EPFL

Thèse n° 9118

Predicting in Uncertain Environments: Methods for Robust Machine Learning

Présentée le 14 octobre 2022

Faculté des sciences et techniques de l'ingénieur
Laboratoire de systèmes d'information et d'inférence
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Paul Thierry Yves ROLLAND

Acceptée sur proposition du jury

Prof. M. Jaggi, président du jury
Prof. V. Cevher, directeur de thèse
Prof. G. Peyré, rapporteur
Prof. P. Ravikumar, rapporteur
Prof. N. Flammarion, rapporteur

École polytechnique fédérale de Lausanne

2022

Essentially all models are wrong,
but some are useful.
— George Box

Acknowledgements

This thesis is the result of several collaborations involving many people. To begin with, I would like to express my gratitude to the primary instigator of this great adventure, which is my supervisor Volkan Cevher. I thank you for your encouragements, your contagious motivation and optimism, your creativity in finding algorithm names, your humour, your ability to gather people and create connections. This last quality was particularly important to me, since it allowed me to work with a myriad of wonderful people.

I was honored to have Gabriel Peyré, Nicolas Flammarion, Pradeep Ravimukar and Martin Jaggi as members of my thesis defense comittee. I am thankful for their time and the interest they had in my work. Special thanks to Gabriel Peyré for hosting me at ENS Paris for three months, and introduced me to the field of Optimal Transport. While no publication has emerged from this short collaboration, it still inspired one of Gabriel's famous Twitter posts, which is priceless.

I thank Francesco Locatello for hosting me at Amazon Tübingen. I spent four great months there, discovering the field of causality, and I had the opportunity to interact with one of the world leaders of this field, namely, Bernhard Schölkopf.

What gave all the flavour to this adventure is certainly the people I have been fortunate to collaborate with at LIONS. The team at LIONS is very diverse, in terms of knowledge, nationalities and personalities, and I quickly realized everything I could gain from interacting with all these people. I would like to thank Thomas Pethick for his infinite kindness, his crazy ability to always find something positive in every aspect of life. I am thankful to everything he brought me into, such as planting trees, cook a lentilles dahl or swimming in the lake. I am also thankful for being a great climbing partner, and being consistently motivated for doing crazy things, such as the Lausanne triathlon under the storm. I thank Leello for everything he is. He is very good at asking disturbing questions making one rethink our whole conception of things. This ranges from philosophical questions, with his "fifty francs on the floor" dilemma, to technical questions in Machine Learning. He also knows the best riddles, one of them still begin unsolved. I thank Igor Krawczuk for being a model of dedication and rigour, for the deep political debates while performing pull-ups and push-ups at 7am in the rain. It is impressive to have that much energy while I literally never saw him eat. Thank you also for your help on improving my hand-stand and for introducing me to the acroyoga Lausanne association.

I always had a great time hanging out with all the people from the lab. Huge thanks to the team members that started the PhD at the same time as me: Thomas Sanchez for being a loyal

mate for all these years at EPFL and the tidiest person I know, Ali Kavis for his great laugh and jokes, Mehmet Fatih for his peaceful personality and Fabian Latorre for begin an amazing collaborator, able to explain anything and make it look trivial right away, even to Reviewer #2.

I thank the people that were already at LIONS when I started: Ilija Bogunovic and Jonathan Scarlett, which were the first persons I worked with at LIONS, during my Master thesis, and made me do my first steps in the academic world. Ya-Ping Hsieh, whose qualitative description would be as long as incomprehensible if you never interacted with him. But behind his complexity lies a truly inspiring person. Ahmet Alacaoglu for his great advices on gaining weight, which can be summarized as: "Eat food!". And Kamalaruban Parameswaran for his warm smile every time I would come to ask a question.

And I thank the people that joined LIONS during my PhD: Luca Viano for his eternal enthousiasm, Pedro Abranches for his teasing, Fanghui for is valuable and limited time, Stratis Skoulakis (or Souvlakis?) for his high fives and warm huges, Kimon Antonakopoulos for being the boss, Grigorios Chrysos for his pragmatism in life and Ali Ramezani for keeping us up-to-date on the best Twitter posts in Machine Learning, Yurii Malitsky for his smart and funny personality, and his great tutorial on Variational Inequalities, and Nadav Hallak for his look when watching the snow for the first time in Switzerland in the middle of a meeting and screaming "Wow, I need to call my wife!"

Warm thanks to Gosia Baltaian, our secretary, for her efficiency in organizing all the meeting, booking, conferences for such a big lab with perfect reliability.

Thanks also to all my friends. I would not dare making a list, being too scared to forget anyone. Thank you to Elina, my girlfriend for all these years, who supported me, and nodded in a very convincing way every time I would speak about my work.

Last but surely not least, I express my gratitude to my family. In particular to my parents Gilles and Virginie, for giving me the complete freedom to study anything I wanted, and always encouraging me in everything I do.

Lausanne, September 30, 2022 P. R.

Abstract

One of the main goal of Artificial Intelligence (AI) is to develop models capable of providing valuable predictions in real-world environments. In particular, Machine Learning (ML) seeks to design such models by *learning* from examples coming from this same environment. However, the real world is often not static, and the environment in which the model is used can differ from the one in which it is trained. It is hence desirable to design models that are *robust* to changes of environments. This encapsulates a large family of topics in ML, such as adversarial robustness, meta-learning, domain adaptation and others, depending on the way the environment is perturbed.

In this dissertation, we focus on methods for training models whose performance does not drastically degrade when applied to environments differing from the one the model has been trained in. Various types of environmental changes will be treated, differing in their structure or magnitude. Each setup defines a certain kind of robustness to certain environmental changes, and leads to a certain optimization problem to be solved. We consider 3 different setups, and propose algorithms for solving each associated problem using 3 different types of methods, namely, **min-max optimization** (Chapter 2), **regularization** (Chapter 3) and **variable selection** (Chapter 4).

Leveraging the framework of distributionally robust optimization, which phrases the problem of robust training as a min-max optimization problem, we first aim to train robust models by directly solving the associated min-max problem. This is done by exploiting recent work on game theory as well as first-order sampling algorithms based on the Langevin dynamics. Using this approach, we propose a method for training robust agents in the scope of Reinforcement Learning.

We then treat the case of adversarial robustness, i.e., robustness to small arbitrary perturbation of the model's input. It is known that neural networks trained using classical optimization methods are particularly sensitive to this type of perturbations. The adversarial robustness of a model is tightly connected to its smoothness, which is quantified by its so-called *Lipschitz constant*. This constant measures how much the model's output changes upon any bounded input perturbation. We hence develop a method to estimate an upper bound on the Lipschitz constant of neural networks via polynomial optimization, which can serve as a robustness certificate against adversarial attacks. We then propose to penalize the Lipschitz constant during training by minimizing the *1-path-norm* of the neural network, and we develop an algorithm for solving the resulting regularized problem by efficiently computing the proximal

operator of the 1-path-norm term, which is non-smooth and non-convex.

Finally, we consider a scenario where the environmental changes can be arbitrary large (as opposed to adversarial robustness), but need to preserve a certain *causal structure*. Recent works have demonstrated interesting connections between robustness and the use of causal variables. Assuming that certain mechanisms remain invariant under some change of the environment, it has been shown that knowing the underlying causal structure of the data at hand allows to train models that are invariant to such changes. Unfortunately, in many cases, the causal structure is unknown. We thus propose a causal discovery algorithm from observational data in the case of non-linear additive models.

We emphasize that, while we make the relation to robustness explicit in each chapter, the focus in this thesis is put on the tools involved in the different algorithms rather than the resulting method itself. Hence the scope of this work extends to other fields of research than robust ML, such as first-order sampling methods, generalization and causal discovery.

Résumé

Un des objectifs principaux en Intelligence Artificielle est de développer des modèles capables de fournir des prédictions valides dans des environnements réels. En particulier, l'apprentissage automatique (ou Machine Learning) cherche à construire de tels modèles en *apprenant* à partir de données provenant de ce même environnement. Cependant, le monde réel n'est la plupart du temps pas statique, et l'environnement dans lequel le modèle sera utilisé peut varier par rapport à celui dans lequel il a été entrainé. Il est donc nécéssaire de développer des modèles qui sont *robustes* à ces changements d'environnement. Ce problème englobe une vaste classes de sujets en Machine Learning, telles que la robustesse antagoniste, le meta-learning, ou l'adaptation de domaine, dépendamment de la manière dont l'environnement est perturbé.

Cette thèse traite de méthodes pour entrainer des modèles dont la performance ne se dégrade pas drastiquement lorsque ceux-ci sont appliqués dans des environnement différents de l'environnement d'entrainement. Nous considérerons plusieurs types de changements d'environnements, différant en terme de structure et de magnitude. Nous traitons 3 différents types de robustesse, et proposons des algorithmes capables d'entrainer des modèles satisfaisant chacun de ces types de robustesse. Ces algorithmes sont basés sur des techniques très différentes pour chaque scénario : **l'optimisation min-max** (Chapter 2), la **regularisation** (Chapter 3) et **la sélection de variables** (Chapter 4).

Dans un premier temps, nous proposons en méthode d'entrainement robuste en introduisant un adversaire perturbant le modèle durant l'entrainement, afin de le rendre moins sensible aux possibles modifications de l'environnement. Cette tâche peut s'exprimer comme un problème min-max, dans lequel un modèle est entrainé de sorte à minimiser une fonction de coût et un adversaire est entrainé simultanément afin de maximiser ce coût. À l'aide de récents travaux sur la théorie des jeux, ainsi que sur les algorithmes d'échantillonnage via la dynamique de Langevin, nous proposons un algorithme permettant d'entrainer des agents robustes dans le cadre de l'apprentissage par renforcement.

Nous traitons ensuite le cas de la robustesse aux petites perturbations de l'entrée. Les réseaux de neurones entrainés avec des méthodes d'optimisation classiques sont particulièrement sensibles à ce genre de perturbations. Ce type de robustesse est fortement lié à la régularité du modèle, caractérisée par sa *constante de Lipschitz*. Cette constante mesure jusqu'à quel point la sortie du modèle peut varier lorsque l'on modifie son entrée. Nous proposons donc une méthode permettant d'estimer cette quantité dans le cas de modèles paramétrés par

des réseaux de neurones. Cela permet notamment d'obtenir un certificat de robustesse aux perturbations de l'éntrée. Nous proposons ensuite une méthode pénalisant la constante de Lipschitz durant la phase d'entrainement, en minimisant une certaine norme (la norme *1-path*) servant de proxy à la constante de Lipschitz. Nous développons ensuite un algorithme pour résoudre le problème associé en utilisant la méthode du gradient proximal. Pour ce faire, nous proposons une méthode efficace pour calculer l'opérateur proximal de la norme *1-path*, qui est non-régulière et non-convexe.

Enfin, nous considérons le scénario où les changements de l'environnement préservent une certaine structure causale. En supposant que certains mécanismes causaux restent invariants entre l'environnement d'entrainement et l'environnement de test, il a été démontré que connaître la structure causale des variables étudiées permet d'entrainer des modèles qui restent invariants à ce type de changement. Malheureusement, dans de nombreux cas, cette structure causale est inconnue. Nous proposons donc un algorithme d'inférence de la structure causale à partir de données observationnelles dans le cas de modèles additifs non-linéaires.

Nous appuyons sur le fait que, bien que nous explicitons la relation à la robustesse dans chaque chapitre, cette thèse se concentre principalement sur les outils développés dans les différents algorithmes plutôt que sur la méthode résultante elle même. La portée de cette dissertation s'étend donc à des domaines autres que la robustesse en Machine Learning, tels que les méthodes d'échantillonnage, la généralisation et la découverte de structures causales.

Bibliographic Note

This dissertation is based on the following publications:

- Paul Rolland, Armin Eftekhari, Ali Kavis, Volkan Cevher. "Double-loop Unadjusted Langevin Algorithm." International Conference on Machine Learning (ICML), 2020.
- Parameswaran Kamalaruban, Yu-Ting Huang, Ya-Ping Hsieh, Paul Rolland, Cheng Shi, Volkan Cevher. "Robust reinforcement learning via adversarial training with Langevin dynamics." Advances in Neural Information Processing Systems (NeurIPS), 2020.
- Fabian Latorre, Paul Rolland, Volkan Cevher. "Lipschitz constant estimation of neural networks via sparse polynomial optimization." International Conference on Learning Representation (ICLR), 2020.
- Fabian Latorre,* Paul Rolland,* Nadav Hallak,* Volkan Cevher. "Efficient Proximal Mapping of the 1-path-norm of Shallow Networks." International Conference on Machine Learning (ICML), 2020.
- Nadav Hallak*, Paul Rolland*, Fabian Latorre*, Volkan Cevher. "Efficient Proximal Mapping of the 1-path-norm Regularizer of unit-width Deep Neural Networks." Work in progress.
- Paul Rolland, Volkan Cevher, Matthäus Kleindessner, Chris Russel, Bernhard Schölkopf, Dominik Janzing, Francesco Locatello. "Score matching enables causal discovery of nonlinear additive noise models." International Conference on Machine Learning (ICML), 2022.

Bibliographic notes are added at the end of some sections to specify my own contributions. If no note appears, it means that I contributed to all the results within the section.
Here are my other publications that I worked on during my PhD, but which are not included in this dissertation:

- Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, Volkan Cevher. "High-dimensional Bayesian optimization via additive models with overlapping groups." International conference on artificial intelligence and statistics (AISTATS), 2018.
- Paul Rolland, Ali Kavis, Alexander Immer, Adish Singla, Volkan Cevher. "Efficient learning of smooth probability functions from Bernoulli tests with guarantees." International Conference on Machine Learning (ICML), 2019.

- Ya-Ping Hsieh, Ali Kavis, Paul Rolland, Volkan Cevher. "Mirrored Langevin dynamics." Advances in Neural Information Processing Systems (NeurIPS), 2018.
- Fabian Latorre, Leello Tadesse Dadi, Paul Rolland, Volkan Cevher. "The Effect of the Intrinsic Dimension on the Generalization of Quadratic Classifiers." Advances in Neural Information Processing Systems (NeurIPS), 2021.
- Paul Rolland, Luca Viano, Norman Schuerhoff, Boris Nikolov, Volkan Cevher. "Identifiability and generalizability from multiple experts in Inverse Reinforcement Learning." Advances in Neural Information Processing Systems (NeurIPS), 2022.

Contents

Acknowledgements i

Abstract (English/Français) iii

1 Introduction 1

2 Robustness to distribution shift using min-max optimization 7
2.1 Preliminaries . . . 7
2.1.1 Min-max optimization . . . 7
2.1.2 Finding mixed Nash equilibria . . . 8
2.2 Double-loop unadjusted Langevin algorithm . . . 10
2.2.1 Introduction . . . 10
2.2.2 Related work . . . 12
2.2.3 Preliminaries . . . 13
2.2.4 DL-ULA for unconstrained sampling . . . 16
2.2.5 DL-MYULA for constrained sampling . . . 22
2.3 Robust Reinforcement Learning via adversarial training with Langevin dynamics 25
2.3.1 Introduction . . . 25
2.3.2 Preliminaries: Markov decision problems and deterministic policy gradient . . . 26
2.3.3 Robust training with two players Markov games . . . 27
2.3.4 Experiments . . . 28
2.4 Bibliographic notes . . . 29

3 Robustness to adversarial perturbation using regularization 33
3.1 Lipschitz constant estimation of neural networks via sparse polynomial optimization . . . 33
3.1.1 Introduction . . . 34
3.1.2 Polynomial optimization formulation . . . 35
3.1.3 Solving the POP using polynomial positivity certificate . . . 37
3.1.4 Reducing the number of variables . . . 38
3.1.5 Relation to Shor's relaxation and Sum-Of-Squares hierarchy . . . 41
3.1.6 Experiments . . . 42
3.2 1-path-norm Regularization using Proximal Gradient Method . . . 45

3.2.1 Introduction . . . 45
3.2.2 Problem setup and preliminaries . . . 47
3.2.3 Path norm regularization of shallow neural networks . . . 49
3.2.4 Path norm regularization of deep neural networks . . . 55
3.2.5 Experiments . . . 62
3.3 Bibliographic notes . . . 67

4 Robustness to structured environmental changes using causal feature selection 69
4.1 Preliminaries: Causality and robustness . . . 69
4.1.1 Causality and structural equation models . . . 69
4.1.2 Robustness via causal features selection . . . 71
4.2 Causal discovery for non-linear additive models . . . 72
4.2.1 Introduction . . . 72
4.2.2 Related Work . . . 74
4.2.3 Preliminaries . . . 75
4.2.4 Causal discovery via score matching . . . 77
4.2.5 Experiments . . . 82

5 Conclusion and future work 89
5.1 Summary of the thesis . . . 89
5.2 Directions for future work . . . 90
5.2.1 Further analysis of DL-ULA . . . 90
5.2.2 Analysis of stochastic prox method for 1-path norm regularization . . . 90
5.2.3 Extension of SCORE to other identifiable models . . . 91

A Appendix for Chapter 2 93
A.1 Proofs of Section 2.2 . . . 93
A.1.1 Proof of Lemma 9 . . . 93
A.1.2 Proof of Lemma 10 . . . 95
A.1.3 Proof of Theorem 11 . . . 96
A.1.4 Proof of Lemma A.1.4 . . . 97
A.1.5 Proof of Theorem 14 . . . 99
A.2 Appendix for Section 2.3 . . . 103

B Appendix for Chapter 3 107
B.1 Proofs of Section 3.1 . . . 107
B.1.1 Proof of Theorem 15 . . . 107
B.1.2 Proof of Proposition 22 . . . 108
B.2 Appendix for Section 3.2: Proximal operator in the multi-output setting . . . 109
B.3 Proofs of Section 3.2 . . . 119
B.3.1 Proof of Theorem 23 . . . 119
B.3.2 Proof of Theorem 24 . . . 120

B.3.3 Proof of Lemma 26 121
B.3.4 Proof of Lemma 27 122
B.3.5 Proof of Lemma 30 122
B.3.6 Proof of Corollary 31 124
B.3.7 Proof of Lemma 32 124
B.3.8 Proof of Lemma 34 125
B.3.9 Proof of Lemma 38 126
B.3.10 Proof of Lemma 42 127
B.3.11 Proof of Theorem 43 127

C Appendix for Chapter 4 **129**
C.1 Appendix for Section 4.2: Additional experiments 129

Bibliography **131**

1 Introduction

In recent years, powerful models able to predict input/output relationships in complex environments with tremendous accuracy have been developed, in particular thanks to the success of Machine Learning (ML), i.e., by learning from training data. Examples of such models include image classification, Reinforcement Learning (RL), or self-driving cars to name a few. The models developed for these tasks are trained using examples, e.g., already classified images in the context of image classification. Thanks to the increasing amount of available data, such methods have shown impressive results in a wide variety of tasks, performing even better than human predictions: CoAtNet-7 obtained 90.88% accuracy on ImageNet (Dai et al., 2021), alphaZero defeated the Go world champion (Silver et al., 2018), and self-driving cars are safer and more reliable than humans in certain contexts (Badue et al., 2021). These great successes made such models omnipresent in our modern world. We indeed regularly interact with them, and often do no doubt the correctness of their predictions.

This increasing trust in these models raises questions about the reliability and robustness of these predictions. ML models expected to perform predictions in a certain environment are trained using data acquired from this same environment. While the trained models show impressive success when tested on the same environment as the one on which there are trained, a slight change in the test environment can lead to catastrophic decrease of the accuracy. Indeed, without appropriate training, image classification algorithms are usually very sensitive to adversarial perturbations, in the sense that small perturbations of the input, almost imperceptible to the human eye, can heavily degrade the accuracy of the prediction (Dong et al., 2020). Similarly, RL agents trained to perform well in a given environment can be severely perturbed by small modifications of this environment (Morimoto and Doya, 2005). Finally, while self-driving cars are safe and reliable in classical scenarios, they have failure cases and may fail to detect unusual obstacle or interpret unusual signs. In 2016, the first fatal case of self-driving cars occurred in Florida. The accident was cause by the Tesla's autopilot misdetecting a white truck because of its similar color with the sky. This discrepancy between the classical test accuracy and the robust accuracy triggered the development of methods for training robust models that are not too sensitive to changes in the environment.

A rather general template describing the problem of training robust models is given as follows

$$\begin{aligned} &\text{Input: } X = \{(\mathbf{x}_i, y_i)\}_{i=1}^N \sim \mathbf{p}^{\text{data}} \\ &\text{Goal: } \min_{f \in \mathscr{F}} \max_{\mathbf{p}^{\text{test}} \in \mathscr{P}} \mathbb{E}_{(\mathbf{x},y)\sim \mathbf{p}^{\text{test}}}[\ell(f(\mathbf{x}), y)], \end{aligned} \tag{ROB}$$

where $\mathbf{p}^{\text{data}}$ denotes the training distribution that we get samples from, $\mathscr{F}$ is a class of models, ℓ is a loss function, $\mathscr{P}$ is a class of test distributions and $\mathbb{E}_{X\sim\mu}$ denotes the expectation operator over the distribution μ. The vectors $\mathbf{x}_i \in \mathbb{R}^d$ denote a set of observable features, and the associated values $y_i \in \mathbb{R}$ denote the quantity that we aim to predict. This problem template is known as Distributionally Robust Optimization (Neumann, 1928). The parameter specifying the robustness of problem (ROB) is the choice of the space $\mathscr{P}$ which describes how different we think the test environment will be from the training one. If $\mathscr{P} = \{\mathbf{p}^{\text{data}}\}$, then we assume no difference between $\mathbf{p}^{\text{data}}$ and $\mathbf{p}^{\text{test}}$, and (ROB) reduces to a classical non-robust supervised learning problem. For (ROB) to yield an interesting solution, we need $\mathscr{P}$ to only contain distributions that are not too different from $\mathbf{p}^{\text{data}}$, i.e., that are *close* to $\mathbf{p}^{\text{data}}$ in some way, or that share a certain structure. If $\mathscr{P}$ is too large, it becomes impossible for the trained model to learn anything.

As a motivating example, consider an image classification task, where the input image $\mathbf{x}$ can be modified at test time by changing the pixels up to a certain budget. This problem can be written as

$$\min_{f\in\mathscr{F}} \max_{\|\delta_i\|\leq\epsilon} \sum_{i=1}^N \ell(f(\mathbf{x}_i + \delta_i), y_i) \tag{1.1}$$

where the test distribution deviation is limited by bounding the norm of the injected perturbations $\|\delta_i\| \leq \epsilon$ for some $\epsilon > 0$ and norm $\|\cdot\|$ to be chosen. We can see that problem (1.1) fits into the framework (2.1). Indeed, letting $\mathscr{P} = \{\nu : \nu = (g \times \text{Id})\#\mathbf{p}^{\text{data}}, \|g(\mathbf{x}) - \mathbf{x}\| \leq \epsilon \ \forall \mathbf{x}\}$ where # denotes the push-forward operator,[1] problem (2.1) boils down to (1.1) (after replacing the expectation by the empirical average).

Several approaches have been developed for training robust models, which can roughly be classified in three categories:

- **Solving the min-max problem** (ROB) **directly:** In the case where we are able to obtain samples from the distributions within $\mathscr{P}$, it is possible to approximately solve problem (ROB) by exploiting recent works on min-max optimization, which has known an increase of interest, in particular with the popularity of Generative Adversarial Networks (Creswell et al., 2018). Even when we do not have direct access to distributions in $\mathscr{P}$, we can mimic the changes between the train and test distributions by directly perturbing the model, the data or the environment during training, in order to make sure that

[1] More precisely, $(g \times \text{Id})\#\mathbf{p}^{\text{data}}$ denotes the distribution obtained by sampling $(\mathbf{x}, y)$ from $\mathbf{p}^{\text{data}}$, and then applying g to $\mathbf{x}$ (Hsieh et al., 2018).

the trained model's accuracy is not severely affected by some small changes. These perturbations can take various forms, leading to different practical algorithms, such as dropout (Hinton et al., 2012) or adversarial training (Ganin et al., 2016). One way to design those perturbations is to simultaneously train a second model that perturbs the learning model in a way that harms the model's accuracy the most, under some budget constraint, also leading to a min-max formulation serving as a proxy for (ROB).

- **Regularize the model:** Another possibility is to directly penalize the sensitivity of the trained model to small changes in the input. As mentioned previously, non-robustly trained models can make very different predictions after a slight perturbation of the input. This is unacceptable for most practical tasks, especially in computer vision. This sensitivity of the model's output to change in the input is precisely characterized by the so-called *Lipschitz constant*, which is defined as the maximal possible ratio between the output difference and the input difference. Methods have hence been designed both to estimate this constant for various models, e.g., neural networks, which can serve as a robustness certificate against adversarial attacks, and to penalize it during training.

- **Carefully select the input variables**: Finally, it is believed that the non-robustness of classically trained models is partially due to the high-dimensionality of the input data, yielding the emergence of non-robust features with high predictive power which are exploited by the trained model (Ilyas et al., 2019). Therefore, reducing the number of input variables generally leads to more robust models. On the other hand, we do not want to loose too much predictive power, and hence we need to identify which variables are necessary for the prediction task at hand. Several variable selection techniques have been designed, improving the robustness and interpretability of the resulting model (Andersen and Bro, 2010).

This dissertation is separated in 3 chapters, each targeting a certain type of robustness, associated with a choice of $\mathscr{P}$ in (ROB), and featuring an algorithm in one of the categories described above.

Chapter 2: Robustness to distribution shift using min-max optimization

In this chapter, we study the case where we have access to the distributions within $\mathscr{P}$, and we aim to solve (ROB) directly, by exploiting an existing approach for nonconvex-nonconcave min-max problems (Hsieh et al., 2019). This algorithm requires to iteratively sample from certain distributions whose normalization constants are unknown. We hence need a method to efficiently perform these sampling tasks.

A popular method for sampling from distributions known up to a normalization constant is the Unadjusted Langevin Algorithm (ULA). This method works by applying Gradient Descent to the negative log density, and adding a certain amount of Gaussian noise at each iterations. Convergence properties of this method have been studied in various settings (Ahn et al., 2012; Cheng and Bartlett, 2017; Dalalyan and Karagulyan, 2017; Durmus et al., 2017, 2018a; Welling

and Teh, 2011). In this dissertation, we present one work analysing the convergence properties of the ULA for both unconstrained and constrained sampling in the log-concave setting. We show that, by exploiting a certain multi-stage step size schedule, we obtain improved convergence guarantees in certain cases.

We then turn our attention to the training of robust models in the scope of Reinforcement Learning (RL), i.e., we want to train an agent in a certain environment so that the agent still performs well in different environments where the transition dynamics is modified to some extend. However, we consider that we do not have direct access to these modified environments, and we instead involve an adversary that perturbs the agent's action during training. Exploiting recent advances on policy gradient methods (Sutton et al., 2000; Silver et al., 2014; Schulman et al., 2015, 2017), this problem can be phrased as a continuous min-max optimization problem with access to stochastic gradients. Hence, we apply the method of (Hsieh et al., 2019) and demonstrate improved performance over state-of-the-art algorithms.

Chapter 3: Robustness to adversarial perturbation using regularization

In this next chapter, we focus on training models which are robust to adversarial attacks, i.e., to small perturbations of the input fed to the model at test time. This can be achieved by constraining or regularizing the Lipschitz constant of the model. However, computing the Lipschitz constant for most models, e.g., neural networks, is a computationally hard task. We hence start by proposing a method estimating an upper bound on the Lipschitz constant of feed-forward neural networks. The estimated value hence provides a robustness certificate to adversarial attacks for trained neural networks.

Directly regularizing the Lipschitz constant during training turns out to be quite computationally inefficient. Instead, we propose to regularize a proxy for the Lipschitz constant known as the 1-path-norm of the network. We show that this quantity provides a general upper bound on the Lipschitz constant, and we develop an efficient algorithm to solve the resulting optimization problem by efficiently computing the proximal mapping of the non-smooth and non-convex path-norm term.

Chapter 4: Robustness to structured environmental changes using causal feature selection

Finally, we consider training robust models by limiting the number of input variables based on the notion of Causality. Coming back to our problem definition of training a robust model (ROB), recall that the main difficulty comes from the difference $\mathbf{p}^{\text{data}} \neq \mathbf{p}^{\text{test}} \in \mathscr{P}$. This difference is enhanced by the high dimensionality of the input data. For example, suppose that the distribution of each input variable is allowed to change within a certain budget. Then, the difference between $\mathbf{p}^{\text{data}}$ and $\mathbf{p}^{\text{test}}$ in Wasserstein (W_2) distance would generally scale as $W_2(\mathbf{p}^{\text{data}}, \mathbf{p}^{\text{test}}) = \mathscr{O}(\sqrt{d})$, so we can expect the distributions to be more and more different as the dimension grows. Therefore, limiting the input dimension is in general an efficient way to improve robustness.

A recent trend of works towards developing robust models has emerged using the framework of Causality (Pearl, 2009), showing optimal robustness properties in cases where we are able to identify the causal variables (Bühlmann, 2020). Indeed, certain variables can be more or less prone to variability when changing the environment, and certain relations between variables can sometimes hardly be modified. Taking a closer look at the way the input data have been generated in a first place can allow to identify certain causal relations among variables: This is known as Causal Discovery.

In this last chapter, we start by introducing the framework of Causality and its relation to robustness, based on the work of Bühlmann (2020). It has been shown that, if we restrict $\mathscr{P}$ in (ROB) to the set of distributions having a similar causal structure as $\mathbf{p}^{\text{data}}$, (see the **invariance of causal mechanisms** Assumption 44), then, the accuracy of any model trained only using the appropriate causal variables remains unchanged under any $\mathbf{p}^{\text{test}} \in \mathscr{P}$. In particular, in the case of linear classification with quadratic loss, it leads to the solution of (ROB). This assumption on $\mathscr{P}$ contrasts with the two previous chapters, which assumed small but unstructured changes between $\mathbf{p}^{\text{data}}$ and $\mathbf{p}^{\text{test}}$, rather than arbitrarily large but structured ones here.

This connection motivates the search for causal structure, since causal relations are generally not known a priori. Hence, the main part of this chapter focuses on Causal Discovery from observational data. We propose an efficient algorithm for estimating the causal graph from data in the case of non-linear additive noise models.

Notation. $\mathbb{R}$ denotes the space of real numbers, $\|\cdot\|_2 : \mathbb{R}^d \to$ denotes the Euclidean norm. The symbol $\mathcal{O}$ denotes an asymptotic upper bound, i.e., for two functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that $f(x) = \mathcal{O}(g(x))$ if there exist constants $C, R > 0$ such that $|f(x)| \leq Cg(x)$ for all x such that $x \geq R$. The symbol $\tilde{\mathcal{O}}$ is the same as $\mathcal{O}$ but hides logarithmic terms, i.e., $f(x) = \tilde{\mathcal{O}}(g(x)) \Leftrightarrow f(x) = \mathcal{O}(g(x)\log^n(x))$ for some integer n. We use the symbol $\lesssim$ to denote functional inequalities that hide possible multiplicative constant, i.e., $f(x) \lesssim g(x)\ \forall x \Leftrightarrow \exists c > 0, f(x) \leq cg(x)\ \forall x$ where c is a constant independent of x. We write $X \sim \mathcal{N}(\mu, \Sigma)$ to indicate that the random variable X follows a Gaussian distribution with mean μ and covariance matrix Σ.

2 Robustness to distribution shift using min-max optimization

Our chosen notion of robustness (ROB) features a min-max formulation, where the *max* part characterizes the possible distribution shift at test time compared to the training distribution. In this section, we propose to solve this problem directly using a min-max optimization method. To this end, we introduce an *adversary* perturbing the learning model, whose goal is to simulate potential situations that can possibly arise at test time due to the distribution shift. The adversary is trained together with the learning model, so as to increase the loss value as most as possible. We hence expect the model to adapt to these perturbation and thus be more robust.

In this chapter, we start with a preliminary section introducing a framework, based on a recent work (Hsieh et al., 2019), for solving min-max problems. The algorithm requires the use of a sampling algorithm, and the authors in (Hsieh et al., 2019) propose to use a method based on Langevin dynamics. We hence present a work analysing the convergence property of the Unadjusted Langevin Algorithm in the log-concave setting. Finally, we apply this framework to the training of robust Reinforcement Learning agents under changes of the environment, by introducing an adversary perturbing the learning agent's actions.

2.1 Preliminaries

2.1.1 Min-max optimization

In its most generic form, a min-max problem, or Saddle Point problem (SPP), can be phrased as follows:

$$\min_{\theta\in\Theta}\max_{\omega\in\Omega} f(\theta,\omega). \tag{2.1}$$

where $\Theta \subseteq \mathbb{R}^n$, $\Omega \subseteq \mathbb{R}^m$. The function $f(\theta,\omega)$ represents the training loss when the model makes a guess based on a model parameterized by θ, which is perturbed according to an adversary parametrized by ω.

Solving (2.1) requires finding a point $(\theta^\star, \omega^\star)$ such that

$$f(\theta, \omega^\star) \leq f(\theta^\star, \omega^\star) \leq f(\theta^\star, \omega), \quad \forall \theta \in \mathbb{R}^n, \omega \in \mathbb{R}^m. \tag{2.2}$$

In the language of game theory, we say that $(\theta^\star, \omega^\star)$ is a *pure* Nash Equilibrium (pure NE). If (2.2) holds only locally, we say that $(\theta^\star, \omega^\star)$ is a local pure NE.

When the function f is convex-concave, i.e., $f(\cdot, \mathbf{y})$ is convex for all $\mathbf{y}$, and $f(\mathbf{x}, \cdot)$ is concave for all $\mathbf{x}$, existence of such a pair is guaranteed to exist provided that the constraint set is compact (Neumann, 1928; Rosen, 1965), and computing such a solution can be reduced to solve a convex program.

However, when f is *not* convex-concave, such a solution is not guaranteed to exist, and various hardness results have been proved. In particular, deciding whether an approximate local Nash equilibrium exists is NP-hard, and finding such a point, even when guaranteed to exist, is PPAD-complete (Daskalakis et al., 2021).

2.1.2 Finding mixed Nash equilibria

In this section, we present an existing framework (Hsieh et al., 2019) for solving problems of the form (2.1). The goal of this section is to present the main ideas of the procedure, and technical details will be omitted. For a more formal treatment of the algorithm's design, please refer to the original paper.

It seems from the previous section that the solution concept of pure Nash equilibrium is not satisfactory in the general case where the objective function f is not convex-concave, since such a solution may not exist, or can be very hard to find. As a possible solution, Hsieh et al. (2019) propose to relax problem (2.1) by introducing stochastic strategies, i.e., they consider the following two players game:

$$\min_{\mu \in \mathscr{P}(\Theta)} \max_{\nu \in \mathscr{P}(\Omega)} \mathbb{E}_{\theta \sim \mu} \left[\mathbb{E}_{\omega \sim \nu} \left[f(\theta, \omega) \right] \right], \tag{2.3}$$

where $\mathscr{P}(\mathscr{Z})$ denotes the space of probability distributions over $\mathscr{Z}$. A pair $(\mu^\star, \nu^\star)$ achieving the min-max value in (2.3) is called a *mixed Nash Equilibrium* (mixed NE).

Although the problem becomes infinite dimensional, due to the optimization over the space of probability measures, it also becomes bilinear thanks to the linearity of the expectation, which guarantees the existence of a *mixed* Nash equilibrium. Moreover, by mimicking the Entropic Mirror Descent algorithm for finite-dimensional bilinear games, Hsieh et al. (2019) propose Algorithm 1 to find such a solution of (2.3). Denoting by $\mathscr{F}(\mathscr{Z})$ the space of functions over $\mathscr{Z}$, the operators $G : \mathscr{P}(\Theta) \to \mathscr{F}(\Omega)$ and its adjoint $G^\dagger : \mathscr{P}(\Omega) \to \mathscr{F}(\Theta)$ in Algorithm 1 are

Algorithm 1 Infinite-dimensional Entropic MD

Input: Initial distributions μ_1, ν_1, learning rate η.
for $t = 1, 2, \ldots, T-1$ **do**
  $\mu_{t+1} \leftarrow \mathrm{MD}_\eta(\mu_t, G^\dagger \nu_t)$
  $\nu_{t+1} \leftarrow \mathrm{MD}_\eta(\nu_t, -G\mu_t)$
Return $\bar{\mu}_T = \frac{1}{T}\sum_{t=1}^{T} \mu_t$ and $\bar{\nu}_T = \frac{1}{T}\sum_{t=1}^{T} \nu_t$.

Algorithm 2 Infinite-dimensional Entropic MD

Input: $\theta_1, \omega_1 \leftarrow$ random initialization, step sizes $\{\gamma_t\}_{t=1}^{T}$, thermal noises $\{\epsilon_t\}_{t=1}^{T}$, warm-up steps $\{K_t\}_{t=1}^{T-1}$, damping factor β.
for $t = 1, 2, \ldots, T-1$ **do**
  $\bar{\theta}_t, \theta^{(1)} \leftarrow \theta_1, \bar{\omega}_t, \omega^{(1)} \leftarrow \omega$
  for $k = 1, 2, \ldots, K_t$ **do**
    $\theta_t^{k+1} \leftarrow \theta_t^k - \gamma_t \nabla_\theta f(\theta_t^{(k)}, \omega_t) + \sqrt{2\gamma_t}\epsilon_t \xi$, where $\xi \sim \mathcal{N}(0, I)$
    $\omega_t^{k+1} \leftarrow \omega_t^k + \gamma_t \nabla_\omega f(\theta_t, \omega_t^{(k)}) + \sqrt{2\gamma_t}\epsilon_t \xi'$, where $\xi' \sim \mathcal{N}(0, I)$
    $\bar{\theta}_t \leftarrow (1-\beta)\bar{\theta}_t + \beta\theta_t^{(k+1)}, \quad \bar{\omega}_t \leftarrow (1-\beta)\bar{\omega}_t + \beta\omega_t^{(k+1)}$
  $\theta_{t+1} \leftarrow (1-\beta)\theta_t + \beta\bar{\theta}_t, \quad \omega t + 1 \leftarrow (1-\beta)\omega_t + \beta\bar{\omega}_t$
Return θ_T, ω_T.

defined as follows:

$$G\mu(\omega) \equiv \mathbb{E}_{\theta\sim\mu}[f(\theta,\omega)]$$
$$G^\dagger \nu(\theta) \equiv \mathbb{E}_{\omega\sim\nu}[f(\theta,\omega)]$$

The Mirror Descent operator MD can be defined as follows: Let μ be an arbitrary probability distribution and h a regular enough function. Then, for $\eta > 0$, the distribution $\mu_+ = \mathrm{MD}_\eta(\mu, h)$ is defined as

$$d\mu_+ = \frac{e^{-\eta h}\,\mathrm{d}\mu}{\int e^{-\eta h}\,\mathrm{d}\mu}.$$

It is shown in (Hsieh et al., 2019) that, with a proper choice of learning rate η, Algorithm 1 achieves a $\mathcal{O}(T^{-1/2})$-NE, i.e., $\max_{\mu,\nu}\left[\mathbb{E}_{\theta\sim\mu_T}\left[\mathbb{E}_{\omega\sim\nu}\left[f(\theta,\omega)\right]\right] - \mathbb{E}_{\theta\sim\mu}\left[\mathbb{E}_{\omega\sim\nu_T}\left[f(\theta,\omega)\right]\right]\right] = \mathcal{O}(T^{-1/2})$.

However, a significant issue with Algorithm 1 is that it is not implementable, since it requires to iteratively sample from certain distributions. Hence, the authors propose an implementable version by approximating each sampling step using Langevin dynamics, which will be the topic of the next section. This gives rise to Algorithm 2 whose goal is to sample a pair θ_T, ω_T approximately following the respective distributions μ_T, ν_T as defined in Algorithm 1.

2.2 Double-loop unadjusted Langevin algorithm

In order to understand the transition from Algorithm 1 to its implementable version (Algorithm 2), we now introduce a sampling algorithm know as the Unadjusted Langevin Algorithm (ULA). ULA is a first order sampling algorithm which can be seen as the sampling counter-part of Gradient Descent in optimization. In this section, we analyse the convergence properties of ULA in the case where the target distribution is (weakly) log-concave, and propose a specific step size schedule that allows to obtain an improved convergence rate in certain regimes.

This section is adapted from the paper (Rolland et al., 2020) published at ICML 2020.

2.2.1 Introduction

Let $\mathrm{d}\mu^*(x) \propto e^{-f(x)}\,\mathrm{d}x$ be a probability measure over $\mathbb{R}^d$, where $f:\mathbb{R}^d \to \mathbb{R}$ is a convex function with Lipschitz continuous gradient. In order to sample from such distributions, first-order sampling schemes based on the discretization of Langevin dynamics and, in particular the Unadjusted Langevin Algorithm (ULA), have found widespread success in various applications (Welling and Teh, 2011; Li et al., 2016b; Patterson and Teh, 2013; Li et al., 2016a). An ever-growing body of literature has been devoted solely to the study of ULA and its variations (Ahn et al., 2012; Chen et al., 2015; Cheng and Bartlett, 2017; Cheng et al., 2017a; Dalalyan and Karagulyan, 2017; Durmus et al., 2017, 2018a; Dwivedi et al., 2018; Luu et al., 2017; Welling and Teh, 2011; Ma et al., 2015).

The ULA iterates are given as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_{k+1}\nabla f(\mathbf{x}_k) + \sqrt{2\gamma_{k+1}}\xi_k, \tag{2.4}$$

where ∇f denotes the gradient of f, $\{\gamma_k\}_{k\geq 0}$ is a non-increasing sequence of positive step-sizes, and the entries of $\xi_k \in \mathbb{R}^d$ are zero-mean and unit-variance Gaussian random variables, independent from each another and everything else. In its standard form (2.4), ULA can provably sample from any log-concave and smooth probability measure (Durmus et al., 2017, 2018a).

The recent analysis of Durmus et al. (2018a) studies ULA through the lens of convex optimization. Their analysis shows strong resemblance with the convergence analysis of stochastic gradient descent (SGD) algorithm for minimizing a convex continuously differentiable function $f:\mathbb{R}^d \to \mathbb{R}$. Starting from $\mathbf{x}_0 \in \mathbb{R}^d$, SGD iterates similarly as (2.4):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_{k+1}\nabla f(\mathbf{x}_k) + \gamma_{k+1}\Theta(\mathbf{x}_k),$$

where $\Theta:\mathbb{R}^d \to \mathbb{R}^d$ is a stochastic perturbation to ∇f. One way of proving convergence guarantees for SGD is to show the following inequality (Beck and Teboulle, 2009):

$$2\gamma_{k+1}(\mathbb{E}[f(\mathbf{x}_{k+1})] - f(\mathbf{x}^*)) \leq \mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}^*\|_2^2\right] - \mathbb{E}\left[\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_2^2\right] + C\gamma_{k+1}^2, \tag{2.5}$$

for some constant $C \geq 0$, $\forall k \geq 0$ and $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$. From this inequality, and using step size $\gamma_k \propto \frac{1}{\sqrt{k}}$, it is then possible to show convergence, in expectation, of the average iterate $\bar{\mathbf{x}}_T = \frac{1}{T}\sum_{t=0}^{T-1} \mathbf{x}_t$ to the optimal value, i.e., $\mathbb{E}[f(\bar{\mathbf{x}}_T)] - f(\mathbf{x}^*) = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$.

In their paper, Durmus et al. (2018a) showed a similar *descent Lemma* as (2.5) for the sequence of generated measures $\{\mu_k\}_{k\geq 0}$ denoting the distributions of the iterates $\{\mathbf{x}_k\}_{k\geq 0}$ in (2.4), in which the objective gap $\mathbb{E}[f(\mathbf{x}_k)] - f(\mathbf{x}^*)$ is replaced with the Kullback-Leibler divergence $\mathrm{KL}(\mu_k;\mu^*)$, and the Euclidean distance $\|\mathbf{x}_k - \mathbf{x}^*\|_2$ is replaced with the 2-Wasserstein distance $\mathrm{W}_2(\mu_k,\mu^*)$, i.e., they showed

$$2\gamma_{k+1}\,\mathrm{KL}(\mu_k;\mu^*) \leq \mathrm{W}_2^2(\mu_k,\mu^*) - \mathrm{W}_2^2(\mu_{k+1},\mu^*) + 2Ld\gamma_{k+1}^2, \tag{2.6}$$

where L is the Lipschitz constant of the gradient of f. Then again, using $\gamma_k \propto \frac{1}{\sqrt{k}}$, it is possible to show convergence of the average sample distribution $\bar{\mu}_T = \frac{1}{T}\sum_{t=0}^{T}\mu_t$ to μ^* in KL divergence, with rate $\mathcal{O}\left(\frac{d^3}{\sqrt{T}}\right)$.

In this work, we build upon the work of Durmus et al. (2018a), and introduce a new multi-stage decaying step size schedule, which proceeds in a double loop fashion by geometrically decreasing the step-size after a certain number of iterations, and that we call Double-loop ULA (DL-ULA). By properly choosing the step size decay and the number of iterations per step size, we can prove new convergence guarantees, that improves the state-of-the-art in a certain range of accuracy ϵ and dimension d, in TV distance and KL divergence.

To the best of our knowledge, all existing convergence proof for ULA use either constant, or polynomially decaying step sizes, i.e. of the form $\gamma_k = k^{-\alpha}$ for some $\alpha \geq 0$, and this is the first work introducing a multistage decaying step size for a sampling algorithm. Interestingly, there is precedence to support our approach in that such step decay schedule can improve convergence of optimization algorithms (Ge et al., 2019; Aybat et al., 2019).

In our analysis, we prove and exploit a new bound that relates the W_2 distance and the KL divergence between any two log-concave distributions (Lemma 10). This inequality serves as an alternative to the powerful T_2 inequality (Gozlan and Léonard, 2010), the latter requiring stronger assumptions on the distributions. The literature on Langevin dynamics commonly proves the convergence of an algorithm in KL divergence and then extends it to the total variation (TV) distance using the famous Pinsker's inequality (Pinsker, 1960; Cheng and Bartlett, 2017; Durmus et al., 2018a). Our new inequality enables to do the same for extending convergence results to W_2 distance in the case of general log-concave distributions.

Finally, we apply this multistage strategy to the constrained sampling algorithm MYULA (Brosse et al., 2017), which allows us to obtain improved convergence guarantees, both in terms of rate and dimension dependence. This approach provides state-of-the-art convergence guarantees for sampling from a log-concave distribution over a general convex set.

We summarize our contributions as follows:

- We introduce a variant of the Unadjusted Langevin Algorithm, using a new multistage decaying step-size schedule as well as a normalization step. Our new approach, called DL-ULA, yields new convergence guarantees, that are not covered by existing convergence results (i.e., either better convergence rate or better dimension dependence compared to state-of-the-art results).

- We apply our new step-size schedule to an existing Langevin-based constrained sampling algorithm, called MYULA (Brosse et al., 2017), and improve its convergence both in terms of iteration and dimension dependences.

- We introduce a new bound relating the 2-Wasserstein and the TV distance between any two log-concave distributions.

A summary of our convergence rates can be found in Tables 2.1 and 2.2.

2.2.2 Related work

Unconstrained sampling. Sampling algorithms based on Langevin dynamics have been widely studied (Ahn et al., 2012; Chen et al., 2015; Cheng and Bartlett, 2017; Cheng et al., 2017a; Dalalyan and Karagulyan, 2017; Durmus et al., 2018a; Dwivedi et al., 2018; Durmus et al., 2017; Luu et al., 2017; Welling and Teh, 2011). Most convergence rates have been established in the strongly log-concave setting, or under log-Sobolev inequality (LSI). In this work, we focus on the general log-concave setting, and we wish in particular to allow for distributions whose tail decay as $e^{-\|x\|_2}$ for which LSI does not hold. Because of their fatter tails, the complexity for sampling from such distributions in particular exhibits larger dimension dependences.

Convergence guarantees for ULA applied to a general unconstrained log-concave distribution have been successively improved over the years (see Table 2.1). Various works feature different variants of ULA, such as averaging (Durmus et al., 2018a), strongly convex regularization (Dalalyan et al., 2019), or underdamping (Zou et al., 2018). In this work, we also include a small modification by involving a normalization step.

We are particularly careful in characterizing the dimension dependence in our convergence guarantees. Various works include the Poincaré constant C_{PI} inside their convergence guarantees (Chewi et al., 2021; Lehec, 2021). However, this constant can include dimension dependence. For distribution with tail decaying as $e^{-\|x\|_2}$, we have $C_{PI} = \mathcal{O}(d)$, which should be taken into consideration when reading Table 2.1.

Constrained sampling. Extensions of ULA have been designed in order to sample from constrained distributions (Bubeck et al., 2018; Brosse et al., 2017; Hsieh et al., 2018; Patterson and Teh, 2013). In (Bubeck et al., 2018), the authors propose to apply ULA, and project the sample onto the constraint at each iteration. They show a convergence rate of $\mathcal{O}(d^{12}\epsilon^{-12})$ in TV distance for log-concave distributions.

In (Brosse et al., 2017), the authors propose to smooth the constraint using its Moreau-Yoshida envelope, and obtain a convergence rate of $\mathcal{O}(d^5\epsilon^{-6})$ in TV distance when the objective distribution is log-concave. To do so, they penalize the domain outside the constraint via its Moreau-Yoshida envelop, and sample from the penalized unconstrained distribution using a penalty parameter depending on the desired accuracy.

The analysis of MYULA in (Brosse et al., 2017) only holds when the penalty parameter is fixed and chosen in advance, leading to a natural saturation after a certain number of iterations. In this work, we extend this procedure using ourmulti-stage approach. This allows us to obtain improved convergence both in terms of rate and dimension dependence, i.e., $\mathcal{O}(d^{3.5}\epsilon^{-5})$ in TV distance, and to ensure asymptotic convergence of the algorithm since the penalty is allowed to vary along the iterations.

In a different paper (Hsieh et al., 2018), which is not presented in details here, we also solve the special case of sampling from the simplex, i.e., $\{\mathbf{x} \in \mathbb{R}^d : \sum_{i=1}^d x_i \leq 1, x_i \geq 0\}$, by introducing Mirrored Langevin Dynamics (MLD). Our work relies on finding a mirror map for the given constrained domain, and then performing ULA in the dual space. This method yields the best known convergence rates for constrained sampling, matching the complexity of unconstrained sampling. However, this method requires log-concavity of the distribution in the dual space, which is not straightforward to ensure in general. Moreover, finding a suitable mirror map for a general convex set is not an easy task.

2.2.3 Preliminaries

Various measures between distributions

Let us recall classical distances/divergences between probability measures which will be used in this section. The Kullback–Leibler (KL) divergence between two probability measures μ, ν on $\mathbb{R}^d$ is defined as

$$\mathrm{KL}(\mu;\nu) = \mathbb{E}_\mu \log(\mathrm{d}\mu/\,\mathrm{d}\nu), \tag{2.7}$$

assuming that μ is dominated by ν. Their Total Variation (TV) distance is defined as

$$\|\mu - \nu\|_{\mathrm{TV}} = \sup_{S \subseteq \mathbb{R}^d} |\mu(S) - \nu(S)|, \tag{2.8}$$

where the supremum is over all measurable subsets S of $\mathbb{R}^d$.

Finally, the 2-Wasserstein distance (W_2) between μ and ν is defined as

$$\mathrm{W}_2^2(\mu,\nu) = \inf_{\gamma \in \Phi(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|\mathbf{x}-\mathbf{y}\|_2^2 d\gamma(\mathbf{x},\mathbf{y}), \tag{2.9}$$

where $\Phi(\mu,\nu)$ denotes the set of all joint probability measures γ on $\mathbb{R}^{2d}$ that marginalize to μ

and ν, namely, such that for all measurable sets $A, B \subseteq \mathbb{R}^d$, $\gamma(A \times \mathbb{R}^d) = \mu(A)$ and $\gamma(\mathbb{R}^d \times B) = \nu(B)$.

The main difference between W_2 and TV distances is that W_2 associates a higher cost when the difference between the distributions occurs at points that are further appart (in terms of Euclidean distance). Due to this property, errors occurring at the tail of the distributions (i.e., when $\|x\|_2 \to \infty$) can have a small impact in terms of TV distance, but a major impact in terms of W_2 distance.

Log-concave distributions and tail properties

We now recall the basic properties that we will assume on the probability measure. We will then present some known results about this class of measures which will be exploited in the convergence analysis of our algorithm.

Definition 1. *We say that a function $f : \mathbb{R}^d \to \mathbb{R}$ has L-Lipschitz continuous gradient for $L \geq 0$ if $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,*

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_2.$$

Definition 2. *We say a function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $\forall 0 \leq t \leq 1$ and $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,*

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq t f(\boldsymbol{x}) + (1-t) f(\boldsymbol{y}).$$

Definition 3. *We say that probability measure $\mu \propto e^{-f(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}$ is log-concave if f is convex. Moreover, we say that μ is L-smooth if f has a L-Lipschitz continuous gradient.*

As mentioned previously, bounding the Wasserstein distance between two probability measures requires controlling the error at the tail of the distributions. In order to deal with such a distance without injecting large dimension dependence, we make the following assumption on the tail of the target distribution, which is quite standard when working with unconstrained non-strongly log-concave distributions (Durmus et al., 2018a, 2017):

Assumption 4. *There exists $\eta > 0, M_\eta > 0$ such that for all $x \in \mathbb{R}^d$ such that $\|\boldsymbol{x}\|_2 \geq M_\eta$,*

$$f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \geq \eta \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2,$$

where $\boldsymbol{x}^\star = \arg\min_{x \in \mathbb{R}^d} f(x)$. For simplicity, we will also assume $\boldsymbol{x}^\star = 0$ and $f(\boldsymbol{x}^\star) = 0$.

Note that in the case of a distribution constrained to a set $\Omega \subset \mathbb{R}^d$, this assumption is naturally satisfied with η arbitrary, and $M_\eta = \mathrm{diam}(\Omega)$ where $\mathrm{diam}(\Omega)$ is the diameter of Ω.

In order to see how this assumption transfers into a constraint on the tail of the distribution, we recall two following results shown in (Durmus et al., 2018a) and (Lovász and Vempala, 2007), respectively.

Lemma 5 (Durmus et al. (2018a)). *Let $X \in \mathbb{R}^d$ be a random vector from a log-concave distribution μ satisfying Assumption 4. Then, it holds that*

$$\mathbb{E}_{X\sim\mu}\left[\|X\|_2^2\right] \leq \frac{2d(d+1)}{\eta^2} + M_\eta^2.$$

Lemma 6 (Lovász and Vempala (2007)). *Let $X \in \mathbb{R}^d$ be a random vector from a log-concave distribution μ such that $\mathbb{E}\left[\|X\|_2^2\right] \leq C^2$. Then, for any $R > 1$, we have that*

$$\Pr\left(\|X\|_2 > RC\right) < e^{-R+1}. \tag{2.10}$$

It is thus possible to combine both Lemmata to show that any distribution satisfying Assumption 4 necessarily has a sub-exponential tail. This property will allow us to control the Wasserstein distance in terms of the Total Variation distance.

Lemma 7. *Let X be a random vector from a log-concave distribution μ satisfying Assumption 4. Then, $\forall R > 1$, we have that*

$$\Pr\left(\|X\|_2 > R\sqrt{\frac{2d(d+1)}{\eta^2} + M_\eta}\right) < e^{-R+1}. \tag{2.11}$$

Unadjusted Langevin Algorithm

Finally, we recall the standard Unadjusted Langevin Algorithm as well as a very useful inequality bounding the KL divergence between the target distribution and the k-th iterate distribution.

Consider the probability space $(\mathbb{R}^d, \mathscr{B}, \mu^*)$, where $\mathscr{B}$ is the Borel sigma algebra and μ^* is the target distribution. Suppose that μ^* is log-concave and is dominated by the Lebesgue measure on $\mathbb{R}^d$, namely,

$$\mathrm{d}\mu^*(\mathbf{x}) = Ce^{-f(\mathbf{x})}\,\mathrm{d}\mathbf{x}, \qquad \forall x \in S, \tag{2.12}$$

where C is an unknown normalizing constant and the function $f : \mathbb{R}^d \to \mathbb{R}$ is convex and has L-Lipschitz continuous gradient.

A well-known scheme for sampling for such a distribution without calculating its normalizing constant C is called ULA. Initialized at $\mathbf{x}_0 \in \mathbb{R}^d$, the ULA constructs a random sequence of iterates as defined in equation (2.4).

Let μ_k be the probability measure associated to iterate x_k, $\forall k \geq 0$. It is well-known that ULA

converges to the target measure in KL divergence. More specifically, for $n \geq n_\epsilon = \mathcal{O}(d^3 L\epsilon^{-2})$ iterations, we reach $\mathrm{KL}(\overline{\mu}_n; \mu^*) \leq \epsilon$, where $\overline{\mu}_n = \frac{1}{n}\sum_{k=1}^n \mu_k$ is the average of the probability measures associated to the iterates $\{x_k\}_{k=0}^n$ (Durmus et al., 2018a). The averaging sum $\frac{1}{n}\sum_{k=1}^n \mu_k$ is to be understood in the sense of measures, i.e., sampling from the $\bar{\mu}_n$ is equivalent to choosing an index k uniformly at random among $\{1, ..., n\}$, and then sampling from μ_k.

To prove this result, the authors showed the following useful inequality that we will exploit in our analysis:

Lemma 8 (Durmus et al. (2018a))**.** *Suppose that we apply ULA* (2.4) *for sampling from a L-smooth log-concave distribution* $\mu^* \propto e^{-f(x)}\,\mathrm{d}x$ *with constant step-size* $0 < \gamma < \frac{1}{L}$*, starting from* $x_0 \sim \mu_0$*. Then,* $\forall\, n > 0$*, it holds that*

$$\mathrm{KL}(\bar{\mu}_n; \mu^*) \leq \frac{\mathrm{W}_2^2(\mu_0, \mu^*)}{2\gamma n} + Ld\gamma. \tag{2.13}$$

2.2.4 DL-ULA for unconstrained sampling

In this section, we present a modified version of the standard ULA for sampling from an unconstrained distribution and provide convergence guarantees. This modified version of ULA involves a new step size schedule as well as a projection step.

DL-ULA algorithm

We consider the problem of sampling from a smooth and unconstrained probability measure $\mu^* \propto e^{-f(x)}\,\mathrm{d}x$, where $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable and convex. To this end, we apply the standard ULA in a double-loop fashion, and decrease the step size only between each inner loop. Moreover, each inner loop is followed by a projection step onto some Euclidean ball. The procedure is summarized in Algorithm 3.

The projection step appears to be crucial in our analysis in order to control the tail of the sample distribution, which is necessary for bounding its Wasserstein distance to the target distribution.

In the following sections, we derive the convergence rate for Algorithms 3. The global idea is to use the inequality (2.13) recursively between each successive outer loop. We denote as $\bar{\mu}_k$ the average distribution associated with the iterates of outer iteration k just *before* the projection step. Similarly, we denote as $\hat{\mu}_k$ the same distribution *after* the projection step.

Each outer iteration k uses as a starting point a sample from the previous outer iteration $x_{k,0} \sim \hat{\mu}_{k-1}$. Therefore, we can apply the inequality (2.13) to the outer iteration k to obtain

$$\mathrm{KL}(\bar{\mu}_k; \mu^*) \leq \frac{\mathrm{W}_2^2(\hat{\mu}_{k-1}, \mu^*)}{2\gamma_k n_k} + Ld\gamma_k. \tag{2.14}$$

Algorithm 3 Double-loop Unadjusted Langevin Algorithm (DL-ULA)

Input: Smooth unconstrained probability measure μ^*, step sizes $\{\gamma_k\}_{k\geq 0}$, number of (inner) iterations $\{n_k\}_{k\geq 0}$, thresholds $\{\tau_k\}_{k\geq 1}$. and initial probability measure μ_0 on $\mathbb{R}^d$.
Initialization: Draw a sample $\mathbf{x}_0$ from the probability measure μ_0.
for $k = 0, \dots$ **do**
  $\mathbf{x}_{k,0} \leftarrow \mathbf{x}_k$
  Draw N_k uniformly from $\{1, \cdots, n_k\}$.
  for $n = 0, \dots, N_k - 1$ **do**
    $\mathbf{x}_{k,n+1} \leftarrow \mathbf{x}_{k,n} - \gamma_k \nabla f(\mathbf{x}_{k,n}) + \sqrt{2\gamma_k}\xi_{k,n}$, where $\xi_{k,n} \sim \mathcal{N}(0, I_d)$.
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_{k,N_k}$.
  if $\|\mathbf{x}_{k+1}\|_2 > \tau_{k+1}$ **then**
    $\mathbf{x}_{k+1} \leftarrow \tau_{k+1}\mathbf{x}_{k+1}/\|\mathbf{x}_{k+1}\|_2$.

In order to unfold the recursion, we must have a bound on $\mathrm{W}_2^2(\hat{\mu}_{k-1}, \mu^*)$ in terms of $\mathrm{KL}(\bar{\mu}_{k-1}, \mu^*)$. Using the tail property of log-concave distributions (Lemma 7), it is easy to obtain a bound between $\mathrm{W}_2^2(\hat{\mu}_{k-1}, \mu^*)$ and $\mathrm{W}_2^2(\bar{\mu}_{k-1}, \mu^*)$. However, it is not clear how to bound $\mathrm{W}_2^2(\bar{\mu}_{k-1}, \mu^*)$ by $\mathrm{KL}(\bar{\mu}_{k-1}, \mu^*)$.

As an intermediate step in the convergence analysis, we derive in the next section a bound between the W_2-distance and the TV-distance between two general log-concave probability measures, which can then be extended to a W_2-KL bound using Pinsker's inequality.

Relation Between W_2- and TV-Distances

The Total Variation distance can be seen as a Wasserstein distance, where the Euclidean distance is replaced with an indicator function. Indeed, we can write (Gibbs and Su, 2002)

$$\|\mu - \nu\|_{\mathrm{TV}} = \inf_{\gamma \in \Phi(\mu,\nu)} \mathbb{E}_{(X,Y)\sim\gamma}[1_{X\neq Y}].$$

The challenge for bounding W_2 with TV distance is that the former can associate arbitrary large cost when the error occurs *at the tail* of the distribution, i.e., when $\|x\|$ becomes very large. For any distributions μ and ν compactly supported on an Euclidean ball of diameter D, the cost involved in W_2 can be bounded, and we hence have $\mathrm{W}_2(\mu,\nu) \leq D\sqrt{\|\mu-\nu\|_{\mathrm{TV}}}$ (Gibbs and Su, 2002).

When μ and ν are not compactly supported, the transportation cost can be arbitrary large, especially when transporting mass from/to the set $\bar{B}_R = \{x \in \mathbb{R}^d : \|x\|_2 \geq R\}$. However, if both μ and ν are log-concave, we can exploit their tail property (Lemma 7) to ensure that the mass of $\bar{B}_R$ under both distributions decreases exponentially with R, hence inducing the following bound:

Lemma 9. *Let μ, ν be distributions both satisfying the following inequality, for some $c, C > 0$*

and $R \geq C$:

$$\Pr(\|X\|_2 \geq t) \leq ce^{-\frac{t}{C}}, \ \forall t \geq R. \tag{2.15}$$

Then, it holds that

$$\mathrm{W}_2^2(\mu,\nu) \lesssim R^2\|\mu-\nu\|_{\mathrm{TV}} + R^2 e^{-\frac{R}{C}}. \tag{2.16}$$

Note that Lemma 9 does not require the distributions to be log-concave, but only to have sub-exponential tails. This will be useful in our convergence proof, since the iterate distributions of ULA (using finite step size) are in general not log-concave. Finally, since log-concave distributions, under Assumption 4, satisfy equation (2.15) with $C = \sqrt{\frac{2d(d+1)}{\eta^2} + M_\eta}$ (Lemma 7), we can show the following bound between W_2 and TV distances:

Lemma 10 (W_2-TV distances inequality)**.** *Let μ,ν be log-concave probability measures on $\mathbb{R}^d$ satisfying Assumption 4 with (η, M_η). Then, we have that*

$$\mathrm{W}_2(\mu,\nu) \lesssim \sqrt{\frac{2d(d+1)}{\eta^2} + M_\eta}\max\left(\log\left(\frac{1}{\|\mu-\nu\|_{\mathrm{TV}}}\right), 1\right)\sqrt{\|\mu-\nu\|_{\mathrm{TV}}}. \tag{2.17}$$

In a sense, (2.17) is an alternative to the powerful T_2 inequality which does not hold generally in our setting (Gozlan and Léonard, 2010). Indeed, for $C_\mu > 0$, recall that a probability measure μ satisfies Talagrand's $T_2(C_\mu)$ transportation inequality if we have

$$\mathrm{W}_2(\mu,\nu) \leq C_\mu\sqrt{\mathrm{KL}(\mu;\nu)}, \tag{2.18}$$

for any probability measure ν. Above, C_μ depends only on μ and, in particular, if μ is κ-strongly log-concave,[1] then (2.18) holds with $C_\mu = \mathcal{O}(1/\sqrt{\kappa})$ (Gozlan and Léonard, 2010). In this work, the target measure that we consider is not necessarily strongly log-concave measures, leaving us in need for a replacement to (2.18).

In the case where the difference between the mean of the two distributions can be controlled, using Pinsker's inequality (Pinsker, 1960) on (2.17) yields

$$\mathrm{W}_2(\mu,\nu) = \widetilde{\mathcal{O}}(\mathrm{KL}(\mu;\nu)^{\frac{1}{4}}), \tag{2.19}$$

which can serve as a replacement for (2.18). Equation (2.17) is of interest in its own right, especially when working with non-strongly log-concave measures.

Convergence Analysis of DL-ULA

Now that we have covered the necessary technical tools above, we turn our attention to the convergence analysis of Algorithm 3, summarized in Theorem 11. We provide here the main lines of the analysis, and postpone the detailed proof to Appendix A.1.3.

[1] If $\mathrm{d}\mu \propto e^{-f}\,\mathrm{d}x$, then we say that μ is κ is strongly log-concave if f is κ-strongly convex.

Literature	W_2	TV	KL
Durmus et al. (2018a)	-	$\widetilde{\mathcal{O}}\left(Ld^3\epsilon^{-4}\right)$	$\widetilde{\mathcal{O}}\left(Ld^3\epsilon^{-2}\right)$
Durmus et al. (2017)	-	$\widetilde{\mathcal{O}}\left(L^2d^5\epsilon^{-2}\right)$	-
Zou et al. (2018)	$\widetilde{\mathcal{O}}\left(L^2d^{10.5}\epsilon^{-6}\right)^*$	-	-
Dalalyan et al. (2019)	$\widetilde{\mathcal{O}}\left(Ld^9\epsilon^{-6}\right)^\natural$	-	-
Chewi et al. (2021)	-	$\widetilde{\mathcal{O}}(L^2d^2C_{PI}^2\epsilon^{-2})^\star$	$\widetilde{\mathcal{O}}(L^2d^2C_{PI}^2\epsilon^{-1})^\star$
Lehec (2021)	$\widetilde{\mathcal{O}}\left(L_f^2d^4C_{PI}^3\epsilon^{-4}\right)^\#$	-	-
Our work	$\widetilde{\mathcal{O}}\left(Ld^9\epsilon^{-6}\right)$	$\widetilde{\mathcal{O}}\left(Ld^3\epsilon^{-3}\right)$	$\widetilde{\mathcal{O}}\left(Ld^3\epsilon^{-\frac{3}{2}}\right)$

Table 2.1 – Complexity of sampling from a smooth and log-concave probability distribution when using different variants of ULA. For each metric, the entry corresponds to the total number of iterations to use in order to reach an ϵ accuracy in the specified metric.
* Zou et al. (2018) make the assumption that $\mathbb{E}_{X\sim\mu}\left[\|X\|_2^4\right] \leq \bar{U}d^2$ for some scalar $\bar{U}$. For comparison purpose, we extended the proof in Zou et al. (2018) in the case where the distribution satisfies the weaker Assumption 4.
$^\natural$ Dalalyan et al. (2019) analyse a variant of ULA, called α-LMC, which runs ULA on the regularized distribution $\mathrm{d}\mu_\alpha^*(x) \propto e^{-f(x)-\alpha\|x\|^2}$ for small enough α so as to leverage convergence results on strongly log-concave distributions.
* C_{PI} denotes the Poincaré constant of the target distribution (Bobkov, 1999; Bakry et al., 2008). This constant can be dimension dependent, especially for weakly log-concave distribution. Under Assumption 4, we have the bound $C_{PI} = \mathcal{O}(d)$.
$^\#$ In (Lehec, 2021), the function f is assumed to be L_f-Lipschitz continuous instead of gradient Lipschitz.

Recall that $\bar{\mu}_k$ denotes the iterate distribution after k outer steps of Algorithm 3 just *before* the projection step, i.e., the distribution of $\mathbf{x}_k$ in line 8, and $\hat{\mu}_k$ denotes the same distribution after the projection, which is used as initialisation for the next outer iteration. Hence, after running n_k iterations of ULA (equation 2.4) starting with a sample from $\hat{\mu}_{k-1}$, we obtain a sample from $\bar{\mu}_k$.

Hence, starting from equality (2.14) and using Pinsker inequality, we have for all $k \geq 1$

$$\begin{aligned}\|\bar{\mu}_{k+1} - \mu^*\|_{\mathrm{TV}} &\leq \sqrt{2\,\mathrm{KL}(\bar{\mu}_{k+1};\mu^*)} \\ &\leq \sqrt{\frac{W_2^2(\hat{\mu}_k,\mu^*)}{\gamma_k n_k} + 2Ld\gamma_k} \\ &\leq \frac{\mathrm{W}_2(\hat{\mu}_k,\mu^*)}{\sqrt{\gamma_k n_k}} + \sqrt{2Ld\gamma_k}. \end{aligned} \tag{2.20}$$

To obtain a recursion formula, we hence need to bound $\mathrm{W}_2(\hat{\mu}_k,\mu^*)$ by $\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}}$. To this end, we first bound $\mathrm{W}_2(\hat{\mu}_k,\mu^*)$ using $\|\hat{\mu}_k - \mu^*\|_{\mathrm{TV}}$ as allowed by Lemma 9.

Thanks to the log-concavity of μ^*, Lemma 7 implies that μ^* satisfies the condition (2.15) with $C = C_\eta \equiv \sqrt{\frac{2d(d+1)}{\eta^2} + M_\eta}$. Moreover, the projected distribution $\hat{\mu}_k$ naturally satisfies

condition (2.15) as long as $R \geq \tau_k$ since $\Pr_{X\sim\hat{\mu}_k}(\|X\|_2 > \tau_k) = 0$. Hence, by applying Lemma 7 with $C = C_\eta$ and $R = \tau_k$, and choosing $\tau_k \equiv C_\eta k$, we obtain

$$\mathrm{W}_2^2(\hat{\mu}_k, \mu^*) \lesssim C_\eta^2 k^2 \|\hat{\mu}_k - \mu^*\|_{\mathrm{TV}} + C_\eta^2 k^2 e^{-k}. \tag{2.21}$$

We now need to bound $\|\hat{\mu}_k - \mu^*\|_{\mathrm{TV}}$ with $\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}}$. Using the triangle inequality, and recalling that $\hat{\mu}_k$ is the projected version of $\bar{\mu}_k$ onto a ball of radius $\tau_k = C_\eta k$, we have

$$\begin{aligned}\|\hat{\mu}_k - \mu^*\|_{\mathrm{TV}} &\leq \|\hat{\mu}_k - \bar{\mu}_k\|_{\mathrm{TV}} + \|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}} \\ &\leq \Pr_{X\sim\bar{\mu}_k}(\|X\|_2 > C_\eta k) + \|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}}.\end{aligned}$$

Then, using the fact that $\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}} \geq |\Pr_{X\sim\bar{\mu}_k}(\|X\|_2 > C_\eta k) - \Pr_{X\sim\mu^*}(\|X\|_2 > C_\eta k)|$ and $\Pr_{X\sim\mu^*}(\|X\|_2 > C_\eta k) \lesssim e^{-k}$, we obtain

$$\|\hat{\mu}_k - \mu^*\|_{\mathrm{TV}} \lesssim 2\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}} + e^{-k}. \tag{2.22}$$

Hence, putting equations (2.20), (2.21) and (2.23) together, we obtain

$$\|\bar{\mu}_{k+1} - \mu^*\|_{\mathrm{TV}} \lesssim \frac{kC_\eta\sqrt{\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}}} + kC_\eta e^{-k/2}}{\sqrt{\gamma_k n_k}} + \sqrt{2Ld\gamma_k}. \tag{2.23}$$

It is then a matter of properly choosing the sequences $\{\gamma_k\}_{k\geq 0}$ and $\{n_k\}_{k\geq 0}$ in order to obtain the best possible rate. Let us choose $\gamma_k = \frac{1}{Ld}e^{-2k}$ and $n_k = LdC_\eta^2 k^2 e^{3k}$. Plugging these values in (2.23), we have that

$$\|\bar{\mu}_{k+1} - \mu^*\|_{\mathrm{TV}} \lesssim e^{-k/2}\sqrt{\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}}} + e^{-k}. \tag{2.24}$$

It is easy to show that, from this recursive inequality, it follows $\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}} \lesssim e^{-k}\ \forall k \geq 1$ (Lemma 48). Hence, for $\epsilon > 0$, running Algorithm 3 for $K = \log(1/\epsilon)$ outer iteration ensures that $\|\bar{\mu}_K - \mu^*\|_{\mathrm{TV}} \leq \epsilon$. This corresponds to a total number of ULA iterations of

$$N = \sum_{k=1}^{K} n_k = \sum_{k=1}^{K} LdC_\eta^2 k^2 e^{3k} \propto LdC_\eta^2 K^2 e^{3K} = LdC_\eta^2 \log^2(1/\epsilon)\epsilon^{-3}. \tag{2.25}$$

Since $C = \mathcal{O}(d)$, we hence obtain an iteration complexity in TV distance of $\tilde{\mathcal{O}}(Ld^3\epsilon^{-3})$. The corresponding convergence rates in KL divergence and W_2 distance can be similarly derived, and are summarized in the following Theorem:

Theorem 11 (Iteration complexity of DL-ULA)**.** *Let μ^* be a L-smooth log-concave distribution satisfying Assumption 4 with parameters η, M_η. For every $k \geq 0$, let*

$$n_k = LdC_\eta^2 k^2 e^{3k}, \tag{2.26}$$

$$\gamma_k = \frac{1}{Ld} e^{-2k}, \tag{2.27}$$

$$\tau_k = C_\eta k. \tag{2.28}$$

Let $\bar{\mu}_k$ be the average distribution associated with the iterates of outer iteration k of DL-ULA (Algorithm 3) using the parameters above, just before the projection step. Then, $\forall \epsilon > 0$, the following hold:

- *After $N^{\mathrm{KL}} = \tilde{\mathcal{O}}(Ld^3\epsilon^{-\frac{3}{2}})$ total iterations, we obtain $\mathrm{KL}(\bar{\mu}_k; \mu^*) \leq \epsilon$.*
- *After $N^{\mathrm{TV}} = \tilde{\mathcal{O}}(Ld^3\epsilon^{-3})$ total iterations, we obtain $\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}} \leq \epsilon$.*
- *After $N^{\mathrm{W}_2} = \tilde{\mathcal{O}}(Ld^9\epsilon^{-6})$ total iterations, we obtain $\mathrm{W}_2(\bar{\mu}_k, \mu^*) \leq \epsilon \log(1/\epsilon)$.*

A few remarks about Theorem 11 are in order.

Geometric sequences. Theorem 11 prescribes a geometric sequence for the choice of $\{\gamma_k\}_{k\geq 0}$ and $\{n_k\}_{k\geq 0}$. As the outer iteration counter k increases, more and more ULA (inner) iterations are performed with the constant step-size γ_k. Asymptotically, we observe that the step size decreases at a rate $n^{-\frac{2}{3}}$ where n is the total number of ULA iterations. This decaying rate is faster than the more classical decaying rate of $n^{-\frac{1}{2}}$ for ULA (Durmus et al., 2018a).

In contrast to convex optimization where a global optimum can provably be reached with constant step-size, running ULA with constant step size γ yields a stationary distribution $\mu_\gamma \neq \mu^*$, which converges to μ^* only when $\gamma \to 0$. There is hence an intrinsic limitation to the sampling accuracy controlled by the magnitude of the step size. It is thus intuitively desirable to use a fast decaying step size.

Projection step. Even when assuming that the initial and target distributions are both log-concave, and thus have a sub-exponential tail, the sample distributions $\bar{\mu}_k$ are generally not log-concave, and it is not clear how to characterize their sub-exponential tail properties. This question is related to bounding the Poincaré constant of the ULA distribution iterates in the log-concave setting (Gromov and Milman, 1983; Gozlan, 2010), which is, to the best of our knowledge, still an open problem. Instead, we rely on the projection step at the end of each outer iteration as a way to enforce the tail property (2.15). Note that, since $\lim_{k\to\infty} \tau_k = \infty$, the projection step is applied less and less often.

Convergence rate comparison. Table 2.1 summarizes various convergence rates of Langevin dynamics based methods applied to general log-concave distributions. Compared to Durmus et al. (2017), the convergence rate in TV distance is worse in terms of accuracy ϵ but enjoys much better dimension dependence, and is also better in terms of Lipschitz constant dependence. Dalalyan et al. (2019) showed the same convergence guarantees in W_2 distance. The convergence bounds derived by Chewi et al. (2021) also outperforms ours in terms of accuracy ϵ. However, due to possible dimension dependence of the Poincaré constant, their guarantees can be worse in terms of dimension, and in terms of Lipschitz constant L. Finally, by assuming Lipschitz continuity of the potential f instead of its gradient, Lehec (2021) obtained the best known convergence guarantees in W_2 distance in term of accuracy.

2.2.5 DL-MYULA for constrained sampling

We now apply the same multistage idea to an existing constrained sampling algorithm, and show that it allows both to obtain an asymptotic convergence and improved convergence guarantees.

DL-MYULA algorithm

Consider sampling from a log-concave distribution over a convex set $\Omega \subset \mathbb{R}^d$, i.e.,

$$\mu^*(x) = \begin{cases} e^{-f(x)} / \int_\Omega e^{-f(x')} dx' & x \in \Omega \\ 0 & x \notin \Omega. \end{cases} \tag{2.29}$$

In Durmus et al. (2018b); Brosse et al. (2017), the authors propose to reduce this problem to an unconstrained sampling problem by penalizing the domain outside Ω directly inside the probability measure using its Moreau-Yoshida envelop. More precisely, they propose to sample from the following unconstrained probability measure $\mathrm{d}\mu_\lambda(\mathbf{x}) \propto e^{-f_\lambda(x)}\,\mathrm{d}\mathbf{x}$ where $f_\lambda : \mathbb{R}^d \to \mathbb{R}$ is defined as:

$$f_\lambda(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{x} - \mathrm{proj}_\Omega(\mathbf{x})\|_2^2, \qquad \forall \mathbf{x} \in \mathbb{R}^d, \tag{2.30}$$

where $\mathrm{proj}_\Omega : \mathbb{R}^d \to \Omega$ is the standard projection operator onto Ω defined as $\mathrm{proj}_\Omega(\mathbf{x}) = \arg\min_{\mathbf{y}\in\Omega} \|\mathbf{x}-\mathbf{y}\|_2$. Note that this penalty is easily differentiable as soon as the projection onto Ω can be computed since $\nabla f_\lambda(\mathbf{x}) = \nabla f(\mathbf{x}) + \frac{1}{\lambda}(\mathbf{x} - \mathrm{proj}_\Omega(\mathbf{x}))$.

By bounding the TV distance between μ_λ and μ^*, they showed that, by sampling from μ_λ with λ small enough, it is possible to sample from μ^* with arbitrary precision. This algorithm is called Moreau-Yoshida ULA (MYULA).

Building on this approach, we apply our double loop algorithm, by modifying both the step

Algorithm 4 DL-MYULA

Input: Smooth constrained probability measure μ^*, step sizes $\{\gamma_k\}_{k\geq 0}$, penalty parameters $\{\lambda_k\}_{k\geq 1}$, number of (inner) iterations $\{n_k\}_{k\geq 0}$, thresholds $\{\tau_k\}_{k\geq 1}$ and initial probability measure μ_0 on $\mathbb{R}^d$.
Initialization: Draw a sample x_0 from the probability measure μ_0.
for $k = 0, \dots$ **do**
    $\mathbf{x}_{k,0} \leftarrow \mathbf{x}_k$
    Draw N_k uniformly from $\{1, \cdots, n_k\}$.
    for $n = 0, \dots, N_k - 1$ **do**
        $\mathbf{x}_{k,n+1} \leftarrow \mathbf{x}_{k,n} - \gamma_k(\nabla f(\mathbf{x}_{k,n}) + \frac{1}{\lambda_{k+1}}(\mathbf{x}_{k,n} - \mathrm{proj}_\Omega(\mathbf{x}_{k,n}))) + \sqrt{2\gamma_k}\xi_{k,n}$, $\xi_{k,n} \sim \mathcal{N}(0, I_d)$.
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_{k,N_k}$.
    if $\|\mathbf{x}_{k+1}\|_2 > \tau_{k+1}$ **then**
        $\mathbf{x}_{k+1} \leftarrow \tau_{k+1}\mathbf{x}_{k+1}/\|\mathbf{x}_{k+1}\|_2$.

size as well as the penalty parameter λ between each inner loop (Algorithm 4).

In addition to providing improved rate, as we will show later, our algorithm also has the advantage to use a decreasing penalty parameter λ so as to guarantee asymptotic convergence of the algorithm to the target distribution. On the other hand, MYULA uses constant penalty λ, and thus saturates after a certain number of iterations. Although this looks like a trivial extension, using a varying penalty parameter makes the analysis more challenging since the target distribution of the algorithm is changing.

Convergence analysis of DL-MYULA

We now analyze the convergence of DL-MYULA. In Algorithm 4, both the step-size γ and the penalty parameter λ are decreased after each outer iteration. Therefore, at each outer iteration k, we aim to sample from the unconstrained penalized distribution $\mathrm{d}\mu_{\lambda_k}(\mathbf{x}) \propto e^{-f_{\lambda_k}(\mathbf{x})}\mathrm{d}\mathbf{x}$ where f_{λ_k} is defined in equation (2.30).

Similarly as for DL-ULA, we use Lemma 2.13 after each outer iteration, where the target distribution μ^* is replaced by μ_{λ_k}. Due to the strongly convex regularizer, the smoothness constant L_k of μ_{λ_k} increases as $L_k = L + \frac{1}{\lambda_k}$. We hence have the following:

$$\mathrm{KL}(\bar{\mu}_{k+1}; \mu_{\lambda_{k+1}}) \leq \frac{\mathrm{W}_2^2(\hat{\mu}_k, \mu_{\lambda_{k+1}})}{2\gamma_k n_k} + L_{k+1} d\gamma_k,$$

where again, $\bar{\mu}_k$ denotes the average iterate distribution of outer iteration k just before the projection step, and $\hat{\mu}_k$ is the one just after the projection step.

In order to use a similar recursion argument as previously, we need to bound $\mathrm{W}_2(\hat{\mu}_k, \mu_{\lambda_{k+1}})$ by

$\mathrm{W}_2(\hat{\mu}_k, \mu_{\lambda_k})$. Using the triangle inequality for W_2, we have

$$\mathrm{W}_2(\hat{\mu}_k, \mu_{\lambda_{k+1}}) \leq \mathrm{W}_2(\hat{\mu}_k, \mu_{\lambda_k}) + \mathrm{W}_2(\mu_{\lambda_k}, \mu^*) + \mathrm{W}_2(\mu_{\lambda_{k+1}}, \mu^*).$$

Brosse et al. (2017) showed a bound for $\|\mu_\lambda - \mu^*\|_{\mathrm{TV}}$ in terms of $\lambda > 0$, and it is easy to extend their proof to obtain a bound for $\mathrm{W}_2(\mu_\lambda, \mu^*)$ (Lemma A.1.4).

In order to prove our result, we make the same assumptions on the constraint set Ω as in (Brosse et al., 2017):

Assumption 12. *There exist $r, R, \Delta_1 > 0$ such that*

1. $B(0, r) \subset \Omega \subset B(0, D)$ *where* $B(0, r_0) = \{\boldsymbol{y} \in \mathbb{R}^d : \|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq r_0\}$ $\forall\, r_0 > 0$,
2. $e^{\inf_{\Omega^c}(f) - \max_\Omega(f)} \geq \Delta_1$, *where* $\Omega^c = \mathbb{R}^d \backslash \Omega$.

Lemma 13. *Let $\Omega \subset \mathbb{R}^d$ satisfy Assumption 12. Then, $\forall\, \lambda < \frac{r^2}{8d^2}$, it holds that*

$$\mathrm{W}_2^2(\mu_\lambda, \mu^*) \leq C_\Omega^2 d\sqrt{\lambda} \tag{2.31}$$

for some scalar $C_\Omega > 0$ depending on D, r and Δ_1.

Using these results, the convergence proof is then very similar as for DL-ULA, and is summarized in Theorem 14.

Theorem 14 (Iteration complexity of DL-MYULA)**.** *Let $\Omega \subset \mathbb{R}^d$ be a convex set satisfying Assumption 12 and μ^* be a log-concave distribution given by* (2.29) *where f has a L-Lipschitz continuous gradient. For every $k \geq 0$, let*

$$\lambda_k = \frac{1}{\frac{8d^2}{r^2} + de^{2k}}, \tag{2.32}$$

$$n_k = Ldk^2 e^{5k}, \tag{2.33}$$

$$\gamma_k = \frac{1}{Ld} e^{-4k}, \tag{2.34}$$

$$\tau_k = Dk, \tag{2.35}$$

Let $\bar{\mu}_k$ be the average distribution associated with the iterates of outer iteration k of DL-MYULA (Algorithm 4) using the parameters above, just before the projection step. Then, $\forall \epsilon > 0$, the following hold:

- *After $N^{\mathrm{TV}} = \mathcal{O}\left(d^{3.5}\epsilon^{-5}\right)$ total iterations, we obtain $\|\bar{\mu}_K - \mu^*\|_{\mathrm{TV}} \leq \epsilon$.*

Algorithm	TV	Literature
PLMC	$d^{12}\widetilde{O}(\epsilon^{-12})$	Bubeck et al. (2018)
MYULA	$d^{5}\widetilde{O}(\epsilon^{-6})$	Brosse et al. (2017)
DL-MYULA	$d^{3.5}\widetilde{O}(\epsilon^{-5})$	Rolland et al. (2020)

Table 2.2 – Upper bounds on the number of iterations required in order to guarantee an error smaller than ϵ in TV distance for various constrained sampling algorithms for log-concave distributions.

- *After* $N^{\mathrm{W}_2} = \tilde{\mathcal{O}}\left(d^{3.5}\epsilon^{-10}\right)$ *total iterations, we obtain* $\mathrm{W}_2(\bar{\mu}_K, \mu^*) \lesssim \epsilon$.

Smoothness of μ_{λ_k}. One can notice that the number of iterations in the inner loops of DL-MYULA increases faster than in DL-ULA. In order to explain this choice, first observe that the smoothness constant associated with the penalized distribution μ_λ grows as $\mathcal{O}\left(\frac{1}{\lambda}\right)$ as λ goes to 0. As k increases and λ_k decreases, μ_{λ_k} becomes less and less smooth. Since sampling from less smooth distributions requires to use a smaller step size and more iterations, this explains the fact that γ_k must decrease even faster than for DL-ULA.

We note that the choice for λ_k ensures that $\lambda_k < \frac{r^2}{8d^2}$ as required for Lemma A.1.4 to be applicable.

Convergence rate comparison. Table 2.2 summarizes convergence rates in TV distance for various first-order constrained sampling algorithms. We can see that DL-MYULA outperforms existing approaches, both in terms of rate and dimension dependence. Note that, similarly as in (Bubeck et al., 2018) and (Brosse et al., 2017), we omitted the dependence on the volume and the diameter of the constraint set, which are thus assumed to be dimension independent, in order to make a fair comparison.

2.3 Robust Reinforcement Learning via adversarial training with Langevin dynamics

Now that we have gained more insight about the sampling algorithms involved in the Infinite-dimensional Entropic Mirror Descent (Algorithm 2), we demonstrate one particular application of this method, related to the training of robust models via Reinforcement Learning.

2.3.1 Introduction

Reinforcement learning aims to train an agent evolving in a given environment so as to maximize a certain reward function. To this end, we can simulate any agent evolving in this environment, and observe the obtained reward, enabling us to improve the agent's policy. By properly parameterizing the agent's strategy, or policy, it becomes possible, by evaluating the

agent's performance in the environment, to compute the gradient of the expected reward with respect to the policy parameters. This is known as the Policy Gradient method. Hence, RL can be seen as a continuous optimization problem with access to first-order information.

Despite the success of deep RL in many automation tasks with beyond-human performance (Mnih et al., 2015; Silver et al., 2017; Lillicrap et al., 2015; Levine et al., 2016), trained RL agent are usually brittle when it comes to testing on environments differing from the training environment, seriously questioning their applicability in real-life applications involving safety and security issues.

A powerful framework to learning robust policies is to interpret the changing of the environment as an adversarial perturbation. This notion naturally lends itself to a two-player max-min problem involving a pair of agents, a protagonist and an adversary, where the protagonist learns to fulfill the original task goals while being robust to the disruptions generated by its adversary. Two prominent examples along this research vein, differing in how they model the adversary, are the Robust Adversarial Reinforcement Learning (RARL) (Pinto et al., 2017) and Noisy Robust Markov Decision Process (NR-MDP) (Tessler et al., 2019).

Despite the impressive empirical progress, the training of the robust RL objectives remains an open and critical challenge. In particular, Tessler et al. (2019) prove that it is in fact strictly suboptimal to directly apply (deterministic) policy gradient steps to their NR-MDP max-min objectives. Owing to the lack of a better algorithm, the policy gradient is nonetheless still employed in their experiments; similar comments also apply to (Pinto et al., 2017).

In this work, we aim to solve the resulting min-max problem using Entropic Mirror Descent (Algorithm 2), exploiting existing policy gradient methods for RL, namely DDPG.

2.3.2 Preliminaries: Markov decision problems and deterministic policy gradient

Classical RL deals with the training of a single agent evolving in an environment. This environment is characterised by a Markov Decision Process (MDP) $\mathscr{M} = \{\mathscr{S}, \mathscr{A}, T, \gamma, R, P_0\}$, where $\mathscr{S}, \mathscr{A}$ are the sets of states and actions respectively, $T : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to [0,1]$ is the state transition probability, i.e., $T(s'|s,a)$ denotes the probability of landing in state s' by taking action a in state s. $R : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ denotes the reward function, γ the discount factor and P_0 is the initial state distribution. At each time step t, the agent observes the current state $s_t \in \mathscr{S}$ and takes an action $a_t = \mu(s_t) \in \mathscr{A}$ where μ is the agent's policy. The agent gets a reward $r_t = R(s_t, a_t)$ and transitions to a new state s_{t+1} sampled according to the transition probability T. The goal is to maximize the cumulative sum of discounted rewards over the agent's policy

$$\max_{\mu} J(\mu) \equiv \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| \mathscr{M}\right] \tag{2.36}$$

Problem (2.36) looks difficult, since the way the objective function depends on the policy μ is quite intricate, and it is not clear intuitively how to make progress in order to improve the policy. Luckily, if we parametrize the policy μ_θ in a differentiable way, then the gradient of the function $J(\theta) \equiv J(\mu_\theta)$ is given by

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t \sim \rho^{\mu_\theta}} \left[\nabla_a Q^{\mu_\theta}(s,a)|_{s=s_t, a=\mu(s_t)} \nabla_\theta \mu_\theta(s_t) \right], \tag{2.37}$$

where $Q^\mu(s_t, a_t) \equiv \mathbb{E}_{r_{i-1}, s_i, a_i=\mu(s_i), i>t}[\sum_{i=t}^{\infty} \gamma^{i-t} r_i]$ is the so-called Q-function, and ρ^μ denotes the (discounted) state-visitation distribution under policy μ. Hence, by simulating an agent with policy μ_θ, we can both learn the function Q^{μ_θ}, and obtain samples s_t from ρ^{μ_θ}. We can thus compute MCMC estimates of $\nabla_\theta J(\theta)$ thanks to (2.37). Problem (2.36) can thus be approximately solved using some form of stochastic gradient descent. This approach is called *Deterministic Policy Gradient*.

Since the stochastic gradient estimates are very noisy, and can be biased depending on the Q function parametrization, various techniques have been developed to stabilise the training. However, we will not enter into too much details here. The algorithm we will use is called *Deep Deterministic Policy Gradient* (Lillicrap et al., 2015). The Q functions and policy μ^θ in DDPG are both parametrized using neural networks. Moreover, two pairs of Q function and policy are trained simultaneously, one representing the actual trained models, and the other one being a time-delayed version of the first pair, used when computing the stochastic gradients (2.37), and is introduced in order to stabilize the training.

2.3.3 Robust training with two players Markov games

By maximizing the function J, we want to make sure that the policy performs well within the environment described by the MDP $\mathscr{M}$. However, robust policy training requires that the agent also performs well in the case where the environment undergoes some small changes. To this end, we introduce a second adversarial agent, whose goal is to perturbe the original agent during training.

More concretely, consider a two-player zero-sum Markov game (Littman, 1994; Perolat et al., 2015), where at each step of the game, both players simultaneously choose an action. The reward each player gets after one step depends on the state and the joint action of both players. Furthermore, the transition dynamics of the game is controlled jointly by both players.

This game can be described by a 2-players MDP $\mathscr{M}_2 = \left(\mathscr{S}, \mathscr{A}, \mathscr{A}', T_2, \gamma, R_2, P_0\right)$, where $\mathscr{A}$ and $\mathscr{A}'$ are the continuous set of actions the players can take, $T_2 : \mathscr{S} \times \mathscr{A} \times \mathscr{A}' \times \mathscr{S} \to \mathbb{R}$ is the state transition probability, and $R_2 : \mathscr{S} \times \mathscr{A} \times \mathscr{A}' \to \mathbb{R}$ is the reward for both players. Consider an agent executing a policy $\mu : \mathscr{S} \to \mathscr{A}$, and an adversary executing a policy $\nu : \mathscr{S} \to \mathscr{A}'$ in the environment $\mathscr{M}$. At each time step t, both players observe the state s_t and take actions $a_t = \mu(s_t)$ and $a'_t = \nu(s_t)$. In the zero-sum game, the agent gets a reward $r_t = R_2\left(s_t, a_t, a'_t\right)$ while the adversary gets a negative reward $-r_t$.

This two-player zero-sum Markov game formulation has been used to model the following robust RL settings:

- Robust Adversarial Reinforcement Learning (RARL) (Pinto et al., 2017), where the power of the adversary is limited by its action space $\mathscr{A}'$.
- Noisy Robust Markov Decision Process (NR-MDP) (Tessler et al., 2019), where $\mathscr{A}' = \mathscr{A}$, $T_2\left(s_{t+1} \mid s_t, a_t, a'_t\right) = T_1\left(s_{t+1} \mid s_t, \bar{a}_t\right)$, and $R_2\left(s_t, a_t, a'_t\right) = R_1\left(s_t, \bar{a}_t\right)$, with $\bar{a}_t = (1-\delta)a_t + \delta a'_t$, for a chosen $\delta \in (0,1)$, which limits the adversary.

In our adversarial game, we consider the following performance objective:

$$J(\mu,\nu) \;=\; \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \;\middle|\; \mu, \nu, \mathscr{M}_2\right],$$

where $\sum_{t=1}^{\infty} \gamma^{t-1} r_t$ is the random cumulative return. In particular, we consider the parameterized policies $\left\{\mu_\theta : \theta \in \Theta\right\}$, and $\{\nu_\omega : \omega \in \Omega\}$. By an abuse of notation, we denote $J(\theta,\omega) = J\left(\mu_\theta, \nu_\omega\right)$. We consider the following objective:

$$\max_{\theta\in\Theta} \min_{\omega\in\Omega} J(\theta,\omega). \tag{2.38}$$

Note that J is neither convex in θ nor concave in ω. Instead of solving (2.38) directly, we focus on the mixed strategy formulation of (2.38). In other words, we consider the set of all probability distributions over Θ and Ω, and we search for the optimal distribution that solves the following program:

$$\max_{p\in\mathscr{P}(\Theta)} \min_{q\in\mathscr{P}(\Omega)} f\left(p,q\right) := \mathbb{E}_{\theta\sim p}\left[\mathbb{E}_{\omega\sim q}\left[J(\theta,\omega)\right]\right]. \tag{2.39}$$

Similarly as in the single player case, it possible to compute estimates of $\nabla_\theta J, \nabla_\omega J$ by simulating the agents in the environment and evaluating the obtained rewards, as shown in (Tessler et al., 2019, Proposition 5). Hence, we can use the techniques from Section 2.1.2 to solve the above problem. By adapting DDPG (Lillicrap et al., 2015) to this mixed strategy 2-players game, we obtain Algorithm 14. Note that, while two policy networks are trained, one for each player, we only train a single network for learning the Q function of the joint policy.

2.3.4 Experiments

In this section, we demonstrate the effectiveness of using the MixedNE-LD framework to solve the robust RL problem. We consider NR-MDP setting with $\delta = 0.1$ (as recommended in Section 6.3 of (Tessler et al., 2019)). We compare the solution given by solving the mixed formulation (2.39) using MixedNE-LD (Algorithm 14) and the one given by solving the pure strategy formulation (2.38) using classical min-max algorithm, namely Gradient-Ascent-Descent (GAD)

as proposed in (Tessler et al., 2019) and Extra-Gradient (EG) Gidel et al. (2018). These two latter methods are summarized in Algorithm 15.

The comparison is performed on classical RL tasks available on OpenAI Gym Brockman et al. (2016) utilizing the MuJoCo environment Todorov et al. (2012). We consider 8 different tasks, namely Walker, Hopper, Half-Cheetah, Ant, Swimmer, Reacher, Humanoid, and InvertedPendulum (see Brockman et al. (2016) for more details about these environments).

All parameterized functions, i.e., the agent's and adversary's policies as well as the Q function are represented using two-hidden layers feed-forward neural networks with 64 neurons per layer and tanh activation function. The algorithms' hyperparameters are described in Table A.1. For exploration-related hyperparameters, the table specifies a set of values, meaning that the corresponding hyperparameter has been optimized using grid search over this set for each algorithm-environment pair, and are given in Table A.2. Each algorithm is trained on 0.5M samples, i.e., 0.5M time steps in the environment. For each environment, we run each algorithm with 5 different seeds. The exploration noise is turned off for evaluation.

In order to test the robustness of the learnt policies, we evaluate the obtained cumulative reward in modified environments, by changing the mass of the agent and the friction parameter (which are common parameters in all environments) but without the adversarial perturbation that was present during training (Figures 2.1 and 2.2). We observe that the policies learnt using MixedNE-LD (Algorithm 14) outperform the one trained using the baselines (Algorithm 15). We emphasize in particular the superior performance for InvertedPendulum, which was a failure case in (Tessler et al., 2019).

For further experimental evaluations and more detailed ablation study of the algorithms, please refer to the original paper Kamalaruban et al. (2020).

2.4 Bibliographic notes

The RL experiments in the last section have been performed by Yu-Ting Huang.

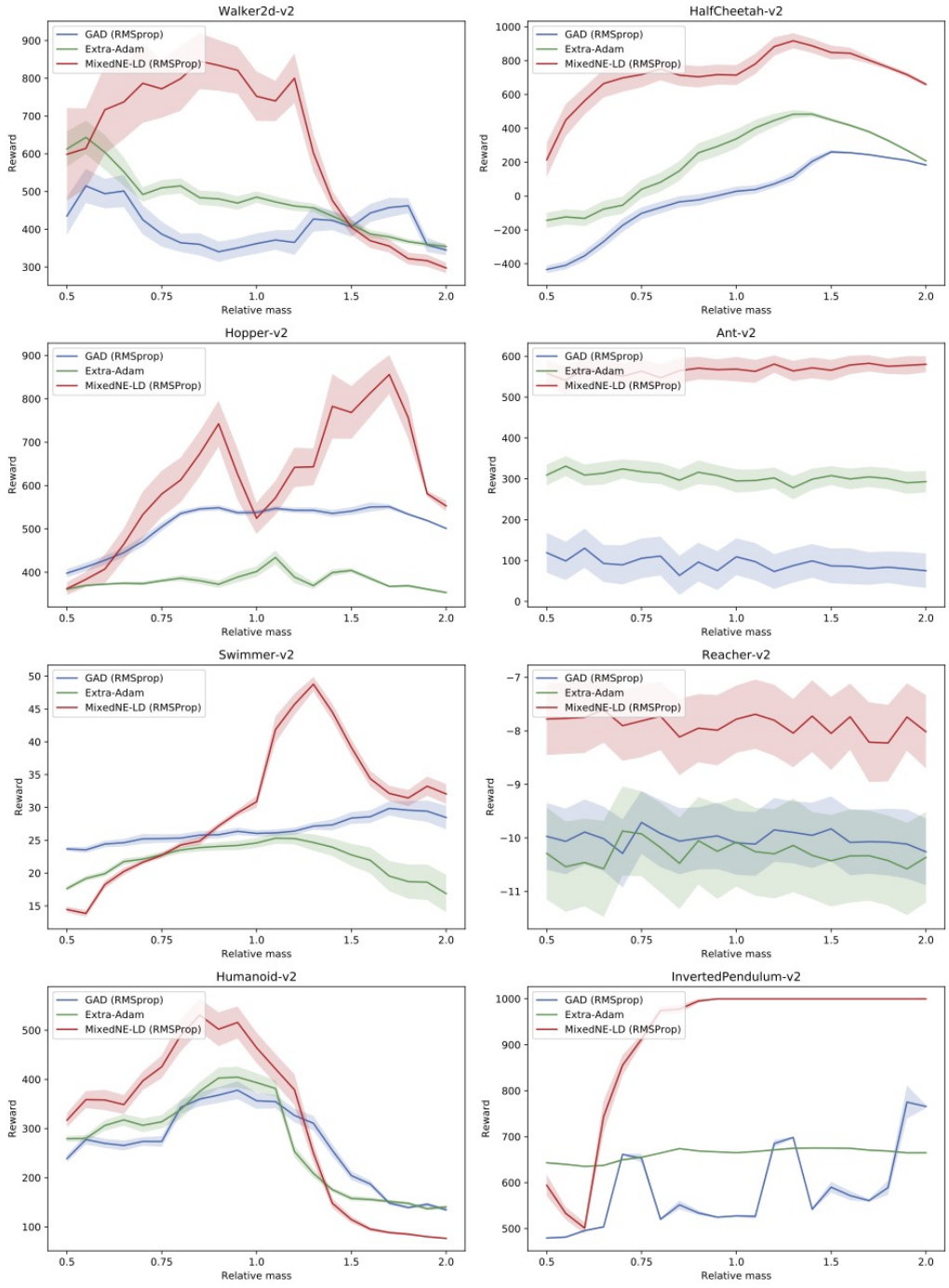


Figure 2.1 – Average performance (over 5 seeds) of Algorithm 14, and Algorithm 15 (with GAD and Extra-Adam), under the NR-MDP setting with $\delta = 0.1$. The evaluation is performed without adversarial perturbations, on a range of mass values not encountered during training.

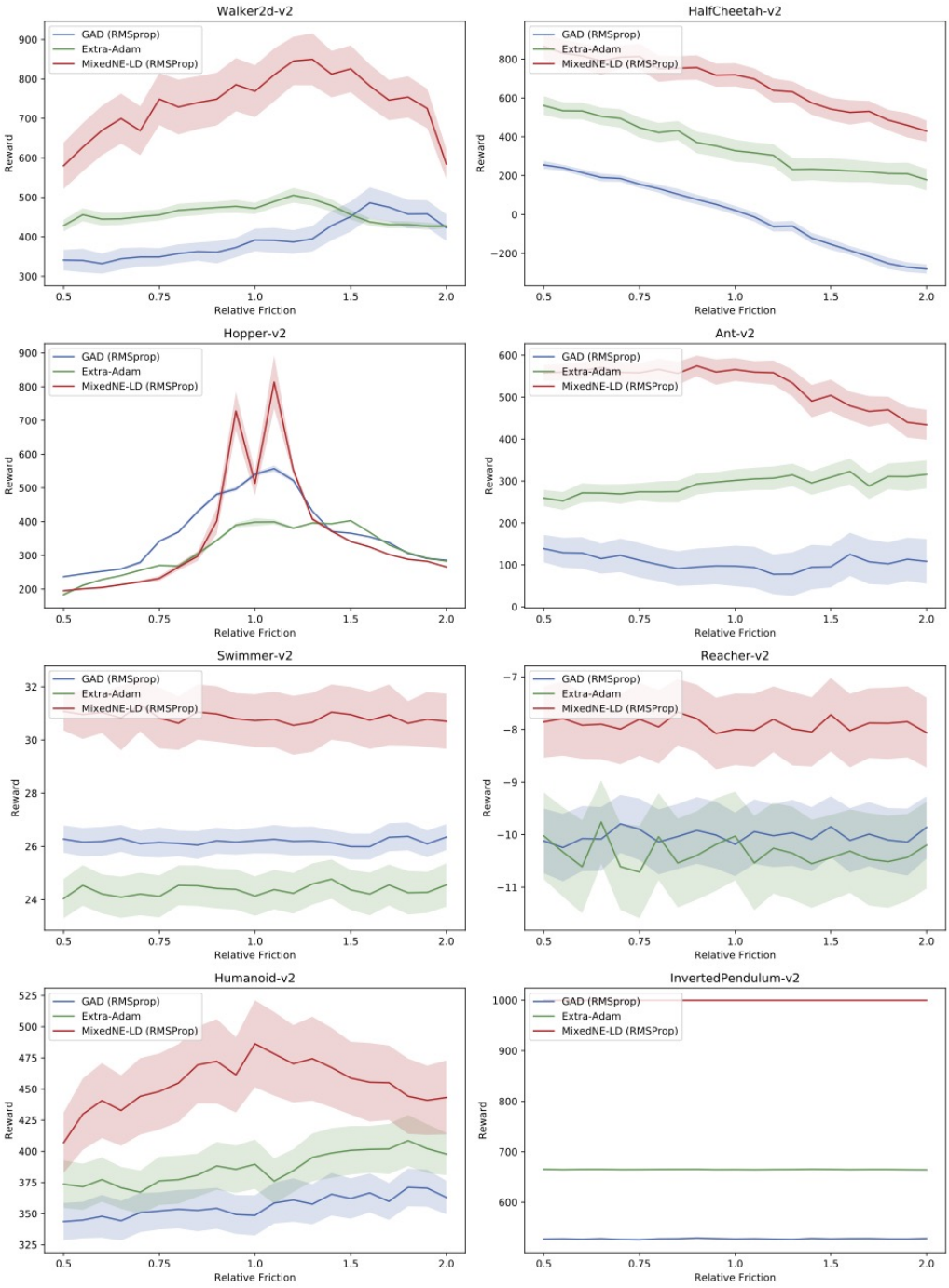


Figure 2.2 – Average performance (over 5 seeds) of Algorithm 14, and Algorithm 15 (with GAD and Extra-Adam), under the NR-MDP setting with $\delta = 0.1$. The evaluation is performed without adversarial perturbations, on a range of friction values not encountered during training.

3 Robustness to adversarial perturbation using regularization

In the previous chapter, we saw how to train a robust model by involving an adversary perturbing the model during the training phase. However, nonconvex-nonconcave minimax optimization is a challenging problem and convergence properties of practical algorithms for solving such a task are not fully understood. Moreover, it is not exactly clear what model is trained using this method, i.e., what are the specific properties of the model that we favour when including a perturbing adversary.

In this chapter, we propose to train a single model while penalizing a certain quantity during training that is meant to improve its robustness. Most of the models that are used in practice are over-parameterized, meaning that we optimize much more parameters than necessary in order to solve the task. This implies that many different models can perfectly fit the training data, i.e., achieve a training loss of 0. However, not all of these models may have the same robustness properties.

In order to further specify what model we want to train among all the ones achieving low training loss, one method is to add another term in the objective, in addition to the training loss. This method is called **regularization**. We will argue that using the Lipschitz constant of the model as regularizer seems reasonable, since it is closely related to the adversarial robustness of the model.

In the first part of this chapter, we propose an algorithm for estimating the Lipschitz constant of neural networks, allowing one to certify the robustness of a given network. Then, as a proxy to the Lipschitz constant, we propose to penalize the 1-path norm of the network, and develop a proximal gradient algorithm for solving the associated regularized problem.

3.1 Lipschitz constant estimation of neural networks via sparse polynomial optimization

This section is based on the paper (Latorre et al., 2020a) published at ICLR 2020.

3.1.1 Introduction

Let $f : \mathbb{R}^n \to \mathbb{R}$ be an arbitrary function. We say that f is Lipschitz continuous if there exists a constant $L_f > 0$ such that for all $x, y \in \mathbb{R}^n$, we have

$$|f(x) - f(y)| \leq L_f \|x - y\|, \tag{3.1}$$

where $\|\cdot\|$ is a norm in $\mathbb{R}^n$ to be specified. The minimum over all such values satisfying this inequality condition is called the Lipschitz constant of f, and is denoted by $L(f)$.

The Lipschitz constant of a function is linked to various of its properties. In particular, we see from equation (3.1) that $L(f)$ quantifies the deviation in the output given a certain perturbation in the input. Indeed, suppose that we evaluate f at two close points x, y such that $\|x - y\| \leq \epsilon$. Then, the Lipschitz continuity implies that $|f(x) - f(y)| \leq L\epsilon$. Hence, the smaller the Lipschitz constant, the less the function changes given any input perturbation. This is of particular interest in the scope of adversarial robustness, where we wish to train a model that is not too much perturbed given a small perturbation of the input. An upper bound on the Lipschitz constant hence provides a robustness certificate of a model against adversarial perturbation at any given point.

In this work, we focus on estimating the Lipschitz constant of a scalar-valued neural network f_d with depth d defined by recursion as

$$f_1(x) := W_1 x \qquad f_i(x) := W_i \sigma(f_{i-1}(x)), \quad i = 2, \ldots, d \tag{3.2}$$

where the weight matrices $W_i \in \mathbb{R}^{n_i \times n_{i+1}}$ (i.e., n_0, n_{d+1} denote the input and output dimensions respectively), and σ denotes the activation function applied element-wise. In this work, for simplicity, we only treat the case of single output networks, i.e., with $n_{d+1} = 1$. However, bounds for the multi-output case can be obtained in a similar manner.

While trivial upper bounds can easily be obtained, i.e., be computing the product of the layer-wise Lipschitz constants, these are often overly pessimistic. There is hence a growing need for methods that provide tighter upper bounds on $L(f_d)$, even at the cost of increased computational expenses. For example Raghunathan et al. (2018a); Jin and Lavaei (2018); Fazlyab et al. (2019) derive upper bounds based on *semidefinite programming (SDP)*. While expensive to compute, these type of certificates are in practice surprisingly tight.

In this section, we present a general approach, called **LiPopt**, for upper bounding the Lipschitz constant of a given neural network via polynomial optimization. We start by showing that the problem of computing the Lipschitz constant can be phrased as an optimization problem, which can itself be relaxed as a *polynomial optimization problem* (POP) (Lasserre, 2015). Hence, solving the relaxed POP provides an upper bound on the Lipschitz constant. Moreoever, while the initial problem related to exactly estimating the Lipschitz constant is NP-hard (Virmaux and Scaman, 2018), the relaxed problem (which remains NP-hard to solve

exactly) can be efficiently approximated by exploiting existing methods from polynomial optimization.

We extend this approach to the case where the neural network is sparse, hence improving the computational complexity of our method. While our approach covers both ℓ_2- and $\ell_\infty-$ Lipschitz constant estimations, we solely focus on the $\ell_\infty-$ Lipschitz constant, for which we experiment on networks with random weights as well as networks trained on MNIST (Lecun et al., 1998). We show that the proposed method can provide tighter upper bounds on the Lipschitz constant than existing approaches.

Related work. Virmaux and Scaman (2018); Combettes and Pesquet (2019); Fazlyab et al. (2019); Jin and Lavaei (2018) worked on estimations of $L(f_d)$ with respect to the ℓ_2-norm. The method **SeqLip** (Virmaux and Scaman, 2018) attemps to estimate the Lipschitz constant instead of providing a proper upper bound, and hence cannot be used for robustness certification. On the other hand, **LipSDP** (Fazlyab et al., 2019) provides true upper bounds on $L(f_d)$. However, its formulation is restricted to the ℓ_2-norm Lipschitz constant. While it is possible to extend it to ℓ_∞-norm Lipschitz constant by multiplying the obtained bound by the square root of the input dimension, this often results in rather loose bounds, which is confirmed empirically in our experiments (see Section 3.1.6).

Notation. We denote by n_i the number of columns of the matrix W_i in the definition (3.2) of the network. This corresponds to the size of the i-th layer, where we identify the input as the first layer. We let $n = n_1 + \ldots + n_d$ be the total number of neurons in the network. For a vector $\mathbf{x}$, Diag($\mathbf{x}$) denotes the square matrix with $\mathbf{x}$ in its diagonal and zeros everywhere else. For an array X, vec(X) is the *flattened* array. The support of a sequence supp(α) is defined as the set of indices j such that α_j is nonzero. For $\mathbf{x} = [x_1, \ldots, x_n]$ and a sequence of nonnegative integers $\gamma = [\gamma_1, \ldots, \gamma_n]$ we denote by $\mathbf{x}^\gamma$ the monomial $x_1^{\gamma_1} x_2^{\gamma_2} \ldots x_n^{\gamma_n}$.

3.1.2 Polynomial optimization formulation

In this section, we start by showing that the computation of the Lipschitz constant can be cast as an optimization problem. When applied to the a neural network given by (3.2), we show that this optimization problem takes a specific form that can be relaxed as a POP. We start with the following equivalent characterization of the Lipschitz constant:

Theorem 15. *Let f be a differentiable and Lipschitz continuous function on an open, convex subset $\mathcal{X}$ of an Euclidean space. Let $\|\cdot\|_*$ be the dual norm defined as $\|\boldsymbol{x}\|_* \equiv \sup_{\|t\|\leq 1} \boldsymbol{t}^T \boldsymbol{x}$. The Lipschitz constant of f is given by*

$$L(f) = \sup_{x \in \mathcal{X}} \left\| \nabla f(x) \right\|_* . \tag{3.3}$$

This theorems only applies to functions f that are differentiable. In order to apply it to neural networks, we hence require the activation function to be Lipschitz continuous and

differentiable, ruling out the common ReLU activation function $\sigma(x) = \max(0, x)$. However, this assumption remains true for other standard choices of activation functions such as the Exponential Linear Unit (ELU) (Clevert et al., 2015) or softplus.

We hence observe that the Lipschitz constant is closely tied to the gradient of the function. In order to specify this result to the case of neural networks, let us compute explicitly the gradient of f_d in (3.2) using the chain rule:

$$\nabla f_d(x) = W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(\sigma'(f_i(x))) W_{i+1}^T \tag{3.4}$$

Using the definition of a dual norm, i.e., $\|x\|_* = \sup_{\|t\|\leq 1} t^T x$, the Lipschitz constant of f_d can be written as

$$L(f_d) = \sup_{x\in\mathbb{R}^{n_1}, \|t\|\leq 1} t^T W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(\sigma'(f_i(x))) W_{i+1}^T. \tag{3.5}$$

We now observe that, for common differentiable activation functions such as ELU or softplus, their derivative is bounded between 0 and 1. Therefore, by introducing new variables $\mathbf{s}_i = \sigma'(f_i(x)) \in \mathbb{R}^{n_{i+1}},\ i = 1,\ldots,d-1$, we can relax the formulation (3.5) to obtain the following upper bound:

$$L(f_d) \leq \max\left\{ t^T W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(\mathbf{s}_i) W_{i+1}^T : 0 \leq \mathbf{s}_i \leq 1, \|t\| \leq 1 \right\}. \tag{3.6}$$

We can see that the objective in (3.6) is a polynomial in $t, \{\mathbf{s}_i\}_{i=1,\ldots,d-1}$.We will refer to this polynomial objective as the **norm-gradient polynomial**, i.e.,

$$p(\mathbf{s}_1,\ldots,\mathbf{s}_{d-1}, t) = t^T W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(\mathbf{s}_i) W_{i+1}^T. \tag{3.7}$$

Moreover, for any positive integer q (or $q = \infty$), the constraint $\|t\|_q \leq 1$ can be expressed as a polynomial constraint. In the rest of this work, we will focus on the ℓ_∞-norm constraint, i.e., $\|t\|_\infty \leq 1$ which is equivalent to $-1 \leq t_i \leq 1$ for $i = 1,\ldots,n_1$. Hence, problem (3.6) boils down to maximizing a polynomial objective under polynomial constraints, and is hence a POP.

To motivate the use of the ℓ_∞-norm, we note that this is the most commonly used norms to assess robustness in the adversarial examples literature. Moreover, it has been shown that, in practice, ℓ_∞-norm robust networks are also robust in other more plausible measures of perceptibility, like the Wasserstein distance (Wong et al., 2019).

3.1.3 Solving the POP using polynomial positivity certificate

We now focus on solving the relaxed POP (3.6) using classical techniques in polynomial optimization. First note that, by defining $\mathbf{s}_0 \equiv (t+1)/2$, and denoting by p the norm-gradient polynomial, the POP (3.6) can be written as

$$\max_{\mathbf{x}\in[0,1]^n} p(\mathbf{x})$$

where $\mathbf{x} = \text{vec}([\mathbf{s}_0, \ldots, \mathbf{s}_{d-1}])$ is the concatenation of all variables.

The central idea in polynomial optimization is that maximizing a polynomial p over a domain $\mathscr{X}$ is equivalent to finding the smallest value λ such that $\lambda - p$ is positive over $\mathscr{X}$. This positivity constraint can be ensured by writing the polynomial $\lambda - p$ in such a way that makes it clear that this polynomial is non-negative over $[0,1]^n$, e.g., by writing it as a sum of squared polynomials. In this work, we use the so-called *Krivine's positivity certificate* (also known as *Krivine's Positivstellensatz*) which, adapted to our setting, reads as follows:

Theorem 16. *(Adapted from Krivine (1964); Stengle (1974); Handelman (1988)) If the polynomial $\lambda - p$ is strictly positive on $[0,1]^n$, then there exist finitely many positive weights $c_{\alpha\beta} \in \mathbb{R}_+$ such that*

$$\lambda - p = \sum_{(\alpha,\beta)\in\mathbb{N}^{2n}} c_{\alpha\beta} h_{\alpha\beta}, \qquad h_{\alpha\beta}(x) := \prod_{j=1}^{n} x_j^{\alpha_j}(1-x_j)^{\beta_j} \tag{3.8}$$

Example 17. *Consider the polynomial $q(x_1,x_2,x_3) = 1.1 - x_3 - x_1x_2 + x_1x_3$. A priori, it is not trivial to guess whether q is strictly positive over $[0,1]^3$. However, writing q as $q(\boldsymbol{x}) = 0.1 + x_1(1-x_2) + (1-x_1)(1-x_3)$ provides us with a certificate that q is indeed strictly positive on $[0,1]^3$.*

Note that the degree of the certificate sometimes needs to be higher than that of the resulting polynomial. Take for example the degree 2 polynomial $q(\mathbf{x}) = 1.1 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3$. Then, its smallest degree Krivine certificate is given by $q(\mathbf{x}) = 0.1 + x_1x_2x_3 + (1-x_1)(1-x_2)(1-x_3)$ which is of degree 3.

Hence, one issue with this certificate is that the maximum degree of the polynomials $h_{\alpha\beta}$ potentially needs to be larger (and sometimes much larger) than that of p, thus the number of terms required in the decomposition (3.8) can be very large.

In order for this certificate to be usable, we need to truncate the degree of the polynomial involved in the decomposition (3.8), leading to the following *hierarchy* of LP problems (Lasserre, 2015, Section 9):

$$\theta_k := \min_{c\geq 0,\lambda} \left\{ \lambda : \lambda - p = \sum_{(\alpha,\beta)\in\mathbb{N}_k^{2n}} c_{\alpha\beta} h_{\alpha\beta} \right\} \tag{3.9}$$

where $\mathbb{N}_k^{2n}$ is the set of nonnegative integer sequences of length $2n$ adding up to at most k.

For each k, problem (3.9) is clearly an LP, since the objective function is simply λ, and the constraint $\lambda - p = \sum_{(\alpha,\beta)\in\mathbb{N}^{2n}_k} c_{\alpha\beta} h_{\alpha\beta}$ can be equivalently written by equating the coefficients of each polynomial in the canonical basis, giving rise to a set of linear constraints in λ and c. For this constraint to be feasible, we need the degree k of the certificate to be greater or equal to the degree of the norm-gradient polynomial p, i.e., $k \geq d$.

The sequence $\{\theta_k\}_{k=1}^{\infty}$ is non-incresing and converges to the maximum of the upper bound (3.6) thanks to Theorem 16. Note that for any level of the hierarchy, the solution of the LP (3.9) provides a valid upper bound on $L(f_d)$.

On remaining issue is that the size of the LP for θ_k grows rapidly with k. Indeed, the dimension of the variable c in (3.9) is

$$|\mathbb{N}^{2n}_k| = \mathcal{O}(n^k), \tag{3.10}$$

corresponding to the size of the monomial canonical basis of degree k in dimension n. For instance, if we consider the MNIST dataset and a one-hidden-layer network with 100 neurons, we have $\left|\mathbb{N}^{2n}_2\right| \approx 1.5 \times 10^6$ while $\left|\mathbb{N}^{2n}_3\right| \approx 9.3 \times 10^8$. To make this approach more scalable, we exploit in the next section the sparsity of the polynomial p to find LPs of drastically smaller size than (3.9), but with similar approximation properties.

3.1.4 Reducing the number of variables

In many practical cases, trained neural networks involve sparse weight matrices. These sparsity patterns can either come from the use of convolutional layers, or from network pruning, since it has been empirically observed that up to 90% of network weights can be set to zero without harming the accuracy (Frankle and Carbin, 2019). In this case, the norm-gradient polynomial inherits this sparsity pattern, which can be used to reduce the search space when decomposing it using Krivine's positivity certificate (3.8).

The question is hence the following: Can we exploit the structure of the polynomial p to decompose in order to restrict the set of functions $\{h_{\alpha\beta}\}_{(\alpha,\beta)\in\mathbb{N}^{2n}}$ required to construct the Krivine certificate (3.8)?

Consider for example a positive polynomial $q : [0,1]^n \to \mathbb{R}$ that can be written as $q(\mathbf{x}) = \sum_{i=1}^{m} q_i(\mathbf{x}_{I_i})$, where $I_i \subseteq \{1,\dots,n\}$, $I_i \cap I_j = \emptyset$ for $i \neq j$, and q_i is a polynomial only depending on variables $\{x_j : j \in I_i\}$. When searching for a positivity certificate for q, it makes sense to decompose each q_i separately, and hence not to include polynomials $h_{\alpha\beta}$ involving interacting variables from different sets I_i and I_j. In this case, the number of possible decompositions drastically reduces, since we would only consider polynomials $h_{\alpha\beta}$ such that $\mathrm{supp}(\alpha) \cap \mathrm{supp}(\beta) \subseteq I_i$ for some $i \in \{1,\dots,m\}$.

However, when the sets $\{I_i\}_{i=1}^{m}$ overlap, this strategy does not always work, and we may not find a certificate with the same structure as q. Nonetheless, it can be shown that this method applies if the sets $\{I_i\}_{i=1}^{m}$ satisfy a *valid sparsity pattern*, which we define as follows:

Definition 18. *Let $I = \{1,\dots,n\}$ and p be a polynomial with variable $x \in \mathbb{R}^n$. A valid sparsity pattern of p is a sequence $\{I_i\}_{i=1}^m$ of subsets of I, called cliques, such that $\bigcup_{i=1}^m I_i = I$ and*

- *$p = \sum_{i=1}^m p_i$ where p_i is a polynomial that depends only on the variables $\{x_j : j \in I_i\}$,*
- *for all $i = 1,\dots,m-1$ there is an $l \le i$ such that $(I_{i+1} \cap \bigcup_{r=1}^i I_r) \subseteq I_l$.*

In the case where the objective polynomial of a POP is known to have a valid sparsity pattern, we can provide further characterizations of the decomposition involved in the Krivine's certificate. The following result is a consequence of (Weisser et al., 2018), and is referred to as the *sparse Krivine's certificate*:

Theorem 19 (Adapted from Weisser et al. (2018))**.** *Let a polynomial p have a valid sparsity pattern $\{I_i\}_{i=1}^m$. Define N_i as the set of sequences $(\alpha,\beta) \in \mathbb{N}^{2n}$ where the support of both α and β is contained in I_i. If $\lambda - p$ is strictly positive over $K = [0,1]^n$, there exist finitely many positive weights $c_{\alpha\beta}$ such that*

$$\lambda - p = \sum_{i=1}^m h_i, \qquad h_i = \sum_{(\alpha,\beta)\in N_i} c_{\alpha\beta} h_{\alpha\beta} \tag{3.11}$$

where the polynomials $h_{\alpha\beta}$ are defined as in (3.8)*.*

Depending on the sparsity pattern, sparse Krivine's certificate can drastically reduce the search space when looking for a positivity certificate.

It turns out that the norm-gradient polynomial has a natural sparsity pattern given by its construction. Indeed, variables s_{ij} in the gradient-norm polynomial p in (3.7) correspond to the neurons in the neural networks. By definition of p, we know that the gradient-norm polynomial does not involve monomials containing several variables associated with the same layer. Actually, it only contains monomials of the form $t s_{1,i_1} s_{2,i_2} \dots s_{d-1,i_{d-1}}$ for $i_j \in 1,\dots,n_{j+1}$. Moreover, the sparsity pattern of p also depends on the inner sparsity of its weight matrices. In order to characterize the induced sparsity pattern of norm-gradient polynomials, we first introduce a graph that depends on the network f_d.

Definition 20. *Let f_d be a neural network with weights $\{W_i\}_{i=1}^d$. Define a directed graph $G_d = (V,E)$ as follows:*

$$\begin{aligned} V &= \left\{s_{i,j} : 0 \le i \le d-1,\, 1 \le j \le n_i\right\}, \\ E &= \left\{(s_{i,j}, s_{i+1,k}) : 0 \le i \le d-2, [W_i]_{k,j} \ne 0\right\}, \end{aligned} \tag{3.12}$$

which we call the computational graph of the network f_d.

The graph G_d is intuitively composed of all neurons of the neural network denoted by $\{s_{i,j}\}$, and constructed by connecting each neuron to the one of the next layer for which the associated weight is non-zero.

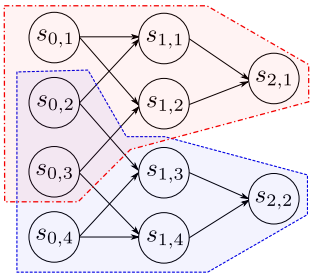


Figure 3.1 – Induced sparsity pattern for a network of depth three.

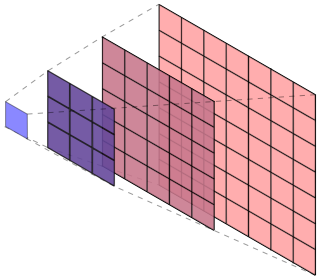


Figure 3.2 – Structure of the induced sparsity pattern for a network with 2D convolutional layers with 3×3 filters.

Definition 21. *Let f_d be a neural network with associated graph G_d defined as in Definition 20. We define the sparsity pattern $\{I_i\}_{i=1}^{n_d}$* ***induced by*** *G_d as*

$$I_i := \{s_{(d-1,i)}\} \cup \{s_{(j,k)} : \text{ there exists a directed path from } s_{(j,k)} \text{ to } s_{(d-1,i)} \text{ in } G_d\}. \tag{3.13}$$

An example of induced sparsity pattern is depicted in Figure 3.1. However, while, for fully connected graphs, the induced sparsity pattern is indeed a valid sparsity pattern, it may not be the case for sparse networks, since the second condition (3.13) in Definition 21 might not hold. In that case, we loose the guarantee that the values of the corresponding LPs converge to the maximum of the POP (3.19). Nevertheless, it still provides a valid positivity certificate that we can use to upper bound $L(f_d)$. In Section 2.3.4 we show that in practice it provides upper bounds of good enough quality, while significantly accelerating the computations.

We now quantify how much using a certain sparsity pattern $\{I_i\}_{i=1}^m$ when searching for a Krivine decomposition reduces the size of the associated LP. Let $s = \max_{i=1,\dots,m} |I_i|$, and let $N_{i,k}$ be the subset of N_i (defined in Theorem 19) composed of sequences summing up to k. The number of different polynomials $h_{\alpha\beta}$ involved in the k-th LP of the hierarchy given by the sparse Krivine's certificate can be bounded as follows:

$$\left| \bigcup_{i=1}^{m} N_{i,k} \right| \leq \sum_{i=1}^{m} \binom{2|I_i|+k}{k} = \mathcal{O}\left(ms^k\right). \tag{3.14}$$

We immediately see that the dependence on the number of subsets m is really mild (linear) but the size of the associated subsets I_i as well as the degree of the hierarchy can greatly impact the size of the optimization problem. Note that this upper bound can be quite loose since polynomials $h_{\alpha\beta}$ that depend only on variables in the intersection of two or more subsets are counted more than once.

In the case of sparsity pattern induced by a neural network, as defined in Definition 21, the number of subsets m corresponds to the size of the last hidden layer n_d, and the subsets I_i depend on the inner sparsity of the network. In the following, we describe various sparsity

patterns induced by networks with different architectures.

Fully connected networks. In this case, all subsets I_i have the same size $|I_i| = n_1 + \ldots + n_{d-1} + 1$. Hence, using the bound of (3.14), the total number of variables in the LP for computing θ_k (3.9) is $\mathcal{O}(n_d(n_1 + \ldots + n_{d-1} + 1)^{k-1})$, improving upon the full size (3.10) when $n_d > 1$. Moreover, the resulting pattern is a valid sparsity pattern (Proposition 22).

Proposition 22. *Let f_d be a dense network (all weights are nonzero). Then, the sparsity pattern induced by the network's graph is a valid sparsity pattern for the norm-gradient polynomial of f_d.*

Unstructured sparsity. In the case of networks obtained by pruning (Hanson and Pratt, 1989) or generated randomly from a distribution over graphs (Xie et al., 2019), the sparsity pattern can be arbitrary. In this case the size of the resulting LPs varies at runtime. Under the layer-wise assumption that any neuron is connected to at most r neurons in the previous layer, the size of the subsets of the induced sparsity pattern is bounded as $s = \mathcal{O}(r^d)$. This estimate has an exponential dependency on the depth but ignores that many neurons might share connections to the same inputs in the previous layer, thus being potentially loose. The bound (3.14) implies that the number of different polynomials is $\mathcal{O}(n_d r^{dk})$.

2D Convolutional networks. The sparsity in the weight matrices of convolutional layers has a certain *local structure*; neurons are connected to contiguous inputs in the previous layer. Adjacent neurons also have many input pixels in common (see Figure 3.2). Assuming a constant number of channels per layer, the size of the cliques in (3.13) is $\mathcal{O}(d^3)$. Intuitively, such number is proportional to the volume of the pyramid depicted in Figure 3.2 where each dimension depends linearly on d. Using (3.14) we get that there are $\mathcal{O}(n_d d^{3k})$ different polynomials in the sparse Krivine's certificate. This is a drastic decrease in complexity when compared to the unstructured sparsity case.

The general procedure for upper bounding the Lipschitz constant, given a sparsity pattern and a hierarchy degree k, is described in Algorithm 5.

Algorithm 5 LipOpt for ELU activations

Input: matrices $\{W_i\}_{i=1}^d$, sparsity pattern $\{I_i\}_{i=1}^m$, hierarchy degree k.

$p \leftarrow (2\mathbf{s}_0 - 1)^T W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(\mathbf{s}_i) W_{i+1}^T$ ▷ compute norm-gradient polynomial

for $i = 1, \ldots, m$ **do**

  $N_{i,k} \leftarrow \{(\alpha, \beta) \in \mathbb{N}_k^{2n} : \mathrm{supp}(\alpha) \cap \mathrm{supp}(\beta) \subseteq I_i\}$

$N_k \leftarrow \cup_{i=1}^m N_{i,k}$

return $\min\{\lambda : \lambda - p = \sum_{(\alpha,\beta)\in N_k} c_{\alpha\beta} h_{\alpha\beta}, c_{\alpha\beta} \geq 0\}$ ▷ solve LP

3.1.5 Relation to Shor's relaxation and Sum-Of-Squares hierarchy

Shor's relaxation. Any POP can be written as a *quadratically constrained quadratic program* (QCQP) by introducing extra variables representing higher order interactions. This procedure

is described in (Park and Boyd, 2017, Section 2.1). While QCQPs are NP-hard to solve in general (since, e.g., MAX-CUT can be phrased as a QCQP), there exists a relaxation of such problems to an SDP known as the *Shor's relaxation* (Park and Boyd, 2017, Section 3.3). The idea behind Shor's relaxation is to introduce a matrix $X = \mathbf{x}\mathbf{x}^T$ representing the quadratic interactions in the QCQP, and then relax the non-convex constraint $X = \mathbf{x}\mathbf{x}^T$ by the semi-definite one $X \succeq 0$, hence removing the constraint that $\text{rank}(X) = 1$.

One drawback of this approach is that it includes a further relaxation step from (3.6), thus being fundamentally limited in how tightly it can upper bound the value of $L(f_d)$. Moreover when compared to LP solvers, off-the-shelf semidefinite programming solvers are, in general, much more limited in the number of variables they can efficiently handle.

Sum-Of-Squares hierarchy. The main idea we used for maximizing a polynomial p is to minimize a lower bound λ for p, i.e., such that $\lambda - p(x) \geq 0 \ \forall x \in [0,1]^n$. We then used Krivine decomposition to certify this constraint. However, other positivity certificates can be used. A well-known certificate is the sum-of-squares (SOS) decompositions, which involves squared polynomials which are obviously non-negative. For certifying non-negativity over $[0,1]$ using SOS certificate, we would look for a decomposition of the form

$$\lambda - p(x) = s_0(x) + \sum_{i=1}^{n} (s_i(x)x_i + s_i'(x)(1 - x_i)),$$

where s_i, s_i' are squared polynomials. Similarly as Krivine decomposition, this gives rise to a hierarchy of decompositions, by constraining the degrees of the polynomials s_i, s_i'.

Solving the resulting optimization problem using the SOS certificate requires dealing with the sum of squares constraint for the polynomials s_i, s_i', which can be cast as a semi-definite constraint. Hence, using this certificate requires solving an SDP. Actually, the first degree of the SOS hierarchy, i.e., restricting the polynomials s_i, s_i' to be of degree 2, is equivalent to the Shor's relaxation (Lasserre, 2000).

3.1.6 Experiments

We consider the following bounds on $L(f_d)$ with respect to the ℓ_∞-norm:

Name	Description
SDP	Upper bound arising from the solution of the Shor's relaxation described in Section 3.1.5.
LipOpt-k	Upper bound arising from the k-th degree of the LP hierarchy (3.9) based on the sparse Krivine Positivstellenstatz.
LipSDP	Upper bound from Fazlyab et al. (2019) multiplied $\sqrt{n_1}$, where n_1 is the input dimension of the network.
UBP	Upper bound determined by the product of the layer-wise Lipschitz constants with $\ell_\infty-$ metric.
LBS	Lower bound obtained by sampling 50000 random points around zero, and evaluating the dual norm of the gradient.

Experiments on random networks

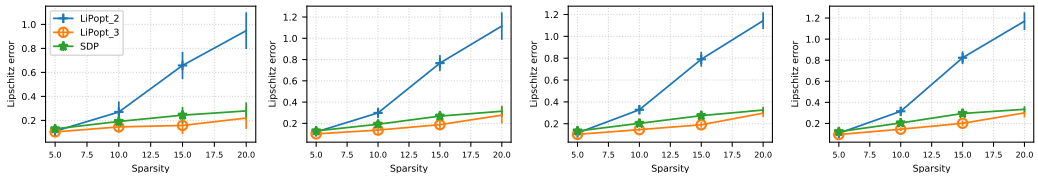
We compare the bounds obtained by the algorithms described above on networks with random weights and either one or two hidden layers. We define the sparsity level of a network as the maximum number of neurons any neuron in one layer is connected to in the next layer. For example, the network represented on Figure 3.1 has sparsity 2. The non-zero weights of network's i-th layer are sampled uniformly in $[-\frac{1}{\sqrt{n_i}}, \frac{1}{\sqrt{n_i}}]$ where n_i is the number of neurons in layer i.

For different configurations of width and sparsity, we generate 10 random networks and average the obtained Lipschitz bounds. For better comparison, we plot the relative error. Since we do not know the true Lipschitz constant, we cannot compute the true relative error. Instead, we take as reference the lower bound given by **LBS**. Figures 3.3 and 3.5 show the relative error, i.e., $(\hat{L} - L_{LBS})/L_{LBS}$ where L_{LBS} is the lower bound computed by **LBS** and $\hat{L}$ is the estimated upper bound.

When the chosen degree for **LiPopt-k** is the smallest as possible, i.e., equal to the depth of the network, we observe that the method is already competitive with the **SDP** method, especially in the case of 2 hidden layers. When we increment the degree by 1, **LiPopt-k** becomes uniformly better than **SDP** over all tested configurations. We remark that the upper bounds given by **UBP** are too large to be shown in the plots. Similarly, for 1-hidden layer networks, the bounds from **LipSDP** are too large to be plotted.

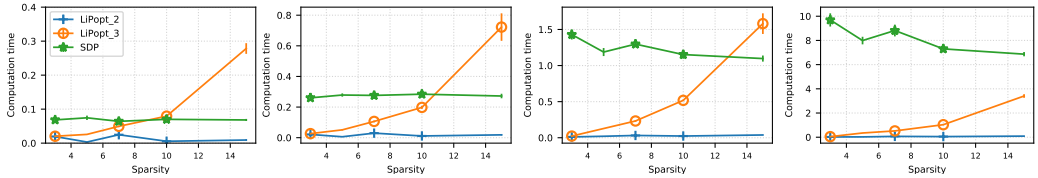
Finally, we measure the computation time of the different methods on each tested network (Figures 3.4 and 3.6). We observe that the computation time for **LiPopt-k** heavily depends on the network sparsity, which reflects the fact that such a structure is exploited in the algorithm. In contrast, the time required for **SDP** does not depend on the sparsity, but only on the size of the network. Therefore, as the network size grows (with fixed sparsity level), **LipOpt-k** obtains a better upper bound and runs faster. Also, with our method, we see that it is possible to increase the computation power in order to compute tighter bounds when required, making it more flexible than **SDP** in terms of computation/accuracy tradeoff. **LipOpt-k** uses the Gurobi

LP solver, while **SDP** uses Mosek. All methods run on a single machine with Core i7 2.8Ghz quad-core processor and 16Gb of RAM.



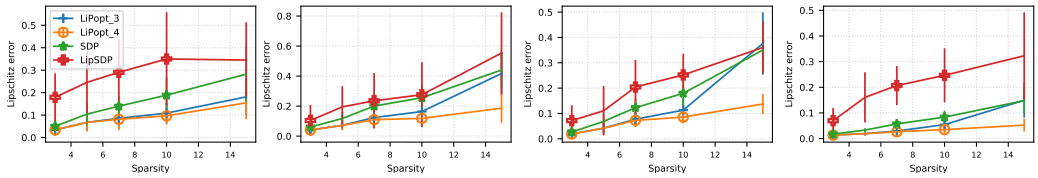
(a) 40 × 40 (b) 80 × 80 (c) 160 × 160 (d) 320 × 320

Figure 3.3 – Lipschitz approximated relative error for 1-hidden layer networks



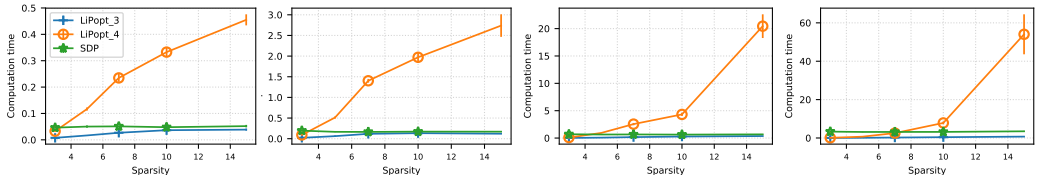
(a) 40 × 40 (b) 80 × 80 (c) 160 × 160 (d) 320 × 320

Figure 3.4 – Computation times for 1-hidden layer networks (seconds)



(a) 5 × 5 × 10 (b) 10 × 10 × 10 (c) 20 × 20 × 10 (d) 40 × 40 × 10

Figure 3.5 – Lipschitz approximated relative error for 2-hidden layer networks



(a) 5 × 5 × 10 (b) 10 × 10 × 10 (c) 20 × 20 × 10 (d) 40 × 40 × 10

Figure 3.6 – Computation times for 2-hidden layer networks (seconds)

Experiments on trained networks

Similarly, we compare these methods on networks trained on MNIST. The architecture we use is a fully connected network with two hidden layers with 300 and 100 neurons respectively, and with one-hot output of size 10. Since the output is multi-dimensional, we restrict the network to a single output, and estimate the Lipschitz constant with respect to label 8. We

chose this label because it is the one which is most sensitive to adversarial attacks, since it can be easily changed to a 3, a 6, a 9 or a 0 only using small perturbations.

We train the network using the pruning strategy described in Han et al. (2015)[1]. After training the full network using standard techniques, the weights of smallest magnitude are set to zero. Then, the network is trained for additional iterations, only updating the nonzero parameters. Doing so, we were able to remove 95% of the weights, while preserving the same test accuracy. We recorded the Lipschitz bounds for various methods in Table 3.1.6. We observe a clear improvement of the Lipschitz bound obtained from **LiPopt-k** compared to **SDP** method, even when using the smallest allowed value $k = 3$ (since the network is of depth 3). Also note that the input dimension is too large for the method **Lip-SDP** to provide competitive bound, since it requires multiplying the obtained bound by the square root of the image size, i.e., $\sqrt{784}$.

Algorithm	**LBS**	**LiPopt-4**	**LiPopt-3**	**SDP**	**UBP**
Lipschitz bound	84.2	88.3	94.6	98.8	691.5

3.2 1-path-norm Regularization using Proximal Gradient Method

We now turn our attention to the training of robust networks. While the previous section describes a method for estimating an upper bound on the Lipschitz constant, extending it to an algorithm for training neural networks would yield an algorithm with rather large per iteration complexity. Instead, we consider in this section a surrogate quantity related to the Lipschitz constant, and propose an efficient algorithm for approximately solving the associated regularized optimization problem.

This section is based on the paper (Latorre et al., 2020c) published at ICML 2020.

3.2.1 Introduction

The use of the Lipschitz constant as a regularizer has shown desirable properties in terms of robustness and generalization both in theory and practice (Raghunathan et al., 2018b; Cisse et al., 2017; Jiang* et al., 2020). However, its direct use as a regularizer leads to quite a complex optimization problem to solve. Therefore, there is a need for designing upper bounds on the Lipschitz constant whose associated regularized problem can be efficiently solved. An ideal bound would achieve a balance between two properties: It should provide a good estimate of the constant while being fast and easy to minimize with iterative first-order algorithms.

Recently, the *path-norm* of the network (Neyshabur et al., 2015) has emerged as a complexity measure that is highly-correlated with generalization (Jiang* et al., 2020). Thus, its use as a regularizer holds an increasing interest for researchers in the field.

[1]For training we used the code from this reference. It is publicly available in https://github.com/mightydeveloper/Deep-Compression-PyTorch

However, our understanding of the optimization aspects of the path-norm-regularized objective is lacking. Jiang* et al. (2020) refrained from using automatic-differentiation methods in this case because, as they argue, the optimization could fail, thus providing no conclusion about its qualities.

It is then natural to ask: *How do we properly optimize the path-norm-regularized objective with theoretical guarantees? What conclusions can we draw about the robustness and sparsity of path-norm-regularized networks?* We focus on the *1-path-norm* and provide partial answers to those questions, further advancing our understanding of this measure. We treat the problem separately between the case of 1-hidden layer neural networks, for which we develop a method for computing the proximal mapping exactly and efficiently (Section 3.2.3), and the general case of deep networks for which we need to resolve to approximations of the proximal mapping (Section 3.2.4). Let us summarize our main contributions:

- In the case of 1-hidden layer networks, we show the following properties:
  - Despite its non-convexity, the 1-path norm of 1-hidden layer neural networks admits an efficient *proximal mapping* (Algorithm 16), allowing the use of proximal-gradient type methods which are, as of now, the only first-order optimization algorithms to provide guarantees of convergence for composite non-smooth and non-convex problems (Bolte et al., 2013).
  - The 1-path norm provides an upper bound on the (ℓ_∞, ℓ_1)-Lipschitz constant, which is always tighter than the naive product of spectral norms bound.
  - Neural network regularization schemes promoting sparsity in a principled way are of great interest in the growing field of *compression* in Deep Learning (Han et al., 2016; Cheng et al., 2017b). Our analysis provides a formula (cf. Lemma 29) for choosing the *strength* of the regularization, which enforces a desired bound on the sparsity level of the iterates generated by the proximal gradient method. This is a suprising, yet intuitive, result, as the sparsity-inducing properties of non-smooth regularizers have been observed before in convex optimization and signal processing literature, see e.g., (Bach et al., 2012; Eldar and Kutyniok, 2012).
- For deeper networks,
  - We derive a tractable procedure (8) to compute the non-convex non-smooth proximal mapping operator with respect to the 1-path-norm, restricted to a single path of a network with arbitrary depth, i.e., a unit-width network.
  - For the general case of a deep network of arbitrary width, we propose two practical heuristical methods (Algorithms 9 and 11) that try to bypass the implementation limitations of an exact proximal gradient scheme.
- We illustrate the benefits of our proposed algorithms in a variety of experimental setups. First, in synthetic scenarios where it is possible to derive the full proximal map, we show

that the proposed approach outperforms Adam (Kingma and Ba, 2015) and SGD in iteration complexity. Finally in experiments on FashionMNIST Xiao et al. (2017a) and CIFAR10 Krizhevsky (2009) where we employ our proposed heuristics, we observe they outperform SGD and compare favorably to Adam when the network size is sufficiently large.

3.2.2 Problem setup and preliminaries

For a d-layers feedforward neural network (3.2) with weight matrices $\mathbf{W} \equiv [W_1, \ldots, W_d]$ which we now denote as $f_{\mathbf{W}}$ to emphasize the dependence on the weights, its 1-path-norm can be defined in two equivalent ways:

$$\begin{aligned} P_1(\mathbf{W}) &\equiv \mathbf{1}^T |W_L||W_{L-1}|\cdots|W_1|\,\mathbf{1} \\ &= \sum_{s \in S} \prod_{i=1}^{d} |W_i(s_{i+1}, s_i)| \end{aligned} \tag{3.15}$$

where $S \equiv [n_1] \times \cdots \times [n_{d+1}]$ with $[n_i] \equiv \{0, 1, \ldots, n_i - 1\}$, $|W_i|$ is the matrix obtained by entry-wise application of the absolute value function, the symbol $\mathbf{1}$ denotes an all-ones column vector with dimension inferred by the context, and $W_i(j,k)$ denotes the j,k-th entry of a matrix W_i.

To understand the definition (3.15), notice that an element of the set S, say $s = (s_1, \ldots, s_{d+1})$, can be understood as a choice of one neuron per layer: the s_1-th neuron in the 1-st layer (input layer), the s_2-th neuron in the 2-nd layer and so on. This sequence of neurons forms a path in the network, from input to output layer. For the i-th edge in this path, we associate the value $W_i(s_{i+1}, s_i)$, which is precisely the parameter in W_i that connects the chosen neurons in the computational graph.

With this terminology, we can describe the 1-path-norm of a network (3.15) as the sum of the absolute value of the product of the weights along each path from input to output layer. From now on we will refer to any such choice of one neuron per layer as one *path*. The following is the 1-path-norm regularized empirical risk minimization problem on N labeled training samples $(x_i, y_i) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_{d+1}}$, loss function $\mathscr{L}$ and regularization parameter $\lambda \geq 0$:

$$\min_{\mathbf{W}} \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_{\mathbf{W}}(x_i), y_i\right) + \lambda P_1(\mathbf{W}). \tag{3.16}$$

When $L \geq 2$, common choices of loss function $\mathscr{L}$, such as the cross-entropy loss, lead to a composite optimization objective in (3.16) that is non-convex and non-smooth with a non-convex non-smooth regularizer due to the presence of absolute values and products. Evidently, such a non-convex model cannot be solved globally unless more restrictive assumptions are imposed.

Thus, instead of global optimality, we rather consider developing algorithms with non-asymptotic

rates of convergence to first-order stationarity via the proximal gradient approach. For the type of problem in consideration, this still constitutes a highly challenging task since computing the proximal mapping requires solving a non-convex non-smooth problem as-well.

The Prox-GD Method

The **proximal gradient** method aims to solve optimization problems of the form

$$\min_{x\in\mathbb{R}^n} F(x) \equiv f(x) + g(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function with L-Lipschitz continuous gradient, and $g : \mathbb{R}^n \to \mathbb{R}$ is non-smooth, but admits a computable proximal operator as defined in equation (3.17).

$$\text{prox}_g(x) = \arg\min_{y\in\mathbb{R}^n} g(y) + \frac{1}{2}\|x-y\|_2^2. \tag{3.17}$$

The Prox-GD method is described by Algorithm 6. Since g is nonconvex, the operator in (3.17) can be a set of solutions. Any element of this set can then be chosen in the algorithm.

Algorithm 6 Prox-GD

Input: $z^0 \in \mathbb{R}^n$, $\{\eta^k\}_{k\geq 0}$.

1: **for** $k \geq 1$ **do**
2: Compute $G^{k-1} = \nabla f(z^{k-1})$
3: $z^k \leftarrow \text{prox}_{\eta^{k-1}g}(z^{k-1} - \eta^{k-1}G^{k-1})$
4: **return** $\{z^k\}_{k\geq 0}$

Theoretical guarantees for the Prox-GD method with respect to a nonconvex regularizer were established by Bolte et al. (2013) (for a more general proximal-gradient type scheme).

Theorem 23 (Convergence guarantees)**.** *Let $\{z^k\}_{k\geq 0}$ be a sequence generated by Algorithm 6 with $\{\eta^k\}_{k\geq 0} \subseteq (0, 1/L)$. Then, the following hold:*

1. *Any accumulation point of $\{z^k\}_{k\geq 0}$ is a critical point of $f + g$.*
2. *If f satisfies the Kurdyka-Lojasiewicz (KL) property (Attouch et al., 2010), then $\{z^k\}_{k\geq 0}$ converges to a critical point.*
3. *Suppose that η_k is chosen such that there exists $c > 0$ such that $\sum_{k=0}^{K} \frac{1}{\eta_k} \geq cK$ for any integer $K > 0$. Then*

$$\min_{k=0,\ldots,K} \|z^{k+1} - z^k\|_2 \leq \sqrt{\frac{2(F(z^0) - F_*)}{(c-L)K}},$$

where $F_* \equiv \min_{x\in\mathbb{R}^n} F(x)$.

Point 2 of Theorem 23 states that Algorithm 6 is globally convergent under the Kurdyka–Lojasiewicz (KL) property Attouch et al. (2010). The broad classes of semi-algebraic and subanalytic functions, widely used in optimization, satisfy the KL property (see e.g. (Bolte et al., 2013, Section 5)), and in particular, most convex functions encountered in finite dimensional applications satisfy it (see (Bolte et al., 2013, Section 5.1)). We refer the reader to the works Attouch et al. (2010, 2011); Bolte et al. (2013), in particular to (Bolte et al., 2013, Sections 3.2-3.5) for additional information and results.

3.2.3 Path norm regularization of shallow neural networks

We first consider the special case of 1-hidden layer neural networks, e.g., $L = 2$, which we re-write

$$f_{V,W}(x) = V^T \sigma(Wx)$$

where $V \in \mathbb{R}^{n\times p}, W \in \mathbb{R}^{n\times m}$; m, p denote the input and output dimensions respectively, and n the number of neurons in the hidden layer of the network.

Relation to the Lipschitz constant

Since one of the goal of 1-path norm regularization is to improve the robustness of the model, we show how it relates to the Lipschitz constant. It provides an upper bound on the Lipschitz constant when using the ℓ_∞ norm for the input space and the ℓ_1 norm for the input space, i.e., we define $L_{V,W}$ as the smallest scalar $L \in \mathbb{R}$ such that

$$\|f_{V,W}(y) - f_{V,W}(x)\|_1 \leq L\|y - x\|_\infty \ \forall x, y \in \mathbb{R}^m. \tag{3.18}$$

A naive upper bound on $L_{V,W}$ is the product $\|V^T\|_{\infty,1}\|W\|_\infty$, where $\|W\|_\infty$ is the operator norm of a matrix W with respect to the ℓ_∞ norm for both input and output space, which is equal to the maximum ℓ_1-norm of its rows, and $\|V\|_{\infty,1}$ is the operator norm of the matrix V with respect to the ℓ_∞ norm in input space and ℓ_1-norm in output space, which is equal to the sum of the ℓ_1 norm of its columns. However, this bound can be quite loose, and we show that the 1-path norm always provide a tighter upper bound on $L_{V,W}$.

Theorem 24. *Let $f_{V,W}(x) = V^T\sigma(Wx)$ be a network such that the derivative of the activation σ is globally bounded between zero and one, and let $P_1(V,W) = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^p |W_{ij}V_{ik}|$ be its 1-path norm. The Lipschitz constant $L_{V,W}$ of the network is bounded as follows:*

$$L_{V,W} \leq P_1(V,W) \leq \|V^T\|_{\infty,1}\|W\|_\infty. \tag{3.19}$$

Notice that although the path-norm and layer wise product bounds can be equal, this only happens in the special case where, for the weight matrix in the first layer, the 1-norm of every rows are equal. Thus, in practice the bounds can differ drastically.

Computing the proximal mapping

We now turn our attention to the task of solving the path nom regularized risk minimization problem (3.16) in the 1-hidden layer case, i.e.,

$$\min_{V\in\mathbb{R}^{n\times p},W\in\mathbb{R}^{n\times m}} \frac{1}{n}\sum_{i=1}^{n}\mathscr{L}\left(f_{V,W}(x_i),y_i\right)+\lambda P_1(V,W), \tag{3.20}$$

using the proximal gradient method (Algorithm 6). To this end, it is necessary to design an efficient way to compute the proximal mapping of P_1, i.e., to solve for any $X\in\mathbb{R}^{n\times p}$, $Y\in\mathbb{R}^{n\times m}$

$$\operatorname{prox}_{\lambda P_1}(X,Y)\equiv\operatorname*{arg\,min}_{V,W}\lambda P_1(V,W)+\frac{1}{2}(\|V-X\|_{\mathscr{F}}^2+\|W-Y\|_{\mathscr{F}}^2) \tag{3.21}$$

where $\|\cdot\|_{\mathscr{F}}$ denotes the Frobenius norm. We first notice that the objective in (3.21) is coercive and lower bounded, implying that there exists an optimal solution (Beck, 2014, Thm. 2.32).

Lemma 25 (Well-posedness of (3.21))**.** *For any $\lambda\geq 0$ and any (X,Y), the problem* (3.21) *has a global optimal solution.*

In this section, we assume for simplicity that the neural network $f_{V,W}$ has a single output, i.e., $p=1$. The multi-output setting uses similar arguments, but its analysis requires a more delicate treatment which is deferred to Appendix B.2.

The problem (3.21) can hence be written for $\mathbf{x}\in\mathbb{R}^n$, $Y\in\mathbb{R}^{n\times m}$ as

$$\operatorname{prox}_{\lambda P_1}(\mathbf{x},Y)\equiv\operatorname*{arg\,min}_{\mathbf{v},W}\lambda\sum_{i=1}^{n}\sum_{j=1}^{m}|W_{ij}v_i|+\frac{1}{2}(\|\mathbf{v}-\mathbf{x}\|_2+\|W-Y\|_{\mathscr{F}}^2) \tag{3.22}$$

$$=\operatorname*{arg\,min}_{\mathbf{v},W}\sum_{i=1}^{n}\left(\lambda\sum_{j=1}^{m}|W_{ij}v_i|+\frac{1}{2}(v_i-x_i)_2+\sum_{j=1}^{m}(W_{ij}-Y_{ij})^2)\right). \tag{3.23}$$

We can see that the problem (3.23) is separable, in the sense that it involves minimizing a sum of terms that depend on separate sets of variables. Therefore, it suffices to solve the optimization problem for each term separately and then combine the results. We hence further simplify the problem and aim to solve for any $x\in\mathbb{R},\mathbf{y}\in\mathbb{R}^m$

$$\operatorname*{arg\,min}_{v,\mathbf{w}\in\mathbb{R}\times\mathbb{R}^m}\lambda|v|\sum_{j=1}^{m}|w_j|+\frac{1}{2}(v-x)^2+\frac{1}{2}\sum_{j=1}^{m}(w_j-y_j)^2. \tag{3.24}$$

We next observe that the signs of the variables $v,\mathbf{w}$ solving (3.24) are determined by the signs of $x,\mathbf{y}$. Since the sign of $v,\mathbf{w}$ does not impact the value of the term $|v|\sum_{j=1}^m|w_j|$, it should be chosen so as to minimize the rest of the terms $\frac{1}{2}(v-x)^2+\frac{1}{2}\sum_{j=1}^m(w_j-y_j)^2$. The best choice is hence to choose the signs of $v,\mathbf{w}$ to be the same as the one of $x,\mathbf{y}$. Therefore, by an appropriate

change of variable, it suffices to solve

$$\underset{v,\mathbf{w}\in\mathbb{R}_+\times\mathbb{R}_+^m}{\operatorname{argmin}} \lambda v \sum_{j=1}^{m} w_j + \frac{1}{2}(v-|x|)^2 + \frac{1}{2}\sum_{j=1}^{m}(w_j-|y_j|)^2. \tag{3.25}$$

Lemma 26. *Let* $(v^*, \boldsymbol{w}^*) \in \mathbb{R}+\times\mathbb{R}+^n$ *be an optimal solution of* (3.25). *Then* $(sign(x)\cdot v^*, sign(\mathbf{y})\circ \boldsymbol{w}^*)$ *is an optimal solution of problem* (3.24).

Let us denote

$$h_\lambda(v,\mathbf{w};x,\mathbf{y}) = \lambda v \sum_{j=1}^{m} w_j + \frac{1}{2}(v-|x|)^2 + \frac{1}{2}\sum_{j=1}^{m}(w_j-|y_j|)^2.$$

Although h_λ is nonconvex, we will show that a global optimum to (3.25) can be obtained efficiently by utilizing several tools, the first being the first-order optimality conditions of (3.25) (Beck, 2014, Ch. 9) given below.

Lemma 27 (Stationarity conditions)**.** *Let* $(v^*, \boldsymbol{w}^*) \in \mathbb{R}+\times\mathbb{R}+^m$ *be an optimal solution of* (3.25) *for a given* $(x,\mathbf{y}) \in \mathbb{R}\times\mathbb{R}^m$. *Then*

$$w_j^* = \max\{0, |y_j| - \lambda v^*\} \quad \text{for any } j = 1,2,\ldots,m, \tag{3.26}$$

$$v^* = \max\left\{0, |x| - \lambda \sum_{j=1}^{m} w_j^*\right\}. \tag{3.27}$$

Remark 1. *The special case where* $v^* = 0, w_j^* = |y_j|$ *yields a trivial solution where the path norm is* 0. *In the following, we aim to find candidate non-trivial solution, i.e., satisfying* $v^* > 0$. *At the end of the procedure, it is however necessary to compare the candidate solutions with the trivial one, and output the one achieving the smallest objective value* h_λ.

A key insight following Lemma 27 is that the elements of $\mathbf{w}^*$ solving (3.25) satisfy a monotonic relation in magnitude, correlated with the magnitude of the elements of $\mathbf{y}$; this is formulated by the next Corollary.

Corollary 28. *Let* $(v^*, \boldsymbol{w}^*) \in \mathbb{R}_+\times\mathbb{R}_+^m$ *be an optimal solution of* (3.25) *for a given* $(x,\mathbf{y}) \in \mathbb{R}\times\mathbb{R}^m$. *Then, we have the following:*

1. *The vector* $\boldsymbol{w}^*$ *satisfies that for any* $j,l \in \{1,2,\ldots,m\}$ *it holds that* $w_j^* \geq w_l^*$ *if only if* $|y_j| \geq |y_l|$.

2. *Let* $\bar{\mathbf{y}}$ *be the sorted vector of* $\mathbf{y}$ *in descending magnitude order. Suppose that* $v^* > 0$ *and let* $s = |\{j : w_j^* > 0\}|$. *Then,*

$$v^* = \frac{1}{1-s\lambda^2}\left(|x| - \lambda\sum_{j=1}^{s}|\bar{y}_j|\right), \tag{3.28}$$

where we use the convention that $\sum_{j=1}^{0}|\bar{y}_j| = 0$.

Proof. The first part follows trivially from the stationarity conditions on $\mathbf{w}^*$ given in Lemma 27.

Suppose that $\mathbf{w}^*$ contains precisely s non-zero entries. From the first order condition (3.26) of Lemma 27 and the observation of point 1, the non-zero entries must be the one associated with the y_j with largest magnitude, i.e., $\{w_j*, j = 1,\dots,m : w_j* > 0\} = \{\bar{y}_j, j = 1,\dots,s\}$. Hence, we have

$$\sum_{j=1}^{m} w_j^* = \sum_{j=1}^{s} |\bar{y}_j| - \lambda s v^*.$$

By plugging the latter in the stationarity condition (3.27) and assuming that $v^* > 0$, we obtain

$$v^* = |x| - \lambda \sum_{j=1}^{s} |\bar{y}_j| + \lambda^2 s v^*,$$

which implies the result (3.28) after solving for v^*. □

Without loss of generality, **we assume hereafter that the vector y is already sorted in decreasing magnitude order**, such that the s non-zero entries of w^* are always the first s entries. To supplement the results above, we now show that we can actually upper-bound the sparsity level of the prox-grad output by adjusting the value of λ.

Lemma 29 (Sparsity bound)**.** *Let $(v^*, \boldsymbol{w}^*) \in \mathbb{R}_+ \times \mathbb{R}_+^m$ be an optimal solution of* (3.25) *for a given $(x, \mathbf{y}) \in \mathbb{R} \times \mathbb{R}^m$. Suppose that $v^* > 0$, and denote $S = \{j : w_j^* > 0\}$. Then $|S| \leq \lambda^{-2}$.*

Proof. Since $(v^*, \mathbf{w}^*)$ is an optimal solution of (3.25) and the objective function in (3.25) is twice continuously differentiable, $(v^*, \mathbf{w}^*)$ satisfies the second order necessary optimality conditions (Bertsekas, 1999, Ex. 2.1.10). That is, for any $d \in \mathbb{R} \times \mathbb{R}^m$ satisfying that $(v^*, \mathbf{w}^*) + d \in \mathbb{R}_+ \times \mathbb{R}_+^m$ and $d^T \nabla h_\lambda(v^*, \mathbf{w}^*; x, \mathbf{y}) = 0$ it holds that

$$d^T \nabla^2 h_\lambda(v^*, \mathbf{w}^*; x, \mathbf{y}) d = d^T \begin{pmatrix} 1 & \lambda & \cdots & \lambda \\ \lambda & 1 & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ \lambda & 0 & 0 & 1 \end{pmatrix} d \geq 0,$$

where the first row/column corresponds to v and the others correspond to w. Noting that for any $j \in S$ it holds that $\frac{\partial h_\lambda}{\partial w_j}(v^* \mathbf{w}^*; x, \mathbf{y}) = 0$, we have that the submatrix of $\nabla^2 h_\lambda(v^*, \mathbf{w}^*; x, \mathbf{y})$ containing the rows and columns corresponding to the positive coordinates in $(v^*, \mathbf{w}^*)$ must be positive semidefinite.

Since the the minimal eigenvalue of this submatrix equals $1 - \lambda\sqrt{|S|}$, we have that $\lambda^{-2} \geq |S|$. □

Combining the previous results, we conclude that we can restrict the solution of (3.25) to the

one of the form

$$\begin{aligned} v^{(s)} &= \frac{1}{1-s\lambda^2}\left(|x| - \lambda \sum_{j=1}^{s} |y_j|\right) \\ w_j^{(s)} &= |y_j| - \lambda v^{(s)} \text{ for } j \in [s], \text{ and } w_j^{(s)} = 0 \text{ otherwise.} \end{aligned} \tag{3.29}$$

for any $s \in \{0,\dots,\bar{s}\}$ where $\bar{s} = \min(\lfloor \lambda^{-2} \rfloor, m)$. Moreover, the function h_λ is monotonically decreasing in the sparsity level s, which implies that instead of exhaustively checking the value of h_λ for any sparsity level, we can perform a binary search.

Lemma 30. *Let $\bar{s} = \min(\lfloor \lambda^{-2} \rfloor, m)$. For all integer $s \in \{2,3,\dots,\bar{s}\}$, we have that*

$$h_\lambda(v^{(s)}, \boldsymbol{w}^{(s)}; x, \boldsymbol{y}) < h_\lambda(v^{(s-1)}, \boldsymbol{w}^{(s-1)}; x, \boldsymbol{y}). \tag{3.30}$$

At this point, since the value of h_λ decreases with the sparsity level s, we may be tempted to compute the maximal allowed value $\bar{s} = \min(\lfloor \lambda^{-2} \rfloor, m)$ and return the associated pair $(v^{(\bar{s})}, \mathbf{w}^{(\bar{s})})$ as defined in (3.29). However, recall that the solution to (3.25) must satisfy the non-negativity constraint $v, \mathbf{w} \in \mathbb{R}_+ \times \mathbb{R}_+^m$. Since this constraint is not ensured by the definition of $(v^{(s)}, \mathbf{w}^{(s)})$ in (3.29), it must be additionally imposed to the choice of the optimal sparsity level s^*.

Corollary 31. *Suppose that there exists a non-trivial optimal solution of* (3.25). *Denote $\bar{s} = \min(\lfloor \lambda^{-2} \rfloor, m)$ and let*

$$s^* = \max\left\{s \in \{0,\dots,\bar{s}\} : v^{(s)}, w_s^{(s)} > 0\right\}. \tag{3.31}$$

Then $(v^{(s^*)}, w^{(s^*)})$ is an optimal solution of (3.25).

Note that since, by definition, the s first entries of the vector $\mathbf{w}^{(s)}$ are ordered in decreasing order, the constrained $w_s^{(s)} > 0$ ensures that the full vector $\mathbf{w}^{(s)}$ has exactly s nonzero entries, which are all strictly positive.

The final ingredient required for designing an efficient algorithm is the following monotone property of the positivity criterion in problem (3.31):

Lemma 32. *For any $k \in [\bar{s}]$, we have*

$$v^{(k)} > 0, \boldsymbol{w}^{(k)} > 0 \Rightarrow v^{(i)} > 0, \boldsymbol{w}^{(i)} > 0, \ \ \forall i < k.$$

This property implies that the optimal sparsity parameter s^* can be efficiently found using a binary search approach.

We conclude this section by combining all the ingredients above to develop Algorithm 7, and to prove that it yields a solution to (3.24).

Theorem 33 (Prox computation)**.** *Let $(v^*, \boldsymbol{w}^*)$ be the output of Algorithm 7 with input $x, \boldsymbol{y}, \lambda$, assuming that $\boldsymbol{y}$ is sorted in decreasing magnitude order. Then $(v^*, \boldsymbol{w}^*)$ is a solution to* (3.24).

Algorithm 7 Single-output robust-sparse proximal mapping

Input: $x \in \mathbb{R}$, $\mathbf{y} \in \mathbb{R}^m$ sorted in decreasing magnitude order, $\lambda > 0$.

$v^* = 0, \mathbf{w}^* = |\mathbf{y}|$
$s_{\text{lb}} \leftarrow 0$, $s_{\text{ub}} \leftarrow \min(\lfloor \lambda^{-2} \rfloor, m)$, $s \leftarrow \lceil (s_{\text{lb}} + s_{\text{ub}})/2 \rceil$
while $s_{\text{lb}} \neq s_{\text{ub}}$ **do**
    $v^{(s)} = \dfrac{1}{1 - s\lambda^2} \left(|x| - \lambda \sum_{j=1}^{s} |y_j| \right)$
    $w_j^{(s)} = |y_j| - \lambda v^{(s)}$, $j \in [s]$ and $w_j^{(s)} = 0$ otherwise
    if $v > 0$, $w_s > 0$ **then**
        $s_{\text{lb}} \leftarrow s$, $s \leftarrow \lceil (s_{\text{lb}} + s_{\text{ub}})/2 \rceil$
        $(v^*, \mathbf{w}^*) \leftarrow (v, \mathbf{w})$
    else if $v < 0$ **then** $s_{\text{ub}} \leftarrow s$, $s \leftarrow \lceil (s_{\text{lb}} + s_{\text{ub}})/2 \rceil$
    else $s_{\text{lb}} \leftarrow s$, $s \leftarrow \lceil (s_{\text{lb}} + s_{\text{ub}})/2 \rceil$
return $(\text{sign}(x) \cdot v^*, \text{sign}(\mathbf{y}) \circ \mathbf{w}^*)$

Proof. We will show that $(v^*, \mathbf{w}^*)$ is an optimal solution to (3.24) by arguing that Algorithm 7 chooses the point with the smallest h_λ value out of a feasible set of solutions containing an optimal solution of (3.24).

By Lemma 26 it is sufficient to prove that $(|v^*|, |\mathbf{w}^*|)$ is an optimal solution of (3.25), as this will imply the optimality of (v^*, w^*); Recall that Lemma 25 establishes that there exists an optimal solution to (3.25).

If the trivial solution is the only optimal solution to (3.25), then obviously it will be the output of Algorithm 7. Otherwise, the point described in Corollary 31 is an optimal solution. Assume that Algorithm 7 returned the point $(v^{(s_{\text{out}})}, \mathbf{w}^{(s_{\text{out}})})$ for some $s_{\text{out}} \in [\bar{s}]$, meaning in particular that $(v^{(s_{\text{out}})}, \mathbf{w}^{(s_{\text{out}})}) > 0$. By definition, $s^* \geq s_{\text{out}}$. If $s_{\text{out}} < s^*$, then at some $s < s^*$ we had that $v^{(s)} < 0$. Since the value of $v^{(i)}$ is monotonic decreasing in the sparsity level, this implies that $v^{(s^*)} < 0$, which is a contradiction.

Hence, if Algorithm 7 did not return the trivial solution, then $(v^*, \mathbf{w}^*) = (v^{(s^*)}, \mathbf{w}^{(s^*)})$, meaning that $(\text{sign}(x) \cdot v^*, \text{sign}(\mathbf{y}) \circ \mathbf{w}^*)$ is a solution to (3.24). □

Time complexity of Algorithm 7. In the worst case where $m \leq \lambda^{-2}$, the number of searches for finding s^* is at most $\log_2(m)$. Each step of the binary search requires to compute $v^{(s)}$, and in particular $\sum_{j=1}^{s} |y_j|$, as well as $w_j^{(s)}$, $j = 1, \ldots, s$, each taking $\mathcal{O}(s)$ steps. Thus, the overall loop complexity is $\mathcal{O}(m \log m)$.

Moreover, this algorithm assumes that the input vector $\mathbf{y}$ is already sorted in decreasing magnitude order. This can easily be achieved by a sorting procedure in time $\mathcal{O}(m \log m)$.

3.2.4 Path norm regularization of deep neural networks

We now treat the more general case of deep neural networks. Similarly, we will attempt to design an efficient procedure to compute the proximal operator of P_1 defined in (3.15). However, this task turns out to be quite complicated due to the exponential number of terms involved in the path norm.

Therefore, we first develop an efficient procedure for estimating the proximal operator of the path norm associated with a single path of arbitrary length, i.e., corresponding to a network with unit width and of arbitrary depth. Then, we propose two heuristic methods to approximate the proximal operator of the path-norm in the general case.

The Proximal Mapping of a Single Path

The proximal mapping of the 1-path norm associated with a unit-width neural network with depth d is defined as follows:

$$\operatorname{prox}_{\lambda P_1}(\mathbf{z}) = \operatorname*{arg\,min}_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}-\mathbf{z}\|^2 + \lambda \prod_{i=1}^{d} |w_i|\,. \tag{3.32}$$

Similarly as for the previous single layer case, since the solution $\mathbf{w}^*$ tends to minimize both the ℓ_2 distance to $\mathbf{z}$ and the symmetric term $\prod_{i=1}^{d}|w_i|$, w_i^* must have the same sign as z_i (or be 0). Hence, we only need to solve for the magnitude of w_i, i.e., solve

$$\operatorname*{arg\,min}_{\mathbf{w}\in\mathbb{R}_+^m} \frac{1}{2}\sum_{i=1}^{m}(w_i-|z_i|)^2 + \lambda\, w_1\cdots w_d, \tag{3.33}$$

Moreover, since the regularizer in (3.33) is symmetric, the order of the elements with respect to their magnitudes is maintained by the optimal solution.

Lemma 34. *Suppose that* $|z_1|\geq|z_2|\geq\cdots\geq|z_d|$, *and let* $\boldsymbol{w}^*$ *be an optimal solution of* (3.33). *Then*

$$w_1^*\geq w_2^*\geq\cdots\geq w_d^*. \tag{3.34}$$

It is not hard to derive from (3.33) that if one weight is set to zero, then the regularizer has no influence on the solution, and thus the solution is trivial. From Lemma 34, the zero element must be the one corresponding to the smallest z_i.

Corollary 35. *If there exists an optimal solution* w^* *of* (3.32) *with* $w_i^*=0$ *for some* i. *Then* $|z_i|=\min_j|z_j|$, *and the optimal solution satisfies that* $w_j^*=|z_j|$ *for all* $j\neq i$.

Similarly as in the single layer analysis, let us make the following conventions until the end of this section:

A. **Order:** It holds that $|z_1| \geq |z_2| \geq \cdots \geq |z_d|$;

B. **Nontrivial Solution:** There exists a nontrivial solution $\mathbf{w}^*$ of (3.33), that is, satisfying that $w_i^* > 0$ for all $i = 1, 2, \ldots, d$.

We will now solve (3.33) under the assumptions above and we will regard any solution as positive, i.e., $\mathbf{w} > 0$. Once the possible solution satisfying $\mathbf{w} > 0$ is found, it must be compared to the *trivial* possible solution of Corollary 35 in terms of the objective value in (3.32). The one achieving the smallest objective value must then be the solution.

Lemma 36 (First-order optimality conditions)**.** *Let $\boldsymbol{w}^*$ be an optimal solution of* (3.33)*. Then, we have:*

$$w_i^* - |z_i| + \lambda \frac{w_1^* \cdot w_2^* \cdots w_d^*}{w_i^*} = 0, \qquad i = 1, 2, \ldots, d. \tag{3.35}$$

Proof. This set of equations is obtained by setting the gradient of the objective of (3.33) to 0. □

The optimality conditions imply the following useful result.

Corollary 37. *Let $\boldsymbol{w}^*$ be an optimal solution of* (3.33)*. Then, we have:*

$$w_i^*(|z_i| - w_i^*) = w_j^*(|z_j| - w_j^*), \qquad \forall i, j = 1, 2, \ldots, d. \tag{3.36}$$

By fixing the value of w_1^* and solving the quadratic equation (3.36) for w_i^*, $i = 2, \ldots, d$, we find that

$$w_i^* = \frac{1}{2}\left(|z_i| \pm \sqrt{|z_i|^2 - 4w_1^*(|z_1| - w_1^*)}\right). \tag{3.37}$$

For $i = 2, \ldots, d-1$, we will argue that the + sign is the only possibility in equation (3.37). For $i = d$, the choice of sign is not clear, and we instead write

$$w_d^* = |z_d| - \lambda w_1^* \cdot w_2^* \cdots w_{d-1}^* \tag{3.38}$$

as given by equation (3.35) with $i = d$. Hence, solution variables w_i^*, $i = 2, \ldots, d$ can all be expressed only in term of the solution variable w_1^*. The problem hence boils down to finding w_1^*. To this end, we again use equation (3.35) using $i = 1$ so that w_1^* is a solution of the following nonlinear equation

$$w_1^* = |z_1| - \lambda w_2^*(w_1^*) \cdot w_3^*(w_1^*) \cdots w_d^*(w_1^*), \tag{3.39}$$

where for $i = 2, \ldots, d$, $w_i^*(w_1^*)$ is the value of w_i^* given w_1^* as given in equations (3.37) and (3.38). We now formulate all of this discussion properly.

Lemma 38 (Properties of solutions for (3.33))**.** *Let $\boldsymbol{w}^*$ be an optimal solution of* (3.33) *(such that $\boldsymbol{w}^* > 0$). Then, the following hold:*

1. *For any $i = 1, 2, \ldots, d-1$, the element w_i^* satisfies that*

$$\frac{1}{2}|z_i| + \frac{1}{2}\sqrt{|z_i|^2 - |z_d|^2} \leq w_i^* \leq |z_i|. \tag{3.40}$$

2. *For any $i = 2, \ldots, d-1$, the element w_i^* satisfies that*

$$w_i^* = \frac{1}{2}\left(|z_i| + \sqrt{|z_i|^2 - 4w_1^*(|z_1| - w_1^*)}\right),$$

and

$$w_d^* = |z_d| - \lambda w_1^* \cdot w_2^* \cdots w_{d-1}^*.$$

3. *It holds that*

$$\begin{aligned} w_1^* = |z_1| - \frac{\lambda}{2^{d-2}}\Bigg(|z_d| - \frac{\lambda}{2^{d-2}} w_1^* \prod_{i=2}^{d-1}\left(|z_i| + \sqrt{|z_i|^2 - 4w_1^*(|z_1| - w_1^*)}\right)\Bigg) \\ \cdot \prod_{i=2}^{d-1}\left(|z_i| + \sqrt{|z_i|^2 - 4w_1^*(|z_1| - w_1^*)}\right). \end{aligned} \tag{3.41}$$

Lemma 38 suggests that candidates for an optimal solution of (3.33) can be found by finding solutions w_1^* of the nonlinear univariate equation (3.41) over a specific bounded interval $\frac{1}{2}|z_1| + \frac{1}{2}\sqrt{|z_1|^2 - |z_d|^2} \leq w_1^* \leq |z_1|$ (equation (3.40)). This is considered an easy task in optimization, e.g., using grid-search. Algorithm 8 is thus obtained by making the change of variable $\beta = w_1^* - \frac{|z_1|}{2}$.

Complexity of Algorithm 8: The first step in the procedure involves sorting the weights along the path, and has complexity $\mathcal{O}(d \log d)$. Although we do not theoretically bound the number of solutions of equation (3.42), we empirically observe that this equation has a finite number of solutions (at most 4 in practice), independently of the path's depth. Since evaluating the RHS of equation (3.42) takes time $\mathcal{O}(d)$, and since grid search can be trivially parallelized on GPU, applying grid search over a bounded domain has the same complexity $\mathcal{O}(d)$. Overall, we thus conclude that the total complexity of Algorithm 8 is $\mathcal{O}(d \log d)$.

Heuristic extension to the general case: Stochastic regularization

We now leverage the single-path proximal mapping procedure derived in the previous section to try to solve the general problem (3.15). We point out two crucial facts:

First, recalling equation (3.15), the regularizer has a finite sum structure,

$$P_1(\mathbf{W}) = \sum_{s \in S} g(W[s]), \quad g(w_1, w_2, \ldots, w_d) \equiv \prod_{i=1}^{d} |w_i| \tag{3.43}$$

Algorithm 8 Single path proximal operator

Input: Weights along the path $\mathbf{z} = (z_1, \ldots, z_d) \in \mathbb{R}^d$, regularization parameter $\lambda \geq 0$.
Output: $\text{prox}_{\lambda P_1}(\mathbf{z})$

1: $\pi \leftarrow \text{argsort}(|z|)$ in decreasing order.
2: $\tilde{z}_i \leftarrow |z|_{\pi(i)}$
3: Define $f_\lambda(\tilde{\mathbf{z}}, \beta) \equiv 2^{2-d}\lambda \prod_{i=2}^{d-1}\left(\tilde{z}_i + \sqrt{4\beta^2 + \tilde{z}_i^2 - \tilde{z}_1^2}\right)$
4: Find the set B of values β satisfying:

$$\beta = \frac{\tilde{z}_1}{2} - \tilde{z}_d f_\lambda(\tilde{\mathbf{z}}, \beta) - (2\beta + \tilde{z}_1) f_\lambda^2(\tilde{\mathbf{z}}, \beta) \tag{3.42}$$

subject to $\frac{1}{2}\sqrt{\tilde{z}_1^2 - \tilde{z}_d^2} \leq \beta \leq \frac{1}{2}\tilde{z}_1$
5: For each value of β, compute:

- $w_1^\beta \leftarrow \beta + \frac{\tilde{z}_1}{2}$
- $w_i^\beta \leftarrow \frac{1}{2}(\tilde{z}_i + \sqrt{\tilde{z}_i^2 - 4w_1^\beta(\tilde{z}_1 - w_1^\beta)}),\ i = 2, \ldots, d$
- $w^\beta \leftarrow \text{sign}(z) \odot \pi^{-1}(w^\beta)$, where $\odot$ stands for element-wise multiplication

6: Compute the trivial candidate solution $w_i^0 = 0$ for $i = \arg\min_k |z_k|$, $w_j^0 = z_j$ for $j \neq i$
7: **Return** $w \in \{w^\beta : \beta \in B\} \cup \{w^0\}$ achieving the smallest objective value in (3.32)

where for a path $s = [s_1, s_2, \ldots, s_{d+1}]$, $W[s] \equiv [W_1(s_2, s_1), W_2(s_3, s_2), \ldots, W_d(s_{d+1}, s_d)]$. Hence, even though there are exponentially many terms in the sum, it is possible to obtain a stochastic estimate by sampling a random subset of paths.

Second, if two paths s, s' do not share any variable we can compute the proximal mapping of their sum $g(W[s]) + g(W[s'])$. This is achieved by applying the single-path proximal operator independently for s and s'. This is valid for any number of paths not sharing any weight due to the following known result.

Lemma 39. *Let $\boldsymbol{z}^{(i)} \in \mathbb{R}^{n_i}$ and let $\boldsymbol{z} = [\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(k)}] \in \mathbb{R}^{n_1 + \cdots + n_k}$. Suppose that $f(\boldsymbol{z}) = \sum_i f_i(\boldsymbol{z}^{(i)})$ for some functions $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$. Then,* $\text{prox}_f(\boldsymbol{z}) = [\text{prox}_{f_1}(\boldsymbol{z}^{(1)}), \ldots, \text{prox}_{f_k}(\boldsymbol{z}^{(k)})]$.

This motivates the search for a stochastic estimator of the 1-path-norm, that is composed of paths that do not share any variables. Such estimator would allow the application of the single-path prox over multiple paths in the network. We hence need a distribution $\mathcal{T}$ over sets of paths satisfying the following properties:

Assumption 40. *Let $\mathcal{T}$ be a distribution over sets paths in S. Assume that*

1. ***Unbiased estimation of the regularizer:*** *$\mathcal{T}$ covers the set of paths evenly, in the sense that for any two paths $s, s' \in S$ and $T \sim \mathcal{T}$, $p(s \in T) = p(s' \in T)$. This implies that estimating P_1 stochastically using paths generated using $\mathcal{T}$ yields an unbiased estimator, i.e.,*

$$P_1^T(\boldsymbol{W}) \equiv \frac{|S|}{|T|} \sum_{s \in T} g(W[s]) \tag{3.44}$$

is such that $\mathbb{E}_{T\sim\mathscr{T}} P_1^T(\boldsymbol{W}) = P_1(\boldsymbol{W})$.

2. ***Non-overlapping paths:*** *$\mathscr{T}$ only generates sets of independent paths, i.e., for any $T \sim \mathscr{T}$, any two paths $s, s' \in T$ satisfy $s \cap s' = \emptyset$, meaning that they do not share any variable.*

3. ***Constant size:*** *$\mathscr{T}$ only generates sets of path with the same cardinality, i.e., $\exists k \in \mathbb{N}^*$ such that $\mathscr{T} \in \mathscr{P}(S^k)$.*

The third assumption is made to simplify the algorithm. To substantiate the above, let us consider the following trivial example.

Example 41 (single-path sampler)**.** *Define the trivial sampler $\mathscr{T}_0 \equiv Unif(S)$, i.e., it generates sets of cardinality* 1 *containing a single path taken uniformly at random from S. It is easy to see that this sampler satisfies all conditions of Assumption 40.*

Obviously, we would like to use a sampler that generates as largest sets as possible. Before developing an efficient path sampler $\mathscr{T}$, let us describe the proposed optimization procedure for solving (3.16) (Algorithm 9). The idea is to use the stochastic proximal gradient descent, by replacing at each iteration the path-norm regularizer with a stochastic unbiased estimate P_1^T, $T \sim \mathscr{T}$. A stochastic gradient update is then performed on the smooth term $\frac{1}{N}\sum_{i=1}^N \mathscr{L}\left(f_{\mathbf{W}}(x_i), y_i\right)$, and a proximal step is performed on the estimate of the non smooth term $\lambda P_1(\mathbf{W})$. The latter step is done by repeatedly using Algorithm 8 over the paths in the set T. We call this algorithm Stochastic Gradient with Stochastic Prox (SGSP) and the pseudo-code is presented in Algorithm 9.

Algorithm 9 SGSP

Input: Initial weight matrices $\mathbf{W}^0$, step size $\eta > 0$, regularization parameter $\lambda > 0$, path sampler $\mathscr{T}$ satisfying Assumption 40

1: **for** $k \geq 1$ **do**
2: Sample a Stochastic Gradient $\tilde{G}^{k-1}$ of $\frac{1}{N}\sum_{i=1}^N \mathscr{L}\left(f_{\mathbf{W}}(x_i), y_i\right)$ at $\mathbf{W} = \mathbf{W}^{k-1}$.
3: $\mathbf{W}^{k-\frac{1}{2}} \leftarrow \mathbf{W}^{k-1} - \eta\tilde{G}^{k-1}$
4: Sample non-overlapping paths $T_k \sim \mathscr{T}$
5: $\mathbf{W}^k \leftarrow \text{prox}_{\lambda\eta P_1^{T_k}}(\mathbf{W}^{k-\frac{1}{2}})$

This way of sampling paths avoids the situation where two or more paths share weights. In such cases, the proximal mapping requires solving an optimization problem whose solution is currently out of reach due to the complexity of the first-order conditions and the non-convexity of the path-norm.

Intuitively, the use of a stochastic regularizer in Algorithm 9, and the use of stochastic gradients in SGD, are solutions to a similar problem: the complexity of minimizing a sum with a large number of terms. Algorithm 9 simultaneously avoids the complexity of computing the full

gradient and the full proximal operator by using unbiased stochastic estimators using batches smaller size.

Our *stochastic regularization* approach is also similar to randomized proximal coordinate descent schemes (e.g., Lin et al. (2014); Fercoq and Richtárik (2015)), only that in our case, there are no predetermined blocks (they change at each iteration according to the set T_k). In this way, it is closely related to randomized Proximal Coordinate Descent schemes (e.g., Lin et al. (2014); Fercoq and Richtárik (2015)).

We now turn to the task of providing a sampling mechanism to sample a large number of non-overlapping paths satisfying Assumption 40, as obviously, the procedure in Example 41 would yield a poor estimator with large variance.

Sampling a large number of non-overlapping paths: This is crucial for the performance of our training algorithm: (i) It determines the amount of weights that are regularized at each iteration; and more importantly, (ii) It significantly affects the variance of the path-norm estimator.

We first establish the maximal number of concurrent non-overlapping paths that can be sampled. Here we identify a path with a sequence of integers corresponding to the indices of the neurons that are traversed by the path.

Lemma 42. *Two paths* $(s_1,\dots,s_{d+1})$ *and* $(s'_1,\dots,s'_{d+1})$ *are called non-overlapping if for all* $0 \leq i \leq d$, $s_i = s'_i$ *implies* $s_{i+1} \neq s'_{i+1}$. *Let* T *be a set of non-overlapping paths in a network with layer sizes* $n_1,\dots,n_{d+1}$. *It holds that* $|T| \leq k^\star \equiv \min_{i=1}^{d} n_i n_{i+1}$.

In Algorithm 10, we implement a low-complexity procedure yielding a distribution $\mathcal{T}_{max}$ satisfying Assumption 40 generating sets of paths paths of maximum possible size $k^\star$ (Theorem 43). We only require that the set in line 7 of Algorithm 10 is non-empty, which can always be achieved in practice by slightly increasing the layer sizes.

In summary, Algorithm 10 samples a uniform set of neurons at each layer in an independent fashion, and connects them in a way that ensures the non-overlapping condition while achieving the largest possible number of paths.

The complexity of Algorithm 10 is $\tilde{\mathcal{O}}(dk^\star)$. This is at most the complexity of a forward-pass through the network and hence, can be called at each iteration of the training loop without increasing the overall complexity. However, note that the main *for loop* can be executed in parallel which can make the algorithm $\tilde{\mathcal{O}}(k^\star)$ in practice, if enough cores are available.

Theorem 43. *Let* $\mathcal{T}_{max}$ *denote the distribution over sets of paths generated by Algorithm 10. Then,* $\mathcal{T}_{max}$ *satisfies Assumption 40, and generates sets of paths with maximal cardinality* $k^\star \equiv \min_i n_i n_{i+1}$.

While appealing, we emphasize the heuristic nature of this approach. Although, recently,

Algorithm 10 Maximal non-overlapping path sampler

Input: Layer sizes $(n_1, \ldots, n_{d+1})$, $k^\star = \min_i n_i n_{i+1}$
Output: A set of paths $T \sim \mathcal{T}_{max}$ in the form of a matrix of dimensions $(k^\star, d+1)$

$i^\star \leftarrow \arg\min\{n_i n_{i+1} : i = 1, \ldots, d\}$
for $i = 0, \ldots, L$ **do**
  if $i \equiv i^\star \pmod 2$ **then**
    $a \leftarrow n_{i^\star+1}, b \leftarrow n_{i^\star}$
  else
    $a \leftarrow n_{i^\star}, b \leftarrow n_{i^\star+1}$
  $r \leftarrow \text{Uniform}\{U \subseteq [n_{i+1}] : |U| = a\}$ ▷ uniformly random ordered subset
  for $0 \le k \le b-1, 0 \le j \le a-1$ **do**
    if $i \equiv i^\star \pmod 2$ **then**
      $M(jb+k, i) \leftarrow r_j$
    else
      $M(ka+j, i) \leftarrow r_j$
Return M

some significant advances have been made in this regard (Xu et al., 2019a; Davis and Drusvyatskiy, 2019; Xu et al., 2019b; Metel and Takeda, 2019; Hallak et al., 2021; Tran-Dinh et al., 2021), the stochastic Prox-Grad approach with the 1-path-norm regularizer is still without any controllable guarantees.

Heuristic extension to the general case: Layer-wise proximal estimation

SGSP still suffers from the fact that not all weights in the network are regularized at each iteration, given that only a few paths are sampled. We propose an alternative heuristic approach for approximating the proximal operator of the full 1-path-norm regularizer P_1. Recall that the proximal operator of λP_1 is given by

$$\text{prox}_{\lambda P_1}(\mathbf{Z}) = \underset{W_1,\ldots,W_d}{\arg\min} \frac{1}{2} \sum_{i=1}^{d} \|W_i - Z_i\|_F^2 + \lambda \mathbf{1}^T |W_d||W_{d-1}| \cdots |W_1| \mathbf{1}. \tag{3.45}$$

For some layer i, let us now fix the value of variables W_j, $j \neq i$ to $W_j = Z_j$, for all weights but the i-th layer, and solve (3.45) only for W_i, i.e., denote

$$\Phi_i^\lambda(\mathbf{Z}) \equiv \underset{W_i}{\arg\min} \frac{1}{2} \|W_i - Z_i\|_F^2 + \lambda \mathbf{1}^T |Z_d| \cdots |Z_{i+1}||W_i||Z_{i-1}| \cdots |Z_1| \mathbf{1}. \tag{3.46}$$

The problem now becomes an easy convex problem with a closed form solution. Indeed, the right term can be written as $\lambda \text{Tr}(M_i |W_i|)$ where $M_i := |Z_{i-1}| \cdots |Z_1| \mathbf{1}\mathbf{1}^T |Z_d| \cdots |Z_{i+1}|$. Solving (3.46) is thus equivalent to solving a weighted ℓ_1-norm proximal operator, and its solution is therefore given by the soft-thresholding operator:

$$\Phi_i^\lambda(Z) = \text{sign}(Z_i) \odot \max(|Z_i| - \lambda M_i^T, 0). \tag{3.47}$$

where $\odot$ denotes element-wise multiplication. The structure of the regularizer makes it possible to compute these values for each layer in a sequential manner, and the complexity is equivalent to that of a forward-backward pass through the network.

Hence, the overall complexity of the training procedure given by Algorithm 11, named *Heuristic Layer-wise Proximal* (HLP), remains the same as the automatic differentiation approach.

Complexity of Algorithms 9 and 11: In Algorithm 9, the complexity of each proximal step is $\tilde{\mathcal{O}}(dk^\star)$, which is in general smaller than the complexity of computing one stochastic gradient (even with batch-size 1). In Algorithm 11, computing the approximate proximal update can be done efficiently with one additional forward pass over the network. Thus, it also enjoys the same complexity as the gradient step. This makes the iterations of such algorithms comparable to those of SGD/Adam, despite small speed differences that arise in practice due to a non-optimized/non-compiled implementation on a high-level programming language (such as Python).

Algorithm 11 HLP

Input: Initial weight matrices $\mathbf{W}^0$, step size $\eta > 0$, regularization parameter $\lambda > 0$

for $k \geq 1$ **do**
  Sample Stochastic Gradient $\tilde{\nabla} H(\mathbf{W}^{k-1})$
  $\mathbf{W}^{k-\frac{1}{2}} \leftarrow \mathbf{W}^{k-1} - \eta \tilde{\nabla} H(\mathbf{W}^{k-1})$
  for $i = 1, \ldots, d$ **do**
    $\mathbf{W}_i^k \leftarrow \Phi_i^{\eta\lambda}(\mathbf{W}^{k-\frac{1}{2}})$

3.2.5 Experiments

We now evaluate the empirical performance of using the 1-path norm proximal operator for solving 1-path norm regularized problem. We first treat the case of 1-hidden layer networks, for which the proximal operator can be exactly computed, and then proceed to the general case of deep networks using the two proposed heuristics.

1-path norm regularization for shallow networks

The goals in this section are two-fold: (i) First, we empirically demonstrate that using proximal updates yields a better optimizer than using auto-differentiation and (ii) We show that 1-path norm regularization yields more robust models compared to the more classical ℓ_1-norm regularization.

Experimental setup. Our benchmark datasets are MNIST (LeCun and Cortes, 2010), Fashion-MNIST (Xiao et al., 2017b) and Kuzushiji-MNIST (Clanuwat et al., 2018). We train models on these tasks by solving problem (3.20) using the cross-entropy loss. We compare the following two optimizers: SGD, i.e., using the Pytorch auto-differentiation module on the non-smooth

1-path norm term, and Prox-SGD (Algorithm 6). Both SGD and Prox-SGD are ran for 20 epochs using constant learning rate and with batch size set to 100. For each combination of parameters, we train 6 single-layer networks with 100 hidden neurons using the default random initialization.

In addition to the 1-path norm regularized problem, we introduce two baseline algorithms for solving the classification task:

- **Layer-wise regularization (Parseval Networks).** We minimize the cross-entropy loss with a hard constrain on the ℓ_∞-operator-norm of the weight matrices i.e., $\|W\|_\infty \leq \lambda^{-1}$ and $\|V\|_\infty \leq \lambda^{-1}$, as described by Cisse et al. (2017). The projection on such a set is achieved by projecting each row of the matrices onto an ℓ_1-ball using efficient algorithms (Duchi et al., 2008; Condat, 2016). This approach is meant to control the product bound on the right hand side of equation (3.19), which also yields an upper bound on the Lipschitz constant, although less tight than the 1-path norm.

- **ℓ_1-regularization.** We penalize the ℓ_1-norm of the parameters of the network, i.e., $g(V,W) = \|\mathrm{vec}(V)\|_1 + \|\mathrm{vec}(W)\|_1$ and use the Prox-SGD method, given that the proximal operator of the ℓ_1 norm is simply given by soft-thresholding. The ℓ_1-norm regularizer provides an upper boud on the already loose product bound (Neyshabur et al., 2015, Eq. (4)), which makes it less attractive as a regularizer for penalizing the Lipschitz constant.

Convergence of SGD vs Proximal-SGD. We first examine the ability of the optimizers to minimize the regularized empirical loss. Due to the non-differentiability of the ℓ_1- and path-norm regularizers, we expect Prox-SGD to converge faster, and to lower values of the regularized loss, when compared to SGD. This is examined in Figure 3.7, where we plot the value of the loss function across iterations. For both SGD and Prox-SGD, the loss function decays rapidly in the first few epochs. We then enter a second regime where SGD suffers from slow convergence, whereas Prox-SGD continues to reduce the loss at a fast rate. At the end of the 20-epochs, Prox-SGD consistently achieves a lower value of the loss compared to SGD.

Robustness-Sparsity trade-off. Another advantage of Proximal-SGD over plain SGD is that the proximal mappings of both the ℓ_1- and path-norm regularizers can set many weights to *exactly* zero. In Figure 3.8 we plot the average error and robust test error obtained, as functions of the sparsity of the network. Compared to ℓ_1 regularization, the sparsity pattern induced by the 1-path-norm correlates with the robustness to a higher degree. As a drawback, it appears that in more difficult datasets like KMNIST, the 1-path-norm struggles to obtain good accuracy and sparsity simultaneously.

Robustness-Accuracy trade-off. Next, we examine the robustness-accuracy trade-off achieved by 1-path norm regularization, and compare it with that achieved by ℓ_1-norm and layer-wise regularizations. Any training procedure which promotes robustness of a classifier may decrease its accuracy, and this effect is consistently observed in practice (Tsipras et al., 2019).

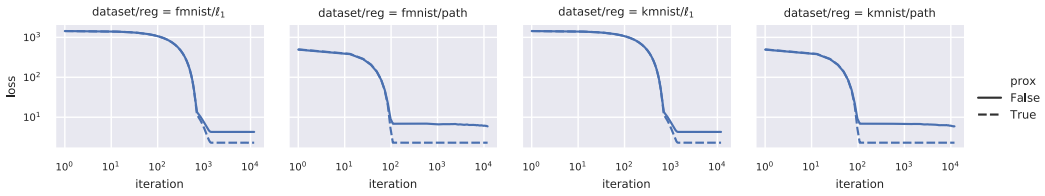


Figure 3.7 – value of regularized cross-entropy loss across iterations.

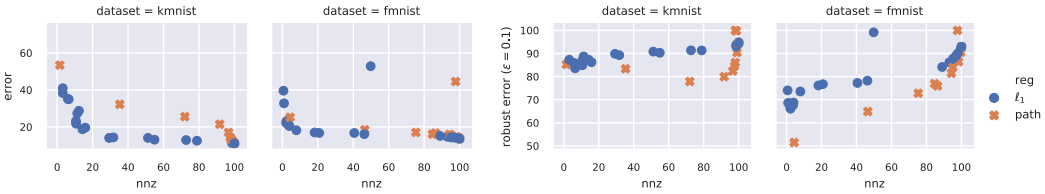


Figure 3.8 – Misclassification test error (left) and robust test error (right) as a function of the percentage of nonzero weights.

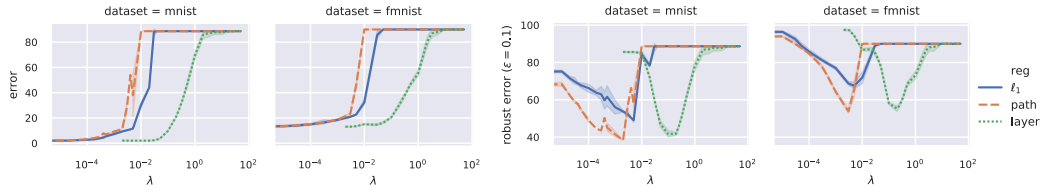


Figure 3.9 – Misclassification test error (left) and robust test error (right) on the test set, as a function of the regularization parameter λ.

Hence, the merits of a regularizer should be measured by how efficiently it can trade-off accuracy for robustness.

Figure 3.9 shows the misclassification error on clean and adversarial examples as a function of λ, and corresponds to the learning rate minimizing the error on clean samples. The adversarial perturbations were obtained by PGD (Madry et al., 2018). We observe that for all three regularization schemes, there exists choices of λ that attain the best possible error on clean samples.

The error obtained by the ℓ_1 regularization degrades significantly. The layer-wise and 1-path-norm regularization achieve a noticeably low error on adversarial examples. Comparing the latter schemes, the 1-path-norm regularization shows only a slight advantage over the layer-wise method.

1-path norm regularization for deep networks

We now turn to evaluating the performance of Prox-SGD for training deeper neural networks with 1-path norm regularization.

Proximal Point Algorithm vs. Automatic Differentiation. We assess the benefits of a proximal-

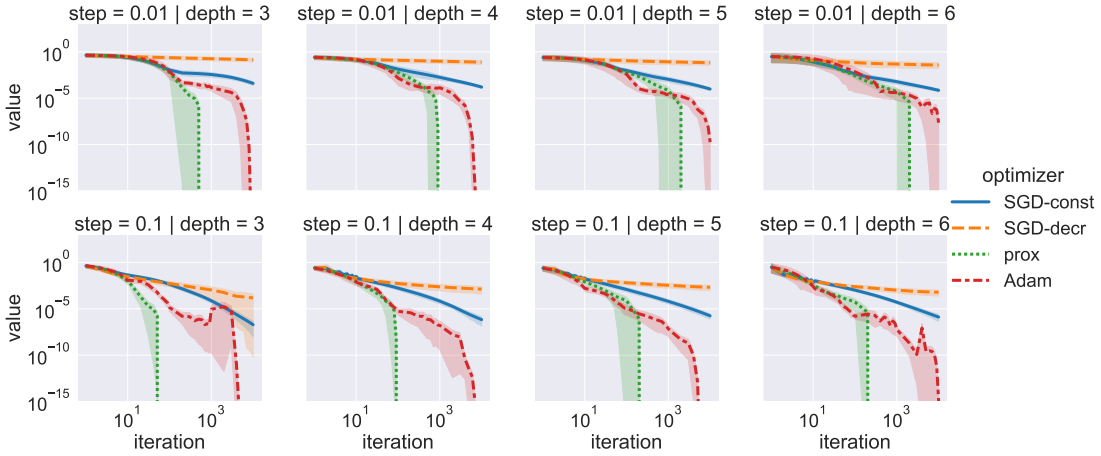


Figure 3.10 – Value of the 1-path-norm of a single path as a function of iteration, for different values of depth. Two step sizes and four different optimizers are considered, SGD with either constant or decreasing step-size, Adam and the proximal point algorithm based on 8 (prox). 1000 repeated runs with random initialization were performed for each parameter combination.

gradient approach in a synthetic scenario of a neural network with a single path $f_{\mathbf{w}}(x) = w_d\sigma(\cdots\sigma(w_1 x))$ with weights $\mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{R}^d$. The loss function is set to be identically zero i.e., $\mathscr{L}(f_{\mathbf{w}}(x), y) = 0$, which allows us to test the effectiveness of our method in minimizing the non-convex non-smooth regularizer in isolation. In this setup, the proximal-gradient scheme (Algorithm 9) with constant step-size $\eta > 0$ boils down to a non-convex version of the proximal point algorithm:

$$\mathbf{w}^{(k+1)} = \text{prox}_{\eta P_1}(\mathbf{w}^{(k)}). \tag{3.48}$$

For different values of depth and step-size, we compare this algorithm against the auto-differentiation alternative, using either SGD (with constant or decreasing step-size) or *Adam* (Kingma and Ba, 2014). We observe that not only the proximal point iterations in (3.48) can achieve the global minimum, equal to zero in this case, but they consistently outperform both SGD and Adam (Figure 3.10). After a large number of iterations, SGD has still not converged and achieves a suboptimal value. Adam is able to get to the global minimum but with orders of magnitude more iterations compared to the proximal method.

Performance on networks composed of independent paths. We generate a synthetic dataset with 100 elements $(x_i, y_i) \in \mathbb{R}^2$ sampled as $y_i \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.1)$, $x_i \overset{i.i.d.}{\sim} \text{Unif}[0, 1]$. We consider the regression problem with quadratic loss, i.e., $\mathscr{L}(a, b) = (a-b)^2$ and 1-path-norm regularization, as in (3.16). We use a neural network architecture composed of multiple non-ovelapping paths and ELU activations. This architecture corresponds to an ensemble of models of the form $f_{\mathbf{w}}(x) = w_d\sigma(\cdots\sigma(w_1 x))$ with $\mathbf{w} \in \mathbb{R}^d$.

In this setting SGSP (Algorithm 9) is precisely the Prox-SGD algorithm, given that we can compute the true proximal operator by applying Algorithm 8 independently over each path.

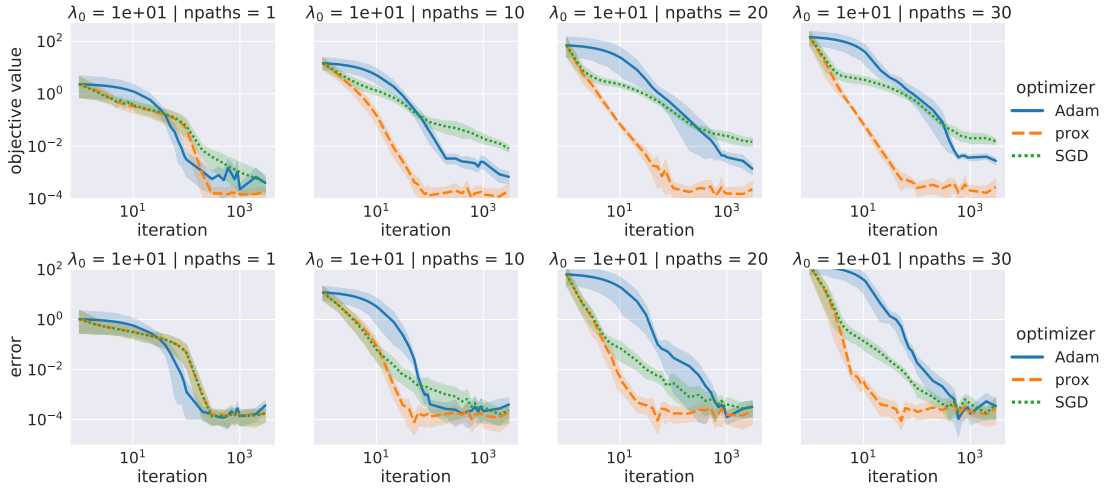


Figure 3.11 – Mean Squared Error per iteration (ℓ_2-Regression) using a neural network architecture composed of *npaths* non-overlapping paths using different optimizers: Adam (blue, solid), SGD (green, dotted) and 9 (SGSP) here labelled *prox* (orange, dashed). We plot the results for the dataset $y_i \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.1)$, $x_i \overset{i.i.d.}{\sim} \text{Unif}[0,1]$. The top row shows the training error while the bottom row shows the test error.

This scenario helps us illustrate the expected performance of the proximal approach in the idealistic case where we have access to the full proximal operator of the network. We divide the regularization parameter by the total number of paths in the network i.e., $\lambda = \lambda_0 / \prod_{i=1}^{d+1} n_i$. This way, a fixed value of the regularization parameter λ_0 can be easily compared for varying sizes of networks, as it effectively corresponds to regularizing the *average* 1-path-norm.

We plot the objective as a function of iteration in Figure 3.11. The proposed SGSP (here called *prox*) outperforms SGD and Adam, and the difference is larger as we increase the size of the network. It is also remarkable that the improvement in convergence speed is not only appreciated for the training loss, but the test loss is also minimized faster using SGSP (bottom row of Figure 3.11).

Performance on real datasets. We train 1-path-norm regularized neural networks on the FashionMNIST Xiao et al. (2017a) and CIFAR10 Krizhevsky (2009) datasets, using SGSP (Algorithm 9) and HLP (Algorithm 11) as well as automatic differentiation (SGD and Adam). We use the cross-entropy loss, ELU activations, batch-size of 200 and train for 700K iterations.

Figure 3.12 summarizes the results. Both SGD/Adam and HLP show a slow speed of convergence, and appear to get stuck at suboptimal values. In contrast, SGSP can attain much lower values of the objective function for an appropriate choice of step size. The momentum/adaptivity of Adam are not useful in this case, i.e., automatic differentiation might not succeed for 1-path-norm regularization. We remark that larger stepsizes than those shown in the figure resulted in divergence. The fact that HLP does not perform as well as SGSP, **serves the purpose of showing the hardness of coming up with good heuristics** that workaround the lack of a

full proximal operator.

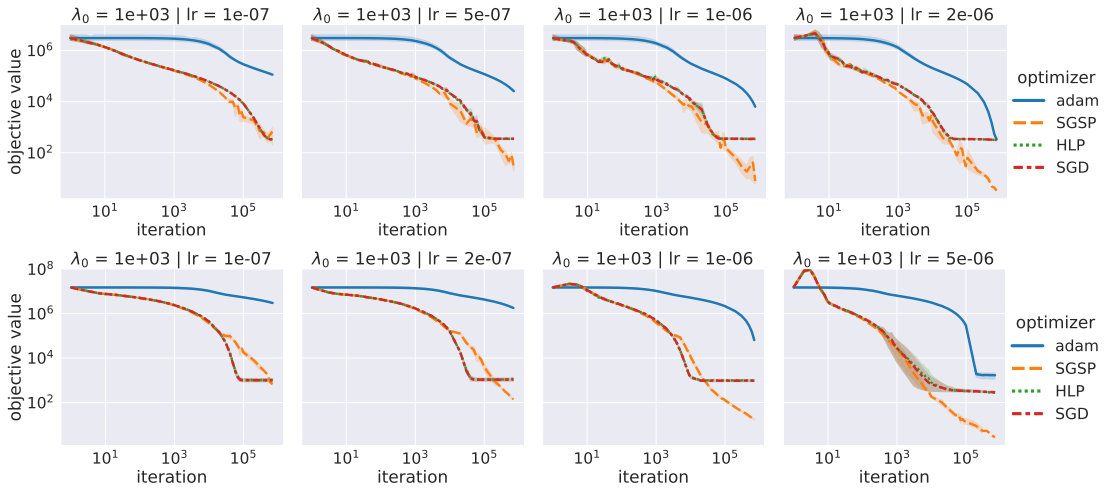


Figure 3.12 – Comparing SGD (red-dashed/dot), Adam (blue-solid), 9 (SGSP, orange-dashed) and 11 (HLP, green-dotted): Value of cross-entropy with average 1-path-norm regularization ($\lambda_0 = 10^3$), as a function of iteration, for different values of learning rate; larger learning rates than that of the last column result in divergence. Six repeated runs with different random splits of the training set were performed for each parameter combination. Architecture chosen is a Fully Connected Network with 6 layers. FMNIST dataset (top) corresponds to a network with layer sizes (28×28, 1000, 784, 1000, 784, 10). CIFAR10 (bottom) corresponds to a network with layer sizes ($3 \times 32 \times 32$, 1000, 1536, 1000, 1536, 10).

Time per iteration. In Figure 3.12 we plot the error per iteration. As we showed, the complexity per iteration of SGSP is equivalent to that of Adam/SGD. However, there might be slight differences due to implementation or constants hidden by the asymptotic complexity analysis. We compute the iterations-per-second of the algorithms: SGD: 6.92, Adam: 5.01, SGSP: 7.07, HLP: 6.24. This shows that our algorithms are competitive, and that the plots on Figure 3.12 would only vary slightly if we change the x-axis to wall-clock time.

3.3 Bibliographic notes

In the work "Lipschitz Constant Estimation of Neural Networks via Sparse Polynomial Optimization" (Section 3.1), the idea of expressing the Lipschitz constant computation as a POP comes from Fabian Latorre. The candidate's contribution to this work was the scaling of the algorithm in the case of sparse networks (Section 3.1.4) as well as the numerical experiments.

In the part on 1-path norm regularization, the candidate contributed to all the results, in collaboration with Nadav Hallak and Fabian Latorre.

4 Robustness to structured environmental changes using causal feature selection

In the two previous chapters, we developed robust methods by focusing on the trained model itself. This was achieved by either perturbing the learner during training, or using explicit regularization. In this chapter, we do not focus on the model training procedure, but rather on the choice of variables that are used by the model in order to solve the desired task.

In the first part of this chapter, we introduce the framework of **Causality**, as well as its relation to robustness based on the work of Bühlmann (2020). This motivates the importance of knowing the right causal structure underlying the data generation. In the second part, we propose an algorithm inferring the causal graph from observational data, in the case of non-linear additive noise models.

4.1 Preliminaries: Causality and robustness

4.1.1 Causality and structural equation models

In classical Machine Learning tasks, we wish to answer questions such as: "If I observe $X = x$, what do I expect the distribution of Y to be?" Such question can be answered by estimating the conditional probability distribution $Y|X = x$. In Causality, we aim to answer different kinds of questions, such as "If I observe $(X, Y) = (x, y)$ and that I change the value of X to x', what do I expect the new distribution of Y to be?" Such question cannot be answered by simply knowing the distribution $Y|X$, but requires to know more about the process underlying the data generation.

Let us mention a concrete toy example, taken from (Peters et al., 2017). Suppose that we observe the altitude A and temperature T of various cities. It is clear that these two variables are correlated, and it is possible to infer from data the conditional distributions $A|T$ as well as $T|A$ so that it becomes possible to approximately predict the temperature given the altitude and vice versa (Figure 4.1). However, a more causal question would be "If the altitude of a city was modified, what would happen to the temperature?" or similarly, "If the temperature of a

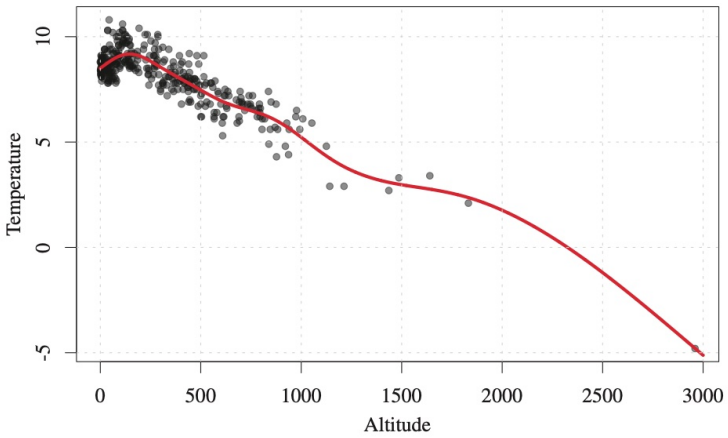


Figure 4.1 – Altitude vs temperature of different cities (Peters et al., 2017).

city was modified, what would happen to its altitude?" Such question requires us to describe the origin of the correlation between A and T.

From physical insights, it is clear that a modification of the altitude would change the temperature, and not vice versa. We hence say that *A causes T*. However, it is not clear how to arrive to such conclusion only by looking at the data of Figure 4.1. Breaking the causal asymmetry between variables is the topic of Causal Discovery, and will be discussed in the second part of this chapter.

In order to characterize the causal relations among all variables, we need to understand how the observed data have been generated in a first place. This is classically done by introducing a so-called Structural Equation Model (SEM), which sequentially describes how each variable is generated based on the already generated one:

$$X_i \leftarrow f_i(\mathrm{pa}_i(X), \epsilon_i), \tag{SEM}$$

where f_i's are arbitrary functions, ϵ_i's are independent random variables and $\mathrm{pa}_i(X)$ selects the coordinates of X which are parents of the node i in some graph. The corresponding directed graph is called the causal graph, and describes what information is needed in order to generate each variable. By construction, since the variables are generated sequentially, and only once, the causal graph must be acyclic, and is hence a Directed Acyclic Graph (DAG).

The sign $\leftarrow$ in (SEM) is to be understood as an assignment, so that the SEM describes a way to generate data by sequentially assigning values to each variable. Each function f_i hence characterizes the distribution $X_i|\mathrm{pa}_i(X)$ which is to be understood as the causal mechanisms relating the causes to the effect. In particular, if one would intervene on the value of a certain variable X_i, then the SEM also describes the effect of such an intervention on the descendant of the node i in the causal graph. On the other hand, the variables which are not descendent

of the node i would not be affected by any intervention on X_i.

Going back to our example about the altitude and the temperature of cities, the corresponding SEM would look like

$$A \leftarrow \epsilon_A,$$
$$T \leftarrow f_T(A, \epsilon_T),$$

and the causal graph would simply be $A \rightarrow T$. The function $f_T(\cdot, \epsilon_T)$ describes the way altitude affects the temperature.

4.1.2 Robustness via causal features selection

It turns out that knowing the causal relations between input and output variables can greatly help in developing robust models. In this section, we present a connection between causal feature selection and robustness, based on the work of Bühlmann (2020).

Recall that feasibility of problem (ROB) requires some similarity assumption between $\mathbf{p}^{\text{data}}$ and $\mathbf{p}^{\text{test}}$. In Chapter 2, we assumed that an adversary could perturb the environment within some limited budget, while in Chapter 3 we considered that the input data could be directly modified within some ball with limited radius. In this Chapter, we will assume that the distributions share the same causal mechanisms.

Let $\mathscr{S}^{\text{data}}$ and $\mathscr{S}^{\text{test}}$ denote the Structural Equation Models associated with $\mathbf{p}^{\text{data}}$ and $\mathbf{p}^{\text{test}}$ respectively. Each model describes the generation of the pair of variables $(X, Y) \in \mathbb{R}^{d+1}$ in their respective environment. Note that Y need not be an effect of variables in X, but can also be a cause, in the sense that Y can be involved in the generation of some variables X_i in the SEMs.

At this point, the distributions $\mathbf{p}^{\text{data}}$ and $\mathbf{p}^{\text{test}}$ could be arbitrarily different. We now describe the set $\mathscr{P}$ involved in (ROB) by making the following assumption on the similarity between $\mathbf{p}^{\text{data}}$ and $\mathbf{p}^{\text{test}}$:

Assumption 44. *The structural equation for generating Y from $pa_Y(X)$ are the same in $\mathscr{S}^{data}$ and $\mathscr{S}^{test}$. That means, the causal graphs associated with $\mathscr{S}^{data}$ and $\mathscr{S}^{test}$ share the same edges pointing towards Y, and the structural equation for generating Y*

$$Y \leftarrow f_Y(pa_Y(X), \epsilon_Y)$$

is the same in both models, i.e., the function f_Y and the random variable ϵ_Y are the same.

Assumption 44 is known as the *invariance of causal mechanisms*. Instead of assuming small unstructured deviation between $\mathbf{p}^{\text{data}}$ and $\mathbf{p}^{\text{test}}$ as was done in the previous chapters, we instead assume here arbitrary large but structured differences. Under this assumption, we can draw an interesting connection between causality and the solution of the associated robust

problem (ROB). Suppose that we perform linear regression using the quadratic loss. Let $\mathscr{P}^{\text{inv}}$ be the space of distribution $\mathbf{p}^{\text{test}}$ satisfying Assumption 44 with respect to $\mathbf{p}^{\text{data}}$. Then, we have

$$b^* = \underset{b\in\mathbb{R}^d}{\arg\min} \max_{\mathbf{p}^{\text{test}}\in\mathscr{P}^{\text{inv}}} \mathbb{E}_{(X,Y)\sim\mathbf{p}^{\text{test}}}[|Y - Xb|^2] = \text{causal parameters},$$

meaning that $\text{supp}(b^*) = \text{pa}_Y(X)$. Hence, when performing linear regression with quadratic loss, only using the causal parameters for Y leads to the most robust classifier in the worst-case sense.

In the more general case, it can be shown that, under Assumption 44, we have

$$Y^{\text{data}}|X^{\text{data}}_{S_{\text{causal}}} \overset{law}{=} Y^{\text{test}}|X^{\text{test}}_{S_{\text{causal}}} \tag{4.1}$$

where the set of causal variables $S_{\text{causal}} \equiv \text{pa}_Y$, and $X \overset{law}{=} Y$ means that the random variables X and Y follow the same distribution. Therefore, if we perfectly predict Y from $X_{S_{\text{causal}}}$ in the training environment, the prediction should perfectly translate into the test environment. Note that the set of parameters S_{causal} may not be the only one satisfying the invariance property (4.1). However, in cases where we know the causal graph, this gives a systematic way to find a set of parameters satisfying such a property.

4.2 Causal discovery for non-linear additive models

In the previous section, we motivated the usefulness of knowing the process underlying the data generation, as opposed to just knowing the joint probability distribution. It allows to pick specific variables to be used for predictions, yielding strong robust properties. However, in practice, we only have access to data sampled from a certain distribution, and we do not have direct access to the causal graph. Therefore, we need a method for estimating the causal graph underlying a certain distribution from samples.

This section is based on the paper Rolland et al. (2022) published at ICML 2022.

4.2.1 Introduction

We focus on causal discovery from purely observational data, i.e., finding a causal Directed Acyclic Graph (DAG) underlying a distribution given samples from this distribution, i.e., from *observational* data. In general, the problem of causal discovery from observational data is ill-posed, since there may be several generative models SEM with various causal structures that yield the same data distribution. Therefore, in order to make the problem well-posed, we need to rely on extra assumptions on the generative process. A popular solution is to assume that the noise injected during the generation of each variable is additive (see equation (4.2)). If, in addition, the link functions are non-linear, it has been shown that such model is identifiable from purely observational data (Peters et al., 2014).

Many causal discovery algorithms maximize a suitable loss function over the set of DAGs. Unfortunately, solving such problem using classical loss functions is known to be NP-hard (Chickering, 1996). Therefore, recent methods focused on heuristic approximations, e.g., 1) by using a greedy approach (PC, FCI (Spirtes et al., 2000; Zhang, 2008), GES (Chickering, 2002), CAM (Bühlmann et al., 2014) and others (Teyssier and Koller, 2012; Larranaga et al., 1996; Singh and Valtorta, 1993; Cooper and Herskovits, 1992; Bouckaert, 1992)), 2) by expressing the problem as a continuous non-convex optimization problem and applying first-order optimization methods (GraNDAG (Lachapelle et al., 2019), NOTEARS (Zheng et al., 2018)), or 3) by using Reinforcement Learning methods (RL-BIC (Zhu et al., 2019), CORL (Wang et al., 2021)).

There are two distinct aspects that make the search over DAGs difficult: the size of the set of DAGs, which grows super-exponentially with the number of nodes, and the acyclicity constraint. In order to reduce the impact of these two difficulties, approaches called order-based methods (Teyssier and Koller, 2012) tackle the problem in two phases. First, we find a certain *topological ordering* of the nodes, such that a node in the ordering can be a parent only of the nodes appearing after it in this ordering. This constrains the DAG to be a subgraph of the fully connected DAG having such a topological order. Then, the graph is pruned in order to remove spurious edges, e.g., by using sparse regression (Bühlmann et al., 2014). While the first step still requires to solve a combinatorial problem, the set of permutations is much smaller than the set of DAGs. Moreover, once a topological order is fixed, the acyclicity constraint is naturally enforced, making the pruning step easier to solve.

The algorithm that we propose is an order-based method, where the topological order is estimated based on an approximation of the *score* of the data distribution. The score of a distribution with a differentiable probability density $p(x)$ is defined as the map $\nabla \log p(x)$.[1] We show that for a non-linear additive Gaussian noise model, it is possible to identify leaves of the causal graph by analysing the score of the associated data distribution. By sequentially identifying the leaves of the causal graph, and removing the identified leaf variables, one can obtain a complete topological order, since any reverted sequence of leaves gives a topological order. Classical pruning techniques can then be used in order to obtain the final graph. While the proposed algorithm is designed for additive Gaussian noise models, we show that the main required ingredient for our method to work is the additive structure of the model, rather than the noise type. Hence, we expect similar methods to also be applicable to other types of noise (i.e, non-Gaussian).

In order to approximate the score of the data distribution from a sample, we exploit and extend recent work on score matching and density gradient estimation (Li and Turner, 2017). Score approximation methods from observational data have shown success in general machine learning tasks such as generative (Song and Ermon, 2019) and discriminative models (Zimmer-

[1] The term *score* has been used in the causality literature with a different meaning. Classical works (Chickering, 2002) use this term referring to the objective of an optimization problem yielding the causal structure as solution. In the present work, the term score means $\nabla \log p(x)$ as in the statistics literature (Wilks, 1962).

mann et al., 2021), leading to increased interest in developing scalable and efficient solutions. In particular, score-based generative models have shown state-of-the-art performance for image generation (Song and Ermon, 2019; Song et al., 2020b,a; Song and Ermon, 2020). As much of the prior work on causal discovery approaches has focused on leveraging machine/deep learning (Lachapelle et al., 2019; Zheng et al., 2018; Zhu et al., 2019; Wang et al., 2021) to provide a tractable approximation to an NP-hard problem, our work is especially relevant to bridge the gap between provably identifying the causal structure and leveraging advances in deep generative models to scale to large sample sizes and high dimensions.

Hereafter, we summarize our contributions:

- We start by showing that, in the case of non-linear additive Gaussian noise model, knowing the distribution's score function is sufficient to recover the full causal graph, and we provide a method for doing so. To the best of our knowledge, the link between the score function and the causal graph structure established in Lemmata 45 and 46 is not only useful, but also novel.
- We propose a method for estimating the score's Jacobian over a set of observations, exploiting and extending an existing method based on Stein's identity, which can be of independent interest. This method is then used to design a practical algorithm for estimating the causal topological order.
- We finally evaluate our proposed algorithm on both synthetic and real world data and show competitive results compared to state-of-the-art methods, while being significantly faster (10× faster than CAM (Bühlmann et al., 2014) on 20 nodes graphs and 5× faster than GraN-DAG (Lachapelle et al., 2019) on 50 nodes). We also show that our method is robust to noise misspecification and works well when the additive noise is non-Gaussian.

4.2.2 Related Work

Causal discovery for non-linear additive models. Many algorithms have been proposed in the past few years for the specific problem studied in this work. GraN-DAG (Lachapelle et al., 2019) aims to maximise the likelihood of the observed data under this model, and uses a continous contraint for the acyclicity of the causal graph, proposed in (Zheng et al., 2018), in order to use a continuous optimization method to find a first order stationary point of the problem. CAM (Bühlmann et al., 2014) further assumes that the link functions f_i in (4.2) also have an additive structure. They first estimate a topological order by greedily maximizing the data likelihood, and then prune the DAG using sparse regression techniques.

In the scope of linear additive models, (Ghoshal and Honorio, 2018) first proposed an approach to provably recover, under some hypothesis on the noise variances, the causal graph in polynomial time and sample complexity. Their approach can be seen as an order-based method, where the ordering is estimated by sequentially identifying leaves based on an

estimation of the precision matrix. In spirit, their method is closely related to ours. For instance, if the link functions f_i in (4.2) are all linear, then the score of the joint distribution of X is given by $s(x) = -\Theta x$, where Θ is the precision matrix. Hence, the score's Jacobian, which is used in our algorithm to identify the causal graph, can be seen as a non-linear generalization of the precision matrix, which has shown success for identifying causal relations in linear settings (Loh and Bühlmann, 2014).

While our work focuses on the identifiable non-linear additive Gaussian noise model, other works target more general non-parametric model, but must then rely on different kinds of assumptions such as faithfulness, restricted faithfulness or sparsest Markov representation (Spirtes et al., 2000; Raskutti and Uhler, 2018; Solus et al., 2021). These works apply conditional independence tests, and learn a graph that matches the identified conditional independence relations (Spirtes et al., 2000; Zhang, 2008).

Score estimation. In the scope of generative modelling (Song and Ermon, 2019), the score function is learned by fitting a neural network minimizing the empirical Fisher divergence (Hyvärinen and Dayan, 2005). While performing well in practice, such method is quite computationally expensive and requires tuning of several training parameters.

For our purpose, we chose to instead minimize the kernelized Stein discrepancy, since this approach provides a close form solution, allowing fast estimation at all observations. In practice, such method performs similarly as score matching while being much faster to compute. Asymptotic consistency of the Stein gradient estimator, and its relation to score matching were analyzed in (Barp et al., 2019) and (Zhou et al., 2020).

4.2.3 Preliminaries

Causal discovery for non-linear additive Gaussian noise models

Assume that a random variable $X \in \mathbb{R}^d$ is generated using the following special case of (SEM):

$$X_i = f_i(\mathrm{pa}_i(X)) + \epsilon_i, \tag{4.2}$$

$i = 1, \ldots, d$. The noise variables $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ are jointly independent. The functions f_j are assumed to be twice continuously differentiable and non-linear in every component. That is, if we denote the parents $\mathrm{pa}_j(X)$ of X_j by $X_{k_1}, X_{k_2}, \ldots, X_{k_l}$, then, for all $a = 1, \ldots, l$, the function $f_j(x_{k_1}, \ldots, x_{k_{a-1}}, \cdot, x_{k_{a+1}}, \ldots, x_{k_l})$ is assumed to be nonlinear for some $x_{k_1}, \ldots, x_{k_{a-1}}, x_{k_{a+1}}, \ldots, x_{k_l} \in \mathbb{R}^{l-1}$.

This model is known to be identifiable from observational data (Peters et al., 2014), meaning that it is possible to recover the DAG underlying the generative model (4.2) from the knowledge of the joint probability distribution of X. We aim to identify the causal graph from the score function $\nabla \log p(x)$, which has a one-to-one correspondence with $p(x)$. Hence, any model identifiable from observational data will be identifiable from the knowledge of the data score

function.

Score matching

The goal of score matching is to learn the score function $s(x) \equiv \nabla \log p(x)$ of a distribution with density $p(x)$ given a sample $\{x^k\}_{k=1,\dots,n}$ from p. We present here a method developed by Li and Turner (2017) for estimating the score at the sample points, i.e., approximating $\mathbf{G} \equiv (\nabla \log p(\mathbf{x}^1), \dots, \nabla \log p(\mathbf{x}^n))^T \in \mathbb{R}^{n \times d}$.

This estimator is based on the well known Stein identity (Stein, 1972), which states that for any test function $\mathbf{h} : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that $\lim_{\mathbf{x}\to\infty} \mathbf{h}(\mathbf{x}) p(\mathbf{x}) = 0$, we have

$$\mathbb{E}_p[\mathbf{h}(\mathbf{x}) \nabla \log p(\mathbf{x})^T + \nabla \mathbf{h}(\mathbf{x})] = 0, \tag{4.3}$$

where $\nabla \mathbf{h}(\mathbf{x}) \equiv (\nabla h_1(\mathbf{x}), \dots, \nabla h_{d'}(\mathbf{x}))^T \in \mathbb{R}^{d' \times d}$.

By approximating the expectation in (4.3) with the empirical average, we obtain

$$-\frac{1}{n} \sum_{k=1}^{n} \mathbf{h}(\mathbf{x}^k) \nabla \log p(\mathbf{x}^k)^T + \text{err} = \frac{1}{n} \sum_{k=1}^{n} \nabla \mathbf{h}(\mathbf{x}^k), \tag{4.4}$$

where err is a random error term with mean zero, and which vanishes as $n \to \infty$ almost surely. By denoting $\mathbf{H} = (\mathbf{h}(\mathbf{x}^1), \dots, \mathbf{h}(\mathbf{x}^n)) \in \mathbb{R}^{d' \times n}$ and $\overline{\nabla \mathbf{h}} = \frac{1}{n} \sum_{k=1}^{n} \nabla \mathbf{h}(\mathbf{x}^k)$, equation (4.4) reads

$$-\frac{1}{n} \mathbf{H}\mathbf{G} + \text{err} = \overline{\nabla \mathbf{h}}.$$

Hence, by using ridge regression and using the kernel trick, the Stein gradient estimator is defined as:

$$\hat{\mathbf{G}}^{\text{Stein}} \equiv \arg\min_{\hat{\mathbf{G}}} \| \overline{\nabla \mathbf{h}} + \frac{1}{n} \mathbf{H}\hat{\mathbf{G}} \|_F^2 + \frac{\eta}{n^2} \| \hat{\mathbf{G}} \|_F^2 \tag{4.5}$$

$$= -(\mathbf{K} + \eta \mathbf{I})^{-1} \langle \nabla, \mathbf{K} \rangle, \tag{4.6}$$

where $\mathbf{K} \equiv \mathbf{H}^T\mathbf{H}$, $\mathbf{K}_{ij} = \kappa(\mathbf{x}^i, \mathbf{x}^j) \equiv \mathbf{h}(\mathbf{x}^i)^T \mathbf{h}(\mathbf{x}^j)$, $\langle \nabla, \mathbf{K} \rangle = n\mathbf{H}^T \overline{\nabla \mathbf{h}}$, $\langle \nabla, \mathbf{K} \rangle_{ij} = \sum_{k=1}^{n} \nabla_{x_j^k} \kappa(\mathbf{x}^i, \mathbf{x}^k)$ and $\eta \geq 0$ is a regularization parameter. The estimator (4.6) hence gives an efficient way to estimate the score function at every sample point. It requires the choice of a kernel κ that satisfies Stein's identity, such as the RBF kernel as shown in (Liu et al., 2016).

In the following section, we will exploit and extend this approach in order to obtain estimates of the score's Jacobian over the observations, which will be used in order to estimate a topological order for the causal DAG.

4.2.4 Causal discovery via score matching

We now show how to recover the causal graph from the score function $\nabla \log p(x)$ for a non-linear additive model (4.2). We design our method in the case where the additive noise is Gaussian, and then discuss extensions to other types of noise.

Deduce the causal graph from the score of the data distribution

Suppose that we have access to enough observational data coming from an additive Gaussian noise model (4.2) so that we can accurately approximate the score function of the underlying data distribution. In order to extract information about the graph structure from the score function, let us write it in closed form for a model of the form (4.2). The associated probability distribution is given by

$$\begin{aligned} p(\mathbf{x}) &= \prod_{i=1}^{d} p(x_i|\mathrm{pa}_i(\mathbf{x})) \\ \log p(\mathbf{x}) &= \sum_{i=1}^{d} \log p(x_i|\mathrm{pa}_i(\mathbf{x})) \\ &= -\frac{1}{2}\sum_{i=1}^{d}\left(\frac{x_i - f_i(\mathrm{pa}_i(\mathbf{x}))}{\sigma_i}\right)^2 - \frac{1}{2}\sum_{i=1}^{d}\log(2\pi\sigma_i^2). \end{aligned}$$

Thus, the score function $s(\mathbf{x}) \equiv \nabla \log p(\mathbf{x})$ reads

$$\boxed{s_j(\mathbf{x}) = -\frac{x_j - f_j(\mathrm{pa}_j(\mathbf{x}))}{\sigma_j^2} + \sum_{i\in\mathrm{children}(j)} \frac{\partial f_i}{\partial x_j}(\mathrm{pa}_i(\mathbf{x}))\frac{x_i - f_i(\mathrm{pa}_i(\mathbf{x}))}{\sigma_i^2}.} \tag{4.7}$$

An immediate observation from equation (4.7) is that, if j is a leaf, i.e., $\mathrm{children}(j) = \emptyset$, then $s_j(\mathbf{x}) = -\frac{x_j - f_j(\mathrm{pa}_j(\mathbf{x}))}{\sigma_j^2}$. Since $j \notin \mathrm{pa}_j(\mathbf{x})$, we have that $\frac{\partial s_j(\mathbf{x})}{\partial x_j} = -\frac{1}{\sigma_j^2}$, and hence, it holds that $\mathrm{Var}_X\left(\frac{\partial s_j(X)}{\partial x_j}\right) = 0$. The following Lemma shows that this condition is also sufficient for j to be a leaf, providing a way to *provably identify a leaf of the graph from the knowledge of the Jacobian of the score function.*

Lemma 45. *Let p be the probability density function of a random variable X defined via a non-linear additive Gaussian noise model* (4.2)*, and let $s(x) = \nabla \log p(x)$ be the associated score function. Then, $\forall\, j \in \{1,\dots,d\}$, we have:*

1. *j is a leaf $\Leftrightarrow \forall \boldsymbol{x}$, $\frac{\partial s_j(\boldsymbol{x})}{\partial x_j} = c$, with $c \in \mathbb{R}$ independent of $\boldsymbol{x}$, i.e., $Var_X\left[\frac{\partial s_j(X)}{\partial x_j}\right] = 0$.*
2. *If j is a leaf, i is a parent of $j \Leftrightarrow s_j(\boldsymbol{x})$ depends on x_i, i.e., $Var_X\left[\frac{\partial s_j(X)}{\partial x_i}\right] \neq 0$.*

Proof. (1) Equation (4.7) implies the "⇒" direction as described above.

We prove the other direction by contradiction. Suppose that j is not a leaf and that $\frac{\partial s_j(\mathbf{x})}{\partial x_j} = c$ $\forall x$. We can thus write:

$$s_j(\mathbf{x}) = cx_j + g(\mathbf{x}_{-j}),$$

where $g(\mathbf{x}_{-j})$ can depend on any variable but x_j. By plugging equation (4.7) in s_j, we get

$$\frac{f_j(\mathrm{pa}_j(\mathbf{x}))}{\sigma_j^2} + \sum_{i \in \mathrm{children}(j)} \frac{\partial f_i}{\partial x_j}(\mathrm{pa}_i(\mathbf{x})) \frac{x_i - f_i(\mathrm{pa}_i(\mathbf{x}))}{\sigma_i^2} = \left(c + \frac{1}{\sigma_j^2}\right) x_j + g(\mathbf{x}_{-j}).$$

Let i_c be a child of node j such that $\forall i \in \mathrm{children}(j)$, $i_c \notin \mathrm{pa}_i$. Such a node always exist since j is not a leaf, and it suffices to pick a child of j appearing at last in some topological order. We then have

$$\begin{aligned} \frac{\partial f_{i_c}}{\partial x_j}(\mathrm{pa}_{i_c}(\mathbf{x})) \frac{x_{i_c} - f_{i_c}(\mathrm{pa}_{i_c}(\mathbf{x}))}{\sigma_{i_c}^2} - g(\mathbf{x}_{-j}) = \left(c + \frac{1}{\sigma_j^2}\right) x_j - \frac{f_j(\mathrm{pa}_j(\mathbf{x}))}{\sigma_j^2} \\ - \sum_{i \in \mathrm{children}(j), i \neq i_c} \frac{\partial f_i}{\partial x_j}(\mathrm{pa}_i(x)) \frac{x_i - f_i(\mathrm{pa}_i(\mathbf{x}))}{\sigma_i^2}. \end{aligned} \tag{4.8}$$

Now, due to the specific choice of i_c, we have that the RHS of (4.8) does not depend on x_{i_c} (note that we are here speaking about functional dependence on variables, not statistical dependence on a random variable). Hence, we have

$$\frac{\partial}{\partial x_{i_c}} \left(\frac{\partial f_{i_c}}{\partial x_j}(\mathrm{pa}_{i_c}(\mathbf{x})) \frac{x_{i_c} - f_{i_c}(\mathrm{pa}_{i_c}(\mathbf{x}))}{\sigma_{i_c}^2} - g(\mathbf{x}_{-j}) \right) = 0 \quad \Rightarrow \quad \frac{\partial f_{i_c}}{\partial x_j} = \sigma_{i_c}^2 \frac{\partial g(\mathbf{x}_{-j})}{\partial x_{i_c}}.$$

Since g does not depend on x_j, this means that $\frac{\partial f_{i_c}}{\partial x_j}$ does not depend on x_j neither, implying that f_{i_c} is linear in x_j, contradicting the non-linearity assumption.

(2) If j is a leaf, then, by equation (4.7), we have

$$s_j(\mathbf{x}) = -\frac{x_j - f_j(\mathrm{pa}_j(\mathbf{x}))}{\sigma_j^2}.$$

If i is not a parent of j, then $\frac{\partial s_j}{\partial x_i} \equiv 0$, and hence we have $\mathrm{Var}_X \left[\frac{\partial s_j(X)}{\partial x_i} \right] = 0$. On the other hand, if i is a parent of j, then we have $\frac{\partial s_j}{\partial x_i}(x) = \frac{1}{\sigma_j^2} \frac{\partial f_j}{\partial x_i}(\mathrm{pa}_j(\mathbf{x}))$. Moreover, since f_j cannot be linear in x_i, $\frac{\partial f_j}{\partial x_i}(\mathrm{pa}_j(\mathbf{x}))$ cannot be a constant, and hence $\mathrm{Var}_X \left[\frac{\partial s_j(X)}{\partial x_i} \right] \neq 0$.

□

Lemma 45 shows that, for non-linear additive Gaussian noise models, leaf nodes (and only leaf nodes) have the property that the associated diagonal element in the score's Jacobian is a

constant. This hence provides a way to identify a leaf of the causal graph from the knowledge of the variance of the score's Jacobian diagonal elements. By repeating this method and always removing the identified leaves, we can estimate a full topological order. This procedure is summarized in Algorithm 12. In the following section, we present a new approach, exploiting Stein identities, to compute estimates of the score's Jacobian over a set of samples.

Note that the use of empirical variance to identify identically 0 function $\frac{\partial s_j}{\partial x_j}$ is not necessary. However, we did not find any empirical benefit when using other deviation measures, such as the average distance to the median for example.

DAG pruning. Once a topological order is estimated, the DAG becomes constrained to be a sub-graph of a certain fully connected DAG. However, it is necessary to prune this fully connected DAG to remove spurious edges. In theory, it would be possible to make use of the learnt score for this purpose, by using property (2) of Lemma 45. However, more classical methods such as CAM appears to perform better in practice. The idea behind CAM is to assume that the link functions f_i in (4.2) have an additive structure. We then fit a generalized additive model (Hastie and Tibshirani, 1987) on each component and use hypothesis testing for additive models (Marra and Wood, 2011) to decide upon existence of edges. For further details about this pruning technique, please refer to the original paper (Bühlmann et al., 2014).

Algorithm 12 SCORE-matching causal order search

Input: Data matrix $X \in \mathbb{R}^{n \times d}$.
Initialize $\pi = []$, nodes $= \{1, \dots, d\}$
for $k = 1, \dots, d$ **do**
    Estimate the diagonal part of the Jacobian of the score function $s_{nodes} = \nabla \log p_{nodes}$ (e.g., using Algorithm 13).
    Estimate $V_j = \mathrm{Var}_{X_{nodes}}\left[\frac{\partial s_j(X)}{\partial x_j}\right]$.
    $l \leftarrow$ nodes$[\arg\min_j V_j]$
    $\pi \leftarrow [l, \pi]$
    nodes $\leftarrow$ nodes $- \{l\}$
    Remove l-th column of X
Get the final DAG by pruning the full DAG associated with the topological order π.

Approximation of the score's Jacobian

The Stein gradient estimator $\hat{\mathbf{G}}^{\text{Stein}}$ enables us to estimate the score function point-wise at each of our sample points. However, in order to implement Algorithm 12, we need an estimate of the Jacobian of the score at all samples, in order to estimate its variance. Since we do not have a functional approximation of the score, we cannot use tricks such as auto-differentiation in order to obtain higher order derivative approximations. In this section, we extend the idea of Stein based estimator to obtain estimates for the score's Jacobian.

For this purpose, we will use the second-order Stein identity (Diaconis et al., 2004; Zhu, 2021).

Assuming that p is twice differentiable, for any $q : \mathbb{R}^d \to \mathbb{R}$ such that $\lim_{\mathbf{x}\to\infty} q(\mathbf{x})p(\mathbf{x}) = 0$ and such that $\mathbb{E}[\nabla^2 q(\mathbf{x})]$ exists, the second-order Stein identity states that

$$\mathbb{E}[q(\mathbf{x})p(\mathbf{x})^{-1}\nabla^2 p(\mathbf{x})] = \mathbb{E}[\nabla^2 q(\mathbf{x})], \tag{4.9}$$

which can be rewritten as

$$\mathbb{E}[q(\mathbf{x})\nabla^2 \log p(\mathbf{x})] = \mathbb{E}[\nabla^2 q(\mathbf{x}) - q(\mathbf{x})\nabla \log p(\mathbf{x})\nabla \log p(\mathbf{x})^T]. \tag{4.10}$$

Recall that, in order to identify a leaf of the causal graph, we are only interested in estimating the diagonal elements of the score's Jacobian at the sample points, i.e.,
$J \equiv (\mathrm{diag}(\nabla^2 \log p(\mathbf{x}^1)), \ldots, \mathrm{diag}(\nabla^2 \log p(\mathbf{x}^n)))^T \in \mathbb{R}^{n\times d}$. Using the diagonal part of the matrix equation (4.10) for various test functions gathered in $\mathbf{h} : \mathbb{R}^d \to \mathbb{R}^{d'}$, we can write

$$\mathbb{E}[\mathbf{h}(\mathbf{x})\mathrm{diag}(\nabla^2 \log p(\mathbf{x}))^T] = \mathbb{E}[\nabla^2_{\mathrm{diag}}\mathbf{h}(\mathbf{x}) - \mathbf{h}(\mathbf{x})\mathrm{diag}((\nabla \log p(\mathbf{x})\nabla \log p(\mathbf{x})^T))], \tag{4.11}$$

where $(\nabla^2_{\mathrm{diag}}\mathbf{h}(\mathbf{x}))_{ij} = \frac{\partial^2 h_i(\mathbf{x})}{\partial x_j^2}$. By approximating the expectations by empirical averages, we obtain, similarly as in (4.4),

$$\frac{1}{n}\sum_{k=1}^{n} \mathbf{h}(\mathbf{x}^k)\mathrm{diag}(\nabla^2 \log p(\mathbf{x}^k))^T + \mathrm{err} = \frac{1}{n}\sum_{k=1}^{n}\left(\nabla^2_{\mathrm{diag}}\mathbf{h}(\mathbf{x}^k) - \mathbf{h}(\mathbf{x}^k)\mathrm{diag}\left(\nabla \log p(\mathbf{x}^k)\nabla \log p(\mathbf{x}^k)^T\right)\right) \tag{4.12}$$

with $\mathrm{err} \overset{n\to\infty}{\longrightarrow} 0$ almost surely.

By denoting $\mathbf{H} = (\mathbf{h}(\mathbf{x}^1), \ldots, \mathbf{h}(\mathbf{x}^n)) \in \mathbb{R}^{d'\times n}$ and $\overline{\nabla^2_{\mathrm{diag}}\mathbf{h}} \equiv \frac{1}{n}\sum_{k=1}^{n} \nabla^2_{\mathrm{diag}}\mathbf{h}(\mathbf{x}^k)$, equation (4.12) reads

$$\frac{1}{n}\mathbf{H}\mathbf{J} + \mathrm{err} = \overline{\nabla^2_{\mathrm{diag}}\mathbf{h}} - \frac{1}{n}\mathbf{H}\mathrm{diag}(\mathbf{G}\mathbf{G}^T).$$

By using the Stein gradient estimator for $\mathbf{G}$, we define the Stein Hessian estimator as the ridge regression solution of the previous equation, i.e.,

$$\begin{aligned}\hat{\mathbf{J}}^{\mathrm{Stein}} &\equiv \arg\min_{\hat{\mathbf{J}}} \left\| \frac{1}{n}\mathbf{H}\hat{\mathbf{J}} + \frac{1}{n}\mathbf{H}\mathrm{diag}\left(\hat{\mathbf{G}}^{\mathrm{Stein}}\left(\hat{\mathbf{G}}^{\mathrm{Stein}}\right)^T\right) - \overline{\nabla^2_{\mathrm{diag}}\mathbf{h}}\right\|_F^2 + \frac{\eta}{n^2}\|\hat{\mathbf{J}}\|_F^2 \\ &= -\mathrm{diag}\left(\hat{\mathbf{G}}^{\mathrm{Stein}}\left(\hat{\mathbf{G}}^{\mathrm{Stein}}\right)^T\right) + (\mathbf{K} + \eta\mathbf{I})^{-1}\langle \nabla^2_{\mathrm{diag}}, \mathbf{K}\rangle, \end{aligned} \tag{4.13}$$

where $\mathbf{K}_{ij} = \kappa(\mathbf{x}^i, \mathbf{x}^j) \equiv \mathbf{h}(\mathbf{x}^i)^T\mathbf{h}(\mathbf{x}^j)$, $\langle \nabla^2_{\mathrm{diag}}, \mathbf{K}\rangle = n\mathbf{H}^T\overline{\nabla^2_{\mathrm{diag}}\mathbf{h}}$, $\langle \nabla^2_{\mathrm{diag}}, \mathbf{K}\rangle_{ij} = \sum_{i=1}^{n} \frac{\partial^2 \kappa(\mathbf{x}^i, \mathbf{x}^k)}{\partial (x_j^k)^2}$ and $\hat{\mathbf{G}}^{\mathrm{Stein}}$ is defined in (4.6). The regularization parameter η lifts the eigenvalues of the same

matrix $\mathbf{K}$ as in the Stein gradient estimator $\hat{\mathbf{G}}^{\text{Stein}}$. We hence decide to use the same parameter for both ridge regression problems.

Choice of kernel. Estimating the score's Jacobian with the method above requires a choice of kernel κ. A widely used kernel is the RBF kernel $\kappa_s(\mathbf{x},\mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2s^2}}$, which has one parameter s called the bandwidth. This parameter can be estimated from the data to be fitted, using the commonly used median heuristic, i.e., choosing s to be the median of the pairwise distances between vectors in X. This bandwidth estimation procedure even enjoys theoretical convergence properties (Garreau et al., 2017). Note that, when using Algorithm 13 for causal discovery in Algorithm 12, the kernel bandwidth is re-computed each time a node is removed from the data matrix X.

Algorithm 13 Estimating the Jacobian of the score

1: Input: Data matrix $X \in \mathbb{R}^{n\times d}$, regularisation parameter $\eta > 0$.
2: $s \leftarrow \text{median}(\{\|x_i - x_j\|_2 : i, j = 1,\dots,n, x_k = X[k,:]\})$.
3: Compute $\hat{\mathbf{J}}^{\text{Stein}}$ using RBF kernel κ_s, regularisation parameter η and data matrix X based on (4.13).

Algorithm complexity. Estimating the topological order requires inverting d times an $n \times n$ kernel matrix, hence the complexity is $\mathcal{O}(dn^3)$ (but could be improved using, e.g., Strassen's algorithm (Strassen, 1969)). Including the pruning step, the final complexity is hence $\mathcal{O}(dn^3 + dr(n,d))$ where $r(n,d)$ is the complexity of fitting a generalized additive model (Hastie and Tibshirani, 1987) using n data points in d dimensions. In comparison, the complexity of CAM is $\mathcal{O}(d^2 r(n,d))$. The total computational complexity of GraNDAG is not discussed in (Lachapelle et al., 2019); it is difficult to specify it since it depends on the number of iterations used in the Augmented Lagrangian method, which may depend on the dimension and number of samples. However, GraNDAG is particularly slow due the computation of the acyclicity constraint at each iteration, which requires computing the exponential of a $d \times d$ matrix, taking $\mathcal{O}(d^3)$ operations.

In practice, in our method, the time for estimating the topological order is much smaller than the time for pruning it (30% of the total time for $(d,n) = (20,1000)$ and 5% of the total time for $(d,n) = (50,1000)$). In comparison, CAM spends most of the time estimating the topological order (more than 95% of the total time in all tested scenari). Hence, we expect the dominant term in SCORE's time complexity to be $dr(n,d)$, thus improving upon CAM's complexity by a factor of d. Moreover, in the case where n becomes very large, it is possible to use kernel approximation methods to reduce the time complexity of our method (Si et al., 2014).

Extension to non-Gaussian additive noise models

In the previous section, we exploited the structure of the additive Gaussian noise model to deduce the causal graph from the score function (4.2). Actually, the main ingredient required in our analysis is the additive structure. Indeed, for any additive noise model (including

non-Gaussian noise), the score function has a similar structure as in (4.7).

Lemma 46. *Suppose that the random variable X is generated from* (4.2) *where the noise variables ϵ_i are i.i.d. with smooth probability distribution function p^ϵ. Then, the score function s of X is given by*

$$s_j(\boldsymbol{x}) = \frac{d\log p^\epsilon}{dx}(x_j - f_j(pa_j(\boldsymbol{x}))) - \sum_{i\in children(j)} \frac{\partial f_i}{\partial x_j}(pa_i(\boldsymbol{x}))\frac{d\log p^\epsilon}{dx}(x_i - f_i(pa_i(\boldsymbol{x}))). \quad (4.14)$$

Proof. The proof follows exactly the same lines as for showing the score decomposition (4.7) for the Gaussian noise model. □

The decomposition of the score's components j into a common term $\frac{d\log p^\epsilon}{dx}(x_j - f_j(\mathrm{pa}_j(\mathbf{x})))$ and a term involving only the parents of the node j is hence characteristic of additive noise models. Recall that our method identifies leaves by identifying non-linearity in the components of the score. When the common term is linear in x_j, as it is the case with Gaussian noise, the second term is the only one carrying non-linearities, and the leaves can hence be perfectly identified with this method (see Lemma 45). However, intuitively speaking, even when the noise is non-Gaussian, i.e., when the common term carries non-linearities, the second term still carries non-linearities proportionally to the number of parents of node j. Hence, we may expect that the proposed algorithm can work in the more general case of additive models, even when the noise is non-Gaussian. While this does not provide a formal identifiability statement, we will show in the experimental section that SCORE outperforms other state-of-the-art algorithms on non-Gaussian additive models.

4.2.5 Experiments

We now apply Algorithm 12 with Algorithm 13 as score estimator to synthetic and real-world datasets and compare its performance to state-of-the-art methods, such as CAM (Bühlmann et al., 2014), GraNDAG (Lachapelle et al., 2019), SELF (Cai et al., 2018) and GES (Chickering, 2002). Some other methods such as NOTEARS, PC or FCI are omitted since they perform much worse (Bühlmann et al., 2014; Lachapelle et al., 2019).

Recent work (Reisach et al., 2021) warned about the fact that simulated data sometimes lead to scenari where a topological order can simply be estimated by sorting the nodes variances. In order to defend ourselves against this, we randomly generate the noise variances in the generative model, and show that the estimated order when sorting the variance is much worse than the one estimated by Algorithm 12. The code can be found in https://github.com/paulrolland1307/SCORE/.

2D toy example

Before we present the results on high dimensional data, let us first analyze how SCORE is applied to a 2D toy problem. Consider the following simple example:

$$\begin{aligned} X_1 &\leftarrow \xi_1 \\ X_2 &\leftarrow f(X_1) + \xi_2, \end{aligned} \tag{4.15}$$

where $\xi_1, \xi_2 \overset{i.i.d}{\sim} \mathcal{N}(0,1)$ and $f(x) = \sin(x)$. The ground truth causal graph is hence $X_1 \to X_2$. We sample 1000 data points from this generative model, and obtain the dataset shown in Figure (4.2). Since the model is given analytically, we can compute the score function and its Jacobian exactly:

$$\begin{aligned} s(\mathbf{x}) &= \begin{pmatrix} -x_1 + \cos(x_1)(x_2 - \sin(x_1)) \\ -x_2 + \sin(x_1) \end{pmatrix} \\ \nabla s(\mathbf{x}) &= \begin{pmatrix} -1 - \sin(x_1)(x_2 - \sin(x_1)) + \cos^2(x_1) & \cos(x_1) \\ -x_2 + \sin(x_1) & -1 \end{pmatrix} \end{aligned}$$

Notice the value -1 on the lower-right part of the Jacobian ∇s. This means that $s_2(\mathbf{x})$ is linear in x_2, which is a sign of X_2 being a leaf that we want to identify in SCORE.

We can then estimate the score function from the data, using our Stein estimator, or other ones. The estimator on the right of Figure (4.2) is actually obtained using the Score Matching method of Hyvärinen and Dayan (2005), since it also provides out-of-samples estimates and hence allows to nicely plot the estimator over the full domain.

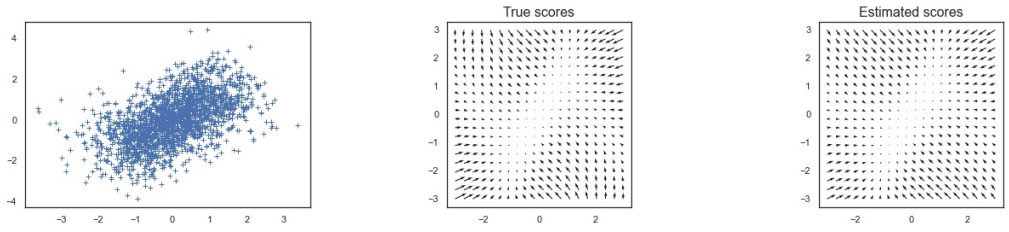


Figure 4.2 – Left: 1000 samples from generative model (4.15). Middle: True data distribution score function s. Right: Estimated score from the data $\tilde{s}$.

We can then compute $\mathrm{Var}(\nabla s(X))$ both theoretically and empirically:

$$\mathrm{Var}(\nabla s(X)) = \begin{pmatrix} 0.54 & 0.22 \\ 0.22 & 0 \end{pmatrix} \qquad \tilde{\mathrm{Var}}(\nabla \tilde{s}(X)) = \begin{pmatrix} 0.57 & 0.26 \\ 0.26 & 0.07 \end{pmatrix} \tag{4.16}$$

where $\tilde{\text{Var}}$ denotes the empirical estimator for the variance. While the estimated value for $\text{Var}\left(\frac{\partial s_2(\mathbf{x})}{\partial x_2}\right)$ is not exactly 0, it is still much smaller that $\text{Var}\left(\frac{\partial s_1(\mathbf{x})}{\partial x_1}\right)$. Hence, SCORE picks X_2 as the first leaf, and thus outputs $(1,2)$ as topological order. Then, applying CAM pruning procedure, we fit a generalized additive model to predict X_2 from X_1, and we find that X_1 is a parent of X_2. We hence recover the correct causal graph $X_1 \to X_2$.

Synthetic data

We test our algorithm on synthetic data generated from a non-linear additive Gaussian noise model (4.2). Mimicking (Lachapelle et al., 2019; Zhu et al., 2019), we generate the link functions f_i by sampling Gaussian processes with a unit bandwidth RBF kernel. The noise variances σ_i^2 are independently sampled uniformly in $[0.4, 0.8]$. The causal graph is generated using the Erdös-Rényi model (Erdös and Rényi, 2011). For a fixed number of nodes d, we vary the sparsity of the sampled graph by setting the average number of edges to be either d (ER1) or $4d$ (ER4). Moreover, to test the robustness of the algorithm against noise type misspecification, we also generate data with Laplace noise instead of Gaussian noise. Additional experiments, using Gumbel noise and scale free graphs (Barabási and Albert, 1999) can be found in Appendix C.1.

For each method, we compute the structural Hamming distance (SHD) between the output and the true causal graph, which counts the number of missing, falsely detected or reversed edges, as well as the structural intervention distance (SID) (Peters and Bühlmann, 2015) which counts the number of interventional distribution which would be miscalculated using the chosen causal graph.

For all order-based causal discovery methods, we always apply the same pruning procedure, i.e., CAM with the same cutoff parameter of 0.001. Moreover, we compute a quantity measuring how well the topological order is estimated. For an ordering π, and a target adjacency matrix A, we define the topological order divergence $D_{top}(\pi, A)$ as

$$D_{top}(\pi, A) = \sum_{i=1}^{d} \sum_{j:\pi_i > \pi_j} A_{ij}.$$

If π is a correct topological order for A, then $D_{top}(\pi, A) = 0$. Otherwise, $D_{top}(\pi, A)$ counts the number of edges that cannot be recovered due to the choice of topological order. It hence provides a lower bound on the SHD of the final algorithm (irrespective of the pruning method). The results of the synthetic experiments are shown in Tables 4.1 to 4.6. The computed quantities are averages over 10 independent runs.

We can see that, for sparser graphs (ER1), our method performs similarly as the best method CAM. However, for denser graphs (ER4), our method performs better, and in particular seems to estimate a better topological order, since the D_{top} value is smaller. For 50 nodes graphs, the two best methods are CAM and ours, which both perform similarly. Note that, in order to run it within a reasonable time frame, we had to restrict the maximum number of neighbours,

hence providing a sparsity prior to the algorithm, which fits the correct graph in this situation, since sparse Erdös-Renyi graphs usually do not contain high degree nodes. Since we restricted the number of neighbours in the graph, the order search in CAM does not yield a single topological order, hence we could not compute D_{top} in this setting. We also observe that the topological ordering resulting from sorting the variances (VarSort) is much worse than using the order-based methods (either CAM or SCORE), showing that finding a topological order for the generated datasets is not a trivial task. Finally, we observe that our method is quite robust to noise misspecification, since the accuracy remains very similar for Laplace noise.

In terms of running time (Table 4.7), we see that our method is significantly faster compared to the other competitive algorithms CAM and GraN-DAG. Actually, in SCORE, most of the time (95% for $d = 50$) is spent on pruning the final DAG.

Table 4.1 – Synthetic experiment for $d = 10$ with Gaussian noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	$\mathbf{1.1 \pm 0.9}$	$\mathbf{4.5 \pm 5.3}$	$\mathbf{0.4 \pm 0.6}$	$\mathbf{19.5 \pm 2.9}$	$\mathbf{35.0 \pm 9.1}$	$\mathbf{0.3 \pm 0.3}$
CAM	1.7 ± 1.0	6.4 ± 4.2	$\mathbf{0.4 \pm 0.5}$	24.4 ± 3.1	45.2 ± 10.2	4.4 ± 3.2
GraN-DAG	1.5 ± 1.4	6.5 ± 7.2	–	22.2 ± 2.6	42.0 ± 6.2	–
SELF	8.4 ± 1.6	32.5 ± 7.6	–	37.2 ± 2.1	83.0 ± 5.2	–
GES	7.8 ± 2.7	32.5 ± 13.6	–	34.3 ± 3.0	78.9 ± 6.0	–
VarSort	–	–	1.9 ± 1.1	–	–	9.7 ± 3.1

Table 4.2 – Synthetic experiment for $d = 20$ with Gaussian noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	$\mathbf{2.6 \pm 1.9}$	$\mathbf{9.9 \pm 8.5}$	1.2 ± 1.7	$\mathbf{47.5 \pm 4.5}$	$\mathbf{177.5 \pm 11.6}$	$\mathbf{3.1 \pm 1.5}$
CAM	3.5 ± 1.6	14.3 ± 9.8	$\mathbf{0.8 \pm 1.0}$	54.2 ± 5.4	201.9 ± 29.0	13.6 ± 6.9
GraN-DAG	7.6 ± 4.2	31.6 ± 22.7	–	49.3 ± 4.5	211.4 ± 36.6	–
SELF	16.6 ± 2.1	89.9 ± 31.2	–	75.5 ± 1.6	336.8 ± 31.2	–
GES	17.7 ± 3.8	77.3 ± 30.5	–	67.4 ± 6.1	322.9 ± 21.7	–
VarSort	–	–	3.7 ± 1.6	–	–	18.3 ± 6.7

Table 4.3 – Synthetic experiment for $d = 50$ with Gaussian noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	10.4 ± 3.9	$\mathbf{50.9 \pm 32.9}$	3.9 ± 2.4	$\mathbf{131.5 \pm 7.5}$	$\mathbf{1262 \pm 110}$	16.3 ± 6.1
CAM	$\mathbf{8.3 \pm 2.9}$	53.7 ± 31.9	–	140.8 ± 5.5	1337 ± 94	–
GraN-DAG	20.2 ± 6.1	135.3 ± 45.9	–	140.8 ± 9.5	1432 ± 110	–
SELF	45.4 ± 3.5	326.6 ± 74.3	–	192.7 ± 3.2	2097 ± 103	–
GES	50.5 ± 4.2	233.5 ± 60.8	–	182.9 ± 7.3	2003 ± 105	–
VarSort	–	–	8.8 ± 3.0	–	–	43.3 ± 9.7

Table 4.4 – Synthetic experiment for $d = 10$ with Laplace noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	1.4 ± 0.8	4.5 ± 4.7	0.8 ± 0.7	$\mathbf{19.6 \pm 2.5}$	$\mathbf{31.9 \pm 7.9}$	$\mathbf{0.2 \pm 0.4}$
CAM	1.5 ± 1.3	6.1 ± 6.5	$\mathbf{0.5 \pm 0.5}$	24.4 ± 1.5	44.4 ± 8.1	1.5 ± 1.6
GraN-DAG	$\mathbf{1.3 \pm 1.4}$	$\mathbf{4.4 \pm 4.9}$	–	20.3 ± 2.7	39.3 ± 13.0	–
SELF	9.7 ± 2.5	33.4 ± 10.8	–	38.2 ± 1.8	86.9 ± 4.3	–
GES	8.9 ± 2.2	28.3 ± 12.0	–	33.7 ± 2.3	78.9 ± 7.4	–
VarSort	–	–	1.6 ± 1.3	–	–	7.2 ± 2.3

Table 4.5 – Synthetic experiment for $d = 20$ with Laplace noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	$\mathbf{1.6 \pm 1.2}$	$\mathbf{6.8 \pm 11.4}$	0.5 ± 0.9	$\mathbf{48.0 \pm 4.0}$	199.8 ± 21.4	$\mathbf{4.9 \pm 1.8}$
CAM	2.3 ± 1.4	10.0 ± 7.0	$\mathbf{0.3 \pm 0.5}$	52.4 ± 3.9	208.7 ± 17.5	11.6 ± 7.9
GraN-DAG	4.9 ± 2.1	27.5 ± 13.2	–	48.2 ± 3.8	$\mathbf{198.3 \pm 42.8}$	–
SELF	16.4 ± 3.6	87.5 ± 32.3	–	77.4 ± 2.2	349.5 ± 19.0	–
GES	17.7 ± 6.8	72.6 ± 25.5	–	69.7 ± 7.1	325.5 ± 28.3	–
VarSort	–	–	3.4 ± 2.0	–	–	20.8 ± 4.5

Table 4.6 – Synthetic experiment for $d = 50$ with Laplace noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	11.0 ± 4.5	71.8 ± 50.2	4.0 ± 2.5	$\mathbf{128.1 \pm 7.9}$	1384 ± 131	19.8 ± 3.5
CAM	$\mathbf{10.1 \pm 3.4}$	$\mathbf{66.1 \pm 47.9}$	–	134.6 ± 7.2	$\mathbf{1361 \pm 136}$	–
GraN-DAG	21.9 ± 3.9	165.7 ± 46.2	–	138.3 ± 8.8	1603 ± 166	–
SELF	42.4 ± 2.9	361.4 ± 112.5	–	191.4 ± 2.9	2053 ± 110	–
GES	52.4 ± 7.7	292.2 ± 105.9	–	182.5 ± 7.2	2028 ± 120	–
VarSort	–	–	8.1 ± 4.2	–	–	47.3 ± 8.7

Table 4.7 – Run time (in seconds) comparison of the algorithms on ER1. The first row corresponds to the time spent for finding the topological order in our method. $(*)$ In order to run CAM on 50 nodes within a reasonable time, we had to use preliminary neighbour search while restricting the maximum number of neighbours to 20 (Bühlmann et al., 2014).

	$d = 10$	$d = 20$	$d = 50$
SCORE order	3.3 ± 0.1	8.5 ± 0.8	31 ± 2.9
SCORE	6.3 ± 0.2	32.7 ± 6.7	257 ± 17
CAM	30.1 ± 3.7	313 ± 80	$1143 \pm 79^{(*)}$
GraN-DAG	185 ± 26	357 ± 47	1410 ± 73

Real data

We now compare the algorithms on a popular real-world dataset for causal discovery (Sachs et al., 2005) (11 nodes, 17 edges and 853 observations), as well as the pseudo-real dataset

sampled from SynTReN generator (Van den Bulcke et al., 2006) (Table 4.8). We can see that on Sachs, our method matches the SHD of CAM while improving the SID. On the SynTReN datasets, GraN-DAG seems to perform best, although the confidence intervals overlap.

Table 4.8 – Comparison of several algorithms on the real world dataset Sachs and 10 datasets sampled from SynTReN.

	Sachs		SynTReN	
	SHD	SID	SHD	SID
SCORE	**12**	**45**	36.2 ± 4.7	193.4 ± 60.2
CAM	**12**	55	40.5 ± 6.8	$\mathbf{152.3 \pm 48.0}$
GraN-DAG	13	47	$\mathbf{34.0 \pm 8.5}$	161.7 ± 53.4

5 Conclusion and future work

5.1 Summary of the thesis

In this dissertation, we focused on the training of robust models, i.e., models whose performance do not drastically degrade when applied to environments differing from the one they where trained in. The choice of method for a given task mainly depends on the two following aspects:

- The desired robustness property: Robustness generally comes with a price on the accuracy on the trained environment. This price usually increases with the number of environments we wish to generalize to. Hence, it is important to wisely choose the space $\mathscr{P}$ of test environments so that the resulting model still performs well in the training environment.
- The available information to us for training. In addition to training data, we saw that additional elements, such as knowledge of the causal structure, or access to different environments or adversary, can help us in the training of robust models.

In each chapter, we study a specific robustness setup, and provide a method for solving the resulting optimization problem (ROB).

Chapter 2 The first setup we study is the case where we have access to samples from the distributions in $\mathscr{P}$. In this setting, we propose to solve the min-max problem (ROB) directly using a recent method for finding the mixed Nash equilibrium. This method relies on iterative sampling steps that are implemented using the Unadjusted Langevin Algorithm, and we provide an analysis of ULA in the log-concave scenario. We thus apply this method to the training of a Reinforcement Learning agent, where the access to different test environments is mimicked by introducing an adversarial agent perturbing the learner's actions.

Chapter 3 We then consider the setup of adversarial robustness, i.e., we wish to train models that are robust to any small change of the input. To this end, we identify a quantity, namely the

Lipschitz constant of the model, which quantifies how robust the model is to any adversarial perturbation. We hence propose a method to upper bound this quantity for neural networks, leading to a certificate for adversarial robustness. Then, for the purpose of training adversarially robust neural networks, we propose to regularize the network using its 1-path-norm, serving as a proxy for its Lipschitz constant. In order to solve the resulting regularized problem, we provide a method for computing (or estimating in the case of deep networks) the proximal mapping of the 1-path-norm.

Chapter 4 Finally, we motivate the use of causal features for training robust models. Indeed, it has been shown that models using causal features only are robust to changes of environments sharing the same causal structure and causal mechanisms. Therefore, identifying the causal relations between the variables at hand seems of great importance for developing robust models. We hence present an algorithm for identifying causal graphs in additive non-linear Gaussian noise models from observational data, based on the approximation of the score of the data distribution.

5.2 Directions for future work

5.2.1 Further analysis of DL-ULA

In DL-ULA (Algorithm 3), we use a projection step in order to control the sub-exponentiality constant. It is an interesting question whether this step is actually required. What is needed in the proof is a bound on the tail of the iterate distributions as in (2.11), so that we can apply Lemma 9 as done in equation (2.21). Such a tail property can be obtained by bounding the Poincaré constant of the iterate distributions. However, controlling the Poincaré constant of the ULA iterates in the non-strongly log-concave setting appears to remain an open problem.

Moreover, DL-ULA mainly differs form the classical ULA by its choice of a multistage step size decay. In particular, the step size in DL-ULA for unconstrained sampling (Algorithm 3) decays as $\mathcal{O}(t^{-2/3})$, i.e., more rapidly than $\mathcal{O}(t^{-1/2})$ as commonly used in this setting. By applying a similar analysis as Durmus et al. (2018a) to this new choice of step-size decay, we obtained improved convergence guarantees compared to the same analysis using $\mathcal{O}(t^{-1/2})$ step size decay. An interesting future direction is hence to extend other analyses of ULA using this modified step size schedule to see whether we can obtain similar improvements.

5.2.2 Analysis of stochastic prox method for 1-path norm regularization

In the single hidden layer case, we developed an efficient method for computing the exact proximal operator of the 1-path norm. However, in the case of deep networks, it seems that computing the exact proximal mapping of the full 1-path norm is quite hard, and we instead propose to compute the exact proximal mapping of an unbiased stochastic estimator of the 1-path norm. However, while intuitive, this method is not theoretically grounded, and a

convergence analysis of such a method is still missing.

5.2.3 Extension of SCORE to other identifiable models

In theory, for any identifiable model, it is possible to recover the causal graph from the knowledge of the data score function. In the case of non-linear additive Gaussian noise models, we identified a simple condition to *read* the causal graph (or at least informations about the leaves) from the score function (Lemma 45). In order to extend this approach to other models, we need to design ways to link the score function to the causal graph associated with the studied model. In particular, for additive non-Gaussian noise models, the score decomposes in a similar way as for the additive Gaussian noise model (Lemma 46). Hence, if we know the noise type, we should be able to find conditions for identifying the leaves, similarly as in SCORE. Other extensions should include the use of interventional data, and deal with the presence of hidden confounders, i.e., unobserved causal variables.

The main novelty of this work is that it links causal discovery with score estimation, which became very popular recently due to the emergence and the great performance of score-based generative models. Hence, several methods have been developed, and continue to be improved, for estimating the score function of a distribution from data. The most used algorithm for this task is *Score Matching*, together with several variants, which showed great performance, even for very high dimensional data, e.g., high resolution images. We expect this method to outperform our Stein estimator, especially in very high dimensions, where kernel methods usually perform rather poorly. However, naively applying Score Matching to our problem requires training d different neural networks, which would drastically affect the scalability of the method. Nonetheless, the fitting problems to be solved are not completely independent, since the score function of a distribution after removing a variable (a leaf in the case of SCORE), shares some similarities with the score function of the original distributions. Hence, we do not need to retrain a network from scratch after each leaf removal. There is hence hope for applying Score Matching to our causal discovery method in a scalable way.

A Appendix for Chapter 2

A.1 Proofs of Section 2.2

A.1.1 Proof of Lemma 9

Before proving Lemma 9, we first prove some intermediate Lemmata.

Lemma 47. *Let μ, ν be any two distributions. Then, $\forall R > 0$, we have*

$$\begin{aligned} \mathrm{W}_2^2(\mu,\nu) \leq & 4R^2\|\mu-\nu\|_{\mathrm{TV}} + 2\mathbb{E}_{X\sim\mu}\left[\|X\|_2^2 1_{\{\|X\|_2>R\}}\right] + 2R^2\mathbb{E}_{X\sim\mu}\left[1_{\{\|X\|_2>R\}}\right] \\ & + 2\mathbb{E}_{Y\sim\nu}\left[\|Y\|_2^2 1_{\{\|Y\|_2>R\}}\right] + 2R^2\mathbb{E}_{Y\sim\nu}\left[1_{\{\|Y\|_2>R\}}\right], \end{aligned}$$

where $1_{\{\|X\|_2>R\}}$ is the indicator function of the set $B(0,R)^c = \{\boldsymbol{x}\in\mathbb{R}^d : \|\boldsymbol{x}\|_2 > R\}$.

Proof. Let $X\sim\mu, Y\sim\nu$. Recall that the W_2-distance between probability measures μ and ν is defined as

$$\mathrm{W}_2^2(\mu,\nu) = \inf_{\gamma\in\Phi(\mu,\nu)} \mathbb{E}_{(X,Y)\sim\gamma}\|X-Y\|_2^2, \tag{A.1}$$

where the minimization is over all probability measures γ that marginalize to μ,ν, namely,

$$\gamma(A\times\mathbb{R}^d) = \mu(A), \qquad \gamma(\mathbb{R}^d\times B) = \nu(B), \tag{A.2}$$

for any measurable sets $A, B\subseteq\mathbb{R}^d$. For a fixed such coupling γ, let us decompose the right-hand side of (A.1) as

$$\mathbb{E}\|X-Y\|_2^2 = \mathbb{E}\left[\|X-Y\|_2^2 1_{E_R}\right] + \mathbb{E}\left[\|X-Y\|_2^2 1_{E_R^c}\right], \tag{A.3}$$

where 1_{E_R} stands for the indicator of the event $E_R = \{\|X\|_2\leq R, \|Y\|_2\leq R\}$. Above, E_R^c is the

complement of E_R. For the first expectation on the right-hand side above, we write that

$$\begin{aligned}\mathbb{E}\left[\|X-Y\|_2^2 1_{E_R}\right] &\leq 4R^2\mathbb{E}\left[1_{X\neq Y}1_{E_R}\right]\\ &\leq 4R^2\mathbb{E}[1_{X\neq Y}].\end{aligned} \tag{A.4}$$

For the second expectation on the right-hand side of (A.3), we write that

$$\mathbb{E}\left[\|X-Y\|_2^2 1_{E_R^c}\right] \leq 2\mathbb{E}\left[\|X\|_2^2 1_{E_R^c}\right] + 2\mathbb{E}\left[\|Y\|_2^2 1_{E_R^c}\right]. \qquad ((a+b)^2 \leq 2a^2+2b^2) \tag{A.5}$$

Let us in turn focus on, say, the first expectation on the right-hand side of (A.5). Since

$$1_{E_R^c} = 1_{\{\|X\|_2>R\}} + 1_{\{\|X\|_2\leq R\}}1_{\{\|Y\|_2>R\}},$$

we can write that

$$\begin{aligned}\mathbb{E}\left[\|X\|_2^2 1_{E_R^c}\right] &= \mathbb{E}\left[\|X\|_2^2 1_{\{\|X\|_2>R\}}\right] + \mathbb{E}\left[\|X\|_2^2 1_{\{\|X\|_2\leq R\}}1_{\{\|Y\|_2>R\}}\right]\\ &\leq \mathbb{E}\left[\|X\|_2^2 1_{\{\|X\|_2>R\}}\right] + R^2\mathbb{E}\left[1_{\{\|Y\|_2>R\}}\right].\end{aligned} \tag{A.6}$$

Bounding $\mathbb{E}\left[\|Y\|_2^2 1_{E_R^c}\right]$ similarly, we finally obtain

$$\begin{aligned}\mathbb{E}\|X-Y\|_2^2 \leq& 4R^2\mathbb{E}[1_{X\neq Y}] + 2\mathbb{E}_{X\sim\mu}\left[\|X\|_2^2 1_{\{\|X\|_2>R\}}\right] + 2R^2\mathbb{E}_{X\sim\mu}\left[1_{\{\|X\|_2>R\}}\right]\\ &+ 2\mathbb{E}_{Y\sim\nu}\left[\|Y\|_2^2 1_{\{\|Y\|_2>R\}}\right] + 2R^2\mathbb{E}_{Y\sim\nu}\left[1_{\{\|Y\|_2>R\}}\right]\end{aligned}$$

We now use the fact that $\|\mu-\nu\|_{\mathrm{TV}} = \min_{\gamma\in\Phi(\mu,\nu)}\mathbb{E}_{(X,Y)\sim\gamma}[1_{X\neq Y}]$ (Gibbs and Su, 2002). Hence, using $\gamma^* = \arg\min_{\gamma\in\Phi(\mu,\nu)}\mathbb{E}_{(X,Y)\sim\gamma}[1_{X\neq Y}]$, we have

$$\begin{aligned}\mathrm{W}_2^2(\mu,]nu) &\leq \mathbb{E}_{(X,Y)\sim\gamma^*}\|X-Y\|_2^2\\ &\leq 4R^2\|\mu-\nu\|_{\mathrm{TV}} + 2\mathbb{E}_{X\sim\mu}\left[\|X\|_2^2 1_{\{\|X\|_2>R\}}\right] + 2R^2\mathbb{E}_{X\sim\mu}\left[1_{\{\|X\|_2>R\}}\right]\\ &\quad + 2\mathbb{E}_{Y\sim\nu}\left[\|Y\|_2^2 1_{\{\|Y\|_2>R\}}\right] + 2R^2\mathbb{E}_{Y\sim\nu}\left[1_{\{\|Y\|_2>R\}}\right]\end{aligned}$$

□

Lemma. *Let μ,ν be distributions both satisfying, for some $c, C>0$ and $R\geq C$:*

$$\Pr(\|X\|_2\geq t) \leq ce^{-\frac{t}{C}},\ \forall\, t\geq R. \tag{A.7}$$

Then,

$$\mathrm{W}_2^2(\mu,\nu) \lesssim R^2\|\mu-\nu\|_{\mathrm{TV}} + R^2 e^{-\frac{R}{C}}. \tag{A.8}$$

Proof. We start from the result of Lemma 47. The goal is then to bound the each term on the

right hand side using the tail property of log-concave distributions (Lemma 6).

First notice that

$$\begin{aligned}\int_{\|\mathbf{x}\|_2>R}\int_0^\infty 1_{\{\|\mathbf{x}\|_2\geq z\}} z dz d\mu(\mathbf{x}) &= \int_{\|\mathbf{x}\|_2>R}\int_0^{\|\mathbf{x}\|_2} z dz d\mu(\mathbf{x}) \\ &= \frac{1}{2}\int_{\|\mathbf{x}\|_2>R} \|\mathbf{x}\|_2^2 d\mu(\mathbf{x}) \\ &= \frac{1}{2}\mathbb{E}_{X\sim\mu}\left[\|X\|_2^2 1_{\{\|X\|_2\geq R\}}\right].\end{aligned}$$

We hence have

$$\begin{aligned}\mathbb{E}_{X\sim\mu}\left[\|X\|_2^2 1_{\{\|X\|_2>R\}}\right] &= 2\int_{\|\mathbf{x}\|_2>R}\int_{z\in\mathbb{R}} 1_{\{\|\mathbf{x}\|_2\geq z\}} z dz d\mu(\mathbf{x}) \\ &= 2\int_{z\in\mathbb{R}} z dz \int_{\|\mathbf{x}\|_2\geq \max(R,z)} d\mu(\mathbf{x}) \\ &= 2\int_{z\in\mathbb{R}} z \Pr\left[\|X\|_2\geq \max(R,z)\right] dz \\ &= 2\Pr[\|X\|_2\geq R]\int_0^R z dz + 2\int_R^\infty z\Pr[\|X\|_2\geq z] dz \\ &\leq cR^2 e^{-\frac{R}{C}} + 2c\int_R^\infty z e^{-\frac{z}{C}} dz \qquad \text{(see (A.7))} \\ &\leq c\left(R^2+2CR+2C^2\right)e^{-\frac{R}{C}}. \end{aligned} \tag{A.9}$$

As a direct consequence of (A.7), we also have

$$\mathbb{E}[1_{\{\|X\|_2>R\}}] = \Pr[\|X\|_2>R] \leq ce^{-\frac{R}{C}}. \tag{A.10}$$

Doing the same calculation for Y and replacing the terms in Lemma 47 provides the result. □

A.1.2 Proof of Lemma 10

Lemma (W_2-TV distances inequality)**.** *Let μ,ν be log-concave probability measures on $\mathbb{R}^d$ satisfying Assumption 4 with (η, M_η). Then,*

$$W_2(\mu,\nu) \lesssim \sqrt{\frac{2d(d+1)}{\eta^2} + M_\eta}\max\left(\log\left(\frac{1}{\|\mu-\nu\|_{TV}}\right),1\right)\sqrt{\|\mu-\nu\|_{TV}}. \tag{A.11}$$

Proof. Let us apply Lemma 9 using

$$R = C\max\left(\log\left(\frac{1}{\|\mu-\nu\|_{TV}}\right),1\right).$$

With this choice of R, we have

$$e^{-\frac{R}{C}} \leq \|\mu - \nu\|_{\mathrm{TV}}. \tag{A.12}$$

Thus, Lemma 9 gives

$$\begin{aligned} \mathrm{W}_2^2(\mu,\nu) &\lesssim C^2 \max\left(\log^2\left(\frac{1}{\|\mu-\nu\|_{\mathrm{TV}}}\right), 1\right)\|\mu-\nu\|_{\mathrm{TV}} + C^2\left(1+\max\left(\log\left(\frac{1}{\|\mu-\nu\|_{\mathrm{TV}}}\right),1\right)\right)^2\|\mu-\nu\|_{\mathrm{TV}} \\ &\lesssim C^2 \max\left(\log^2\left(\frac{1}{\|\mu-\nu\|_{\mathrm{TV}}}\right), 1\right)\|\mu-\nu\|_{\mathrm{TV}}. \end{aligned} \tag{A.13}$$

The result then follows from taking the square root of (A.13) and using $C^2 = \frac{d(d+1)}{\eta^2} + M_\eta$ according to Lemma 5.

□

A.1.3 Proof of Theorem 11

Theorem. *Let μ^* be a L-smooth log-concave distribution satisfying Assumption 4 with parameters η, M_η. For every $k \geq 0$, let*

$$n_k = LdC_\eta^2 k^2 e^{3k} \tag{A.14}$$

$$\gamma_k = \frac{1}{Ld} e^{-2k} \tag{A.15}$$

$$\tau_k = C_\eta k. \tag{A.16}$$

Let $\bar{\mu}_k$ be the average distribution associated with the iterates of outer iteration k of DL-ULA (Algorithm 3) using the parameters above, just before the projection step. Then, $\forall \epsilon > 0$, we have:

- *After $N^{\mathrm{KL}} = \tilde{\mathcal{O}}(Ld^3\epsilon^{-\frac{3}{2}})$ total iterations, we obtain $\mathrm{KL}(\bar{\mu}_k;\mu^*) \leq \epsilon$.*
- *After $N^{\mathrm{TV}} = \tilde{\mathcal{O}}(Ld^3\epsilon^{-3})$ total iterations, we obtain $\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}} \leq \epsilon$.*
- *After $N^{\mathrm{W}_2} = \tilde{\mathcal{O}}(Ld^9\epsilon^{-6})$ total iterations, we obtain $\mathrm{W}_2(\bar{\mu}_k,\mu^*) \leq \epsilon\log(1/\epsilon)$.*

Proof. The proof of convergence in TV distance is explained is Section 2.2.4. For KL divergence, we use, as in Section 2.2.4, equation (2.14) and Lemma 9 to obtain

$$\mathrm{KL}(\bar{\mu}_k;\mu^*) \leq \frac{\mathrm{W}_2^2(\hat{\mu}_{k-1},\mu^*)}{2\gamma_k n_k} + Ld\gamma_k \leq 2\,\mathrm{TV}((\hat{\mu}_{k-1},\mu^*))e^{-k} + e^{-2k} \lesssim e^{-2k}.$$

Hence, after $K^{\mathrm{KL}} = \frac{1}{2}\log(1/\epsilon)$ outer iterations, we obtain $\mathrm{KL}(\bar{\mu}_k;\mu^*) \lesssim \epsilon$. Repeating the computaiton of equation (2.25), this corresponds to a total number of iterations of $N^{\mathrm{KL}} = \tilde{\mathcal{O}}(Ld^3\epsilon^{-\frac{3}{2}})$. Note that we bound $\mathrm{KL}(\bar{\mu}_k;\mu^*)$ and not $\mathrm{KL}(\bar{\mu}_k;\mu^*)$ because the projection step makes the KL divergence blow up.

Finally, for W_2 distance, we use equation (2.21) and obtain

$$\mathrm{W}_2^2(\hat{\mu}_k,\mu^*) \lesssim C_\eta^2 k^2 \|\hat{\mu}_k - \mu^*\|_{\mathrm{TV}} + C_\eta^2 k^2 e^{-k} \lesssim C_\eta^2 k^2 e^{-k},$$

i.e., $\mathrm{W}_2(\hat{\mu}_k,\mu^*) \lesssim C_\eta k e^{-k/2}$. Hence, after $K^{\mathrm{W}_2} = 2\log(C_\eta/\epsilon)$ outer iterations, we obtain $\mathrm{W}(\bar{\mu}_k;\mu^*) \lesssim \epsilon\log(1/\epsilon)$. This corresponds to a total number of iterations of $N^{\mathrm{W}_2} = \tilde{\mathcal{O}}\left(Ld^3\left(\frac{C_\eta}{\epsilon}\right)^6\right) = \tilde{\mathcal{O}}\left(Ld^9\epsilon^{-6}\right)$.

□

Lemma 48. *Let $\{u_k\}_{k\geq 0}$ be a real sequence satisfying*

$$u_{k+1} \leq C\left(\sqrt{u_k}e^{-k/2} + e^{-k}\right)$$

for some constant $C > 0$, and $u_0 \leq 1$. Then, $u_k \leq C'e^{-k}\ \forall k \geq 0$ for some constant $C' > 0$ depending only on C.

Proof. Define the sequence $\{v_k\}_{k\geq 0}$ as $v_k = e^{k-1}u_k$ for all $k \geq 0$. We hence have

$$v_{k+1} = e^k u_{k+1} \leq Ce^{k/2}\sqrt{u_k} + C = C\sqrt{ev_k} + C. \tag{A.17}$$

Since the function $f(x) = C\sqrt{ex} + C$ is strictly increasing, we have that $v_k \leq w_k$ where the sequence $\{w_k\}_{k\geq 0}$ is defined as $w_{k+1} = C\sqrt{ew_k} + C$. It is then easy to see that $w_k \leq W$ where W is the only fixed point solution to the equation $x = C\sqrt{ex} + C$ given by $W = \frac{1}{2}(2C + C^2\sqrt{e} + \sqrt{(2C + C^2\sqrt{e}) - 4C^2})$, and assuming that $W \geq 1$.

We hence have

$$u_k = e^{-k+1}v_k \leq e^{-k+1}w_k \leq e^{-k+1}W = C'e^{-k}$$

for $C' \equiv eW$.

□

A.1.4 Proof of Lemma A.1.4

Lemma. *Let $\Omega \subset \mathbb{R}^d$ satisfy Assumption 12. Then $\forall \lambda < \frac{r^2}{8d^2}$,*

$$\mathrm{W}_2^2(\mu_\lambda,\mu^*) \leq C_\Omega^2 d\sqrt{\lambda} \tag{A.18}$$

for some scalar $C_\Omega > 0$ depending on D, r and Δ_1.

Proof. A similar result has been shown in Brosse et al. (2017) (Proposition 5) for W_1 distance, and it is only a matter of trivial technicalities to extend their result to W_2 distance. Since the full proof requires to introduce several concepts that are out of the scope of this paper, we only present the required modifications that allow us to extend the result from W_1- to W_2-distance.

Using Villani (2009), Theorem 6.15, we have:

$$W_2^2(\mu_\lambda, \mu^*) \leq 2 \int_{\mathbb{R}^d} \|x\|_2^2 |\mu^*(x) - \mu_\lambda(x)| dx = A + B \tag{A.19}$$

where

$$A = \int_{K^c} \|x\|_2^2 \mu_\lambda(x) dx \ , \ B = \left(1 - \frac{\int_K e^{-f}}{\int_{\mathbb{R}^d} e^{-f_\lambda}}\right) \int_K \|x\|_2^2 \mu^*(x) dx \tag{A.20}$$

Following very closely the proof in Brosse et al. (2017) (equations 48 to 51), we can easily obtain:

$$A \leq \Delta_1^{-1} \sum_{i=0}^{d-1} \left(\frac{d}{r}\sqrt{\frac{\pi\lambda}{2}}\right)^{d-i} \left(R^2 + 2R\sqrt{\lambda(d-i+2)} + \lambda(d-i+2)\right). \tag{A.21}$$

Therefore, for $\lambda \leq \frac{r^2}{2\pi d^2}$,

$$A \leq \Delta_1^{-1} \sqrt{2\pi\lambda} d r^{-1} \left(R^2 + 2Rr\sqrt{\frac{3}{2d\pi}} + r^2 \frac{3}{2d\pi}\right). \tag{A.22}$$

Moreover, it is also shown in Brosse et al. (2017) (equations 17, 30, 42) that $\left(1 - \frac{\int_K e^{-f}}{\int_{\mathbb{R}^d} e^{-f_\lambda}}\right) \leq \Delta_1^{-1} 2\pi\lambda d r^{-1}$, which implies:

$$B \leq \Delta_1^{-1} \sqrt{2\pi\lambda} d r^{-1} R^2 \tag{A.23}$$

We thus showed that $W_2(\mu_\lambda, \mu^*) \leq C\sqrt{d}\lambda^{\frac{1}{4}}$ for some $C > 0$ depending on D, r, Δ_1.

□

A.1.5 Proof of Theorem 14

Theorem (Iteration complexity of DL-MYULA)**.** *Let $\Omega \subset \mathbb{R}^d$ be a convex set satisfying Assumption 12 and μ^* be a log-concave distribution given by* (2.29) *where f has a L-Lipschitz continuous gradient. For every $k \geq 0$, let*

$$\lambda_k = \frac{1}{\frac{8d^2}{r^2} + de^{2k}} \tag{A.24}$$

$$n_k = Ldk^2e^{5k} \tag{A.25}$$

$$\gamma_k = \frac{1}{Ld}e^{-4k} \tag{A.26}$$

$$\tau_k = Dk \tag{A.27}$$

Then, $\forall \epsilon > 0$, we have:

- *After $N^{\mathrm{TV}} = \mathcal{O}\left(d^{3.5}\epsilon^{-5}\right)$ total iterations, we obtain $\|\hat{\mu}_K - \mu^*\|_{\mathrm{TV}} \leq \epsilon$.*

- *After $N^{\mathrm{W}_2} = \tilde{\mathcal{O}}\left(d^{3.5}\epsilon^{-10}\right)$ total iterations, we obtain $\mathrm{W}_2(\hat{\mu}_K, \mu^*) \lesssim \epsilon$.*

The proof of Theorem 14 is very similar to the one for DL-ULA. Before presenting it, we will need an auxiliary Lemma, showing the light tail property of the distributions μ_λ.

Lemma 49. *For $\lambda \leq \frac{r^2}{8d^2}$, the distribution μ_λ as defined in equation* (2.30) *satisfies*

$$Pr_{X\sim\mu_\lambda}(\|X\|_2 \geq R) \leq \sigma e^{-\frac{R}{D}}$$

for some scalar $\sigma > 0$ and any $R > 0$, where D is the diameter of the constraint set Ω.

Proof. Suppose first that $R \geq 2D$. Then,

$$
\begin{aligned}
\Pr_{X\sim\mu_\lambda}[\|X\|_2 \geq R] &= \frac{\int_{\mathrm{B}(0,R)^c} e^{-f(\mathbf{x})-\frac{1}{2\lambda}\|\mathbf{x}-\mathrm{proj}_\Omega(\mathbf{x})\|_2^2}\,\mathrm{d}\mathbf{x}}{\int_\Omega e^{-f(\mathbf{x})}\,\mathrm{d}\mathbf{x}+\int_{\Omega^c} e^{-f(\mathbf{x})-\frac{1}{2\lambda}\|\mathbf{x}-\mathrm{proj}_\Omega(\mathbf{x})\|_2^2}\,\mathrm{d}\mathbf{x}} \\
&\leq \Delta_1 \frac{\int_{\mathrm{B}(0,R)^c} e^{-\frac{1}{2\lambda}(\|\mathbf{x}\|_2-D)^2}\,\mathrm{d}\mathbf{x}}{\mathrm{Vol}(\Omega)} \\
&\leq \Delta_1 \mathrm{Vol}(\Omega)^{-1}\int_R^\infty u^{d-1}e^{-\frac{1}{2\lambda}(u-D)^2}\,\mathrm{d}u \\
&= \Delta_1 \mathrm{Vol}(\Omega)^{-1} d\mathrm{Vol}(B(0,1))\int_R^\infty u^{d-1}e^{-\frac{1}{2\lambda}(u-D)^2}\,\mathrm{d}u \\
&\leq \Delta_1 d\frac{\mathrm{Vol}(B(0,1))}{\mathrm{Vol}(B(0,r))}D^{d-1}\int_{R-D}^\infty (u+D)^{d-1}e^{-\frac{1}{2\lambda}u^2}\,\mathrm{d}u \\
&\leq \Delta_1 d\frac{1}{r^d}\int_{R-D}^\infty (2u)^{d-1}e^{-\frac{1}{2\lambda}u^2}\,\mathrm{d}u \qquad \text{since } u\geq R-D\geq D \\
&\leq \Delta_1 d\frac{1}{r^d}2^{d-1}\int_{\frac{1}{2\lambda}(R-D)^2}^\infty (2v\lambda)^{\frac{d-1}{2}}e^{-v}\sqrt{\frac{\lambda}{2v}}\,\mathrm{d}u \qquad (v=\frac{1}{2\lambda}u^2) \\
&\leq \Delta_1 d\frac{2^{\frac{3}{2}d-3}\lambda^{\frac{d}{2}}}{r^d}\Gamma\left(\frac{d}{2};\frac{1}{2\lambda}(R-D)^2\right) \qquad \text{where } \Gamma(s;x) \text{ is the incomplete Gamma function} \\
&\leq \Delta_1 d\frac{2^{-3}}{d^d}\frac{d}{2}\left(\frac{1}{2\lambda}(R-D)^2\right)^{\frac{d}{2}}e^{-\frac{1}{2\lambda}(R-D)^2} \qquad \text{since for } x\geq s,\ \Gamma(s;x)\leq sx^se^{-x},\ \lambda\leq\frac{r^2}{8d^2} \\
&\leq \left(\Delta_1^{\frac{1}{d^2}}2^{\frac{-4}{d^2}}d^{\frac{2}{d^2}}\left(\frac{(R-D)^2}{2\lambda d^2}\right)^{\frac{1}{2d}}e^{-\frac{1}{2\lambda d^2}(R-D)^2}\right)^{d^2} \\
&\leq \left(c_d e^{-\frac{1}{\sqrt{2\lambda}d}(R-D)}\right)^{d^2} \qquad \text{since } xe^{-x^2}\leq e^{-x}\ \forall x\geq 0 \text{ and } \frac{1}{2\lambda d^2}(R-D)^2\geq 1
\end{aligned}
$$

where in the last line, $c_d = \Delta_1^{\frac{1}{d^2}}2^{\frac{-4}{d^2}}d^{\frac{2}{d^2}}$. If $c_d e^{-\frac{\sqrt{\frac{1}{2\lambda}}}{d}(R-D)}\geq 1$, then, this does not provide a useful bound, and we can always write $\Pr_{X\sim\mu_\lambda}[\|X\|_2\geq R]\leq 1\leq c_d e^{-\frac{\sqrt{\frac{1}{2\lambda}}}{d}(R-D)}$. On the other hand, if $c_d e^{-\frac{\sqrt{\frac{1}{2\lambda}}}{d}(R-D)}\leq 1$, then we have $\Pr_{X\sim\mu_\lambda}[\|X\|_2\geq R]\leq\left(c_d e^{-\frac{\sqrt{\frac{1}{2\lambda}}}{d}(R-D)}\right)^{d^2}\leq c_d e^{-\frac{\sqrt{\frac{1}{2\lambda}}}{d}(R-D)}$.

Therefore, we can write:

$$
\begin{aligned}
\Pr_{X\sim\mu_\lambda}[\|X\|_2\geq R] &\leq c_d e^{-\frac{\sqrt{\frac{1}{2\lambda}}}{d}(R-D)} \\
&\leq c_d e^{-2(\frac{R}{D}-1)} \qquad \text{since } \lambda\leq\frac{r^2}{8d^2}\leq\frac{D^2}{8d^2} \\
&\leq \max(1,c_d)e^2e^{-\frac{R}{D}}.
\end{aligned}
$$

Moreover, in the case $R\leq 2D$, we have $\max(1,c_d)e^2e^{-\frac{R}{D}}\geq 1\geq\Pr_{X\sim\mu_\lambda}[\|X\|_2\geq R]$. We thus showed the result with $\sigma=\max(1,c_d)e^2$. Note that although c_d depends on d, it is bounded

and converges to 1 as $d \to \infty$, thus it does not involve any asymptotic dependence in d.

□

Using this Lemma, we can now prove our convergence result for DL-MYULA (Theorem 14).

Proof of Theorem 14. Let us denote $\mu_k \equiv \mu_{\lambda_k}$ the target distributions of the ULA iterations at outer iteration $k \geq 1$, and μ_{init} the initial distribution. It is straightforward to show that the distributions μ_k are L_k-smooth with $L_k = L + \frac{1}{\lambda_k}$.

The proof then goes exactly the same way as for Theorem 11. We will show a similar recursive inequality for $\|\bar{\mu}_k - \mu_k\|_{\mathrm{TV}}$ as in (2.24).

For any $k \geq 1$, we have:

$$\begin{aligned}
\|\bar{\mu}_{k+1} - \mu_{k+1}\|_{\mathrm{TV}} &\leq \sqrt{2\,\mathrm{KL}(\bar{\mu}_{k+1}; \mu_{k+1})} \qquad \text{(Pinsker's inequality)} \\
&\leq \sqrt{\frac{W_2^2(\hat{\mu}_k, \mu_{k+1})}{\gamma_k n_k} + 2L_{k+1} d\gamma_k} \\
&\leq \frac{W_2(\hat{\mu}_k, \mu_{k+1})}{\sqrt{\gamma_k n_k}} + \sqrt{2L_{k+1} d\gamma_k} \\
&\leq \frac{W_2(\hat{\mu}_k, \mu_k)}{\sqrt{\gamma_k n_k}} + \frac{W_2(\mu_k, \mu^*)}{\sqrt{\gamma_k n_k}} + \frac{W_2(\mu_{k+1}, \mu^*)}{\sqrt{\gamma_k n_k}} + \sqrt{2L_{k+1} d\gamma_k}
\end{aligned} \tag{A.28}$$

For the second and third term, we use Lemma A.1.4 and the fact that $\lambda_k \leq d^{-1} e^{-2k}$ to show that $\forall k \geq 1$,

$$W_2(\mu_k, \mu^*) \leq C_\Omega d^{\frac{1}{4}} e^{-\frac{k}{2}} \tag{A.29}$$

For the first term, we use the fact that $\Pr_{X \sim \hat{\mu}_{k-1}}(\|X\|_2 \geq Dk) = 0$ thanks to the projection step, and Lemma 49 to apply Lemma 9 with $R = Dk$. We hence obtain

$$W_2^2(\hat{\mu}_k, \mu_k) \lesssim D^2 k^2 \|\hat{\mu}_{k-1} - \mu_{k-1}\|_{\mathrm{TV}} + D^2 k^2 e^{-k}. \tag{A.30}$$

Moreover, similarly as for DL-ULA, we use Lemma 49 to show that

$$\|\hat{\mu}_k - \mu_k\|_{\mathrm{TV}} \lesssim 2\|\bar{\mu}_k - \mu_k\|_{\mathrm{TV}} + e^{-k} \tag{A.31}$$

By replacing (A.29), (A.30) and (A.31) in (A.28), we obtain the following recursive inequality:

$$\|\bar{\mu}_{k+1} - \mu_{k+1}\|_{\mathrm{TV}} \lesssim \sqrt{\|\bar{\mu}_k - \mu_k\|_{\mathrm{TV}}} e^{-k/2} + \sqrt{L_{k+1} d\gamma_k} \tag{A.32}$$

Using the definitions of n_k, γ_k, λ_k, and noting that $L_{k+1} d\gamma_k = \left(1 + \frac{8d^2}{Lr^2}\right) e^{-4k} + \frac{d}{L} e^{-2k}$, we hence

obtain

$$\|\bar{\mu}_{k+1} - \mu_{k+1}\|_{\mathrm{TV}} \lesssim \sqrt{\|\bar{\mu}_k - \mu_k\|_{\mathrm{TV}}} e^{-k/2} + \sqrt{\frac{d}{L}} e^{-k} + \sqrt{1 + \frac{8d^2}{Lr^2}} e^{-2k} \tag{A.33}$$

Using Lemma 50, this sequence hence satisfies

$$\|\bar{\mu}_{k+1} - \mu_{k+1}\|_{\mathrm{TV}} \lesssim \sqrt{d} e^{-k} + d e^{-2k}.$$

Using the result from (Brosse et al., 2017) that $\|\mu_\lambda - \mu^*\|_{\mathrm{TV}} \lesssim d\sqrt{\lambda}$, and using the triangle inequality for the TV distance, we have

$$\|\bar{\mu}_{k+1} - \mu^*\|_{\mathrm{TV}} \lesssim \sqrt{d} e^{-k} + d e^{-2k}.$$

since $\sqrt{\lambda} \leq d^{-1} e^{-2k}$. Hence, for $\epsilon > 0$, after $K = \log(\sqrt{d}/\epsilon)$ outer iteration of DL-MYULA, we have $\|\bar{\mu}_k - \mu^*\|_{\mathrm{TV}} \lesssim \epsilon$. This corresponds to a total number of iterations of

$$N^{\mathrm{TV}} = \sum_{k=1}^{K} n_k = \sum_{k=1}^{K} L d k^2 e^{5k} \propto L d K^2 e^{5K} = L d^{3.5} \log^2(\sqrt{d}/\epsilon) \epsilon^{-5}. \tag{A.34}$$

The complexity can equivalently be computed for W_2.

□

Lemma 50. *Let $\{u_k\}_{k \geq 0}$ be a real sequence satisfying*

$$u_{k+1} \leq C \left(\sqrt{u_k} e^{-k/2} + \sqrt{d} e^{-k} + d e^{-2k} \right) \tag{A.35}$$

for some constants $C, d > 0$, and $u_0 \leq 1$. Then, $u_{k+1} \leq C'(\sqrt{d} e^{-k} + d e^{-2k})$ $\forall k \geq 0$ for some constant $C' > 0$ depending only on C.

Proof. We prove this result by induction. Let us pick a constant C' satisfying $C(1 + \sqrt{2e^2 C'}) \leq C'$, i.e., $C' = \mathcal{O}(C^2)$. We show by induction on k that $u_{k+1} \leq C'(\sqrt{d} e^{-k} + d e^{-2k})$. Suppose that $u_k \leq C'(\sqrt{d} e^{-k+1} + d e^{-2k+2})$. Then, applying equation (A.35), we have

$$
\begin{aligned}
u_{k+1} &\le C\left(\sqrt{u_k}e^{-k/2} + \sqrt{d}e^{-k} + de^{-2k}\right) \\
&\le C\left(\sqrt{C'(\sqrt{d}e^{-k+1} + de^{-2k+2})}e^{-k/2} + \sqrt{d}e^{-k} + de^{-2k}\right) \\
&= C\left(\sqrt{C'(\sqrt{d}e + de^{-k+2})}e^{-k} + \sqrt{d}e^{-k} + de^{-2k}\right) \\
&\le C\left(\sqrt{d}(1+\sqrt{2e^2C'})e^{-k} + de^{-2k}\right) \\
&\le C'(\sqrt{d}e^{-k} + de^{-2k}).
\end{aligned}
$$

The property naturally holds for $k = 0$ since $u_0 \le 1 < C'$.

□

A.2 Appendix for Section 2.3

Table A.1 – Common hyperparameters for Algorithm 14 and Algorithm 15, where most of the values are chosen from Dhariwal et al. (2017).

Hyperparameter	Value
critic optimizer	Adam
critic learning rate	10^{-3}
target update rate τ	0.999
mini-batch size N	128
discount factor γ	0.99
damping factor β	0.9
replay buffer size	10^6
action noise parameter σ	$\{0, 0.01, 0.1, 0.2, 0.3, 0.4\}$
RMSProp parameter α	0.999
RMSProp parameter ϵ	10^{-8}
RMSProp parameter η	10^{-4}
thermal noise σ_t (Algorithm 14)	$\sigma_0 \times (1 - 5 \times 10^{-5})^t$, where $\sigma_0 \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$
warmup steps K_t (Algorithm 14)	$\min\{15, \lfloor (1 + 10^{-5})^t \rfloor\}$

Table A.2 – Exploration-related hyperparameters for Algorithm 14 and Algorithm 15 chosen via grid search (for NR-MDP setting with $\delta = 0.1$).

	Alg. 14: (σ_0, σ)	Alg. 15 (with GAD): σ	Alg. 15 (with Extra-Adam): σ
Walker-v2	$(10^{-2}, 0.01)$	0	0.3
HalfCheetah-v2	$(10^{-2}, 0)$	0.2	0.01
Hopper-v2	$(10^{-3}, 0.2)$	0.2	0.3
Ant-v2	$(10^{-4}, 0.2)$	0.4	0.01
Swimmer-v2	$(10^{-5}, 0.4)$	0.4	0.4
Reacher-v2	$(10^{-3}, 0.2)$	0.4	0.2
Humanoid-v2	$(10^{-4}, 0.01)$	0	0.01
InvertedPendulum-v2	$(10^{-3}, 0.01)$	0.1	0.01

Algorithm 14 DDPG with MixedNE-LD (pre-conditioner = RMSProp)

Hyperparameters: see Table A.1
Initialize (randomly) policy parameters ω_1, θ_1, and Q-function parameter ϕ.
Initialize the target network parameters $\omega_{\text{targ}} \leftarrow \omega_1$, $\theta_{\text{targ}} \leftarrow \theta_1$, and $\phi_{\text{targ}} \leftarrow \phi$.
Initialize replay buffer $\mathscr{D}$.
Initialize $m \leftarrow \mathbf{0}$; $m' \leftarrow \mathbf{0}$.
$t \leftarrow 1$.
repeat
Observe state s, and select actions $a = \mu_{\theta_t}(s) + \xi$; $a' = \nu_{\omega_t}(s) + \xi'$, where $\xi, \xi' \sim \mathscr{N}(0, \sigma I)$
Execute the action $\bar{a} = (1-\delta)a + \delta a'$ in the environment.
Observe reward r, next state s', and done signal d to indicate whether s' is terminal.
Store $\left(s, \bar{a}, r, s', d\right)$ in replay buffer $\mathscr{D}$.
If s' is terminal, reset the environment state.
if it's time to update **then**
for however many updates **do**
$\bar{\omega}_t, \omega_t^{(1)} \leftarrow \omega_t$; $\bar{\theta}_t, \theta_t^{(1)} \leftarrow \theta_t$
for $k = 1, 2, \ldots, K_t$ **do**
Sample a random minibatch of N transitions $B = \left\{\left(s, \bar{a}, r, s', d\right)\right\}$ from $\mathscr{D}$.
Compute targets $y\left(r, s', d\right) = r + \gamma\left(1-d\right) Q_{\phi_{\text{targ}}}\left(s', (1-\delta)\mu_{\theta_{\text{targ}}}\left(s'\right) + \delta \nu_{\omega_{\text{targ}}}\left(s'\right)\right)$.
Update critic by one step of (preconditioned) gradient descent using $\nabla_\phi L\left(\phi\right)$, where

$$L\left(\phi\right) = \frac{1}{N} \sum_{(s,\bar{a},r,s',d) \in B} \left(y\left(r, s', d\right) - Q_\phi\left(s, \bar{a}\right)\right)^2.$$

Compute the (agent and adversary) policy gradient estimates:

$$\widehat{\nabla_\theta J(\theta, \omega_t)} = \frac{1-\delta}{N} \sum_{s \in \mathscr{D}} \nabla_\theta \mu_\theta(s) \nabla_{\bar{a}} Q_\phi(s, \bar{a})\big|_{\bar{a} = (1-\delta)\mu_\theta(s) + \delta \nu_{\omega_t}(s)}$$

$$\widehat{\nabla_\omega J(\theta_t, \omega)} = \frac{\delta}{N} \sum_{s \in \mathscr{D}} \nabla_\omega \nu_\omega(s) \nabla_{\bar{a}} Q_\phi(s, \bar{a})\big|_{\bar{a} = (1-\delta)\mu_{\theta_t}(s) + \delta \nu_\omega(s)}.$$

$g \leftarrow \left[\widehat{\nabla_\theta J(\theta, \omega_t)}\right]_{\theta = \theta_t^{(k)}}$; $m \leftarrow \alpha m + (1-\alpha)\, g \odot g$; $C \leftarrow \text{diag}\left(\sqrt{m + \epsilon}\right)$
$\theta_t^{(k+1)} \leftarrow \theta_t^{(k)} + \eta C^{-1} g + \sqrt{2\eta}\sigma_t C^{-\frac{1}{2}} \xi$, where $\xi \sim \mathscr{N}(0, I)$
$g' \leftarrow \left[\widehat{\nabla_\omega J(\theta_t, \omega)}\right]_{\omega = \omega_t^{(k)}}$; $m' \leftarrow \alpha m' + (1-\alpha)\, g' \odot g'$; $D \leftarrow \text{diag}\left(\sqrt{m' + \epsilon}\right)$
$\omega_t^{(k+1)} \leftarrow \omega_t^{(k)} - \eta D^{-1} g' + \sqrt{2\eta}\sigma_t D^{-\frac{1}{2}} \xi'$, where $\xi' \sim \mathscr{N}(0, I)$
$\bar{\omega}_t \leftarrow \left(1-\beta\right)\bar{\omega}_t + \beta \omega_t^{(k+1)}$; $\bar{\theta}_t \leftarrow \left(1-\beta\right)\bar{\theta}_t + \beta \theta_t^{(k+1)}$
Update the target networks:

$$\begin{aligned} \phi_{\text{targ}} &\leftarrow \tau \phi_{\text{targ}} + (1-\tau)\phi \\ \theta_{\text{targ}} &\leftarrow \tau \theta_{\text{targ}} + (1-\tau)\theta_t^{(k+1)} \\ \omega_{\text{targ}} &\leftarrow \tau \omega_{\text{targ}} + (1-\tau)\omega_t^{(k+1)} \end{aligned}$$

$\omega_{t+1} \leftarrow \left(1-\beta\right)\omega_t + \beta\bar{\omega}_t$; $\theta_{t+1} \leftarrow \left(1-\beta\right)\theta_t + \beta\bar{\theta}_t$
$t \leftarrow t+1$.
until convergence
Output: ω_T, θ_T.

Algorithm 15 DDPG with GAD (pre-conditioner = RMSProp) / Extra-Adam

Hyperparameters: see Table A.1
Initialize (randomly) policy parameters ω_1, θ_1, and Q-function parameter ϕ.
Initialize the target network parameters $\omega_{\text{targ}} \leftarrow \omega_1$, $\theta_{\text{targ}} \leftarrow \theta_1$, and $\phi_{\text{targ}} \leftarrow \phi$.
Initialize replay buffer $\mathscr{D}$.
Initialize $m \leftarrow \mathbf{0}$; $m' \leftarrow \mathbf{0}$.
$t \leftarrow 1$.
repeat
Observe state s, and select actions $a = \mu_{\theta_t}(s) + \xi$; $a' = \nu_{\omega_t}(s) + \xi'$, where $\xi, \xi' \sim \mathcal{N}(0, \sigma I)$
Execute the action $\bar{a} = (1-\delta)a + \delta a'$ in the environment.
Observe reward r, next state s', and done signal d to indicate whether s' is terminal.
Store $\left(s, \bar{a}, r, s', d\right)$ in replay buffer $\mathscr{D}$.
If s' is terminal, reset the environment state.
if it's time to update **then**
for however many updates **do**
Sample a random minibatch of N transitions $B = \left\{\left(s, \bar{a}, r, s', d\right)\right\}$ from $\mathscr{D}$.
Compute targets $y\left(r, s', d\right) = r + \gamma\left(1-d\right) Q_{\phi_{\text{targ}}}\left(s', (1-\delta)\mu_{\theta_{\text{targ}}}\left(s'\right) + \delta \nu_{\omega_{\text{targ}}}\left(s'\right)\right)$.
Update critic by one step of (preconditioned) gradient descent using $\nabla_\phi L\left(\phi\right)$, where

$$L\left(\phi\right) = \frac{1}{N} \sum_{(s,\bar{a},r,s',d) \in B} \left(y\left(r, s', d\right) - Q_\phi\left(s, \bar{a}\right)\right)^2.$$

Compute the (agent and adversary) policy gradient estimates:

$$\widehat{\nabla_\theta J(\theta, \omega_t)} = \frac{1-\delta}{N} \sum_{s \in \mathscr{D}} \nabla_\theta \mu_\theta\left(s\right) \nabla_{\bar{a}} Q_\phi\left(s, \bar{a}\right)|_{\bar{a}=(1-\delta)\mu_\theta(s)+\delta\nu_{\omega_t}(s)}$$

$$\widehat{\nabla_\omega J(\theta_t, \omega)} = \frac{\delta}{N} \sum_{s \in \mathscr{D}} \nabla_\omega \nu_\omega\left(s\right) \nabla_{\bar{a}} Q_\phi\left(s, \bar{a}\right)|_{\bar{a}=(1-\delta)\mu_{\theta_t}(s)+\delta\nu_\omega(s)}.$$

GAD (pre-conditioner = RMSProp):
$g \leftarrow \left[\widehat{\nabla_\theta J(\theta, \omega_t)}\right]_{\theta=\theta_t}$; $m \leftarrow \alpha m + (1-\alpha)\, g \odot g$; $C \leftarrow \text{diag}\left(\sqrt{m + \epsilon}\right)$
$\theta_{t+1} \leftarrow \theta_t + \eta C^{-1} g$
$g' \leftarrow \left[\widehat{\nabla_\omega J(\theta_t, \omega)}\right]_{\omega=\omega_t}$; $m' \leftarrow \alpha m' + (1-\alpha)\, g' \odot g'$; $D \leftarrow \text{diag}\left(\sqrt{m' + \epsilon}\right)$
$\omega_{t+1} \leftarrow \omega_t - \eta D^{-1} g'$
Extra-Adam: use Algorithm 4 from Gidel et al. (2018).
Update the target networks:

$$\begin{aligned}
\phi_{\text{targ}} &\leftarrow \tau \phi_{\text{targ}} + (1-\tau)\phi \\
\theta_{\text{targ}} &\leftarrow \tau \theta_{\text{targ}} + (1-\tau)\theta_{t+1} \\
\omega_{\text{targ}} &\leftarrow \tau \omega_{\text{targ}} + (1-\tau)\omega_{t+1}
\end{aligned}$$

$t \leftarrow t+1$.
until convergence
Output: ω_T, θ_T.

B Appendix for Chapter 3

B.1 Proofs of Section 3.1

B.1.1 Proof of Theorem 15

Theorem. *Let f be a differentiable and Lipschitz continuous function on an open, convex subset $\mathscr{X}$ of a Euclidean space. Let $\|\cdot\|$ be the dual norm. The Lipschitz constant of f is given by*

$$L(f) = \sup_{x\in\mathscr{X}} \left\|\nabla f(x)\right\|_*$$

Proof. First we show that $L(f) \leq \sup_{x\in\mathscr{X}} \left\|\nabla f(x)\right\|_*$.

$$\begin{aligned}
\left|f(y) - f(x)\right| &= \left|\int_0^1 \nabla f((1-t)x + ty)^T (y-x)\, dt\right| \\
&\leq \int_0^1 \left|\nabla f((1-t)x + ty)^T (y-x)\right| dt \\
&\leq \int_0^1 \|\nabla f((1-t)x + ty)\|_* \, dt \, \|y-x\| \\
&\leq \sup_{x\in\mathscr{X}} \|\nabla f(x)\|_* \|y-x\|
\end{aligned}$$

were we have used the convexity of $\mathscr{X}$.

Now we show the reverse inequality $L(f) \geq \sup_{x\in\mathscr{X}} \|\nabla f(x)\|_*$. To this end, we show that for any positive ϵ, we have that $L(f) \geq \sup_{x\in\mathscr{X}} \|\nabla f(x)\|_* - \epsilon$.

Let $z \in \mathscr{X}$ be such that $\|\nabla f(z)\|_* \geq \sup_{x\in\mathscr{X}} \|\nabla f(x)\|_* - \epsilon$. Because $\mathscr{X}$ is open, there exists a sequence $\{h_k\}_{k=1}^{\infty}$ with the following properties:

1. $\langle h_k, \nabla f(z)\rangle = \|h_k\| \|\nabla f(z)\|_*$
2. $z + h_k \in \mathscr{X}$

3. $\lim_{k\to\infty} h_k = 0$.

By definition of the gradient, there exists a function δ such that $\lim_{h\to 0}\delta(h) = 0$ and the following holds:

$$f(z+h) = f(z) + \langle h, \nabla f(z)\rangle + \delta(h)\|h\|.$$

For our previously defined iterates h_k we then have

$$\Rightarrow \left|f(z+h_k) - f(z)\right| = \left|\|h_k\|\|\nabla f(z)\|_* + \delta(h_k)\|h_k\|\right|$$

Dividing both sides by $\|h_k\|$ and using the definition of $L(f)$ we finally get

$$\begin{aligned}
&\Rightarrow L(f) \geq \left|\frac{f(z+h_k) - f(z)}{\|h_k\|}\right| = \left|\|\nabla f(z)\|_* + \delta(h_k)\right| \\
&\Rightarrow L(f) \geq \lim_{k\to\infty}\left|\|f(z)\|_* + \delta(h_k)\right| = \|\nabla f(z)\|_* \\
&\Rightarrow L(f) \geq \sup_{x\in\mathcal{X}} \|\nabla f(x)\|_* - \epsilon.
\end{aligned}$$

□

B.1.2 Proof of Proposition 22

Proposition. *Let f_d be a dense network (all weights are nonzero). Then, the sparsity pattern induced by the network's graph is a valid sparsity pattern for the norm-gradient polynomial of f_d.*

Proof. First we show that $\cup_{i=1}^m I_i = I$. This comes from the fact that any neuron in the network is connected to at least one neuron in the last layer. Otherwise such neuron could be removed from the network altogether.

Now we show the second property of a valid sparsity pattern. Note that the norm-gradient polynomial is composed of monomials corresponding to the product of variables in a path from input to a final neuron. This imples that if we let p_i be the sum of all the terms that involve the neuron $s_{(d-1,i)}$ we have that $p = \sum_i p_i$, and p_i only depends on the variables in I_i.

We now show the last property of the valid sparsity pattern. This is the only part where we use that the network is dense. For any network architecture the first two conditions hold. We will use the fact that the maximal cliques of a chordal graph form a valid sparsity pattern (see for example Lasserre (2006)).

Because the network is dense, we see that the clique I_i is composed of the neuron in the last layer $s_{(d-1,i)}$ and all neurons in the previous layers. Now, consider the extension of the

computational graph $\hat{G}_d = (V, \hat{E})$ where

$$\hat{E} = E \cup \{(s_{j,k}, s_{l,m}) : j, l \leq d-2)\},$$

which consists of adding all the edges between the neurons that are not in the last layer. We show that this graph is chordal. Let $(a_1, \ldots, a_r, a_1)$ be a cycle of length at least 4 ($r \geq 4$). notice that because neurons in the last layer are not connected between them in $\hat{G}$, no two consecutive neurons in this cycle belong to the last layer. This implies that in the subsequence $(a_1, a_2, a_3, a_4, a_5)$ at most three belong to the last layer. A simple analysis of all cases implies that it contains at least two nonconsecutive neurons not in the last layer. Neurons not in the last layer are always connected in $\hat{G}$. This constitutes a chord. This shows that $\hat{G}_d$ is a chordal graph. Its maximal cliques correspond exactly to the sets in the proposition.

□

B.2 Appendix for Section 3.2: Proximal operator in the multi-output setting

In this section, we generalize the computation of the proximal mapping we derived for the single-output scenario to the multi-output case. When the network has multiple-output, the proximal operator $\text{prox}_{\lambda P_1}(X, Y)$ can be written as the solution set of

$$\begin{aligned}\text{prox}_{\lambda P_1}(X, Y) &= \underset{V \in \mathbb{R}^{n\times p}, W \in \mathbb{R}^{n\times m}}{\text{argmin}} \lambda \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} |W_{ij}V_{ik}| + \frac{1}{2}\|V - X\|_F^2 + \frac{1}{2}\|W - Y\|_F^2 \\ &= \underset{V \in \mathbb{R}^{n\times p}, W \in \mathbb{R}^{n\times m}}{\text{argmin}} \sum_{i=1}^{n}\left(\lambda \sum_{j=1}^{m}\sum_{k=1}^{p} |W_{ij}V_{ik}| + \frac{1}{2}\sum_{k=1}^{p}(V_{ik} - X_{ik})^2 + \frac{1}{2}\sum_{j=1}^{m}(W_{ij} - Y_{ij})^2\right).\end{aligned} \tag{B.1}$$

As in the single-output case, we observe that the proximal mapping (B.1) is separable with respect to the i-th rows of the matrices V and W, and that the signs of the decision variables are determined by the signs of (X, Y). Therefore, it is enough to solve, for any $\mathbf{x} \in \mathbb{R}^p, \mathbf{y} \in \mathbb{R}^m$,

$$\min_{\mathbf{v}\in\mathbb{R}^p_+, \mathbf{w}\in\mathbb{R}^m_+} h_\lambda(\mathbf{v}, \mathbf{w}; \mathbf{x}, \mathbf{y}) \equiv \lambda \sum_{k=1}^{p} v_k \sum_{j=1}^{m} w_j + \frac{1}{2}\sum_{k=1}^{p}(v_k - |x_k|)^2 + \frac{1}{2}\sum_{j=1}^{m}(w_j - |y_j|)^2, \tag{B.2}$$

where $\mathbf{v}, \mathbf{w}, \mathbf{x}, \mathbf{y}$ represent one particular row of V, W, X, Y respectively. To improve readability, we will just write $h_\lambda(\mathbf{v}, \mathbf{w})$, assuming that $(\mathbf{x}, \mathbf{y})$ is understood from context.

Similarly as in the single-output case, let us write the first order stationary condition of problem (B.2). The proof is the same as in the single output case.

Lemma 51 (Stationarity conditions)**.** *Let* $(\boldsymbol{v}^*, \boldsymbol{w}^*) \in \mathbb{R}^p_+ \times \mathbb{R}^m_+$ *be an optimal solution of* (B.2) *for*

a given $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^p \times \mathbb{R}^m$. *Then*

$$w_j^* = \max\left\{0, |y_j| - \lambda \sum_{k=1}^{p} v_k^*\right\} \text{ for any } j = 1,2,\ldots,m,$$
$$v_k^* = \max\left\{0, |x_k| - \lambda \sum_{j=1}^{m} w_j^*\right\} \text{ for any } k = 1,2,\ldots,p.$$

The following Lemma expands on the monotonic relation in magnitude originally established for single-output networks in Corollary 28.

Corollary 52. *Let* $(\boldsymbol{v}^*, \boldsymbol{w}^*) \in \mathbb{R}_+^p \times \mathbb{R}_+^m$ *be an optimal solution of* (B.2) *for a given* $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^p \times \mathbb{R}^m$.

1. *The vector* $\boldsymbol{w}^*$ *satisfies that for any* $j, l \in \{1,2,\ldots,m\}$ *it holds that* $w_j^* \geq w_l^*$ *only if* $|y_j| \geq |y_l|$.

2. *The vector* $\boldsymbol{v}^*$ *satisfies that for any* $k, l \in \{1,2,\ldots,p\}$ *it holds that* $v_k^* \geq v_l^*$ *only if* $|x_k| \geq |x_l|$.

3. *Let* $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$ *be the sorted vector of* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *respectively in descending magnitude order. Let* $s_v = |\{k : v_k^* > 0\}|$ *and* $s_w = |\{j : w_j^* > 0\}|$. *Suppose that* $s_v, s_w > 0$. *Then, for all* k, j *such that* $v_k^* > 0, w_j^* > 0$, *we have*

$$v_k^* = |x_k| + \frac{1}{1 - s_v s_w \lambda^2}\left(\lambda^2 s_w \sum_{l=1}^{s_v} |\bar{x}_l| - \lambda \sum_{j=1}^{s_w} |\bar{y}_j|\right), \tag{B.3}$$
$$w_j^* = |y_j| + \frac{1}{1 - s_v s_w \lambda^2}\left(\lambda^2 s_v \sum_{l=1}^{s_w} |\bar{y}_l| - \lambda \sum_{k=1}^{s_v} |\bar{x}_k|\right). \tag{B.4}$$

Proof. The two first points are direct applications of the stationary conditions of Lemma 51.

From the conditions in Lemma 51 we have that

$$\begin{aligned}
\sum_{j=1}^{m} w_j^* &= \sum_{j=1}^{s_w} |\bar{y}_j| - \lambda s_w \sum_{k=1}^{p} v_k^* \\
\sum_{k=1}^{p} v_k^* &= \sum_{k=1}^{s_v} |\bar{x}_k| - \lambda s_v \sum_{j=1}^{m} w_j^* \\
&= \sum_{k=1}^{s_v} |\bar{x}_k| - \lambda s_v \sum_{j=1}^{s_w} |\bar{y}_j| + \lambda^2 s_v s_w \sum_{k=1}^{p} v_k^* \\
&= \frac{1}{1 - \lambda^2 s_v s_w}\left(\sum_{k=1}^{s_v} |\bar{x}_k| - \lambda s_v \sum_{j=1}^{s_w} |\bar{y}_j|\right).
\end{aligned}$$

Plugging this equality in the stationarity condition for $\mathbf{w}^*$ (Lemma 51) gives the result for $\mathbf{w}^*$.

We also have

$$\sum_{j=1}^{m} w_j^* = \sum_{j=1}^{s_w} |\bar{y}_j| - \frac{\lambda s_w}{1-\lambda^2 s_v s_w}\left(\sum_{k=1}^{s_v} |\bar{x}_k| - \lambda s_v \sum_{j=1}^{s_w} |\bar{y}_j|\right)$$
$$= \frac{1}{1-\lambda^2 s_v s_w}\left(-\lambda s_w \sum_{k=1}^{s_v} |\bar{x}_k| + \sum_{j=1}^{s_w} |\bar{y}_j|\right).$$

Plugging the latter to the stationarity condition for $\mathbf{v}^*$ ((Lemma 51) gives the result for $\mathbf{v}^*$. □

We now show that the second order stationarity condition constraints the ranges of sparsities of $\mathbf{v}^*$ and $\mathbf{w}^*$.

Lemma 53 (Sparsity bound)**.** *Let* $(\boldsymbol{v}^*, \boldsymbol{w}^*) \in \mathbb{R}_+^p \times \mathbb{R}_+^m$ *be an optimal solution of* (B.2)*. Denote* $s_v = |\{j : w_j^* > 0\}|$ *and* $s_w = |\{j : w_j^* > 0\}|$*. Then* $s_v s_w \leq \lambda^{-2}$*.*

Proof. Since $(\mathbf{v}^*, \mathbf{w}^*)$ is an optimal solution of (B.2) and the objective function in (B.2) is twice continuously differentiable, $(\mathbf{v}^*, \mathbf{w}^*)$ satisfies the second order necessary optimality conditions. That is, for any $d \in \mathbb{R}^p \times \mathbb{R}^m$ satisfying that $(\mathbf{v}^*, \mathbf{w}^*) + d \in \mathbb{R}_+^p \times \mathbb{R}_+^m$ and $d^T \nabla h_\lambda(\mathbf{v}^*, \mathbf{w}^*) = 0$ it holds that

$$d^T \nabla^2 h_\lambda(\mathbf{v}^*, \mathbf{w}^*) d = d^T \begin{pmatrix} I_{p\times p} & \Lambda_{p\times m} \\ \Lambda_{m\times p} & I_{m\times m} \end{pmatrix} d \geq 0,$$

where the first row/column corresponds to $\mathbf{v}$ and the others correspond to $\mathbf{w}$, I denotes the identity matrix and Λ denotes a matrix completely filled with λ. Similarly as in the single output case, we require that the submatrix of $\nabla^2 h_\lambda(\mathbf{v}^*, \mathbf{w}^*)$ containing the rows and columns corresponding to the positive coordinates in $(\mathbf{v}^*, \mathbf{w}^*)$ is positive semidefinite. Since the minimal eigenvalue of this submatrix equals $1 - \lambda\sqrt{|S_v||S_w|}$, we have that

$$\lambda^{-2} \geq |S_v||S_w|.$$

□

Without loss of generality, **we assume hereafter that the vectors $\mathbf{x}, \mathbf{y}$ are already sorted in decreasing order of magnitude**. Corollary 52 shows that for each pair (s_v, s_w), $s_v = 1, \ldots, p$, $s_w = 1, \ldots, m$, there exists a stationary point $(\mathbf{v}^{(s_v,s_w)}, \mathbf{w}^{(s_v,s_w)})$ of $h_\lambda(\cdot,\cdot)$ such that $|\{k : v_k^{(s_v,s_w)} > 0\}| = s_v$, $|\{j : w_j^{(s_v,s_w)} > 0\}| = s_w$, given by

$$v_k^{(s_v,s_w)} = |x_k| + \frac{1}{1-s_v s_w \lambda^2}\left(\lambda^2 s_w \sum_{l=1}^{s_v} |x_l| - \lambda \sum_{j=1}^{s_w} |y_j|\right) \text{ for } k = 1, 2, \ldots, s_v, \text{ and } v_k^{(s_v,s_w)} = 0 \text{ otherwise}$$
$$w_j^{(s_v,s_w)} = |y_j| + \frac{1}{1-s_v s_w \lambda^2}\left(\lambda^2 s_v \sum_{l=1}^{s_w} |y_l| - \lambda \sum_{k=1}^{s_v} |x_k|\right) \text{ for } j = 1, 2, \ldots, s_w, \text{ and } w_j^{(s_v,s_w)} = 0 \text{ otherwise.} \tag{B.5}$$

Finding a non-trivial solution (i.e., such that $s_v, s_w \neq 0$) to Problem (B.2) hence boils down to solving

$$\begin{aligned} &\min_{s_v \in \{1,\dots,p\}, s_w \in \{1,\dots,m\}} h_\lambda(\mathbf{v}^{(s_v,s_w)}, \mathbf{w}^{(s_v,s_w)}) \\ &\text{such that } s_v s_w \leq \lambda^{-2}, \mathbf{v}^{(s_v,s_w)} \geq 0, \mathbf{w}^{(s_v,s_w)} \geq 0 \end{aligned} \tag{B.6}$$

A possible way of solving problem (B.6) would be to exhaustively compute the value of h_λ for each stationary point associated with sparsities $s_v = 1,\dots,p$, $s_w = 1,\dots,m$ such that $s_v s_w \leq \lambda^{-2}$ and $\mathbf{w}^{(s_v,s_w)} \geq 0, \mathbf{w}^{(s_v,s_w)} \geq 0$. However, trying all possible pairs of sparsities (s_v, s_w) is computationally costly. Similarly as is the single output case, we can exploit some structure of the objective function h_λ in order to reduce the candidate optimal pairs of sparsities.

Lemma 54. *Given $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^p \times \mathbb{R}^m$, for all $s_v, s_w \in \{0,\dots,p\} \times \{0,\dots,m\}$ satisfying $s_v s_w < \lambda^{-2}$, we have*

$$\begin{aligned} h_\lambda(\boldsymbol{v}^{(s_v,s_w)}, \boldsymbol{w}^{(s_v,s_w)}) &< h_\lambda(\boldsymbol{v}^{(s_v,s_w-1)}, \boldsymbol{w}^{(s_v,s_w-1)}), \\ h_\lambda(\boldsymbol{v}^{(s_v,s_w)}, \boldsymbol{w}^{(s_v,s_w)}) &< h_\lambda(\boldsymbol{v}^{(s_v-1,s_w)}, \boldsymbol{w}^{(s_v-1,s_w)}). \end{aligned}$$

Proof. The proof follows the same lines as in the single output case. Plugging the definitions

from equation (B.5), we have

$$
\begin{aligned}
h_\lambda\left(\mathbf{v}^{(s_v,s_w)},\mathbf{w}^{(s_v,s_w)}\right) &= \frac{s_v}{2}\left(\frac{1}{1-\lambda^2 s_v s_w}\left(\lambda^2 s_w\sum_{k=1}^{s_v}|x_k|-\lambda\sum_{j=1}^{s_w}|y_j|\right)\right)^2+\frac{1}{2}\sum_{k=s_v+1}^{p}x_k^2\\
&+\frac{s_w}{2}\left(\frac{1}{1-\lambda^2 s_v s_w}\left(\lambda^2 s_v\sum_{j=1}^{s_w}|y_j|-\lambda\sum_{k=1}^{s_v}|x_k|\right)\right)^2+\frac{1}{2}\sum_{j=s_w+1}^{m}y_j^2\\
&+\frac{\lambda}{(1-\lambda^2 s_v s_w)^2}\left(\sum_{k=1}^{s_v}|x_k|-\lambda s_v\sum_{j=1}^{s_w}|y_j|\right)\left(-\lambda s_w\sum_{k=1}^{s_v}|x_k|+\sum_{j=1}^{s_w}|y_j|\right)\\
&=\frac{1}{2(1-\lambda^2 s_v s_w)^2}\left(\left(\sum_{k=1}^{s_v}|x_k|\right)^2(\lambda^4 s_v s_w^2+\lambda^2 s_w-2\lambda^2 s_w)+\left(\sum_{j=1}^{s_w}|y_j|\right)^2(\lambda^2 s_v+\lambda^4 s_v^2 s_w-2\lambda^2 s_v)\right.\\
&\left.\left(\sum_{k=1}^{s_v}|x_k|\right)\left(\sum_{j=1}^{s_w}|y_j|\right)(-2\lambda^3 s_v s_w-2\lambda^3 s_v s_w+2\lambda+2\lambda^3 s_v s_w)\right)+\frac{1}{2}\sum_{k=s_v+1}^{p}x_k^2+\frac{1}{2}\sum_{j=s_w+1}^{m}y_j^2\\
&=\frac{1}{2(1-\lambda^2 s_v s_w)}\left(-\lambda^2 s_w\left(\sum_{k=1}^{s_v}|x_k|\right)^2-\lambda^2 s_v\left(\sum_{j=1}^{s_w}|y_j|\right)^2+2\lambda\left(\sum_{k=1}^{s_v}|x_k|\right)\left(\sum_{j=1}^{s_w}|y_j|\right)\right)+\frac{1}{2}\sum_{k=s_v+1}^{p}x_k^2+\frac{1}{2}\sum_{j=s_w+1}^{m}y_j^2
\end{aligned}
\tag{B.7}
$$

$$
\begin{aligned}
&=\left(1+\frac{\lambda^2 s_v}{1-\lambda^2 s_v s_w}\right)\frac{1}{2(1-\lambda^2 s_v(s_w-1))}\left(-\lambda^2(s_w-1)\left(\sum_{k=1}^{s_v}|x_k|\right)^2-\lambda^2\left(\sum_{k=1}^{s_v}|x_k|\right)^2\right.\\
&\left.-\lambda^2 s_v\left(\left(\sum_{j=1}^{s_w-1}|y_j|\right)^2+2\lambda|y_{s_w}|\sum_{j=1}^{s_w-1}|y_j|+y_{s_w}^2\right)+2\lambda\sum_{k=1}^{s_v}|x_k|\left(\sum_{j=1}^{s_w-1}|y_j|+|y_{s_w}|\right)\right)
\end{aligned}
\tag{B.8}
$$

$$
+\frac{1}{2}\sum_{k=s_v+1}^{p}x_k^2+\frac{1}{2}\sum_{j=s_w-1+1}^{m}y_j^2-\frac{1}{2}y_{s_w}^2. \tag{B.9}
$$

By applying equation (B.7) at s_v, s_w-1, we can express the right hand side of equation (B.8) in terms of $h_\lambda\left(v^{(s_v,s_w-1)},w^{(s_v,s_w-1)}\right)$ as:

$$
\begin{aligned}
h_\lambda\left(v^{(s_v,s_w)},w^{(s_v,s_w)}\right) &= h_\lambda\left(v^{(s_v,s_w-1)},w^{(s_v,s_w-1)}\right)+\frac{1}{2(1-\lambda^2 s_v(s_w-1))}\left(-\lambda^2\left(\sum_{k=1}^{s_v}|x_k|\right)^2\right.\\
&\left.-\lambda^2 s_v|y_{s_w}|\left(2\sum_{j=1}^{s_w-1}|y_j|+|y_{s_w}|\right)+2\lambda|y_{s_w}|\sum_{k=1}^{s_v}|x_k|\right)+\frac{\lambda^2 s_v}{2(1-\lambda^2 s_v s_w)(1-\lambda^2 s_v(s_w-1))}\left(-\lambda^2 s_w\left(\sum_{k=1}^{s_v}|x_k|\right)^2\right.\\
&\left.-\lambda^2 s_v\left(\sum_{j=1}^{s_w}|y_j|\right)^2-2\lambda\left(\sum_{k=1}^{s_v}|x_k|\right)\left(\sum_{j=1}^{s_w}|y_j|\right)\right)-\frac{1}{2}y_{s_w}^2.
\end{aligned}
$$

Therefore:

$$
\begin{aligned}
&h_\lambda\left(v^{(s_v,s_w)}, w^{(s_v,s_w)}\right) - h_\lambda\left(v^{(s_v,s_w-1)}, w^{(s_v,s_w-1)}\right) \\
&\quad = -\frac{1}{2(1-\lambda^2 s_v(s_w-1))}\left(-2\lambda|y_{s_w}|\left(\sum_{k=1}^{s_v}|x_k| - \lambda s_v\sum_{j=1}^{s_w}|y_j|\right) - \lambda^2 s_v|y_{s_w}|^2 + \lambda^2\left(\sum_{k=1}^{s_v}|x_k|\right)^2\right. \\
&\quad \left. + \frac{\lambda^2 s_v}{1-\lambda^2 s_v s_w}\left(\lambda^2 s_w\left(\sum_{k=1}^{s_v}|x_k|\right)^2 + \lambda^2 s_v\left(\sum_{j=1}^{s_w}|y_j|\right)^2 - 2\lambda\left(\sum_{k=1}^{s_v}|x_k|\right)\left(\sum_{j=1}^{s_w}|y_j|\right)\right) + (1-\lambda^2 s_v s_w + \lambda^2 s_v)|y_{s_w}|\right) \\
&\quad = -\frac{1}{2(1-\lambda^2 s_v(s_w-1))}\left((1-\lambda^2 s_v s_w)y_{s_w}^2 - 2\lambda|y_{s_w}|(1-\lambda^2 s_v s_w)\sum_{k=1}^{s_v} v_k^{(s_v,s_w)} + \frac{\lambda^2}{1-\lambda^2 s_v s_w}\left(\sum_{k=1}^{s_v}|x_k| - \lambda s_v\sum_{j=1}^{s_w}|y_j|\right)^2\right) \\
&\quad = -\frac{1-\lambda^2 s_v s_w}{2(1-\lambda^2 s_v(s_w-1))}\left(|y_{s_w}| - \lambda\sum_{k=1}^{s_v} v_k^{(s_v,s_w)}\right)^2 < 0.
\end{aligned}
$$

The second result is obtain directly by symmetry between $\mathbf{v}$ and $\mathbf{w}$. □

Moreover, the feasibility conditions $\mathbf{v}^{(s_v,s_w)} \geq 0, \mathbf{w}^{(s_v,s_w)} \geq 0$ in (B.6) also have a monotonic property:

Lemma 55. *Let $(k,l) \in [p] \times [m]$ be such that $kl \leq \lambda^{-2}$.*

If $\boldsymbol{v}^{(k,l)} \geq 0$ and $\boldsymbol{w}^{(k,l)} \geq 0$, then, $\boldsymbol{v}^{(i,j)} \geq 0$ and $\boldsymbol{w}^{(i,j)} \geq 0$ $\forall\, i = 1,\ldots,k$ and $\forall\, j = 1,\ldots,l$.

Proof. Since the first k entries of $\mathbf{v}^{(k,l)}$ are ordered in decreasing order, we have that $\mathbf{v}^{(k,l)} \geq 0$ if and only if $v_k^{(k,l)} \geq 0$. Similarly, $\mathbf{w}^{(k,l)} \geq 0$ if and only if $w_l^{(k,l)} \geq 0$.

Suppose that $\mathbf{v}^{(k,l)} \geq 0$ and $\mathbf{w}^{(k,l)} \geq 0$. By induction, in order to prove the result, it is sufficient to prove that $v_{k-1}^{(k-1,l)} \geq 0$, $v_k^{(k,l-1)} \geq 0$, $w_l^{(k-1,l)} \geq 0$ and $w_{l-1}^{(k,l-1)} \geq 0$. We only prove the result for $\mathbf{v}$, as the proof for $\mathbf{w}$ is identical.

Using equation (B.5), we have that

$$
\begin{aligned}
(1-kl\lambda^2)v_k^{(k,l)} &= (1-kl\lambda^2)|x_k| + \lambda^2 l\sum_{i=1}^{k}|x_i| - \lambda\sum_{j=1}^{l}|y_j| \qquad \text{(B.10)} \\
&= (1-kl\lambda^2)|x_k| + (1-(k-1)l\lambda^2)|x_{k-1}| - (1-(k-1)l\lambda^2)|x_{k-1}| + \lambda^2 l\sum_{i=1}^{k-1}|x_i| + \lambda^2 l|x_k|\lambda\sum_{j=1}^{l}|y_j| \\
&= (1-(k-1)l\lambda^2)v_{k-1}^{(k-1,l)} + (1-(k-1)l\lambda^2)(|x_k| - |x_{k-1}|).
\end{aligned}
$$

Therefore:

$$
v_{k-1}^{(k-1,l)} = \frac{1-(k-1)l\lambda^2}{1-kl\lambda^2}v_k^{(k,l)} + |x_{k-1}| - |x_k| \geq 0,
$$

since the vector x is ordered in decreasing order of magnitude, and thus $|x_{k-1}| - |x_k| \geq 0$.

Using again equation (B.10), we have that

$$\begin{aligned}(1-kl\lambda^2)\,v_k^{(k,l)} &= (1-kl\lambda^2)|x_k| + (1-k(l-1)\lambda^2)|x_k| - (1-k(l-1)\lambda^2)|x_k| \\ &\quad + \lambda^2(l-1)\sum_{i=1}^{k}|x_i| + \lambda^2\sum_{i=1}^{k}|x_i| - \lambda\sum_{j=1}^{l-1}|y_j| - \lambda|y_l| \\ &= (1-k(l-1)\lambda^2)\,v_k^{(k,l-1)} - k\lambda^2|x_k| + \lambda^2\sum_{i=1}^{k}|x_i| - \lambda|y_l|,\end{aligned}$$

where the last equality follows from equation (B.10) for $v_k^{(k,l-1)}$. Thus,

$$(1-k(l-1)\lambda^2)\,v_k^{(k,l-1)} = (1-kl\lambda^2)\,v_k^{(k,l)} + k\lambda^2|x_k| - \lambda^2\sum_{i=1}^{k}|x_i| + \lambda|y_l|. \tag{B.11}$$

From the definition of $v_k^{(k,l)}$ (equation (B.5)), we have that $v_k^{(k,l)} \geq 0$ is equivalent to the condition:

$$|x_k| \geq \frac{\lambda\sum_{j=1}^{l}|y_j| - l\lambda^2\sum_{i=1}^{k}|x_i|}{1-kl\lambda^2}.$$

Plugging this inequality in equation (B.11), we obtain:

$$\begin{aligned}(1-k(l-1)\lambda^2)\,v_k^{(k,l-1)} &\geq (1-kl\lambda^2)\,v_k^{(k,l)} + \frac{k\lambda^2}{1-kl\lambda^2}\left(\lambda\sum_{j=1}^{l}|y_j| - l\lambda^2\sum_{i=1}^{k}|x_i|\right) + \lambda|y_l| - \lambda^2\sum_{i=1}^{k}|x_i| \\ &= (1-kl\lambda^2)\,v_k^{(k,l)} + \frac{\lambda}{1-kl\lambda^2}\left(k\lambda^2\sum_{j=1}^{l}|y_j| - kl\lambda^3\sum_{i=1}^{k}|x_i| + (1-kl\lambda^2)|y_l| - \lambda(1-kl\lambda^2)\sum_{i=1}^{k}|x_i|\right) \\ &= (1-kl\lambda^2)\,v_k^{(k,l)} + \frac{\lambda}{1-kl\lambda^2}\left(k\lambda^2\sum_{j=1}^{l}|y_j| + (1-kl\lambda^2)|y_l| - \lambda\sum_{i=1}^{k}|x_i|\right). \quad \text{(B.12)}\end{aligned}$$

From the definition of $w_l^{(k,l)}$ (equation (B.5)), we have that $w_l^{(k,l)} \geq 0$ is equivalent to the condition:

$$(1-kl\lambda^2)|y_l| + k\lambda^2\sum_{j=1}^{l}|y_j| - \lambda\sum_{i=1}^{k}|x_i| \geq 0. \tag{B.13}$$

Since the expression of equation (B.13) is exactly the same as the one inside the parentheses of equation (B.12), plugging this relation to (B.11) thus shows that $(1-k(l-1)\lambda^2)\,v_k^{(k,l-1)} \geq 0$, i.e. $v_k^{(k,l-1)} \geq 0$. □

To properly address the complications arising from handling two intertwining sparsity levels at the same time, we introduce the notion of *maximal feasibility boundary (MFB)* which acts a frontier of possible sparsity levels.

Definition 56 (Maximal feasibility boundary)**.** *We say that a sparsity pair* $(s_v, s_w) \in \{0,\dots,p\} \times \{0,\dots,m\}$ *is on the maximal feasibility boundary (MFB) if incrementing either* s_v *or* s_w *results with a non-stationary point. That is, if both of the following conditions hold:*

- $v_{s_v+1}^{(s_v+1,s_w)} < 0$ *or* $w_{s_w}^{(s_v+1,s_w)} < 0$ *or* $(s_v+1)s_w > \lambda^{-2}$,
- $v_{s_v}^{(s_v,s_w+1)} < 0$ *or* $w_{s_w+1}^{(s_v,s_w+1)} < 0$ *or* $s_v(s_w+1) > \lambda^{-2}$.

The efficient computation of the multi-output robust-sparse proximal mapping is based on the fact that we only need to compute the value of h_λ for sparsity levels that are on the MFB. This allows us to find the optimal sparsity in time $\mathcal{O}(p+m)$, improving upon the $\mathcal{O}(pm)$ complexity of the exhaustive search. Algorithm 16 implements the above by employing a binary search type procedure defined in Algorithm 17 to calculate the MFB.

Theorem 57 (Multi-output prox computation)**.** *Let* $(V^*_{:,i}, W^*_{i,:})$ *be the output of Algorithm 16 with input* $X_{:,i}, Y_{i,:}, \lambda$*, where each* $X_{:,i}$*,* $Y_{i,:}$ *are sorted in decreasing magnitude order. Then* (V^*, W^*) *is a solution to* (3.17)*.*

Algorithm 16 Multi-output robust-sparse proximal mapping

Input: $x \in \mathbb{R}^p$, $y \in \mathbb{R}^m$ ordered in decreasing magnitude order, $\lambda > 0$.

```
Employ Algorithm 17: Find the set of sparsity pairs S = {(s_v, s_w)} that are on the MFB
h_opt ← ∞
for (s_v, s_w) ∈ S do
    Compute v^(s_v,s_w) and w^(s_v,s_w) as given in equation (B.5)
    if h_λ(v^(s_v,s_w), w^(s_v,s_w); x, y) < h_opt then
        h_opt = h_λ(v^(s_v,s_w), w^(s_v,s_w); x, y)
        v* ← v^(s_v,s_w), w* ← w^(s_v,s_w)
return (sign(x) ∘ v*, sign(y) ∘ w*)
```

Time complexity of Algorithm 16. It is easy to see that the maximal feasibility boundary contains at most $\min(m,p)$ pairs, and Algorithm 17 finds them all in time $\mathcal{O}(m+p)$. Then, for each such pair (s_v, s_w), we must compute $\mathbf{v}^{(s_v,s_w)}$ and $\mathbf{w}^{(s_v,s_w)}$ and $h_\lambda(\mathbf{v}^{(s_v,s_w)}, \mathbf{w}^{(s_v,s_w)})$, which takes time $\mathcal{O}(m+p)$. The total complexity of Algorithm 16 is thus $\mathcal{O}(\min(m,p)(m+p))$. In most practical application, the output layer size p can be considered $\mathcal{O}(1)$, so that the complexity of computing this proximal mapping is comparable to the complexity of computing one stochastic gradient.

Lemma 58. *The set S returned by Algorithm 17 contains all, and only, the sparsity pairs that are on the maximal feasibility boundary.*

Proof. First recall that the MFB is defined as all pairs $(s_v, s_w) \in \{0,\dots,p\} \times \{0,\dots,m\}$ satisfying the conditions:

1. $v_{s_v}^{(s_v,s_w)} > 0$ and $w_{s_w}^{(s_v,s_w)} > 0$ and $s_v s_w \le \lambda^{-2}$,

Algorithm 17 Finding sparsity pairs on the maximal feasibility boundary

Input: $\mathbf{x} \in \mathbb{R}^p$, $\mathbf{y} \in \mathbb{R}^m$ ordered in decreasing magnitude order, $\lambda > 0$.

1: $s_v \leftarrow 0$, $s_w \leftarrow m$
2: $S \leftarrow \emptyset$
3: $maximal \leftarrow True$
4: **while** $s_v \leq p$ and $s_w \geq 0$ **do**
5: Compute $v_{s_v}^{(s_v,s_w)}$ and $w_{s_w}^{(s_v,s_w)}$ as shown in equation (B.5)
6: **if** $v_{s_v}^{(s_v,s_w)} < 0$ or $w_{s_w}^{(s_v,s_w)} < 0$ or $s_v s_w \geq \lambda^{-2}$ **then**
7: **if** $maximal$ **then**
8: $S \leftarrow S \cup \{(s_v - 1, s_w)\}$
9: $maximal \leftarrow False$
10: $s_w \leftarrow s_w - 1$
11: **else**
12: $s_v \leftarrow s_v + 1$
13: $maximal \leftarrow True$
14: **if** $s_v == p + 1$ **then**
15: $S \leftarrow S \cup \{(s_v - 1, s_w)\}$
16: **return** S

2. $v_{s_v+1}^{(s_v+1,s_w)} \leq 0$ or $w_{s_w}^{(s_v+1,s_w)} \leq 0$ or $(s_v + 1)s_w > \lambda^{-2}$ or $s_v = p$,
3. $v_{s_v}^{(s_v,s_w+1)} \leq 0$ or $w_{s_w+1}^{(s_v,s_w+1)} \leq 0$ or $s_v(s_w + 1) > \lambda^{-2}$ or $s_w = m$.

Algorithm 17 plays on the properties of *feasibility-infeasibility* of the sparsity levels to build the MFB. We say that a pair of the sparsity pair $(i, j) \in \{0, \ldots, p\} \times \{0, \ldots, m\}$ is *feasible* if $v_i^{(i,j)} \geq 0$, $w_j^{(i,j)} \geq 0$ and $ij < \lambda^{-2}$, and denote this by the property $P(i, j)$, i.e.

$$(i, j) \text{ is feasible } \Leftrightarrow P(i, j).$$

Our claim can be read as: Let $(i, j) \in \{0, \ldots, p\} \times \{0, \ldots, m\}$, then (i, j) is added to S by Algorithm 17 if and only if (i, j) belongs to the MFB, i.e.,

$$(i, j) \in \text{MFB} \Leftrightarrow (i, j) \in S.$$

Obviously, only feasible sparsity pairs belong to the MFB, and it is quite easy to see that only feasible sparsity pairs will belong to an output S of Algorithm 17. Indeed, Algorithm 17 monotonically decrements s_w starting from $s_w = m$ and increments s_v starting from $s_v = 0$. For each value of s_w, it increases s_v while the current pair (s_v, s_w) is feasible (lines $12 - 15$). Once it reaches an infeasible point (i, s_w), and in the case where s_v has been increased at least once for this particular value of s_w, it adds to S the pair encountered just before, i.e., $(i - 1, s_w)$, and then decrements s_w (lines $6 - 11$).

We first prove the $\Rightarrow$ statement. Suppose that some pair (i, j) belongs to the MFB. Let us first leave aside the corner cases, and assume that $i < p$ and $j < m$.

Suppose first that s_w reaches j before s_v reaches i, i.e., $s_v < i$. Since the pair (i, j) is feasible, and due to the monotonicity property of the feasibility condition (Lemma 54), all pairs (k, s_w) with $k \leq i$ must be feasible. Therefore, s_v will be increased until reaching $i+1$. By definition of the MFB, the pair $(i+1, j)$ must be infeasible. Since s_v has necessarily been increased at least once for this value of $s_w = j$, and so the pair $(i+1-1, j) = (i, j)$ will be added to S before decrementing s_w.

In the special case where $i = p$, no infeasible point will be found. The loop will thus finish with $s_w = j$ and $s_v = p+1$. The condition at line 17 will thus hold, and the pair (p, j) will be added to S.

Suppose now that s_v reaches i before s_w reaches j, i.e., $s_w > j$. Since (i, j) is in the MFB, then the pair $(i, j+1)$ must be infeasible. Thanks to the monotonicity property of the feasibility condition (Lemma 54), all pairs (s_v, k) with $k \geq i$ must also be infeasible. Therefore, s_w will be decreased until reaching $s_w = j$. Then, similarly as in the previous case, since (i, j) is feasible, s_v will be increased, and the pair (i, j) added to S.

We now prove the $\Leftarrow$ statement. We show that if (i, j) is added to S, then it must belong to the MFB, i.e., it satisfies all three properties recalled in the beginning of the proof.

Let us first show that for each pair (s_v, s_w) encountered during the algorithm, the pair (s_v-1, s_w) is always feasible (or $s_v = 0$). We can show that this property is conserved each time the algorithm either increases s_v or decreases s_w. First note that the pair $(0, m)$ is always feasible. The algorithm will then necessarily first goes to the pair $(1, m)$ and $P(1, m)$ is true. Then suppose that $P(s_v, s_w)$ is true for some pair (s_v, s_w) encountered during the algorithm. Then, if s_v is increases, it means that the pair (s_v, s_w) is feasible. The next encountered pair is then (s_v+1, s_w) and $P(s_v+1, s_w)$ is true. On the other hand, suppose that s_w is decreased. The next encountered pair is thus (s_v, s_w-1). Since $P(s_v, s_w)$ is true, it means that (s_v-1, s_w) is feasible. By Lemma 54, it implies that (s_v-1, s_w-1) is also feasible, and thus $P(s_v, s_w-1)$ is true. We thus proved that $P(s_v, s_w)$ is true for any pair (s_v, s_w) encountered during the algorithm. Therefore, since any pair added to S is of the form (s_v-1, s_w) for some pair (s_v, s_w) encountered during the algorithm, then any pair added to S must be feasible.

The second property of the MFB is straightforward to show. Indeed, if $(i-1, j)$ is added to S, it means that the pair (i, j) is infeasible due to condition on line 6.

Finally, the third property follows from the fact that, when reaching $s_w = j$, s_v must be increased at least once for adding a pair of the form (i, j) to S. Let $s_v^{(j)}$ be the value of s_v when the algorithm reaches $s_w = j$. We necessarily have $s_v^{(j)} \leq i$. This implies that the pair $(s_v^{(j)}, j+1)$ is infeasible, otherwise s_v would have been increased to a greater value at the previous value $s_w = j+1$. By Lemma 54, and since $s_v^{(j)} \leq i$ this implies that the pair $(i, j+1)$ is also infeasible, hence the result.

□

B.3 Proofs of Section 3.2

B.3.1 Proof of Theorem 23

Theorem. *Let $\{z^k\}_{k\geq 0}$ be a sequence generated by Algorithm 6 with $\{\eta^k\}_{k\geq 0} \subseteq (0, 1/L)$. Then*

1. *Any accumulation point of $\{z^k\}_{k\geq 0}$ is a critical point of $f+g$.*
2. *If f satisfies the Kurdyka-Lojasiewicz (KL) property (Attouch et al., 2010), then $\{z^k\}_{k\geq 0}$ converges to a critical point.*
3. *Suppose that η_k is chosen such that there exists $c > 0$ such that $\sum_{k=0}^{K} \frac{1}{\eta_k} \geq cK$ for any integer $K > 0$. Then*

$$\min_{k=0,\ldots,K} \|z^{k+1} - z^k\|_2 \leq \sqrt{\frac{2(F(z^0) - F_*)}{(c-L)K}},$$

where $F_* \equiv \min_{x\in\mathbb{R}^d} F(x)$.

Proof. The first and second parts follow from the results established by Bolte et al. (2013). For the third point, we will rely on the following Sufficient Decrease property from Bolte et al. (2013).

Lemma 59 (Sufficient decrease property (Bolte et al., 2013, Lemma 2))**.** *Let $\Psi : {}^n \to$ be a continuously differentiable function with gradient assumed L_Ψ-Lipschitz continuous, and let $\sigma : {}^n \to (-\infty, \infty]$ be a proper lower semi-continuous function satisfying that $\inf \sigma > -\infty$. Fix any $t \in (0, 1/L_\Psi)$. Then, for any $\mathbf{u} \in {}^n$ and any $\mathbf{u}^+ \in {}^n$ defined by*

$$\mathbf{u}^+ \in \operatorname{prox}_{\sigma t}(\mathbf{u} - t\nabla\Psi(\mathbf{u}))$$

we have

$$\Psi(\mathbf{u}) + \sigma(\mathbf{u}) - \Psi(\mathbf{u}^+) - \sigma(\mathbf{u}) \geq \frac{1 - tL_\Psi}{2t}\|\mathbf{u}^+ - \mathbf{u}\|^2.$$

By Lemma 59 we have that

$$\mathscr{F}(z^k) - \mathscr{F}(z^{k+1}) = f(z^k) + \lambda g(z^k) - f(z^{k+1}) - \lambda g(z^{k+1})) \geq \frac{1 - L\eta_k}{2\eta_k}\|z^{k+1} - z^k\|^2. \tag{B.14}$$

Hence $\{f(z^k) + \lambda g(z^k)\}_{k\geq 0}$ is a non-increasing sequence that strictly decreasing unless a critical point is obtained in a finite number of steps. By summing (B.14) over $k = 0, 1, \ldots, K$ and using the fact that $\{f(z^k) + \lambda g(z^k)\}_{k\geq 0}$ is non-increasing and is bounded below by $\mathscr{F}_*$, we obtain that

$$\begin{aligned}\mathscr{F}(z^0) - \mathscr{F}_* &\geq \sum_{k=0}^{K} \frac{1 - L\eta_k}{2\eta_k}\|z^{k+1} - z^k\|^2 \\ &\geq \frac{1}{2}(c - L)K \min_{k=0,\ldots,K} \|z^{k+1} - z^k\|_2^2.\end{aligned}$$

Consequently,

$$\min_{k=0,\dots,K} \|z^{k+1} - z^k\|_2 \leq \sqrt{\frac{2(\mathscr{F}(z^0) - \mathscr{F}_*)}{(c-L)K}}.$$

□

B.3.2 Proof of Theorem 24

Theorem. *Let $f_{V,W}(x) = V^T\sigma(Wx)$ be a network such that the derivative of the activation σ is globally bounded between zero and one, and let $P_1(V,W) = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^p |W_{ij}V_{ik}|$ be its 1-path norm. The Lipschitz constant $L_{V,W}$ of the network is bounded as follows:*

$$L_{V,W} \leq P_1(V,W) \leq \|V^T\|_{\infty,1}\|W\|_\infty.$$

Proof. We prove the result in the single-output case, which is trivially extended to the general case. Because the output space is $\mathbb{R}$, the ℓ_1-norm is just the absolute value of the output. In this case the Lipschitz constant of the single-output function $f_{V,W}$ is equal to the supremum of the ℓ_1-norm of its gradient, over its domain (c.f., Latorre et al. (2020b, Theorem 1)).

$$\begin{aligned}
L_{V,W} &= \sup_x \|\nabla f_{V,W}(x)\|_1 \\
&= \sup_x \sup_{\|t\|_\infty \leq 1} t^T \nabla h_{V,W}(x) \\
&= \sup_x \sup_{\|t\|_\infty \leq 1} t^T W^T \sigma'(Wx) V \\
&\leq \sup_{0\leq s\leq 1} \sup_{\|t\|_\infty \leq 1} t^T W^T \operatorname{Diag}(s) V \\
&= \sup_{0\leq s\leq 1} \sup_{\|t\|_\infty \leq 1} \sum_{i=1}^n \sum_{j=1}^m t_i (W^T \operatorname{Diag}(V))_{i,j} s_j \\
&\leq \sum_{i=1}^n \sum_{j=1}^m \sup_{0\leq s_j\leq 1} \sup_{-1\leq t_i\leq 1} t_i (W^T \operatorname{Diag}(V))_{i,j} s_j \\
&= \sum_{i=1}^n \sum_{j=1}^m |W^T \operatorname{Diag}(V)|_{i,j} = \sum_{i=1}^n \sum_{j=1}^m |W_{i,j} V_{i,1}|
\end{aligned}$$

This shows the first inequality. We now show the second inequality. Denote the i-th row of the

matrix W as w_i:

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{m}|W_{i,j}V_{i,1}| &= \sum_{i=1}^{n}|V_{i,1}|\sum_{j=1}^{m}|W_{i,j}| \\ &= \sum_{i=1}^{n}|V_{i,1}|\|w_i\|_1 \\ &\leq \sum_{i=1}^{n}|V_{i,1}|\max_{j=1,\dots,m}\|w_j\|_1 \\ &= \sum_{i=1}^{n}|V_{i,1}|\|W\|_\infty \\ &= \|V\|_1\|W\|_\infty\end{aligned}$$

In the fourth line we have used the fact that the ℓ_∞ operator norm of a matrix is equal to the maximum ℓ_1-norm of the rows.

□

B.3.3 Proof of Lemma 26

Lemma. *Let* $(v^*, \boldsymbol{w}^*) \in \mathbb{R}_+ \times \mathbb{R}_+^n$ *be an optimal solution of* (3.25). *Then* $(sign(x)\cdot v^*, sign(\boldsymbol{y})\circ \boldsymbol{w}^*)$ *is an optimal solution of problem* (3.24).

Proof. Let $\tilde{h}_\lambda(v,\mathbf{w};x,\mathbf{y})$ denote the objective function of problem (3.24). We have that

$$\begin{aligned}\tilde{h}_\lambda(v,\mathbf{w};x,\mathbf{y}) &\equiv \frac{1}{2}(v-x)^2 + \frac{1}{2}\sum_{j=1}^{m}(w_j-y_j)^2 + \lambda|v|\sum_{j=1}^{m}|w_j| \\ &= \frac{1}{2}(\text{sign}(x)\,v-|x|)^2 + \frac{1}{2}\sum_{j=1}^{m}(\text{sign}(y_j)\,w_j-|y_j|)^2 + \lambda|v|\sum_{j=1}^{m}|w_j| \\ &\geq \frac{1}{2}(|v|-|x|)^2 + \frac{1}{2}\sum_{j=1}^{m}(|w_j|-|y_j|)^2 + \lambda v\sum_{j=1}^{m}w_j \\ &\geq h_\lambda(v^*,\mathbf{w}^*;x,\mathbf{y}),\end{aligned}$$

where the last inequality follows from the fact that (v^*, w^*) is an optimal solution of (3.25). Since equality with the lower bound is attained by setting $(v,\mathbf{w}) = (\text{sign}(x)\cdot v^*, \text{sign}(\mathbf{y})\circ \mathbf{w}^*)$, we conclude that $(\text{sign}(x)\cdot v^*, \text{sign}(\mathbf{y})\circ \mathbf{w}^*)$ is an optimal solution of (3.24). □

B.3.4 Proof of Lemma 27

Lemma. *Let* $(v^*, \boldsymbol{w}^*) \in \mathbb{R}_+ \times \mathbb{R}_+^m$ *be an optimal solution of* (3.25) *for a given* $(x, \boldsymbol{y}) \in \mathbb{R} \times \mathbb{R}^m$. *Then*

$$w_j^* = \max\{0, |y_j| - \lambda v^*\} \quad \textit{for any } j = 1, 2, \ldots, m,$$
$$v^* = \max\left\{0, |x| - \lambda \sum_{j=1}^m w_j^*\right\}.$$

Proof. The stationarity (first-order) conditions of (3.25) (cf. (Beck, 2014, Ch. 9.1)) state that

$$\frac{\partial h_\lambda}{\partial v}(v^*, \mathbf{w}^*; x, \mathbf{y}) \begin{cases} = 0, & v^* > 0, \\ \geq 0, & v^* = 0, \end{cases} \quad \text{and} \quad \frac{\partial h_\lambda}{\partial w_j}(v^*, \mathbf{w}^*; x, \mathbf{y}) \begin{cases} = 0, & w_j^* > 0, \\ \geq 0, & w_j^* = 0, \end{cases}$$

which translates to

$$v^* - |x| + \lambda \sum_{j=1}^m w_j^* \begin{cases} = 0, & v^* > 0, \\ \geq 0, & v^* = 0, \end{cases} \quad \text{and} \quad w_j^* - |y_j| + \lambda v^* \begin{cases} = 0, & w_j^* > 0, \\ \geq 0, & w_j^* = 0, \end{cases}$$

and the required follows. □

B.3.5 Proof of Lemma 30

Lemma. *Let* $\bar{s} = \min(\lfloor \lambda^{-2} \rfloor, m)$. *For all integer* $s \in \{2, 3, \ldots, \bar{s}\}$, *we have that*

$$h_\lambda(v^{(s)}, \boldsymbol{w}^{(s)}; x, \boldsymbol{y}) < h_\lambda(v^{(s-1)}, \boldsymbol{w}^{(s-1)}; x, \boldsymbol{y}).$$

Proof. Recall that $h_\lambda(v, \mathbf{w}; x, \mathbf{y}) := \frac{1}{2}(v - |x|)^2 + \frac{1}{2}\sum_{j=1}^m (w_j - |y_j|)^2 + \lambda v \sum_{j=1}^m w_j$. By plugging $\mathbf{w}^{(s)}$ defined in (3.29) in h_λ we obtain

$$\begin{aligned} h_\lambda(v^{(s)}, \mathbf{w}^{(s)}; x, \mathbf{y}) &= \frac{1}{2}(v^{(s)} - |x|)^2 + \frac{1}{2}\sum_{i=1}^s (|y_i| - (|y_i| - \lambda v^{(s)}))^2 + \frac{1}{2}\sum_{i=s+1}^m |y_i|^2 + \lambda v^{(s)} \sum_{i=1}^s (|y_i| - \lambda v^{(s)}) \\ &= \frac{1}{2}(v^{(s)} - |x|)^2 + \frac{\lambda^2}{2} s (v^{(s)})^2 + \frac{1}{2}\|\mathbf{y}\|_2^2 - \frac{1}{2}\sum_{i=1}^s |y_i|^2 + \lambda v^{(s)} \sum_{i=1}^s |y_i| - \lambda^2 s (v^{(s)})^2. \end{aligned}$$

Consequently, plugging $v^{(s)}$, defined in (3.29), yields

$$
\begin{aligned}
h_\lambda(v^{(s)},\mathbf{w}^{(s)};x,\mathbf{y}) &= \frac{1}{2}\left(\frac{\lambda^2 s}{1-\lambda^2 s}|x| - \frac{\lambda}{1-\lambda^2 s}\sum_{i=1}^{s}|y_i|\right)^2 - \frac{\lambda^2 s}{2(1-\lambda^2 s)^2}\left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right)^2 \\
&\quad + \frac{\lambda}{1-\lambda^2 s}\sum_{i=1}^{s}|y_i|\left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right) - \frac{1}{2}\sum_{i=1}^{s}|y_i|^2 + \frac{1}{2}\|\mathbf{y}\|_2^2 \\
&= \frac{\lambda^2 s}{2(1-\lambda^2 s)^2}x^2(\lambda^2 s - 1) + \frac{\lambda^2}{2(1-\lambda^2 s)^2}\left(\sum_{i=1}^{s}|y_i|\right)^2(1-\lambda^2 s - 2(1-\lambda^2 s)) \\
&\quad + |x|\sum_{i=1}^{s}|y_i|\left(-\frac{\lambda^3 s}{(1-\lambda^2 s)^2} + \frac{\lambda^3 s}{(1-\lambda^2 s)^2} + \frac{\lambda}{1-\lambda^2 s}\right) - \frac{1}{2}\sum_{i=1}^{s}|y_i|^2 + \frac{1}{2}\|\mathbf{y}\|_2^2 \\
&= \frac{1}{2(1-\lambda^2 s)}\left(-\lambda^2 s x^2 - \left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right)^2 + x^2\right) - \frac{1}{2}\sum_{i=1}^{s}|y_i|^2 + \frac{1}{2}\|\mathbf{y}\|_2^2 \\
&= -\frac{1}{2(1-\lambda^2 s)}\left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right)^2 + \frac{1}{2}\|x\|_2^2 - \frac{1}{2}\sum_{i=1}^{s}|y_i|^2 + \frac{1}{2}\|\mathbf{y}\|_2^2 \\
&= -\left(1 + \frac{\lambda^2}{1-\lambda^2 s}\right)\frac{1}{2(1-\lambda^2(s-1))}\left(|x| - \lambda\sum_{i=1}^{s-1}|y_i| - \lambda|y_s|\right)^2 + \frac{1}{2}\|x\|_2^2 - \frac{1}{2}\sum_{i=1}^{s}|y_i|^2 + \frac{1}{2}\|\mathbf{y}\|_2^2 \\
&= h_\lambda(v^{(s-1)}, w^{(s-1)};x,y) - \frac{1}{2(1-\lambda^2 s+\lambda^2)}\left(-2\lambda|y_s|\left(|x| - \lambda\sum_{i=1}^{s-1}|y_i|\right) + \lambda^2|y_s|^2\right) \\
&\quad - \frac{\lambda^2}{2(1-\lambda^2 s)(1-\lambda^2 s+\lambda^2)}\left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right)^2 - \frac{1}{2}|y_s|^2.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&h_\lambda(v^{(s)},\mathbf{w}^{(s)};x,\mathbf{y}) - h_\lambda(v^{(s-1)}, w^{(s-1)};x,y) \\
&= -\frac{1}{2(1-\lambda^2 s+\lambda^2)}\left(-2\lambda|y_s|\left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right) - \lambda^2|y_s|^2 + \frac{\lambda^2}{1-\lambda^2 s}\left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right)^2 + (1-\lambda^2 s+\lambda^2)|y_s|^2\right) \\
&= -\frac{1}{2(1-\lambda^2 s+\lambda^2)}\left((1-\lambda^2 s)|y_s|^2 - 2\lambda|y_s|\left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right) + \frac{\lambda^2}{1-\lambda^2 s}\left(|x| - \lambda\sum_{i=1}^{s}|y_i|\right)^2\right) \\
&= -\frac{1-\lambda^2 s}{2(1-\lambda^2 s+\lambda^2)}\left(|y_s|^2 - 2\lambda|y_s|v^{(s)} + \lambda^2(v^{(s)})^2\right) \\
&= -\frac{1-\lambda^2 s}{2(1-\lambda^2 s+\lambda^2)}\left(|y_s| - \lambda v^{(s)}\right)^2 \le 0,
\end{aligned}
$$

meaning that

$$
h_\lambda(v^{(s)},\mathbf{w}^{(s)};x,\mathbf{y}) \le h_\lambda(v^{(s-1)},\mathbf{w}^{(s-1)};x,\mathbf{y}).
$$

□

B.3.6 Proof of Corollary 31

Corollary. *Suppose that there exists a non-trivial optimal solution of* (3.25). *Denote* $\bar{s} = \min(\lfloor \lambda^{-2} \rfloor, m)$ *and let*

$$s^* = \max\left\{ s \in \{0, \dots, \bar{s}\} : v^{(s)}, w_s^{(s)} > 0 \right\}.$$

Then $(v^{(s^*)}, \boldsymbol{w}^{(s^*)})$ *is an optimal solution of* (3.25).

Proof. By Lemma 27, $(v^{(s^*)}, \mathbf{w}^{(s^*)})$ is a stationary point of (3.25). Moreover, according to Corollary 28 and Lemma 29, $(v^{(s^*)}, \mathbf{w}^{(s^*)})$ belongs to the set of $\bar{s}$ stationary points that are candidates to be optimal solutions of (3.25). Invoking Lemma 30, we have that

$$h_\lambda(v^{(s^*)}, \mathbf{w}^{(s^*)}; x, \mathbf{y}) < h_\lambda(v^{(j)}, \mathbf{w}^{(j)}; x, \mathbf{y}), \quad \forall s^* > j. \tag{B.15}$$

Hence, $(v^{(j)}, \mathbf{w}^{(j)})$ is not an optimal solution for any $j < s^*$.

Let us now consider the complementary case. By Lemma 29, for any $i > \bar{s}$ the pair $(v^{(i)}, \mathbf{w}^{(i)})$ does not satisfy the second-order optimality conditions, and therefore is not an optimal solution. On the other hand, by the definition of s^*, for any $\bar{s} > i > s^*$ the pair $(v^{(i)}, w^{(i)})$ is not a feasible solution , and subsequently not a stationary point. To conclude, $h_\lambda(v^{(s^*)}, \mathbf{w}^{(s^*)}; x, \mathbf{y}) < h_\lambda(v^{(j)}, \mathbf{w}^{(j)}; x, \mathbf{y})$ holds for any $j \neq s^*$ such that $(v^{(j)}, \mathbf{w}^{(j)})$ is a stationary point, meaning that $(v^{(s^*)}, \mathbf{w}^{(s^*)})$ is an optimal solution of (3.25).

□

B.3.7 Proof of Lemma 32

Lemma. *For any* $k \in [\bar{s}]$, *we have*

$$v^{(k)} > 0, \boldsymbol{w}^{(k)} > 0 \Rightarrow v^{(i)} > 0, \boldsymbol{w}^{(i)} > 0, \ \ \forall i < k.$$

Proof. Suppose that $(v^{(k)}, \mathbf{w}^{(k)}) > 0$ for some $k \in \{2, \dots, \bar{s}\}$. By induction principle, it is sufficient to show that $(v^{(k-1)}, \mathbf{w}^{(k-1)})$ is feasible in order to prove the result.

By (3.29), we have:

$$(1 - k\lambda^2)\, v^{(k)} = |x| - \lambda \sum_{j=1}^{k} |y_j| = (1 - k\lambda^2 + \lambda^2)\, v^{(k-1)} - |y_k|.$$

which implies

$$v^{(k-1)} = \frac{1}{(1 - k\lambda^2 + \lambda^2)} ((1 - k\lambda^2)\, v^{(k)} + |y_k|) \geq 0.$$

For $\mathbf{w}^{(k)}$, it is easy to see from (3.29) that, since the vector $\mathbf{y}$ is sorted in decreasing order of magnitude, the vector $\mathbf{w}^{(k)}$ is also sorted in decreasing order, and thus $\mathbf{w}^{(k)} > 0$ if and only if

$w_k^{(k)} > 0$.

$$\begin{aligned}(1-k\lambda^2)\,w_k^{(k)} &= (1-k\lambda^2)|y_k| - \lambda|x| + \lambda^2 \sum_{j=1}^{k} |y_j| \\ &= -\lambda|x| + (1-(k-1)\lambda^2)|y_{k-1}| + \lambda^2 \sum_{j=1}^{k-1} |y_j| + \lambda^2|y_k| + (1-k\lambda^2)|y_k| - (1-(k-1)\lambda^2)|y_{k-1}| \\ &= (1-(k-1)\lambda^2)\,w_{k-1}^{(k-1)} + (1-k\lambda^2+\lambda^2)(|y_k| - |y_{k-1}|),\end{aligned}$$

where the last line uses the identity of the first line for $k-1$. We thus have:

$$w_{k-1}^{(k-1)} = \frac{1}{(1-(k-1)\lambda^2)}(1-k\lambda^2)\,w_k^{(k)} + |y_{k-1}| - |y_k| > 0,$$

since $|y_{k-1}| \geq |y_k|$ and $k < \lambda^{-2}$.

□

B.3.8 Proof of Lemma 34

Lemma. *Suppose that* $|z_1| \geq |z_2| \geq \cdots \geq |z_d|$, *and let* $\boldsymbol{w}^*$ *be an optimal solution of* (3.33). *Then*

$$w_1^* \geq w_2^* \geq \cdots \geq w_d^*. \tag{B.16}$$

Proof. Assume the contrary, that there exist an optimal solution of (3.33) such that (3.33) does not hold. Without loss of generality, suppose that $w_1^* < w_2^*$, and consider the solution $\tilde{\mathbf{w}}$ given by

$$\tilde{w}_i = \begin{cases} w_i^*, & i = 3,4,\ldots,L, \\ w_1^*, & i = 2, \\ w_2^*, & i = 1. \end{cases}$$

Then by the optimality of $\mathbf{w}^*$, and our assumptions that $|z_1| \geq |z_2|$ and $w_1^* < w_2^*$, we obtain

$$\begin{aligned}0 &\geq \frac{1}{2}\sum_{i=1}^{d}(w_i^* - |z_i|)^2 + \lambda\, w_1^* \cdot w_2^* \cdots w_d^* - \frac{1}{2}\sum_{i=1}^{l}(\tilde{w}_i - |z_i|)^2 - \lambda\, \tilde{w}_1 \cdot \tilde{w}_2 \cdots \tilde{w}_d \\ &= \frac{1}{2}\left[(w_1^* - |z_1|)^2 + (w_2^* - |z_2|)^2 - (\tilde{w}_1 - |z_1|)^2 - (\tilde{w}_2 - |z_2|)^2\right] \\ &= \frac{1}{2}\left[(w_1^* - |z_1|)^2 + (w_2^* - |z_2|)^2 - (w_2^* - |z_1|)^2 - (w_1^* - |z_2|)^2\right] \\ &= (|z_1| - |z_2|)(w_2^* - w_1^*) > 0,\end{aligned}$$

which is a contradiction. □

B.3.9 Proof of Lemma 38

Lemma. *Let $\boldsymbol{w}^*$ be an optimal solution of* (3.33) *(such that $\boldsymbol{w}^* > 0$). Then:*

1. *For any $i = 1, 2, \ldots, d-1$, the element w_i^* satisfies that*

$$\frac{1}{2}|z_i| + \frac{1}{2}\sqrt{|z_i|^2 - |z_d|^2} \leq w_i^* \leq |z_i|$$

2. *For any $i = 2, \ldots, d-1$, the element w_i^* satisfies that*

$$w_i^* = \frac{1}{2}\left(|z_i| + \sqrt{|z_i|^2 - 4w_1^*(|z_1| - w_1^*)}\right),$$

and

$$w_d^* = |z_L| - \lambda w_1^* \cdot w_2^* \cdots w_{d-1}^*.$$

3. *It holds that*

$$w_1^* = |z_1| - \frac{\lambda}{2^{d-2}}\left(|z_d| - \frac{\lambda}{2^{d-2}} w_1^* \prod_{i=2}^{d-1}\left(|z_i| + \sqrt{|z_i|^2 - 4w_1^*(|z_1| - w_1^*)}\right)\right)$$
$$\cdot \prod_{i=2}^{d-1}\left(|z_i| + \sqrt{|z_i|^2 - 4w_1^*(|z_1| - w_1^*)}\right).$$

Proof. 1. First, it is obvious that $w_i^* \leq |z_i|$ for any $i = 1, 2, \ldots, d$. For the lower bound, notice that, thanks to Corollary 37, it holds that for any $i = 1, 2, \ldots, d-1$,

$$w_i^*(|z_i| - w_i^*) = w_d^*(|z_d| - w_d^*) \leq \frac{|z_d|^2}{4}$$

which implies that:

$$w_i^* \leq \frac{1}{2}\left(|z_i| - \sqrt{|z_i|^2 - |z_d|^2}\right) \text{ or } w_i^* \geq \frac{1}{2}\left(|z_i| + \sqrt{|z_i|^2 - |z_d|^2}\right). \tag{B.17}$$

We now argue that for $i = 1, \ldots, d-1$, we have $w_i^* \geq \frac{|z_i|}{2}$, and hence, the right inequality of (B.17) must hold. To this end, for some $i = 1, \ldots, d-1$, define α as the value achieving

$$w_i^*(|z_i| - w_i^*) = w_d^*(|z_d| - w_d^*) = \alpha.$$

By solving the above quadratic equations, we have

$$w_i^* = \frac{1}{2}\left(|z_i| \pm \sqrt{z_i^2 - 4\alpha}\right)$$
$$w_d^* = \frac{1}{2}\left(|z_d| \pm \sqrt{z_d^2 - 4\alpha}\right).$$

Consider the solution $w_i^* = \frac{1}{2}\left(|z_i| - \sqrt{z_i^2 - 4\alpha}\right)$. Whatever the solution for w_d^*, we see

that $w_d^* \geq \frac{1}{2}\left(|z_d| - \sqrt{z_d^2 - 4\alpha}\right)$. Hence,

$$\begin{aligned} 2(w_i^* - w_d^*) &\leq |z_i| - |z_d| + \sqrt{z_d^2 - 4\alpha} - \sqrt{z_i^2 - 4\alpha} \\ &= |z_i| - |z_d| + \frac{z_d^2 - z_i^2}{\sqrt{z_i^2 - 4\alpha} + \sqrt{z_d^2 - 4\alpha}} \\ &= (|z_i| - |z_d|)\left(1 - \frac{|z_i| + |z_d|}{\sqrt{z_i^2 - 4\alpha} + \sqrt{z_d^2 - 4\alpha}}\right) < 0 \end{aligned}$$

since $|z_i| > |z_d|$ by assumption. Hence, we have $w_i^* < w_d^*$, which contradicts the ordering property given by Lemma 32. Therefore, we must have $w_i^* = \frac{1}{2}\left(|z_i| + \sqrt{z_i^2 - 4\alpha}\right) \geq \frac{|z_i|}{2}$ as desired.

2. The equalities for $i = 1, \ldots, d-1$ follows from Corollary 37 by finding roots of the order two polynomial together with the fact that $w_i^* \geq \frac{|z_i|}{2}$ (from point 1.) which excludes one of the two solutions. The equality for w_d^* trivially follows from Lemma 36.

3. By Lemma 36, we have that

$$|z_1| = w_1^* + \lambda w_2^* \cdot w_3^* \cdots w_d^*.$$

The result follows by plugging the expressions of w_i, $i = 2, \ldots, d$ in terms of w_1^* obtained in part 2.

□

B.3.10 Proof of Lemma 42

Lemma. *Two paths $(s_1, \ldots, s_{d+1})$ and $(s'_1, \ldots, s'_{d+1})$ are called non-overlapping if for all $0 \leq i \leq d$, $s_i = s'_i$ implies $s_{i+1} \neq s'_{i+1}$. Let T be a set of non-overlapping paths in a network with layer sizes $n_1, \ldots, n_{d+1}$. It holds that $|T| \leq k^\star \equiv \min_{i=1}^{d} n_i n_{i+1}$.*

Proof. In essence, Lemma 42 states that we can sample a number of non-overlapping paths at most equal to the size of the smallest weight matrix in the network. Consider for each $i = 1, \ldots, d$ the map $\psi_i : T \to [n_i] \times [n_{i+1}]$ defined as $\psi_i(s_1, \ldots, s_{d+1}) = (s_i, s_{i+1})$. The non-overlapping property implies that all such maps are injective. Thus, $|T| \leq n_i n_{i+1}$ for each $i = 1, \ldots, d$ which implies that $|T| \leq \min_{i=1,\ldots,d} n_i n_{i+1}$. □

B.3.11 Proof of Theorem 43

Theorem. *Let $\mathcal{T}_{max}$ denote the distribution over sets of paths generated by Algorithm 10. Then, $\mathcal{T}_{max}$ satisfies Assumption 40, and generates sets of paths with maximal cardinality*

$k^\star = \min_{i=1}^{d} n_i n_{i+1}$.

Proof. We only prove the unbiased estimation property of the sampler. Let M be the matrix output by Algorithm 10. We need to show that for any path $(s_1, \ldots, s_{d+1})$,

$$\text{Prob}\{(s_1, \ldots, s_{d+1}) \in M\} = \frac{d_{i^\star} d_{i^\star+1}}{\prod_{i=1}^{d+1} n_i} \tag{B.18}$$

where recall that $i^\star = \arg\min_{i=1}^{d} n_i n_{i+1}$ and we abuse the notation and write that $(s_1, \ldots, s_{d+1}) \in M$ if the path $(s_1, \ldots, s_{d+1})$ appears as one of the rows of the matrix M.

We will show that (B.18) holds by induction on the length d of the path. We will denote as $M[d]$ as the matrix obtained from M by taking only its first $d+1$ columns. Note that we will do induction on d while keeping the value of $d_{i^\star}$ fixed.

Denote by U_i the sets of vertices sampled uniformly in line 7 of Algorithm 10. Without loss of generality we will assume that $i^\star$ is even. By this assumption we have that $|U_i| = n_{i^\star}$ if i is even and $|U_i| = n_{i^\star+1}$ if i is odd.

For the base case when $d = 1$ we have that

$$\text{Prob}\{(s_1, s_2) \in M[1]\} = \text{Prob}(s_1 \in U_1)\text{Prob}(s_2 \in U_2) = \frac{n_{i^\star} n_{i^\star+1}}{n_1 n_2} \tag{B.19}$$

where in the first equality we have used the independence of the random sets U_1, U_2, as well as the fact that our algorithm connects all neurons sampled in the input layer and those sampled in the next layer. Now we proceed by induction as follows

$$\text{Prob}\{(s_1, \ldots, s_{d+1}) \in M[d]\} = \text{Prob}\{(s_1, \ldots, s_d) \in M[d-1]\} \cdot \Psi(s_{d+1}) \tag{B.20}$$

$$\text{where } \Psi(s_{d+1}) := \text{Prob}\{s_{d+1} \in U_{d+1} \text{ and it gets assigned next to } (s_1, \ldots, s_d) \text{ in line 10}\} \tag{B.21}$$

where the equality is due again to independence of the sampling at each layer. We need only compute $\Psi(s_{d+1})$. Assume that $d+1$ is even, the other case is analogous. The probability that $s_{d+1} \in U_{d+1}$ is $n_{i^\star}/n_{d+1}$ and then it gets assigned uniformly at random in the block of $n_{i^\star}$ paths that end in s_d. Hence, the probability that it gets assigned at a particular row is $1/n_{i^\star}$, and we have that $\Psi(s_{d+1}) = n_{i^\star}/(n_{d+1} n_{i^\star}) = 1/n_{d+1}$. Plugging this value in (B.20) yields the desired result (B.18).

□

C Appendix for Chapter 4

C.1 Appendix for Section 4.2: Additional experiments

We show here additional synthetic experiments. Tables C.1, C.2 and C.3 show the results for additive noise model with Gumbel noise on Erdös-Renyi graphs. Tables C.4, C.5 and C.5 show the results for Gaussian noise with Scale-free graphs.

Table C.1 – Synthetic experiment for $d = 10$ with Gumbel noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	$\mathbf{1.1 \pm 1.2}$	$\mathbf{4.5 \pm 5.0}$	$\mathbf{0.4 \pm 0.5}$	$\mathbf{21.7 \pm 2.9}$	$\mathbf{35.3 \pm 7.4}$	$\mathbf{0.3 \pm 0.4}$
CAM	2.0 ± 1.5	6.1 ± 5.8	1.6 ± 0.8	27.2 ± 1.8	48.9 ± 9.0	3.8 ± 2.5
GraN-DAG	2.1 ± 1.9	9.7 ± 10.4		22.9 ± 3.2	43.2 ± 11.7	
SELF	8.8 ± 2.7	37.0 ± 8.9	–	38.9 ± 1.2	85.9 ± 5.0	–
GES	7.6 ± 2.4	29.6 ± 11.5	–	34.9 ± 3.5	81.9 ± 5.3	–
VarSort			1.9 ± 0.8			8.2 ± 3.0

Table C.2 – Synthetic experiment for $d = 20$ with Gumbel noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	$\mathbf{3.3 \pm 2.6}$	$\mathbf{12.0 \pm 11.5}$	$\mathbf{0.7 \pm 0.9}$	$\mathbf{52.9 \pm 4.4}$	$\mathbf{205.5 \pm 35.5}$	$\mathbf{5.1 \pm 1.6}$
CAM	5.8 ± 1.5	24.6 ± 13.0	3.0 ± 2.0	57.1 ± 4.2	230.0 ± 39.3	10.7 ± 5.8
GraN-DAG	7.4 ± 2.5	29.2 ± 11.3		54.9 ± 4.3	239.5 ± 43.6	
SELF	19.2 ± 2.1	96.2 ± 27.9	–	77.7 ± 1.4	342.9 ± 15.2	–
GES	19.0 ± 3.9	84.0 ± 32.7	–	72.7 ± 4.2	323.2 ± 28.5	–
VarSort			3.8 ± 1.7			20.8 ± 6.6

Table C.3 – Synthetic experiment for $d = 50$ with Gumbel noise

	ER1			ER4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	$\mathbf{11.3 \pm 4.6}$	$\mathbf{68.2 \pm 45.1}$	4.1 ± 2.5	$\mathbf{132.6 \pm 8.0}$	1390 ± 132	$\mathbf{19.7 \pm 3.4}$
CAM	$\mathbf{11.0 \pm 3.7}$	69.7 ± 48.8	–	141.1 ± 6.7	$\mathbf{1350 \pm 137}$	–
GraN-DAG	22.5 ± 4.2	167.1 ± 47.3	–	139.9 ± 7.0	1552 ± 143	–
SELF	46.3 ± 3.7	306.5 ± 41.1	–	193.3 ± 3.1	2100 ± 102	–
GES	51.0 ± 5.1	273.0 ± 57.9	–	182.1 ± 3.2	2012 ± 105	–
VarSort	–	–	8.8 ± 1.6	–	–	45.5 ± 8.0

Table C.4 – Synthetic experiment for $d = 10$ with Gaussian noise on scale free graphs

	SF1			SF4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	$\mathbf{0.3 \pm 0.6}$	$\mathbf{2.7 \pm 5.8}$	$\mathbf{0.1 \pm 0.3}$	$\mathbf{4.6 \pm 1.7}$	$\mathbf{21.5 \pm 9.6}$	$\mathbf{0.5 \pm 0.9}$
CAM	0.4 ± 0.5	2.8 ± 3.6	0.3 ± 0.3	9.6 ± 2.0	40.4 ± 11.4	4.1 ± 1.6
GraN-DAG	1.4 ± 1.0	12.5 ± 9.7	–	4.7 ± 1.8	23.0 ± 7.3	–
SELF	10.4 ± 2.7	60.2 ± 16.2	–	26.8 ± 1.4	84.6 ± 3.6	–
GES	12.5 ± 3.3	57.2 ± 15.2	–	22.7 ± 4.1	76.6 ± 7.2	–
VarSort	–	–	2.8 ± 1.7	–	–	7.0 ± 3.2

Table C.5 – Synthetic experiment for $d = 20$ with Gaussian noise on scale free graphs

	SF1			SF4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	$\mathbf{0.9 \pm 0.9}$	13.8 ± 12.6	0.7 ± 0.6	17.5 ± 3.5	179.2 ± 23.8	$\mathbf{3.6 \pm 1.4}$
CAM	$\mathbf{0.9 \pm 0.9}$	$\mathbf{12.9 \pm 14.0}$	$\mathbf{0.5 \pm 0.4}$	26.4 ± 3.9	253.7 ± 28.8	4.6 ± 3.2
GraN-DAG	3.2 ± 1.9	25.5 ± 15.6	–	$\mathbf{14.7 \pm 4.0}$	$\mathbf{168.0 \pm 39.2}$	–
SELF	18.9 ± 2.9	245.7 ± 28.2	–	65.9 ± 2.6	369.2 ± 7.8	–
GES	23.6 ± 3.6	166.4 ± 47.6	–	60.0 ± 4.0	345.7 ± 11.4	–
VarSort	–	–	7.4 ± 2.5	–	–	20.2 ± 7.2

Table C.6 – Synthetic experiment for $d = 50$ with Gaussian noise on scale free graphs

	SF1			SF4		
	SHD	SID	$D_{top}(\pi, A)$	SHD	SID	$D_{top}(\pi, A)$
SCORE (ours)	4.6 ± 2.4	132.6 ± 75.8	4.0 ± 1.0	68.3 ± 3.6	1724 ± 109	21.8 ± 5.0
CAM	$\mathbf{3.6 \pm 1.9}$	$\mathbf{115.4 \pm 72.6}$	–	85.3 ± 4.2	1935 ± 99	–
GraN-DAG	9.2 ± 3.3	281.8 ± 129.8	–	$\mathbf{63.8 \pm 9.7}$	$\mathbf{1677 \pm 118}$	–
SELF	57.6 ± 6.6	1780 ± 150	–	176.0 ± 4.0	2424 ± 16	–
GES	81.3 ± 8.8	1049 ± 174	–	167.6 ± 9.2	2289 ± 49	–
VarSort	–	–	21.0 ± 4.0	–	–	73.0 ± 10.6

Bibliography

Ahn, S., Korattikara, A., and Welling, M. (2012). Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv preprint arXiv:1206.6380*.

Andersen, C. M. and Bro, R. (2010). Variable selection in regression—a tutorial. *Journal of chemometrics*, 24(11-12):728–737.

Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. (2010). Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-lojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457.

Attouch, H., Bolte, J., and Svaiter, B. F. (2011). Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1-2):91–129.

Aybat, N. S., Fallah, A., Gurbuzbalaban, M., and Ozdaglar, A. (2019). A universally optimal multistage accelerated stochastic gradient method. *arXiv preprint arXiv:1901.08022*.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). Optimization with sparsity-inducing penalties. *Found. Trends Mach. Learn.*, 4(1):1–106.

Badue, C., Guidolini, R., Carneiro, R. V., Azevedo, P., Cardoso, V. B., Forechi, A., Jesus, L., Berriel, R., Paixao, T. M., Mutz, F., et al. (2021). Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816.

Bakry, D., Barthe, F., Cattiaux, P., and Guillin, A. (2008). A simple proof of the poincaré inequality for a large class of probability measures. *Electronic Communications in Probability*, 13:60–66.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439):509–512.

Barp, A., Briol, F.-X., Duncan, A. B., Girolami, M., and Mackey, L. (2019). Minimum stein discrepancy estimators. *arXiv preprint arXiv:1906.08283*.

Beck, A. (2014). *Introduction to nonlinear optimization*, volume 19 of *MOS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena scientific Belmont.

Bobkov, S. G. (1999). Isoperimetric and analytic inequalities for log-concave probability measures. *The Annals of Probability*, 27(4):1903–1921.

Bolte, J., Sabach, S., and Teboulle, M. (2013). Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494.

Bouckaert, R. R. (1992). Optimizing causal orderings for generating dags from data. In *Uncertainty in Artificial Intelligence*, pages 9–16. Elsevier.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Brosse, N., Durmus, A., Moulines, É., and Pereyra, M. (2017). Sampling from a log-concave distribution with compact support with proximal langevin monte carlo. *arXiv preprint arXiv:1705.08964*.

Bubeck, S., Eldan, R., and Lehec, J. (2018). Sampling from a log-concave distribution with projected langevin monte carlo. *Discrete & Computational Geometry*, 59(4):757–783.

Bühlmann, P. (2020). Invariance, causality and robustness. *Statistical Science*, 35(3):404–426.

Bühlmann, P., Peters, J., and Ernest, J. (2014). Cam: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556.

Cai, R., Qiao, J., Zhang, Z., and Hao, Z. (2018). Self: structural equational likelihood framework for causal discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Chen, C., Ding, N., and Carin, L. (2015). On the convergence of stochastic gradient mcmc algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pages 2278–2286.

Cheng, X. and Bartlett, P. (2017). Convergence of langevin mcmc in kl-divergence. *arXiv preprint arXiv:1705.09048*.

Cheng, X., Chatterji, N. S., Bartlett, P. L., and Jordan, M. I. (2017a). Underdamped langevin mcmc: A non-asymptotic analysis. *arXiv preprint arXiv:1707.03663*.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2017b). A survey of model compression and acceleration for deep neural networks.

Chewi, S., Erdogdu, M. A., Li, M. B., Shen, R., and Zhang, M. (2021). Analysis of langevin monte carlo from poincar\'e to log-sobolev. *arXiv preprint arXiv:2112.12662*.

Chickering, D. M. (1996). Learning bayesian networks is np-complete. In *Learning from data*, pages 121–130. Springer.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 854–863, International Convention Centre, Sydney, Australia. PMLR.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical japanese literature.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv e-prints*, page arXiv:1511.07289.

Combettes, P. L. and Pesquet, J.-C. (2019). Lipschitz Certificates for Neural Network Structures Driven by Averaged Activation Operators. *arXiv e-prints*, page arXiv:1903.01014.

Condat, L. (2016). Fast projection onto the simplex and the l1 ball. *Math. Program.*, 158(1–2):575–585.

Cooper, G. F. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65.

Dai, Z., Liu, H., Le, Q. V., and Tan, M. (2021). Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977.

Dalalyan, A. S., Karagulyan, A., and Riou-Durand, L. (2019). Bounding the error of discretized langevin algorithms for non-strongly log-concave targets. *arXiv preprint arXiv:1906.08530*.

Dalalyan, A. S. and Karagulyan, A. G. (2017). User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *arXiv preprint arXiv:1710.00095*.

Daskalakis, C., Skoulakis, S., and Zampetakis, M. (2021). The complexity of constrained min-max optimization. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1466–1478.

Davis, D. and Drusvyatskiy, D. (2019). Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. (2017). Openai baselines. https://github.com/openai/baselines.

Diaconis, P., Stein, C., Holmes, S., and Reinert, G. (2004). Use of exchangeable pairs in the analysis of simulations. In *Stein's Method*, pages 1–25. Institute of Mathematical Statistics.

Dong, Y., Fu, Q.-A., Yang, X., Pang, T., Su, H., Xiao, Z., and Zhu, J. (2020). Benchmarking adversarial robustness on image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 321–331.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML 2008, page 272–279, New York, NY, USA. Association for Computing Machinery.

Durmus, A., Majewski, S., and Miasojedow, B. (2018a). Analysis of langevin monte carlo via convex optimization. *arXiv preprint arXiv:1802.09188*.

Durmus, A., Moulines, E., et al. (2017). Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587.

Durmus, A., Moulines, E., and Pereyra, M. (2018b). Efficient bayesian computation by proximal markov chain monte carlo: when langevin meets moreau. *SIAM Journal on Imaging Sciences*, 11(1):473–506.

Dwivedi, R., Chen, Y., Wainwright, M. J., and Yu, B. (2018). Log-concave sampling: Metropolis-hastings algorithms are fast! *arXiv preprint arXiv:1801.02309*.

Eldar, Y. C. and Kutyniok, G. (2012). *Compressed sensing: theory and applications*. Cambridge university press.

Erdös, P. and Rényi, A. (2011). *On the evolution of random graphs*, pages 38–82. Princeton University Press.

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. J. (2019). Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. *arXiv e-prints*, page arXiv:1906.04893.

Fercoq, O. and Richtárik, P. (2015). Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023.

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030.

Garreau, D., Jitkrittum, W., and Kanagawa, M. (2017). Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*.

Ge, R., Kakade, S. M., Kidambi, R., and Netrapalli, P. (2019). The step decay schedule: A near optimal, geometrically decaying learning rate procedure. *arXiv preprint arXiv:1904.12838*.

Ghoshal, A. and Honorio, J. (2018). Learning linear structural equation models in polynomial time and sample complexity. In *International Conference on Artificial Intelligence and Statistics*, pages 1466–1475. PMLR.

Gibbs, A. L. and Su, F. E. (2002). On choosing and bounding probability metrics. *International statistical review*, 70(3):419–435.

Gidel, G., Berard, H., Vignoud, G., Vincent, P., and Lacoste-Julien, S. (2018). A variational inequality perspective on generative adversarial networks. *arXiv preprint arXiv:1802.10551*.

Gozlan, N. (2010). Poincaré inequalities and dimension free concentration of measure. In *Annales de l'IHP Probabilités et statistiques*, volume 46, pages 708–739.

Gozlan, N. and Léonard, C. (2010). Transport inequalities. a survey. *arXiv preprint arXiv:1003.3852*.

Gromov, M. and Milman, V. D. (1983). A topological application of the isoperimetric inequality. *American Journal of Mathematics*, 105(4):843–854.

Hallak, N., Mertikopoulos, P., and Cevher, V. (2021). Regret minimization in stochastic non-convex learning via a proximal-gradient approach. In *To appear in Proceedings of the 38th International Conference on Machine Learning*. PMLR.

Han, S., Mao, H., and Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *International Conference on Learning Representations*, abs/1510.00149.

Handelman, D. (1988). Representing polynomials by positive linear functions on compact convex polyhedra. *Pacific J. Math.*, 132(1):35–62.

Hanson, S. J. and Pratt, L. Y. (1989). Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems 1*, pages 177–185. Morgan-Kaufmann.

Hastie, T. and Tibshirani, R. (1987). Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hsieh, Y.-P., Kavis, A., Rolland, P., and Cevher, V. (2018). Mirrored langevin dynamics. In *Advances in Neural Information Processing Systems*, pages 2883–2892.

Hsieh, Y.-P., Liu, C., and Cevher, V. (2019). Finding mixed nash equilibria of generative adversarial networks. In *International Conference on Machine Learning*, pages 2810–2819.

Hyvärinen, A. and Dayan, P. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4).

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32.

Jiang*, Y., Neyshabur*, B., Krishnan, D., Mobahi, H., and Bengio, S. (2020). Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*.

Jin, M. and Lavaei, J. (2018). Stability-certified reinforcement learning: A control-theoretic perspective. *arXiv e-prints*, page arXiv:1810.11505.

Kamalaruban, P., Huang, Y.-T., Hsieh, Y.-P., Rolland, P., Shi, C., and Cevher, V. (2020). Robust reinforcement learning via adversarial training with langevin dynamics. *Advances in Neural Information Processing Systems*, 33:8127–8138.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Krivine, J.-L. (1964). Anneaux préordonnés. *Journal d'analyse mathématique*, 12:p. 307–326.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. (2019). Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*.

Larranaga, P., Kuijpers, C. M., Murga, R. H., and Yurramendi, Y. (1996). Learning bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, 26(4):487–493.

Lasserre, J. B. (2000). Convergent lmi relaxations for nonconvex quadratic programs. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, volume 5, pages 5041–5046 vol.5.

Lasserre, J. B. (2006). Convergent sdp-relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, 17(3):822–843.

Lasserre, J. B. (2015). *An Introduction to Polynomial and Semi-Algebraic Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press.

Latorre, F., Rolland, P., and Cevher, V. (2020a). Lipschitz constant estimation of neural networks via sparse polynomial optimization. *arXiv preprint arXiv:2004.08688*.

Latorre, F., Rolland, P., and Cevher, V. (2020b). Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *International Conference on Learning Representations*.

Latorre, F., Rolland, P., Hallak, N., and Cevher, V. (2020c). Efficient proximal mapping of the 1-path-norm of shallow networks. In *International Conference on Machine Learning*, pages 5651–5661. PMLR.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Lehec, J. (2021). The langevin monte carlo algorithm in the non-smooth log-concave case. *arXiv preprint arXiv:2101.10695*.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.

Li, C., Chen, C., Carlson, D., and Carin, L. (2016a). Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Li, W., Ahn, S., and Welling, M. (2016b). Scalable mcmc for mixed membership stochastic blockmodels. In *Artificial Intelligence and Statistics*, pages 723–731.

Li, Y. and Turner, R. E. (2017). Gradient estimators for implicit models. *arXiv preprint arXiv:1705.07107*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, Q., Lu, Z., and Xiao, L. (2014). An accelerated proximal coordinate gradient method. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3059–3067, Cambridge, MA, USA. MIT Press.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings*. Elsevier.

Liu, Q., Lee, J., and Jordan, M. (2016). A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR.

Loh, P.-L. and Bühlmann, P. (2014). High-dimensional learning of linear causal networks via inverse covariance estimation. *The Journal of Machine Learning Research*, 15(1):3065–3105.

Lovász, L. and Vempala, S. (2007). The geometry of logconcave functions and sampling algorithms. *Random Structures & Algorithms*, 30(3):307–358.

Luu, T., Fadili, J., and Chesneau, C. (2017). Sampling from non-smooth distribution through langevin diffusion.

Ma, Y.-A., Chen, T., and Fox, E. (2015). A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, pages 2917–2925.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Marra, G. and Wood, S. N. (2011). Practical variable selection for generalized additive models. *Computational Statistics & Data Analysis*, 55(7):2372–2387.

Metel, M. and Takeda, A. (2019). Simple stochastic gradient methods for non-smooth non-convex regularized optimization. In *International Conference on Machine Learning*, pages 4537–4545.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Morimoto, J. and Doya, K. (2005). Robust reinforcement learning. *Neural computation*, 17(2):335–359.

Neumann, J. v. (1928). Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320.

Neyshabur, B., Tomioka, R., and Srebro, N. (2015). Norm-based capacity control in neural networks. In *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 1376–1401, Paris, France. PMLR.

Park, J. and Boyd, S. (2017). General heuristics for nonconvex quadratically constrained quadratic programming. *arXiv preprint arXiv:1703.07870*.

Patterson, S. and Teh, Y. W. (2013). Stochastic gradient riemannian langevin dynamics on the probability simplex. In *Advances in neural information processing systems*, pages 3102–3110.

Pearl, J. (2009). *Causality*. Cambridge university press.

Perolat, J., Scherrer, B., Piot, B., and Pietquin, O. (2015). Approximate dynamic programming for two-player zero-sum Markov games. In *International Conference on Machine Learning*.

Peters, J. and Bühlmann, P. (2015). Structural intervention distance for evaluating causal graphs. *Neural computation*, 27(3):771–799.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. (2014). Causal discovery with continuous additive noise models.

Pinsker, M. S. (1960). Information and information stability of random variables and processes.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. (2017). Robust adversarial reinforcement learning. In *International Conference on Machine Learning*.

Raghunathan, A., Steinhardt, J., and Liang, P. (2018a). Certified defenses against adversarial examples. In *International Conference on Learning Representations*.

Raghunathan, A., Steinhardt, J., and Liang, P. (2018b). Certified defenses against adversarial examples. In *International Conference on Learning Representations*.

Raskutti, G. and Uhler, C. (2018). Learning directed acyclic graph models based on sparsest permutations. *Stat*, 7(1):e183.

Reisach, A. G., Seiler, C., and Weichwald, S. (2021). Beware of the simulated dag! varsortability in additive noise models. *NeurIPS*.

Rolland, P., Cevher, V., Kleindessner, M., Russel, C., Schölkopf, B., Janzing, D., and Locatello, F. (2022). Score matching enables causal discovery of nonlinear additive noise models. *arXiv preprint arXiv:2203.04413*.

Rolland, P., Eftekhari, A., Kavis, A., and Cevher, V. (2020). Double-loop unadjusted langevin algorithm. In *International Conference on Machine Learning*, pages 8169–8177. PMLR.

Rosen, J. B. (1965). Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, pages 520–534.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Si, S., Hsieh, C.-J., and Dhillon, I. (2014). Memory efficient kernel approximation. In *International Conference on Machine Learning*, pages 701–709. PMLR.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

Singh, M. and Valtorta, M. (1993). An algorithm for the construction of bayesian network structures from data. In *Uncertainty in Artificial Intelligence*, pages 259–265. Elsevier.

Solus, L., Wang, Y., and Uhler, C. (2021). Consistency guarantees for greedy permutation-based causal inference algorithms. *Biometrika*, 108(4):795–814.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11918–11930.

Song, Y. and Ermon, S. (2020). Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*.

Song, Y., Garg, S., Shi, J., and Ermon, S. (2020a). Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. (2000). *Causation, prediction, and search*. MIT press.

Stein, C. (1972). A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the sixth Berkeley symposium on mathematical statistics and probability, volume 2: Probability theory*, volume 6, pages 583–603. University of California Press.

Stengle, G. (1974). A nullstellensatz and a positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207(2):87–97.

Strassen, V. (1969). Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Tessler, C., Efroni, Y., and Mannor, S. (2019). Action robust reinforcement learning and applications in continuous control. *arXiv preprint arXiv:1901.09184*.

Teyssier, M. and Koller, D. (2012). Ordering-based search: A simple and effective algorithm for learning bayesian networks. *arXiv preprint arXiv:1207.1429*.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE.

Tran-Dinh, Q., Pham, N. H., Phan, D. T., and Nguyen, L. M. (2021). A hybrid stochastic optimization framework for composite nonconvex optimization. *Mathematical Programming*, pages 1–67.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness may be at odds with accuracy. In *International Conference on Learning Representations*.

Van den Bulcke, T., Van Leemput, K., Naudts, B., van Remortel, P., Ma, H., Verschoren, A., De Moor, B., and Marchal, K. (2006). Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC bioinformatics*, 7(1):1–12.

Villani, C. (2009). Optimal transport–old and new, volume 338 of a series of comprehensive studies in mathematics.

Virmaux, A. and Scaman, K. (2018). Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems 31*, pages 3835–3844. Curran Associates, Inc.

Wang, X., Du, Y., Zhu, S., Ke, L., Chen, Z., Hao, J., and Wang, J. (2021). Ordering-based causal discovery with reinforcement learning. *arXiv preprint arXiv:2105.06631*.

Weisser, T., Lasserre, J. B., and Toh, K.-C. (2018). Sparse-bsos: a bounded degree sos hierarchy for large scale polynomial optimization with sparsity. *Mathematical Programming Computation*, 10(1):1–32.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.

Wilks, S. S. (1962). Mathematical statistics.

Wong, E., Schmidt, F., and Kolter, Z. (2019). Wasserstein adversarial examples via projected Sinkhorn iterations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6808–6817, Long Beach, California, USA. PMLR.

Xiao, H., Rasul, K., and Vollgraf, R. (2017a). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xiao, H., Rasul, K., and Vollgraf, R. (2017b). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

Xie, S., Kirillov, A., Girshick, R., and He, K. (2019). Exploring Randomly Wired Neural Networks for Image Recognition. *International Conference on Computer Vision*.

Xu, Y., Jin, R., and Yang, T. (2019a). Non-asymptotic analysis of stochastic methods for non-smooth non-convex regularized problems. In Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Xu, Y., Qi, Q., Lin, Q., Jin, R., and Yang, T. (2019b). Stochastic optimization for DC functions and non-smooth non-convex regularizers with non-asymptotic convergence. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6942–6951. PMLR.

Zhang, J. (2008). On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873–1896.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. P. (2018). Dags with no tears: Continuous optimization for structure learning. *arXiv preprint arXiv:1803.01422*.

Zhou, Y., Shi, J., and Zhu, J. (2020). Nonparametric score estimators. In *International Conference on Machine Learning*, pages 11513–11522. PMLR.

Zhu, J. (2021). Hessian estimation via stein's identity in black-box problems. *arXiv preprint arXiv:2104.01317*.

Zhu, S., Ng, I., and Chen, Z. (2019). Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477*.

Zimmermann, R. S., Schott, L., Song, Y., Dunn, B. A., and Klindt, D. A. (2021). Score-based generative classifiers. *arXiv preprint arXiv:2110.00473*.

Zou, D., Xu, P., and Gu, Q. (2018). Stochastic variance-reduced hamilton monte carlo methods. *arXiv preprint arXiv:1802.04791*.