Generalization in Deep Learning and Bayesian Graph Cut

Taborsky, Petr

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Generalization in Deep Learning and Bayesian Graph Cut

Petr Taborsky

Kongens Lyngby 2022

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

The focus of the thesis is a puzzling capability of deep neural networks to "work well" on previously unseen data, i.e., to generalize well. The long-lived open problem in machine learning with links to the cognitive abilities of biological neural networks.

The thesis consists of an introductory chapter, three theoretical chapters, addressing the main topic in increasing depth and complexity, and two complementary papers. After the introductory chapter sets the scene, the thesis addresses the problem with increasing complexity. It starts with presenting a novel perspective on an evolution of the eigenspectrum of general loss function during the gradient descent (GD) optimization of the neural network model parameters. The following chapters specialize in a broad class of Bregman divergence losses, that includes common DL objectives. Several novel theoretical and experimental results are presented. These include the formulation of the so-called Self Regularized Bregman Objective (SeReBrO) and its equivalence to stochastic gradient descent optimizing Bregman losses and proving the presence of a latent regularizer.

Next, the following fundamental research question is addressed: *Could a small gradient noise, possibly negligible when updating individual weight parameters by mini-batch back-prop, have a significant effect by imposing a large model variance prior through its non-negligible norm concentrating away from zero in high dimensional (overparameterized) settings?*

The main contribution of this thesis is a positive answer to this question obtained in a sequence of new theoretical results for isotropic and further generalized to

an arbitrary gradient noise covariance matrix. In the result, it is shown that the generalization impact of an overparameterization and noise is limited by the rank of a noisy gradient covariance matrix. Further, it is put into perspective, shedding light on existing experimental and theoretical challenges in generalization in deep learning, and demonstrating that it provides also a practical tool leading to better generalizing models.

An experiment on denoising auto-encoders (DAE) is presented in which recommended explicit regularizers are replaced by stochasticity and overparameterization to boost the rank of gradient noise covariance matrix along the lines of this thesis, and, opposing the previous expectation, are shown to learn to generalize well.

The thesis concludes with a paper on unsupervised Bayesian learning on graphs by making use of generative modeling of uncertainty that allows for inference, verified on real-world data sets and images. The experiment section demonstrates that Bayesian Cut outperforms the popular spectral and modularity-based methods and renders itself as their probabilistic alternative. Bayesian Cut source code has been made available.

Overall, the thesis benefits from taking a probabilistic approach when addressing deep learning in the first part and unsupervised learning in the Bayesian Cut. Intriguingly, a deeper link emerged in a form of hypergeometric functions, that in both cases present the solution to underlying mathematical problems.

Summary (Danish)

Målet med denne afhandling er en forvirrende evne af dybe neurale netværk til at 'arbejde godt' på tidligere usete data, dvs. at generalisere godt. Den langlivede åben problem i maskinlæring med forbindelser til biologiske kognitive evner neurale netværk. Specialet består af et indledende kapitel, tre teoretiske kapitler, ing hovedemnet i stigende dybde og kompleksitet, og to komplementære papirer. Efter det indledende kapitel sætter scenen, behandler afhandlingen problem med stigende kompleksitet. Det starter med at præsentere et nyt perspektive på en udvikling af egenspektret af generel tabsfunktion under gradient descent (GD) optimering af de neurale netværksmodelparametre.

De følgende kapitler specialiserer sig i en bred klasse af Bregman divergenstab, der inkluderer fælles DL mål. Flere nye teoretiske og eksperimenterendesamlede resultater præsenteres. Disse omfatter formuleringen af det såkaldte Selv Regularized Bregman Objective (SeReBrO) og dets ækvivalens til stokastisk gradient-nedstigning optimerer Bregman div. og beviser tilstedeværelsen af en latent regularizer.

Dernæst behandles følgende grundlæggende forskningsspørgsmål: *Kunne en lille gradientstøj, muligvis ubetydelig ved opdatering af individuelle vægtparametre ved mini-batch back-prop, have en betydelig effekt ved at pålægge en stor model varians forud gennem dens ikke-ubetydelige norm, der koncentrerer sig væk fra nul høje dimensionelle (overparametrerede) indstillinger?*

Hovedbidraget i denne afhandling er et positivt svar på dette opnåede spørgsmål i en sekvens af nye teoretiske resultater for isotrop og yderligere generaliseret til en vilkårlig gradient støj kovarians matrix. I resultatet er det vist, at generalise-

ringseffekten af en overparameterisering og støj er begrænset af rangeringen af en støjende gradient-kovarians matrix. Ydermere sættes det i perspektiv, udgydelse lys på eksisterende eksperimentelle og teoretiske udfordringer i generalisering i dyb læring og demonstrere, at det også giver et praktisk værktøj til at lede til bedre at generalisere modeller.

Et eksperiment med denoising auto-encoders (DAE) præsenteres, hvori eksplicitte regularisatorer erstattes af stokasticitet og overparameterisering for at øge rangeringen af gradient støj kovarians matrix i overensstemmelse med denne afhandling, og i modsætning til den tidligere forventning er det vist, at de lærer at generalisere godt.

Specialet afsluttes med et papir om uovervåget Bayesiansk læring på grafer af gør brug af generativ modellering af usikkerhed, der giver mulighed for slutninger, verificeret på datasæt og billeder fra den virkelige verden. Eksperimentdelen viser det Bayesian Cut overgår de populære spektral- og modularitetsbaserede metoder og fremstiller sig selv som deres sandsynlige alternativ. Kildekode til Bayesian Cut metoden er gjort tilgængelig.

Samlet set drager specialet godt af at tage en sandsynlighedstilgang, når den adresserer dyb læring i første del og uovervåget læring i Bayesian Cut. Spændende nok opstod et dybere link i en form for hypergeometriske funktioner, dvs i begge tilfælde præsentere løsningen på underliggende matematiske problemer.

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science, Technical University of Denmark, in fulfillment of the requirements for acquiring a Ph.D. in Engineering.

During the Ph.D., a research stay was conducted at Telenor Research Center, Fornebu, Norway in collaboration with the Norwegian University of Science and Technology, Trondheim. The research considered differential private machine learning and automatic speech recognition systems, from which the findings are not disclosed in this thesis.

The thesis was funded by the Technical University of Denmark and Telenor DK with guidance under Professor Lars Kai Hansen. The work was carried out between January 1, 2018, and April 4, 2022.

The thesis is a monograph that includes 2 research papers.

Lyngby, 04-April-2022

Petr Taborsky

Contributions

The thesis is a standalone monograph.

Papers included in the thesis

A P. Taborsky, L. K. Hansen, Mechanisms that support generalization in deep learning, *currently in proceedings of ICML 2022 (Phase 1 Accepted) at submission date of this thesis.*

B Taborsky, P., Vermue, L., Korzepa, M., and Morup, M. (2021). The bayesian cut. *Ieee Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4111–4124

Papers not included in the thesis

1. Vamsee K. Kodali, P. Taborsky, G. Canright, Using Differential Privacy to Disclose Tourism Statistics with a Quantified Limit on Risk of Re-identification, ©Telenor Research, published internally.

Acknowledgements

I would like to thank $\{Viktor, Marta, Petra\}$ and all permutations of this set for being a never ending source of inspiration, energy and joy during the work on the thesis.

And special thanks to Lars Kai Hansen for his brilliant and human centric support. This has led to a PhD study that has been both fun and challenging.

I would also like to warmly thank my research collaborators: Morten Mørup, Laurent Vermue, Maciej Korzepa, Geoffrey Canright, Vamsee K. Kodali for interesting discussions and learnings throughout the research.

Thanks to the Technical University of Denmark and Telenor Denmark for funding the research.

Contents

Chapter 1

Introduction

... The great question certainly was, what? Alice looked all round her at the flowers and the blades of grass, but she did not see anything that looked like the right thing to eat or drink under the circumstances. There was a large mushroom growing near her, about the same height as herself; and when she had looked under it, and on both sides of it, and behind it, it occurred to her that she might as well look and see what was on the top of it. ...

Alice meets the Caterpillar, Alice's Adventures in Wonderland, Charles Ludtwidge Dodgson (Lewis Carroll, 1865)

1.1 Thesis Outline and Contributions

The focus of the thesis is a long puzzling capability of deep neural networks to "work well" on previously unseen data, i.e., to generalize. This has been a very lively and open problem in machine learning ever since the emergence of current Deep Learning (DL), [Hinton et al., 2006, Salakhutdinov and Hinton, 2007] in 2006 with the latest challenges summarized in [Zhang et al., 2021, Bengio et al., 2021]. It originated from and still has the potential to help understand the cognitive abilities of biological neural networks.

The thesis consists of an introductory chapter, three theoretical chapters addressing the main topic in increasing depth and complexity, and two papers, one published and one in the proceedings. The introductory chapter sets the scene by reviewing necessary existing work and approaches and lays out necessary definitions. The aim is to give a necessary overview of the field and introduce the approach of the thesis by placing it into context.

Chapter 2, GD Evolution of the Hessian for General DL Objectives, analyzes the evolution of the eigenspectrum of general loss function during the gradient descent (GD) optimization of the neural network parameters. The flatness of the Hessian of loss has been often used as one of the proxies for good generalization properties, [Hochreiter and Schmidhuber, 1997]. After reviewing the recent related work on the topic of the chapter the novel results in a form of two Lemmas are proven. It concludes with an interpretation of the results in the context of practical applications with links to supporting experimental works. It includes a discussion on the pros and cons of the general loss approach motivating the next chapter.

Chapter3, Self Regularized Bregman Objective (SeReBrO), reflecting on an ambiguity of a general approach of Chapter 2 this chapter specializes in the broad class of Bregman divergence losses, that includes common DL objectives. Several novel theoretical and experimental results are presented. These include several lemmas leading to the formulation and proof of the first main result of the thesis, the "SeReBrO" Theorem 3.15 with its Corollary, which shows the equivalence between stochastic gradient descent and Self Regularized Bregman Objective (SeReBro) proving the presence of a latent regularizer in SGD optimization of Bregman losses.

Chapter4, Generalization of Deep Learning Optimizing Bregman Divergences, building upon previous this chapter applies SeRebrO on the stochasticity of deep learning in high dimensions present due to overparameterization of the network. First, it builds the intuition for high dimensions leading to the following fundamental research question.

Research Question *Could a small gradient noise, possibly negligible when updating individual weight parameters by mini-batch back-prop, have a significant effect by imposing a large model variance prior through its norm concentrating away from zero in high dimensions?*

The rest of the chapter answers this question positively and constitutes the main contribution of this thesis.

This is done in a sequence of new theoretical results to the Theorem 4.3 and its Corollary 4.4 providing the basis for a proof of Theorem 4.6 addressing the research question for isotropic noise. Further, it generalizes the results to an arbitrary gradient noise covariance through the novel Theorem 4.8 and its Corollary 4.11 showing that the generalization impact of overparameterization and noise is limited by the rank of noisy gradient covariance matrix Σ. These results not only shed light on existing experimental and theoretical works often suggesting ambiguous results, see for instance [Kawaguchi et al., 2017], but also provide a practical tool leading towards better generalizing models.

To demonstrate it the chapter concludes with an experiment on denoising auto-encoders (DAE) [Vincent et al., 2008] in which recommended explicit regularizers are replaced by stochasticity and overparameterization to boost the rank of Σ along the lines of this thesis, and, opposing the previous expectation, are shown to learn to generalize.

Chapter5, Discussion and Conclusion, addresses the generalization challenges raised by work [Zhang et al., 2016] and more recent [Zhang et al., 2021]. Especially the reported ability of deep models to fit random labels with zero training error, which that in first sight is a contradiction to the theory of this thesis. In particular, deep networks trained by SGD should not be allowed to fit random labels (corresponding to the signature of overfitting and highly complex network) by the mere presence of latent SeReBrO regularizers. It is shown that adaptive learning rate training methods, e.g., RMSProp, Adam, etc., used in [Zhang et al., 2016] to fit the random label labels effectively cancel out the SeReBrO regularization because they normalize the gradients by their (moving average) norm. In conclusion, the reported overfitting of adaptive methods is a supporting argument in favor of the thesis' conclusions.

Paper A, Mechanisms that Support Generalization in Deep Learning, presents a complementary geometrical view to previous chapters focused on "flattening" of the representations during the stochastic gradient descent training of deep models. It uses the generalized Pythagorean theorem on the output layer geometry to interpret back-prop training as minimizing the loss of the model on the output layer along parameterized (by activations of network) curves determined by weights general position. It argues that the back-propagation

projects network layer representations into a low-dimensional manifold, while non-linearity, further amplified by the depth of the network, stratifies the model and updates its feature representations locally to assist generalization. In conclusion, several mechanisms, such as depth and batch normalization, are identified to help deep models generalize well, supported by experiments.

Paper B, The Bayesian Cut, derives a novel probabilistic method to graphs cuts, such as the "min-cut" problem. It proposes an alternative to spectral methods, e.g, Normalized cut, by proposing a computational framework for cutting graphs into communities or groups that accounts for parameter uncertainty through Bayesian modeling. It extends the stochastic block model (dc-SBM) and provides an analytical solution to the corresponding constrained integral representation.

As with the rest of the thesis, this paper benefits from taking the (Bayesian) probabilistic approach to both, deep learning models as well as to generative modeling used in this paper. And, albeit distant, there is even a deeper link between this paper and the rest of the thesis in the form of hypergeometric functions, that in both cases present the solution to the underlying problem, the marginal likelihood in the case of The Bayesian Cut and probability of stochastic gradient norm being larger than the non-stochastic norm in Corollary 4.4.

Overall, it leads to a better recovery of the true underlying partitioning structure of nodes into groups and more reliable inference. For illustrative purposes, the results are demonstrated in the context of social network modeling in which the true partitioning structure is known, and as well on image segmentation in which results can easily be visually inspected. The computational framework developed has applications beyond social network modeling and computer vision to the many domains in which graph cuts are currently used.

1.2 Setting Up the Scene

Let a neural network $f : \mathbb{R}^b \times \Upsilon \to \mathbb{R}^d$ of L layers be defined as the composition:

$$f(\boldsymbol{x}, \overrightarrow{\boldsymbol{w}}) = \varphi_L(\boldsymbol{W}^{(L)}) \circ \varphi_{L-1}(\boldsymbol{W}^{(L-1)}) \circ \ldots \circ \varphi_1(\boldsymbol{W}^{(1)})(\boldsymbol{x}) \tag{1.1}$$

where each vector function $\varphi_l(\boldsymbol{W}^{(l)})(\boldsymbol{v}) = a_l\left(\boldsymbol{W}^{(l)}\boldsymbol{v}\right)$ is an activation function a_l applied onto a result of matrix $\boldsymbol{W}^{(l)}$ and vector $\boldsymbol{v}$ product. We denoted collation of all network weights into a tensor as $\overrightarrow{\boldsymbol{w}} \in \Upsilon$.

Definition of Generalization Error We follow definition from widely accepted deep learning book [Goodfellow et al., 2016]. For the sake of brevity we don't use bold typeset for vectors in here as opposed to the main body of the paper. The cost function can be written as an average over the training set, such as

$$J(w) = E_{\hat{p}_{data}}(x, y)\ell(f(x; w), y) \qquad \text{(training error)}$$

where ℓ is the per-example loss function (such as Bregman divergence), and $f(x; w)$ is the predicted output when the input is x, and $\hat{p}_{data}$ is the empirical distribution. In the supervised learning case, y is the target output.

Equation (training error) defines an objective function with respect to the training set. We would usually prefer to minimize the corresponding objective function where the expectation is taken across the data-generating distribution p_{data} rather than just over the (finite) training set:

$$J^*(w) = E_{p_{data}(x,y)}\ell(f(x; w), y) \qquad \text{(generalization error)}$$

In practice a generalization error is estimated on another (test) data set, randomly sampled from p_{data}.

Following definition of a manifold is relatively self-contained and comes from [Carmo and Flaherty, 1992].

Definition 1.1 (Differentiable (Smooth) Manifold)**.** *A differentiable manifold of dimension n is a set M and a family of injective mappings $\boldsymbol{x}_\alpha : U_\alpha \subset \mathbb{R}^n \to M$ of open sets U_α of $\mathbb{R}^n$ into M such that:*

1. $\bigcup_\alpha \boldsymbol{x}_\alpha(U_\alpha) = M$
2. *for any pair α, β, with $\boldsymbol{x}_\alpha(U_\alpha) \cap \boldsymbol{x}_\beta(U_\beta) = W \neq \emptyset$, the sets $\boldsymbol{x}_\alpha^{-1}(W)$ and $\boldsymbol{x}_\beta^{-1}(W)$ are open sets in $\mathbb{R}^n$ and the mapping $\boldsymbol{x}_\beta^{-1} \circ \boldsymbol{x}_\alpha$ is differentiable.*

3. The family $\{(U_\alpha, \boldsymbol{x}_\alpha)\}$ is maximal relative to the conditions (1) and (2).

The pair $(U_\alpha, \boldsymbol{x}_\alpha)$ (or the mapping $\boldsymbol{x}_\alpha$) with $p \in \boldsymbol{x}_\alpha(U_\alpha)$ is called a parametrization (or system of coordinates) of M at p; $\boldsymbol{x}_\alpha(U_\alpha)$ is than called a coordinate neighborhood at p.

For alternative more fundamental definition of Topological Manifold we refer reader to works of [Tu, 2011, Tu, 2017, Hauser, 2018] or to excellent visual work on differential geometry [Needham, 2021].

Definition 1.2. *(Diffeomorphism, [Carmo and Flaherty, 1992], p.10, Def. 2.9) Let M_1 and M_2 be differentiable manifolds. A mapping $\varphi : M_1 \to M_2$ is a* diffeomorphism *if it is differentiable, bijective, and its inverse φ^{-1} is differentiable.*

Proposition 1.3. *(Differential, [Carmo and Flaherty, 1992], p.9, Proposition 2.7) Let M_1^n and M_2^m be differentiable manifolds and let $\varphi : M_1 \to M_2$ be a differentiable map. For every $p \in M_1$ and for each $v \in T_pM_1$, choose a differentiable curve $\alpha : (-\epsilon, \epsilon) \to M_1$ with $\alpha(0) = p, \alpha'(0) = v$. Take $\beta = \varphi \circ \alpha$. The mapping $d\varphi_p : T_pM_1 \to T_{\varphi(p)}M_2$ given by $d\varphi_p(v) = \beta'(0)$ is a linear mapping that does not depend on α. It is called* differential of φ as p.

Proof. cf. [Carmo and Flaherty, 1992] □

On top of the previous we define loss function as Bregman divergence [Banerjee et al., 2005] in Chapter 3 and 4:

$$\begin{aligned}\mathcal{L}(\boldsymbol{z}, \boldsymbol{y}) &= d_\Phi(\boldsymbol{z}, \boldsymbol{y}) \\ &= \Phi(\boldsymbol{z}) - \Phi(\boldsymbol{y}) - \langle \boldsymbol{z} - \boldsymbol{y}, \nabla_y \Phi(\boldsymbol{y}) \rangle \end{aligned} \tag{1.2}$$

, where $\Phi : \mathbb{R}^d \to \mathbb{R}$ is strictly convex function and ∇_y denotes the gradient of Φ.

Overall, following the generalization framework of [Kawaguchi et al., 2017] we aim to minimize $\ell(\overrightarrow{\boldsymbol{w}}; \mathcal{D}, f)$ given training dataset indexed by set $\mathcal{D}$ and hypothesis captured in composition of f:

$$\ell(\overrightarrow{\boldsymbol{w}}; \mathcal{D}, f) = \int\limits_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}} \mathcal{L}(f(\boldsymbol{x}, \overrightarrow{\boldsymbol{w}}), \boldsymbol{y}) d\mathbb{P}(\boldsymbol{x}, \boldsymbol{y}) \tag{1.3}$$

From now on we will abuse notation and use $\boldsymbol{w}$ instead of $\overrightarrow{\boldsymbol{w}}$ to denote either all or subset of weight(s) depending on the context. We will also use index instead

of a function argument to denote dataset over which loss is evaluated, i.e. $\ell_{\mathcal{D}}(\boldsymbol{w})$ instead of $\ell(\overrightarrow{\boldsymbol{w}}; \mathcal{D}, f)$. And we refer to value of loss over batch $B_i \subset \mathcal{D}$ as $\ell_{B_i}(\boldsymbol{w})$.

Further in this paper we consider back-prop training of network f using stochastic gradient descent (SGD) with the constant learning rate η over mini-batch samples indexed by B_i

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta \nabla_w \ell_{B_i}(\boldsymbol{w}_k) \tag{1.4}$$

In case of mini-batch being whole dataset we may refer to it as (full) gradient descent (GD) throughout the text.

Throughout this thesis $\nabla_y g$ denotes the gradient of $g : \mathbb{R}^d \to \mathbb{R}$, defined as $1-$form on a vector space, i.e. a single-vector y input function, for an exact definiton of forms see, [Lee, 2013, Tu, 2011]). However when $g : \mathbb{R}^d \to \mathbb{R}^h$ is vector-valued function then we abuse the notation and $\nabla_y g$ then denotes the Jacobian of map g of $d \times h$ elements.

For the completeness we state here a definition of Lipschitz function as follows.

Definition 1.4 ($C-$Lipschitz function)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ such that $|f(x) - f(y)| \leq C\|x - y\|_2$ for all x and y, where C is a constant independent of x and y and $\|\cdot\|_2$ is a norm induced by Euclidean metric on $\mathbb{R}^d$, is called $C-$Lipschitz function.*

For example, any continuous function with the first derivative bounded by B must be $B-$Lipschitz.

1.3 Related Recent Work

The convergence of learning and generalization in deep neural networks (DNNs) has been a lively research topic for more than two decades. Receding from an ambition to cover even the latest research compellingly this section only mentions the most related and influential works for this thesis with an emphasis to describe different approaches used to address generalization in DL.

To start with a convergence of stochastic gradient descent (SGD) to optima, multi-layer *linear* networks were analyzed by [Glorot and Bengio, 2010] and [Kawaguchi, 2016] and showed that every local minimum is global. There is progress on understanding convergence to global optima in general [Zhu et al., 2019], for two-layer ReLU networks in [Du et al., 2018] and ResNet's in [Du et al., 2019]. The [Hardt et al., 2016] performs the stability analysis of SGD and shows that the faster (stepwise) the training is the more stable the algorithm converges. Overall, convergence and generalization properties of deep models are argued to be related to three following factors by [Jastrzębski et al., 2017]: learning rate, batch size, and gradient covariance. In particular, it is argued that the ratio of learning rate to batch size is a key determinant of SGD dynamics and of the width of the final minima.

Other interesting views on both, convergence and generalization, have been produced by a line of work using a continuous approximation to inherently discrete SGD training leading to representing SGD in a form of a stochastic process guided by (stochastic) differential equations. Early works of [Hansen et al., 1993] showed that SGD upon (adiabatic) convergence to stationary distribution never settles at the optimum but rather fluctuates around with an isotropic covariance. Recently under certain smoothness and moments assumptions works of [Li et al., 2017, Li et al., 2019] derived the expected distribution of SGD not only at optima but weakly along the finite trajectories as well with comments on the switch between "convergence" vs. "fluctuation" regime conditioned on eigenvalues of Hessian to learning rate ratio. Also by dynamical approximation, it has been argued that SGD exponentially favors the flat minima [Xie et al., 2020]. Notably, besides the continuous approximation that requires an infinitesimal learning rate along the paths and in practice is almost never met, it assumes closeness of gradient covariance matrix and Hessian. That is a commonly used assumption but was also argued to be violated. Strictly speaking, it does not hold unless the model reaches the true parameter of underlying data generating distribution, [Pawitan, 2004].

Zooming in on generalization, merits are most commonly attributed to stochasticity in gradient descent (GD) optimization [Roberts, 2021, Smith et al., 2021] claiming that stochastic gradient descent (SGD) implicitly regularizes generaliza-

tion error. But it is not clear how it happens and what role the architecture data and problem plays. The work [Kawaguchi et al., 2017] claims that generalization is determined by all these factors and SGD may and may not even play a role. Also, [Geiping et al., 2021] raises questions about the impact the stochasticity of GD has on generalization whatsoever.

Moreover, there are two fronts of generalization research. On one hand, there is an argument for a presence of an implicit bias of SGD towards well generalizing solutions such as in recent [Huh et al., 2021]. On the other hand, another line of work shows that "bad" local minima exist and SGD may reach them [Liu et al., 2019] supporting the notion that the current understanding of the generalization in DNNs still requires a revision, [Zhang et al., 2016, Liu et al., 2019, Zhang et al., 2021].

(Hessian) flatness-based approaches. The flatness of the loss around (local) optimum is by intuition and as well arguably [Hochreiter and Schmidhuber, 1997, Goodfellow et al., 2016] related to the good generalization of the model around the optima. Intuition is based on local quadratic approximation (in the case of the Gaussian model it would correspond to inverse covariance (precision) matrix, [Bishop, 2006]) and then flat Hessian[1], i.e., with small eigenvalues, corresponds to the model being fitted, that has a large variance and thus is not forced by maximum likelihood to fit training data exactly.

Despite a curvature (Hessian eigenspectrum) depends on parameterization, also challenged in [Zhang et al., 2016] and depending on the settings SGD may never converge to an optimum in finite time due to a presence of (small) negative eigenvalues [Yao et al., 2020] and "asymmetric valleys" in the loss [He et al., 2019a], it has been widely used to explore the generalization abilities of neural networks.

Analyzing the spectrum of Hessian, related to Chapter 2 of the thesis, [Cohen et al., 2021] demonstrates that full-batch gradient descent on neural network training objectives typically operates in an Edge of Stability regime, characterized by the maximum eigenvalue of the training loss Hessian hovers just above the value 2/(step size), and the training loss behaves non-monotonically over short timescales, yet consistently decreases over long timescales.

Further experimental analysis of neural net optimization via Hessian eigenvalues has been enabled by the recent appearance of software such as PyHessian, [Yao et al., 2020] or methods such as in [Ghorbani et al., 2019].

These works provide experimental inputs for Chapter 2 of the thesis. In non-

[1] positive definite at the (global) optimum [Pawitan, 2004]

batch normalized networks, the rapid appearance of large isolated eigenvalues in the spectrum was observed, along with a concentration of the gradient in the corresponding eigenspaces. In batch normalized networks, these two effects are almost absent. Another thesis inspiring work, [Jastrzebski et al., 2021], shows the evidence that the early phase of training a deep neural network has a dramatic effect on the local curvature of the loss function. It is presented that SGD implicitly penalizes the trace of the Fisher Information Matrix (FIM), a measure of the local curvature, from the start of training and that poor generalization coincides with the trace of the FIM attaining a large value early in training.

Representation learning Related, other aspects of deep learning require a deeper understanding, such as the ability of deep networks to learn important features and learn to ignore the irrelevant noisy features (argued to be related to better generalization) as emphasized by [Bengio et al., 2021]. Information theory-based arguments applied for representation learning in [Achille and Soatto, 2018a, Achille and Soatto, 2018b, Achille and Soatto, 2016] argue that invariance to nuisance factors in a deep neural network is equivalent to information minimality of the learned representation and that stacking layers and injecting noise during training naturally bias the network towards learning invariant representations.

When we refrain from a vanilla stochastic gradient training, i.e., without additional explicit methods helping generalization, there are two massive fields that have been shown to help generalization, "normalization" and "explicit regularization".

Normalization It is well established that normalization, i.e., an affine transform of the inputs to produce zero means and unit variance random variables, accelerates convergence [LeCun et al., 2012]. Since then many other normalization methods have been proposed.

Batch normalization (BN) [Ioffe and Szegedy, 2015] is an algorithmic method that makes the training of DNN faster and more stable. It consists of normalizing activation vectors from hidden layers using the first and the second statistical moments (mean and variance) of the current batch. This normalization step is applied right before (or right after) the nonlinear function. BN has been shown to provide several training merits. These include prevention of exploding or vanishing gradients, robustness to different settings of hyper-parameters such as learning rate and initialization scheme and keeping most of the activations away from saturation regions of non-linearities. Work in [Kohler et al., 2018] based on Gaussian input assumptions showed that BN reduces the cross-dependency between layers in DNN. Due to this dependency reduction, a gradient-based optimization with an adaptive step size can enjoy a linear convergence rate.

Weight Normalization (WN) [Salimans and Kingma, 2016] decouples the length

of the weight vector from its direction. The optimization of these two components runs separately. Notably, WN has been shown empirically to benefit from similar acceleration properties as BN. While BN fits well in feed-forward architectures it is more troublesome to apply in recurrent networks. Other caveats include a dependence on mini-batch size and deterioration if test set sample distribution differs from the training set (see also below).

Layer Normalization (LN) [Ba et al., 2016], which normalizes all hidden units of the layer by shared mean and variance computed on a single training sample, was developed to mitigate all these issues and empirically demonstrated to work well, especially for recurrent networks, [Ba et al., 2016].

While all the above-mentioned techniques relate the benefits of normalization to optimization, the generalization merits drawn significantly less attention. The original work [Ioffe and Szegedy, 2015] conjectured that BN implicitly regularizes training to prevent over-fitting. [Zhang et al., 2016] and later in [Zhang et al., 2021] presents BN as an implicit regularizer drawing on experimental evidence. Injecting random noise in the input layer is known to be equivalent to regularization by weight decay (L2 regularization), as shown in [Bishop, 1995] and generalized to DNNs or autoencoders (AE), [Poole et al., 2014]. Dropout [Srivastava et al., 2014] can be seen as noise injection in DNN layers, [Poole et al., 2014].

Nevertheless, to the best of our knowledge generalization properties of solutions found by BN are not yet fully understood. Moreover, [Lian and Liu, 2019] points out three downsides of BN, shown experimentally, as follows:

- BN overfits when trained with unit mini-batch size.
- Solution found by BN is mini-batch size-dependent.
- BN linear model fails to converge to high accuracy (compared to alternative logistic regression model) if data are with large variation[2].

BN was empirically shown to improve generalization in DNN [Luo et al., 2018, Santurkar et al., 2018, Ba et al., 2016]. Yet, theoretical understanding is still in progress. The original work [Ioffe and Szegedy, 2015] hypothesized that BN effectiveness was mostly due to internal covariate shift (ICS) reduction. The idea was challenged by [Santurkar et al., 2018] using counterexamples and rather stressed the smoothing effect of BN on the optimization landscape. Specifically, it is demonstrated that there does not seem to be any specific link between the

[2]This particular experiment is on the shallow linear network, implementing logistic regression trained on synthetic data

convergence gain of BN and the reduction of ICS. In fact, it is shown that in a certain sense BN might not even reduce ICS. Here we follow to argue that it is not *reduction* of ICS, but rather the pervasive *presence* of ICS throughout the training process that assists generalization.

Explicit Regularization There are many regularizers at hand to be combined with SGD training and that work provably and empirically well. Next, we review the most common regularization techniques.

The classic and one of the most effective and efficient regularizers is the **early stopping** combined with a common random initialization of weights around zero, i.e. of [He et al., 2015, Glorot and Bengio, 2010] zero mean and variance that corrects for the number of units of the network weights are gradually updated over the course of the learning and suggest that vanilla SGD leads towards a complex and eventually over-fitting network f characterized by large weights. Early stopping is an effective and robust way to stop weights along the way, [Li et al., 2020a].

Another widely effective regularization stemming from numerical methods as total variation and Tikhonov regularization [Bishop, 1995] is the **weight decay** [Goodfellow et al., 2016]

One of the most common regularization techniques developed specifically to regularize neural networks is the **drop-out**, [Srivastava et al., 2014]. Dropout regularizes by multiplying, a random or deterministic, a subset of layers' outputs by zero [Srivastava et al., 2014, Goodfellow et al., 2016] and thus deactivating weights leading to those units from gradient update at the given step, slows the growth of the weight parameters similarly to BN.

Note that the authors of batch normalization in their work [Ioffe and Szegedy, 2015], also see above, based on experiments suggest that batch normalization reduces, partially or completely, the need for drop-out, suggesting a similar effect on the training. Or alternatively, [Hinton et al., 2012b] approximates the dropout effect by the full model with outgoing weights of node i multiplied by the probability of including the unit i. That is weight noise-related regularization.

Noise injection, including the above mentioned dropout or use of a mini-batch stochastic gradient[3], is also widely known and working regularizer [Neelakantan et al., 2016, He et al., 2019b, Noh et al., 2017, Bishop, 1995, Poole et al., 2014]. Note that the data augmentation can be viewed as a noise injection into input data [Goodfellow et al., 2016, Poole et al., 2014].

[3] as opposed to a full (non-stochastic) gradient evaluated on all training data at every step

Besides explicit regularization or normalization generalization can be imposed "by a principle". As in a view provided by information-theoretic approaches based on compressing the information flowing through the network [Tishby and Zaslavsky, 2015, Tishby et al., 2000]. They draw the links between SGD training and information bottleneck (IB) variational principle (regularizer) based on maximizing the (mutual) information between quantization of the inputs and targets while minimizing (mutual) information (forgetting irrelevant features) between input and quantized (compressed) version.

Deep Networks In the case of deep networks, vanilla SGD suffers from vanishing gradients and fails to train whatsoever. Or if trained the results are suboptimal compared to networks trained with help of explicit methods such as batch-normalization described above. For recent deep architectures with **skip connections**, i.e., identity maps between activations from different layers, often "skipping" one or more layers, [He et al., 2016, He et al., 2020], these skip connections have been shown to have a regularization effect due to imposing smoothness prior on a layer to layer maps composing the network f in [Hauser, 2018].

Initialization has been explored as one of the fundamental building blocks of good generalization [Frankle and Carbin, 2018] and more recent in [Malach et al., 2020]. The Lottery ticket hypothesis [Frankle and Carbin, 2018] brings into play the role of weights initialization [Sutskever et al., 2013, Daniely et al., 2016] linked to random matrix theory and argues that dense, randomly-initialized, feed-forward networks contain subnetworks (winning tickets), significantly simpler in terms of a number of parameters, that, when trained in isolation, reach test accuracy comparable to the original network in a similar number of iterations, i.e., generalize the same. This suggests that depth may not have a larger role than initialization in deep networks.

While depth has been shown to exponentially increase the expressivity of the network compared to wide networks [Hanin and Rolnick, 2019], the role of depth with respect to generalization is largely unknown. On top, the same work of [Hanin and Rolnick, 2019] has shown that the practical expressivity of deep networks – the functions they can learn rather than express – is often far from the theoretical maximum, and realizing the full expressivity of deep networks may not be possible in practice, at least with current methods. This thesis provides yet another view and argues that lower practical expressivity may be the price to pay for the better generalization imposed by the depth, under circumstances. There is no free lunch, is there? :-)

1.3.0.1 Alternative Approaches to Generalization

Besides the most common definition of generalization error [Goodfellow et al., 2016] from the Definitions section the following presents selected alternative approaches for completeness.

Fisher Rao Metric and Complexity Recent work [Liang et al., 2019] proposed Fisher-Rao norm as a capacity measure of generalization ability

$$\|\boldsymbol{w}\|_{fr} = \boldsymbol{w}^T F \boldsymbol{w} \tag{1.5}$$

where $\boldsymbol{w}$ denotes flattened vectorized version of all weights of the neural network model $f(\circ, \boldsymbol{w})$. It is shown therein that this norm has desirable invariance properties and relates to generalization capacity defined as Rademacher complexity

$$\mathcal{R}_N(\Omega) = E_\sigma \sup_{w \in \Omega} \frac{1}{\mathcal{D}} \sum_{s=1}^{|\mathcal{D}|} \sigma_i \mathcal{L}(f(\boldsymbol{x}_s|\boldsymbol{w}), \boldsymbol{y}_s). \tag{1.6}$$

The Rademacher complexity directly gives an upper bound on generalization error (see e.g. [Mohri et al., 2012]), Theorem B.1 Suppose the loss function is bounded in $[0, c]$ and is L-Lipschitz in the first argument. Then with probability at least $1 - \delta$ over sample $\mathcal{D}$ of size $|\mathcal{D}|$:

$$\sup_{w \in \Omega} \left\{ \mathcal{L}(f(w)) - \mathcal{L}_{\mathcal{D}}(f(w)) \right\} \leq 2L\mathcal{R}_{\mathcal{D}}(\Omega) + 3c\sqrt{\frac{\log(2/\delta)}{2|\mathcal{D}|}} \tag{1.7}$$

where $\mathcal{F}$ is defined as a set of functioned generated by weights architecture and dataset $f(\boldsymbol{x}_s, \boldsymbol{w}), s \in \mathcal{D}$ and $\mathcal{L}$ without index denotes expected loss over the latent data distribution as opposed to empirical loss $\mathcal{L}_{\mathcal{D}}$ we optimize over.

Therefore, as long as we can bound the Rademacher complexity of a certain function class that contains our learned predictor, we can obtain a generalization bound.

In particular for deep *linear* networks [Liang et al., 2019, Neyshabur et al., 2015] has shown that Rademacher complexity is bounded from top by this norm. In the definition we used *primal* formulation of the loss 3.0.1.2, $d_\Phi(f(\boldsymbol{x}|\boldsymbol{w}), \boldsymbol{y})$.

In the case of ReLu networks bounding Rademacher complexity by Fisher-Rao norm $\|\cdot\|_{fr}$ is challenging. However, it is shown therein that other norms (spectral, group and *path* norm) can be viewed as a subset of the unit Fisher-Rao norm ball, and Rademacher complexity on these subsets is bounded by $L \cdot \frac{2^{depth} Polylog}{\sqrt{N}}$.

Expressivity vs. Generalization Work [Raghu et al., 2017] has shown visually and theoretically, that arc length $l(f(x(t)))$ of the trajectory of the network $f(x)$ along the (nonlinear) curve $x(t)$ grows exponentially with the depth of the network and show that this bound is asymptotically tight in the limit of large initialization variance.

1.3.1 Learning in the Manifold of Distributions (LiMoD)

This section introduces the probabilistic view taken by the thesis from Chapter 3 onwards, the Learning in the Manifold of Distributions. It connects works of [Hauser, 2018] and [Amari, 2016] and provides the rigorous foundation for SGD optimization of Bregman divergence with regards to parameters of its argument.

It is defined by seeing layers of the neural network model being seen as coordinate representations of a data manifold[4] M following [Hauser, 2018] and imposes the Exponential family probability model over the output layer. One model per every training datum (or pair of input-output in case of supervised learning). Let us dub it *Learning in the Manifold of Distributions*, or "LiMoD" in short for the rest of the thesis.

In this view, the neural network f is a composition of maps, parametrized by the learnable weights, that define layer-by-layer transformations of coordinate representations of M. Altogether it forms by assumption a smooth[5] input-output map.

As presented in [Hauser, 2018] an input layer is a rather arbitrary representation of a data manifold M. For example, an RGB representation of the image used for image recognition is used just because RGB is a convenient format for image software and displays. However, it is not a good representation for the labeling/image recognition task. The output layer is considered to be embedded in the Euclidean space.

In the case of the Bregman divergence loss, the output layer has a dually flat geometry following the Information Geometry, [Amari, 2016]. By Theorem 2.2 of [Amari, 2016] there is a bijection between regular Exponential families and the Bregman divergences using the result of [Banerjee et al., 2005]. And further, Theorem 6.2, [Amari, 2016], instigates that a geometry derived from the Bregman divergence is dually flat and the affine coordinate systems are natural and mean value parameters.

Let's define the index set $I_{\mathcal{D}} = \{1, \ldots, |\mathcal{D}|\}$, where $\mathcal{D}$ is training data set. Then using the *dual* interpretation (3.0.1.2) and (3.31), the loss over training data is

[4] For the definition of a manifold see previous section or any differential geometry book. Intuitively speaking a manifold is a deformed Euclidean space with typically different global topology, e.g., a sphere vs. a flat plane in 3D as examples of two different 2-dimensional manifolds. An n-dimensional manifold is a set of points such that each point has n-dimensional extension in its neighborhood and every such neighborhood is topologically equivalent to an n-dimensional Euclidean space - that gives rise to a local coordinate system.

[5] ReLu can be seen as a limit case of smooth NN models using softmax activations

given by

$$\sum_{s\in I_{\mathcal{D}}} \mathcal{L}(f(\boldsymbol{x}_s, \boldsymbol{w}), \boldsymbol{y}_s) = -\log \prod_{s\in I_{\mathcal{D}}} p_{\theta_s}(\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{y}}_s) + \underbrace{\sum_{s\in I_{\mathcal{D}}} \Psi(\tilde{\boldsymbol{y}}_s)}_{const.w.r.t.\boldsymbol{w}} \tag{LiMoD}$$

where $\theta_s = f(\boldsymbol{x}_s, \boldsymbol{w})$ are natural parameters of *dual* exponential family with cumulant $\Phi(\cdot)$, derived from $d_\Psi(\nabla\Phi(\boldsymbol{y}), \nabla\Phi(f(\boldsymbol{x}_s, \boldsymbol{w})))$ over random variables $\tilde{\boldsymbol{y}} = \nabla\Phi(\boldsymbol{y})$.

Under a common assumption that training data is independently sampled from a latent data distribution, the factors of the product $\prod_{s\in I_{\mathcal{D}}} p_{\theta_s}(\tilde{\boldsymbol{y}})$ are mutually and conditionally independent distributions, given weights $\boldsymbol{w}$.

An example of such a product of two one dimensional Gaussians with zero means and unit variance is presented in Fig 1.1. It depicts the settings with training data $\mathcal{D}$ comprising two data points with target values $\{0, 0\}$. In other words, the output layer distribution is fully determined by $|\mathcal{D}|$, targets $\boldsymbol{y}_s, s \in I_{\mathcal{D}}$ and choice of Φ[6].

Thus there are two interpretation at hand. First, optimizing the (LiMoD) may be seen as maximum likelihood optimization of one large Exponential Family "product" model with the dimensionality $|\mathcal{D}| \times (dim(\boldsymbol{y}) - 1)$ with conditionally independent factors corresponding to individual observations indexed by s in (LiMoD). Or the second, it is also a maximum likelihood (ML) optimization of many, i.e. $|\mathcal{D}|$ conditionally independent $dim(\boldsymbol{y})$-dimensional models derived from *dual* formulation of loss 3.0.1.2.

Note that optimum of distribution $p_{\theta_s}(\tilde{\boldsymbol{y}})$ is attained at $\nabla\Phi(\boldsymbol{y}_s)$ for every s because it is ML estimate for the case of one single observation. And further making use of independent samples of training data, the unique optimum (from the strict convexity of Φ the gradient mapping $\nabla\Phi(\boldsymbol{y}_s)$ is invertible) of the large "product" model is also known and given by coordinates $[\boldsymbol{y}_s], s \in I_{\mathcal{D}}$. Let's denote it $\hat{f}$.

Given independently sampled finite training data set $\mathcal{D}$ from a latent data distribution we call stochastic gradient descent optimization of LiMoD with regards to weights **Learning in the Manifold of Distributions** (LiMoD). As already outlined LiMoD provides a novel perspective of the learning in the probabilistic manifold with known geometry derived from Φ (or equivalently

[6] together with some base measure. However, it is irrelevant with regards to optimizing LiMoD w.r.t. to weights. Thus without loss of generality, it can be set to Lebesgue or counting measure in case of discrete targets and further disregarded for brevity.

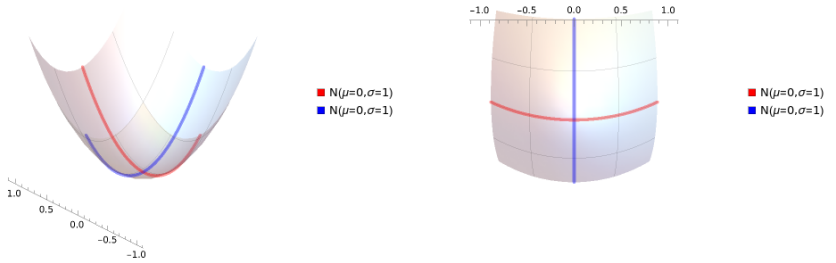


***Figure 1.1:** Log density of the product of two zero mean unit variance Gaussian distributions.*

from Ψ) that corresponds to optimizing finitely many known distributions, $p_{\theta_s}(\tilde{\boldsymbol{y}}$, as opposed to a standard one-probability-model view.

So optimizing LiMoD seems quite straightforward, isn't it? We know the optimum of the output layer, i.e. $\hat{f}$, and we just need to put the finger on it using the weights $\boldsymbol{w}$ parametrizing the f. In the deep learning settings, there are often by orders of magnitude more weight parameters than there are training data samples $|\mathcal{D}|$. The optimum $\hat{f}$ may be expressed by the multitude of weight configurations/sub-manifolds of weight manifold, [Zhang et al., 2016], that all have the same loss value[7]. They may differ in generalization properties of f they define though. This is also true for linear over-parametrized networks, e.g., overparameterized linear regression, for instance.

The challenge is thus not only to converge to any one of optima in weight space but to converge to the one that generalizes well.

The LiMoD perspective will be picked up in Chapter 3 and extended therein. Further, the thesis will explore how SGD training, overparameterization, and architecture of network f impact its generalization using the framework of LiMoD.

[7] alongside with other local optima with larger loss values in general

Chapter 2

GD Evolution of the Hessian for General DL Objectives

. . . As soon as she had made out the proper way of nursing it, (which was to twist it up into a sort of knot, and then keep tight hold of its right ear and left foot, so as to prevent its undoing itself,) she carried it out into the open air. . . .

Alice and pig baby, Alice's Adventures in Wonderland, Charles Ludtwidge Dodgson (Lewis Carroll, 1865)

2.0.1 Chapter Introduction

The most of deep learning theory books consider the neural network model $f(\boldsymbol{x}, \boldsymbol{w})$ as a (unnormalized) probability model either explicitly, or implicitly, parameterized by weights w. Then loss optimization corresponds to maximizing likelihood directly or by lower bound (ELBO) in variational/bayesian networks or else.

Such approaches allow for an application for wide range of well developed methods from statistical learning theory yet come short at fully explaining some phenomenons observed in deep learning, such as its generalization capability among others, [Zhang et al., 2016, Zhang et al., 2021].

Another approach uses geometric (re)formulation of the deep learning problem as optimizing a whole family of distributions (so-called probability manifold, [Amari, 2016, Hauser, 2018]) as opposed to one single model.

Alternatively, instead of relying on the implicit regularization one may regularize model explicitly[Goodfellow et al., 2016]. This is well known and long used concept stemming from numerical and statistical methods used for instance in inverse problems such as Tikhonov regularization or total variation [Bishop, 1995]. In fact there is a line of work arguing for nonexistence of implicit regularizer as in [Liu et al., 2019] or its needlessness [Geiping et al., 2021], see Related Work below. More over wast and active field of research on Bayesian deep learning or Gaussian processes [Rasmussen, 2003], deploying priors (believes) can be seen as generalizing concept of "hard" regularizers ported into probabilistic learning [Bishop, 2006, Goodfellow et al., 2016]. [Luo et al., 2018].

In this initial chapter we take general loss and analyze the evolution of the eigen-spectrum of the Hessian of loss under following assumptions.

Definition 2.1. *(Weak Assumptions)*

1. *weights lie in a differentiable manifold, denoted $\mathcal{W}$, i.e. it is a topological space. For the purpose pf deep learning we assume $\mathcal{W} = \mathbb{R}^m$*

2. *stochastic gradient descent step is reversible and smooth map, i.e., diffeomorphism, see Definition 1.2, on the weight manifold*

3. *the training objective (a loss function) is trice differentiable and with a bounded gradient on any closed interval of the domain*

4. *standard random weight initialization, [Sutskever et al., 2013, Daniely*

et al., 2016, *Glorot and Bengio, 2010*, *Goodfellow et al., 2016*)[1]

5. *(technical) SGD updates are evaluated at an arbitrary precision*[2]

Remark. *We explicitly note that a variable learning rate η_t and arbitrary loss ℓ is accounted for in this Chapter.*

In what follows this chapter shows in Lemma 2.2 that under Weak Assumptions 2.1 the Hessian of the objective at the point T is closely approximated by $H_{w_T}(B_T) = A \prod_{t=1}^{T-1} \left(I - \eta_t H_{w_t}(B_t)\right)^{-2}$, where $A = H_{w_0}(B_T)$ is the Hessian after a random initialization at time $t = 0$ and where B_t denotes a randomly sampled batch of training data at step t.

This is to the best of authors knowledge a novel results that links a curvature of the solution (Hessian) to the convergence path and not only sheds light on deep learning phenomenons and recent experimental results, see Related Work, but also suggests the ways to improve generalization.

Interpretation of the main result of this chapter, Lemma 2.2, is presented therein including insights into the long puzzling mysteries of deep learning by (S)GD, such as its generalization ability in spite the over parameterization and convergence to global optimum in spite of a generally rugged landscape.

2.0.2 Chapter Related Preliminaries and Definitions

Diffeomorphism, defined by 1.2, introduces a notion of equivalence between differentiable manifolds. As an example take differentiable and invertible change of coordinates (change between bases of a vector space) in Euclidean spaces $\mathbb{R}^n$.

2.1 GD Evolution of the Hessian Spectrum

Let's define mapping $\varphi : \mathcal{W} \to \mathcal{W}$ that represents one GD step t as follows

$$\varphi(t) : w \to w - \eta_t \frac{\partial}{\partial w}(\ell(B_t, w)) \qquad \text{(gradient step)}$$

[1] replaceable by other less practical assumptions

[2] technical requirement allowing to work with matrix inverse for the sake of simplicity and instead of Moore-Penrose pseudo inverse, in case this assumption was violated

It is straightforward to see that given training data (in a batch of full) denoted B_t the φ is *locally* a diffeomorphism, because gradient map is locally an invertible affine map. We'll see that *locally* is "good enough" for this entry Chapter as all its results are valid under "small gradient regime", i.e, $\left\|\nabla\ell(\varphi(w_0))\right\|_2 \leq \epsilon$, and thus the whole Chapter operates under "first order" approximation.

2.1.1 GD as a Sequence of Local Coordinate Changes

As shown in [Carmo and Flaherty, 1992], p26, if $\varphi : \mathcal{W} \to \mathcal{W}$ is a diffeomorphism, $v \in T_p\mathcal{W}$ and ℓ is a differentiable function in a neighborhood of $\varphi(p)$, we have

$$(d\varphi_t(v)\ell)\varphi(p) = v(\ell \circ \varphi_t)(p) \quad \forall t \in \mathbb{R}. \tag{2.1}$$

It follows by a derivative chain rule applied on a differentiable parameterized curve $\alpha(t), t \in \mathbb{R}$ and using the fact that the differential $d\varphi(v)$ is independent on α as in Proposition 1.3 in Preliminaries.

Note that for some $p \in \mathbb{W}$ the v denotes a tangent vector[3] of a form $\sum_{i=1}^{m} a_i(p)\frac{\partial}{\partial x_i}(p)$, where x_i represent coordinate curves of the tangent space $T_p\mathcal{W}$ and $a_i(p)$ are its coordinates. If $(\frac{\partial}{\partial y_1}, \ldots, \frac{\partial}{\partial y_m})(\varphi(p))$ denotes coordinate system of this tangent space then differential $d\varphi$ then maps v to $d\varphi(v) \in T_{\varphi(v)}\mathcal{W}$, i.e., vector $\sum_{i=1}^{m} a_i(\varphi(p))\frac{\partial}{\partial y_i}(\varphi(p))$.

Detailed proof requires some more technical definitions and is found in [Carmo and Flaherty, 1992]. We only need its special case when $\mathcal{W} = \mathbb{R}^m$. In this case RHS of (2.1) reduces to *a standard chain rule of derivatives* with (2.1) defining the corresponding Jacobian of the coordinates change.

Considering the φ defined above as the GD update the following lemma links a curvature of the loss at step T to the sequence of the GD updates leading to it.

Lemma 2.2. *(Full batch) Consider full batch gradient descent (GD) optimization of the loss ℓ over weight parameters of dimension d and φ_t defined in* (gradient step)*. Under general Assumptions 2.1 consider loss function ℓ having bounded absolute values of third partial derivatives by $0 \leq \gamma < \infty$ on any closed interval of its domain. Further at GD step $T \in \mathbb{N}^+$ define composition $\varphi(w_0) := \varphi_{T-1} \circ \cdots \circ \varphi_0)(w_0)$.*

If the norm of the gradient $\left\|\nabla\ell(\varphi(w_0))\right\|_2 \leq \epsilon$ at T is upper bounded by $\epsilon \leq 0$ the

[3] (or 1-form, see [Tu, 2017])

following holds for the Hessian of ℓ at the step T

$$H_{\varphi(w_0)} = H_{w_0} J^2 + \varepsilon, \quad \|\varepsilon\|_{max} \leq \epsilon\gamma \tag{2.2}$$

where J is a real (symmetric) inverse of Jacobian of $\varphi(w_0)$ w.r.t. w_0

$$J = \prod_{t=1}^{T-1} \left(I - \eta_t H_{w_t}\right)^{-1} \tag{2.3}$$

and where ε denotes a real symmetric $d \times d$ "error matrix" with all elements in absolute value lower than $\epsilon\gamma$.

Proof. For the sake of simplicity we show the case for two dimensions. Higher dimensions are exactly same just more involved.

Let denote by $(x, y) \to (u(x, y), v(x, y))$ the "change of variable" diffeomorphism. Consider $\ell((u(x, y), v(x, y)))$ and its Hessian w.r.t. (x, y) and (u, v).

First and second derivatives of the loss are

$$\ell_x = \ell_u u_x + \ell_v v_x \tag{2.4}$$
$$\ell_{xy} = \ell_{uu} u_x u_y + \ell_{uv} u_x v_y + \ell_u u_{xy} + \ell_{vu} v_x u_y + \ell_{vv} v_x v_y + \ell_v v_{xy} \tag{2.5}$$

where subscript denotes derivative of ℓ with regards to it. The full hessian is

$$\underbrace{\begin{pmatrix} \ell_{xx} & \ell_{yx} \\ \ell_{xy} & \ell_{yy} \end{pmatrix}}_{H_x} = \begin{pmatrix} \ell_{xx}(u(x,y), v(x,y)) & \ell_{yx}(u(x,y), v(x,y)) \\ \ell_{xy}(u(x,y), v(x,y)) & \ell_{yy}(u(x,y), v(x,y)) \end{pmatrix} \tag{2.6}$$

with the elements given by (2.5) with appropriate index.

By assumptions ℓ has bounded third partial derivatives by γ. Set $w_0 = (x_0, y_0)$ and $(u_1(x_0, y_0), v_1(x_0, y_0)) = \varphi(w_0) = w_1$. Thus (2.5) becomes

$$\ell_{xy} = \ell_{uu} u_x u_y + \ell_{uv} u_x v_y + \ell_{vu} v_x u_y + \ell_{vv} v_x v_y + \varepsilon_{xy}, \quad |\varepsilon_{xy}| \leq \gamma\epsilon, \quad \forall x, y \tag{2.7}$$

where the upper bound on ε follows from $\ell_{\varphi_0(w_0)}((u_1(x_0, y_0), v_1(x_0, y_0))) < \epsilon$ by assumption of the lemma and by the Cauchy-Schwartz inequality.

While (2.2) gives lower left corner element of $H_{(x,y)}$ a straightforward matrix multiplication formula verifies that following gives full Hessian

$$\underbrace{\begin{pmatrix} \ell_{xx} & \ell_{yx} \\ \ell_{xy} & \ell_{yy} \end{pmatrix}}_{H_{(x,y)}} = \underbrace{\begin{pmatrix} u_x, v_x \\ u_y, v_y \end{pmatrix}}_{J_x} \underbrace{\begin{pmatrix} \ell_{uu} & \ell_{uv} \\ \ell_{vu} & \ell_{vv} \end{pmatrix}}_{H_{(u,v)}} \underbrace{\begin{pmatrix} u_x, u_y \\ v_x, v_y \end{pmatrix}}_{J_x^T} + \underbrace{\varepsilon}_{\text{an error matrix } \varepsilon} \tag{2.8}$$

where ε is real symmetric matrix with elements ε_{xy} defined above and d denotes dimension of weights from the statement of the lemma (in this particular case $d = 2$). The symmetry of ε, i.e., $\varepsilon_{xy} = \varepsilon_{yx}$, follows directly from exchangeability of the order of differentiation $\frac{\partial^2}{\partial x \partial y} = \frac{\partial^2}{\partial y \partial x}$ for any twice differentiable function.

While Jacobian J_x are not symmetric in general in case of SGD diffeomorphism φ defined in (gradient step) and by taking derivative

$$J_{w_0} := \begin{pmatrix} \frac{\partial u_\varphi}{\partial x}, \frac{\partial u_\varphi}{\partial y} \\ \frac{\partial v_\varphi}{\partial x}, \frac{\partial v_\varphi}{\partial y} \end{pmatrix} (w_0) = I - \eta H_{(x,y)}(w_0) \tag{2.9}$$

where (u_φ, v_φ) denotes vectorized version of map $\varphi : (x, y) \to (u, v)$, we see that it is real, symmetric due to interchangebility of the order of derivatives in and invertible (diffeomorphism is invertible by definition).

In the previous we derived the result without considering φ being composition diffeomorphisms (corresponding to gradient descent steps). Because composition of diffeomorphisms is again diffeomorphism [Carmo and Flaherty, 1992] this extension is straightforward by plugging the definition of $\varphi(w_0) := \varphi_{T-1} \circ \cdots \circ \varphi_0)(w_0)$ from the Lemma into (2.9) and applying a chain rule to obtain $J_{w_0}^{w_T} = \frac{\partial \varphi}{\partial w_0} = J_{w_{T-1}}^{w_T} J_{w_{T-2}}^{w_{T-1}} \dots J_{w_0}^{w_1} = \prod_{t=0}^{T-1} (I - \eta_t H_{w_t})$.

Because all the Jacobians involved are real symmetric matrices as well as H_{w_0} is they all commute. If the product of two symmetric matrices is symmetric, then they must commute ($AB = (AB)^T = B^T A^T = BA$) (Or more formally they are all unitarily diagonizable (by SVD) and because a change of orthonormal basis does not change eigenvalues, they are simultanously unitarily diagonizable, [Gantmakher, 1959], and thus normal. Normal matrices commute.).

Invertibility and symmetry allows to set $J := (J_{w_0}^{w_T})^{-1}$ and the statement of the lemma follows. □

Remark. *Notably only gradient at T is required to be small while gradients in between are arbitrary (bounded by Assumptions 2.1). That is once SGD hits low level of the loss, i.e., norm of gradients are below ϵ, at some point T the Lemma 2.2 allows for inspection of its curvature depending at the path leading to this point.*

Remark. *(Bounded 3^{rd} derivatives of ℓ)The third derivative bound γ is always finite on finite datasets and at finite T, i.e. after finite number of back-prop steps, assuming a smoothness of f and initialization around zero, as elaborated earlier. Never the less, the absolute value of the (finite) bound may be ever increasing with number of steps and thus one needs some mechanism (implicit or explicit) to regularize the f.*

While this Chapter considers a general loss function, we lack such mechanism in this generality. In the next Chapter 3 we reduce our attention to still a wide class of Bregman losses, comprising the most common objectives, as will be shown. It will be argued that when optimized by a stochastic gradient descent over-parameterized neural network f tends to be constraint (and thus ℓ) rendering the results of this Chapter usefull in such contexts.

Note that because $\varphi_t, \forall t$ are diffeomorphisms then $(I - \eta_t H_{w_t})$'s are invertible for every t and J is well defined.

2.1.1.1 Difficulties in Generalizing Lemma 2.2 to Stochastic Settings

While the Lemma 2.2 assumes a full batch gradient descent we'd like to have simillar statement for a stochastic version *in expectation* w.r.t. weight initialization probability distribution at least.

However even that seems to be quite challenging without further assumptions, as also noted in [Cohen et al., 2021] for different reasons, perhaps. Let's demonstrate it in one-dimensional case.

First note, that in stochastic gradient descent settings both gradient and Hessian depends on the batch. We'll indicate the corresponding batch at step t in their arguments, i.e., the $H_{w_0}(B_t)$ denotes the sample Hessian of loss *ell* evaluated over batch B_t, given step t. In this respect the Lemma 2.2 worked for full batch (GD) with a simplified notation $H_{w_0} = H_{w_0}(B_T) = H_{w_0}(B_0)$.

Let's assume H is one dimensional variable and let's denote its expectation by H_0. Let's assume that ξ represents series of stochastic updates of Hessian H_0 over $k > 1$ SGD steps thus $H = \xi(H_0)$. It represents a "stochastic" and one dimensional version (function) of Equation (2.2) from the Lemma 2.2. Then by the Taylor expansion of $\xi(H)$ around H_0 we can write

$$E[\xi(H)] = \xi(H_0) + E[\xi''(x_H)(H - H_0)^2]. \tag{2.10}$$

In case φ had bounded second derivative, i.e. $|\xi''(x_H)| \leq C$ for all H we could write

$$|E[\xi(H)] - \xi(H_0)| \leq CE[(H - H_0)^2] = C\mathrm{Var}(H). \tag{2.11}$$

This would be exactly what we wanted. However tricky part is bounded second derivative of transformation ξ in $|\xi''(x_H)| \leq C$. By inspection of Equation (2.2)

from the Lemma 2.2 we see that $\xi(H_0)$ involves $\prod_{t=1}^{T-1} (I - \eta_t H_{w_t})^{-2}$ meaning that if $\xi(H_0)$ get close to $1/\eta_t$ at any of k steps the bound C explodes rendering the approximation above not useful.

It may be possible to work around it by showing under some additional assumptions, on an initial and sampling distribution as well as on a learning rate η_t perhaps, that this case has low probability to happen and obtain the stochastic version of the Lemma 2.2 in expectation. It is left for future work however and we follow the Chapter with full gradient descent (GD) in mind.

Remark. *(On generalization of the Lemma 2.2 to an arbitrary manifold $\mathcal{W}$) From the differential geometry point of view the Hessian is an extrinsic curvature (second fundamental form), [Carmo and Flaherty, 1992] and as such depends on the coordinate system. Hence it changes under diffeomorphism (gradient step) and one needs the assumptions of the Lemma, i.e., small gradient assumption $\left\|\nabla \ell(\varphi(w_0))\right\|_2^2 \leq \epsilon$ and some analogy of the third derivatives on manifold at T to generalize (2.2) to arbitrary differential manifold $\mathcal{W}$. We do not go into this general case here however and leave it for future work, as it is not required to explain the current deep learning applications and is beyond the scope of this thesis.*

2.1.2 Regime of A Small Gradient

For reasons outlined in the previous section the rest of this chapter assumes a full gradient and full descent (GD) only unless stated explicitly.

As will be shown shortly when average gradient $\nabla \ell$ gets small at some entry time t_0 (the following sequence of) Hessian H_{w_t} for $t > t_0$ is approximated by polynomial function of $H_{w_{t_0}}$. Under such regime we are able to track approximately the evolution of spectrum of H_t over time.

First we derive the relation between the H_{w_t} and $H_{w_{t_0}}$ for $t > t_0$. A spectral density transformations and their dynamics are described in the follow up at the rest of the section.

2.1.2.1 Transformations of initial spectral density by GD

Consider n eigenvalues of real symmetric matrix H_t of dimension n, denoted $\lambda_1^t(H_t) < \lambda_2^t(H_t) < \cdots < \lambda_n^t(H_t)$. Because H_t's for all t are symmetric or

Hermitian, these eigenvalues are all real. Further because of the Assumption 2.1, No.5, (the arbitrary high precision) all eigenvalues are distinct. In a realm of deep learning this corresponds to random initialization producing distinct values almost surely. These values are then mapped by diffeomorphisms evaluated at high precision and thus are distinct along the training almost surely (w.r.t. initialization probability distribution).

When H_{w_0} is random Wigner matrix with i.i.d. elements from any probability distribution with finite moments we'd have initial spectral distribution given by semicircle probability distribution given by

$$\sigma(\lambda)d\lambda = \frac{1}{2\pi}\sqrt{4-x^2}\mathbf{1}_{|\lambda|\leq 2}d\lambda. \qquad \text{(semi-circle law)}$$

Never the less work of [Baskerville et al., 2022] and links therein points out that Hessians in deep learning applications do not follow this law (we'll shortly see why). While it is empirically shown for later stages of the training it is unclear how much the spectrum after random initialization differs from (semi-circle law). Because of this uncertainty the thesis does not make use of the result above. We thus also omit the random theory definitions and refer reader to [Feier, 2012] for proof and more in case of interest. We leave this for future work and proceeds assuming general "arbitrary" spectrum of H_{w_0} with finite support.

Will make use of the following known lemma stating that polynomial function h applied on real symmetric diagonizable matrix gives matrix with eigenvalues $h(\lambda)$.

Lemma 2.3. *(Eigenvalues of $h(A)$) Consider polynomial matrix function $h(\boldsymbol{z}) = \sum_k \alpha_k \boldsymbol{z}^k$ and assume a $n \times n$ real symmetric matrix A. Then eigenvalues of matrix of $h(A)$ are given by*

$$\{h(\lambda_i(A))\}_{i=1}^n. \tag{2.12}$$

Proof. By spectral theorem [Bhatia, 1997, Bishop, 2006] real symmetric matrix is diagonalizable with respect to an orthonormal basis, $A = UDU^{-1}$ where D is a diagonal matrix with eigenvalues on the diagonal. Then we have

$$\begin{aligned} h(D) &= \sum_{k=0}^m \alpha_k D^k = \sum_{k=0}^m \alpha_k \text{diag}(\lambda_1, \ldots, \lambda_n)^k = \sum_{k=0}^m \alpha_k \text{diag}(\lambda_1^k, \ldots, \lambda_m^k) \\ &= \text{diag}(\sum_{k=0}^m \alpha_k \lambda_1^k, \ldots, \sum_{k=0}^m \alpha_k \lambda_n^k) = \text{diag}(h(\lambda_1), \ldots, h(\lambda_n)) \end{aligned} \tag{2.13}$$

Because of U are orthonormal $(UDU^{-1})^2 = (UDU^{-1})(UDU^{-1}) = (UD^2U^{-1})$

and so on, and h is polynomial we have

$$h(A) = h(UDU^{-1}) = U\mathrm{diag}(h(\lambda_1), \ldots, h(\lambda_n))U^{-1}, \tag{2.14}$$

i.e., $h(A)$ is similar to
$\mathrm{diag}(h(\lambda_1), \ldots, h(\lambda_n))$ and the only eigenvalues are $\{h(\lambda_i(A))\}_{i=1}^n$. □

To make use of this lemma we need to show that GD can be represented as polynomial function applied on eigenvalues of H_{w_0}. While this does not hold in general the following lemma shows that it approximately holds when (average) gradient $\nabla\ell$ is small.

Recall that gradient of loss $\nabla\ell$ is at GD evaluated as average gradient over training data. This means that while gradient $\nabla\ell$ may be small for instance at local or global optimum its variance (inverse Hessian) need not. In fact under Gaussian assumption, which for instance can be invoked by central limit theorem, these two moments are independent.

Let's assume GD (average) gradient $\|\nabla\ell\|_2 \leq \epsilon$ at some entry time t_0 and remains under this threshold until some time denoted $T < \infty$. Following lemma as a direct consequence of the Lemma 2.2 approximates Hessian at T by polynomial function of $H_{w_{t_0}}$.

Lemma 2.4. *Under assumptions of the Lemma (2.2) assume gradient* $\|\nabla\ell\|_2 \leq \epsilon$ *at time* t_0 *and remains under this threshold until time step* $T \in \mathbb{N}^+, T < \infty$.

Then the Hessian of the loss H_{w_T} *at step* T *is approximated by* $1/P(H_{w_{t_0}})$, *where* P *is a polynomial*[4] *matrix function of a finite degree given by*

$$H_{w_T} = \underbrace{\prod_{t=t_0}^{T-1} L_t(H_{w_{t_0}})}_{\stackrel{def}{=} 1/P(H_{w_{t_0}})} + \varepsilon(T - t_0)^2/2 \tag{2.15}$$

and $L_t(\cdot)$*'s are given by a recurrent relation*

$$L_t(Z) = \prod_{k=t_0}^{t} \left(I - \eta_k L_{k-1}(Z)\right)^{-2} \tag{2.16}$$

[4] One could define instead of polynom P a Laurent series $1/P$. In general Laurent series are defined as a formal power series of a form $\sum_{n\in\mathbb{Z}} a_n x^n$ which are allowed to have a finite number of negative exponents.

The order of a formal Laurent series is defined as the smallest n such that $a_n \neq 0$. This is kind of like the degree of a polynomial, but for negative integers. The degree of a formal Laurent series is defined in the same way as the degree of a polynomial, though the degree may not exist (since all of the a_n for $n > 0$ are still allowed to be nonzero).

where error terms of ϵ^2 and higher are neglected. The ε is the "error" real symmetric matrix from the Lemma 2.2.

Proof. Proof follows by the Lemma 2.2 applied on $H_{w_0} = H_{w_{t_0}}$ and error manipulation using Taylor series of the first order in terms of ϵ and neglecting higher terms.

To simplify the notation of the proof assume without loss of generality that $t_0 = 0$. We prove the lemma by induction in T.

Step ($T = 1$): For $H_{w_0} = H_{w_{t_0}}$ and $T = 1$ the Lemma (2.2) gives the Hessian H_{w_1} is approximated by following polynomial function of $H_{w_{t_0}}$

$$\begin{aligned} H_{w_1} =& \left(I - \eta_k H_{w_0}\right)^{-1} H_{w_0} \left(I - \eta_k H_{w_0}\right)^{-1} + \varepsilon \\ =& H_{w_0} \left(I - \eta_k H_{w_0}\right)^{-2} + \varepsilon \stackrel{def.}{=} L_{t_0}(H_{w_{t_0}}) + \varepsilon \end{aligned} \tag{2.17}$$

where the (Laurent polynom) $L_{t_0}(H_{w_{t_0}})$ is defined in the statement and ε is from the Lemma 2.2. The last step follows from the fact that if the product of two symmetric matrices is symmetric, then they must commute (they are all unitarily diagonizable (by SVD) and because a change of orthonormal basis does not change eigenvalues, they are simultanously unitarily diagonizable, [Gantmakher, 1959]). We obtained the statement of the lemma for $t_0 = 1$ and $T = 1$ which concludes the first part of the proof.

Step ($T - 1 \to T$): First part of (2.15) follows exactly by the same arguments as the previous step, i.e. Lemma 2.2 applied on $t_0 = T - 1$ and T same as in this step.

For the "error" part involving ε we note that an induction step adds one element to the product (2.15) defined by recurrent definition of $L_T(Z) = \left(I - \eta_k L_{T-1}(Z)\right)^{-2}$, i.e., (2.16), is a product of $T - t_0$ terms every of which has an error ε by the Lemma 2.2 and neglecting terms ϵ^2 and higher. Thus the step T adds $(T - t_0 - 1)\varepsilon$ to the overall error sum. The statement follows from summing up the finite series $\sum_{t=t_0}^{T} (T - t_0)\varepsilon = \varepsilon(T - t_0)^2/2$ proving the induction step and the statement. □

2.1.3 Analysis in a Small Gradient Regime of GD

The previous lemma 2.4 provides an insight to an evolution of a spectrum of Hessian under a small gradient regime. The rest of the chapter is dedicated to

analysis of the Lemma 2.2 and Lemma 2.4 and its consequences for gradient descent (GD) back-prop for general loss objectives.

Definition 2.5. *Given notation of this chapter* bending *coefficient is defined as follows*

$$\xi(\lambda) := (1 - \eta_t \lambda)^{-2}. \tag{2.18}$$

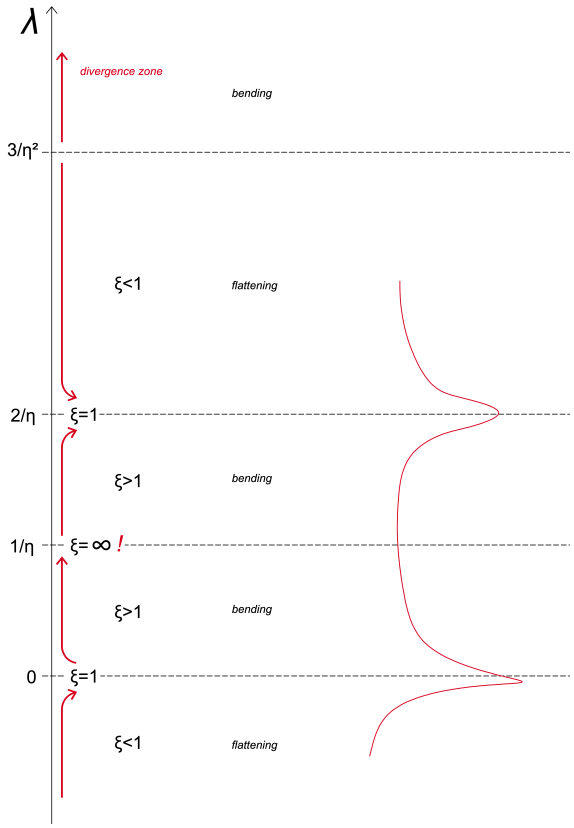


Figure 2.1: *Flattening/bending effect of an GD update on the Hessian eigen-spectrum at time t, i.e., spectrum of H_{w_t}. ξ is a bending coefficient from Definition 2.18 as a function of λ. Eigenvalues are depicted on y-axis. Boundaries between flattening/bending zones are horizontal dashed lines and occur at levels given by η_t,i.e., $(0, \frac{1}{\eta_t}, \frac{2}{\eta_t}, \frac{3}{\eta_t^2})$. Lines in red depicts an update direction on the eigenvalues within that range. Right most is a sketch of cumulation of eigenvalues over training (a sort of eigenspectrum ghost).*

Overall under assumptions of the Lemma 2.4 we can deduct following dynamics of the spectrum of Hessian during GD back-prop training also captured in Fig.2.1:

- H_T has negative eigenvalues almost surely w.r.t. probability distribution used to initialize weights. This is due to the first element H_{w_0} that is present in (2.2). The spectrum of this initial element of recurrence relation H_{w_0} has, given by a random initialization around zero [Sutskever et al., 2013, Glorot and Bengio, 2010, Goodfellow et al., 2016], negative eigenvalues almost surely. In polynom P it only multiplies positive second powers of the spectrum of $\left(I - \eta_t H_{w_t}(B_t)\right)^{-1}$ because the Jacobian of every step is present twice in the product. Hence even close to optimum H_T has always escape path towards the lower loss levels.

  The line of thoughts above only works if gradients have been low, i.e., under the threshold over the whole training, which is barely the case in practice. On the other hand, recent works on the Hessian spectrum, [Cohen et al., 2021, Yao et al., 2020] show that Hessian loss has a bunch of negative eigenvalues (be they small) during the whole training process. These experiments suggest the regime of large gradients acts rather as a "shake up" of the spectrum (possibly due to large ε's) yet it seems to always leave a significant portion of eigenvalues negative (Note this is empirical observation only). If so, the small gradient regime then keeps them negative as argued above, possibly enabling a steady path to low loss levels, [Choromanska et al., 2015].

  It does not grant global minimum convergence in finite time though, yet gradients updates of vanilla GD are mostly along the largest eigen-directions and further analysis is needed.

- Another interesting point raised in [Cohen et al., 2021] is that GD does not diverge when sharpness at time T, i.e, $\lambda_m^T ax$, gets above the level of $\frac{2}{\eta_t}$ as it should based on a quick analogy with one dimensional quadratic objective $l(x) = \frac{1}{2}ax^2 + bx + c$, where GD with step size η will diverge if a exceeds the threshold $2/\eta$, as showed in [Cohen et al., 2021]. It shown there that update step $x_t = (1 - \eta a)^t (x_0 - x^*) + x^*$ where x^* is the optimum. In case $a > \frac{2}{\eta}$ sequence of $\{x_t\}$ will oscillate around x^* with ever-increasing magnitude. Notably, in our case of eigenvalue sequence, we rather have an update of the form $\lambda^{t+1} = \xi(\lambda^t) = (1 - \eta\lambda^t)^{-2}$, i.e. reciprocal to the quadratic case.

  Therefore, instead of diverging, whenever $\lambda^t \geq \frac{2}{\eta_t}$ the update is $\xi(\lambda^t) \leq 1$ resulting in next λ^{t+1} in sequence being lowered (dubbed "flattening" of the Hessian in Fig.2.1 as opposed to the "bending" opposite case of $\xi > 1$) back under the level $\frac{2}{\eta_t}$ along the red arrows depicted in Fig.2.1. Therein, depending on where the λ^t on the real axis is, the (red) arrows show the direction of the update $\lambda^{t+1} - \lambda^t$ holistically.

  As a consequence, the GD is "bouncing back" from large "sharpness" regions imposing stability above the level of $\frac{2}{\eta}$. Note that divergence

happens approximately above the level $\frac{3}{\eta^2}$ when considering the increase of λ's, i.e., of the sharpness or curvature of the Hessian, incurred in the "out of small gradients regime", i.e., $\nabla\ell \leq \epsilon$ from the Lemma is violated. This is because increasing curvature of Hessian leads to larger gradients and at one point they happen to be large enough to increase sharpness (via weight update) more than (first-order) dumping effect of $\xi(\lambda^t)$ can balance. Nevertheless, we omit the details for brevity.

- Singularity of $\xi(\lambda^t)$ at $\lambda_t = 1/\eta_t$. At this point, sharpness explodes. On the other hand, with a fixed learning rate the GD will hit this value exactly with zero probability. Nevertheless, especially with a small learning rate that slowly but steadily increases sharpness due to "bending" updates of the spectrum, see Fig.2.2, it will get to its vicinity very likely. Depending on the closeness to $1/\eta_t$ such a step would eject GD from the "small gradient regime". Notably, when learning rate η_t decays, as is common practice in DL, or GD hits the loss plateau, the updates $\xi(\lambda^t)$ become negligible while points $(0, \frac{1}{\eta_t}, \frac{2}{\eta_t}, \frac{3}{\eta_t^2})$ get far apart leading to a stagnating spectrum.

The results above are well supported by the above-mentioned paper [Cohen et al., 2021] the Figure 1 of which is also brought in here and presented in Fig.2.2 for convenience.

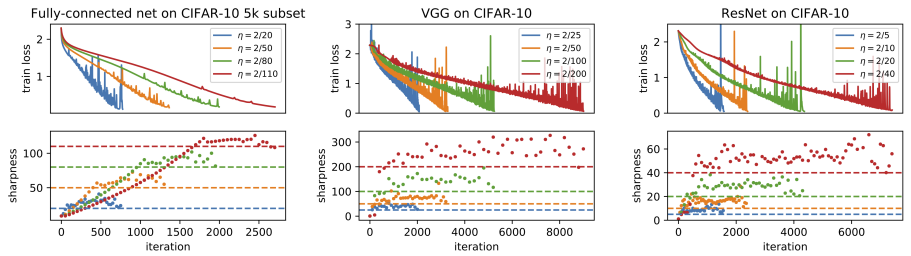


Figure 2.2: **Figure 1** *from [Cohen et al., 2021] showing the sharpness, i.e., the largest eigenvalue of the Hessian, converging and hovering above the level of* $2/\eta_t$ *supporting experimentally the theoretical results of this chapter.*

2.1.3.1 Conclusions

As follows from the discussion above under the small gradient regime spectrum of the Hessian tends to cumulate around two points, 0 and $\frac{2}{\eta}$ as also suggested visually in Fig.2.1 by the (red) curve on the right. Despite the evolution has been derived under some weak assumptions 2.1 and more stringent ones on "gradient

norm" and smoothness, as stated in the Lemma 2.4, it supports the recent empirical results in mentioned works [Cohen et al., 2021, Yao et al., 2020, Geiger et al., 2019] and provides some new perspectives on the curvature of the loss (the Hessian).

On the upside, the results of this Chapter are valid for general loss functions and provide new and detailed perspectives on the spectrum of the Hessian. One of its main downsides is that it only covers a full gradient descent back-prop training while stochasticity is argued and supported by experiments to play a major role in the generalization of deep networks. This is the topic the next section addressed at length at the cost of limiting our focus to a wide class of the Bregman divergence losses.

Chapter 3

Self Regularized Bregman Objective (SeReBrO)

. . . Suddenly she came upon a little three-legged table, all made of solid glass; there was nothing on it except a tiny golden key, and Alice's first thought was that it might belong to one of the doors of the hall; but, alas! either the locks were too large, or the key was too small, but at any rate it would not open any of them. However, on the second time round, she came upon a low curtain she had not noticed before, and behind it was a little door about fifteen inches high: she tried the little golden key in the lock, and to her great delight it fitted! . . .

Alice finding tiny door behind the curtain, Alice's Adventures in Wonderland, Charles Ludtwidge Dodgson (Lewis Carroll, 1865)

3.0.1 Preliminaries on Bregman Divergences

From now on for the rest of the thesis and for reasons to be revealed shortly let a loss to be the Bregman divergence [Amari, 2016, Banerjee et al., 2005] defined as follows

$$\ell(\boldsymbol{z}, \boldsymbol{y}) = d_\Phi(\boldsymbol{z}, \boldsymbol{y}) = \Phi(\boldsymbol{z}) - \Phi(\boldsymbol{y}) - \langle \boldsymbol{z} - \boldsymbol{y}, \nabla_y \Phi(\boldsymbol{y}) \rangle \tag{3.1}$$

, where $\Phi : \mathbb{R}^d \to \mathbb{R}$ is a strictly convex function.

The definition of the *generalization error* holds and was defined in generalization error. An important property of the Bregman divergence is that its derivative w.r.t the first argument at datum $(\boldsymbol{x}_s, \boldsymbol{y}_s)$ evaluates as

$$\nabla_x d_\Phi(x, y) = \nabla\Phi(\boldsymbol{x}) - \nabla\Phi(\boldsymbol{y}) \tag{3.2}$$

Further it can be shown that there exist an isomorphic dual space such that

$$d_\Phi(\boldsymbol{z}, \boldsymbol{y}) = d_\Psi(\nabla\Phi(\boldsymbol{y}), \nabla\Phi(\boldsymbol{z})) \tag{3.3}$$

, where Ψ is a convex conjugate to Φ. For more details see [Hiriart-Urruty and Lemaréchal, 2012].

Definition 3.1 (Convex sets and Cones)**.** *A set $C \subseteq \mathbb{R}^d$ is* convex *if for all $x, y \in C$ and $\alpha \in [0, 1]$, the set C also contains the point $\alpha x + (1 - \alpha)y$. A* cone *K is a set such that for any $x \in K$, the ray $\{|\lambda > 0\}$ also belongs to K. A* convex cone *is a cone that is also convex.*

Definition 3.2 (Convex and Affine Hulls)**.** *A* linear combination *of elements $x_1, x_2, \ldots, x_k$ from set S is a sum $\sum_{i=1}^{k} \alpha_i x_i$, for arbitrary scalars $\alpha_i \in \mathbb{R}$. An* affine combination *is a linear combination with a restriction $\sum_{i=1}^{k} \alpha_i = 1$ and a* convex combination *is an affine combination with further restriction $\alpha_i \geq 0$ for all $i = 1, \ldots, k$. The* affine hull *of a set S, denoted $aff(S)$, is the smallest set that contains all affine combinations. Similarly, the* convex hull *of a set S, denoted $conv(S)$, is the smallest set that contains all its convex combinations. Note that $conv(S)$ is a convex set by definition.*

Definition 3.3 (Polyhedra)**.** Polyhedron *is a set that can be represented as the intersection of a finite number of half-spaces*

$$P = \{x \in \mathbb{R}^d | \langle a_j, x \rangle \leq b_j, \quad \forall j \in \mathcal{J}\} \tag{3.4}$$

where for each $j \in \mathcal{J}$, the pair (a_j, b_j) parameterizes a particular half-space. A bounded polyhedron is called polytope*.*

Definition 3.4 (Shannon entropy)**.**

$$H(p) = \int_{\mathcal{X}} \log p(x) p(x) \nu(dx) \tag{3.5}$$

w.r.t. some base measure ν*.*

Definition 3.5. *(see. [Rashid et al., 2019]) Let* $f : K \subset \mathbb{R} \to \mathbb{R}$ *is exponentially convex function, if* f *is positive and* $\forall u, v \in K$ *and* $t \in [0, 1]$ *we have*

$$e^{f((1-t)u+tv)} \leq (1-t)e^{f(u)} + te^{f(v)} \tag{3.6}$$

Alternative and higher dimensional definition of exponentially convex function follows (see[Banerjee et al., 2005]).

Definition 3.6. *A function* $f : \Theta \to \mathbb{R}^{++}, \Theta \subset \mathbb{R}^d$ *is called exponentially convex if the kernel* $Kf(a, b) = f(a + b)$*, where* $a + b \in \Theta$*, satisfies,*

$$\sum_i \sum_j K(\theta_i, \theta_j) u_i \overline{u}_j \geq 0 \tag{3.7}$$

for any set $\{\theta_1, \theta_2, \ldots, \theta_n\} \subseteq \Theta$*,* $\forall i, j$ *and* $\{u_1, \ldots, u_n\} \subseteq \mathbb{C}$ *and* $\overline{u}_j$ *denotes complex copnjugate of* u_j*, i.e., kernel* $K(\theta_i, \theta_j)$ *is positive semi-definite.*

Definition 3.7. *(Regular Bregman divergence, [Banerjee et al., 2005]) Let* $f : \Omega \to \mathbb{R}^{++}, \Omega \subset \mathbb{R}^d$ *be a continuous exponentially convex function such that* Ω *is open and* $\Psi(\theta) = \log(f(\theta))$ *is strictly convex. Let* Φ *be the conjugate function of* Ψ*. Then we say that the Bregman divergence* d_Φ *derived from* Φ *is a regular Bregman divergence.*

[Banerjee et al., 2005] proves that there is a bijection between regular exponential families and regular Bregman divergences in a form of (3.25). The crux of the argument relies on results in harmonic analysis connecting positive definiteness to integral transforms. In particular, following result due to [Devinatz, 1955], and taken from [Banerjee et al., 2005], is used that relates exponentially convex functions to Laplace transforms of bounded non-negative measures.

Definition 3.8 (Push-forward measure)**.** *Given measurable spaces* $X_1, \mathcal{S}_1$ *and* $X_2, \mathcal{S}_2$*, a measurable mapping* $f \colon X_1 \to X_2$ *and a measure* $\mu \colon \mathcal{S}_1 \to [0, +\infty]$*, the* push-forward *of* μ *is defined to be the measure*

$$f_*(\mu) \colon \mathcal{S}_2 \to [0, +\infty] \tag{3.8}$$

given by

$$(f_*(\mu))(B) = \mu\left(f^{-1}(B)\right) \text{ for } B \in \mathcal{S}_2. \tag{3.9}$$

Theorem 3.9 (3.6.1. Theorem, [Bogachev and Ruas, 2007]). *Under notation of Definition 3.8 of push-forward measure ($f_*(\mu)$ above, for every $\mathcal{S}_2$-measurable function on X_2 following holds*

$$\int_{X_2} u(y) df_*(\mu)(y) = \int_{X_1} u(f(x)) d\mu(x) \tag{3.10}$$

if at least one of the sides is well defined.

Theorem 3.10. *([Devinatz, 1955]) Let $\Omega \subset \mathbb{R}^d$ be an open convex set. A necessary and sufficient condition that there exists a unique, bounded, non-negative measure ν such that $f : \Omega \to \mathbb{R}^{++}$ can be represented as*

$$f(\boldsymbol{\theta}) = \int_{\mathbb{R}^d} e^{\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle} d\nu(\boldsymbol{x}) \tag{3.11}$$

is that f is continuous and exponentially convex.

Basics of Exponential Families we use definitions and notation as close as possible to an inspiring and exciting work of [Wainwright and Jordan, 2008].

Exponential family is a parameterized family of densities, taken with respect to some underlying (base) measure.

Given random vector $(X_1, X_2, \ldots, X_m)$ taking values in some space $\mathcal{X}^m = \otimes_{s=1}^m \mathcal{X}_s$, let be $t = (t_\alpha, \alpha \in \mathcal{I})$ be a collection of functions $t_\alpha : \mathcal{X} \to \mathbb{R}$, so-called *sufficient statistics* and where $\mathcal{I}$ is a index set with $d = |\mathcal{I}|$ elements to be specified so that $t(X)$ defines vector-valued map from $\mathcal{X}^m \to \mathbb{R}^d$. For a given vector of sufficient statistics t_α, let $\theta = (\theta_\alpha, \alpha \in \mathcal{I})$ a vector of *canonical* or *natural* parameters. For fixed $x \in \mathcal{X}^m$ the $\langle \theta, t(x) \rangle$ denotes Euclidean inner product in $\mathbb{R}^d$ of the two vectors θ and $t(x)$.

With this notation, the *exponential family* associated with t consists of the following parameterized collection of density functions

$$p_{\theta,\Psi}(x_1, x_2, \ldots, x_m) = \exp\{\langle \theta, t(x) \rangle - \Psi(\theta)\}, \tag{3.12}$$

defined with respect to $d\nu$. The function Ψ, known as *log partition* or *cumulant function*, is defined by the integral

$$\Psi(\theta) = \int_{\mathcal{X}^m} \exp\langle \theta, t(x) \rangle \nu(dx). \tag{3.13}$$

The canonical parameters θ belong to the set

$$\Omega := \{\theta \in \mathbb{R}^d | \Psi(\theta) \leq +\infty\}. \tag{3.14}$$

Corresponding mean value parameter space is defined as

$$\mathcal{M} := \{\boldsymbol{\mu} \in \mathbb{R}^d | \exists p \text{ s. t. } E_p[\boldsymbol{t}(\boldsymbol{x})] = \boldsymbol{\mu}\}. \tag{3.15}$$

Notably density p, defined with respect to the underlying measure ν, *is not* restricted to the exponential family associated with sufficient statistics $t(x)$ and base measure ν. However it turns out that under suitable tehnical conditions, this vector provides an alternative parameterization of this exponential family, see [Wainwright and Jordan, 2008], chapter 3.

Embraced with definitions above we can characterize specifics of exponential families defined above.

Regular families is a exponential family for which domain Ω is an open set.

Minimal is an exponential family such that its sufficient statistics are linearly independent, i.e., there does *not exist* vector $\boldsymbol{a} \in \mathbb{R}^m$ such that

$$\langle \boldsymbol{a}, \boldsymbol{t}(x) \rangle = \sum_{\alpha \in \mathcal{I}} a_\alpha t_\alpha(x) = c \tag{3.16}$$

almost everywhere with regards to $\nu(x)$ and where c is a real constant. Bold font is used to emphasize that all elements involved are vectors. This gives rise to *minimal representation* of the exponential family where each distribution is given by a unique parameter vector θ.

Overcomplete is a exponential family that is not minimal in the sense of (3.16). In such representation there exist affine subsets of Ω that are associated with the same distribution.

Note that exp. family is defined with respect to some carrier measure. Then density p_0 correspond to Radon-Nykodym derivative $\frac{dP_0(\omega)}{d\lambda(\omega)}$ where P_0 is absolutely continuous w.r.t. the Lebesque or counting measure λ for continuous and discrete r.v. respectively in an alignment with [Wainwright and Jordan, 2008], [Banerjee et al., 2005].

It can be easily seen that if $\boldsymbol{x} \in \mathbb{R}^d$ denotes the natural statistic $T(\omega)$, then the probability density function $g(x; \theta)$ (with respect to the appropriate measure dx) given by

$$g(\boldsymbol{x}; \theta) = \exp\left(\langle \boldsymbol{\theta}, \boldsymbol{x} \rangle - \Psi(\boldsymbol{\theta})\right) dP_0(\boldsymbol{x}) \tag{3.17}$$

is such that $f(\omega; \theta)/g(\boldsymbol{x}; \theta)$ does not depend on θ. Thus, $\boldsymbol{x}$ is a sufficient statistic [Amari, 2016] for the family, and in fact, can be shown (Barndorff-Nielsen, 1978) to be minimally sufficient.

Example, Gaussian 1D For instance, the natural statistic for the one-dimensional Gaussian distributions denoted by

$$f(\omega;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{(\omega-\mu)^2}{\sigma^2} \tag{3.18}$$

is given by $x = [\omega, \omega^2]$ and the corresponding natural parameter turns out to be $\theta = [\mu/\sigma^2, 1/2\sigma^2]$, which can be easily verified to be minimally sufficient.

In this thesis it is convenient to work with the minimal natural sufficient statistic $\boldsymbol{x}$ and hence, we redefine regular exponential families in terms of the probability density of $\boldsymbol{x} \in \mathbb{R}^d$, noting that the original probability space can actually be quite general.

This generality is for instance reflected in formulation of exponential family in a form

$$\exp\left(\langle \boldsymbol{\theta}, \boldsymbol{x}\rangle - \Psi(\boldsymbol{\theta})\right) dH(\boldsymbol{x}) \tag{3.19}$$

, i.e., with respect to measure $H(\boldsymbol{x})$ as opposed to Lebesgue $d\boldsymbol{x}$. Suppose H is a non-decreasing function of a real variable. Then Lebesgue–Stieltjes integrals with respect to $\mathrm{d}\, H(\mathbf{x})$ are integrals with respect to the reference measure of the exponential family generated by H. When the reference measure is finite, see also Theorem 5 in [Banerjee et al., 2005], it can be normalized and H is actually the cumulative distribution function of a probability distribution. If H is absolutely continuous it can be written $\mathrm{d}\, H(x) = h(x)\,\mathrm{d}\, x$ so the formulas reduce to that of the previous paragraphs. Alternatively, we can write the probability measure directly as

$$P(d\boldsymbol{x};\theta) = \exp\left(\langle \boldsymbol{\theta}, \boldsymbol{x}\rangle - \Psi(\boldsymbol{\theta})\right) \nu(d\boldsymbol{x}) \tag{3.20}$$

where $h(\boldsymbol{x})$ is a Radon-Nykodym derivative of ν with respect to Lebesgue measure dx, i.e., $d\nu = h(\boldsymbol{x})d\boldsymbol{x}$. Then we can write both (3.19) and (3.20) as

$$\exp\left(\langle \boldsymbol{\theta}, \boldsymbol{x}\rangle - \Psi(\boldsymbol{\theta})\right) h(\boldsymbol{x})d\boldsymbol{x} \tag{3.21}$$

and analogically for counting measure instead of $d\boldsymbol{x}$.

Table 3.1 from p.42 [Wainwright and Jordan, 2008] enlists some common distributions of the exponential family, including "Gaussian Location family" which is the imposed distribution when Squared loss is used in deep learning as will shortly be shown.

3.0.1.1 Mean Parameter Space

The previous definition 3.15 of Mean Parameter Space, taken from [Wainwright and Jordan, 2008], is used in the following.

Family	$\mathcal{X}$	ν, $h(\cdot)$	*Suff. stats* $\langle\theta, t(x)\rangle$	$\Psi(\theta)$
Bernoulli	$\{0,1\}$	Counting	θx	$\log(1+\exp(\theta))$
Gaussian Location f.	$\mathbb{R}$	Lebesgue, $\frac{\exp\{-x^2/2\}}{\sqrt{2\pi}}$	θx	$\frac{1}{2}\theta^2$
Gaussian	$\mathbb{R}$	Lebesgue, $\frac{1}{\sqrt{2\pi}}$	$\theta_1 x + \theta_2 x^2$	$-\frac{\theta_1^2}{4\theta_2^2} - \frac{1}{2}\log(-2\theta_2)$
Exponential	$\{0,+\infty\}$	Lebesgue	θx	$-\log(\theta_1)$
Poisson	$\mathbb{N}\cup\{0\}$	Counting $\frac{1}{x!}$	θx	$\exp(\theta)$
Beta	$(0,1)$	Lebesgue	$\theta_1 \log x + \theta_2 \log(1-x)$	$\sum_{i\in\{1,2\}} \log\Gamma(\theta_i+1) - \log\Gamma(\sum_{i\in\{1,2\}}(\theta_i+1))$

Table 3.1: *Several well-known classes of scalar random variables as exponential families.*

Let p be a given density defined with respect to the underlying base measure ν; for the moment, we do not assume that p is a member of an exponential family defined with respect to ν. The mean parameter μ_α associated with a sufficient statistic $t_\alpha : \mathcal{X}^m \to \mathbb{R}$ is defined by the expectation

$$\mu_\alpha = E_p[t_\alpha(X)] = \int t_\alpha(x)p(x)\nu(dx), \quad \forall\alpha \in \mathcal{I} \tag{3.22}$$

In this way, we define a vector of mean parameters $(\mu_1, \mu_2, \ldots, \mu_d)$, one for each of the $|\mathcal{I}| = d$ sufficient statistics t_α, with respect to an arbitrary density p. An interesting object is the set of all such vectors $\mu \in \mathbb{R}^d$ traced out as the underlying density p is varied.

More formally, corresponding mean value parameter space is defined in 3.15 as

$$\mathcal{M} := \{\boldsymbol{\mu} \in \mathbb{R}^d | \exists p \text{ s. t. } E_p[\boldsymbol{t}(\boldsymbol{x})] = \boldsymbol{\mu}\}. \tag{3.23}$$

corresponds to all realizable mean parameters. It is important to note that in this definition, we have not restricted the density p to the exponential family associated with the sufficient statistics t and base measure ν. However, it turns out that under suitable technical conditions, this vector provides an alternative parameterization of this exponential family.

Example [Gaussian, (3.18), continued] We illustrate the mean value parameter space $\mathcal{M}$ on a d-dimensional Gaussian distribution, introduced for $d = 1$ in 3.18, just with a change of notation from ω to column vector $\boldsymbol{x} \in \mathbb{R}^d$. Then sufficient statistics become $\boldsymbol{t}(x) = [\boldsymbol{x}, \langle\boldsymbol{x}, \boldsymbol{x}\rangle]$.

For this particular model, it is straightforward to characterize the set $\mathcal{M}$ of globally realizable mean parameters (μ, Σ). We begin by recognizing that if (μ, Σ) are realized by some distribution (not necessarily Gaussian), then $\Sigma - \mu\mu^T$) must be a valid covariance matrix of the random vector X, implying that the positive semidefiniteness (PSD) condition $\Sigma - \mu\mu^T \succcurlyeq 0$ must hold. Conversely, any pair (μ, Σ) for which the PSD constraint holds, we may construct a multivariate Gaussian distribution with mean μ, and (possibly degenerate) covariance $\Sigma - \mu\mu^T$, which by construction realizes (μ, Σ).

Thus, we have established that for a Gaussian Markov random field, the set $\mathcal{M}$ has the form

$$\mathcal{M} = \{(\mu, \Sigma) \in \mathbb{R}^d \times PSD^d | \Sigma - \mu\mu^T \succcurlyeq 0\} \tag{3.24}$$

where PSD^d denotes the set of $d \times d$ symmetric positive definite matrices. Figure 3.1 illustrates this set in the scalar case ($d = 1$).

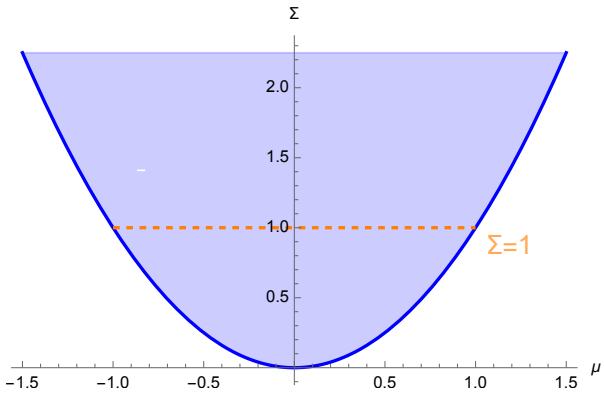


Figure 3.1: *Blue filled area depicts a set $\mathcal{M}$ for one dimensional Gaussian distribution $N(\mu, \Sigma)$ given by contraint on positive definitness $\Sigma - \mu\mu^T \succcurlyeq 0$ that is $\Sigma \geq \mu^2$ in this $d = 1$ dimensional case. The orange level line shows the range $(-1, 1)$ of the mean parameter μ that is "allowed" under exponential family incurred by strictly convex function $\Phi(\mu) := \frac{1}{2}\langle\mu, \mu\rangle$, i.e., in one dimensional case $\mu^2/2$. All other values are not*

Notably the mean parameter space does not span the whole available space, in this case $\mathbb{R}^2$, but is a (convex) **subspace** of it.

As shown further in [Wainwright and Jordan, 2008] and for brevity is also presented here, the set $\mathcal{M}$ is always a convex subset of $\mathbb{R}^d$. Indeed, if $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$ are both elements of $\mathcal{M}$, then there must exist distributions p and p' that realize them, meaning that $E_p[\boldsymbol{t}(\boldsymbol{x})] = \boldsymbol{\mu}$ and $E'_p[\boldsymbol{t}(\boldsymbol{x})] = \boldsymbol{\mu}'$. For any $\lambda \in [0, 1]$, the

convex combination $\boldsymbol{\mu}(\lambda) := \lambda\boldsymbol{\mu}+(1-\lambda)\boldsymbol{\mu}'$ is realized by the mixture distribution $\lambda p + (1-\lambda)p'$, so that $\boldsymbol{\mu}(\lambda)$ also belongs to $\mathcal{M}$.

It can also be shown by Minkowski-Weyl theorem, that convex polytope $\mathcal{M}$ is equal to the intersection of a finite collection of half-spaces, see [Wainwright and Jordan, 2008], p.54 and Appendix A.2.

This property of $\mathcal{M}$ stating that all mean parameters realizable by some distribution (and hence also by exponential family distribution, see [Wainwright and Jordan, 2008]) are constrained to lie in the convex subspace, turns out to be essential for implict regularization of neural networks, as will be shown shortly.

Bijection of Regular Exponential Families and Bregman Divergences

As presented in [Banerjee et al., 2005], Theorem 4, there exists a one-to-one mapping between the regular exp. family of distributions $p_{(\Psi,\boldsymbol{\theta})}(\boldsymbol{x})$ generated by sufficient statistics, reference measure and Bregman div. $d_\Phi(\boldsymbol{x},\boldsymbol{y})$,

$$p_{(\Psi,\boldsymbol{\theta})}(\boldsymbol{x}) = \exp(-d_\Phi(\boldsymbol{x},\boldsymbol{\mu}))b_\Phi(\boldsymbol{x}) \tag{3.25}$$

where the related exponential family has the following form

$$g(\boldsymbol{x};\theta) = \exp\left(\langle\boldsymbol{\theta},\boldsymbol{x}\rangle - \Psi(\boldsymbol{\theta})\right) dP_0(\boldsymbol{x}) \tag{3.26}$$

and

$$b_\Phi(\boldsymbol{x}) = \exp(\Phi(\boldsymbol{x}))p_0(\boldsymbol{x}) \tag{3.27}$$

is uniquely determined given the measure $P_0(\omega)$ and $dP_0(\omega) = p_0(\boldsymbol{x})d\boldsymbol{x}$. In this case $p_0(\boldsymbol{x})$ denotes reference measure corresponding to $h(x) = \frac{d\nu}{dx}$ from (3.21).

Because relation between Bregman divergences and exponential families is essential to this thesis in what follows we show the relation of the right hand sides of (3.25) and (3.17) in detail.

Let *mean* and *natural* parameters be denoted $\boldsymbol{\mu}$ and $\boldsymbol{\theta}$ respectively. Since Φ and Ψ are Legendre (convex) duals there are also following known properties, see. [Wainwright and Jordan, 2008]

$$E_{\boldsymbol{\theta}}[\boldsymbol{x}] = \boldsymbol{\mu}(\boldsymbol{\theta}) \tag{3.28}$$
$$\nabla\Psi(\boldsymbol{\theta}) = \boldsymbol{\mu}, \nabla\Phi(\boldsymbol{\mu}) = \boldsymbol{\theta} \tag{3.29}$$

for $\boldsymbol{\mu} \in int(dom(\Phi))$ so that $\nabla\Phi$ exists.

The conjugate function can be expressed as $\Phi(\boldsymbol{\mu}) = \langle \nabla\Phi(\boldsymbol{\mu}), \boldsymbol{\mu}\rangle - \Psi(\nabla\Phi(\boldsymbol{\mu}))$,[1] and thus we can write log likelihood of $p_{(\Psi,\boldsymbol{\theta})}(\boldsymbol{x})$ from Eq.(3.17) as

$$\begin{aligned}\langle \boldsymbol{x}, \boldsymbol{\theta}\rangle - \Psi(\boldsymbol{\theta}) &= (\langle \boldsymbol{\mu}, \boldsymbol{\theta}\rangle - \Psi(\boldsymbol{\theta})) + \langle \boldsymbol{x} - \boldsymbol{\mu}, \theta\rangle \\ &= \Phi(\boldsymbol{\mu}) + \langle \boldsymbol{x} - \boldsymbol{\mu}, \nabla\Phi(\boldsymbol{\mu})\rangle \end{aligned} \tag{3.30}$$

Therefore for any $\boldsymbol{x} \in dom(\Phi)$ and $\boldsymbol{\mu} \in int(dom(\Phi))$ we can write:

$$\langle \boldsymbol{x}, \boldsymbol{\theta}\rangle - \Psi(\boldsymbol{\theta}) - \Phi(\boldsymbol{x}) = -d_\Phi(\boldsymbol{x}, \boldsymbol{\mu}). \tag{3.31}$$

For further technical details we refer reader to [Banerjee et al., 2004].

3.0.1.2 Dually Coupled Exponential Families

There is an intriguing property of the Bregman divergences stating that the Bregman divergence d_Ψ equals to the Bregman divergence d_Φ on the dual space defined by gradient mapping $\nabla\Phi$.

Thus, by means of (3.25), the "dual" Bregman divergence $d_\Psi(\nabla\Phi(\boldsymbol{y}), \nabla\Phi(f(\boldsymbol{x}|\boldsymbol{w})))$ defines the coupled Exponential family $p_{\Phi, f(\boldsymbol{x}|\boldsymbol{w})}(\boldsymbol{y})$ over the dual space.

In lieu of this duality we have the loss $\ell(\boldsymbol{z}, \boldsymbol{y}) = d_\Phi(\boldsymbol{z}, \boldsymbol{y}) = d_\Psi(\nabla\Phi(\boldsymbol{y}), \nabla\Phi(\boldsymbol{z}))$ represents two coupled exponential families ...

1. *(primal)* ... defined by a cumulant function Ψ and neural network $f(\boldsymbol{x}|\boldsymbol{w})$ being parametrized sufficient statistics. Mean value parameters are given by targets $\boldsymbol{y}$, given by Eq. (3.25) applied on loss $\ell(f(\boldsymbol{x}|\boldsymbol{w}), \boldsymbol{y}) = d_\Phi(f(\boldsymbol{x}|\boldsymbol{w}), \boldsymbol{y})$.

2. *(dual)* ... defined by a cumulant function Φ and sufficient statistics $\nabla\Phi(\boldsymbol{y})$ derived from $\ell(f(\boldsymbol{x}|\boldsymbol{w}), \boldsymbol{y}) = d_\Psi(\nabla\Phi(\boldsymbol{y}), \nabla\Phi(f(\boldsymbol{x}|\boldsymbol{w})))$. The neural network $f(\boldsymbol{x}|\boldsymbol{w})$ maps inputs $\boldsymbol{x}$ into the natural parameter space of this family.

In the primary view the loss is mapped to an exponential family with sufficient statistics $f(\boldsymbol{x}, \boldsymbol{w})$ given by neural network f and mean parameters are given by $\boldsymbol{y}$ (thus maximum likelihood estimate of natural parameter is given by $\hat{\theta} = \nabla\Phi(\boldsymbol{y})$ if $\boldsymbol{y} \in \mathcal{M}$).

[1] follows from definition of $\Psi(\boldsymbol{\theta}) := \sup_{\boldsymbol{\mu}\in dom(\Phi)} \langle \boldsymbol{\theta}, \boldsymbol{\mu}\rangle - \Phi(\boldsymbol{\mu}))$ and because the supremum is attained at $\boldsymbol{\theta} = \nabla\Phi(\boldsymbol{\mu})$. We skip technicalities in definitions for a supremum to be attainable for the sake of space and brevity, see [Wainwright and Jordan, 2008] for details.

3.0.2 Self-Regularized Bregman Objective (SeReBrO)

Recall that in the LiMoD formulation of a neural network back-prop training from Chapter 1 every data point defines one probability distribution from the Exponential Family, defined by the Bregman loss with strictly convex function Φ, sufficient statistics and base measure.

By duality and bijection between regular exponential families and regular Bregman divergences, making use of the primal view 1, we can write *the loss* for any given data pair $(\boldsymbol{x}, \boldsymbol{y})$ *on training data sample* s as

$$d_\Phi(f(\boldsymbol{x}_s; w), \boldsymbol{y}_s) \overset{(3.25)}{=} -\log q_{(\boldsymbol{\theta}=\nabla\Phi(\boldsymbol{y}_s),\Psi,\boldsymbol{w})}(\boldsymbol{x}_s) + \Phi(f(\boldsymbol{x}_s, \boldsymbol{w})) \tag{3.32}$$

where exponential family distribution q is

$$q_{(\boldsymbol{\theta},\Psi,\boldsymbol{w})}(\boldsymbol{x}) := \exp\left(\langle \boldsymbol{\theta}, f(\boldsymbol{x}, \boldsymbol{w})\rangle - \Psi(\boldsymbol{\theta})\right) h_x(\boldsymbol{x}) \tag{3.33}$$

will be reconstructed and elaborated in detail shortly in (3.42) in the next section. Note that both elements functionally depend on weights and are optimized by SGD.

In what follows we construct exponential family from given strictly convex function Ψ (and thus its convex dual Φ) allowing us to interpret both terms and arriving at the Theorem 3.15, the main result of this chapter, at the end of this section.

3.0.2.1 (Re)construction of the objective from Ψ

This section gives meaning to both terms of 3.32 by constructing both elements of the equation from given function Ψ (or Φ).

By Theorem 3.11 from [Devinatz, 1955], rewritten for brevity in Preliminaries, for a given strictly convex Φ there exists a unique (almost everywhere) bounded positive measure, denoted ν_z, on (latent) space, denoted $\mathcal{Z} \subset \mathbb{R}^d$, such that,

$$e^{\Psi(\boldsymbol{\theta})} = \int_{\mathbb{R}^d} e^{\langle \boldsymbol{z}, \boldsymbol{\theta}\rangle} d\nu_z(\boldsymbol{z}). \tag{3.34}$$

By the other perspective, theorem restates, that there exist (bounded positive) measure ν_z such that $e^{\Psi(\boldsymbol{\theta})}$ can be expressed as the Laplace transform of its Radon-Nikodym derivative $\frac{d\nu_z}{d\lambda}$, where λ is either Lebesgue (or analogously

counting measure for discrete case) and assuming ν being absolutely continuous w.r.t. λ

Note that as a consequence $d\nu_z(\boldsymbol{z})$ is independent of θ by construction.

Then, as shown in [Banerjee et al., 2005], divergence derived from convex dual Φ gives rise to exponential family distribution parameterized by θ, as defined in (3.25), with density

$$g_{(\boldsymbol{\theta},\Psi)}(\boldsymbol{z}) := \exp\left(\langle \boldsymbol{\theta}, \boldsymbol{z}\rangle - \Psi(\boldsymbol{\theta})\right) d\nu_z(\boldsymbol{z}) = \exp\left(\langle \boldsymbol{\theta}, \boldsymbol{z}\rangle - \Psi(\boldsymbol{\theta})\right) h_z(\boldsymbol{z}) d\boldsymbol{z} \quad (3.35)$$

where h_z is Radon-Nikodym derivative $\frac{d\nu_z(\boldsymbol{z})}{d\lambda(\boldsymbol{z})}$ assuming ν_z is absolutely continuos w.r.t. to Lebesgue (or counting) measure $\lambda(\boldsymbol{z})$. For brevity we retract to usual notation for Lebesgue measure $\lambda(\boldsymbol{z})$ as $d\boldsymbol{z}$ with an abuse of notation for counting measure as well to cover for the case of discrete distributions.

Following lemma's derive well known identities of cumulant function of exponential family. They show however that these properties follow from strict convexity of Φ and duality between Φ and Ψ.

Lemma 3.11. *Under previous notation following holds*

$$\nabla\Psi(\theta) = E_{g_{(\boldsymbol{\theta},\Psi)}}[\boldsymbol{z}] \quad (3.36)$$

Proof. From (3.34) we have $\Psi(\boldsymbol{\theta}) = \log \int_{\mathbb{R}^d} e^{\langle \boldsymbol{z},\boldsymbol{\theta}\rangle} d\nu_z(\boldsymbol{z})$. Because ν_z is independent of θ,by taking derivative, i.e., $\nabla\Psi$ we obtain statement of the lemma as follows

$$\nabla\Psi(\theta) = \frac{\int_{\mathbb{R}^d} \boldsymbol{z} e^{\langle \boldsymbol{z},\boldsymbol{\theta}\rangle} d\nu_z(\boldsymbol{z})}{\underbrace{\int_{\mathbb{R}^d} e^{\langle \boldsymbol{z},\boldsymbol{\theta}\rangle} d\nu_z(\boldsymbol{z})}_{\overset{(3.34)}{=}\Psi(\theta)}} = \int_{\mathbb{R}^d} \boldsymbol{z} e^{\langle \boldsymbol{z},\boldsymbol{\theta}\rangle - \Psi(\theta)} d\nu_z(\boldsymbol{z}) = E_{g_{(\boldsymbol{\theta},\Psi)}}[\boldsymbol{z}].$$

□

Lemma 3.12. *Under previous notation and assuming that ν_z is normalized the following holds*

$$\Phi(\nabla\Psi(\theta)) = D_{KL}(g_{(\boldsymbol{\theta},\Psi)} || d\nu_z) \quad (3.37)$$

Proof. From convex conjugacy of Ψ and Φ, i.e., $\Phi(\mu) = \sup_{\theta} \langle \mu, \theta\rangle - \Psi(\theta)$ and by taking derivative and setting it to zero to find supremum is attained at

$\mu = \nabla\Psi(\theta)$, we have $\Phi(\nabla\Psi(\theta)) = \langle\nabla\Psi(\theta), \theta\rangle - \Psi(\theta)$ and by previous lemma (3.36) this equals to

$$\Phi(\nabla\Psi(\theta)) = \langle E_{g_{(\boldsymbol{\theta},\Psi)}}[\boldsymbol{z}], \theta\rangle - \Psi(\theta) = \int_{\mathbb{R}^d} (\langle \boldsymbol{z}, \boldsymbol{\theta}\rangle - \Psi(\theta)) e^{\langle \boldsymbol{z},\boldsymbol{\theta}\rangle - \Psi(\theta)} d\nu_z(\boldsymbol{z})$$

$$= \int_{\mathcal{Z}} \log\left(\frac{g_{(\boldsymbol{\theta},\Psi)}}{d\nu_z}\right) g_{(\boldsymbol{\theta},\Psi)}(\boldsymbol{z}) \overset{def.}{=} D_{KL}(g_{(\boldsymbol{\theta},\Psi)} || d\nu_z) \tag{3.38}$$

where $D_{KL}(g||d\nu_z)$ denotes Kullback-Leiber divergence as defined in [Amari, 2016, Bishop, 2006]. □

Remark. *Note that one can assume ν_z to be normalized without loss of generality as adding its normalization constant, that is independent on $\boldsymbol{w}$, to the objective does not change the optimization trajectory and position of the optima. Thus for the rest of this chapter consider ν_z to be probability measure, i.e., normalized measure, on $\mathcal{Z}$.*

Next we extend $g(\boldsymbol{z})$, i.e., the distribution defined by Φ over the latent space $\mathcal{Z}$, to the distribution $q(\boldsymbol{x})$ over inputs $\mathcal{X}$.

Given set $\mathcal{X}$ and σ algebra $\mathcal{S}$ on $\mathcal{X}$ with (probability) measure $P(\boldsymbol{x})$, i.e., measurable space $(\mathcal{X}, \mathcal{S}, P(\boldsymbol{x}))$, we can define probability distribution over $\mathcal{X}$ by considering sufficient statistics $f(\boldsymbol{x}, \boldsymbol{w})$ with $\boldsymbol{x} \in \mathcal{X}$ under mild condition on f.

The only condition required is existence of some positive measure $\nu_x(x)$ such that following integral is finite

$$\int_{\mathcal{X}} e^{\langle f(\boldsymbol{x},\boldsymbol{w}),\boldsymbol{\theta}\rangle} d\nu_x(\boldsymbol{x}) < \infty \tag{3.39}$$

as then the resulting density over $\mathcal{X}$ can be normalized and thus well defined.

We prove by construction that such ν_x exists under general yet sufficient conditions to meet this requirement in what follows.

For $\mathcal{X} \subset \mathbb{R}^m$, such as in machine and deep learning, a measurability of f is ensured for instance when f is almost everywhere continuous on $\mathbb{R}^m$ or, in discrete case, for finite f.

Let's assume $f : \mathcal{X} \to \mathcal{Z}$ is surjective and P-measurable and P is absolutely continuous w.r.t. Lebesgue or counting measure λ on $\mathcal{X}$. Let's define ν_x as follows

$$\nu_x(E) := \nu_z(f(E)) \tag{3.40}$$

To check this is well-defined, we have to show that $f(E)$ is a Borel set in $\mathcal{Z}$ for any Borel set E in $\mathcal{X}$. This uses the surjectivity of f and the fact that Borel algebra is the smallest σ-algebra containing all closed sets.

It is also straightforward to see from surjectivity of f giving $f(f^{-1}(B)) = B$ that

$$\nu_x(f^{-1}(B)) = \nu_z(B) \text{ for } B \in \mathcal{Z} \tag{3.41}$$

where $f^{-1}(E) := \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x} \in E \in \mathcal{Z})\}$, the so-called pre-image (and NOT an inverse of f).

Lemma 3.13. *That is ν_z is the push-forward measure of ν_x denoted ν_{x_*}, i.e., $\nu_z = \nu_{x_*}$*

Proof. From definition. □

Construction above has shown the well known fact that strictly convex Φ and continuous and finite sufficient statistics f (that is given parameters $\boldsymbol{w}$ in case f represents neural network) give rise to exponential family over $(\mathcal{X}, \mathcal{S}, P)$ with following density

$$q(\boldsymbol{x}; \boldsymbol{\theta}, \Psi) := \exp\left(\langle \boldsymbol{\theta}, f(\boldsymbol{x}, \boldsymbol{w})\rangle - \Psi(\boldsymbol{\theta})\right) d\nu_x(\boldsymbol{x}) \tag{3.42}$$

$$= \exp\left(\langle \boldsymbol{\theta}, f(\boldsymbol{x}, \boldsymbol{w})\rangle - \Psi(\boldsymbol{\theta})\right) h_x(\boldsymbol{x}) d\boldsymbol{x} \tag{3.43}$$

where h_x is Radon-Nikodym derivative $\frac{d\nu_x(\boldsymbol{x})}{d(\boldsymbol{x})}$ assuming ν_x is absolutely continuous w.r.t. to Lebesgue (or counting) measure $d\boldsymbol{x}$.

Note that while the measure $d\nu_z$ does not depend on statistics f the $d\nu_x$ and thus also $h_x(\boldsymbol{x})$ does.

Lemma 3.14. *(change of measures) Let $(\mathcal{X}, \mathcal{S}, \nu_x)$ and $(\mathcal{Z}, \mathcal{T}, \nu_z)$ denote two measurable spaces, with ν_x and ν_z defined above by (3.40) and (3.34) respectively.*

Then for ν_x-measurable $f : \mathcal{X} \to \mathcal{Z}$ the following holds

$$\int_{f(E)} g(\boldsymbol{z}; \boldsymbol{\theta}, \Psi) = \int_E q(\boldsymbol{x}; \boldsymbol{\theta}, \Psi), \qquad \forall E \in \mathcal{S} \tag{3.44}$$

Proof. By construction of ν_x in (3.40) we have $\nu_z = f_* \nu_x$ as shown earlier, see lemma 3.13. Because $\nu_z = f(\nu_x)$, i.e., ν_z is a pushforward of ν_x, the assumptions

of the change of measure theorem 3.10 are met. Its direct application on the definition of $g(\boldsymbol{z})$, (3.35) gives statement of the lemma as follows

$$\int\limits_{f(E)} \exp\left(\langle \boldsymbol{\theta}, \boldsymbol{z}\rangle - \Psi(\boldsymbol{\theta})\right) d\nu_z(\boldsymbol{z}) = \int\limits_{E} \exp\left(\langle \boldsymbol{\theta}, f(\boldsymbol{x})\rangle - \Psi(\boldsymbol{\theta})\right) d\nu_x(\boldsymbol{x}). \tag{3.45}$$

□

Embraced by previous lemmas we are ready to state the main theorem of this chapter.

Theorem 3.15. *(Self Regularized Bregman Objective (SeReBrO)) Under notation of this chapter and for $f : \mathcal{X} \to \mathcal{Z}$ continuous optimization of the loss $\ell(f(\boldsymbol{x}_s; \boldsymbol{w}), \boldsymbol{y}_s) = d_\Phi(f(\boldsymbol{x}_s; \boldsymbol{w}), \boldsymbol{y}_s)$ with respect to weights $\boldsymbol{w}$ is equivalent to optimizing objective*

$$-\log q_{\nabla\Phi(\boldsymbol{y}_s),\Psi}(\boldsymbol{x}_s) + D_{KL}(g_{(\nabla\Phi(f(\boldsymbol{x}_s,\boldsymbol{w}),\Psi)} || d\nu_z) \tag{3.46}$$

w.r.t. $\boldsymbol{w}$ and where g and q are defined in (3.35) and (3.42) respectively and both are functionally dependent on weights $\boldsymbol{w}$.

Proof. For any given data pair $(\boldsymbol{x}, \boldsymbol{y})$ and by bijection between regular exponential families and Bregman divergence (3.25) we can write loss on the training data sample s as follows

$$\begin{aligned} d_\Phi(f(\boldsymbol{x}_s; w), \boldsymbol{y}) \overset{(3.25)}{=} & \underbrace{-\left(\langle \boldsymbol{\theta}, f(\boldsymbol{x}_s; w)\rangle - \Psi(\boldsymbol{\theta})\right) dP_0(f(\boldsymbol{x}_s; w))}_{A} \\ & + \underbrace{\Phi(f(\boldsymbol{x}_s, \boldsymbol{w}))}_{B} + \text{const. w.r.t. } \boldsymbol{w} \end{aligned} \tag{3.47}$$

Part A: Note that $dP_0(f(\boldsymbol{x}_s; w))$ from [Banerjee et al., 2005] and (3.25) corresponds to $\nu_z(f(\boldsymbol{x}_s; w))$ of this section. Than part A above equals $A = -\log g(\boldsymbol{z}; \boldsymbol{\theta}, \Psi)$ evaluated at $\boldsymbol{z} = f(\boldsymbol{x}_s; w)$.

Next we proceed by the "change of measures" lemma 3.14 applied on set $\mathcal{X}$ containing data pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ with i indexing data points and $E = (\boldsymbol{x}_s, \boldsymbol{y}_s)$, which is particular training data point with index s given by assumption of the theorem. Note $\mathcal{X}$ is a joint input-target set not only inputs as would notation falsely suggest. Lemma 3.14 gives

$$A = -\log q(\boldsymbol{x}; \boldsymbol{\theta}, \Psi)$$

which is the first part of the statement of the theorem.

Part B: Because Φ is strictly convex $\nabla\Psi$ from natural to mean parameters is a one-to-one, i.e., invertible, mapping with inverse $\nabla\Phi$,see [Wainwright and Jordan, 2008], and thus $\nabla\Psi(\theta) = f(\boldsymbol{x}_s; w)$ is equivalent to $\theta = \nabla\Phi(f(\boldsymbol{x}_s; w))$. Now the second part of the statement to be proven follows directly from the lemma 3.12 applied on $\theta = \nabla\Phi(f(\boldsymbol{x}_s; w))$. □

Corollary 3.16. *For batch $\mathcal{B}$ randomly sampled from the (latent) data distribution, denoted P_{data}, and under notation of Theorem 3.15 the optimization of the loss $\sum_{s\in\mathcal{B}} \ell(f(\boldsymbol{x}_s; \boldsymbol{w}), \boldsymbol{y}_s) = \sum_{s\in\mathcal{B}} d_\Phi(f(\boldsymbol{x}_s; \boldsymbol{w}), \boldsymbol{y}_s)$ with respect to weights $\boldsymbol{w}$ is equivalent to optimizing objective*

$$D_{KL}(P_{data}(\mathcal{B})||\prod_{s\in\mathcal{B}} q_{\nabla\Phi(\boldsymbol{y}_s),\Psi}(\boldsymbol{x}_s)) + \sum_{s\in\mathcal{B}} D_{KL}(g_{(\nabla\Phi(f(\boldsymbol{x}_s,\boldsymbol{w}),\Psi)}||d\nu_z). \quad (3.48)$$

Proof. Straightforward from independence of data samples by assumption of the corollary (a standard practice in deep learning tasks). □

Remark. *Theorem 3.15 and its corollary casts loss as optimizing two opposite Kullback-Leibler divergences, where the first corresponds to maximizing the likelihood while the second of opposite and reduces the divergence.*

Remark. *(Not all SGD are the same) Importantly it does not claim that every gradient descent optimizes these two divergences. If gradient descent is performed over parameter space only as in classic parameter estimation problem of one probability model, e.g. linear or logistic regression with Gaussian errors, etc. then only the likelihood is maximized as the second term is constant w.r.t. weights.*

Remark. *Link to the concept of Entropy. The entropy of some probability distribution p can be seen as Kullback-Leibler divergence between p and uniform distribution (straight-forward from the definition of $D_{KL}(p||Unif)$[2]), see [Giffin, 2008].*

3.0.3 SeReBrO via Cumulants Matching

While the previous section presents the decomposition of the Bregman divergence loss in general, this section analyzes the effect of both, stochastic and full, gradient descent optimization of this objective. We argue that in the case of mini-batch training inherent noise in the gradient keeps the variance of gradients larger compared to the full gradient even in a regime of small gradient norms (plateau

[2]Uniform distribution to be well defined requires finite domain. Otherwise one can take the limit case of uniform distributions over finite domains, in a similar way to the use of the improper priors in Bayesian methods

or local/global optima). This imposes the (soft) constraint on the variance of the model over targets, preventing it from collapsing at low loss levels and thus preventing the net f from over-fitting.

3.0.3.1 Minimizing $D_{KL}(g||d\nu_z)$, Matching Cumulants

As was pointed out in the previous chapter the crucial different between maximum likelihood estimation and optimizing neural networks is the second element $+\sum_{s\in\mathcal{B}} D_{KL}(g_{(\nabla\Phi(f(\boldsymbol{x}_s,\boldsymbol{w}),\Psi)}||d\nu_z)$ in Corollary 3.16 corresponding to minimizing the batch average KL divergence from g to ν as stated by Lemma 3.12.

To shed more light on what's going on let's have a look at ν_z first. By its definition it is a unique positive probability[3] measure defined in (3.34) as

$$e^{\Psi(\boldsymbol{\theta})}e^{-\Psi(\mathbf{0})} = e^{-\Psi(\mathbf{0})}\int_{\mathbb{R}^d} e^{\langle \boldsymbol{z},\boldsymbol{\theta}\rangle} d\nu_z(\boldsymbol{z}) \tag{3.49}$$

It follows by taking the derivatives of (3.49) at $\theta = 0$ that Ψ is a cumulant generating function of ν_z. For instance the first cumulant, i.e., mean of sufficient statistics, is obtained as

$$\nabla\Psi(\theta)e^{-\Psi(\mathbf{0})}e^{\Psi(\boldsymbol{\theta})}|_{\theta=\mathbf{0}} = \nabla\Psi(\theta)|_{\theta=\mathbf{0}} = \int_{\mathbb{R}^d} \boldsymbol{z}e^{\langle \boldsymbol{z},\mathbf{0}\rangle} d\nu_z(\boldsymbol{z}) = E_{\nu_z}[\boldsymbol{z}] \tag{3.50}$$

and so on with higher cumulants.

Thus minimizing the KL divergence $D_{KL}(g_{(\nabla\Phi(f(\boldsymbol{x}_s,\boldsymbol{w}),\Psi)}||d\nu_z)$ from g to ν_z from the Theorem 3.15 and its corollaries corresponds to matching the cumulants (or moments, both define a bounded! probability distribution uniquely [Billingsley, 1995]) of these two distributions [Bishop, 2006]. Minimizing KL divergence between two exponential family models by matching moments is well known and used in variational methods [Bishop, 2006] and Expectation Propagation [Minka, 2001, Bishop, 2006]. The method above is similar in its nature but extends (3.50) the moment matching between g from Exponential family and ν_x that is inherently ambient given by Theorem 3.11.

Moreover, while in methods of approximate inference the moments are analytically known and matched, the backprop training optimizes the weights of the model

[3] It is explicitly normalized by a constant $e^{\Psi(\mathbf{0})}$) in here. Thus a probability measure as was assumed throughout the chapter, abusing notation and denoting it also ν_z, is obtained explicitly.

f to meet both objectives in Theorem 3.15 including the likelihood besides the "cumulant" term. Thus the cumulants are matched only approximately acting as a regularizer. One could compare the cumulants matching by minimizing the KL divergence $D_{KL}(g_{(\nabla\Phi(f(\boldsymbol{x}_s,\boldsymbol{w}),\Psi)}||d\nu_z)$ to a soft constraint similar to prior belief on moments in Bayesian methods [Gelman et al., 2013].

Note that cumulants of $g_{(\nabla\Phi(f(\boldsymbol{x}_s,\boldsymbol{w}),\Psi)}$ are given by derivatives of Ψ evaluated at $\theta = f(\boldsymbol{x}_s, \boldsymbol{w})$ given by outputs of f.

Matching the First Cumulants making use of (3.50) we have that backprop minimization of loss leads to

$$\nabla\Psi(\theta)|_{\theta=\mathbf{0}} = \int_{\mathbb{R}^d} \boldsymbol{z} e^{\langle \boldsymbol{z},\mathbf{0}\rangle} d\nu_z(\boldsymbol{z}) = E_{\nu_z}[\boldsymbol{z}] \overset{backprop}{\approx} \nabla\Psi(f(\boldsymbol{x}_s,\boldsymbol{w}))\ \forall \boldsymbol{x}_s. \quad (3.51)$$

Since Ψ is strictly convex the exact match is only achieved by $f \equiv 0$. Because f includes bias term in general such a setting works for arbitrary values of targets. Note that after a standard random weights initialization around zero [Glorot and Bengio, 2010, Goodfellow et al., 2016] $f \equiv 0$ is true in expectation w.r.t. initializing distribution. While the second "KL" term in Theorem 3.15 is optimal (in expectation), the "likelihood" term is typically randomly "bad" at initialization.

During the course of backprop training, the model strives to strike the balance between maximizing the "likelihood" term and minimizing the "KL" term acting as a regularizer. Intuitively the Equation (3.51) keeps the outputs of f close to $\theta = 0$, i.e., the origin of the output manifold[4]. That means the effect of matching the first cumulant (3.51) is to keep weight parameters not "far away" from the bias parameter of *the output layer* and it can be intuitively seen as a sort of translated "weight decay" implicit regularizer of the output layer with an (intuitively) diminishing effect on the layers towards the input [Bishop, 1995, Goodfellow et al., 2016].

As it will be shown shortly, the LiMoD, model "per sample" perspective, will be used to reformulate SGD as a back-prop with adaptive step dependent learning rate τ_t. In turn, this learning rate will have a step-dependent regularization effect, evolving over the course of the training and depending on hyperparameters.

Before moving on let's note that standardization of training data (i.e., centralizing and dividing by sample variance), which is standard practice nowadays leads

[4] note again, that f includes bias terms in general, so this does not mean output f is kept zero, but rather "target data mean", i.e. what the bias parameter of the last layer converges to.

to balanced first cumulants matching regularization viewed by a prism of this subsection. On the contrary unnormalized training, data lead to divergence "KL" term being large along with certain directions possibly leading to a model biased towards certain features.

Matching the Second Cumulants By repeating (3.50) we obtain the second cumulant matching condition as follows

$$\nabla^2\Psi(0) \overset{backprop}{\approx} \nabla^2\Psi(f(\boldsymbol{x}_s, \boldsymbol{w})). \tag{3.52}$$

Since KL divergence is asymmetric minimizing $KL(g||\nu)$ and $KL(\nu||g)$ leads to different solution in general, see [Bishop, 2006], Chapter 10. In particular minimizing KL divergence from g to ν_z leads to g that avoids areas of domain where ν_z is negligible as could be seen from definition of KL and is also visually shown elaborately in [Bishop, 2006], Chapter 10. In other words it favors

$$\left\|\nabla^2\Psi(0)\right\| \leq \left\|\nabla^2\Psi(f(\boldsymbol{x}_s, \boldsymbol{w}))\right\| \tag{3.53}$$

for some matrix norm (due to norms equivalence on finite spaces, [Bhatia, 1997]). For instance the matrix norm induced by a vector norm $\|\cdot\|$ as $\|\Sigma\| = \sup\{\|\Sigma x\| : \|x\| \leq 1\}$.

Recall that by convex duality we have $\nabla^2\Psi = (\nabla^2\Phi)^{-1}$ and that Φ is a cumulant function for the "dual" view probability model over targets $\tilde{\boldsymbol{y}}$ that are sufficient statistics as well and parametrized by outputs of f, see (3.0.1.2).

That means minimizing $D_{KL}(g_{(\nabla\Phi(f(\boldsymbol{x}_s,\boldsymbol{w}),\Psi)}||d\nu_z)$ leads to a model whose *variance* at predicted parameters $f(\boldsymbol{x}_s, \boldsymbol{w})$ is approximately $\nabla^2\Phi(0)$, Eq. (3.54), and biased to be bounded from the top by the second cumulant of ν_z in (3.54) as follows

$$\nabla^2\Phi(0) \approx \nabla^2\Phi(f(\boldsymbol{x}_s, \boldsymbol{w})) \tag{3.54}$$

$$\left\|\nabla^2\Phi(0)\right\| \geq \left\|\nabla^2\Phi(f(\boldsymbol{x}_s, \boldsymbol{w}))\right\|. \tag{3.55}$$

3.0.3.2 Minimizing $D_{KL}(g||d\nu_z)$ Shapes the Net f

Eq. 3.54 means that the (dual view) model derived from

$$\ell(f(\boldsymbol{x}|\boldsymbol{w}), \boldsymbol{y}) = d_\Psi(\nabla\Phi(\boldsymbol{y}), \nabla\Phi(f(\boldsymbol{x}|\boldsymbol{w})))$$

over targets $\tilde{\boldsymbol{y}} := \nabla\Phi(\boldsymbol{y})$, defined in 3.0.1.2, has a given variance by $\nabla^2\Phi(0)$, approximately (in a sense of the divergence $D_{KL}(g||d\nu_z)$) from (3.16)). For

instance in the case of "squared loss", given by $\Phi(\boldsymbol{\mu}) = \frac{1}{2}\langle \boldsymbol{\mu}, \boldsymbol{\mu} \rangle$, it'd be an identity matrix I. This can be seen as a prior belief (in a form of degenerate one point delta function probability distribution) imposed on variance parameter of the model in the Bayesian statistics sense [Gelman et al., 2013].

If this imposed variance $\nabla^2\Phi(0)$ is larger than the variance of target *data distribution* over $\tilde{\boldsymbol{y}}$ than $f(\boldsymbol{x}_s, \boldsymbol{w})$ is given a slack by $\nabla^2\Phi(0)$ to fit $\boldsymbol{y}$, because gradients of likelihood would tend to vanish when $f(\boldsymbol{x}_s, \boldsymbol{w})$ is in a high likelihood region of $\boldsymbol{y}$, determined by variance $\nabla^2\Phi(0)$. In such setting, the $f(\boldsymbol{x}_s, \boldsymbol{w})$ is not updated to fit $\boldsymbol{y}_s$ exactly, i.e., to overfit, as opposed to the case of low(er) variance of $\nabla^2\Phi(0)$. This is well known behaviour from regression Gaussian models ([Rasmussen, 2003]) with priors on (or given) variance of the model for instance. Next section shows how this variance changes over training and how it depends on a batch size and learning rate.

More over this is intuitively in line with matching/minimizing the first moment as well as it is expected that fitting $\boldsymbol{y}_s$ exactly produces larger distance of the outputs $f(\boldsymbol{x}_s, \boldsymbol{w})$ from the origin than only getting to its vicinity (going from the origin as network is initialized close to it, $f(\boldsymbol{x}_s, \boldsymbol{w}) = 0$ for all $\boldsymbol{x}_s$).

Alternative interpretation rises from the use 1^{st} order Taylor approximation to f as a function of $\boldsymbol{x}_s$ that gives a variance approximation $\nabla_x f var(X) \nabla_x f$ [5] while assuming finite data variance $var(X) < \infty$. Then bounding this variance leads to penalizing the norm of the gradient $\|\nabla_x f\|_2$ and thus also $\|\nabla_x \ell\|_2$ that is argued to be connected to all sorts of norm regularizes, e.g., path norm [Yoshida and Miyato, 2017, Neyshabur et al., 2015, Kawaguchi et al., 2017], $l_2$norm [Goodfellow et al., 2016], depending on activation functions used in the model.

More broadly. Under the LiMoD perspective, where individual sample dependent models indexed by s belong to the Ψ-imposed Exponential family, the joint model over pairs $(\boldsymbol{x}_s, \boldsymbol{y}_s)$ belongs to Exponential family as well, [Wainwright and Jordan, 2008, Amari, 2016]. Then imposing the priors on the model via Φ and its moments while parameterizing the mean function $\nabla\Phi(f(\boldsymbol{x}|\boldsymbol{w})))$ can be seen as a generalization of Gaussian models, [Rasmussen, 2003]. In particular, and under the model hypothesis, any subset of data points defines a joint probability distribution from (some) Exponential family meeting constraints given by Φ and with parametric (by weights) mean value function. Details are left for the future work.

[5] also by the delta method in case of CLT was legit to apply

3.0.3.3 LiMoD Back-prop Formulation

For reasons to be revealed shortly, let's lump together learning rate multiplied by a l_2 norm of a gradient over batch size, [Goodfellow et al., 2016] into one step dependent hyper parameter denoted $\tau_t := \frac{\eta_t \|\nabla \ell_t\|_2}{|B_t|}$.

Then back-prop step t proceeds with a unit gradient $\overrightarrow{g}_t$ multiplied by a "learning rate" τ_t.

Since Bregman divergence is linear in the first parameter, [Banerjee et al., 2005], and assuming $\tau_t > 0$, one can absorb scalar τ_t into a Φ and consider the $\Phi_t = \tau_t \Phi$, a "new" strictly convex and step dependent function that generates a corresponding Bregman divergence (3.31) to be minimized in every step.

To summarize it we have defined the following step dependent entities recasting the back-prop training into step dependent optimization, i.e., the objective at every step is different in general and given by Φ_t

Definition 3.17. *(LiMoD back-prop training formulation)*

$$\begin{aligned} \tau_t &:= \frac{\eta_t \|\nabla \ell_t\|_2}{|B_t|} && \text{(LiMoD learning rate)} \\ \overrightarrow{g}_t &:= \frac{\nabla \ell_t}{\|\nabla \ell_t\|_2} && \text{(LiMoD gradient)} \\ \Phi_t &:= \tau_t \Phi. && \text{(LiMoD objective)} \end{aligned}$$

Relating it to (3.55) one sees that the smaller learning rate or gradient norm over batch size the smaller variance of the model is allowed by (3.55) and thus f has to fit targets closely. This may happen by learning rate decay, a standard practice in deep learning, see [Goodfellow et al., 2016], during and at the end of the training allowing for "overfitting". In other words, we have theoretically recovered highly intuitive and well-known, and observed phenomena. On contrary, the novel results will follow in the next chapter by analyzing the last ingredient of τ_t, a norm of gradient, whose effects are complex, especially in higher dimensions.

To motivate it to consider stochastic training with the constant learning rate η_t and batch size $|B_t|$. Then a decreasing norm of gradient, occurring possibly at (a vicinity of) optimum or plateau, brings τ_t small and thus imposes smaller variance on the targets model (3.54) using $\Phi_t = \tau_t \Phi$ in place of Φ as described

above. In particular, we obtain

$$\tau_t \nabla^2 \Phi(0) \approx \nabla^2 \Phi_t(f(\boldsymbol{x}_s, \boldsymbol{w})) \tag{3.56}$$

$$\tau_t \left\| \nabla^2 \Phi(0) \right\| \geq \left\| \nabla^2 \Phi_t(f(\boldsymbol{x}_s, \boldsymbol{w})) \right\|. \tag{3.57}$$

where $\nabla^2 \Phi_t(f(\boldsymbol{x}_s, \boldsymbol{w}))$ is a variance of the model over targets at update t as described in previous section. One can see that scalar τ_t controls the variance of the model. The most interesting constituting element of τ_t is norm of gradients and will be elaborated next.

Chapter 4

Generalization of Deep Learning Optimizing Bregman Divergences

. . . 'There's more evidence to come yet, please your Majesty,' said the White Rabbit, jumping up in a great hurry; 'this paper has just been picked up.' . . .

King reflecting in court, Alice's Adventures in Wonderland, Charles Ludtwidge Dodgson (Lewis Carroll, 1865)

This chapter builds upon the previous one and shows that stochasticity in training and depth of the network together, have consequences on a norm of noisy gradient that will concentrate **away** from zero as a function of dimension and as opposed to the behavior of the low dimensional models and common intuition. That put together with previous findings of this chapter will allow us to present a statement linking the depth and stochasticity of back-prop training to generalization in deep learning.

4.0.0.1 Going Noisy with Mini-Batch Training

Compared to full batch (gradient descent) a mini-batch (stochastic gradient descent) training could be seen as back-propagation of a full batch gradient with an additive random noise [Li et al., 2020b, Du et al., 2018, Li et al., 2017, Li et al., 2019, Mandt et al., 2017, Goodfellow et al., 2016] and many others.

Formally in what follows we use the construction from [Li et al., 2017] and consider a unit batch size of 1 for simplicity. The stochastic gradient descent (SGD) replaces the full gradient $\nabla \ell$ with a sampled version, serving as an unbiased estimator. In its simplest form, the SGD iteration is written as

$$w_{t+1} \to w_t - \eta_t \nabla \ell_{\gamma_t}(w_t) \tag{4.1}$$

where $t \geq 0$ and γ_t are i.i.d uniform variates defining the subset of training data (random mini-batch) taking values in $\{1, 2, \ldots, |\mathcal{D}|\}$ where $\mathcal{D}$ denotes set of training data. Now we can rewrite (4.1) as

$$w_{t+1} \to w_t - \eta_t \nabla \ell(w_t) + \sqrt{\eta_t} V_t \qquad \text{(stochastic gradient step)}$$

where $V_t = \sqrt{\eta_t}(\nabla \ell(w_t) - \nabla \ell_{\gamma_t}(w_t))$ is a $d-$dimensional random vector. Given w_t, V_t has mean 0 and covariance matrix $\eta \Sigma(w_t)$ with

$$\Sigma(w) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} (\nabla \ell(w) - \nabla \ell_i(w))(\nabla \ell(w) - \nabla \ell_i(w))^T. \tag{4.2}$$

Define gradient errors as

$$\varepsilon_i := \nabla \ell_i(w) - \nabla \ell(w) \tag{4.3}$$

equivalently written as

$$\varepsilon_i(w) = |\mathcal{D}| \nabla \ell(w) - \sum_{j=1, j \neq i}^{|\mathcal{D}|} \nabla \ell_j(w). \tag{4.4}$$

Consider a large enough training data size, e.g., the common deep learning applications with more than hundreds of training data samples, etc., and bounded moments of errors (4.3) $\varepsilon_i, \forall i$. Then by the Central Limit Theorem (CLT), [Gelman et al., 2013], applied on errors (4.4) and on (4.2) one obtains that ε_i are (asymptotically in training data size $|\mathcal{D}|$) Gaussian and sample covariance of errors, Σ, is asymptotically Gaussian estimator of variance of gradient error ε at given w.

The construction above considers a unit batch size. For the larger batch sizes, the construction and Gaussian approximation follow by the same lines taking γ_t to be randomly sampled set of indices of training data and using general CLT as γ_t are now (weakly) dependent.

4.0.1 Going Deep

As opposed to low dimensional case, the norm of random vector in high dimensional spaces behaves a bit unintuitively, i.e., random vector of zero mean and unit variance coordinates in $\mathbb{R}^n$ will concentrate in the distance $\sqrt{n}$ from zero.

Gaussian and Spherical Distributions A prequel for such a statement is the following insightful lemma that will turn useful in higher dimensions. It shows that Gaussian distribution in Cartesian coordinates corresponds to Uniform (direction) and χ^2 (radius) distribution in polar coordinates. Lemma is also an Excercise 3.3.7 in [Vershynin, 2018]. It is presented here for completeness with proof.

Lemma 4.1. *(The Gaussian as a Spherical Distribution) If $X \sim N(\mathbf{0}, \mathbb{I}_n)$, then $X = R \cdot U$, where*

$$R^2 = \|X\|_2^2 \sim \chi^2(n) \qquad \text{and} \tag{4.5}$$

$$U = \frac{X}{\|X\|_2} \sim \mathit{Unif}(S_{n-1}) \qquad \text{(Uniform distribution on a sphere)}$$

and R and U are independent. Further $(n-1)$-sphere of radius $R > 0$ is defined $S_{n-1}(R) = \{x \in \mathbb{R}^n : \|x\|_2 = R\}$ and S_{n-1} denotes the unit radius $(n-1)$-sphere and $\|\cdot\|_2$ is the Euclidean metric induced l_2 norm.

Proof. Suppose $X = \{X_1, \ldots, X_n\}$ is a multivariate random vector with independent and identically distributed standard Gaussian coordinates $X_i \sim N(0, 1)$. Then $X \sim N(\mathbf{0}, \mathbb{I})$ is a multivariate normal distribution with zero mean and identity covariance matrix $\mathbb{I}$ and $R^2 := \|X\|_2^2 = (X_1^2 + \cdots + X_n^2) \sim \chi^2(n)$ follows χ^2 distribution by definition, see [Gelman et al., 2013, Bishop, 2006].

Let O be an orthogonal (square and real) matrix, i.e, $O^T = O^{-1}$. Than XO has same distribution as X. It follows from the fact that X is multivariate Gaussian $N(\mathbf{0}, \mathbb{I})$ then OX is as well multivariate Gaussian (by plugging into density) and then these are equal iff means and variances equal, [Bishop, 2006] for instance. But that is straightforward to see as $E[OX] = OE[X] = 0 = E[X]$ and $E[(OX)^2] = E[(OX)^T(OX)] = E[(X^T O^T(OX)] = E[X^2]$. Because orthogonality of O gives also $\|OX\|_2 = OO^T\|X\|_2^2 = \|X\|_2$ (we just proved it) then we have $\frac{OX}{\|OX\|_2}$ is identically distributed with $\frac{X}{\|X\|_2}$.

We have gotten that $U := \frac{X}{\|X\|_2}$ is invariant under rotations and U lies on the unit sphere. That is (equivalent with) a definition of the uniform distribution on the unit sphere, [Vershynin, 2018], and shows that U uniformly distributed on the n-dimensional unit sphere.

Independence of U and R follows by showing $P(U, R) = P(U|R)P(R) = P(U)P(R)$ (independence), that is equivalent to $P(U|R) = P(U)$ (for case $P(R) > 0$, otherwise trivially). But for given $R = c$ where c is constant we have $P(U|R) = P(X/\|X\|_2 \,|\|X\|_2 = c)$. Since X is multivariate normal with zero mean its density only depends on $\|X\|_2$ and thus is constant for $\|X\|_2 = c$, which means $X|\|X\|_2 = c$ is uniformly distributed over all vectors x such $\|x\|_2 = c$. And thus $P(X/\|X\|_2 \,|\|X\|_2 = c)$ is uniformly distributed over vectors on a unit sphere, i.e. it is $P(U)$. We showed $P(U|R) = P(U)$ for all values of R which concludes the proof. □

In the following the fact that in the (high) dimension n the variance of radius R, defined in previous Lemma, Eq. (4.5), shrinks (concentrate) around $\sqrt{n}$ exponentially. This will allow us to show that zero mean errors ε_i are uniformly distributed on a sphere with radius $\sqrt{n}$.

4.0.2 SGD Gradient Norm in Deep Networks

Deep networks. The word *deep* in the context of this chapter means *high dimensional weights* (altogether with biases), i.e. vector of parameters $\boldsymbol{w} \in \mathcal{W}$ is embedded in *high dimensional Euclidean space* $\mathcal{W} \subseteq \mathbb{R}^{|\mathcal{W}|}$. Thus also gradient of loss $\nabla \ell$ is of a (high) dimension $|\mathcal{W}|$. As outlined before in high dimensional Euclidean spaces a vector with zero centered random coordinates concentrates away from zero opposing the intuition from low dimensions.

Research Question: *Could a small gradient noise, possibly negligible when updating individual weight parameters by mini-batch back-prop, have a significant*

effect by imposing a large model variance (3.56) SeReBrO prior through its norm concentrating away from zero in high dimensions?

It is not obvious even if the norm grows with dimensionality as outlined because the norm of noisy gradient is only upper-bounded by[1] and it is not a sum of norms (of the noisy and the full gradient)[2] in general. Consequently, the noise may cancel out or remain negligibly small during the training or upon convergence living up to intuition from low dimensions.

It will be shown that this cancellation does not happen when stochastic (mini-batch) training is combined with an over-parameterization. The main contribution of this chapter is providing a positive answer to the *Research question* above. In particular, showing that it holds almost surely with the growing dimension of weights.

4.0.2.1 Concentration of the Norm in High Dimensions

Challenging our low dimensional intuition, despite the standard normal distribution $N(\mathbf{0}, \mathbb{I}_n)$ in high dimensions has the density maximal at origin, yet it concentrates in a thin spherical shell around the sphere of radius $\sqrt{N}$ and width $O(1)$. The concentration inequality in the next theorem 4.7 from [Vershynin, 2018] states that the *norm* of the random vector lies far from zero, as also shown in Fig. 4.3 later.

4.0.2.2 Intuition

To begin with, let's build some intuition for a high-dimensional setting. For instance, what can we expect about a randomly picked vector from $N(\mathbf{0}, \mathbb{I}_n)$? Should it be close to the origin (by density and low dimensional intuition) or rather away from zero?

We argue that a random point is more likely to be close to the shell than in the center. Not because the probability in the center is "low" (the opposite is true), but just because there are increasingly more points overall (in a sense of area/volume density integrals) away from the origin, despite being spread thin.

[1] the triangle inequality of the Euclidean norm

[2] unless they are orthogonal, which is sometimes assumed and can be shown to be true around global optimum, i.e., at true parameter [Pawitan, 2004]. However, we'd like to have this property throughout the training and out of optima (does the training ever converge to any optimum in fact? [Goodfellow et al., 2016]).

This can be seen already in two dimensions, see Fig. 4.1. The probability of a randomly picked point inside the (blue) circle with radius 0.5 is $\approx 0.11(sampled)$ and $0.146(theoretical)$, while probability of randomly picked point between the (blue) and (red)[3] circle is $\approx 0.28(sampled)$ and $0.32(theoretical)$ and beyond the (red) line being ≈ 0.61[4] (and $0.534(theoretical)$)

On the other hand, consider the arbitrary circle S of a radius r. Let's compare the probability of picking the point from inside of two such circles with different centers. I.e., the areas are the same, and positions differ. It follows that the probability of picking a point inside of the S closer to the origin is larger because the density increases towards the center where it is highest, as also a heatmap in Fig. 4.1 presents.

As opposed to a density being the infinitesimal limit of the probability per area the above-mentioned concentration is rather an "unnormalized" probability over the area. Because the area grows with radius and exponentially in dimension, i.e. $\sim r^n$, the probability accumulates radially dropping density (Gaussian marginals of n-dimensional Gaussians) over a radially expanding area, exponentially expanding with increasing dimension. It leads to an "equilibrium" away from the origin, the mentioned shell of the hyper-sphere of radius $\sqrt{n}$.

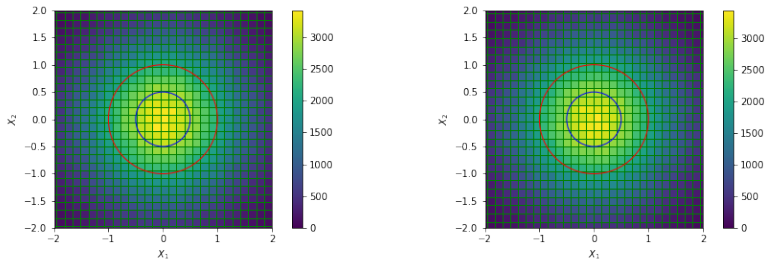


Figure 4.1: *Following contains 2D histogram of* 10^6 *samples from two dimensional* $N(\mathbf{0}, \mathbb{I}_n)$*, i.e, for* $n = 2$*. On the (left) drawn as* $N(0, 1)$ *Cartesian coordinates, on the (right) drawn from spherical normal* $R \cdot U$*,* $U \sim Unif(S_1), R^2 \sim \chi^2(2)$ *in polar coordinates, see Lemma 4.1.*
To motivate intuitively the concept of a "concentration" of a random vector with zero centered standard Gaussian coordinates $N(0, 1)$ *in high dimensions away from origin, see theorem 4.7 from [Vershynin, 2018], note, that a probability of a random vector to lie* outside *of the (red) circle with radius* 1*, i.e. radius of* 1 *standard coordinate deviation (std), is* $\approx 1 - 0.68^2 = 0.537$*, that is larger than* $\frac{1}{2}$*. And thus it is more likely for a random point to be picked from that "outer" area as opposed to 1D case, where probability to lie* within $\pm 1std$ *is larger,* ≈ 0.68*, as well known.*

[3] of radius 1
[4] simulations done using Numpy package, python [vanRossum, 1995].

4.0.2.3 Concentration in High Dimensions

As before, consider a random vector $X = \{X_1, \ldots, X_n\}$ in $\mathbb{R}^n$ with independent and identically distributed standard Gaussian coordinates $X_i \sim N(0,1)$. Then $X \sim N(\mathbf{0}, \mathbb{I}_n)$ is a multivariate normal distribution with zero mean and identity covariance $n \times n$ matrix $\mathbb{I}_n$ and let its norm be denoted $R^2 := \|X\|_2^2 = (X_1^2 + \cdots + X_n^2) \sim \chi^2(n)$[5]. Note that $R \in \mathbb{R}$ is a non-negative real number. Then

$$E[R^2] := E\|X\|_2^2] = \sum_{i=1}^{n} E[X_i^2] = n. \tag{4.6}$$

The length of X can be expected to be around $\sqrt{n}$. The following theorem proven in [Vershynin, 2018] states that it is indeed true with high probability, exponentially vanishing as a function of distance t from a sphere of radius $\sqrt{n}$. We use its lighter version here for brevity.

Theorem 4.2. *(Norm concentration in high dimensions, Informal, see blog of Anmol Goel and [Vershynin, 2018] manuscript) Let X be a random vector in $\mathbb{R}^n$ with independent coordinates. Then*

$$P(|\,\|X\|_2 - \sqrt{n}\,| \geq t) \leq 2\exp\{-ct^2\} \tag{4.7}$$

where $c > 0$ is a constant, and $t \geq 0$ and $\|\cdot\|_2$ is the Euclidean vector norm (l_2-norm).

Proof. The formal statement (Theorem 3.1.1.) for sub-gaussian distributions and detailed proof is to be found in an excellent book on high dimensional probability [Vershynin, 2018], section 3.1. □

To clarify the relation of Gaussian r.v. to spherical distribution as shown in Lemma 4.1 and let us represent X in polar coordinates

$$X = r\alpha \tag{4.8}$$

where r is the length and $\alpha = \frac{X}{\|X\|_2}$ is the direction of X. Then X becomes approximately only a function of direction given n

$$X \approx \sqrt{n}\alpha \sim \mathit{Unif}(\sqrt{n}S_{n-1}) \tag{4.9}$$

with an exponentially vanishing error with increasing dimension n. A relation between Gaussian r.v. in Cartesian and polar coordinates is general and can be

[5] χ^2 distribution by definition

seen demonstrated by sampling from Cartesian and spherical representation in two dimensions in Fig. 4.1 (left) and (right), respectively.

Alternatively put, the concentration inequality (4.7) says that the standard normal distribution in high dimensions is close to the uniform distribution on the sphere of radius $\sqrt{n}$, i.e. $X \sim N(\mathbf{0}, \mathbb{I}_n) \approx Unif(S_{n-1}(\sqrt{n}))$.

Note that this in line with general representation of Gaussian as a spherical distribution as by Lemma 4.1, just with radius becoming increasingly "shrinked" around its expected value.

Fig. 4.2, taken from [Vershynin, 2018], Figure 3.6, presents a high level intuition behind the norm concentration theorem In the follow up the Figure 4.3 reports

Figure 4.2: *Figure 3.6 from the book [Vershynin, 2018], p.53. A Gaussian point cloud in two dimensions (left) and its* intuitive *visualization in high dimensions (right). In high dimensions, the standard normal distribution is very close to the uniform distribution on the sphere of radius* $\sqrt{n}$.

a synthetic data experiment and demonstrates the statement of the Theorem empirically on $30,000$ vectors from $\mathbb{R}^{10^3}$. To sum it up a high-dimensional random normal vector $X \in \mathbb{R}^n$ is (exponentially) tightly concentrated around a sphere of radius $\sqrt{n}$ with norm $\|X\|_2 \approx \sqrt{n}$.

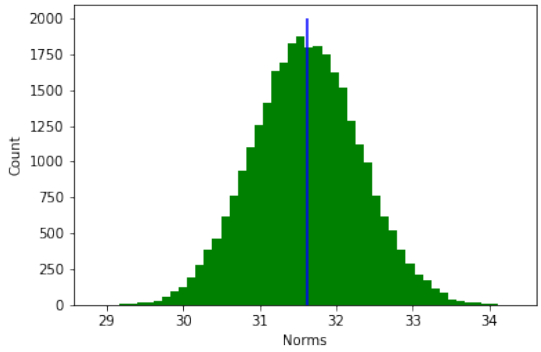


Figure 4.3: *Histogram of Euclidean norms of* 30000 *vectors in* $\mathbb{R}^{10^3}$ *with randomly sampled standard* ***zero mean*** *Gaussian coordinates* $N(0,1)$*. Vectors in this* 10^3*-dimensional space are apparently following the norm concentration Theorem from [Vershynin, 2018], stated above, showing the norm concentrates around* $\sqrt{1000} \approx 31.6$ *(blue line), that is far from zero as opposing the intuition from lower dimensions.*

4.0.2.4 Noisy Gradient is Larger Then Full Batch Gradient in Probability

Recall that considering mini-batch training gradient error in a batch i was defined in (4.3) as

$$\varepsilon_i := \nabla \ell_i(w) - \nabla \ell(w) \tag{4.10}$$

Under a large enough training data size assumption allowing Central Limit Theorem (CLT) to be invoked, ε_i were shown to be (asymptotically in training data size $|\mathcal{D}|$) Gaussian with zero mean and sample covariance of errors, Σ, is asymptotically Gaussian estimator of the variance of gradient error ε at given w.

So a "noisy gradient" can be written as a sum of two following vectors where $\nabla \ell$ is fixed, while ε_i is randomly distributed following $N(\mathbf{0}, \Sigma)$, i.e.

$$\nabla \ell_i = \nabla \ell + \varepsilon_i. \tag{4.11}$$

Let's denote the dimension of the gradient by N. Then by Lemma 4.1 a random vector ε_i with Gaussian coordinates can be represented in polar coordinates as $R \cdot U$, where U is a direction random vector with uniform distribution on the

unit $(N-1)$-sphere and R is a random radius vector such that $R^2 \sim \chi^2(N)$ in polar coordinates.

Next, we prove the following theorem in a $\mathbb{R}^2$ plane for its instructive nature. The corollary for higher dimensions follows.

Theorem 4.3. *Consider vector $\nabla\ell_i = \nabla\ell + \varepsilon \in \mathbb{R}^2$, where $\nabla\ell$ is a given constant vector and $\varepsilon \sim$ Unif$(S_1(r))$, is uniformly distributed random vector on the circle of the given radius $r > 0$ i.e. $\|\varepsilon\|_2 = r$. If $r \leq 2\|\nabla\ell\|_2$ then*

$$P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) = \frac{1}{2} + \frac{\arcsin\left(\frac{\|\varepsilon\|_2}{2\|\nabla\ell\|_2}\right)}{\pi} \tag{4.12}$$

and 1 otherwise, i.e., for $r > 2\|\nabla\ell\|_2$. The $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^2$.

Proof. We opted for a geometrical proof as presented in Fig.4.4 for its visual suggestivity.

First note that for $\|\varepsilon\|_2 = r \geq 2\|\nabla\ell\|_2$ the

$$\begin{aligned}
\|\nabla\ell_i\|_2^2 =& \|\nabla\ell + \varepsilon\|_2^2 = \|\nabla\ell\|_2^2 + \|\varepsilon\|_2^2 + 2\langle\varepsilon, \nabla\ell\rangle = && (4.13)\\
& \|\nabla\ell\|_2^2 + \|\varepsilon\|_2^2 + 2\|\nabla\ell\|_2\|\varepsilon\|_2 \cos\left(\measuredangle_{\varepsilon}^{\nabla\ell}\right) && (4.14)\\
\geq& \|\nabla\ell\|_2^2 + \underbrace{r^2 - 4\|\nabla\ell\|_2^2}_{\geq 0} && (4.15)\\
\geq& \|\nabla\ell\|_2^2 && (4.16)
\end{aligned}$$

is always true, and thus $P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) = 1$ by posititivity of the norm proving the second part of the statement. It should be also visually straightforward from Fig. 4.4.

Let's further consider the case $r \leq 2\|\nabla\ell\|_2$. Given the radius r and $\|\nabla\ell\|_2$ (deterministic gradient) consider the triangle PQR with sides $|PR| = \|\nabla\ell_i\|_2$, $|PQ| = \|\nabla\ell\|_2$ and $|QR| = r$, see Fig.4.4. Point R of the triangle follows the uniform distribution on the circle with a center Q and radius r and thus considering polar coordinate system with origin at Q we have $\|\varepsilon\|_2 = r$ and $R \sim$ Unif$(S_1(r))$ has a coordinate θ measured from vertical line going through Q pointing upwards, as in Fig.4.4.

Now consider the angle α defining a point $R = (r, \theta = \alpha/2)$ such that $|PR| = |PQ|$, i.e., the length of the sides of the triangle with the common vertex P are equal. By elementary geometry, it follows that $\measuredangle QPR = \alpha$.

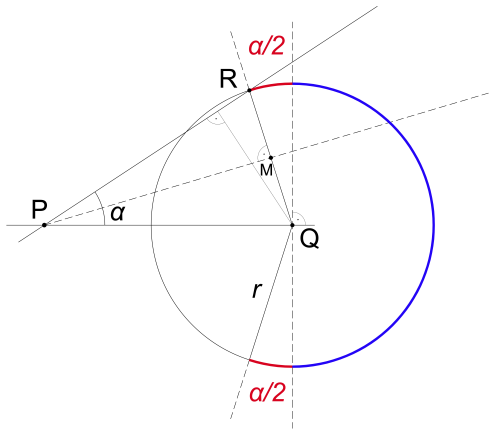


Figure 4.4: *Consider the angle $\measuredangle QPR = \alpha$ such that the triangle PQR has lengths of sides $|PQ|$ and $|PR|$ equal. Such $\alpha \in (0, \pi)$ always exists for any given radius $2|PQ| > r > 0$ of a circle $S_1(r)$ with a center at Q. It follows the length of (blue) arc, i.e., a semi-circle, plus nonzero lenghts of two (red) arcs are always larger than half of the circumference of $S_1(r)$, i.e., the length of (blue) arc. Consider R uniformly distributed on the circle $S_1(r)$. Then the probability (in a sense of Unif$(S_1(r))$) of $|PR|$ being larger then $|PQ|$ is proportional to arc lengths and is larger than $\frac{1}{2}$ for any $2|PQ| > r > 0$. Note that for $r \geq 2|PQ|$ this probability is always 1.*

This is a boundary position in the sense that the norm of $\nabla \ell_i$ is strictly larger if the point R lies anywhere on the (blue) or the (red) *arc* of the circle and strictly smaller in the opposite case, as depicted in Fig. 4.4. It follows by the well-known triangle inequality combined with the Pythagorean theorem applied on triangle PMR for instance.

Note that (blue) arc is a semi-circle. Thus it follows that (blue)+(red) arc length is strictly larger than half of the circumference for any $r > 0$. Since R is uniformly distributed on the circle the arc lengths are proportional to probabilities and we can write

$$P\left(\|\nabla \ell_i\| \geq \|\nabla \ell\|\right) = \frac{\text{red + blue arc}}{\text{circumference}} = \frac{(\pi + \alpha)r}{2\pi r} = \frac{1}{2} + \frac{\alpha}{2\pi} \tag{4.17}$$

Finally from triangle PMQ we get that $\sin \frac{\alpha}{2} = \frac{r/2}{\nabla \ell}$ and thus

$$\alpha = 2 \arcsin \frac{\|\varepsilon\|_2}{2\|\nabla \ell\|_2} \tag{4.18}$$

where we plugged in $|PR| = \|\nabla \ell_i\|_2$, $|PQ| = \|\nabla \ell\|_2$ and $|QR| = r = \|\varepsilon\|_2$. □

Corollary 4.4. *Consider vector* $\nabla\ell_i = \nabla\ell + \varepsilon \in \mathbb{R}^N$, *where* $\nabla\ell$ *is a given constant vector and* $\varepsilon \sim Unif(rS_{N-1})$, *is uniformly distributed random vector on the circle of the given radius* $r > 0$ *i.e.* $\|\varepsilon\|_2 = r$. *If* $r \leq 2\|\nabla\ell\|_2$ *then*

$$P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) = 1 - \frac{1}{2} I_{\cos^2(\alpha/2)}\left(\frac{N-1}{2}, \frac{1}{2}\right). \tag{4.19}$$

where

$$\alpha = 2\arcsin\frac{\|\varepsilon\|_2}{2\|\nabla\ell\|_2} \tag{4.20}$$

as in Lemma 4.3 and 1 *otherwise, i.e., for* $r > 2\|\nabla\ell\|_2$.

The $I_{\cos^2\alpha/2}\left(\frac{n-1}{2}, \frac{1}{2}\right)$ *is a regularized Beta function, [Myland et al., 2008], defined as* $I(z;a,b) = B(z;a,b)/B(a,b)$ *where* $B(z;a,b)$ *is incomplete beta function* $B(z;a,b) = \int_0^z u^{(a-1)}(1-u)^{(b-1)}du$ *and* $B(a,b)$ *is a complete Beta function defined as* $B(1;a,b)$.

Proof. Note that the second part of the statement for $\mathbb{R}^N$ follows by an identical argument as in $\mathbb{R}^2$ because it is based on well known Euclidean norm relation that is independent of dimension, see Theorem 4.3.

Overall, the idea of the proof follows the same line of arguments as in R^2 just instead of the circle in $\mathbb{R}^2$ the $(N-1)$-spheres are considered. Because of the direction is uniformly distributed over the hyper-sphere rS_{N-1} the probability $P\left(\|\nabla\ell_i\| \leq \|\nabla\ell\|\right)$ is proportional to a ratio of the area of hyperspherical cap to the area of the whole hypersphere in an exact analogy to R^2.

An area of the hyperspherical cap on the $S_n(r)$ given by a co-latitude θ, the angle between vector of the sphere and its positive n^th axis, i.e., the axis from center of the sphere to the "north" pole), is by work [Li, 2010] given as

$$\frac{1}{2}A_n(r)I_{\sin^2\theta}\left(\frac{n-1}{2}, \frac{1}{2}\right) \qquad \text{(Area of a hyperspherical cap on } S_{n-1}(r)\text{)}$$

where $A_n(r)$ is a area of $(n-1)$-sphere $S_{n-1}(r)$ and $I_{\sin^2\alpha}\left(\frac{n-1}{2}, \frac{1}{2}\right)$ is the regularized Beta function as defined in the corollary. Note that in our case, relating to Fig. 4.4, the co-latitude is $\theta = \frac{\pi-\alpha}{2}$.

For more details on regularized Beta function kindly refer to the Chapter 4 of this thesis[6] or [Myland et al., 2008] or Wolfram Math World *https://mathworld.wolfram.com/IncompleteBetaFunction.html*.

[6] IEEE TPAMI paper "Bayesian Cut" uses incomplete Beta function and some if its properties.

Note that (Area of a hyperspherical cap on $S_{n-1}(r)$) corresponds to $P\left(\|\nabla\ell_i\| \leq \|\nabla\ell\|\right)$, that is a complementary probability to the one wanted. Further it contains a factor $A_n(r)$, i.e., an area of the whole hypersphere. Then ratio of hypershperical cap area over the area of whole sphere $A_n(r)$ simply equals the value of the regularized Beta function. Hence the probability $P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right)$ and statement of the corollary to be proven is given as

$$P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) = 1 - \frac{1}{2} I_{\cos^2(\alpha/2)}\left(\frac{N-1}{2}, \frac{1}{2}\right). \tag{4.21}$$

A relation between α and ε translates to high dimensions unchanged from $\mathbb{R}^2$ because the high dimensional case is symmetric around the n^{th} co-latitude axis the same way as $\mathbb{R}^2$ case w.r.t. PR, see Fig.4.4)

$$\alpha = 2 \arcsin \frac{\|\varepsilon\|_2}{2\|\nabla\ell\|_2} \tag{4.22}$$

which concludes the proof. □

Note the value of the regularized incomplete Beta in the Corollary 4.4 can be interpreted as the probability of a random vector on a hemisphere falling onto a cap defined by co-latitude $\frac{\pi-\alpha}{2}$. Letting alone that such a view may serve as a basis for efficient sampling from such spherical distribution, see [Li, 2010], it may also be used as a numerical test (numerically efficient evaluation of Beta functions are available in major ML software platforms,e.g., Python [vanRossum, 1995], see also the Chapter 4) that training follows the hereby described behavior. Exploring these avenues is left for future work.

Up until this point we have worked in a "marginal" regime with a given fixed radius set to an arbitrary yet given value r. Next, we alleviate this marginality and consider stochastic back-prop training under LiMoD settings of this Chapter in the full generality.

Definition 4.5. *(SGD setting) Consider SGD mini-batch training given by given by Equation (4.11) with batch index i where realization of "noise" variable ε for the batch i is denoted ε_i and "noisy" i^{th} batch gradient $\nabla\ell_i$. In SGD mini-batch training we have $\nabla\ell_i = \nabla\ell + \varepsilon$ by (4.11) and ε is considered to be zero centered Gaussian $\varepsilon \sim N(\mathbf{0}, \Sigma)$ as derived earlier following by CLT in (4.4).*

Let's define radius and direction of ε given by

$$R_{sgd}(N) := \|\varepsilon\|_2, R^2_{sgd}(N) \sim \chi^2(N) \qquad \text{(SGD radius)}$$

$$U_{sgd}(N) := \frac{\varepsilon}{\|\varepsilon\|_2} \sim \mathit{Unif}(S_{(N-1)}) \qquad \text{(SGD direction)}$$

the random variables given by Lemma 4.5. Further we have

$$E[\|\varepsilon\|_2] = \sqrt{N}. \tag{4.23}$$

and by the Concentration theorem 4.7 applied on ε we have that R_{sgd} fluctuates (exponentially) close to the radius $\sqrt{N}$.

Note that by the corollary 4.4 the probability of noisy (stochastic) gradient $\nabla\ell_i$ to be larger than "not-noisy" (full batch) $\nabla\ell$ is always larger then $\frac{1}{2}$ for any given positive value of $R_{sgd}(N) = r = \|\varepsilon_i\| > 0$ and this probability only depends on this norm, i.e., radius $R_{sgd}(N)$ of the sphere and not on the direction $U_{sgd}(N)$.

Ultimately we are interested in dependence on dimension N. From the corollary 4.4 it follows it depends on N through $R_{sgd}(N)$, defining area of the hyper-spherical cap through $\alpha = 2\arcsin\frac{R_{sgd}(N)}{2\|\nabla\ell\|_2}$ as well as directly through the first argument of $I_{\cos^2(\alpha/2)}\left(\frac{N-1}{2}, \frac{1}{2}\right)$ from (4.19).

The next theorem and its corollary show that this double dependence on N is not in contradiction and moreover is monotonically increasing with the rank of the covariance matrix of the noise ε.

Theorem 4.6. *(Norm of isotropic noisy gradient) Under SGD settings 4.5 with identity stochastic gradient covariance matrix $\Sigma = \mathbb{I}_N$ the following holds*

$$P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) > \frac{1}{2} \quad \textit{for any } N > 1, \textit{ and} \tag{4.24}$$

$$\lim_{N\to\infty} P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) = 1 \tag{4.25}$$

almost surely. More over $P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right)$ is monotonically increasing to 1 with increasing rank(Σ), i.e., N.

Proof. Consider an arbitrary step t of stochastic back-prop training 4.5. Then value of the full gradient $\nabla\ell$ at t is deterministic, i.e. non-random, while $\nabla\ell_i$ at t is a random zero mean isotropic Gaussian variable according to assumptions.

$P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) > 1/2$ *a.s.:* The first statement of this theorem is a direct consequence of the corollary 4.4 that yields the probability $P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) > \frac{1}{2}$ whenever value of $r = \|\varepsilon\|_2 > 0$. But this is true with probability 1 because $\varepsilon \sim N(\mathbf{0}, \Sigma)$ and $P(\varepsilon = 0) = 0$. In other words the statement holds "almost surely" (a.s.), supported by Fig.4.4 geometrically.

Monotonicity: Recall the norm of full gradient $\|\nabla\ell\|$ is given constant. Consider $\mathbb{R}^2$ case in Fig.4.4. Radius $R_{sgd} = \|\varepsilon\|_2$ defining radius of circle r is now random

variable. Never the less, note that $P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) = 1$ whenever value of R_{sgd} on batch i satisfies $\|\varepsilon_i\|_2 \geq 2\|\nabla\ell\|_2$ by the Theorem 4.3 and its Corollary 4.4 for $\mathbb{R}^N$. But from concentration theorem 4.7 we know that R_{sgd} concentrates exponentially closely around $\sqrt{N}$ and thus increases with dimension N.

Following this idea we $P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) =$ by random radius R_{sgd} and prove that both factors are monotone and increasing with N. Formally we use sum rule of probability

$$P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right)(N) = \underbrace{P\left(\{\|\nabla\ell_i\| \geq \|\nabla\ell\|\} \;\&\; \{R_{sgd}(N) < 2\|\nabla\ell\|_2\}\right)}_{P_S(N)} \tag{4.26}$$

$$+ \underbrace{P(R_{sgd}(N) \geq 2\|\nabla\ell\|_2}_{P_R(N)} \tag{4.27}$$

and show that $P_S(N)$ is increasing and $P(R_{sgd}(N)$ is non-decreasing with increasing N until they reach 1.

Monotonicity of $P_S(N)$ Note that by corollary 4.4 $P_S(N) = 1$ when $r > 2\|\nabla\ell\|_2$ and thus we only need to prove the case $r \leq 2\|\nabla\ell\|_2$.

In this case the same corollary gives direct formula for $P_S(N)$

$$P\left(\|\nabla\ell_i\| \geq \|\nabla\ell\|\right) = 1 - \frac{1}{2} I_{\cos^2(\alpha/2)}\left(\frac{N-1}{2}, \frac{1}{2}\right) \tag{4.28}$$

where $\alpha \in (0, \pi)$. In such case $\cos^2(\alpha/2)$ is monotonically decreasing on $\alpha \in (0, \pi)$ with stationary points $0, \pi$. Since $\alpha = 2\arcsin\frac{\|\varepsilon\|_2}{2\|\nabla\ell\|_2}$.

Now using the fact that the incomplete Beta $B_z(a, b)$ is monotonically decreasing function of both arguments a and b and monotonically increasing in z[7], [Myland et al., 2008], p.604, yields required.

In a larger detail: For the proof of monotonicity w.r.t. a (and b, by symmetry) we only need it for $z \in [0, 1]$ because we have $z = cos^2$ fixed and bounden by 1 from above. Then decreasing monocity can be shown from the definition $B(z; a, b) = \int_0^z u^{(a-1)}(1-u)^{(b-1)}du$ because increasing a decreases the inside of the integral for all u, i.e. on the whole interval $u \in (0, 1)$. Integral can be approximated by the Riemanian sum ($B(z; a, b)$ smooth function w.r.t. Lebesgue measure) over $f(u)du$, where f denotes inside of the integral. Every sum is strictly smaller for $a' = a + \epsilon, \epsilon > 0$ compared to the the same sum with a and by taking the limit $du \to 0$ and continuity the monotonicity follows. For a geometric intuition see Fig.4.4 where increasing N means increasing $r = \|\varepsilon\|_2$ and thus

[7] fixing all other arguments but one in focus

$\alpha = 2\arcsin\frac{\|\varepsilon\|_2}{2\|\nabla\ell\|_2}$ leading to increasing length of a (red) arc. That $B(z;a,b)$ is monotonically increasing in $z, z \in [0,1]$ is straightforward from its definition above and strict positivity of the defining integrand on $u \in [0,z)$.

Asymptotics: Making use of [Myland et al., 2008] and *https://mathworld.wolfram.com/IncompleteBetaFunction.html* following approximations can be found and used to shed light on asymptotic behaviour of $I_{\cos^2(\alpha/2)}\left(\frac{N-1}{2},\frac{1}{2}\right)$ when $N \to \infty$

$$B(z;a,b) \propto \frac{z^a}{a}\left(1+O(z)\right) \tag{4.29}$$

$$B(a,b) \propto \Gamma(b)a^{-b} \tag{4.30}$$

Then, fixing everything but N, the limit is

$$\lim_{N\to\infty} I_{\cos^2(\alpha/2)}\left(\frac{N-1}{2},\frac{1}{2}\right) = \lim_{N\to\infty}\frac{(\cos^2(\alpha/2))^{N-1}(N-1)^{-1/2}}{\Gamma(1/2)}\left(1+O(\cos^2(\alpha/2))\right) = 0 \tag{4.31}$$

by $cos^2(\cdot)$ being bounded from above by 1.

Non-decreasing $P_R(N)$ The norm concentration theorem 4.7 says that $P(\mid R_{sgd} - \sqrt{N} \mid \geq t) \leq 2\exp\{-ct^2\}$ where RHS is independent on N, c is positive constant and $t > 0$ is a distance from $S_{N-1}(\sqrt{N})$. In other words $R_{sgd}(N)$ lies in the interval $(\sqrt{N}-t, \sqrt{N}+t)$ with probability $1-2\exp\{-ct^2\}$ and this probability does not depend on N.

Next fix N and take t such that $\sqrt{N}-t = 2\|\nabla\ell\|$ and consider some $T > N$. Then we also have $R_{sgd}(T)$ lies in the interval $(\sqrt{T}-t, \sqrt{T}+t)$ with the same probability probability $1-2\exp\{-ct^2\}$. Because both these probabilities are same and independent on T and N for all c and t their one sided probabilities also equal (given c and t), i.e., $P(R_{sgd}(N) \geq 2\|\nabla\ell\|_2) = P(R_{sgd}(T) \geq \sqrt{T}-t)$. Because probability being positive measure on sigma algebra it follows that if $B \subseteq A$ then $P(A) \geq P(B)$. And thus

$$P(R_{sgd}(T) \geq 2\|\nabla\ell\|_2) \geq P(R_{sgd}(T) \geq \sqrt{T}-t) = P(R_{sgd}(N) \geq 2\|\nabla\ell\|_2) \tag{4.32}$$

because $(\sqrt{T}-t, \infty) \subseteq (2\|\nabla\ell\|_2, \infty)$. Since we took arbitrary $T > N$ it proves $P_R(N)$ is non-decreasing in N. □

4.0.3 The Norm of the Gradient as Anti-Overfitting Prior

Let us recall what the role of the norm of the (stochastic) gradient is in the back-prop learning. We consider LiMoD training formulation of a standard SGD training using (LiMoD learning rate) in combination with normalized gradient (LiMoD gradient) and step dependent loss (LiMoD objective) as defined in Definition 3.17 in the first part of this chapter.

In particular we have shown (restating the formulas (3.57) here) that

$$\tau_t \nabla^2 \Phi(0) \approx \nabla^2 \Phi_t(f(\boldsymbol{x}_s, \boldsymbol{w})) \tag{4.33}$$

$$\tau_t \left\| \nabla^2 \Phi(0) \right\| \geq \left\| \nabla^2 \Phi_t(f(\boldsymbol{x}_s, \boldsymbol{w})) \right\| \tag{4.34}$$

where $\nabla^2 \Phi_t(f(\boldsymbol{x}_s, \boldsymbol{w}))$ is a variance of the model being trained by back-prop minimizing (3.16) over targets at update t. Thus larger τ_t leads to a wider variance of the model driven by a larger $D_K L(g||\nu_z)$ penalty for over-fitting.

In other words rephrasing one of the main results of the thesis, the corollary 3.16, the loss is minimized not by maximizing the likelihood itself but rather by striking a balance between "exact fit" and "small variance" (over-fitting) penalty.

4.0.3.1 Role of Other Cumulant Priors

Intriguingly, the norm of the gradient is not the only implicit regularizer acting during the back-prop training. In the LiMoD training, defined in 3.17, **matching the second cumulants** (3.52), imposing the variance prior that prefers large gradient norms, **is always accompanied by matching the first**[8] **cumulants** (3.51) that prevent the outputs of f to be "large", as elaborated in Section 3.0.3.1.

Notably, this does not mean that all weights have to be small to generalize well. In fact, large weights responding to so-called "irrelevant features"[9] [Bengio et al., 2021] are allowed, i.e., are not implcitly regularized, by SeReBrO. In a sense, it suggests a diffusion to irrelevant directions is not only allowed but may be helpful in terms of generalization. Further research along these lines is needed, encouraged, and is left for future work.

[8] and higher cumulants depending on the chosen loss function, or rather Ψ from which it derives. The higher cumulants impose skewness and other priors in general. Further research is needed and is left for future work.

[9] features that do not correlate with changes of target variables

4.0.3.2 Arbitrary Noise Covariance $\Sigma \succeq 0$

Because τ_t, defined in (LiMoD learning rate) and restated here for brevity as $\frac{\eta_t \|\nabla \ell_t\|_2}{|B_t|}$, depends on the norm of $\|\nabla \ell_t\|_2$ one can see that norm of the gradient $\|\nabla \ell_t\|_2$ controls the variance of the model being fitted and acts as a (step t) prior. Note that this prior changes every step in general.

If this is so and considering the results of the previous theorem, we should see all deep models generalize well or gradients explode. Why is it not that so? In practice, stochastic gradients are not isotropic as in the previous theorem and are "dampened" by a spectrum of their covariance matrix $\Sigma \succeq 0$ (besides other factors such as learning rate, etc.).

In what follows, the previous results will be leveraged to shed light on the training with an arbitrary $\Sigma \succeq 0$.

Definition 4.7. *(kernel and image of linear operator) Let A be a linear operator on a vector space V. Then subspace $ker(A) = \{v \in V : Av = 0\}$ is called* kernel *of A and the subspace $im(A) = \{Av : v \in V\}$ is called the* image *of A.*

Consider an arbitrary real square $D \times D$ noise covariance matrix $\Sigma \succeq 0$ of rank N. Consider a standard Euclidean vector space R^D and define the null space of matrix A as $ker(A) = \{x : Ax = 0\}$ a null space of Σ, [Roman et al., 2005]. Because $\Sigma \succeq 0$ then its rank $N \leq D$ and $D - N$ is the dimension of the null space of Σ, i.e. and also a number of zero eigenvalues.

In an analogy to previous chapter let's denote N *positive* eigenvalues of a real symmetric matrix (so it has real eigenvalues) Σ of dimension N, by $0 < \lambda^+_{min}(\Sigma) = \lambda^+_1(\Sigma) \leq \lambda^+_2(\Sigma) \leq \cdots \leq \lambda^+_N(\Sigma) = \lambda^+_{max}(\Sigma)$.

Because Σ is a real and symmetric one can invoke a well known singular value decomposition (SVD) [Roman et al., 2005, Bhatia, 1997, Gantmakher, 1959] to write it as $\Sigma = U \Lambda U^T$ where U is an orthonormal matrix with eigenvectors of Σ in columns and Λ is a diagonal matrix with $\{\lambda^+_i\}_{i=1}^N$ on diagonal and the rest of diagonal elements $N - D$ are zeros.

Let's define the $D \times D$ matrix $\Sigma^{\frac{1}{2}} := (U \Lambda^{\frac{1}{2}})$ where $\Lambda^{\frac{1}{2}}$ is a diagonal matrix with $\{\sqrt{\lambda^+_i}\}_{i=1}^N$ on diagonal. The rest of diagonal elements of Λ $N - D$ are zeros. Let's assume without loss of generality the and the diagonal Λ is in descending order and thus first N columns of U are eigenvalues corresponding to positive eigenvalues, the rest span null space of Σ. Then $\Sigma^{\frac{1}{2}} \Sigma^{\frac{1}{2}^T} = \Sigma$.

Theorem 4.8. *(Expected norm of $\Sigma^{\frac{1}{2}}\varepsilon$) Under notation of this section consider a random Gaussian vector $\varepsilon \sim N(\mathbf{0}, \mathbb{I}_D)$. Then the expected norm of random vector $\Sigma^{\frac{1}{2}}\varepsilon$ is lower bounded by*

$$\sqrt{rank(\Sigma)\lambda^+_{min}(\Sigma)} \leq E\left\|\Sigma^{\frac{1}{2}}\varepsilon\right\|_2 \leq \sqrt{rank(\Sigma)\lambda^+_{max}(\Sigma)} \tag{4.35}$$

where $\lambda^+_{min}(A)$ and $\lambda^+_{max}(A)$ denotes the smallest and largest positive *eigenvalue of Σ respectively.*

Proof. Because Σ is a linear operator on $\mathbb{R}^D$ we can write $\mathbb{R}^D$ as a direct sum $\mathbb{R}^D = ker(\Sigma) \oplus im(\Sigma)$, see [Roman et al., 2005], Theorem 2.21, and we also have $ker^{\perp}(\Sigma) = im(\Sigma)$[10]. Hence we write $\varepsilon = \varepsilon^{\perp} + \varepsilon^0$, where $\varepsilon^0 \in ker(\Sigma)$ and $\varepsilon^{\perp} \in im(\Sigma)$. Recall that N denotes the $rank(\Sigma)$, i.e., number of its positive eigenvalues.

Since $\varepsilon \sim N(\mathbf{0}, \mathbb{I}_D)$ and for Gaussian distribution "orthogonal is independent", [Bishop, 2006, Gelman et al., 2013], expectation over $\mathbb{R}^D = ker(\Sigma) \oplus im(\Sigma)$ is expectation over kernel $ker(\Sigma)$ + expectation over the image $im(\Sigma)$ as follows

$$E\left[\left\|\Sigma^{\frac{1}{2}}\varepsilon\right\|_2^2\right] = E_{ker(\Sigma)}\underbrace{\left[\left\|\Sigma^{\frac{1}{2}}\varepsilon^0\right\|_2^2\right]}_{=\mathbf{0}} + E_{ker^{\perp}(\Sigma)}\left[\left\|\Sigma^{\frac{1}{2}}\varepsilon^{\perp}\right\|_2^2\right] \tag{4.36}$$

where $E_{ker(\Sigma)}$ and $E_{ker^{\perp}(\Sigma)}$ denotes the conditional expectation over kernel and image of Σ respectively.

Further we make use of the Rayleigh quotient formula on a real symmetric A^TA that holds for any real matrix A, [Parlett, 1998],

$$\sup_{x\neq 0}\frac{\|Ax\|_2^2}{\|x\|_2^2} = \lambda_{max}(A^TA) \implies \|Ax\|_2^2 \leq \lambda_{max}(A^TA)\|x\|_2^2 \tag{4.37}$$

$$\inf_{x\neq 0}\frac{\|Ax\|_2^2}{\|x\|_2^2} = \lambda_{min}(A^TA) \implies \|Ax\|_2^2 \geq \lambda_{min}(A^TA)\|x\|_2^2 \tag{4.38}$$

and apply it on the $\left\|\Sigma^{\frac{1}{2}}\varepsilon\right\|_2^2$ with constraint on $\varepsilon \notin ker(\Sigma)$, i.e. $\left\|\Sigma^{\frac{1}{2}}\varepsilon^{\perp}\right\|_2^2$.

Note that this constraint ensures the minimum of the norm is positive. And because positive eigenvalues of matrices $\Sigma^{\frac{1}{2}}\Sigma^{\frac{1}{2}^T}$, $\Sigma^{\frac{1}{2}^T}\Sigma^{\frac{1}{2}}$ and Σ are the same by the construction above, the inside of the expectation (4.36) has following bounds

$$\|\varepsilon\|_2^2\,\lambda^+_{min}(\Sigma) \leq \left\|\Sigma^{\frac{1}{2}}\varepsilon\right\|_2^2 \leq \|\varepsilon\|_2^2\,\lambda^+_{max}(\Sigma). \tag{4.39}$$

[10] as also follows from (SVD) above noting eigenvectors corresponding to zero eigenvalues are orthogonal to non-zero ones

Now taking the conditional expectation $E_{ker^{\perp}(\Sigma)}$ and plugging the (4.36), that showed the conditional expectation over image of Σ equals the unconditional expectation $E_{ker^{\perp}(\Sigma)}\left[\left\|\Sigma^{\frac{1}{2}}\varepsilon^{\perp}\right\|_2^2\right] = E\left\|\Sigma^{\frac{1}{2}}\varepsilon\right\|_2^2$, we can replace the middle term in inequalities above and obtain

$$\lambda^{+}_{min}(\Sigma)E_{ker^{\perp}(\Sigma)}\|\varepsilon\|_2^2 \leq \left\|\Sigma^{\frac{1}{2}}\varepsilon\right\|_2^2 \leq \lambda^{+}_{max}(\Sigma)E_{ker^{\perp}(\Sigma)}\|\varepsilon\|_2^2 . \tag{4.40}$$

Further ε has i.i.d. coordinates and is rotation invariant, i.e., invariant with respect to orthonormal change of basis, we can effectively replace conditional expectation of the square of the norm over the N-dimensional subspace $ker^{\perp}(\Sigma)$ by the expectation over $\mathbb{R}^N \subseteq \mathbb{R}^D$ of the original space $\mathbb{R}^D$ shown to be equal N

$$N\lambda^{+}_{min}(\Sigma) \leq E\left\|\Sigma^{\frac{1}{2}}\varepsilon\right\|_2^2 \leq N\lambda^{+}_{max}(\Sigma). \tag{4.41}$$

which by taking square root (all elements are strictly positive) concludes the proof. □

Definition 4.9. *(Lipschitz functions on metric spaces). Let (X, d_X) and (Y, d_Y) be metric spaces. A function $f : X \to Y$ is called Lipschitz if there exists $L \in \mathbb{R}$ such that*

$$d_Y(f(u), f(v)) \leq L d_X(u, v) \quad \forall u, v \in X \tag{4.42}$$

The infimum of all L in this definition is called the Lipschitz norm of f and is denoted $\|f\|_{Lip}$.

For completeness let's note there is a general version of concentration for Lipschitz functions on a sphere as stated in [Vershynin, 2018], Chapter 5.

Theorem 4.10. *(Theorem 5.4.1, [Vershynin, 2018], Concentration of Lipschitz functions on the sphere) Consider a random vector $X \sim Unif(S_{N-1}(\sqrt{N}))$, i.e. X is uniformly distributed on the Euclidean sphere of radius $\sqrt{N-1}$. Consider a Lipschitz function $f : (S_{N-1}(\sqrt{N})) \to \mathbb{R}$. Then for every $t \geq 0$, we have*

$$P\{|f(X) - E[f(X)]| \geq t\} \leq 2\exp\left\{-\frac{ct^2}{\|f\|_{Lip}}\right\}. \tag{4.43}$$

Note that if we take $f : X \to \left\|\Sigma^{\frac{1}{2}}X\right\|_2$ and knowing the norm is Lipschitz, [Vershynin, 2018], [Bhatia, 1997][11] than we have by previous theorem that $\left\|\Sigma^{\frac{1}{2}}X\right\|_2$

[11] Follows by defining matrix norm from a vector norm $\|\cdot\|$ as $\|\Sigma\| = \sup\{\|\Sigma x\| : \|x\| \leq 1\}$ and triangle inequality, one gets Lipschitz constant $L = \|\Sigma\|$.

concentrates around its expected value tightly in t and reciprocally w.r.t. Lipschitz constant.

Corollary 4.11. *(Gradient norm is bounded by rank of Σ in Deep Learning) Consider LiMoD training from Definition 3.17 and the stochastic gradient (4.11) with an arbitrary covariance matrix $\Sigma \succeq 0$ with rank $d > 1$, i.e., with noise $\varepsilon \sim N(0, \Sigma)$. Then expected norm of stochastic gradient on batch i and at arbitrary step t is approximately bounded from above by*

$$\|\nabla \ell_i\|_2^2 \leq \|\nabla \ell\|_2^2 + rank(\Sigma)\lambda_{max}(\Sigma) + t \tag{4.44}$$

with probability $1 - 2\exp\left\{-\frac{ct^2}{\|f\|_{Lip}}\right\}$.

Proof. (Sketch) By triangle inequality we have

$$\|\nabla \ell_i\|_2 \leq \|\nabla \ell\|_2 + \|\varepsilon\|_2 . \tag{4.45}$$

Further from the theorem 4.10 when taking $f : X \to \left\|\Sigma^{\frac{1}{2}} X\right\|_2$ we have

$$P(E[f(X)] - t \leq f(X) \leq E[f(X)] + t) \leq 2\exp\left\{-\frac{ct^2}{\|f\|_{Lip}}\right\}. \tag{4.46}$$

Finally, taking the upper bound on expected norm from previous theorem 4.8 and plugging into the RHS of $f(X) \leq E[f(X)] + t$ above the statement follows. □

4.0.4 Implications

The theorems 4.6 for isotropic noise and theorem 4.8 together with Theorem 5.4.1 [Vershynin, 2018] for arbitrary Sigma all show that stochastic gradient norm is larger than the norm of a full gradient and more likely so with the growing number of parameters, i.e. the dimension of weights.

But as opposed to isotropic noise the theorem 4.8 and its corollary 4.11 show that the large dimension of weights is only enabler and it is effectively reduced by rank of covariance matrix Σ. This is in the author's opinion a highly interesting result with many theoretical and practical implications some of which are stated in what follows.

First and foremost, the main conclusion of this chapter is that any method increasing the rank of noise covariance Σ prevents overfitting. Let's further elaborate on this in detail.

4.0.4.1 The Larger Rank of Noise Covariance Σ, The Less Overfitting

This is a direct consequence of theorems 4.6 for isotropic noise and theorem 4.8 together with Theorem 5.4.1 [Vershynin, 2018] that show the stochastic gradient leads to a larger τ_t and thus imposes a wider prior on variance of the model $\tau_t \left\| \nabla^2 \Phi(0) \right\|$ from (3.57) 3.17 leading to arguably "simpler", i.e., not over-fitting, network as argued in Section 4.0.3.

4.0.4.2 SGD Generalizes Better Than GD in Overparameterized networks

In this perspective, the "full" gradient learning (GD) corresponds to Σ with rank 0. Starting there the mini-batch (or stochastic gradient) (SGD) training [Robbins and Monro, 1951], known to be improving generalization in theory and practice [Bottou and Bousquet, 2007, Goodfellow et al., 2016], increases the rank of Σ by the construction detailed in the Section (4.0.0.1). Such doing leads by the theory developed in this thesis to wider priors during the training resulting in better generalization of SGD compared to GD in deep learning as also argued and empirically demonstrated by numerous papers, [Goodfellow et al., 2016, Zhang et al., 2016, Bengio et al., 2021, Bishop, 2006, Zhang et al., 2021, Hinton, 2012] and many others.

4.0.4.3 Common Regularizers Put Into LiMoD Perspective

Notably relying on "vanilla" SGD may not be enough. Letting alone an influence of hyperparameters, the "anti-overfitting" prior controlled by $rank(\Sigma)$, acting in τ_t through a norm of the SGD gradient, is in every step t of the vanilla SGD case bounded by the size of a training data, denoted earlier as $|\mathcal{D}|$. This is straightforward to see from Section 4.0.0.1 with Σ being the sample covariance, i.e. sum of matrices of $|\mathcal{D}|$ unit ranks.

So it is common practice to combine SGD with additional regularizers some of which can be cast as methods boosting the $rank(\Sigma)$ by injecting the noise one way or the other (see below), such as

1. *data augmentation* [Goodfellow et al., 2016], combined with SGD increases $rank(\Sigma)$ by augmenting sample data size as Σ is in this case a sample covariance, as in Section 4.0.0.1

2. injected *input and/or target data noise* effectively increases rank of Σ in the same way as augmentation, [Poole et al., 2014]

3. injecting *gradient or weight noise* often imposes Σ with a full rank, i.e., $N = D$ in Theorem 4.8 and its corollary, due to i.i.d. of its coordinates. It depends on the type of noise injected. One way or the other, the gradient covariance is a mixture of the injected noise covariance and the sample covariance (of rank 0 in the case of GD instead of SGD). The work of [Neelakantan et al., 2016] et. al. analyzes different methods of gradient noise with strong empirical evidence supporting its generalization merits. Weights noise injection is argued to help robustness against adversarial attacks in [Noh et al., 2017].

4. injecting *activation noise* including popular regularizers as drop-out (can be also seen as an ensemble averaging over range of random sub-models due to dropped out units, [Hinton et al., 2012b, Srivastava et al., 2014], see next point), batch normalization [Ioffe and Szegedy, 2015], see next point, etc. Injecting noise into activations during the SGD training together with the bottleneck architecture is used in denoising autoencoders (DAE) in [Poole et al., 2014], where it is argued and practically shown to generalize well. In the next Experimental section, we show that bottle-neck architecture can be omitted, when the noise and depth are applied as a regularizer instead. A novel and direct experiment in support of the conclusions of this thesis.

5. *batch normalization* [Ioffe and Szegedy, 2015, Santurkar et al., 2018] As opposed to vanilla SGD training, BN ensures all activations (after normalization) fluctuate (so-called "internal covariate shift", [Ioffe and Szegedy, 2015]) around zero with the second moment ≈ 1 even in the small gradients regime, causing output layer parameters fluctuate. Thus even on the flat areas of loss gradient (norm) is protected from collapsing, imposing the wider variance prior $\tau_t \left\| \nabla^2 \Phi(0) \right\|$ compared to a case without BN.

6. *skip connections* as in ResNet [He et al., 2016], shown to be helping generalization [He et al., 2020, Rousseau and Fablet, 2018, Hauser and Ray, 2017] besides making deep model trainable. ResNets can be seen as a ensemble of networks [Hansen and Salamon, 1990] as presented in [Veit et al., 2016]. From the LiMoD perspective an ensemble averaging (could also include dropout [Hinton et al., 2012b, Srivastava et al., 2014, Goodfellow et al., 2016]) also increases the rank of $rank(\Sigma)$ by averaging of sample covariances over many "ensemble" models. This is due to "ensemble" models having different weights and thus value of loss functions on the same (batch) data inputs at the particular step t.

4.1 Experiment: DAEs Self-Regularized by Width, Depth, and Rank of the Gradient Noise Σ

The experiment of this section is as well partially reported in Paper A, Figures 1. and 2., to motivate the claim that explicit regularizer, in this case, the bottle-neck layer and injected data input noise in denoising auto-encoders (DAE), [Vincent et al., 2008], can be replaced by an over-parameterization of the model, in this case, both width and depth are large, trained with SGD and batch-normalization (BN).

More precisely, DAEs are an extension of auto-encoders trained to reconstruct a clean version of input from its corrupted version [Poole et al., 2014]. To demonstrate the ability of deep networks trained with a stochastic gradient to learn the robust solution we train the DAE model to fit the identity function *without adding noise to input data* as opposed to a recommended procedure in [Vincent et al., 2008, Poole et al., 2014].

The main methods used to regularize DAE are bottle-neck architecture or noise injected into inputs during training. Otherwise, DAE learns an identity map. In the experiment below neither of these is used. The architecture of convolutional layers is wide of an "anti-bottleneck" shape '<>' instead of recommended bottleneck '><', and *there is not any noise injected* into training samples.

Instead we used depth combined with mini-batch (SGD) training, see Section 4.0.0.1, that ensures the rank of gradient noise covariance Σ from previous section is positive and thus imposing an implicit LiMoD wide priors on model variance (3.56).

In this case, it is 25 layers deep auto-encoder trained with stochastic (mini-batch) gradient descent with batch-normalization (BN). In detail, two following auto-encoder models are compared on the widely known and used hand-written digits MNIST dataset. The version from PyTorch datasets module, [Paszke et al., 2017].

- **Shallow AE+BN** (1 (encoder) + 2 (decoder) CNN layers, ReLU activations, with BN), 784 neurons in first and 200704 neurons in the widest second layer, decoded back to 784 of the output.

- **Deep AE+BN** (15 (encoder) + 10 (decoder) CNN layers, ReLU activations, with BN). The last decoder layer comprises 200704 neurons as in the Shallow model for the good comparison.

According to [Vincent et al., 2008] these learn identity function and perform poorly on de-noising noisy inputs[12].

Results As opposed to the Shallow model (at the bottom of the Figure), Fig.4.5 demonstrates that the wide and "Deep $15 + 10$ CNN layers AE+BN" ReLU auto-encoder trained without noise injection, converged to a solution robust to input noise[13] supporting the implicit LiMoD regularization effect caused by over-parametrization combined with a large rank of gradient noise covariance Σ due to the use of mini-batch SGD and batch normalization as predicted by this thesis.

[12] This behavior is recovered in AE experiments without BN or with vanilla SGD on models with *sine* activations presented in Supplementary Material of Paper A

[13] The noise in test samples is additive Gaussian $N(0, 0.3)$

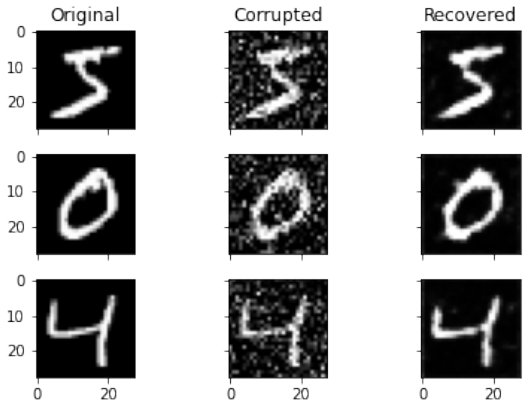


Figure 4.5: ***Deep AE+BN*** *15 (encoder) + 10 (decoder) CNN ReLU layers of wide "<>" (as opposed to bottleneck "><") architecture (*200704 *neurons in the widest layer, compared to* 784 *of the input and output) and trained with batch normalization de-noising identity map that is* ***robust to noise****, while the model was trained only on clean images: For clean images it operates as the identity, while on noisy inputs (middle column) it recovers the clean original, cf., 'Recovered', 'Original' columns.*

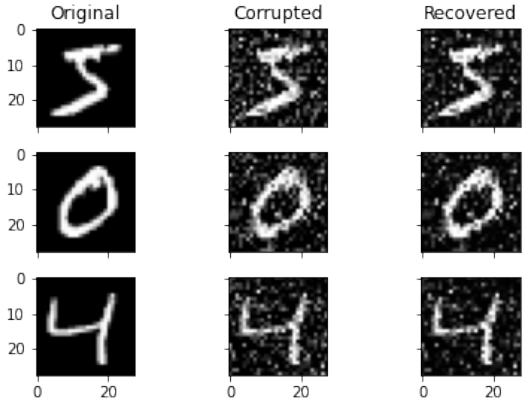


Figure 4.6: ***Shallow AE+BN****, 1 (encoder) + 2 (decoder) CNN ReLU layers of wide "<>" architecture with* 200704 *neurons in the widest layer as above. Despite being trained with batch normalization result is not robust against the input noise, compared to its deep variant shown in Fig.4.5 above.*

Further, Figure 4.7 shows an evolution of the train and test (reconstruction) error of two models during training. It supports the claim that noise (SGD + BN) combined with the over-parametrization (deep and wide <> architecture 25

layers with over $2 \cdot 10^5$ neurons in the middle) on MNIST "protect" deep model (blue) from over-fitting as opposed to the shallow model (red).

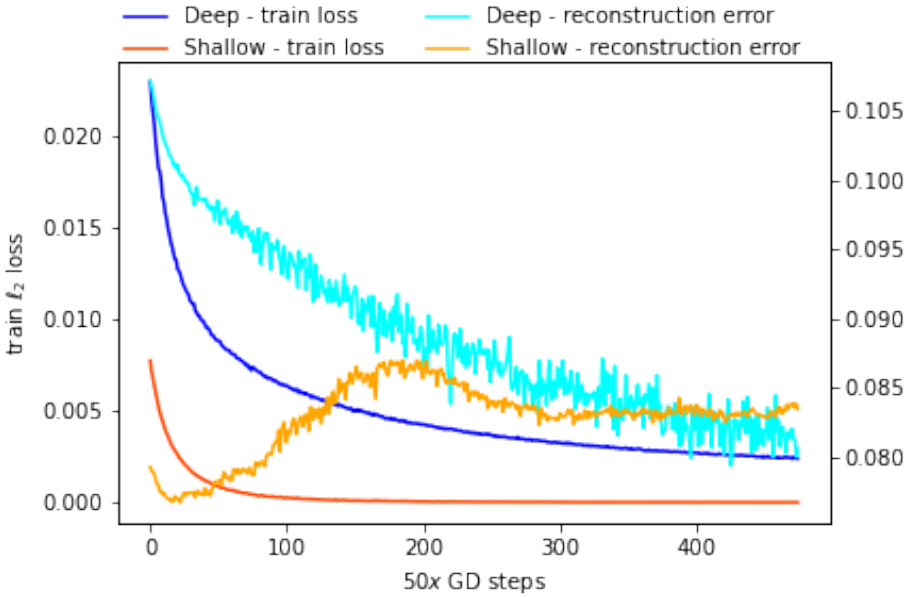


Figure 4.7: ***Shallow AE+BN****, 1 (encoder) + 2 (decoder) vs.* ***Deep AE+BN*** *15 (encoder) + 10 (decoder) CNN ReLU models from Figure 4.6 and evolution of their Mean Squared Error (MSE) on training (left axis) and test samples (right axis). It shows that training error (red) of the shallow model dropped close to zero already after* 5000 *steps (having learnt an identity map) while (blue) training error of the Deep model never reached such levels due to anti-overfitting prior imposed by batch normalization and over-parametrization (both wide and deep model). Eventually the deep model outperformed shallow one (after* $\approx$ 20000 *SGD updates) and test (called "reconstruction") error (light blue) continued dropping steadily further while (yellow) shallow model continued leveling out.*

Chapter 5

Discussion and Conclusion

. . . 'If there's no meaning in it,' said the King, 'that saves a world of trouble, you know, as we needn't try to find any. And yet I don't know,' he went on, spreading out the verses on his knee, and looking at them with one eye; 'I seem to see some meaning in them, after all. . . .

King reflecting in court, Alice's Adventures in Wonderland, Charles Ludtwidge Dodgson (Lewis Carroll, 1865)

Having praised the over-parametrization in combination with noise as an implicit anti-overfitting regularizer in the previous section there is apparently an elephant in the room to be addressed.

5.0.1 The Elephant in the Room

As widely known the over-parameterized deep models trained by SGD are readily over-fitting, even fitting the random labels as shown in [Zhang et al., 2021, Zhang et al., 2016]. How does it go along with the results of this thesis suggesting the opposite?

It has been shown in [Zhang et al., 2016, Zhang et al., 2021] that deep nets are capable of fitting random labels reaching zero training error. As reported therein, it has been reached using Adam optimizer. Experiments on MNIST datasets during work on this thesis showed that vanilla SGD was not able to fit random labels even after an extensive amount of training epochs unless adaptive methods were used. Based on the idea of RProp [Hinton et al., 2012a], both, RMSprop and Adam, are based on normalizing gradients by moving averages of their norms, see [Goodfellow et al., 2016].

RMSprop,[Hinton et al., 2012a], uses an exponentially weighted moving average of the norm of the gradient to normalize and Adam, [Kingma and Ba, 2014], combines it with momentum, again weighted moving average of gradients. Overall they are all using the adaptive normalization of the gradient similar to the form shown in (5.1).

The recent work [Zou et al., 2021], alongside others, shows that Adam may find worse solutions that vanilla SGD.

We argue this is a consequence of normalizing gradients and thus effectively canceling the "anti-overfitting" priors arising from minimization of $D_K L(g||\nu_z)$ in Corollary 3.16. In other words adaptive work against matching cumulants (3.51) and (3.52) as introduced by this thesis.

In particular, the weight update of the Rprop (*sign* of gradient, [Hinton et al., 2012a]) can also be written as standard gradient descent with the learning rate decayed by the norm of the gradient

$$W_t - \frac{\eta}{\|g\|_2} g \quad \equiv \quad W_t - \frac{\tau_t}{\|\nabla \ell_t\|_2} \overrightarrow{g}_t \tag{5.1}$$

where g denotes gradient of loss w.r.t. to weights and when we adapted it to LiMoD training notation defined in 3.17. In a light of LiMoD training, the prior

on variance of the model (3.57) controlled by τ_t that is driven large by the norm of the gradient $\|\nabla \ell_t\|_2$, the Rprop (and RMSprop and Adam, as noted above) cancels the norm of the gradient from τ_t allowing for the observed over-fitting [Zhang et al., 2016].

Secondly note that (5.1) has a similar (de-regularizing) effect on matching the first cumulant (3.51) that intuitively prevents weights of the outer layers from going extreme during training.

From the weight space perspective, adaptive learning allows the model to acquire large weights minimizing likelihood while neglecting the LiMoD implicit regularizer $D_{KL}(g||\nu_z)$ and converging to max likelihood solution with zero-training error. By (5.1) the larger the norm of the gradient g the smaller the effective learning rate $\frac{\eta}{\|g\|_2}$ of adaptive methods is. So once in the area of large gradients (fitting maximum likelihood and neglecting $D_{KL}(g||\nu_z)$) it stays in the over-fitting regime. The resulting model has very likely a signature of a poorly generalizing model.

Having said above it does not mean that adaptive methods overfit necessarily. On contrary, with a suitable setup of hyperparameters and with *explicit* regularizers they may be by far the fastest method to achieve excellent results, [Kingma and Ba, 2014]. This is rather to point out that they are not implicitly regularized.

Through the prism of the LiMoD perspective, the very discussed capability of deep models to fit random labels is in support of the results of this thesis. Moreover, the works [Zhang et al., 2016, Zhang et al., 2021] provide valuable empirical evidence for the conclusions of the thesis.

5.0.2 On Weights and Diffusion to Irrelevant Directions

Following on Section 4.0.3.1, the norm of the gradient is not the only implicit regularizer acting during the back-prop training. In the LiMoD training, defined in 3.17, **matching the second cumulants** (3.52), leading to the variance prior from the previous section that prefers large gradient norms, **is always accompanied by matching the first** and higher cumulants depending on the chosen loss function, or rather Ψ from which it derives. The higher cumulants impose skewness and other priors in general. Further research is needed and is left for future work. The first **cumulants** (3.51) matching prevent the outputs of f to be "large", as elaborated in Section 3.0.3.1.

Notably, this does not mean that all weights have to be small to generalize well. In fact, large weights of "irrelevant features" [Bengio et al., 2021], i.e.,

those that do not correlate with changes of target variables and thus do not influence the outputs of f if network architecture allows, e.g., linear model vs. shallow non-linear vs. deep non-linear model, etc. In a sense, it suggests a diffusion to irrelevant directions is not only allowed but may be helpful in terms of generalization. Further research along these lines is needed, encouraged, and is left for future work.

Chapter 6

Mechanisms that support generalization in deep learning (Paper A)

Mechanisms that support generalization in deep learning

Anonymous Authors[1]

Abstract

A generalized Pythagorean theorem for the output layer geometry is used to explore how back-prop projects network layer representations into a low-dimensional manifold, while non-linearity, further amplified by the depth of the network, stratifies the model and updates its feature representations locally to assist generalization. This is in contrast to the more global representations of linear or shallow models. Such stratification allows the deep model to flatten in locally irrelevant (noisy) feature directions enabling a globally robust model. Batch-normalization, dropout or (smaller) batch sizes are further enabling mechanisms. Our evidence includes experimental results with auto-encoders as well as with supervised models on various feed-forward and convolutional architectures.

1. Introduction

The generalization ability that has led to massive successes of deep neural networks (DNN) in many applications is still poorly understood (Zhang et al., 2021; 2016).

In this work we will address factors that enable supervised or semi-supervised model to learn and generalize well. The motivating example based on autoencoders (AEs) is presented in Fig.1, where the depth of the model combined with batch-normalization (BN) leads to a model that can de-noise images *without being trained* for it. We address the problem by taking a geometrical perspective on the layers of a DNN model. In particular we consider the layers as transformations of a data manifold acting as parametrizations of the output layer (Hauser, 2018), and analyze their behaviour during (stochastic) gradient descent training. Under this view the forward and backward passes of back-prop (LeCun et al., 1988), update the configurations of the layer activations, i.e., update the parameters of the output layer.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

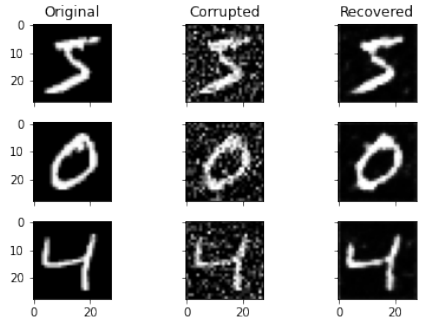


Figure 1: Deep AE+BN *15 (encoder) + 10 (decoder) CNN ReLU layers of wide "<>" (as opposed to bottleneck "><") architecture (*200704 *neurons in the widest layer, compared to* 784 *of the input and output) and trained with batch normalization de-noising identity map that is* ***robust to noise****, while the model was trained only on clean images: For clean images it operates as the identity, while on noisy inputs (middle column) it recovers the clean original, cf., 'Recovered', 'Original' columns.*

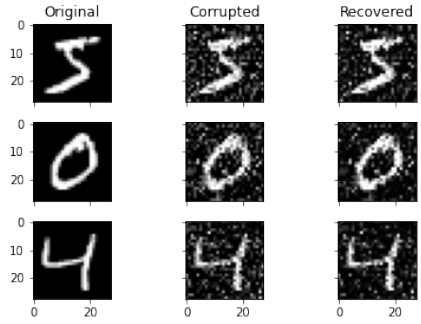


Figure 2: Shallow AE+BN*, 1 (encoder) + 2 (decoder) CNN ReLU layers of wide "<>" architecture with* 200704 *neurons in the widest layer as above. Despite being trained with batch normalization result is not robust against the input noise, compared to its deep variant shown in Fig.1 above.*

Let's start by stating what we believe is a natural but essential assumption on target distributions implicitly present in

both supervised and semi-supervised deep learning problems.

Implicit Target Assumption (ITA): *The target variables, denoted* $\{\boldsymbol{y}_s\}$ *and indexed by training data index* s *are subject to some general constraints* $g(\boldsymbol{y}_s) = 0$*, such that the interior of the graph of* $g(\boldsymbol{y}) = 0$ *defines a manifold, which is compact and smooth*[1]. This could be a simplex equation for binary classification model, $P(y_s = 1) + P(y_s = 0) = 1\ \forall s$, or could capture functional constraints on moments of target distribution, as exemplified by a Gaussian process (Rasmussen, 2003), or $\boldsymbol{y}_s$'s being close to a low dimensional manifold, in regression problems.

The (ITA) is relevant when the targets encode some (learnable) pattern. We will argue that this is the case for the combination of implicit targets distribution traits, such as fixed variance (second moment) around mean curve in regression for instance, and the explicit choice of loss (probability model), such as mean square error (MSE), cross-entropy, or log-likelihood. We note that this view, which focuses on targets, differs from a low-dimensional data manifold assumption, often assumed in representation learning, (LeCun et al., 2015), based on all data (not only targets).

Motivating Example Let us consider the example of a binary classifier and, for simplicity assume a unit mini-batch. The two classes represent a model parameterized by probability of success of respective class, p_0 and p_1. A pattern to learn is represented by making the two probabilities dependent, i.e., to constrain them to form a simplex $p_0 + p_1 = 1$. Formally we just defined a one-dimensional Bernoulli model by constraining a two dimensional Binomial model by the softmax representation, see e.g., (Goodfellow et al., 2016). The corresponding (ITA) for labels encoded as 0 and 1 is $g(\boldsymbol{y}_s) = P(y = 0) + P(y = 1) - 1 = 0$, where P is sample mean of class indicators $I[y_s = 0]$ and $I[y_s = 1]$ over batch. It defines a one dimensional curve on the two dimensional Binomial distribution (exponential family) probabilistic manifold incurred by the cross-entropy loss (a Bregman divergence, see later).

Accuracy and Flattening Back-prop training minimizes the loss, i.e. divergence along $\overrightarrow{\Delta}$ in Fig.3. By a generalized Pythagorean Theorem this divergence can be decomposed into two orthogonal components. Namely "accuracy" component along $\overrightarrow{\Delta_a}$, defined by constraint $g(y) = 0$, and "normal" component along $\overrightarrow{\Delta_n}$, in analogy to Pythagoras theorem in Euclidean space. Back-prop minimizes along $\overrightarrow{\Delta_a}$ increasing accuracy and along $\overrightarrow{\Delta_n}$, projecting outputs into the low dimensional manifold, $g(y) = 0$. Because $\overrightarrow{\Delta_n}$ points in the "irrelevant" direction not contributing to accuracy within target manifold (at P), this projection trains the model to become robust to input signals by sending outputs in irrelevant directions. We further argue that increasing

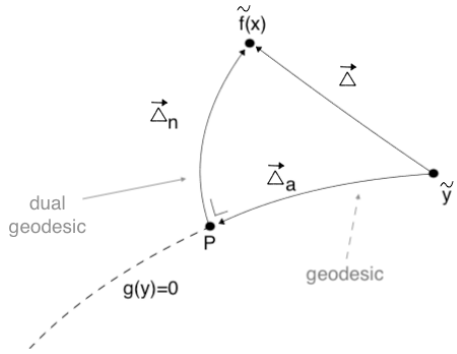


Figure 3: *Generalized Pythagorean Theorem, see (Amari, 2016), Theorem 1.2. Loss* ℓ *along* $\overrightarrow{\Delta}$ *and defined as a divergence of targets* $\tilde{\boldsymbol{y}}$ *and prediction* $\tilde{f}(x)$ *is a sum of divergences along* $\overrightarrow{\Delta_a}$ *and* $\overrightarrow{\Delta_n}$*.*

such robustness is linked to activations in the network. As shown already in (LeCun et al., 1991), the activations (and their sample moments) have an impact on convergence and generalization of the model. In the pursuit of better optimization and convergence properties batch normalization (BN) was introduced to address inherent distributional shifts in activations. The original idea was based on the observation that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change, the so-called *internal covariance shift* (ICS) (Ioffe & Szegedy, 2015). This was claimed to slow down the training by requiring lower learning rates and careful parameter initialization, making it hard to train models with saturating non-linearitiesl like ReLU's. To address this 'negative' effect of ICS the *BatchNorm* layer, normalizing activations of hidden layers using the first and the second moments (mean and variance) of the current batch, was developed.

The practical success of BN is indisputable. It improves speed and stability of DNN training. By now, it is used by default in most deep learning models. A full theoretical understanding of why BN works is lacking behind its empirical success, however, cf., Section 2.

Contributions

We unify supervised and semi-supervised back-prop training and reformulate it as an intuitive geometric procedure that flattens and bends curves. Evidence that these two mechanisms support generalization is given in a form of analysis and experiments throughout the paper. We formulate and present evidence for the following:

[1] see (Carmo & Flaherty, 1992) and Supplementary Material for definition

Hypothesis: *The generalization of an unregularized neural network is contingent on training batches having divergence component along $\overrightarrow{\Delta_a}$ much smaller than along $\overrightarrow{\Delta_n}$, with these directions defined in Fig.3. In addition, by random initialization combined with depth and batch normalization (BN) convergence is more likely. The analysis points to alternatives to BN, such as noise in layers (incl. inputs) or smaller batch-to-learning rate ratio.*

2. Further Related Work

The convergence of learning and generalization in DNNs has been a lively research topic over more than two decades. The case of SGD dynamics in multi-layer linear networks was analyzed by (Glorot & Bengio, 2010) and (Kawaguchi, 2016) showed that every local minimum is global. There is progress on understanding convergence to global optima in general (Zhu et al., 2019), for two layer ReLU networks in (Du et al., 2018) and ResNet's in (Du et al., 2019). Recent work argue for implicit bias of SGD towards well generalizing solutions (Huh et al., 2021). Also a line of work shows that "bad" local minima exist and SGD may reach them supporting the notion that current understanding of the generalization in DNNs still requires a revision, (Zhang et al., 2016; Liu et al., 2019; Zhang et al., 2021).

It is well established that normalization, i.e., an affine transform of the inputs to produce zero means and unit variance random variables, accelerates convergence (LeCun et al., 2012). Since then many other normalization methods have been proposed. Batch normalization (Ioffe & Szegedy, 2015) is an algorithmic method which makes the training of DNN faster and more stable. It consists of normalizing activation vectors from hidden layers using the first and the second statistical moments (mean and variance) of the current batch. This normalization step is applied right before (or right after) the nonlinear function. BN has been shown to provide several training merits. These include prevention of exploding or vanishing gradients, robustness to different settings of hyper-parameters such as learning rate and initialization scheme, and keeping most of the activations away from saturation regions of non-linearities.

Work in (Kohler et al., 2018) based on Gaussian input assumptions showed that BN reduces the cross-dependency between layers in DNN. Due to this dependency reduction a gradient-based optimization with an adaptive step-size can enjoy a linear convergence rate. While BN fits well in feed forward architectures it more troublesome to apply in recurrent networks. Other caveats include a dependence on mini-batch size and deterioration if test set sample distribution differs from training set (see also below).

While all above mentioned techniques relate benefits of normalization techniques to optimization, generalization merits drawn significantly less attention. The original work (Ioffe & Szegedy, 2015) conjectured that BN implicitly regularizes training to prevent over-fitting. (Zhang et al., 2016) and later in (Zhang et al., 2021) presents BN as an implicit regularizer drawing on experimental evidence. Injecting random noise in the input layer is known to be equivalent to regularization by weight decay (L2 regularization), as shown in (Bishop, 1995) and generalized to DNNs or autoencoders (AE), (Poole et al., 2014). Dropout (Srivastava et al., 2014) can be seen as noise injection in DNN layers, (Poole et al., 2014).

BN was empirically shown to improve generalization in DNN (Luo et al., 2018; Santurkar et al., 2018; Ba et al., 2016). Yet, theoretical understanding is still in progress. The original work (Ioffe & Szegedy, 2015) hypothesized that BN effectiveness was mostly due to ICS reduction a notion challenged by (Santurkar et al., 2018) using counter examples and rather stresses the smoothing effect of BN on the optimization landscape. Specifically it is demonstrated that there does not seem to be any specific link between the convergence gain of BN and the reduction of ICS. In fact, it is shown that in a certain sense BN might not even reduce ICS. Here we follow to argue that it is not *reduction* of ICS, but rather the pervasive *presence* of ICS throughout the training process that assists generalization.

3. Formulation

For the sake of brevity, definitions of *test* and *generalization error* as well as definition of *smooth manifold* is given in Supplementary Material. Further, this paper considers Bregman divergence loss functions (Amari, 2016; Banerjee et al., 2005) defined as $\ell(\boldsymbol{z}, \boldsymbol{y}) = d_\Phi(\boldsymbol{z}, \boldsymbol{y}) = \Phi(\boldsymbol{z}) - \Phi(\boldsymbol{y}) - \langle \boldsymbol{z} - \boldsymbol{y}, \nabla_y \Phi(\boldsymbol{y}) \rangle$, where $\Phi : \mathbb{R}^d \to \mathbb{R}$ is a strictly convex function. This choice allows us to derive general results for a wide range of losses, including classification and prediction problems.

An important property of the Bregman divergence is that its derivative w.r.t the first argument at datum $(\boldsymbol{x}_s, \boldsymbol{y}_s)$ evaluates as

$$\nabla_x d_\Phi(x, y) = \nabla\Phi(\boldsymbol{x}) - \nabla\Phi(\boldsymbol{y}) \tag{1}$$

Further it can be shown that there exist an isomorphic dual space such that $d_\Phi(\boldsymbol{z}, \boldsymbol{y}) = d_\Psi(\nabla\Phi(\boldsymbol{y}), \nabla\Phi(\boldsymbol{z}))$, where Ψ is a convex conjugate to Φ. For more details see (Hiriart-Urruty & Lemaréchal, 2012).

Batch Normalization In the following, a notation established in (Ba et al., 2016) is used for all three normalization methods analyzed.Consider the l^{th} hidden layer in a deep feed-forward neural network, and let a^l be the vector representation of the summed inputs to the neurons in that layer. The summed inputs are computed through a linear projection with the weight matrix W^l and the bottom-up inputs

z^l given as $a_i^l = W^{(l)T}[:,i]z^l, z_i^{l+1} = q(a_i^l + b_i^l)$, where $q(\cdot)$ is an element-wise non-linear function and $W^{(l)T}[:,i]$ is the i^{th} column of the weight matrix (vector of incoming weights to the i^{th} hidden units and b_i^l is the scalar bias parameter. The parameters in the neural network are learned using gradient-based optimization algorithms, with the gradients being computed by back-propagation.

BN normalizes the summed inputs to each hidden unit z_i^l of the layer l over the training cases. Specifically, for the i^{th} summed input in the l^{th} layer, it rescales the summed inputs according to their variances under the distribution of the data $\bar{a}_i^l = \frac{g_i}{\sigma_i}(a_i^l - \mu_i^l), \mu_i^l = E[\bar{a}^l], \sigma_i^l = \sqrt{E[(a_i^l - \mu_i^l)^2]}$, where $\bar{a}_i^l$ is normalized sum of the inputs to the i^{th} hidden unit in the l^{th} layer and g_i is a gain parameter scaling the normalized activation before the non-linear activation function. The expectation is over whole training data distribution and it is is replaced by sample estimates from current batches in practice.

BN normalizes the summed inputs a_i to a neuron through the two scalars μ and σ. They also learn an adaptive bias b and gain g for each neuron after the normalization $z_i = q\left(\frac{g_i}{\sigma_i}(a_i - \mu_i) + b_i\right)$, where $q(\cdot)$ is a non-linearity used for the hidden unit i and where we left out upper index of the layer l hidden unit i belongs to for brevity.

Learning in the Manifold of Distributions Our ambition is to relate the variations in the activations z_i^l in the layers, representing features, to changes of the loss function. We use the concept of Information Geometry(Amari, 2016) combined with Riemannian geometry of the neural networks derived in (Hauser, 2018) and introduce Learning in the Manifold of Distributions (LiMoD).

In the LiMoD setting, following (Hauser, 2018), layers and their activations are representations of the (latent) data manifold[2] and, at the same time, they represent parameterizations of the output layer Exponential family. As opposed to a more common view o deep learning as a weight optimization, the LiMoD presents it as an optimization over the activations (parameters of the model) rather than weights.

Let a neural network $f : \mathbb{R}^b \times \Upsilon \to \mathbb{R}^d$ of L layers be defined as the composition:

$$f(\boldsymbol{x}, \vec{\boldsymbol{w}}) = \varphi_L(\boldsymbol{W}^{(L)}) \circ \varphi_{L-1}(\boldsymbol{W}^{(L-1)}) \circ \ldots \circ \varphi_1(\boldsymbol{W}^{(1)})(\boldsymbol{x}) \quad (2)$$

where each vector function $\varphi_l(\boldsymbol{W}^{(l)})(\boldsymbol{v}) = a^l\left(\boldsymbol{W}^{(l)}\boldsymbol{v}\right)$ is an activation function a^l applied onto a result of matrix $\boldsymbol{W}^{(l)}$ and vector $\boldsymbol{v}$ product. Let H^l denote number of hidden units of the layer l.

In these settings the output layer is a parametric manifold of the Exponential family probability distributions over the target/label variables induced by the choice of the convex function Φ, e.g., $\Phi := \frac{1}{2}\langle \boldsymbol{x}, \boldsymbol{x} \rangle$ for square loss and $\Phi(\boldsymbol{x}) := \sum_{j=1}^{d} x_j \log_2 x_j$ s.t. $\sum_{j=1}^{d} x_j = 1$ for KL divergence, which covers negative log likelihood models and cross-entropy loss for classification tasks, cf. (Banerjee et al., 2005; Hiriart-Urruty & Lemaréchal, 2012). Given index of the data sample s, the output of the neural network model $f(\boldsymbol{x}_s, \boldsymbol{w})$ defines a natural parameterization of the probability model over targets $\tilde{\boldsymbol{y}} := \nabla\Phi(\boldsymbol{y}_s)$ and network f is further parameterized by weights $\boldsymbol{w}$. Tilde notation for a gradient map $\nabla\Phi$ is used throughout the paper, including the Introduction and Fig.3.

By the construction above, the strictly convex function Φ gives rise to Bregman divergence and, through a bijection between Bregman divergences and regular Exponential families, cf. (Banerjee et al., 2005), it endows the output layer as a probabilistic manifold with every point representing a distribution from this Exponential family[3]. According to (Amari, 2016), Chapter 2, by this construction, the output layer is a (dually flat) Riemannian manifold with Fisher Information Matrix (FIM) as a metric. In fact, in the case of Bregmann divergence loss, the Fisher Information Matrix is given by the Hessian of ψ (convex dual of Φ)

$$FIM(\theta_s)\,|_{\theta_s = f(\boldsymbol{x}_s, \boldsymbol{w})} = \nabla\nabla\psi := \left\{\partial_\kappa \partial_\lambda \psi(\theta_s)\right\}_{\kappa,\lambda} \quad (3)$$

where κ, λ are indexes of the output layer coordinate system corresponding to natural parameters of the related Exponential Family.

3.1. Back-prop, Straight and Curved

As the previous section outlined, activations z^l represent inputs, transformed by a composition of previous layers. Let's call them features, slightly deviating from the usual use of this word (as independent input variables). In this paper, features are real values of activations. Further consider the i^{th} feature of the layer l evaluated at data point $\boldsymbol{x}_s$. Let's denote such real number $z_i^l(\boldsymbol{x}_s) \in \mathbb{R}$.

In the network (2), for a given layer l and its i-th activation, the layer by layer composition defines a map from layer l to the output layer projecting i^{th} feature as $\Phi_{l_i}(t) : t \to \varphi_L(\boldsymbol{W}^{(L)}) \circ \varphi_{L-1}(\boldsymbol{W}^{(L-1)}) \circ \cdots \circ \varphi_l(\boldsymbol{W}^{(l)})(\boldsymbol{z}^l + t\boldsymbol{e}_i)$ where $t \in \mathbb{R}$ and $\boldsymbol{e}_i \in \mathbb{R}^{H^l}$ is unit vector with $H^l - 1$ zero elements and 1 on position i.

Assuming necessary smoothness of f, $\Phi_{l_i}(t)$ defines a parameterized curve on the output layer, see. (Carmo & Flaherty, 1992).

[2] or rather its immersions/submersions into layer manifolds

[3] with a cumulant function given by convex dual of Φ, denoted ψ, see Preliminaries section

Tangent vector ("speed") of general curve $\gamma(t)$ at t is given by $\gamma'(t)$, where ' denotes a derivative with respect to a real parameter. Similarly, fixing the layer l and index of a hidden unit i, by taking derivative of $\Phi_{l_i}(t)$ w.r.t. a_i we get $q'\frac{g_i}{\sigma_i}$ and by applying chain rule, we obtain the vector field along the curve $\Phi_{l_i}(t)$ of the dimension of the output layer $dim(\boldsymbol{y}_s)$ whose j^{th} element evaluates as

$$\Phi'_{l_i}(t)[j] = \sum_{p\in\mathcal{BP}_j^{l_i}} W(p)\prod_{k=l}^{L} Q(t,k)\frac{g^k_{p(k)}}{\sigma^l_{p(k)}} \tag{4}$$

where $\mathcal{BP}_j^{l_i}$ is a set of all paths connecting hidden unit i of the layer l to an output layer node j through the network f such that each layer k along the path p has exactly one node present in the path, denoted $p(k)$. Then $W(p)$ denotes the *product* of weights along the path p and $Q(t,k)$ is a product of activation derivatives along the path $p(k)$.

Back-propagated gradient of the weight w_m^{l-1} leading to node i of the layer l, considering unit mini-batch for sake of simplicity, is written as

$$\frac{\partial \ell}{\partial w_m^{l-1}} = \frac{\partial \ell(\theta)}{\partial \theta}\bigg|_{\theta=f(\boldsymbol{x}_s,\boldsymbol{w})} \frac{d\Phi_{l_i}(t)}{dt}\bigg|_{t=z_i^l(\boldsymbol{x}_s)} q'(a_m^{l-1})z_m^{l-1} \tag{5}$$

using the chain rule, also see (Bishop, 2006). For Bregmann divergence loss ℓ the first factor of RHS evaluates element-wise as

$$\begin{aligned}\Delta_j(s) :=& \frac{\partial \ell(\theta)}{\partial \theta_j}\bigg|_{\theta=f(\boldsymbol{x}_s,\boldsymbol{w})} \\ =& \nabla\Phi(f(\boldsymbol{x}_s,\boldsymbol{w})[j]) - \nabla\Phi(\boldsymbol{y}_s[j])\end{aligned} \tag{6}$$

where we used (1) applied element-wise at the point $\theta \in R^{dim(\boldsymbol{y})}$ of the output layer. Note that Φ, that defined loss in (6), denotes a convex dual function to Ψ and is essentially different from curve Φ denoted in italic. For a square loss, Δ_j from (6) is a vector of "errors", whose j^{th} element is a squared error between prediction $f(x,w)[j]$ and target $y[j]$.

Note, that back-propagation (5) includes a direction of forward-propagated feature z_i^l evaluated at the end of interval $(0, z_i^l)$ (corresponding to data sample $\boldsymbol{x}_s$), i.e. namely $\frac{d\Phi_{l_i}(t)}{dt}\Big|_{t=z_i^l(\boldsymbol{x}_s)}$ shortly denoted as $\Phi'_{l_i}(z_i^l(\boldsymbol{x}_s))$.

Following (5) and assuming non-saturated state $q'(a_m^{l-1}) \neq 0$ a backward pass minimizes a standard ℓ_2-inner product, projecting the "error" vector Δ onto vector $\Phi'_{l_i}(z_i^l(\boldsymbol{x}_r))$. As such it is a function of a tangent $\Phi'_{l_i}(z_i^l(\boldsymbol{x}_r))$.

Assume value of feature $z_i^l(\cdot)$ at four close (in layer representation) input data samples with indices r, t, m, n (points R, T, M, N in Fig.4 respectively). Back-prop, evaluated on current data sample $\boldsymbol{x}_s$ (green cross), expands $\Phi(t)$ along $-\vec{\Delta}$ by updating incoming weights (5).

By this scaling, see Fig. 4 depicting two back-prop updates, RN segment, before insensitive to errors along $\vec{\Delta}$ has been split and MN (red) part of has become partially sensitive to this direction.

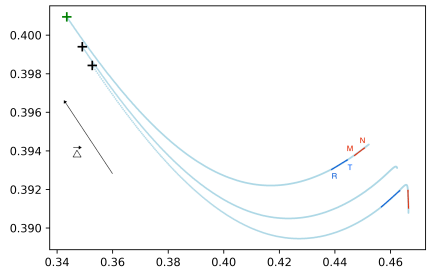


Figure 4: *Parameterized features curves $\Phi(t), t \in (0, 0.5)$ (light blue) of the "two moons" model (see Experiments sec.). Presented is feature of the 3^{rd} layer, evaluated at the same data sample ($\boldsymbol{x}_s$) over two back-prop steps (black crosses) at the early stage of the training. An image of the zero feature, $\Phi(0)$ (green cross), denotes starting point, (blue) arrow shows the $\vec{\Delta}_s$ direction. Besides translation and due to an expansion in $-\vec{\Delta}_s$ direction length of the $\Phi(t)$ curve on interval $(0, 0.5)$ gets longer over steps, following the underlying curvature of $\Phi(t)$. Neighbours RT (blue segment) are kept close and unaffected over updates, because they lie on segment orthogonal to $\vec{\Delta}_s$, while features in MN (red segment) have become sensitive to this direction.*

Overall, back-prop can be viewed as procedure updating curves (representations of features) in such a way that only parts linearly aligned with $\vec{\Delta}$ get updated, i.e., scaled, by updating weights. Further curvature of $\Phi_{l_i}(t)$, if present, causes different parts of $z_i^l(\cdot)$'s feature range being responsive to different directions, localizing the back-prop for future updates.

$\Phi(t)$ is curved in general, whose curvature depends on weights along the path to the output layer as well as on other features of layer l through $Q(t,k)$ and $W(p)$ in (5) respectively. It is affected by their updates throughout the training. Nevertheless, back-prop proceeds asynchronously and updates a particular weight having all other weights and features fixed. That means curvature of Φ_{l_i} for one update is fixed.

Further, by smoothness of f there exist a $\boldsymbol{z}^l$-neighborhood of feature $\boldsymbol{z}^l(\boldsymbol{x}_s)$ on which the 4 is constant vector field $\Phi'_{l_i}(t)$. If this vector field is updates so it becomes non-constant on this $\boldsymbol{z}^l$-neighborhood then it either does not

affect $\Phi'_{l_i}(t)$ on other $\boldsymbol{z}^l$-neighborhoods, e.g., features of other training data samples, or it stratifies them too. In either case curvature enables localization of back-prop.

3.2. Small Accuracy Divergence Enables Robustness

Another ingredient contributing to better generalization is the "flattening" phenomenon of the layer representations during back-prop training outlined in the Introduction. It is experimentally demonstrated in Fig.5, (Hauser, 2018) and additional experiments in Supplementary Material and further references.

(ITA) constraints $g(\boldsymbol{y}) = 0$ define, generally curved, sub-manifold of the probabilistic manifold. While the example in the Introduction presents binary classification model with $g(y) = 0$, being simplex line (geodesic on natural parameter space) of related Binomial distribution, in the general case of curved $g(y) = 0$, such as regression problems, general Pythagorian Theorem applies *locally* on geodesic neighborhoods of targets $\{\boldsymbol{y}_s\}$ in training data.

Thanks to (assumed) smoothness of probability manifold and (ITA) by definition of the manifold, there exists (infitesimally) small $\boldsymbol{y}_s$-neighborhood of $\boldsymbol{y}_s$ such that tangent space of $g(\boldsymbol{y}) = 0$ at given $\boldsymbol{y}_s$ approximate (by local isomorphism from definition[4]) manifold to arbitrary precision., for all s. Locally any geodesic segment in $\boldsymbol{y}_s$-neighborhood forms the base of the geodesic triangle of Fig.3 onto which $\overrightarrow{\Delta}_n$ propagates feature curves. Neighborhoods $\boldsymbol{y}_s$-neighborhood indexed by s have different directions of normal components due to curvature of $g(\boldsymbol{y}_s) = 0$, however.

On $\boldsymbol{y}_s$-neighborhoods back-prop makes feature curves insensitive to fluctuations of features propagated orthogonal to $g(\boldsymbol{y}_s) = 0$ by the mechanism of Fig.3 and therefore also on their union, that is global.

Never the less, back-prop updates are in a direction of a $\overrightarrow{\Delta}$ and it coincides with $\overrightarrow{\Delta}_n$ only if divergence along $\overrightarrow{\Delta}_a$ is small compared to divergence in $\overrightarrow{\Delta}_n$ direction. Thus robustness can only be built on those $\boldsymbol{y}_s$-neighborhoods that have larger "normal" divergence components than "accuracy". Summarizing the previous arguments we arrive at:

Postulate 1 (*Informal*)**.** *Let $d_\Phi(f(\boldsymbol{x}_s), \boldsymbol{y}_s)$ denote Bregman divergence loss ℓ (3). Then, following notation of Fig.3, a unit batch size back-prop (5) on training datum s updates weights in direction of $\overrightarrow{\Delta}_n$, building robustness, only if $d_\Phi(\boldsymbol{y}_s, \nabla\Psi(P)) << d_\Phi(\nabla\Psi(P), f(\boldsymbol{x}_s))$.*

Depth and Random Initialization One straightforward argument for depth supporting generalization is to link the depth to higher accuracy, (Goodfellow et al., 2016), and then apply Postulate 1.

Further on a geometric view built up in this paper reveals a link between robustness and random initialization we present in what follows. We have shown before that higher curvature of the feature curves $\Phi(z_{l_i})$ during training, reported in Fig.5, means an interval of activation values z_{l_i} in split into larger number of sub-intervals each of which responds to a different direction of gradient of loss $\overrightarrow{\Delta}$, determined by inner product $\Phi'(z_{l_i})\overrightarrow{\Delta}$ in back-prop (5).

Following Proposition 1 proven in Supplementary Material states that the deeper the network the higher probability of curved $\Phi(z_{l_i})$'s there is.

Proposition 1. *Consider random weight initialization producing i.i.d. distribution of layer activations, conditionally on inputs, with finite moments*[5]*. Then in randomly initialized deep neural network (2) a probability of a curved $\Phi(z_{l_i})$ grows linearly with depth L.*

Assuming $\|\cdot\|_2$ of these sub-intervals were distributed randomly with finite first and second moments (as by popular random weight initialization, (Glorot & Bengio, 2010) or He (He et al., 2015) initialization) by smoothness of $\Phi(t)$ it follows that the larger number of such sub-intervals produces smaller[6] linear neighbourhoods in the output layer, these sub-intervals are mapped on through $\Phi(z_{l_i})$. In the extreme case, each neighborhood contains at max one training datum output $f(\boldsymbol{x}_s)$ and gets highly specialized to fit the loss at this particular point, bringing the model to a state where divergence along $\overrightarrow{\Delta}_a$ is close to zero. Not necessarily zero training error (loss ℓ), because minimization of divergence along $\overrightarrow{\Delta}_n$ may have been hindered by large $\overrightarrow{\Delta}_a$, see Postulate 1 and corroborating experiments in Supplementary Material.

Proposition 1 states that the probability of reaching this training point increases with the depth of the model. If there are no updates along $\overrightarrow{\Delta}_n$, such a solution, even with zero training error, has features with arbitrary tangents $\Phi'(t)$ pointing also normal direction $\overrightarrow{\Delta}_n$ and thus arbitrarily low robustness in general.

Learning Robustness

Following up on previous section, a starting point for this section is condition of Postulate 1, $d_\Phi(\boldsymbol{y}_s, \nabla\Psi(P)) << d_\Phi(\nabla\Psi(P), f(\boldsymbol{x}_s))$. Further let's analyze settings that prevent/support minimizing divergence along $\overrightarrow{\Delta}_n$.

Batch size In case of non-unit mini-batch of size B with samples indexed by b the loss is averaged over batch samples, (Goodfellow et al., 2016). Assume training data to be

[4] follows by approximation of discrete set of $|\mathcal{D}|$ equations $g(\boldsymbol{y}_s) = 0$ by smooth manifold assumed by (ITA) and then by local isomorphism of manifold to Euclidean (tangent) spaces from definition of manifold, see (Tu, 2011; Carmo & Flaherty, 1992)

[5] such as common Xavier (Glorot & Bengio, 2010) or He (He et al., 2015) initialization

[6] in FIM metric of output layer

independently sampled from underlying latent data distribution. Then $\overrightarrow{\Delta}$ is average of derivatives of $d_\Phi(f(\boldsymbol{x}_s), \boldsymbol{y}_s)$ over batch. Recall $\overrightarrow{\Delta}$ and its $\overrightarrow{\Delta}_n$ and $\overrightarrow{\Delta}_a$ components are vectors living in (dually) flat probabilistic manifold, (Amari, 2016), parameterized by natural coordinates of related Exponential Family and back-prop sums them up. That means that components also average.

Notably, if samples of non-zero loss are close to each other on the same geodesic of $g(\boldsymbol{y}_b) = 0$ then the average of their normal components stays approximately orthogonal to this geodesic and does not cause an increase of loss in $\overrightarrow{\Delta}_a$ direction. The more apart they are in target manifold $g(\boldsymbol{y}_b) = 0$ the more this average affects also the "accuracy" component if it is curved.

Note that in the similar way the $\sum_b \overrightarrow{\Delta}_a/B$ averages over batch as well during training. That may cause back-prop to stay in local optima, as in Fig.5, 3^{rd} row of the panel. When batch size decreased training continues as $\sum_b \overrightarrow{\Delta}_a/B$ increased. Never the less, also in local optimum when $\sum_b \overrightarrow{\Delta}_a/B$ is small, by Postulate 1 flattening follows, if divergence along $\sum_b \overrightarrow{\Delta}_n/B$ non-zero and robustness is built.

Overall, the larger the mini-batch size B the higher chance that resulting in $\sum_b \overrightarrow{\Delta}_n/B$ average out over the batch, causing lower flattening in the irrelevant direction leading to lower robustness. Note also, that increasing the learning rate, in line with keeping the same learning rate to batch ratio helps only partially, as averaging happens first and in case it zeros out the learning rate have next to nothing effect. On the other hand, the average will never be exactly zero on all batches, assuming training with a constant learning rate, (Hansen et al., 1993) and then increasing learning rate to batch ratio may help, as confirmed in Experiments in Supplementary Material.

Batch normalization (BN) & noise in layers Training with BN only makes sense with batches of the sizes that ensure data distribution in the batch is close to a training data distribution, (Ioffe & Szegedy, 2015; Santurkar et al., 2018). In such settings averaging of $\overrightarrow{\Delta}$'s, described in the previous section affects convergence and flattening.

As opposed to vanilla SGD training, however, BN ensures all activations z_i^l (after normalization) fluctuate around zero with the second moment ≈ 1 even in the small gradients regime, such as in the case of averaging over batches. Tho imposed variations in after effect cause the feature curves $\Phi(z_{l_i})$ evaluated on batch samples propagate along the curves reviving the loss and its gradient. This behavior is reported in experiments in (Santurkar et al., 2018), showing the internal covariant shift (variations of activations) is in certain cases in fact enhanced instead of reduced by BN, and gradients norms are kept non-zero by BN.

Assuming loss along "accuracy" component has reached its local minimum (otherwise SGD escapes in next steps and training continues) the only way the back-prop decreases loss is to make features insensitive along $\overrightarrow{\Delta}_n$ as these do not affect "accuracy" component.

This causes weights to be updated so that model ignores changes of features in these "irrelevant" directions (normal to output data manifold) while maintaining "accuracy" divergence in its low (or evacuating local optima due to increase of divergence along $\sum_b \overrightarrow{\Delta}_a/B$ by flattening).

Applied on $l = 0$, i.e. the input layer, back-prop 'shapes' the network $f(\boldsymbol{x}|\boldsymbol{w})$ so that it has low sensitivity (derivative) along feature curves of the inputs training f to generalize well on inputs.

On the other hand, since divergence along $\sum_b \overrightarrow{\Delta}_a/B$ may still be significant even in the local optimum, the flattening with BN only happens till divergence along $\sum_b \overrightarrow{\Delta}_n/B$ gets on the comparable level to $\sum_b \overrightarrow{\Delta}_a/B$ due to Postulate 1. This can explain the reason behind BN solutions generalizing worse than those found by vanilla SGD and smaller batch size.

Nevertheless, BN (and other similar normalization methods, as layer normalization, (Ba et al., 2016)) enables back-prop to train extremely deep models, as ResNets, (He et al., 2020), maintaining gradient flow by normalization and preventing vanishing gradients (case with vanilla SGD in deep models).

Noise in layer activations and inputs Work (Poole et al., 2014) showed that injecting noise to layers and inputs helps generalization of auto-encoders and in deep networks, both theoretically and experimentally. Getting back to the main experiment of the paper in Fig.1 noise in the inputs is the classical way to train denoising auto-encoders (DAE), (Poole et al., 2014).

From the perspective of this paper, injecting noise into the activations of layers and inputs, including methods such as dropout, (Srivastava et al., 2014), contributes to generalization by the same mechanisms as BN in the previous section. On top, it does not require large mini-batch sizes. On the downside, we conjecture that scale of the noise is an important hyper-parameter that may prevent reaching low levels of "accuracy" divergence $\sum_b \overrightarrow{\Delta}_a/B$ if over-boosted and thus also lower robustness on data samples from such mini-batches.

4. Experiments

To demonstrate the ability of deep networks trained with BN to learn the robust solution we train the denoising auto-

encoder (DAE) (Vincent et al., 2008) model. As argued therein DAE is learning the identity function. More precisely, Denoising autoencoders (DAEs) are an extension of autoencoders trained to reconstruct a clean version of input from its corrupted version (Poole et al., 2014). The main methods used to regularize DAE are bottle-neck architecture or noise injected into inputs during training. Otherwise, DAE learns an identity map. In our experiments, we use neither of these. Architecture of convolutional layers is of a shape '<>' instead of recommended bottleneck '><', and *there is not any noise injected* into training samples. Further details in Fig.1.

According to (Vincent et al., 2008) these learn identity function and perform poorly on de-noising noisy inputs. This behavior is recovered in AE experiments without BN or with vanilla SGD on models with $sine$ activations presented in Supplementary Material.

On contrary, Fig. 1 from the Introduction, demonstrates that "Deep $15 + 10$ CNN layers AE+BN" ReLU auto-encoder trained without noise injection, converged to a solution robust to input noise[7].

Throughout the paper experiments with binary classifier models on synthetic "two moons" scikit-learn datasets, (Pedregosa et al., 2011) are reported extending experiments with AE to supervised learning. The "Two Moons" model is a feed-forward model with 2D inputs + 7 *tanh* or *relu* or *linear* (3-5-10-10-5-2-2) layers, followed by *sigmoid*. Further experiments with both deep and shallow architectures are in Supplementary Material.

Evacuation of a local optimum example enabled by curved feature curves and executed by increased learning rate to batch ratio. The network f maps all training data into a small area (single point) at the beginning, given by the general position of weights after standard random initialization, (Daniely et al., 2016; Sutskever et al., 2013), as depicted in Fig.5, top row, the first and the second image[8].

As the training progresses, following the previous section, outputs of f are propagated into a one-dimensional manifold, captured by the second row, showing outputs of f before sigmoid (leftmost) and the same effect on feature curves in the middle of the second row of Fig.4.

After the model has stuck in the local (linear model) optimum (3^{rd} row), decreasing the mini-batch size from 20 to 2 enhanced gradients[9]. On batches producing large loss, the gradients extend the scale of the layer representation so much that some of the large loss features are mapped into previously unused non-linear regions of $\Phi_{l_i}(\delta)$ map.

[7]The noise in test samples is additive Gaussian $N(0, 0.3)$

[8]zoomed in after 30 SGD updates

[9]similar effect had an increase of a learning rate $10\times$.

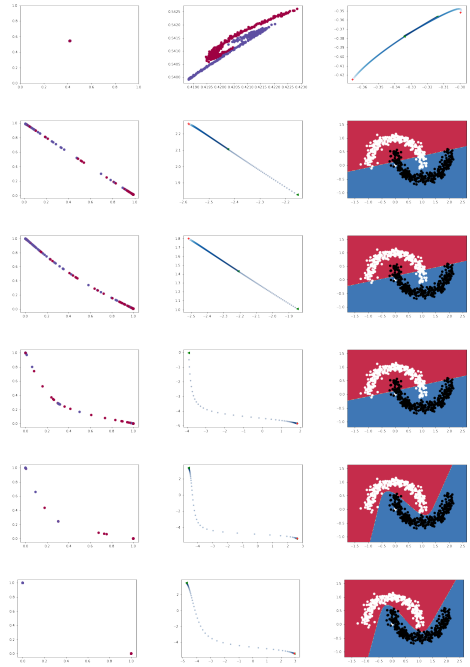


Figure 5: *"Two moons" binary classifier with 7 hidden linear layers and* $\tanh$ *activation. The top three rows showcase the first* $10,000$ *epochs of training with a mini-batch size of* 20*, the bottom three rows demonstrate the change caused by mini-batch size reduction to* 2 *for the last* $10,000$ *epochs. This change enlarged gradients expanding the arc-length of feature curves to reach its curved regions (middle column) and thus making the features at opposite ends absorb back-propagated gradients of different directions, see Fig.4. Overall, decreasing batch size enabled the model to continue training from a local optimum at epoch* $10,000$ *with test accuracy* 0.9 *(third row, right) and converge to a better one (test accuracy* 1*), last two bottom rows (epochs,* $11,500$ *and* $20,000$*).*

This has an effect of "localizing" the back-prop enabling to fit parts with large errors while keeping the other ends unchanged in line with previous arguments and Fig.4, see Fig.5, middle column.

5. Discussion and Conclusions

Proposition 1 links work on the "lottery ticket" hypothesis, (Frankle & Carbin, 2018; Malach et al., 2020), that argue the initialization determines the convergence trajectory and resulting properties of the solution. From our results and Proposition 1, it follows that initialization produces different shapes of the feature curves. Fig.5 and 4 support the theoretical argument that back-prop expands the arc-length

of the feature curves by up-scaling parts linearly aligned with gradient of loss. In effect, it exploits an underlying curvature of features shaped by initialization and following weight updates (depending again on initialization). Hence it is initialization that gives rise to different gradient flows and stratifications of features, resulting in different trajectories and generalization properties.

The phenomenon of "flattening representations" has been reported in deep learning experiments (Hauser, 2018), but to the best of authors' knowledge not yet fully understood. The geometric view presented in this paper provides a unifying view, shedding light on this fundamental phenomenon. Theoretical arguments, corroborated by experiments with AEs and classifiers are in favor of the Hypothesis stated in the Introduction, that it is depth, combined with BN (or alternatively with a adjusted learning rate to batch size ratio or noise in layers), which delivers a well generalizing solution.

References

Amari, S.-i. *Information geometry and its applications*, volume 194. Springer, 2016.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. 2016.

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749, 2005.

Bishop, C. M. Regularization and complexity control in feed-forward networks. 1995.

Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

Carmo, M. P. and Flaherty, F. J. *Riemannian geometry*. Birkhäuser, 1992. ISBN 3764334908, 0817634908, 9780817634902, 9783764334901.

Daniely, A., Frostig, R., and Singer, Y. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *Advances In Neural Information Processing Systems*, 29:2253–2261, 2016.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1675–1685. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/du19c.html.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Hansen, L. K., Pathria, R., and Salamon, P. Stochastic dynamics of supervised learning. *Journal of Physics A: Mathematical and General*, 26(1):63, 1993.

Hauser, M. B. Principles of riemannian geometry in neural networks. 2018.

He, F., Liu, T., and Tao, D. Why resnet works? residuals generalize. *IEEE transactions on neural networks and learning systems*, 31(12):5349–5362, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

Hiriart-Urruty, J.-B. and Lemaréchal, C. *Fundamentals of convex analysis*. Springer Science & Business Media, 2012.

Huh, M., Mobahi, H., Zhang, R., Cheung, B., Agrawal, P., and Isola, P. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Kawaguchi, K. Deep learning without poor local minima. *arXiv preprint arXiv:1605.07110*, 2016.

Kohler, J., Daneshmand, H., Lucchi, A., Zhou, M., Neymeyr, K., and Hofmann, T. Towards a theoretical understanding of batch normalization. *stat*, 1050:27, 2018.

LeCun, Y., Touresky, D., Hinton, G., and Sejnowski, T. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pp. 21–28, 1988.

LeCun, Y., Kanter, I., and Solla, S. A. Second order properties of error surfaces: Learning time and generalization. In *Advances in neural information processing systems*, pp. 918–924, 1991.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Liu, S., Papailiopoulos, D., and Achlioptas, D. Bad global minima exist and sgd can reach them. *arXiv preprint arXiv:1906.02613*, 2019.

Luo, P., Wang, X., Shao, W., and Peng, Z. Towards understanding regularization in batch normalization. *arXiv preprint arXiv:1809.00846*, 2018.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pp. 6682–6691. PMLR, 2020.

Needham, T. *Visual Differential Geometry and Forms: A Mathematical Drama in Five Acts*. Princeton University Press, 2021.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Poole, B., Sohl-Dickstein, J., and Ganguli, S. Analyzing noise in autoencoders and deep networks. *arXiv preprint arXiv:1406.1831*, 2014.

Rasmussen, C. E. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *Proceedings of the 32nd international conference on neural information processing systems*, pp. 2488–2498, 2018.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Tu, L. W. Manifolds. In *An Introduction to Manifolds*, pp. 47–83. Springer, 2011.

Tu, L. W. *Differential geometry: connections, curvature, and characteristic classes*, volume 275. Springer, 2017.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *ICML*, pp. 7654–7663, 2019.

Zou, D., Cao, Y., Li, Y., and Gu, Q. Understanding the generalization of adam in learning neural networks with proper regularization. *arXiv e-prints*, pp. arXiv–2108, 2021.

A. Supplementary Material

A.0.1. Necessary Definitions

Following definition of a manifold is relatively self-contained and comes from (Carmo & Flaherty, 1992).

Definition A.1 (Differentiable (Smooth) Manifold)**.** *A differentiable manifold of dimension n is a set M and a family of injective mappings $\boldsymbol{x}_\alpha : U_\alpha \subset \mathbb{R}^n \to M$ of open sets U_α of $\mathbb{R}^n$ into M such that:*

1. $\bigcup_\alpha \boldsymbol{x}_\alpha(U_\alpha) = M$
2. *for any pair α, β, with $\boldsymbol{x}_\alpha(U_\alpha) \cap \boldsymbol{x}_\beta(U_\beta) = W \neq \emptyset$, the sets $\boldsymbol{x}_\alpha^{-1}(W)$ and $\boldsymbol{x}_\beta^{-1}(W)$ are open sets in $\mathbb{R}^n$ and the mapping $\boldsymbol{x}_\beta^{-1} \circ \boldsymbol{x}_\alpha$ is differentiable.*
3. *The family $\{(U_\alpha, \boldsymbol{x}_\alpha)\}$ is maximal relative to the conditions (1) and (2).*

The pair $(U_\alpha, \boldsymbol{x}_\alpha)$ (or the mapping $\boldsymbol{x}_\alpha$) with $p \in \boldsymbol{x}_\alpha(U_\alpha)$ is called a parametrization (or system of coordinates) of M at p; $\boldsymbol{x}_\alpha(U_\alpha)$ is than called a coordinate neighborhood at p.

For alternative more fundamental definition of Topological Manifold we refer reader to works of (Tu, 2011; 2017; Hauser, 2018) or to excellent visual work on differential geometry (Needham, 2021).

Definition of Generalization Error We follow definition from widely accepted deep learning book (Goodfellow et al., 2016). For the sake of brevity we don't use bold typeset for vectors in here as opposed to the main body of the paper. The cost function can be written as an average over the training set, such as

$$J(w) = E_{\hat{p_{data}}(x,y)} \ell(f(x; w), y) \qquad \text{(training error)}$$

, where ℓ is the per-example loss function (such as Bregman divergence), and $f(x; w)$ is the predicted output when the input is x, and $\hat{p_{data}}$ is the empirical distribution. In the supervised learning case, y is the target output.

Equation (training error) defines an objective function with respect to the training set. We would usually prefer to minimize the corresponding objective function where the expectation is taken across the data-generating distribution p_{data} rather than just over the (finite) training set:

$$J^*(w) = E_{p_{data}(x,y)} \ell(f(x; w), y) \quad \text{(generalization error)}$$

In practice a generalization error is estimated on another (test) data set, randomly sampled from p_{data}.

A.0.2. "Two Moons" models

The "Two Moons" models are feed-forward neural networks with 2D inputs. Deeper version is has 7 *tanh* or *relu* or *linear* (3-5-10-10-5-2-2) layers, followed by *sigmoid*. Shallow has 2 *tanh* or *relu* or *linear* (5-2) layers + *sigmoid*.

Trained with constant learning rate (10^{-2} or 10^{-3}) and 200,20 and 2 batch sizes. Training and test data comprises 600 data points generated by scikit learn, (Pedregosa et al., 2011).

A.0.3. Flattening experiments

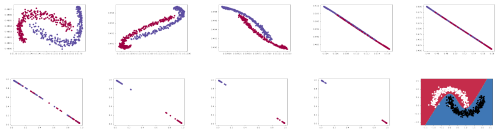


Figure 6: *Flattening of the outputs of 7 layer FNN "Two Moons" model, trained by back-prop with a constant learning rate and no regularizer.*

A.0.4. Effect of the depth

Proof of the Proposition 1 (also restated here).

Proposition 2. *Consider random weight initialization producing i.i.d. distribution of layer activations, conditionally on inputs, with finite moments*[10]*. Then in randomly initialized deep neural network (2) a probability of a curved $\Phi(z_{l_i})$ grows linearly with depth L.*

Proof. According to assumption, activations of layers z_{l_i} have i.i.d. distribution, given inputs, with finite moments. Assume that model uses non-linear activation functions and derivative of which, w.r.t. its argument, is different outside of the interval I from the derivative inside of I, i.e., $I = (-1, 1)$ for *tanh*, or $I = (0, \infty)$ for ReLU, etc.

Let's denote $P(z_{l_i} \notin I)$ a probability that activations z_{l_i} is out of I. Then the probability that *any* activation is out of I, i.e. $\sum_{l \leq L} \sum_{n \leq H^l} P(z_{l_n} \notin I)$ scales linearly in L for network with L layers and H^l activations in layer l.

On a side note, common initialization methods (He et al., 2015; Glorot & Bengio, 2010) normalize variance by number of nodes in layers, thus $\sum_{n \leq H^l}$ above is not reflected upon in general case, to allow for such normalization. □

Feature curves in deep vs. Shallow models experiment reported in Fig.7.

[10] such as common Xavier (Glorot & Bengio, 2010) or He (He

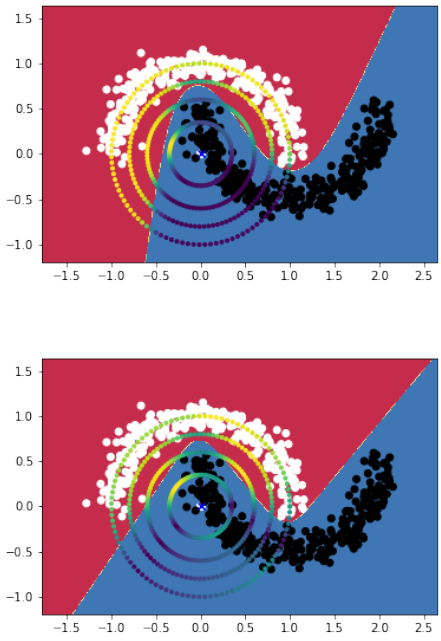


Figure 7: Deep vs. Shallow *w/o BN. Same as in Fig.9 colors encode ℓ_2 distance of $\Phi(t)$ images of (blue cross) from the $\Phi(t)$ images of the points on the circles,independently, i.e. colors are standardized for each circles irrespectively on the radius. In case of deeper model (upper) colors indicate "distances" distributions vary with radius, while in case of shallow model (bottom) they do less so. It visually demonstrates that "closeness" of two features, in deeper model, changes also radially and not only with direction as in shallower model. We argue this is due to larger curvature of $\Phi(t)$ enabling larger localization of the output metric in line with Fig.4. Both models have been trained for* $10,000$ *epochs, mini-batches of* 20 *,instead of* 200 *as those would not converge, on* 600 *training data samples with constant learning rate* 0.01.

A.1. Fitting Noisy & Random Labels

If targets differ for nearby inputs, as in case of noisy or even random labels, than large errors in "accuracy" component $\overrightarrow{\Delta_a}$ direction are produced and resulting Δ does not point along normal directions not producing "flattening" effect but rather large weight update in general. See Fig.10.

It has been shown in (Zhang et al., 2016; 2021) that deep nets are capable of fitting random labels with zero training

et al., 2015) initialization

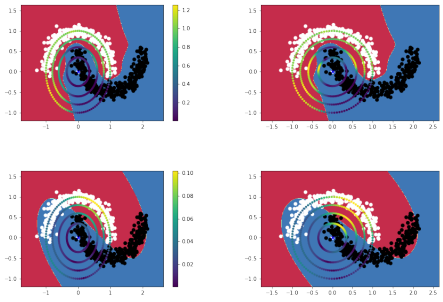


Figure 8: Deep vs. Shallow *+ BN. Both models have been trained for* $10,000$ *epochs, mini-batches of* 200 *on* 600 *training data samples with constant learning rate* 0.001.

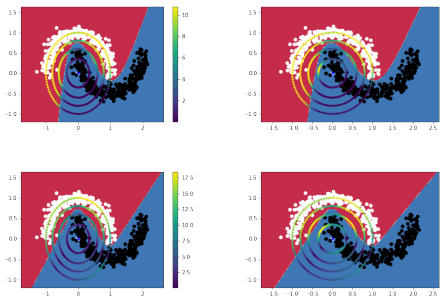


Figure 9: Deep vs. Shallow *w/o BN. Both models have been trained for* $10,000$ *epochs, mini-batches of* 20 *on* 600 *training data samples with constant and larger learning rate* 0.01 *compared to BN version.*

error. As reported therein it has been reached using Adam optimizer. Additional experiments in Supplementary Material show that normalizing gradient leads to over-fitting in some cases, in line with work (Zou et al., 2021) showing that Adam may find worse solutions that vanilla SGD. We argue this is a consequence of convergence conditions for training with normalized gradients.

Weight update of the Rprop (*sign* of gradient, (Hinton et al., 2012)) can also be written as standard gradient descent with learning rate decayed by norm of gradient

$$W_t - \frac{\eta}{\|g\|} g \tag{7}$$

where g denotes gradient of loss w.r.t. to weights. From (5) one can see norm $\|g\|$ is a function of weights through involved weight products $W(p)$ and it follows that the larger weight products are the slower training gets. In other words,

such adaptive methods are biased towards "large weights" configurations reciprocally to weight products $W(p)$ of (5).

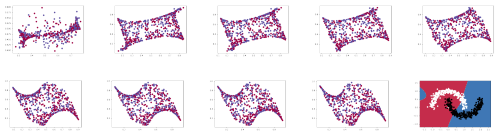


Figure 10: No Flattening for Noisy Labels *A shallow "Two Moons" feed-forward network model of 3 layers, trained by back-prop with a constant learning rate and BN. Labels are 30-times more noisy than in Fig.5. Noisy labels prevent flattening of the representations in reported 8000 epochs.*

Further, an increasingly large curvature of feature curves by promoting large weight products $W(p)$ by adaptive optimizers enables escaping from local optima and converging to zero-training error.

In such regime of the effectively small learning rate does not provide any chance to flatten and model stays in the over-parametrized regime. Such model has a signature of a poorly generalizing model (Zhang et al., 2016).

This is further demonstrated in Fig.13 with Adam and Fig.14 where Rprop is implemented[11].

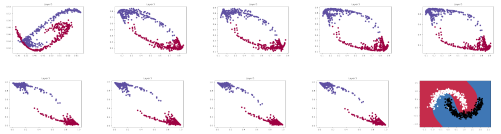


Figure 11: Normalized (BN) *feed-forward network of 3 layers. On the top first 2000 epochs of the training (snapshots by 500) are shown, the bottom row reports snapshots from epoch 8000, followed by resulting model with decision boundary on test data. Model, used $\tanh$ activations, constant learning rate of 0.001, mini-batch of 20, training and test data of 600 samples.*

A.1.1. Adam and RMSprop Experiments

Fig.14 shows RMSprop with no exp. decay hyper-parameter $\alpha = 0$. Further unreported experiments with varying α showed little or no-dependence on this hyper-parameter. Effectively it means that RMSprop reduces to training with gradients of unit norm multiplied by $sign()$ of gradient w.r.t. respective weight.

Fig.15, demonstrates the very same AE model as in Fig.14, just trained with mini batch size of 100 instead of unit mini-batch in Fig.14.

[11] through optim. RMSprop PyTorch implementation with hyper-parameter, controlling the length of moving average of previous gradients considered set to zero, $\alpha = 0$, (Paszke et al., 2017)

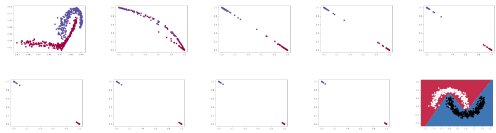
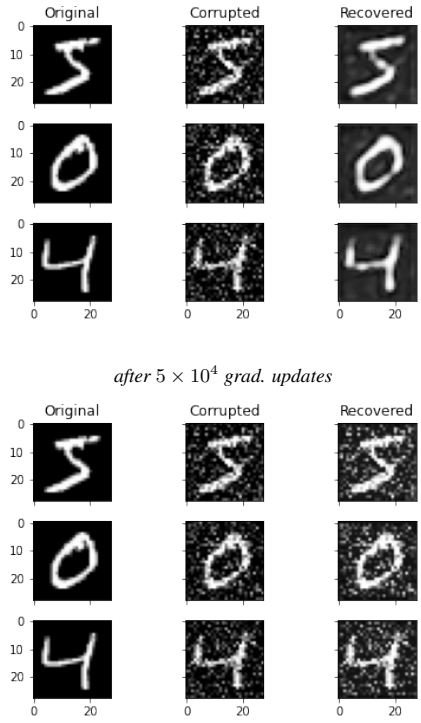


Figure 12: *The same model as in Fig.11 (3-layer FFN), trained with the same settings just **without BN**. Resulted in more profound and rapid flattening of the output representation.*



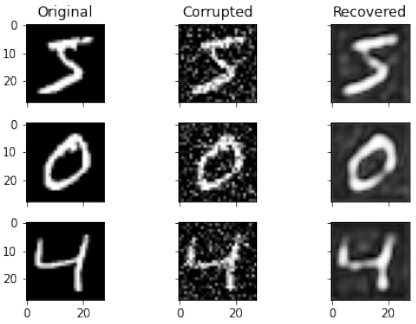
after 5×10^4 grad. updates

after 5×10^5 grad. updates.

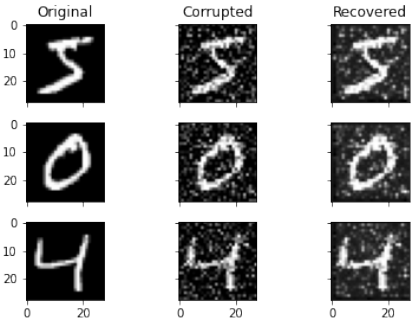
Figure 13: *DeepSIREN w/o BN, Adam, exp. decay with default first and second gradient moments hyper-parameters $\beta_1 = 0.9, \beta_2 = 0.999$ respectively, trained with unit mini-batch size.*

A.1.2. SIREN Experiments

- **DeepSIREN w/o BN**, 15 (encoder) + 10 (decoder) CNN layers, Sine activations, without BN.



after 5×10^4 *grad. updates*

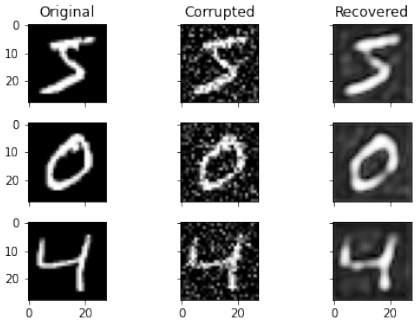


after 5×10^5 *grad. updates*

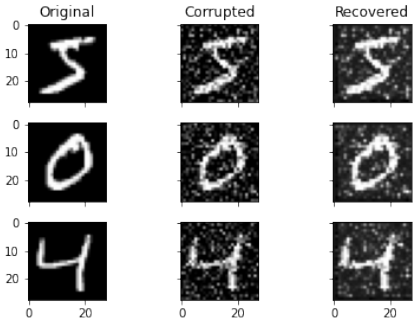
Figure 14: *DeepSIREN w/o BN, RMSprop, exp. decay hyper-parameter* $\alpha = 0$*, unit mini-batch size*

- **DeepSIREN+BN**, 15 (encoder) + 10 (decoder) CNN layers, Sine activations, with BN.

Both models are compared in versions with and without batch normalization.



after 5×10^4 *grad. updates*



after 5×10^5 *grad. updates*

Figure 15: *DeepSIREN w/o BN, RMSprop, exp. decay hyper-parameter* $\alpha = 0$*, mini-batch size of* 100

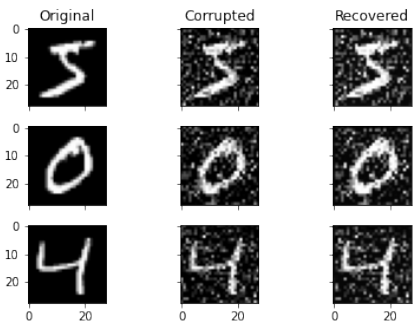


Figure 16: DeepSIREN w/o BN*. The depth alone without normalization was also not enough to find model insensitive to noise.*

Chapter 7

The Bayesian Cut (Paper B)

The Bayesian Cut

Petr Taborsky, Laurent Vermue, Maciej Korzepa, and Morten Mørup

Abstract—An important task in the analysis of graphs is separating nodes into densely connected groups with little interaction between each other. Prominent methods here include flow based graph cutting procedures as well as statistical network modeling approaches. However, adequately accounting for the holistic community structure in complex networks remains a major challenge. We present a novel generic Bayesian probabilistic model for graph cutting in which we derive an analytical solution to the marginalization of nuisance parameters under constraints enforcing community structure. As a part of the solution a large scale approximation for integrals involving multiple incomplete gamma functions is derived. Our multiple cluster solution presents a generic tool for Bayesian inference on Poisson weighted graphs across different domains. Applied on three real world social networks as well as three image segmentation problems our approach shows on par or better performance to existing spectral graph cutting and community detection methods, while learning the underlying parameter space. The developed procedure provides a principled statistical framework for graph cutting and the Bayesian Cut source code provided enables easy adoption of the procedure as an alternative to existing graph cutting methods.

Index Terms—normalized cut, ratio cut, graph cut, modularity, degree-corrected stochastic block modeling, Bayesian inference, incomplete gamma function, image segmentation.

1 Introduction

In the analysis of graphs, partitioning nodes into groups that are highly intra-connected with few inter-group connections has become important in disparate scientific fields - from network science for the identification of communities [1], [2], computer vision for image segmentation [3], [4] and the extraction of superpixel representations [5], scene reconstruction from large community photo collections [6], video decomposition [7], to physics for the splitting of materials [8]. In fact, many problems can be rephrased as a graph partitioning problem. This includes clustering problems based on pair-wise similarity in which graph partitioning approaches have found to have merits over traditional k-means and agglomerative hierarchical clustering procedures [9], and semi-supervised learning problems in which a popular solution procedure is to use graph cuts constrained according to the labelled observations [10], [11].

A variety of computational tools have been developed for graph partitioning. As such, methods based on minimizing flow between the separated entities have been devised based on various quality measures of cutting graphs. Two prominent procedures are the ratio cut [12] and normalized cut [3], for a review see also [4], [9]. On the other end, flexible in objective function, are methods minimizing certain classes of submodular energies in pairwise Markov Random Fields with applications in computer vision [13] and extended to certain nonsubmodular functions in [14]. Recently, inference in sparse graphs recovering true partitions using side information was introduced in [15]. While providing general optimisation frameworks these methods face scaling issues. Within network science a prominent procedure to identify communities is based on optimizing the modularity measure proposed in [1], which contrasts intra-group connectivity structure relative to the connectivity structure as would be expected according to the nodes' degree distribution. Within the social sciences identifying subgroups in graphs has been addressed using stochastic block-models (SBM) [16], [17] that identify homogeneous groups with similar connectivity profiles. This framework has been advanced to community detection by constraining parameters specifying intra-connectivity to be higher than inter-connectivity based on an information theoretic compression imposing intra and inter link constraints [18] or through Bayesian modeling constraining the parameters specifying intra and inter group link densities [19]. When partitioning networks a limitation of the SBM is that it is driven by grouping nodes according to their degree distribution. This issue has been alleviated by the degree-corrected stochastic block model (dc-SBM) proposed in [20] and its non-parametric Bayesian counterpart defined in [21]. Recently, it has been proven that modularity is a special case of maximum-likelihood estimation in the dc-SBM [22] assuming a planted l-partition model [23] in which link densities within l groups are specified only by two parameters; a within community η_{in} and between community strength η_{out} and further assuming the network is community structured, i.e. $\eta_{in} > \eta_{out}$. This then corresponds to the generalized modularity quality function proposed in [24] in which modularity is perfectly recovered when $\frac{\eta_{in}-\eta_{out}}{\log(\eta_{in})-\log(\eta_{out})} = 1$ [22].

In this paper, we propose a novel computational framework for cutting graphs into communities or groups that accounts for parameter uncertainty through Bayesian modeling. Our starting point is the dc-SBM in which we explicitly impose community structure requiring the parameters specifying intra-connectivity to be strictly larger than the corresponding inter-connectivity. Although less flexible than the dc-SBM, our model is more realistic than the planted l-partition model as we endow each community separate link-densities. We derive a Bayesian inference procedure and provide an analytical solution to the corresponding constrained integral representation. On three social networks

• *Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kgs. Lyngby, Denmark. E-mail: {ptab, lauve, mjko, mmor}@dtu.dk*

TABLE 1: Summary of the notation used.

Notation	Meaning	Definition
A	Adjacency Matrix	
A_{ij}	Link strength between node i and j	
b_c	Hyperparameter link density gap between the inter clusters expected link density and expected link density in community c	
C	Number of communities/clusters	
G	Undirected Graph	
d_i	Degree of node i	$A_{ii}/2 + \sum_{j \neq i} A_{ij}$
D_c	Sum of node degrees in cluster c.	$\sum_{i:z_i=c} d_i$
n	Total number of nodes in graph G	
n_c	Number of nodes in cluster c	$\sum_{i:z_i=c} 1$
n_{out}	Number of nodes between the clusters	$\sqrt{n^2 - \sum_{c=1}^{C} n_c^2}$
N	Total number of links in the graph G	$\sum_i A_{ii}/2 + \sum_{i<j} A_{ij}$
N_c	Number of links in cluster c	$\sum_{i:z_i=c} A_{ii}/2$ $+\sum_{i:z_i=c, j<i:z_j=c} A_{ij}$
N_{out}	Number of links between the clusters	$N - \sum_{c=1}^{C} N_c$
z_i	Cluster assignment of node i	
$\boldsymbol{z}$	Set of node assignments z_i for all nodes n	$\{z_1, z_2, \ldots, z_n\}$
Z_G	Normalizing constant of the graph G	$\prod_{i<j} A_{ij}! \prod_i \frac{A_{ii}}{2}! 2^{\frac{A_{ii}}{2}}$.
Z_{BC}	Normalizing constant of the constrained distribution	see (5)
α_c	A priori assumed link counts within community c, $\alpha_c \in \mathbb{R}^+$	
α_{out}	A priori assumed link counts between communities, $\alpha_{out} \in \mathbb{R}^+$	
β_c	A priori assumed number of network entries within community, $\beta_c \in \mathbb{R}^+$	
β_{out}	A priori assumed number of network entries between communities, $\beta_{out} \in \mathbb{R}^+$	
$\boldsymbol{\eta}$	Set of all η parameters	$\{\eta_1, \ldots, \eta_C, \eta_{out}\}$
γ	Degree correction hyperparameter	
η_c	Parameter controlling expected density of links within cluster c	
η_{out}	Parameter controlling expected density of links between clusters	
ϕ_i	Weight of node i	
$\boldsymbol{\phi}$	Set of node weights ϕ_i for all nodes n	$\{\phi_1, \phi_2, \ldots, \phi_n\}$
θ_i	Node degree control weight for node i	$n_{z_i} \phi_i$
$B(\boldsymbol{x})$	Multivariate Beta function	$\frac{\prod_k \Gamma(x_k)}{\Gamma(\sum_k x_k)}$

we demonstrate the importance of correctly accounting for community-structure when clustering nodes in graphs and that our Bayesian approach to cutting graphs have merits in contrast to the prominent graph cutting procedures outlined above. This includes better recovery of the true underlying partitioning structure of nodes into groups and more reliable inference. We further highlight the utility of the procedure for image segmentation considering both the Fast Marching Method (FMM) of [25] and the mean color regional adjacency graph (RAG) of [26] where normalized cut is typically applied. Notably, our results are for illustrative purposes demonstrated in the context of social network modeling in which the true partitioning structure is known, and image segmentation in which results can easily be visually inspected. However, we note that the computational framework developed has application beyond social network modeling and computer vision to the many domains in which graph cuts are currently used.

2 METHOD

Let G be an undirected graph with adjacency matrix A (i.e., $A_{ij} = A_{ji}$) whose elements A_{ij} are equal to the number of links between nodes i and j for $i \neq j$ and for computational reasons [20] twice that number for $i = j$. Let further n define the total number of nodes in the graph.

Following the dc-SBM [20] we assume that G is partitioned into a fixed number of C communities and the number of links between nodes i and j follow a Poisson distribution:

$$A_{ij} = \begin{cases} Poisson(\theta_i \theta_j \eta_{z_i z_j}) \text{ for } i \neq j \\ Poisson(\frac{1}{2} \theta_i^2 \eta_{z_i z_i}) \text{ for } i = j \end{cases}, \quad (1)$$

in which the parameter η_{ce} controls the probability of links between communities c and e, θ_i regulates the probability of links connected to the node i based on the degree of that node, and z_i defines the community assignment of node i. The factor of $\frac{1}{2}$ for $i = j$ results from the factor of two in the definition of diagonal elements of the adjacency matrix. In particular in all presented application in this paper self-links A_{ii} are constant. For the social networks presented they are zeros given by data, while in image applications with well defined similarities (following a common sense that node/pixel is similar to itself) they obtain maximal similarity.

As noted in [20] typically in large scale applications (i.e. images) self-links do not play a role as their effect diminish with scale ($\sim 1/n$). If necessary they can be marginalized as

suggested in [21]. Although it may be undesired to account for self-links they add to generality of the model that makes computations and (approximate) optimisation easier, i.e. [27].

In order to keep analytic tractability of the constrained model that will be introduced later we assume all links between different communities are generated using the same value, i.e. $\eta_{ce} = \eta_{out}$ for $c \neq e$. We will also refer to η_{cc} simply as η_c and $\boldsymbol{\eta}$ as the set of all $\{\eta_1, \ldots, \eta_C, \eta_{out}\}$ parameters. Accordingly, the probability of graph G can be written as:

$$\begin{aligned} P(G|\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{z}) &= \prod_{i<j} \frac{(\theta_i\theta_j\eta_{z_iz_j})^{A_{ij}}}{A_{ij}!} \exp(-\theta_i\theta_j\eta_{z_iz_j}) \\ &\times \prod_i \frac{(\frac{1}{2}\theta_i^2\eta_{z_iz_i})^{A_{ii}/2}}{(A_{ii}/2)!} \exp(-\frac{1}{2}\theta_i^2\eta_{z_iz_i}) \\ &= \frac{1}{Z_G}\eta_{out}^{N_{out}} \exp(-\frac{n_{out}^2}{2}\eta_{out}) \\ &\times \left[\prod_c \eta_c^{N_c} \exp(-\frac{n_c^2}{2}\eta_c)\right] \left[\prod_i \theta_i^{d_i}\right]. \end{aligned} \tag{2}$$

We have here used that $d_i = A_{ii}/2 + \sum_{j\neq i} A_{ij}$ is the degree of node i; $n_c = \sum_{i:z_i=c} 1$, $N_c = \sum_{i:z_i=c} A_{ii}/2 + \sum_{i:z_i=c, j<i:z_j=c} A_{ij}$ are respectively the number of nodes and links in community c; $n_{out}^2 = n^2 - \sum_{c=1}^C n_c^2$ and $N_{out} = N - \sum_{c=1}^C N_c$ with $N = \sum_i A_{ii}/2 + \sum_{i<j} A_{ij}$, whereas $Z_G = \prod_{i<j} A_{ij}! \prod_i \frac{A_{ii}}{2}! 2^{\frac{A_{ii}}{2}}$. Following [21], given partition $\boldsymbol{z}$, we define a constraint $\sum_{i:z_i=c} \theta_i = n_c$ and parametrize $\theta_i = n_{z_i}\phi_i$ such that parameters $(\phi_i)_{z_i=c}$ for each community c lie on a simplex. We endow all parameters with priors thereby accounting for uncertainty using Bayesian modeling. Thus, for given partition $\boldsymbol{z}$, we assign Dirichlet priors for the $(\phi_i)_{z_i=c}$ parameters of each community c. Further we impose Gamma priors for the elements of $\boldsymbol{\eta}$ and we obtain:

$$\begin{aligned} p(\boldsymbol{\phi}|\boldsymbol{z}) &= \prod_c \frac{1}{B(\gamma \mathbf{1}_{n_c})} \prod_{i:z_i=c} \phi_i^{\gamma-1}, \\ p(\boldsymbol{\eta}) &= \frac{\beta_{out}^{\alpha_{out}}}{\Gamma(\alpha_{out})} \eta_{out}^{\alpha_{out}-1} \exp(-\beta_{out}\eta_{out}) \\ &\times \prod_c \frac{\beta_c^{\alpha_{in}}}{\Gamma(\alpha_c)} \eta_c^{\alpha_c-1} \exp(-\beta_c\eta_c), \end{aligned} \tag{3}$$

where $B(\boldsymbol{x}) = \frac{\prod_k \Gamma(x_k)}{\Gamma(\sum_k x_k)}$ denotes the multivariate Beta function, and γ is a hyperparameter that allows to infer the optimal strength of degree correction for a given graph such that if $\gamma \to \infty$, then $\phi_i \to \frac{1}{n_c}$ and $\theta_i \to 1$ and the model reduces to the corresponding SBM [21]. On the other hand, if $\gamma \to 0$, then $\phi_{i^*} \to 1$ and $\theta_{i^*} \to n_c$ for some node i^* in each community c and thus a network generated according to this prior becomes dominated by a few greedy nodes. α_c and α_{out} denotes the a priori assumed number of links within community c and between communities (i.e., the prior shape parameter of the Gamma distribution) whereas β_c and β_{out} denotes the corresponding a priori imposed number of network entries (i.e., the prior rate parameter of the Gamma distribution) within community c and between communities. Assuming further an uniform prior on $\boldsymbol{z}$, $P(\boldsymbol{z}) = C^{-n}$, we obtain:

$$\begin{aligned} P(G, \boldsymbol{z}) &= \int P(G|\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{z}) p(\boldsymbol{\phi}) p(\boldsymbol{\eta}) P(\boldsymbol{z}) d\boldsymbol{\eta} d\boldsymbol{\phi} \\ &= \frac{C^{-n}}{Z_G} \frac{\Gamma(N_{out}+\alpha_{out})\beta_{out}^{\alpha_{out}}}{\left(\frac{n_{out}^2}{2}+\beta_{out}\right)^{N_{out}+\alpha_{out}} \Gamma(\alpha_{out})} \\ &\times \prod_c \frac{\Gamma(N_c+\alpha_c)\beta_c^{\alpha_c}}{\left(\frac{n_c^2}{2}+\beta_c\right)^{N_c+\alpha_c} \Gamma(\alpha_c)} \frac{B(\gamma \mathbf{1}_{n_c} + (d_i)_{i:z_i=c})}{B(\gamma \mathbf{1}_{n_c})} \eta_c^{D_c}, \end{aligned} \tag{4}$$

where $D_c = \sum_{i:z_i=c} d_i$ is the sum of node degrees in community c. The marginalized parameters $\boldsymbol{\eta} = \{\eta_1, \ldots, \eta_C, \eta_{out}\}$ can be interpreted as the densities of links within each community and between the communities respectively.

To ensure community structure in the graph, we presently restrict the model such that the within-community densities are larger than the between-community density. This has previously been considered in the context of the SBM [18], [19] but not in the context of the dc-SBM and without fully analytical tractable solutions to the constraints as presently derived. We constrain $\boldsymbol{\eta}$ parameters such that $\eta_c b_c \geq \eta_{out}$ for each community c where each b_c is a hyperparameter within range $[0, 1]$ specifying a density gap between the inter and intra community densities as considered in the context of the standard SBM in [19]. We introduce this constraint by defining the following constrained prior on the $\boldsymbol{\eta}$ parameters

$$\begin{aligned} p_{BC}(\boldsymbol{\eta}) &= \frac{1}{Z_{BC}} \eta_{out}^{\alpha_{out}-1} \exp(-\beta_{out}\eta_{out}) \\ &\times \left(\prod_{c=1}^C \eta_c^{\alpha_c-1} \exp(-\beta_c\eta_c)\right) I(\boldsymbol{\eta}), \end{aligned} \tag{5}$$

where $I(\boldsymbol{\eta}) = \prod_c \chi_{[0;\infty[}(\eta_c - b_c\eta_{out})$ is an indicator function evaluating to 1 if the constraints are satisfied and zero otherwise ($\chi_{[a;b]}(x)$ is the standard step function evaluating to one if $x \in [a; b]$ and 0 otherwise). Z_{BC} is the normalizing constant of this constrained distribution. For a summary of the notation used see table 1.

Combining priors with the likelihood function and

marginalizing the $\boldsymbol{\phi}$ and $\boldsymbol{\eta}$ parameters gives:

$$
\begin{aligned}
p(G,\boldsymbol{z}) &= \int p(G|\boldsymbol{\phi},\boldsymbol{\eta},\boldsymbol{z})p(\boldsymbol{\phi}|\boldsymbol{z})p_{BC}(\boldsymbol{\eta})p(\boldsymbol{z})d\boldsymbol{\eta}d\boldsymbol{\phi} \\
&= \int \eta_{out}^{N_{out}+\alpha_{out}-1}\exp\left(-\eta_{out}(\frac{n_{out}^2}{2}+\beta_{out})\right) \\
&\times \left[\prod_c \eta_c^{N_c+\alpha_c-1}\exp\left(-\eta_c(\frac{n_c^2}{2}+\beta_c)\right) I(\boldsymbol{\eta}) \right. \\
&\left. \times \frac{B\left(\gamma\mathbf{1}_{n_c}+(d_i)_{i:z_i=c}\right)}{B(\gamma\mathbf{1}_{n_c})} n_c^{D_c}\right] d(\boldsymbol{\eta}) \times \frac{C^{-n}}{Z_G Z_{BC}} \\
&= \int_0^\infty e^{-\eta_{out}(\frac{n_{out}^2}{2}+\beta_{out})}\eta_{out}^{N_{out}+\alpha_{out}-1} \\
&\times \prod_{c=1}^{C}\Gamma\left(N_c+\alpha_c, \eta_{out}\times\left(\frac{\frac{n_c^2}{2}+\beta_c}{b_c}\right)\right)d\eta_{out} \\
&\times \left[\prod_{c=1}^{C}\left(\frac{n_c^2}{2}+\beta_c\right)^{-(N_c+\alpha_c)}\right. \\
&\left. \times \frac{B\left(\gamma\mathbf{1}_{n_c}+(d_i)_{i:z_i=c}\right)}{B(\gamma\mathbf{1}_{n_c})} n_c^{D_c}\right] \times \frac{C^{-n}}{Z_G Z_{BC}},
\end{aligned}
\tag{6}
$$

where in the second step we used change of variables $s = \eta_c(\frac{n_c^2}{2}+\beta_c)$ to obtain each of c integrals in the form of an upper incomplete gamma function (in the following simply referred to as incomplete gamma function) given by [28]:

$$
\Gamma(\alpha,x) = \int_x^\infty s^{\alpha-1}e^{-s}ds \tag{7}
$$

A major challenge that remains and we presently solve is to analytically marginalize η_{out} in the above expression thereby solving analytically for the constraints specified by $I(\boldsymbol{\eta})$.

2.1 Marginalization of constrained η parameters

According to eq. (6) marginalizing under the constraint imposed by $I(\boldsymbol{\eta})$ requires the solution to an integral of the following form:

$$
\int_0^\infty e^{-B_0 x}x^{\mu_0-1}\left(\prod_{c=1}^{C}\Gamma(\mu_c,B_c x)\right)dx, \tag{8}
$$

(Marginalizing integral)

Where we have used the following substitutions, $x=\eta_{out}$, $\mu_c := N_c+\alpha_c$, $\mu_0 := N_{out}+\alpha_{out}$, $B_c := \frac{n_c^2}{2}+\beta_c$, $B_0 := \frac{n_{out}^2}{2}+\beta_{out}$, and ignored all terms independent on η_{out}. As a result, the μ_c and B_c elements in (8) relate respectively to scale and rate parameters of the involved incomplete gamma functions.

We outline what is to the best of our knowledge a novel approach solving integrals of the form presented in Eq.(8). We exploit the following known recurrence property of incomplete gamma functions (see Theorem 1 in [29]): $\Gamma(a+1,x) = a\Gamma(a,x)+x^a e^{-x}$ for $a\in\mathbb{R}, a>0$. This can be considered a generalization of $\Gamma(n+1)=n\Gamma(n)$ to the incomplete Gamma function. By a simple recursion of this property we obtain

$$
\Gamma(a,x) = \frac{\Gamma(a+K,x)}{(a)^{\dot{K}}} - x^a e^{-x}\sum_{i=0}^{K-1}\frac{x^i}{(a)^{\dot{i+1}}}
$$

(K-recurrence of $\Gamma' s$)

where $(a)^{\dot{n}}$ is the Pochhammer symbol (a.k.a. "rising factorial") defined as $(a)^{\dot{n}} = \Gamma(a+n)/\Gamma(a)$. This recursion formalizes idea of "shifting" of shape parameters of gamma distribution as shown in figure 1.

The following theorem presents application of the "shifting" method described above to solve the multidimensional incomplete gamma integral in equation (8) up to an arbitrary precision.

Theorem 2.1. *For every $C\in\mathbb{N}^+$, $\mu_i, B_i\in\mathbb{R}, \mu_i>0, B_i>0$ for $i\in\{1,...,C\}$ and $K\in\mathbb{N}^+$ following equality holds:*

$$
\begin{aligned}
&\int_0^\infty e^{-xB_0}x^{\mu_0-1}\prod_{c=1}^{C}\Gamma(\mu_c,B_c x)dx \qquad (9)\\
&= \sum_{m_1=1}^{C}\sum_{\substack{m_2=1,\\ m_2\neq m_1}}^{C}\cdots\sum_{\substack{m_C=1,\\ m_C\neq m_1,...,m_{C-1}}}^{C}\sum_{i_1=0}^{K-1}\cdots\sum_{i_C=0}^{K-1} \\
&\prod_{w=1}^{C}\frac{B_{m_w}^{\mu_{m_w}}(B_0+\sum_{j=1}^{w-1}B_{m_j})^{i_w}}{\left(\mu_0+\sum_{j=1}^{w-1}(\mu_{m_j}+i_j)\right)^{\dot{i_w+1}}} \\
&\times \frac{\Gamma\left(\mu_0+\sum_{j=1}^{C}(\mu_{m_j}+i_j)\right)}{(B_0+\sum_{j=1}^{C}B_{m_j})^{(\mu_0+\sum_{j=1}^{C}(\mu_{m_j}+i_j))}} \qquad (10)\\
&+ E(K), \qquad (11)
\end{aligned}
$$

where the error term $E(K)$ satisfies $\lim_{K\to\infty}E(K)=0$.

Proof. Detailed proof altogether with additional two proven lemmas is to be found in appendix. (6.3) □

To evaluate the joint distribution $p(G,\boldsymbol{z})$ the integral (8) is to be evaluated twice. First to compute prior normalization factor of hyperparameters (α's being gamma priors), denoted Z_{BC}, and second to evaluate the integral (6) with shape parameters μ's that are result of α's added together with link counts from the respective clusters.

While the former can be efficiently solved by theorem 2.1 as the prior values are typically small requiring a small value of K, the latter imposes substantial computational challenges especially for large and dense graphs where the use of theorem 2.1 becomes computationally heavy as the required K has to be in orders of magnitudes of the number of links in the largest cluster.

Rather than resorting to analytical integration one could opt for the use of point estimates in the large setting where the posterior distribution can be expected to be peaked and thereby point estimates to provide reasonable accuracy or apply simple normal approximations through the Laplace

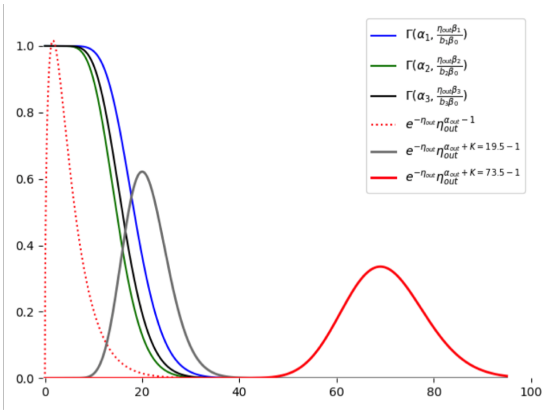


Fig. 1: Decomposition of integrand of Part A in lemma 6.1 into elements and shift of original gamma pdf (dotted red) using (K-recurrence of $\Gamma's$) to the inadequately shifted (gray curve K=19.5) and adequately shifted (red curve K=73.5) with close to zero-mass area of all the considered incomplete gamma functions (blue, green, and black curves) whereby the product in Part A becomes close to zero. As a result, the size of the shift controls the closeness to zero of Part A.

procedure also potentially accounting for the constraints using the result of the work of Hartman at al. [30] from 2017. Notably, a simple point estimate would be the maximum a posteriori of $\boldsymbol{\eta}$ under the required constraint and as the posterior is convex with convex constraints on $\boldsymbol{\eta}$ the MAP estimation of the constrained $\boldsymbol{\eta}$ is convex. Alternatively, η_{out} could be sampled and conditioned on the sampled value of η_{out}, η_c could be analytically marginalized using the incomplete Gamma function. While these approaches are scalable they are approximate and for the large scale setting we therefore opt for the following analytic procedure accounting explicitly for the uncertainty of $\boldsymbol{\eta}$ while keeping complexity at $O(C)$ for evaluating (8) which is the same as can be achieved by use of point estimates.

2.2 Large Scale Settings

Up until now there were no limitations set on values of η and in particular of hyperparameters μ and B, besides being real and positive. In large scale applications however, we are often facing large values of $\mu_{c,c\in\{1,\dots,C\}}$. In such case, it is convenient to consider evaluation of the integral for integer values of the $\mu's$. As we present in the following theorem, for integer $\mu's$ the integral is proportional to the CDF of the Negative Multinomial distribution with easy to evaluate limiting distribution. Notably, it is shown in section 3.1 that resorting to the integer setting imposes no significant constraints for most large scale applications.

Next we present main result of this section: exact evaluation of the integral (8) in case of integer shape parameters of involved gamma densities:

Theorem 2.2. *For* $C \in \mathbb{N}^+$, $\mu_i \in \mathbb{N}^+$ *and* $B_i \in \mathbb{R}^+$, $i \in \{0, ..., C\}$ *integral* (8) *is proportional to the cumulative distribution function of the Negative Multinomial (NMn) distribution and the following equality holds:*

$$\int_0^\infty e^{-xB_0} x^{\mu_0-1} \prod_{i\in\{1,\dots,C\}} \Gamma(\mu_i, B_i x)dx$$

$$= \frac{\prod_{c=0}^{C}\Gamma(\mu_c)}{B_0^{\mu_0}} \times \tag{12}$$

$$\times \sum_{i_1=0}^{\mu_1-1} \cdots \sum_{i_c=0}^{\mu_c-1} \frac{\Gamma(\mu_0+i_1+\ldots+i_c)}{\Gamma(\mu_0)i_1!\ldots,i_c!} \left(\frac{B_0}{B}\right)^{\mu_0} \prod_{c=1}^{C}\left(\frac{B_c}{B}\right)^{i_c}, \tag{13}$$

where $B := \sum_{i=0}^{C} B_i$

Proof. To be found in Appendix (6.4). For Negative Multinomial distribution definition and properties refer to [31]. □

The connection to Negative Multinomial distribution shown in theorem 2.2 also allows for an interpretation of the marginalized posterior (8) probability. If we consider sequence of independent multinomial trials in each of which event E_i occurs with probability $p_{i,i\in\{0,\dots,C\}}, \sum_{i=0}^{C} p_i = 1$ and let X_i be the frequency of $E_{i,i\in\{1,\dots,C\}}$ "successes" before predefined number μ_0 of X_0 "failures" appears, then $(X_0, X_1, ..., X_C)$ follows the Negative Multinomial distribution NMn [31].

Hence integral (8) is proportional to the likelihood of observing μ_i "successes" (links within clusters) before number of "failures" (links between clusters) reaches at most μ_0, given that number of links in graph follows multinomial distribution with probability of links appearing in cluster c being $\frac{B_c}{B}$, which is positively related to the relative size of a cluster (proportion of nodes in cluster) c.

In the following section we make use of favourable asymptotics of the Negative Multinomial distribution to derive a fast evaluation of the integral for the large scale setting.

3 Inference

We presently show how to efficiently evaluate Theorem 2.1 for $C = 2$ clusters. In this case, the formula (omitting the error term) can be written as:

$$B_1^{\mu_1} B_2^{\mu_2} \Gamma(\mu_0) \sum_{m=1}^{2} \sum_{i_1=0}^{K-1} \sum_{i_2=0}^{K-1} \frac{(1+B_m)^{i_2}}{(1+B_1+B_2)^{\mu_T+i_1+i_2}}$$
$$\times \frac{(\mu_0+i_1+1)^{(\mu_m-1)}\Gamma(\mu_T+i_1+i_2)}{\Gamma(\mu_0+\mu_m+i_1+i_2+1)},$$

where $\mu_T = \mu_0+\mu_1+\mu_2$. If we apply substitution $v = i_1+i_2$, we can rewrite the above expression as:

$$B_1^{\mu_1} B_2^{\mu_2} \Gamma(\mu_0) \sum_{m=1}^{2} \sum_{v=0}^{2(K-1)} \frac{(1+B_m)^{v}\Gamma(\mu_T+v)}{(1+B_1+B_2)^{\mu_T+v}}$$
$$\times \sum_{i_1=0}^{min(v,K-1)} \frac{(\mu_0+i_1+1)^{(\mu_m-1)}}{(1+B_m)^{i_1}}.$$

We notice that the sums dependent on v or i_1 can be evaluated independently in $\mathcal{O}(K)$ time which allows for efficient evaluation compared to the original $\mathcal{O}(K^2)$ time.

With regards to control of the approximation error Theorem 2.1 gives for arbitrary error thresholds ϵ the existence of K that evaluates this integral up to ϵ precision. However, the Theorem is not explicit about the choice of a sufficient value of K. One simple approach for finding K to control approximation error we used to produce the results presented in section 2.1 is to set K such that the mode of inter cluster link density $\frac{\mu_0+K-1}{B_0}$ is equal or greater than the q-quantile of all gamma distributions controlling intra clusters link densities. An accuracy is then controlled by setting values of q. Results of this application on karate network are shown in figure 2. There are many alternative choices for K, however, we found this approach to be easy and efficient in practice. For the purpose of error evaluation we compared results of Theorem 2.1 with results of the scipy.integrate.quad function from the scipy 1.2.0 python package. From the figure we can observe how increasing K, corresponding to increasing the q-quantile according to the method described above, controls the absolute error on the evaluation of the integral. For the results obtained in the following we used $q = 0.9999$, given that this guarantees an absolute error close to 10^{-5}, but in most cases will range around 10^{-9}.

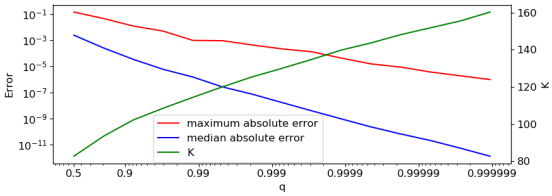


Fig. 2: Maximum and median absolute approximation error and corresponding number of added observations T for karate network based on 100 chains with 100 samples each.

Typically, a graph cut is obtained by optimizing a given cost function. In case of Bayesian Cut, the cost function is defined by the posterior distribution $p(\boldsymbol{z}|G)$ which specifies probability of every possible partition of graph G. While the full posterior would provide lots of insight into different ways of cutting the graph, due to its high complexity, it is not possible to determine it fully. Instead, the most reasonable approach is to search for the maximum of the posterior (MAP) $\boldsymbol{z}_{MAP} = \text{argmax}_{\boldsymbol{z}} p(\boldsymbol{z}|G)$. While one could opt for optimization of the posterior distribution of $\boldsymbol{z}$ possibly making use of wide arsenal of approximation methods i.e. [14], [13], [32] or other discrete optimisation methods [33] to this NP hard problem, we advocate using MCMC sampling (for reference see [34], Chapter 11) before performing optimization for a few reasons. First of all, optimization might get stuck in local maxima while sampling given enough time will find the global maximum. In practice, within the sampling budget, the sampler will likely focus on some high density region of the posterior, but it will still explore multiple modes within that region. A comparison of only using optimization compared to using the sampler can be found in the appendix, see section 6.3. Secondly, by using sampling we are able to infer values of specific hyperparameters to create a more plausible model that explains the observed data better and thus learn about the underlying structure of the problem. Finally, sampling produces not only a point estimate but an approximation of the true posterior (more or less accurate depending on its complexity and sampling budget) that can be used to answer more complex questions than what the most probable cut is. To perform MCMC sampling, we use Gibbs sampling and sample each element z_i of $\boldsymbol{z}$ independently:

$$p(z_i|G) = \frac{p(G,\boldsymbol{z})}{p(G,\boldsymbol{z}_{-i})} \propto p(G,\boldsymbol{z}).$$

We treat the hyperparameter γ as a random variable while fixing other parameters to a constant value. We use the non-informative prior $p(\gamma) = \gamma^{-1}$ and after each Gibbs sweep over all nodes in the graph, we perform 20 Metropolis-Hastings (MH) updates using the proposal distribution $\gamma^* = \gamma \exp(\epsilon), \epsilon \sim N(0, \sigma = 0.1)$. Alternatively, if one is not interested in inferring γ, it can be set to 1 which assumes any configuration of node-specific parameters $(\phi_i)_{z_i=c}$ of community c is equally probable (i.e., corresponding to the uniform distribution over the $(n_c$–$1)$-simplex). Furthermore, we fix all α and β parameters to a non-informative value 0.01 and set b to 1 unless specified otherwise. After running out of the sampling budget, we apply deterministic optimization by switching node assignments only when it leads to higher likelihood and we stop when in a full sweep over all nodes we do not observe any further improvement.

3.1 Inference for large graphs

Posterior distribution of η (expected density of links) in BC model has the same form as prior (due to the conjugacy between Poisson and Gamma) where 'shape' μ_c and 'rate' B_c, in general positive real parameters of the involved gamma densities, are by definition priors $\alpha_c \in \mathbb{R}^+$ and $\beta_c \in \mathbb{R}^+$ updated by the added number of links and nodes in cluster c respectively. This often results in large, in general real, values when dealing with large graphs. In order to find a fast evaluation algorithm first let us note that for the cutting of large graphs limiting ourselves to integer shape hyperparameters of both prior and posterior gamma densities while updating real 'rate' impose any relevant constraints in most applications as the prior is overwhelmed by the observed data. Technicaly speaking transformation from $\mu' = \lceil \mu \rceil, B' = \frac{\lceil \mu \rceil}{\mu} B$ keeps mean of posterior gamma distribution unchanged ($\frac{\mu'}{B'}$) while increases its variance (or uncertainty) ($\frac{\mu'}{B'^2}$) by factor diminishing with scale. Therefore and especially with uninformative priors resorting to integer 'shape' should have an insignificant and asymptotically zero effect on posterior for large values of μ and if not the general Theorem (2.1) should be applied.

Secondly, as shown in [31], a limiting distribution of the negative multinomial decomposes into a product of Poisson distributions as $\mu_0 \to \infty$. Making use of this limiting distribution we obtain a large scale (asymptotic) solution of our integral. In the following we make use of the fact that the cummulative density function (cdf) of a Poisson distributed variable $F_{Pois(\lambda)}(\mu_i - 1)$ can be written as $\Gamma(\mu_i - 1; \lambda)$. Let m denote the threshold beyond which the asymptotic is applied. To determine m we analyze in Figure 3 how well the asymptotic approximation of the marginalized integral (8) behaves. The figure shows that absolute error of log integral

is close to zero but for the bipartite setting, corresponding to a graph in which all links/similarities are between clusters while there is zero density of link/similarity within clusters. In such setting it is still possible to evaluate the integral exactly using Theorem (2.2) with complexity $O(N)$. However, if the observed graph G has bipartite structure (can be detected prior to application of the method) then the proposed asymptotic becomes expensive. This does not impose any issues for most applications, in particular, for image segmentation where bipartite structures are unlikely. Formally, proposed method to evaluate the marginalized integral (8) in large scale settings depends on sum of weights (in our case number of links) between clusters, μ_0:

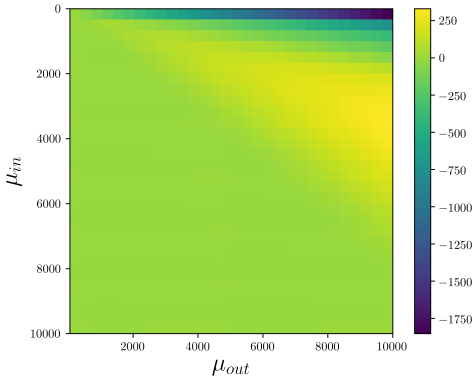


Fig. 3: Error of logarithm of integral (8) evaluated by "shifted" method of theorem (2.1) with bounded error of 10^{-5} and logarithm of same integral evaluated by asymptotic method of section (3.1). For experiments we fixed $B_0 = 60$ and $B_1 = B_2 = 70$ while ranging $\mu_{out} \in (51, 10^3)$ and $\mu_1 = \mu_2 = \mu_{in} \in (0, 10^3)$.

Large $\mu_0 > m$: If μ_0 is sufficiently large, then (41) resolves asymptotically into:

$$\prod_{i=1}^{C} \Gamma(\mu_i - 1; \mu_0 B_i / B_0) \tag{14}$$

Small $\mu_0 \leq m$: In this case there are 2 options:

- In case the smallest of μ_c's, $_{c\in\{1,\dots,C\}}$, is sufficiently large $\min_c(\mu_c) > m$ we apply 'per-partes' on (41) to rotate elements of integral and asymptotic decomposition on each of C summands resulting in:

$$B_0^{-\mu_0} \prod_{i=0}^{C} \Gamma(\mu_i) - \sum_{j=1}^{C} \prod_{\substack{i=0,\\ i\neq j}}^{C} \Gamma(\mu_i - 1; \alpha_j B_i / B_0) \tag{15}$$

- Else, when one or more μ_c's, $_{c\in\{1,\dots,C\}}$, are small $(\min_c(\mu_c) < m)$, asymptotic properties of NMn are of no use. This corresponds to a degenerated case when nodes within one or more clusters are dissimilar or respective clusters contain few nodes. In either case this does not correspond to a preferable cut. Let's note that it is unlikely that the Metropolis - Hastings/ Gibbs MCMC sampler appears to be sampling from an assignment corresponding to this case unless observed graph has aforementioned bipartite structure. In degenerate case sampler would need to accept low probability proposals against the imposed constraints on the link densities. So unless initial assignments of sampler are degenerate or number of clusters C is extremely large compared to nodes in the considered graph, it is unlikely to end up in such case during sampling. Anyway, in such case we evaluate the integral of theorem 2.2 directly at cost of higher complexity $O(N)$ instead of $O(C)$.

In the procedure above m represents a threshold above which asymptotic apply. In our image experiments (non bipartite structure) we applied $m = 50$ given results of fig. 3, striking balance between accuracy and runtimes. More elaborate and/or conservative choices may be better suited, depending on use.

3.2 Reference methods

We contrast the proposed BC to the corresponding dc-SBM without community constraints given by (4) as well as to modularity optimization (Mod), ratio-cut (RC) and normalized cut (NC). The solutions obtained by RC and NC were derived using the spectral clustering procedure described in [9] whereas the modularity objective was optimized using the spectral approach described in [1]. We note that the spectral optimization procedure may be suboptimal to other inference approaches, however, we presently use these solutions for illustrative purposes to characterize the methods and contrast favourable configurations by these approaches to the favorable configurations using the proposed BC procedure.

To evaluate the modularity score of a given partition we use the modularity objective function described in [1], given by

$$\begin{aligned} Q(\boldsymbol{z}) = \frac{1}{4m} \sum_{c=1}^{C} \Bigg[\sum_{i:z_i=c} \Bigg(\sum_{j:z_j=c} (A_{ij} - \frac{k_i k_j}{2m}) \\ - \sum_{j:z_j\neq c} (A_{ij} - \frac{k_i k_j}{2m}) \Bigg) \Bigg], \qquad m = \frac{1}{2} \sum_{i=1}^{n} k_i \end{aligned} \tag{16}$$

To evaluate solutions in the domain of NC and RC we use their respective cost functions as defined in [9]

$$RC(\boldsymbol{z}) = \frac{1}{2} \sum_{c=1}^{C} \frac{1}{n_c} \sum_{i:z_i=c} \sum_{j:z_j\neq c} A_{ij}, \tag{17}$$

$$NC(\boldsymbol{z}) = \frac{1}{2} \sum_{c=1}^{C} \frac{1}{K_c} \sum_{i:z_i=c} \sum_{j:z_j\neq c} A_{ij}. \tag{18}$$

3.3 Visualization technique for solution landscape

To show the solutions supported by each procedure we plot the solution landscapes similar to the method proposed in [35]. These landscapes were created by obtaining a set of V $\boldsymbol{z}$ vectors for the models under scrutiny. This set of vectors is expanded by 50% to cover the in between solution

space through pseudo-random vectors, i.e. a new vector is generated by randomly taking two distinct $\boldsymbol{z}$ vectors and combining half of the elements of each vector. The score or likelihood for each vector is subsequently obtained by running the specified model with each unique $\boldsymbol{z}$ vector. As measure of distance between partition vectors we use the Variation of Information [36] between all V vectors. The resulting $V \times V$ dimensional distance matrix is reduced to two dimensions using Multidimensional Scaling [37]. Discrete Sibson Interpolation [38] is subsequently used to obtain a meshgrid of the remaining two dimensions.

4 Results and Discussion

In the following we analyze the properties of the proposed Bayesian Cut (BC) model for community detection in social networks and image segmentation for computer vision.

We first present results on a set of simple synthetic networks (Section 4.1) followed by results for community detection in social networks (Section 4.2) that have an advantage of available "ground truth" as well as unified definition of an adjacency matrix across methods we compare with. Hence presented comparison provides insights on graph cutting performance more clearly than in the subsequent image applications where cuts are used in connection with disparate similarity matrices. In Section 4.3 two often used similarity matrices are presented to demonstrate utility of BC model as a generic tool for graph cuts. We further apply multiple cluster solutions on the images.

The Bayesian Cut source code used for these experiments is provided through a public source code repository, hosted on Github (https://github.com /DTUComputeCognitiveSystems /bayesian_cut), and through the Python Package Index (https://pypi.org/project/bayesian-cut/) to allow a straightforward installation of the package. To ensure accessibility and reproducibility of the results, the repository includes image of "bears" used in experiments (original downloaded from: https://images.app.goo.gl /Mvdra73AwjfRfp629) and the software, that is accompanied by instructions on how to use the package and Jupyter Notebooks that show how the results were obtained.

4.1 Synthetic networks

We test the proposed algorithm on synthetic networks to demonstrate the effect of imposed connectivity constraint on the inference. In this experiment, we fix the total number of nodes to $n = 100$ and links to $N = 1000$. We assume networks are partitioned into two communities having equal number of nodes ($n_1 = n_2 = \frac{n}{2}$) and links ($N_1 = N_2 = N_{in}$). For different values of intra- to inter-community link density ratio ($\eta_{in}/\eta_{out} = \frac{2N_{in}}{N_{out}}$), we generate network to match these predefined properties. We fix γ to 10^6 (to remove effects of degree correction), α_{out} to 10^{-6} (to remove the difference coming from marginalizing the constrained vs. unconstrained prior) while keeping values of the other hyperparameters as specified in Section 3. In Figure 4, we show the posterior densities (up to a constant) of the partition for a wide range of η_{in}/η_{out} density ratios for a constrained (Bayesian Cut) and corresponding unconstrained (dc-SBM) model to demonstrate the effect of the constraint. Condition $\eta_{in}/\eta_{out} < 1$ represents an extent of constraint violation - the closer it is to 0, the stronger the violation. At extreme of 0, the partition represents a bipartite network which is a structure exactly opposite to a community structure. As it can be seen in the figure, unconstrained dc-SBM assigns very high probability to partitions where there is very distinct difference between intra- and inter-community densities, even if inter-community density is higher. On the other hand, the constrained model penalizes partitions that violate the constraint and assigns them even lower probability than to partitions with the density of links uniformly distributed over the whole graph.

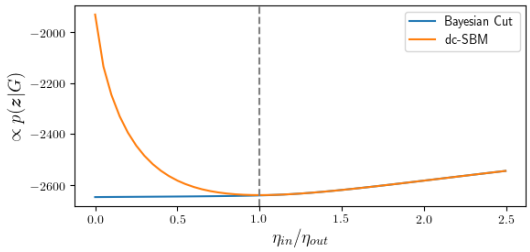


Fig. 4: Experiment on synthetic networks confirms that constrained BC model "Bayesian cut" strongly penalizes partitions that violate graph connectivity constraint, $\eta_{in} \geq \eta_{out}$, compared to unconstrained "dc-SBM" model that assigns very high probability to partitions where there is very distinct difference between intra- and inter-community densities, even if inter-community density is higher.

4.2 Community detection in social networks

For community detection the properties of the proposed Bayesian Cut (BC) model are analyzed based on three real world social networks and contrasted to ratio-cut, normalised cut, modularity and the unconstrained dc-SBM. The networks considered are:

Karate: A social undirected network studied by Zachary [39] of ties in a Karate club that turned out to split in two. The network consists of 34 nodes and 78 edges and was partitioned using modularity in [1].

Polblogs: The political blogosphere (Polblogs) network on US politics assembled by [40]. We consider the largest connected component of the network in the undirected form used in the dc-SBM analysis of [20] which contains 1222 nodes and 16714 edges.

HIV-1: Sexual partnership network extracted from the first study (Colorado Springs Project 90) in HIV Transmission Network Metastudy Project [41]. We consider the largest connected component of the network consisting of 1888 nodes and 2096 edges.

In all analyses we used $C = 2$ corresponding to the ground-truth structure of the split in Karate club and political blogs along party line. Notably, when $C = 2$ there is only one η_{out} parameter in the dc-SBM and our analyses correspond to the dc-SBM parametrization with and with-

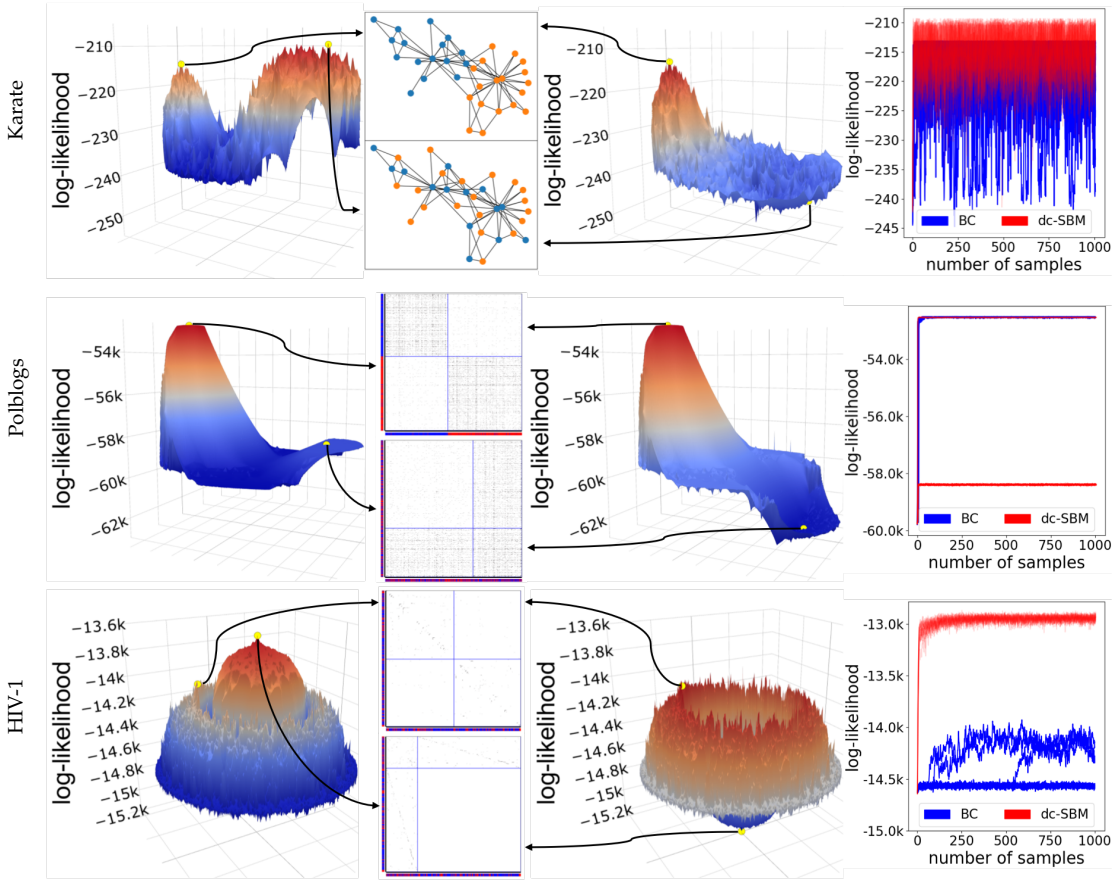


Fig. 5: Comparison of the dc-SBM (left column) and BC (right column) solution landscapes based on $p(\boldsymbol{z}|G)$ as well as the resulting cuts performed on the three networks. The outer right column shows the trace plots obtained running each model with 15 chains and 1000 samples. The dc-SBM model exhibits for all three networks modes and thus resulting cuts that violate the constraint $\eta_{in} \geq \eta_{out}$. In the corresponding adjacency matrices it can be seen that whenever the constraint is violated (lower adjacency matrix/network for each example), the off-diagonal blocks have a higher density than at least one of the diagonal blocks. In contrast, the proposed BC model gives those regions of the solution landscape that violate the constraint lower likelihoods, which leads only to modes and thus resulting cuts that do not violate the constraint.

out the community constraint. For model inference in the dc-SBM and BC we use Gibbs sampling to infer $\boldsymbol{z}$.

4.2.1 Comparison of dc-SBM and BC

Figure 5 shows the results of the unconstrained dc-SBM and our Bayesian Cut (BC) procedure for $b_1 = b_2 = 1$, i.e. imposing the constraint $\eta_1 \geq \eta_{out}$ and $\eta_2 \geq \eta_{out}$. Furthermore, a non-informative prior is used, i.e. $\alpha_{in} = \alpha_{out} = \beta_{in} = \beta_{out} = 0.01$. For the Karate network (top panel) we observe that the conventional Bayesian dc-SBM (given by the likelihood in (4)) creates a substantially different solution from our proposed BC. While our BC peaks around the true split of the Karate network, we observe that the samples of the conventional dc-SBM concentrate around two modes of the distribution in which the other mode represents a configuration that does not comply with the notion of community structure, but has a significantly higher likelihood.

For the larger Polblogs network we again observe that the dc-SBM exhibits one mode that does not comply with the community structure and creates a split leading to one community with high link density and one community with a bipartite structure, while the mode shared with our proposed model corresponds well to a separation along political orientation (i.e., democrat vs. republican). In the bottom panel for the HIV-1 network we observe a substantial difference between the dc-SBM and our proposed BC procedure with no shared modes. Here the unconstrained model identifies a bipartite structure in which one community has very low link density as compared to the inter community link density, whereas the constrained model by

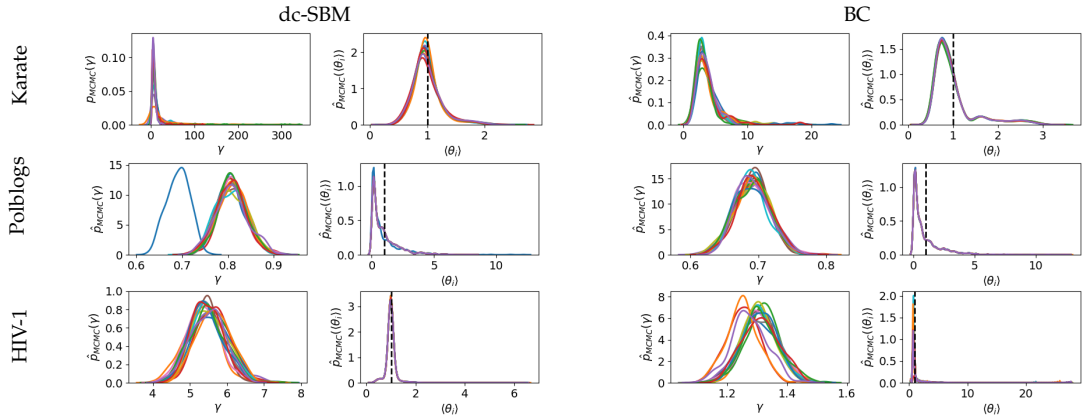


Fig. 6: Gamma inference and resulting node degree correction (theta) of the dc-SBM and BC for all three networks. The dc-SBM and BC models show substantial differences, since the parameter inference resulting from the BC model is more reliable, because it contrary to the dc-SBM does not get stuck in local optima that violate the constraint.

only giving community structure support strives to separate the network according to identifying separate communities.

Overall it can be seen that the BC with its constraints is more in line with the natural splits in the Karate and Polblogs networks and suggests a more sensible split for the HIV-1 network. The sub-optimal congruity of the unconstrained model can be attributed to the local modes of the posterior observed in Figure 5 that are unsupported by the BC procedure.

On the outer right side in Figure 5 the convergence of the dc-SBM and BC is illustrated for 15 chains and 1000 samples. Notably, we observe that for the Karate and political blogosphere networks the unconstrained model explores the mode not complying with community structure. For the political blogosphere the inference for most of the chains is stuck in the local sub-optimal mode of the posterior distribution, incapable of escaping this mode by the Gibbs sampler and recovering the underlying correct structure leading to the lower cut shown in the middle panel of figure 5. In contrast all chains of the BC model converge to the underlying partitioning structure for both networks. When considering the HIV-1 network it can be observed that the solution space supporting community structure is consisting of a vast number of local optima, contrary to the non-community supporting structure, which has a strong global mode. This is causing the community structure inferred to be less reliable and the chains to end in local modes of the community constrained posterior.

The influence of BC and dc-SBM on inferring the parameter controlling for degree (γ) is shown in figure 6. Here the gamma inference as well as the node degree correction distribution of each chain of both the dc-SBM and BC model is shown for the three networks. Focusing on the left column, a substantial difference within the inference of the γ parameter, i.e. controlling the degree correction, is observable. Subsequently, the derived θ parameters differ based on the modes preferred by the models. As previously shown, the dc-SBM model often gets stuck in local modes or exhibits globally preferred modes that do not support the community structure. Accordingly, the parameter inference is biased by the modes in the non-community structure region in those cases. In the above analysis we used non-informative priors on $\boldsymbol{\eta}$, however, we could also impose an informed prior favoring community structure in the dc-SBM. This and role of constraint parameter b is further addressed exemplary on the karate network in the appendix, section 6.4.

TABLE 2: Comparison of Cuts running 100 chains with 1000 samples without and with (in parenthesis) deterministic optimization in terms of their modularity value (Mod.) and correspondence to ground truth partition structure (avaible for Karate and Polblogs) as quantified using normalized mutual information (NMI). $\langle\cdot\rangle$ denotes average value and $\lceil\cdot\rceil$ maximum value.

	Score	RC	NC	MOD	dc-SBM	BC
Karate	⟨NMI⟩	0.415	0.732	-	0 (0)	**0.837 (0.837)**
	⌈NMI⌉	0.578	0.732	**0.837**	0 (0)	**0.837 (0.837)**
	⟨Mod.⟩	0.236	0.356	-	-0.267 (-0.258)	**0.371 (0.371)**
	⌈Mod.⌉	0.313	0.356	**0.371**	-0.267 (-0.258)	**0.371 (0.371)**
Polblogs	⟨NMI⟩	0.017	0.017	-	0.143 (0.146)	**0.717 (0.718)**
	⌈NMI⌉	0.017	0.017	0.693	0.727 (0.737)	**0.739 (0.739)**
	⟨Mod.⟩	0.001	0.001	-	-0.057 (-0.062)	**0.426 (0.426)**
	⌈Mod.⌉	0.001	0.001	0.424	**0.426 (0.426)**	**0.426 (0.426)**
HIV	⟨Mod.⟩	0.045	0.045	-	-0.363 (-0.365)	**0.185 (0.411)**
	⌈Mod.⌉	0.045	0.045	0.190	-0.357 (-0.359)	**0.385 (0.463)**

4.2.2 *Comparison of dc-SBM and BC to Modularity, NC and RC*

In Table 2 we quantify the correspondence as measured by normalized mutual information (NMI) between the inferred partitions and the partition defined by the underlying split with highest support for each of the considered methods in the Karate network and separation according to party line in Polblogs. Furthermore, we measure the adherence to community structures of each model by calculating the modularity for the inferred partitions using the formula defined in eq. 16. For each calculated metric and network we point out the average and maximum score achieved by that particular method.

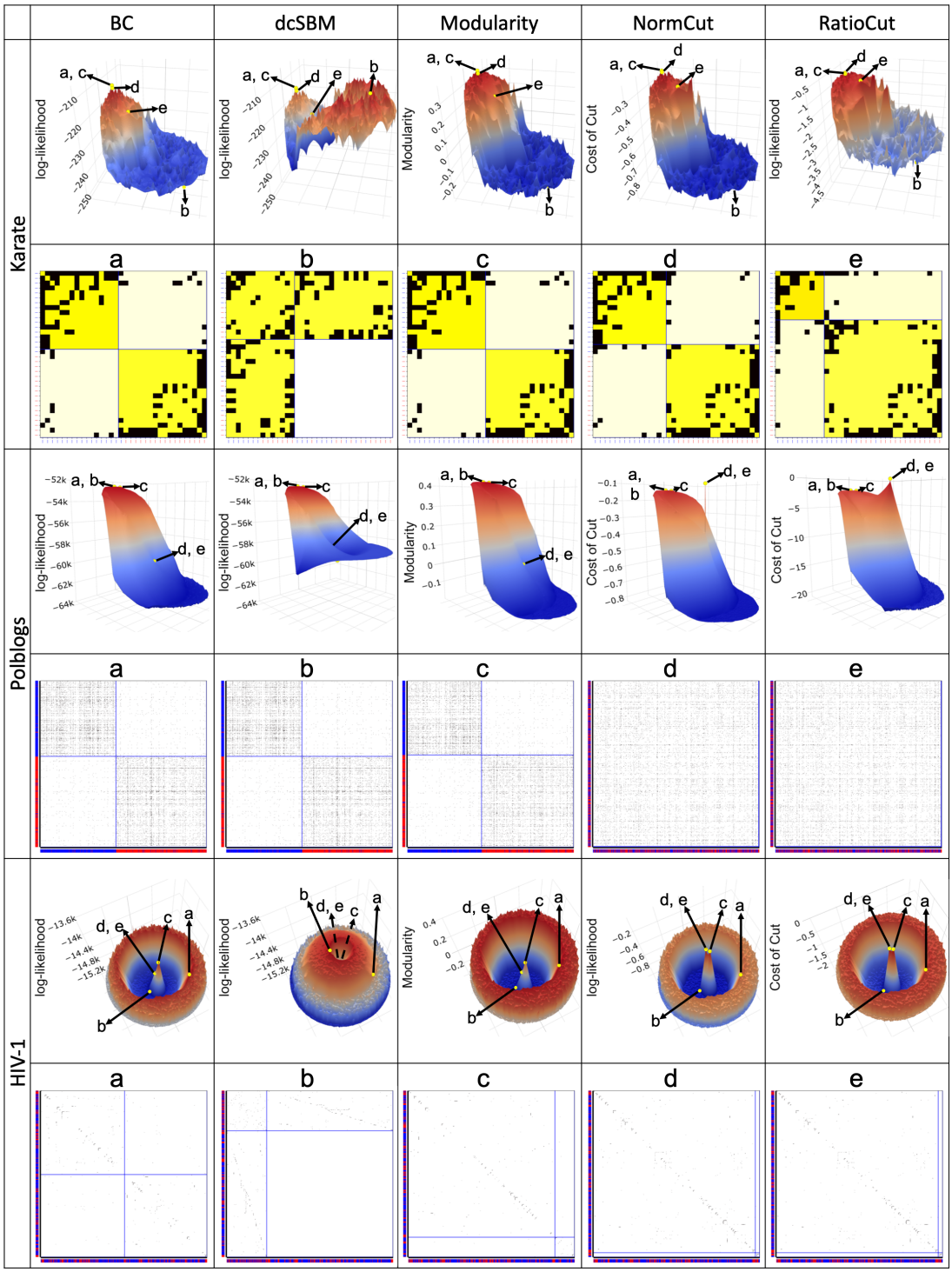


Fig. 7: Solution landscape comparison of BC, dc-SBM, Modularity, NormCut, RatioCut on the three networks. To explore the space, 100 samples from 15 chains were taken for the Bayesian methods, while for the spectral cuts 200 different solutions were generated for each method by randomly alternating 1% of the links within the networks. The costs of Normcut and Ratiocut are inverted to allow for direct landscape comparisons.

We observe here that the BC achieves superior or on par performance on all three networks. In these results we again observe that the BC differs substantially from the dc-SBM, which is explained by the underlying supported configurations of the model likelihood $P(\boldsymbol{z}|G)$ shown in figure 5. In figure 7 we explore the solution space also of the ratio-cut (RC), normalized cut (NC) and modularity (Q) and how these solutions are supported by their corresponding objective functions.

We notice that the solutions supported (and thus the inference landscape) by the proposed BC is more in agreement with these existing community detection/graph partitioning procedures than the dc-SBM. However, we also observe notable differences of the proposed Bayesian Cut (BC) and these alternative partitioning procedures. In particular, neither RC nor NC provide as balanced solutions as the BC and they provide higher support for solutions further away from the underlying community structure. Here we pay particular attention to the cuts proposed for the Polblogs and HIV-1 network as these appear unsubstantiated due to the fact that they exclude a very small group of persons from the overall population.

For Polblogs both RC and NC exhibit very extreme and local optima in their solution landscape, which lead to a cut that excludes 4 persons from the other 1218 persons in both cases. In the case of RC, defined in eq 17, the dominance of the cost by the flow, i.e. links between two groups, is obvious. Since these two group are only connected by 1 inter-link the cost for performing this is cut is extremely low. In contrast, the true cut along party lines leads to one group with 662 nodes and one with 560, which share 1217 inter-links. To obtain lower costs, no-more than 76 inter-links would be allowed.

One way of alleviating this strong influence of the cut flow is to use NC, defined in eq. 18, which does not divide the cut flow by the number of nodes, but according to the degree of the cluster. However, even though this subtle difference changes the solution landscape in non-community supporting regions as shown in figure 7, the preference for cuts that separate unbalanced groups having very low flow remains. In this case the extreme cut leaves the small group with a degree of 5 and the bigger group with a degree of 16710, which results in very low costs. The above mentioned cut of our model results in a degree of 9464 and 8467 for the group with 662 nodes and 560 nodes respectively. In this case 894 inter-links would already give lower costs for our cut, which shows the improvement over RC, but still is not sufficient.

The highest congruence can be found between the BC and the Modularity method, confirming the community detection support of our proposed BC. Here we observe that both methods exhibit almost identical solution landscapes, which is reflected in the identical or very similar solution landscapes obtained by the methods. Interestingly, for the HIV-1 network the BC obtains a solution with a significantly higher modularity than the spectral modularity method itself identifies. In addition, this solution seems to be more balanced, since it achieves almost equally sized groups, while the proposed solution of the spectral modularity method partitions the network into a small and a large group. This highlights that BC strives for balanced modular structures.

4.3 Image Segmentation

In following we present results of image segmentation suitable for foreground-background or scene recognition. We compare BC model to NC and dc-SBM (with shared density of links out η_{out} in case of more than two segments $C > 2$).

NC implementations are often in practice combined with specific similarity matrices and we make use of the following two widely used procedures:

Mean color RAG: Mean color Regional Adjacency Graph is used to compute similarity matrices on super pixel graphs (RAG) [26] that serves as an input for NC in popular python package for image processing skimage https://scikit-image.org/docs/dev/api/skimage.future.graph.html. To compare with the BC method "cameraman" image and "coffee"available in the skimage package was used.

Fast Marching Method (FMM): This method (a.k.a. geodesical distance) is besides many used in the Graclus software presented in [25]. We used the MATLAB implementation of Jianbo Shi from https://www.cis.upenn.edu/∼jshi/software/ to generate the FMM similarity matrix. Graclus software optimizes normcut objective in a hierarchical manner [25] with results presented at https://www.cis.upenn.edu/∼jshi/software/demo2.html. For comparison purposes we use the public image of "baby" from the same site.

These methods produce similarity matrices $\mathbf{S}$ with elements in $[0; 1]$. To convert them into graphs with countable links required by the BC model we follow similar procedure as aforementioned Graclus software [25]. Graclus runs $\mathbf{A} = \lceil 100 * \mathbf{S} \rceil$ while BC implements $\mathbf{A} = \lfloor 100 * \mathbf{S} \rfloor$, both element wise.

Results of the BC model applied on images of "cameraman" and "bears" using RAG can be found in figure (8), "coffee" is presented in Appendix (13) and the results on "baby" using FMM can be found in figure (9). Notably BC model was applied on similarity matrices produced by respective implementations of RAG and FMM described above without further adjustments. In case of RAG and "bears" we adjust sigma for the Gaussian similarity kernel [1] in case of "cameraman" we leave it on default setting. "Cameraman" and "bears" experiments with Mean Color RAG have been ran with no degree correction (corresponds to hyper parameter γ set extremely large 10^7).

In all applications mentioned BC performs on par or superior to the compared methods (not necessarily state of the art though). In two partitions version considered for the "cameraman" the BC method separates objects from sky. In case of the four partition scenario used on "bears" BC recognizes foreground objects (cub and surrounding), background and adult bear while the other methods only partially succeed. For the "coffee cup" in appendix the BC

1. future.graph.rag_mean_color(img, labels1, mode='similarity', sigma=70**2, segmentation.slic(img, compactness=0.3, n_segments=100)

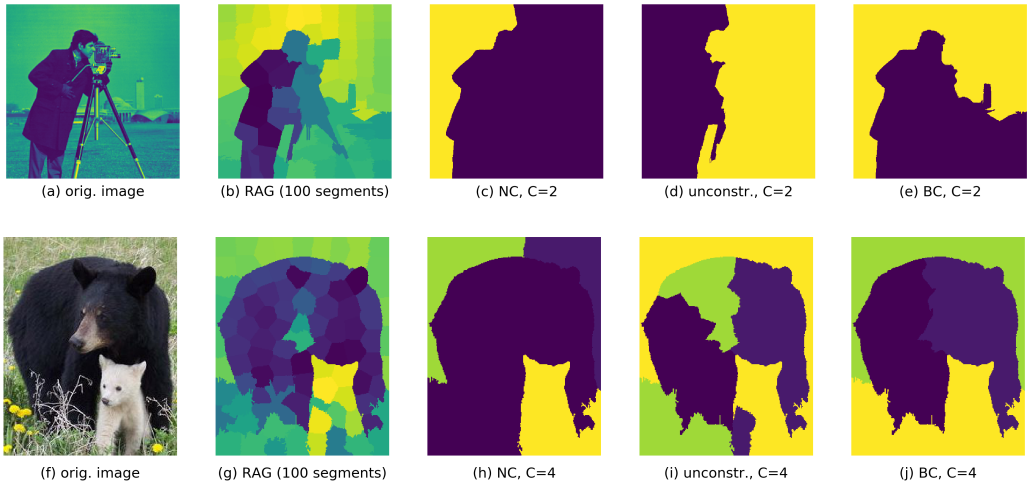


Fig. 8: **Top panel, Cameraman:** Resulting cut of BC model for C=2 (e) segments compared with unconstrained dc-SBM with shared η_{out} as well as spectral Norm cut. Fig (b) shows for reference RAG super pixel graph that is used to compute similarity matrix. Results demonstrate on par or better results of BC against referenced methods. Also it shows effect of constraint included in the model (we emphasized the effect of constraint by setting b=10 corresponding to 10x times higher within segment links density compared to links density among segments): (e) vs (d) a constraint model improves the results. Resulting cuts were obtained from 50 MCMC chains, 1000 samples each.
Bottom panel, Bears: Resulting cut of BC model for C=4 segments (j) compared with unconstrained dc-SBM with shared η_{out} (i) as well as spectral Norm cut (h) applied on mean color RAG similarity matrix. Similar to previous results figures BC demonstrates on par or better results against referenced methods. Resulting cuts were obtained from 50 MCMC chains, 1000 samples each with hyperparameters set on $b = 10^3$ and without degree correction

Fig. 9: Bayesian cut (BC) applied on similarity matrix obtained by fast marching method implemented by Jianbo Shi from https://www.cis.upenn.edu/~jshi/software/. Resulting cut is sampled MAP obtained from 20 MCMC chains, 1000 samples each. (a) original, (b) $C = 2, b = 10$, with degree correction hyper parameter γ being inferred. Maximum of its posterior obtained at: $\gamma_{MAP} = 4.07$, (c) $C = 2, b = 10, \gamma = 0.0001$.

removes more of the background than the unconstrained dc-SBM and captures more of the coffee cup object than NC. In case of the "baby" image see figure 9 the effect of degree correction parameter γ controlling "greediness" of clusters is showed. In the more greedy settings, (c) as opposed to gamma being inferred in option (b), fixing it to "greedy" mode recognizes focal object's boundary more complete yet produces artifacts.

In summary, the presented image segmentation results by BC are on-par or superior to NC and the unconstrained version. However, we noted during experiments that the multiple MCMC runs produced slightly different cuts confirming that the inference is prone to sub optimal solutions and multiple restarts are therefore recommended.

5 Conclusion

We have proposed the Bayesian Cut (BC) advancing the degree-corrected stochastic block-model (dc-SBM) to explicitly account for community structure. In contrast to the dc-SBM only one parameter specified inter-group connectivity strength (η_{out}), however, in contrast to the generalized modularity as conforming to an l-partition model with shared link density across communities the proposed BC include more flexible community specific link-densities. We derived a fully Bayesian procedure and demonstrated that the imposed community constraints are analytically tractable even for large graphs by deriving a novel general solution to integrals involving multiple incomplete gamma functions. We expect the presented small and large scale solutions to the integral will have applications beyond community detection in social networks and image segmentation considered in this paper. For instance, for collapsed inference in the performance analysis of cognitive radio networks [42]. We observed that the constraints had significant impact on the inference providing more reliable results in compliance with ground truth for network exhibiting community structure

and it was also empirically confirmed that the constraint had merits for image segmentation in computer vision. We also observed that strictly enforcing community structure enabled to identify configurations where traditional block-modeling would identify bipartite structure. Notably, our Bayesian Cut provides favorable partitions when compared to traditional graph cutting procedures such as the ratio and normalized cut. In particular, we empirically observed that our BC procedure has meritorious properties balancing the partitions more favorable than these existing graph partitioning procedures. We have also derived fast large scale multiple cluster solution that presents generic tool for Bayesian inference.

We presently considered a uniform prior on the partition $P(\boldsymbol{z}) = C^{-n}$ to highlight the influence of the specification of the likelihood $p(G|\boldsymbol{z})$ in identifying partitions. However, we note that within the Bayesian modeling framework other (non-uniform) priors could be applied including the Pólya-urn (i.e., marginalized Dirichlet-Categorical) representation and its infinite limit given by the non-parametric Chinese restaurant process (CRP) also used in stochastic block-modeling [43].

Overall this work presents generic graph based clustering method that can be applied on wide range of similarity matrices. For illustrative purposes we presently applied our BC approach in the context of identifying communities in social networks and image segmentation, however, the approach extends to the many applications in which graph cuts are used. Flexibility with regards to similarity matrix allows for possible applications in areas such as scene reconstruction from large set of community photos [6], where the image set is partitioned into groups of related images, based on the visual structure represented in the image connectivity graph for the collection. Connectivity graph and corresponding similarity matrix is based on scale invariant feature transform, SIFT [44], that extracts image representative features that are used to find matches and define similarity between each image pair. Another possible area of application is Video summarization and scene detection [7], where similarity used for graph partitioning are based on color similarity and temporal frame distance.

In the outlook, although MCMC sampling are suitable for network structure inference, in order to find optimal cuts, future work should investigate alternatives while keeping the properties of the proposed framework. Further concerning image segmentation, this work made use of two popular similarities, Fast Marching Method and Mean Color, that rather relate pixels based on color intensities as opposed to spatial features. As suggested above we leave as future work to explore possibilities of BC applied on other existing or new similarities as well as extension of hereby presented bayesian generative hierarchical BC model to allow for contextual spatial or other features [34].

REFERENCES

[1] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[2] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[3] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[4] B. Peng, L. Zhang, and D. Zhang, "A survey of graph theoretical approaches to image segmentation," *Pattern Recognition*, vol. 46, no. 3, pp. 1020–1038, 2013.

[5] Z. Li and J. Chen, "Superpixel segmentation using linear spectral clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1356–1363.

[6] N. Snavely, I. Simon, M. Goesele, R. Szeliski, and S. M. Seitz, "Scene reconstruction and visualization from community photo collections," *Proceedings of the IEEE*, vol. 98, no. 8, pp. 1370–1390, 2010.

[7] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang, "Video summarization and scene detection by graph modeling," *IEEE Transactions on circuits and systems for video technology*, vol. 15, no. 2, pp. 296–305, 2005.

[8] M. Witman, S. Ling, P. Boyd, S. Barthel, M. Haranczyk, B. Slater, and B. Smit, "Cutting materials in half: A graph theory approach for generating crystal surfaces and its prediction of 2d zeolites," *ACS central science*, vol. 4, no. 2, pp. 235–245, 2018.

[9] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[10] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01, 2001, pp. 19–26.

[11] J. Wang, T. Jebara, and S.-F. Chang, "Semi-supervised learning using greedy max-cut," *Journal of Machine Learning Research*, vol. 14, no. Mar, pp. 771–800, 2013.

[12] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[13] V. Kolmogorov and R. Zabih, "What energy functions can be minimizedvia graph cuts?" *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 2, pp. 147–159, 2004.

[14] V. Kolmogorov and C. Rother, "Minimizing nonsubmodular functions with graph cuts-a review," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 7, pp. 1274–1279, 2007.

[15] D. J. Foster, D. Reichman, and K. Sridharan, "Inference in sparse graphs with pairwise measurements and side information," *arXiv preprint arXiv:1703.02728*, 2017.

[16] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.

[17] K. Nowicki and T. A. B. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American statistical association*, vol. 96, no. 455, pp. 1077–1087, 2001.

[18] M. Rosvall and C. T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," *Proceedings of the National Academy of Sciences*, vol. 104, no. 18, pp. 7327–7331, 2007.

[19] M. Mørup and M. N. Schmidt, "Bayesian community detection," *Neural computation*, vol. 24, no. 9, pp. 2434–2456, 2012.

[20] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, no. 1, p. 016107, 2011.

[21] T. Herlau, M. N. Schmidt, and M. Mørup, "Infinite-degree-corrected stochastic block model," *Physical review E*, vol. 90, no. 3, p. 032819, 2014.

[22] M. E. Newman, "Equivalence between modularity optimization and maximum likelihood methods for community detection," *Physical Review E*, vol. 94, no. 5, p. 052315, 2016.

[23] A. Condon and R. M. Karp, "Algorithms for graph partitioning on the planted partition model," *Random Structures & Algorithms*, vol. 18, no. 2, pp. 116–140, 2001.

[24] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, no. 1, p. 016110, 2006.

[25] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[26] A. Trémeau and P. Colantoni, "Regions adjacency graph applied to color image segmentation," *IEEE Transactions on image processing*, vol. 9, no. 4, pp. 735–744, 2000.

[27] Y. Zhang, Z. Ghahramani, A. J. Storkey, and C. A. Sutton, "Continuous relaxations for discrete hamiltonian monte carlo," in *Advances in Neural Information Processing Systems*, 2012, pp. 3194–3202.

[28] R. AlAhmad, "Products of incomplete gamma functions," *Analysis*, vol. 36, no. 3, pp. 199–203, 2016.
[29] G. Jameson, "The incomplete gamma functions," *The Mathematical Gazette*, vol. 100, no. 548, pp. 298–306, 2016.
[30] M. Hartmann, "Extending owen's integral table and a new multivariate bernoulli distribution," *arXiv preprint arXiv:1704.04736*, 2017.
[31] M. Sibuya, I. Yoshimura, and R. Shimizu, "Negative multinomial distribution," *Annals of the Institute of Statistical Mathematics*, vol. 16, no. 1, pp. 409–426, Dec 1964. [Online]. Available: https://doi.org/10.1007/BF02868583
[32] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows," 1988.
[33] R. G. Parker and R. L. Rardin, *Discrete optimization*. Elsevier, 2014.
[34] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. CRC press, 2013.
[35] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Science advances*, vol. 3, no. 5, p. e1602548, 2017.
[36] M. Meilă, "Comparing clusterings by the variation of information," in *Learning theory and kernel machines*. Springer, 2003, pp. 173–187.
[37] I. Borg and P. Groenen, "Modern multidimensional scaling: theory and applications," *Journal of Educational Measurement*, vol. 40, no. 3, pp. 277–280, 2003.
[38] S. W. Park, L. Linsen, O. Kreylos, J. D. Owens, and B. H. Hamann, "Discrete sibson interpolation," 2006.
[39] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.
[40] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 36–43.
[41] M. Morris and R. Rothenberg, "Hiv transmission network metastudy project: An archive of data from eight network studies, 1988–2001," 2011.
[42] B. Van Nguyen, H. Jung, D. Har, and K. Kim, "Performance analysis of a cognitive radio network with an energy harvesting secondary transmitter under nakagami-m fading," *IEEE Access*, vol. 6, pp. 4135–4144, 2018.
[43] M. N. Schmidt and M. Morup, "Nonparametric bayesian modeling of complex networks: An introduction," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 110–128, 2013.
[44] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.
[45] P. Taborsky, L. Vermue, M. Korzepa, and M. Morup, "The bayesian cut," *IEEE Transactions on Pattern Analysis Machine Intelligence*, no. 01, pp. 1–1, may 5555.
[46] G. Jameson, "A simple proof of stirling's formula for the gamma function," *The Mathematical Gazette*, vol. 99, no. 544, pp. 68–74, 2015.

Petr Taborsky received his M.Sc. degree in Mathematics, Mathematical Statistics and Probability at Charles University, Prague. Currently he's PhD understudy at the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. He's also in AI team, Telenor Danmark. His research interests include federated machine learning, neural networks, and complex network modeling.

Laurent Vermue received his M.Sc. degree in Industrial Engineering and Management at the Technical University of Berlin and MMSc. degree in Management Science and Engineering at the Tongji University. Currently he is a Ph.D. student at the Section for Statistics and Data Analysis and the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. His research interests include machine learning, complex network modeling and open research software.

Maciej Korzepa received his M.Sc. degree in Digital Media Engineering at the Technical University of Denmark. He is currently a Ph.D. student at the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. His research interests include machine learning, intelligent interfaces, and complex network modeling.

Morten Mørup received his M.S. and Ph.D. degrees in applied mathematics at the Technical University of Denmark and he is currently Professor at the Section for Cognitive Systems at DTU Compute, Technical University of Denmark. He has been associate editor of IEEE Transactions on Signal Processing and his research interests include machine learning, neuroimaging, and complex network modeling.

6 SUPPLEMENTARY MATERIAL/APPENDIX

6.1 Theoretical results and proofs

Due to incomplete gamma recurrence (K-recurrence of $\Gamma's$) it is possible to rewrite the integral (8) according to the following lemma (6.1):

Lemma 6.1. *For $C \in \mathbb{N}$ and all real $\mu_i > 0, B_i > 0, i \in \{0, ..., C\}$ the following equalities hold*

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx \tag{19}$$

$$= \frac{\Gamma(\mu_0)}{B_0^{\mu_0}} \prod_{c=1}^{C} \Gamma(\mu_c) - \sum_{m=1}^{C}$$

$$\frac{B_m^{\mu_m}}{B_0^{\mu_0}} \int_0^\infty \Gamma(\mu_0, B_0 x) e^{-B_m x} x^{\mu_m - 1} \prod_{\substack{c=1 \\ c \neq m}}^{C} \Gamma(\mu_c, B_c x) dx \tag{20}$$

$$= \underbrace{\int_0^\infty \frac{e^{-xB_0} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx}_{\text{Part A}} + \sum_{m=1}^{C} \sum_{i=0}^{K-1} \tag{21}$$

$$\underbrace{\frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \int_0^\infty e^{-x(B_0 + B_m)} x^{\mu_0 + \mu_m + i - 1} \prod_{\substack{c=1 \\ c \neq m}}^{C} \Gamma(\mu_c, B_c x) dx}_{\text{Part B}} \tag{22}$$

Proof. The first equality follows directly by applying integration by parts using $u' = e^{-xB_0} x^{\mu_0 - 1}$ and $v = \prod_{c=1}^{C} \Gamma(\mu_c, B_c x)$ and the second by the use of (K-recurrence of $\Gamma's$). □

Using the above lemma, the aim is to select the number of recurrences K such that Part A is made arbitrarily small as illustrated by the underlying terms given in Figure 1. The following lemma proofs that Part A can indeed be made arbitrarily small.

Lemma 6.2. *For all real $\mu_i, B_i \in \mathbb{R}, \mu_i > 0, B_i > 0, i \in \{0, ..., C\}$ and constant $Q, Q \in \mathbb{R}, Q \geq 0$ following limit exists and holds:*

$$\lim_{K \to \infty} K^Q \times \int_0^\infty \frac{e^{-xB_0} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx = 0 \tag{23}$$

Proof. We proof the lemma by showing that for every K there exists q such that integral over (q, ∞) is below threshold $\epsilon/2$ and consequently there exists K large enough such that the integral over $(0, q]$ is also below $\epsilon/2$.

Without loss of generality we assume $B_0 = 1$. (If $B_0 \neq 1$, we apply a transformation of variables $y = B_0 x$ on (23) and take $\epsilon \times B_0^{\mu_0}$ as a new epsilon and $\frac{B_c}{B_0}$ as new $B_c's$).

Denote $f(x, K) = \frac{e^{-x} x^{\mu_0 + K - 1}}{\Gamma(\mu_0 + K)}$, which is a density function of the gamma distribution with shape parameter $\mu_0 + K$ and rate 1 such that $f(0) = 0$. For every $K, K \geq 1$ $f(x, K)$ is positive and increasing on $(0, m(K))$, where $m(K)$ is the mode $\mu_0 + K - 1$, controlled by K which follows well known characteristics of the gamma distribution.

Upper bound on $(0, q]$: Denote the regularized upper incomplete gamma function $\frac{\Gamma(\alpha, bx)}{\Gamma(\alpha)}$ which can be written as 1 - the cumulative distribution function [29] of the gamma distribution and is therefore positive, decreasing on $\mathbb{R}^+$, and bounded from the top by 1.

For every $q \in \mathbb{R}, q > 0$ and for every $K \in \mathbb{N}^+$ large enough to ensure that q is below the mode of the gamma distribution with density $f(x, K)$, that means $K > q - \mu_0 + 1$, the following inequality holds:

$$K^Q \times \int_0^q \frac{e^{-x} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx$$

$$\leq q K^Q f(q, K) \prod_{c=0}^{C} \Gamma(\mu_c) \tag{24}$$

We make use of Stirling's formula [46]:

$$\Gamma(x) \sim (2\pi)^{\frac{1}{2}} x^{x - \frac{1}{2}} e^{-x} \text{ as } x \to \infty, \tag{25}$$

where notation $g(x) \sim h(x)$ means that $\frac{g(x)}{h(x)} \to 1$ as $x \to \infty$. Taking $K \to \infty$ and using Stirling's formula above on $f(q, K)$ gives us existence of the following limit:

$$\lim_{K \to \infty} q K^Q f(q, K) * \prod_{c=0}^{C} \Gamma(\mu_c) = 0 \tag{26}$$

From (ϵ, δ)-limit definition of (26) we get that for every $q \in \mathbb{R}, q > 0$ and every $\epsilon' > 0, \epsilon \in \mathbb{R}^+$ there exists $K' \in \mathbb{N}, K' > 0$ such that for every $K \geq \max(K', q - \mu_0 + 1), K \in \mathbb{N}$ equation (24) $\leq \epsilon'$.

Upper bound on $[q, \infty)$: For every $q > 0$ we have from definition that all upper incomplete gamma functions are smooth and decreasing on $[q, \infty)$ and can be bounded from top by its value at q. We thereby get the following upper bound by taking the integral over region from q to ∞, recognizing the upper incomplete gamma function and using $\frac{\Gamma(a,x)}{\Gamma(a)} \leq 1$ (which trivially follows from the definition of $\Gamma(a, x)$):

$$K^Q \int_q^\infty \frac{e^{-x} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx \tag{27}$$

$$\leq K^Q \frac{\Gamma(\mu_0) \prod_{c=1}^{C} \Gamma(\mu_c, B_c q)}{\Gamma(\mu_0 + K))} \int_q^\infty e^{-x} x^{\mu_0 + K - 1} dx$$

$$= K^Q \frac{\Gamma(\mu_0 + K, q) \prod_{c=1}^{C} \Gamma(\mu_c, B_c q)}{\Gamma(\mu_0 + K)} \Gamma(\mu_0)$$

$$\leq K^Q \prod_{c=1}^{C} \Gamma(\mu_c, B_c q) \Gamma(\mu_0) \tag{28}$$

Following limit exists and follows directly from definition of incomplete gamma function (7):

$$\lim_{q \to \infty} \prod_{c=1}^{C} \Gamma(\mu_c, B_c q) \Gamma(\mu_0) = 0 \tag{29}$$

From the above limit we can now select q such that the error is below $\epsilon/2$ and for this q we select K such that (26) is also below $\epsilon/2$.

□

Theorem 6.3. *For every $C \in \mathbb{N}^+$, $\mu_i, B_i \in \mathbb{R}, \mu_i > 0, B_i > 0$ for $i \in \{1, ..., C\}$ and $K \in \mathbb{N}^+$ following equality holds:*

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \prod_{c=1}^{C} \Gamma(\mu_c, B_c x) dx \tag{30}$$

$$= \sum_{m_1=1}^{C} \sum_{\substack{m_2=1,\\ m_2 \neq m_1}}^{C} \cdots \sum_{\substack{m_C=1,\\ m_C \neq m_1,\ldots,m_{C-1}}}^{C} \sum_{i_1=0}^{K-1} \cdots \sum_{i_C=0}^{K-1}$$

$$\prod_{w=1}^{C} \frac{B_{m_w}^{\mu_{m_w}} (B_0 + \sum_{j=1}^{w-1} B_{m_j})^{i_w}}{\left(\mu_0 + \sum_{j=1}^{w-1} (\mu_{m_j} + i_j)\right)^{i_w + 1}}$$

$$\times \frac{\Gamma\left(\mu_0 + \sum_{j=1}^{C} (\mu_{m_j} + i_j)\right)}{(B_0 + \sum_{j=1}^{C} B_{m_j})^{(\mu_0 + \sum_{j=1}^{C} (\mu_{m_j} + i_j))}} \tag{31}$$

$$+ E(K), \tag{32}$$

where $E(K)$ satisfies $\lim_{K \to \infty} E(K) = 0$.

Proof. We proof this by induction on C.

Case C=1: According to lemma 6.2 (for Q=0) we can for every given $\epsilon > 0$ find K′ such that Part A in lemma 6.1 will be below ϵ for all $K > K'$ whereas the infinite integral in Part B reduces to the Gamma function and we thereby obtain:

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \Gamma(\mu_1, B_1 x) dx$$

$$= \epsilon + \sum_{i=0}^{K-1} \frac{B_0^i B_1^{\mu_1}}{(\mu_0)^{\dot{i+1}}} \frac{\Gamma(\mu_0 + \mu_1 + i)}{(B_0 + B_1)^{(\mu_0 + \mu_1 + i)}}. \tag{33}$$

Induction step from C to $C+1$: We again apply lemma 6.1 (for Q=0) on $C+1$, and thereby obtain:

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \prod_{c=1}^{C+1} \Gamma(\mu_c, B_c x) dx \tag{34}$$

$$= \underbrace{\int_0^\infty \frac{e^{-xB_0} x^{\mu_0 + K - 1}}{(\mu_0)^{\dot{K}}} \prod_{c=1}^{C+1} \Gamma(\mu_c, B_c x) dx}_{\text{Part C+1}} \tag{35}$$

$$+ \sum_{m=1}^{C+1} \sum_{i=0}^{K-1} \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \tag{36}$$

$$\underbrace{\int_0^\infty e^{-x(B_0 + B_m)} x^{\mu_0 + \mu_m + i - 1} \times \prod_{\substack{c=1 \\ c \neq m}}^{C+1} \Gamma(\mu_c, B_c x) dx}_{\text{Part C}} \tag{37}$$

Part C+1 in the above captures the error $E_{C+1}(K)$ introduced reducing the integral involving C+1 to the integral involving C incomplete gamma functions (due to the $c \neq m$ in the sum) given in Part C in the above, and according to the induction we can assume the theorem holds for Part C up to the error term $E_C(K)$:

$$E_{C+1}(K) + \sum_{m=1}^{C+1} \sum_{i=0}^{K-1} \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \times E_C(K)$$

$$\leq E_{C+1}(K) + K(C+1) \max_{i \in \{1,\ldots,K-1\}} \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \times E_C(K). \tag{38}$$

Next we show that elements

$$Q(K) := (C+1) \max_{i \in \{1,\ldots,K-1\}} \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} \tag{39}$$

are bounded from the top when $K \to \infty$. To see this we note that it can be split into $\max_{i \leq L}(\dots)$ that is maximum over a finite set of integers lower than some $L \in \mathbb{N}$ (that always has finite upper bound) and maximum over $\{i \geq L\}$ such that maximum over this region is reached by the first index L (that follows from limit $\lim_{i \to \infty} (C+1) \frac{B_0^i B_m^{\mu_m}}{(\mu_0)^{\dot{i+1}}} = 0$ which we get from Stirling formula (25) applied on $\frac{\Gamma(\mu_0}{\Gamma(\mu_0 + i)}$ in Pochhammer symbol in (39)). So for all sufficiently large K such that $K \geq L$ we can bound $Q(K)$ by a constant we denote $Q^{(1)}$.

Notably, steps from (34) to (37) using lemma 6.1 together with formula for sum of geometric finite sum reveal that total error term of (38) including integral $E_{C+1}(K)$ and sums of $E_C(K)$ comprises not more than $\frac{1-(K(C+1))^{(C+1)}}{1-K(C+1)} = 1 + K(C+1) + (K(C+1))^2 + \cdots + (K(C+1))^C$ integrals of the same form as (23) multiplied by fractions of the same structure as (39). Using same logic as for $Q^{(1)}$ earlier that maximized fractions from 1 induction step and again leveraging Stirling formula (25), [46], we get that there exists L such that for all sufficiently large $K \geq L$ we can bound all these fractions from the top. We take their maximum, denoted $Q^{(2)}$.

Since $Q^{(1)}$ is now maximized withing $Q^{(2)}$ we can bound the total approximation error of (34) by:

$$\leq Q^{(2)} \times K \left(\frac{1 - (K(C+1))^{(C+1)}}{1 - K(C+1)} \right) \times$$

$$\max_{i \in \{0,\ldots,I\}} \left(\int_0^\infty \frac{e^{-xB_0'(i)} x^{\mu_0'(i) + K - 1}}{(\mu_0'(i))^{\dot{K}}} \prod_c \Gamma(\mu_c'(i), B_c' x(i)) dx \right)$$

where we omitted exact expression for the sake of simplicity and rather used symbolic notation instead for all combinations of parameters $\mu'(i), B'(i)$ and finite number of indexes (simplified as i).

Application of lemma 6.2 for the choice of Q such that $K(\frac{1-(K(C+1))^{(C+1)}}{1-K(C+1)}) \leq K^Q$ with each integral in max and taking $K \to \infty$ brings the limit of this upper bound to 0. That concludes the proof. □

6.2 Large Scale settings

Assuming hyperparameter priors can be chosen to be integer values following theorem presents exact evaluation of the integral (8):

Theorem 6.4. *For* $C \in \mathbb{N}^+$, $\mu_i \in \mathbb{N}^+$ *and* $B_i \in \mathbb{R}^+$, $i \in \{0, ..., C\}$ (8) *is proportional to the cumulative distribution function of the Negative Multinomial (NMn) distribution and the following equality holds:*

$$\int_0^\infty e^{-xB_0} x^{\mu_0 - 1} \prod_{i \in \{1,\dots,C\}} \Gamma(\mu_i, B_i x) dx$$

$$= \frac{\prod_{c=0}^{C} \Gamma(\mu_c)}{B_0^{\mu_0}} \times \tag{40}$$

$$\times \sum_{i_1=0}^{\mu_1-1} \cdots \sum_{i_c=0}^{\mu_c-1} \frac{\Gamma(\mu_0 + i_1 + \ldots + i_c)}{\Gamma(\mu_0) i_1! \ldots, i_c!} \left(\frac{B_0}{B}\right)^{\mu_0} \prod_{c=1}^{C} \left(\frac{B_c}{B}\right)^{i_c}, \tag{41}$$

where $B := \sum_{i=0}^{C} B_i$

Proof. Equality follows from applying (K-recurrence of $\Gamma's$) from section K-recurrence of $\Gamma's$ on $\Gamma(\mu, x) = \Gamma(1 + (\mu - 1), x)$ setting $K = \mu_i - 1$ and $a = 1$ and further simplifying as follows:

$$\frac{\Gamma(1 + (\mu - 1), x)}{\Gamma(\mu)} = \Gamma(1, x) + x e^{-x} \sum_{i=0}^{\mu-2} \frac{x^i}{\Gamma(i+2)}$$

$$= e^{-x} + e^{-x} \sum_{i=1}^{\mu-1} \frac{x^i}{\Gamma(i+1)} = e^{-x} \sum_{i=0}^{\mu-1} \frac{x^i}{\Gamma(i+1)}$$

('Euler and inc. gamma') (42)

where $a^{\dot{n}}$ is the Pochhammer symbol (a.k.a. "rising factorial") defined as $a^{\dot{n}} = \Gamma(a + n)/\Gamma(a)$. We used the fact that $\Gamma(1, x) = e^{-x}$ (trivially from definition of upper incomplete gamma $\Gamma(a, x)$).

Result follows from changing the order of integration and integrating out x. That leads to gamma functions (by definition) and gives formula to be proven. Alternatively one can recognize inner integral over x as a Laplace transform of $x^{\mu_0 + i_1 + \ldots + i_c}$. □

Note: A similar but infinite sum formula of theorem (6.4) also holds for real parameters μ. It can be shown by use of binomial series expansion $(1 + x)^\alpha, \alpha \in \mathcal{R}^+$ and Laplace transform. However, in a result we obtain infinite series.

6.3 Comparison of sampling vs. optimization

To show the advantage of using the inference method described in section 3 over an optimization heuristic as proposed in [20], we compare these two methods on the HIV-1 network (ref. 4.2). Figure 10 shows the Box-Whisker-Plots for the best obtained cut of each run/chain for each method based on the log-likelihood $p(\boldsymbol{z}|G)$. The two left Box-Whisker-Plots show the results of the optimization heuristic, whereas the third Box-Whisker-Plot shows the results for pure sampling without any optimization. In this direct comparison the optimization obtains better results. However, if the sampling method is followed by the optimization heuristic as proposed in this paper, we achieve much better results compared to pure optimization. As argued in Section 3, the optimization heuristic seems to get stuck in local maxima. This can be observed in the two left Box-Whisker-Plots. Running more optimization initiations does not help to improve the results, since each run apparently gets stuck in the same local maxima, which would explain the marginal difference of obtained best cuts in log-likelihood between 100 runs and 1000 runs. In contrary, the sampler will likely focus on some high density region of the posterior, but it will still explore multiple modes within that region, which can be seen on the wide range covered by its Box-Whisker-Plot. When these obtained best samples are subsequently used with the optimization heuristic, the obtained best cuts are significantly better than the cuts obtained using the optimization heuristic only, as to be seen in the fourth Box-Whisker-Plot. Interestingly, when using the optimization heuristic before sampling to obtain a starting point, we observe that sampling with or without a final optimization obtains the same best cuts as the pure optimization. This is most likely due to the optimization finding an extreme local maximum, as mentioned above, which even the sampler cannot escape.

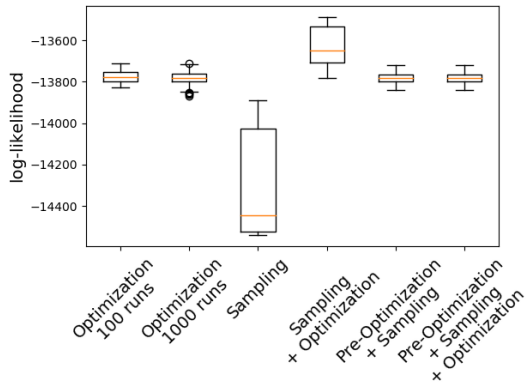
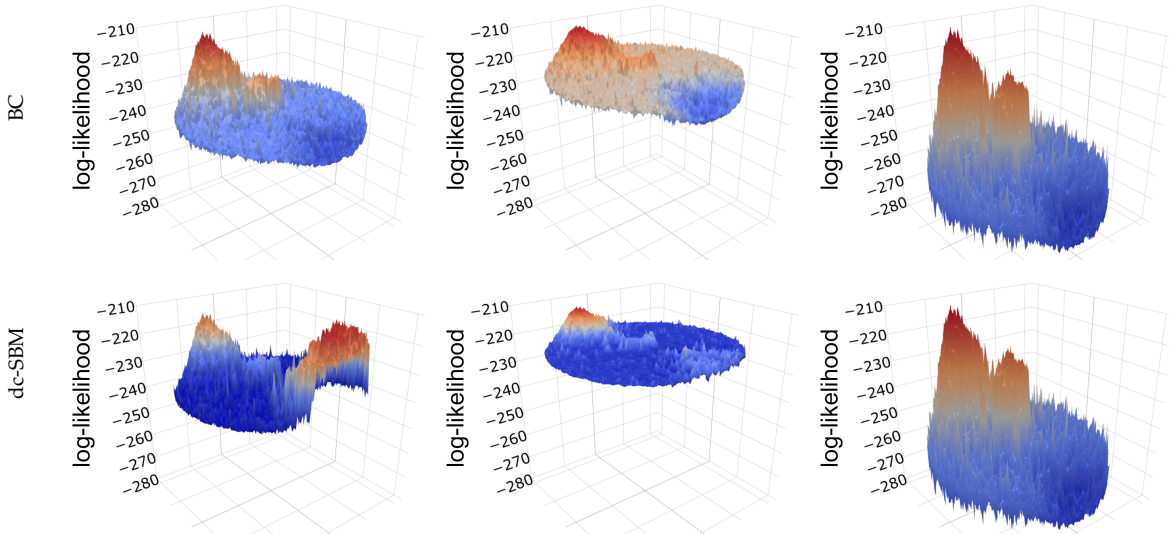


Fig. 10: Comparison of sampling and pure optimization according to [20] on the HIV-1 network through Box-Whisker-Plots of the best obtained cut of each chain/run for each method based on the log-likelihood $p(\boldsymbol{z}|G)$. The sampling results are based on 100 chains and 1000 samples per chain. Sampling followed by optimization shows superior performance compared to only using the optimization or using the optimization before sampling.

6.4 Influence of priors and constraints

In figure 11 the influence of the prior on both models considering also a weak and strongly informative prior on community structure is investigated. The non-informative prior allows the models to find their preferred modes. In this case the dc-SBM clearly exhibits the strongest support in the non-community structure region as discussed in section 4.2.1 . Even though the medium community enforcing prior

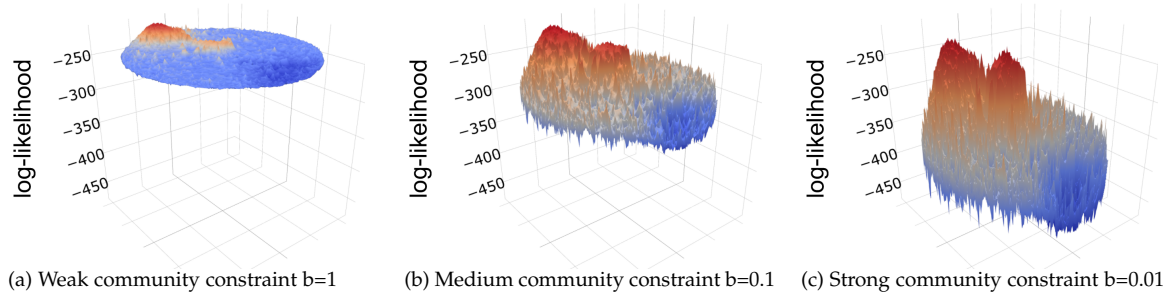


(a) Non-informative prior ($\alpha_{in} = \alpha_{out} = \beta_{in} = \beta_{out} = 0.01$)

(b) Medium community enforcing prior ($\alpha_{in} = 19.5, \alpha_{out} = 39, \beta_{in} = 144.5, \beta_{out} = 289$)

(c) Strong community enforcing prior ($\alpha_{in} = 39, \alpha_{out} = 0.01, \beta_{in} = 144.5, \beta_{out} = 289$)

Fig. 11: Comparison of different priors for α and β on the solution landscape of the Karate network ($b = 1$ for the BC model). The solution landscape was created using all posterior samples generated by running both models with 15 chains and 100 posterior samples for each configuration.



(a) Weak community constraint b=1

(b) Medium community constraint b=0.1

(c) Strong community constraint b=0.01

Fig. 12: Comparison of different constraints (b) on the solution landscape of the Karate network with non-informative prior ($\alpha_{in} = \alpha_{out} = \beta_{in} = \beta_{out} = 0.01$). The solution space is based on the samples generated in figure 11

lowers the support for the non-community structure region of the solution space, it still maintains a mode in the non-desired region. Only the strong community enforcing prior forces the dc-SBM to abandon these regions. However, imputing such a strong prior belief effectively makes the prior the posterior distribution, which is the reason that in this case the solution landscapes for both models look identical. In conclusion, the constraint imposed by BC distinguishes itself by allowing to explore the posterior with a non-informative prior while still enforcing community-structure.

The role of the b parameter, i.e. the strength of the constraint can be seen in figure 12. The lower b, the more the model enforces community structure the more it drops those regions of the solution space not supporting the community structure. Accordingly, b sets the boundaries of the constraints that can also be learned as part of the model inference.

6.5 Additional Experiment

(a) original image (b) norm cut, C=2 (c) unconstrained cut, C=2 (d) bayesian cut (BC), C=2

Fig. 13: **Coffee:** Resulting cut of BC model for C=2 (d) segments compared with unconstrained dc-SBM with shared η_{out} as well as spectral Norm cut. As experiments in the main body also these results demonstrate on par or better results of BC against referenced methods. Resulting cuts were obtained from 50 MCMC chains, 1000 samples each.

Bibliography

[Achille and Soatto, 2016] Achille, A. and Soatto, S. (2016). Information dropout: learning optimal representations through noise.

[Achille and Soatto, 2018a] Achille, A. and Soatto, S. (2018a). Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980.

[Achille and Soatto, 2018b] Achille, A. and Soatto, S. (2018b). Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2897–2905.

[Amari, 2016] Amari, S.-i. (2016). *Information geometry and its applications*, volume 194. Springer.

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.

[Banerjee et al., 2004] Banerjee, A., Dhillon, I., Ghosh, J., and Merugu, S. (2004). An information theoretic analysis of maximum likelihood mixture estimation for exponential families. In *Proceedings of the twenty-first international conference on Machine learning*, page 8.

[Banerjee et al., 2005] Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749.

[Baskerville et al., 2022] Baskerville, N. P., Granziol, D., and Keating, J. P. (2022). Appearance of random matrix theory in deep learning. *Physica A: Statistical Mechanics and its Applications*, 590:126742.

[Bengio et al., 2021] Bengio, Y., Lecun, Y., and Hinton, G. (2021). Deep learning for ai. *Communications of the Acm*, 64(7):58–65.

[Bhatia, 1997] Bhatia, R. (1997). *Matrix Analysis*. Springer New York.

[Billingsley, 1995] Billingsley, P. (1995). *Probability and measure*. Wiley.

[Bishop, 1995] Bishop, C. M. (1995). Regularization and complexity control in feed-forward networks.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[Bogachev and Ruas, 2007] Bogachev, V. I. and Ruas, M. A. S. (2007). *Measure theory*, volume 1. Springer.

[Bottou and Bousquet, 2007] Bottou, L. and Bousquet, O. (2007). The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20.

[Carmo and Flaherty, 1992] Carmo, M. P. and Flaherty, F. J. (1992). *Riemannian geometry*. Birkhäuser.

[Choromanska et al., 2015] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR.

[Cohen et al., 2021] Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. (2021). Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*.

[Daniely et al., 2016] Daniely, A., Frostig, R., and Singer, Y. (2016). Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *Advances In Neural Information Processing Systems*, 29:2253–2261.

[Devinatz, 1955] Devinatz, A. (1955). The representation of functions as a laplace-stieltjes integrals. *Duke Mathematical Journal*, 22(2):185–191.

[Du et al., 2019] Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. 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 1675–1685. PMLR.

[Du et al., 2018] Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

[Feier, 2012] Feier, A. R. (2012). *Methods of proof in random matrix theory*. PhD thesis, Harvard University.

[Frankle and Carbin, 2018] Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

[Gantmakher, 1959] Gantmakher, F. R. (1959). *The theory of matrices*, volume 131. American Mathematical Soc.

[Geiger et al., 2019] Geiger, M., Spigler, S., d'Ascoli, S., Sagun, L., Baity-Jesi, M., Biroli, G., and Wyart, M. (2019). Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115.

[Geiping et al., 2021] Geiping, J., Goldblum, M., Pope, P. E., Moeller, M., and Goldstein, T. (2021). Stochastic training is not necessary for generalization. *arXiv preprint arXiv:2109.14119*.

[Gelman et al., 2013] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian data analysis*. CRC press.

[Ghorbani et al., 2019] Ghorbani, B., Krishnan, S., and Xiao, Y. (2019). An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR.

[Giffin, 2008] Giffin, A. (2008). Maximum entropy: The universal method for inference. *Ph. D. Thesis*.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[Hanin and Rolnick, 2019] Hanin, B. and Rolnick, D. (2019). Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems*, 32.

[Hansen et al., 1993] Hansen, L. K., Pathria, R., and Salamon, P. (1993). Stochastic dynamics of supervised learning. *Journal of Physics A: Mathematical and General*, 26(1):63.

[Hansen and Salamon, 1990] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001.

[Hardt et al., 2016] Hardt, M., Recht, B., and Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234.

[Hauser and Ray, 2017] Hauser, M. and Ray, A. (2017). Principles of riemannian geometry in neural networks. *Advances in Neural Information Processing Systems 30 (nips 2017)*, 30.

[Hauser, 2018] Hauser, M. B. (2018). Principles of riemannian geometry in neural networks.

[He et al., 2020] He, F., Liu, T., and Tao, D. (2020). Why resnet works? residuals generalize. *IEEE transactions on neural networks and learning systems*, 31(12):5349–5362.

[He et al., 2019a] He, H., Huang, G., and Yuan, Y. (2019a). Asymmetric valleys: Beyond sharp and flat local minima. *Advances in neural information processing systems*, 32.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[He et al., 2019b] He, Z., Rakin, A. S., and Fan, D. (2019b). Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 588–597.

[Hinton et al., 2012a] Hinton, G., Srivastava, N., and Swersky, K. (2012a). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2.

[Hinton, 2012] Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[Hinton et al., 2012b] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[Hiriart-Urruty and Lemaréchal, 2012] Hiriart-Urruty, J.-B. and Lemaréchal, C. (2012). *Fundamentals of convex analysis*. Springer Science & Business Media.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Flat minima. *Neural computation*, 9(1):1–42.

[Huh et al., 2021] Huh, M., Mobahi, H., Zhang, R., Cheung, B., Agrawal, P., and Isola, P. (2021). The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

[Jastrzebski et al., 2021] Jastrzebski, S., Arpit, D., Astrand, O., Kerg, G. B., Wang, H., Xiong, C., Socher, R., Cho, K., and Geras, K. J. (2021). Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In *International Conference on Machine Learning*, pages 4772–4784. PMLR.

[Jastrzębski et al., 2017] Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. (2017). Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*.

[Kawaguchi, 2016] Kawaguchi, K. (2016). Deep learning without poor local minima. *arXiv preprint arXiv:1605.07110*.

[Kawaguchi et al., 2017] Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kohler et al., 2018] Kohler, J., Daneshmand, H., Lucchi, A., Zhou, M., Neymeyr, K., and Hofmann, T. (2018). Towards a theoretical understanding of batch normalization. *stat*, 1050:27.

[LeCun et al., 2012] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

[Lee, 2013] Lee, J. M. (2013). Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–31. Springer.

[Li et al., 2020a] Li, M., Soltanolkotabi, M., and Oymak, S. (2020a). Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4313–4324. PMLR.

[Li et al., 2017] Li, Q., Tai, C., and Weinan, E. (2017). Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pages 2101–2110.

[Li et al., 2019] Li, Q., Tai, C., and Weinan, E. (2019). Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. *J. Mach. Learn. Res.*, 20:40–1.

[Li, 2010] Li, S. (2010). Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70.

[Li et al., 2020b] Li, X., Gu, Q., Zhou, Y., Chen, T., and Banerjee, A. (2020b). Hessian based analysis of sgd for deep nets: Dynamics and generalization. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 190–198. SIAM.

[Lian and Liu, 2019] Lian, X. and Liu, J. (2019). Revisit batch normalization: New understanding and refinement via composition optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3254–3263. PMLR.

[Liang et al., 2019] Liang, T., Poggio, T., Rakhlin, A., and Stokes, J. (2019). Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 888–896. PMLR.

[Liu et al., 2019] Liu, S., Papailiopoulos, D., and Achlioptas, D. (2019). Bad global minima exist and sgd can reach them. *arXiv preprint arXiv:1906.02613*.

[Luo et al., 2018] Luo, P., Wang, X., Shao, W., and Peng, Z. (2018). Towards understanding regularization in batch normalization. *arXiv preprint arXiv:1809.00846*.

[Malach et al., 2020] Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. (2020). Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR.

[Mandt et al., 2017] Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*.

[Minka, 2001] Minka, T. P. (2001). *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology.

[Mohri et al., 2012] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). Foundations of machine learning.[sl].

[Myland et al., 2008] Myland, J., Myland, Jan, C., Spanier, J., Oldham, K., and Oldham, Keith, B. (2008). The incomplete beta function b(v,,x). *An Atlas of Functions*, pages 603–609.

[Needham, 2021] Needham, T. (2021). *Visual Differential Geometry and Forms: A Mathematical Drama in Five Acts*. Princeton University Press.

[Neelakantan et al., 2016] Neelakantan, A., Vilnis, L., Le, Q. V., Kaiser, L., Kurach, K., Sutskever, I., and Martens, J. (2016). Adding gradient noise improves learning for very deep networks.

[Neyshabur et al., 2015] Neyshabur, B., Tomioka, R., and Srebro, N. (2015). Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401. PMLR.

[Noh et al., 2017] Noh, H., You, T., Mun, J., and Han, B. (2017). Regularizing deep neural networks by noise: Its interpretation and optimization. *Advances in Neural Information Processing Systems*, 30.

[Parlett, 1998] Parlett, B. N. (1998). *The symmetric eigenvalue problem*. SIAM.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[Pawitan, 2004] Pawitan, Y. (2004). *In all likelihood : statistical modelling and inference using likelihood*. Clarendon Press.

[Poole et al., 2014] Poole, B., Sohl-Dickstein, J., and Ganguli, S. (2014). Analyzing noise in autoencoders and deep networks. *arXiv preprint arXiv:1406.1831*.

[Raghu et al., 2017] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR.

[Rashid et al., 2019] Rashid, S., Noor, M. A., and Noor, K. I. (2019). New estimates for exponentially convex functions via conformable fractional operator. *Fractal and Fractional*, 3(2):19.

[Rasmussen, 2003] Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Roberts, 2021] Roberts, D. A. (2021). Sgd implicitly regularizes generalization error. *arXiv preprint arXiv:2104.04874*.

[Roman et al., 2005] Roman, S., Axler, S., and Gehring, F. (2005). *Advanced linear algebra*, volume 3. Springer.

[Rousseau and Fablet, 2018] Rousseau, F. and Fablet, R. (2018). Residual networks as geodesic flows of diffeomorphisms. *arXiv preprint arXiv:1805.09585*.

[Salakhutdinov and Hinton, 2007] Salakhutdinov, R. and Hinton, G. (2007). Deep belief networks.

[Salimans and Kingma, 2016] Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29:901–909.

[Santurkar et al., 2018] Santurkar, S., Tsipras, D., Ilyas, A., and Mądry, A. (2018). How does batch normalization help optimization? In *Proceedings of the 32nd international conference on neural information processing systems*, pages 2488–2498.

[Smith et al., 2021] Smith, S. L., Dherin, B., Barrett, D. G., and De, S. (2021). On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR.

[Taborsky et al., 2021] Taborsky, P., Vermue, L., Korzepa, M., and Morup, M. (2021). The bayesian cut. *Ieee Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4111–4124.

[Tishby et al., 2000] Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

[Tishby and Zaslavsky, 2015] Tishby, N. and Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE.

[Tu, 2011] Tu, L. W. (2011). Manifolds. In *An Introduction to Manifolds*, pages 47–83. Springer.

[Tu, 2017] Tu, L. W. (2017). *Differential geometry: connections, curvature, and characteristic classes*, volume 275. Springer.

[vanRossum, 1995] vanRossum, G. (1995). Python reference manual. *Department of Computer Science [CS]*, (R 9525).

[Veit et al., 2016] Veit, A., Wilber, M. J., and Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29.

[Vershynin, 2018] Vershynin, R. (2018). *High dimensional probability. An introduction with applications in Data Science.* Cambridge University Press.

[Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

[Wainwright and Jordan, 2008] Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference.

[Xie et al., 2020] Xie, Z., Sato, I., and Sugiyama, M. (2020). A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. *arXiv e-prints*, pages arXiv–2002.

[Yao et al., 2020] Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. (2020). Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pages 581–590. IEEE.

[Yoshida and Miyato, 2017] Yoshida, Y. and Miyato, T. (2017). Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*.

[Zhang et al., 2016] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

[Zhang et al., 2021] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

[Zhu et al., 2019] Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. (2019). The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *ICML*, pages 7654–7663.

[Zou et al., 2021] Zou, D., Cao, Y., Li, Y., and Gu, Q. (2021). Understanding the generalization of adam in learning neural networks with proper regularization. *arXiv e-prints*, pages arXiv–2108.