upper bound of the local Lipschitz constant, i.e., $L_{F_k,\varepsilon,\|\cdot\|} \leq L_{F_k,\|\cdot\|}$.

By Lemma 5.2 and semialgebraicity of ReLU, ∂ReLU described in 5.1, 5.3, the *Lipschitz constant estimation problem* for deep neural network F w.r.t. norm $\|\cdot\|$, is formulated as follows: for $k \in [K]$,

$$\max_{\mathbf{x}_i \in \mathbb{R}^{p_i},\, \mathbf{z}_i \in \mathbb{R}^{p_i},\, \mathbf{t} \in \mathbb{R}^{p_0}} \quad \mathbf{t}^T \Big(\prod_{i=1}^{L} \mathbf{A}_i^T \operatorname{Diag}(\mathbf{z}_i) \Big) \mathbf{c}_k \tag{LipDNN-k}$$

$$\text{s.t.} \quad \begin{cases} \mathbf{z}_i \circ (\mathbf{z}_i - 1) \leq 0,\ \mathbf{z}_i \circ (\mathbf{A}_i \mathbf{x}_{i-1} + \mathbf{b}_i) \geq 0, \\ (\mathbf{z}_i - 1) \circ (\mathbf{A}_i \mathbf{x}_{i-1} + \mathbf{b}_i) \geq 0,\ i \in [L], \\ \mathbf{x}_{i-1} \circ (\mathbf{x}_{i-1} - \mathbf{A}_{i-1}\mathbf{x}_{i-2} - \mathbf{b}_{i-1}) = 0, \\ \mathbf{x}_{i-1} - \mathbf{A}_{i-1}\mathbf{x}_{i-2} - \mathbf{b}_{i-1} \geq 0,\ \mathbf{x}_{i-1} \geq 0,\ 2 \leq i \leq L, \\ \mathbf{t}^2 \leq 1,\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \leq \varepsilon. \end{cases}$$

[Latorre *et al.* 2020] only use the constraint $0 \leq \mathbf{z}_i \leq 1$ on the variables $\mathbf{z}_i$, only capturing the Lipschitz character of the considered activation function. We could use the same constraints, which allows us to use activation functions that do not have semialgebraic representations such as ELU function. However, such a relaxation is much coarser than the one we propose. Indeed, (LipDNN-k) treats an *exact formulation* of the subdifferential of ReLU function by exploiting its semialgebraic character.

For MONs: Now we consider a different type of neural network, fully-connected MON, as introduced in Section 1.1.2. The only difference between a MON and a DNN is that we only have one hidden layer in MON, and the evaluation of the neurons in the hidden layer is realized by a fixed point equation. Similar with the case of DNNs, we assume the MON is a classifier activated by ReLU function. Let $\mathbf{W} = (\mathbf{A}, \mathbf{B}, \mathbf{b}; \mathbf{C})$ be the parameters of F, where $\mathbf{A} \in \mathbb{R}^{p_1 \times p_1}, \mathbf{B} \in \mathbb{R}^{p_1 \times p_0}, b \in \mathbb{R}^{p_1}, \mathbf{C} \in \mathbb{R}^{K \times p_1}$, p_0 is the number of neurons in the input layer, p_1 is the number of neurons in the hidden layer, K is the number of labels. We say that network F

has size (p_0, p_1) if F has p_0 neurons in the input layer and p_1 neurons in the hidden layer. We are going to bound the Lipschitz constant of F with respect to input perturbation. Let $L_{F,\|\cdot\|}$ (resp, $L_{F,\varepsilon,\|\cdot\|}$) be the global (resp. local) Lipschitz constant of F, and let $\mathbf{s}$ be any subgradient of the implicit variable, i.e., $\mathbf{s} \in \partial\mathrm{ReLU}(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b})$. By the semialgebraicity of ∂ReLU described in (5.3), we can write equivalently a system of polynomial inequalities for variable $\mathbf{s}$:

$$\mathbf{s}(\mathbf{s}-1) \leq 0,\ \mathbf{s}(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \geq 0,\ (\mathbf{s}-1)(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \geq 0. \tag{5.11}$$

Similar with the case of DNNs, we are able to calculate the upper bound of the Lipschitz constant of F by solving an optimization problem, as the following lemma states:

Lemma 5.3. *Suppose $F : \Omega \to \mathbb{R}^K$ is a fully-connected MON activated by ReLU function. Let $L_{F,\varepsilon,\|\cdot\|}$ be the local Lipschitz constant of F w.r.t. norm $\|\cdot\|$. Then the optimal value of the following problem*

$$\tilde{L}_{F,\varepsilon,\|\cdot\|} := \max_{\substack{\mathbf{t},\mathbf{x}_0 \in \mathbb{R}^{p_0},\ \mathbf{s},\mathbf{x}_1,\mathbf{h},\mathbf{r} \in \mathbb{R}^{p_1},\ \mathbf{v},\mathbf{w} \in \mathbb{R}^K}} \mathbf{t}^T\mathbf{B}^T\mathbf{h} \tag{LipMON}$$

$$s.t. \quad \begin{cases} \|\mathbf{t}\| \leq 1,\ \mathbf{w}^T\mathbf{v} \leq 1,\ \|\mathbf{w}\| \leq 1,\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \leq \varepsilon, \\ \mathbf{r} - \mathbf{A}^T\mathbf{h} = \mathbf{C}^T\mathbf{v},\ \mathbf{h} = \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{r}, \\ \mathbf{s} \circ (\mathbf{s}-1) \leq 0,\ \mathbf{s} \circ (\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \geq 0, \\ (\mathbf{s}-1) \circ (\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \geq 0, \\ \mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b}) = 0, \\ \mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b} \geq 0,\ \mathbf{x}_1 \geq 0, \end{cases}$$

is a valid upper bound of $L_{F,\varepsilon,\|\cdot\|}$, i.e., $L_{F,\varepsilon,\|\cdot\|} \leq \tilde{L}_{F,\varepsilon,\|\cdot\|}$.

In order to prove Lemma 5.3, we need some preliminaries.

Definition 5.2. (Clarke's generalized Jacobian [Clarke 1983]) Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a locally Lipschitz vector-valued application, denote by Ω_f any zero measure set such that f is differentiable outside Ω_f. For $\mathbf{x} \notin \Omega_f$, denote by $\mathcal{J}_f(\mathbf{x})$ the *Jacobian matrix* of f evaluated at $\mathbf{x}$. For any $\mathbf{x} \in \mathbb{R}^n$, the *generalized Jacobian*, or *Clarke Jacobian*, of f evaluated at $\mathbf{x}$, denoted by $\mathcal{J}_f^C(\mathbf{x})$, is defined as the convex hull of all $m \times n$ matrices obtained as the limit of a sequence of the form $\mathcal{J}_f(\mathbf{x}_i)$ with $\mathbf{x}_i \to \mathbf{x}$ and $\mathbf{x}_i \notin \Omega_f$. Symbolically, one has

$$\mathcal{J}_f^C(\mathbf{x}) := \mathrm{conv}\{\lim \mathcal{J}_f(\mathbf{x}_i) : \mathbf{x}_i \to \mathbf{x},\ i \to \infty,\ \mathbf{x}_i \notin \Omega_f\}. \tag{5.12}$$

Definition 5.3. For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the *operator norm* of $\mathbf{A}$ induced by norm $\|\cdot\|$, denoted by $|||\cdot|||$, is defined by

$$|||\mathbf{A}||| := \inf\{\lambda \geq 0 : \|\mathbf{A}\mathbf{x}\| \leq \lambda\|\mathbf{x}\|, \forall \mathbf{x} \in \mathbb{R}^n\}. \tag{5.13}$$

Lemma 5.4. *Let $F : \Omega \to \mathbb{R}^K$ be a fully-connected MON. The local Lipschitz constant of F is upper bounded by the supremum of the operator norm of its generalized Jacobian, i.e., define*

$$\bar{L}_{F,\varepsilon,\|\cdot\|} := \sup_{\substack{\mathbf{t},\mathbf{x}_0\in\mathbb{R}^{p_0},\ \mathbf{v},\mathbf{w}\in\mathbb{R}^K,\ \mathbf{J}\in\mathcal{J}^C_{\mathbf{x}_1}(\bar{\mathbf{x}})}} \mathbf{t}^T\mathbf{J}^T\mathbf{C}^T\mathbf{v} \qquad (5.14)$$
$$s.t. \quad \|\mathbf{t}\| \le 1,\ \mathbf{w}^T\mathbf{v} \le 1,\ \|\mathbf{w}\| \le 1,\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \le \varepsilon,$$

then $L_{F,\varepsilon,\|\cdot\|} \le \bar{L}_{F,\varepsilon,\|\cdot\|}$.

Proof. (of Lemma 5.4) Since $\mathbf{x}_1(\mathbf{x}_0) = \mathrm{ReLU}(\mathbf{A}\mathbf{x}_1(\mathbf{x}_0) + \mathbf{B}\mathbf{x}_0 + \mathbf{b})$ by definition of MON, $\mathbf{x}_1(\mathbf{x}_0)$ is Lipschitz according to [Pabbaraju *et al.* 2021, Theorem 1]. Furthermore, $\mathbf{x}_1(\mathbf{x}_0)$ is semialgebraic by the semialgebraicity of ReLU function described in (5.1). Therefore, the Clarke Jacobian of $\mathbf{x}_1$ is conservative. Indeed by [Clarke 1983, Proposition 2.6.2], the Clarke Jacobian is included in the product of subgradients of its coordinates which is a conservative field by [Bolte & Pauwels 2021, Lemma 3, Theorems 2 and 3]. Since $F(\mathbf{x}_0) = \mathbf{C}\mathbf{x}_1$, the mapping $\mathbf{C}\mathcal{J}^C_{\mathbf{x}_1} : \mathbf{x}_0 \rightrightarrows \mathbf{C}\mathbf{J}$, where $\mathbf{J} \in \mathcal{J}^C_{\mathbf{x}_1}$, is conservative for F by [Bolte & Pauwels 2021, Lemma 5]. So it satisfies an integration formula along segments. Let $\mathbf{u}_1, \mathbf{u}_2 \in \Omega$, and let $\gamma : [0,1] \to \mathbb{R}^{p_0}$ be a parametrization of the segment defined by $\gamma(t) = \mathbf{u}_1 + t(\mathbf{u}_2 - \mathbf{u}_1)$ (which is absolutely continuous). For almost all $t \in [0,1]$, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}F(\gamma(t)) = \mathbf{C}\mathbf{J}\gamma'(t) = \mathbf{C}\mathbf{J}(\mathbf{u}_2 - \mathbf{u}_1), \qquad (5.15)$$

for all $\mathbf{J} \in \mathcal{J}^C_{\mathbf{x}_1}(\gamma(t))$.

Let $M := \sup\{|||\mathbf{C}\mathbf{J}||| : \mathbf{J} \in \mathcal{J}^C_{\mathbf{x}_1}(\mathbf{x}_0), \mathbf{x}_0 \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)\}$ be the supremum of the operator norm $|||\mathbf{C}\mathbf{J}|||$ for all $\mathbf{J} \in \mathcal{J}^C_{\mathbf{x}_1}(\mathbf{x}_0)$ and all $\mathbf{x}_0 \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)$. We prove that $M < \infty$. Indeed, $\mathbf{x}_1(\mathbf{x}_0)$ is Lipschitz, hence there exists $N > 0$ such that $|||\mathbf{J}||| < N$ for all $\mathbf{J} \in \mathcal{J}^C_{\mathbf{x}_1}(\mathbf{x}_0)$ and all $\mathbf{x}_0 \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)$. The value M is thus upper bounded by $|||\mathbf{C}|||N$. Therefore, for almost all $t \in [0,1]$, $\|\frac{\mathrm{d}}{\mathrm{d}t}F(\gamma(t))\| \le M\|\mathbf{u}_2 - \mathbf{u}_1\|$, and by integration,

$$\|F(\mathbf{u}_2) - F(\mathbf{u}_1)\| = \left\| \int_0^1 \frac{\mathrm{d}}{\mathrm{d}t}F(\gamma(t))\mathrm{d}t \right\| \le \int_0^1 \left\| \frac{\mathrm{d}}{\mathrm{d}t}F(\gamma(t)) \right\| \mathrm{d}t \le M\|\mathbf{u}_2 - \mathbf{u}_1\|,$$

which proves that $L_{F,\varepsilon,\|\cdot\|} \le M$. We show that $M = \bar{L}_{F,\varepsilon,\|\cdot\|}$. Fix $\mathbf{x}_0 \in \mathbb{R}^{p_0}$ and $\mathbf{J} \in \mathcal{J}^C_{\mathbf{x}_1}(\mathbf{x}_0)$. By the definition of operator norm,

$$\begin{aligned} |||\mathbf{C}\mathbf{J}||| &= \left|\left|\left|(\mathbf{C}\mathbf{J})^T\right|\right|\right|^* = \sup_{\mathbf{v}\in\mathbb{R}^K} \{\|\mathbf{J}^T\mathbf{C}^T\mathbf{v}\|^* : \|\mathbf{v}\|^* \le 1\} \\ &= \sup_{\mathbf{t}\in\mathbb{R}^{p_0},\ \mathbf{v}\in\mathbb{R}^K} \{\mathbf{t}^T\mathbf{J}^T\mathbf{C}^T\mathbf{v} : \|\mathbf{t}\| \le 1,\ \|\mathbf{v}\|^* \le 1\} \\ &= \sup_{\mathbf{t}\in\mathbb{R}^{p_0},\ \mathbf{v},\mathbf{w}\in\mathbb{R}^K} \{\mathbf{t}^T\mathbf{J}^T\mathbf{C}^T\mathbf{v} : \|\mathbf{t}\| \le 1,\ \mathbf{w}^T\mathbf{v} \le 1,\ \|\mathbf{w}\| \le 1\}, \end{aligned} \qquad (5.16)$$

where $\|\cdot\|^*$ denotes the dual norm of $\|\cdot\|$ defined by $\|\mathbf{v}\|^* := \sup_{\mathbf{w}\in\mathbb{R}^K}\{\mathbf{w}^T\mathbf{v} : \|\mathbf{w}\| \le$

1} for all $\mathbf{v} \in \mathbb{R}^K$, and the first equality is due to the fact that the operator norm of matrix $\mathbf{CJ}$ induced by norm $\|\cdot\|$ is equal to the operator norm of its transpose $(\mathbf{CJ})^T$ induced by the dual norm $\|\cdot\|^*$. Indeed, by definition of operator norm and dual norm, we have

$$\begin{aligned}
|||\mathbf{CJ}||| &= \sup_{\mathbf{x}_0 \in \mathbb{R}^{p_0}} \{\|\mathbf{CJ}\mathbf{x}_0\| : \|\mathbf{x}_0\| \leq 1\} \\
&= \sup_{\mathbf{x}_0 \in \mathbb{R}^{p_0},\, \mathbf{x}_1 \in \mathbb{R}^{p_1}} \{\mathbf{x}_1^T \mathbf{CJ}\mathbf{x}_0 : \|\mathbf{x}_0\| \leq 1,\ \|\mathbf{x}_1\|^* \leq 1\} \\
&= \sup_{\mathbf{x}_0 \in \mathbb{R}^{p_0},\, \mathbf{x}_1 \in \mathbb{R}^{p_1}} \{\mathbf{x}_0^T (\mathbf{CJ})^T \mathbf{x}_1 : \|\mathbf{x}_0\| \leq 1,\ \|\mathbf{x}_1\|^* \leq 1\} \\
&= \sup_{\mathbf{x}_1 \in \mathbb{R}^{p_1}} \{\|(\mathbf{CJ})^T \mathbf{x}_1\|^* : \|\mathbf{x}_1\|^* \leq 1\} = \left|\left|\left|(\mathbf{CJ})^T\right|\right|\right|^*. \qquad (5.17)
\end{aligned}$$

The quantity $\bar{L}_{F,\varepsilon,\|\cdot\|}$ is just the maximization of Equation (5.16) for all $\mathbf{x}_0 \in \mathbb{R}^{p_0}$ and all $\mathbf{J} \in \mathcal{J}^C_{\mathbf{x}_1}(\mathbf{x}_0)$ and therefore equals M, which concludes that $L_{F,\varepsilon,\|\cdot\|} \leq M = \bar{L}_{F,\varepsilon,\|\cdot\|}$. □

Now we have enough materials to prove Lemma 5.3.

Proof. (of Lemma 5.3) The function $\mathbf{x}_1$ is semialgebraic, and therefore, there exists a closed zero measure set $\Omega_{\mathbf{x}_1}$ such that $\mathbf{x}_1$ is continuously differentiable on the complement of $\Omega_{\mathbf{x}_1}$. For any $\mathbf{x}_0 \notin \Omega_{\mathbf{x}_1}$, since $\mathbf{x}_1$ is C^1 at $\mathbf{x}_0$, we have $\mathcal{J}^C_{\mathbf{x}_1}(\mathbf{x}_0) = \{\mathcal{J}_{\mathbf{x}_1}(\mathbf{x}_0)\}$ by definition of the Clarke Jacobian. Fix $\mathbf{x}_0 \notin \Omega_{\mathbf{x}_1}$ arbitrarily. According to [Clarke 1983, page 75, Corollary of Theorem 2.6.6], we have

$$\begin{aligned}
\mathcal{J}^C_{\mathbf{x}_1}(\mathbf{x}_0) &\subseteq \operatorname{conv}\{\mathcal{J}^C_{\mathrm{ReLU}}(\mathbf{A}\mathbf{x}_1(\mathbf{x}_0) + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \cdot \mathcal{J}^C_{\mathbf{A}\mathbf{x}_1(\mathbf{x}_0)+\mathbf{B}\mathbf{x}_0+\mathbf{b}}(\mathbf{x}_0)\} \\
&= \operatorname{conv}\{\mathcal{J}^C_{\mathrm{ReLU}}(\mathbf{A}\mathbf{x}_1(\mathbf{x}_0) + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \cdot (\mathbf{A} \cdot \mathcal{J}_{\mathbf{x}_1}(\mathbf{x}_0) + \mathbf{B})\} \\
&= \mathcal{J}^C_{\mathrm{ReLU}}(\mathbf{A}\mathbf{x}_1(\mathbf{x}_0) + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \cdot (\mathbf{A} \cdot \mathcal{J}_{\mathbf{x}_1}(\mathbf{x}_0) + \mathbf{B}), \qquad (5.18)
\end{aligned}$$

where the first inclusion is from the cited Corollary, the first equality is because $\mathbf{x}_1$ is C^1 at $\mathbf{x}_0$ so that the chain rule applies, and the last one is due to the convexity of the Clarke Jacobian.

Fix any $\mathbf{x}_0 \in \mathbb{R}^{p_0}$, then by Definition 5.2, we have

$$\mathcal{J}^C_{\mathbf{x}_1}(\bar{\mathbf{x}}) = \operatorname{conv}\{\lim \mathcal{J}_{\mathbf{x}_1}(\mathbf{u}_i) : \mathbf{u}_i \to \mathbf{x}_0,\ i \to +\infty,\ \mathbf{u}_i \notin \Omega_{\mathbf{x}_1}\}. \qquad (5.19)$$

Let $\{\mathbf{u}_i\}_{i\in\mathbb{N}}$ be a sequence not in $\Omega_{\mathbf{x}_1}$ and converging to $\bar{\mathbf{x}}$. For each $\mathbf{u}_i \notin \Omega_{\mathbf{x}_1}$, we have by (5.18) that $\mathcal{J}_{\mathbf{x}_1}(\mathbf{u}_i) \in \mathcal{J}^C_{\mathrm{ReLU}}(\mathbf{A}\mathbf{x}_1(\mathbf{u}_i)+\mathbf{B}\mathbf{u}_i+\mathbf{b})\cdot(\mathbf{A}\cdot\mathcal{J}_{\mathbf{x}_1}(\mathbf{u}_i)+\mathbf{B})$, i.e., there exists $\mathbf{H}_i \in \mathcal{J}^C_{\mathrm{ReLU}}(\mathbf{A}\mathbf{x}_1(\mathbf{u}_i) + \mathbf{B}\mathbf{u}_i + \mathbf{b})$ such that $\mathcal{J}_{\mathbf{x}_1}(\mathbf{u}_i) = \mathbf{H}_i(\mathbf{A} \cdot \mathcal{J}_{\mathbf{x}_1}(\mathbf{u}_i) + \mathbf{B})$. By [Clarke 1983, proposition 2.6.2 (b)], $\mathcal{J}^C_{\mathrm{ReLU}}$ has closed graph. Therefore, by continuity of $\mathbf{x}_1$, up to a subsequence, $\mathbf{H}_i \to \mathbf{H} \in \mathcal{J}^C_{\mathrm{ReLU}}(\mathbf{A}\mathbf{x}_1(\mathbf{x}_0) + \mathbf{B}\mathbf{x}_0 + \mathbf{b})$ for $i \to +\infty$, which means

$$\mathcal{J}^C_{\mathbf{x}_1}(\mathbf{x}_0) \subseteq \{\mathbf{J} : \mathbf{H} \in \mathcal{J}^C_{\mathrm{ReLU}}(\mathbf{A}\mathbf{x}_1(\mathbf{x}_0) + \mathbf{B}\mathbf{x}_0 + \mathbf{b}),\ \mathbf{J} = \mathbf{H}(\mathbf{A}\mathbf{J} + \mathbf{B})\}, \qquad (5.20)$$

for all $\mathbf{x}_0 \in \mathbb{R}^{p_0}$. Let $\mathbf{H} \in \mathcal{J}^C_{\mathrm{ReLU}}(\mathbf{A}\mathbf{x}_1(\mathbf{x}_0)+\mathbf{B}\mathbf{x}_0+\mathbf{b})$, since we have coordinate-wise application of ReLU, we obtain $\mathbf{H} = \mathrm{Diag}(\mathbf{s})$ with $\mathbf{s} \in \partial\mathrm{ReLU}(\mathbf{A}\mathbf{x}_1(\mathbf{x}_0)+\mathbf{B}\mathbf{x}_0+\mathbf{b})$. By equation (5.20), the right-hand side of equation (5.14) is upper bounded by

$$\max_{\mathbf{t},\mathbf{x}_0\in\mathbb{R}^{p_0},\, \mathbf{s},\mathbf{x}_1\in\mathbb{R}^{p_1},\, \mathbf{v},\mathbf{w}\in\mathbb{R}^K,\, \mathbf{J}\in\mathbb{R}^{p_1\times p_0}} \quad \mathbf{t}^T\mathbf{J}^T\mathbf{C}^T\mathbf{v} \tag{LipMON-a}$$

$$\text{s.t.} \quad \begin{cases} \|\mathbf{t}\| \le 1,\ \mathbf{w}^T\mathbf{v} \le 1,\ \|\mathbf{w}\| \le 1,\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \le \varepsilon, \\ \mathbf{J} = \mathrm{Diag}(\mathbf{s}) \cdot (\mathbf{A} \cdot \mathbf{J} + \mathbf{B}), \\ \mathbf{s} \circ (\mathbf{s} - 1) \le 0,\ \mathbf{s} \circ (\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \ge 0, \\ (\mathbf{s} - 1) \circ (\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \ge 0, \\ \mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b}) = 0, \\ \mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b} \ge 0,\ \mathbf{x}_1 \ge 0. \end{cases}$$

Note that in problem (LipMON-a), we have a matrix variable $\mathbf{J}$ of size $p_1 \times p_0$, i.e., containing $p_1 \times p_0$ many variables, which is too large for any SDP solvers. To make the optimization tractable, we use the *vector-matrix product* trick proposed by [Winston & Kolter 2020] to reduce the size of the unknown variables. From equation $\mathbf{J} = \mathrm{Diag}(\mathbf{s}) \cdot (\mathbf{A} \cdot \mathbf{J} + \mathbf{B})$, we have $\mathbf{J} = (\mathbf{I}_{p_1} - \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{A})^{-1} \cdot \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{B}$. This inversion makes sense because of the strong monotonicity of $\mathbf{I}_{p_1} - \mathbf{A}$ and the fact that all entries of $\mathbf{s}$ lie in $[0, 1]$, see [Winston & Kolter 2020, Proposition 1]. Hence

$$\mathbf{v}^T\mathbf{C}\mathbf{J} = \mathbf{v}^T\mathbf{C} \cdot (\mathbf{I}_{p_1} - \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{A})^{-1} \cdot \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{B} = \mathbf{r}^T \cdot \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{B}\,, \tag{5.21}$$

where $\mathbf{r}^T = \mathbf{v}^T\mathbf{C} \cdot (\mathbf{I}_{p_1} - \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{A})^{-1}$, which means $\mathbf{r} - \mathbf{A}^T \cdot \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{r} = \mathbf{C}^T\mathbf{v}$. Set $\mathbf{h} = \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{r}$ and transpose both sides of equation (5.21), we have $\mathbf{J}^T\mathbf{C}^T\mathbf{v} = \mathbf{B}^T\mathbf{h}$ with $\mathbf{r} - \mathbf{A}^T \cdot \mathbf{h} = \mathbf{C}^T\mathbf{v}$. We can then rewrite the objective function of (LipMON-a) as $\mathbf{t}^T\mathbf{B}^T\mathbf{h}$, leading to the following equivalent problem

$$\max_{\mathbf{t},\mathbf{x}_0\in\mathbb{R}^{p_0},\, \mathbf{s},\mathbf{x}_1,\mathbf{h},\mathbf{r}\in\mathbb{R}^{p_1},\, \mathbf{v},\mathbf{w}\in\mathbb{R}^K} \quad \mathbf{t}^T\mathbf{A}^T\mathbf{h} \tag{LipMON-b}$$

$$\text{s.t.} \quad \begin{cases} \|\mathbf{t}\| \le 1,\ \mathbf{w}^T\mathbf{v} \le 1,\ \|\mathbf{w}\| \le 1,\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \le \varepsilon, \\ \mathbf{r} - \mathbf{A}^T\mathbf{h} = \mathbf{C}^T\mathbf{v},\ \mathbf{h} = \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{r}, \\ \mathbf{s} \circ (\mathbf{s} - 1) \le 0,\ \mathbf{s} \circ (\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \ge 0, \\ (\mathbf{s} - 1) \circ (\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}) \ge 0, \\ \mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b}) = 0, \\ \mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b} \ge 0,\ \mathbf{x}_1 \ge 0. \end{cases}$$

We have shown that (LipMON-b) is the right hand side of Equation (LipMON) in Lemma 5.3 and is an upper bound of the right hand side of Equation (5.14) in Lemma 5.4, i.e., $\bar{L}_{F,\varepsilon,\|\cdot\|} \le \tilde{L}_{F,\varepsilon,\|\cdot\|}$. □

Remark. In order to avoid some possible numerical issues, we add some redundant

constraints to problem (LipMON), we use the *slope restriction* condition of ReLU function as proposed by [Hu *et al.* 2020]: for $i \neq j$,

$$(x_1^{(j)} - x_1^{(i)})(\mathbf{A}^{(j,:)}\mathbf{x}_1 + \mathbf{B}^{(j,:)}\mathbf{x}_0 + b^{(j)} - \mathbf{A}^{(i,:)}\mathbf{x}_1 - \mathbf{B}^{(i,:)}\mathbf{x}_0 - b^{(i)}) - (x_1^{(j)} - x_1^{(i)})^2 \geq 0. \tag{5.22}$$

There are $\binom{p_1}{2} = O(p_1^2)$ many constraints in (5.22), which will be very large if p_1 increases. In practice, we choose those constraints with a fixed index $i \in [p_1]$ so that we reduce the number of redundant constraints to $p_1 - 1$.

Summarize the discussion above, we show the total number of variables and constraints for the Lipschitz constant estimation problem of DNNs and MONs in Table 5.1.

Table 5.1: Summary of Lipschitz constant estimation problem for DNNs and MONs.

Network	Norm	# of variables	# of constraints
DNN	$\frac{L_2}{L_\infty}$	$2p_0 + 2\sum_{i=1}^{L-1} p_i + p_L$	$\frac{1 + p_0 + 3\sum_{i=1}^{L} p_i + 3\sum_{i=2}^{L} p_i}{2p_0 + 3\sum_{i=1}^{L} p_i + 3\sum_{i=2}^{L} p_i}$
MON	$\frac{L_2}{L_\infty}$	$2p_0 + 4p_1 + 2K$	$\frac{4 + 8p_1}{1 + 2p_0 + 8p_1 + K}$

5.2.2 Algorithms

Designing a general algorithm for all kind of neural networks is difficult. Here we consider specifically three types of neural networks: fully-connected DNNs with 1 and 2 hidden layers (denoted by $\mathbf{FcDNN}_1$ and $\mathbf{FcDNN}_2$), and fully-connected MONs (denoted by **FcMON**). We are going to discuss about the convex relaxation method for each type of network respectively. In the following discussion, we write $x^{(i)}$ the i-th entry of vector $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{A}^{(i,:)}$ the i-th row vector of matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$.

For $\mathbf{FcDNN}_1$: Suppose the network F has only one hidden layer. In this case, the Lipschitz constant estimation problem (LipDNN-k) reduces as

$$\max_{\mathbf{t},\mathbf{x}_0\in\mathbb{R}^{p_0},\, \mathbf{z}_1\in\mathbb{R}^{p_1}} \quad \mathbf{t}^T\mathbf{A}_1^T \operatorname{Diag}(\mathbf{z}_1)\mathbf{c}_k \tag{5.23}$$

$$\text{s.t.} \quad \begin{cases} \mathbf{z}_1 \circ (\mathbf{z}_1 - 1) \leq 0,\ \mathbf{z}_1 \circ (\mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1) \geq 0,\ (\mathbf{z}_1 - 1) \circ (\mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1) \geq 0, \\ \mathbf{t}^2 \leq 1,\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \leq \varepsilon. \end{cases}$$

In order to produce more sparsity patterns, we introduce a new decision variable $\mathbf{u}_1 = \mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1$ and replace it in problem (5.23). Then we obtain an equivalent

form of (5.23):

$$\max_{\mathbf{t},\mathbf{x}_0\in\mathbb{R}^{p_0},\,\mathbf{z}_1,\mathbf{u}_1\in\mathbb{R}^{p_1}} \quad \mathbf{t}^T\mathbf{A}_1^T\,\mathrm{Diag}(\mathbf{z}_1)\mathbf{c}_k \qquad \text{(LipFcDNN}_1\text{-}k\text{)}$$
$$\text{s.t.} \quad \begin{cases} \mathbf{u}_1-\mathbf{A}_1\mathbf{x}_0-\mathbf{b}_1=0, \\ \mathbf{z}_1\circ(\mathbf{z}_1-1)\le 0,\ \mathbf{z}_1\circ\mathbf{u}_1\ge 0,\ (\mathbf{z}_1-1)\circ\mathbf{u}_1\ge 0; \\ \mathbf{t}^2\le 1,\ \|\mathbf{x}_0-\bar{\mathbf{x}}\|\le\varepsilon. \end{cases}$$

By semialgebraicity of L_p norm, problem (LipFcDNN$_1$-k) is a POP for $p\in\mathbb{N}_+\cup\{\infty\}$. In particular, for L_2 and L_∞ norm, it is a nonconvex QCQP. Therefore, we can apply Lasserre's relaxation and sublevel relaxation to compute upper bounds of the optimal value of (LipFcDNN$_1$-k). We first apply Shor's relaxation (a.k.a. first-order Lasserre's relaxation) without exploiting sparsity, then design a sublevel relaxation which is benefited from the sparsity of (LipFcDNN$_1$-k) and enhance Shor's relaxations. Shor's relaxation applied to (LipFcDNN$_1$-k) is the following SDP:

$$\inf_{\mathbf{y}} \quad L_{\mathbf{y}}(\mathbf{t}^T\mathbf{A}_1^T\,\mathrm{Diag}(\mathbf{z}_1)\mathbf{c}_k) \qquad \text{(SHOR-LipFcDNN}_1\text{-}k\text{)}$$
$$\text{s.t.} \quad \begin{cases} L_{\mathbf{y}}(1)=1,\ \mathbf{M}_1(\mathbf{y})\succeq 0, \\ L_{\mathbf{y}}(u_1^{(i)}-\mathbf{A}_1^{(i,:)}\mathbf{x}_0+b_1^{(i)})=0,\ i\in[p_1], \\ L_{\mathbf{y}}(z_1^{(i)}(z_1^{(i)}-1))\le 0,\ L_{\mathbf{y}}(z_1^{(i)}u_1^{(i)})\ge 0,\ L_{\mathbf{y}}((z_1^{(i)}-1)u_1^{(i)})\ge 0,\ i\in[p_1], \\ L_{\mathbf{y}}(1-(t^{(i)})^2)\ge 0,\ i\in[p_0], \\ L_{\mathbf{y}}(\varepsilon^2-(\mathbf{x}_0-\bar{\mathbf{x}})^T(\mathbf{x}_0-\bar{\mathbf{x}}))\ge 0, \quad \text{(for } L_2 \text{ norm)} \\ L_{\mathbf{y}}(\varepsilon^2-(x_0^{(i)}-\bar{x}^{(i)})^2)\ge 0,\ i\in[p_0]. \quad \text{(for } L_\infty \text{ norm)} \end{cases}$$

Based on Shor's relaxation, we are going to enhance the relaxation by adding second-order moment matrices and first-order localizing matrices, which results in a sublevel relaxation. Define subsets $I^{(i)}=\{t^{(i)},x_0^{(i)}\}$ for $i\in[p_0]$ w.r.t. variables $\mathbf{t},\mathbf{x}_0$, and $J^{(j)}=\{z_1^{(j)},u_1^{(j)}\}$ for $j\in[p_1]$ w.r.t. variables $\mathbf{z}_1,\mathbf{u}_1$. Let Γ_{dense} be the index set of dense constraints $u_1^{(i)}-\mathbf{A}_1^{(i,:)}\mathbf{x}_0+b_1^{(i)}$ (and $\varepsilon^2-(\mathbf{x}_0-\bar{\mathbf{x}})^T(\mathbf{x}_0-\bar{\mathbf{x}})$ for L_∞ norm), let Γ_{sparse} be the index set of the remaining sparse constraints. The idea is to endow the dense constraints with 0-th order localizing matrices (i.e., Riesz linear functionals), and the sparse constraints with first-order localizing matrices w.r.t. subsets $I^{(i)}$ and $J^{(j)}$. Using the terminology described in Definition 4.3, let $\mathbf{l}=\{\mathbf{0}_{|\Gamma_{dense}|},\mathbf{2}_{|\Gamma_{sparse}|}\}$ and $\mathbf{q}=\mathbf{1}_{|\Gamma_{dense}|+|\Gamma_{sparse}|}$. For subsets I_i and J_j, define the

second-order $(\mathbf{l}, \mathbf{q})$-sublevel relaxation as follows:

$$\inf_{\mathbf{y}} \quad L_{\mathbf{y}}(\mathbf{t}^T \mathbf{A}_1^T \operatorname{Diag}(\mathbf{u}_1)\mathbf{c}_k) \qquad (\text{Sub}_2\text{-LipFcDNN}_1\text{-}k)$$
$$\text{s.t.} \quad \begin{cases} L_{\mathbf{y}}(1) = 1,\ \mathbf{M}_1(\mathbf{y}) \succeq 0, \\ \mathbf{M}_2(\mathbf{y}, I^{(i)}) \succeq 0,\ i \in [p_0],\ \mathbf{M}_2(\mathbf{y}, J^{(j)}) \succeq 0,\ j \in [p_1], \\ L_{\mathbf{y}}(u_1^{(i)} - \mathbf{A}_1^{(i,:)}\mathbf{x}_0 + b_1^{(i)}) = 0,\ i \in [p_1], \\ \mathbf{M}_1(z_1^{(i)}(z_1^{(i)} - 1), J^{(j)}) \le 0,\ \mathbf{M}_1(z_1^{(i)}u_1^{(i)}, J^{(j)}) \ge 0, \\ \qquad\qquad \mathbf{M}_1((z_1^{(i)} - 1)u_1^{(i)}, J^{(j)}) \ge 0,\ j \in [p_1], \\ \mathbf{M}_1(1 - (t^{(i)})^2, I^{(i)}) \ge 0,\ i \in [p_0], \\ L_{\mathbf{y}}(\varepsilon^2 - (\mathbf{x}_0 - \bar{\mathbf{x}})^T(\mathbf{x}_0 - \bar{\mathbf{x}})) \ge 0, \quad (\text{for } L_2 \text{ norm}) \\ \mathbf{M}_1(\varepsilon^2 - (x_0^{(i)} - \bar{x}^{(i)})^2, I^{(i)}) \ge 0,\ i \in [p_0]. \quad (\text{for } L_\infty \text{ norm}) \end{cases}$$

For FcDNN_2: Suppose the network F has two hidden layers. Now problem (LipDNN-k) reads:

$$\max_{\mathbf{t} \in \mathbb{R}^{p_0},\, \mathbf{x}_i \in \mathbb{R}^{p_i},\, \mathbf{z}_i \in \mathbb{R}^{p_i}} \quad \mathbf{t}^T \mathbf{A}_1^T \operatorname{Diag}(\mathbf{z}_1)\mathbf{A}_2^T \operatorname{Diag}(\mathbf{z}_2)\mathbf{c}_k \qquad (5.24)$$
$$\text{s.t.} \quad \begin{cases} \mathbf{z}_1 \circ (\mathbf{z}_1 - 1) \le 0,\ \mathbf{z}_1 \circ (\mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1) \ge 0,\ (\mathbf{z}_1 - 1) \circ (\mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1) \ge 0, \\ \mathbf{z}_2 \circ (\mathbf{z}_2 - 1) \le 0,\ \mathbf{z}_2 \circ (\mathbf{A}_2\mathbf{x}_1 + \mathbf{b}_2) \ge 0,\ (\mathbf{z}_2 - 1) \circ (\mathbf{A}_2\mathbf{x}_1 + \mathbf{b}_2) \ge 0, \\ \mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{A}_1\mathbf{x}_0 - \mathbf{b}_1) = 0,\ \mathbf{x}_1 - \mathbf{A}_1\mathbf{x}_0 - \mathbf{b}_1 \ge 0,\ \mathbf{x}_1 \ge 0, \\ \mathbf{t}^2 \le 1,\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \le \varepsilon. \end{cases}$$

Similarly, let $\mathbf{u}_i = \mathbf{A}_i\mathbf{x}_{i-1} + \mathbf{b}_i$ for $i = 1, 2$. We obtain the Lipschitz constant estimation problem for DNNs with two hidden layers:

$$\max_{\mathbf{t} \in \mathbb{R}^{p_0},\, \mathbf{x}_i \in \mathbb{R}^{p_i},\, \mathbf{z}_i \in \mathbb{R}^{p_i}} \quad \mathbf{t}^T \mathbf{A}_1^T \operatorname{Diag}(\mathbf{z}_1)\mathbf{A}_2^T \operatorname{Diag}(\mathbf{z}_2)\mathbf{c}_k \qquad (\text{LipFcDNN}_2\text{-}k)$$
$$\text{s.t.} \quad \begin{cases} \mathbf{u}_1 - \mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1 = 0,\ \mathbf{u}_2 - \mathbf{A}_2\mathbf{x}_1 + \mathbf{b}_2 = 0, \\ \mathbf{z}_1 \circ (\mathbf{z}_1 - 1) \le 0,\ \mathbf{z}_1 \circ \mathbf{u}_1 \ge 0,\ (\mathbf{z}_1 - 1) \circ \mathbf{u}_1 \ge 0, \\ \mathbf{z}_2 \circ (\mathbf{z}_2 - 1) \le 0,\ \mathbf{z}_2 \circ \mathbf{u}_2 \ge 0,\ (\mathbf{z}_2 - 1) \circ \mathbf{u}_2 \ge 0, \\ \mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{u}_1) = 0,\ \mathbf{x}_1 - \mathbf{u}_1 \ge 0,\ \mathbf{x}_1 \ge 0, \\ \mathbf{t}^2 \le 1,\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \le \varepsilon. \end{cases}$$

Different from the case of one hidden layer, problem (LipFcDNN_2-k) is no longer a nonconvex QCQP, since the objective in (LipFcDNN_2-k) is a degree-3 polynomial in variable $\mathbf{t}, \mathbf{z}_1, \mathbf{z}_2$. Hence Shor's relaxation is not effective any more. The smallest order of Lasserre's relaxation that can be applied to (LipFcDNN_2-k) is 2, which means we need to encode second-order moment matrices. When the number of neurons in F is large (eg. $p_1 = p_2 = 100$), it is untractable for any current SDP solvers. Fortunately, we may reduce the size of moment matrices by choosing only some submatrices of the full second-order moment matrices. Precisely, instead of

considering the dense moment matrix $\mathbf{M}_2(\mathbf{y})$, we are going to select those submatrices of $\mathbf{M}_2(\mathbf{y})$ that involve variables $t^{(i)}, u_1^{(j)}, u_2^{(k)}$ appearing in the objective of (LipFcDNN$_2$-k). Define, for $i \in [p_0], j \in [p_1], k \in [p_2]$,

$$\mathbf{M}_2^{sub}(\mathbf{y}, \{t^{(i)}, u_1^{(j)}, u_2^{(k)}\}) = L_{\mathbf{y}}\left(\begin{pmatrix} 1 & t^{(i)} & u_1^{(j)}u_2^{(k)} \\ t^{(i)} & (t^{(i)})^2 & t^{(i)}u_1^{(j)}u_2^{(k)} \\ u_1^{(j)}u_2^{(k)} & t^{(i)}u_1^{(j)}u_2^{(k)} & (u_1^{(j)}u_2^{(k)})^2 \end{pmatrix}\right). \tag{5.25}$$

If the full moment matrix $\mathbf{M}_2(\mathbf{y})$ is positive semidefinite, then the submatrices $\mathbf{M}_2^{sub}(\mathbf{y}, \{t^{(i)}, u_1^{(j)}, u_2^{(k)}\})$ are also positive semidefinite for all $i \in [p_0], j \in [p_1], k \in [p_2]$. Thus, the constraints $\mathbf{M}_2^{sub}(\mathbf{y}, \{t^{(i)}, u_1^{(j)}, u_2^{(k)}\}) \succeq 0$ are necessary conditions for $\mathbf{M}_2(\mathbf{y})$ being positive semidefinite. Replacing the full moment matrix $\mathbf{M}_2(\mathbf{y})$ by its submatrices $\mathbf{M}_2^{sub}(\mathbf{y}, \{t^{(i)}, u_1^{(j)}, u_2^{(k)}\})$ and applying Riesz functional to all the constraints, we obtain a relaxation between Shor's relaxation and second-order Lasserre's relaxation:

$$\sup_{\mathbf{y}} \quad L_{\mathbf{y}}(\mathbf{t}^T \mathbf{A}_1^T \operatorname{Diag}(\mathbf{z}_1)\mathbf{A}_2^T \operatorname{Diag}(\mathbf{z}_2)\mathbf{c}) \tag{Sub$_1$-LipFcDNN$_2$-k}$$

$$\text{s.t.} \quad \begin{cases} L_{\mathbf{y}}(1) = 1,\ \mathbf{M}_1(\mathbf{y}) \succeq 0, \\ \mathbf{M}_2^{sub}(\mathbf{y}, \{t^{(i)}, u_1^{(j)}, u_2^{(k)}\}) \succeq 0,\ i \in [p_0],\ j \in [p_1],\ k \in [p_2], \\ L_{\mathbf{y}}(\mathbf{u}_1 - \mathbf{A}_1\mathbf{x}_0 - \mathbf{b}_1) = 0,\ L_{\mathbf{y}}(\mathbf{u}_2 - \mathbf{A}_2\mathbf{x}_1 - \mathbf{b}_2) = 0, \\ L_{\mathbf{y}}(\mathbf{z}_1 \circ (\mathbf{z}_1 - 1)) = 0,\ L_{\mathbf{y}}(\mathbf{z}_1 \circ \mathbf{u}_1) \geq 0,\ L_{\mathbf{y}}((\mathbf{z}_1 - 1) \circ \mathbf{u}_1) \geq 0, \\ L_{\mathbf{y}}(\mathbf{z}_2 \circ (\mathbf{z}_2 - 1)) = 0,\ L_{\mathbf{y}}(\mathbf{z}_2 \circ \mathbf{u}_2) \geq 0,\ L_{\mathbf{y}}((\mathbf{z}_2 - 1) \circ \mathbf{u}_2) \geq 0, \\ L_{\mathbf{y}}(\mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{u}_1)) = 0,\ L_{\mathbf{y}}(\mathbf{x}_1 - \mathbf{u}_1) \geq 0,\ L_{\mathbf{y}}(\mathbf{x}_1) \geq 0, \\ L_{\mathbf{y}}(1 - \mathbf{t}^2) \geq 0, \\ L_{\mathbf{y}}(\varepsilon^2 - (\mathbf{x}_0 - \bar{\mathbf{x}})^T(\mathbf{x}_0 - \bar{\mathbf{x}})) \geq 0, \quad \text{(for } L_2 \text{ norm)} \\ L_{\mathbf{y}}(\varepsilon^2 - (x_0^{(i)} - \bar{x}^{(i)})^2, I^{(i)}) \geq 0,\ i \in [p_0]. \quad \text{(for } L_\infty \text{ norm)} \end{cases}$$

Moreover, we can strengthen relaxation (Sub$_1$-LipFcDNN$_2$-k) by adding sublevel structures. Let Γ_{dense} be the index set of dense constraints $\mathbf{u}_1 - \mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1 = 0$, $\mathbf{u}_2 - \mathbf{A}_2\mathbf{x}_1 + \mathbf{b}_2 = 0$, and $\|\mathbf{x}_0 - \bar{\mathbf{x}}\| \leq \varepsilon$ for L_2 norm. Let Γ_{sparse} be the index set of the remaining sparse constraints. Define subsets $I^{(i)} = \{t^{(i)}, x_0^{(i)}\}$ for $i \in [p_0]$; $J_1^{(j)} = \{x_1^{(j)}, u_1^{(j)}\}$, $J_2^{(j)} = \{z_1^{(j)}, u_1^{(j)}\}$ for $j \in [p_1]$; $K^{(k)} = \{z_2^{(k)}, u_2^{(k)}\}$ for $k \in [p_2]$. Set level $\mathbf{l} = \{\mathbf{0}_{|\Gamma_{dense}|}, \mathbf{2}_{|\Gamma_{sparse}|}\}$ and depth $\mathbf{q} = \mathbf{1}_{|\Gamma_{dense}|+|\Gamma_{sparse}|}$. Replacing the full moment matrix $\mathbf{M}_2(\mathbf{y})$ by its submatrices $\mathbf{M}_2^{sub}(\mathbf{y}, \{t^{(i)}, u_1^{(j)}, u_2^{(k)}\})$, and applying second-order $(\mathbf{l}, \mathbf{q})$-sublevel relaxation w.r.t. subsets $I^{(i)}, J_1^{(j)}, J_2^{(j)}, K^{(k)}$ to problem

(LipFcDNN$_2$-k), we obtain an enhanced relaxation (Sub$_1$-LipFcDNN$_2$-k):

$$\sup_{\mathbf{y}} \quad L_{\mathbf{y}}(\mathbf{t}^T\mathbf{A}_1^T\operatorname{Diag}(\mathbf{z}_1)\mathbf{A}_2^T\operatorname{Diag}(\mathbf{z}_2)\mathbf{c}_k) \qquad \text{(Sub}_2\text{-LipFcDNN}_2\text{-}k\text{)}$$

$$\text{s.t.} \quad \begin{cases} L_{\mathbf{y}}(1)=1,\ \mathbf{M}_1(\mathbf{y})\succeq 0, \\ \mathbf{M}_2^{sub}(\mathbf{y},\{t^{(i)},u_1^{(j)},u_2^{(k)}\})\succeq 0,\ i\in[p_0],\ j\in[p_1],\ k\in[p_2], \\ \mathbf{M}_2(\mathbf{y},I_1^{(i)})\succeq 0,\ \mathbf{M}_2(\mathbf{y},J_1^{(j)})\succeq 0,\ \mathbf{M}_2(\mathbf{y},J_2^{(j)})\succeq 0,\ \mathbf{M}_2(\mathbf{y},K^{(k)})\succeq 0, \\ L_{\mathbf{y}}(\mathbf{u}_1-\mathbf{A}_1\mathbf{x}_0-\mathbf{b}_1)=0,\ L_{\mathbf{y}}(\mathbf{u}_2-\mathbf{A}_2\mathbf{x}_1-\mathbf{b}_2)=0, \\ \mathbf{M}_1(z_1^{(j)}(z_1^{(j)}-1)\mathbf{y},J_2^{(j)})=0,\ \mathbf{M}_1(z_1^{(j)}u_1^{(j)}\mathbf{y},J_2^{(j)})\succeq 0, \\ \qquad\qquad \mathbf{M}_1((z_1^{(j)}-1)u_1^{(j)}\mathbf{y},J_2^{(j)})\succeq 0,\ j\in[p_1], \\ \mathbf{M}_1(z_2^{(j)}(z_2^{(j)}-1)\mathbf{y},K^{(k)})=0,\ \mathbf{M}_1(z_2^{(j)}u_2^{(j)}\mathbf{y},K^{(k)})\succeq 0, \\ \qquad\qquad \mathbf{M}_1((z_2^{(j)}-1)u_2^{(j)}\mathbf{y},K^{(k)})\succeq 0,\ k\in[p_2], \\ \mathbf{M}_1(x_1^{(j)}(x_1^{(j)}-u_1^{(j)})\mathbf{y},J_1^{(j)})=0,\ \mathbf{M}_1((x_1^{(j)}-u_1^{(j)})\mathbf{y},J_1^{(j)})\succeq 0, \\ \qquad\qquad \mathbf{M}_1(x_1^{(j)}\mathbf{y},J_1^{(j)})\succeq 0,\ j\in[p_1], \\ \mathbf{M}_1((1-(t^{(i)})^2)\mathbf{y},I^{(i)})\succeq 0,\ i\in[p_0], \\ L_{\mathbf{y}}(\varepsilon^2-(\mathbf{x}_0-\bar{\mathbf{x}})^T(\mathbf{x}_0-\bar{\mathbf{x}}))\geq 0, \quad \text{(for } L_2 \text{ norm)} \\ \mathbf{M}_1(\varepsilon^2-(x_0^{(i)}-\bar{x}^{(i)})^2,I^{(i)})\geq 0,\ i\in[p_0]. \quad \text{(for } L_\infty \text{ norm)} \end{cases}$$

In this way, we add $p_0p_1p_2$ moment matrices $\mathbf{M}_2^{sub}(\mathbf{y},\{t^{(i)},u_1^{(j)},u_2^{(k)}\})$ of size 3, and $p_0p_1p_2+p_1p_2$ moment variables $L_{\mathbf{y}}(t^{(i)}u_1^{(j)}u_2^{(k)})$, $L_{\mathbf{y}}((u_1^{(j)})^2(u_2^{(k)})^2)$. A variant of this technique is to enlarge the size of the moment matrices but in the meantime reduce the number of moment matrices. For instance, consider the following submatrices of the second-order moment matrix $\mathbf{M}_2(\mathbf{y})$: for $k\in[p_2]$,

$$\mathbf{M}_2^{sub}(\mathbf{y},\{\mathbf{t},\mathbf{u}_1,u_2^{(k)}\})=L_{\mathbf{y}}\left(\begin{pmatrix} 1 & \mathbf{t}^T & \mathbf{u}_1^Tu_2^{(k)} \\ \mathbf{t} & \mathbf{t}\mathbf{t}^T & \mathbf{t}\mathbf{u}_1^Tu_2^{(k)} \\ \mathbf{u}_1u_2^{(k)} & \mathbf{u}_1\mathbf{t}^Tu_2^{(k)} & \mathbf{u}_1\mathbf{u}_1^T(u_2^{(k)})^2 \end{pmatrix}\right). \tag{5.26}$$

We still have all the moments of the cubic terms $t^{(i)}u_1^{(j)}u_2^{(k)}$ from those submatrices $\mathbf{M}_2^{sub}(\mathbf{y},\{\mathbf{t},\mathbf{u}_1,u_2^{(k)}\})$. However, in this case, we only add p_2 moment matrices of size $1+p_0+p_1$, and $p_0p_1p_2+p_1^2p_2$ new variables $L_{\mathbf{y}}(\mathbf{u}_1\mathbf{t}^Tu_2^{(k)})$, $L_{\mathbf{y}}(\mathbf{u}_1\mathbf{u}_1^T(u_2^{(k)})^2)$.

For FcMON: Suppose F is a fully-connected MON. For L_2 and L_∞ norm, the Lipschitz constant estimation problem (LipMON) is a nonconvex QCQP, we

directly apply Shor's relaxation to it and obtain:

$$\max_{\mathbf{y}} \quad L_{\mathbf{y}}(\mathbf{t}^T\mathbf{B}^T\mathbf{h}) \qquad \text{(SHOR-LipFcMON)}$$

$$\text{s.t.} \quad \begin{cases} L_{\mathbf{y}}(1) = 1,\ \mathbf{M}_1(\mathbf{y}) \succeq 0, \\ L_{\mathbf{y}}(1 - \mathbf{w}^T\mathbf{v}) \geq 0, \\ L_{\mathbf{y}}(1 - \mathbf{t}^T\mathbf{t}) \geq 0,\ L_{\mathbf{y}}(1 - \mathbf{w}^T\mathbf{w}) \geq 0, \quad \text{(for } L_2 \text{ norm)} \\ L_{\mathbf{y}}(1 - (t^{(i)})^2) \geq 0,\ i \in [p_0],\ L_{\mathbf{y}}(1 - (w^{(k)})^2) \geq 0,\ k \in [K], \quad \text{(for } L_2 \text{ norm)} \\ L_{\mathbf{y}}(\mathbf{r} - \mathbf{A}^T\mathbf{h} - \mathbf{C}^T\mathbf{v}) = 0,\ L_{\mathbf{y}}(\mathbf{h} - \mathrm{Diag}(\mathbf{s}) \cdot \mathbf{r}) = 0, \\ L_{\mathbf{y}}(\mathbf{s} \circ (\mathbf{s} - 1)) \leq 0,\ L_{\mathbf{y}}(\mathbf{s} \circ (\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b})) \geq 0, \\ L_{\mathbf{y}}((\mathbf{s} - 1) \circ (\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b})) \geq 0, \\ L_{\mathbf{y}}(\mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b})) = 0, \\ L_{\mathbf{y}}(\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b}) \geq 0,\ L_{\mathbf{y}}(\mathbf{x}_1) \geq 0, \\ L_{\mathbf{y}}(\varepsilon^2 - (\mathbf{x}_0 - \bar{\mathbf{x}})^T(\mathbf{x}_0 - \bar{\mathbf{x}})) \geq 0, \quad \text{(for } L_2 \text{ norm)} \\ L_{\mathbf{y}}(\varepsilon^2 - (x_0^{(i)} - \bar{x}^{(i)})^2, I^{(i)}) \geq 0,\ i \in [p_0]. \quad \text{(for } L_\infty \text{ norm)} \end{cases}$$

5.2.3 Experiments

In this section, we provide numerical results for the *global* and *local* Lipschitz constants of *random* $\mathbf{FcDNN}_1$ and $\mathbf{FcDNN}_2$ with various sparsities. We also consider a *real* pre-trained $\mathbf{FcDNN}_1$ and **FcMON**. For DNNs we focus on L_∞ norm, and for MONs we consider both L_2 and L_∞ norms. Let us first take an overview of the methods with which we compare our results.

- **SHOR**: Shor's relaxation applied to $\mathbf{FcDNN}_1$ by (SHOR-LipFcDNN$_1$-k) and **FcMON** by (SHOR-LipFcMON);
- **Sub-1**: first-order sublevel relaxation applied to $\mathbf{FcDNN}_2$ by (Sub$_1$-LipFcDNN$_2$-k);
- **Sub-2**: second-order sublevel relaxation applied to $\mathbf{FcDNN}_1$ by (Sub$_2$-LipFcDNN$_1$-k) and $\mathbf{FcDNN}_2$ by (Sub$_2$-LipFcDNN$_2$-k);
- **LipOpt-d**: LP-based relaxation by [Latorre *et al.* 2020] with degree d;
- **LBS**: lower bound obtained by sampling 50000 random points.

The reason why we list **LBS** here is because **LBS** provides valid lower bounds on the Lipschitz constant. Therefore all methods should provide results greater than **LBS**, a basic necessary condition of consistency.

As discussed before, if we want to estimate the global Lipschitz constant of a neural network $F : \Omega \to \mathbb{R}^K$, we need the input space Ω to be the whole space. In consideration of numerical issues, we set Ω to be the ball of radius 10 around the origin. For local Lipschitz constant, we set by default the radius of input ball as $\varepsilon = 0.1$. In both cases, we compute the Lipschitz constant with respect to the first label ($k = 1$). We use the (Python) code provided by [Latorre *et al.* 2020][2] to

[2] https://openreview.net/forum?id=rJe4_xSFDB.

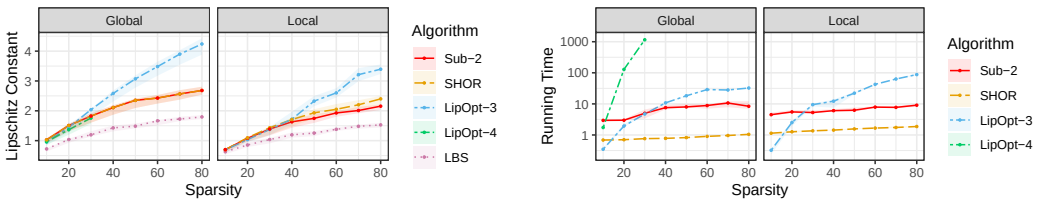
execute the experiments for **LipOpt-d** with Gurobi solver. For **Sub-2** and **SHOR**, we use the YALMIP toolbox (MATLAB) [Löfberg 2004] with MOSEK as a backend to calculate the Lipschitz constants for *random* networks. For *trained* network, we implement our algorithm on Julia [Bezanson *et al.* 2017] with MOSEK optimizer to accelerate the computation. All experiments are run on a personal laptop with a 4-core i5-6300HQ 2.3GHz CPU and 8GB of RAM.

Remark. The crossover option[3] 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 solving time taken by the LP/SDP solver (Gurobi/Mosek) and *Out of Memory (OfM)* means running out of memory during building up the LP/SDP model.

Random FcDNN$_1$: We first compare the upper bounds for $(80, 80)$ networks, whose weights $\mathbf{A}_1 \in \mathbb{R}^{80\times 80}$ and biases $\mathbf{b}_1 \in \mathbb{R}^{80}$ are randomly generated. We define a certain sparsity structure of those DNNs as proposed by [Latorre *et al.* 2020]. For illustration purpose, consider a neural network F with one single hidden layer, and 4 nodes in each layer. The network F is said to have *sparsity 2* if its weight matrix $\mathbf{A} \in \mathbb{R}^{4\times 4}$ is symmetric with diagonal blocks of size at most 2×2:

$$\begin{pmatrix} * & * & 0 & 0 \\ * & * & * & 0 \\ 0 & * & * & * \\ 0 & 0 & * & * \end{pmatrix}. \tag{5.27}$$

Larger sparsity values refer to symmetric matrices with band structure of a given size. We use the codes provided by [Latorre *et al.* 2020] to generate networks with various sparsities. For each fixed sparsity, we generate 10 different random networks, and apply all the methods to them repeatedly. Then we compute the average upper bound and average running time of those 10 experiments.



(a) Upper bounds (b) Computation time

Figure 5.3: Lipschitz constant upper bounds and solving time for $(80, 80)$ networks.

We generate random networks of size $(80, 80)$ with sparsity 10, 20, 30, 40, 50, 60, 70, 80. In the meantime, and calculate median and quartiles over 10 random

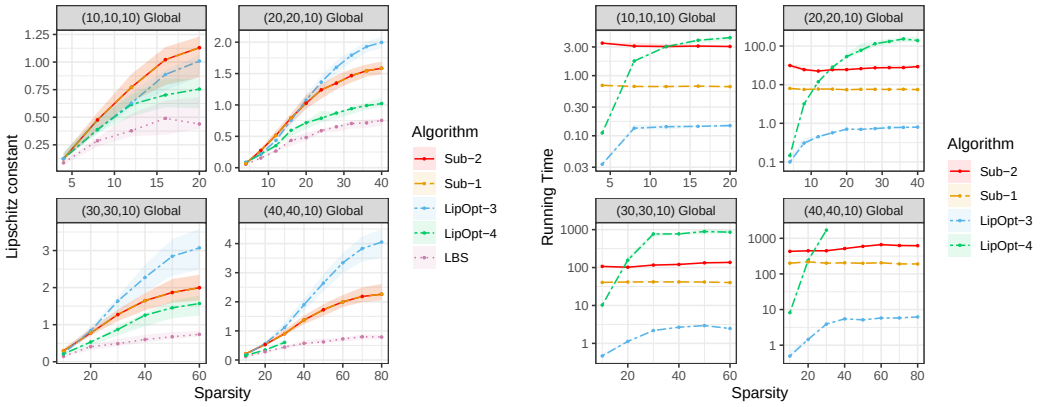
[3] https://www.gurobi.com/documentation/9.0/refman/crossover.html

networks draws. Figure 5.3 displays a comparison of average upper bounds of global and local Lipschitz constants, and average solving time with respect to L_∞ norm obtained by **SHOR**, **Sub-2**, **LipOpt-3** and **LipOpt-4**. For global bounds, we see from Figure 5.3a 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 **Sub-2**. As the sparsity increases, **Sub-2** provides tighter bounds. Figure 5.3b shows that **LipOpt-3** is more efficient than **Sub-2** only for sparsity 10. **LipOpt-4** shows great potential on the results. However, it is not applicable when the sparsity increases, due to the high computational burden. 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 **Sub-2** give nearly the same upper bounds. However, in the local case, **Sub-2** provides strictly tighter bounds than **SHOR**. In both global and local cases, **SHOR** has smaller computational time than **Sub-2**.

Random FcDNN$_2$: We generate 2-hidden layer random networks with 10, 20, 30, 40 neurons in the input layer and first hidden layer, we fix the number of neurons in the second hidden layer to 10. For size $(10, 10, 10)$, we consider sparsity 4, 8, 12, 16, 20; for size $(20, 20, 10)$, we consider sparsity 4, 8, 12, 16, 20, 24, 28, 32, 36, 40; for size $(30, 30, 10)$, we consider sparsity 10, 20, 30, 40, 50, 60; for size $(40, 40, 10)$, we consider sparsity 10, 20, 30, 40, 50, 60, 70, 80. In the meantime, we display median and quartiles over 10 random networks draws. Figure 5.4 displays the average upper bounds of global Lipschitz constants and running time of different algorithms for 2-hidden layer random networks of different sizes and sparsities. We see from Figure 5.4a that the SDP-based method **Sub-2** performs worse than the LP-based method **LipOpt-3** for networks of size $(10, 10, 10)$. However, as the size and the sparsity of the network increase, the gap between **Sub-2** and **LipOpt-3** becomes smaller (and **Sub-2** performs even better). For networks of size $(20, 20, 10)$, $(30, 30, 10)$ and $(40, 40, 10)$, with sparsity greater than 10, **Sub-2** provides strictly tighter bounds than **LipOpt-3**, with the price of higher computational time.

Trained FcDNN$_1$ for MNIST: We use the same MNIST classifier *SDP-NN* described in [Raghunathan *et al.* 2018a][4]. The network is of size $(784, 500)$ for 10-classification, we calculate the upper bounds with respect to label 2 ($k = 3$). In Table 5.2, 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 **Sub-2** still work and moreover, **Sub-2** provides tighter upper bounds than **SHOR** in both global and local cases. As a trade-off, the running time of **Sub-2** is around 5 times longer than that of **SHOR**.

[4] https://worksheets.codalab.org/worksheets/0xa21e794020bb474d8804ec7bc0543f52/



(a) (b)

Figure 5.4: Global Lipschitz constant for 2-hidden layer networks.

Table 5.2: Upper bounds of Lipschitz constant and solving time on SDP-NN.

	Global				Local			
	Sub-2	**SHOR**	**LipOpt-3**	**LBS**	**Sub-2**	**SHOR**	**LipOpt-3**	**LBS**
Bound	14.56	17.85	OfM	9.69	12.70	16.07	OfM	8.20
Time	12246	2869	OfM	-	20596	4217	OfM	-

Trained FcMON for MNIST: The MON we use consists of a fully-connected hidden layer with 87 neurons and we set its monotonicity factor m to be 20. The training is based on the normalized MNIST database and all training hyperparameters are set to be the same as in [Winston & Kolter 2020, Table D1][5]. We use the same normalization setting on each test example with mean $\mu = 0.1307$ and standard deviation $\sigma = 0.3081$, which means that each input is an image of size 28×28 with entries varying from -0.42 to 2.82. For every perturbation ε, we also take the normalization into account, i.e., we use the normalized perturbation ε/σ for each input. We compare the upper bounds of Lipschitz constants computed by (SHOR-LipFcMON) with the ones proposed by [Pabbaraju *et al.* 2021]. Notice that the upper bounds in [Pabbaraju *et al.* 2021] is dedicated to estimate the Lipschitz constant of function $\mathbf{x}_1(\mathbf{x}_0)$, not $F(\mathbf{x}_0)$. Denote by $\mathbf{Pab}^2_{\mathbf{x}_1}$ the upper bound of the Lipschitz constant of $\mathbf{x}_1$ w.r.t. the L_2 norm, which is given by $\mathbf{Pab}^2_{\mathbf{x}_1} = \|\mathbf{B}\|_2/m$ according to [Pabbaraju *et al.* 2021], where $\mathbf{B}$ is the parameter of the network and m is the monotonicity factor. We can then compute the upper bound w.r.t. L_∞ norm by $\mathbf{Pab}^\infty_{\mathbf{x}_1} = \sqrt{p_0}\cdot\mathbf{Pab}^2_{\mathbf{x}_1}$, where p_0 is the input dimension. The upper bound of Lipschitz constant of F is computed via the upper bound of $\mathbf{x}_1$: $\mathbf{Pab}^q_F = \|\mathbf{C}\|_q \cdot \mathbf{Pab}^q_{\mathbf{x}_1}$ for $q = 2, \infty$. Denote similarly by $\mathbf{SHOR}^q_F$ the upper bounds of Lipschitz constants of F provided by (SHOR-LipFcMON) w.r.t. L_q norm, we are able to compare the

[5]The training code (in Python) is available at `https://github.com/locuslab/monotone_op_net`.

tightness of upper bounds given by the two methods.

Table 5.3: Upper bounds of Lipschitz constant and solving time for L_2, L_∞ norm.

	$q=2$		$q=\infty$	
	bound	time (s)	bound	time (s)
$\mathbf{Pab}_F^q$	4.80	-	824.14	-
$\mathbf{SHOR}_F^q$	4.67	1756.58	108.84	1898.65

From Table 5.3, we see that Shor's relaxation provides consistently tighter upper bounds than the ones in [Pabbaraju *et al.* 2021]. Especially for L_∞ norm, the upper bound computed by equation $|||\mathbf{C}|||_\infty \cdot \mathbf{Pab}_{\mathbf{x}_1}^\infty$ is rather crude compared to the bound obtained directly by (SHOR-LipFcMON). As a trade-off, we can obtain results immediately by $\mathbf{Pab}_F^q$, while suffering half an hour by Shor's relaxation.

5.3 Abstract Domain Propagation

The approach we are concerned here is about abstract interpretation, which enables one to prove program properties on a set of inputs without actually running the program. Formally, given a function $f : \mathbb{R}^m \to \mathbb{R}^n$, a set of inputs $\mathcal{X} \subseteq \mathbb{R}^m$, and a property $\mathcal{C} \subseteq \mathbb{R}^n$, the goal is to determine whether the property holds, i.e., whether $f(\mathbf{x}) \in \mathcal{C}$ for all $\mathbf{x} \in \mathcal{X}$. Usually the set $\mathcal{X}$ can be very large or infinite, so that we cannot enumerate all points in $\mathcal{X}$ to verify whether $f(\mathbf{x}) \in \mathcal{C}$. Instead, abstract interpretation overapproximates sets defined by abstract domains and verifies the properties on the resulting abstract domains. Since abstract interpretation employs overapproximation, it is sound, but is not complete, i.e., may fail to prove the properties when they hold.

In this section, we focus on overapproximating the output region of a neural network F by an abstract domain under ReLU functions. The most popular abstract domains are: box, zonotope, polyhedra and ellipsoid. Different abstract domains have their advantages but also their drawbacks. For example, to encode boxes is significantly faster to encode polyhedra, while polyhedra are much more precise than boxes. It is easy to describe ellipsoid and compute its volume, despite that one needs to calculate the determinant of a symmetric matrix. Related works includes robustness verification by abstract interpretation [Gehr *et al.* 2018], safety verification and robustness analysis of neural networks by SDP and quadratic constraints [Fazlyab *et al.* 2020], reachability analysis of closed-loop dynamic systems with neural network controllers by SDP [Hu *et al.* 2020], and reachability analysis combining ellipsoids and zonotopes by [Kousik *et al.* 2022].

5.3.1 Problem setting

Consider neural network $F : \Omega \to \mathbb{R}^K$ with $\Omega \in \mathbb{R}^{p_0}$, activated by ReLU function. We assume that the input region Ω is one of the following abstract domains:

box, zonotope, polyhedron adn ellipsoid. Note that all these abstract domains are semialgebraic:

- *Box*: a box $\mathcal{B}^n \subseteq \mathbb{R}^n$ has form $\{\mathbf{x} \in \mathbb{R}^n : a_i \leq x_i \leq b_i\}$ with $a_i \leq b_i$ for all $i \in [n]$;
- *Zonotope*: a zonotope $\mathcal{Z} \subseteq \mathbb{R}^n$ has form $\{\mathbf{z} \in \mathbb{R}^n : \mathbf{z} = \mathbf{A}\mathbf{x}+\mathbf{b}, \mathbf{x} \in \mathcal{B}^m\}$ where $\mathbf{A} \in \mathbb{R}^{n\times m}, \mathbf{b} \in \mathbb{R}^n$. In other words, a zonotope is an affine transformation of a box;
- *Polyhedron*: a polyhedron $\mathcal{P} \subseteq \mathbb{R}^n$ has form $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$;
- *Ellipsoid*: an ellipsoid $\mathcal{E} \subseteq \mathbb{R}^n$ has form

$$\mathcal{E} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{Q}\mathbf{x} + \mathbf{q}\|_2 \leq 1\}, \tag{5.28}$$

  where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{q} \in \mathbb{R}^n$. Or equivalently,

$$\mathcal{E} = \{\mathbf{x} \in \mathbb{R}^n : -\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}^T\mathbf{x} + c \geq 0\}, \tag{5.29}$$

  where $\mathbf{Q} \in \mathbb{S}^n_+, \mathbf{q} \in \mathbb{R}^n, c \in \mathbb{R}$, and $\mathbb{S}^n_+$ is the cone of positive semidefinite matrices of size $n \times n$, with an additional constraint $\begin{pmatrix} \mathbf{Q} & \mathbf{q}/2 \\ \mathbf{q}^T/2 & 1-c \end{pmatrix} \succeq 0$.

Denote by $F(\Omega) \subseteq \mathbb{R}^K$ the image of Ω under network F. The goal of this section is to find a proper abstract domain that overapproximate the set $F(\Omega)$. Let $\mathcal{D}$ be an abstract domain such that $F(\Omega) \subseteq \mathcal{D}$, we want that the volume of $\mathcal{D}$ is as small as possible, which is indeed to solve an optimization problem:

$$\min_{\mathcal{D}}\{\mathrm{Vol}(\mathcal{D}) : F(\Omega) \subseteq \mathcal{D}\}. \tag{5.30}$$

Throughout the following discussion, we are going to focus on the case where Ω is an L_p ball ($p = 2, \infty$) and $\mathcal{D}$ is an ellipsoid. Precisely, suppose the input region $\Omega = B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|_p)$ is an L_p ball for a given input $\bar{\mathbf{x}} \in \mathbb{R}^{p_0}$ and perturbation $\varepsilon > 0$, and $\mathcal{D} \subseteq \mathbb{R}^K$ is an ellipsoid either parameterized by $\mathbf{Q} \in \mathbb{R}^{K\times K}, \mathbf{q} \in \mathbb{R}^K$ and equation (5.28), or by $\mathbf{Q} \in \mathbb{S}^K_+, \mathbf{q} \in \mathbb{R}^K, c \in \mathbb{R}$ and equation (5.29). We first consider the characterization of (5.28). Then if $F(\Omega) \subseteq \mathcal{D}$, for all $\mathbf{x}_0 \in \mathbb{R}^{p_0}$, we have $-F(\mathbf{x}_0)^T\mathbf{Q}F(\mathbf{x}_0) + \mathbf{q}^TF(\mathbf{x}_0) + c \geq 0$. Moreover, the volume of ellipsoid $\mathcal{D}$ is inversely proportional to the determinant of matrix $\mathbf{Q}$. In other words, to minimize $\mathrm{Vol}(\mathcal{D})$ is equivalent to maximize $\det(\mathbf{Q})$. Therefore, in this case, the *ellipsoid propagation problem* (5.30) can be formulated as

$$\max_{\mathbf{Q}\in\mathbb{S}^K,\, \mathbf{q}\in\mathbb{R}^K,\, c\in\mathbb{R}} \quad \det(\mathbf{Q}) \tag{Ellip}$$
$$\text{s.t.} \quad \begin{cases} -F(\mathbf{x}_0)^T\mathbf{Q}F(\mathbf{x}_0) + \mathbf{q}^TF(\mathbf{x}_0) + c \geq 0, \ \forall \mathbf{x}_0 \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|_p), \\ \begin{pmatrix} \mathbf{Q} & \mathbf{q}/2 \\ \mathbf{q}^T/2 & 1-c \end{pmatrix} \succeq 0. \end{cases}$$

For DNNs: Suppose $F : \Omega \to \mathbb{R}^K$ is a fully-connected DNN with ReLU activation function, and let $\mathbf{W} = (\mathbf{A}_1, \mathbf{b}_1; \ldots; \mathbf{A}_L, \mathbf{b}_L; \mathbf{C})$ be its parameters. Then for input $\mathbf{x}_0 \in \mathbb{R}^{p_0}$, $F(\mathbf{x}_0) = \mathbf{C}\mathbf{x}_L$ where $\mathbf{x}_i = \mathrm{ReLU}(\mathbf{A}_i\mathbf{x}_{i-1} + \mathbf{b}_i)$. Let $\mathcal{O}_{dnn} \subseteq \mathbb{R}^{p_L}$ be the output region of $\mathbf{x}_L$ for input $\mathbf{x}_0 \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|_p)$, i.e.,

$$\mathcal{O}_{dnn} := \{\mathbf{x}_L \in \mathbb{R}^{p_L} : \mathbf{x}_i = \mathrm{ReLU}(\mathbf{A}_i\mathbf{x}_{i-1} + \mathbf{b}_i),\ i \in [L],\ \mathbf{x}_0 \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|_p)\}.$$

Then the ellipsoid propagation problem (Ellip) for DNNs reads

$$\begin{aligned} \max_{\mathbf{Q}\in\mathbb{S}^K,\, \mathbf{q}\in\mathbb{R}^K,\, c\in\mathbb{R}} \quad & \det(\mathbf{Q}) \\ \text{s.t.} \quad & \begin{cases} -\mathbf{x}_L^T\mathbf{C}^T\mathbf{Q}\mathbf{C}\mathbf{x}_L + \mathbf{q}^T\mathbf{C}\mathbf{x}_L + c \geq 0,\ \forall \mathbf{x}_L \in \mathcal{O}_{dnn}, \\ \begin{pmatrix} \mathbf{Q} & \mathbf{q}/2 \\ \mathbf{q}^T/2 & 1-c \end{pmatrix} \succeq 0. \end{cases} \end{aligned} \tag{EllipDNN}$$

For MONs: Let $F : \Omega \to \mathbb{R}^K$ be a fully-connected MON activated by ReLU function, and let $\mathbf{W} = (\mathbf{A}, \mathbf{B}, \mathbf{b}; \mathbf{C})$ be the parameters of F. For $\mathbf{x}_0 \in \mathbb{R}^{p_0}$, we have $F(\mathbf{x}_0) = \mathbf{C}\mathbf{x}_1$ where $\mathbf{x}_1 = \mathrm{ReLU}(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b})$. Let $\mathcal{O}_{mon} \subseteq \mathbb{R}^{p_1}$ be the output region of $\mathbf{x}_1$ for input $\mathbf{x}_0 \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|_p)$, i.e.,

$$\mathcal{O}_{mon} := \{\mathbf{x}_1 \in \mathbb{R}^{p_1} : \mathbf{x}_1 = \mathrm{ReLU}(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}),\ \mathbf{x}_0 \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|_p)\}.$$

Then the ellipsoid propagation problem (Ellip) for MONs is formulated as

$$\begin{aligned} \max_{\mathbf{Q}\in\mathbb{S}^K,\, \mathbf{q}\in\mathbb{R}^K,\, c\in\mathbb{R}} \quad & \det(\mathbf{Q}) \\ \text{s.t.} \quad & \begin{cases} -\mathbf{x}_1^T\mathbf{C}^T\mathbf{Q}\mathbf{C}\mathbf{x}_1 + \mathbf{q}^T\mathbf{C}\mathbf{x}_1 + c \geq 0,\ \forall \mathbf{x}_1 \in \mathcal{O}_{mon}, \\ \begin{pmatrix} \mathbf{Q} & \mathbf{q}/2 \\ \mathbf{q}^T/2 & 1-c \end{pmatrix} \succeq 0. \end{cases} \end{aligned} \tag{EllipMON}$$

5.3.2 Algorithms

By semialgebraicity of ReLU function described in (5.1), the output region $\mathcal{O}_{dnn}$ and $\mathcal{O}_{mon}$ are both semialgebraic. Precisely, for DNNs,

$$\begin{aligned} \mathcal{O}_{dnn} = \{&\mathbf{x}_L \in \mathbb{R}^{p_L} : \mathbf{x}_i \circ (\mathbf{x}_i - \mathbf{A}_i\mathbf{x}_{i-1} - \mathbf{b}_i) = 0, \\ &\mathbf{x}_i - \mathbf{A}_i\mathbf{x}_{i-1} - \mathbf{b}_i \geq 0, \mathbf{x}_i \geq 0,\ i \in [L],\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\|_p \leq \varepsilon\}, \end{aligned} \tag{5.31}$$

and for MONs,

$$\begin{aligned} \mathcal{O}_{mon} = \{&\mathbf{x}_1 \in \mathbb{R}^{p_1} : \mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b}) = 0, \\ &\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b} \geq 0,\ \mathbf{x}_1 \geq 0, \|\mathbf{x}_0 - \bar{\mathbf{x}}\|_p \leq \varepsilon\}. \end{aligned} \tag{5.32}$$

Note that in problems (EllipDNN) and (EllipMON), we have nonnegative (ellipsoidal) constraints with variables lying in semialgebraic sets $\mathcal{O}_{dnn}$ and $\mathcal{O}_{mon}$.

Applying the similar idea presented in Section 3.2.2, one can replace the nonnegative constraints by WSOSs, which results in relaxations of the original problems. Precisely, recall that the SOS cone $\Sigma[\mathbf{x}]$ is contained in the nonnegative cone $\mathcal{P}[\mathbf{x}]$. The constraint $g(\mathbf{x}_L) := -\mathbf{x}_L^T\mathbf{C}^T\mathbf{Q}\mathbf{C}\mathbf{x}_L + \mathbf{q}^T\mathbf{C}\mathbf{x}_L + c \geq 0$ indicates that $g \in \mathcal{P}[\mathbf{x}_L]$. By enforcing g to be a WSOS w.r.t. semialgebraic sets $\mathcal{O}_{dnn}$ and $\mathcal{O}_{mon}$, we have $g \in \Sigma[\mathbf{x}_L] \subseteq \mathcal{P}[\mathbf{x}_L]$. The Lasserre's hierarchy, sublevel hierarchy can be then applied to polynomial g. We discuss in details the modeling and reformulation for DNNs and MONs respectively.

For DNNs Suppose the network F is a fully-connected DNN. Denote by $\mathbf{x}_{0:L}$ the union of variables $\mathbf{x}_i$ for $0 \leq i \leq L$, and $\mathbf{x}_i^{(j:k)}$ the union of variables $x_i^{(j)}, x_i^{(j+1)}, \ldots, x_i^{(k)}$ for $j \leq k$. Replace the nonnegative constraint $-\mathbf{x}_L^T\mathbf{C}^T\mathbf{Q}\mathbf{C}\mathbf{x}_L + \mathbf{q}^T\mathbf{C}\mathbf{x}_L + c \geq 0$ in problem (EllipDNN) by WSOS w.r.t. semialgebraic set $\mathcal{O}_{dnn}$, we obtain the following d-th order SOS problem: for $d \geq 1$,

$$\max_{\mathbf{Q}\in\mathbb{S}^K,\, \mathbf{q}\in\mathbb{R}^K,\, c\in\mathbb{R}} \quad \det(\mathbf{Q}) \qquad \text{(SOS}_d\text{-EllipDNN)}$$

$$\text{s.t.} \quad \begin{cases} -\mathbf{x}_L^T\mathbf{C}^T\mathbf{Q}\mathbf{C}\mathbf{x}_L + \mathbf{q}^T\mathbf{C}\mathbf{x}_L + c = \sigma^{(0)}(\mathbf{x}_{0:L}) \\ \quad + \sum_{i=1}^{L}\sum_{j=1}^{p_i}\Big(\tau_{i,j}(\mathbf{x}_{0:L})\cdot x_i^{(j)}(x_i^{(j)} - \mathbf{A}_i^{(j,:)}\mathbf{x}_{i-1} - b_i^{(j)}) \\ \qquad +\sigma_{i,j}^{(1)}(\mathbf{x}_{0:L})\cdot(x_i^{(j)} - \mathbf{A}_i^{(j,:)}\mathbf{x}_{i-1} - b_i^{(j)}) + \sigma_{i,j}^{(2)}(\mathbf{x}_{0:L})\cdot x_i^{(j)}\Big) \\ \quad +\sigma^{(3)}(\mathbf{x}_{0:L})\cdot(\varepsilon^2 - (\mathbf{x}_0 - \bar{\mathbf{x}})^T(\mathbf{x}_0 - \bar{\mathbf{x}})) \quad (L_2 \text{ norm}) \\ \quad + \sum_{j=1}^{p_0}\sigma_i^{(3)}(\mathbf{x}_{0:L})\cdot(\varepsilon^2 - (x_0^{(j)} - \bar{x}^{(j)})^2) \quad (L_\infty \text{ norm}), \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall \mathbf{x}_i \in \mathbb{R}^{p_i},\ 0 \leq i \leq L, \\ \begin{pmatrix} \mathbf{Q} & \mathbf{q}/2 \\ \mathbf{q}^T/2 & 1-c \end{pmatrix} \succeq 0. \end{cases}$$

where $\tau_{i,j} \in \mathbb{R}_{2d-2}[\mathbf{x}_{0:L}], \sigma^{(0)} \in \Sigma_d[\mathbf{x}_{0:L}], \sigma_{i,j}^{(1)}, \sigma_{i,j}^{(2)}, \sigma^{(3)}, \sigma_i^{(3)} \in \Sigma_{d-1}[\mathbf{x}_{0:L}]$ for $j \in [p_i], i \in [L]$. The SOS problem (SOS$_d$-EllipDNN) is dense and thus is not of much interest. Now we are going to explore sparsity patterns and design sublevel structures to employ sublevel hierarchy to (SOS$_d$-EllipDNN).

Construct subsets $I_{i,j,k}$ according to the cyclic heuristic **H2** as follows: fix a level $l \geq 0$, for $j \in [p_i]$, $k \in [p_{i-1} - l + 1]$, and $i \in [L]$, define

$$I_{i,j,k} := \{\mathbf{x}_{i-1}^{(k:k+l-1)}; x_i^{(j)}\}. \tag{5.33}$$

For each i and j, the $p_i - l + 1$ many subsets $I_{i,j,k}$ form a finite (cyclic) covering of the variables $\{\mathbf{x}_{i-1}; x_i^{(j)}\}$, whose corresponding SOS polynomials serve as weights of the sublevel relaxation for each constraint. Note that $|I_{i,j,k}| = l + 1$ for all i, j, k.

Let $\mathbf{l} = (l+1)\cdot\mathbf{1}_N$ and $\mathbf{q} = [\mathbf{q}_i]_{i=0}^L$, where

$$N = \begin{cases} 1 + 3\sum_{i=1}^{L} p_i, & \text{for } L_2 \text{ norm;} \\ p_0 + 3\sum_{i=1}^{L} p_i, & \text{for } L_\infty \text{ norm,} \end{cases}$$

and

$$\mathbf{q}_0 = \begin{cases} p_0 - l + 1, & \text{for } L_2 \text{ norm;} \\ (p_0 - l + 1)\cdot\mathbf{1}_{p_1}, & \text{for } L_\infty \text{ norm,} \end{cases}$$
$$\mathbf{q}_i = (p_{i-1} - l + 1)\cdot\mathbf{1}_{p_i},\ i \in [L].$$

With the subsets defined in (5.33), the d-th order $(\mathbf{l},\mathbf{q})$-sublevel relaxation w.r.t. subsets $I_{i,j,k}$ applied to problem (EllipDNN) reads: for $d \geq 1$,

$$\max_{\mathbf{Q}\in\mathbb{S}^K,\,\mathbf{q}\in\mathbb{R}^K,\,c\in\mathbb{R}} \quad \det(\mathbf{Q}) \qquad \text{(Sub}_d\text{-EllipDNN)}$$
$$\text{s.t.} \quad \begin{cases} -\mathbf{x}_L^T\mathbf{C}^T\mathbf{Q}\mathbf{C}\mathbf{x}_L + \mathbf{q}^T\mathbf{C}\mathbf{x}_L + c = \sigma^{(0)}(\mathbf{x}_{0:L}) \\ \quad + \sum_{i=1}^{L}\sum_{j=1}^{p_i}\sum_{k=1}^{p_i-l+1}\Big(\sigma_{i,j,k}^{(0)}(\mathbf{x}_{i-1}^{(k:k+l-1)}; x_i^{(j)}) + \\ \qquad + \tau_{i,j,k}(\mathbf{x}_{i-1}^{(k:k+l-1)}; x_i^{(j)})\cdot x_i^{(j)}(x_i^{(j)} - \mathbf{A}_i^{(j,:)}\mathbf{x}_{i-1} - b_i^{(j)}) \\ \qquad + \sigma_{i,j,k}^{(1)}(\mathbf{x}_{i-1}^{(k:k+l-1)}; x_i^{(j)})\cdot(x_i^{(j)} - \mathbf{A}_i^{(j,:)}\mathbf{x}_{i-1} - b_i^{(j)}) \\ \qquad + \sigma_{i,j,k}^{(2)}(\mathbf{x}_{i-1}^{(k:k+l-1)}; x_i^{(j)})\cdot x_i^{(j)}\Big) \\ \quad + \sum_{k=1}^{p_0-l+1}\sigma_k^{(3)}(\mathbf{x}_0^{(k:k+l-1)}; x_1^{(1)})\cdot(\varepsilon^2 - (\mathbf{x}_0 - \bar{\mathbf{x}})^T(\mathbf{x}_0 - \bar{\mathbf{x}})) \quad (L_2 \text{ norm}) \\ \quad + \sum_{j=1}^{p_0}\sum_{k=1}^{p_0-l+1}\sigma_{i,k}^{(3)}(\mathbf{x}_0^{(k:k+l-1)}; x_1^{(1)})\cdot(\varepsilon^2 - (x_0^{(j)} - \bar{x}^{(j)})^2) \quad (L_\infty \text{ norm}), \\ \qquad\qquad \forall \mathbf{x}_i \in \mathbb{R}^{p_i},\ 0 \leq i \leq L, \\ \begin{pmatrix} \mathbf{Q} & \mathbf{q}/2 \\ \mathbf{q}^T/2 & 1-c \end{pmatrix} \succeq 0. \end{cases}$$

where $\tau_{i,j,k} \in \mathbb{R}_{2d-2}[\mathbf{x}_{i-1}^{(k:k+l-1)}; x_i^{(j)}]$, $\sigma^{(0)} \in \Sigma_{d-1}[\mathbf{x}_{0:L}]$, $\sigma_{i,j,k}^{(0)} \in \Sigma_d[\mathbf{x}_{i-1}^{(k:k+l-1)}; x_i^{(j)}]$, $\sigma_{i,j,k}^{(1)}, \sigma_{i,j,k}^{(2)}, \sigma_k^{(3)}, \sigma_{i,k}^{(3)} \in \Sigma_{d-1}[\mathbf{x}_{i-1}^{(k:k+l-1)}; x_i^{(j)}]$ for $j \in [p_i]$, $k \in [p_i - l + 1]$, $i \in [L]$.

For MONs: Suppose now the network F is a fully-connected MON. We use formulation (5.29) instead of (5.28) to characterize an ellipsoid. Similar with the case for DNNs in the SOS problem (SOS$_d$-EllipDNN), we consider the first-order SOS relaxation applied to problem (EllipMON), which results in the following SOS

problem:

$$\max_{\mathbf{Q}\in\mathbb{S}^K,\,\mathbf{q}\in\mathbb{R}^K} \quad \det(\mathbf{Q}) \qquad \text{(SOS}_1\text{-EllipMON)}$$

$$\text{s.t.} \quad \begin{cases} 1 - \|\mathbf{Q}\mathbf{C}\mathbf{x}_1 + \mathbf{q}\|_2^2 = \sigma^{(0)}(\mathbf{x}_{0:1}) \\ \quad + \sum_{j=1}^{p_1} \Big(\tau_j \cdot x_1^{(j)}(x_1^{(j)} - \mathbf{A}^{(j,:)}\mathbf{x}_1 - \mathbf{B}^{(j,:)}\mathbf{x}_0 - b^{(j)}) \\ \qquad + \sigma_j^{(1)} \cdot (x_1^{(j)} - \mathbf{A}^{(j,:)}\mathbf{x}_1 - \mathbf{B}^{(j,:)}\mathbf{x}_0 - b^{(j)}) + \sigma_j^{(2)} \cdot x_1^{(j)} \Big) \\ \quad + \sigma^{(3)} \cdot (\varepsilon^2 - (\mathbf{x}_0 - \bar{\mathbf{x}})^T(\mathbf{x}_0 - \bar{\mathbf{x}})) \quad (L_2 \text{ norm}) \\ \quad + \sum_{j=1}^{p_0} \sigma_j^{(3)} \cdot (\varepsilon^2 - (x_0^{(j)} - \bar{x}^{(j)})^2) \quad (L_\infty \text{ norm}), \\ \qquad\qquad\qquad\qquad \forall \mathbf{x}_i \in \mathbb{R}^{p_i},\ 0 \le i \le L. \end{cases}$$

where $\tau_j \in \mathbb{R}$, $\sigma^{(0)} \in \Sigma_1[\mathbf{x}_{0:1}]$, $\sigma_j^{(1)}, \sigma_j^{(2)}, \sigma_j^{(3)}, \sigma^{(3)} \in \mathbb{R}_+$.

Problem (SOS$_1$-EllipMON) provides an ellipsoid feasible for (EllipMON), i.e., an ellipsoid which contains $F(\Omega)$. In practice, the determinant is replaced by a log-det objective because there exist efficient solver dedicated to optimize such objectives on SDP constraints. In this thesis, we only consider L_2, L_∞ norm, and first-order relaxation ($d = 1$). In this case, problem (EllipMON) is in fact equivalent to a problem with log-det objective and SDP constraints. Indeed, the SOS constraint in problem (SOS$_1$-EllipMON) can be written as

$$\begin{aligned} \sigma^{(0)}(\mathbf{x}_{0:1}) = - \Big(& \|\mathbf{Q}\mathbf{C}\mathbf{x}_1 + \mathbf{q}\|_2^2 - 1 \quad (=: f_1(\mathbf{x}_{0:1})) \\ & + \sum_{j=1}^{p_1} \tau_j \cdot x_1^{(j)}(x_1^{(j)} - \mathbf{A}^{(j,:)}\mathbf{x}_1 - \mathbf{B}^{(j,:)}\mathbf{x}_0 - b^{(j)}) \quad (=: f_2(\mathbf{x}_{0:1})) \\ & + \sum_{j=1}^{p_1} \sigma_j^{(1)} \cdot (x_1^{(j)} - \mathbf{A}^{(j,:)}\mathbf{x}_1 - \mathbf{B}^{(j,:)}\mathbf{x}_0 - b^{(j)}) \quad (=: f_3(\mathbf{x}_{0:1})) \\ & + \sum_{j=1}^{p_1} \sigma_j^{(2)} \cdot x_1^{(j)} \quad (=: f_4(\mathbf{x}_{0:1})) \\ & + \sigma^{(3)} \cdot (\varepsilon^2 - (\mathbf{x}_0 - \bar{\mathbf{x}})^T(\mathbf{x}_0 - \bar{\mathbf{x}})) \quad (L_2 \text{ norm}) \quad (=: f_5(\mathbf{x}_{0:1})) \\ & + \sum_{j=1}^{p_0} \sigma_j^{(3)} \cdot (\varepsilon^2 - (x_0^{(j)} - \bar{x}^{(j)})^2) \quad (L_\infty \text{ norm}) \quad (=: f_5(\mathbf{x}_{0:1})) \Big) \\ = & - \sum_{i=1}^{5} f_i(\mathbf{x}_{0:1}) =: -f(\mathbf{x}_{0:1}). \end{aligned}$$

Denote by $\mathbf{M}_i$ the Gram matrix of polynomial $f_i(\mathbf{x}_{0:1})$ for $i \in [5]$ and $\mathbf{M}$ the Gram matrix of polynomial $f(\mathbf{x}_{0:1})$, with basis $[\mathbf{x}_0^T, \mathbf{x}_1^T, 1]$. Then we have explicitly

$\mathbf{M} = \sum_{i=1}^{5} \mathbf{M}_i$, where $\mathbf{M}_i$ has the following forms

$$\mathbf{M}_1 = \begin{bmatrix} \mathbf{0}_{p_0\times p_0} & \mathbf{0}_{p_0\times p_1} & \mathbf{0}_{p_0\times 1} \\ \mathbf{0}_{p_1\times p_0} & \mathbf{C}^T\mathbf{Q}^2\mathbf{C} & \mathbf{C}^T\mathbf{Q}\mathbf{q} \\ \mathbf{0}_{1\times p_0} & \mathbf{q}^T\mathbf{Q}\mathbf{C} & \mathbf{q}^T\mathbf{q}-1 \end{bmatrix},$$

$$\mathbf{M}_2 = \begin{bmatrix} \mathbf{0}_{p_0\times p_0} & -\frac{1}{2}\mathbf{B}^T\cdot\mathrm{Diag}(\tau_1,\dots,\tau_{p_1}) & \mathbf{0}_{p_0\times 1} \\ -\frac{1}{2}\mathrm{Diag}(\tau_1,\dots,\tau_{p_1})\cdot\mathbf{B} & \mathrm{Diag}(\tau_1,\dots,\tau_{p_1})\cdot(\mathbf{I}_{p_1}-\mathbf{A}) & -\frac{1}{2}\mathrm{Diag}(\tau_1,\dots,\tau_{p_1})\cdot\mathbf{b} \\ \mathbf{0}_{1\times p_0} & -\frac{1}{2}\mathbf{b}^T\cdot\mathrm{Diag}(\tau_1,\dots,\tau_{p_1}) & 0 \end{bmatrix},$$

$$\mathbf{M}_3 = \begin{bmatrix} \mathbf{0}_{p_0\times p_0} & \mathbf{0}_{p_0\times p_1} & -\frac{1}{2}\mathbf{B}^T\cdot\sigma^{(1)} \\ \mathbf{0}_{p_1\times p_0} & \mathbf{0}_{p_1\times p_1} & \frac{1}{2}(\mathbf{I}_{p_1}-\mathbf{A}^T)\cdot\sigma^{(1)} \\ -\frac{1}{2}(\sigma^{(1)})^T\cdot\mathbf{B} & \frac{1}{2}(\sigma^{(1)})^T\cdot(\mathbf{I}_{p_1}-\mathbf{A}) & -(\sigma^{(1)})^T\cdot\mathbf{b} \end{bmatrix},$$

$$\mathbf{M}_4 = \begin{bmatrix} \mathbf{0}_{p_0\times p_0} & \mathbf{0}_{p_0\times p_1} & \mathbf{0}_{p_0\times 1} \\ \mathbf{0}_{p_1\times p_0} & \mathbf{0}_{p_1\times p_1} & \frac{1}{2}\sigma^{(2)} \\ \mathbf{0}_{1\times p_0} & \frac{1}{2}(\sigma^{(2)})^T & 0 \end{bmatrix},$$

$$\mathbf{M}_5 = \begin{cases} \sigma^{(3)}\begin{bmatrix} -\mathbf{I}_{p_0} & \mathbf{0}_{p_0\times p_1} & \mathbf{x}_0 \\ \mathbf{0}_{p_1\times p_0} & \mathbf{0}_{p_1\times p_1} & \mathbf{0}_{p_1\times 1} \\ \mathbf{x}_0^T & \mathbf{0}_{1\times p_1} & \varepsilon^2-\mathbf{x}_0^T\mathbf{x}_0 \end{bmatrix}, & \text{for } L_2\text{-norm;} \\ \begin{bmatrix} -\mathrm{Diag}(\sigma_1^{(3)},\dots,\sigma_{p_0}^{(3)}) & \mathbf{0}_{p_0\times p_1} & \mathrm{Diag}(\sigma_1^{(3)},\dots,\sigma_{p_0}^{(3)})\cdot\mathbf{x}_0 \\ \mathbf{0}_{p_1\times p_0} & \mathbf{0}_{p_1\times p_1} & \mathbf{0}_{p_1\times 1} \\ \mathbf{x}_0^T\cdot\mathrm{Diag}(\sigma_1^{(3)},\dots,\sigma_{p_0}^{(3)}) & \mathbf{0}_{1\times p_1} & (\sigma^{(3)})^T(\varepsilon^2-\mathbf{x}_0^2) \end{bmatrix}, & \text{for } L_\infty\text{-norm.} \end{cases}$$

Since $\sigma_0(\mathbf{x}_{0:1})$ is an SOS polynomial of degree at most 2, its Gram matrix must be positive semidefinite, i.e., $-\mathbf{M} \succeq 0$. The above discussion indicates the following lemma:

Lemma 1. *Let $F : \Omega \to \mathbb{R}^K$ be a fully-connected MON activated by ReLU function. For $p = 2$ or $p = \infty$, problem (SOS$_1$-EllipMON) is equivalent to an optimization problem with SDP constraints:*

$$\max_{\mathbf{Q}\in\mathbb{S}^K,\,\mathbf{q}\in\mathbb{R}^K,\,\sigma_j^{(1)},\sigma_j^{(2)},\sigma_j^{(3)},\sigma^{(3)}\in\mathbb{R}_+,\,\tau_j\in\mathbb{R}} \{\det(\mathbf{Q}) : -\mathbf{M}\succeq 0\}, \qquad \text{(SDP-EllipMON)}$$

where $\mathbf{M} \in \mathbb{S}^{(p_0+p_1+1)\times(p_0+p_1+1)}$ is a symmetric matrix parametrized by decision variables $(\mathbf{Q},\mathbf{q})$, coefficients $(\tau_j,\sigma_j^{(1)},\sigma_j^{(2)},\sigma_j^{(3)},\sigma^{(3)})$, and parameters of the network $(\mathbf{A},\mathbf{B},\mathbf{b};\mathbf{C})$.

Proof : We have already shown that $\mathbf{M} = \sum_{i=1}^{5}\mathbf{M}_i$ and $-\mathbf{M}\succeq 0$. The only problem comes from the fact that matrix $\mathbf{M}_1$ contains quadratic terms $\mathbf{Q}^2$. In fact, according to [Fazlyab *et al.* 2019a, Lemma 5], the constraint $-\mathbf{M}\succeq 0$ is equivalent to an SDP constraint using *Schur complements*, which concludes the proof. □

Remark. If we add redundant constraints (5.22), the corresponding Gram matrix

of the SOS combination of these constraints with basis $[\mathbf{x}_0^T, \mathbf{x}_1^T, 1]$ has form

$$\mathbf{M}_6 = \begin{bmatrix} \mathbf{B} & \mathbf{A} & \mathbf{b} \\ \mathbf{0}_{p_1\times p_0} & \mathbf{I}_{p_1} & \mathbf{0}_{p_1\times 1} \\ \mathbf{0}_{1\times p_0} & \mathbf{0}_{1\times p_1} & 1 \end{bmatrix}^T \begin{bmatrix} \mathbf{0}_{p_0\times p_0} & \mathbf{T} & \mathbf{0}_{p_0\times 1} \\ \mathbf{T} & -2\mathbf{T} & \mathbf{0}_{p_1\times 1} \\ \mathbf{0}_{1\times p_0} & \mathbf{0}_{1\times p_1} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{A} & \mathbf{b} \\ \mathbf{0}_{p_1\times p_0} & \mathbf{I}_{p_1} & \mathbf{0}_{p_1\times 1} \\ \mathbf{0}_{1\times p_0} & \mathbf{0}_{1\times p_1} & 1 \end{bmatrix},$$

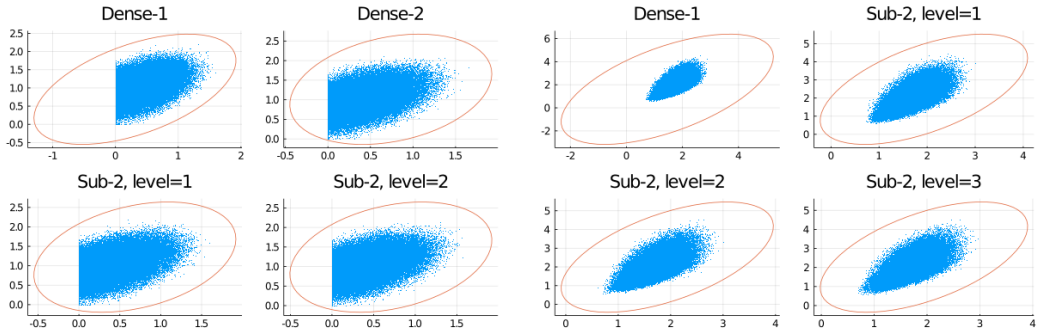
where $\mathbf{T} = \sum_{i=1}^{p_1-1}\sum_{j=i+1}^{p_1} \lambda_{ij}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T$ with $\lambda_{ij} \geq 0$ for all $i < j$, and $\{\mathbf{e}_i\}_{i=1}^{p_1} \subseteq \mathbb{R}^{p_1}$ is the canonical basis of $\mathbb{R}^{p_1}$. In this case, Lemma 1 is still valid with $\mathbf{M} = \sum_{i=1}^{6} \mathbf{M}_i$.

5.3.3 Experiments

In this section, we only show the numerical results of sublevel relaxation applied to $\mathbf{FcDNN}_1$ and $\mathbf{FcDNN}_2$, obtained by (Sub$_d$-EllipDNN) for $d = 2$. The ellipsoid propagation for MONs will be discussed in Section 5.4.3 together with Lipschitz constant estimation problem and robustness verification problem. Let $F : \Omega \to \mathbb{R}^K$ be a fully-connected DNN of size $(p_0, p_1, \ldots, p_L)$. For illustration purpose, we set $K = 2$ so that we are able to draw the graph of the overapproximation ellipsoid on the plane. Based on the discussion in Section 5.3.2, we consider the following approaches for ellipsoid propagation of F:

- **Dense-d**: d-th order SOS relaxation by (SOS$_d$-EllipDNN) for $d = 1, 2$;
- **Sub-2**: second-order sublevel relaxation by (Sub$_d$-EllipDNN) for $d = 2$.

All the networks are generated randomly. We implement the codes in Julia with CVX [Grant & Boyd 2014] package, and use Mosek as a backend to solve the targeted optimization problems (with log-det objective and SDP constraints). All experiments are run on a personal laptop with a 4-core i5-6300HQ 2.3GHz CPU and 8GB of RAM.



(a) $(20, 2)$ network (b) $(20, 20, 2)$ network

Figure 5.5: Ellipsoid propagation of $(20, 2)$ and $(20, 20, 2)$ networks.

In Figure 5.5, the blue point cloud is the output region of the input region (unit ball in $\mathbb{R}^{p_0}$) under neural network F. The red curve is the overapproximation ellipsoid given by different algorithms. We see that, for network of size $(20, 2)$, the

second-order sublevel relaxation of level 1, namely **Sub-2**, already provides overapproximation as good as the dense second-order relaxation **Dense-2**. However, for both networks, the dense first-order relaxation **Dense-1** is far away precise enough compared to the second-order relaxations. Moreover, Table 5.4 shows that **Sub-2** not only provides precise approximations, but also suffers much less computational burden compared to **Dense-2**. For network of size $(20, 20, 2)$, the second order dense relaxation **Dense-2** is out of memory while the sublevel relaxations provide valid results, which is better than the results given by **Dense-1**, in modest time.

Table 5.4: Ellipsoid propagation of $(20, 2)$ and $(20, 20, 2)$ networks.

	$(20, 2)$ network				$(20, 20, 2)$ network				
	Dense-1	**Dense-2**	**Sub-2**		**Dense-1**	**Dense-2**	**Sub-2**		
			level 1	level 2			level 1	level 2	level 3
Objective	0.48	**0.62**	**0.60**	**0.61**	0.07	OfM	**0.18**	**0.20**	**0.21**
Time	0.06	216.39	**5.96**	**8.15**	0.37	OfM	**103.93**	**151.53**	**204.59**

5.4 Robustness Verification

Even with high test accuracy, DNNs are very sensitive to tiny input perturbations, see e.g. [Szegedy *et al.* 2014, Goodfellow *et al.* 2015], hence become crucial for evaluating the safety of a neural network. Robustness to input perturbation has been investigated in many different works with various techniques, including SDP relaxation by [Raghunathan *et al.* 2018b], fast first-order SDP algorithm by [Dathathri *et al.* 2020], abstract interpretation by [Gehr *et al.* 2018], multi-neuron convex relaxations by [Müller *et al.* 2022], GPU-based method by [Müller *et al.* 2021a] which can scale to large networks with one million neurons and 34 layers, scalable quantitative verification framework by [Baluta *et al.* 2021], SMT solver by [Katz *et al.* 2017], analytical certification with Fast-lin, Fast-lip by [Weng *et al.* 2018a], CROWN by [Zhang *et al.* 2018] and their extension to convolutional neural networks with CNN-Cert by [Boopathy *et al.* 2019]. All these methods are restricted to DNNs or CNNs and do not directly apply to MONs.

There are less works related to the application of robustness verification to MONs. The robustness verifier proposed by [Müller *et al.* 2021b] is a zonotope-based scalable and precise method, which can be regarded as an extension of AI2 [Gehr *et al.* 2018] to MONs. [Wei & Kolter 2022] proposed an interval bound propagation (IBP) method to certify robustness of MONs. The IBP algorithm for a typical MON is usually unstable, i.e., the related fixed point equation is not guaranteed to have a solution. By modifying and adding additional constraints to the layers of a MON, the authors propose a special type of MONs, called *IBP-MON*, with a guaranteed unique fixed point with provable interval bounds on the fixed point value. Based on IBP, [Li *et al.* 2022] also proposed a robust training method for DEQs, called *CerDEQ*, which can achieve state-of-the-art performance compared with models using regular convolution and linear layers on l_∞ tasks with $\varepsilon = 8/255$.

5.4.1 Problem setting

We are now going to formally describe the certification problem in terms of mathematical language. Consider a neural network F for classification with ReLU activation function. Then F is an application from the input space $\Omega \subseteq \mathbb{R}^{p_0}$ to the output space $\mathbb{R}^K$, where p_0 is the dimension of input and K is the number of labels. For a given input $\bar{\mathbf{x}} \in \mathbb{R}^{p_0}$, we want to study the robustness of network F with respect to input $\bar{\mathbf{x}}$. Denote by $\bar{y}$ the prediction of $\bar{\mathbf{x}}$, i.e., $\bar{y} = \arg\max_i F(\mathbf{x})_i$. Let $\|\cdot\|$ be a norm on $\mathbb{R}^{p_0}$ and $B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)$ the ball centered at $\bar{\mathbf{x}}$ with radius $\varepsilon > 0$ for norm $\|\cdot\|$, i.e., $B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|) := \{\mathbf{x} \in \mathbb{R}^{p_0} : \|\mathbf{x} - \bar{\mathbf{x}}\| \leq \varepsilon\}$. For a different input $\mathbf{x} \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)$, if the prediction of $\mathbf{x}$ varies from that of $\bar{\mathbf{x}}$, then the network F is not robust at $\bar{\mathbf{x}}$ w.r.t. perturbation ε. And it is natural to define the robustness of a network:

Definition 5.4. The network F is *ε-robust* at point $\bar{\mathbf{x}}$ w.r.t. norm $\|\cdot\|$, if for any $\mathbf{x} \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)$, the prediction of $\mathbf{x}$ is the same as that of $\bar{\mathbf{x}}$.

Let y be the prediction of $\mathbf{x} \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)$. By Definition 5.4, if network F is ε-robust, then $y = \bar{y}$, i.e., for any label k different from $\bar{y}$, $F_k(\mathbf{x}) \leq F_{\bar{y}}(\mathbf{x})$. Therefore, in order to verify the robustness of network F, it is equivalent to check if inequality $F_k(\mathbf{x}) \leq F_{\bar{y}}(\mathbf{x})$ holds for any input $\mathbf{x} \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)$ and label $k \neq \bar{y}$, which leads to the following definition:

Definition 5.5. For network F, given input $\bar{\mathbf{x}} \in \mathbb{R}^{p_0}$ and its prediction $\bar{y}$. The *robustness verification problem* of network F at input $\bar{\mathbf{x}} \in \mathbb{R}^{p_0}$ for perturbation ε, w.r.t. norm $\|\cdot\|$ can be formulated as the following optimization problem: for $k \neq \bar{y}$,

$$\delta_k = \max_{\mathbf{x} \in B(\bar{\mathbf{x}}, \varepsilon, \|\cdot\|)} F_k(\mathbf{x}) - F_{\bar{y}}(\mathbf{x}). \tag{Cert-k}$$

Remark. If $\delta_k < 0$ for all $k \neq \bar{y}$, then F is ε-robust at $\bar{\mathbf{x}}$ w.r.t. norm $\|\cdot\|$.

For DNNs: Suppose we have a pre-trained fully-connected deep ReLU neural network F for K-classifications with parameters $\mathbf{W} = (\mathbf{A}_1, \mathbf{b}_1; \ldots; \mathbf{A}_L, \mathbf{b}_L; \mathbf{C})$ as described in Section 1.1.1. Denote by $\mathbf{x}_0$ the variables in the input layer and $\mathbf{x}_i$ the variables in each hidden layer for $i = 1, \ldots, L$. The value of $\mathbf{x}_i$ is computed by equation $\mathbf{x}_i = \mathrm{ReLU}(\mathbf{A}_i \mathbf{x}_{i-1} + \mathbf{b}_i)$. By Definition 5.5, the robustness verification problem of deep neural network F is written as: for $k \neq \bar{y}$,

$$\begin{aligned} \max_{\mathbf{x}_i \in \mathbb{R}^{p_i}} \quad & F_k(\mathbf{x}_0) - F_{\bar{y}}(\mathbf{x}_0) \\ \text{s.t.} \quad & \begin{cases} \mathbf{x}_i = \mathrm{ReLU}(\mathbf{A}_i \mathbf{x}_{i-1} + \mathbf{b}_i),\ i \in [L], & \text{(ReLU constraint)} \\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \leq \varepsilon. & \text{(Bound constraint)} \end{cases} \end{aligned} \tag{5.34}$$

The output of $\mathbf{x}_0$ is an affine combination of $\mathbf{x}_L$, i.e., $F(\mathbf{x}_0) = \mathbf{C}\mathbf{x}_L$. Denote by $\mathbf{C}^{(k,:)}$ the k-th row vector of matrix $\mathbf{C}$ and let $\mathbf{c}_{k,\bar{y}} := (\mathbf{C}^{(k,:)} - \mathbf{C}^{(\bar{y},:)})^T$, then the objective of (5.34) is simply $\mathbf{c}_{k,\bar{y}}^T \mathbf{x}_L$. Moreover, by the semialgebraicity of ReLU

function (5.1), the ReLU constraint in (5.34) can be formulated as

$$\mathbf{x}_i \circ (\mathbf{x}_i - \mathbf{A}_i\mathbf{x}_{i-1} - \mathbf{b}_i) = 0,\ \mathbf{x}_i - \mathbf{A}_i\mathbf{x}_{i-1} - \mathbf{b}_i \geq 0,\ \mathbf{x}_i \geq 0, \tag{5.35}$$

where $\circ$ denotes the element-wise product, and all the (in)equalities are element-wise. The robustness verification problem for DNNs (5.34) can be then rewritten as: for $k \neq \bar{y}$,

$$\begin{aligned} \max_{\mathbf{x}_i \in \mathbb{R}^{p_i}} \quad & \mathbf{c}_{k,\bar{y}}^T \mathbf{x}_L \qquad \text{(CertDNN-}k\text{)} \\ \text{s.t.} \quad & \begin{cases} \mathbf{x}_i \circ (\mathbf{x}_i - \mathbf{A}_i\mathbf{x}_{i-1} - \mathbf{b}_i) = 0,\ \mathbf{x}_i - \mathbf{A}_i\mathbf{x}_{i-1} - \mathbf{b}_i \geq 0,\ \mathbf{x}_i \geq 0,\ i \in [L], \\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \leq \varepsilon. \end{cases} \end{aligned}$$

For MONs: Now let F be a fully-connected MON activated by ReLU function with parameters $\mathbf{W} = (\mathbf{A}, \mathbf{B}, \mathbf{b}; \mathbf{C})$ as described in Section 1.1.2. Similar with the case of DNN, except that the variables $\mathbf{x}_1$ in the hidden layer is evaluated by fixed-point equation $\mathbf{x}_1 = \mathrm{ReLU}(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b})$. By definition 5.5, the robustness verification problem for MONs is written as: for $k \neq \bar{y}$,

$$\begin{aligned} \max_{\mathbf{x}_i \in \mathbb{R}^{p_i}} \quad & F_k(\mathbf{x}_0) - F_{\bar{y}}(\mathbf{x}_0) \qquad (5.36) \\ \text{s.t.} \quad & \begin{cases} \mathbf{x}_1 = \mathrm{ReLU}(\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_0 + \mathbf{b}), & \text{(ReLU constraint)} \\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \leq \varepsilon. & \text{(Bound constraint)} \end{cases} \end{aligned}$$

Using the semialgebraicity of ReLU function, the ReLU constraint in (5.36) is equivalent to

$$\mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b}) = 0,\ \mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b} \geq 0,\ \mathbf{x}_1 \geq 0. \tag{5.37}$$

Keep the notation $\mathbf{c}_{k,\bar{y}}$, then problem (5.36) is rewritten as: for $k \neq \bar{y}$,

$$\begin{aligned} \max_{\mathbf{x}_i \in \mathbb{R}^{p_i}} \quad & \mathbf{c}_{k,\bar{y}}^T \mathbf{x}_1 \qquad \text{(CertMON-}k\text{)} \\ \text{s.t.} \quad & \begin{cases} \mathbf{x}_1 \circ (\mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b}) = 0,\ \mathbf{x}_1 - \mathbf{A}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_0 - \mathbf{b} \geq 0,\ \mathbf{x}_1 \geq 0, \\ \|\mathbf{x}_0 - \bar{\mathbf{x}}\| \leq \varepsilon. \end{cases} \end{aligned}$$

The number of variables in problem (CertDNN-k) depends on the input layer and each hidden layer. Different from DNNs, we only have one hidden layer in MON. Therefore, the number of variables in problem (CertMON-k) only depends on the number of input neurons p_0 and hidden neurons p_1. Summarizing the discussion above, we show the total number of variables and constraints of robustness verification problem for each case in Table 5.5. Note that for L_2 norm, the bound constraint is equivalent to one dense polynomial constraint; while for L_∞ norm, we have p_0 sparse constraints.

Table 5.5: Summary of verification problem for DNNs and MONs.

Network	Norm	# of variables	# of constraints
DNNs	$\frac{L_2}{L_\infty}$	$\sum_{i=0}^{L} p_i$	$\frac{1+3\sum_{i=1}^{L} p_i}{p_0+3\sum_{i=1}^{L} p_i}$
MONs	$\frac{L_2}{L_\infty}$	p_0+p_1	$\frac{1+3p_1}{p_0+3p_1}$

5.4.2 Algorithms

In Section 5.2 and 5.3, we discuss about the Lipschitz constant estimation problem and ellipsoid propagation problem for both fully-connected DNNs and MONs with ReLU activation function. In fact, Lipschitz constants and overapproximation ellipsoids can also be used to verify robustness of neural networks.

Proposition 5.5. *For neural network $F:\Omega\to\mathbb{R}^K$, where $\Omega=B(\bar{\mathbf{x}},\varepsilon,\|\cdot\|)\subseteq\mathbb{R}^{p_0}$ and $\bar{\mathbf{x}}\in\Omega$ is a fixed point. Let $\bar{y}$ be the prediction of $\bar{\mathbf{x}}$ by network F. For $k\neq\bar{y}$:*

- *Let δ_k^{cert} be the optimal value of problem (Cert-k);*
- *Let $L_{F,\varepsilon,\|\cdot\|}$ be the local Lipschitz constant of F. Define $\delta^{lip}:=\varepsilon\cdot L_{F,\varepsilon,\|\cdot\|}$ and $\tau:=F_{\bar{y}}(\bar{\mathbf{x}})-\max_{k\neq\bar{y}}F_k(\bar{\mathbf{x}})$;*
- *Let $\mathcal{E}=\{\mathbf{x}\in\mathbb{R}^K:\|\mathbf{Q}\mathbf{x}+\mathbf{b}\|_2\leq 1\}$, where $\mathbf{Q}\in\mathbb{R}^{K\times K}$ and $\mathbf{b}\in\mathbb{R}^K$, be the overapproximation ellipsoid given by (Ellip). Define*

$$\delta_k^{ellip}:=\max_{\mathbf{x}\in\mathbb{R}^K}\{x^{(i)}-x^{(\bar{y})}:\|\mathbf{Q}\mathbf{x}+\mathbf{b}\|_2\leq 1\}. \tag{5.38}$$

If one of the following assertions holds:
(1) $\delta_k^{cert}<0$ for all $k\neq\bar{y}$;
(2) $2\delta^{lip}<\tau$;
(3) $\delta_k^{ellip}<0$ for all $k\neq\bar{y}$.
Then the network F is ε-robust at $\bar{\mathbf{x}}$ w.r.t. norm $\|\cdot\|$.

Proof : (1) is trivial by Definition 5.5.
(2) For a given input $\bar{\mathbf{x}}\in\mathbb{R}^{p_0}$ and any $\mathbf{x}_0\in B(\bar{\mathbf{x}},\varepsilon,\|\cdot\|)$, we have

$$\|F(\mathbf{x}_0)-F(\bar{\mathbf{x}})\|\leq L_{F,\varepsilon,\|\cdot\|}\cdot\|\mathbf{x}_0-\bar{\mathbf{x}}\|\leq\varepsilon L_{F,\varepsilon,\|\cdot\|}\leq\varepsilon L_{F,\|\cdot\|}\,, \tag{5.39}$$
$$\|F(\mathbf{x}_0)-F(\bar{\mathbf{x}})\|\leq|||\mathbf{C}|||\cdot L_{\mathbf{x}_1,\varepsilon,\|\cdot\|}\cdot\|\mathbf{x}_0-\bar{\mathbf{x}}\|\leq\varepsilon|||\mathbf{C}|||\cdot L_{\mathbf{x}_1,\varepsilon,\|\cdot\|}\,. \tag{5.40}$$

By equations (5.39), (5.40), and using the fact that $\|\mathbf{x}\|_\infty\leq\|\mathbf{x}\|$ for all $\mathbf{x}\in\mathbb{R}^K$, we have

$$\|F(\mathbf{x}_0)-F(\bar{\mathbf{x}})\|_\infty\leq\|F(\mathbf{x}_0)-F(\bar{\mathbf{x}})\|\leq L_{F,\varepsilon,\|\cdot\|}\cdot\|\mathbf{x}_0-\bar{\mathbf{x}}\|\leq\delta^{lip}. \tag{5.41}$$

Hence, for $k \neq \bar{y}$, by definition of δ_k^{cert},

$$\begin{aligned}
\delta_k^{cert} &= \max_{\mathbf{x}_0 \in \Omega} \{F_k(\mathbf{x}_0) - F_{\bar{y}}(\mathbf{x}_0)\} \\
&= \max_{\mathbf{x}_0 \in \Omega} \{F_k(\mathbf{x}_0) - F_k(\bar{\mathbf{x}}) + F_k(\bar{\mathbf{x}}) - F_{\bar{y}}(\bar{\mathbf{x}}) + F_{\bar{y}}(\bar{\mathbf{x}}) - F_{\bar{y}}(\mathbf{x}_0)\} \\
&\leq \max_{\mathbf{x}_0 \in \Omega} \{|F_k(\mathbf{x}_0) - F_k(\bar{\mathbf{x}})| + \max_{k \neq \bar{y}} \{F_k(\bar{\mathbf{x}})\} - F_{\bar{y}}(\bar{\mathbf{x}}) + |F_{\bar{y}}(\bar{\mathbf{x}}) - F_{\bar{y}}(\mathbf{x}_0)|\} \\
&\leq \max_{\mathbf{x}_0 \in \Omega} \{\|F(\mathbf{x}_0) - F(\bar{\mathbf{x}})\|_\infty - \tau + \|F(\mathbf{x}_0) - F(\bar{\mathbf{x}})\|_\infty\} \\
&\leq 2\delta^{lip} - \tau < 0, \qquad (5.42)
\end{aligned}$$

where the last inequality is due to (5.41). By Proposition 5.5 (1), F is ε-robust.

(3) The overapproximation ellipsoid $\mathcal{E}$ contains $F(\Omega)$, i.e., $F(\mathbf{x}_0) \in \mathcal{E}$ for all $\mathbf{x}_0 \in \mathbb{R}^{p_0}$. If $\delta_k^{ellip} < 0$ for all $k \neq \bar{y}$, then $F_k(\mathbf{x}_0) - F_{\bar{y}}(\mathbf{x}) < 0$ for all $k \neq \bar{y}$ and $\mathbf{x}_0 \in \mathbb{R}^{p_0}$. Thus $\delta_k^{cert} < 0$ for all $k \neq \bar{y}$ and F is ε-robust. □

For DNNs, we consider the Lipschitz constant estimation problem (LipDNN-k) and apply Proposition 5.5 (2) to certify robustness. For MONs, we consider all the three problems (robustness verification problem, Lipschitz constant estimation problem, ellipsoid propagation problem) and apply Proposition 5.5 (1) (2) (3) to certify robustness. The authors in [Pabbaraju *et al.* 2021] use inequality (5.40) with L_2 norm to certify robustness of MONs, as they provide an upper bound of $L_{\mathbf{x}_1,\varepsilon,\|\cdot\|_2}$. In contrast, our formulation (LipMON) provides upper bounds on Lipschitz constants of F or $\mathbf{x}_1$ for arbitrary L_q norms. We directly focus on estimating the value of $L_{F,\varepsilon,\|\cdot\|_q}$ instead of $L_{\mathbf{x}_1,\varepsilon,\|\cdot\|_q}$. In fact the quantity $\|\mathbf{C}\|_q \cdot L_{\mathbf{x}_1,\varepsilon,\|\cdot\|_q}$ can be regarded as an upper bound of $L_{F,\varepsilon,\|\cdot\|_q}$, the local Lipschitz constant of F. Therefore, we are able to certify more examples using $\mathbf{SHOR}_F^q$ than $\mathbf{Pab}_F^q$, see Table 5.8.

Remark. Shor's relaxation of Lipschitz constant estimation problems for DNN has already been extensively investigated in [Fazlyab *et al.* 2019b, Chen *et al.* 2020]. If one want to certify robustness for several input test examples, one may choose the input region Ω to be a big ball containing all such examples with an additional margin of ε. Choosing a big ball for all input points requires to solve only one optimization problem, while choosing one ball for each input point (say among N) requires to solve N optimization problems, which is much more costly. In this case, it is more favorable to apply the certification problem (CertMON-k) directly.

5.4.3 Experiments

In this section, we use the same notation of algorithms and neural networks described in Section 5.2.2 and 5.3.2. We verify the robustness of DNNs by estimating their Lipschitz constants, and apply all the three approaches to MONs to compare the effectiveness of each method. All experiments are performed on a personal laptop with an Intel 8-Core i7-8665U CPU @ 1.90GHz Ubuntu 18.04.5 LTS, 32GB RAM.

Trained FcDNN$_1$ for 2 labels: 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 ReLU network of size (80, 80) 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 $\bar{\mathbf{x}} \in \mathbb{R}^{80}$ and a perturbation $\varepsilon > 0$, we compute an upper bound of the local Lipschitz constant $L_{F,\varepsilon,\|\cdot\|_\infty}$ w.r.t. L_∞ norm. Denote by $\bar{y}$ the output of $\bar{\mathbf{x}}$, by Proposition 5.5, if the output $\bar{y}$ is negative (resp. positive) and $\bar{y} + \varepsilon L_{F,\varepsilon,\|\cdot\|_\infty} < 0$ (resp. $\bar{y} - \varepsilon L_{F,\varepsilon,\|\cdot\|_\infty} > 0$), then F is ε-robust at $\bar{\mathbf{x}}$ w.r.t. L_∞ norm. With this criteria, if we have n inputs to verify, we need to execute n experiments. However, for large networks based on MNIST dataset, this is impractical, even if we verify the inputs directly without using Lipschitz constants (see [Raghunathan *et al.* 2018a]). Therefore, we compute the Lipschitz constant with respect to $\bar{\mathbf{x}} = \mathbf{0}$ and $\varepsilon = 3$, denoted by L, as an upper bound of the local Lipschitz constants. Then we generate 10^6 random points in the box $\mathbf{B} = \{\mathbf{x} \in \mathbb{R}^{80} : \|\mathbf{x}\|_\infty \leq 2.9\}$. For any $\varepsilon \leq 0.1$ and $\bar{\mathbf{x}} \in \mathbf{B}$, we have $L_{F,\varepsilon,\|\cdot\|_\infty} \leq L$, then F is ε-robust at $\bar{\mathbf{x}}$ if $\bar{y}(\bar{y} - \text{sign}(\bar{y})\varepsilon L) > 0$. Instead of running 10^6 times the algorithm, we are able to verify robustness of large number of inputs by only doing one experiment. Table 5.6 shows the ratio of verified inputs in the box $\mathbf{B}$ by **Sub-2** and **LipOpt-3**. We see that **Sub-2** can always verify more examples than **LipOpt-3**.

Table 5.6: Ratios of verified examples for $(80, 80)$ network.

ε	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Sub-2	87.51%	75.02%	62.46%	49.89%	37.22%	24.36%	8.15%	2.75%	0.76%	0.12%
LipOpt-3	69.03%	37.84%	4.78%	0.15%	0%	0%	0%	0%	0%	0%

Trained FcDNN$_1$ for MNIST: The SDP-NN network is a well-trained network of size $(784, 500)$ to classify the handwritten digit images from 0 to 9. Denote the parameters of this network by $\mathbf{A}_1 \in \mathbb{R}^{500\times784}, \mathbf{b}_1 \in \mathbb{R}^{500}, \mathbf{C} \in \mathbb{R}^{10\times500}$. Denote by $\bar{y}$ the prediction of input $\bar{\mathbf{x}} \in \mathbb{R}^{784}$. Let $\varepsilon > 0$, if for all $k \neq \bar{y}$ and $\mathbf{x}_0 \in \mathbb{R}^{784}$ such that $\|\mathbf{x}_0 - \bar{\mathbf{x}}\|_\infty \leq \varepsilon$, $F_k(\mathbf{x}_0) - F_{\bar{y}}(\mathbf{x}_0) < 0$, then F is ε-robust at $\bar{\mathbf{x}}$ by Proposition 5.5 (1). Alternatively, denote by $L_{f_{k,\bar{y}},\varepsilon,\|\cdot\|_\infty}$ the local Lipschitz constant of function $f_{k,\bar{y}}(\mathbf{x}_0) := F_k(\mathbf{x}_0) - F_{\bar{y}}(\mathbf{x}_0)$ with respect to L_∞ norm in the ball $\{\mathbf{x} \in \mathbb{R}^{784} : \|\mathbf{x} - \bar{\mathbf{x}}\|_\infty \leq \varepsilon\}$. If for all $k \neq \bar{y}$, $f_{k,\bar{y}}(\bar{\mathbf{x}}) + \varepsilon L_{f_{k,\bar{y}},\varepsilon,\|\cdot\|_\infty} < 0$. Then F is ε-robust at $\bar{\mathbf{x}}$ by Proposition 5.5 (2). Since the 28×28 MNIST images are flattened and normalized into vectors taking value in $[0, 1]$, we compute the local Lipschitz constant by **Sub-2** with respect to $\bar{\mathbf{x}} = \mathbf{0}$ and $\varepsilon = 2$. Define matrix $\mathbf{L} := [L_{f_{i,j},\varepsilon,\|\cdot\|_\infty}]_{i,j}$ for $i \neq j$, the

complete matrix $\mathbf{L}$ (without diagonal entries) is as follows:

$$\mathbf{L} = \begin{pmatrix} * & 7.94 & 7.89 & 8.28 & 8.64 & 8.10 & 7.66 & 8.04 & 7.46 & 8.14 \\ 7.94 & * & 7.74 & 7.36 & 7.68 & 8.81 & 8.06 & 7.55 & 7.36 & 8.66 \\ 7.89 & 7.74 & * & 7.63 & 8.81 & 10.23 & 8.18 & 8.13 & 7.74 & 9.08 \\ 8.28 & 7.36 & 7.63 & * & 8.52 & 7.74 & 9.47 & 8.01 & 7.37 & 7.96 \\ 8.64 & 7.68 & 8.81 & 8.52 & * & 9.44 & 7.98 & 8.65 & 8.49 & 7.47 \\ 8.10 & 8.81 & 10.23 & 7.74 & 9.44 & * & 8.26 & 9.26 & 8.17 & 8.55 \\ 7.66 & 8.06 & 8.18 & 9.47 & 7.98 & 8.26 & * & 10.18 & 8.00 & 9.83 \\ 8.04 & 7.55 & 8.13 & 8.01 & 8.65 & 9.26 & 10.18 & * & 8.28 & 7.65 \\ 7.46 & 7.36 & 7.74 & 7.37 & 8.49 & 8.17 & 8.00 & 8.28 & * & 7.87 \\ 8.14 & 8.66 & 9.08 & 7.96 & 7.47 & 8.55 & 9.83 & 7.65 & 7.87 & * \end{pmatrix}. \tag{5.43}$$

Note that if we replace the vector $\mathbf{c}_k$ in (LipDNN-k) by $-\mathbf{c}_k$, the problem is equivalent to the original one. Therefore, the matrix $\mathbf{L}$ is symmetric, and we only need to compute the upper triangle of $\mathbf{L}$.

We take different values of ε 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 5.7. Note that for $\varepsilon = 0.1$, we improve a little bit by 67% compared to **Grad-cert** (65%) proposed by [Raghunathan *et al.* 2018a], as we use an exact formulation of the subdifferential of ReLU function.

Table 5.7: Ratios of verified test inputs for SDP-NN by **Sub-2**.

ε	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Ratios	98.80%	97.24%	95.16%	92.84%	90.18%	87.10%	83.01%	78.34%	73.54%	67.63%

Trained FcMON for MNIST: The MON we use is the same as in Section 5.2.3. Consider $\varepsilon = 0.1$ for L_2 norm and $\varepsilon = 0.1, 0.05, 0.01$ for L_∞ norm. For each verification approach described in Proposition 5.5, we compute the ratio of certified test examples among the first 100 test inputs and the average computation time (with unit second) for one example of each method. The ratio in parentheses of the column "Lipschitz Model" are computed by the Lipschitz constant given in [Pabbaraju *et al.* 2021] (see Section 5.2.2 for details). Exact binomial 95% confidence intervals are given in bracket. Following [Pabbaraju *et al.* 2021], we also compute the *projected gradient descent (PGD)* attack accuracy using Foolbox library [Rauber *et al.* 2020], which indicates the ratio of non-successful attacks among our 100 inputs. Note that the ratio of verified examples should always be less or equal than the ratio of non-successful attacks. The gaps between them shows how many test inputs there are for which we are neither able to verify robustness nor find adversarial attacks.

From Table 5.8, we see that the MON we consider is robust to all the 100 test examples for L_2 norm and $\varepsilon = 0.1$ (the only example that we can not verify is because the label itself is wrong). However, it is not robust for L_∞ norm at the same level of perturbation (all our three verification approaches cannot certify any examples), and the PGD algorithm finds adversarial examples for 85% of the inputs. The network becomes robust again for the L_∞ norm when we reduce the

perturbation ε to 0.01. Overall, we see that verification approach is the one with best performance as it provides the highest ratio, ellipsoid approach is the second best model compared to verification approach, and Lipschitz approach provides the lowest ratio. As a trade-off, for each test example, verification approach requires to consider at most 9 optimization problems, each one being solved in around 150 seconds. While ellipsoid approach requires to consider only one problem, which is solved in around 500 seconds. We only need to calculate one (global) Lipschitz constant, which takes around 1500 seconds, so that we are able to certify any number of inputs. Each approach we propose provide better or equal certification accuracy compared to [Pabbaraju *et al.* 2021], and significant improvements for L_∞ perturbations.

Table 5.8: Ratio of verified test inputs and running time.

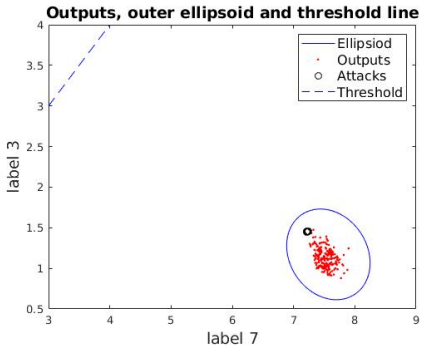
Norm	ε	Verification approach (1350s / example)	Lipschitz approach (1500s in total)	Ellipsoid approach (500s / example)	PGD Attack
L_2	0.1	99% [>94]	91% (91%) [>83]	99%[>94]	99%[>94]
L_∞	0.1	0% [<4]	0% (0%) [<4]	0%[<4]	15% [8, 24]
	0.05	24% [16, 34]	0% (0%) [<4]	0% [<4]	82% [73, 89]
	0.01	99% [>94]	24% [16, 34] (0%) [<4]	92% [>84]	99% [>94]

Remark. The ellipsoid approach for robustness verification has a geometric explanation: for $k \neq \bar{y}$, denote by $\mathcal{P}_k$ the projection map from output space $\mathbb{R}^K$ to its 2-dimensional subspace $\mathbb{R}_{\bar{y}} \times \mathbb{R}_k$, i.e., $\mathcal{P}_k(\mathbf{x}) = [x^{(\bar{y})}, x^{(k)}]^T$ for all $\mathbf{x} \in \mathbb{R}^K$. Let $\mathcal{L}_i$ be the line in subspace $\mathbb{R}_{\bar{y}} \times \mathbb{R}_k$ defined by $\{[x^{(\bar{y})}, x^{(k)}]^T \in \mathbb{R}_{\bar{y}} \times \mathbb{R}_k : x^{(\bar{y})} = x^{(k)}\}$. Then the network F is ε-robust at $\bar{\mathbf{x}}$ if the projection $\mathcal{P}_k(\mathcal{E})$ lies strictly below the line $\mathcal{L}_k$ for all $k \neq \bar{y}$.

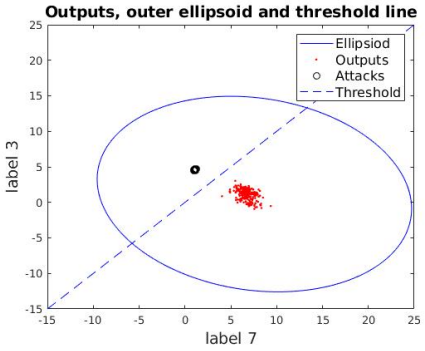
Take the first MNIST test example (which is classified as 7) for illustration. For $\varepsilon = 0.1$, this input is certified to be robust for L_2 norm but not for L_∞ norm. We show the landscape of the projections onto $\mathbb{R}_7 \times \mathbb{R}_3$, i.e., the x-axis indicates label 7 and the y-axis indicates label 3. In Figure 5.6, the red points are projections of points in the image $F(\Omega)$, the black circles are projections of some (successful and unsuccessful) adversarial examples found by the PGD algorithm. Notice that the adversarial examples also lie in the image $F(\Omega)$. The blue curve is the boundary of the projection of the overapproximation ellipsoid (which is an ellipse), and the blue dashed line plays the role of a certification threshold. Figure 5.6a shows the landscape for L_2 norm, we see that the ellipse lies strictly below the threshold line, which means that for all points $\mathbf{x} \in \mathcal{E}$, we have $x_3 < x_7$. Hence for all $\mathbf{x} \in F(\Omega)$, we also have $x_3 < x_7$. On the other hand, for L_∞ norm, we see from Figure 5.6b that the threshold line crosses the ellipse, which means that we are not able to verify robustness of this example using ellipsoid approach. Indeed, we can find adversarial examples with the PGD algorithm, as shown in Figure 5.6b by the black circles that lie above the threshold line.

Figure 5.7 is the visualization of one of the attack examples. The picture on the left is the original digit which is corrected classified as 7. The picture on the right

is the attack example found by PGD algorithm, which looks also the digit 7 but is incorrectly classified as 3.

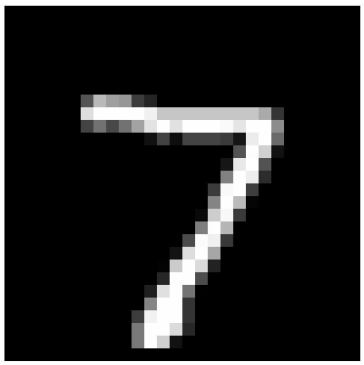


(a) Verified example for L_2 norm.



(b) Unverified example for L_∞ norm.

Figure 5.6: Visualization of the overapproximation ellipsoids and output region.



(a) Original example, classified as 7.

(b) Adversarial example, classified as 3.

Figure 5.7: An adversarial example of the first MNIST test example.

Conclusions and Future Works

In this thesis, we first develop a general sublevel moment-SOS hierarchy as an alternative to the standard Lasserre hierarchy. We show that the sublevel hierarchy is not only useful for optimization problems, but also provides new methodologies to verify neural networks. Even though the ReLU function and its subdifferential are semialgebraic, as shown in (5.1) and (5.3), optimization problems derived from neural networks are generally not sparse. Therefore, polynomial optimization approaches exploiting parsimony (such as Lasserre relaxation exploiting correlative sparsity, TSSOS exploiting term sparsity) do not work. The sublevel hierarchy gives possible intermediate relaxations for dense or nearly-dense POPs.

We also present three general models for verifying the robustness of neural networks: Lipschitz constant estimation, ellipsoidal propagation, and robustness verification. All three models can be applied to verify robustness by Proposition 5.5. There is a trade-off between accuracy and efficiency among these approaches. For robustness verification, this is an exact model describing the problem. However, we need to run N experiments if we want to verify N inputs. On the contrary, we only need to calculate a single Lipschitz constant to verify all inputs located in the corresponding input region. The Lipschitz constant is also important for robust training, and ellipsoidal propagation can be generalized to abstract domain propagation, which is widely used in dynamical systems.

Our approaches are based on the semialgebraic representation of the ReLU function, which limits our approach to the case of ReLU neural networks. Moreover, the efficiency of our approaches depends on the power of the SDP solvers. Therefore, our methods cannot be applied to large networks such as ResNet, LeNet, etc. Even though sublevel relaxation shows promising improvements for small networks. When the number of neurons increases, even Shor's relaxation cannot apply. Models and algorithms for large-scale and convolutional neural networks are interesting future topics of this thesis. Moreover, if one develops a highly efficient first-order SDP solver in place of interior point algorithms, our SDP-based approaches may be applicable to larger and more complicated cases. The design of first-order SDP algorithms is also a valuable future topic.

All in all, robustness verification is indeed a difficult problem both theoretically and practically. The approaches proposed in this thesis give only one direction to deal with them. There are many unresolved difficulties and future topics to overcome and investigate.

Bibliography

[Akintunde *et al.* 2018] Michael Akintunde, Alessio Lomuscio, Lalit Maganti and Edoardo Pirovano. *Reachability Analysis for Neural Agent-Environment Systems.* In Michael Thielscher, Francesca Toni and Frank Wolter, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018, pages 184–193. AAAI Press, 2018. (Cited on pages 15 and 24.)

[Albarghouthi 2021] Aws Albarghouthi. *Introduction to Neural Network Verification.* Foundations and Trends® in Programming Languages, vol. 7, no. 1–2, pages 1–157, 2021. (Cited on page 14.)

[Antun *et al.* 2021] Vegard Antun, Nina M. Gottschling, Anders C. Hansen and Ben Adcock. *Deep Learning in Scientific Computing: Understanding the Instability Mystery.* SIAM NEWS MARCH 2021, 2021. (Cited on page 8.)

[Athalye *et al.* 2018] Anish Athalye, Logan Engstrom, Andrew Ilyas and Kevin Kwok. *Synthesizing Robust Adversarial Examples.* In International conference on machine learning, pages 284–293. PMLR, 2018. (Cited on pages xxii and xxvi.)

[Bai *et al.* 2019] Shaojie Bai, Zico Kolter and Vladlen Koltun. *Deep Equilibrium Models.* In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019. (Cited on pages xxii, xxvi and 3.)

[Baluta *et al.* 2021] T. Baluta, Z. Chua, K. S. Meel and P. Saxena. *Scalable Quantitative Verification for Deep Neural Networks.* In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), pages 248–249, Los Alamitos, CA, USA, may 2021. IEEE Computer Society. (Cited on page 84.)

[Barthel & Hübener 2012] Thomas Barthel and Robert Hübener. *Solving Condensed-Matter Ground-State Problems by Semidefinite Relaxations.* Physical Review Letters, vol. 108, no. 20, May 2012. (Cited on page 36.)

[Bastani *et al.* 2016] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori and Antonio Criminisi. *Measuring Neural Net Robustness with Constraints.* In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc., 2016. (Cited on pages 22 and 24.)

[Bezanson *et al.* 2017] Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B Shah. *Julia: A Fresh Approach to Numerical Computing.* SIAM review, vol. 59, no. 1, pages 65–98, 2017. (Cited on page 73.)

[Bibi *et al.* 2019] Adel Bibi, Bernard Ghanem, Vladlen Koltun and Rene Ranftl. *Deep Layers as Stochastic Solvers.* In International Conference on Learning Representations, 2019. (Cited on page 4.)

[Bolte & Pauwels 2021] Jérôme Bolte and Edouard Pauwels. *Conservative Set Valued Fields, Automatic Differentiation, Stochastic Gradient Methods and Deep Learning.* Mathematical Programming, vol. 188, no. 1, pages 19–51, 2021. (Cited on pages 60, 61 and 64.)

[Boopathy *et al.* 2019] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu and Luca Daniel. *CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks.* In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 3240–3247, 2019. (Cited on pages xxii, xxvii, 10, 21, 24, 58 and 84.)

[Brown *et al.* 2022] Robin Brown, Edward Schmerling, Navid Azizan and Marco Pavone. *A Unified View of SDP-based Neural Network Verification through Completely Positive Programming*, 2022. (Cited on page 20.)

[Bunel *et al.* 2018] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli and Pawan K Mudigonda. *A Unified View of Piecewise Linear Neural Network Verification.* In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. (Cited on pages 15, 16, 21 and 24.)

[Campos *et al.* 2022] Juan S. Campos, Ruth Misener and Panos Parpas. *Partial Lasserre Relaxation for Sparse Max-Cut.* Optimization and Engineering, Aug 2022. (Cited on pages 36, 42, 44 and 47.)

[Chen *et al.* 2020] Tong Chen, Jean-Bernard Lasserre, Victor Magron and Edouard Pauwels. *Semialgebraic Optimization for Lipschitz Constants of ReLU Networks.* In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 19189–19200. Curran Associates, Inc., 2020. (Cited on pages xxiii, xxviii, 42 and 88.)

[Chen *et al.* 2021] Tong Chen, Jean-Bernard Lasserre, Victor Magron and Edouard Pauwels. *Semialgebraic Representation of Monotone Deep Equilibrium Models and Applications to Certification.* In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, volume 34, pages 27146–27159. Curran Associates, Inc., 2021. (Cited on pages xxiii and xxviii.)

[Chen *et al.* 2022] Tong Chen, Jean-Bernard Lasserre, Victor Magron and Edouard Pauwels. *A Sublevel Moment-SOS Hierarchy for Polynomial Optimization.* Computational Optimization and Applications, vol. 81, no. 1, pages 31–66, 2022. (Cited on pages xxiii, xxvii and xxviii.)

[Cheng *et al.* 2017] Chih-Hong Cheng, Georg Nührenberg and Harald Ruess. *Maximum Resilience of Artificial Neural Networks.* CoRR, vol. abs/1705.01040, 2017. (Cited on pages 15 and 24.)

[Clarke 1983] F. H. Clarke. Optimization and Nonsmooth Analysis. Wiley New York, 1983. (Cited on pages 61, 63, 64 and 65.)

[Combettes & Pesquet 2020] Patrick L. Combettes and Jean-Christophe Pesquet. *Lipschitz Certificates for Layered Network Structures Driven by Averaged Activation Operators.* SIAM Journal on Mathematics of Data Science, vol. 2, no. 2, pages 529–557, 2020. (Cited on page 58.)

[Cousot & Cousot 1977] Patrick Cousot and Radhia Cousot. *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* In Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, POPL '77, page 238–252, New York, NY, USA, 1977. Association for Computing Machinery. (Cited on page 17.)

[Cybenko 1989] G. Cybenko. *Approximation by Superpositions of a Sigmoidal Function.* Mathematics of Control, Signals and Systems, vol. 2, no. 4, pages 303–314, Dec 1989. (Cited on pages xxi, xxv and 1.)

[Dathathri *et al.* 2020] Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang and Pushmeet Kohli. *Enabling Certification of Verification-Agnostic Networks via Memory-Efficient Semidefinite Programming.* In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 5318–5331. Curran Associates, Inc., 2020. (Cited on pages xxiii, xxviii, 58 and 84.)

[de Moura & Bjørner 2008] Leonardo de Moura and Nikolaj Bjørner. *Z3: An Efficient SMT Solver.* In C. R. Ramakrishnan and Jakob Rehof, editors, Tools and Algorithms for the Construction and Analysis of Systems, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. (Cited on page 16.)

[Dutta *et al.* 2018] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan and Ashish Tiwari. *Output Range Analysis for Deep Feedforward Neural Networks.* In Aaron Dutle, César Muñoz and Anthony Narkawicz, editors,

NASA Formal Methods, pages 121–138, Cham, 2018. Springer International Publishing. (Cited on pages 15, 17 and 24.)

[Dvijotham *et al.* 2018] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann and Pushmeet Kohli. *A Dual Approach to Scalable Verification of Deep Networks.* In UAI, volume 1, page 3, 2018. (Cited on pages 19, 24 and 58.)

[Ehlers 2017] Rüdiger Ehlers. *Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks.* In ATVA, 2017. (Cited on pages xxii, xxvii, 13, 14 and 24.)

[Fazlyab *et al.* 2019a] Mahyar Fazlyab, Manfred Morari and George J. Pappas. *Probabilistic Verification and Reachability Analysis of Neural Networks via Semidefinite Programming.* In 2019 IEEE 58th Conference on Decision and Control (CDC), pages 2726–2731, 2019. (Cited on page 82.)

[Fazlyab *et al.* 2019b] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari and George Pappas. *Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks.* In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019. (Cited on pages 58 and 88.)

[Fazlyab *et al.* 2020] Mahyar Fazlyab, Manfred Morari and George J. Pappas. *Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming.* IEEE Transactions on Automatic Control, 2020. (Cited on pages 58 and 76.)

[Floudas & Pardalos 1990] Christodoulos A. Floudas and Panos M. Pardalos. A collection of test problems for constrained global optimization algorithms, volume 455. Springer Science & Business Media, 1990. (Cited on page 41.)

[Furini *et al.* 2019] Fabio Furini, Emiliano Traversi, Pietro Belotti, Antonio Frangioni, Ambros Gleixner, Nick Gould, Leo Liberti, Andrea Lodi, Ruth Misener, Hans Mittelmann*et al. QPLIB: a Library of Quadratic Programming Instances.* Mathematical Programming Computation, vol. 11, no. 2, pages 237–265, 2019. (Cited on page 45.)

[Gehr *et al.* 2018] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri and Martin Vechev. *AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation.* In 2018 IEEE Symposium on Security and Privacy (SP), pages 3–18, 2018. (Cited on pages xxii, xxvii, 18, 24, 76 and 84.)

[Goodfellow *et al.* 2015] Ian Goodfellow, Jonathon Shlens and Christian Szegedy. *Explaining and Harnessing Adversarial Examples.* In International Conference on Learning Representations, 2015. (Cited on pages xiii, 8 and 84.)

[Goodfellow 2018] Ian J. Goodfellow. *Gradient Masking Causes CLEVER to Overestimate Adversarial Perturbation Size*. CoRR, vol. abs/1804.07870, 2018. (Cited on page 22.)

[Grant & Boyd 2014] Michael Grant and Stephen Boyd. *CVX: Matlab Software for Disciplined Convex Programming, version 2.1*, 2014. (Cited on page 83.)

[Haim *et al.* 2020] Arbel Haim, Richard Kueng and Gil Refael. *Variational-Correlations Approach to Quantum Many-body Problems*. arXiv preprint arXiv:2001.06510, 2020. (Cited on page 36.)

[He *et al.* 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Deep Residual Learning for Image Recognition*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. (Cited on page 13.)

[Henrion & Lasserre 2005] Didier Henrion and Jean-Bernard Lasserre. *Detecting Global Optimality and Extracting Solutions in GloptiPoly*. In Positive polynomials in control, pages 293–310. Springer, 2005. (Cited on page 40.)

[Henrion *et al.* 2009] Didier Henrion, Jean-Bernard Lasserre and Johan Löfberg. *GloptiPoly 3: Moments, Optimization and Semidefinite Programming*. Optimization Methods & Software, vol. 24, no. 4-5, pages 761–779, 2009. (Cited on page 41.)

[Hornik *et al.* 1989] Kurt Hornik, Maxwell Stinchcombe and Halbert White. *Multilayer Feedforward Networks are Universal Approximators*. Neural Networks, vol. 2, no. 5, pages 359–366, 1989. (Cited on pages xxi and xxv.)

[Hornik 1991] Kurt Hornik. *Approximation Capabilities of Multilayer Feedforward Networks*. Neural Networks, vol. 4, no. 2, pages 251–257, 1991. (Cited on pages xxii, xxvi and 1.)

[Howard *et al.* 2017] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. CoRR, vol. abs/1704.04861, 2017. (Cited on page 13.)

[Hu *et al.* 2020] Haimin Hu, Mahyar Fazlyab, Manfred Morari and George J Pappas. *Reach-SDP: Reachability Analysis of Closed-Loop Systems with Neural Network Controllers via Semidefinite Programming*. In 2020 59th IEEE Conference on Decision and Control (CDC), pages 5929–5934. IEEE, 2020. (Cited on pages 67 and 76.)

[Huang *et al.* 2017a] Gao Huang, Zhuang Liu, Laurens Van Der Maaten and Kilian Q. Weinberger. *Densely Connected Convolutional Networks*. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2261–2269, 2017. (Cited on pages 13 and 22.)

[Huang *et al.* 2017b] Xiaowei Huang, Marta Kwiatkowska, Sen Wang and Min Wu. *Safety Verification of Deep Neural Networks.* In International conference on computer aided verification, pages 3–29. Springer, 2017. (Cited on pages 16 and 24.)

[Huang *et al.* 2020] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu and Xinping Yi. *A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, testing, adversarial attack and defence, and interpretability.* Computer Science Review, vol. 37, page 100270, 2020. (Cited on pages xxii, xxvii and 14.)

[Huster *et al.* 2018] Todd Huster, Cho-Yu Jason Chiang and Ritu Chadha. *Limitations of the Lipschitz Constant as a Defense Against Adversarial Examples.* In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 16–29. Springer, 2018. (Cited on page 58.)

[Josz & Molzahn 2018] Cédric Josz and Daniel K. Molzahn. *Lasserre Hierarchy for Large Scale Polynomial Optimization in Real and Complex Variables.* SIAM Journal on Optimization, vol. 28, no. 2, pages 1017–1048, 2018. (Cited on pages 36 and 42.)

[Kakade & Lee 2018] Sham M Kakade and Jason D Lee. *Provably Correct Automatic Subdifferentiation for Qualified Programs.* In Advances in neural information processing systems, pages 7125–7135, 2018. (Cited on page 60.)

[Katz *et al.* 2017] Guy Katz, Clark Barrett, David L Dill, Kyle Julian and Mykel J Kochenderfer. *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks.* In International conference on computer aided verification, pages 97–117. Springer, 2017. (Cited on pages xxii, xxvii, 13, 14, 24 and 84.)

[Klep *et al.* 2021] Igor Klep, Victor Magron and Janez Povh. *Sparse Noncommutative Polynomial Optimization.* Mathematical Programming, Jan 2021. (Cited on page 36.)

[Kochenberger *et al.* 2013] Gary A Kochenberger, Jin-Kao Hao, Zhipeng Lü, Haibo Wang and Fred Glover. *Solving Large Scale Max-Cut Problems via Tabu Search.* Journal of Heuristics, vol. 19, no. 4, pages 565–571, 2013. (Cited on page 46.)

[Kolter & Madry 2018] Zico Kolter and Aleksander Madry. *Adversarial Robustness - Theory and Practice.* https://adversarial-ml-tutorial.org/, 2018. (Cited on pages xxii, xxvi and 8.)

[Kousik *et al.* 2022] Shreyas Kousik, Adam Dai and Grace X. Gao. *Ellipsotopes: Uniting Ellipsoids and Zonotopes for Reachability Analysis and Fault Detection.* IEEE Transactions on Automatic Control, pages 1–13, 2022. (Cited on page 76.)

[Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks.* In F. Pereira, C.J. Burges, L. Bottou and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012. (Cited on page 13.)

[Lasserre *et al.* 2017] Jean-Bernard Lasserre, Kim-Chuan Toh and Shouguang Yang. *A Bounded Degree SOS hierarchy for Polynomial Optimization.* EURO Journal on Computational Optimization, vol. 5, no. 1-2, pages 87–117, 2017. (Cited on page 35.)

[Lasserre 2001] Jean-Bernard Lasserre. *Global Optimization with Polynomials and the Problem of Moments.* SIAM Journal on optimization, vol. 11, no. 3, pages 796–817, 2001. (Cited on pages xxiii, xxvii, 31 and 41.)

[Lasserre 2006] Jean-Bernard Lasserre. *Convergent SDP-Relaxations in Polynomial Optimization with Sparsity.* SIAM Journal on Optimization, vol. 17, no. 3, pages 822–843, 2006. (Cited on pages xxiii, xxvii, 33, 36, 40 and 42.)

[Latorre *et al.* 2020] Fabian Latorre, Paul Rolland and Volkan Cevher. *Lipschitz Constant Estimation of Neural Networks via Sparse Polynomial Optimization.* In International Conference on Learning Representations, 2020. (Cited on pages xxiii, xxviii, 58, 62, 72 and 73.)

[Laurent 2003] Monique Laurent. *A Comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 Programming.* Mathematics of Operations Research, vol. 28, no. 3, pages 470–496, 2003. (Cited on page 35.)

[Lecun *et al.* 1998] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner. *Gradient-Based Learning Applied to Document Recognition.* Proceedings of the IEEE, vol. 86, no. 11, pages 2278–2324, 1998. (Cited on page 13.)

[Lecun 1988] Yann Lecun. *A Theoretical Framework for Back-Propagation.* In D. Touretzky, G. Hinton and T. Sejnowski, editors, Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA, pages 21–28. Morgan Kaufmann, 1988. (Cited on pages xxi, xxii, xxv, xxvi, 1 and 13.)

[Li *et al.* 2019] Jianlin Li, Jiangchao Liu, Pengfei Yang, Liqian Chen, Xiaowei Huang and Lijun Zhang. *Analyzing Deep Neural Networks with Symbolic Propagation: Towards Higher Precision and Faster Verification.* In Static Analysis, page 296–319, Berlin, Heidelberg, 2019. Springer-Verlag. (Cited on page 18.)

[Li *et al.* 2020] Linyi Li, Xiangyu Qi, Tao Xie and Bo Li. *Sok: Certified Robustness for Deep Neural Networks.* arXiv preprint arXiv:2009.04131, 2020. (Cited on pages xxii, xxvii and 14.)

[Li *et al.* 2022] Mingjie Li, Yisen Wang and Zhouchen Lin. *CerDEQ: Certifiable Deep Equilibrium Model*. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pages 12998–13013. PMLR, 17–23 Jul 2022. (Cited on page 84.)

[Liu *et al.* 2021] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer*et al. Algorithms for Verifying Deep Neural Networks*. Foundations and Trends® in Optimization, vol. 4, no. 3-4, pages 244–404, 2021. (Cited on page 14.)

[Löfberg 2004] Johan Löfberg. *YALMIP: A Toolbox for Modeling and Optimization in MATLAB*. In Proceedings of the CACSD Conference, volume 3. Taipei, Taiwan, 2004. (Cited on page 73.)

[Lomuscio & Maganti 2017] Alessio Lomuscio and Lalit Maganti. *An Approach to Reachability Analysis for Feed-Forward ReLU Neural Networks*. 2017. (Cited on pages xxii, xxvii, 15 and 24.)

[Lu *et al.* 2017] Jiajun Lu, Hussein Sibai, Evan Fabry and David Forsyth. *No Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles*. arXiv preprint arXiv:1707.03501, 2017. (Cited on pages xxii and xxvi.)

[Magron *et al.* 2017] Victor Magron, George Constantinides and Alastair Donaldson. *Certified Roundoff Error Bounds using Semidefinite Programming*. ACM Transactions on Mathematical Software (TOMS), vol. 43, no. 4, pages 1–31, 2017. (Cited on page 36.)

[Magron 2018] Victor Magron. *Interval Enclosures of Upper Bounds of Roundoff Errors using Semidefinite Programming*. ACM Transactions on Mathematical Software (TOMS), vol. 44, no. 4, pages 1–18, 2018. (Cited on page 36.)

[Mai *et al.* 2020] Ngoc Hoang Anh Mai, Victor Magron and Jean-Bernard Lasserre. *A Sparse Version of Reznick's Positivstellensatz*. arXiv preprint arXiv:2002.05101, 2020. (Cited on page 36.)

[Majumdar *et al.* 2014] A. Majumdar, A. A. Ahmadi and R. Tedrake. *Control and Verification of High-Dimensional Systems with DSOS and SDSOS Programming*. In Proceedings of the 53rd IEEE Conference on Decision and Control, pages 394–401. IEEE, 2014. (Cited on page 35.)

[Meng *et al.* 2022] Mark Huasong Meng, Guangdong Bai, Sin Gee Teo, Zhe Hou, Yan Xiao, Yun Lin and Jin Song Dong. *Adversarial Robustness of Deep Neural Networks: A Survey from a Formal Verification Perspective*. IEEE Transactions on Dependable and Secure Computing, 2022. (Cited on page 14.)

[Mirman *et al.* 2018] Matthew Mirman, Timon Gehr and Martin Vechev. *Differentiable Abstract Interpretation for Provably Robust Neural Networks.* In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pages 3578–3586. PMLR, 10–15 Jul 2018. (Cited on page 18.)

[Molnar 2019] Christoph Molnar. *Interpretable Machine Learning.* `https://christophm.github.io/interpretable-ml-book/`, 2019. (Cited on pages xxi and xxvi.)

[Motzkin 1967] Theodore Samuel Motzkin. *The Arithmetic-Geometric Inequality.* Inequalities (Proc. Sympos. Wright-Patterson Air Force Base, Ohio, 1965), pages 205–224, 1967. (Cited on page 29.)

[Müller *et al.* 2021a] Christoph Müller, François Serre, Gagandeep Singh, Markus Püschel and Martin Vechev. *Scaling Polyhedral Neural Network Verification on GPUs.* In A. Smola, A. Dimakis and I. Stoica, editors, Proceedings of 4th MLSys Conference, volume 3, pages 733–746, 2021. (Cited on page 84.)

[Müller *et al.* 2021b] Mark Niklas Müller, Robin Staab, Marc Fischer and Martin T. Vechev. *Effective Certification of Monotone Deep Equilibrium Models.* CoRR, vol. abs/2110.08260, 2021. (Cited on page 84.)

[Müller *et al.* 2022] Mark Niklas Müller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel and Martin Vechev. *PRIMA: General and Precise Neural Network Certification via Scalable Convex Hull Approximations.* Proc. ACM Program. Lang., vol. 6, no. POPL, jan 2022. (Cited on page 84.)

[Narodytska *et al.* 2018] Nina Narodytska, Shiva Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv and Toby Walsh. *Verifying Properties of Binarized Deep Neural Networks.* In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. (Cited on page 15.)

[Narodytska 2018] Nina Narodytska. *Formal Analysis of Deep Binarized Neural Networks.* In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 5692–5696. International Joint Conferences on Artificial Intelligence Organization, 7 2018. (Cited on pages 15 and 24.)

[Neumaier & Shcherbina 2004] Arnold Neumaier and Oleg Shcherbina. *Safe Bounds in Linear and Mixed-Integer Linear Programming.* Mathematical Programming, vol. 99, no. 2, pages 283–296, Mar 2004. (Cited on page 17.)

[Nie & Demmel 2009] Jiawang Nie and James Demmel. *Sparse SOS Relaxations for Minimizing Functions that are Summations of Small Polynomials.* SIAM Journal on Optimization, vol. 19, no. 4, pages 1534–1558, 2009. (Cited on pages 42 and 43.)

[Pabbaraju *et al.* 2021] Chirag Pabbaraju, Ezra Winston and Zico Kolter. *Estimating Lipschitz Constants of Monotone Deep Equilibrium Models.* In International Conference on Learning Representations, 2021. (Cited on pages 64, 75, 76, 88, 90 and 91.)

[Pardalos & Phillips 1990] Panos M. Pardalos and A. T. Phillips. *A Global Optimization Approach for Solving the Maximum Clique Problem.* International Journal of Computer Mathematics, vol. 33, no. 3-4, pages 209–216, 1990. (Cited on page 48.)

[Peck *et al.* 2017] Jonathan Peck, Joris Roels, Bart Goossens and Yvan Saeys. *Lower Bounds on the Robustness to Adversarial Perturbations.* In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. (Cited on pages 21 and 24.)

[Pál & Vértesi 2009] Károly F. Pál and Tamás Vértesi. *Quantum Bounds on Bell Inequalities.* Physical Review A, vol. 79, no. 2, Feb 2009. (Cited on page 36.)

[Raghunathan *et al.* 2018a] Aditi Raghunathan, Jacob Steinhardt and Percy Liang. *Certified Defenses against Adversarial Examples.* In International Conference on Learning Representations, 2018. (Cited on pages 19, 74, 89 and 90.)

[Raghunathan *et al.* 2018b] Aditi Raghunathan, Jacob Steinhardt and Percy S Liang. *Semidefinite Relaxations for Certifying Robustness to Adversarial Examples.* In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. (Cited on pages xxii, xxiii, xxvi, xxvii, xxviii, 24, 58 and 84.)

[Rauber *et al.* 2020] Jonas Rauber, Roland Zimmermann, Matthias Bethge and Wieland Brendel. *Foolbox Native: Fast Adversarial Attacks to Benchmark the Robustness of Machine Learning Models in PyTorch, TensorFlow, and JAX.* Journal of Open Source Software, vol. 5, no. 53, page 2607, 2020. (Cited on page 90.)

[Rendl *et al.* 2007] Franz Rendl, Giovanni Rinaldi and Angelika Wiegele. *A Branch and Bound Algorithm for Max-Cut Based on Combining Semidefinite and Polyhedral Relaxations.* In International Conference on Integer Programming and Combinatorial Optimization, pages 295–309. Springer, 2007. (Cited on page 45.)

[Ruan *et al.* 2018] Wenjie Ruan, Xiaowei Huang and Marta Kwiatkowska. *Reachability Analysis of Deep Neural Networks with Provable Guarantees.* In Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, page 2651–2659. AAAI Press, 2018. (Cited on pages 17, 23 and 24.)

[Ruan *et al.* 2019] Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening and Marta Kwiatkowska. *Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance.* In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pages 5944–5952. International Joint Conferences on Artificial Intelligence Organization, 7 2019. (Cited on pages 17 and 24.)

[Schlosser & Korda 2020] Corbinian Schlosser and Milan Korda. *Sparse Moment-Sum-of-Squares Relaxations for Nonlinear Dynamical Systems with Guaranteed Convergence.* arXiv preprint arXiv:2012.05572, 2020. (Cited on page 36.)

[Simonyan & Zisserman 2014] Karen Simonyan and Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos.* In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 27. Curran Associates, Inc., 2014. (Cited on pages 13 and 16.)

[Singh *et al.* 2018] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel and Martin Vechev. *Fast and Effective Robustness Certification.* In Advances in Neural Information Processing Systems, pages 10802–10813, 2018. (Cited on page 10.)

[Singh *et al.* 2019] Gagandeep Singh, Timon Gehr, Markus Püschel and Martin Vechev. *An Abstract Domain for Certifying Neural Networks.* Proceedings of the ACM on Programming Languages, vol. 3, no. POPL, page 41, 2019. (Cited on page 10.)

[Szegedy *et al.* 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow and Rob Fergus. *Intriguing Properties of Neural Networks.* In International Conference on Learning Representations, 2014. (Cited on page 84.)

[Szegedy *et al.* 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. *Going Deeper with Convolutions.* In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, 2015. (Cited on page 13.)

[Tacchi *et al.* 2020] Matteo Tacchi, Carmen Cardozo, Didier Henrion and Jean-Bernard Lasserre. *Approximating Regions of Attraction of a Sparse Polyno-*

mial Differential System. IFAC-PapersOnLine, vol. 53, no. 2, pages 3266–3271, 2020. 21th IFAC World Congress. (Cited on page 36.)

[Tacchi *et al.* 2021] Matteo Tacchi, Tillmann Weisser, Jean-Bernard Lasserre and Didier Henrion. *Exploiting Sparsity for Semi-Algebraic Set Volume Computation.* Foundations of Computational Mathematics, Mar 2021. (Cited on page 36.)

[Tjeng *et al.* 2019] Vincent Tjeng, Kai Y. Xiao and Russ Tedrake. *Evaluating Robustness of Neural Networks with Mixed Integer Programming.* In International Conference on Learning Representations, 2019. (Cited on pages xxii, xxvii, 15, 24 and 58.)

[Urban & Miné 2021] Caterina Urban and Antoine Miné. *A Review of Formal Methods applied to Machine Learning*, 2021. (Cited on page 14.)

[Virmaux & Scaman 2018] Aladin Virmaux and Kevin Scaman. *Lipschitz Regularity of Deep Neural Networks: Analysis and Efficient Estimation.* In Advances in Neural Information Processing Systems, pages 3835–3844, 2018. (Cited on page 58.)

[Waki *et al.* 2006] Hayato Waki, Sunyoung Kim, Masakazu Kojima and Masakazu Muramatsu. *Sums of Squares and Semidefinite Program Relaxations for Polynomial Optimization Problems with Structured Sparsity.* SIAM Journal on Optimization, vol. 17, no. 1, pages 218–242, 2006. (Cited on pages xxiii, xxvii and 36.)

[Wang & Magron 2021] Jie Wang and Victor Magron. *Exploiting Term Sparsity in Noncommutative Polynomial Optimization.* Computational Optimization and Applications, vol. 80, no. 2, pages 483–521, Nov 2021. (Cited on page 36.)

[Wang *et al.* 2018a] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang and Suman Jana. *Efficient Formal Safety Analysis of Neural Networks.* In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. (Cited on pages 21 and 24.)

[Wang *et al.* 2018b] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang and Suman Jana. *Formal Security Analysis of Neural Networks using Symbolic Intervals.* In 27th USENIX Security Symposium (USENIX Security 18), pages 1599–1614, 2018. (Cited on pages 21 and 24.)

[Wang *et al.* 2021a] Jie Wang, Martina Maggio and Victor Magron. *SparseJSR: A Fast Algorithm to Compute Joint Spectral Radius via Sparse SOS Decompositions.* In 2021 American Control Conference (ACC), pages 2254–2259, 2021. (Cited on page 36.)

[Wang *et al.* 2021b] Jie Wang, Victor Magron and Jean-Bernard Lasserre. *Chordal-TSSOS: a Moment-SOS Hierarchy that Exploits Term Sparsity with Chordal Extension.* SIAM Journal on Optimization, vol. 31, no. 1, pages 114–141, 2021. (Cited on page 36.)

[Wang *et al.* 2021c] Jie Wang, Victor Magron and Jean-Bernard Lasserre. *TSSOS: A Moment-SOS Hierarchy that Exploits Term Sparsity.* SIAM Journal on Optimization, vol. 31, no. 1, pages 30–58, 2021. (Cited on page 36.)

[Wang *et al.* 2021d] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh and Zico Kolter. *Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification.* In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, volume 34, pages 29909–29921. Curran Associates, Inc., 2021. (Cited on pages xxii, xxvii, 21 and 24.)

[Wang *et al.* 2022] Jie Wang, Victor Magron, Jean-Bernard Lasserre and Ngoc Hoang Anh Mai. *CS-TSSOS: Correlative and Term Sparsity for Large-Scale Polynomial Optimization.* ACM Trans. Math. Softw., oct 2022. Just Accepted. (Cited on pages 36, 46 and 47.)

[Wei & Kolter 2022] Colin Wei and Zico Kolter. *Certified Robustness for Deep Equilibrium Models via Interval Bound Propagation.* In International Conference on Learning Representations, 2022. (Cited on page 84.)

[Weisser *et al.* 2018] Tillmann Weisser, Jean-Bernard Lasserre and Kim-Chuan Toh. *Sparse-BSOS: a Bounded Degree SOS Hierarchy for Large Scale Polynomial Optimization with Sparsity.* Mathematical Programming Computation, vol. 10, no. 1, pages 1–32, 2018. (Cited on page 36.)

[Weng *et al.* 2018a] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning and Inderjit Dhillon. *Towards Fast Computation of Certified Robustness for ReLU Networks.* In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pages 5276–5285. PMLR, 10–15 Jul 2018. (Cited on pages 10, 20, 24, 58 and 84.)

[Weng *et al.* 2018b] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Aurelie Lozano, Cho-Jui Hsieh and Luca Daniel. *On Extensions of CLEVER: A Neural Network Robustness Evaluation Algorithm.* In 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pages 1159–1163. IEEE, 2018. (Cited on pages 10, 22 and 24.)

[Weng *et al.* 2018c] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh and Luca Daniel. *Evaluating the Robustness*

of Neural Networks: An Extreme Value Theory Approach. In International Conference on Learning Representations, 2018. (Cited on pages xxii, xxvii, 10, 22, 24 and 58.)

[Winston & Kolter 2020] Ezra Winston and Zico Kolter. *Monotone Operator Equilibrium Networks.* In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 10718–10728. Curran Associates, Inc., 2020. (Cited on pages xxii, xxvi, 4, 66 and 75.)

[Wong & Kolter 2018] Eric Wong and Zico Kolter. *Provable Defenses Against Adversarial Examples via the Convex Outer Adversarial Polytope.* In International Conference on Machine Learning, pages 5286–5295. PMLR, 2018. (Cited on pages 19, 24 and 58.)

[Wu *et al.* 2020] Min Wu, Matthew Wicker, Wenjie Ruan, Xiaowei Huang and Marta Kwiatkowska. *A Game-Based Approximate Verification of Deep Neural Networks with Provable Guarantees.* Theor. Comput. Sci., vol. 807, pages 298–329, 2020. (Cited on pages 16 and 24.)

[Xiang *et al.* 2018a] Weiming Xiang, Hoang-Dung Tran and Taylor T. Johnson. *Output Reachable Set Estimation and Verification for Multilayer Neural Networks.* IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 11, pages 5777–5783, 2018. (Cited on pages 18 and 24.)

[Xiang *et al.* 2018b] Weiming Xiang, Hoang-Dung Tran, Joel A. Rosenfeld and Taylor T. Johnson. *Reachable Set Estimation and Safety Verification for Piecewise Linear Systems with Neural Network Controllers.* In 2018 Annual American Control Conference (ACC), pages 1574–1579, 2018. (Cited on page 18.)

[Xu *et al.* 2021] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin and Cho-Jui Hsieh. *Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers.* In International Conference on Learning Representations, 2021. (Cited on pages 20 and 24.)

[Xue *et al.* 2022] Anton Xue, Lars Lindemann, Alexander Robey, Hamed Hassani, George J. Pappas and Rajeev Alur. *Chordal Sparsity for Lipschitz Constant Estimation of Deep Neural Networks*, 2022. (Cited on page 58.)

[Zhang *et al.* 2018] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh and Luca Daniel. *Efficient Neural Network Robustness Certification with General Activation Functions.* In Advances in Neural Information Processing Systems (NuerIPS), dec 2018. (Cited on pages xxii, xxvii, 10, 20, 24, 58 and 84.)

[Zhang *et al.* 2019] Huan Zhang, Pengchuan Zhang and Cho-Jui Hsieh. *RecurJac: An Efficient Recursive Algorithm for Bounding Jacobian Matrix of Neural Networks and Its Applications.* Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pages 5757–5764, Jul. 2019. (Cited on pages 20 and 24.)

[Zhang *et al.* 2020] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning and Cho-Jui Hsieh. *Towards Stable and Efficient Training of Verifiably Robust Neural Networks.* In International Conference on Learning Representations, 2020. (Cited on pages 20 and 24.)