UNIVERSITY OF CALIFORNIA
Los Angeles

A Picture of the Energy Landscape of Deep Neural Networks

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Pratik Anil Chaudhari

2018

ABSTRACT OF THE DISSERTATION

A Picture of the Energy Landscape of Deep Neural Networks

by

Pratik Anil Chaudhari

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2018

Professor Stefano Soatto, Chair

This thesis characterizes the training process of deep neural networks. We are driven by two apparent paradoxes. First, optimizing a non-convex function such as the loss function of a deep network should be extremely hard, yet rudimentary algorithms like stochastic gradient descent are phenomenally successful at this. Second, over-parametrized models are expected to perform poorly on new data, yet large deep networks with millions of parameters achieve spectacular generalization performance.

We build upon tools from two main areas to make progress on these questions: statistical physics and a continuous-time point-of-view of optimization. The former has been popular in the study of machine learning in the past and has been rejuvenated in recent years due to the strong correlation of empirical properties of modern deep networks with existing, older analytical results. The latter, i.e., modeling stochastic first-order algorithms as continuous-time stochastic processes, gives access to powerful tools from the theory of partial differential equations, optimal transportation and non-equilibrium thermodynamics.

The confluence of these ideas leads to fundamental theoretical insights that explain observed phenomena in deep learning as well as the development of state-of-the-art algorithms for training deep networks.

The dissertation of Pratik Anil Chaudhari is approved.

Arash Ali Amini

Stanley J Osher

Fei Sha

Ameet Shrirang Talwalkar

Stefano Soatto, Committee Chair

University of California, Los Angeles

2018

To my parents, Jyotsna and Anil

TABLE OF CONTENTS

List of Figures

List of Tables

Acknowledgments

My first thanks go to my advisor Stefano Soatto. I have been extremely lucky to work with him for the past four years and have benefited immensely from his open-mindedness, generosity and laser-sharp focus. Members of my thesis committee have provided helpful advice on the material discussed in this thesis: Stan Osher, Ameet Talwalkar, Arash Amini and Fei Sha. Emilio Frazzoli and Sertac Karaman have been crucial in shaping my thinking in the early stages of my research career at MIT. Hemendra Arya supported my journey into robotics with enthusiasm at IIT Bombay.

I have been extremely fortunate to interact and work with numerous researchers over the past few years, their vision has inspired me and their mentorship has added new tools to my repertoire: Adam Oberman, Stan Osher, Riccardo Zecchina, Carlo Baldassi, Anna Choromanska, Yann LeCun, Jennifer Chayes, Christian Borgs, Yuval Peres, Rene Vidal, Anima Anandkumar and Stephane Mallat. Karl Iagnemma at nuTonomy has been a mentor and well-wisher.

UCLA has been an intellectually inspiring, fertile and extremely enjoyable environment for me. For this, I would like to thank my office-mates over the years: Alhussein Fawzi, Konstantine Tsotsos, Nikos Karianakis, Alessandro Achille, Virginia Estellers and others from the UCLA Vision Lab, as also the regulars at the Level Set Seminar series.

My friends, from Los Angeles, Boston, and from India, have a special place in my heart. Graduate school so far away from home is not easy and this experience was made joyful and memorable by their laughter.

My family has been an unconditional source of strength and encouragement throughout my graduate studies. No words can do justice to the sacrifices my parents have made to set me up on this path.

The research presented in this thesis was supported by the Balu and Mohini Balakrishnan Fellowship and grants from ONR, AFOSR, ARO and AWS. Their support is gratefully acknowledged.

Vitæ

2006 - 2010	Bachelor of Technology, Aerospace Engineering, Indian Institute of Technology Bombay, India.
2010 - 2012	Master of Science, Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA.
2012 - 2014	Engineer, Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA.
2014 - 2016	Principal Autonomous Vehicle Engineer, nuTonomy Inc., Cambridge, MA.
2014 - present	Research Assistant, Computer Science Department, University of California, Los Angeles, CA.

CHAPTER 1

Introduction

Why study deep networks instead of, say, building new applications with them? Notwithstanding the enormous potential they have shown in almost every field they have touched in the past half-decade (LeCun et al., 2015), translating this potential into real-world applications, or even simply replicating these results, is—today—a dark art. That deep networks work is beyond debate but why they work is almost a mystery, and it is the latter that we wish to focus upon. This thesis has two goals: (i) capturing this art as the basic principles of training deep networks and, (ii) building optimization algorithms that leverage upon this understanding to push the envelope of practice.

What tools enable this study? The techniques to be seen in the remainder of this thesis are highly unusual in computer science. Ideas from statistical physics of disordered systems help characterize the hardness of the optimization problem in deep learning. Physics of learning algorithms is used to new develop new loss functions for deep learning that not only aid optimization but also improve the prediction performance on new data. These ideas from physics have deep connections to the theory of partial differential equations, stochastic control and convex optimization. The continuous-time point-of-view of stochastic first-order optimization algorithms undertaken in the above analysis is further exploited using ideas from optimal transportation and non-equilibrium thermodynamics, these in turn are connected back to known results in Bayesian inference and information theory. These techniques, diverse in their reach and esoteric in their nature, lead to state-of-the-art algorithms for training deep networks and that is our answer to questions of "why should I study these tools" from the reader.

With this background, we next provide a statement of our technical contributions. Section 1.2 discusses the organization of the manuscript.

1.1. Statement of contributions

The first contribution is to model the loss function of a deep network as a spin glass Hamiltonian. This enables us to compute the statistics of the critical points, their location in the energy landscape and a detailed analysis of the topology of the energy landscape at low temperatures. Under this model, the energy landscape of a deep network is expected to have exponentially many critical points at all energy levels. We use the technique of topology trivialization to modulate this complexity of such a landscape using a magnetic field. Sharp thresholds for the magnitude of the magnetic field are available which leads to a technique called Trivialized-SGD for training a deep network.

Our next contribution is a new optimization algorithm called Entropy-SGD for training deep neural networks that is motivated by the local geometry of the energy landscape. Empirically, regions with low generalization error have a large proportion of almost-zero eigenvalues in the Hessian with very few positive or negative eigenvalues. We devise a loss function called Local Entropy that leverages upon this observation and biases optimization towards such regions. This loss can be minimized using techniques from Markov chain Monte Carlo and Entropy-SGD resembles two nested loops of stochastic gradient descent (SGD). We show that the new objective has a smoother energy landscape and show improved generalization over SGD using uniform stability, under certain assumptions.

The third contribution is to observe that Local Entropy which was motivated from statistical physics is equivalent to smoothing using the viscous Hamiltonian-Jacobi partial differential equation. This gives a

stochastic optimal control interpretation that shows that a modified version of Entropy-SGD converges faster than SGD. Well-established PDE regularity results allow us to analyze the geometry of the relaxed energy landscape. Stochastic homogenization theory allows us to connect Entropy-SGD with popular distributed algorithms such as Elastic-SGD.

The fourth contribution is an algorithm called Parle for parallel training of deep networks that converges 2-4$\times$ faster than a data-parallel implementation of SGD, while achieving significantly improved error rates that are nearly state-of-the-art on several benchmarks including CIFAR-10 and CIFAR-100, without introducing any additional hyper-parameters. We exploit the phenomenon of wide minima that has been shown to lead to improved generalization error for deep networks. Parle requires very infrequent communication with the parameter server and instead performs more computation on each client, which makes it well-suited to both single-machine, multi-GPU settings and distributed implementations.

The fifth contribution is to prove a number of widely accepted yet unproven properties of stochastic gradient descent. We show that SGD minimizes an average potential over the posterior distribution of weights along with an entropic regularization term. This potential is however not the original loss function in general. In other words, SGD perform variational inference, but for a different loss than the one used to compute the gradients. More surprisingly, most likely trajectories of SGD resemble closed loops in the parameter space instead of Brownian motion around critical points. We prove that such "out-of-equilibrium" behavior is a consequence of highly non-isotropic gradient noise in SGD; the covariance matrix of mini-batch gradients for deep networks has a rank as small as 1% of its dimension.

Throughout, the theory developed in this thesis is validated through experiments on prototypical deep networks. The material presented in this thesis has been disseminated through the following articles:

1. Pratik Chaudhari, and Stefano Soatto, On the energy landscape of deep networks. arXiv:1511.06485, 2015.
2. Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: biasing gradient descent into wide valleys. In Proc. of the International Conference of Learning and Representations (ICLR), arXiv:1611.01838, 2017.
3. Pratik Chaudhari, Adam Oberman, Stanley Osher, Stefano Soatto and Guillaume Carlier. Deep Relaxation: partial differential equations for optimizing deep neural networks. Accepted for Research in the Mathematical Sciences, arXiv:1704.04932, 2018.
4. Pratik Chaudhari, Carlo Baldassi, Riccardo Zecchina, Stefano Soatto, Ameet Talwalkar and Adam Oberman. Parle: parallelizing stochastic gradient descent. arXiv:1707.00424, 2017.
5. Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In Proc. of the International Conference of Learning and Representations (ICLR), arXiv:1710.11029, 2017.

1.2. READING OF THE THESIS

This thesis is divided into three parts.

The first part in Chapter 2 develops a model for a prototypical deep network. This model is motivated from the theory of spin glasses in statistical physics and suggests that training a deep network is an extremely difficult optimization problem: the energy landscape of deep networks has exponentially many critical points at all energy levels. This chapter then devises techniques from topology trivialization to reduce the complexity and make this landscape more accessible to first order algorithms.

Chapters 3 to 5 form the second part of the thesis. The aim is to develop algorithms that build upon and amplify empirically observed properties of stochastic gradient descent for deep networks. Chapter 3 devises a loss function called Local Entropy and discusses a Markov chain Monte Carlo techniques to minimize this modified loss function. Chapter 4 builds upon a simple, yet powerful observation about Local Entropy, the fact

that it is connected to smoothing using a certain Hamilton-Jacobi partial differential equation. This observation leads to connections with stochastic optimal control, proximal point iteration and new algorithms for training deep networks. Chapter 5 exploits the fact that 1-step replica symmetry breaking as performed by the MCMC algorithm to minimize Local Entropy is equivalent to training an ensemble of neural networks in parallel with proximal forces attracting them to each other. This leads to a state-of-the-art algorithm for distributed training of deep networks called Parle.

Chapter 6 forms the third part of this thesis. It develops a continuous-time model for stochastic gradient descent and characterizes the steady-state distribution of the underlying Markov process. This chapter is theoretical in nature and elucidates a number of—widely believed yet never proven—properties of stochastic first-order algorithms. In particular, it shows that SGD has an explicit regularization term that stems from its stochasticity, an inductive bias that leads to solutions that are different from the critical points of the original loss function and that SGD converges to tail phenomena such as limit cycles; these are marked characteristics of non-equilibrium systems in statistical physics. These results are connected to the generalization performance of similar large deviations loss functions like Local Entropy and acceleration of MCMC algorithms by breaking detailed balance.

Finally, in Chapter 7 we discuss directions for future development of the ideas explored in this thesis.

1.2.1. Notation

When in doubt, the reader is advised to consult this section to clarify notation used throughout this thesis. However, overloading of notation has been difficult to avoid, we will duly remind the reader of such instances in the narrative. The chapters of this thesis are mostly self-contained in their material. We have made the effort to keep the notation concise, sometimes, at the cost of introducing non-standard usage.

x, y, z	parameters or weights of a neural network
X, Y, Z	random variables
d	dimensionality of parameter space
N	size of the training set
Ξ	training dataset, e.g., images, text etc.
$\xi \in \Xi$	sample from the training dataset
μ or Y	ground-truth labels for data in Ξ
n	number of replicas (copies of the parameter space)
$a, b \ldots$	index of the replica
$\mathcal{b}$	overloaded notation that denotes both the mini-batch size and the set of indices that form the mini-batch
$\nabla f_k(x)$	gradient of the loss function on the k^{th} datum, i.e., with $\mathcal{b} = \{k\}$
$\nabla f_{\mathcal{b}}(x)$	average gradient of the loss function on the mini-batch $\mathcal{b}$, i.e., $\nabla f_{\mathcal{b}}(x) = \mathcal{b}^{-1} \sum_{k \in \mathcal{b}} \nabla f_k$
∇	gradient operator in Euclidean space
η	step-size/learning rate in first-order algorithms
γ	scoping parameter of local entropy, bandwidth of the Gaussian smoothing and the time scale for the partial differential equations
$\nabla \cdot$	divergence operator
Δ	Laplacian operator, this is equal to the composition of the gradient with the divergence: $\nabla \cdot \nabla$
$\mathscr{P}(\Omega)$	probability distributions supported on $\Omega \subseteq \mathbb{R}^d$

$\mathscr{L}^2(\Omega)$	$\mathscr{L}^2$ integrable functions, usually probability densities supported on $\Omega \subseteq \mathbb{R}^d$
$W_2^2(\Omega)$	2-Wasserstein metric on Ω
$H(\rho)$	Shannon entropy of the probability distribution $\rho \in \mathscr{P}(\Omega)$, this is defined as $H(\rho) = -\int \mathrm{d}\rho \log \rho$
$\mathrm{KL}(p \,\|\| \, q)$	Kullback-Leibler divergence between two distributions $p, q \in \mathscr{P}(\Omega)$ defined as $\mathrm{KL}(p \,\|\| q) = \int \mathrm{d}p \, \log \frac{p}{q}$

CHAPTER 2

What is the shape of the energy landscape of deep neural networks?

There are three primary kind of analysis one would like to perform in machine learning: (i) the richness of the model class that the function represents, (ii) the hardness of the optimization problem while training this model and, (iii) the generalization capability of models of its class for new data. Deep networks are quite opaque objects for the purposes of such analysis: while their capability of being universal function approximators is very well-known, we do not have a good theoretical understanding of the latter two problems.

This chapter proposes a model that lends insight into the computational hardness of the optimization process for deep networks. Our goal is to compare and contrast known empirical and theoretical results for deep networks with those of other prototypical hard non-convex optimization problems—with an aim to see where contemporary models fall short. We are motivated by the analysis of spin glasses which, incidentally, were among the first models for neural networks (Hopfield, 1982). Techniques from statistical physics first devised for studying spin glasses have been extended to combinatorial optimization problems such as random k-SAT, LDPC codes and compressed sensing (Mezard and Montanari, 2009). Over the past decade, they have led to an elaborate understanding of the energy landscape of these problems and have also resulted in state-of-the-art algorithms (Krzakala et al., 2012; Braunstein et al., 2005). The motivation to study this is model is also one of feasibility, in spite of a high-dimensional non-convex landscape, we can compute global quantities such as the expected number of local minima (Auffinger et al., 2013) and the structure of the Gibbs distribution (Talagrand, 2003) for spin glasses; this is a much harder task for other models where the results are of a local nature.

We consider a model for deep networks with random, sparse weights; we show that the loss function of such a network resembles the Hamiltonian of a spin glass which is a well-studied model in statistical physics (Talagrand, 2003). Existing results for this model point to a complex energy landscape with exponentially many local minima and saddle points (Auffinger et al., 2013). Empirically, this manifests in deep networks as sensitivity of the training procedure to initialization and learning rates (Sutskever et al., 2013; Singh et al., 2015) or even divergence for complex architectures (Cai et al., 2016). The energy landscape also has a layered structure, i.e., high-order saddle points with many negative eigenvalues of the Hessian lie near the surface.

This understanding can help design new algorithms for optimization. An effective way of modifying a hard energy landscape is to perturb the Hamiltonian by a random magnetic field (Fyodorov, 2013) or another uncorrelated Hamiltonian (Talagrand, 2010b). In this chapter, we employ the former. We show that there exists a critical threshold of the perturbation below which the exponential regime persists and above which the landscape trivializes to leave only one local minimum. This phase transition is not sharp, i.e., that there exists a small band around the perturbation threshold that results in polynomially many local minima. Using a control parameter, we can thus smoothly transition between the original complex landscape and a completely degenerate loss function. Moreover, we prove that such an annealing does not change the locations of the local minima of the original problem in the low-temperature regime. We motivate an algorithm called Trivialized-SGD that builds upon this phenomenon to demonstrate good empirical results on modern deep networks.

2.1. A MODEL FOR DEEP NETWORKS

Modified notation in this chapter: The notation of this chapter is slightly different from the chapters that follow. We have tried to remain consistent with standard notation in the theory of spin glasses in order to be

able to compare relevant literature easily. In particular, the dimensionality of the optimization problem will be N in this chapter as opposed to d in the rest of the thesis. The notation x^k denotes the weights of the k^{th} layer of a neural network. The number of layers is denoted by p and d is the number of non-zero weights that connect each neuron to the layer below it.

2.1.1. Sparse, random deep networks

Let us consider a fully-connected deep network with p hidden layers and N neurons on each layer. We assume that binary data $\xi \in \{0,1\}^N$ is generated by a binary feature vector $h \in \{0,1\}^N$. More precisely,

$$\xi = g\left(x^{1^\top}\, g\left(x^{2^\top} \dots\, g\left(x^{p^\top}\, h\right)\dots\right)\right); \tag{2.1}$$

where $x^k \in [-1,\ 1]^{N\times N}$ for $k \le p$ and $g(x) = \mathbf{1}_{\{x\ge 0\}}$ are threshold nonlinearities. $\mathbf{1}_A$ denotes the indicator function of the set A. The output of the intermediate layers is given by

$$h^{k-1} = g\left(x^{k^\top}\, g\left(x^{k+1^\top} \dots\, g\left(x^{p^\top}\, h\right)\dots\right)\right)$$

where $h^{k-1} \in \{0,1\}^N$. With this notation, we have $\xi = h^0$ and $h = h^p$. To make this model analytically tractable, we make the following assumptions:

(i) h has ρ non-zero entries and $\rho\left(\frac{d}{2}\right)^p$ is a constant;
(ii) for $d = N^{1/p}$, every entry x^k_{ij} is an iid zero-mean random variable with

$$\mathbb{P}(x^k_{ij} > 0) = \mathbb{P}(x^k_{ij} < 0) = \frac{d}{2N}$$

and zero otherwise. This results in an average d non-zero input or output synapses for every neuron;
(iii) the distribution of x^k_{ij} is supported on a finite set of size $\Theta(N)$ is not all concentrated near zero.

If every entry of the hidden vector h is an indicator of some class, the first assumption implies that at most ρ classes are present in every data sample ξ. The Chernoff bound then implies that each datum ξ has $\rho\left(\frac{d}{2}\right)^p$ fraction non-zero entries with high probability, which we assume to be constant. The third assumption on the support of the weight distribution ensures that the weights of the deep network are well-conditioned, the reason for this assumption will be elucidated in the sequel.

This model enjoys a number of useful properties by virtue of it being random and sparsely connected. For high sparsity ($d < N^{1/5}$ or equivalently, $p > 5$), the network exhibits weight-tying, a technique popular in training denoising auto-encoders, whereby the hidden layer can be computed (w.h.p) by running the network in reverse:

$$h = g\left(x^p g\left(x^{p-1} \dots\, g\left(x^1 \xi - \frac{d}{3}\mathbb{1}_N\right) - \frac{d}{3}\mathbb{1}_N\right)\dots\right); \tag{2.2}$$

where $\mathbb{1}_N$ is a vector of ones (Arora et al., 2013). We exploit this property in the following section to connect the loss of such a network to the spin glass Hamiltonian.

We will consider a typical binary classification setting: a support vector machine (SVM) or a soft-max layer is used on the feature vector h to obtain an output $Y \in \{0,1\}$. We model this as

$$Y = g\left(x^{p+1} h - \frac{d}{3}\mathbb{1}_N\right). \tag{2.3}$$

2.1.2. Deep networks as spin glasses

A p-spin glass for an integer $p \ge 1$ with N spins is given by an energy function, also known as the Hamiltonian,

$$-H_{N,p}(\sigma) = \frac{1}{N^{(p-1)/2}} \sum_{i_1,i_2,\dots,i_p=1}^{N} J_{i_1,\dots,i_p}\sigma_{i_1}\dots\sigma_{i_p}; \tag{2.4}$$

where $\sigma = (\sigma_1, \ldots, \sigma_N) \in \mathbb{R}^N$ is a configuration of the spins and the disorder $J_{i_1,\ldots,i_p}$ are iid standard Gaussian random variables. Note that $H_{N,p}(\sigma)$ is zero-mean and Gaussian for every σ. The term $N^{(p-1)/2}$ is a normalizing constant that ensures that the Hamiltonian scales as $\Theta(N)$. In spin glass computations, for ease of analysis one typically assumes a spherical constraint on the spins, viz., $\sum_i \sigma_i^2 = N$ or in other words, $\sigma \in S^{N-1}(\sqrt{N})$ which is the $(N-1)$-dimensional hypersphere of radius $\sqrt{N}$.

We now connect the model for deep networks in Section 2.1.1 to the Hamiltonian of a p-spin mean-field glass. Consider the classification model of (2.2). We first get rid off the non-linearities and write (2.2) and (2.3) in terms of the "active paths", i.e., all paths with non-zero activations from an input neuron ξ_i to the output Y. We then have

$$Y = \sum_{i=1}^{N} \sum_{\gamma \in \Gamma_i} \xi_i \, x_\gamma; \tag{2.5}$$

where Γ_i is the set of all active paths connecting the i^{th} input neuron to the output Y. The term x_γ is the product of the weights along each path γ. Now define $\gamma_{i,i_1,\ldots,i_p}$ to be an indicator random variable that is 1 iff the path $i \to i_1 \to \ldots \to i_p \to Y$, i.e., the one that connects the i_1^{th} neuron in the first hidden layer, i_2^{th} neuron in the second hidden layer and so on, is active. With some abuse of notation, we will use the variable γ to denote both the path and the tuple $(i, i_1, i_2, \ldots, i_p)$.

With an aim to analyze Y when data and labels are drawn from a generic data distribution, we now work towards removing the dependency of the input ξ in (2.5). First split the path into two parts, the first edge on layer 1 and the rest of the path as: $\gamma_{i,i_1,i_2,\ldots,i_p} = \gamma_{i,i_1} \, \gamma_{i_1,\ldots,i_p}$ (conditional upon $\gamma_{i,i_1} = 1$, the sub-paths are independent). This gives

$$Y = \sum_{i,i_1,i_2,\ldots,i_p=1}^{N} \left(\gamma_{i,i_1} \, \xi_i \, x_{i,i_1}\right) \left(\gamma_{i_1,\ldots i_p} \, x_{i_1,\ldots,i_p}\right).$$

This expression has correlations between different paths that are introduced by the sum over inputs ξ_i which we approximate next. Define the net coupling due to the sum over the inputs as

$$Z_i := \xi_i \, \gamma_{i,i_1} \, x_{i,i_1}, \quad Z := \sum_{i=1}^{N} Z_i.$$

Since $\xi_i \in \{0,1\}$, we have var $(X_i) \leq 1/4$ (the variance is maximized for a Bernoulli random variable with probability $1/2$), and according to our assumptions, $x_{i,i+1}$ is positive or negative with equal probability $d/(2n)$ while $\gamma_{i,i_1} = 1$ with probability d/n for a fixed i_1. This gives

$$\mathbb{E}\, Z_i = 0 \quad \text{and} \quad \mathbb{E}\,(Z_i^2) \leq \frac{d^2}{4N^2}.$$

We now use Bernstein's inequality to see that:

$$P\left(\left|\sum_{i=1}^{N} Z_i\right| > t\right) \leq \exp\left(-\frac{t^2/2}{\text{var}\,(Z) + t/3}\right)$$
$$\Rightarrow |Z| \leq N^{-\frac{1}{2}+\frac{1}{p}+\varepsilon} \leq N^{-1/5} \qquad \text{for } p > 5;$$

with high probability for any small $\varepsilon > 0$ (we picked $\varepsilon = 1/5$ above). This also gives var $(Z) \leq \frac{N^{-2/5}}{4}$. We now get a lower bound on var (Z) using Chebyshev's inequality where we use the assumption that the weight distribution is not all concentrated around zero. In particular, let us assume that $\mathbb{P}\left(\left|x_{ij}^k\right| > t\right) \geq t$ for some small $t = N^{-2/15}$; note that the special form $\mathbb{P}(x > t) \geq t$ is only for convenience. This gives var$(x) \geq t^3 = N^{-2/5}$ and finally

$$\frac{N^{-2/5}}{4} \geq \text{var}\,(Z) \geq \mathbb{E}\,(\xi^2)\, N^{-2/5+1/p}. \tag{2.6}$$

We have thus shown that the net coupling due to inputs in our model is bounded and has variance of the order $\Theta(N^{-2/5})$.

We now compute the probability of a path $\gamma_{i_1,\ldots,i_p}$ being active. Note that hidden units on the k^{th} layer are uniform, but because of correlations of the layers above, they are not simply Bernoulli with probability $\frac{\rho}{N}\left(\frac{d}{2}\right)^{p-k}$. However, for networks that are at most $p=\mathscr{O}(\log_d n)$ deep, we can bound the probability of the ℓ^{th} neuron being active on the k^{th} layer (Arora et al., 2013, Proof of Lemma 6):

$$\mathbb{P}\left(h_\ell^k=1\right)\in\left[\frac{3\rho}{4n}\,\left(\frac{d}{2}\right)^{p-k},\frac{5\rho}{4n}\,\left(\frac{d}{2}\right)^{p-k}\right];$$

which gives $P\left(\gamma_{i_1,\ldots,i_p}=1\right)=\Theta(N^{-(p-1)/2})$. The random variable Z $\gamma_{i_1,\ldots i_p}$ is zero mean with variance of the order $\Theta(N^{-(p-1)/2-2/5})$ and we therefore replace it with a zero-mean standard Gaussian $K_{i_1,\ldots,i_p}$

$$Y_J\overset{law}{=}\frac{J}{N^{(p-1)/4+1/5}}\sum_{i_1,i_2,\ldots,i_p=1}^{N}K_{i_1,\ldots,i_p}x_{i_1,\ldots,i_p},\tag{2.7}$$

for some constant $J>0$.

Since the entries of weight matrices x^k are iid, the weights along each path $\gamma\in\Gamma_i$, i.e., $x_{i_1,\ldots,i_p}$ are independent and identically distributed. Note that there are at most $\mathscr{O}(m^p)$ distinct values that $x_{i_1,\ldots,i_p}$ can take since each $x_{i_k,i_{k+1}}$ is supported over a set of cardinality $m=\Theta(N)$ in the interval $[-1,\ 1]$. For $\sigma_{l_k}\in\text{supp}(x_{i_k,i_{k+1}})$ with $l_k\leq N$, we now write $x_{i_1,\ldots,i_p}$ as a sum of weighted indicator variables to get

$$x_{i_1,\ldots,i_p}=\sum_{l_1,\ldots,l_p=1}^{m}\mathbf{1}\left\{x_{i_1,i_2}=\sigma_{l_1},\ldots,x_{i_p,Y}=\sigma_{l_p}\right\}\,\left(\prod_{k=1}^{p}\sigma_{l_k}\right),$$

to get

$$\begin{aligned}Y_J&=\frac{J}{N^{(p-1)/4+1/5}}\sum_{l_1,\ldots,l_p=1}^{m}\sum_{i_1,i_2,\ldots,i_p=1}^{N}K_{i_1,\ldots,i_p}\mathbf{1}\left\{x_{i_1,i_2}=\sigma_{l_1},\ldots,x_{i_p,Y}=\sigma_{l_p}\right\}\,\left(\prod_{k=1}^{p}\sigma_{l_k}\right),\\&\triangleq\frac{J}{N^{(p-1)/4+1/5}}\sum_{l_1,\ldots,l_p=1}^{m}J_{l_1,\ldots,l_p}\left(\prod_{k=1}^{p}\sigma_{l_k}\right);\end{aligned}$$

Note that $J_{l_1,\ldots,l_p}$ is a zero-mean Gaussian with variance $(n/m)^p$ and we can write the scaled output $\widehat{Y}=\frac{N^{-(p-3)/4+1/5}}{\sqrt{m}}\,Y_J$ after a renaming of indices as

$$\widehat{Y}\overset{\text{law}}{=}\frac{J}{m^{(p-1)/2}}\sum_{i_1,i_2,\ldots,i_p=1}^{m}J_{i_1,\ldots,i_p}\,\sigma_{i_1}\ldots\sigma_{i_p}.\tag{2.8}$$

Since each σ_i has the same distribution as x_{ij}^k, the p-product $\sigma_{i_1}\ldots\sigma_{i_p}$ can take at most $\binom{N}{p}$ distinct values for every configuration σ. Using Stirling's approximation, we can see that this is also of the order $\mathscr{O}(m^p)$ for $p=\mathscr{O}(\log N)$ and $m=\Theta(N)$, i.e., if the discretization of weights grows with the number of neurons we do not lose any representational power by replacing the weights by spins. In the sequel, we will simply set $m=N$ the formulas undergo only minor changes if $m=\Theta(N)$.

Let us now deal with the loss function. We do not have a model for the true labels Y^t which presents a hurdle while approximating the loss. Since $\widehat{Y}\in\{-1,1\}$, the zero-one loss is given by

$$\ell(\widehat{Y})=\begin{cases}1-\widehat{Y} & \text{if } Y^t=1\\ 1+\widehat{Y} & \text{else.}\end{cases}$$

disregarding a scalar multiplier of $1/2$. We now make the following observation: we can set all the labels in the dataset are $Y^t=1$. In other words, if there exists a datum with label $Y^t=-1$, we can flip the signs on the corresponding input ξ so that the label becomes $Y^t=1$ for that sample. It is easy to see that this is true for the case of a single layer neural network and we can show that it remains so under the sparsity assumption. This

gives the loss

$$H_{N,p} \triangleq L = 1 - \widehat{Y}.$$

The Hamiltonian has the same distribution as $\widehat{Y}$ and we will disregard the constant henceforth.

Let us emphasize that the "spins" resulting from the above approximations are the quantization of weights of a deep network on a set $[-1,1]$. This quantization is currently necessary to map the $\mathscr{O}(N^p)$ distinct paths x_γ into the set of p-products of N spins $(\sigma_1,\ldots,\sigma_N)$. For the sake of building intuition in the sequel, one can simply think of σ as the real-valued weights themselves. We will also make the assumption that $\sigma \in S^{N-1}(\sqrt{N})$ in the sequel.

Remark 2.1 (Comments on the approximations). The derivation towards a spin-glass Hamiltonian in this section makes a number of approximations, we now comment on their nature. We don't aim to create a model for a deep network that is faithful for all analyses one might wish to do for a deep network but simply to elucidate the hardness of the optimization problem. We would like to capture the non-convexity of the loss function, the statistics of the local minima and saddle points, the attraction basins of local minima, locations of local minima etc. Roughly speaking, we have created a sparse neural network (each neuron is connected to a constant number of other neurons) with N neurons on each layer and with $\log n$ layers. The most critical assumption in this section is while connecting the loss function to the output of such a network $\widehat{Y}$. We have assumed that the training set is such that the sign of each pixel in the datum $\xi\{-1,1\}^d$ can be flipped to flip the corresponding true label $Y^t \in \{-1,1\}$. This is true for a single-layer perceptron with discrete-valued weights or linear deep networks without non-linearities.

2.2. Energy landscape of a spin-glass

2.2.1. Asymptotic scaling of critical points

For any $u \in \mathbb{R}$, let the total number of critical points in the set $(-\infty, u]$ be

$$\mathrm{crt}(u) = |\{\sigma :\ \nabla H(\sigma) = 0,\ H(\sigma) \leq Nu\}|.$$

We will denote $\mathrm{crt}(\infty)$ by $\mathrm{crt}(H)$. The index of a critical point $\sigma \in \mathrm{crt}(H)$ is the number of negative eigenvalues of the Hessian of $H_{N,p}(\sigma)$ and we denote critical points of index k by $\mathrm{crt}_k(u)$, e.g., local minima are simply $\mathrm{crt}_0(u)$. Using statistics of the eigen-spectrum of the Gaussian Orthogonal Ensemble (GOE), one can show:

Lemma 2.2 (Expected number of critical points). The expected number of critical points of index k for a p-spin glass Hamiltonian with energy below Nu, denoted by $\mathrm{crt}_k(u)$ is

$$\lim_{N\to\infty} \frac{1}{N} \log \mathbb{E}\ \mathrm{crt}_k(u) = \Theta_k(u), \tag{2.9}$$

$$\lim_{N\to\infty} \frac{1}{N} \log \mathbb{E}\ \mathrm{crt}(u) = \Theta(u); \tag{2.10}$$

for any $k \geq 0$ and $p \geq 2$ (Auffinger et al., 2013). The functions $\Theta_k(u)$ and $\Theta(u)$ are known as the "complexity" of a spin glass and are given by

$$\Theta(u) = \begin{cases} \frac{1}{2}\log(p-1) - \frac{(p-2)}{4(p-1)}\ u^2 - I_1(u) & \text{if } u \leq -E_\infty \\ \frac{1}{2}\log(p-1) - \frac{(p-2)}{4(p-1)}\ u^2 & \text{if } -E_\infty \leq u \leq 0 \\ \frac{1}{2}\log(p-1) & \text{else,} \end{cases} \tag{2.11}$$

and

$$\Theta_k(u) = \begin{cases} \frac{1}{2}\log(p-1) - \frac{(p-2)}{4(p-1)}\ u^2 - (k+1)I_1(u) & \text{if } u \leq -E_\infty \\ \frac{1}{2}\log(p-1) - \frac{(p-2)}{p}, & \text{else.} \end{cases} \tag{2.12}$$

The rate function $I_1(u)$ is defined to be

$$\begin{aligned} I_1(u) &= \frac{2}{E_\infty^2}\int_u^{-E_\infty} (z^2 - E_\infty^2)^{1/2}\, \mathrm{d}z \\ &= -\frac{u^2}{E_\infty^2}\sqrt{u^2 - E_\infty^2} - \log\left(-u + \sqrt{u^2 - E_\infty^2}\right) + \log E_\infty. \end{aligned}$$

In particular, note that $\Theta(u)$ and $\Theta_k(u)$ are non-decreasing functions on $\mathbb{R}$.

Based on the above lemma, Figure 2.1 plots the complexity of a spherical spin-glass Hamiltonian at different energies.

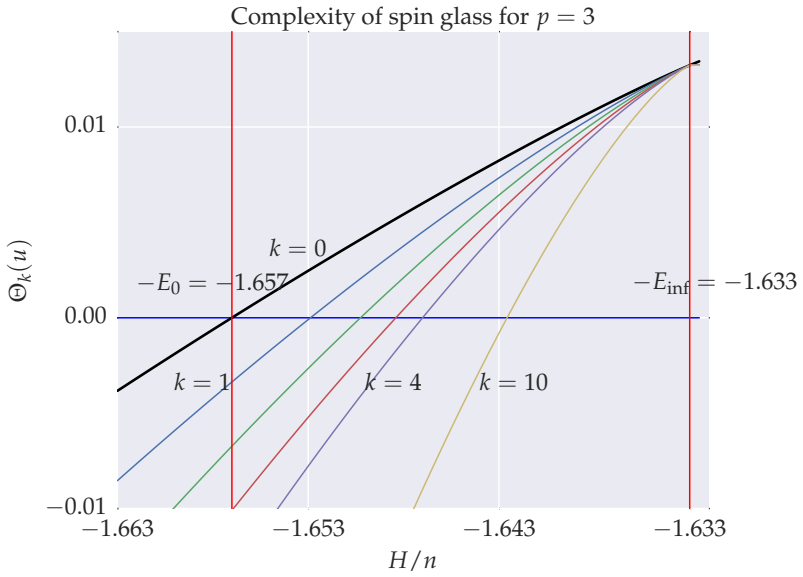


FIGURE 2.1. The number of critical points in $(-\infty, u]$ scales as $e^{N\Theta_k(u)}$. The black curve denotes $\Theta_0(u)$, the complexity of local minima. Below $\lim_{N\to\infty} \inf H(\sigma)/N = -E_0$ which is where $\Theta_0(u)$ intersects the x-axis, there are no local minima in the logarithmic scaling (Auffinger et al., 2013). Similarly, below the point where $\Theta_k(u)$ intersects the x-axis, there are no critical points of index higher than k. The Hamiltonian thus shows a layered landscape, higher-order saddle points to the right and in fact, local minima concentrated near $-E_0$ on the far left (Subag and Zeitouni, 2015).

Remark 2.3 (Convergence in probability for estimates of critical points). Lemma 2.2 studies the asymptotic expected number of critical points for the spherical p-spin Hamiltonian. For $p \geq 7$, near the global minimum, i.e., for small values of u, one can use the second moment method to show that

$$\frac{\mathrm{crt}_k(u)}{\mathbb{E}\left[\mathrm{crt}_k(u)\right]} \to 1 \quad \text{in } \mathscr{L}^2.$$

Remark 2.4 (Exponentially many critical points at all energies). The total number of local minima is exponential in the number of spins. Moreover, they exist at all energy levels above the global minimum $-E_0$. Beyond the threshold value $-E_\infty$, the number of critical points is split evenly among different indices. This shows that the energy landscape is extremely rugged and difficult to optimize. First-order optimization algorithms such as stochastic gradient descent are expected to get stuck in local minima at high energies. The following lemma lends an important insight.

Lemma 2.5 (Auffinger et al. (2013), Lemma 3.2). Assume without loss of generality that $\sigma_0 = (1,0,\ldots,0)^\top$ is a critical point of the Hessian. The gradient $\nabla H(\sigma)$ is zero-mean Gaussian with variance Np and is independent of $\nabla^2 H(\sigma_0)$ and $H(\sigma_0)$. Also, conditional on $H(\sigma_0) = Nu$, the normalized Hessian $N^{-1/2}\,\nabla^2 H(\sigma_0)$ has the distribution

$$\sqrt{2(N-1)p(p-1)}M - puI; \tag{2.13}$$

where $M \in \mathbb{R}^{(N-1)\times(N-1)}$ is from the Gaussian Orthogonal Ensemble and $I \in \mathbb{R}^{(N-1)\times(N-1)}$ is the identity.

This shows that the expected sum of the eigenvalues of the Hessian, or the trace of the Hessian, is $-puN$. The trace is negative for $u \geq 0$ which suggests that on average, there are descent directions to critical points at high energies while an eigenvalue is more likely to be an ascent direction for a critical point below $u = 0$. This is also seen in the experimental results in Mehta et al. (2013, Section IV.E).

Remark 2.6 (Proportion of critical points of different indices). Let us consider the formula for $\Theta_k(u)$ in (2.12) for $u \leq -E_\infty$. Let us define a quantity

$$\chi_k(u) = \frac{\mathrm{d}\,\mathrm{crt}_k(u)}{\mathrm{d}\,u} \approx \frac{\mathrm{d}}{\mathrm{d}\,u} e^{-N(k+1)I_1(u)-Ncu},$$

i.e., $\chi_k(u)$ is the number of critical points of index k in an infinitesimal neighborhood of energy u. We now have

$$\frac{1}{N}\log\frac{\chi_k(u)}{\chi_{k+1}(u)} \approx I_1(u).$$

The function $I_1(u)$ is a monotonically decreasing (and positive) function of u which suggests that at any energy level u, the ratio of critical points of index k is exponentially more than the critical points of index $(k+1)$. In particular, the energy landscape is completely dominated by local minima. This is also seen in Figure 2.1: the line for the cumulative complexity of critical points $\Theta_k(u)$ is always above the line for $\Theta_{k+1}(u)$.

2.2.2. Low temperature landscape: multiple disconnected valleys

The low temperature energy landscape of spin glasses has been studied in detail in the works of Subag (2017); Chatterjee (2014); Jagannath (2017); Talagrand (2010a). The general picture that emerges is that, under the thermodynamic limit as $N \to \infty$ at low enough temperatures, the Gibbs distribution splits into a mixture of "pure states"; each pure state is given by a of these bands has very little overlap with the rest of the Gibbs distribution. See Subag (2017) for a detailed rigorous exposition of this "multiple valleys" picture. We believe, this difference in the shape of the local minima is the most prominent difference between the energy landscape of spin glasses with known results in deep neural networks. For instance, even for a single-layer neural network with no non-linearities, i.e., matrix factorization,

$$X = A\,B;$$

here $X, A, B \in \mathbb{R}^{n\times n}$ there exist symmetries in the solution space wherein if A^*, B^* is a solution, $A^*\,P^{-1}$ and $P\,B$ is also a solution of the same problem

$$\min_{A,B} \|X - A\,B\|_{\mathrm{F}}^2.$$

Recent results in the deep learning literature build upon this observation to show that in the over-parametrized regime, any two weight configurations of a neural network can be connected by (not necessarily straight) paths such that the training loss does not decrease along these paths (Freeman and Bruna, 2016; Venturi et al., 2018). This suggests that the level set of global minima of deep networks are connected, in particular there do not exist isolated global minima for deep networks. Let us note that this observation about matrix factorization can also be used to show that every local minimum of a two-layer neural network is a global minimum (Nouiehed and Razaviyayn, 2018), under certain assumptions. This structure of non-isolated local or global minima is not the case for the mean-field spin glass model considered in this chapter. There are however other popular models in statistical physics such as XY models (Nerattini et al., 2013) and planted spin glasses (Krzakala and Zdeborová, 2009) which possess non-isolated critical points and may serve as good models for deep

neural networks. These are avenues for future investigations into the complexity of the optimization problem underlying deep networks.

2.3. Perturbations of the Hamiltonian

In this section, we discuss the asymptotic scaling of critical points, i.e., all local minima and saddle points. We then show how the energy landscape changes under an external magnetic field in Section 2.3.1 and Section 2.3.3. Section 2.3.4 describes how this perturbation affects the locations of the local minima.

2.3.1. Modifying the energy landscape

Let now consider a perturbation to the Hamiltonian $H_{N,p}$ given by

$$-\widetilde{H}(\sigma) = -H(\sigma) + h^\top \sigma; \tag{2.14}$$

where $h = (h_1, \ldots, h_N) \in \mathbb{R}^N$ is the external perturbation. We let h_i be iid zero-mean Gaussian random variables with

$$h_i \sim \mathrm{N}(0,\ \nu^2) \quad \text{for all } i \le n.$$

We now define a parameter

$$B = \frac{J^2 p(p-2) - \nu^2}{J^2 p^2 + \nu^2} \in (-1,\ 1 - 2/p]. \tag{2.15}$$

that governs the transition between the exponential, polynomial and trivialized regimes. The value $B = 0$, i.e.,

$$\nu_c := J\sqrt{p(p-2)}$$

demarcates the exponential regime from total trivialization.

Theorem 2.7 (Fyodorov (2013)). For large N, the expected number of critical points of the perturbed Hamiltonian is given by

$$\mathbb{E}\ \mathrm{crt}(\widetilde{H}) = \begin{cases} 2 & \text{if } B = -\Omega(N^{-1}), \\ \frac{2n}{\sqrt{\pi}}\tau^{-3/2} & \text{if } B = -\frac{\tau}{N}, \\ 4N^{1/2}\sqrt{\frac{1+B}{\pi B}}\exp\left(\frac{N}{2}\log\frac{1+B}{1-B}\right) & \text{if } B > 0. \end{cases} \tag{2.16}$$

where $\tau \ll n$ is a constant.

For $\nu < \nu_c$, i.e., $B > 0$ in the third case, we have the (non-asymptotic version of) exponential regime in (2.9) and (2.10). The first case shows total trivialization because any smooth function on the sphere $S^{N-1}(\sqrt{N})$ has at least one minimum and at least one maximum, thus there is exactly one minimum if the perturbations are larger than $\nu_c = J\sqrt{p(p-2)}$. Indeed, such a trivialization occurs for all regularized loss functions when the regularization term overwhelms the data term. However, for most loss functions unlike spin glass Hamiltonians, we cannot compute how the regularization term affects the solution landscape; an exception to this is LASSO where the Least Angle Regression (LARS) algorithm exploits the piecewise-linear solution paths to efficiently compute the best regularization coefficient (Efron et al., 2004).

More importantly for us, the phase transition between the exponential and trivial regime is not sharp—otherwise there would be no way of modifying the complexity of the landscape without destroying the loss function. For a band of size $\mathcal{O}(N^{-1})$ below zero, as shown in Theorem 2.7, the spin glass has only $\mathcal{O}(N)$ critical points. A more elaborate analysis on a finer scale, viz., using an order parameter $B = -\tau/n$ for a constant $|\tau| \ll n$ gives the following theorem.

Theorem 2.8 (Fyodorov (2013)). For $B = -\tau/n$, the expected number of critical points is given by

$$\lim_{N\to\infty} \frac{\mathbb{E}\ \mathrm{crt}(\widetilde{H})}{N} = \begin{cases} \frac{2}{\sqrt{\pi}}\ \tau^{-3/2} & \text{for } \tau \gg 1, \\ \frac{2}{\sqrt{\pi|\tau|}}\ 2e^{|\tau|} & \text{for } \tau \ll -1 \end{cases} \tag{2.17}$$

Note that the $\tau \gg 1$ case in (2.17) gives the polynomial regime in (2.16) with $\mathscr{O}(N)$ critical points while the second case recovers the $B > 0$ limit of the exponential regime in (2.16). Indeed, as Theorem 2.8 shows, the polynomial band $|B| = \mathscr{O}(N^{-1})$ extends on either size of $B = 0$, the exponential regime for $B = \Omega(N^{-1})$ lies to the right and the trivialized regime lies to the left of this band with $B = -\Omega(N^{-1})$.

2.3.2. Scaling of local minima

The expected number of local minima of the Hamiltonian also undergoes a similar phase transition upon adding an external perturbation with the same critical threshold, viz., $\nu_c = J\sqrt{p(p-2)}$. If $\nu < \nu_c$, the energy landscape is relatively unaffected and we have exponentially many local minima as predicted by (2.9) and exactly one minimum under total trivialization. Similar to Theorem 2.7, there is also an "edge-scaling" regime for local minima that smoothly interpolates between the two.

Theorem 2.9 (Fyodorov (2013)). For $B = -\frac{\kappa}{2}N^{-1/3}$ and some constant $C > 0$, the expected number of local minima is given by

$$\lim_{N\to\infty} \mathbb{E}\ \mathrm{crt}_0(\widetilde{H}) = \begin{cases} 1 & \text{for } \kappa \gg 1, \\ C\exp\left(\frac{\kappa^2}{24} + \frac{4\sqrt{2}|\kappa|^{3/2}}{3}\right) & \text{for } \kappa \ll -1. \end{cases} \tag{2.18}$$

First, note that this suggests $\mathscr{O}\left(\exp(\log n)^2\right)$ (quasi-polynomial) local minima for $\kappa = \log n$. Also, the band around $B = 0$ for polynomially many local minima is wider; it is $\mathscr{O}(N^{-1/3})$ as compared to $\mathscr{O}(N^{-1})$ for critical points. Above the critical perturbation threshold, in particular, between $-N^{-1/3} \lesssim B \lesssim -N^{-1}$ this shows an interesting phenomenon wherein almost all critical points are simply local minima. This has been previously discussed and empirically observed in the deep learning literature (Dauphin et al., 2014). A more precise statement is proved for spin glasses in (Fyodorov and Williams, 2007).

2.3.3. Annealing scheme

If we set $B = -\tau/n$ in (2.15), for $p \gg 1$ resulting in $\nu_c \approx Jp$, we have

$$\nu = \nu_c\left(1 + \frac{2\tau}{N}\right)^{1/2} \tag{2.19}$$

for the perturbation to be such that our random, sparse deep network has polynomially many critical points. For $0 < \tau \ll n$, we have $\nu > \nu_c$ but the spin glass is still within the polynomial band whereas large values of τ result in total trivialization. For large negative values of τ, the external magnetic field is almost zero and we have exponentially many local minima as given by the second case in Theorem 2.8.

We can now exploit this phenomenon to construct an annealing scheme for τ. For the i^{th} iteration of SGD, we set

$$\tau(i) = n\left(e^{-i/\tau_0} - \frac{1}{2}\right). \tag{2.20}$$

In practice, we treat ν_c and τ_0 as tunable hyper-parameters. Considerations of different values of $\tau(0)$, $\lim_{k\to\infty}\tau(k)$ and speed of annealing can result in a variety of annealing schemes; we have used exponential and linear decay to similar experimental results. Algorithm 1 provides a sketch of the implementation for a general deep network. Note that Line 10 can be adapted to the underlying implementation of SGD, in our experiments, we use SGD with Nesterov's momentum and Adam (Kingma and Ba, 2014).

Algorithm 1: Trivialized-SGD

Input: weights of a deep network w;
Estimate: #layers p, #neurons n;
Parameters: constant τ_0, coupling J, learning rate η;
$h \sim J\sqrt{p(p-2)}\,\mathrm{N}(0,\, I)$;
$i = 0$;
while True **do**
  $\xi_i \leftarrow$ sample batch;
  $\tau_i \leftarrow n\left(e^{-i/\tau_0} - \frac{1}{2}\right)$;
  $h_{\text{curr}} \leftarrow \left(1 + \frac{2\tau_i}{N}\right)^{1/2} h$;
  $w \leftarrow x - \eta\,(\nabla \text{loss}(x,\, \xi_i) + h_{\text{curr}})$;
  $i \leftarrow i + 1$

2.3.3.1. Annealing is tailored to critical points. Recent analyses suggest that even non-stochastic gradient descent always converges to local minima instead of saddle points (Lee et al., 2016; Choromanska et al., 2015). It seems contradictory then that we have used the scaling for critical points derived from Theorem 2.8 instead of the corresponding local minima version, viz., Theorem 2.9 which would give

$$\nu = \nu_c \left(1 + \frac{\kappa \, \log n}{N^{1/3}}\right)^{1/2}; \tag{2.21}$$

surely if SGD never converges on saddle points, reducing the complexity of local minima is more fruitful. We are motivated by two considerations to choose the former. Firstly, in our experiments on convolutional neural networks (2.21) does not perform as well as (2.19), probably due to the asymptotic nature of our analysis. Secondly, even if SGD never converges to saddle points, training time is marred by the time that it spends in their vicinity. For instance, Kramers law argument for Langevin dynamics predicts that the time spent is inversely proportional to the smallest negative eigenvalue of the Hessian at the saddle point (see Bovier and den Hollander (2006) for a precise version).

2.3.4. Quality of perturbed local minima

Section 2.3.1 describes how the perturbation h changes the number of local minima and critical points. It is however important to characterize the quality of the modified minima, indeed it could be that even if there are lesser number of local minima the resulting energy landscape is vastly different. In this section, we show that close to the global minimum, the locations of the local minima in the polynomial regime are only slightly different than the their original locations and this difference vanishes in the limit $N \to \infty$. This section can also be seen as a low-temperature analysis of the Gibbs distribution for a spin-glass Hamiltonian.

We construct a quadratic approximation to the unperturbed Hamiltonian at a critical point and analyze the gradient $\nabla H(\sigma)$ and $\nabla^2 H(\sigma)$ separately. We can show that for a spherical spin glass, the gradient at a critical point is zero-mean Gaussian with variance np while the Hessian is simply a scaled GOE matrix Lemma 2.5. This helps us bound the location and the energy of the perturbed minimum as shown in the following theorem.

The main result of this section is Theorem 2.13 which says that if the external field is such that if ν is a constant, the perturbations of the normalized Hamiltonian are ν in the limit while the perturbations of local minima are at most $\mathcal{O}(N^{-1/3})$. The development here is based on some lemmas in Subag and Zeitouni (2015) which undergo minor modifications for our problem. The basic idea consists of approximating the perturbed Hamiltonian as a quadratic around a local minimum of the original Hamiltonian and analyzing the corresponding local minimum.

Let us consider the Hamiltonian

$$-\widetilde{H}(\sigma) = -H(\sigma) + \frac{1}{\sqrt{N}}\, h^\top \sigma; \tag{2.22}$$

where $h \in \mathbb{R}^N$ is a vector with zero-mean Gaussian iid entries with variance ν^2 and

$$-H(\sigma) = N^{-(p-1)/2} \sum_{i_1, i_2, \ldots, i_p} J_{i_1,\ldots,i_p}\, \sigma_{i_1} \ldots \sigma_{i_p}.$$

Note that this is the same model considered in (2.14) except that we have used a different normalization for the perturbation term. We first get rid off the spherical constraint by re-parameterizing the Hamiltonian around a local minimum $\sigma \in \mathrm{crt}_0(H)$. For $x = (x_1, \ldots, x_{N-1}) \in \mathbb{R}^{N-1}$ with $\|x\| \le 1$ and any $y \in \mathbb{R}^N$ consider the functions

$$\begin{aligned} P_E(x) &= \left(x_1, x_2, \ldots, x_{N-1}, \sqrt{1 - \|x\|^2} \right), \\ S(y) &= \sqrt{N}\, y. \end{aligned}$$

Now define

$$f_\sigma(x) \triangleq -\frac{1}{\sqrt{N}}\, H \circ \theta_\sigma \circ S \circ P_E(x); \tag{2.23}$$

where $\sigma \in S^{N-1}(1)$ is the normalized spin on the unit-sphere and θ_σ is a rotation that sends the the north pole to σ. The function $f_\sigma(x)$ is thus the re-parameterization of $H(\sigma)$ around a local minimum σ, but normalized to have a unit variance. We define $\widetilde{f}_\sigma(x)$ similarly and also define $\overline{f}_\sigma(x)$ for the perturbation term $h^\top \sigma$ to see that (2.22) becomes

$$\sqrt{N}\, \widetilde{f}_\sigma(x) = \sqrt{N} f_\sigma(s) + \overline{f}_\sigma(x).$$

We can now construct the quadratic approximation of $\widetilde{f}_\sigma(x)$ (which we assume to be non-degenerate at local minima, i.e, the Hessian has full rank) to be

$$\sqrt{N}\, \widetilde{f}_{\sigma,\mathrm{approx}}\,(x) \triangleq \sqrt{N}\, f_{\sigma,\mathrm{quad}}(x) + \overline{f}_{\sigma,\mathrm{lin}}(x);$$

where we used the quadratic approximation of the original Hamiltonian and the linear approximation of the perturbation term:

$$\begin{aligned} f_{\sigma,\mathrm{quad}}(x) &= f_\sigma(0) + \langle \nabla f_\sigma(0), x \rangle + \frac{1}{2} \left\langle \nabla^2 f_\sigma(0) x,\, x \right\rangle, \\ \overline{f}_{\sigma,\mathrm{lin}}(x) &= \overline{f}_\sigma(0) + \left\langle \nabla \overline{f}_\sigma(0), x \right\rangle. \end{aligned} \tag{2.24}$$

Let Y_σ be the critical point of $\sqrt{N}\, \widetilde{f}_{\sigma,\mathrm{approx}}$ and V_σ be its value. We have,

$$Y_\sigma \triangleq -\frac{1}{\sqrt{N}}\, \left(\nabla^2 f_\sigma(0) \right)^{-1}\, \nabla \overline{f}_\sigma(0); \tag{2.25}$$

$$\begin{aligned} \Delta_\sigma &\triangleq \sqrt{N}\, \widetilde{f}_\sigma(Y_\sigma) - \sqrt{N}\, f_\sigma(0) - \overline{f}_\sigma(0), \\ &= -\frac{1}{2\sqrt{N}}\, \left(\nabla \overline{f}_\sigma(0) \right)^\top \left(\nabla^2 f_\sigma(0) \right)^{-1}\, \nabla \overline{f}_\sigma(0). \end{aligned} \tag{2.26}$$

The term $\sqrt{N} f_\sigma(0)$ is the value of the original Hamiltonian at the unperturbed minimum and thus Δ_σ is the perturbation in the critical value of the Hamiltonian due to the perturbations minus the contribution of the perturbation $\overline{f}_\sigma(0)$ itself. We now allude to the following lemma that gives the distribution of the gradient and the Hessian of the Hamiltonian at the critical point, viz., $\nabla f_\sigma(0)$ and $\nabla^2 f_\sigma(0)$.

We now compute bounds for $\|Y_\sigma\|$ and $\|\Delta_\sigma\|$ using the following two lemmas but let us describe a few quantities that we need before. It can be shown that the value of a p-spin glass Hamiltonian at local minima concentrates around a value

$$m_N = -N E_0 + \frac{\log N}{2\Theta'(-E_0)} - K_0; \tag{2.27}$$

where E_0 is the lower energy bound in Figure 2.1, Θ is the complexity of critical points discussed in Figure 2.1. In fact, the Hamiltonian is distributed as a negative Gumbel distribution with m_N as the location parameter.

The constants C_0 and K_0 are computed in Subag and Zeitouni (2015, Eqn. 2.6). For a small $\delta \in (0, p(E_0, E_{\text{inf}}))$, define $\mathscr{V}(\delta)$ to be the set of real symmetric matrices with eigenvalues in the interval

$$(pE_0 - 2\sqrt{p(p-1)} - \delta,\ pE_0 + 2\sqrt{p(p-1)} + \delta);$$

and define the set $B_1 = \sqrt{N}\mathscr{V}(\delta)$. The proof of Lemma 2.12 hinges upon the observation in Subag (2015) that the second negative term (apart from the first GOE term) in the Hessian at a critical point (2.13) does not change the eigenvalue distribution of the GOE much.

Lemma 2.10. For any $L > 0$ and

$$g_3(\sigma) = \Delta_\sigma + \frac{\nu^2}{2\sqrt{N}} \operatorname{Tr}\left\{ \left(\nabla^2 f_\sigma(0)\right)^{-1} \right\}, \quad B_3 = \left(-N^{-\frac{1}{2}+\varepsilon}, N^{-\frac{1}{2}+\varepsilon}\right);$$

there exists a sequence $e_{\varepsilon,\delta}(N) \to 0$ such that

$$\mathbb{P}\left(g_3 \notin B_3 :\ \nabla^2 f_\sigma(0) = A_{N-1},\ \nabla f_\sigma(0) = 0,\ f_\sigma(0) = u\right) \le e_{\varepsilon,\delta}(N),$$

for all $u \in N^{-1/2}\ (m_N - L, m_N + L)$ and $A_{N-1} \in \sqrt{N}\ \mathscr{V}(\delta)$.

Proof. First note that $\nabla \overline{f}_\sigma(0) = h$ and the term Δ_σ evaluates to $-\frac{1}{2\sqrt{N}}\ h^\top \left(\nabla^2 f_\sigma(0)\right)^{-1} h$. The gradient of the perturbation $\nabla \overline{f}_\sigma(0) = h$ is independent of the disorder that defines $\nabla^2 f_\sigma(0)$. Thus, conditional on $\nabla^2 f_\sigma(0) = A_{N-1} \in B_1$, we have that $\nabla \overline{f}_\sigma(0)$ is distributed as

$$\nu \sum_{i=1}^{N-1} W_i\, a_i;$$

where $W_i \sim N(0,1)$ are standard Gaussian random variables and a_i are the eigenvectors of A_{N-1}. Because $h \sim N(0, \nu^2 I)$, we now see that $g_3(\sigma)$ has the same distribution as

$$\frac{\nu^2}{2\sqrt{N}} \sum_{i=1}^{N-1} (1 - W_i^2)\ \frac{1}{\lambda(A_{N-1})};$$

where $\lambda(A_{N-1})$ are the eigenvalues of A_{N-1} and $W_i \sim N(0,1)$ are standard Gaussian random variables. This is a zero-mean random variable with bounded second moment, indeed

$$\left(\frac{\nu^2 C}{2n}\right)^2 \sum_{i=1}^{N-1} \mathbb{E}\left\{ \left(1 - W_i^2\right)^2 \right\} = 2\left(\frac{\nu^2 C}{2n}\right)^2 (N-1); \tag{2.28}$$

for some constant $C > 0$. The lemma now follows by Chebyshev's inequality. ■

Remark 2.11. Note that we had used a different normalization for the magnetic field in (2.22) as compared to (2.14). Switching back to the original normalization, i.e., replacing $\nu \leftarrow \nu N^{1/2}$, we see from (2.28) that we still have a bounded second moment if ν is a constant, in particular, if it does not grow with N.

Lemma 2.12 (Subag and Zeitouni (2015), Lemma 17). For $\tau_{\varepsilon,\delta} \to 0$ as $N \to \infty$ slowly enough and $L > 0$

$$g_4(\sigma) = \frac{\nu^2}{2\sqrt{N}} \operatorname{Tr}\left\{ \left(\nabla^2 f_\sigma(0)\right)^{-1} \right\}, \quad B_4 = (C_0 - \tau_{\varepsilon,\delta}(N),\ C_0 + \tau_{\varepsilon,\delta}(N));$$

we have

$$\lim_{N\to\infty} \mathbb{E}\left\{ |\sqrt{N}\ \sigma : H(\sigma) \in [m_N - L, m_N + L],\ g_4(\sigma) \notin B_4,\ \nabla^2 f_\sigma(0) \in \sqrt{N}\mathscr{V}(\delta)| \right\} = 0.$$

We can now easily combine the estimates of Lemma 2.10 and Lemma 2.12 to obtain:

Theorem 2.13. The perturbation of the normalized Hamiltonian is ν in the limit $N \to \infty$ while the perturbed local minima lie within a ball of radius $N^{-\alpha}$ with $\alpha \in (1/4, 1/3)$ centered at the original local minima if ν is a

constant. More precisely for all critical points $\sigma \in \mathrm{crt}_0(H)$,

$$\lim_{N\to\infty} \frac{1}{\sqrt{N}} |\widetilde{f}_\sigma(Y_\sigma) - f_\sigma(0)| = \nu,$$
$$\|Y_\sigma\| \leq N^{-\alpha}.$$

Proof. The above two lemmas imply that $|g_4(\sigma)| \leq C_0$ and $|\Delta_\sigma + g_4(\sigma)| \leq N^{-1/2+\varepsilon}$ with high probability. Together, this gives that

$$|\Delta_\sigma| \leq C_0 + N^{-1/2+\varepsilon},$$
$$\sqrt{N}|\widetilde{f}_\sigma(Y_\sigma) - f_\sigma(0)| \leq \|h\| + C_0 + N^{-1/2+\varepsilon}. \tag{2.29}$$

We now set $\nu \leftarrow \nu N^{1/2}$ in (2.22) to match the normalization used in the preceding development to get that the perturbations to the normalized Hamiltonian are

$$\frac{1}{\sqrt{N}}|\widetilde{f}_\sigma(Y_\sigma) - f_\sigma(0)| \leq \nu + \frac{C_0}{N} + N^{-3/2+\varepsilon}. \tag{2.30}$$

Since the gradient $\nabla f_\sigma(0)$ is independent of $\nabla^2 f_\sigma(0)$ and $f_\sigma(0) = u$, we have the following stochastic domination

$$\|Y_\sigma\| \leq \frac{c}{N} \|h\|;$$

for some constant $c > 0$. The conditional probability in Lemma 2.10 is now computed to be

$$\mathbb{P}\left(Q_{N-1} \geq \frac{N^{\frac{1}{2}-\alpha}}{c\nu}\right); \tag{2.31}$$

where Q_{N-1} is a standard Chi random variable with $N-1$ degrees of freedom, which goes to zero if ν is a constant that goes not grow with N. In particular, we have that $\|Y_\sigma\| \leq N^{-a}$. ■

In other words, the polynomial regime does not cause the local minima of the Hamiltonian to shift. Roughly, it only smears out small clusters of nearby local minima.

2.4. Experiments

In this section, we first empirically compute the energy landscape of a spin glass Hamiltonian and show how it changes with different levels of external perturbation. This gives a novel insight into structure of saddle points in the high-dimensional parameter space of a deep network. We then demonstrate our annealing procedure described in Section 2.3.3 on fully-connected and convolutional neural networks for image classification on two datasets, viz., the MNIST dataset (LeCun et al., 1998) with $70,000$ 28×28 gray-scale images of digits, and the CIFAR-10 dataset (Krizhevsky, 2009) with $60,000$ 32×32 RGB images of 10 classes of objects. Even for large CNNs that are close to the state-of-the-art on CIFAR-10, we show that Trivialized-SGD results in improved generalization.

2.4.1. Energy landscape of spin glasses

Consider the perturbed 3-spin glass Hamiltonian with $n = 100$ spins, in our notation this is a sparsely connected neural network with 100 neurons on each layer, three hidden layers, threshold non-linearities and a fourth loss layer:

$$-\widetilde{H}_{N,3} = \frac{1}{N} \sum_{i,j,k=1}^{N} J_{i,j,k}\, \sigma_i \sigma_j \sigma_k + \sum_i h_i\, \sigma_i; \tag{2.32}$$

where $h_i \in N(0, \nu^2)$ and $J_{i,j,k} \sim N(0,1)$. For a given a disorder J we empirically find the local minima of the above Hamiltonian for values of h in the three regimes:

(i) $\nu = 1/N$ for exponentially many critical points;

(ii) $\nu = \sqrt{3}(1-1/N)^{1/2}$ to be in the polynomial regime;
(iii) and a large value $\nu = 3$ to be in the totally trivialized regime with a single minimum.

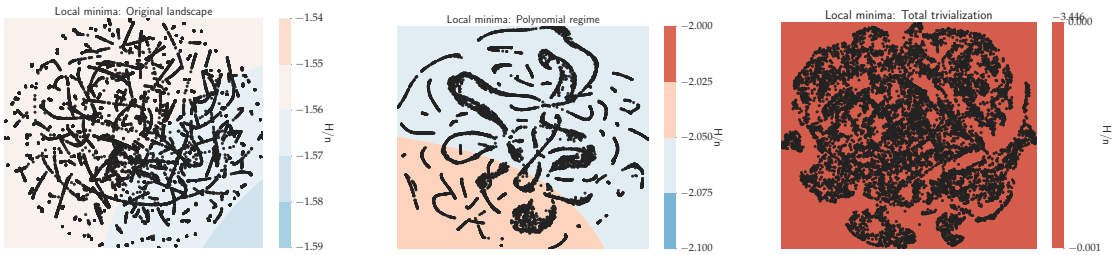
For an initial configuration σ^1 that is sampled uniformly on $S^{N-1}(\sqrt{N})$, we perform gradient descent with a fixed step size $\eta = 0.1$ and stop when $\|\nabla \widetilde{H}\|_2 \leq 10^{-4}$. The gradient can be computed to be

$$-\left(\nabla \widetilde{H}_{N,3}\right)_i = \frac{1}{N} \sum_{j,k} \left[J_{ijk}\sigma_j\sigma_k + J_{jik}\sigma_j\sigma_k + J_{jki}\sigma_j\sigma_k\right] + h_i.$$

Before each weight update, we project the gradient on the tangent plane of $S^{N-1}(\sqrt{N})$ and re-project $\sigma^{k+1} \leftarrow \sigma^k - \eta \nabla \widetilde{H}(\sigma^k)$ back on to the sphere.

We next perform a t-distributed stochastic neighbor embedding (t-SNE) (Van der Maaten and Hinton, 2008) for 20,000 runs of gradient descent from independent initial configurations. t-SNE is a nonlinear dimensionality reduction technique that is capable of capturing the local structure of the high-dimensional data while also revealing global structure such as the presence of clusters at several scales. This has been successfully used to visualize the feature spaces of deep networks (Donahue et al., 2014). We use it to visualize the high-dimensional energy landscape.

The results of this simulation are shown in Figure 2.2. Local minima are seen as isolated points with surrounding energy barriers in Figure 2.2a. The saddle points in this t-SNE of the 100-dimensional hyper-sphere are particularly interesting. They are seen as long "canyons", i.e., they are regions where the gradient is non-negative in all but a few directions, gradient descent is only free to proceed in the latter. As multiple independent runs progress along this canyon, we get the narrow connected regions in Figure 2.2a and Figure 2.2b.



(A) (B) (C)

FIGURE 2.2. Two-dimensional t-SNE (Van der Maaten and Hinton, 2008) of 20,000 of local minima discovered by gradient descent for the exponential, polynomial and trivial regimes. The background is colored by the kernel density estimate of the value of the normalized Hamiltonian H/N. The numerical values of the normalized Hamiltonian in Figure 2.2a are the non-asymptotic versions ($n = 100$) of the values in Figure 2.1.

Exponential regime (Figure 2.2a): Local minima are seen here as isolated dots surrounded by high energy barriers while saddle points are seen as narrow connected regions, viz., regions where the gradient is very small in all but a few directions.

Polynomial regime (Figure 2.2b): The number of isolated clusters, i.e., local minima, is significantly smaller as compared to Figure 2.2a. As the discussion in Section 2.3.2 predicts, the energy landscape seems to be full of saddle points in the polynomial regime.

Trivial regime (Figure 2.2c): Gradient descent always converges to the same location. The average cosine distance (on $S^{N-1}(\sqrt{N})$) here is 0.02 as compared to 1.16 for Figure 2.2a which suggests that this is indeed a unique local minimum.

2.4.2. Setup for deep neural networks

The theoretical model in Section 2.1 and the annealing scheme (2.19) depend on two main parameters: the number of neurons on each layer N and the number of hidden layers p. Modern deep networks with multiple feature maps, convolutional kernels etc. do not conform to our theoretical architecture and we therefore use a heuristic to set these parameters during training. For all networks considered here, we set

$$N := \left(\frac{\#\text{ weights}}{p}\right)^{1/2}$$

where p is taken to be the number of layers. This estimate for the number of neurons is accurate for a fully-connected, dense neural network of p layers. For CNNs, we consider the combined block of a convolution operation followed by batch normalization and max-pooling as a single layer of the theoretical model. All linear and convolutional layers in what follows have rectified linear units (ReLUs) non-linearities and we use the cross-entropy loss to train all our networks.

Theorem 2.7 depends on the spherical constraint introduced in Section 2.1 quite crucially. Indeed, the fact that the spherical spin glass Hamiltonian is isotropic on the hyper-sphere is essential for us to be able to pick the perturbation h in a random direction. Modern deep networks are not trained with additional constraints on weights, they instead use ℓ_2 regularization (weight decay) which can be seen as a soft ℓ_2 constraint on the weights. This case can also be analyzed theoretically and the annealing strategy only undergoes minor variations (Fyodorov, 2013) (the polynomial regime is achieved for $|B| = \mathcal{O}(N^{-1/2})$. We however use the annealing scheme in Section 2.3.3 with $|B| = \mathcal{O}(N^{-1})$ here.

For MNIST, we scale each pixel to lie between $[-1, 1]$. For CIFAR-10, we perform global contrast normalization and ZCA-whitening (Goodfellow et al., 2013) as the preprocessing steps. We do not perform any data augmentation. For both datasets, we use a subset of size 12% of the total training data as a validation set and use it for hyper-parameter tuning of the standard pipeline without gradient perturbation. For the best parameters, we run both the standard optimizer and Trivialized-SGD with the same random seed, i.e., both networks have the same weight initialization, batch-sampling and the same hyper-parameters, and report the error rates on the test set averaged over a few runs.

2.4.3. MNIST

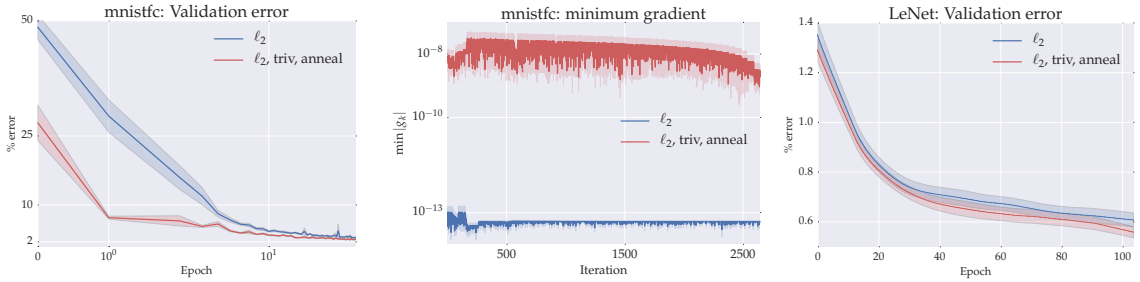
As a sanity check, we construct a thin, very-deep fully-connected network to demonstrate topology trivialization. Deep networks with a large number of connections, viz., fully-connected ones, are often challenging to train due to poorly initialized weights and vanishing gradients (Sutskever et al., 2013). The external magnetic field results in a gradient bias and thus artificially boosts the magnitude of the gradient which should effectively counter this problem.

Our "mnistfc" network has 16 hidden layers with 128 hidden units and we use a momentum-based optimizer Adam (Kingma and Ba, 2014) with learning rate 10^{-3} for 50 epochs to train it. We set $J = 10^{-3}$ and $\tau_0 = 500$ along with an ℓ_2 regularization of 10^{-5}. As Figure 2.3a shows, averaged over 10 runs, Trivialized-SGD results in dramatically faster training in the beginning when the external field is stronger. Moreover, as predicted, the minimum of the absolute value of the back-propagated gradient with Trivialized-SGD in Figure 2.3b is significantly larger than the original gradient. The final validation error for annealing is also slightly smaller (2.48%) as compared to the error without it (2.92%).

We now train a LeNet-style network (LeCun et al., 1998) on 28×28 gray-scale images with the following architecture:

$$\text{block}_{32} \to \text{block}_{64} \to \text{linear}_{512} \to \text{drop}_{\,0.1} \to \text{softmax}.$$

The block_{32} layer denotes a convolutional layer with a 5×5 kernel and 32 features, 3×3 max-pooling and batch-normalization. Note that as compared to the standard LeNet, we have replaced the drop layer after the convolutional layer with batch-normalization; this is common in recent papers (Ioffe and Szegedy, 2015). We



(A) (B) (C)

FIGURE 2.3. Figure 2.3a: mnistfc trained with Adam (blue) vs. Trivialized-SGD (red). Figure 2.3b: Minimum absolute value of the back-propagated gradient during training for Adam (blue) versus Trivialized-SGD (red). Figure 2.3c: validation error for lenet with Adam (blue) versus Trivialized-SGD (red).

train using Adam with a learning rate $\eta = 10^{-4}$ for 100 epochs with a batch size of 256 and ℓ_2 regularization of 10^{-3} and compare it with Trivialized-SGD using $J = 10^{-4}$ and $\tau_0 = 10^3$.

Figure 2.3c shows the average validation error over 10 runs. Typically, convolutional networks are very well-regularized and the speedup here in training is smaller than Figure 2.3. However, we still have an improvement in generalization error: $0.55 \pm 0.03\%$ final validation error with annealing compared to $0.61 \pm 0.04\%$ without it.

2.4.3.1. CIFAR-10. As a baseline for the CIFAR-10 dataset, we consider a slight modification of the All-CNN-C network of Springenberg et al. (2014). This was the state-of-the-art until recently and consists of a very simple, convolutional structure with carefully chosen strides:

$$\text{input}_{3\times32\times32} \to \text{drop}_{\,0.2} \to \big(\text{block}_{96}\big)_{\times3} \to \text{drop}_{\,0.5} \to \big(\text{block}_{192}\big)_{\times3} \to \text{drop}_{\,0.5} \to \big(\text{block}_{192}\big)_{\times2} \dots$$
$$\dots \to \text{mean-pool}_{8\times8} \to \text{softmax}.$$

The block_{96} layer denotes a $\text{conv}_{3\times3\times96}$ layer followed by batch normalization (which is absent in the original All-CNN-C network). Unfortunately, we could not get Adam to prevent over-fitting on this network and hence we train with SGD for 200 epochs with a batch-size of 128 and Nesterov's momentum. We set momentum to 0.9 and the initial learning rate to 0.1 which diminishes by a factor of 5 after every 60 epochs. We also use an ℓ_2 regularization of 10^{-3}. The optimization strategy and hyper-parameters were chosen to obtain error rates comparable to Springenberg et al. (2014); however as Table 2.1 shows, we obtain a 0.66% improvement even without annealing on their results (8.42% test error in 200 epochs vs. 9.08% error in 350 epochs). This can probably be attributed to batch normalization.

We compare the training of the baseline network against Trivialized-SGD with $J = 10^{-4}$ and $\tau_0 = 10^3$ as the hyper-parameters over two independent runs. Table 2.1 shows a comparison with other competitive networks in the literature. We note that All-CNN-BN with annealing results in a significantly lower test error (7.95%) than the baseline (8.42%).

Model	Test error (%)
NiN (Lin et al., 2013)	10.41
All-CNN-C (Springenberg et al., 2014)	9.04
P4-All-CNN (Cohen and Welling, 2016)	8.84
All-CNN-BN (baseline)	8.42 $\pm$ 0.08
All-CNN-BN (resampled, annealed noise)	8.16 $\pm$ 0.07
All-CNN-BN (with topology trivialization)	7.95 $\pm$ 0.04
ELU (Clevert et al., 2015)	6.55

TABLE 2.1. Error rates on CIFAR-10 without data augmentation.

2.4.3.2. Comparison to gradient noise. Annealed additive gradient noise has also been recently explored for complex recurrent network architectures in Neelakantan et al. (2015). We compare Trivialized-SGD with this approach and as Table 2.1 shows, although resampled noise improves slightly upon the baseline, fixing the direction of the additive term works better. This is in line with our analysis using an external magnetic field — as Figure 2.4 shows — resampling is detrimental because it forces the weights to align in different uncorrelated directions at each iteration whereas the annealing scheme simply introduces a (modulated) bias in a fixed direction.

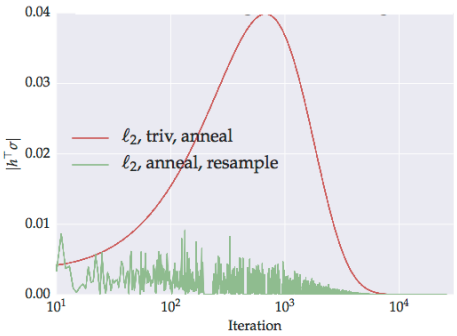


FIGURE 2.4. Alignment of the weights with the perturbation term $\left(|h^\top \sigma|\right)$ for All-CNN-BN trained with Trivialized-SGD (red) vs. resampled, annealed additive noise (green) (cf. Table 2.1 for error).

CHAPTER 3

Where do empirically successful algorithms converge?

The previous chapter constructs a model for the energy landscape of deep neural networks. Under a mean-field p-spin model, the loss function of deep networks has exponentially many critical points at all energy levels. In other words, it is a very difficult to find low energy local minima because greedy, local algorithms like stochastic gradient descent may get stuck in sub-optimal ones. In practice however, it is quite possible—even easy—to train even large neural networks to obtain near-zero error on the training data.

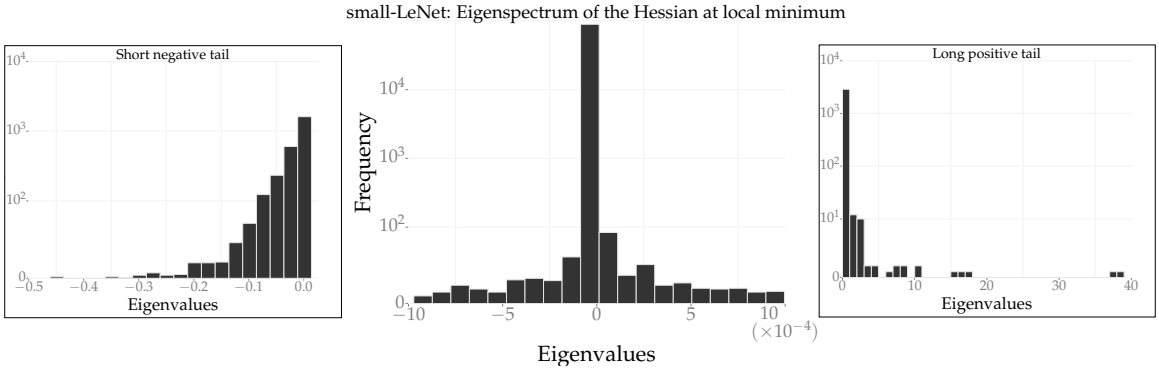


FIGURE 3.1. Eigenspectrum of the Hessian at a local minimum of a CNN on MNIST (two independent runs). The central plot shows the eigenvalues in a small neighborhood of zero whereas the left and right insets show the entire tails of the eigenspectrum.

This chapter takes a slightly different recourse to the problem of understanding the energy landscape. It begins with an empirical result showcasing the local geometry of the solutions found by optimization algorithms for deep networks. Consider the histogram in Figure 3.1 showing the spectrum of the Hessian at an extremum discovered by Adam (Kingma and Ba, 2014) for a convolutional neural network on MNIST (LeCun et al., 1998) ($\approx 47,000$ weights, cf. Section 3.3.1). It is evident that:

(i) a large number of directions ($\approx 94\%$) have near-zero eigenvalues with magnitude less than 10^{-4},
(ii) positive eigenvalues (right inset) have a long tail with the largest one being almost 40,
(iii) negative eigenvalues (left inset), which are directions of descent that the optimizer missed have a much faster decay, the largest negative eigenvalue is only -0.46.
(iv) the location is not a local minimum, it is a saddle point with both positive and negative eigenvalues.

This trend is not unique to this particular network; its qualitative properties are shared across a variety of network architectures, network sizes, datasets or optimization algorithms. Local minima that generalize well and are discovered by stochastic gradient descent lie in "wide valleys" of the energy landscape, rather than in sharp, isolated minima. Based on this understanding of how the local geometry looks at the end of optimization, can we modify SGD to actively seek such regions? Motivated by the work of Baldassi et al.

(2015) on shallow networks, instead of minimizing the original loss $f(x)$, we propose to minimize

$$f_\gamma(x) = -\log \int_\Omega \exp\left(-f(y) - \frac{1}{2\gamma}\,|x-y|^2\right)\,\mathrm{d}y. \tag{3.1}$$

This function which we will call Local Entropy is a local log-partition function that measures both the loss at a location x and the average loss of its neighborhood via an explicit integration on the variable y. Wider the region at x, smaller is the e. This chapter constructs an algorithm called Entropy-SGD to minimize (3.1).

Local Entropy is the convolution of the exponentiated loss function $e^{-f(x)}$ with a Gaussian measure $\exp\left(-\frac{|x-y|^2}{2\gamma}\right)$ and as such, is smoother than the original loss $f(x)$. This accelerates stochastic gradient descent on this modified loss function and also leads to bounds on the generalization error via techniques from uniform stability. While it is extremely surprising that greedy and local algorithms like SGD discover "rare" wide regions in the parameter space, explicitly biasing the optimization towards wide regions results in faster algorithms with better generalization error. This chapter shows results on benchmark neural networks with competitive baselines and shows improved performance in terms of both convergence rate and generalization error.

3.1. Local Entropy

We first provide a simple intuition for the concept of Local Entropy of an energy landscape. This section builds upon the results of Baldassi et al. (2016a) and extends them for the case of continuous variables.

Consider a cartoon energy landscape in Figure 3.2 where the x-axis denotes the configuration space of the parameters. We have constructed two local minima: a shallower although wider one at $\widehat{x}_{f_\gamma}$ and a very sharp global minimum at $\hat{x}_f$. Under a Bayesian prior on the parameters, say a Gaussian of a fixed variance at locations $\widehat{x}_{f_\gamma}$ and $\widehat{x}_f$ respectively, the wider local minimum has a higher marginalized likelihood than the sharp valley on the right.

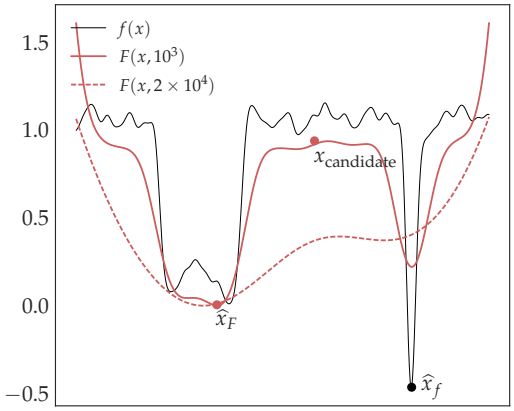


FIGURE 3.2. Local Entropy concentrates on wide valleys in the energy landscape. $F(x,g) = f_\gamma(x)$ in this picture

The above discussion suggests that parameters that lie in wider local minima like $\widehat{x}_{f_\gamma}$, which may possibly have a higher loss than the global minimum, should generalize better than the ones that are simply at the global minimum. In a neighborhood of $\widehat{x}_{f_\gamma}$, Local Entropy in (3.1) is small because it includes the contributions from a large region of good parameters; conversely, near $\widehat{x}_f$, there are fewer such contributions and the resulting Local Entropy is higher. Local Entropy thus provides a way of picking large, approximately flat, regions of the landscape over sharp, narrow regions in spite of the latter possibly having a lower loss or being more numerous. Quite conveniently, Local Entropy is also computed from the partition function with a local re-weighting term.

3.1.1. Energy landscape of the binary perceptron

The binary perceptron is a prototypical constraint satisfaction problem: the goal is to fit a binary training dataset where each datum $\xi \in \Xi$ lies in a discrete space $\xi \in \{-1,1\}^d$, each label $y \in Y$ is just the sign $y \in \{-1,1\}$ and the weights are discrete

$$x \in \{-1,1\}^d.$$

A single-layer perceptron predicts an output

$$\widehat{y} = \text{sign}(x \cdot \xi)$$

which incurs an error according to the zero-one loss as

$$f(x) = \prod_{i=1}^{N} \mathbf{1}_{\{y_i \neq \widehat{y}_i\}} = \prod_{i=1}^{N} \Theta\left(y_i \, \frac{x \cdot \xi_i}{\sqrt{d}}\right) \tag{3.2}$$

where $\Theta(\cdot)$ is the Heavyside step function. The above loss function is a hard constraint satisfaction problem; it is zero if and only if the perceptron classifies the entire dataset correctly. Note that we have normalized by $\sqrt{d}$ to ensure that the argument to the Heavyside step function is always of order unity. There are two problems one can study under this setup:

(i) a classification task where each $\xi \in \Xi$ consists of iid random entries from $\{-1,1\}$ and we aim to minimize $f(x)$, and
(ii) a generalization task where a teacher-student model wherein a teacher perceptron, say x^* labels random vectors $\xi_{i \leq N}$ drawn from $\{-1,1\}^d$. Note that if a candidate student x is equivalent to the teacher, i.e., it has the same weights as x^*, it classifies all samples in any dataset Ξ correctly and also generalizes perfectly.

The second aspect will be the key to understanding the notion of generalization and motivation for Local Entropy; standard approaches in statistical learning theory such as VC theory (Vapnik, 1998)), Rademacher and Gaussian complexity of Bartlett and Mendelson (2002); Koltchinskii and Panchenko (2000, 2002); Mendelson (2002), uniform stability of Bousquet and Elisseeff (2002)) or PAC-Bayes bounds of McAllester (2013) do not consider this setup. It can be shown that if

$$\alpha = \frac{N}{d}$$

is less than $\alpha_c = 0.833$, the probability of finding a solution to the classification problem with zero error drops to zero. On the other hand, there exist exponentially many solutions for the teacher-student scenario up to $\alpha \leq \alpha_{\text{TS}} = 1.245$ and there is only one solution, the teacher, after that. In the case of a binary perceptron, the generalization error of a student can be computed in closed form as

$$\varepsilon = \frac{1}{\pi} \cos^{-1}\left(\frac{x \cdot x^*}{d}\right); \tag{3.3}$$

note that this is simply the probability that two perceptrons, x and x^* in this case, agree on a randomly sampled input vector $\xi \in \{-1,1\}^d$.

The first step of a standard analysis of a perceptron involves finding the total number of solutions. It begins with identifying that all the labels in the dataset can be set to $y_i = 1$ because we can simply flip the signs of the corresponding input vector ξ_i without changing the parameters of the perceptron. The partition function

$$Z = \sum_{x \in \{-1,1\}^d} \prod_{i=1}^{N} \Theta\left(\frac{x \cdot \xi_i}{\sqrt{d}}\right) \, \exp\left[\beta \, x \cdot x^*\right] \tag{3.4}$$

counts the total number of solutions to a binary perceptron for a given dataset Ξ that are weighted by an inverse temperature parameter β. This summation can be evaluated to obtain

$$Z = \sum_{\tilde{q} \in [-1,1]} \exp\left[d\left(s(\tilde{q}) + \beta \, \tilde{q}\right)\right]$$

where $\tilde{q} = \frac{x \cdot x^*}{d}$ is the overlap between a weight vector x and the teacher x^* and $e^{ds(\tilde{q})}$ is the total number of solutions with this overlap. Note that $\beta = 0$ recovers the total number of solutions. The above partition function is written for a single dataset Ξ, the next step is to average the log-partition function over random input samples to estimate the so-called quenched average:

$$\Big\langle \log Z \Big\rangle_{\Xi}.$$

either via Jensen's inequality (also known as annealed approximation in statistical physics) or replica techniques (Engel and Van den Broeck, 2001). The picture of the energy landscape that emerges from this analysis can be summarized as follows:

(i) typical solutions are isolated from each other for all values of α with a distance of $\mathcal{O}(d)$ between any two solutions,
(ii) the analytical generalization error of typical solutions is higher than the generalization error obtained by empirically successful algorithms such as reinforced Belief Propagation (Braunstein et al., 2005) and survey propagation (Braunstein and Zecchina, 2006). The solutions of these algorithms are not typical, they belong to regions in the parameter space which possess a high density of solutions.

This motivated (Baldassi et al., 2015; Huang et al., 2013) to construct a new energy which measures not only the error on the training dataset but also the total number of solutions in its neighborhood. This amounts to studying the energy

$$f_{\gamma}^{\text{discrete}}(x) = f(x) \left(\sum_{y} f(y) \, \mathbf{1}_{\left\{ \frac{x \cdot y}{d} = 1 - 2\gamma \right\}} \right); \tag{3.5}$$

the indicator variable singles out all solutions $y \in \{-1,1\}^d$ that are at a distance $d(1-2\gamma)$ from the references solution $x \in \{-1,1\}^d$. The log-partition function of $f_{\gamma}^{\text{discrete}}$ is

$$Z = -\frac{1}{d\beta} \, \log \sum_{x} \left(f_{\gamma}^{\text{discrete}}(x) \right)^{\beta}$$

which can again be studying using techniques from replica theory. The picture of the energy landscape that emerges from this exercise is summarized as a cartoon in Figure 3.3. One can also forgo the constraint that the reference configuration x is a solution of the perceptron (Baldassi et al., 2016c). A very interesting phenomenon emerges in this case: even if x is not constrained to be a solution, if we only seek configurations such that most of their neighbors y are solutions, the probability that x itself is a solution goes to one as the distance d approaches zero.

Remark 3.1 (Local entropy landscape at higher energies). This section has discussed the connectivity of the solution space of the perceptron. The question of how the entire energy landscape looks remains open. While the zero-one loss (3.2) only picks solutions of the entire training set, we can study an energy of the form

$$f(x) = \prod_{i=1}^{N} \Theta\left(y_i \frac{x \cdot \xi_i}{\sqrt{d}} - \kappa \right)$$

that measures the solutions with stability of at least κ. For $\kappa > 0$, this set of solutions is a subset of the solutions of the original problem where as for $\kappa < 0$, it allows us to study the regions in the parameter space which misclassify a few samples by a margin κ.

Remark 3.2 (Local entropy as tilting the Gibbs distribution). This section discussed the typical and dense solutions of the binary perceptron. Let us focus on a few important facts:

(i) Empirically successful algorithms for constraint satisfaction such as reinforced Belief Propagation (BP) (Braunstein et al., 2005) and stochastic gradient descent for deep networks reach dense clusters and wide regions respectively in the parameter space,
(ii) Local Entropy is a technique to artificially bias local algorithms like BP and SGD towards such regions, it puts more probability mass on atypical dense clusters of solutions which are a large deviations

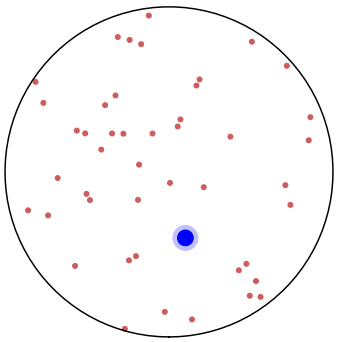


FIGURE 3.3. A cartoon of the energy landscape of the binary perceptron: Each point in the circle denotes one binary perceptron, the red dots and the blue cluster are all perceptrons which have zero training error. A replica theoretic analysis Baldassi et al. (2015) predicts that the dense blue cluster of solutions have better generalization performance than the red solutions. Note that while a perceptron has exponentially many solutions in its dimensionality, i.e., the number of red dots is exponential in d, only a constant number of dense clusters exist, each of $\mathcal{O}(1)$ width in the Hamming space. In other words, the number of solutions in a dense cluster is sub-dominant as compared to the number of isolated solutions and hence can only be discovered by a large deviations phenomenon.

phenomenon. It can therefore be seen as tilting the Gibbs measure on the solutions which puts equal mass on all solutions.

(iii) Based on the analysis of binary perceptron, we expect wide valleys to be highly rare in the parameter space and it is highly surprising that SGD finds these regions as shown in Figure 3.1.

Effectively, one may conjecture that just as minimizing Local Entropy or performing reinforced BP are means of tilting the equilibrium Gibbs distribution, SGD itself tilts this distribution. In fact, we prove this in Chapter 6 and discover a much more general phenomenon than wide regions in the parameter space; we show that the steady-state distribution of SGD is a non-equilibrium distribution which is quite different from the the Gibbs distribution and leads to many more large deviation behaviors.

3.1.2. Local Entropy for continuous parameters

For a parameter vector $x \in \mathbb{R}^d$, consider a Gibbs distribution corresponding to a given energy landscape $f(x)$:

$$\rho^{\mathrm{eq}}(x) = \frac{1}{Z_\beta} e^{-\beta f(x)}; \tag{3.6}$$

where β is the inverse temperature and Z_β is a normalizing constant, also known as the partition function. As $\beta \to \infty$, the probability distribution above concentrates on the global minimizers of $f(x)$ given by

$$x^* = \arg\min_x \ f(x), \tag{3.7}$$

which establishes the link between the Gibbs distribution and a generic optimization problem (3.7). We would instead like the probability distribution—and therefore the underlying optimization problem—to focus on wide regions such as $\widehat{x}_{f_\gamma}$ in Figure 3.2. With this in mind, let us construct a modified Gibbs distribution:

$$\rho_\gamma(y;\, x) = \frac{1}{Z_{x,\beta,\gamma}} \ \exp\left(-\beta f(y) - \frac{\beta}{2\gamma}\,|x-y|^2\right). \tag{3.8}$$

The distribution in (3.8) is a function of a dummy variable y and is parameterized by the original location x. The parameter γ biases the modified distribution (3.8) towards x; a small γ results in a $\rho(y;\, x)$ with all its mass

near x irrespective of the energy term $f(y)$. For large values of γ, the term $f(y)$ in the exponent dominates and the modified distribution is similar to the original Gibbs distribution in (3.6). For the purposes of the exposition, in this section, we will set the inverse temperature $\beta^{-1} = 1$.

Definition 3.3 (Local Entropy). Local Entropy is defined to be the negative log-partition function of the modified Gibbs distribution in (3.8) at inverse temperature $\beta^{-1} = 1$:

$$\begin{aligned} f_\gamma(x) &= -\log Z_{x,1,\gamma} \\ &= -\log \int_\Omega \exp\left(-f(y) - \frac{1}{2\gamma}\,|x-y|_2^2\right)\,\mathrm{d}y. \end{aligned} \tag{3.9}$$

The parameter γ is used to focus the modified Gibbs distribution upon a local neighborhood of x and we call it a "scope" henceforth.

Figure 3.2 shows Local Entropy for two different values of γ. We expect x_{robust} to be more robust than $x_{\text{non-robust}}$ to perturbations of data or parameters and thus generalize well and indeed, the negative Local Entropy in Figure 3.2 has a global minimum near x_{robust}. For large values of γ, the energy landscape is significantly smoother than the original landscape and possesses our desired characteristic, viz. global minimum at a wide valley. It is easy to see that as $\gamma \to 0$, the Local Entropy converges to the original loss function.

3.2. ENTROPY-SGD

The Entropy-SGD algorithm minimizes Local Entropy using gradient descent. The gradient of Local Entropy requires a sequence of Markov Chain Monte Carlo (MCMC) updates to evaluate an expectation. Obtaining the gradient is a straightforward computation and is seen to be

$$-\nabla f_\gamma(x) = \gamma^{-1}\ \langle x-y\rangle\,; \tag{3.10}$$

here the expression $\langle\cdot\rangle$ denotes an expectation over the modified Gibbs distribution (3.8):

$$\begin{aligned} \langle x-y\rangle &= \int_\Omega (x-y)\ \mathrm{d}\rho_\gamma(y;\,x) \\ &= x - \frac{1}{Z_{x,\beta,\gamma}}\ \int_\Omega y\ \exp\left(-f(y) - \frac{1}{2\gamma}|x-y|^2\right)\,\mathrm{d}y \end{aligned} \tag{3.11}$$

The Entropy-SGD algorithm solves the following problem:

$$x^*_{\text{Entropy-SGD}} = \arg\min_{x\in\Omega}\ f_\gamma(x). \tag{3.12}$$

Remark 3.4 (Connection to Shannon entropy). The quantity we have defined as Local Entropy in Definition 3.3 is different from Shannon entropy which measures the volume of likely configurations under a given distribution. For a continuous parameter space, this is given by

$$\mathrm{H}\left(\rho_\gamma(y;x)\right) = -\int_y \log\rho_\gamma(y;\,x)\ \mathrm{d}\rho_\gamma(y;\,x)$$

for the Gibbs distribution in (3.8). Minimizing Shannon entropy however does not differentiate between wide regions that have very high loss versus wide regions that lie deeper in the energy landscape. For instance in Figure 3.2, Shannon entropy is smallest in the neighborhood of $x_{\text{candidate}}$ which is a large region with very high loss on the training dataset and is unlikely to generalize well. We could minimize a modified loss function of the form $f(x) + \lambda^{-1}\mathrm{H}(\rho_\gamma)$ whose gradient can be computed to be

$$\nabla f(x) + \lambda^{-1}\nabla\,\mathrm{H}(\rho_\gamma) = \nabla f(x) - \frac{1}{\lambda\gamma}\,\mathrm{corr}(g(y),\,x-y); \tag{3.13}$$

where we have defined the cross-correlation as

$$\mathrm{corr}(g(y),\,x-y) := \langle g(y)\ (x-y)\rangle - \langle g(y)\rangle\,\langle x-y\rangle\,;$$

with $g(y) = f(y) + \frac{\gamma}{2}\,|x-y|^2$.

3.2.1. Evaluating the gradient

The integral in (3.11) is high-dimensional and as such it is difficult to estimate it accurately. In this section, we discuss stochastic gradient Langevin dynamics (Welling and Teh, 2011) as a way to evaluate such integrals.

For a parameter vector $x \in \mathbb{R}^d$ with a prior distribution $p(x)$ and if the probability of generating a data item $\xi_k \in \Xi$ given a model parameterized by x is $p(\xi_k \,|\, x)$, the posterior distribution of the parameters based on N data items can be written as

$$p\,(x \,|\, \xi_{\,k \leq N}) \;\propto\; p(x) \prod_{k=1}^{N} p\,(\xi_k \,|\, x)\,. \tag{3.14}$$

Langevin dynamics (Neal, 2011) injects Gaussian noise into maximum-a-posteriori (MAP) updates to prevent over-fitting the solution x^* of the above equation. The updates can be written as

$$\Delta x_t = \frac{\eta}{2} \left(\nabla \log p(x_t) + \sum_{k=1}^{N} \nabla p(\xi_k \,|\, x_t) \right) + \sqrt{\eta}\; \varepsilon_t; \tag{3.15}$$

where $\varepsilon_t \sim \mathrm{N}(0, \varepsilon^2)$ is Gaussian noise and η is the learning rate. In this form, Langevin dynamics faces two major hurdles for applications to large datasets. First, computing the gradient $\sum_{k=1}^{N} \nabla p(\xi_k \,|\, x_t)$ over all samples for each update Δx_t becomes prohibitive. However, as Welling and Teh (2011) show, one can instead simply use the average gradient over m data samples (mini-batch) as follows:

$$\Delta x_t = \frac{\eta_t}{2} \left(\nabla \log p(x_t) + \frac{N}{m} \sum_{k=1}^{m} \nabla p(\xi_k \,|\, x_t) \right) + \sqrt{\eta_t}\; \varepsilon_t. \tag{3.16}$$

Secondly, Langevin dynamics in (3.15) is the discrete-time approximation of a continuous-time stochastic differential equation (Mandt et al., 2016a) thereby necessitating a Metropolis-Hastings (MH) rejection step (Roberts and Stramer, 2002) which again requires computing $p(\xi_k \,|\, x)$ over the entire dataset. However, if the learning rate $\eta_t \to 0$, we can also forgo the MH step (Chen et al., 2014). Welling and Teh (2011) also argue that the sequence of samples x_t generated by updating (3.16) converges to the correct posterior (3.14) and one can hence compute the statistics of any function $g(x)$ of the parameters using these samples. Concretely, the posterior expectation $\mathbb{E}\,[g(x)]$ is given by

$$\mathbb{E}\,[g(x)] \approx \frac{\sum_{s=1}^{t} \eta_t \; g(x_t)}{\sum_{s=1}^{t} \eta_t};$$

which is the average computed by weighing each sample by the corresponding learning rate in (3.16). We will employ a uniform prior on the parameters x and hence the first term in (3.16), viz., $\nabla \log p(x_t)$ vanishes.

Let us note that there is a variety of increasingly sophisticated MCMC algorithms applicable to our problem, e.g., Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) by Chen et al. (2014) based on volume preserving flows in the "parameter-momentum" space, stochastic annealing thermostats (Santa) by Chen et al. (2015a) etc. We can also employ these techniques, although we use SGLD for ease of implementation; the authors in Ma et al. (2015) provide an elaborate overview.

3.2.2. Algorithm and Implementation details

Algorithm 2 provides the pseudo-code for one iteration of the Entropy-SGD algorithm. At each iteration, lines 2-5 perform L iterations of Langevin dynamics to estimate $z = \langle y \rangle$. The weights x are updated with the modified gradient on line 6.

Algorithm 2: Entropy-SGD algorithm

Input : current weights x, Langevin iterations L
Hyper-parameters : scope γ, learning rate η, SGLD step size η'

// SGLD iterations;
1 $y,\ z \leftarrow x$;
2 **for** $\ell \leq L$ **do**
3 $\quad \mathrm{d}y \leftarrow \nabla f_b(y) - \gamma^{-1}\ (x-y)$;
4 $\quad y \leftarrow y - \eta\ \mathrm{d}y + \sqrt{2\eta\beta^{-1}}\ \mathrm{N}(0,\mathrm{I})$;
5 $\quad z \leftarrow \alpha z + (1-\alpha)\ y$;

// Update weights;
6 $x \leftarrow x - \eta'\ \gamma^{-1}\ (x-z)$

Although we have written Algorithm 2 in the classical SGD setup, we can easily modify it to include techniques such as momentum and gradient pre-conditioning (Duchi et al., 2011) by changing lines 5 and 7. In our experiments, we have used SGD with Nesterov's momentum (Sutskever et al., 2013) and Adam for outer and inner loops with similar qualitative results. We use exponential averaging to estimate z in the SGLD loop (line 5) with $\alpha = 0.25$ so as to put more weight on the later samples, this is akin to a burn-in in standard SGLD.

The number of SGLD iterations is set to $L \in [5, 20]$ depending upon the complexity of the dataset. The parameter β^{-1} in SGLD on line 4 is the thermal noise and we fix this to $\beta^{-1} \in [10^{-8}, 10^{-6}]$. The remaining parameter γ can be tuned by matching the magnitude of the gradient of the Local Entropy term $\gamma^{-1}\ (x-z)$ with that of the gradient for vanilla SGD $\nabla f_b(x)$.

3.2.3. Scoping of γ

The scope γ is fixed in Algorithm 2. For small values of γ, the SGLD updates happen in a small neighborhood of the current parameters x while large values of γ allow SGLD to explore further away from x. Annealing the scope γ, i.e. decreasing γ as training progresses has the effect of exploring the energy landscape on progressively finer scales. We call this process "scoping" which is similar to that of Baldassi et al. (2016a) and use a simple exponential schedule given by

$$\gamma(t) = \gamma_0\ (1-\gamma_1)^t;$$

for the t^{th} parameter update. We have experimented with linear, quadratic and bounded exponential scoping schedules and obtained qualitatively similar results.

Scoping of γ interferes with the learning rate annealing that is typically done in deep learning; this is a direct consequence of the update step on line 6 of Algorithm 2. In practice, we therefore scale down the Local Entropy gradient by γ before the weight update and modify the line to read

$$x \leftarrow x - \eta'(x-z).$$

We can further set $\eta' = 1$ which simply gives $x \leftarrow z$ as the update on line 6.

3.2.4. Theoretical Properties

We show that Entropy-SGD results in a smoother loss function and obtains better generalization error than the original loss function $f(x)$. With some overload of notation, we assume that the original loss $f(x)$ is μ-smooth, i.e., for all $x, y \in \Omega$, we have

$$|\nabla f(x) - \nabla f(y)| \leq \mu\ |x-y|.$$

We additionally assume for the purpose of analysis that no eigenvalue of the Hessian $\nabla^2 f(x)$ lies in the set $[-2\gamma^{-1} - c, c]$ for some small $c > 0$.

Lemma 3.5. The objective $f_\gamma(x)$ is $\frac{\alpha}{1+\gamma\, c}$-Lipschitz and $\frac{\mu}{1+\gamma\, c}$-smooth.

Proof. The gradient $\nabla f_\gamma(x)$ is $\gamma^{-1}\,\langle x-y\rangle$. Consider the term

$$\begin{aligned}\langle x-y\rangle &= Z^{-1}\,\int_y y e^{-f(y)-\frac{1}{2\gamma}|x-y|^2}\,\mathrm{d}y \\ &\approx x - Z^{-1}\,\int_y (x+s)\; e^{-f(x)-\nabla f(x)^\top s-\frac{1}{2}s^\top\left(\nabla^2 f(x)+\gamma^{-1}I\right)s}\,\mathrm{d}s \\ &= -Z^{-1}\,\int_y s\; e^{-f(x)-\nabla f(x)^\top s-\frac{1}{2}s^\top\left(\nabla^2 f(x)+\gamma^{-1}I\right)s}\,\mathrm{d}s\end{aligned}$$

We can estimate the above integral using a Laplace approximation to be

$$\langle x-y\rangle \approx \left(\nabla^2 f(x)+\gamma^{-1}I\right)^{-1}\nabla f(x);$$

this approximation is exact if the objective is locally convex. If

$$A^{-1} = \gamma\,\nabla^2 f(x) + I$$

we have

$$\begin{aligned}|\nabla f_\gamma(x)-\nabla f_\gamma(y)| &= |A(x)\;\nabla f(x) - A(y)\;\nabla f(y)| \\ &\leq \left(\sup_x |A(x)|\right)\;\mu\;|x-y|.\end{aligned}$$

We get a uniform bound if we assume that for a small constant $c>0$, no eigenvalue of $\nabla^2 f(x)$ lies in the set $[-2\gamma^{-1}-c, c]$. This gives

$$\left(\sup_x |A(x)|\right) \leq \frac{1}{1+\gamma\, c}.$$

This shows that a large value of γ results in a smoother energy landscape, except at places with very flat directions. The Lipschitz constant α of the original loss $f(x)$ decreases by the same factor. ■

The Local Entropy objective is thus smoother than the original objective. Let us now obtain a bound on the improvement in generalization error. We denote an optimization algorithm, viz., SGD or Entropy-SGD by $A(\Xi)$, it is a function of the dataset Ξ and outputs the parameters x^* upon termination. Stability of the algorithm (Bousquet and Elisseeff, 2002) is then a notion of how much its output differs in loss upon being presented with two datasets, Ξ and Ξ', that differ in at most one sample:

$$\sup_{\xi\,\in\,\Xi\,\cup\,\Xi'} \left[f\left(A(\Xi),\xi\right) - f\left(A(\Xi'),\xi\right)\right] \leq \varepsilon.$$

Hardt et al. (2016) connect uniform stability to generalization error and show that an ε-stable algorithm $A(\Xi)$ has generalization error bounded by ε, i.e., if $A(\Xi)$ terminates with parameters x^*,

$$\left|\mathop{\mathbb{E}}_{\Xi}\left(R_\Xi(x^*) - R(x^*)\right)\right| \leq \varepsilon;$$

where the left hand side is the generalization error: it is the difference between the empirical loss $R_\Xi(x) := N^{-1}\sum_{k=1}^N\; f(x,\xi_k)$ and the population loss $R(x) := \mathbb{E}_\xi\; f(x,\xi)$. We now employ the following theorem that bounds the stability of an optimization algorithm through the smoothness of its loss function and the number of iterations on the training set.

Theorem 3.6 (Hardt et al. (2016)). For an α-Lipschitz and μ-smooth loss function, if SGD converges in T iterations on N samples with decreasing learning rate $\eta_t \leq 1/t$ the stability is bounded by

$$\begin{aligned}\varepsilon &\leq \frac{1}{N-1}\;\left(1+\frac{1}{\beta c}\right)\;\left(2c\alpha^2\right)^{\frac{1}{1+\beta c}}\;T^{\frac{\beta c}{1+\beta c}} \\ &\lessapprox \frac{1}{N}\;\alpha^{(1+\beta)^{-1}}\;T^{1-(1+\beta)^{-1}}.\end{aligned}$$

Using Lemma 3.5 and Theorem 3.6 we have

$$\varepsilon_{\text{Entropy-SGD}} \lessapprox \left(\alpha\, T^{-1}\right)^{\left(1-\frac{1}{1+\gamma\, c}\right)\mu}\ \varepsilon_{\text{ SGD}}, \tag{3.17}$$

which shows that Entropy-SGD generalizes better than SGD for all $T > \alpha$ if they are both executed for T epochs over the dataset.

Let us note that while the number of passes over the dataset for Entropy-SGD and SGD are similar for our experiments on CNNs, Entropy-SGD makes only half as many passes as SGD for our experiments on RNNs. As an aside, it is easy to see from the proof of Lemma 3.5 that for a convex loss function $f(x)$, the Local Entropy objective does not change the minimizer of the original problem.

Remark 3.7. The above analysis hinges upon an assumption that the Hessian $\nabla^2 f(x)$ does not have eigenvalues in the set $[-2\gamma^{-1} - c, c]$ for a constant $c > 0$. This is admittedly unrealistic, for instance, the eigenspectrum of the Hessian at a local minimum in Figure 3.1 has a large fraction of its eigenvalues almost zero. Let us however remark that the result in Theorem 3.6 by Hardt et al. (2016) assumes global conditions on the smoothness of the loss function; one imagines that (3.17) remains qualitatively the same (with respect to T in particular) even if this assumption is violated to an extent before convergence happens. Obtaining a rigorous generalization bound without this assumption would require a dynamical analysis of SGD and seems out of reach currently.

3.3. Experiments

In Section 3.3.1, we discuss experiments that suggest that the characteristics of the energy landscape around local minimal accessible by SGD are universal to deep architectures. We then present experimental results on two standard image classification datasets, viz. MNIST and CIFAR-10 and two datasets for text prediction, viz. PTB and the text of War and Peace. Table 3.1 summarizes the results of these experiments on deep networks.

3.3.1. Universality of the Hessian at local minima

We use automatic differentiation[1] to compute the Hessian at a local minimum obtained at the end of training for the following networks:

(i) **small-LeNet**: This network has $47,658$ parameters and is similar to lenet but with 10 and 20 channels respectively in the first two convolutional layers and 128 hidden units in the fully-connected layer. We train this with Adam to obtain a validation error of 2.4%.

(ii) **smallfc**: A fully-connected network ($50,890$ parameters) with one layer of 32 hidden units, ReLU non-linearities and cross-entropy loss; it converges to a validation error of 2.5% with momentum-based SGD.

(iii) **char-lstm for text generation**: This is a recurrent network with 48 hidden units and Long Short-Term Memory (LSTM) architecture (Hochreiter and Schmidhuber, 1997b). It has $32,640$ parameters and we train it with Adam to re-generate a small piece of text consisting of 256 lines of length 32 each and 96-bit one-hot encoded characters.

(iv) **allcnn on CIFAR-10**: This is similar to the All-CNN-C network (Springenberg et al., 2014) with ≈ 1.6 million weights (cf. Section 3.3.3) which we train using Adam to obtain an error of 11.2%. Exact Hessian computation is in this case expensive and thus we instead compute the diagonal of the Fisher information matrix (Wasserman, 2013) using element-wise first and second moments of the gradients that Adam maintains, i.e.,

$$\text{diag I} = \mathbb{E}(g^2) - (\mathbb{E}\ g)^2$$

where g is the back-propagated gradient. Fisher information measures the sensitivity of the log-likelihood of data given parameters in a neighborhood of a local minimum and thus is exactly equal to the Hessian of the negative log-likelihood. We will consider the diagonal of the empirical Fisher information matrix as a proxy for the eigenvalues of the Hessian, as is common in the literature.

[1] https://github.com/HIPS/autograd

We choose to compute the exact Hessian and to keep the computational and memory requirements manageable, the first three networks considered above are smaller than standard deep networks used in practice. For the last network, we sacrifice the exact computation and instead approximate the Hessian of a large deep network. We note that recovering an approximate Hessian from Hessian-vector products (Pearlmutter, 1994) could be a viable strategy for large networks.

Figure 3.1 in the beginning of this chapter shows the eigenspectrum of the Hessian for small-LeNet while Figure 3.4 shows the eigenspectra for the other three networks. A large proportion of eigenvalues of the Hessian are very close to zero or positive with a very small (relative) magnitude. This suggests that the local geometry of the energy landscape is almost flat at local minima discovered by gradient descent. This agrees with theoretical results such as Baldassi et al. (2016c) where the authors predict that wide regions of the landscape generalize better. Standard regularizers in deep learning such as convolutions, max-pooling and dropout seem to bias SGD towards wider regions in the energy landscape. Away from the origin, the right tails of the eigenspectra are much longer than the left tails. Indeed, as discussed in numerous places in literature (Bray and Dean, 2007; Dauphin et al., 2014; Choromanska et al., 2015), SGD finds low-index critical points, i.e., optimizers with few negative eigenvalues of the Hessian. What is interesting and novel is that the directions of descent that SGD misses do not have a large curvature.

3.3.2. MNIST

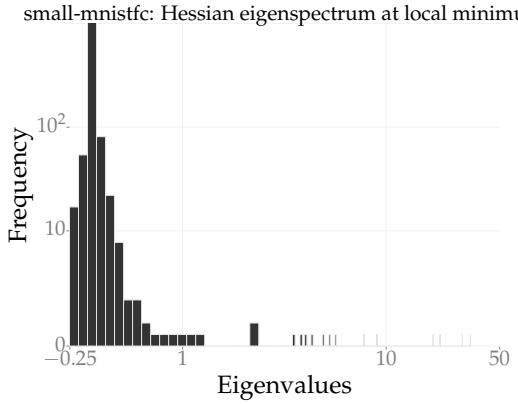
We consider two prototypical networks: the first is called mnistfc and is a fully-connected network with two hidden layers of 1024 units each and the second is a convolutional neural network with the same size as lenet but with batch-normalization (Ioffe and Szegedy, 2015); both use a dropout of probability 0.5 after each layer. As a baseline, we train for 100 epochs with Adam and a learning rate of 10^{-3} that drops by a factor of 5 after every 30 epochs to obtain an average error of $1.39 \pm 0.03\%$ and $0.51 \pm 0.01\%$ for mnistfc and lenet respectively over 5 independent runs.

For both these networks, we train Entropy-SGD for 5 epochs with $L = 20$ and reduce the dropout probability (0.15 for mnistfc and 0.25 for lenet). The learning rate of the SGLD updates is fixed to $\eta = 0.1$ while the outer loop's learning rate is set to $\eta' = 1$ and drops by a factor of 10 after the second epoch; we use Nesterov's momentum for both loops. The thermal noise in SGLD updates (line 4 of Algorithm 2) is set to $\beta^{-1} = 10^{-6}$. We use an exponentially increasing value of γ for scoping, the initial value of the scope is set to $\gamma = 10^4$ and this decreases by a factor of $1 + 10^{-3}$ after each parameter update. The results in Figure 3.5a and Figure 3.5b show that Entropy-SGD obtains a comparable generalization error: $1.37 \pm 0.03\%$ and $0.50 \pm 0.01\%$, for mnistfc and lenet respectively. While Entropy-SGD trains slightly faster in wall-clock time for lenet; it is marginally slower for mnistfc which is a small network and trains in about two minutes anyway.

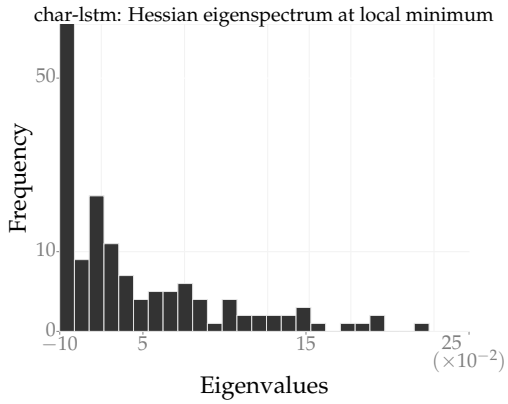
Remark on the computational complexity: Since Entropy-SGD runs L steps of SGLD before each parameter update, the effective number of passes over the dataset is L times that of SGD or Adam for the same number of parameter updates. We therefore plot the error curves of Entropy-SGD in Figures 3.5–3.7 against the "effective number of epochs", i.e. by multiplying the abscissæby a factor of L. We define $L = 1$ for SGD or Adam. This is a direct measure of wall-clock time, agnostic to the underlying hardware and implementation.

3.3.3. CIFAR-10

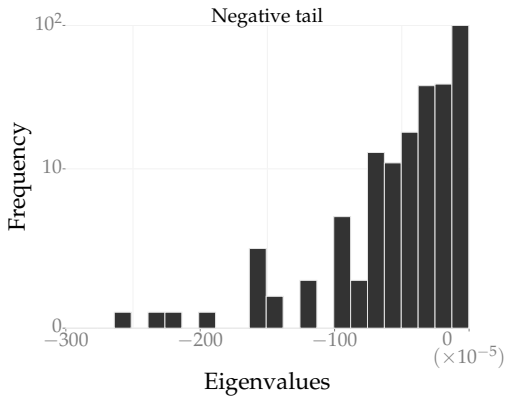
We train on CIFAR-10 without data augmentation after performing global contrast normalization and ZCA whitening (Goodfellow et al., 2013). We consider the All-CNN-C network of Springenberg et al. (2014) with the only difference that a batch normalization layer is added after each convolutional layer; all other architecture and hyper-parameters are kept the same. We train for 200 epochs with SGD and Nesterov's momentum during which the initial learning rate of 0.1 decreases by a factor of 5 after every 60 epochs. We obtain an average error of $7.71 \pm 0.19\%$ in 200 epochs vs. 9.08% error in 350 epochs that the authors in Springenberg et al. (2014) report and this is thus a very competitive baseline for this network. Let us note

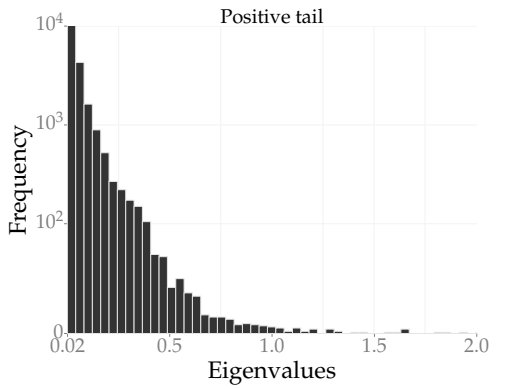


(A) small-mnistfc (2 runs): Peak (clipped here) at zero ($|\lambda| \leq 10^{-2}$) accounts for 90% of the entries.



(B) char-LSTM (5 runs): Almost 95% eigenvalues have absolute value below 10^{-5}.



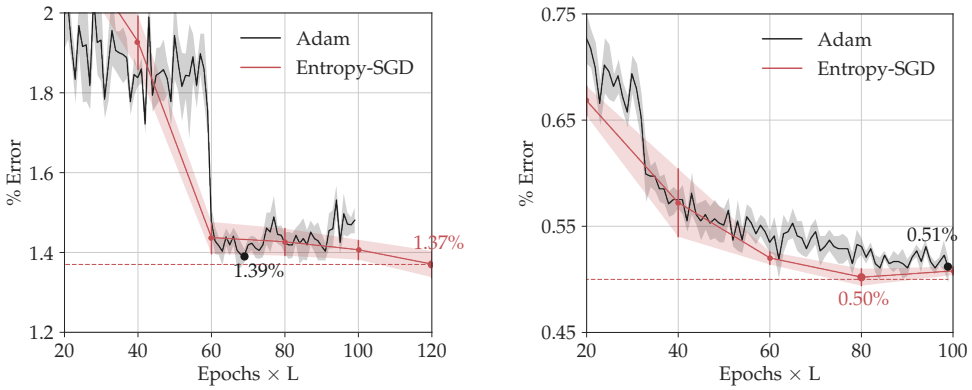


(C) Negative and positive eigenvalues of the Fisher information matrix of allcnn at a local minimum (4 independent runs). The origin has a large peak with $\approx 95\%$ near-zero ($|\lambda| \leq 10^{-5}$) eigenvalues (clipped here).

FIGURE 3.4. Universality of the Hessian: for a wide variety of network architectures, sizes and datasets, optima obtained by SGD are mostly wide (large peak near zero), they always have a few directions with large positive curvature (long positive tails). A very small fraction of directions have negative curvature, and the magnitude of this curvature is extremely small (short negative tails).

that the best result in the literature on non-augmented CIFAR-10 is the ELU-network by Clevert et al. (2015) with 6.55% validation error.

We train Entropy-SGD with $L = 20$ for 10 epochs with the original dropout of 0.5. The initial learning rate of the outer loop is set to $\eta' = 1$ and drops by a factor of 5 every 4 epochs, while the learning rate of the SGLD updates is fixed to $\eta = 0.1$ with thermal noise $\beta^{-1} = 10^{-8}$. As the scoping scheme, we set the initial value of the scope to $\gamma_0 = 30$ which decreases by a factor of 1.001 after each parameter update. Figure 3.6 shows the training and validation error curves for Entropy-SGD compared with SGD. It shows that Local Entropy performs as well as SGD on a large CNN; we obtain a validation error of $7.81 \pm 0.09\%$ in about 160 effective epochs.

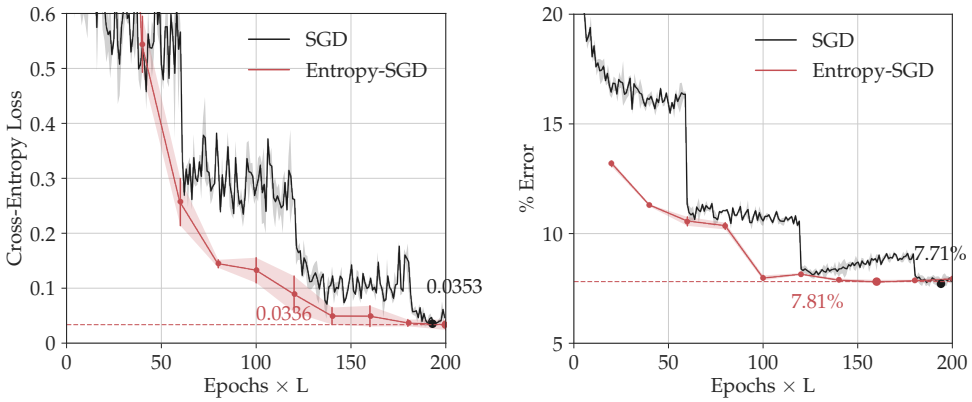


(A) mnistfc: Validation error

(B) lenet: Validation error

FIGURE 3.5. Entropy-SGD vs. Adam on MNIST

We see almost no plateauing of training loss or validation error for Entropy-SGD in Figure 3.6a; this trait is shared across different networks and datasets in our experiments and is an indicator of the additional smoothness of the Local Entropy landscape coupled with a good scoping schedule for γ.



(A) allcnn: Training loss

(B) allcnn: Validation error

FIGURE 3.6. Entropy-SGD vs. SGD on CIFAR-10

3.3.4. Penn Tree Bank

We train an LSTM network on the Penn Tree Bank (PTB) dataset for word-level text prediction. This dataset contains about one million words divided into a training set of about 930,000 words, a validation set of 74,000 words and 82,000 words with a vocabulary of size 10,000. Our network called PTB-LSTM consists of two layers with 1500 hidden units, each unrolled for 35 time steps; note that this is a large network with about 66 million weights. We recreated the training pipeline of Zaremba et al. (2014) for this network (SGD without momentum) and obtained a word perplexity of 81.43 ± 0.2 on the validation set and 78.6 ± 0.26 on the test set with this setup; these numbers closely match the results of the original authors.

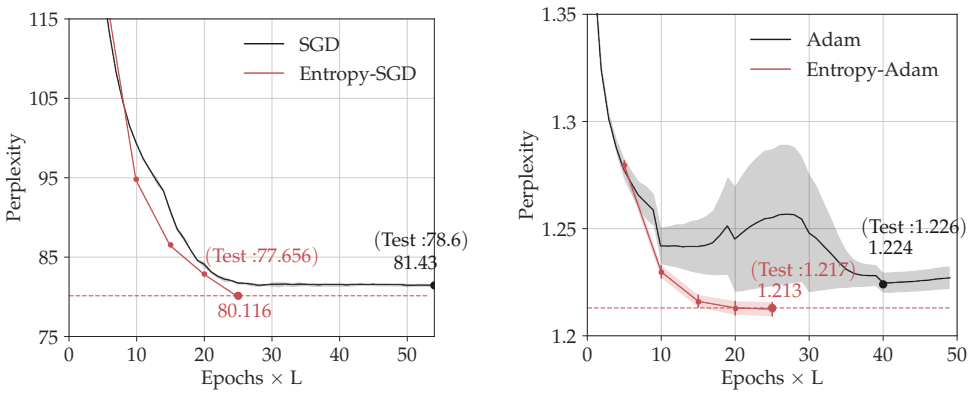
We run Entropy-SGD on PTB-LSTM for 5 epochs with $L = 5$, note that this results in only 25 effective epochs. We do not use scoping for this network and instead fix $\gamma = 10^3$. The initial learning rate of the outer loop is $\eta' = 1$ which reduces by a factor of 10 at each epoch after the third epoch. The SGLD learning rate is

fixed to $\eta = 1$ with $\beta^{-1} = 10^{-8}$. We obtain a word perplexity of 80.116 ± 0.069 on the validation set and 77.656 ± 0.171 on the test set. As Figure 3.7a shows, Entropy-SGD trains significantly faster than SGD (25 effective epochs vs. 55 epochs of SGD) and also achieves a slightly better generalization perplexity.

3.3.5. char-LSTM on War and Peace

Next, we train an LSTM to perform character-level text-prediction. As a dataset, following the experiments of Karpathy et al. (2015), we use the text of War and Peace by Leo Tolstoy which contains about 3 million characters divided into training (80%), validation (10%) and test (10%) sets. We use an LSTM consisting of two layers of 128 hidden units unrolled for 50 time steps and a vocabulary of size 87. We train the baseline with Adam for 50 epochs with an initial learning rate of 0.002 that decays by a factor of 2 after every 5 epochs to obtain a validation perplexity of 1.224 ± 0.008 and a test perplexity of 1.226 ± 0.01.

As noted in Section 3.2.2, we can easily wrap Algorithm 2 inside other variants of SGD such as Adam; this involves simply substituting the Local Entropy gradient in place of the usual back-propagated gradient. For this experiment, we constructed Entropy-Adam which is equivalent to Adam with the Local Entropy gradient (which is computed via SGLD). We run Entropy-Adam for 5 epochs with $L = 5$ and a fixed $\gamma = 100$ with an initial learning rate of $\eta' = 0.01$ that decreases by a factor of 2 at each epoch. Note that this again results in only 25 effective epochs, i.e. half as much wall-clock time as SGD or Adam. We obtain a validation perplexity of 1.213 ± 0.007 and a test perplexity of 1.217 ± 0.005 over 4 independent runs which is better than the baseline. Figure 3.7b shows the error curves for this experiment.



(A) PTB-LSTM: Validation, test perplexity

(B) char-LSTM: Validation, test perplexity

FIGURE 3.7. Entropy-SGD vs. SGD/Adam on RNNs

Model	Entropy-SGD		SGD / Adam	
	Error (%) / Perplexity	Epochs	Error (%) / Perplexity	Epochs
mnistfc	1.37 ± 0.03	120	1.39 ± 0.03	66
lenet	0.5 ± 0.01	80	0.51 ± 0.01	100
allcnn	7.81 ± 0.09	160	7.71 ± 0.19	180
PTB-LSTM	77.656 ± 0.171	25	78.6 ± 0.26	55
char-LSTM	1.217 ± 0.005	25	1.226 ± 0.01	40

TABLE 3.1. Experimental results: Entropy-SGD vs. SGD / Adam

Tuning the momentum in Entropy-SGD was crucial to getting good results on RNNs. While the SGD baseline on PTB-LSTM does not use momentum (and in fact, does not train well with momentum) we used a momentum

of 0.5 for Entropy-SGD. On the other hand, the baseline for char-LSTM was trained with Adam with $\beta_1 = 0.9$ (β_1 in Adam controls the moving average of the gradient) while we set $\beta_1 = 0.5$ for Entropy-Adam. In contrast to this observation about RNNs, all our experiments on CNNs used a momentum of 0.9. In order to investigate this difficulty, we monitored the angle between the Local Entropy gradient and the vanilla SGD gradient during training. This angle changes much more rapidly for RNNs than for CNNs which suggests a more rugged energy landscape for the former; Local Entropy gradient is highly uncorrelated with the SGD gradient in this case.

3.3.6. Comparison with SGLD

We use stochastic gradient Langevin dynamics to estimate the gradient of Local Entropy in Algorithm 2. It is natural then, to ask the question whether vanilla SGLD performs as well as Local Entropy. To this end, we compare the performance of SGLD on two prototypical networks: lenet on MNIST and allcnn on CIFAR-10. We follow the experiments in Welling and Teh (2011) and Chen et al. (2015a) and set the learning rate schedule to be $\eta/(1+t)^b$ where the initial learning rate η and b are hyper-parameters. We make sure that other architectural aspects (dropout, batch-normalization) and regularization (weight decay) are consistent with the experiments in Section 3.3.

After a hyper-parameter search, we obtained a validation error on lenet of $0.63 \pm 0.1\%$ after 300 epochs and $9.89 \pm 0.11\%$ on allcnn after 500 epochs. Even if one were to disregard the slow convergence of SGLD, its generalization error is much worse than our experimental results; we get $0.50 \pm 0.01\%$ on lenet and $7.81 \pm 0.09\%$ on allcnn with Entropy-SGD. For comparison, the authors in Chen et al. (2015a) report 0.71% error on MNIST on a slightly larger network. Our results with Local Entropy on RNNs are much better than those reported in Gan et al. (2016) for SGLD. On the PTB dataset, we obtain a test perplexity of 77.656 ± 0.171 vs. 94.03 for the same model whereas we obtain a test perplexity of 1.213 ± 0.007 vs. 1.3375 for char-LSTM on the War and Peace dataset.

Training deep networks with SGLD, or other more sophisticated MCMC algorithms such as SGHMC, SGNHT etc. (Chen et al., 2014; Neal, 2011) to errors similar to those of SGD is difficult, and the lack of such results in the literature corroborates our experimental experience. Roughly speaking, Local Entropy is so effective because it operates on a transformation of the energy landscape that exploits entropic effects. Conventional MCMC techniques such as SGLD or Nose'-Hoover thermostats (Ding et al., 2014) can only trade energy for entropy via the temperature parameter which does not allow the direct use of the geometric information of the energy landscape and does not help with narrow minima.

3.4. Related work

Our above observation about the spectrum of Hessian (further discussed in Section 3.3) is similar to results on a perceptron model in Dauphin et al. (2014) where the authors connect the loss function of a deep network to a high-dimensional Gaussian random field. They also relate to earlier studies such as Baldi and Hornik (1989); Fyodorov and Williams (2007); Bray and Dean (2007) which show that critical points with high training error are exponentially likely to be saddle points with many negative directions and all local minima are likely to have error that is very close to that of the global minimum. The authors also argue that convergence of gradient descent is affected by the proliferation of saddle points surrounded by high error plateaus — as opposed to multiple local minima. One can also see this via an application of Kramer's law: the time spent by diffusion is inversely proportional to the smallest negative eigenvalue of the Hessian at a saddle point (Bovier and den Hollander, 2006).

The existence of multiple — almost equivalent — local minima in deep networks has been predicted using a wide variety of theoretical analyses and empirical observations, e.g., papers such as Choromanska et al. (2015); Chaudhari and Soatto (2015) that build upon results from statistical physics as also others such as Haeffele and Vidal (2015) and Janzamin et al. (2015) that obtain similar results for matrix and tensor factorization problems. Although assumptions in these works are somewhat unrealistic in the context of deep networks used in practice,

similar results are also true for linear networks which afford a more thorough analytical treatment (Saxe et al., 2013). For instance, Soudry and Carmon (2016) show that with mild over-parameterization and dropout-like noise, training error for a neural network with one hidden layer and piece-wise linear activation is zero at every local minimum. All these results suggest that the energy landscape of deep neural networks should be easy to optimize and they more or less hold in practice—it is easy to optimize a prototypical deep network to near-zero loss on the training set (Hardt et al., 2016; Goodfellow and Vinyals, 2014).

Obtaining good generalization error, however, is challenging: complex architectures are sensitive to initial conditions and learning rates (Sutskever et al., 2013) and even linear networks (Kawaguchi, 2016) may have degenerate and hard to escape saddle points (Ge et al., 2015; Anandkumar and Ge, 2016). Techniques such as adaptive (Duchi et al., 2011) and annealed learning rates, momentum (Tieleman and Hinton, 2012), as well as architectural modifications like dropout (Srivastava et al., 2014), batch-normalization (Ioffe and Szegedy, 2015; Cooijmans et al., 2016), weight scaling (Salimans and Kingma, 2016) etc. are different ways of tackling this issue by making the underlying landscape more amenable to first-order algorithms. However, the training process often requires a combination of such techniques and it is unclear beforehand to what extent each one of them helps.

Closer to the subject of this chapter are results by Baldassi et al. (2015, 2016a,b) who show that the energy landscape of shallow networks with discrete weights is characterized by an exponential number of isolated minima and few very dense regions with lots of local minima close to each other. These dense local minima can be shown to generalize well for random input data; more importantly, they are also accessible by efficient algorithms using a novel measure called "robust ensemble" that amplifies the weight of such dense regions. The authors use belief propagation to estimate Local Entropy for simpler models such as committee machines considered there. A related work in this context is Elastic-SGD (Zhang et al., 2015a) which trains multiple deep networks in parallel and modulates the distance of each worker from the ensemble average. Such an ensemble training procedure enables improved generalization by ensuring that different workers land in the same wide valley and indeed, it turns out to be closely related to the replica theoretic analysis of Baldassi et al. (2016a).

Our work generalizes the Local Entropy approach above to modern deep networks with continuous weights. It exploits the same phenomenon of wide valleys in the energy landscape but does so without incurring the hardware and communication complexity of replicated training or being limited to models where one can estimate Local Entropy using belief propagation. The enabling technique in our case is using Langevin dynamics for estimating the gradient of e, which can be done efficiently even for large deep networks using mini-batch updates.

Motivated by the same final goal, viz. flat local minima, the authors in Hochreiter and Schmidhuber (1997a) introduce hard constraints on the training loss and the width of local minima and show using the Gibbs formalism (Haussler and Opper, 1997) that this leads to improved generalization. As the authors discuss, the effect of hyper-parameters for the constraints is intricately tied together and they are difficult to choose even for small problems. Our Local Entropy based objective instead naturally balances the energetic term (training loss) and the entropic term (width of the valley). The role of γ is clear as a focusing parameter (cf. Section 3.2.3) and effectively exploiting this provides significant computational advantages. Among other conceptual similarities with our work, let us note that Local Entropy in a flat valley is a direct measure of the width of the valley which is similar to their usage of Hessian, while the Gibbs variant to averaging in weight space (Equation 33 of Hochreiter and Schmidhuber (1997a)) is similar to our (3.10). Indeed, Gibbs formalism used in their analysis is a very promising direction to understanding generalization in deep networks (Zhang et al., 2016).

Homotopy continuation methods convolve the loss function to solve sequentially refined optimization problems (Allgower and Georg, 2012; Mobahi and Fisher III, 2015), similarly, methods that perturb the weights or activations to average the gradient (Gulcehre et al., 2016b) do so with an aim to smooth the rugged energy landscape. Such smoothing is however very different from e. For instance, the latter places more weight on wide local minima even if they are much shallower than the global minimum (cf. Figure 3.2); this effect cannot

be obtained by smoothing. In fact, smoothing can introduce an artificial minimum between two nearby sharp valleys which is detrimental to generalization. In order to be effective, continuation techniques also require that minimizers of successively smaller convolutions of the loss function lie close to each other (Hazan et al., 2016); it is not clear whether this is true for deep networks. Local Entropy, on the other hand, exploits wide minima which have been shown to exist in a variety of learning problems (Monasson and Zecchina, 1995; Cocco et al., 1996).

CHAPTER 4

Smoothing of non-convex energy landscapes

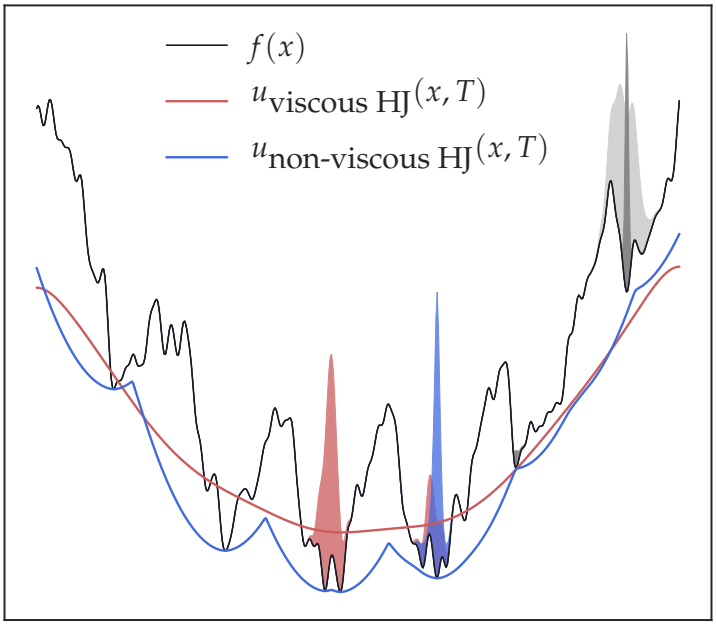


FIGURE 4.1. One dimensional loss function $f(x)$ (black) and smoothing of $f(x)$ by the viscous Hamilton-Jacobi equation (red) and the non-viscous Hamilton Jacobi equation (blue). The initial density (light gray) is evolved using the Fokker-Planck equation (4.33) which uses solutions of the respective PDEs. The terminal density obtained using SGD (dark gray) remains concentrated in a local minimum. The function $u_{\text{viscous HJ}}(x,T)$ (red) is convex, with the global minimum close to that of $f(x)$, consequently the terminal density (red) is concentrated near the minimum of $f(x)$. The function $u_{\text{non-viscous HJ}}(x,T)$ (blue) is non-convex, so the presence of gradients pointing away from the global minimum slows down the evolution. Nevertheless, the corresponding terminal density (blue) is concentrated in a nearly optimal local minimum. Note that $u_{\text{non-viscous HJ}}(x,T)$ can be interpreted as a generalized convex envelope of f, it is given by the Hopf-Lax formula (4.16). Also note that both the value and location of some local extrema for non-viscous HJ are unchanged, while the regions of convexity widen. On the other hand, local maxima located in smooth concave regions see their concave regions shrink, value eventually decreases. This figure was produced by solving (4.33) using monotone finite differences (Oberman, 2006).

In Chapter 3, we introduced a modification of the loss function of a neural network called Local Entropy which we denoted by $f_\gamma(x)$. The authors in Chaudhari et al. (2016) showed that this modification is effective for

training modern, high-dimensional deep neural networks. Our first result is to show that $f_\gamma(x)$ is the solution of a nonlinear partial differential equation (PDE); see Section 4.2. Define the viscous Hamilton-Jacobi PDE

$$\frac{\partial u}{\partial t} = -\frac{1}{2}|\nabla u|^2 + \frac{\beta^{-1}}{2}\Delta u, \qquad \text{for } 0 < t \le \gamma \tag{4.1}$$

along with an initial value $u(x,0) = f(x)$. The Laplacian is defined as

$$\Delta u = \sum_{i=1}^{d} \frac{\partial u}{\partial x_i^2}$$

We first show in this chapter that

$$f_\gamma(x) = u(x,\gamma)$$

where $u(x,t)$ is the solution of (4.1). Using standard semi-concavity estimates in Section 4.5, the PDE interpretation immediately leads to regularity results for the solutions that are consistent with empirical observations in Section 3.3.

Algorithmically, our starting point is the continuous-time stochastic gradient descent equation,

$$\mathrm{d}X(t) = -\nabla f(X)\,\mathrm{d}t + \beta^{-1/2}\,\mathrm{d}B(t) \tag{4.2}$$

for $t \ge 0$ where $B(t) \in \mathbb{R}^d$ is d-dimensional Brownian motion and the coefficient β^{-1} is the variance of a stochastic estimate of the gradient $\nabla f(x)$. (3.9) computes the gradient of Local Entropy by first computing the invariant measure, $\rho_\gamma(y;\, X)$, of the auxiliary stochastic differential equation (SDE)

$$\mathrm{d}Y(s) = -\frac{1}{\gamma}Y(s)\,\mathrm{d}s - \nabla f\left(X - Y(s)\right)\mathrm{d}s + \beta^{-1/2}\,\mathrm{d}B(s) \tag{4.3}$$

and updating $X(t)$ by averaging against this invariant measure

$$\frac{\mathrm{d}\,X(t)}{\mathrm{d}\,t} = -\int \nabla f\,(X-y)\,\rho_\gamma(\mathrm{d}y;\, X) \tag{4.4}$$

The effectiveness of the algorithm may be partially explained by the fact that for small value of γ, that solutions of Fokker-Planck equation associated with the SDE (4.3) converge exponentially quickly to the invariant measure ρ. Tools from optimal transportation (Santambrogio, 2015) or functional inequalities (Bakry and Émery, 1985) imply that there is a threshold parameter related to the semi-convexity of $f(x)$ for which the auxiliary SGD converges exponentially; we discuss this in Remark 4.6. Beyond this threshold, the auxiliary problem can be, in general, slow to converge.

In Section 4.3, we prove using homogenization of SDEs (Pavliotis and Stuart, 2008) that the dynamics in (4.4) recovers the gradient of Local Entropy. In other words, in the homogenization limit, (4.4) is equivalent to

$$\mathrm{d}\overline{X}(t) = -\nabla u\left(\overline{X}(t), \gamma\right)\,\mathrm{d}t \tag{4.5}$$

where $u\left(\overline{X}, \gamma\right)$ is the solution of (4.1). Thus even if smoothing the original loss function by computing the solution of the PDE directly is not practical due to the curse of dimensionality, the auxiliary SDE (4.3) accomplishes this naturally.

The Elastic-SGD algorithm (Zhang et al., 2015a) is an influential algorithm which trains multiple deep networks in parallel to obtain faster convergence in the distributed setting. We prove in Section 4.3.3 that the Elastic-SGD is equivalent to minimizing Local Entropy by the Entropy-SGD algorithm. The equivalence is a surprising result, since the former computes the average of a coupled realizations of SGD over multiple processors while the latter computes temporal averages to evaluate (4.4) over a single processor. This result follows naturally if the gradient dynamics is ergodic once the homogenization framework is established for Entropy-SGD.

The parameter γ can be modulated during the optimization: if γ is large, the smoothing achieved by the PDE is larger while a small γ recovers the true gradients of the original loss function $f(x)$ towards the end. Modulating

this parameter, which we call scoping is equivalent to replacing (4.5) by

$$\mathrm{d}\overline{X}(t) = -\nabla u\left(\overline{X}(t), T-t\right)\ \mathrm{d}t$$

where T is a termination time for the algorithm. Scoping, which was introduced as effective heuristic in Entropy-SGD in the previous chapter, is rigorously shown to be effective in Section 4.4. Additionally, we discuss its connections to nonlinear forward-backward equations, which appear in Mean Field Games; see Remark 4.10.

It is unusual in stochastic non-convex optimization to obtain a proof of the superiority of a particular algorithm compared to standard stochastic gradient descent. We obtain such a result in Theorem 4.11 of Section 4.4 where we prove an improvement (for slightly modified dynamics) in the expected value of the loss function using the Local Entropy dynamics as compared to that of SGD. This result is obtained using well-known techniques from stochastic optimal control theory (Fleming and Soner, 2006) and the comparison principle for viscosity solutions.

The effectiveness of the PDE regularization is illustrated Figure 4.1.

4.1. A MODEL FOR SGD

Our developments in this chapter hinge upon interpreting SGD as a continuous-time stochastic differential equation. To remind the reader, the discrete update equations of SGD are given by

$$x_{k+1} = x_k - \eta\ \nabla f_{\mathscr{b}}(x_k). \tag{4.6}$$

Along the lines of (Ghadimi and Lan, 2013), we assume that the stochastic gradient $\nabla f_{\mathscr{b}}(x_k)$ in (4.6) is unbiased with respect to the full-gradient and has bounded variance, i.e., for all $x \in \Omega$,

$$\mathbb{E}\left[\nabla f_{\mathscr{b}}(x)\right] = \nabla f(x),$$
$$\mathbb{E}\left[|\nabla f_{\mathscr{b}}(x) - \nabla f(x)|^2\right] \leq \beta_{\mathscr{b}}^{-1};$$

for some $\beta_{\mathscr{b}}^{-1} \geq 0$. We use the notation to emphasize that the noise is coming from the mini-batch gradients. The coefficient $\beta_{\mathscr{b}}^{-1}$ is zero if the mini-batch size is equal to the size of the dataset.

We omit the source of the noise and simply write β^{-1} while developing the theory; we will refer to the sub-script $\mathscr{b}$ again in Section 4.6.1 while providing algorithmic details. The discrete-time dynamics in (4.6) can be modeled by the stochastic differential equation (Li et al., 2017c)

$$\mathrm{d}X(t) = -\nabla f(X(t))\ \mathrm{d}t + \beta^{-1/2}\ \mathrm{d}B(t).$$

The generator L corresponding to (4.2) is defined for smooth functions φ as

$$L\varphi = -\nabla f\ \cdot \nabla\ \varphi + \frac{\beta^{-1}}{2}\Delta\varphi. \tag{4.7}$$

The adjoint operator L^*, also known as the Fokker-Planck operator, is given by

$$L^*\rho = \nabla\cdot(\nabla f\ \rho) + \frac{\beta^{-1}}{2}\Delta\rho.$$

Given a function $V(x) : \Omega \to \mathbb{R}$, define

$$u(x,t) = \mathbb{E}\left[V(X(T))\ \middle|\ X(t) = x\right] \tag{4.8}$$

to be the expected value at the final time for the path (4.2) with initial data $X(t) = x$. By taking expectations in the Itô formula, it can be established that u satisfies the backward Kolmogorov equation

$$\frac{\partial u}{\partial t} = Lu, \quad \text{for } t < s \leq T,$$

along with the terminal value $u(x,T) = V(x)$. Furthermore, $\rho(x,t)$, the probability density of $X(t)$ at time t, satisfies the Fokker-Planck equation (Risken, 1996)

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (\nabla f\ \rho)\ + \frac{\beta^{-1}}{2}\ \Delta \rho \tag{4.9}$$

along with the initial condition $\rho(x,0) = \rho_0(x)$ which represents the probability density of the initial distribution. With mild assumptions on $f(x)$, in particular even if f is non-convex, $\rho(x,t)$ converges to the unique stationary solution of (4.9) as $t \to \infty$ (Pavliotis, 2016, Section 4.5). Elementary calculations show that the stationary solution is the Gibbs distribution

$$\rho^{\text{eq}}(x;\ \beta) = \frac{1}{Z_\beta} e^{-\beta f(x)}, \tag{4.10}$$

for a normalizing constant $Z(\beta)$. The notation for the parameter $\beta > 0$ comes from physics, where it corresponds to the inverse temperature. From (4.10), as $\beta \to \infty$, the Gibbs distribution ρ^{eq} concentrates on the global minimizers of $f(x)$. Momentum is often used to accelerate convergence to this invariant distribution; this corresponds to running second-order Langevin dynamics (Pavliotis, 2016, Chapter 6) given by

$$\begin{aligned} \mathrm{d}q(t) &= p(t)\ dt \\ c^{-1}\mathrm{d}p(t) &= -\Big(\nabla f(q(t)) + p(t)\Big)\ \mathrm{d}t + \beta^{-1/2}\ \mathrm{d}B(t). \end{aligned}$$

where c is the friction coefficient. In the over-damped limit $c \to \infty$, we recover (4.2); see (Stoltz et al., 2010, Section 2.2.4) for a more elaborate discussion.

The celebrated paper by Jordan et al. (1998b) interpreted the Fokker-Planck equation as a gradient descent in the Wasserstein metric d_{W_2} (Santambrogio, 2015) of the energy functional

$$J(\rho) = \int\ V(x)\ \rho\ dx + \frac{\beta^{-1}}{2} \int\ \rho\ \log \rho\ dx;$$

See Section 6.2.1 for a more detailed exposition. If $V(x)$ is λ-convex function $f(x)$, i.e., $\nabla^2 V(x) + \lambda I$, is positive definite for all x, then the solution $\rho(x,t)$ converges exponentially in the Wasserstein metric with rate λ to ρ^∞ (Carrillo et al., 2006),

$$d_{W_2}\left(\rho(x,t),\ \rho^{\text{eq}}\right) \leq d_{W_2}\left(\rho(x,0),\ \rho^{\text{eq}}\right)\ e^{-\lambda t}.$$

Let us note that a less explicit, but more general, criterion for speed of convergence in the weighted L^2 norm is the Bakry-Emery criterion which uses PDE-based estimates of the spectral gap of the potential function (Bakry and Émery, 1985).

4.1.1. Metastability and simulated annealing

Under mild assumptions for non-convex functions $f(x)$, the Gibbs distribution ρ^∞ is still the unique steady solution of (4.9). However, convergence of $\rho(x,t)$ can take an exponentially long time. Such dynamics are said to exhibit metastability: there can be multiple quasi-stationary measures which are stable on time scales of order one. Kramers' formula (Kramers, 1940) for Brownian motion in a double-well potential is the simplest example of such a phenomenon: if $f(x)$ is a double-well with two local minima at locations $x_1, x_2 \in \mathbb{R}$ with a saddle point $x_3 \in \mathbb{R}$ connecting them, we have

$$\mathop{\mathbb{E}}_{x_1}[\tau_{x_2}]\ \propto \frac{1}{|f''(x_3)\ f''(x_1)|^{1/2}} \exp\Big(\beta(f(x_3) - f(x_1))\Big);$$

where τ_{x_2} is the time required to transition from x_1 to x_2 under the dynamics in (4.2). The one dimensional example can be generalized to the higher dimensional case (Bovier and den Hollander, 2006). Observe that there are two terms that contribute to the slow time scale: (i) the denominator involves the Hessian at a saddle point x_3, and (ii) the exponential dependence on the difference of the energies $f(x_1)$ and $f(x_2)$. In the higher dimensional case the first term can also go to infinity if the spectral gap goes to zero.

Simulated annealing (Kushner, 1987; Chiang et al., 1987) is a popular technique which aims to evade metastability and locate the global minimum of a general non-convex $f(x)$. The key idea is to control the variance of Brownian motion, in particular, decrease it exponentially slowly by setting $\beta = \beta(t) = \log(t+1)$ in (4.6). There are theoretical results that prove that simulated annealing converges to the global minimum (Geman and Hwang, 1986); however the time required is still exponential.

4.1.2. Smoothing of the loss function

A number of models of deep neural networks have been used in the literature to analyze characteristics of the energy landscape. If one forgoes the nonlinearities, a deep network can be seen as a generalization of principal component analysis (PCA), i.e., as a matrix factorization problem; this is explored by Baldi and Hornik (1989); Haeffele and Vidal (2015); Soudry and Carmon (2016); Saxe et al. (2013), among others. Based on empirical results such as Dauphin et al. (2014), authors in Bray and Dean (2007); Fyodorov and Williams (2007); Choromanska et al. (2015) and Chaudhari and Soatto (2015) have modeled the energy landscape of deep neural networks as a high-dimensional Gaussian random field. In spite of these analyses being drawn from vastly diverse areas of machine learning, they suggest that, in general, the energy landscape of deep networks is highly non-convex and rugged.

Smoothing is an effective way to improve the performance of optimization algorithms on such a landscape and it is our primary focus in this chapter. This can be done by convolution with a kernel (Chen, 2012); for specific architectures this can be done analytically (Mobahi, 2016) or by averaging the gradient over random, local perturbations of the parameters (Gulcehre et al., 2016a). A recent paper by Li et al. (2017b) suggests using extensive empirical analysis that the rugged energy landscape of deep architectures can be made smoother using residual architectures, e.g., by forcing the model to be an identity mapping (He et al., 2016) if all the weights are zero. Our results provide a means to smoothen the energy landscape using new optimization algorithms for the same neural architecture. This chapter compares and contrasts different optimization-based smoothing techniques in a unified mathematical framework.

4.2. PDE INTERPRETATION OF LOCAL ENTROPY

Let us remind the reader of Local Entropy from Chapter 3. It is a modified loss function first introduced as a means for studying the energy landscape of the discrete perceptron, i.e., a "shallow" neural network with one layer and discrete parameters, e.g., $x \in \{-1,1\}^d$ (Baldassi et al., 2016c). An analysis of Local Entropy using the replica method predicts dense clusters of solutions to perceptron problem. Moreover, parameters that lie within such clusters yield better error on samples outside the training set, i.e. they have improved generalization error (Baldassi et al., 2015). We rewrite the Local Entropy function as:

$$f_\gamma(x) = -\frac{1}{\beta}\log\Big(G_{\beta^{-1}\gamma} * \exp(-\beta f(x))\Big); \tag{4.11}$$

where $G_\gamma(x) = (2\pi\gamma)^{-d/2}\exp\left(-\frac{|x|^2}{2\gamma}\right)$ is the heat kernel.

4.2.1. Derivation of the viscous Hamilton-Jacobi PDE

The Cole-Hopf transformation (Evans, 1998, Section 4.4.1) is a classical tool used in PDEs. It relates solutions of the heat equation to those of the viscous Hamilton-Jacobi equation. For the convenience of the reader, we restate it here.

Lemma 4.1 (Cole-Hopf transformation). Local Entropy defined by (4.11) is the solution of the initial value problem for the viscous Hamilton-Jacobi equation (4.1)

$$\frac{\partial u}{\partial t} = -\frac{1}{2}|\nabla u|^2 + \frac{\beta^{-1}}{2}\Delta u, \qquad \text{for } 0 < t \le \gamma$$

with initial values $u(x,0) = f(x)$. Moreover, the gradient is given by

$$\nabla u(x,t) = \int_\Omega \frac{x-y}{t} \rho_1^\infty(\mathrm{d}y;\, x) \tag{4.12}$$

$$= \int_\Omega \nabla f(x-y)\, \rho_2^\infty(\mathrm{d}y;\, x) \tag{4.13}$$

where the two probability distributions are given by

$$\begin{aligned} \rho_1^\infty(y;\, x) &= Z_1^{-1} \exp\left(-\beta f(y) - \frac{\beta}{2t}|x-y|^2 \right) \\ \rho_2^\infty(y;x) &= Z_2^{-1} \exp\left(-\beta f(x-y) - \frac{\beta}{2t}|y|^2 \right) \end{aligned} \tag{4.14}$$

and $Z_i = Z_i(x)$ are normalizing constants for $i = 1, 2$.

Proof. Define $u(x,t) = -\beta^{-1} \log v(x,t)$. From (4.11), we can check that $v = \exp(-\beta u)$ solves the heat equation

$$v_t = \frac{1}{2}\beta^{-1}\Delta v$$

with initial data $v(x,0) = \exp(-\beta f(x))$. Taking partial derivatives gives

$$\begin{aligned} v_t &= -\beta\ v\ u_t, \\ \nabla v &= -\beta\ v\ \nabla u, \\ \Delta v &= -\beta\ v\ \Delta u + \beta^2\ v\ |\nabla u|^2. \end{aligned}$$

Combining these expressions results in (4.1). Differentiating $v(x,t) = \exp(-\beta u(x,t))$ using the convolution in (4.11), gives

$$\begin{aligned} \nabla u(x,t) = \nabla_x \left(G_{\beta^{-1}t} \ast e^{-\beta f(x)} \right) &= \nabla_x \int G_{\beta^{-1}t}(x-y)\, e^{-\beta f(y)}\, \mathrm{d}y \\ &= \nabla_x \int G_{\beta^{-1}t}(y)\, e^{-\beta f(x-y)}\, \mathrm{d}y. \end{aligned}$$

up to a constant factor. Evaluating the two expressions leads to (4.13) and (4.14) respectively. ■

4.2.2. Hopf-Lax formula for the Hamilton-Jacobi equation and dynamics for the gradient

In addition to the connection with (4.1) provided by Lemma 4.1, we can also explore the non-viscous Hamilton-Jacobi equation which corresponds to the limiting case of (4.1) as the viscosity term $\beta^{-1} \to 0$:

$$u_t = -\frac{1}{2}\ |\nabla u|^2. \tag{4.15}$$

There are several reasons for studying this equation. It has a simple, explicit formula for the gradient. In addition, semi-concavity of the solution follows directly from the Hopf-Lax formula. Moreover, under certain convexity conditions, the gradient of the solution can be computed as an exponentially convergent gradient flow. The deterministic dynamics for the gradient are a special case of the stochastic dynamics for the viscous-HJ equation which are discussed in another section.

In the following lemma, we apply the well-known Hopf-Lax formula (Evans, 1998) for the solution of the HJ equation. It is also called the inf-convolution of the functions $f(x)$ and $\frac{1}{2t}|x|^2$ (Cannarsa and Sinestrari, 2004) and is closely related to the proximal operator (Moreau, 1965; Rockafellar, 1976).

Lemma 4.2. Let $u(x,t)$ be the viscosity solution of (4.15) with $u(x,0) = f(x)$. Then

$$u(x,t) = \inf_y \left\{ f(y) + \frac{1}{2t}\ |x-y|^2 \right\}. \tag{4.16}$$

Define the proximal operator

$$\text{prox}_{tf}(x) = \arg\min_y \left\{ f(y) + \frac{1}{2t}|x-y|^2 \right\} \tag{4.17}$$

The gradient $\nabla_x u(x,t)$ exists if $y^* = \text{prox}_{tf}(x)$ is a singleton and

$$\nabla_x u(x,t) = \frac{x-y^*}{t} = \nabla f(y^*). \tag{4.18}$$

Proof. The Hopf-Lax formula for the solution of the HJ equation can be found in (Evans, 1998). Danskin's theorem (Bertsekas et al., 2003, Prop. 4.5.1) allows us to pass a gradient through an infimum, by applying it at the argminimum. The first equality is a direct application of this to (4.16). For the second equality, rewrite (4.16) with $z = (x-y)/t$ and the drop the prime, to obtain,

$$u(x,t) = \inf_z \left\{ f(x-tz) + \frac{t}{2}\,|z|^2 \right\}.$$

From this, we again have

$$\nabla u(x,t) = \nabla f(x-tz^*) = \nabla f(y^*).$$

■

We next give a lemma which shows that under an auxiliary convexity condition, we can find $\nabla u(x,t)$ using exponentially convergent gradient dynamics.

Lemma 4.3 (Dynamics for HJ). For a fixed x, define

$$h(x,y;\,t) = f(x-ty) + \frac{t}{2}\,|y|^2.$$

Suppose that $t > 0$ is chosen so that $h(x,y;t)$ is λ-convex as a function of y (meaning that $\nabla_y^2\, h(x,y,t) - \lambda I$ is positive definite). The gradient $p = \nabla_x\, u(x,t)$ is then the unique steady solution of the dynamics

$$y'(s) = -\nabla_y\, h(x,y(s);\,t) = -t\Big(y(s) - \nabla f(x-t\,y(s))\Big). \tag{4.19}$$

Moreover, the convergence to p is exponential, i.e.,

$$|y(s)-p| \le |y(0)-p|\; e^{-\lambda s}.$$

Proof. Note that (4.19) is the gradient descent dynamics on h which is λ-convex by assumption. Thus by standard techniques from ordinary differential equation (ODE) (see, for example (Santambrogio, 2015)) the dynamics is a contraction, and the convergence is exponential. ■

4.2.3. The invariant measure for a quadratic function

Local entropy computes a non-linear average of the gradient in the neighborhood of x by weighing the gradients according to the steady-state measure $\rho^\infty(y;\,x)$. It is useful to compute $\nabla f_\gamma(x)$ when $f(x)$ is quadratic, since this gives an estimate of the quasi-stable invariant measure near a local minimum of f. In this case, the invariant measure $\rho^\infty(y;\,x)$ is a Gaussian distribution.

Lemma 4.4. Suppose $f(x)$ is quadratic, with $p = \nabla\, f(x)$ and $Q = \nabla^2 f(x)$, then the invariant measure $\rho_2^\infty(y;\,x)$ of (4.14) is a normal distribution with mean μ and covariance Σ given by

$$\mu = x - \Sigma\, p \quad \text{and} \quad \Sigma = (Q + \gamma\, I)^{-1}.$$

In particular, for $\gamma > |Q|_2$, we have

$$\mu \approx x - \left(\gamma^{-1}\, I - \gamma^{-2} Q\right) p \quad \text{and} \quad \Sigma \approx \gamma^{-1}\, I - \gamma^{-2} Q,$$

plus higher order terms in γ.

Proof. Without loss of generality, a quadratic $f(x)$ centered at $x = 0$ can be written as

$$f(x) = f(0) + p\,x + \frac{1}{2}\,x^\top Q x,$$
$$\Rightarrow f(x) + \frac{\gamma}{2}\,x^2 = f(0) - \mu^\top p + \frac{1}{2}\,(x-\mu)^\top (Q+\gamma\,I)\,(x-\mu)$$

by completing the square. The distribution $\exp\left(-f(y) - \frac{1}{2\gamma}\,|y|^2\right)$ is thus a normal distribution with mean $\mu = x - \Sigma\,p$ and variance $\Sigma = (Q+\gamma\,I)^{-1}$ and an appropriate normalization constant. The approximation $\Sigma = \gamma^{-1}\,I - \gamma^{-2}Q$ which avoids inverting the Hessian follows from the Neumann series for the matrix inverse; it converges for $\gamma > |Q|_2$ and is accurate up to $\mathscr{O}\left(\gamma^{-3}\right)$. ■

4.3. Derivation of Local Entropy via homogenization of SDEs

The Hopf-Cole formula and the corresponding formula for the gradient in Lemma 4.1 involves high-dimensional integrals cannot be evaluated efficiently. In this section we obtain stochastic dynamics which through averaging (homogenization), give a computationally effective method to compute the gradient of the solution of the HJB PDE.

In addition, the homogenization interpretation allows us to rigorously show that an algorithm called Elastic-SGD (Zhang et al., 2015a) that was heuristically connected to a distributed version of Local Entropy in Baldassi et al. (2016a) is indeed equivalent to Local Entropy.

4.3.1. Background on homogenization

Homogenization is a technique used to analyze dynamical systems with multiple time-scales that have a few fast variables that may be coupled with other variables which evolve slowly. Computing averages over the fast variables allows us to obtain averaged equations for the slow variables in the limit that the times scales separate. We refer the reader to Pavliotis and Stuart (2008, Chap. 10, 17) or E (2011) for details. The convergence is strong: for rigorous convergence statements see Pavliotis and Stuart (2008, Theorem 17.1).

Consider the system of SDEs given by

$$\begin{aligned} \mathrm{d}X(s) &= h(X,\,Y)\,\mathrm{d}s \\ \mathrm{d}Y(s) &= \frac{1}{\varepsilon}\,g(X,\,Y)\,\mathrm{d}s + \frac{1}{\sqrt{\varepsilon\beta}}\,\mathrm{d}B(s); \end{aligned} \tag{4.20}$$

where h, g are sufficiently smooth functions, $B(s) \in \Omega$ is standard Brownian motion and $\varepsilon > 0$ is the homogenization parameter which introduces a fast time-scale for the dynamics of $Y(s)$. Let us define the generator for the second equation to be

$$L_0 = g(x,y)\,\cdot\,\nabla_y + \frac{\beta^{-1}}{2}\,\Delta_y.$$

We can assume that for fixed x the fast-dynamics $Y(s)$ has a unique invariant probability measure denoted by $\rho^\infty(y;\,x)$ and the operator L_0 has a one-dimensional null-space characterized by

$$L_0\,1(y) = 0, \quad L_0^*\,\rho^\infty(y;\,x) = 0;$$

here $1(y)$ are all constant functions in y and L_0^* is the adjoint of $\mathscr{L}_0$. It now follows that in the limit $\varepsilon \to 0$, the dynamics for $X(s)$ in (4.20) converges to

$$\mathrm{d}\overline{X}(s) = \overline{h}(\overline{X})\,\mathrm{d}s$$

where the homogenized vector field $\overline{X}$ is defined as the average against the invariant measure. Moreover, by ergodicity, in the limit, the spatial average is equal to a long term average over the dynamics independent of the initial value $Y(0)$:

$$\overline{h}(X) = \int h(X,y)\,\rho^\infty(dy;\,X) = \lim_{T\to\infty}\,\frac{1}{T}\int_0^T h(X,y(s))\,\mathrm{d}s$$

for all $X \in \Omega$.

4.3.2. Derivation of Local Entropy via homogenization of SDEs

Recall (4.13) of Lemma 4.1

$$\nabla f_\gamma(x) = -\gamma^{-1} \int_\Omega (x-y)\, \rho_1^\infty(y;\, x)\, \mathrm{d}y.$$

Hence, let us consider the following system of SDEs

$$\begin{aligned} \mathrm{d}X(s) &= -\gamma^{-1}\,(X-Y)\,\mathrm{d}s \\ \mathrm{d}Y(s) &= -\frac{1}{\varepsilon}\left[\nabla f(Y) + \frac{1}{\gamma}\,(Y-X)\right]\mathrm{d}s + \frac{\beta^{-1/2}}{\sqrt{\varepsilon}}\,\mathrm{d}B(s). \end{aligned} \tag{4.21}$$

Write

$$H(x,y;\,\gamma) = f(y) + \frac{1}{2\gamma}|x-y|^2.$$

The Fokker-Planck equation for the density of $y(s)$ is given by

$$\rho_t = L_0^*\,\rho = \nabla_y \cdot (\nabla_y\, H\, \rho) + \frac{\beta^{-1}}{2}\,\Delta_y\,\rho; \tag{4.22}$$

The invariant measure for this Fokker-Planck equation is thus

$$\rho_1^\infty(y;\,x) = \frac{1}{Z}\exp\left(-\beta H(x,y;\,\gamma)\right)$$

which agrees with (4.14) in Lemma 4.1.

Theorem 4.5. As $\varepsilon \to 0$, the system (4.21) converges to the homogenized dynamics given by

$$\mathrm{d}\overline{X}(s) = -\nabla f_\gamma(\overline{X})\, ds.$$

Moreover $-\nabla f_\gamma(x) = -\gamma^{-1}\ \langle x-y\rangle$ where

$$\langle x-y\rangle = \int (x-y)\,\rho_1^\infty(\mathrm{d}y;\,X) = \lim_{T\to\infty}\frac{1}{T}\int_0^T\ y(s)\,\mathrm{d}s \tag{4.23}$$

where $y(s)$ is the solution of the second equation in (4.21) for fixed x.

Proof. The result follows immediately from the convergence result in Section 4.3.1 for the system (4.20). The homogenized vector field is

$$\overline{h}(\overline{X}) = -\gamma^{-1}\int (\overline{X}-y)\,\rho_1^\infty(\mathrm{d}y;\,\overline{X})$$

which is equivalent to $\nabla f_\gamma(X)$ from Lemma 4.1. ■

By Lemma 4.1, the following dynamics also converge to the gradient descent dynamics for f_γ.

$$\begin{aligned} \mathrm{d}X(s) &= -\nabla f(X-Y)\,\mathrm{d}s \\ \mathrm{d}Y(s) &= -\frac{1}{\varepsilon}\left[\frac{Y}{\gamma} - \nabla f(X-Y)\right]\mathrm{d}s + \frac{\beta^{-1/2}}{\sqrt{\varepsilon}}\,\mathrm{d}B(s). \end{aligned}$$

Remark 4.6 (Exponentially fast convergence). Theorem 4.5 relies upon the existence of an ergodic, invariant measure $\rho_1^\infty(y;x)$. Convergence to such a measure is exponentially fast if the underlying function, $H(x,y;\,\gamma)$, is convex in y. In our case this happens if $\nabla^2 f(x) + \gamma^{-1} I$ is positive definite for all x.

4.3.3. Elastic-SGD as Local Entropy

Elastic-SGD is an algorithm introduced by Zhang et al. (2015a) for distributed training of deep neural networks and aims to minimize the communication overhead between a set of workers that together optimize replicated

copies of the original function $f(x)$. For $n > 0$ distinct workers, Let $y = (y_1, \ldots, y_n)$, and let

$$\overline{Y} = \frac{1}{n} \sum_{i=1}^{n} Y^i$$

be the average value. Define the modified loss function

$$\min_{x^1,\ldots,x^n} \frac{1}{n} \sum_{i=1}^{n} \left(f(x^i) + \frac{1}{2\gamma} |x^i - \overline{x}|^2 \right). \tag{4.24}$$

The formulation in (4.24) lends itself easily to a distributed optimization where worker i performs an update which depends only on the average of the other workers

$$\mathrm{d}Y^i(s) = -\nabla f(Y^i(s))\, \mathrm{d}s - \frac{1}{\gamma} \left(Y^i(s) - \overline{Y} \right) \mathrm{d}s + \beta^{-1/2} \mathrm{d}B^i(s) \tag{4.25}$$

This algorithm was shown to be connected to the Local Entropy loss $f_\gamma(x)$ by Baldassi et al. (2016a) using arguments from replica theory. The results from the previous section can be modified to prove that the dynamics above converges to gradient descent of the Local Entropy. Consider the system

$$\mathrm{d}X(s) = \frac{1}{\gamma} \left(X - \overline{Y} \right) \mathrm{d}s$$

along with (4.25). As in Section 4.3.2, define

$$H(X, Y;\, \gamma) = \sum_{i=1}^{n} f(Y^i) + \frac{1}{2\gamma} |X - \overline{Y}|^2.$$

The invariant measure for the $Y(s)$ dynamics, and consequently $\overline{Y}$, corresponding to the ε scaling of (4.25) is given by

$$\rho^\infty(y;\, x) = \frac{1}{Z} \exp\left(-\beta\, H(x, y;\, \gamma) \right),$$

and the homogenized vector field is

$$\overline{h}(\overline{X}) = \gamma^{-1} \left\langle \overline{X} - \overline{Y} \right\rangle$$

where the angular brackets compute the expectation over $\rho^\infty(y;\, x)$ above. By ergodicity, we can replace the average over the invariant measure with a combination of the temporal average and the average over the workers

$$\left\langle \overline{Y} \right\rangle = \lim_{T \to \infty} \frac{1}{T} \int_0^T \overline{Y}(s)\, \mathrm{d}s \tag{4.26}$$

Thus the same argument as in Theorem 4.5 applies, and we can conclude that the dynamics also converges to gradient descent for Local Entropy as $n \to \infty$.

Remark 4.7 (Variational interpretation of the distributed algorithm). An alternate approach to understanding the distributed version of the algorithm, which we hope to study in the future, is modeling it as the gradient flow in the Wasserstein metric of a non-local functional. This functional is obtained by including an additional term in the Fokker-Planck functional,

$$J(\rho) = \int f_\gamma(x)\, \rho \mathrm{d}x + \frac{\beta^{-1}}{2} \int \log \rho\, \mathrm{d}\rho + \frac{1}{2\gamma} \int \int |x - y|^2\, \rho(x) \rho(y)\, \mathrm{d}x\, \mathrm{d}y;$$

see Santambrogio (2015).

4.3.4. Heat equation versus the viscous Hamilton-Jacobi equation

Lemma 4.1 showed that Local Entropy is the solution to the viscous Hamilton-Jacobi equation (4.1). The homogenization dynamics in (4.21) can thus be interpreted as a way of smoothing of the original loss $f(x)$ to aid optimization. Other partial differential equations could also be used to achieve a similar effect. The

following dynamics that corresponds to gradient descent on the solution of the the heat equation:

$$\begin{aligned} \mathrm{d}X(s) &= -\nabla f(X-Y)\,\mathrm{d}s \\ \mathrm{d}Y(s) &= -\frac{1}{\varepsilon\gamma}\,y\,\mathrm{d}s + \frac{1}{\sqrt{\varepsilon\beta}}\,\mathrm{d}B(s). \end{aligned} \tag{4.27}$$

Using a very similar analysis as above, the invariant distribution for the fast variables is $\rho^{\infty}(y;\,X) = G_{\beta^{-1}\gamma}(y)$. In other words, the fast-dynamics for Y is decoupled from X and $Y(s)$ converges to a Gaussian distribution with mean zero and variance $\beta^{-1}\gamma I$. The homogenized dynamics is given by

$$\mathrm{d}X(s) = -\nabla f(X) * G_{\beta^{-1}\gamma}\,\mathrm{d}s = -\nabla v(X,\gamma) \tag{4.28}$$

where $v(x,\gamma)$ is the solution of the heat equation $v_t = \frac{\beta^{-1}}{2}\Delta v$ with initial data $v(x,0) = f(x)$. The dynamics corresponds to a Gaussian averaging of the gradients.

Remark 4.8 (Comparison between Local Entropy and heat equation dynamics). The extra term in the $Y(s)$ dynamics for Section 4.3.2 with respect to (4.27) is exactly the distinction between the smoothing performed by Local Entropy and the smoothing performed by heat equation. The latter is common in the deep learning literature under different forms, e.g., Gulcehre et al. (2016a) and Mobahi (2016). The former version however has much better empirical performance (cf.experiments in Section 4.6 and Chaudhari et al. (2016)) as well as improved convergence in theory (cf. Theorem 4.11). Moreover the gradient term at a critical point vanishes whereby (4.27) and Section 4.3.2 coincide.

4.4. Stochastic control interpretation

In this section we consider the following controlled SDE

$$\mathrm{d}X(s) = -\nabla f(X(s))\,\mathrm{d}s - \alpha(s)\,\mathrm{d}s + \beta^{-1/2}\,\mathrm{d}B(s), \quad \text{for } t \le s \le T, \tag{4.29}$$

along with $X(t) = x$. Controlled stochastic differential equations (Fleming and Soner, 2006; Fleming and Rishel, 2012) are generalizations of SDEs discussed earlier in Section 4.1. The use of stochastic control in this context is unusual, we are not aware of any works other than Li et al. (2017c).

4.4.1. Stochastic control for a variant of Local Entropy

For fixed controls α, the generator for (4.29) is given by

$$L_\alpha := (-\nabla f - \alpha)\cdot\nabla + \frac{\beta^{-1}}{2}\,\Delta$$

Define the cost functional for a given solution $x(\cdot)$ of (4.29) corresponding to the control $\alpha(\cdot)$,

$$\mathscr{C}(x,\,\alpha(\cdot)) = \mathbb{E}\left[V(X(T)) + \frac{1}{2}\int_t^T |\alpha(s)|^2\,\mathrm{d}s \mid X(t) = x\right]. \tag{4.30}$$

Here the terminal cost is a given function $V : \Omega \to \mathbb{R}$ and we use the prototypical quadratic running cost. Define the value function

$$u(x,t) = \min_{\alpha(\cdot)} \mathscr{C}(x,\,\alpha(\cdot)).$$

to be the minimum expected cost, over admissible controls, and over paths which start at $x(t) = x$. From the definition, the value function satisfies

$$u(x,T) = V(x).$$

The dynamic programming principle expresses the value function as the unique viscosity solution of a Hamilton-Jacobi-Bellman (HJB) PDE (Fleming and Soner, 2006; Fleming and Rishel, 2012), $u_t = H(\nabla u,\,\nabla^2 u)$ where

the operator H is defined by

$$\begin{aligned} H(p,Q) &= \min_{\alpha} \left\{ L_\alpha(p,Q) + \frac{1}{2}|\alpha|^2 \right\} \\ &:= \min_{\alpha} \left\{ (-\nabla f - \alpha)\cdot p + \frac{1}{2}\,|\alpha|^2 + \frac{\beta^{-1}}{2}\,\mathrm{Tr}\,Q \right\} \end{aligned}$$

The minimum is achieved at $\alpha^* = p$, which results in

$$H(p,Q) = -\nabla f \cdot p - \frac{1}{2}|p|^2 + \frac{\beta^{-1}}{2}\,\mathrm{Tr}\,Q.$$

The resulting PDE for the value function is

$$-u_t(x,t) = -\nabla f(x)\cdot\nabla u(x,t) - \frac{1}{2}\,|\nabla u(x,t)|^2 + \frac{\beta^{-1}}{2}\,\Delta u(x,t), \qquad \text{for } t \le s \le T, \tag{4.31}$$

along with the terminal values $u(x,T) = V(x)$. We make note that in this case, the optimal control is equal to the gradient of the solution

$$\alpha^*(x,t) = \nabla u(x,t). \tag{4.32}$$

Remark 4.9. We could also obtain a stochastic control interpretation for (4.1) by dropping the $\nabla f(x(s))$ term in (4.29). Then, by a similar argument, (4.31) results in (4.1). Thus we obtain the interpretation of solutions of (4.1) as the value function of an optimal control problem, minimizing the cost function (4.30) but with the modified dynamics. The reason for including the ∇f term in (4.29) is that it allows us to prove the comparison principle in the next subsection.

Remark 4.10 (Forward-backward equations and mean-field games). Note that the PDE (4.31) is a terminal value problem in backwards time. This equation is well-posed, and by relabeling time, we can obtain an initial value problem in forward time. The corresponding forward equation for the evolution is of a density under the dynamics (4.29), involving the gradient of the value function. Together, these PDEs are the forward-backward system

$$\begin{aligned} -u_t &= -\nabla f\cdot\nabla u - \frac{1}{2}\,|\nabla u|^2 + \frac{\beta^{-1}}{2}\,\Delta u \\ \rho_t &= -\nabla\cdot\left(\nabla u\,\rho\right) + \Delta\rho, \end{aligned} \tag{4.33}$$

for $0 \le s \le T$ along with the terminal and the initial data $u(x,T) = V(x)$ and $\rho(x,0) = \rho_0(x)$.

More generally, we can consider stochastic control problems where the control and the cost depend on the density of other players. In the case of "small" players, this can lead to mean field games (Lasry and Lions, 2007; Huang et al., 2006). In the distributed setting is it natural to consider couplings (control and costs) which depend on the density (or moments of the density). In this context, it may be possible to use mean field game theory to prove an analogue of Theorem 4.11 and obtain an estimate of the improvement in the terminal reward.

4.4.2. Improvement in the value function

The optimal control interpretation above allows us to provide the following theorem for the improvement in the value function obtained by the dynamics (4.29) using the optimal control (4.32), compared to stochastic gradient descent.

Theorem 4.11. Let $X_{\mathrm{csgd}}(s)$ and $X_{\mathrm{sgd}}(s)$ be solutions of (4.29) and (4.2), respectively, with the same initial data $X_{\mathrm{csgd}}(0) = X_{\mathrm{sgd}}(0) = x_0$. Fix a time $t \ge 0$ and a terminal function, $V(x)$. Then

$$\mathbb{E}\left[V(X_{\mathrm{csgd}}(t))\right] \le \mathbb{E}\left[V(X_{\mathrm{sgd}}(t))\right] - \frac{1}{2}\,\mathbb{E}\left[\int_0^t |\alpha(X_{\mathrm{csgd}}(s),s)|^2\,\mathrm{d}s\right].$$

The expectations above are with respect to Brownian motion and conditioned on the events $X_{\text{csgd}}(0) = x_0$ and $X_{\text{sgd}}(0) = x_0$ as appropriate. The optimal control is given by

$$\alpha^*(x,t) = \nabla u(x,t),$$

where $u(x,t)$ is the solution of (4.31) along with terminal data $u(x,T) = V(x)$.

Proof. First consider (4.2) with the generator $L = -\nabla f(x) \cdot \nabla + \frac{\beta^{-1}}{2} \Delta$ given in (4.7). Note the PDEs are backward parabolic with terminal values. To be consistent with the rest of the development, we simply reverse time, which does not change the substance of the proof. The expected value function $v(x,t) = \mathbb{E}[V(X(t))]$ is the solution of the PDE $v_t = Lv$. Next, let $u(x,t)$ be the solution of (4.31). The PDE (4.31) says

$$u_t - Lu - \frac{1}{2}|\nabla u|^2 = 0.$$

In particular,

$$u_t - Lu \geq 0.$$

We also have $u(x,T) = v(x,T) = V(x)$. By the maximum principle (Evans, 1998), (with some additional technical assumptions such as $|x| \to \infty$), we can conclude

$$u(x,s) \leq v(x,s), \quad \text{for all } t \leq s \leq T; \tag{4.34}$$

Use the definition to get

$$v(x,t) = \mathbb{E}[V(X(t)) \mid X(0) = x_0];$$

the expectation is taken over the paths of (4.2). Similarly,

$$u(x,t) = \mathbb{E}\left[V(X(t)) + \frac{1}{2} \int_0^t |\alpha^*|^2 \, \mathrm{d}s \mid X(0) = x_0\right]$$

over paths of (4.29). Here $\alpha^*(\cdot)$ is the optimal control. The interpretation along with (4.34) gives the desired result. Note that the second term on the right hand side in Theorem 4.11 above can also be written as

$$\frac{1}{2} \mathbb{E}\left[\int_0^t |\nabla u(X_{\text{csgd}}(s), s)|^2 \, ds\right].$$

■

4.5. Regularization, widening and semi-concavity

The PDE approaches discussed in Section 4.2 result in a smoother loss function. We exploit the fact that our regularized loss function is the solution of the viscous HJ equation. This PDE allows us to apply standard semi-concavity estimates from PDE theory (Evans, 1998; Cannarsa and Sinestrari, 2004) and quantify the amount of smoothing. Note that these estimates do not depend on the coefficient of viscosity so they apply to the HJ equation as well. Indeed, semi-concavity estimates apply to solutions of PDEs in a more general setting which includes (4.1) and (4.15).

We begin with special examples which illustrate the widening of local minima. In particular, we study the limiting case of the HJ equation, which has the property that local minima are preserved for short times. We then prove semi-concavity and establish a more sensitive form of semi-concavity using the harmonic mean. Finally, we explore the relationship between wider robust minima and the Local Entropy.

4.5.1. Widening for solutions of the Hamilton-Jacobi PDE

A function $f(x)$ is semi-concave with a constant C if $f(x) - \frac{C}{2}|x|^2$ is concave, refer to Cannarsa and Sinestrari (2004). Semi-concavity is a way of measuring the width of a local minimum: when a function is semi-concave with constant C, at a local minimum, no eigenvalue of the Hessian can be larger than C. The semi-concavity estimates below establish that when we evolve f by (4.1) high curvature local minima widen faster than the

ones with low curvature. The development in this section is a generalization of the Laplace approximation used in the proof of Lemma 3.5.

It is illustrating to consider the following example.

Example 4.12. Let $u(x,t)$ be the solution of (4.1) with $u(x,0) = c|x|^2/2$. Then

$$u(x,t) = \frac{|x|^2}{2(t+c^{-1})} + \frac{\beta^{-1}n}{2}\log(t+c^{-1}).$$

In particular, in the non-viscous case, $u(x,t) = \frac{|x|^2}{2(t+c^{-1})}$.

Proof. This follows from taking derivatives:

$$\begin{aligned}
\nabla u(x,t) &= \frac{x}{t+c^{-1}},\\
\frac{|\nabla u|^2}{2} &= \frac{|x|^2}{2(t+c^{-1})^2},\\
\Delta u(x,t) &= \frac{n}{t+1} \text{ and,}\\
u_t &= -\frac{|x|^2}{2(t+c^{-1})^2} + \frac{\beta^{-1}d}{2}\frac{1}{t+c^{-1}}.
\end{aligned}$$

■

In the previous example, the semi-concavity constant of $u(x,t)$ is $C(t) = 1/(c^{-1}+t)$ where c measures the curvature of the initial data. This shows that for very large c, the improvement is very fast, for example $c = 10^8$ leads to $C(t = .01) \approx 10$, etc. In the sequel, we show how this result applies in general for both the viscous and the non-viscous cases. In this respect, for short times, solutions of (4.1) and (4.15) widen faster than solutions of the heat equation.

Example 4.13 (Slower rate for the heat equation). In this example we show that the heat equation can result in a very slow improvement of the semi-concavity. Let $v_1(x)$ be the first eigenfunction of the heat equation, with corresponding eigenvalue λ_1, (for example $v_1(x) = \sin(x)$). In many cases λ_1 is of order 1. Then, since $v(x,t) = \exp(-\lambda_1 t)\ v_1(x)$ is a solution of the heat equation, the semi-concavity constant for $v(x,t)$ is $C(t) = c_0\exp(-\lambda_1 t)$, where c_0 is the constant for $v_1(x)$. For short time, $C(t) \approx c_0(1-\lambda_1 t)$ which corresponds to a much slower decrease in $C(t)$.

Another illustrative example is to study the widening of convex regions for the non-viscous Hamilton-Jacobi equation (4.15). As can be seen in Figure 4.1, solutions of the Hamilton-Jacobi have additional remarkable properties with respect to local minima. In particular, for small values of t (which depend on the derivatives of the initial function) local minima are preserved. To motivate these properties, consider the following examples. We begin with the Burgers equation.

Example 4.14 (Burgers equation in one dimension). There is a well-known connection between Hamilton-Jacobi equations in one dimension and the Burgers equation (Evans, 1998) which is the prototypical one-dimensional conservation law. Consider the solution $u(x,t)$ of the non-viscous HJ equation (4.15) with initial value $f(x)$ where $x \in \mathbb{R}$. Let $p(x,t) = \nabla_x\, u(x,t)$. Then p satisfies the Burgers equation

$$p_t = -p\ p_x$$

We can solve the Burgers equation using the method of characteristics for a short time depending on the regularity of $f(x)$. Set $p(x,t) = u_x(x,t)$. We use the fact that $p(x,t)$ is the fixed point of (4.19) which leads to

$$p(x,t) = f'(x - tp(x,t)) \tag{4.35}$$

In particular, the solutions of this equation remain classical until the characteristics intersect, and this time t^* is determined by

$$t^* f''(x - pt^*) + 1 = 0 \tag{4.36}$$

which recovers the convexity condition of Lemma 4.3.

Example 4.15 (Widening of convex regions in one dimension). In this example, we show that the widening of convex regions for the one dimensional example in Figure 4.1 occurs in general. Let x be a local minimum of the smooth function $f(x)$ and let $0 \le t < t^*$ where the critical t^* is defined by (4.36). Define the convexity interval as:

$$I(x,t) = \text{ the widest interval containing } x \text{ where } u(x,t) \text{ is convex}$$

Let $I(x,0) = [x_0,\, x_1]$, so that $f''(x) \ge 0$ on $[x_0,\, x_1]$ and $f''(x_0) = f''(x_1) = 0$. Then $I(x,t)$ contains $I(x,0)$ for $0 \le t \le t^*$. Differentiating (4.35), and the solving for $p_x(x,t)$ leads to

$$\begin{aligned} p_x &= f''(x - tp)\,(1 - tp_x) \\ \Rightarrow p_x &= \frac{f''(x-tp)}{1 + t\, f''(x-tp)}. \end{aligned}$$

Since $t \le t^*$, the last equation implies that p_x has the same sign as $f''(x-tp)$. Recall now that x_1 is an inflection point of $f(x)$. Choose x such that $x_1 = x - tp(x,t)$, in other words, $u_{xx}(x,t) = 0$. Note that $x > x_1$, since $p(x,t) > 0$. Thus the inflection point moves to the right of x_1. Similarly the inflection point x_0 moves to the left. We have thus established that the interval is larger.

Remark 4.16. In higher dimensions, well-known properties of the inf-convolution can be used to establish the following facts. If $f(x)$ is convex, $u(x,t)$ given by (4.16) is also convex; thus, any minimizer of f also minimizes $u(x,t)$. Moreover, for bounded $f(x)$, using the fact that $|y^*(x) - x| = \mathcal{O}(\sqrt{t})$, one can obtain a local version of the same result, viz., for short times t, a local minimum persists. Local minima in the solution vanish for longer times like we saw in the example in Figure 4.1.

4.5.2. Semi-concavity estimates

In this section we establish semi-concavity estimates. These are well known, and are included for the convenience of the reader.

Lemma 4.17. Suppose $u(x,t)$ is the solution of (4.1) and let $\beta^{-1} \ge 0$. If

$$C_k = \sup_x u_{x_k x_k}(x,0) \quad \text{and} \quad C_{\text{Lap}} = \sup_x \Delta\, u(x,0),$$

then

$$\sup_x u_{x_k x_k}(x,t) \le \frac{1}{C_k^{-1} + t}, \qquad \text{and} \qquad \sup_x \Delta u(x,t) \le \frac{1}{C_{\text{Lap}}^{-1} + t/d}.$$

Proof. First consider the non-viscous case $\beta^{-1} = 0$. Define $g(x,y) = f(y) + \frac{1}{2t}\,|x-y|^2$ to be a function of x parametrized by y. Each $g(x,y)$, considered as a function of x, is semi-concave with constant $1/t$. Note that $u(x,t)$ is the solution of (4.15) which is given by the Hopf-Lax formula (4.16) and is expressed as the infimum of functions like $g(x,y)$. This shows that $u(x,t)$ is also semi-concave with the same constant $1/t$.

Next consider the viscous case $\beta^{-1} > 0$. Let $w = u_{x_k x_k}$, differentiating (4.1) twice gives

$$w_t + \nabla u \,\cdot\, \nabla w + \sum_{i=1}^{d} u_{x_i x_k}^2 = \frac{\beta^{-1}}{2}\,\Delta w. \tag{4.37}$$

Using $u_{x_k x_k}^2 = w^2$, we have

$$w_t + \nabla u \cdot \nabla w - \frac{\beta^{-1}}{2}\,\Delta w \le -w^2.$$

Note that $w(x,0) \leq C_k$ and the function

$$\phi(x,t) = \frac{1}{C_k^{-1} + t}$$

is a spatially independent super-solution of the preceding equation with $w(x,0) \leq \phi(x,0)$. By the comparison principle applied to w and ϕ, we have

$$w(x,t) = u_{x_k x_k}(x,t) \leq \frac{1}{C_k^{-1} + t}$$

for all x and for all $t \geq 0$, which gives the first result. Now set $v = \Delta u$ and sum (4.37) over all k to obtain

$$v_t + \nabla u \cdot \nabla v + \sum_{i,j=1}^{d} u_{x_i x_j}^2 = \frac{\beta^{-1}}{2} \Delta v$$

Thus

$$v_t + \nabla u \cdot \nabla v - \frac{\beta^{-1}}{2} \Delta v \leq -\frac{1}{d} v^2$$

by the Cauchy-Schwartz inequality. Now apply the comparison principle to $v'(x,t) = \left(C_{\text{Lap}}^{-1} + t/d\right)^{-1}$ to obtain the second result. ■

4.5.3. Estimates on the Harmonic mean of the spectrum

Our semi-concavity estimates in Section 4.5.2 gave bounds on $\sup_x u_{x_k x_k}(x,t)$ and the Laplacian $\sup_x \Delta u(x,t)$. In this section, we extend the above estimates to characterize the eigenvalue spectrum more precisely. Our approach is motivated by experimental evidence Figure 3.1. The Hessian of a typical deep network at a location discovered by SGD has a very large proportion of its eigenvalues that are close to zero. For such a Hessian, instead of bounding the largest eigenvalue or the trace (which is also the Laplacian), we obtain estimates of the harmonic mean of the eigenvalues. This effectively discounts large eigenvalues and we get an improved estimate of the ones close to zero, that dictate the width of a minimum. The harmonic mean of a vector $x \in \Omega$ is

$$\text{HM}(x) = \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{x_i} \right)^{-1}.$$

The harmonic mean is more sensitive to small values as seen by

$$\min_i x_i \leq \text{HM}(x) \leq d \min_i x_i.$$

which does not hold for the arithmetic mean. To give an example, the eigenspectrum in Figure 3.1 has an arithmetic mean of 0.0029 while its harmonic mean is $\approx 10^{-10}$ which better reflects the large proportion of almost-zero eigenvalues.

Lemma 4.18. If Λ is the vector of eigenvalues of the symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$ and $D = (a_{11}, \ldots, a_{nn})$ is its diagonal vector,

$$\text{HM}(\Lambda) \leq \text{HM}(D).$$

Proof. Majorization is a partial order on vectors with non-negative components (Marshall et al., 1979, Chap.3) and for two vectors $x, y \in \Omega$, we say that x is majorized by y and write $x \preceq y$ if $x = Sy$ for some doubly stochastic matrix S. The diagonal of a symmetric, positive semi-definite matrix is majorized by the vector of its eigenvalues (Schur, 1923). For any Schur-concave function $f(x)$, if $x \preceq y$ we have

$$f(y) \leq f(x)$$

from Marshall et al. (1979, Chap.9). The harmonic mean is also a Schur-concave function which can be checked using the Schur-Ostrowski criterion

$$(x_i - x_j) \left(\frac{\partial \text{ HM}}{\partial x_i} - \frac{\partial \text{ HM}}{\partial x_j} \right) \leq 0 \quad \text{for all } x \in \Omega_+$$

and we therefore have

$$\mathrm{HM}(\Lambda) \leq \mathrm{HM}(D).$$

■

Lemma 4.19. If $u(x,t)$ is a solution of (4.1) and $C \in \Omega$ is such that $\sup_x \; u_{x_k x_k}(x,t) \leq C_k$ for all $k \leq n$, then at a local minimum x^* of $u(x,t)$,

$$\mathrm{HM}(\Lambda) \leq \frac{1}{t + \mathrm{HM}(C)^{-1}}$$

where Λ is the vector of eigenvalues of $\nabla^2 u(x^*,t)$.

Proof. For a local minimum x^*, we have $\mathrm{HM}(L) \leq \mathrm{HM}(D)$ from the previous lemma; here D is the diagonal of the Hessian $\nabla^2 u(x_{\min},t)$. From Lemma 4.17 we have

$$u_{x_k x_k}(x^*,t) \leq C_k(t) = \frac{1}{t + C_k^{-1}}.$$

The result follows by observing that

$$\mathrm{HM}(C(t)) = \frac{n}{\sum_{i=1}^n \; C_k(t)^{-1}} = \frac{1}{t + \mathrm{HM}(C)^{-1}}.$$

■

4.5.3.1. Wider minima for controlled SGD. We connected the width of regions that SGD converges to to generalization error of deep network in Chapter 3. Larger the width smaller is the difference between the performance of a classifier on the training set and an unknown validation set. In this section, we explain how we can use the regularized loss function $f_\gamma(x)$ as a proxy for a robust definition of the width of a local minimum. This is achieved by expanding $f(y)$ around the minimum x^*.

Definition 4.20 (Width of a region). The width of a region $x \in \Omega$ for the energy landscape $f(x)$ is defined to be the negative Local Entropy, i.e., it is simply $-f_\gamma(x)$.

Note the width as defined above is a function of the parameter γ. Suppose x is a local minimum of the function $f(x)$, which need not be smooth. Given the tolerance $\varepsilon > 0$, define the robust width, γ^* of the minimum at x to be the largest value γ^* so that

$$f_\gamma(x) - f(x) \leq \varepsilon, \quad \text{for all } 0 \leq \gamma \leq \gamma^* \tag{4.38}$$

This definition is motivated by the fact that near a local minimum, $f_\gamma(x)$ is approximated by the convolution of f with a Gaussian of width γ. So if (4.38) holds, the values of $f(x)$ are nearly constant in the neighborhood $|y-x| \leq \gamma^*$.

For small enough γ,

$$\begin{aligned}
f_\gamma(x^*) &= -\log \int_y \exp\left(-\frac{|x^*-y|^2}{2\gamma} - f(y)\right) \, \mathrm{d}y \\
&\approx f(x^*) - \log \int_y \exp\left[-\frac{1}{2}\,(x^*-y)^\top \left(\nabla^2 f(x^*) + \gamma^{-1} I\right)(x^*-y)\right] \, \mathrm{d}y \\
&= f(x^*) + \log \det\left(\nabla^2 f(x^*) + \frac{I}{\gamma}\right) + \text{constant} \\
&= \sum_{i=1}^n \lambda_i\left(\nabla^2 f(x^*) + \frac{1}{\gamma}\right),
\end{aligned}$$

up to constants; here $\lambda_i(A)$ is the i^{th} eigenvalue of the matrix A. Note that since x^* is a minimum $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is a positive semi-definite matrix. Smaller the eigenvalues λ_i, smaller the Local Entropy and wider the minimum.

We now have from Theorem 4.11 that

$$\mathbb{E}\left[f_\gamma(X_{\text{csgd}}(t))\right] \le \mathbb{E}\left[f_\gamma(X_{\text{sgd}}(t))\right] - \frac{1}{2}\,\mathbb{E}\left[\int_0^t |\alpha(X_{\text{csgd}}(s),s)|^2\,\mathrm{d}s\right];$$

for any fixed $t > 0$. This suggests that the width of a region obtained by $X_{\text{csgd}}(t)$ is larger than the width of a region obtained by $X_{\text{sgd}}(t)$ if both controlled SGD and SGD are trained for the same time t.

Remark 4.21 (Measuring the width of a region). A definition of the "width" of a region in a high-dimensional energy landscape is the subject of much debate in contemporary literature. In this thesis, we are primarily concerned with the width as the volume of the entire level set of a particular location. The Local Entropy integral

$$f_\gamma(x) = -\log \int_{y\in\Omega} e^{-f(y)-\frac{1}{2\gamma}|y-x|^2}\,\mathrm{d}y$$

measures this volume and is quite different from a localized measure of the width, say the Hessian as considered in Dinh et al. (2017) or the "sharpness of a region" as used in Keskar et al. (2016) etc. The loss functions of deep networks with ReLU non-linearities are piecewise-linear (Montufar et al., 2014). We expect that the loss layer such as softmax changes adds a small amount of curvature; nevertheless the Local Entropy integral is meaningful while the Hessian is degenerate for such loss functions.

4.6. Empirical validation

4.6.1. Algorithmic details

4.6.1.1. Entropy-SGD. We first derive the discrete time update equations for Entropy-SGD again using the SDE-based development in this chapter. The equation for $Y(s)$ in (4.21) is discretized using the Euler-Maruyama method with time step η_y as

$$y^{k+1} = y^k - \eta_y\,\left[\nabla f(y^k) + \frac{y^k - x^k}{\gamma}\right] + \sqrt{\eta_y\,\beta^{-1}}\,\varepsilon^k \tag{4.39}$$

where ε_k are zero-mean, unit variance Gaussian random variables. In the absence of the full gradient $\nabla f(y^k)$, we use the mini-batch gradient $\nabla f_{\text{b}}(y^k)$ in its place with the corresponding increased variance.

$$y^{k+1} = y^k - \eta_y\,\left[\nabla f_{\text{b}}(y^k) + \frac{y^k - x^k}{\gamma}\right] + \sqrt{\eta_y\,\beta_{\text{ex}}^{-1}}\,\varepsilon^k. \tag{4.40}$$

which corresponds to (4.39) with

$$\beta^{-1} = \beta_{\text{b}}^{-1} + \beta_{\text{ex}}^{-1}$$

In practice, we can set $\beta_{\text{ex}}^{-1} = 0$, i.e., we can rely only on the noise of the mini-batch updates. We initialize $y^k = x^k$ if k/L is an integer. The number of time steps taken for the Y variable is set to be L. This corresponds to $\varepsilon = 1/L$ in (4.20) and to $T = 1$ in (4.23). This results in a discretization of the equation for x in (4.21) given by

$$x^{k+1} = \begin{cases} x^k - \eta\,\gamma^{-1}\,\left(x^k - \langle y\rangle^k\right) & \text{if } k/L \text{ is an integer,} \\ x^k & \text{otherwise;} \end{cases} \tag{4.41}$$

Since we may be far from the limit $\varepsilon \to 0$, $T \to \infty$, instead of using the linear averaging for $\langle Y\rangle$ in (4.23), we will use a forward looking average. Such an averaging gives more weight to the later steps which is beneficial for large values of γ when the invariant measure $\rho^\infty(y;\,x)$ in (4.23) does not converge quickly. The two algorithms we describe will differ in the choices of L and β_{ex}^{-1} and the definition of $\langle y\rangle$. For Entropy-SGD, we set $L = 20$, $\beta_{\text{ex}}^{-1} = 10^{-8}$, and set $\langle Y\rangle$ to be

$$\langle y\rangle^{k+1} = \alpha\,\langle y\rangle^k + (1-\alpha)\,y^{k+1};$$

The parameter α is used to perform an exponential averaging of the iterates y^k. We set $\alpha = 0.25$ for experiments in Section 4.6 using Entropy-SGD. This is equivalent to the implementation in Section 3.3. The step-size for the y^k dynamics is fixed to $\eta_y = 0.1$ whereas we set $\eta = 0.1$ and reduce it by a factor of 5 after every 3 epochs.

4.6.1.2. Non-viscous Hamilton-Jacobi (HJ). For the other algorithm which we denote as **HJ**, we set $L = 5$ and $\beta_{\text{ex}}^{-1} = 0$, and set $\langle y \rangle^k = y^k$ in (4.40) and (4.41), i.e., no averaging is performed and we simply take the last iterate of the y^k updates and take $\alpha = 0$.

We can also construct an equivalent system of updates for **HJ** by discretizing Section 4.3.2 and again setting $\beta_{\text{ex}}^{-1} = 0$; this exploits the two different ways of computing the gradient in Lemma 4.1 and gives

$$\begin{aligned} y^{k+1} &= \left(1 - \gamma^{-1}\ \eta_y\right)\ y^k + \eta_y\ \nabla f_{\acute{e}}(x^k - y^k) \\ x^{k+1} &= \begin{cases} x^k - \eta\ \nabla f_{\acute{e}}(x^k - y^k) & \text{if } k/L \text{ is an integer,} \\ x^k & \text{else;} \end{cases} \end{aligned} \tag{4.42}$$

and we initialize $y^k = 0$ if k/L is an integer.

4.6.1.3. Heat equation. We perform the update corresponding to (4.28)

$$x^{k+1} = x^k - \frac{\eta}{L}\ \left[\sum_{i=1}^{L} \nabla f_{\acute{e}}(x^k + \varepsilon^i)\right] \tag{4.43}$$

where ε^i are Gaussian random variables with zero mean and variance γI. Note that we have implemented the convolution in (4.28) as an average over Gaussian perturbations of the parameters x. We again set $L = 20$ for our experiments in Section 4.6.

Other details such as adding momentum to first order stochastic updates and scoping of the parameter γ are standard and as discussed in Section 3.2.3.

This gives us four algorithms to compare and contrast in the sequel:

(i) **Entropy-SGD:** the algorithm from Chapter 3 and as implemented according to Section 4.6.1.1,
(ii) **HEAT:** smoothing by the heat equation described in Section 4.6.1.3,
(iii) **HJ:** updates for the non-viscous Hamilton-Jacobi equation described in Section 4.6.1.2,
(iv) **SGD:** stochastic gradient descent given by (4.6),

4.6.2. MNIST

Consider a "fully-connected" network on MNIST

$$\text{mnistfc}: \ \text{input}_{784} \to \text{drop}_{0.2} \to \underbrace{\text{fc}_{1024} \to \text{drop}_{0.2}}_{\times 2} \to \text{fc}_{10} \to \text{softmax}.$$

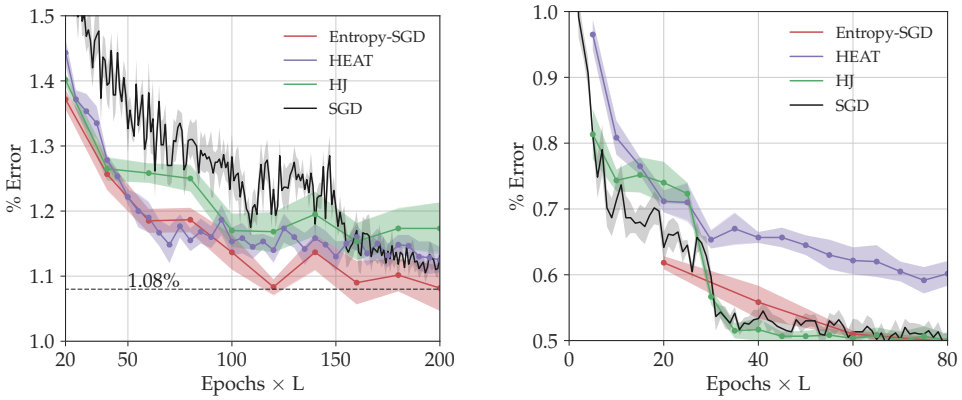
The input layer reshapes each MNIST image as a vector of size 784. The notation fc_d denotes a "fully connected" dense matrix with d output dimensions, followed by the ReLU nonlinearity and an operation known as batch normalization (Ioffe and Szegedy, 2015) which whitens the output of each layer by subtracting the mean across a mini-batch and dividing by the standard deviation. The notation drop_p denotes the dropout layer (Srivastava et al., 2014) which randomly sets a fraction p of the weights to zero at each iteration; this is a popular regularization technique in deep learning, see Kingma et al. (2015); Achille and Soatto (2016) for a Bayesian perspective. For 10 classes in the classification problem, we create an output vector of length 10 in the last fc layer followed by softmax which picks the largest element of this vector, which is interpreted as the output (or "prediction") by this network. We use the standard cross-entropy loss for penalizing incorrect predictions for all networks considered in this article. The mnistfc network has $n = 1.86$ million parameters.

As Figure 4.2a and Table 4.1 show, the final validation error for all algorithms is quite similar. The convergence rate varies, in particular, Entropy-SGD converges fastest in this case. Note that mnistfc is a small network and the difference in the performance of the above algorithms, e.g., 1.08 % for Entropy-SGD versus 1.17 % for HJ, is minor.

Our second network for MNIST is a convolutional neural network (CNN) denoted as follows:

$$\text{lenet}: \ \text{input}_{28\times 28} \ \to \text{conv}_{20,5,3} \to \text{drop}_{0.25} \to \text{conv}_{50,5,2} \to \text{drop}_{0.25} \to \text{fc}_{500} \to \text{drop}_{0.25} \to \text{fc}_{10} \to \text{softmax}.$$

The notation $\text{conv}_{c,k,m}$ denotes a 2D convolutional layer with c output channels, each of which is the sum of a channel-wise convolution operation on the input using a learnable kernel of size $k \times k$. It further adds ReLU nonlinearity, batch normalization and an operation known as max-pooling which down-samples the image by picking the maximum over a patch of size $m \times m$. Convolutional networks typically perform much better than fully-connected ones on image classification task despite having fewer parameters, lenet has only $n = 131,220$ parameters.



(A) mnistfc: Validation error

(B) lenet: Validation error

FIGURE 4.2. PDE-based optimization algorithms on mnistfc and lenet on MNIST

The results for lenet are described in Figure 4.2b and Table 4.1. This is a convolutional neural network and performs better than mnistfc on the same dataset. The final validation error is very similar for all algorithms at 0.50 % with the exception of the heat equation which only reaches 0.59 %. We can also see that the other algorithms converge quickly, in about half the number of effective epochs as that of mnistfc.

4.6.3. CIFAR

The CIFAR-10 dataset is more complex than MNIST and fully-connected networks typically perform poorly. We will hence employ a convolutional network for this dataset. We use the All-CNN-C architecture introduced by Springenberg et al. (2014) and add batch-normalization:

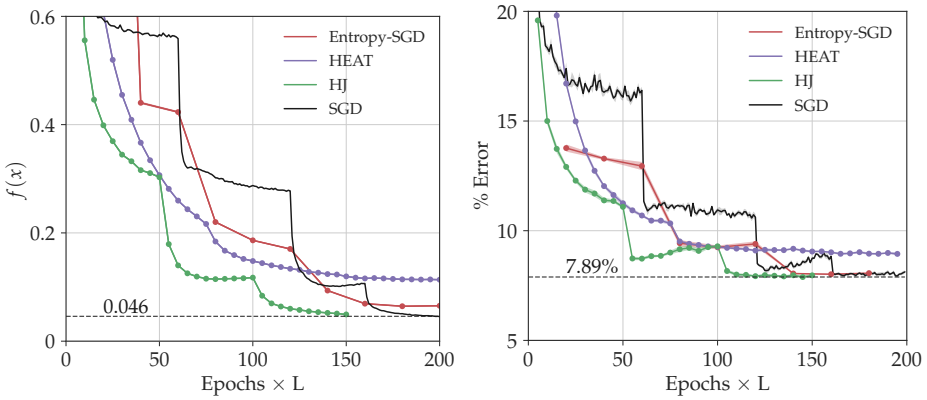
$$\text{allcnn}: \ \text{input}_{3\times 32\times 32} \ \to \text{drop}_{0.2} \to \text{block}_{96,3} \to \text{block}_{192,3} \to \underbrace{\text{conv}_{192,3}}_{\times 2} \to \text{conv}_{10} \to \text{mean-pool}_{10} \to \text{softmax}.$$

where

$$\text{block}_{d,3}: \ \text{conv}_{d,3,1} \to \text{conv}_{d,3,1} \ \to \text{conv}^{*}_{d,3,1} \to \text{drop}_{0.5}.$$

The final convolutional layer in the above block denoted by conv^{*} is different from others; while they perform convolution at every pixel otherwise known as a "stride" of 1 pixel, conv^{*} on the other hand uses a stride of 2; this results in the image being down-sampled by a factor of two. Note that $\text{conv}_{c,k,m}$ with $m = 1$ does not result in any down-sampling. Max-pooling usually results in a drastic reduction of the image size and by replacing it with a strided convolution, allcnn achieves improved performance with much fewer parameters than many other networks on CIFAR-10. The final layer denoted as mean-pool takes an input of size $10 \times 8 \times 8$ and computes the spatial average to obtain an output vector of size 10. This network has $n = 1.67$ million parameters.

Figure 4.3a and Figure 4.3b show the training loss and validation error for the allcnn network on the CIFAR-10 dataset. The Hamilton-Jacobi equation (HJ) obtains a validation error of 7.89 % in 145 epochs and thus



(A) allcnn: Training loss

(B) allcnn: Validation error

FIGURE 4.3. PDE-based optimization algorithms on CIFAR-10

performs best among the algorithms tested here; it also has the lowest training cross-entropy loss of 0.046. Note that both HJ and Entropy-SGD converge faster than SGD. The heat equation performs again poorly on this dataset and has a much higher validation error than others (9.04%). This is consistent with our discussion in Section 4.3.4 which suggests that the viscous or non-viscous HJ equations result in better smoothing than the heat equation.

Model	Entropy-SGD	HEAT	HJ	SGD
mnistfc	**1.08 ± 0.02 @ 120**	1.13 ± 0.02 @ 200	1.17 ± 0.04 @ 200	1.10 ± 0.01 @ 194
lenet	**0.5 ± 0.01 @ 80**	0.59 ± 0.02 @ 75	**0.5 ± 0.01 @ 70**	**0.5 ± 0.02 @ 67**
allcnn	7.96 ± 0.05 @ 160	9.04 ± 0.04 @ 150	**7.89 ± 0.07 @ 145**	7.94 ± 0.06 @ 195

TABLE 4.1. Summary of experimental results: Validation error (%) @ Effective epochs. The first two rows correspond to low-dimensional problems. In these cases, all algorithms obtain comparable, state-of-the-art validation error; Entropy-SGD is significantly faster at training mnistfc while SGD is slightly faster for lenet. In the largest problem on CIFAR-10, HJ has the best validation error and is the fastest.

4.7. DISCUSSION

Our results apply nonlinear PDEs, stochastic optimal control and stochastic homogenization to the analysis of two recent and effective algorithms for the optimization of neural networks. Our analysis also leads to new and improved algorithms for non-convex optimization.

We replaced the standard stochastic gradient descent (SGD) algorithm for the function $f(x)$, with SGD on the two variable function $H(x,y;\gamma) = f(y) + \gamma^{-1}|x-y|^2/2$, along with a small parameter ε (4.21). Using the new system, in the limit $\varepsilon \to 0$, we can provably, and quantitatively, improve the expected value of the original loss function. The effectiveness of our algorithm comes from the connection, via homogenization, of system of SDEs to the gradient of the function $u(x,t)$ which is the solution of the (4.1) PDE with initial data $f(x)$. The function H is more convex in y that the original function f. The convexity of H in y is related to the exponentially fast convergence of the dynamics, a connection explained by the celebrated gradient flow interpretation of Fokker-Planck equation (Jordan et al., 1998b).

Section 4.3 shows that minimizing Local Entropy is equivalent to the influential distributed algorithm Elastic-SGD (Zhang et al., 2015a) if the underlying stochastic dynamics is ergodic. We will exploit this insight in the next chapter to develop new state-of-the-art algorithms for distributed training of deep networks.

On a practical level, training deep neural networks with a large number of hyper-parameters is very costly, in terms of both human effort and computer time. Our analysis lead to better understanding of the parameters involved in the algorithms, and provides an insight into the choice of hyper-parameters for these methods. In particular (i) the parameter γ is now understood as the time t, in the PDE (4.1) (ii) scoping of γ, which was seen as a heuristic, can now be interpreted as sending a regularization parameter to zero (iii) we set the extrinsic noise parameter β_{ex}^{-1} to zero, resulting in a simpler, more effective algorithm, and (iv) we now understand that below a critical value of γ (which is determined by the requirement that H be convex in y), the convergence of the dynamics in (4.21) to the invariant measure is exponentially fast.

While simulated annealing and related methods work by modulating the level of noise in the dynamics, our algorithm works by modulating the smoothness of the underlying loss function. While most algorithms used in deep learning derive their motivation from the literature on convex optimization, the algorithms we have presented here are specialized to non-convex loss functions and have been shown to perform well on these problems both in theory and in practice.

CHAPTER 5

Distributed algorithms for training deep networks

The dramatic success of deep networks has fueled the growth of massive datasets, e.g. Google's JFT dataset has 100 million images. This in turn has prompted researchers to employ even larger models. Our desiderata in such a scenario is as follows. We would like algorithms that can:

(i) parallelize a deep network and its dataset over large compute clusters, i.e., scale well in terms of both computational and communication efficiency,
(ii) improve, or at least preserve, generalization performance and,
(iii) avoid introducing additional hyper-parameters in an already complex distributed system.

In this chapter, we present an algorithm that achieves these demands. It is named Parle and it (a) replicates the model and splits the dataset over multiple GPUs, (b) has very low communication requirements, (c) significantly improves upon the generalization performance of stochastic gradient descent, and (d) exploits proximal point algorithms from the convex optimization literature to be insensitive to hyper-parameters.

We demonstrate extensive empirical evidence that it obtains significant performance improvements and obtains nearly state-of-the-art generalization errors; it also obtains a $2-4\times$ wall-clock time speedup over data-parallel SGD. Moreover, we use ideas from the convex optimization literature to show that Parle is insensitive to hyper-parameters; all experiments in this chapter were conducted with the same hyper-parameters. We do not introduce any additional hyper-parameters over SGD.

5.0.1. Motivation

Training independent copies of a deep network in parallel is difficult. Let us discuss an experiment that explores averaging such models and motivates our approach here. We trained 6 instances of the allcnn architecture of Springenberg et al. (2014) on the CIFAR-10 dataset. These networks converge to a training error of $4.08 \pm 0.9\%$ and a validation error of $8.04 \pm 0.16\%$. Averaging their softmax predictions performs only slightly better than any individual network and the ensemble gets a validation error of 7.84%, indeed if we look at the correlation of the softmax predictions of these networks, it shows that they make mistakes on the same examples. This marginal improvement comes at the cost of a large test-time performance penalty.

A model that consists of the average weights of these independent networks ("one shot averaging") performs extremely poorly: it obtains 89.9% validation error, which is close to random guessing. This is indeed expected given the non-convexity of the loss function for deep networks: although individual networks might converge to good local minima their average need not even be locally optimal.

A typical deep network possess permutation symmetries, i.e., if the first and last layers are fixed, filters on intermediate layers can be permuted without changing the predictions. For a fully-connected network, this is akin to permuting the rows and columns of successive weight matrices. Post-training, we aligned each network to the first one by permuting its filters using a greedy layer-wise matching algorithm. Figure 5.1 shows the average overlap of the weights of these aligned filters; it can be seen as a permutation invariant metric between two networks. Even after aligning with respect to permutations, these copies are very far away from each other in the weight space.

It is surprising that while a naively averaged model has close to 90% validation error, a model obtained by averaging after aligning the weights performs much better at 18.7% validation error. This suggests that if we

1.00	0.22	0.22	0.23	0.22	0.23
0.22	1.00	0.22	0.22	0.22	0.22
0.22	0.22	1.00	0.23	0.23	0.23
0.23	0.22	0.23	1.00	0.23	0.23
0.22	0.22	0.23	0.23	1.00	0.23
0.23	0.22	0.23	0.23	0.23	1.00

FIGURE 5.1. Permutation invariant distance of independently trained neural networks: we plot the quantity $\frac{\langle x^a, x^b \rangle}{|x^a||x^b|}$ for six networks trained independently using SGD on CIFAR-10. Note that the overlap between any two independent networks is quite similar. This matrix is reminiscent of the 1-RSB (replica symmetry breaking) ansatz in statistical physics, a phenomenon where the Gibbs distribution concentrates on multiple equivalent local minima. The algorithm Parle developed in this chapter exploits this observation for distributed training.

can force the copies to remain aligned to each other during training, we can obtain one single average model at the end that combines these copies. Parle uses a quadratic distance to align two copies during training. The loss function in (5.3) ensures that two replicas x^a and x^b are aligned through the quadratic forcing term $\sum_{a=1}^{n} \|x^a - x\|^2$. This encourages members of the ensemble to be close to each other in Euclidean space. For a small enough $\rho > 0$, these replicas are constrained to have a large overlap while still minimizing their individual loss functions $f(x^a)$. As $\rho \to 0$ towards the end of training, the overlap goes to unity.

5.0.2. Approach

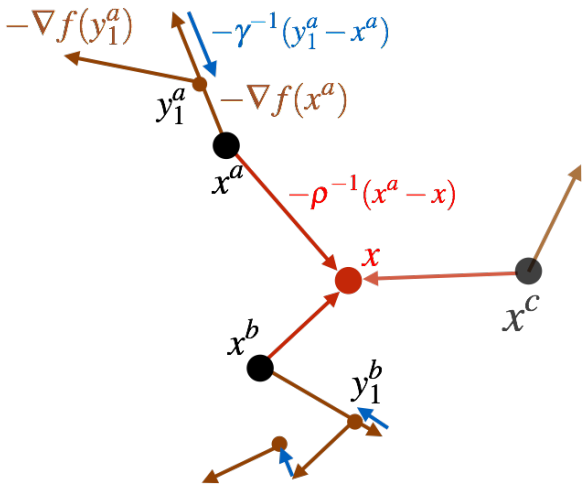


FIGURE 5.2. Parle: schematic of the updates

Parle creates multiple replicas of a deep network along with one master. Each replica maintains its own local dataset that is either replicated or possibly disjoint subsets of a larger dataset. If the number of replicas is n, we use the following notation.

$$\begin{aligned} \text{replicas}: &\ \left\{x^1, x^2, \ldots, x^n\right\}, \\ \text{datasets}: &\ \left\{\Xi^1, \Xi^2, \ldots, \Xi^n\right\}, \\ \text{master}: &\ x. \end{aligned}$$

We denote different replicas by alphabetical indices, $x^a, x^b, \ldots$ etc. Let $f(x^a,\ \Xi^a)$ denote the training loss, say cross-entropy with weight-decay, of the replica x^a on its dataset Ξ^a.

A simple technique to train such a system is to couple the replicas using a quadratic penalty:

$$\underset{x,\ x^1,\ \ldots,\ x^n}{\arg\min} \sum_{a=1}^{n} f(x^a,\ \Xi^a) + \frac{1}{2\rho}\ \|x^a - x\|^2. \tag{5.1}$$

This is similar to Elastic-SGD of Zhang et al. (2015a) except that have made each replica work with its own dataset. The parameter ρ controls how far a replica x^a is allowed to move from the master x. Solving this problem involves taking one stochastic gradient step on each replica and then communicating the weights to-and-from the master. Therefore, even though Elastic-SGD has good computational efficiency, it has poor communication efficiency if n is large.

In order to reduce this communication, we replace $f(x^a,\ \Xi^a)$ above by the convolution

$$f_\gamma(x^a,\ \Xi^a) := -\log\left(G_\gamma\ *\ e^{-f(x^a,\ \Xi^a)}\right); \tag{5.2}$$

where G_γ is the Gaussian kernel with variance γ. This function is called Local Entropy that we introduced in Chapter 3 and it can be minimized by a sequential algorithm called Entropy-SGD (Chaudhari et al., 2016). This new loss function leads to two advantages: (i) it does not need to communicate with the master as frequently, and (ii) as shown in Chapter 4, running one copy of Local Entropy is equivalent to running the entire Elastic-SGD system in (5.1).

Parle therefore solves the optimization problem

$$\underset{x,\ x^1,\ \ldots,\ x^n}{\arg\min} \sum_{a=1}^{n} f_\gamma(x^a,\ \Xi^a) + \frac{1}{2\rho}\ \|x^a - x\|^2. \tag{5.3}$$

The three algorithms differ in their communication requirements: Elastic-SGD involves large communication overhead with the master, Entropy-SGD is a sequential algorithm and thus cannot be parallelized. Parle leverages upon both of them, conceptually, running Parle is akin to running Elastic-SGD with $\mathcal{O}(n^2)$ replicas without increasing the communication cost at all.

We also let

$$\gamma,\ \rho \to 0$$

as training progresses. It increases the coupling strength in Elastic-SGD and reduces the bandwidth of the Gaussian kernel in Local Entropy. This is particularly powerful when replicas operate with disjoint datasets. Reducing ρ forces them to converge to the same region in the weight space and thereby, a replica x^b performs well on the data Ξ^a without even seeing it. Thus, while Parle maintains n replicas during training, scoping collapses them together with the master towards the end of training to result in one single model. This technique leads to significant gains in generalization error in our experiments.

5.1. Related work

There are two primary ways to parallelize the training of deep networks. The first, called **model parallelism**, distributes a large model across multiple GPUs or compute nodes (Krizhevsky, 2014; Dean et al., 2012). This does not scale well because intermediate results of the computation need to be transmitted to different compute

nodes quickly, which puts severe limits on the latencies that the system can tolerate, e.g., large neural networks with recurrent or fully-connected layers are not amenable to model parallelism.

Data parallelism. This is more widely used (Jin et al., 2016; Moritz et al., 2015; Taylor et al., 2016) and maintains multiple copies of the network on different GPUs or compute nodes. Each mini-batch is split evenly amongst these copies who compute the gradient on their samples in parallel. A parameter server then aggregates these partial gradients and updates the weights. This aggregation can be either synchronous or asynchronous. The former is guaranteed to be exactly equivalent to a non-distributed, sequential implementation while the latter can work with large communication latencies but offers guarantees only for convex or sparse problems (Recht et al., 2011; Duchi et al., 2013).

Gradient aggregation. Both synchronous and asynchronous gradient aggregation necessitate a high per-worker load (Qi et al., 2016) if they are to scale well. This is difficult to do in practice because small mini-batches require frequent communication while large batch-sizes suffer from a degradation of the generalization performance. Previous work in the literature has mostly focused on minimizing the communication requirements using stale gradients in HogWild! fashion (Recht et al., 2011; Zhang et al., 2015b), extreme quantization of gradients (Seide et al., 2014) etc. These heuristics have shown impressive performance although it is hard to analyze their effect on the underlying optimization for deep networks. Synchronous approaches often require specialized hardware and software implementations to hide communication bottlenecks (Wu et al., 2015; Abadi et al., 2015; Chen et al., 2015b). Recent work by Goyal et al. (2017) shows that one can indeed obtain generalization performance comparable to small batch-sizes by a careful tuning of the learning rate. While large batches, in theory, can enable large learning rates, as is often seen in practice, optimization of a deep network with a large learning rate is difficult and even the training loss may not decrease quickly. The authors demonstrate that a warm-up scheme that increases the learning rate from a small initial value followed by the usual annealing leads to similar training and validation error curves as those of small mini-batches. Parle can benefit from both synchronous and asynchronous gradient aggregation, indeed, each replica in (5.3) can itself be data-parallel. Our work shows that exploiting the properties of the optimization landscape leads to better generalization performance.

Ensemble methods. Ensemble methods train multiple models and average their predictions to improve generalization performance[1]. Deploying ensembles of deep networks in practice however remains challenging due to both the memory footprint and test-time latency of state-of-the-art networks. For instance, an ensemble of an object detection network like YOLO (Redmon et al., 2016) that works at 45 frames per second on PASCAL VOC can no longer be used in a real-time environment like a self-driving car (Leonard et al., 2008) with limited computational budget. In contrast to an ensemble, Parle results in one single model that performs well at test-time. Second, training multiple models of an ensemble is computationally expensive (Loshchilov and Hutter, 2016; Huang et al., 2017), and as our experiment in Section 5.0.1 shows, the ensemble obtains a marginal improvement over each individual model. Parle maintains a correlated, robust ensemble during training and returns the average model that obtains better errors than a naive ensemble.

5.2. Background

This section introduces Elastic-SGD in Section 5.2.1 and Entropy-SGD in Equation (5.5). We then discuss how they are in fact equivalent to each other in Section 5.2.3. For clarity in notation, we will suppress the dependence of the loss of each replica on its the dataset Ξ^a in this section. One can simply replace x^a by (x^a, Ξ^a) to get the updates that show the dependence on the dataset.

[1] the top-performing methods on ImageNet are ensembles of deep networks (http://image-net.org/challenges/LSVRC/2016/results)

5.2.1. Elastic-SGD

Each replica one step of SGD on its individual loss and communicates its updates to the master in Elastic-SGD. The updates at the k^{th} iteration are:

$$x^a_{k+1} = x^a_k - \eta \left[\nabla f_{\mathcal{B}}(x^a_k) + \frac{1}{\rho_k}\,(x^a_k - x_k)\right] \quad \forall\, a \leq n \tag{5.4a}$$

$$x_{k+1} = x_k - \frac{n\,\eta'}{\rho}\left(x_k - \frac{1}{n}\sum_{a=1}^{n} x^a\right), \tag{5.4b}$$

for all replicas x^a with $a \leq n$. The master therefore updates itself towards the average of the replicas.

5.2.2. Entropy-SGD

This is the sequential algorithm discussed in Chapter 3 that minimizes Local Entropy. It maintains two variables: the iterates y^a_k that live in the inner loop and the "snapshot" weights x^a_k that are updated only in an outer loop. We can write these together as follows.

$$y^a_{k+1} = y^a_k - \eta \left[\nabla f_{\mathcal{B}}(y^a_k) + \frac{1}{\gamma}\,(y^a_k - x^a_k)\right] \tag{5.5a}$$

$$z^a_{k+1} = \alpha\, z^a_k + (1-\alpha)\, y^a_{k+1} \tag{5.5b}$$

$$x^a_{k+1} = \begin{cases} x^a_k - \frac{\eta'}{\gamma}\left(x^a_k - z^a_{k+1}\right) & \text{if } \frac{k}{L} \text{ is an integer;} \\ x^a_k & \text{else.} \end{cases} \tag{5.5c}$$

The variables y^a_k are re-initialized to x^a_k every time $\frac{k}{L}$ is an integer. The y^a_k variable performs stochastic gradient descent on the original loss function with a proximal term

$$f(y^a_k) + \frac{1}{2\gamma}\|y^a_k - x^a_k\|^2.$$

This ensures that successive updates of y^a_k stay close to x^a_k. The z^a_k variables maintain a running exponential average of y^a_k.

We showed in Chapter 3 that these updates are equivalent to performing gradient descent on Local Entropy using a Markov chain Monte Carlo (MCMC) algorithm called stochastic gradient Langevin dynamics (Welling and Teh, 2011). The entire system Equation (5.5) is thus simply

$$x^a_{k+1} = x^a_k - \eta\, \nabla f_\gamma(x^a_k)$$

where

$$\nabla f_\gamma(x^a_k) = \gamma^{-1}\left(x^a_k - \left\langle y^a_k \right\rangle\right).$$

The notation $\left\langle y^a_k \right\rangle$ denotes the average of the y_k iterates; we have used exponential averaging in Equation (5.5b).

In this chapter, we introduce a slight modification in Equation (5.5a) as compared to Chaudhari et al. (2016), namely, we do not add any external noise. The gradients in a deep network are computed on mini-batches and therefore there is already some inherent stochasticity in Equation (5.5a). There are two hyper-parameters in Entropy-SGD:

$$\begin{aligned} \text{the number of inner loops} \quad & L, \\ \text{exponential averaging parameter} \quad & \alpha. \end{aligned} \tag{5.6}$$

5.2.3. Equivalence of Entropy-SGD and Elastic-SGD

Observe the iterations in Equation (5.4) and Equation (5.5). While Equation (5.4b) averages over the multiple replicas, the update in Equation (5.5c) does not talk to other replicas but updates using the run-time average of its inner loop computed in Equation (5.5b). This resemblance is not a coincidence. We proved in Chapter 4 that Elastic-SGD is equivalent to Entropy-SGD if the underlying loss function $f(\cdot)$ is ergodic. A special case

of this is when the sub-objective of Equation (5.5a) given by

$$y \mapsto f(y) + \frac{1}{2\gamma}\|y - x_k^a\|^2$$

is strictly convex in y which is true if

$$\nabla^2 f(y) + \gamma^{-1}\, I \succ 0.$$

This condition implies that the stochastic process of the y_k updates in Equation (5.5a) has an ergodic steady-state distribution whereby temporal averages in Equation (5.5b) are equivalent to spatial averages Equation (5.4b). One can also use techniques from statistical physics to show that the two objectives are equivalent (Baldassi et al., 2016a).

Note that Entropy-SGD is a sequential algorithm and hence hard to parallelize. The y_k^a updates in Equation (5.5a) form a single trajectory of L steps before the x_k^a update Equation (5.5c) and it is difficult to execute chunks of this trajectory independently. On the other hand, Elastic-SGD is a naturally parallelizable algorithm but suffers from a large communication overhead; every weight update Equation (5.4b) requires a reduce operation from all the replicas x_k^a and another broadcast of x_{k+1} to each of them. This becomes prohibitive for large deep networks. Fortunately, the two algorithms are equivalent, and we leverage this in Parle.

5.2.4. Scoping

We can see from (5.2) that

$$\begin{aligned} &\lim_{\gamma\to 0} f_\gamma(x^a) = f(x^a) \quad \text{and,} \\ &\lim_{\gamma\to\infty} f_\gamma(x^a) = \text{constant.} \end{aligned} \tag{5.7}$$

This is also true for the loss function of Elastic-SGD: as $\rho \to 0$, all the replicas and the master converge to the same global minimizer of $f(x)$. We therefore let both γ and ρ in Parle go to zero as training progresses. This collapses all replicas, including the master, to the same weight vector.

5.3. PARLE

We are now ready to give the updates for Parle. These updates amount to optimizing the loss function Equation (5.8) by stochastic gradient descent and can be thought of as "Entropy-SGD running in an inner loop of Elastic-SGD". For all replicas $a \le n$, we have:

$$y_{k+1}^a = y_k^a - \eta^{(0)}\left[\nabla f(y_k^a) + \frac{1}{\gamma_k}\,(y_k^a - x_k^a)\right] \tag{5.8a}$$

$$z_{k+1}^a = \alpha\, z_t^a + (1-\alpha)\, y_{k+1}^a \tag{5.8b}$$

$$x_{k+1}^a = \begin{cases} x_k^a - \frac{\eta^{(1)}}{\gamma_k}\left(x_k^a - z_{k+1}^a\right) - \frac{\eta^{(2)}}{\rho_k}\,\left(x_k^a - x_k\right) & \text{if } \frac{k}{L} \text{ is an integer} \\ x_k^a & \text{else,} \end{cases} \tag{5.8c}$$

$$x_{k+1} = \begin{cases} x_k - \frac{\eta^{(3)}\, n}{\rho_k}\left(x_k - \frac{1}{n}\,\sum_{a=1}^n x_k^a\right) & \text{if } \frac{k}{L} \text{ is an integer} \\ x_k & \text{else.} \end{cases} \tag{5.8d}$$

We again re-initialize y_k^a to x_k^a every time $\frac{k}{L}$ is an integer. Notice that Equation (5.8a) and Equation (5.8b) are the same as Equations (5.5a) and (5.5b). The update for x_k in Equation (5.8d) is also the same as the update for the reference variable x_k in Equation (5.4). The only difference is that Equation (5.8c) takes a gradient step using both the gradient of $f_\gamma(x^a)$ and the gradient of the elastic term $\rho^{-1}\left(x_k^a - x_k\right)$.

5.3.1. Many deputies under one sheriff

Consider an alternative optimization problem:

$$\underset{x,\, x^1,\ldots,\, x^n,\, y^1,\ldots,\, y^n}{\arg\min} \sum_{a=1}^{n} \left(\sum_{b=1}^{n} f(y^b) + \frac{1}{2\gamma} \|y^b - x^a\|^2 \right) + \frac{1}{2\rho} \|x^a - x\|^2. \tag{5.9}$$

Under an ergodicity assumption, from the discussion in Section 5.2, this is equivalent to (5.3). It is also equivalent to Equation (5.4) with a modified coupling between the workers where the constant ρ in (5.1) takes two different values. This loss function provides a completely distributed version of Parle and has the interpretation of "deputies" x^a connected to a "sheriff" x through the term $\frac{1}{2\rho}\|x^a - x\|^2$. Each deputy in turn controls the workers y^b through the proximal term $\frac{1}{2\gamma}\|y^a - x^a\|^2$.

Note that the variables $x, x^a, y^b \in \mathbb{R}^d$ are high-dimensional and there are n^2 copies. Optimizing (5.9) therefore involves a large communication overheard: each deputy communicates $\mathcal{O}(2nd)$ bits with its workers and the sheriff communicates another $\mathcal{O}(2nd)$ bits with each deputy. The total communication complexity is thus quadratic $\mathcal{O}(n^2d^2)$ at each weight update which is prohibitive for large d. On the other hand, using the updates in Equation (5.5) and Equation (5.4) to minimize (5.3) results in an amortized communication complexity of $\mathcal{O}(2nd/L)$ while remaining equivalent to (5.9).

Remark 5.1 (Running Parle on diverse computational platforms). We can extend the above loss function further and again replace $f(y^b)$ with either $f_\delta(y^b)$ or even the Elastic-SGD loss function. Coupled with the idea of HogWild! or stale gradients, one can achieve a seamless trade-off between replicas that may have a lot of computational budget but relatively scarce communication resources, e.g., GPUs, and replicas that can communicate faster than they can compute, e.g., CPUs and mobile devices. Entropy-SGD is naturally suited to the former and Elastic-SGD is naturally suited to the latter, the loss function of Parle shows how to couple such diverse computational platforms together.

Let us note that for different parameter choices, Parle degenerates into the following special cases:

(i) Stochastic gradient descent for $L = 1$, $n = 1$ with $\gamma = \infty$ and $\rho = \infty$,
(ii) Elastic-SGD for $L = 1$, $\gamma = \infty$,
(iii) Entropy-SGD for $n = 1$, $\rho = \infty$.

5.3.2. Hyper-parameter choice

We have used different learning rates in Equation (5.8) for each of the update steps, we now normalize them. First note that the learning rate η is typically annealed during the course of training for a deep network. Our scoping on the parameters γ_k and ρ_k interferes with the usual learning rate annealing. Moreover, normalizing the learning rates also makes our other hyper-parameter choices more robust. We set $\eta^{(0)}$ to the initial learning rate η_0, $\eta^{(1)} = \eta\gamma_k$, $\eta^{(2)} = \eta$, and $\eta^{(3)} = \frac{\rho_k}{n}$. Note that we do not scale $\eta^{(2)}$ with ρ_k in Equation (5.8c) so as to automatically get a weighted combination of the gradients of Local Entropy and the elastic term.

For all the experiments in this chapter, the parameters of Parle in Equation (5.8) are fixed to $L = 25$ and $\alpha = 0.25$. We use an exponential schedule for γ_k and ρ_k:

$$\gamma_k = \gamma_0 \left(1 - \frac{1}{2B}\right)^{\lfloor k/L \rfloor} \quad \text{and} \quad \rho_k = \rho_0 \left(1 - \frac{1}{2B}\right)^{\lfloor k/L \rfloor}, \tag{5.10}$$

where $\gamma_0 = 10^2$ and $\rho_0 = 1$ and B is the number of mini-batches in the global dataset. We clip γ at 1 and ρ at 0.1. The only remaining parameter is the learning rate η which we drop by a factor of $5 - 10$ when the validation error plateaus, Section 5.4 provides more details.

Remark 5.2 (Robustness to hyper-parameters). We have found that this algorithm is very robust to the hyper-parameters L, α and the two scoping parameters γ_0 and ρ_0. Increasing L implies that each replica takes

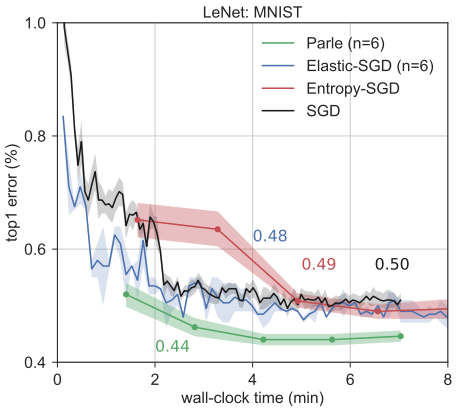


FIGURE 5.3. Validation error of Parle on lenet on MNIST

more gradient steps in the Entropy-SGD inner loop before it communicates with the master. Both the speed of convergence and the final generalization error are insensitive to the exact values of γ_0 or ρ_0.

Remark 5.3 (Nesterov's momentum). We use Nesterov's momentum (fixed to 0.9) for updating the variables y_k^a in Equation (5.8a) and x_k^a in Equation (5.8c). With our specific choice of $\eta^{(3)} = \rho_k/n$, the x_k update does not have first-order dynamics, and we therefore do not use momentum to update it. With other choices of $\eta^{(3)}$, one could use momentum for Equation (5.8d) as well, but we found this to converge marginally slower.

5.4. EMPIRICAL VALIDATION

This section discusses experimental results on a variety of benchmark image classification datasets, namely, MNIST (Section 5.4.2), CIFAR-10 and CIFAR-100 (Section 5.4.3) and SVHN (Section 5.4.4). Parle obtains nearly state-of-the-art errors with a significant wall-clock time speed up without any additional hyperparameters. In Section 5.5, we show that Parle obtains error rates better than SGD with full data even in the case when each replica only has access to a subset of the data.

For all our experiments, we use SGD with Nesterov's momentum (fixed to 0.9) as a baseline and compare the performance of Parle, Entropy-SGD and Elastic-SGD. We report the mean and standard deviation of the validation error over 3 runs for each algorithm with random initialization. For some networks that take a long time to train, we only report the results of a single run (see Table 5.1 and Table 5.2).

5.4.1. Implementation details and communication latencies

We implemented a parallel version of the updates in Equation (5.8), i.e., we execute multiple neural networks in the same process on a standard desktop with 3 GPUs. The communication with the master for the x_k variable happens via optimized NCCL[2] routines on PCI-E. This is implemented using PyTorch[3] in Python but it scales well to $n \approx 16$ replicas. In particular, the reduce operation required in Equations (5.8c) to (5.8d) does not incur a large communication overhead. For instance, on our machine, each mini-batch of size 128 for the wide residual network WRN-28-10 in Section 5.4.3 takes 528 ms on an average, while steps Equations (5.8c) to (5.8d) take 2.8 ms, their ratio is 0.52% and is therefore negligible. For the allcnn network in Section 5.5, this ratio is 0.43%. Let us however note that addressing communication bottlenecks in a distributed implementation is non-trivial. As an example, the authors in Goyal et al. (2017) perform gradient aggregation and backprop in parallel. Parle can also benefit from such techniques although that is not the focus here.

[2] https://devblogs.nvidia.com/parallelforall/fast-multi-gpu-collectives-nccl

[3] http://pytorch.org

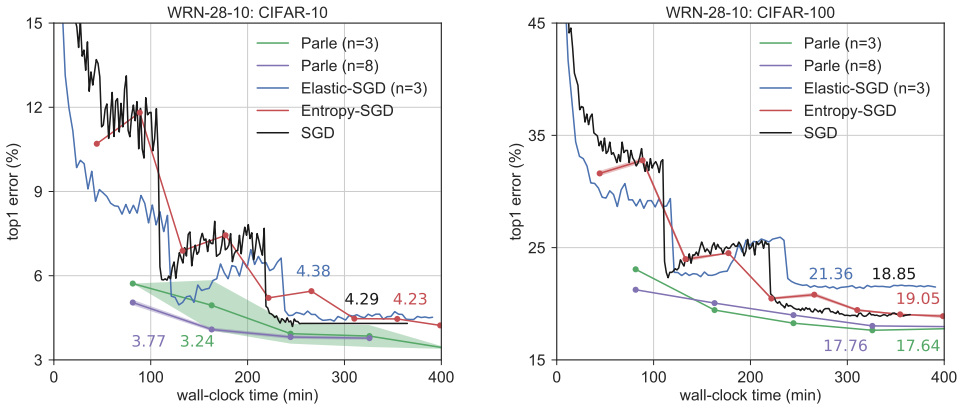
Remark 5.4 (Plotting against wall-clock time). Entropy-SGD and SGD are sequential algorithms while Parle and Elastic-SGD can run on all available GPUs simultaneously. In order to obtain a fair comparison, we run the former two in data-parallel fashion on three GPUs and plot all error curves against the wall-clock time. For the networks considered in our experiments, with a batch-size of 128, the efficiency of a data-parallel implementation in PyTorch is above 90% (except for lenet).

5.4.2. MNIST

We use the standard lenet architecture (LeCun et al., 1998) with ReLU nonlinearities (Nair and Hinton, 2010), batch-normalization (Ioffe and Szegedy, 2015). It has two convolutional layers with 20 and 50 channels, respectively, followed by a fully-connected layer of 500 hidden units that culminates into a 10-way softmax. Both the convolutional and fully-connected layers use a dropout (Srivastava et al., 2014) of probability 0.25. We do not perform any pre-processing for MNIST. The learning rate is initialized to 0.1 and dropped by a factor of 10 at epochs $[30, 60, 90]$ for SGD and only once, after the second epoch, for Entropy-SGD and Parle. Figure 5.3 shows the validation errors for lenet, Parle obtains a validation error of $0.44 \pm 0.01\%$ with three replicas as compared to about 0.48-0.5% on lenet with SGD, Entropy-SGD and Elastic-SGD.

5.4.3. CIFAR-10 and CIFAR-100

We use the WRN-28-10 network of Zagoruyko and Komodakis (2016) for these datasets; this architecture was shown to have very good empirical performance in spite of not being very deep. We use the same training pipeline and hyper-parameters as that of the original authors to enable an easy comparison. In particular, we perform global contrast normalization followed by ZCA normalization and train with data-augmentation which involves random mirror-flipping (with probability 0.5) and random crops of size 32×32 after padding the image by 4 pixels on each side. We use a dropout of probability 0.3 and weight decay of 5×10^{-4}. The learning rate is initialized to 0.1 and dropped by a factor of 5 at epochs $[60, 120, 180]$ for SGD and $[2, 4, 6]$ for Entropy-SGD and Parle. The former is taken from the original paper while the later was constructed using the heuristic that Parle and Entropy-SGD use $L = 25$ gradient evaluations per weight update.



(A) CIFAR-10 (B) CIFAR-100

FIGURE 5.4. Validation error of Parle on WRN-28-10 on CIFAR-10 (Figure 5.4a) and CIFAR-100 (Figure 5.4b)

Figures 5.4a to 5.4b show the empirical performance on CIFAR-10 and CIFAR-100 respectively with the same data summarized in Table 5.1. In our implementation, we obtained a validation error of 4.29% with SGD as compared to 3.89% by Zagoruyko and Komodakis (2016) for CIFAR-10, however our baseline for CIFAR-100 matches well with theirs, both obtain 18.85% validation error. On CIFAR-10, Parle with $n = 3$

replicas obtains a significantly better validation error of 3.24% while it obtains 17.64% error on CIFAR-100. Note that these are both more than 1% better than the baseline SGD with exactly the same network and pre-processing. For these datasets, we found that the benefit of adding more replicas is small, with $n = 8$, we see an initial speed up for both CIFAR-10 and CIFAR-100, but the network converges to a worse error with the same hyper-parameters. Note that if more GPUs are available, each replica can itself be run in a data-parallel fashion to further accelerate training time. Nevertheless, both versions of Parle are better than the baseline SGD implementation.

It is instructive to compare the performance of Elastic-SGD and Entropy-SGD with Parle. Since Parle essentially combines these two algorithms, as we saw in Section 5.3, it is a more powerful than either of them. This is corroborated by our experimental evidence. We also observed that Entropy-SGD obtains very similar errors as those of Elastic-SGD with scoping on the ρ parameter. Scoping was adapted from the results of Chaudhari et al. (2016) and in our experience, it improves the performance of Elastic-SGD significantly.

Our errors on CIFAR-10 and CIFAR-100 are better than those reported previously in published literature for a single model [4]. In fact, our result on CIFAR-10 is better than the 3.44% reported error in Huang et al. (2017) on an ensemble of six DenseNet-100 networks (Huang et al., 2016). Our result on CIFAR-100 is only slightly worse than a DenseNet-100 ensemble which gets 17.41% (Huang et al., 2017).

Model	Parle		Elastic-SGD		Entropy-SGD		SGD	
	Error	Time	Error	Time	Error	Time	Error	Time
lenet (MNIST, $n = 6$)	$\mathbf{0.44 \pm 0.01}$	**4.24**	0.48 ± 0.01	5	0.49 ± 0.01	6.5	0.5 ± 0.01	5.6
WRN-28-10 (CIFAR-10, $n = 3$)	$\mathbf{3.24 \pm 0.1}$	**400**	4.38	289	4.23	400	4.29	355
WRN-28-10 (CIFAR-100, $n = 3$)	**17.64**	**325**	21.36	317	19.05 ± 0.03	400	18.85	355
WRN-16-4 (SVHN)	1.68 ± 0.01	592	**1.57**	**429**	1.64	481	1.62	457

TABLE 5.1. Summary of experimental results: Validation error (%) at wall-clock time (min)

5.4.4. SVHN

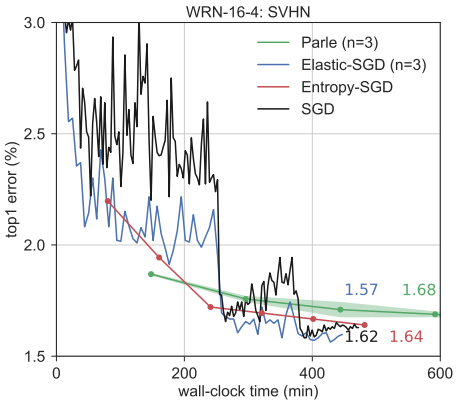


FIGURE 5.5. Validation error of Parle on WRN-16-4 on the SVHN dataset

SVHN is a dataset consisting of house numbers from Google's Street View and contains about 600,000 images of digits. We use the WRN-16-4 network of (Zagoruyko and Komodakis, 2016). We perform global contrast normalization for the input images and do not perform data augmentation. The dropout probability is set to

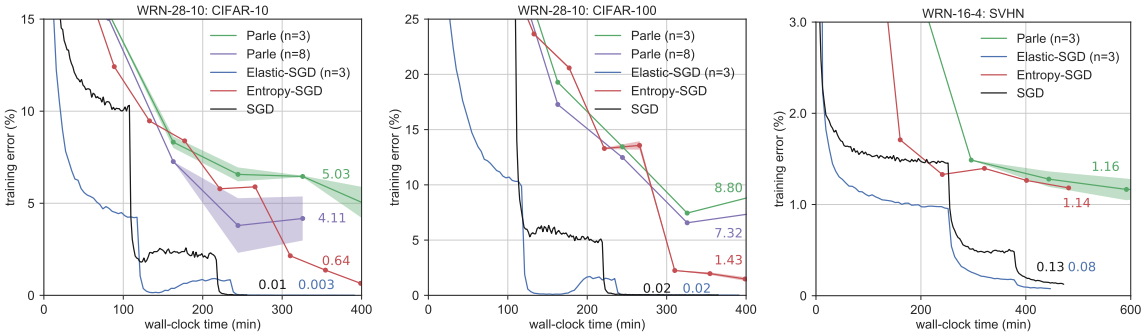
[4] Recent work by Gastaldi (2017) reports an error of 2.8% on CIFAR-10 and 15.85% on CIFAR-100 using "shake-shake" regularization on a three-branch residual network that is trained for 1800 epochs

0.4 and weight decay is set to 5×10^{-4}. The learning rate is initialized to 0.01 and dropped by a factor of 10 at epochs $[80, 120]$ for SGD and epochs $[2, 4]$ for Entropy-SGD and Parle.

Figure 5.5 shows the validation error for SVHN using Parle with three replicas, Entropy-SGD, and SGD. For comparison, the authors in Zagoruyko and Komodakis (2016) report 1.64% error with SGD. All the three algorithms obtain comparable errors in this case, with Elastic-SGD being marginally better. For comparison, the best reported result on SVHN is using a larger network WRN-16-8 which gets 1.54% error, this is very close to our result with Elastic-SGD with scoping. Let us note that Elastic-SGD does not work this well without scoping, we did not get errors below 1.9% on SVHN.

5.4.5. Training error

Let us look at the training error for WRN-28-10 on CIFAR-10 (Figure 5.6a) and CIFAR-100 (Figure 5.6b) and WRN-16-4 on SVHN (Figure 5.6c). Note that while SGD and Elastic-SGD always converge to near-zero training errors, both Entropy-SGD and Parle have much larger training error and do not over-fit as much. The minima discovered by SGD and Parle are qualitatively different: while the former manages to get almost zero training error and converge to the global minimum of the loss function, it ends up over-fitting to the training data and does not generalize as well. Parle obtains superior generalization performance at the cost of under-fitting to the training data. This also sheds light on the structure of the loss function. For the purposes of deep learning, with current regularization techniques, it is not always important to converge to the global optimum. Wide minima, which may exist at higher energy levels instead afford better generalization performance and can be found by algorithms like Entropy-SGD and Parle that are designed to seek them.



(A) CIFAR-10 (B) CIFAR-100 (C) SVHN

FIGURE 5.6. Underfitting of the training loss in Parle: training error on CIFAR-10 (Figure 5.4a), CIFAR-100 (Figure 5.4b) and SVHN (Figure 5.6c)

5.5. SPLITTING THE DATA BETWEEN REPLICAS

In the experiments above, each replica in Parle has access to the entire dataset. Our aim in this section is to explore how much of this improvement can be traded off for speed. If we denote the entire dataset by Ξ, each replica x^a (see Section 5.3) only operates on a subset Ξ^a. We split the dataset evenly amongst the replicas, i.e.,

$$\Xi = \bigcup_{1 \le a \le n} \Xi^a$$

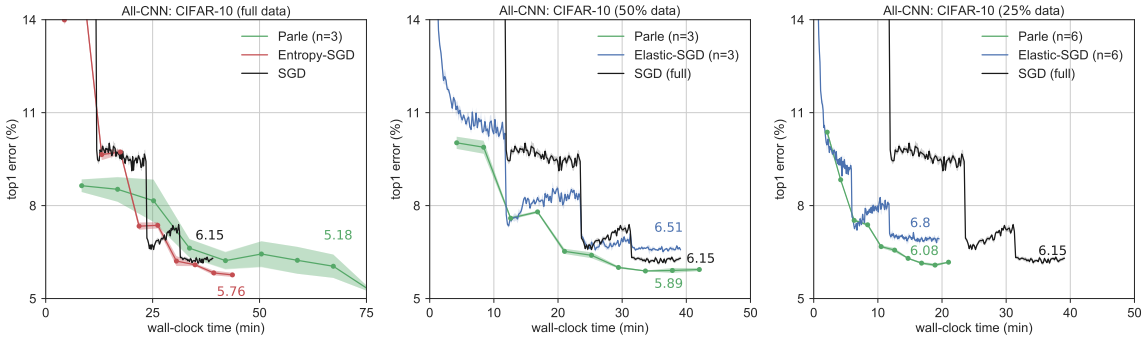
and all Ξ^a are of the same size. In particular, we ensure that each sample lies in at least one of the subsets Ξ^a. By doing so, we would like to explore the efficacy of the proximal term $\frac{1}{2\rho} \|x^a - x\|^2$ in (5.3). Effectively, the only way a replica x^a gets a gradient term on Ξ^b is through this term. We will consider two cases, (i) with $n = 3$ replicas and each gets 50% of the training data, and (ii) with $n = 6$ replicas and each gets 25% of the training data.

We use the allcnn network of Springenberg et al. (2014) for these experiments on CIFAR-10. Again, the hyper-parameters are kept the same as the original authors, in particular, we use a dropout probability of 0.5 with weight decay of 10^{-3} and data augmentation with mirror flips and random crops. Entropy-SGD and SGD are non-distributed algorithms and cannot handle this scenario, we therefore compare Parle with Elastic-SGD (both operate on the same subsets) and a data-parallel SGD on three GPUs with access to the entire dataset. The latter is our baseline and obtains 6.15% validation error. As Figure 5.7b and Figure 5.7c show, quite surprisingly, Parle obtains better error than SGD in spite of the dataset being split between the replicas. The speedup in wall-clock time in Figure 5.7c is a consequence of the fact that Parle has very few mini-batches. Elastic-SGD with split data converges quickly but does not obtain a validation error as good as SGD on the full dataset. For comparison, Parle obtains an error of 5.18% on CIFAR-10 with this network if it has access to the entire dataset in 75 minutes. To our knowledge, this is the best reported error on CIFAR-10 without a residual network, which is itself of importance since allcnn is about $9\times$ faster at test-time than WRN-28-10. In Table 5.2, for comparison, we also report the error of SGD with access to a random subset of the training data (averaged over 3 independent runs); as expected, this is much worse than its error with access to the full dataset.

This experiment shows that the proximal term $\frac{1}{2\rho}\,\|x^a - x\|^2$ is strong enough to pull the replicas towards good regions in the parameter space in spite of their individual gradients being computed on different datasets Ξ^a. If the proximal term is strong enough (as $\rho \to 0$), they can together converge to a region in the parameter space that works for the entire dataset. Exploiting this observation in a data distributed setting (McMahan et al., 2016) to obtain state-of-the-art performance is a promising direction for future work.

Model	Parle		Elastic-SGD		SGD	
	Error	Time	Error	Time	Error	Time
allcnn (full data)	$\mathbf{5.18 \pm 0.06}$	**75**	$5.76^{*} \pm 0.07$	44	6.15 ± 0.05	37
allcnn ($n = 3$, 50% data)	$\mathbf{5.89 \pm 0.01}$	**34**	6.51 ± 0.09	36	$^{*}7.86 \pm 0.12$	20
allcnn ($n = 6$, 25% data)	$\mathbf{6.08 \pm 0.05}$	**19**	6.8 ± 0.05	20	$^{*}10.96 \pm 0.17$	10

TABLE 5.2. Splitting the dataset between replicas on CIFAR-10: Validation error (%) at wall-clock time (min). * SGD performs poorly in these cases because it only has access to a (random) subset of the training data.



(A) 100% data (B) 50% data (C) 25% data

FIGURE 5.7. Validation error of Parle on allcnn on CIFAR-10.

5.6. DISCUSSION

This chapter proposed an algorithm called Parle for training deep neural networks in parallel that exploits the phenomenon of wide regions in the parameter space. Parle requires infrequent communication with the parameter server and instead performs more computation on each client. It scales well to multi-GPU parallel settings and is amenable to a distributed implementation. Our experiments showed that it obtains nearly state-of-the-art performance on benchmark datasets. We obtain significantly better errors than SGD with the same architecture which shows that even with numerous regularization techniques like weight-decay, dropout and batch-normalization, there is still performance left on the table by SGD, which Parle can extract.

In the broader scope of this work, parallelization of non-convex problems like deep neural networks is fundamentally different from convex problems that have been primarily studied in distributed machine learning. Impressive large-scale distributed systems have been built for the specific purposes of deep learning but obtaining a theoretical understanding of popular heuristics used in these systems is hard. Parle is a step towards developing such an understanding, for instance, the loss function used here is a specific way to smooth a rugged, non-convex loss function, similar to the approaches discussed in Chapter 4.

Another interesting offshoot of Parle is that different replicas can have very different computational and communication capabilities. For instance, replicas with GPUs are more suited to run Entropy-SGD while CPU clusters and mobile devices, which can typically communicate quicker than they can compute, are more suited to run Elastic-SGD steps. Coupling these diverse platforms together in a single, interpretable loss function to train a shared model, or multiple coupled models of different sizes, is promising for both scaling up further and learning from private and sensitive data.

CHAPTER 6

Some properties of stochastic gradient descent

The previous chapters construct efficient algorithms that amplify empirically-observed properties of stochastic gradient descent (SGD). However, what makes SGD itself tick? This chapter sheds light upon some its properties, namely implicit regularization and inductive biases, while optimizing high-dimensional energy landscapes.

Our first result is to show precisely in what sense stochastic gradient descent (SGD) implicitly performs variational inference, as is often claimed informally in the literature. For a loss function $f(x)$ with weights $x \in \mathbb{R}^d$, if ρ^{ss} is the steady-state distribution over the weights estimated by SGD,

$$\rho^{\text{ss}} = \underset{\rho \,\in\, \mathscr{L}^2(\Omega)}{\arg\min} \; \underset{X \sim \rho}{\mathbb{E}} \left[\Phi(X) \right] - \frac{\eta}{2\mathscr{b}} \, \mathrm{H}(\rho);$$

where $\mathrm{H}(\rho)$ is the Shannon entropy of the distribution ρ and η and $\mathscr{b}$ are the learning rate and batch-size, respectively. The potential $\Phi(x)$, which we characterize explicitly, is related but not necessarily equal to $f(x)$. It is only a function of the architecture and the dataset. This implies that SGD implicitly performs variational inference with a uniform prior, albeit for a different loss than the one used to compute back-propagation gradients.

We next prove that the implicit potential $\Phi(x)$ is equal to our chosen loss $f(x)$ if and only if the noise in mini-batch gradients is isotropic. This condition, however, is not satisfied for deep networks. Empirically, we find gradient noise to be highly non-isotropic with the rank of its covariance matrix being about 1% of its dimension. Thus, SGD on deep networks implicitly discovers locations where $\nabla\Phi(x) = 0$, these are not the locations where $\nabla f(x) = 0$. This is our second main result: the most likely locations of SGD are not the local minima, nor the saddle points, of the original loss. The deviation of these critical points, which we compute explicitly scales linearly with $\eta/\mathscr{b}$ and is typically large in practice.

When mini-batch noise is non-isotropic, SGD does not even converge in the classical sense. We prove that, instead of undergoing Brownian motion in the vicinity of a critical point, trajectories have a deterministic component that causes SGD to traverse closed loops in the weight space. We detect such loops using a Fourier analysis of SGD trajectories. We also show through an example that SGD with non-isotropic noise can even converge to stable limit cycles around saddle points.

The results in this chapter have been partially published in Chaudhari and Soatto (2017).

6.1. A CONTINUOUS-TIME MODEL OF SGD

Consider the discrete-time updates equations for SGD given by

$$X_{k+1} = X_k - \eta_k \, \nabla f_{\mathscr{b}}(X_k); \quad X_0 = x_0. \tag{6.1}$$

The quantity $\nabla f_{\mathscr{b}}(X_k)$ is a random variable on account of two factors, its argument X_k is a discrete stochastic process while the mini-batch $\mathscr{b}$ is a random subset of the entire dataset. For notational convenience we have overloaded the notation, $\mathscr{b}$ denotes both the mini-batch size and the set of input data that form the mini-batch. Our goal in this chapter is to characterize the solutions of the above update equation. We will study the

stochastic differential equation (SDE) given by

$$\mathrm{d}X = -\nabla f(X)\,\mathrm{d}t + \sqrt{2\beta^{-1}\,D(X)}\,\mathrm{d}B(t); \quad X(0) = x_0 \tag{6.2}$$

where $B(t) : \mathbb{R}_+ \to \mathbb{R}^d$ is Brownian motion, $\beta > 0$ is a scalar and $D(x) \succeq 0$ is a positive semi-definite matrix for all x in the domain. The Euler-Maruyama discretization of (6.2) can be written as

$$X_{k+1} = X_k - \nabla f(X_k)\,\Delta t + (\Delta t)^{1/2}\sqrt{\beta^{-1}D(X_k)}\,Z_k; \quad Z_k \sim N(0, I_{d\times d}) \tag{6.3}$$

where Δt is the time discretization. A comparison of (6.3) and (6.1) suggests that if we take $\Delta t := \eta_k$ and the time parameter of the SDE to be $t := k\eta$, the two equations are similar. The two dynamics are indeed close, not in terms of sample trajectories, but in the sense that the distributions of the trajectories of the two are close. We formalize this notion in Section 6.1.1, we give conditions under which (6.2) approximates (6.1).

6.1.1. Weak convergence

We will assume that each individual $\nabla f_k(x)$, and their average ∇f, are uniformly Lipschitz: for all $x, y \in \Omega$,

$$\big|\nabla f_k(x) - \nabla f_k(y)\big| + \sum_{k\le N} |\nabla f(x) - \nabla f(y)| \le L|x-y|,$$

for some constant $L > 0$. We will additionally assume the following growth condition.

Assumption 6.1 (Gradients grow at most linearly). There exists $M > 0$ such that for all $x \in \Omega$,

$$\big|\nabla f(x)\big| + \sum_{k\le N} |\nabla f_k(x)| \le M(1+|x|)$$

Definition 6.2 (Order-α weak approximation (Kloeden et al., 2012; Milstein, 1994)). Let φ be any function with polynomial growth, i.e., there exists constants K, κ such that $|\varphi(x)| \le K(1+|x|)^\kappa$ for all $x \in \Omega$. The continuous-time SDE (6.2) is an order-α weak approximation of the discrete-time dynamics (6.1) if there exists a constant $c > 0$ independent of η such that for all $k \in \mathbb{N}$

$$\Big|\mathbb{E}[\varphi(x_k)] - \mathbb{E}[\varphi(x(k\eta))]\Big| < c\eta^\alpha.$$

Lemma 6.3 (Diffusion matrix). Let $g_k := \nabla f_k(x)$ and $g := \nabla f(x)$. If mini-batches are sampled with replacement, the variance of mini-batch gradients is

$$\mathrm{var}\Big(\nabla f_{\mathscr{b}}(x)\Big) = \frac{D_{\mathrm{w}}(x)}{\mathscr{b}}$$

where for reasons that will soon become clear, we will call

$$D_{\mathrm{w}}(x) = \left(\frac{1}{N}\sum_{k=1}^N g_k\, g_k^T\right) - g\, g^T \succeq 0 \tag{6.4}$$

the diffusion matrix. Similarly, if mini-batches are sampled without replacement, we have

$$\mathrm{var}\left(\nabla f_{\mathscr{b}}(x)\right) = \frac{D_{\mathrm{wo}}(x)}{\mathscr{b}}\left(1 - \frac{\mathscr{b}}{N}\right)$$

with

$$D_{\mathrm{wo}}(x) = \frac{1}{N-1}\left(\sum_{k=1}^N g_k\, g_k^T\right) - \left(1 - \frac{1}{N-1}\right) g\, g^T \succeq 0. \tag{6.5}$$

Proof. In this computation, although we drop the dependence of g_k on x to keep the notation clear, we emphasize that the diffusion matrix D depends on the weights x.

For mini-batches sampled with replacement, let $i_1, \ldots, i_{\mathscr{b}}$ be $\mathscr{b}$ iid random variables in $\{1, 2, \ldots, N\}$. We would like to compute

$$\text{var}\left(\frac{1}{\mathscr{b}}\sum_{j=1}^{\mathscr{b}} g_{i_j}\right) = \mathop{\mathbb{E}}_{i_1,\ldots,i_{\mathscr{b}}}\left\{\left(\frac{1}{\mathscr{b}}\sum_{j=1}^{\mathscr{b}} g_{i_j} - g\right)\left(\frac{1}{\mathscr{b}}\sum_{j=1}^{\mathscr{b}} g_{i_j} - g\right)^\top\right\}.$$

Note that we have that for any $j \neq k$, the random vectors g_{i_j} and g_{i_k} are independent. We therefore have

$$\text{covar}(g_{i_j}, g_{i_k}) = 0 = \mathop{\mathbb{E}}_{i_j,\, i_k}\left\{(g_{i_j} - g)(g_{i_k} - g)^\top\right\}$$

We use this to obtain

$$\begin{aligned}
\text{var}\left(\frac{1}{\mathscr{b}}\sum_{j=1}^{\mathscr{b}} g_{i_j}\right) = \frac{1}{\mathscr{b}^2}\sum_{j=1}^{\mathscr{b}} \text{var}(g_{i_j}) &= \frac{1}{N\mathscr{b}}\sum_{k=1}^{N}\left((g_k - g)\,(g_k - g)^\top\right) \\
&= \frac{1}{\mathscr{b}}\left(\frac{\sum_{k=1}^N g_k\, g_k^\top}{N} - \frac{\sum_{k=1}^N g_k}{N}\,\frac{\sum_{k=1}^N g_k^\top}{N}\right) \\
&= \frac{1}{\mathscr{b}}\left(\frac{\sum_{k=1}^N g_k\, g_k^\top}{N} - g\, g^\top\right).
\end{aligned}$$

We will set

$$D_{\text{w}}(x) = \frac{1}{N}\left(\sum_{k=1}^{N} g_k\, g_k^\top\right) - g\, g^\top.$$

and assimilate the factor of $\mathscr{b}^{-1}$ in the inverse temperature β.

For mini-batches sampled without replacement, let us define an indicator random variable $\mathbf{1}_{i\in\mathscr{b}}$ that denotes if an example i was sampled in batch $\mathscr{b}$. We can show that

$$\text{var}(\mathbf{1}_{i\in\mathscr{b}}) = \frac{\mathscr{b}}{N} - \frac{\mathscr{b}^2}{N^2},$$

and for $i \neq j$,

$$\text{covar}(\mathbf{1}_{i\in\mathscr{b}}, \mathbf{1}_{j\in\mathscr{b}}) = -\frac{\mathscr{b}(N-\mathscr{b})}{N^2(N-1)}.$$

Similar to Li et al. (2017a), we can now compute

$$\begin{aligned}
\text{var}\left(\frac{1}{\mathscr{b}}\sum_{k=1}^{N} g_k\,\mathbf{1}_{k\in\mathscr{b}}\right) &= \frac{1}{\mathscr{b}^2}\,\text{var}\left(\sum_{k=1}^{N} g_k\,\mathbf{1}_{k\in\mathscr{b}}\right) \\
&= \frac{1}{\mathscr{b}^2}\sum_{k=1}^{N} g_k\, g_k^\top\,\text{var}(\mathbf{1}_{k\in\mathscr{b}}) + \frac{1}{\mathscr{b}^2}\sum_{i,j=1,\, i\neq j}^{N} g_i\, g_j^\top\,\text{covar}(\mathbf{1}_{i\in\mathscr{b}}, \mathbf{1}_{j\in\mathscr{b}}) \\
&= \frac{1}{\mathscr{b}}\left(1 - \frac{\mathscr{b}}{N}\right)\left[\frac{\sum_{k=1}^N g_k\, g_k^\top}{N-1} - \left(1 - \frac{1}{N-1}\right) g\, g^\top\right].
\end{aligned}$$

We will again set

$$D(x) = \frac{1}{N-1}\left(\sum_{k=1}^{N} g_k\, g_k^\top\right) - \left(1 - \frac{1}{N-1}\right) g\, g^\top$$

and assimilate the factor of $\mathscr{b}^{-1}\left(1 - \frac{\mathscr{b}}{N}\right)$ that depends on the batch-size in the inverse temperature β.

■

Remark 6.4 ($D(x)$ does not depend on the optimization algorithm). Note that $D_{\text{w}}(x)$ and $D_{\text{wo}}(x)$ are independent of the learning rate η and the batch-size $\mathscr{b}$. They only depend on the weights x, architecture and the dataset. In particular, the diffusion matrix does not depend on the optimization algorithm.

Theorem 6.5 (Order-1 weak approximation of SGD, (Li et al., 2017c)). For any $T > 0$ and $D(x) : \Omega \to \mathbb{R}^{d\times d}$, if the gradients $\nabla f_k(x)$ are uniformly Lipschitz and grow at most linearly Assumption 6.1, the stochastic process (6.2)

$$\mathrm{d}X = -\nabla f(X)\,\mathrm{d}t + \sqrt{2\beta^{-1}\,D(X)}\,\mathrm{d}B(t); \quad X(0) = x_0$$

interpreted in the Itô sense (Oksendal, 2003) is an order-1 weak approximation of (6.1). For mini-batches sampled with replacement, $D(X)$ is given by (6.4) and

$$\beta^{-1} := \beta_{\mathrm{w}}^{-1} = \frac{\eta}{2\mathscr{b}}; \tag{6.6}$$

while for mini-batches sampled without replacement, the diffusion matrix is given by (6.5) with

$$\beta_{\mathrm{wo}}^{-1} = \frac{\eta}{2\mathscr{b}}\left(1 - \frac{\beta}{N}\right). \tag{6.7}$$

Remark 6.6 (β^{-1} completely captures the effect of optimization). The two hyper-parameters of optimization, the learning rate η and the mini-batch size $\mathscr{b}$ are completely captured by β^{-1}. In particular, the other terms $\nabla f(X)$ and $D(X)$ do not depend upon the specific optimization algorithm used, they are only functions of the dataset, architecture and the weights X.

6.1.2. Fokker-Planck equation

The Fokker-Planck (FPK) equation (Risken, 1996) is a partial differential equation (PDE) that governs the flow of probability densities of stochastic processes such as (6.2). We first define some notation to make our presentation of the FPK compact.

Definition 6.7 (Divergence operator). For a vector field $v(x) : \Omega \to \Omega$, the divergence operator is defined to be

$$\nabla \cdot v(x) = \sum_{k=1}^{d} \frac{\partial v_k}{\partial x_k}$$

where $x = \begin{bmatrix} x_1, x_2, \ldots, x_d \end{bmatrix}^\top$ and $v(x) = \begin{bmatrix} v_1, v_2, \ldots, v_d \end{bmatrix}^\top$. We will overload this operator to also operate on matrices, in which case it returns a vector. The divergence operator on a matrix $M(x) : \Omega \to \Omega \times \Omega$ is defined to be

$$\nabla \cdot M(x) = \begin{bmatrix} \nabla \cdot M_1(x), \nabla \cdot M_2(x), \ldots, \nabla \cdot M_d(x) \end{bmatrix}^\top;$$

where M_k is the k^{th} column of M. In other words, the vector operator is applied column-wise when it operates on a matrix. Note that the $\nabla\cdot$ operator returns a scalar if it operates on a vector and returns a tall vector $\in \Omega$ if it operates a matrix.

We will denote probability densities by $\rho(x) : \Omega \to \mathbb{R}$ and assume that they are square integrable, i.e., $\rho \in \mathscr{L}^2(\Omega)$. Let $\rho(0) = \rho_0 = \delta(x_0)$ which is a Dirac-delta distribution at x_0. The Fokker-Planck equation governs the flow of probability density of $x(t)$, i.e., at each time t, $X(t)$ is sampled from the density $\rho(t)$:

$$X(t) \sim \rho(t);$$

where $\rho(x,t)$ for (6.2) evolves as

$$\frac{\partial \rho}{\partial t} = \nabla \cdot \Big(\nabla f\ \rho + \beta^{-1}\ \nabla \cdot \big(D\ \rho\big)\Big); \quad \rho(0) = \delta(x_0) \tag{6.8}$$

This equation can also be expanded out to look like:

$$\frac{\partial \rho}{\partial t} = \sum_{i=1}^{d} \frac{\partial}{\partial x_i}\Big(\nabla f(x)\rho\Big)_i + \beta^{-1} \sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i x_j}\Big(D_{ij}(x)\ \rho\Big)$$

where $(\cdot)_i$ is the i^{th} element of a vector and $(\cdot)_{ij}$ is the entry at the i^{th} row and j^{th} column of a matrix. The general form of the Fokker-Planck equation is given by

$$\rho_t = \sum_{n=1}^{\infty} \frac{1}{n!} \nabla^{(n)} \left\{ A^{(n)}(x)\, \rho(x) \right\}$$

with drift, diffusion and similar higher-order coefficients denoted by $A^{(k)}$. If the stochastic dynamics as given by (6.2) is a drift term and Brownian motion, it can be shown that all coefficients with $k \geq 3$ are zero and the correct Fokker-Planck equation is (6.8) (Risken, 1996).

Lemma 6.8 (FPK maintains normalization). If the initial condition of FPK is such that $\int_{x\in\Omega} \rho_0(x)\, \mathrm{d}x = 1$, the FPK preserves normalization; the solution has measure unity at any time $t > 0$ as well

$$\int_{x\in\Omega} \rho(t)(x)\, \mathrm{d}x = 1.$$

Proof. We will show that under suitable boundary conditions $\frac{\partial}{\partial t} \int \rho(t)\, dx = 0$.

$$\begin{aligned} \int_\Omega \frac{\partial \rho}{\partial t}\, \mathrm{d}x &= \int_\Omega \nabla \cdot \Big(\nabla f \rho + \beta^{-1}\, \nabla \cdot \big(D\, \rho \big) \Big)\, \mathrm{d}x \\ &= 0. \end{aligned}$$

note that this is divergence of a vector field over Ω and is thus zero by the divergence theorem (also known as Gauss theorem) if no mass flows out normal to the boundary $\partial\Omega$. ■

We can thus think of the density $\rho(t)$ and the probability distribution of $x(t)$ interchangeably and will consider so in the narrative.

Remark 6.9 (The correct form of the Fokker-Planck equation). Depending upon the interpretation of the stochastic process (6.2), one may have different PDEs that govern the flow of probability densities. The interpretation becomes particularly important for state-depending noise $D(X)$. For instance, the Itô interpretation, which we will work with primarily, uses the discretization

$$D^{1/2}(X(t))\, \mathrm{d}B(t) := \lim_{\Delta t \to 0} D^{1/2}\Big(X\big(\alpha t + (1-\alpha)(t+\Delta t)\big)\Big)\, \Big(B(t+\Delta t) - B(t) \Big).$$

Setting $\alpha = 0$ corresponds to the Itô interpretation which leads to the FPK as given by (6.8), setting $\alpha = 1/2$ corresponds to the Stratonovich interpretation of the SDE, whereas $\alpha = 1$ corresponds to the Fick's law (Fick, 1855) or the Klimontovich-Hänggi (Klimontovich, 1990; Hanggi, 1978) version of the Fokker-Planck equation which is given by

$$\rho_t = \nabla \cdot \Big(\nabla f\, \rho + \beta^{-1}\, D\, \nabla \rho \Big). \tag{6.9}$$

Note that the above equation is easier to manipulate because the diffusion matrix comes out of the divergence operator and this is the form popularly employed in the physics and biology literatures. However, while a scientist is satisfied with modeling a physical process as PDE in either form to be able to analyze its behavior, we have an underlying stochastic process, viz., SGD given by (6.1), that we want our results to be faithful to. The stochasticity of SGD is a function of the current location $x(t)$ through the mini-batch gradient $\nabla f_{\acute{b}}(x)$. This necessitates the Itô interpretation of the SDE (6.2) with $\alpha = 0$. We can however conveniently switch across different stochastic calculi using the fact that the corresponding PDE should have the same invariant distribution, a fact we will exploit further in this chapter. See the thesis of Heidernätsch (2014) for a more elaborate discussion on state-dependent diffusion.

We are interested in understanding the solutions of the Fokker-Planck equation and make the following assumption.

Assumption 6.10 (Steady-state solution exists and is unique). We assume that the steady-state solution of the Fokker-Planck equation (6.8) exists and is unique irrespective of the initial condition of the SDE x_0. We

denote this solution by $\rho^{\text{ss}}(x)$ and it satisfies

$$0 = \frac{\partial \rho^{\text{ss}}}{\partial t} = \nabla \cdot \Big(\nabla f \, \rho^{\text{ss}} + \beta^{-1} \, \nabla \cdot \big(D \, \rho^{\text{ss}} \big) \Big). \tag{6.10}$$

This is an important assumption and it says that the Fokker-Planck equation evolves on a convex subspace of probability distributions $\mathscr{P}(\Omega)$ under a metric specific to our problem to be defined later; the steady-state ρ^{ss} is the unique fixed-point of the FPK dynamics in this subspace. Note that this does not implicitly assume that the loss function $f(x)$ is convex.

Definition 6.11 (Fokker-Planck operator). The Fokker-Planck operator is a linear operator $L : \mathscr{L}^2(\Omega) \to \mathbb{R}$ defined as

$$L\rho := \nabla \cdot \Big(\nabla f \, \rho + \beta^{-1} \, \nabla \cdot \big(D \, \rho \big) \Big)$$

whereby (6.8) can be (even more) succinctly written as

$$\rho_t = L\rho.$$

The steady-state solution ρ^{ss} which now satisfies $L\rho^{\text{ss}} = 0$ and therefore lies in the kernel of the operator L. Assumption 6.10 implies that the kernel of L is non-empty and a singleton.

6.1.3. Probability current

This section discusses the different kind of solutions that the Fokker-Planck equation may have. We first define a quantity called the probability current.

Definition 6.12 (Probability current). The quantity

$$-J(x,t) = \Big(\nabla f(x) \, \rho(x,t) + \beta^{-1} \, \nabla \cdot \big(D(x) \, \rho(x,t) \big) \Big) \tag{6.11}$$

is called the probability current. This helps us to write the PDE (6.8) in (yet) another form:

$$\rho_t = -\nabla \cdot J(x,t),$$

which suggests that $J(x,t)$ is the flow of probability mass in space and time. If ρ^{ss} is the steady-state solution of (6.8), we have

$$\begin{aligned} 0 &= \nabla \cdot \Big(\nabla f(x) \, \rho^{\text{ss}} + \beta^{-1} \, \nabla \cdot \big(D(x) \, \rho^{\text{ss}} \big) \Big) \\ &\triangleq \nabla \cdot J^{\text{ss}}(x); \end{aligned} \tag{6.12}$$

for all $x \in \Omega$.

The condition (6.12) can be achieved in two different ways as defined below.

Definition 6.13 (Detailed balance). The probability current can be zero at steady-state for all $x \in \Omega$:

$$J^{\text{ss}}(x) = 0; \quad \text{for all } x \in \Omega. \tag{6.13}$$

This condition is called the "detailed balance" condition for Markov chains. For instance, in the discrete case it implies that at steady-state for any two states x and y the ratio of the transition probabilities $c_{x\to y}$ and $c_{y\to x}$ is the inverse ratio of their respective probability masses

$$\rho^{\text{ss}}(x) \; c_{x\to y} = \rho^{\text{ss}}(y) \; c_{y\to x}.$$

In other words, any trajectory $x \to y \to z$ of a Markov chain with detailed balance can be reversed in time without changing its likelihood.

Definition 6.14 (Broken detailed balance). This is the case where

$$\nabla \cdot J^{\text{ss}}(x) = 0 \text{ for all } x \in \Omega \quad \text{but} \quad J^{\text{ss}}(x) \neq 0 \text{ for all } x \in \Omega. \tag{6.14}$$

This condition is called "broken detailed balance" and the corresponding Markov chain is irreversible. In the discrete case, this gives the following probability mass flow constraint:

$$\rho^{\text{ss}}(x) \sum_{y \neq x} c_{x\to y} = \sum_{x \neq y} \rho^{\text{ss}}(y) \; c_{y\to x};$$

the total probability mass flowing out from a x to states y is matched by the total mass from all states y into the state x. Broken detailed balance is a more general condition than detailed balance and any Markov chain which has a steady-state distribution satisfies the above mass flow constraint.

6.2. VARIATIONAL FORMULATION

The theory of PDEs has powerful connections to variational analysis. This is a link that was discovered in the optimal transport literature by Jordan, Kinderlehrer and Otto (Jordan et al., 1997; Otto, 2001) and further refined in the works of Ambrosio, Gigli and Savaré (Ambrosio et al., 2008); the same flavor of results however date back to the H-theorem in Boltzmann (1872). We will elaborate upon these links in the present section and arrive at the first major result of the present chapter, viz., a variational formulation of stochastic gradient descent which emphasizes its inductive bias and implicit regularization properties.

6.2.1. Wasserstein metric

This section contains a brief introduction to the theory of optimal transport, our aim is to give a visual intuition to the results of this chapter without delving into optimal transport and gradient flows in metric spaces in their full mathematical sophistication. For a more elaborate treatment, the interested reader is referred to the numerous exemplary works in this field, for instance, in increasing order of required mathematical maturity: Santambrogio (2015); Ambrosio et al. (2008) and Villani (2008).

The problem of optimal transport was first posed by Monge (1781): given two probability densities $p, q \in \mathscr{L}^2(\Omega)$, find a map $T : \Omega \to \Omega$ pushing the first one onto the other, i.e.,

$$\int_A q(x)\,\mathrm{d}x = \int_{T^{-1}(A)} f(y)\,\mathrm{d}y \quad \text{for any Borel set } A \in \mathscr{B}(\Omega)$$

that minimizes the quantity

$$\int_\Omega |T(x) - x|\; p(x)\,\mathrm{d}x.$$

In this case, we say that T is the push-forward of p to q and write it as

$$q = T_\#(p).$$

As defined above, the Monge problem stayed without solutions till 1940s until the work of Kantorovich (Kantorovich, 1942) who relaxed the problem as finding a "transport plan" $\gamma \in \Pi(p,q)$ where $\Pi(p,q)$ is the set of joint probability distributions such that the first marginal is p and the second marginal is q. The Kantorovich problem is given by

$$\min\Big\{ \int_{\Omega\times\Omega} c(x,y)\,\mathrm{d}\gamma \;:\; \gamma \in \Pi(p,q) \Big\}. \tag{6.15}$$

The intuitive idea is that instead of considering a single partial x that moves to $T(x)$ in the Monge problem, we consider two locations (x,y) and measure how many particles move from x to y. This formulation is more general than the Monge problem, for instance, while the map T may not exist in the Monge problem, the transport plan always has a candidate solution, the product distribution $p \otimes q \in \Pi(p,q)$. Moreover, if $(id,T)_\# p \in \Pi(p,q)$, then T is also a solution of the Monge problem, i.e., $q = T_\# p$. It can be shown (Brenier, 1987, 1991) that if the measures p, q are absolutely continuous with respect to the Lebesgue measure on Ω and if $c(x,y) = \frac{1}{2}|x-y|^2$ in (6.15), the Kantorovich optimal transport is unique and of the form $(id,T)_\# p$ and there exists a potential $\varphi(x)$ called the Kantorovich potential such that the transport map is the gradient of a convex function: $T(x) = \nabla\left(\frac{x^2}{2} - \varphi(x)\right)$. The converse of this theorem is extremely interesting: the gradient of *any* convex function is the optimal transport between any distribution and its image under the transport.

Definition 6.15 (Wasserstein metric). The Wasserstein metric between two distributions p, q is defined to be

$$W_2^2(p,q) = \min\; \Big\{ \int_{\Omega\times\Omega} |x-y|^2\,\mathrm{d}\gamma \;:\; \gamma \in \Pi(p,q) \Big\}. \tag{6.16}$$

In other words, the Wasserstein metric is the optimal cost of the Kantorovich optimal transport problem with quadratic cost. It is an important quantity in many fields because it gives a natural way to define distances between measures and is in fact a metric on $\mathscr{P}(\Omega)$. Moreover, convergence in the Wasserstein metric is equivalent to weak convergence of probability measures.

The connection of PDEs with optimal transport is motivated from the following observation: an absolutely continuous curve in the space $\mathbb{W}_2(\Omega) = (\mathscr{P}(\Omega), W_2)$ can be thought of as the solution of the continuity equation $\rho_t + \nabla \cdot (v\,\rho)$ where $v \in L^2(\Omega)$ is a vector field (Ambrosio et al., 2008). In fact, the metric space $\mathbb{W}_2(\Omega)$ is a geodesic: for any two probability measures $\rho(0), \rho(1) \in \mathscr{P}(\Omega)$, there exists a path $(\rho(t))_{t\in[0,1]}$ such that for any $0 \le s \le t \le 1$,

$$W_2\big(\rho(s), \rho(t)\big) = (t-s)\ W_2\Big(\rho(0), \rho(1)\Big).$$

This path is a constant-speed geodesic in $\mathbb{W}_2$ and is given by

$$\rho(t) = (T_t)_{\#}\ p \triangleq \Big((1-t)id + tT\Big)_{\#}\ p$$

where T is the Kantorovich optimal transport between p and q. This is known as McCann displacement interpolation in $\mathbb{W}_2$ (Villani, 2008, Chap. 5). The consequence is that looking for the optimal transport and constant-speed geodesics are equivalent: from the optimal plan, one can get geodesics using $v = (T - id) \circ T_t^{-1}$ and geodesics help reconstruct the optimal transport through their velocity field. The celebrated formulation by Benamou and Brenier (2000) solves for such constant-speed geodesics:

$$\min\ \Big\{ \int_0^1 \int_\Omega |v|^2\ \mathrm{d}\rho\ \mathrm{d}t\ \ \rho_t + \nabla \cdot (v\,r) = 0,\ \rho(0) = p,\ \rho(1) = q \Big\}. \tag{6.17}$$

6.2.2. Gradient flows

The results above suggest a natural framework to connect gradient flows in the space $\mathbb{W}_2(\Omega)$ with PDEs of the form of the continuity equation. Consider a function $F(\rho)$ and an iterated minimization problem of the form:

$$\rho_{k+1}^\tau\ \in\ \arg\min_\rho\ \left\{ F(\rho) + \frac{1}{2\tau}\ W_2^2(\rho, \rho_k^\tau) \right\}. \tag{6.18}$$

Just as the gradient gradient flow $\dot{x} = -\nabla F(x)$ corresponds to an iterated minimization problem

$$\min_x\ \left\{ F(x) + \frac{1}{2\tau}\ |x - x_k^\tau|^2 \right\}$$

in the Euclidean metric, we will see that certain PDEs correspond to (6.18) in the Wasserstein metric. The underlying Kantorovich problem is given by

$$\mathscr{T}_{|\cdot|^2}\ (\rho, \rho_k^\tau) := \min\ \left\{ \int\ |x-y|^2\ d\gamma\ :\ \gamma \in \Pi(\rho, \rho_k^\tau) \right\}.$$

Definition 6.16 (First variation of a functional). Given a functional $G : \mathscr{L}^2(\Omega) \to \mathbb{R}$ the first variation of the functional G at ρ denoted by $\frac{\delta G}{\delta \rho}(\rho)$ is the unique function such that

$$\frac{\mathrm{d}}{\mathrm{d}\,\varepsilon} G(\rho + \varepsilon\chi)_{|\varepsilon\to 0} = \int\ \frac{\delta G}{\delta \rho}(\rho)\ \mathrm{d}\chi$$

for every perturbation χ which $\int_\Omega \chi \mathrm{d}x = 0$. We can compute first variations for some functionals of interest: for

$$F(\rho) = \int_\Omega f(\rho(x))\ \mathrm{d}x, \quad V(r) = \int_\Omega V(x)\ \mathrm{d}\rho, \quad W(\rho) = \frac{1}{2}\int_\Omega \int_\Omega W(x-y)\ \mathrm{d}\rho(x)\mathrm{d}\rho(y),$$

we have

$$\frac{\delta F}{\delta \rho}(\rho) = f'(\rho), \quad \frac{\delta V}{\delta \rho}(\rho) = V, \quad \frac{\delta W}{\delta \rho}(\rho) = W * \rho$$

where $*$ denotes the convolution.

In particular the first variation of $\mathscr{T}_{|\cdot|^2}(\rho, \rho_k^\tau)$ is the Kantorovich potential for the transport map from ρ to ρ_k^τ:

$$\frac{\delta \mathscr{T}_{|\cdot|^2}}{\delta \rho}(\rho) = \varphi(\rho, \rho_k^\tau).$$

The optimality conditions for problem (6.18) now give

$$\frac{\delta F}{\delta \rho}(\rho_{k+1}^\tau) + \frac{\varphi\left(\rho_{k+1}^\tau, \rho_k^\tau\right)}{\tau} = \text{constant}.$$

Since we have $T(x) = x - \nabla\varphi(x)$ for the optimal map T, we set

$$-v(x) := \frac{T(x) - x}{\tau} = -\frac{\nabla\varphi(x)}{\tau} = \nabla\left(\frac{\delta F}{\delta \rho}(\rho)\right)(x).$$

The velocity field v is therefore seen as the backward velocity that moves from ρ_{k+1}^τ to ρ_k^τ. Plugging this into the continuity equation gives, as $\tau \to 0$,

$$\frac{\partial \rho}{\partial t} - \nabla \cdot \left(\rho \, \nabla\left(\frac{\delta F}{\delta \rho}(\rho)\right)\right) = 0. \tag{6.19}$$

We next instantiate this PDE to make a few concrete remarks.

Remark 6.17 (Heat flow maximizes Shannon entropy in the Wasserstein metric). If we set

$$F(\rho) = \int f(\rho(x)) \, \mathrm{d}x := \int \rho \log \rho \, \mathrm{d}x$$

to be the negative Shannon entropy, we have from the calculations in Definition 6.16 that $f(t) = t\log t$, $f'(t) = \log t + 1$ and $\nabla(f'(\rho)) = \frac{\nabla\rho}{\rho}$: in other words, the gradient flow associated with the Shannon entropy is the heat equation

$$\rho_t - \nabla \cdot \nabla\rho = 0.$$

On the other hand, the trajectories of the heat equation perform steepest descent in Euclidean metric on the Dirichlet energy functional

$$\frac{1}{2} \int_\Omega |\nabla\rho|^2 \, \mathrm{d}x.$$

This equivalence is also quite natural, the heat equation describes the evolution of the probability density of pure Brownian motion: $\mathrm{d}x = \sqrt{2} \, \mathrm{d}B(t)$. The Wasserstein point-of-view suggests that Brownian motion maximizes the entropy of its state distribution, while the Dirichlet functional suggests that it minimizes the total-variation of its density.

Remark 6.18 (Fokker-Planck equation and the Jordan-Kinderlehrer-Otto (JKO) functional). If $F(\rho)$ is the Jordan-Kinderlehrer-Otto (JKO) functional (Jordan et al., 1997)

$$\begin{aligned} F(\rho) &= \int V(x) \, \mathrm{d}\rho(x) + \int \mathrm{d}\rho \log \rho \\ &= \mathop{\mathbb{E}}_{X \sim \rho}\Big[V(X)\Big] - \mathrm{H}(\rho), \end{aligned} \tag{6.20}$$

where $\mathrm{H}(\rho) = -\int \mathrm{d}\rho \log\rho$ is the Shannon entropy, the Fokker-Planck (FPK) equation given by

$$\rho_t = \nabla \cdot (\rho \, \nabla V) + \Delta \, \rho$$

performs steepest descent on $F(\rho)$. The SDE corresponding to this FPK is

$$\mathrm{d}X = -\nabla V(X) \, \mathrm{d}t + \sqrt{2} \, \mathrm{d}B(t).$$

Note the special form of the JKO functional. It is a sum of two terms: the first is the average of the potential $V(x)$ giving the drift field in FPK and its SDE, and second is the Shannon entropy of the density ρ that corresponds to the Laplacian term in FPK. This result will be very useful for us to study the dynamics of stochastic gradient descent and this split between an energetic term and an entropic terms will also be seen in other results in this chapter; we will draw the reader's attention back to the JKO functional in such cases.

Remark 6.19 (Wasserstein gradient flow for isotropic noise in SGD). If the diffusion matrix is $D(x) = I_{d\times d}$ in the continuous-time approximation of SGD given by (6.2)

$$\mathrm{d}X = -\nabla f(X)\,\mathrm{d}t + \sqrt{2\beta^{-1}}\,\mathrm{d}B(t),$$

Remark 6.18 above shows that the probability density $\rho(t)$ evolves using the Fokker-Planck equation and minimizes

$$F(\rho) = \mathop{\mathbb{E}}_{X\sim\rho}\Big[f(X)\Big] - \beta^{-1}\mathrm{H}(\rho).$$

In other words, the variational problem underlying SGD corresponds to the JKO functional. In this case, the minimizer of the free-energy $F(\rho)$ is

$$\begin{aligned}\rho^* &= \arg\min_{\rho}\ F(\rho)\\ &\propto e^{-\beta f(x)}\end{aligned}$$

as can be seen by taking the first variation. In other words, is the noise of the stochastic updates is isotropic, the steady-state distribution of SGD iterates concentrates on the local minimizers of $f(x)$; as $\beta\to\infty$ it concentrates on the global minimizers.

6.2.3. Non-isotropic gradient noise

The preceding sections laid down a framework for understanding the dynamics of SGD in terms of its continuous-time approximation and the corresponding PDE as steepest descent in the Wasserstein metric of a functional of the probability density. Effectively, we have connected functionals of the form

$$F(\rho) = \mathop{\mathbb{E}}_{X\sim\rho}\Big[f(X)\Big] - \beta^{-1}\mathrm{H}(\rho)$$

with the stochastic process

$$\mathrm{d}X = -\nabla f(X)\,\mathrm{d}t + \sqrt{2\beta^{-1}}\,\mathrm{d}B(t).$$

This section discusses the general case, what are the corresponding functionals when the diffusion matrix

$$D(x) \neq I_{d\times d}$$

for all $x\in\Omega$.

Remark 6.20 (A special SDE). Let us begin with computing the solution of a special kind. The SDE given by

$$\mathrm{d}X = -D(X)\,\nabla\Phi(X)\,\mathrm{d}t + \beta^{-1}\,\nabla\cdot D(X)\,\mathrm{d}t + \sqrt{2\beta^{-1}\,D(X)}\,\mathrm{d}B(t) \tag{6.21}$$

has the Fokker-Planck equation

$$\rho_t = \nabla\cdot\Big(D\,\nabla\Phi\,\rho + \beta^{-1}\,\nabla\cdot(D\,\rho)\Big).$$

It can be seen by a direct computation that

$$\rho^{\mathrm{eq}}(x) = \frac{1}{Z(\beta)}\,e^{-\beta\Phi(x)} \tag{6.22}$$

is the steady-state solution of this Fokker-Planck equation. The normalization constant $Z(\beta) = \int e^{-\beta\Phi(x)}\,\mathrm{d}x$ is the partition function and is assumed to be well-defined. In other words, if we could write $\nabla f(x)$ in terms of the gradient of some function $\Phi(x)$ scaled by the diffusion matrix $D(x)$, we can compute the steady-state distribution of the SGD easily, it is simply $\rho^{\mathrm{eq}}\propto e^{-\beta\Phi(x)}$.

We will use the above development in Remark 6.20 as follows. We have assumed the existence of a unique steady-state solution to the general Fokker-Planck equation (6.8), this is denoted by $\rho^{\mathrm{ss}}(x)$. We can therefore define a function $\Phi(x)$ up to a constant as

$$\Phi(x) = \beta^{-1}\log\rho^{\mathrm{ss}}(x); \tag{6.23}$$

this implicitly assumes that the steady-state can be written as $\rho^{\text{ss}} \propto e^{-\beta\Phi(x)}$ which can always be done under regularity conditions (Doob, 2012). We have implicitly defined $\Phi(x)$ in terms of the steady-state solution $\rho^{\text{ss}}(x)$ in this section; we will prove in the sequel that it is only a function of the dataset, architecture and the weights x and is in fact independent of β^{-1}. In other words, the steady-state potential $\Phi(x)$ is an intrinsic property of the dynamics of SGD and does not depend on the mini-batch size b or the learning rate η.

We now split the vector field $\nabla f(x)$ into two parts, one that comes from the gradient of a potential $\nabla\Phi(x)$ and the second that is a left-over vector field. This definition is motivated from the Helmholtz theorem or the Hodge decomposition which states that any vector field on some bounded domain can be split into a curl-free component and a divergence-free component. For unbounded domains, such as the weights of a neural network, the Helmholtz theorem is true for vector fields that decay fast, say with $1/\|x\|$. However, in view of Assumption 6.1, we will not make such an assumption and instead provide a thermodynamical argument towards the same end.

Definition 6.21 (Remainder force $j(x)$). Given a steady-state potential Φ, the force $j(x)$ is defined be

$$j(x) = -\nabla f(x) + D(x)\,\nabla\Phi(x) - \beta^{-1}\,\nabla\cdot D(x). \tag{6.24}$$

The term $D\,\nabla\Phi$ is curl-free because it is the gradient of a potential Φ. It is natural to think of $j(x)$ as part of the force ∇f that cannot be written as $G\,\nabla\Phi$ for some function Φ and a positive semi-definite matrix G.

We now ask the following question: under what conditions does the steady-state solution of the SDE still remain $\rho^{\text{ss}}(x)$? Roughly speaking, Remark 6.20 shows that if the right hand-side of (6.24) is zero, i.e., if the force $j(x)$ is zero, the steady-state solution is ρ^{ss}. The solution ρ^{ss} also remains unchanged if $j(x)$ is a divergence-free vector field and we make an assumption to that effect.

Assumption 6.22 (The force $j(x)$ is conservative). The force $j(x)$ as defined in Definition 6.21 is conservative, i.e., it is divergence-free

$$\nabla\cdot j(x) = 0; \tag{6.25}$$

for all $x \in \Omega$.

This is a crucial assumption. It will help us prove that the steady-state solution of the general SDE (6.2) is $\rho^{\text{ss}} \propto e^{-\beta\Phi}$. We will however provide two arguments about the legitimacy of such an assumption: one where we discuss the properties of the force $j(x)$ that show that it is a natural generalization of gradient flows discussed in Section 6.2.2, and second where we allude to the second law of thermodynamics under the interpretation that the Fokker-Planck equation represents a physical system that exchanges energy with the environment.

Lemma 6.23 (Properties of $j(x)$). Under Assumption 6.22,

(i) $j(x)$ is non-zero if and only if detailed balance is broken, i.e., $J^{\text{ss}}(x) \neq 0$, the steady-state probability current J^{ss} and potential ρ^{ss} satisfy

$$j(x) = \frac{J^{\text{ss}}(x)}{\rho^{\text{ss}}(x)}, \quad \text{and}$$

(ii) $j(x)$ is orthogonal to the potential Φ:

$$j(x)\,\nabla\Phi = 0.$$

Proof. The Fokker-Planck equation (6.8) can be written in terms of the probability current as

$$\begin{aligned} 0 = \rho_t^{\text{ss}} &= \nabla\cdot\left(-j\,\rho^{\text{ss}} + D\,\nabla\Phi\,\rho^{\text{ss}} - \beta^{-1}(\nabla\cdot D)\,\rho^{\text{ss}} + \beta^{-1}\nabla\cdot(D\rho^{\text{ss}})\right) \\ &= \nabla\cdot\,J^{\text{ss}}. \end{aligned}$$

Since we have $\rho^{\text{ss}} \propto e^{-\beta\Phi(x)}$, from the observation in Remark 6.20, we also have that

$$0 = \rho_t^{\text{ss}} = \nabla\cdot\left(D\,\nabla\Phi\,\rho^{\text{ss}} + \beta^{-1}D\,\nabla\rho^{\text{ss}}\right),$$

and consequently,

$$\begin{aligned} 0 &= \nabla \cdot (j\, \rho^{\text{ss}}) \\ \Rightarrow \quad j(x) &= \frac{J^{\text{ss}}}{\rho^{\text{ss}}}. \end{aligned} \tag{6.26}$$

In other words, the conservative force is non-zero only if detailed balance is broken, i.e., $J^{\text{ss}} \neq 0$. We also have

$$\begin{aligned} 0 &= \nabla \cdot (j\, \rho^{\text{ss}}) \\ &= \rho^{\text{ss}}\, (\nabla \cdot j \;-\; j \cdot \nabla \Phi), \end{aligned}$$

which shows using Assumption 6.22 and $\rho^{\text{ss}}(x) > 0$ for all $x \in \Omega$ that $j(x)$ is always orthogonal to the gradient of the potential

$$\begin{aligned} 0 &= j(x) \cdot \nabla \Phi(x) \\ &= j(x) \cdot \nabla \rho^{\text{ss}}. \end{aligned} \tag{6.27}$$

Using the definition of $j(x)$ in (6.24), we have detailed balance when

$$\nabla f(x) = D(x)\, \nabla \Phi(x) - \beta^{-1} \nabla \cdot D(x). \tag{6.28}$$

■

We now present the main result of this section.

Theorem 6.24 (SGD performs variational inference). The free-energy functional

$$F(\rho) = \beta^{-1}\, \mathrm{KL}\big(\rho \;||\; \rho^{\text{ss}}\big) \tag{6.29}$$

decreases monotonically along the trajectories of the Fokker-Planck equation (6.8) and converges to its minimum, which is zero, at steady-state. Moreover, we also have an energetic-entropic split

$$F(\rho) = \mathop{\mathbb{E}}_{x \in \rho} \Big[\Phi(x)\Big] - \beta^{-1} \mathrm{H}(\rho) + \text{constant}. \tag{6.30}$$

Before presenting the proof of the above theorem, we will collect a few auxiliary results that govern the free-energy functional (6.29) and Assumption 6.22. Our development starts by considering the Fokker-Planck equation (6.8) as a physical system which exchanges energy with an external environment (Ottinger, 2005; Qian, 2014). In our case, this physical system is the gradient dynamics $\nabla \cdot (\nabla f\, \rho)$ while the interaction with the environment is through the term involving temperature: $\beta^{-1} \nabla \cdot (\nabla \cdot (D\, \rho))$. The second law of thermodynamics states that the entropy change of an isolated system is non-negative; in Assumption 6.22 we show how the above assumption is sufficient to satisfy the second law.

Lemma 6.25 (Divergence-free $j(x)$ does not decrease entropy). If the rate of increase of internal entropy is defined to be

$$\beta^{-1}\, \frac{dS_i}{dt} = \int_\Omega \nabla \left(\frac{\delta F}{\delta \rho} \right)\, J(x,t)\, \mathrm{d}x,$$

the condition $\nabla \cdot j = 0$ is sufficient to guarantee

$$\frac{dS_i}{dt} \geq 0.$$

Proof. Within the framework of linear non-equilibrium thermodynamics, it is assumed that the local entropy production is a product of the force $-\nabla \left(\frac{\delta F}{\delta \rho} \right)$ from (6.33) and the probability current $-J(x,t)$ from (6.13). This assumption in this form was first introduced by Prigogine (1955) based on the works of Onsager (1931a,b). See Frank (2005, Sec. 4.5) for a mathematical treatment and Jaynes (1980) for further discussion. The rate of entropy (S_i) increase is given by

$$\beta^{-1}\, \frac{dS_i}{dt} = \int_\Omega \nabla \left(\frac{\delta F}{\delta \rho} \right)\, J(x,t)\, \mathrm{d}x.$$

This can now be written using again as

$$\beta^{-1}\,\frac{dS_i}{dt} = \int\,\rho\; D : \left(\nabla\frac{\delta F}{\delta\rho}\right)\,\left(\nabla\frac{\delta F}{\delta\rho}\right)^{\top} + \int\,j\cdot\rho\,\left(\nabla\frac{\delta F}{\delta\rho}\right)\,dx.$$

The notation $D\,:\,A$ above denotes the Frobenius inner product of two matrices defined as

$$D\,:\,A = \sum_{ij} D_{ij}\,A_{ij}$$

for a symmetric matrix D. The first term in the above expression is non-negative, in order to ensure that $\frac{dS_i}{dt} \geq 0$, we require

$$\begin{aligned} 0 &= \int\,j\cdot\rho\,\left(\nabla\frac{\delta F}{\delta\rho}\right)\,\mathrm{d}x \\ &= \int\,\nabla\cdot(j\rho)\,\left(\frac{\delta F}{\delta\rho}\right)\,\mathrm{d}x; \end{aligned} \tag{6.31}$$

where the second equality again follows by integration by parts. It can be shown (Frank, 2005, Sec. 4.5.5) that the condition in Assumption 6.22, viz., $\nabla\cdot j(x) = 0$, is sufficient to make the above integral vanish and therefore for the entropy generation to be non-negative. ■

The above lemma shows that Assumption 6.22 is reasonable. Note that if the domain Ω were bounded or the gradient field $\nabla f(x)$ were decaying fast enough, we could do away with this assumption and simply claim Helmholtz theorem to the same end, i.e., define a divergence-free $j(x)$. The next lemma shows that $j(x)$ has another interesting property: in an average sense, it is orthogonal to the steepest descent gradient of the free-energy functional.

Lemma 6.26 (A conservative $j(x)$ is orthogonal to the direction of steepest-descent of the free-energy functional). If $\nabla\cdot j = 0$, we have

$$\int\,j\cdot\left(\nabla\frac{\delta F}{\delta\rho}\right)\,\mathrm{d}\rho = 0.$$

The proof is immediate from (6.31). We will see in **??** that this property results in acceleration of the convergence to the steady-state distribution ρ^{ss}. The orthogonal force $j(x)$ drives the dynamics away from regions where the steepest-descent gradient $\frac{\delta F}{\delta\rho}$ is small.

Proof of Theorem 6.24. The KL-divergence is non-negative: $F(\rho) \geq 0$ with equality if and only if $\rho = \rho^{\mathrm{ss}}$. The expression in (6.30) follows after writing

$$\log\rho^{\mathrm{ss}} = -\beta\Phi - \log Z(\beta).$$

We now show that $\frac{dF(\rho)}{dt} \leq 0$ with equality only at $\rho = \rho^{\mathrm{ss}}$ when $F(\rho)$ reaches its minimum and the Fokker-Planck equation achieves its steady-state. The first variation of $F(\rho)$ computed from (6.30) is

$$\frac{\delta F}{\delta\rho}(\rho) = \Phi(x) + \beta^{-1}\big(\log\rho + 1\big), \tag{6.32}$$

which helps us write the Fokker-Planck equation (6.8) as

$$\rho_t = \nabla\cdot\Big(-j\,\rho + \rho\,D\,\nabla\left(\frac{\delta F}{\delta\rho}\right)\Big). \tag{6.33}$$

Together, we can now write

$$\begin{aligned} \frac{\mathrm{d}\,F(\rho)}{\mathrm{d}\,t} &= \int_\Omega\,\rho_t\,\frac{\delta F}{\delta\rho}\,\mathrm{d}x \\ &= \int_\Omega\,\frac{\delta F}{\delta\rho}\,\nabla\cdot(-j\,\rho)\,\mathrm{d}x + \int_\Omega\,\frac{\delta F}{\delta\rho}\,\nabla\cdot\left(\rho\,D\,\nabla\left(\frac{\delta F}{\delta\rho}\right)\right)\mathrm{d}x. \end{aligned}$$

As we show in the proof of Lemma 6.25, the first term above is zero due to Assumption 6.22. Under suitable boundary condition on the Fokker-Planck equation which ensures that no probability mass flows across the boundary of the domain $\partial\Omega$, after an integration by parts, the second term can be written as

$$\begin{aligned}\frac{\mathrm{d}\,F(\rho)}{\mathrm{d}\,t} &= -\int_{\Omega} \rho\, D : \left(\nabla\, \frac{\delta F}{\delta \rho}(\rho)\right)\, \left(\nabla\, \frac{\delta F}{\delta \rho}(\rho)\right)^{\top}\, \mathrm{d}x \\ &\leq 0.\end{aligned}$$

The final inequality with the quadratic form holds because $D(x) \succeq 0$ is a covariance matrix. Moreover, we have from (6.32) that

$$\frac{\mathrm{d}\,F(\rho^{\mathrm{ss}})}{\mathrm{d}\,t} = 0.$$

■

The following companion lemma to Theorem 6.24 shows that the potential Φ in (6.30) is different from the loss function $f(x)$ in (6.1). This is an important lemma: roughly speaking, SGD minimizes a loss function Φ that is different from the loss function f it computes its mini-batch updates on.

Lemma 6.27 (Potential equals original loss iff noise is isotropic). If the diffusion matrix $D(x)$ is isotropic, i.e., a constant multiple of the identity, the implicit potential is the original loss itself

$$D(x) = c\, I_{d\times d} \quad \Leftrightarrow \quad \Phi(x) = f(x). \tag{6.34}$$

Proof. The forward implication can be checked by substituting $\rho^{\mathrm{ss}}(x) \propto e^{-c\,\beta f(x)}$ in the Fokker-Planck equation (6.8) while the reverse implication is true since otherwise (6.27) would not hold.

■

Theorem 6.28 (Potential Φ does not depend on β). The potential Φ is only a function of the dataset, architecture and the weights x, it does not depend on β^{-1}.

We defer the proof of this theorem to Section 6.4. Before closing this section, let us discuss two important properties of stochastic gradient descent that are widely believed in the deep learning literature and elucidated by Theorem 6.24.

Remark 6.29 (Inductive bias and implicit regularization). We have shown in Theorem 6.24 that SGD minimizes a free-energy that is the combination of an "energetic" term and an "entropic" term:

$$F(\rho) = \mathop{\mathbb{E}}_{x\in\rho}\, \Big[\Phi(x)\Big] - \beta^{-1}\mathrm{H}(\rho) + \text{constant}.$$

Two salient features of this loss function are particularly insightful. First, the steady-state distribution of SGD

$$\rho^{\mathrm{ss}}(x) \propto e^{-\beta\, \Phi(x)}$$

is such that it places most of its probability mass in regions of the parameter space with small values of Φ, not the critical points of the original loss $f(x)$. We will call this property the "inductive bias" of SGD. The second key observation is that SGD has a bias towards solutions that maximize the entropy of the distribution ρ; this is the "implicit regularization" in stochastic optimization algorithms, an artifact of the mini-batching and completely absent in gradient descent.

6.2.3.1. Practical implications. As Theorem 6.28 suggests, the potential $\Phi(x)$ does not depend on the learning rate η and the mini-batch size $\mathscr{b}$. Their effect therefore completely determines the strength of the entropic regularization term in (6.30). If $\beta^{-1} \to 0$, the implicit regularization of SGD goes to zero. This implies that

$$\beta^{-1} = \frac{\eta}{2\mathscr{b}} \text{ should not be small}$$

is a good tenet for regularization of SGD.

Remark 6.30 (Learning rate should scale linearly with batch-size to generalize well). In order to maintain the entropic regularization, the learning rate η needs to scale linearly with the batch-size $\mathscr{b}$. This prediction, based on Theorem 6.24, fits very well with empirical evidence wherein one obtains good generalization performance only with small mini-batches in deep networks (Keskar et al., 2016), or via such linear scaling (Goyal et al., 2017).

Remark 6.31 (Sampling with replacement is better than without replacement). The diffusion matrix for the case when mini-batches are sampled without replacement given by (6.5) is very close to the case when they are sampled with replacement (6.4). However, the corresponding inverse temperature is

$$\beta_{\text{wo}}^{-1} = \frac{\eta}{2\mathscr{b}}\left(1 - \frac{\mathscr{b}}{N}\right) \text{ should not be small.}$$

The extra factor of $\left(1 - \frac{\mathscr{b}}{N}\right)$ reduces the entropic regularization in (6.30), as $\mathscr{b} \to N$, the inverse temperature $\beta_{\text{wo}} \to \infty$. As a consequence, for the same learning rate η and batch-size $\mathscr{b}$, Theorem 6.24 predicts that sampling with replacement has better regularization than sampling without replacement. This effect is particularly pronounced at large batch-sizes.

6.2.4. Information bottleneck principle

This section takes a small detour into representation learning to further reveal the connections between Bayesian inference with respect to a prior and implicit regularization of stochastic algorithms.

Let Ξ denote the dataset. We are interested in learning a representation of Ξ, denoted by Z. This representation is relevant to inferring a "task" which is denoted by Y. In other words, the Markov chain of dependencies is given by

$$\Xi \;\to\; Z \;\to\; Y;$$

the representation Z depends on the data Ξ; given Z, the task can be inferred without the need of Ξ. We would like to compute a Z that is sufficient to infer Y, i.e.,

$$\mathrm{I}(Z;\, Y) = \mathrm{I}(\Xi;\, Y);$$

this is a generalization of the notion of a sufficient statistic (Keener, 2011). In particular, while a statistic is limited to being a deterministic function of the data Ξ, the Z above is required to be a stochastic function of Ξ; although note that $Z = \Xi$ satisfies the above condition as well. We now impose some regularization on Z, namely that it retains as little information as possible about the dataset, measured using $\mathrm{I}(Z;\, \Xi)$. This gives the following information bottleneck optimization problem

$$\begin{aligned} &\text{minimize} \quad \mathrm{I}(Z;\, \Xi) \\ &\text{subject to} \quad \mathrm{I}(Z;\, Y) = \mathrm{I}(\Xi;\, Y); \end{aligned} \tag{6.35}$$

the random variable Z preserves the information in Ξ relevant to predicting Y and discards the rest. The hard constraint on mutual information can be enforced in a soft manner using a Lagrange multiplier, which gives the information bottleneck Lagrangian (Tishby et al., 1999) after some algebraic manipulations

$$\mathrm{IB}(Z;\, \Xi, Y) = \beta^{-1}\, \mathrm{I}(Z;\, \Xi) - \mathrm{I}(Z;\, Y) \tag{6.36}$$

where β is a scalar Lagrange multiplier that controls the trade-off between how good Z is for the purpose of inferring Y and how much information it retains about Ξ. A large β corresponds to a smaller penalty on the information retained, consequently we have a representation that aims to do well on the task irrespective of the complexity of the representation; this is analogous to over-fitting. We next instantiate the problem in (6.36).

In general, the information bottleneck Lagrangian is intractable, we need the marginal distribution of the representation Z to compute it. We can however obtain a variational upper bound for $\mathrm{IB}(Z;\, \Xi, Y)$ following Alemi et al. (2016); Achille and Soatto (2016). First note that

$$\mathrm{I}(Z;\, Y) \leq H(Y;\, Z) - H(Y)$$

where $H(Y\,|\,Z) = -E_{Z\sim p(Z|\Xi)} \log q_\theta(Y\,|\,Z)$ is the cross-entropy between Y and Z and

$$\mathrm{I}(Z;\,\Xi) \leq \int \mathrm{d}\xi \mathrm{d}z\; p(\xi)\; p(z\,|\,\xi)\; \log \frac{p(z\,|\,\xi)}{q(z)}$$

for a variational approximation $q(z)$ to the true marginal $p(z)$.

$$\mathrm{IB}_\theta(Z;\,\Xi,Y) \leq \mathop{\mathbb{E}}_{(\Xi,Y)} \left\{ \mathop{\mathbb{E}}_{Z\sim p(Z|\Xi)} \Big[-\log q_\theta\,(Y\,|\,Z) \Big] + \beta^{-1}\, \mathrm{KL}\left(p\big(Z\,|\,\Xi\big)\;||\;q(Z)\right) \right\}. \tag{6.37}$$

The parameter θ parametrizes the variational family $q_\theta(Y\,|\,Z)$ which we would like to be close to the actual distribution $p(Y\,|\,Z)$.

Achille and Soatto (2017) show that the representation Z in the information bottleneck Lagrangian above can be connected with a similar information bottleneck on the weights X. If the mutual information between the (stochastic) weights X and the data Ξ is $\mathrm{I}(X;\Xi)$, we have

$$\mathrm{I}(X;\Xi) \leq \mathrm{I}(\Xi;Z) + \mathrm{TC}(Z) \leq \mathrm{I}(X;\Xi) + c; \tag{6.38}$$

where

$$\mathrm{TC}(Z) = \mathrm{KL}\big(p(Z)\;||\;\prod_{k=1}^{d}\; p(\mathrm{Z}_k)\big)$$

is the multi-variate mutual information or the total correlation (Ver Steeg and Galstyan, 2014) that encourages the representation Z to be disentangled. In other words, provided we have sufficiency, a minimal representation that is also disentangled can be obtained by minimizing the information that the weights retain about the dataset.

Consider therefore a new Lagrangian given by

$$\mathrm{IB}(X;\,\Xi,Y) = \mathop{\mathbb{E}}_{(\Xi,Y)} \left\{ \mathop{\mathbb{E}}_{X\sim q(X|\,\Xi,\,Y)} \Big[-\log p\,(Y\,|\,\Xi,X) \Big] + \beta^{-1}\, \mathrm{KL}\left(q\big(X\,|\,\Xi,Y\big)\;||\;r(X)\right) \right\}. \tag{6.39}$$

Note that the above expression is written in terms of the weights X; this is a different IB Lagrangian than (6.36) and was suggested by (Achille and Soatto, 2017). The distribution $r(X)$ is a variational approximation to the marginal of the weights $p(X|\Xi,Y)$, the latter being intractable. To make the above Lagrangian tractable, $r(X)$ is typically taken to be a factored Gaussian such that its parameters do not depend on the data Ξ, Y.

Remark 6.32 (SGD has an information bottleneck built-in). Denote the density of the posterior distribution $q(X\,|\,\Xi,Y)$ by $\rho(x)$. Define the function

$$f(X) = \mathop{\mathbb{E}}_{(\Xi,Y)} \Big[-\log q\,(Y\,|\,\Xi,X) \Big]$$

and consider a uniform $r(X)$ over the domain Ω to rewrite $\mathrm{IB}(X;\,\Xi,Y)$ in (6.39) as

$$\mathop{\mathbb{E}}_{X\sim\rho} \Big[f(X) \Big] - \beta^{-1}\, \mathrm{H}(\rho)$$

corresponding to stochastic gradient descent (6.2) with isotropic noise $D(x) = I$. This suggests that SGD has an information bottleneck built inside it—a bottleneck not on the representation Z but instead on the weights X; the two are related to each other via the duality bound in (6.38).

Remark 6.33 (Minimality of representation results in implicit regularization). Remark 6.32 provides numerous avenues for interpretation. First, it is surprising that the information bottleneck Lagrangian (6.36) which is an abstract statement about representation learning is so intimately connected to the variational problem that SGD solves. Second, if one constructs the desiderata of a good representation as

(i) sufficiency for the task,
(ii) minimality in size or information,
(iii) invariance to nuisance factors which are present in the data but do not inform the task,
(iv) disentanglement that enables uncorrelated latent factors to be understood independently of each other;

the first two are satisfied in SGD by virtue of mere minimization of an objective $f(x)$ using mini-batch updates. In fact, since the term $\mathrm{I}(\Xi;\, X)$ is directly related to the implicit regularization in Remark 6.32, the desire for minimality is what results in implicit regularization. It should be noted that one cannot get rid off this implicit regularization in SGD, it is a product of stochasticity of mini-batch updates; in other words, for non-zero β^{-1}, SGD always introduces a measure of minimality in the representation.

In fact, as Achille and Soatto (2017) show, provided a representation is sufficient for the task, it is minimal if and only if it is invariant to nuisances. In other words, training a deep network using SGD constructs representations that are invariant to nuisances, say affine transformations of the data. This is indeed corroborated by empirical evidence, deep networks in practice are robust to (small) transformations of input data, convolutional layers result in translational invariance. The final property of disentanglement can be imposed on the representation by employing a factored prior

$$r(Z) = \prod_i r_i(Z_i);$$

such a choice also simplifies calculations in Bayesian inference as we will see in the forthcoming section.

Remark 6.34 (Inductive bias vs. sufficiency of the representation). The reader will note that the functional in Remark 6.33 has the original loss $f(x)$ in the first term while SGD has the potential Φ in (6.30). The preceding development suggests that SGD does more than introducing sufficiency, it creates a sufficient representation only if the noise in mini-batch updates is isotropic, i.e., if $D(x) = I$ for all $x \in \Omega$. This is an interesting avenue for future work because it suggests that sufficiency is not sufficient: there is an additional property one needs to impose on the representation to match what SGD does in practice.

6.2.5. Bayesian inference

The general problem of Bayesian inference in our context can be written down as computing a posterior distribution $q(X)$ on the representation that is close to the true conditional distribution $p(X \,|\, \Xi, Y)$:

$$q^*(X) = \arg\min_{q \in \mathscr{Q}} \mathrm{KL}\left(q(X) \;||\; p(X \,|\, \Xi, Y)\right). \tag{6.40}$$

This optimization problem is in lieu of drawing samples from the posterior $p(X \,|\, \Xi, Y)$ using techniques from Markov chain Monte Carlo. The latter has a long history starting from the celebrated Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970), Georgio Parisi's usage of Langevin dynamics to draw samples (Parisi, 1981; Grenander and Miller, 1994), the Gibbs sampler (Geman and Geman, 1987), to more modern techniques in the chemistry and mathematics literatures (Leimkuhler and Matthews, 2016); see texts such as Doucet et al. (2001a,b); Gelfand and Smith (1990); Robert and Casella (2005) for a thorough treatment.

On the other hand, the variational inference in (6.40) converts this problem into an optimization problem on a family of distributions $\mathscr{Q}$; see Wainwright et al. (2008) for a detailed exposition. The key idea—and the challenge—of this program lies in considering a family $\mathscr{Q}$ that is rich enough to approximate or model the true $p(X \,|\, \Xi)$ and yet simple enough to lead to a tractable optimization problem. This technique, which is complementary to sampling, has also had a long history for both neural networks as well as graphical models; see Anderson and Peterson (1987); Jaakkola and Jordan (1996, 1997); Jordan et al. (1998a); Hinton and Van Camp (1993); MacKay (2003) etc. There has historically been a strong interplay of techniques in sampling and variational inference with those in statistical mechanics (Parisi, 1988; Mézard et al., 1987; Yedidia et al., 2001).

6.2.5.1. Evidence lower bound.

We now compute the so-called evidence lower bound:

$$\begin{aligned}
\mathrm{KL}\big(q(X) \;||\; p(X \,|\, \Xi)\big) &= \mathop{\mathbb{E}}_{X \sim q}\big[\log q(X)\big] - \mathop{\mathbb{E}}_{X \sim q}\big[\log p(X \,|\, \Xi)\big] \\
&= \mathop{\mathbb{E}}_{X \sim q}\big[\log q(X)\big] - \mathop{\mathbb{E}}_{X \sim q}\big[\log p(X, \Xi)\big] + \log p(\Xi) \\
&\geq 0.
\end{aligned}$$

It is typically difficult to estimate the last term, the marginalized likelihood of data Ξ, and it does not depend on the parameters of the representation x. With an eye towards solving (6.40) by minimizing over x, the above inequality suggests the definition:

$$\mathrm{ELBO}(q) = \mathop{\mathbb{E}}_{X\sim q}\big[\log q(X)\big] - \mathop{\mathbb{E}}_{X\sim q}\big[\log p(X,\Xi)\big]$$

which is a lower bound on $\log p(\Xi)$. Let us rewrite ELBO as

$$-\mathrm{ELBO}(x) = \mathop{\mathbb{E}}_{X\sim q_x}\Big[\underbrace{-\log p(\Xi|X)}_{\triangleq f(X)}\Big] + \mathrm{KL}\big(q_x(X\,|\,\Xi)\,||\,p(X)\big); \tag{6.41}$$

where the first is the data-fitting term, the likelihood of a dataset Ξ if the representation is X and the second term is the distance between the variational posterior and the prior $p(X)$. Optimizing ELBO with respect to x involves computing the gradient of the right hand side. The first term is relatively easy to handle, we can compute the gradient using Monte Carlo samples from q_x; this gives an estimate of the term $\nabla f_\theta(x)$.

The gradient of the second term in ELBO is harder, we need to pick a variational family q_x and a prior $p(X)$ such that the computation of the KL-divergence can be done in closed form. The mean-field setup considers

$$q_x(X\,|\,\Xi) = \prod_{i=1}^{d} q_{x_i}(X_i\,|\,\Xi);$$

in other words, the distribution q_x factors in d-independent factors, all of the same parametric form. We do the same for the prior $p(X)$ to split the KL-divergence term into a sum of d-independent KL-divergences, one for each scalar parameter x_i. For computational purposes, popular variants of ELBO make another crucial assumption:

$$\begin{aligned} q_{x_i}(X_i\,|\,\Xi) &= \mathrm{N}(\mu_{x_i},\ \sigma_{x_i}) \\ p(x_i) &= \mathrm{N}(0,1); \end{aligned} \tag{6.42}$$

where, with some abuse of notation, μ_{x_i} and σ_{x_i} together denote the parameter x_i. In this setup, the ELBO function looks like

$$-\mathrm{ELBO}(x) = \mathop{\mathbb{E}}_{X\sim q_x}\Big[f(X)\Big] + \sum_{i=1}^{d} \mathrm{KL}\big(N(\mu_{x_i},\sigma_{x_i})\ ||\ N(0,1)\big). \tag{6.43}$$

The KL-divergence between two scalar Gaussians is easy to compute and turns out to be

$$\mathrm{KL}\big(N(\mu,\sigma)\ ||\ N(0,1)\big) = -\frac{1}{2}\left(1+\log(\sigma^2)-\mu^2-\sigma^2\right). \tag{6.44}$$

We next make a few pointed remarks about Bayesian inference as typically used to train a deep network in the light of our results in Section 6.2.

Remark 6.35 (Phase transition for the variances). Effectively, minimizing (6.43) imposes a quadratic penalty on the means μ while the variances are penalized as $\sigma^2 - \log(\sigma^2)$. Note that the latter penalty has a global minimum at $\sigma = 1$: if $\sigma < 1$ for some synapse, minimizing $-$ELBO favors smaller values of σ in this case while the variances are encouraged to grow unbounded if $\sigma > 1$. Let us note that the case for $\sigma < 1$ can also be achieved by imposing ℓ_2-regularization (weight decay) on the parameters of the neural network. On the other hand, if the prior is uniform, the KL-divergence is negative entropy and we only have a penalty of $-\frac{1}{2}\log\sigma^2$ which encourages all the variances to grow.

Remark 6.36 (Gaussian prior in ELBO vs. uniform prior of SGD). In order to make computations with ELBO tractable, it is popular to optimize over a mean-field Gaussian parameter family as seen in (6.42). Since the gradient of the energetic term $\mathbb{E}_{X\sim q_x}\big[f(X)\big]$ is computing using Monte Carlo samples of X over mini-batches of the dataset Ξ, it implicitly imposes a uniform prior on the steady-state distribution $\rho(x,t)$ as seen in Theorem 6.24, note that the scale of this implicit prior is controlled by the coefficient β^{-1} and one does not have explicit control over it otherwise. It is easy to see that the Gaussian prior as imposed by ELBO conflicts with the uniform prior as imposed by SGD internally, and this is also seen in practice: deep

networks trained using variational inference do not generalize as well as state-of-the-art ones trained using SGD; see (Chen et al., 2014, 2015a) etc.

Resolving this conflict is often done in an ad-hoc way by introducing a scalar multiplier on the KL-divergence term which affords a practitioner more control on this fight (Higgins et al., 2017), or motivated from variational tempering (Mandt et al., 2016b). Indeed, this coefficient is also the Lagrange multiplier in the information bottleneck Lagrangian (6.39).

6.3. Empirical characterization of SGD dynamics

This section shows that the diffusion matrix $D(x)$ for modern deep networks is highly non-isotropic with a very low rank. We also analyze trajectories of SGD and detect periodic components using a frequency analysis in Section 6.3.3; this validates the prediction of

6.3.1. Experimental setup

We consider the following three networks on the MNIST (LeCun et al., 1998) and the CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009).

(i) **small-lenet:** a smaller version of LeNet (LeCun et al., 1998) on MNIST with batch-normalization and dropout (0.1) after both convolutional layers of 8 and 16 output channels, respectively. The fully-connected layer has 128 hidden units. This network has $13,338$ weights and reaches about 0.75% training and validation error.

(ii) **small-fc:** a fully-connected network with two-layers, batch-normalization and rectified linear units that takes 7×7 down-sampled images of MNIST as input and has 64 hidden units. Experiments in Section 6.3.3 use a smaller version of this network with 16 hidden units and 5 output classes ($30,000$ input images); this is called **tiny-fc**.

(iii) **small-allcnn:** this a smaller version of the fully-convolutional network for CIFAR-10 and CIFAR-100 introduced by Springenberg et al. (2014) with batch-normalization and $12,24$ output channels in the first and second block respectively. It has $26,982$ weights and reaches about 11% and 17% training and validation errors, respectively.

We train the above networks with SGD with appropriate learning rate annealing and Nesterov's momentum set to 0.9. We do not use any data-augmentation and pre-process data using global contrast normalization with ZCA for CIFAR-10 and CIFAR-100.

We use networks with about $20,000$ weights to keep the eigen-decomposition of $D(x) \in \mathbb{R}^{d \times d}$ tractable. These networks however possess all the architectural intricacies such as convolutions, dropout, batch-normalization etc. We evaluate $D(x)$ using (6.4) with the network in evaluation mode.

6.3.2. Highly non-isotropic $D(x)$ for deep networks

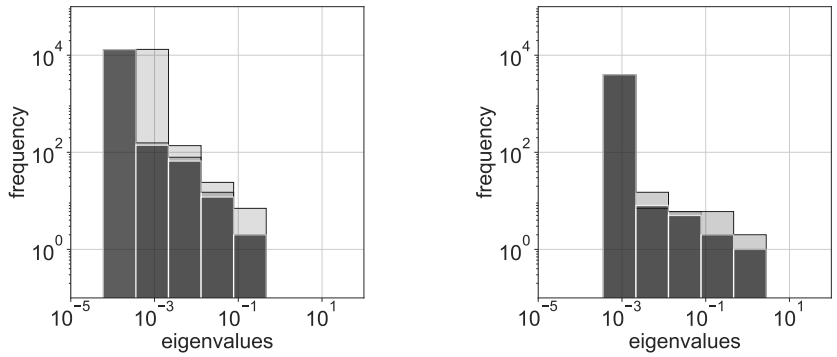
Figures 6.1 and 6.2 show the eigenspectrum[1] of the diffusion matrix. In all cases, it has a large fraction of almost-zero eigenvalues with a very small rank that ranges between 0.3% - 2%. Moreover, non-zero eigenvalues are spread across a vast range with a large variance.

Remark 6.37 (Noise in SGD is largely independent of the weights). The variance of noise in (6.2) is

$$\frac{\eta}{\not{b}} \frac{D(x_k)}{} = 2\,\beta^{-1} D(x_k).$$

We have plotted the eigenspectra of the diffusion matrix in Figure 6.1 and Figure 6.2 at three different instants, 20%, 40% and 100% training completion; they are almost indistinguishable. This implies that the variance of the mini-batch gradients in deep networks can be considered a constant, highly non-isotropic matrix.

[1] thresholded at $\lambda_{\max} \times d \times$ machine-precision. This formula is widely used, for instance, in numpy.



(A) MNIST: small-lenet
$\lambda(D) = (0.3 \pm 2.11) \times 10^{-3}$
$\text{rank}(D) = 1.8\%$

(B) MNIST: small-fc
$\lambda(D) = (0.9 \pm 18.5) \times 10^{-3}$
$\text{rank}(D) = 0.6\%$

FIGURE 6.1. Eigenspectrum of $D(x)$ at three instants during training (20%, 40% and 100% completion, darker is later). The eigenspectrum in Figure 6.1b for the fully-connected network has a much smaller rank and much larger variance than the one in Figure 6.1a which also performs better on MNIST. This indicates that convolutional networks are better conditioned than fully-connected networks in terms of $D(x)$.

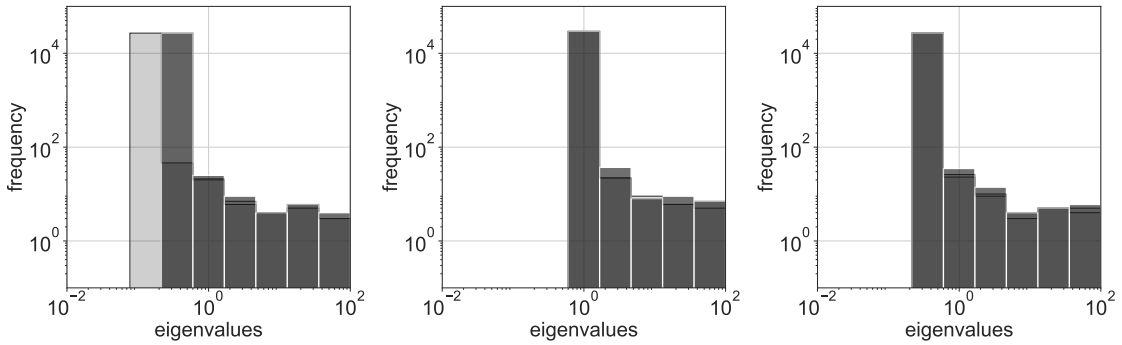
Remark 6.38 (More non-isotropic diffusion if data is diverse). The eigenspectra in Figure 6.2 for CIFAR-10 and CIFAR-100 have much larger eigenvalues and standard-deviation than those in Figure 6.1, this is expected because the images in the CIFAR datasets have more variety than those in MNIST. Similarly, while CIFAR-100 has qualitatively similar images as CIFAR-10, it has $10\times$ more classes and as a result, it is a much harder dataset. This correlates well with the fact that both the mean and standard-deviation of the eigenvalues in Figure 6.2b are much higher than those in Figure 6.2a. Input augmentation increases the diversity of mini-batch gradients. This is seen in Figure 6.2c where the standard-deviation of the eigenvalues is much higher as compared to Figure 6.2a.

Remark 6.39 (Inverse temperature scales with the mean of the eigenspectrum). Remark 6.38 shows that the mean of the eigenspectrum is large if the dataset is diverse. Based on this, we propose that the inverse temperature β should scale linearly with the mean of the eigenvalues of D:

$$\left(\frac{\eta}{\mathscr{b}}\right)\left(\frac{1}{d}\sum_{k=1}^{d}\lambda(D)\right) = \text{constant}; \tag{6.45}$$

where d is the number of weights. This keeps the noise in SGD constant in magnitude for different values of the learning rate η, mini-batch size $\mathscr{b}$, architectures, and datasets. Note that other hyper-parameters which affect stochasticity such as dropout probability are implicit inside D.

Remark 6.40 (Variance of the eigenspectrum informs architecture search). Compare the eigenspectra in Figures 6.1a and 6.1b with those in Figures 6.2a and 6.2c. The former pair shows that small-lenet which is a much better network than small-fc also has a much larger rank, i.e., the number of non-zero eigenvalues ($D(x)$ is symmetric). The second pair shows that for the same dataset, data-augmentation creates a larger variance in the eigenspectrum. This suggests that both the quantities, viz., rank of the diffusion matrix and the variance of the eigenspectrum, inform the performance of a given architecture on the dataset. Note that as discussed in Remark 6.39, the mean of the eigenvalues can be controlled using the learning rate η and the batch-size $\mathscr{b}$.



(A) CIFAR-10
$\lambda(D) = 0.27 \pm 0.84$
$\text{rank}(D) = 0.34\%$

(B) CIFAR-100
$\lambda(D) = 0.98 \pm 2.16$
$\text{rank}(D) = 0.47\%$

(C) CIFAR-10: data augmentation
$\lambda(D) = 0.43 \pm 1.32$
$\text{rank}(D) = 0.32\%$

FIGURE 6.2. Eigenspectrum of $D(x)$ at three instants during training (20%, 40% and 100% completion, darker is later). The eigenvalues are much larger in magnitude here than those of MNIST in Figure 6.1, this suggests a larger gradient diversity for CIFAR-10 and CIFAR-100. The diffusion matrix for CIFAR-100 in Figure 6.2b has larger eigenvalues and is more non-isotropic and has a much larger rank than that of Figure 6.2a; this suggests that gradient diversity increases with the number of classes. As Figure 6.2a and Figure 6.2c show, augmenting input data increases both the mean and the variance of the eigenvalues while keeping the rank almost constant.

This observation is useful for automated architecture search where we can use the quantity

$$\frac{\text{rank}(D)}{d} + \text{var}\left(\lambda(D)\right)$$

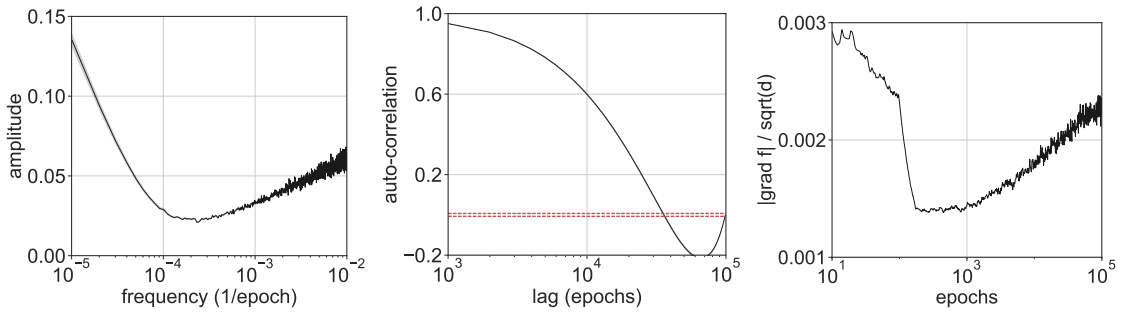
to estimate the efficacy of a given architecture, possibly, without even training, since D does not depend on the weights much. This task currently requires enormous amounts of computational power (Zoph and Le, 2016; Baker et al., 2016; Brock et al., 2017).

6.3.3. Analysis of long-term trajectories

We train a smaller version of small-fc on 7×7 down-sampled MNIST images for 10^5 epochs and store snapshots of the weights after each epoch to get a long trajectory in the weight space. We discard the first 10^3 epochs of training ("burnin") to ensure that SGD has reached the steady-state. The learning rate is fixed to 10^{-3} after this, up to 10^5 epochs.

Remark 6.41 (Low-frequency periodic components in SGD trajectories). Iterates of SGD, after it reaches the neighborhood of a critical point $\|\nabla f(x_k)\| \leq \varepsilon$, are expected to perform Brownian motion with variance $\text{var}\left(\nabla f_{\theta}(x)\right)$, the FFT in Figure 6.3a would be flat if this were so. Instead, we see low-frequency modes in the trajectory that are indicators of a periodic dynamics of the force $j(x)$. These modes are not sharp peaks in the FFT because $j(x)$ can be a non-linear function of the weights thereby causing the modes to spread into all dimensions of x. The FFT is dominated by jittery high-frequency modes on the right with a slight increasing trend; this suggests the presence of colored noise in SGD at high-frequencies.

The auto-correlation (AC) in Figure 6.3b should be compared with the AC for Brownian motion which decays to zero very quickly and stays within the red confidence bands (99%). Our iterates are significantly correlated with each other even at very large lags. This further indicates that trajectories of SGD do not perform Brownian motion.



(A) FFT of $x^i_{k+1} - x^i_k$ (B) Auto-correlation (AC) of x^i_k (C) Normalized gradient $\frac{\|\nabla f(x_k)\|}{\sqrt{d}}$

FIGURE 6.3. Empirical evidence of limit cycles in SGD dynamics: Figure 6.3a shows the Fast Fourier Transform (FFT) of $x^i_{k+1} - x^i_k$ where k is the number of epochs and i denotes the index of the weight. Figure 6.3b shows the auto-correlation of x^i_k with 99% confidence bands denoted by the dotted red lines. Both Figures 6.3a and 6.3b show the mean and one standard-deviation over the weight index i; the standard deviation is very small which indicates that all the weights have a very similar frequency spectrum. Figures 6.3a and 6.3b should be compared with the FFT of white noise which should be flat and the auto-correlation of Brownian motion which quickly decays to zero, respectively. Figure 6.3a and Figure 6.3 therefore show that trajectories of SGD are not simply Brownian motion. Moreover the gradient at these locations is quite large (Figure 6.3c).

Remark 6.42 (Gradient magnitude in deep networks is always large). Figure 6.3c shows that the full-gradient computed over the entire dataset (without burnin) does not decrease much with respect to the number of epochs. While it is expected to have a non-zero gradient norm because SGD only converges to a neighborhood of a critical point for non-zero learning rates, the magnitude of this gradient norm is quite large. This magnitude drops only by about a factor of 3 over the next 10^5 epochs. The presence of a non-zero $j(x)$ also explains this, it causes SGD to be away from critical points, this phenomenon is made precise in Theorem 6.48. Let us note that a similar plot is also seen in Shwartz-Ziv and Tishby (2017) for the per-layer gradient magnitude.

6.4. NON-EQUILIBRIUM PHENOMENA

We now gives an explicit formula for the potential $\Phi(x)$. We also discuss implications of this for generalization in Section 6.4.3.

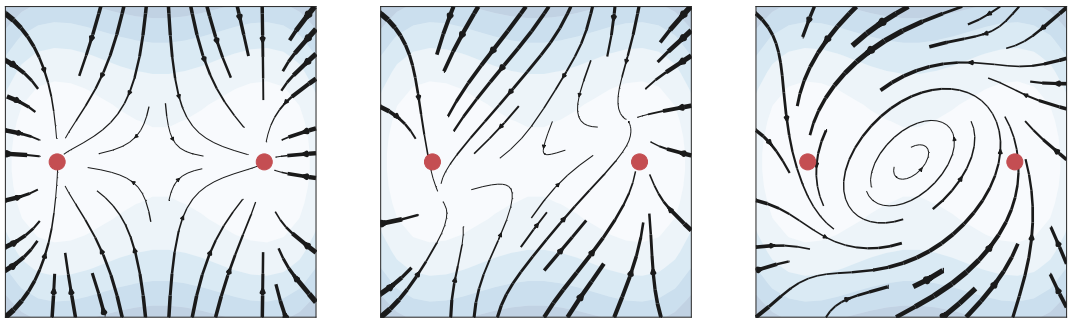
The fundamental difficulty in obtaining an explicit expression for Φ is that even if the diffusion matrix $D(x)$ is full-rank, there need not exist a function $\Phi(x)$ such that $\nabla\Phi(x) = D^{-1}(x)\,\nabla f(x)$ at all $x \in \Omega$. We therefore split the analysis into two cases:

(i) a local analysis near any critical point $\nabla f(x) = 0$ where we linearize $\nabla f(x) = Fx$ and $\nabla\Phi(x) = Ux$ to compute $U = G^{-1}\,F$ for some G, and
(ii) the general case where $\nabla\Phi(x)$ cannot be written as a local rotation and scaling of $\nabla f(x)$.

Let us introduce these cases with an example from Noh and Lee (2015).

Example 6.43 (Double-well potential with limit cycles). Define

$$\Phi(x) = \frac{(x_1^2 - 1)^2}{4} + \frac{x_2^2}{2}.$$



(A) $\lambda = 0$ (B) $\lambda = 0.5$ (C) $\lambda = 1.5$

FIGURE 6.4. Gradient field for the dynamics in Example 6.43: line-width is proportional to the magnitude of the gradient $\|\nabla f(x)\|$, red dots denote the most likely locations of the steady-state $e^{-\Phi}$ while the potential Φ is plotted as a contour map. The critical points of $f(x)$ and $\Phi(x)$ are the same in Figure 6.4a, namely $(\pm 1, 0)$, because the force $j(x) = 0$. For $\lambda = 0.5$ in Figure 6.4b, locations where $\nabla f(x) = 0$ have shifted slightly as predicted by Theorem 6.48. The force field also has a distinctive rotation component, see Remark 6.45. In Figure 6.4c with a large $\|j(x)\|$, SGD converges to limit cycles around the saddle point at the origin. This is highly surprising and demonstrates that the solutions obtained by SGD may be very different from local minima.

Instead of constructing a diffusion matrix $D(x)$, we will directly construct different gradients $\nabla f(x)$ that lead to the same potential Φ; these are equivalent but the later is much easier. The dynamics is given by $\mathrm{d}X = -\nabla f(X)\, dt + \sqrt{2}\, \mathrm{d}B(t)$, where

$$\nabla f(x) = -j(x) + \nabla \Phi(x).$$

We pick $j = \lambda e^{\Phi}\, J^{\mathrm{ss}}(x)$ for some parameter $\lambda > 0$ where

$$J^{\mathrm{ss}}(x) = e^{-\frac{(x_1^2+x_2^2)^2}{4}}\ (-x_2,\, x_1).$$

Note that this satisfies (6.10) and does not change $\rho^{\mathrm{ss}} = e^{-\Phi}$. Figure 6.4 shows the gradient field $f(x)$ along with a discussion.

6.4.1. Local analysis at a critical point

Without loss of generality, let $x = 0$ be a critical point of $f(x)$. This critical point can be a local minimum, maximum, or even a saddle point. We linearize the gradient around the origin and define a fixed matrix $F \in \mathbb{R}^{d\times d}$ (the Hessian) to be $\nabla f(x) = Fx$. Let $D = D(0)$ be the constant diffusion matrix matrix. The dynamics in (6.2) can now be written as

$$\mathrm{d}X = -FX\ \mathrm{d}t + \sqrt{2\beta^{-1}\, D}\ \mathrm{d}B(t). \tag{6.46}$$

Lemma 6.44 (Linearization). The matrix F in (6.46) can be uniquely decomposed into

$$F = (D+Q)\, U; \tag{6.47}$$

D and Q are the symmetric and anti-symmetric parts of a matrix G with $GF^{\top} - FG^{\top} = 0$, to get $\Phi(x) = \frac{1}{2}x^{\top}Ux$.

The above lemma is a classical result if the critical point is a local minimum, i.e., if the loss is locally convex near $x = 0$; this case has also been explored in machine learning before (Mandt et al., 2016a). We refer to Kwon et al. (2005) for the proof that linearizes around any critical point.

Remark 6.45 (Rotation of gradients). We see from Lemma 6.44 that, near a critical point,

$$\nabla f = (D+Q)\,\nabla\Phi - \beta^{-1}\nabla\cdot D - \beta^{-1}\nabla\cdot Q \tag{6.48}$$

up to the first order. This suggests that the effect of $j(x)$ is to rotate the gradient field and move the critical points, also seen in Figure 6.4b. Note that $\nabla\cdot D = 0$ and $\nabla\cdot Q = 0$ in the linearized analysis.

6.4.2. General case

We first give a short description of how most likely trajectories look like. It is natural to imagine that iterates of SGD are concentrated around local minima of a general loss function $f(x)$ for short time intervals. After exponentially long periods, these iterates transition across saddle points in the energy landscape into nearby local minima. This picture suggests that trajectories of SGD are close to Brownian motion most of the time, indeed, as we saw in (6.46) in Section 6.4.1, SGD can be modeled as the Ornstein-Uhlenbeck process around a critical point. In the general case however, when the diffusion matrix D is not identity, most likely trajectories of SGD are closed loops in the parameter space, also known as limit cycles, this is the subject of the following theorem and a characteristic of non-equilibrium systems.

Theorem 6.46 (Most likely trajectories of SGD are limit cycles). The force $j(x)$ does not decrease $F(\rho)$ in (6.30) and introduces a deterministic component in SGD given by

$$\dot{x} = j(x); \tag{6.49}$$

where $j(x) = J^{\text{ss}}/\rho^{\text{ss}}$. The condition $\nabla\cdot j(x) = 0$ in Assumption 6.22 implies that most likely trajectories of SGD traverse closed trajectories in weight space.

Proof. The Fokker-Planck operator

$$L\,\rho = \nabla\cdot\left(-j\,\rho + D\,\nabla\Phi\,\rho + \beta^{-1}D\,\nabla\rho\right)$$

can be split into two operators

$$L = L_S + L_A,$$

where

$$L_S\,\rho = \nabla\cdot\left(D\,\nabla\Phi\,\rho + \beta^{-1}D\,\nabla\rho\right) \tag{6.50}$$

is symmetric and satisfies the detailed balance condition (6.13), and an anti-symmetric operator

$$\begin{aligned} L_A\,\rho &= \nabla\cdot(-j\rho) \\ &= \nabla\cdot(-D\,\nabla\Phi\,\rho + \nabla f\rho) \\ &= \nabla\cdot\left(\beta^{-1}\,D\,\nabla\rho + \nabla f\rho\right). \end{aligned} \tag{6.51}$$

The above split exists for any linear operator and is akin to the fact that any square matrix can be split as a sum of a symmetric and skew-symmetric matrix. The symmetric part L_S is very well-studied in physics/chemistry (Leimkuhler and Matthews, 2016) and mathematics (Pavliotis, 2016) but the skew-symmetric operator L_A which is the purview of non-equilibrium thermodynamics (Ottinger, 2005) will turn out to be very useful for our purposes in this section.

As was also seen in Lemma 6.23, we note that L_A does not affect the free energy $F(\rho)$:

$$\begin{aligned} \frac{d}{\mathrm{d}t}F(\rho) &= \int_\Omega \frac{\delta F}{\delta\rho}\,\rho_t\,\mathrm{d}x \\ &= \int_\Omega \frac{\delta F}{\delta\rho}\,\nabla\cdot(-j\,\rho)\,\mathrm{d}x \\ &= 0, \end{aligned}$$

by Assumption 6.22. This indicates that the dynamics of the skew-symmetric operator is completely deterministic and conserves energy, hence the name "conservative force".

In fact, the equation (6.51) is known as the Liouville equation (Frank, 2005) and describes the density of a completely deterministic dynamics given by (6.49). Note that the conservative force satisfies the two conditions in (6.25) and (6.27):

$$\begin{aligned} \text{(divergence free)} \quad & \nabla \cdot j(x) = 0, \quad \text{and} \\ \text{(flow along contours)} \quad & j \cdot \nabla \Phi(x) = 0. \end{aligned}$$

The first condition creates trajectories that loop in the weight space, these are precisely the low frequency modes we observed in the experiment in Section 6.3.3. ■

We next give the general expression for the deviation of the critical points $\nabla\Phi$ from those of the original loss ∇f.

Remark 6.47 (A-type stochastic calculus). A Fokker-Planck equation is a deterministic partial differential equation (PDE) and every steady-state distribution, $\rho^{\text{ss}} \propto e^{-\beta\Phi}$ in this case, has a unique such PDE that achieves it. However, the same PDE can be tied to different SDEs depending on the stochastic integration scheme, e.g., Itô, Stratonovich (Risken, 1996; Oksendal, 2003), Hanggi (Hänggi, 1978), α-type etc. An "A-type" interpretation is one such scheme (Ao et al., 2007; Shi et al., 2012). It is widely used in non-equilibrium studies in physics and biology (Wang et al., 2008; Zhu et al., 2004) because it allows one to compute the steady-state distribution easily; its implications are supported by other mathematical analyses such as Tel et al. (1989); Qian (2014).

The main result of the section now follows. It exploits the A-type interpretation to compute the difference between the most likely locations of SGD which are given by the critical points of the potential $\Phi(x)$ and those of the original loss $f(x)$.

Theorem 6.48 (Most likely locations are not the critical points of the loss). The Itô SDE

$$\mathrm{d}X = -\nabla f(X)\,\mathrm{d}t + \sqrt{2\beta^{-1}D(X)}\,\mathrm{d}B(t)$$

is equivalent to the A-type SDE (Ao et al., 2007; Shi et al., 2012)

$$\mathrm{d}X = -\Big(D(X)+Q(X)\Big)\,\nabla\Phi(X)\,\mathrm{d}t + \sqrt{2\beta^{-1}D(X)}\,\mathrm{d}B(t) \tag{6.52}$$

with the same steady-state distribution $\rho^{\text{ss}} \propto e^{-\beta\Phi(x)}$ and Fokker-Planck equation (6.8) if for all $x \in \Omega$,

$$\nabla f(x) = \Big(D(x)+Q(x)\Big)\,\nabla\Phi(x) - \beta^{-1}\nabla\cdot\Big(D(x)+Q(x)\Big). \tag{6.53}$$

The anti-symmetric matrix $Q(x)$ and the potential $\Phi(x)$ can be explicitly computed in terms of the gradient $\nabla f(x)$ and the diffusion matrix $D(x)$. The potential $\Phi(x)$ does not depend on β.

Proof. All the matrices below depend on the weights x; we suppress this to keep the notation clear. Our original SDE is given by

$$\mathrm{d}X = -\nabla f\,\mathrm{d}t + \sqrt{2\beta^{-1}\,D}\,\mathrm{d}B(t).$$

We will transform the original SDE into a new SDE

$$G\,\mathrm{d}X = -\nabla\Phi\,\mathrm{d}t + \sqrt{2\beta^{-1}\,S}\,\mathrm{d}B(t) \tag{6.54}$$

where S and A are the symmetric and anti-symmetric parts of G^{-1},

$$\begin{aligned} S &= \frac{G^{-1}+G^{-T}}{2}, \\ A &= G^{-1} - S. \end{aligned}$$

Since the two SDEs above are equal to each other, both the deterministic and the stochastic terms have to match. This gives

$$\begin{aligned}\nabla f(x) &= G\,\nabla\Phi(x)\\ D &= \frac{G+G^\top}{2}\\ Q &= \frac{G-G^\top}{2}.\end{aligned}$$

Using the above expression, we can now give an explicit, although formal, expression for the potential:

$$\Phi(x) = \int_0^1 \Big(G^{-1}(\Gamma(s))\;\nabla f(\Gamma(s))\Big)\;\cdot \mathrm{d}\Gamma(s); \tag{6.55}$$

where $\Gamma : [0,1] \to \Omega$ is any curve such that $\Gamma(1) = x$ and $\Gamma(0) = x(0)$ which is the initial condition of the dynamics in (6.2). Note that $\Phi(x)$ does not depend on β because $G(x)$ does not depend on β.

We now write the modified SDE (6.54) as a second-order Langevin system after introducing a velocity variable p with $q \triangleq X$ and mass m:

$$\begin{aligned}\mathrm{d}q &= \frac{p}{m}\,\mathrm{d}t\\ \mathrm{d}p &= -\,(S+A)\;\frac{p}{m}\,\mathrm{d}t - \nabla_q\Phi(q)\,\mathrm{d}t + \sqrt{2\beta^{-1}\,S}\,\mathrm{d}B(t).\end{aligned} \tag{6.56}$$

The key idea in Yin and Ao (2006) is to compute the Fokker-Planck equation of the system above and take its zero-mass limit. The steady-state distribution of this equation, which is also known as the Klein-Kramer's equation, is

$$\rho^{\mathrm{ss}}(q,p) = \frac{1}{Z'(\beta)}\;\exp\left(-\beta\;\Phi(q) - \frac{\beta p^2}{2m}\right); \tag{6.57}$$

where the position and momentum variables are decoupled. The zero-mass limit is given by

$$\begin{aligned}\rho_t &= \nabla\cdot G\,\Big(\nabla\Phi\;\rho + \beta^{-1}\;\nabla\rho\Big),\\ &= \nabla\cdot\Big(D\;\nabla\Phi\;\rho + Q\;\nabla\Phi\;\rho + (D+Q)\;\beta^{-1}\;\nabla\rho\Big)\\ &= \nabla\cdot\Big(D\;\nabla\Phi\;\rho + Q\;\nabla\Phi\;\rho\Big) + \nabla\cdot\Big(D\;\beta^{-1}\;\nabla\rho\Big) + \beta^{-1}\underbrace{\nabla\cdot(Q\;\nabla)\;\rho}_{*}\end{aligned} \tag{6.58}$$

We now exploit the fact that Q is defined to be an anti-symmetric matrix. Note that $\sum_{i,j}\partial_i\partial_j\,(Q_{ij}\rho) = 0$ because Q is anti-symmetric. Rewrite the third term on the last step $(*)$ as

$$\begin{aligned}\nabla\cdot(Q\;\nabla\rho) &= \sum_{ij}\;\partial_i\Big(Q_{ij}\;\partial_j\rho\Big)\\ &= -\sum_{ij}\;\partial_i\Big(\partial_j Q_{ij}\Big)\rho\\ &= -\nabla\cdot\Big(\nabla\cdot Q\Big)\rho.\end{aligned} \tag{6.59}$$

We now use the fact that (6.2) has $\rho^{\mathrm{ss}} \propto e^{-\beta\Phi}$ as the steady-state distribution as well. Since the steady-state distribution is uniquely determined by a Fokker-Planck equation, the two equations (6.8) and (6.58) are the same. Let us decompose the second term in (6.8):

$$\begin{aligned}&\beta^{-1}\sum_{i,j}\;\partial_i\partial_j\Big[D_{ij}(x)\rho(x)\Big]\\ &= \beta^{-1}\sum_{i,j}\;\partial_i\big\{\,(\partial_j D_{ij})\,\rho\,\big\} + \beta^{-1}\;\sum_{i,j}\;\partial_i\big\{D_{ij}\partial_j\rho\,\big\}.\end{aligned}$$

Observe that the brown terms are equal. Rearranging and matching the drift terms in the two Fokker-Planck equations then gives

$$\nabla f = (D+Q)\,\nabla\Phi - \beta^{-1}\nabla\cdot D - \beta^{-1}\nabla\cdot Q$$

The critical points of Φ are different from those of the original loss f by a term that is $\beta^{-1}\nabla\cdot\,(D+Q)$.

■

Remark 6.49 (SGD is far away from critical points). The time spent by a Markov chain at a state x is proportional to its steady-state distribution $\rho^{\text{ss}}(x)$. While it is easily seen that SGD does not converge in the Cauchy sense due to the stochasticity, it is very surprising that it may spend a significant amount of time away from the critical points of the original loss. If $D(x)+Q(x)$ has a large divergence, the set of states with $\nabla\Phi(x)=0$ might be drastically different than those with $\nabla f(x)=0$. This is also seen in example Figure 6.4c; in fact, SGD may even converge around a saddle point.

This also closes the logical loop we began in Section 6.2 where we assumed the existence of ρ^{ss} and defined the potential Φ using it. Lemma 6.44 and Theorem 6.48 show that both can be defined uniquely in terms of the original quantities, i.e., the gradient term $\nabla f(x)$ and the diffusion matrix $D(x)$. There is no ambiguity as to whether the potential $\Phi(x)$ results in the steady-state $\rho^{\text{ss}}(x)$ or vice-versa.

Remark 6.50 (Consistent with the linear case). Theorem 6.48 presents a picture that is completely consistent with Lemma 6.44. If $j(x)=0$ and $Q(x)=0$, or if Q is a constant like the linear case in Lemma 6.44, the divergence of $Q(x)$ in (6.53) is zero.

Remark 6.51 (Out-of-equilibrium effect can be large even if D is constant). The presence of a $Q(x)$ with non-zero divergence is the consequence of a non-isotropic $D(x)$ and it persists even if D is constant and independent of weights x. So long as D is not isotropic, as we discussed in the beginning of Section 6.4, there need not exist a function $\Phi(x)$ such that $\nabla\Phi(x) = D^{-1}\,\nabla f(x)$ at all x. This is also seen in our experiments, the diffusion matrix is almost constant with respect to weights for deep networks, but consequences of out-of-equilibrium behavior are still seen in Section 6.3.3.

Remark 6.52 (Out-of-equilibrium effect increases with β^{-1}). The effect predicted by (6.53) becomes more pronounced if $\beta^{-1} = \frac{\eta}{2b}$ is large. In other words, small batch-sizes or high learning rates cause SGD to be drastically out-of-equilibrium. Theorem 6.24 also shows that as $\beta^{-1}\to 0$, the implicit entropic regularization in SGD vanishes. Observe that these are exactly the conditions under which we typically obtain good generalization performance for deep networks (Keskar et al., 2016; Goyal et al., 2017). This suggests that non-equilibrium behavior in SGD is crucial to obtain good generalization performance, especially for high-dimensional models such as deep networks where such effects are expected to be more pronounced.

6.4.3. Connections to generalization

It was found that solutions of discrete learning problems that generalize well belong to dense clusters in the weight space (Baldassi et al., 2015, 2016a). Such dense clusters are exponentially fewer compared to isolated solutions. To exploit these observations, the authors proposed a loss called "local entropy" that is out-of-equilibrium by construction and can find these well-generalizable solutions easily. This idea has also been successful in deep learning where Chaudhari et al. (2016) modified SGD to seek solutions in "wide minima" with low curvature to obtain improvements in generalization performance as well as convergence rate (Chaudhari et al., 2017).

Local entropy is a smoothed version of the original loss given by

$$f_\gamma(x) = -\log\left(G_\gamma * e^{-f(x)}\right),$$

where G_γ is a Gaussian kernel of variance γ. Even with an isotropic diffusion matrix, the steady-state distribution with $f_\gamma(x)$ as the loss function is $\rho^{\text{ss}}_\gamma(x) \propto e^{-\beta f_\gamma(x)}$. For large values of γ, the new loss makes the original local minima exponentially less likely. In other words, local entropy does not rely on non-isotropic

gradient noise to obtain out-of-equilibrium behavior, it gets it explicitly, by construction. This is also seen in Figure 6.4c: if SGD is drastically out-of-equilibrium, it converges around the "wide" saddle point region at the origin which has a small local entropy.

Actively constructing out-of-equilibrium behavior leads to good generalization in practice. Our evidence that SGD on deep networks itself possesses out-of-equilibrium behavior then indicates that SGD for deep networks generalizes well because of such behavior.

6.4.4. Related work

6.4.4.1. SGD, variational inference and implicit regularization: The idea that SGD is related to variational inference has been seen in machine learning before (Duvenaud et al., 2016; Mandt et al., 2016a) under assumptions such as quadratic steady-states; for instance, see Mandt et al. (2017) for methods to approximate steady-states using SGD. Our results here are very different, we would instead like to understand properties of SGD itself. Indeed, in full generality, SGD performs variational inference using a new potential Φ that it implicitly constructs given an architecture and a dataset.

It is widely believed that SGD is an implicit regularizer, see Zhang et al. (2016); Neyshabur et al. (2017); Shwartz-Ziv and Tishby (2017) among others. This belief stems from its remarkable empirical performance. Our results show that such intuition is very well-placed. Thanks to the special architecture of deep networks where gradient noise is highly non-isotropic, SGD helps itself to a potential Φ with properties that lead to both generalization and acceleration.

6.4.4.2. SGD and noise: Noise is often added in SGD to improve its behavior around saddle points for non-convex losses, see Lee et al. (2016); Anandkumar and Ge (2016); Ge et al. (2015). It is also quite indispensable for training deep networks (Hinton and Van Camp, 1993; Srivastava et al., 2014; Kingma et al., 2015; Gulcehre et al., 2016a; Achille and Soatto, 2017). There is however a disconnect between these two directions due to the fact that while adding external gradient noise helps in theory, it works poorly in practice (Neelakantan et al., 2015; Chaudhari and Soatto, 2015). Instead, "noise tied to the architecture" works better, e.g., dropout, or small mini-batches. Our results close this gap and show that SGD crucially leverages the highly degenerate noise induced by the architecture.

6.4.4.3. Gradient diversity: Yin et al. (2017) construct a scalar measure of the gradient diversity given by

$$\sum_k \frac{\|\nabla f_k(x)\|}{\|\nabla f(x)\|}$$

and analyze its effect on the maximum allowed batch-size in the context of distributed optimization.

6.4.4.4. Markov Chain Monte Carlo: MCMC methods that sample from a negative log-likelihood $\Phi(x)$ have employed the idea of designing a force $j = \nabla\Phi - \nabla f$ to accelerate convergence, see Ma et al. (2015) for a thorough survey, or Pavliotis (2016); Kaiser et al. (2017) for a rigorous treatment. We instead compute the potential Φ given ∇f and D, which necessitates the use of techniques from physics. In fact, our results show that since $j \neq 0$ for deep networks due to non-isotropic gradient noise, very simple algorithms such as SGLD by Welling and Teh (2011) also benefit from the acceleration that their sophisticated counterparts aim for (Ding et al., 2014; Chen et al., 2015a).

CHAPTER 7

What next?

In this thesis, we have focused on developing an understanding of deep networks from the point-of-view of optimization. The contribution of this thesis is to identify and build upon key phenomena that make deep networks computationally efficient in terms of training and generalize well on new data. While a picture of the many intricacies of deep neural networks is far from complete, we are confident renewed interest in neural networks, and machine learning in general, in the face of stark inadequacies of current technologies will provide the impetus to complete this picture in the coming years.

We have benefited from a combination of diverse techniques spread across statistical physics of learning, combinatorial optimization and constraint satisfaction, Markov chain Monte Carlo, theory of partial differential equations, optimal transportation, Bayesian inference and information theory and non-equilibrium thermodynamics while developing this material. A few key ideas to explore further are as follows.

The continuous-time limit provides powerful tools for the analysis and design of optimization algorithms (Wibisono et al., 2016; Raginsky et al., 2017; Tzen et al., 2018; Weinan et al., 2017) and may lead to new architectures (Ruthotto and Haber, 2018; Li and Shi, 2017) and algorithms for deep learning. Ideas such as underdamped Langevin dynamics from statistical physics for non-convex optimization and mean-field games (Lasry and Lions, 2007), optimal transportation (Mesa et al., 2018) from PDEs for the design of large-scale distributed training algorithms for deep networks.

Non-equilibrium phenomena and their relevance to optimization is only now beginning to be explored and little has been done to exploit these ideas yet. For instance, Markov Chain Monte Carlo samplers on deep networks automatically benefit from these effects (Leimkuhler and Matthews, 2016; Rohrdanz et al., 2011). Arguably, the most pertinent problem in machine learning today is understanding and improving the generalization performance of deep networks Connections of non-equilibrium phenomena to generalization initiated in this thesis may lead to significant advances here.

The diversity of deep architectures, optimization algorithms and regularization techniques is only eclipsed by the problem domains that deep learning seems applicable to. As such, a complete theoretical understanding of all the empirical phenomena seems difficult to achieve—new ones are discovered everyday. The search for "laws of learning": basic principles that are true across a wide variety of everyday situations, becomes even more important. While the success of the previous century was built upon an understanding of the laws of our physical universe, the present century will benefit from laws that govern data and its learning. The author believes that this will be done, not by imposing theoretical models on the learning process, but by identifying empirically testable quantities that sit at the root of this theory.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. 64

Achille, A. and Soatto, S. (2016). Information dropout: learning optimal representations through noise. *arXiv:1611.01353*. 57, 88

Achille, A. and Soatto, S. (2017). On the emergence of invariance and disentangling in deep representations. *arXiv:1706.01350*. 89, 90, 101

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2016). Deep variational information bottleneck. *arXiv:1612.00410*. 88

Allgower, E. and Georg, K. (2012). *Numerical continuation methods: an introduction*, volume 13. Springer. 37

Ambrosio, L., Gigli, N., and Savaré, G. (2008). *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media. 80, 81

Anandkumar, A. and Ge, R. (2016). Efficient approaches for escaping higher order saddle points in non-convex optimization. In *COLT*, pages 81–102. 37, 101

Anderson, J. R. and Peterson, C. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019. 90

Ao, P., Kwon, C., and Qian, H. (2007). On the existence of potential landscape in the evolution of complex systems. *Complexity*, 12(4):19–27. 98

Arora, S., Bhaskara, A., Ge, R., and Ma, T. (2013). Provable bounds for learning some deep representations. *arXiv:1310.6343*. 6, 8

Auffinger, A., Arous, G. B., and Černỳ, J. (2013). Random matrices and complexity of spin glasses. *Communications on Pure and Applied Mathematics*, 66(2):165–201. 5, 9, 10, 11

Baker, B., Gupta, O., Naik, N., and Raskar, R. (2016). Designing neural network architectures using reinforcement learning. *arXiv:1611.02167*. 94

Bakry, D. and Émery, M. (1985). Diffusions hypercontractives. In *Séminaire de Probabilités XIX 1983/84*, pages 177–206. Springer. 40, 42

Baldassi, C., Borgs, C., Chayes, J., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. (2016a). Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *PNAS*, 113(48):E7655–E7662. 23, 29, 37, 46, 48, 66, 100

Baldassi, C., Gerace, F., Lucibello, C., Saglietti, L., and Zecchina, R. (2016b). Learning may need only a few bits of synaptic precision. *Physical Review E*, 93(5):052313. 37

Baldassi, C., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. (2015). Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses. *Physical review letters*, 115(12):128101. 22, 25, 26, 37, 43, 100

Baldassi, C., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. (2016c). Local entropy as a measure for sampling solutions in constraint satisfaction problems. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(2):023301. 25, 32, 43

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58. 36, 43

Bartlett, P. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *JMLR*, 3:463–482. 24

Benamou, J.-D. and Brenier, Y. (2000). A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393. 81

Bertsekas, D., Nedi, A., Ozdaglar, A., et al. (2003). Convex analysis and optimization. 45

Boltzmann, L. (1872). Weitere studien über das wärmegleichgewicht unter gasmolekülen, (Further Studies on the Thermal Equilibrium of Gas Molecules). In *Kinetische Theorie II*. 80

Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research*, 2(Mar):499–526. 24, 30

Bovier, A. and den Hollander, F. (2006). Metastability: a potential theoretic approach. In *International Congress of Mathematicians*, volume 3, pages 499–518. 14, 36, 42

Braunstein, A., Mézard, M., and Zecchina, R. (2005). Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226. 5, 25

Braunstein, A. and Zecchina, R. (2006). Learning by message passing in networks of discrete synapses. *Physical review letters*, 96(3):030201. 25

Bray, A. and Dean, D. (2007). The statistics of critical points of Gaussian fields on large-dimensional spaces. *Physics Review Letter*. 32, 36, 43

Brenier, Y. (1987). Décomposition polaire et réarrangement monotone des champs de vecteurs. *CR Acad. Sci. Paris Sér. I Math.*, 305:805–808. 80

Brenier, Y. (1991). Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417. 80

Brock, A., Lim, T., Ritchie, J., and Weston, N. (2017). SMASH: One-Shot Model Architecture Search through HyperNetworks. *arXiv:1708.05344*. 94

Cai, Z., Fan, Q., Feris, R. S., and Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*. 5

Cannarsa, P. and Sinestrari, C. (2004). *Semiconcave functions, Hamilton-Jacobi equations, and optimal control*, volume 58. Springer Science & Business Media. 44, 51

Carrillo, J. A., McCann, R. J., and Villani, C. (2006). Contractions in the 2-wasserstein length space and thermalization of granular media. *Archive for Rational Mechanics and Analysis*, 179(2):217–263. 42

Chatterjee, S. (2014). *Superconcentration and related topics*. Springer. 11

Chaudhari, P., Baldassi, C., Zecchina, R., Soatto, S., Talwalkar, A., and Oberman, A. (2017). Parle: parallelizing stochastic gradient descent. *arXiv:1707.00424*. 100

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. (2016). Entropy-SGD: biasing gradient descent into wide valleys. In *Proc. of International Conference of Learning and Representations*. 39, 49, 63, 65, 70, 100

Chaudhari, P. and Soatto, S. (2015). On the energy landscape of deep networks. *arXiv:1511.06485*. 36, 43, 101

Chaudhari, P. and Soatto, S. (2017). Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *Proc. of International Conference of Learning and Representations*. 74

Chen, C., Carlson, D., Gan, Z., Li, C., and Carin, L. (2015a). Bridging the gap between stochastic gradient MCMC and stochastic optimization. *arXiv:1512.07962*. 28, 36, 92, 101

Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *ICML*, pages 1683–1691. 28, 36, 92

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015b). Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv:1512.01274*. 64

Chen, X. (2012). Smoothing methods for nonsmooth, nonconvex minimization. *Mathematical programming*, pages 1–29. 43

Chiang, T.-S., Hwang, C.-R., and Sheu, S. (1987). Diffusion for global optimization in rˆn. *SIAM Journal on Control and Optimization*, 25(3):737–753. 43

Choromanska, A., Henaff, M., Mathieu, M., Ben Arous, G., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *AISTATS*. 14, 32, 36, 43

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv:1511.07289*. 21, 33

Cocco, S., Monasson, R., and Zecchina, R. (1996). Analytical and numerical study of internal representations in multilayer neural networks with binary weights. *Physical Review E*, 54(1):717. 38

Cohen, T. S. and Welling, M. (2016). Group equivariant convolutional networks. *arXiv:1602.07576*. 21

Cooijmans, T., Ballas, N., Laurent, C., and Courville, A. (2016). Recurrent batch normalization. *arXiv:1603.09025*. 37

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941. 13, 32, 36, 43

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. (2012). Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231. 63

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R., and Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. In *NIPS*, pages 3203–3211. 36, 101

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. (2017). Sharp minima can generalize for deep nets. *arXiv:1703.04933*. 56

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. 18

Doob, J. (2012). *Classical potential theory and its probabilistic counterpart: Advanced problems*, volume 262. Springer Science & Business Media. 84

Doucet, A., De Freitas, N., and Gordon, N. (2001a). An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer. 90

Doucet, A., de Freitas, N., and Gordon, N. (2001b). Sequential monte carlo methods in practice. Springer. 90

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159. 29, 37

Duchi, J., Jordan, M. I., and McMahan, B. (2013). Estimation, optimization, and parallelism when data is sparse. In *NIPS*. 64

Duvenaud, D., Maclaurin, D., and Adams, R. (2016). Early stopping as non-parametric variational inference. In *AISTATS*, pages 1070–1077. 101

E, W. (2011). *Principles of multiscale modeling*. Cambridge University Press. 46

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499. 12

Engel, A. and Van den Broeck, C. (2001). *Statistical mechanics of learning*. Cambridge University Press. 25

Evans, L. C. (1998). *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society. 43, 44, 45, 51, 52

Fick, A. (1855). Ueber diffusion. *Annalen der Physik*, 170(1):59–86. 78

Fleming, W. H. and Rishel, R. W. (2012). *Deterministic and stochastic optimal control*, volume 1. Springer Science & Business Media. 49

Fleming, W. H. and Soner, H. M. (2006). *Controlled Markov processes and viscosity solutions*, volume 25. Springer Science & Business Media. 41, 49

Frank, T. D. (2005). *Nonlinear Fokker-Planck equations: fundamentals and applications*. Springer Science & Business Media. 85, 86, 98

Freeman, C. D. and Bruna, J. (2016). Topology and geometry of half-rectified network optimization. *arXiv:1611.01540*. 11

Fyodorov, Y. V. (2013). High-dimensional random fields and random matrix theory. *arXiv:1307.2379*. 5, 12, 13, 19

Fyodorov, Y. V. and Williams, I. (2007). Replica symmetry breaking condition exposed by random matrix calculation of landscape complexity. *Journal of Statistical Physics*, 129(5):1081–1116. 13, 36, 43

Gan, Z., Li, C., Chen, C., Pu, Y., Su, Q., and Carin, L. (2016). Scalable Bayesian Learning of Recurrent Neural Networks for Language Modeling. *arXiv:1611.08034*. 36

Gastaldi, X. (2017). Shake-shake regularization. *arXiv:1705.07485*. 70

Ge, R., Huang, F., Jin, C., and Yuan, Y. (2015). Escaping from saddle points — online stochastic gradient for tensor decomposition. In *COLT*, pages 797–842. 37, 101

Gelfand, A. E. and Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409. 90

Geman, S. and Geman, D. (1987). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *Readings in Computer Vision*, pages 564–584. Elsevier. 90

Geman, S. and Hwang, C.-R. (1986). Diffusions for global optimization. *SIAM Journal on Control and Optimization*, 24(5):1031–1043. 43

Ghadimi, S. and Lan, G. (2013). Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368. 41

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv:1302.4389*. 19, 32

Goodfellow, I. J. and Vinyals, O. (2014). Qualitatively characterizing neural network optimization problems. *arXiv:1412.6544*. 37

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv:1706.02677*. 64, 68, 88, 100

Grenander, U. and Miller, M. I. (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 549–603. 90

Gulcehre, C., Moczulski, M., Denil, M., and Bengio, Y. (2016a). Noisy activation functions. In *ICML*, pages 3059–3068. 43, 49, 101

Gulcehre, C., Moczulski, M., Visin, F., and Bengio, Y. (2016b). Mollifying networks. *arXiv:1608.04980*. 37

Haeffele, B. D. and Vidal, R. (2015). Global optimality in tensor factorization, deep learning, and beyond. *arXiv:1506.07540*. 36, 43

Hänggi, P. (1978). On derivations and solutions of master equations and asymptotic representations. *Zeitschrift für Physik B Condensed Matter*, 30(1):85–95. 98

Hanggi, P. (1978). Stochastic-processes. 1. asymptotic-behavior and symmetries. *Helvetica Physica Acta*, 51(2):183–201. 78

Hardt, M., Recht, B., and Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. In *ICML*. 30, 31, 37

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109. 90

Haussler, D. and Opper, M. (1997). Mutual information, metric entropy and cumulative relative entropy risk. *The Annals of Statistics*, 25(6):2451–2492. 37

Hazan, E., Levy, K., and Shalev-Shwartz, S. (2016). On graduated optimization for stochastic non-convex problems. In *ICML*. 38

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. *arXiv:1603.05027*. 43

Heidernätsch, M. (2014). On the diffusion in inhomogeneous systems. 78

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and A, L. (2017). beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework . In *ICLR*. 92

Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM. 90, 101

Hochreiter, S. and Schmidhuber, J. (1997a). Flat minima. *Neural Computation*, 9(1):1–42. 37

Hochreiter, S. and Schmidhuber, J. (1997b). Long short-term memory. *Neural computation*, 9(8):1735–1780. 31

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558. 5

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017). Snapshot ensembles: Train 1, get m for free. *arXiv:1704.00109*. 64, 70

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. (2016). Densely connected convolutional networks. *arXiv:1608.06993*. 70

Huang, H., Wong, K. M., and Kabashima, Y. (2013). Entropy landscape of solutions in the binary perceptron problem. *Journal of Physics A: Mathematical and Theoretical*, 46(37):375002. 25

Huang, M., Malhamé, R. P., Caines, P. E., et al. (2006). Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems*, 6(3):221–252. 50

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*. 19, 32, 37, 57, 69

Jaakkola, T. and Jordan, M. (1997). A variational approach to bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, volume 82, page 4. 90

Jaakkola, T. S. and Jordan, M. I. (1996). Computing upper and lower bounds on likelihoods in intractable networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 340–348. Morgan Kaufmann Publishers Inc. 90

Jagannath, A. (2017). Approximate ultrametricity for random measures and applications to spin glasses. *Communications on Pure and Applied Mathematics*, 70(4):611–664. 11

Janzamin, M., Sedghi, H., and Anandkumar, A. (2015). Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods. *arXiv:1506.08473*. 36

Jaynes, E. T. (1980). The minimum entropy production principle. *Annual Review of Physical Chemistry*, 31(1):579–601. 85

Jin, P. H., Yuan, Q., Iandola, F., and Keutzer, K. (2016). How to scale distributed deep learning? *arXiv:1611.04581*. 64

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998a). An introduction to variational methods for graphical models. In *Learning in graphical models*, pages 105–161. Springer. 90

Jordan, R., Kinderlehrer, D., and Otto, F. (1997). Free energy and the fokker-planck equation. *Physica D: Nonlinear Phenomena*, 107(2-4):265–271. 80, 82

Jordan, R., Kinderlehrer, D., and Otto, F. (1998b). The variational formulation of the Fokker–Planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17. 42, 59

Kaiser, M., Jack, R. L., and Zimmer, J. (2017). Acceleration of convergence to equilibrium in Markov chains by breaking detailed balance. *Journal of Statistical Physics*, 168(2):259–287. 101

Kantorovich, L. (1942). On the transfer of masses (in russian). In *Doklady Akademii Nauk*, volume 37, pages 227–229. 80

Karpathy, A., Johnson, J., and Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. *arXiv:1506.02078*. 35

Kawaguchi, K. (2016). Deep learning without poor local minima. In *NIPS*. 37

Keener, R. W. (2011). *Theoretical statistics: Topics for a core course*. Springer. 88

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv:1609.04836*. 56, 88, 100

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*. 13, 19, 22

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *NIPS*, pages 2575–2583. 57, 101

Klimontovich, Y. L. (1990). Ito, stratonovich and kinetic forms of stochastic equations. *Physica A: Statistical Mechanics and its Applications*, 163(2):515–532. 78

Kloeden, P. E., Platen, E., and Schurz, H. (2012). *Numerical solution of SDE through computer experiments*. Springer Science & Business Media. 75

Koltchinskii, V. and Panchenko, D. (2000). Rademacher processes and bounding the risk of function learning. In *High dimensional probability II*, pages 443–457. Springer. 24

Koltchinskii, V. and Panchenko, D. (2002). Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, pages 1–50. 24

Kramers, H. A. (1940). Brownian motion in a field of force and the diffusion model of chemical reactions. *Physica*, 7(4):284–304. 42

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Master's thesis, Computer Science, University of Toronto. 17, 92

Krizhevsky, A. (2014). One weird trick for parallelizing convolutional neural networks. *arXiv:1404.5997*. 63

Krzakala, F., Mézard, M., Sausset, F., Sun, Y., and Zdeborová, L. (2012). Statistical-physics-based reconstruction in compressed sensing. *Physical Review X*, 2(2):021005. 5

Krzakala, F. and Zdeborová, L. (2009). Hiding quiet solutions in random constraint satisfaction problems. *Physical review letters*, 102(23):238701. 11

Kushner, H. (1987). Asymptotic global behavior for stochastic approximation and diffusions with slowly decreasing noise effects: global minimization via Monte Carlo. *SIAM Journal on Applied Mathematics*, 47(1):169–185. 43

Kwon, C., Ao, P., and Thouless, D. J. (2005). Structure of stochastic dynamics near fixed points. *Proceedings of the National Academy of Sciences of the United States of America*, 102(37):13029–13033. 96

Lasry, J.-M. and Lions, P.-L. (2007). Mean field games. *Japanese Journal of Mathematics*, 2(1):229–260. 50, 102

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. 1

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 17, 19, 22, 69, 92

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016). Gradient descent only converges to minimizers. In *COLT*, pages 1246–1257. 14, 101

Leimkuhler, B. and Matthews, C. (2016). *Molecular dynamics*. Springer. 90, 97, 102

Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., et al. (2008). A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774. 64

Li, C. J., Li, L., Qian, J., and Liu, J.-G. (2017a). Batch size matters: A diffusion approximation framework on nonconvex stochastic gradient descent. *arXiv:1705.07562*. 76

Li, H., Xu, Z., Taylor, G., and Goldstein, T. (2017b). Visualizing the loss landscape of neural nets. *arXiv:1712.09913*. 43

Li, Q., Tai, C., and Weinan, E. (2017c). Stochastic modified equations and adaptive stochastic gradient algorithms. In *ICML*, pages 2101–2110. 41, 49, 77

Li, Z. and Shi, Z. (2017). Deep residual learning and pdes on manifold. *arXiv:1708.05115*. 102

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv:1312.4400*. 21

Loshchilov, I. and Hutter, F. (2016). SGDR: stochastic gradient descent with restarts. *arXiv:1608.03983*. 64

Ma, Y.-A., Chen, T., and Fox, E. (2015). A complete recipe for stochastic gradient MCMC. In *NIPS*, pages 2917–2925. 28, 101

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press. 90

Mandt, S., Hoffman, M., and Blei, D. (2016a). A variational analysis of stochastic gradient algorithms. In *ICML*, pages 354–363. 28, 96, 101

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic Gradient Descent as Approximate Bayesian Inference. *arXiv:1704.04289*. 101

Mandt, S., McInerney, J., Abrol, F., Ranganath, R., and Blei, D. (2016b). Variational tempering. In *AISTATS*, pages 704–712. 92

Marshall, A. W., Olkin, I., and Arnold, B. C. (1979). *Inequalities: theory of majorization and its applications*, volume 143. Springer. 54

McAllester, D. (2013). A pac-bayesian tutorial with a dropout bound. *arXiv:1307.2118*. 24

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. (2016). Communication-efficient learning of deep networks from decentralized data. *arXiv:1602.05629*. 72

Mehta, D., Stariolo, D. A., and Kastner, M. (2013). Energy landscape of the finite-size mean-field 3-spin spherical model. *arXiv:1303.1520*. 11

Mendelson, S. (2002). Rademacher averages and phase transitions in glivenko-cantelli classes. *IEEE transactions on Information Theory*, 48(1):251–263. 24

Mesa, D. A., Tantiongloc, J., Mendoza, M., and Coleman, T. P. (2018). A distributed framework for the construction of transport maps. *arXiv:1801.08454*. 102

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092. 90

Mezard, M. and Montanari, A. (2009). *Information, physics, and computation*. Oxford University Press. 5

Mézard, M., Parisi, G., and Virasoro, M. (1987). *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Company. 90

Milstein, G. N. (1994). *Numerical integration of stochastic differential equations*, volume 313. Springer Science & Business Media. 75

Mobahi, H. (2016). Training Recurrent Neural Networks by Diffusion. *arXiv:1601.04114*. 43, 49

Mobahi, H. and Fisher III, J. (2015). On the link between Gaussian homotopy continuation and convex envelopes. In *Workshop on Energy Minimization Methods in CVPR*, pages 43–56. Springer. 37

Monasson, R. and Zecchina, R. (1995). Weight space structure and internal representations: a direct approach to learning and generalization in multilayer neural networks. *Physical review letters*, 75(12):2432. 38

Monge, G. (1781). Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*. 80

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932. 56

Moreau, J.-J. (1965). Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299. 44

Moritz, P., Nishihara, R., Stoica, I., and Jordan, M. I. (2015). Sparknet: Training deep networks in spark. *arXiv:1511.06051*. 64

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML*. 69

Neal, R. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162. 28, 36

Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. (2015). Adding gradient noise improves learning for very deep networks. *arXiv:1511.06807*. 21, 101

Nerattini, R., Kastner, M., Mehta, D., and Casetti, L. (2013). Exploring the energy landscape of x y models. *Physical Review E*, 87(3):032140. 11

Neyshabur, B., Tomioka, R., Salakhutdinov, R., and Srebro, N. (2017). Geometry of optimization and implicit regularization in deep learning. *arXiv:1705.03071*. 101

Noh, J. D. and Lee, J. (2015). On the steady-state probability distribution of nonequilibrium stochastic systems. *Journal of the Korean Physical Society*, 66(4):544–552. 95

Nouiehed, M. and Razaviyayn, M. (2018). Learning deep models: Critical points and local openness. *arXiv:1803.02968*. 11

Oberman, A. M. (2006). Convergent difference schemes for degenerate elliptic and parabolic equations: Hamilton-Jacobi equations and free boundary problems. *SIAM J. Numer. Anal.*, 44(2):879–895 (electronic). 39

Oksendal, B. (2003). Stochastic differential equations. Springer. 77, 98

Onsager, L. (1931a). Reciprocal relations in irreversible processes. I. *Physical review*, 37(4):405. 85

Onsager, L. (1931b). Reciprocal relations in irreversible processes. II. *Physical review*, 38(12):2265. 85

Ottinger, H. (2005). *Beyond equilibrium thermodynamics*. John Wiley & Sons. 85, 97

Otto, F. (2001). The geometry of dissipative evolution equations: the porous medium equation. 80

Parisi, G. (1981). Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384. 90

Parisi, G. (1988). Statistical field theory. 90

Pavliotis, G. A. (2016). *Stochastic processes and applications*. Springer. 42, 97, 101

Pavliotis, G. A. and Stuart, A. (2008). *Multiscale methods: averaging and homogenization*. Springer Science & Business Media. 40, 46

Pearlmutter, B. A. (1994). Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160. 32

Prigogine, I. (1955). *Thermodynamics of irreversible processes*, volume 404. Thomas. 85

Qi, H., Sparks, E. R., and Talwalkar, A. (2016). Paleo: A performance model for deep neural networks. `https://openreview.net/forum?id=SyVVJ85lg&noteId=SyVVJ85lg`. 64

Qian, H. (2014). The zeroth law of thermodynamics and volume-preserving conservative system in equilibrium with stochastic damping. *Physics Letters A*, 378(7):609–616. 85, 98

Raginsky, M., Rakhlin, A., and Telgarsky, M. (2017). Non-convex learning via Stochastic Gradient Langevin Dynamics: a nonasymptotic analysis. *arXiv:1702.03849*. 102

Recht, B., Re, C., Wright, S., and Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*. 64

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *CVPR*. 64

Risken, H. (1996). The Fokker-Planck Equation. Springer. 42, 77, 78, 98

Robert, C. P. and Casella, G. (2005). Monte carlo statistical methods (springer texts in statistics). 90

Roberts, G. and Stramer, O. (2002). Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357. 28

Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898. 44

Rohrdanz, M. A., Zheng, W., Maggioni, M., and Clementi, C. (2011). Determination of reaction coordinates via locally scaled diffusion map. *The Journal of chemical physics*, 134(12):03B624. 102

Ruthotto, L. and Haber, E. (2018). Deep neural networks motivated by partial differential equations. *arXiv:1804.04272*. 102

Salimans, T. and Kingma, D. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv:1602.07868*. 37

Santambrogio, F. (2015). Optimal transport for applied mathematicians. *Birkäuser, NY*. 40, 42, 45, 48, 80

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*. 37, 43

Schur, I. (1923). Uber eine Klasse von Mittelbildungen mit Anwendungen auf die Determinantentheorie. *Sitzungsberichte der Berliner Mathematischen Gesellschaft*, 22:9–20. 54

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. (2014). 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*. 64

Shi, J., Chen, T., Yuan, R., Yuan, B., and Ao, P. (2012). Relation of a new interpretation of stochastic differential equations to ito process. *Journal of Statistical physics*, 148(3):579–590. 98

Shwartz-Ziv, R. and Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv:1703.00810*. 95, 101

Singh, B., De, S., Zhang, Y., Goldstein, T., and Taylor, G. (2015). Layer-specific adaptive learning rates for deep networks. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 364–368. IEEE. 5

Soudry, D. and Carmon, Y. (2016). No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv:1605.08361*. 37, 43

Springenberg, J., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv:1412.6806*. 20, 21, 31, 32, 58, 61, 72, 92

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958. 37, 57, 69, 101

Stoltz, G., Rousset, M., et al. (2010). *Free energy computations: A mathematical perspective*. World Scientific. 42

Subag, E. (2015). The complexity of spherical p-spin models — A second moment approach. *arXiv:1504.02251*. 16

Subag, E. (2017). The geometry of the gibbs measure of pure spherical spin glasses. *Inventiones mathematicae*, 210(1):135–209. 11

Subag, E. and Zeitouni, O. (2015). The extremal process of critical points of the pure p-spin spherical spin glass model. *arXiv:1509.03098*. 10, 14, 16

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning*, pages 1139–1147. 5, 19, 29, 37

Talagrand, M. (2003). *Spin glasses: A challenge for mathematicians: Cavity and mean field models*, volume 46. Springer Science & Business Media. 5

Talagrand, M. (2010a). Construction of pure states in mean field models for spin glasses. *Probability theory and related fields*, 148(3-4):601–643. 11

Talagrand, M. (2010b). *Mean field models for spin glasses: Volume I: Basic examples*, volume 54. Springer Science & Business Media. 5

Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., and Goldstein, T. (2016). Training neural networks without gradients: A scalable admm approach. In *ICML*. 64

Tel, T., Graham, R., and Hu, G. (1989). Nonequilibrium potentials and their power-series expansions. *Physical Review A*, 40(7):4065. 98

Tieleman, T. and Hinton, G. (2012). Lecture 6.5: RmsProp, Coursera: Neural networks for machine learning. Technical report. 37

Tishby, N., Pereira, F. C., and Bialek, W. (1999). The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377. 88

Tzen, B., Liang, T., and Raginsky, M. (2018). Local optimality and generalization guarantees for the langevin algorithm via empirical metastability. *arXiv:1802.06439*. 102

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85. 18

Vapnik, V. (1998). *Statistical learning theory.*, volume 3. Wiley. 24

Venturi, L., Bandeira, A., and Bruna, J. (2018). Neural networks with finite intrinsic dimension have no spurious valleys. *arXiv:1802.06384*. 11

Ver Steeg, G. and Galstyan, A. (2014). Discovering structure in high-dimensional data through correlation explanation. In *Advances in Neural Information Processing Systems*, pages 577–585. 89

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media. 80, 81

Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305. 90

Wang, J., Xu, L., and Wang, E. (2008). Potential landscape and flux framework of nonequilibrium networks: robustness, dissipation, and coherence of biochemical oscillations. *Proceedings of the National Academy of Sciences*, 105(34):12271–12276. 98

Wasserman, L. (2013). *All of statistics: A concise course in statistical inference*. Springer. 31

Weinan, E., Han, J., and Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380. 102

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, pages 681–688. 28, 36, 65, 101

Wibisono, A., Wilson, A. C., and Jordan, M. I. (2016). A variational perspective on accelerated methods in optimization. *PNAS*, page 201614734. 102

Wu, R., Yan, S., Shan, Y., Dang, Q., and Sun, G. (2015). Deep image: Scaling up image recognition. *arXiv:1501.02876*. 64

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Generalized belief propagation. In *Advances in neural information processing systems*, pages 689–695. 90

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. (2017). Gradient diversity empowers distributed learning. *arXiv:1706.05699*. 101

Yin, L. and Ao, P. (2006). Existence and construction of dynamical potential in nonequilibrium processes without detailed balance. *Journal of Physics A: Mathematical and General*, 39(27):8593. 99

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv:1605.07146*. 69, 70, 71

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv:1409.2329*. 34

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv:1611.03530*. 37, 101

Zhang, S., Choromanska, A., and LeCun, Y. (2015a). Deep learning with elastic averaging SGD. In *NIPS*. 37, 40, 46, 47, 60, 63

Zhang, W., Gupta, S., Lian, X., and Liu, J. (2015b). Staleness-aware Async-SGD for distributed deep learning. *arXiv:1511.05950*. 64

Zhu, X.-M., Yin, L., Hood, L., and Ao, P. (2004). Calculating biological behaviors of epigenetic states in the phage λ life cycle. *Functional & integrative genomics*, 4(3):188–195. 98

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv:1611.01578*. 94