OPTIMIZATION: STOCHASTIC THERMODYNAMICS, MACHINE LEARNING, AND NUMERICAL ALGORITHMS

by

Neha Spenta Wadia

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Biophysics

in the

Graduate Division

of the

University of California, Berkeley

Spring 2022

OPTIMIZATION: STOCHASTIC THERMODYNAMICS, MACHINE LEARNING, AND NUMERICAL ALGORITHMS

Abstract

OPTIMIZATION: STOCHASTIC THERMODYNAMICS, MACHINE LEARNING, AND NUMERICAL ALGORITHMS

by

Neha Spenta Wadia

Doctor of Philosophy in Biophysics

University of California, Berkeley

Professor Michael I. Jordan, Co-chair

Professor Michael R. DeWeese, Co-chair

Optimization is a natural language in which to express a multitude of problems from all reaches of the mathematical sciences. This is a dissertation in three parts, detailing work on three different problems in optimization.

We begin in stochastic thermodynamics, with the problem of computing optimal operating protocols for Brownian machines by minimizing their dissipation on average. We develop a perturbative solution to the Fokker-Planck equation for one-dimensional driven Brownian motion in the limit of slow driving. Calculating the average dissipation in this formalism, we demonstrate the existence of an emergent Riemannian geometric structure in the space of control parameters, whereby the n^{th} order term in the average dissipation contains a tensor of order n. We rigorously derive an exact formula for the order-2 tensor (the metric) and show that up to second-order terms in the perturbation theory, optimal dissipation-minimizing driving protocols minimize the length defined by this metric.

Next, we move to the world of machine learning, and study the generalization properties of models trained with first-order optimizers on whitened data, and with exact second-order optimizers. For a general class of models trained with first-order optimizers, we find that the mutual information between the trained model and the dataset is strictly smaller when the data is whitened than when it isn't. This reduction of access to information relevant for test performance negatively impacts generalization in a dimension-dependent manner, worsening for high dimensional problems. Exploiting analogies between training linear models and wide neural networks with a first-order optimizer on whitened data and with an exact second-order optimizer, we also establish the existence of a generalization gap between first- and exact second-order optimizers in these model classes. Experiments demonstrate that our conclusions hold in a larger class of models than is supported by the theoretical results, but that regularizing a second-order optimizer appropriately can compensate for the reduction in generalization performance observed in the exact setting.

Finally, we study the problem of setting step sizes for discrete-time gradient-based optimization. Taking the view that a discrete-time optimizer is a simulation of an underlying ordinary differential equation (ODE), we set step sizes for the former by controlling the error with which it tracks the latter. The resulting adaptive step size mechanism is highly efficient, requiring just one extra gradient call over all iterations, and applicable to any optimizer with a well-defined continuous-time representation as an ODE. We find empirically that this step size routine can achieve oracle lower bounds on strongly convex functions for first- and second-order optimizers, and that it can also perform competitively with the state of the art on a nonconvex problem, principal components analysis.

To curiosity.

Contents

List of Figures

Acknowledgments

It takes a village to get a PhD, and I am profoundly grateful to mine.

I must begin by thanking my advisors Mike Jordan (MJ) and Mike DeWeese (MD). MD's own cheerful enthusiasm for science and his relentless encouragement to me to work on the problems that "get me out of bed in the morning" made it feel natural to think about problems in a variety of different fields in grad school. Similarly, MJ's insistence that I must "eat [my] meat and potatoes" before anything else gave me the courage to slow down (so it seemed at the time) in the middle of my PhD and devote months to acquiring baseline technical skills in statistics and optimization - the meat and potatoes. Over the years, I have repeatedly been floored by and grateful for MJ's constant support of my development as a researcher. Both Mikes have always been generous with their time, and sources of calm, good, sensible advice. (Idealists like myself are often in need of such.)

I thank Jascha Sohl-Dickstein for mentoring me through the years of my Google PhD Fellowship and since, and for hosting me during my internship at Google Brain in 2019. I thank him for insisting that I ask questions, for fixing my code (and therefore teaching me how to write better code), and for teaching me how to do research in the neural network community. Similarly, I thank Ethan Dyer for functioning as a second mentor at Google. I thank Sam Schoenholz and Daniel Duckworth, also collaborators on our joint work. It was a privilege to learn from you and build experiments with you.

I have been fortunate to have been mentored by a number of postdocs at Berkeley - Dibyendu Mandal, Michael Muehlebach, and Gui França. Dibyendu introduced me to out-of-equilibrium statistical mechanics. Michael and Gui taught me (theory of) optimization and (continue to) cheerfully put up with my erratic work schedule. Michael's astonishing discipline, Gui's effortless creativity, and Dibyendu's understated mastery were and continue to be inspirational.

I learn from everyone I work with. I thank all my other collaborators not already mentioned - Ryan Zarcone, Charles Frye, Kris Bouchard, Andrew Ligeralde, and James Simon. Ryan, in particular - thank you for being my minus-sign catcher!

A nontrivial number of faculty both at Berkeley and at other institutions have supported me and my professional development over the years in various ways. I thank Mike Eisen and Michael Yartsev, whose labs I rotated in in my first year; Bruno Olshausen, Dan Rokhsar, Bin Yu, Marla Feller, and Peter Bartlett, my qualifying exam and dissertation committee members; Jim Hurley and John Wilkening, who facilitated my applications to various graduate fellowships; Mukund Thattai, David Cory, and William Loinaz - it's thanks to you I went to Berkeley; Larry Hunter and Steve Peck - your approach to experimental work informs my thinking still; Patrick Williamson, Bill Bialek, and Anirvan Sengupta for their advice; and Satya Majumdar, Florent Krzakala, Lenka Zdeborova, Jennifer Chayes, and Gavin Crooks.

It has been a profound privilege to be part of the two groups in which I have spent most of my time at Berkeley - the Redwood Center, and the SAIL group. I thank Bruno Olshausen for building and maintaining the former, and MJ for the latter. Both these groups are packed with kind, generous, and outstandingly talented students and postdocs from whom I have learned most of what I learned in graduate school. This, I think, is the privilege of spending time at Berkeley: My peer groups are so stellar that if even the teensiest bit of their talent has rubbed off on me, I'm in good shape. I thank them all.

I have several senior students and postdocs to thank specifically for their guidance, especially during the first years of graduate school and in the last one. In SAIL: Ahmed El Alaoui, Horia Mania, Ashia Wilson, Stephen Bates, Adam Sealfon, Lydia Liu, and Nilesh Tripuraneni. At the Redwood Center: Jesse Livezey, Mayur Mudigonda, Dylan Paiton, Yubei Chen, Eric Weiss, and Shariq Mobin. And in the Biophysics Graduate Group: Jasmine Nirody, Mike Levy, Eric Bolin, and Chris Habrian. I also thank my officemates in Evans for a great time - Louis Kang, Steven Lee, Mayur Mudigonda, and Jesse Livezey - and my floormates in SDH7 - Lydia Liu, Adam Sealfon, Meena Jagadeesan, and Stephen Bates.

My favorite part of being in SAIL was and continues to be the probability reading group consisting of myself, Anastasios Angelopoulos, Reese Pathak, and Mariel Werner. Meeting weekly, it took us about two years to fight our way through Concentration Inequalities by Boucheron, Lugosi, and Massart. Now that we have finished the book we finally feel ready to read it. Anastasios, Reese, and Mariel, thank you for teaching me probability theory, and I look forward to continuing to read and learn with you in the coming years.

Of all the courses I took at Berkeley, Math 205A, Stat 210A, and Stat 210B were the hardest by far. I wouldn't have got through them without my homework buddies - Anastasios Angelopoulos, Akosua Busia, Milind Hegde, Clara Wong-Fannjiang, Jeffrey Epstein, Mike Fang, and Ella Hiesmayr - and the graduate student instructors - Xiao Li, Chiao-Yu Yang, Nilesh Tripuraneni, and Jake Soloff.

I thank Kate Chase, the administrative backbone of the Biophysics program, for her assistance in all things administrative over the years. I'm very grateful. I also thank Naomi Yamasaki, Catherine Dea, Sarah Hamilton, and Nanette Hara in the same vein.

I thank all the many members of the campus community who make Berkeley the special place it is. Shout out to the baristas at the Stanley Hall Yali's, who pull the best espressos on campus, and the baristas at the V&A (formerly Nefeli) Cafe, who make the best hot chocolate, and the librarians at the Math Library. I thank Mary Popylisen, Jeff Chen, and Matthew Garet, physical therapists at the Tang Center, and Bianca Rodriguez at the RSF, for making me stronger.

Where would we all be without our friends? Vasha DuTell, Dar Dahlen, Mike Fang, Jesse Livezey, Sarah Maslov, Mindy Perkins, Ryan Zarcone, Lester Hu, Danielle Ramos, Ruqian Chen, Shenglan Qiao, Matt Fernald, Liz Lawler, Heidi Kleven, Ipsita Agarwal, Maggie Huang, June Pan, Mugdha Sathe, Divya Anjan, Amy Ko, Jeff Spence, and Anne-Marie Bonnel, and many others. Thank you for all the good times, for your care of me, and for your faith in me. And the Friday Night Dinner crew - Aadi Ramdas, Leila Wehbe, Mayur Mudigonda, Vish Banarasi, Siva Bandaru, Madhyama Subramanian, and Divya Anjan, thank you for the laughter and the yummies.

Lastly, I thank all my family. I thank my aunt Anitha for her advice and perspective, and, along with my uncle Ramesh and my grandmother Jayashree, I thank her for spoiling me at every opportunity. I thank my brother Varun - you may be my "little" brother, but you have taught me no small lessons about how to live life. And I thank my parents, Leena and Spenta. Words fail here. Thank you for teaching me to live a life of the mind, for establishing the value system that I rely on in my decision-making every day, and for holding my hand all these many years without question or complaint. I particularly thank my Dad, Spenta, my first and longest-continuing (i.e., longest-suffering) teacher in science. It was he who taught me most of the physics I know, and how to proceed when faced with a blank sheet of paper, a pencil, and a question. And I thank my Mom, for putting her foot down once in a while and making both of us do other things, for staying up with me all night once until I submitted a paper, and who, along with all the women in science of her generation and previous ones, paved my way in this profession.

I end these acknowledgements with an apology to anyone I may have missed here.

Chapter 1

Introduction

Optimization is an indispensable tool that is operative throughout the mathematical sciences. It finds use both at the highest levels of abstraction as an organizing principle, and as a practical methodology for problem solving. An example of the former type is the variational formulation of the laws of gravity, mechanics, and electromagnetism. Through the *action principle*, these laws emerge as the extrema of functionals called *actions*. On the other end of the spectrum of abstraction we have, for example, the standard method of performing a linear fit to a dataset, which is to minimize the sum of squared distances between each data point and a candidate line.

In a small way, this thesis is a celebration of the versatility and ubiquity of optimization. In it, we describe work on three different optimization problems - one in physics (specifically, in thermodynamics), one in machine learning, and one in the engineering discipline known as optimization. In what follows, with a general graduate-level audience in the mathematical sciences in mind, we introduce the problems addressed in subsequent chapters and provide summaries of results.

All the work presented herein is based on references [138], [139], and [140].

Physics, machine learning, and optimization

Optimal protocols for Brownian machines

Extracting work from a machine (extracting forward thrust from a car engine, for example) always comes at the cost of generating waste heat (car engines typically have coolants to absorb this waste heat and maintain a reasonable temperature). The centuries-old laws of thermodynamics characterize this fundamental trade-off, and enable the design of operating protocols that maximize the extraction of useful work while minimizing the release of waste heat, or dissipation. Today, in order to understand the operating principles of *microscopic* machines, such as biomolecules, there is an ongoing research effort to miniaturize the laws of (macroscopic) thermodynamics. At the relevant length scales, where the role of thermal noise is significant, these machines take on probabilistic descriptions, and the theoretical challenges begin at the most fundamental level with the question of how to define quantities such as work, heat, and entropy for a single stochastic trajectory [117]. The subfield of physics in which these problems are dealt with is called stochastic thermodynamics.

In macroscopic thermodynamics, the state of a machine is specified by the values of its external *control* parameters. These parameters are manipulated according to an *operating protocol* in order to extract work. For example, the control parameters of a steam engine are the temperature and volume of the steam. These are manipulated in a cyclic fashion in order to extract work from the engine, say, to drive to a piston or rotate a wheel. The operating protocol that maximizes the extraction of useful work from a steam engine while minimizing dissipation is the celebrated Carnot cycle [36] (for a modern description in English see, for example, Chapter 4 of [115]).[1]

Similarly, in stochastic thermodynamics, a machine can be abstracted as a Brownian motion that depends on a set of tunable external parameters. Since the control parameters vary with time, we say the Brownian motion is *driven.* This abstraction is not merely a theoretical construct. Brownian engines exist [89]. Indeed, this is an area of science in which theory trails experiment. All the more reason, then, to ask: what is the analog of the Carnot cycle for a Brownian engine? Directly inspired by this and similar questions, in Chapter 2 we study the problem of computing dissipation-minimizing *optimal* operating protocols for driven Brownian systems. The problem is formulated as follows: given two points in the space of control parameters of a Brownian system, what is the path between the them that minimizes dissipation on average?

For arbitrary driving dynamics, it is known that this optimization problem is intractable [113]. But in a remarkable line of work in 2012, within a specific driving regime known as the linear response regime, it was demonstrated that the generally intractable problem of minimizing dissipation approximately reduces to the relatively tractable problem of computing geodesics with respect to a Riemannian metric in the space of control parameters of the Brownian system [127, 155]. Some questions follow. Linear response is a theoretical tool. That is, restricting to the linear response regime on paper does not provide a prescription by which an experimentalist can maintain a physical system in that regime. So what is the driving dynamics in which this reduction to a geometric problem holds? Second, while the results of [127, 155] hint at the existence of geometric structure in optimal control problems in stochastic thermodynamics, it was not clear how firm this link to geometry is. Does it hold only approximately, or in an exact sense, and if the latter, what precisely is that sense? In Chapter 2, we provide answers to both these questions.

One of the reasons the optimal control problem for driven Brownian systems is hard is that the probability density of the system is a time-dependent nonequilibrium density (i.e., it is *not* given by the Gibbs measure) to which we typically do not have access. Therefore it is usually not even possible to compute the average dissipation, let alone minimize it. To solve

[1]It should be noted that the Carnot cycle is not a practical operating protocol, because it requires thermodynamic reversibility. However, it provides an upper bound on the efficiency of a certain class of engines, and serves as both a basis and limiting case for practical operating protocols.

this problem, we developed a rigorous method of perturbatively solving the Fokker-Planck equation of a driven one-dimensional Brownian motion, enabled by the spectral properties of the corresponding Schrödinger operator. The perturbative expansion is in a small parameter ν which is the ratio of the system timescale to the characteristic timescale of variation of the fastest-varying control parameter. That is to say, we work in a regime where the system relaxes faster than it is driven. When we computed the average dissipation in the perturbative formalism, we found that each term acquires a tensor structure, revealing an emergent Riemannian structure in the space of control parameters. The geometric structure of the optimal control problem therefore derives from the specific form of the perturbative expansion for the probability density, and persists to all orders of the expansion. The second-order term gives a formula for the Riemannian metric in this space. Up to second order in ν, optimal driving protocols are then characterized by geodesics corresponding to this metric. Thus, up to second order in ν, the result of [127, 155] is *exact*, and not approximate.

The discussion in Chapter 2 is based on joint work [139] with Ryan Zarcone and Michael DeWeese.

Generalization properties of exact second-order optimizers by analogy with data whitening

The canonical problem of machine learning can be stated as follows: given a dataset, learn a function over the data that will perform well (by some metric) when it is evaluated on or used to predict new data. The form of this problem is forced by the fact that we do not usually have knowledge of the data generating process. If we did, we could simply "learn a function over the data" as opposed to learning a function over the *available* ("training") data that will *generalize* to unseen ("test") data. Optimization is the core tool by which we solve this proxy problem over the training data, usually by extremizing a performance metric of some kind. Canonical examples are a cost function over the data (in which case we minimize) or a reward function over actions (in which case we maximize). An important open problem in machine learning is to understand the relationship between the choice of optimizer and the generalization properties of the solutions it finds.

The most widely used optimizers fall roughly into two families, first-order or second-order, so named for the number of derivatives (one or two) of the cost or reward function they require access to. There is some debate in the field on the relative merits of these families, especially with respect to computational complexity and generalization performance. Second-order optimizers are more expensive per iteration, since they require the computation of the second derivative of the cost function, but they also converge in fewer iterations than first-order optimizers, so they could well have lower overall computational complexity in specific problems. Interestingly, there is evidence to suggest that the solutions recovered by second-order optimizers do not always generalize as well as those recovered by first-order optimizers [144]. For this reason, understanding the generalization properties of second-order optimizers is a highly active area of research. In Chapter 3, we discuss some work in this area.

One of our main results is the following: in the high dimensional setting, where the size of the dataset is smaller than the dimensionality of an individual data point, exact second-order optimizers produce models that either cannot generalize or generalize poorly. However, approximate second-order optimizers may not exhibit this pathology. This result is unexpected, if not downright *odd*, for reasons we now describe.

Computing the second derivative of the cost function at each iteration can often be cost-prohibitive both in terms of wall-clock time and processor power. Therefore, when implementing a second-order optimizer in a large-scale machine learning problem, the second derivative is typically approximated in some way that reduces the computational cost of each iteration. These approximations are usually understood as necessary compromises - ideally, it would be better to compute the full second derivative at each iteration. However, in the context of machine learning (as opposed to pure optimization), where we ultimately care about generalization, the main result of Chapter 3 suggests otherwise! The result implies that even if we could compute the full second derivative of the cost function at each iteration during optimization, it may be better not to do so, because the resulting models will have poorer generalization properties than if they had been trained with an approximate second-order optimizer.

Specifically, in Chapter 3, we show that the mutual information between the trained model and the dataset is strictly smaller if the model is trained with an exact second-order method as opposed to a first-order method. Therefore, models trained with exact second-order optimizers are expected to have reduced generalization performance. The result holds in theory for any function that includes a dense matrix multiplication, but we found experimentally that it also holds for more general function classes such as convolutional neural networks. We also show that the drop in generalization caused by training with a second-order optimizer has a dimension dependence, worsening for high dimensional problems. The theoretical results follow from an analogy between exact second-order optimization and whitening, which is a data preprocessing transform that removes second-order correlations in the dataset.

The discussion in Chapter 3 is based on joint work [140] with Jascha Sohl-Dickstein, Ethan Dyer, Sam Schoenholz, and Daniel Duckworth.

Efficient adaptive step size methods for gradient-based optimization

There is a subdiscipline of computer science and electrical engineering called optimization, in which the latter is studied not as a means to an end but as an end in itself. The basic problem is the following - given a function $f(x)$, find either its minimum value, or the argument $x^\star$ that minimizes it, or both.[2] It is understood that the solution to this problem will be implemented on a computer, and so it is natural to think in discrete time.

Beginning with some initial point x^0, an optimizer or optimization algorithm gives a prescription by which to generate a sequence $\{x^0, x^1, x^2, ...\}$, and if this sequence is constructed correctly we can guarantee that the distance between the iterate x^j and the optimum $x^\star$

[2]We will be concerned here with the case $x \in \mathbb{R}^d$.

decreases with iteration number, and goes to zero as j tends to infinity. That is, the sequence *converges* to $x^\star$. The rate at which the distance between x^j and $x^\star$ goes to zero is the *convergence rate* of the optimizer that generated the sequence.

To arrive at x^{j+1} from x^j, the following two pieces of information are required: a direction in which to step, and a step length or step size. Despite this, choosing an optimizer only provides a prescription for computing the first of these. It usually falls to the practitioner to make the additional choice of a step size routine in a problem-specific manner. Let us look at the example of gradient descent,

$$x^{j+1} = x^j - \eta^j \, \nabla f\left(x^j\right), \tag{1.1}$$

in which the update to x^j is given by the product of the gradient of f evaluated at x^j and the j^{th} step size η^j. Only the computation of the update direction is specified by Eq. 1.1. The choice of η^j in Eq. 1.1 is crucial, and can make the difference between convergence, divergence, and other behaviors. Consider, for example, the function $f(x) = x^2$. It has a single optimum at $x^\star = 0$. For any $x^j = y$, applying repeated gradient descent updates with a step size of one will result in the iterate oscillating back and forth between $-y$ and y. Thus the distance between x^j and the optimum $x^\star$ is constant and the algorithm never converges. On the other hand, if at $x^j = y$, we apply the gradient descent update with a step size of $1/2$, then the algorithm converges to $x^\star$ in a single step. Lastly, applying step sizes larger than one will cause the iterate sequence to move away from the optimum and diverge. As illustrated by this example, in general, any prescription for choosing step sizes must negotiate the following mutually opposed requirements: the step size must be large enough so that the algorithm converges in a reasonable number of steps, but small enough to avoid oscillatory or divergent behavior. Methods that modulate the step size at each iteration to meet these requirements are called *adaptive* (as opposed to *constant* step size methods).

Existing adaptive step size methods can be computationally expensive. For example, linesearch is an important method for step size computation in which we search along the next update direction for the step size that approximately maximizes the drop in function value. [There are many details hidden in this description, see, for example, Chapter 9 of [28].] Depending on the specifics of how this is accomplished, linesearch can add significant computational overhead to an optimization routine.

In Chapter 4, we present an efficient alternative to existing adaptive step size methods, inspired by techniques in the numerical analysis of ordinary differential equations. Taking the viewpoint that a discrete-time algorithm is merely a particular discretization of an underlying continuous-time dynamical system, we use a control mechanism to set step sizes for the discrete-time algorithm by controlling the error with which it follows the continuous-time trajectory. This method can be generalized to any discrete-time algorithm with a well-defined continuous-time representation, and, if implemented appropriately, can be surprisingly computationally efficient, requiring exactly *one* extra gradient evaluation over all iterations.

In the complexity theory of optimization, an important class of results called *lower bounds* [93] characterizes the maximal possible rate of convergence of a specific family of optimizers on a specific function class. Let us continue with the example of gradient descent on a quadratic function to unpack this. Quadratic functions are the canonical members of the class of strongly convex functions, and gradient descent is a member of the family of first-order optimizers. The corresponding lower bound puts a limit on how fast *any* first-order optimizer can converge on the class of strongly convex functions. The choice of step size for a specific optimizer can determine whether or not it achieves this lower bound.

We find empirically that the step size routine we develop in Chapter 4 *can* achieve the discrete-time oracle lower bounds for first and second-order optimizers on strongly convex functions, and that it can actually improve the constant in the lower bound compared to constant step size schemes.[3] Furthermore, the step size mechanism is not limited - either in implementation or in good performance - to strongly convex functions. We find, for example, that it also performs competitively with state of the art step size schemes on principal components analysis - a nonconvex problem.

The discussion in Chapter 4 is based on joint work [138] with Michael Muehlebach and Michael Jordan.

[3]Lower bounds typically consist of a constant multiplied by a rate of convergence. These are distinguished by the fact that the latter depends on problem-specific quantities such as the curvature of the function being optimized, and the former does not. Theorems establishing lower bounds usually concentrate on exactly characterizing the rate of convergence, and not the constant.

Chapter 2

Optimal protocols for driven Brownian systems

Driven Brownian motion is a paradigmatic model for a certain class of small (micrometer sized and smaller) stochastic machines [117]. The hallmark of these systems is that quantities such as work and efficiency fluctuate, and are comparable in scale to thermal fluctuations. Their study, stochastic thermodynamics [119], has seen remarkable recent experimental progress, including, for example, the implementation of microscopic single-particle heat engines [24, 89], and much theoretical activity. See [134] for a collection of recent work.

A fundamental problem in stochastic thermodynamics is to understand how small systems do useful work while operating out of equilibrium. A natural framing of this problem is in terms of a notion of optimality out of equilibrium, whereby a system is considered optimal if it minimizes irreversible heat loss to the reservoir on average. Optimal driving protocols can therefore be computed by minimizing the average dissipation over protocols. In general this is a nontrivial optimization problem to solve [113].

The introduction of the thermodynamic metric framework [37, 127] simplified the problem for a restricted class of systems by recasting it in a geometric picture in which the average dissipation is proportional to a measure of length in the space of control parameters of the system. The "length" is defined by a Riemannian metric on this space. An optimal protocol between two points in control space is then given by the minimum of this length, which is generally easier to compute than solutions to the original optimization problem. This framework is a generalization to mesoscale, out-of equilibrium systems of geometrical approaches originally developed for macroscale, endoreversible systems [141, 107, 32, 109, 108, 112, 30].

Since its introduction, the thermodynamic metric framework has found success in predicting optimal protocols for a number of systems both analytically and numerically [155, 154, 105, 126, 106], and in illuminating their general characteristics, opening up a window onto the physics of small machines that operate out of equilibrium.

The concept of a thermodynamic geometry at mesoscopic length scales emerges independently from various different assumptions about the dynamics of the stochastic system.

All these approximations have in common a notion of closeness to equilibrium. In the original work, the approximations were linear response plus slow driving [127]. Subsequent work derived a thermodynamic metric under approximations of derivative truncation [155], and timescale separation [106]. Slow driving was also assumed in order to extend the thermodynamic metric framework to driven discrete-time systems [85].

We provide a new and rigorous derivation of a thermodynamic metric within the framework of the Fokker-Planck equation for Brownian motion with time-varying control parameters. We work in a regime in which the control parameters vary on a timescale that is much longer than the intrinsic timescale of the system, which is set by its relaxation time. The solution to the time-dependent Fokker-Planck equation is obtained as an expansion in a small dimensionless parameter ν that is the ratio of the relaxation time of the system to the the shortest characteristic timescale of variation among the control parameters. The expansion is enabled by the spectral properties of the corresponding Schrödinger operator. The formula for the thermodynamic metric we derive in this framework is exact and has a generalization to higher dimensions.

In addition, we demonstrate an emergent diffeomorphism symmetry in the space of control parameters arising from the expansion in ν of the probability density. Under this symmetry, every term with n indices in the corresponding expansion for the average dissipation is a rank n tensor.

The harmonic potential is a canonical system to study in stochastic thermodynamics, both experimentally and theoretically. For this reason we illustrate our formalism and formulae using the example of a harmonic oscillator with a time-varying spring constant in a time-varying electric field.

2.1 Driven Brownian motion

Consider a small system in contact with a reservoir such as a Brownian particle in a suspension subject to an external potential $V_{\boldsymbol{\lambda}(t)}(x)$ that can depend on a possibly time-dependent control vector $\boldsymbol{\lambda} \in \mathbb{R}^k$. The space $\mathcal{C}$ of all possible values of $\boldsymbol{\lambda}$ is a subset of $\mathbb{R}^k$. The position of the particle is given by $x \in \mathbb{R}$ and its probability density $\rho(x;t)$ evolves according to a Fokker-Planck equation [101],

$$\frac{\partial}{\partial t}\rho(x;t) = \hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}(x)\rho(x;t), \tag{2.1}$$

where $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}(x)$, the Fokker-Planck operator, is a second-order differential operator involving spatial derivatives of the potential. In the overdamped limit, where inertial effects are neglected, $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}(x)$ takes the form

$$\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}(x) = \frac{1}{\gamma}\frac{\partial}{\partial x}\left(V'_{\boldsymbol{\lambda}(t)}(x) + \frac{1}{\beta}\frac{\partial}{\partial x}\right), \tag{2.2}$$

where γ and $\beta = 1/k_BT$ are the friction coefficient and inverse temperature, respectively, and k_B is Boltzmann's constant. Primes denote derivatives with respect to x. Note that

$V'_{\boldsymbol{\lambda}(t)}(x) = -F(x;t)$ where F is the force acting on the system. We consider natural boundary conditions, requiring $\rho(x;t) \to 0$ as $x \to \pm\infty$. $\rho(x;t)$ satisfies the normalization condition

$$\int dx\, \rho(x;t) = 1. \tag{2.3}$$

We use the notation $\int dx$ as shorthand for $\int_{-\infty}^{\infty} dx$.

Eq. 2.1 can also be written in the form of a continuity equation as

$$\frac{\partial}{\partial t}\rho(x;t) = -\frac{\partial}{\partial x}J(x;t), \tag{2.4}$$

where J is the probability current,

$$J(x;t) = -\frac{1}{\gamma}\left(V'_{\boldsymbol{\lambda}(t)}(x) + \frac{1}{\beta}\frac{\partial}{\partial x}\right)\rho(x;t). \tag{2.5}$$

Natural boundary conditions additionally require $J(x;t) \to 0$ as $x \to \pm\infty$.

We note that Eq. 2.1 with $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ as given in Eq. 2.2 is equivalent to the trajectory-level Langevin description

$$\gamma\dot{x} = F(x,t) + \sqrt{\frac{2\gamma}{\beta}}\eta(t), \tag{2.6}$$

where $\eta(t)$ is mean zero δ-correlated Gaussian noise: $\langle\eta(t)\rangle = 0$, $\langle\eta(t)\eta(t')\rangle = \delta(t-t')$. The dot denotes a derivative with respect to time.

At all times, the state space admits the existence of a unique equilibrium distribution $\rho^{eq}_{\boldsymbol{\lambda}(t)}(x)$ such that

$$\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}(x)\rho^{eq}_{\boldsymbol{\lambda}(t)}(x) = 0 \tag{2.7}$$

and

$$\int dx\, \rho^{eq}_{\boldsymbol{\lambda}(t)}(x) = 1. \tag{2.8}$$

$\rho^{eq}_{\boldsymbol{\lambda}(t)}(x)$ is given by

$$\rho^{eq}_{\boldsymbol{\lambda}(t)}(x) = \frac{1}{Z(t)}e^{-\beta V_{\boldsymbol{\lambda}(t)}(x)}, \tag{2.9}$$

where $Z(t)$ is the partition function,

$$Z(t) = \int dx\, e^{-\beta V_{\boldsymbol{\lambda}(t)}(x)}. \tag{2.10}$$

All distributions approach $\rho^{eq}_{\boldsymbol{\lambda}(t)}(x)$ asymptotically with time when $\boldsymbol{\lambda}$ is frozen, and $\rho^{eq}_{\boldsymbol{\lambda}(t)}$ satisfies the detailed balance condition, which requires that the probability current in equilibrium be zero:

$$-\frac{1}{\gamma}\left(V'_{\boldsymbol{\lambda}(t)}(x) + \frac{1}{\beta}\frac{\partial}{\partial x}\right)\rho^{eq}_{\boldsymbol{\lambda}(t)}(x) = 0\ \forall x. \tag{2.11}$$

$\rho^{eq}_{\boldsymbol{\lambda}(t)}$ does not satisfy Eq. 2.1 except in an approximate sense. While Eq. 2.7 is exact, the time derivative of $\rho^{eq}_{\boldsymbol{\lambda}(t)}$ is

$$\frac{\partial}{\partial t}\rho^{eq}_{\boldsymbol{\lambda}(t)}(x) = \sum_{i=1}^{k} \dot{\lambda}_i \frac{\partial}{\partial \lambda_i}\rho^{eq}_{\boldsymbol{\lambda}(t)}(x), \tag{2.12}$$

which is not zero if $\dot{\lambda}_i \neq 0$. The solution to Eq. 2.1 we develop in the following is in the limit of small $\dot{\boldsymbol{\lambda}}$. We will show that the "smallness" of $\dot{\boldsymbol{\lambda}}$ is quantified by a parameter ν, defined as the ratio of the relaxation time τ_{α_1} of the system to the driving timescale τ_λ, which must be chosen such that $\nu \ll 1$. In this limit, the timescale of driving is so long that $\rho^{eq}_{\boldsymbol{\lambda}(t)}$ is roughly stationary on the system timescale, which is set by τ_{α_1}. Thus $\rho^{eq}_{\boldsymbol{\lambda}(t)}$ satisfies Eq. 2.1 to zeroth order in the parameter ν. We return in detail to these ideas in Section 2.1.

We solve Eq. 2.1 using the method of Green's functions. The difficulty in this program is that the Fokker-Planck operator has a zero mode, namely, $\rho^{eq}_{\boldsymbol{\lambda}(t)}$, and is not self-adjoint. We map $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ onto its corresponding Schrödinger operator, which is self-adjoint, and leverage the spectral theory of the latter to construct the Green's function of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$.

For the purposes of solving Eq. 2.1, the partial derivative with respect to time on the left-hand side should be interpreted as acting at fixed $\boldsymbol{\lambda}$. We will show in Section 2.1 that this produces a solution that is *consistent*, in the sense that both the left-hand side of Eq. 2.1 and the time derivative of the solution we find to this equation are $\mathcal{O}(\nu)$.

The associated Schrödinger operator and Green's function

The Fokker-Planck operator $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ is not self-adjoint. However, we can construct a self-adjoint operator $\hat{\mathcal{H}}$ from $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ by making the similarity transformation

$$\hat{\mathcal{H}} = e^{\beta V_{\boldsymbol{\lambda}(t)}/2}\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}e^{-\beta V_{\boldsymbol{\lambda}(t)}/2}. \tag{2.13}$$

We have suppressed the $x-$dependence of the potential and the operators for notational convenience. $\hat{\mathcal{H}}$ and $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ share eigenvalues, and their eigenfunctions are related by a simple transformation that we will discuss shortly. $\hat{\mathcal{H}}$ takes the form

$$\hat{\mathcal{H}}(x) = \frac{1}{\gamma\beta}\left(\frac{\beta}{2}V''_{\boldsymbol{\lambda}(t)}(x) - \left(\frac{\beta}{2}V'_{\boldsymbol{\lambda}(t)}(x)\right)^2 + \frac{\partial^2}{\partial x^2}\right). \tag{2.14}$$

It is related to the one-dimensional single-particle Schrödinger operator $\hat{\mathcal{H}}_{\mathcal{S}}$ as follows:

$$\hat{\mathcal{H}}_{\mathcal{S}} = -\frac{1}{2}\hat{\mathcal{H}}. \tag{2.15}$$

We have

$$\hat{\mathcal{H}}_{\mathcal{S}} = -\frac{1}{2\gamma\beta}\frac{\partial^2}{\partial x^2} + U_{\boldsymbol{\lambda}(t)}(x), \tag{2.16}$$

where the potential $U_{\boldsymbol{\lambda}(t)}$ is given by

$$U_{\boldsymbol{\lambda}(t)}(x) = \frac{1}{2\gamma\beta}\left(\left(\frac{\beta}{2}V'_{\boldsymbol{\lambda}(t)}(x)\right)^2 - \frac{\beta}{2}V''_{\boldsymbol{\lambda}(t)}(x)\right). \tag{2.17}$$

This map between Fokker-Planck and Schrödinger operators is well-known [102, 97]. We use it here to apply the spectral theory of the Schrödinger operator to driven Brownian motion. Any potential for which the spectral decomposition of the Schrödinger operator is known and possesses certain properties then becomes accessible to us for the purposes of solving Eq. 2.1.

As mentioned, the requirements for this approach to be viable involve conditions on the spectrum of $\hat{\mathcal{H}}_{\mathcal{S}}$. Natural boundary conditions on Eq. 2.1 already require $V_{\boldsymbol{\lambda}(t)}(x) \to \infty$ as $x \to \pm\infty$. We additionally require $V_{\boldsymbol{\lambda}(t)}$ to be such that $U_{\boldsymbol{\lambda}(t)}$ is also confining. That is, $U_{\boldsymbol{\lambda}(t)}(x) \to \infty$ as $x \to \pm\infty$. This is satisfied, for example, if $V_{\boldsymbol{\lambda}(t)}$ is harmonic, and not satisfied if it is logarithmic in $|x|$ at large x.

We use E_n and ψ_n to denote the eigenvalues and eigenfunctions of $\hat{\mathcal{H}}_{\mathcal{S}}$. The eigenvalue equation is

$$\hat{\mathcal{H}}_{\mathcal{S}}(x)\psi_n(x) = E_n\psi_n(x),\ n = 0, 1, \ldots. \tag{2.18}$$

For $x \in \mathbb{R}$, with the stated boundary condition on $U_{\boldsymbol{\lambda}(t)}$, we are guaranteed that the spectrum of $\hat{\mathcal{H}}_{\mathcal{S}}$ is discrete, non-degenerate ($E_m \neq E_n$ for $m \neq n$), and ordered ($E_n < E_{n+1}\,\forall n$). The fact that a confining potential confers a discrete non-degenerate spectrum can be argued rigorously (see theorem 10.7 in [62]). From a physical point of view this is reasonable to expect because in one spatial dimension a confining potential has bounded closed orbits which are quantized to give a discrete non-degenerate spectrum. (Tunneling effects can split degenerate energy levels separated by a potential barrier.) The discreteness of the spectrum crucially enables a simple definition of the Green's function of $\hat{\mathcal{H}}_{\mathcal{S}}$. See [75] for a proof of non-degeneracy.

It is simple to check[1] that $E_0 = 0$ and that the zeroth eigenfunction of $\hat{\mathcal{H}}_{\mathcal{S}}$ is given by

$$\psi_0(x) = \frac{1}{\sqrt{Z(t)}}e^{-\beta V_{\boldsymbol{\lambda}(t)}(x)/2}. \tag{2.19}$$

Note that $\rho^{eq}_{\boldsymbol{\lambda}(t)} = \psi_0^2$. The ψ_n are real, and form a complete orthonormal basis [75]:

$$\int dx\ \psi_n(x)\psi_m(x) = \delta_{nm}, \tag{2.20}$$

where δ_{nm} is the Kronecker delta. This guarantees the representation

$$\delta(x-y) = \sum_n \psi_n(x)\psi_n(y) \tag{2.21}$$

[1]Schrödinger operators customarily have nonzero zero-point energies. Here, $E_0 = 0$ due to the specific construction of $U_{\boldsymbol{\lambda}(t)}$, which is "shifted" downward by a factor of $V''_{\boldsymbol{\lambda}(t)}/4\gamma$ such that the usual zero-point energy of Eq. 2.16 is exactly removed.

for the delta function.

For $n > 0$, the eigenvalues of $\hat{\mathcal{H}}_{\mathcal{S}}$ satisfy $E_n > 0$. The proof of this claim is as follows. Left-multiplying Eq. 2.18 by ψ_n and integrating with respect to x, we have

$$E_n = \int dx \left(\frac{1}{2\gamma\beta} \left(\frac{\partial \psi_n}{\partial x} \right)^2 + U_{\boldsymbol{\lambda}(t)} \psi_n^2 \right). \tag{2.22}$$

Writing $\psi_n(x) = \rho_{l,n}(x)\psi_0(x)$ where $\rho_{l,n}$ is a smooth function with n nodes, this is

$$E_n = \int dx \, \frac{1}{2\gamma\beta} \psi_0^2 \left(\frac{\partial \rho_{l,n}}{\partial x} \right)^2 \geq 0, \tag{2.23}$$

with equality holding only for $n = 0$ since $\rho_{l,0} = 1$. The subscript l notation will become clear in the next section.

The function $\rho_{l,n}$ satisfies the eigenvalue equation

$$\frac{1}{\gamma} \left(-V'_{\boldsymbol{\lambda}(t)}(x) + \frac{1}{\beta} \frac{\partial}{\partial x} \right) \frac{\partial \rho_{l,n}}{\partial x} = \hat{\mathcal{L}}^{\dagger}_{\boldsymbol{\lambda}(t)} \rho_{l,n} = -2E_n \rho_{l,n}, \tag{2.24}$$

where $\hat{\mathcal{L}}^{\dagger}_{\boldsymbol{\lambda}(t)}$ is the Kolmogorov backward operator.[2] $\hat{\mathcal{L}}^{\dagger}_{\boldsymbol{\lambda}(t)}$ satisfies the symmetrization relation

$$\hat{\mathcal{H}} = e^{-\beta V_{\boldsymbol{\lambda}(t)}/2} \hat{\mathcal{L}}^{\dagger}_{\boldsymbol{\lambda}(t)} e^{\beta V_{\boldsymbol{\lambda}(t)}/2}. \tag{2.25}$$

Given the structure of the spectrum of $\hat{\mathcal{H}}_{\mathcal{S}}$, its Green's function $G_{\mathcal{S}}(x; y)$ is given by the following standard definition:

$$G_{\mathcal{S}}(x; y) = \sum_{n \neq 0} \frac{1}{E_n} \psi_n(x) \psi_n(y). \tag{2.26}$$

The action of $\hat{\mathcal{H}}_{\mathcal{S}}$ on $G_{\mathcal{S}}$ is

$$\hat{\mathcal{H}}_{\mathcal{S}}(x) G_{\mathcal{S}}(x; y) = \delta(x - y) - \psi_0(x)\psi_0(y). \tag{2.27}$$

Note that the right-hand side of Eq. 2.26 has the form of a projection. It indicates that $\hat{\mathcal{H}}_{\mathcal{S}}$ is only invertible in the space of functions orthogonal to ψ_0.

$\hat{\mathcal{H}}$ and $\hat{\mathcal{H}}_{\mathcal{S}}$ share eigenfunctions ψ_n. Writing α_n for the eigenvalues of $\hat{\mathcal{H}}$, these are given by

$$\alpha_n = -2E_n, \tag{2.28}$$

where $\alpha_0 = 0$ and $\alpha_{n>0} < 0$. The eigenvalue equation for $\hat{\mathcal{H}}$ is

$$\hat{\mathcal{H}}(x)\psi_n(x) = \alpha_n \psi_n(x). \tag{2.29}$$

[2]This operator is self-adjoint under the measure $dm(x)$ defined by $dm(x) = \left(\rho^{eq}_{\boldsymbol{\lambda}(t)}(x) \right)^{-1} dx$.

The Green's function $G_{\mathcal{H}}$ of $\hat{\mathcal{H}}$ is given by Eq. 2.26 with the replacement $E_n \to \alpha_n$:

$$G_{\mathcal{H}}(x;y) = \sum_{n\neq 0} \frac{1}{\alpha_n} \psi_n(x)\psi_n(y). \tag{2.30}$$

The action of $\hat{\mathcal{H}}$ on $G_{\mathcal{H}}$ is

$$\hat{\mathcal{H}}(x) G_{\mathcal{H}}(x;y) = \delta(x-y) - \psi_0(x)\psi_0(y). \tag{2.31}$$

The Green's function of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$

We use the discussion of the previous section to write down the eigenfunctions of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ and $\hat{\mathcal{L}}^{\dagger}_{\boldsymbol{\lambda}(t)}$, and the Green's function of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$.

From Eqs. 2.13, 2.25, and 2.29, we immediately have the relations

$$\hat{\mathcal{L}}(x)\rho_{r,n}(x) = \alpha_n \rho_{r,n}(x), \tag{2.32a}$$
$$\hat{\mathcal{L}}^{\dagger}(x)\rho_{l,n}(x) = \alpha_n \rho_{l,n}(x), \tag{2.32b}$$

where

$$\rho_{r,n}(x) = \psi_0(x)\psi_n(x), \tag{2.33a}$$
$$\rho_{l,n}(x) = (\psi_0(x))^{-1}\psi_n(x). \tag{2.33b}$$

$\rho_{r,n}$ and $\rho_{l,n}$ are called the right and left eigenfunctions, respectively. Together, they form a biorthogonal system that diagonalizes $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$. They are complete,

$$\delta(x-y) = \sum_n \rho_{r,n}(x)\rho_{l,n}(y), \tag{2.34}$$

and orthonormal,

$$\int dx\; \rho_{r,n}(x)\rho_{l,m}(x) = \delta_{nm}. \tag{2.35}$$

Eq. 2.34 follows from Eq. 2.21 and Eq. 2.35 follows from Eqs. 2.20 and 2.34. The zeroth right eigenfunction is the equilibrium distribution of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ corresponding to the specific value of $\boldsymbol{\lambda}$ at time t, and the zeroth left eigenfunction is a constant:

$$\rho_{r,0}(x) = \psi_0^2(x) = \rho^{eq}_{\boldsymbol{\lambda}(t)}(x), \tag{2.36a}$$
$$\rho_{l,0}(x) = 1. \tag{2.36b}$$

Due to Eq. 2.36, the right and left eigenfunctions share the simple relationship

$$\rho_{r,n} = \rho_{r,0}\rho_{l,n}. \tag{2.37}$$

We can now write down the Green's function $G_{\boldsymbol{\lambda}(t)}$ of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$. Using the representation Eq. 2.13 for $\hat{\mathcal{H}}$, and suppressing the subscript $\boldsymbol{\lambda}(t)$ for visual clarity, from Eq. 2.31 we have

$$e^{\beta V(x)/2}\hat{\mathcal{L}}(x)e^{-\beta V(x)/2}G_{\mathcal{H}}(x;y) = \sum_{n\neq 0}\psi_n(x)\psi_n(y). \tag{2.38}$$

Left-multiplying by $e^{-\beta V(x)/2}$, right-multiplying by $e^{\beta V(y)/2}$, and using Eq. 2.33, we arrive at

$$\hat{\mathcal{L}}(x)e^{-\beta V(x)/2}G_{\mathcal{H}}(x;y)e^{-\beta V(y)/2} = \sum_{n\neq 0}\rho_{r,n}(x)\rho_{l,n}(y), \tag{2.39}$$

from which we identify $G_{\boldsymbol{\lambda}(t)}$,

$$G_{\boldsymbol{\lambda}(t)}(x;y) = e^{-\beta V_{\boldsymbol{\lambda}(t)}(x)/2}G_{\mathcal{H}}(x;y)e^{\beta V_{\boldsymbol{\lambda}(t)}(y)/2} = \sum_{n\neq 0}\frac{1}{\alpha_n}\rho_{r,n}(x)\rho_{l,n}(y). \tag{2.40}$$

The action of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ on $G_{\boldsymbol{\lambda}(t)}$ is given by Eq. 2.39. Using Eqs. 2.34 and 2.36, this can be rewritten as

$$\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}(x)G_{\boldsymbol{\lambda}(t)}(x;y) = \delta(x-y) - \rho^{eq}_{\boldsymbol{\lambda}(t)}(x). \tag{2.41}$$

Solution to the Fokker-Planck equation

We can decompose the probability distribution in Eq. 2.1 into the sum of $\rho^{eq}_{\boldsymbol{\lambda}(t)}(x)$ and a correction $\delta\rho(x;t)$:

$$\rho(x;t) = \rho^{eq}_{\boldsymbol{\lambda}(t)}(x) + \delta\rho(x;t). \tag{2.42}$$

We must have $\int dx\ \delta\rho(x;t) = 0$ to preserve normalization. Using this representation for $\rho(x;t)$ in Eq. 2.1, we obtain the dynamics of $\delta\rho(x;t)$:

$$\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}(x)\delta\rho(x;t) = \frac{\partial}{\partial t}\rho(x;t). \tag{2.43}$$

In order to apply the method of Green's functions, we interpret the right-hand side of Eq. 2.43 as a source term. From this follows the relation

$$\delta\rho(x;t) = \int dy\ G_{\boldsymbol{\lambda}(t)}(x;y)\frac{\partial}{\partial t}\rho(y;t). \tag{2.44}$$

Eq. 2.44 contains the quantity $\delta\rho$ on both sides and can be solved iteratively. Thus we arrive at

$$\begin{aligned}\rho(x;t) = {}&\rho^{eq}_{\boldsymbol{\lambda}(t)}(x) + \int dx'\ G_{\boldsymbol{\lambda}(t)}(x;x')\frac{\partial}{\partial t}\rho^{eq}_{\boldsymbol{\lambda}(t)}(x') \\ &+ \int dx''\ G_{\boldsymbol{\lambda}(t)}(x;x'')\ \frac{\partial}{\partial t}\int dx'\ G_{\boldsymbol{\lambda}(t)}(x'';x')\frac{\partial}{\partial t}\rho^{eq}_{\boldsymbol{\lambda}(t)}(x') + \dots,\end{aligned} \tag{2.45}$$

with the partial time derivative of $\rho^{eq}_{\boldsymbol{\lambda}(t)}$ given by Eq. 2.12.

The form of Eq. 2.45 is $\rho(x;t) = \rho^{eq}_{\boldsymbol{\lambda}(t)}(x) + \sum_{n=1}^{\infty} \delta\rho^{(n)}(x;t)$, where the quantities $\delta\rho^{(n)}$ are corrections to $\rho^{eq}_{\boldsymbol{\lambda}(t)}$. We observe that the corrections have a recursive structure, and integrate to zero:

$$\delta\rho^{(n+1)}(x;t) = \int dx'\, G_{\boldsymbol{\lambda}(t)}(x;x')\, \frac{\partial}{\partial t}\delta\rho^{(n)}(x';t), \tag{2.46a}$$

$$\int dx\; \delta\rho^{(n+1)}(x;t) = 0,\ \ n \geq 0. \tag{2.46b}$$

In the above, we have notated $\rho^{eq}_{\boldsymbol{\lambda}(t)}(x)$ as $\delta\rho^{(0)}(x;t)$. The form of Eq. 2.46a indicates that $\delta\rho^{(n+1)}(x;t)$ contains precisely $n+1$ derivatives with respect to time. This motif will be important in Section 2.2, where we will see that it introduces geometric structure to the average dissipation.

The expansion parameter ν

Eq. 2.45 is a derivative expansion. In this section, we justify this claim.

There are two sources of timescales in this problem: the eigenvalues of the Fokker-Planck operator, and the time variation of the control parameters.

The eigenvalues α_n of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ have the physical units of inverse time, and their absolute values set the various natural timescales of the system. Calling these timescales τ_{α_n}, we have $\tau_{\alpha_n} = 1/|\alpha_n|$. Due to the ordering of the α_n, the τ_{α_n} are also ordered. The longest natural timescale in the system is τ_{α_1}, known as the relaxation time.

Each external parameter λ_i has a characteristic timescale τ_{λ_i} associated with its time evolution. We denote the shortest of these timescales as $\tau_\lambda = \min_i \tau_{\lambda_i}$.

Now let us examine the total time variation of $\rho(x;t)$:

$$\frac{d}{dt}\rho(x;t) = \frac{\partial}{\partial t}\rho(x;t) + \sum_i \dot{\lambda}_i \frac{\partial}{\partial \lambda_i}\rho(x;t). \tag{2.47}$$

In the first term on the right-hand side of Eq. 2.47, the time derivative acts at fixed $\boldsymbol{\lambda}$ and the time evolution is generated by the Fokker-Planck operator, i.e., by Eq. 2.1. The second term describes the time variation resulting from the time dependence of the external control parameters, which is not determined by the Fokker-Planck operator.[3] Note that if we replace $\rho(x;t)$ by $\rho^{eq}_{\boldsymbol{\lambda}(t)}(x)$ in Eq. 2.47, the first term on the right-hand side evaluates to zero, exactly consistent with Eq. 2.12.

In this work we consider the scenario in which the dynamics of $\boldsymbol{\lambda}$ is very slow compared to the dynamics generated by the Fokker-Planck operator. This means the longest natural timescale τ_{α_1} must be shorter than the shortest control timescale τ_λ:

$$\tau_\lambda \gg \tau_{\alpha_1}. \tag{2.48}$$

[3]We will see in a later section that this time variation is determined by another principle, namely, the minimization of the average heat produced in the reservoir over the course of driving.

Eq. 2.48 naturally gives rise to a dimensionless small parameter ν, defined as follows: $\nu = \tau_{\alpha_1}/\tau_\lambda \ll 1$. It is the smallness of this parameter that justifies our usage of Eq. (1) to approximate the true dynamics of $\rho(x;t)$, which is given by the left-hand side of Eq. 2.47.

In Eq. 2.45, derivatives with respect to time act (through $G_{\boldsymbol{\lambda}(t)}$ and $\rho^{eq}_{\boldsymbol{\lambda}(t)}$) only on $\boldsymbol{\lambda}(t)$, and so we can rescale time in $\boldsymbol{\lambda}-$space by ν by defining the variable $\tilde{t} = \nu t$. Making the reparameterization $t \to \tilde{t}$ in Eq. 2.45, we arrive at an expansion for $\rho(x;t)$ in the manifestly dimensionless small parameter ν:

$$\begin{aligned}\rho(x;\tilde{t}) =& \rho^{eq}_{\boldsymbol{\lambda}(\tilde{t})}(x) + \nu \int dx'\, G_{\boldsymbol{\lambda}(\tilde{t})}(x;x') \frac{\partial \rho^{eq}_{\boldsymbol{\lambda}(\tilde{t})}}{\partial \tilde{t}}(x') \\ &+ \nu^2 \int dx''\, G_{\boldsymbol{\lambda}(\tilde{t})}(x;x'')\, \frac{\partial}{\partial \tilde{t}} \int dx'\, G_{\boldsymbol{\lambda}(\tilde{t})}(x'';x') \frac{\partial \rho^{eq}_{\boldsymbol{\lambda}(\tilde{t})}}{\partial \tilde{t}}(x') + \dots . \end{aligned} \tag{2.49}$$

What is happening here is that there is a separation of timescales between the laboratory and the control space. In the latter, time must be measured in units of τ_λ. However, the overall timescale of the problem is set by τ_{α_1}, which is fixed by the shape of the potential. Therefore when expanding the density $\rho(x;t)$, it is necessary to measure τ_λ *in units of* τ_{α_1}. This is why time in control space is scaled by ν.

The condition Eq. 2.48 imposes a constraint on the dynamics of the spectrum of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$, which we now discuss. In general, the α_n are functions of all the control parameters λ_i due to the fact that the spectrum of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ depends on $V_{\boldsymbol{\lambda}(t)}$, which is a function of $\boldsymbol{\lambda}$. The time derivative of α_n is

$$\frac{d\alpha_n}{dt} = \sum_i \dot{\lambda}_i \frac{\partial \alpha_n}{\partial \lambda_i}, \tag{2.50}$$

where the variation of α_n with respect to λ_i is given by the Hellmann-Feynman theorem [49]

$$\frac{\partial \alpha_n}{\partial \lambda_i} = \int dx\, \psi_n^2(x) \frac{\partial \hat{\mathcal{H}}(x)}{\partial \lambda_i} = -2 \int dx\, \psi_n^2(x) \frac{\partial U_{\boldsymbol{\lambda}(t)}(x)}{\partial \lambda_i}. \tag{2.51}$$

For every $i \in (1,\dots,k)$, Eq. 2.51 is finite and fully determined by the form of the potential $U_{\boldsymbol{\lambda}(t)}$. Therefore Eq. 2.48, which can equivalently be written as $\max_i \left|\dot{\lambda}_i\right| \ll |\alpha_1|$, together with Eq. 2.50, implies that the quantities $|\dot{\alpha}_n|$ must be small $\forall n$. We can explicitly check that this condition holds. Note that

$$\dot{\lambda}_i = \frac{d\tilde{t}}{dt}\frac{d\lambda}{d\tilde{t}} = \nu \frac{d\lambda}{d\tilde{t}} = \mathcal{O}(\nu), \tag{2.52}$$

and so $\dot{\lambda}_i$ is of order ν. Together with Eq. 2.52, Eq. 2.50 implies that $|\dot{\alpha}_n|$ is also $\mathcal{O}(\nu)$. That is, the condition Eq. 2.48 forces the spectrum of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ to change slowly over the course of driving.

Due to fact that derivatives with respect to time in Eq. 2.49 act only on $\boldsymbol{\lambda}(t)$, Eq. 2.52 also implies that the time derivative of Eq. 2.49 is $\mathcal{O}(\nu)$, which is consistent with the time-dependence of Eq. 2.47 on $\boldsymbol{\lambda}$.

The last point we must address in this timescale analysis is the fact that ν itself is a function of time. Clearly, in order for the expansion in Eq. 2.49 to be stable, we require the time variation of ν to be small. We can check that Eq. 2.48 indeed enforces this. Using Eq. 2.52, we find that

$$\frac{d\nu}{dt} = \mathcal{O}\left(\nu^2\right). \tag{2.53}$$

In fact, the n^{th} time derivative of ν for $n \geq 1$ is of order ν^{n+1}.

Thus as long as the control timescale is chosen such that the slowness condition Eq. 2.48 is satisfied, the procedure we have presented for solving Eq. 2.1 is consistent, and Eq. 2.49 describes the time evolution of $\rho(x;t)$.

In the next section we derive a formula for the thermodynamic metric using Eq. 2.45. We note that in all previous work [127, 155, 106] in which the thermodynamic metric has been derived it is assumed that the timescale of driving is slow with respect to the longest natural timescale of the system. The analysis just given explains why this assumption is necessary: without it, the Fokker-Planck equation is not a good descriptor of the driven Brownian system.

Lastly, we note that other authors have previously made use of eigenfunction expansions of $\rho(x;t)$ to calculate the average dissipation for driven Brownian systems with a single slowly varying control parameter [120, 71]. We will calculate average dissipation in the next section. The authors recognized that their methods must correspond to a perturbative approach to solving Eq. 2.1 as we have presented here, but this idea was not fully developed. In particular, the precise conditions under which the spectral structure of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$ permits a perturbative expansion of $\rho(x;t)$ in ν and the relative importance of the various time scales in the problem were not studied, and τ_λ was not identified.

2.2 The thermodynamic metric

Writing down a driving protocol for a system involves specifying a functional form for the time dependence of the control vector $\boldsymbol{\lambda}$. We say a driving protocol $\boldsymbol{\Lambda}$ is optimal if it minimizes the functional for the average heat $\langle \Delta Q \rangle [\boldsymbol{\Lambda}]$ produced in the reservoir over the course of driving [127]:

$$\boldsymbol{\Lambda}^{\text{opt}} = \arg\min_{\boldsymbol{\Lambda}} \langle \Delta Q \rangle [\boldsymbol{\Lambda}]. \tag{2.54}$$

We are interested in the scenario where the system is driven between two fixed values of $\boldsymbol{\lambda}$ over a fixed time period Ω. Note that we must have $\Omega \gg \tau_\lambda$.

The average heat transferred to the reservoir over the course of driving is given by the formula [118]

$$\begin{aligned}\langle \Delta Q\rangle[\mathbf{\Lambda}] &= -\int_0^\Omega dt \int dx\, V'_{\boldsymbol{\lambda}(t)}(x) J(x;t)\\ &= \int_0^\Omega dt \int dx\, \rho(x;t)\left(\frac{V'^2_{\boldsymbol{\lambda}(t)}(x)}{\gamma} - \frac{V''_{\boldsymbol{\lambda}(t)}(x)}{\gamma\beta}\right). \end{aligned} \tag{2.55}$$

In the second equality, we have replaced $J(x;t)$ with the right-hand side of Eq. 2.5 and integrated by parts. Note that the quantity in parentheses in Eq. 2.55 is, up to a constant factor $4/\beta$, the Schrödinger potential $U_{\boldsymbol{\lambda}(t)}$ at inverse temperature 2β.

In the following, we calculate $\langle \Delta Q\rangle$ using the approximation

$$\rho(x;t) = \rho^{eq}_{\boldsymbol{\lambda}(t)}(x) + \delta\rho^{(1)}(x;t) + \delta\rho^{(2)}(x;t), \tag{2.56}$$

with the corrections $\delta\rho^{(1)}(x;t)$ and $\delta\rho^{(2)}(x;t)$ given by the second and third terms on the right-hand side of Eq. 2.45, respectively:

$$\delta\rho^{(1)}(x;t) = \int dx'\, G_{\boldsymbol{\lambda}(t)}(x;x')\frac{\partial}{\partial t}\rho^{eq}_{\boldsymbol{\lambda}(t)}(x'), \tag{2.57a}$$

$$\delta\rho^{(2)}(x;t) = \int dx'\, G_{\boldsymbol{\lambda}(t)}(x;x')\,\frac{\partial}{\partial t}\delta\rho^{(1)}(x';t). \tag{2.57b}$$

We show that one of the contributions to $\langle \Delta Q\rangle$ coming from $\delta\rho^{(2)}$ contains an integral over a symmetric positive definite matrix in the space of control parameters $\mathcal{C}$, and we identify this as the thermodynamic metric for systems described by Eq. 2.1 with the stated conditions on $V_{\boldsymbol{\lambda}(t)}$ and $U_{\boldsymbol{\lambda}(t)}$. We discuss the emergence of this geometric structure in $\langle \Delta Q\rangle$ and show that it persists to all orders in the expansion of $\rho(x;t)$ (Eq. 2.45).

Calculation of $\langle \Delta Q\rangle$ and derivation of thermodynamic metric

We drop the subscript $\boldsymbol{\lambda}(t)$ for visual clarity.

It is useful to rewrite Eq. 2.55 in the equivalent form

$$\langle \Delta Q\rangle[\mathbf{\Lambda}] = \frac{1}{\gamma\beta^2}\int_0^\Omega dt \int dx\ \rho(x;t) e^{\beta V(x)}\partial_x^2 e^{-\beta V(x)}. \tag{2.58}$$

The first contribution to $\langle \Delta Q\rangle$ from Eq. 2.56 corresponds to approximating $\rho(x;t)$ by $\rho^{eq}(x)$, and it evaluates to zero:

$$\langle \Delta Q\rangle_0 = \frac{1}{\gamma\beta^2}\int_0^\Omega dt \int dx\ \rho^{eq}(x)\ e^{\beta V(x)}\partial_x^2 e^{-\beta V(x)} = 0. \tag{2.59}$$

This is easily seen by using Eq. 2.9 to replace $e^{-\beta V(x)}$ and applying the normalization condition Eq. 2.8.

To calculate the next two terms of $\langle \Delta Q \rangle$, we will make use of the following identity:

$$\int dx\ G(x;x')e^{\beta V(x)}\partial_x^2 e^{-\beta V(x)} = \gamma\beta \int dx\ \left(1-\beta V(x)\right)\hat{\mathcal{L}}(x)G(x;x'). \tag{2.60}$$

This is derived by integrating the left-hand side by parts twice, evaluating the resulting double derivative over the product $G(x;x')e^{\beta V(x)}$, and integrating by parts again. The boundary terms in Eq. 2.60 vanish.

The second contribution to $\langle \Delta Q \rangle$ is

$$\langle \Delta Q \rangle_1 = \frac{1}{\gamma\beta^2}\int_0^\Omega dt \int dx\ \delta\rho^{(1)}(x;t)\ e^{\beta V(x)}\partial_x^2 e^{-\beta V(x)}. \tag{2.61}$$

Replacing $\delta\rho^{(1)}$ with Eq. 2.57a, applying Eq. 2.60 and then Eq. 2.41, we have

$$\begin{aligned}\langle \Delta Q \rangle_1 &= \frac{1}{\beta}\int_0^\Omega dt \int dx'\ \partial_t \rho^{eq}(x') \int dx\ \left(1-\beta V(x)\right)\hat{\mathcal{L}}(x)G(x;x') \\ &= -\frac{1}{\beta}\int_0^\Omega dt \int dx'\ \partial_t\rho^{eq}(x')\beta V(x') + \frac{1}{\beta}\int_0^\Omega dt\int dx'\ \partial_t\rho^{eq}(x')\int dx\ \beta V(x)\rho^{eq}(x).\end{aligned} \tag{2.62}$$

The second term in Eq. 2.62 is zero due to Eq. 2.8, which implies $\partial_t \int dx\ \rho^{eq}(x) = \partial_t 1 = 0$. The first term can be written in terms of the difference in entropy ΔS^{eq} between $\rho^{eq}_{\boldsymbol{\lambda}(0)}(x)$ and $\rho^{eq}_{\boldsymbol{\lambda}(\Omega)}(x)$. We recall the definition of the entropy S^{eq} of an equilibrium distribution:

$$S^{eq}_{\boldsymbol{\lambda}(t)} = -\int dx\ \rho^{eq}_{\boldsymbol{\lambda}(t)}(x)\log \rho^{eq}_{\boldsymbol{\lambda}(t)}(x), \tag{2.63}$$

the time derivative of which is $\int dx\ \beta V(x)\partial_t\rho^{eq}(x)$. Thus we have

$$\langle \Delta Q \rangle_1 = -\frac{1}{\beta}\int_0^\Omega dt\ \partial_t S^{eq}_{\boldsymbol{\lambda}(t)} = -\frac{1}{\beta}\Delta S^{eq}. \tag{2.64}$$

If we truncate the approximation of $\rho(x;t)$ at $\delta\rho^{(1)}(x;t)$, we reproduce the quasistatic Clausius equality for diffusive systems [60, 83, 85]:

$$\beta\langle \Delta Q \rangle [\boldsymbol{\Lambda}] + \Delta S^{eq} = 0. \tag{2.65}$$

The third contribution to $\langle \Delta Q \rangle$ is

$$\langle \Delta Q \rangle_2 = \frac{1}{\gamma\beta^2}\int_0^\Omega dt \int dx\ \delta\rho^{(2)}(x;t)\ e^{\beta V(x)}\partial_x^2 e^{-\beta V(x)}. \tag{2.66}$$

Similar to the calculation of $\langle \Delta Q \rangle_1$, we use Eq. 2.57b to replace $\delta\rho^{(2)}$, apply Eq. 2.60, and then Eq. 2.41. This gives

$$\langle \Delta Q \rangle_2 = -\frac{1}{\beta}\int_0^\Omega dt \left[\int dx''\, \partial_t \delta\rho^{(1)}(x'';t)\beta V(x'') - \int dx''\, \partial_t\delta\rho^{(1)}(x'';t)\int dx\, \rho^{eq}(x)\beta V(x)\right]. \tag{2.67}$$

The second term in Eq. 2.67 is zero due to Eq. 2.46b. Writing $-\beta V(x'') = \log \rho^{eq}(x'') + \log Z$, the first term can be rewritten as

$$\begin{aligned}\langle \Delta Q \rangle_2 = &-\frac{1}{\beta}\int_0^\Omega dt \int dx''\; \delta\rho^{(1)}(x'';t)\; \partial_t \log \rho^{eq}(x'') \\ &-\frac{1}{\beta}\int_0^\Omega dt\; (\partial_t \log Z)\int dx''\; \delta\rho^{(1)}(x'';t) \\ &-\int_0^\Omega dt \int dx''\; \partial_t\left(\delta\rho^{(1)}(x'';t)V(x'')\right).\end{aligned} \tag{2.68}$$

We evaluate the three terms in Eq. 2.68 in reverse order.

The third term is the integral of a total time derivative, and depends only on the initial and final values of $\boldsymbol{\lambda}$ and $\dot{\boldsymbol{\lambda}}$. It can be written as

$$A(\boldsymbol{\lambda}(\Omega), \dot{\boldsymbol{\lambda}}(\Omega)) - A(\boldsymbol{\lambda}(0), \dot{\boldsymbol{\lambda}}(0)) \equiv \Delta A, \tag{2.69}$$

where the function A is given by

$$A = -\sum_i \dot{\lambda}_i \int\int dx\, dx'\; V_{\boldsymbol{\lambda}(t)}(x) G_{\boldsymbol{\lambda}(t)}(x;x') \frac{\partial \rho^{eq}_{\boldsymbol{\lambda}(t)}}{\partial \lambda_i}(x'). \tag{2.70}$$

The second term in Eq. 2.68 evaluates to zero due to Eq. 2.46b.

Lastly, the integral with respect to x'' in the first term in Eq. 2.68 can be rewritten as a quadratic form:

$$-\int dx''\; \delta\rho^{(1)}(x'';t)\; \partial_t \log \rho^{eq}_{\boldsymbol{\lambda}(t)}(x'') = \dot{\boldsymbol{\lambda}}^\top \boldsymbol{\zeta} \dot{\boldsymbol{\lambda}}, \tag{2.71}$$

where the elements of the matrix $\boldsymbol{\zeta}(\boldsymbol{\lambda})$ are given by the formula

$$\zeta_{ij} = -\int dx' dx''\; \rho^{eq}_{\boldsymbol{\lambda}(t)}(x'') \left(\frac{\partial}{\partial \lambda_i}\log \rho^{eq}_{\boldsymbol{\lambda}(t)}(x'')\right) G_{\boldsymbol{\lambda}(t)}(x';x'') \left(\frac{\partial}{\partial \lambda_j}\log \rho^{eq}_{\boldsymbol{\lambda}(t)}(x')\right). \tag{2.72}$$

$\boldsymbol{\zeta}(\boldsymbol{\lambda})$ is clearly symmetric. We now prove that it is also positive definite. In terms of ψ_0 and $G_{\mathcal{S}}$, Eq. 2.72 takes the following simple form:

$$\zeta_{ij} = 2\int dx' dx''\; \frac{\partial \psi_0(x'')}{\partial \lambda_i}\, G_{\mathcal{S}}(x';x'')\, \frac{\partial \psi_0(x')}{\partial \lambda_j}. \tag{2.73}$$

Consider the quadratic form $\dot{\boldsymbol{\lambda}}^{\top}\boldsymbol{\zeta}\dot{\boldsymbol{\lambda}}$. Using Eqs. 2.26 and 2.28 in Eq. 2.73, we have

$$\dot{\boldsymbol{\lambda}}^{\top}\boldsymbol{\zeta}\dot{\boldsymbol{\lambda}} = -\sum_{n\neq 0}\frac{1}{\alpha_n}\left(2\sum_{i=1}^{k}\int dx\ \dot{\lambda}_i\psi_n(x)\frac{\partial\psi_0}{\partial\lambda_i}(x)\right)^2 > 0. \tag{2.74}$$

The last inequality is due to the fact that $-\alpha_{n\neq 0} > 0$. Thus, the eigenvalues of $\boldsymbol{\zeta}(\boldsymbol{\lambda})$ are positive. $\boldsymbol{\zeta}(\boldsymbol{\lambda})$ therefore induces a Riemannian metric on the space $\mathcal{C}$, and can be identified as the thermodynamic metric [127] for driven Brownian systems described by Eq. 2.1 with confining Schrödinger potentials. We note that Eq. 2.72 contains all the time scales in the problem, since $G_{\boldsymbol{\lambda}(t)}$ contains a sum over all the eigenvalues of $\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}$.

It becomes necessary now to distinguish between covariant and contravariant quantities; therefore from this point onward in the discussion we will write control variables with raised indices, as λ^i.

We can explicitly check that $\boldsymbol{\zeta}(\boldsymbol{\lambda})$ transforms correctly under a change of coordinates. Using the representation Eq. 2.73, it is simple to see that under a continuous, invertible transformation (diffeomorphism) $\boldsymbol{\lambda}\to\boldsymbol{\phi}(\boldsymbol{\lambda})$, the elements of the new metric $\tilde{\boldsymbol{\zeta}}(\boldsymbol{\phi})$ in $\boldsymbol{\phi}$-space are given by

$$\tilde{\zeta}_{kl} = \sum_{i,j}\zeta_{ij}\frac{\partial\lambda^i}{\partial\phi^k}\frac{\partial\lambda^j}{\partial\phi^l}. \tag{2.75}$$

This transformation law for the metric holds due to the two partial derivatives with respect to λ^i and λ^j in Eq. 2.73, which in turn derive from the two partial derivatives with respect to time in $\delta\rho^{(2)}(x;t)$. Therefore, even though Eq. 2.55 has no geometric structure in general that we can discover, the specific form of $\delta\rho^{(2)}(x;t)$ introduces geometric structure in the average dissipation. We will see shortly that this emergent structure persists in Eq. 2.55 to all orders in ν.

We emphasize that Eq. 2.72 is distinct from the formula for a thermodynamic metric given in Eq. 12 in [127], which was the first work to derive a thermodynamic metric for mesoscopic systems with time-varying relaxation times. As mentioned previously, this formula was derived in the linear response regime with a slow driving assumption. Evaluating it involves computing an integral with respect to time over the linear response function, which is the average two-point time correlation function of deviations of the conjugate forces from their equilibrium values.

Gathering the contributions from Eqs. 2.59, 2.64, 2.69 and 2.71, we have the following formula for the average heat up to terms of order ν^2 in Eq. 2.45:

$$\beta\langle\Delta Q\rangle[\boldsymbol{\Lambda}] = 0 - \Delta S^{eq} + \int_0^{\Omega} dt\ \dot{\boldsymbol{\lambda}}(t)\boldsymbol{\zeta}(\boldsymbol{\lambda})\dot{\boldsymbol{\lambda}}(t)^{\top} + \beta\Delta A. \tag{2.76}$$

To minimize Eq. 2.76 over protocols, we can define the action

$$S[\boldsymbol{\lambda}(t)] = \beta\Delta A + 2\int_0^{\Omega} dt\ \frac{1}{2}\dot{\boldsymbol{\lambda}}(t)\boldsymbol{\zeta}(\boldsymbol{\lambda})\dot{\boldsymbol{\lambda}}(t)^{\top}. \tag{2.77}$$

The equations of motion follow by setting the variation $\frac{\delta S}{\delta \lambda^i}$ of S with respect to λ^i to zero, subject to the constraints $\delta\lambda^i(0) = \delta\lambda^i(\Omega) = 0 \ \forall i$. These constraints imply $\delta A(0) = \delta A(\Omega) = 0$, and therefore only the second term in Eq. 2.77 contributes to the equations of motion. These are the Euler-Lagrange equations of the Lagrangian $L = \frac{1}{2}\dot{\boldsymbol{\lambda}}^\top \boldsymbol{\zeta}\dot{\boldsymbol{\lambda}}$:

$$\frac{d}{dt}\left(2\sum_j \zeta_{pj}\dot{\lambda}^j\right) = \sum_{i,j}\dot{\lambda}^i \frac{\partial \zeta_{ij}}{\partial \lambda^p}\dot{\lambda}^j,\ p \in (1,\ldots,k). \tag{2.78}$$

Opening out the time derivative on the left-hand side of Eq. 2.78, a straightforward calculation shows that it is equivalent to

$$\ddot{\lambda}^p + \sum_{i,j}\Gamma^p_{ij}\dot{\lambda}^i\dot{\lambda}^j = 0,\ p \in (1,\ldots,k), \tag{2.79}$$

where Γ^p_{ij} is the Christoffel symbol of the second kind:

$$\Gamma^p_{ij} = \frac{1}{2}\sum_m \zeta^{pm}\left(\frac{\partial \zeta_{mi}}{\partial x^j} + \frac{\partial \zeta_{mj}}{\partial x^i} - \frac{\partial \zeta_{ij}}{\partial x^m}\right). \tag{2.80}$$

Eq. 2.79 are also the equations of motion of the Lagrangian $\tilde{L} = \sqrt{\dot{\boldsymbol{\lambda}}\boldsymbol{\zeta}\dot{\boldsymbol{\lambda}}^\top}$ in the arc-length parameterization [99]. In other words, these are geodesic equations of the control parameter space $\mathcal{C}$.

Due to the spectral properties of $\hat{\mathcal{H}}_{\mathcal{S}}$, Eq. 2.74 also indicates that the quadratic form $\dot{\boldsymbol{\lambda}}^\top\boldsymbol{\zeta}\dot{\boldsymbol{\lambda}}$ is always finite. Therefore, if $V_{\boldsymbol{\lambda}(t)}$ is such that $U_{\boldsymbol{\lambda}(t)}$ is confining, and the perturbative expansion Eq. 2.45 holds over the time period Ω, we are guaranteed that $\boldsymbol{\zeta}(\boldsymbol{\lambda})$ exists and is well-defined over the course of driving. Then, up to terms of order ν^2 in Eq. 2.45, optimal protocols $\boldsymbol{\Lambda}^{\text{opt}}$ are geodesics in $\mathcal{C}$ with respect to the length measure defined by $\boldsymbol{\zeta}(\boldsymbol{\lambda})$.

We note that in a specific optimal problem, the invariance of the geodesic equations to reparameterizations of $\mathcal{C}$ is broken by the boundary conditions, in which the *identities* of the control parameters, along with their initial and final values, are specified. For example, in the next section, we consider the harmonic potential $V_{\boldsymbol{\lambda}(t)}(x) = \kappa x^2/2 + Ex$ with time-dependent electric field E and spring constant κ. The choice of these two control parameters breaks the diffeomorphism invariance of Eq. 2.79 for this problem instance.

The diffeomorphism invariance of the geodesic equations suggests that it is appropriate to write $V_{\boldsymbol{\lambda}(t)}$ in such a way that all components of $\boldsymbol{\lambda}$ have matching units. One way to do this is to introduce a fixed length scale ℓ and rescale x as $x \to x/\ell$. For example, in the harmonic potential defined previously, the control parameters κ and E have different units. Rescaling x by ℓ, we can instead write $V_{\boldsymbol{\lambda}(t)}(x/\ell) = (\ell^2\kappa)(x/\ell)^2/2 + (\ell E)x/\ell$. The new control vector is $\boldsymbol{\lambda} = (\ell^2\kappa, \ell E)$, both components of which have units of energy. Applying diffeomorphisms that may scramble the two control parameters now makes sense. We can choose ℓ to be such that $\beta\ell E = 1$, or, equivalently, such that $\beta\ell^2\kappa = 1$.

We end this section with a note on higher-order terms in the average heat production. By calculations analogous to those for $\langle \Delta Q \rangle_2$, it is straightforward to establish that for any $w \geq 2$, the contribution to Eq. 2.55 from $\delta\rho^{(w)}(x;t)$ takes the form

$$\beta \langle \Delta Q \rangle_w = \beta \Delta A_w + \int_0^\Omega dt \sum_{i_1,\ldots,i_w} \dot{\lambda}^{i_1} \ldots \dot{\lambda}^{i_w} \Xi^{(w)}_{i_1 \ldots i_w}, \tag{2.81}$$

where A_w is a term that depends only on the values of $\boldsymbol{\lambda}$ and $\dot{\boldsymbol{\lambda}}$ at times 0 and Ω, and $\boldsymbol{\Xi}^{(w)}$ is an object with w indices. (In the notation of Eq. 2.81, the quantity A defined in Eq. 2.70 is A_2, and the thermodynamic metric $\boldsymbol{\zeta}$ is $\boldsymbol{\Xi}^{(2)}$.) Due to the fact that $\delta\rho^{(w)}(x;t)$ contains exactly w derivatives with respect to time, under a reparameterization $\boldsymbol{\lambda} \to \boldsymbol{\phi}(\boldsymbol{\lambda})$, $\boldsymbol{\Xi}^{(w)}$ obeys the transformation law $\tilde{\Xi}^{(w)}_{j_1 \ldots j_w} = \sum_{i_1,\ldots,i_w} \Xi^{(w)}_{i_1 \ldots i_w} \partial_{\phi^{j_1}} \lambda^{i_1} \ldots \partial_{\phi^{j_w}} \lambda^{i_w}$, and is therefore a rank-w tensor. Thus, if the conditions for the existence of Eq. 2.45 are met, geometric structure is emergent in Eq. 2.55 at all orders in ν.

Up to terms of order ν^k in $\rho(x;t)$, the Lagrangian of the optimal control problem is given by $L^{(w)} = \sum_{w=2}^k \sum_{i_1,\ldots,i_w} \dot{\lambda}^{i_1} \ldots \dot{\lambda}^{i_w} \Xi^{(w)}_{i_1 \ldots i_w}$; like Eq. 2.70, the ΔA_w for $w \geq 3$ do not participate in the Euler-Lagrange equations for $\boldsymbol{\Lambda}^{\text{opt}}$. Predictions of optimal protocols can be refined beyond the solutions of Eq. 2.79 by including terms of order $w = 3$ and higher in $L^{(w)}$. The $\boldsymbol{\Xi}^{(w)}$—and therefore $L^{(w)}$—can easily be expressed in terms of $\rho^{eq}_{\boldsymbol{\lambda}(t)}$ and $G_{\boldsymbol{\lambda}(t)}$. For example, the elements of $\boldsymbol{\Xi}^{(3)}$ are given by

$$\Xi^{(3)}_{ijk} = -\int dx dx' \, \frac{\partial \log \rho^{eq}_{\boldsymbol{\lambda}(t)}(x)}{\partial \lambda^i} G_{\boldsymbol{\lambda}(t)}(x;x') \frac{\partial}{\partial \lambda^j} \left(\int dx'' \, G_{\boldsymbol{\lambda}(t)}(x';x'') \rho^{eq}_{\boldsymbol{\lambda}(t)}(x'') \frac{\partial \log \rho^{eq}_{\boldsymbol{\lambda}(t)}(x'')}{\partial \lambda^k} \right). \tag{2.82}$$

We leave the study of possible interpretations of $\boldsymbol{\Xi}^{(w)}$ for $w \geq 3$ and the development of solutions of the Euler-Lagrange equations of $L^{(w)}$ for $w \geq 3$ to future work.

Relationship of ζ to previously proposed formula for a thermodynamic metric

In [154], the authors propose an approximate formula for a thermodynamic metric involving only $\rho^{eq}_{\boldsymbol{\lambda}(t)}$. Call this metric $\boldsymbol{\chi}$. Using the notation $\Pi^{eq}_{\boldsymbol{\lambda}(t)}$ to refer to the cumulative distribution function

$$\Pi^{eq}_{\boldsymbol{\lambda}(t)}(x) = \int_{-\infty}^{x} dx' \rho^{eq}_{\boldsymbol{\lambda}(t)}(x'), \tag{2.83}$$

the elements of $\boldsymbol{\chi}$ are given by

$$\chi_{ij} = \int dx \, \frac{\gamma \beta}{\rho^{eq}_{\boldsymbol{\lambda}(t)}(x)} \left(\frac{\partial}{\partial \lambda_i} \Pi^{eq}_{\boldsymbol{\lambda}(t)}(x) \right) \left(\frac{\partial}{\partial \lambda_j} \Pi^{eq}_{\boldsymbol{\lambda}(t)}(x) \right). \tag{2.84}$$

The advantage of this formula is that it is entirely local in x, depending only on $\rho^{eq}_{\boldsymbol{\lambda}(t)}$ and not on $G_{\boldsymbol{\lambda}(t)}$, which is nonlocal in x and contains all the natural timescales of the system.

In the case of a harmonic potential, it can be checked by explicit calculation that $\boldsymbol{\zeta}$ and $\boldsymbol{\chi}$ are identical. For more general potentials, we show that in a certain limit, Eq. 2.72 can be written as Eq. 2.84 plus correction terms.

For this part of the discussion only, we restrict ourselves to potentials of the form

$$V_{\boldsymbol{\lambda}(t)}(x) = g(x) + \sum_{i=1}^{m} a_i x^i, \tag{2.85}$$

where $m \geq 4$ is even, and $a_m > 0$. The a_i are functions of $\boldsymbol{\lambda}(t)$. $g(x)$ is any function of x and $\boldsymbol{\lambda}$ that is finite in the limit $|x| \to \infty$. At large x, this potential is dominated by the x^m term. In fact, it contains a natural length scale, x_0, defined as the value of x at which the ratio $V_{\boldsymbol{\lambda}(t)}(x_0)/a_m x_0^m$ is of order 1. For such a potential, it is the case that

$$\lim_{|x|\to\infty} e^{\beta V_{\boldsymbol{\lambda}(t)}(x)/2} \frac{\partial}{\partial \lambda_i} \Pi^{eq}_{\boldsymbol{\lambda}(t)}(x) = 0, \tag{2.86}$$

and integrals over x of the quantity in the limit converge. This can be established using the asymptotic expansion of $1 - \Pi^{eq}_{\boldsymbol{\lambda}(t)}(x_0) \sim \int_{x_0}^{\infty} dy\, e^{-\beta a_m y^m}$:

$$\int_{x_0}^{\infty} dy\, e^{-\beta a_m y^m} \approx \frac{e^{-\beta a_m x_0^m}}{x_0^{m-1}} \left(1 + \mathcal{O}\left(\frac{1}{x_0}\right)\right). \tag{2.87}$$

The first term in the expansion can be verified by differentiating both sides of Eq.2.87 with respect to x_0.

In the following, we drop the subscript $\boldsymbol{\lambda}(t)$ for brevity. We use the notation $\zeta_{ij}^{x_0}$ and $\chi_{ij}^{x_0}$ to denote Eqs. 2.72 and 2.84 with all integrals evaluated between $-x_0$ and x_0.

Using $\partial_x \Pi^{eq}(x) = \rho^{eq}(x)$, Eq. 2.72 can be rewritten as

$$\zeta_{ij}^{x_0} = -\int_{-x_0}^{x_0} dx' dx''\, \frac{\partial^2 \Pi^{eq}(x'')}{\partial \lambda_i \partial x''}\, \frac{G(x';x'')}{\rho^{eq}_{\boldsymbol{\lambda}(t)}(x')}\, \frac{\partial^2 \Pi^{eq}(x')}{\partial \lambda_j \partial x'}. \tag{2.88}$$

Integrating by parts twice, this is

$$\zeta_{ij}^{x_0} = -\int_{-x_0}^{x_0} dx' dx''\, \frac{\partial \Pi^{eq}(x'')}{\partial \lambda_i}\, \Theta(x', x'')\, \frac{\partial \Pi^{eq}(x')}{\partial \lambda_j}, \tag{2.89}$$

where

$$\Theta(x', x'') = \frac{\partial^2}{\partial x' \partial x''} \frac{G(x';x'')}{\rho^{eq}(x')}. \tag{2.90}$$

For potentials of the form Eq. 2.85, the boundary terms in Eq. 2.89 are exponentially suppressed in x_0, that is, they are of order $e^{-\beta a_m x_0^m}$. Opening out the derivatives in Θ, we find that it satisfies the differential equation

$$\frac{1}{\gamma\beta} \frac{\partial}{\partial x'} \rho^{eq}(x') \Theta(x', x'') = \hat{\mathcal{L}}(x') \frac{\partial G(x';x'')}{\partial x''}. \tag{2.91}$$

Applying Eq. 2.41, this is

$$\frac{\partial}{\partial x'}\left(\rho^{eq}(x')\Theta(x',x'')+\gamma\beta\delta\left(x'-x''\right)\right)=0. \tag{2.92}$$

The solution to this differential equation is a family of functions $h_{x'}(x'')$ parameterized by x'. We choose to work with h evaluated at $x'=x_0$, henceforth notated simply as $h(x'')$:

$$h(x'')=\rho^{eq}(x_0)\Theta(x_0,x'')+\gamma\beta\;\delta(x_0-x''). \tag{2.93}$$

In terms of h, Eq. 2.90 can be written as

$$\Theta(x',x'')=\frac{1}{\rho^{eq}(x')}\left(-\beta\gamma\;\delta(x'-x'')+h(x'')\right). \tag{2.94}$$

Substituting this in Eq. 2.89, we find

$$\zeta_{ij}^{x_0}=\chi_{ij}^{x_0}+\Delta_{ij}^{x_0}, \tag{2.95}$$

where

$$\Delta_{ij}^{x_0}=-\int_{-x_0}^{x_0}dx'dx''\;\frac{\beta\gamma}{\rho^{eq}(x')}\frac{\partial\Pi^{eq}(x')}{\partial\lambda_j}h(x'')\frac{\partial\Pi^{eq}(x'')}{\partial\lambda_i}. \tag{2.96}$$

Once again using the asymptotic expansion Eq. 2.87, it can be shown that $\Delta_{ij}^{x_0}$ is of order $e^{-\beta a_m x_0^m}$. We note that it is necessary to evaluate the function $h_{x'}$ at $x'\geq x_0$ to arrive at this conclusion, otherwise it is not clear how to estimate the size of $\Delta_{ij}^{x_0}$. Therefore we finally arrive at

$$\zeta_{ij}^{x_0}=\chi_{ij}^{x_0}+\mathcal{O}(e^{-\beta a_m x_0^m}). \tag{2.97}$$

From Eq. 2.97, we see that in the limit $|x_0|\to\infty$, all correction terms go to zero, and we have $\zeta_{ij}-\chi_{ij}\to 0$. However, this limit is not physically valid—it is simple to check that as $x_0\to\infty$, Eq.2.1 is trivialized to $0=0$. Thus, for general potentials, we cannot expect the two formulae $\boldsymbol{\zeta}$ and $\boldsymbol{\chi}$ to be equivalent. As previously mentioned, the quadratic potential is an interesting exception for which it can be explicitly checked that both $\boldsymbol{\zeta}$ and $\boldsymbol{\chi}$ evaluate to the same quantity.

The calculation leading to Eq. 2.95 is a proof of the formula Eq. 2.84 for polynomial potentials. In [154], the class of potentials for which Eq. 2.84 converges was not established. We further note that we expect a relation similar to Eq. 2.97 to hold for potentials that grow faster than Eq. 2.85; for example, $V(x)=e^{b|x|}$ with $b>0$. The specifics of the asymptotic analysis proving this point will differ from what is presented here.

2.3 The harmonic oscillator in an electric field

We calculate $\boldsymbol{\zeta}$ for a one-dimensional system of charge q in a harmonic potential with time-dependent spring constant $\kappa(t)$ and subject to an external electric field $E(t)$. The control vector is $\boldsymbol{\lambda}(t)=(\kappa(t),E(t))$, where $\kappa>0$ and $E\in\mathbb{R}$. The potential is

$$V_{\boldsymbol{\lambda}(t)}(x)=\frac{1}{2}\kappa x^2-qEx=\frac{1}{2}\kappa\left(x-\theta\right)^2-\frac{\kappa}{2}\theta^2. \tag{2.98}$$

In the second equality we have defined the new variable $\theta = E/\kappa$. The electric field can be interpreted as an offset in the center of the harmonic trap.

The Fokker-Planck operator for this system is

$$\hat{\mathcal{L}}_{\boldsymbol{\lambda}(t)}(x) = \frac{1}{\gamma}\left[\kappa(t) + \kappa(t)\left(x - \theta(t)\right)\frac{\partial}{\partial x} + \frac{1}{\beta}\frac{\partial^2}{\partial x^2}\right]. \tag{2.99}$$

The eigenfunctions ψ_n of the corresponding Schrödinger operator are given by the Hermite functions [101]. Using H_n to denote the n^{th} Hermite polynomial, the right and left eigenfunctions are

$$\rho_{r,n}(x) = \frac{1}{\sqrt{2^n n!}}\sqrt{\frac{\beta\kappa}{2\pi}}e^{-\frac{1}{2}\beta\kappa(x-\theta)^2}H_n\left(\sqrt{\frac{\beta\kappa}{2}}(x-\theta)\right), \tag{2.100a}$$

$$\rho_{l,n}(x) = \frac{1}{\sqrt{2^n n!}}H_n\left(\sqrt{\frac{\beta\kappa}{2}}(x-\theta)\right). \tag{2.100b}$$

The corresponding eigenvalues are $-\kappa n/\gamma$. The equilibrium distribution at any given time t is a normalized Gaussian distribution with mean θ and variance $1/\beta\kappa$:

$$\rho^{eq}_{\boldsymbol{\lambda}(t)}(x) = \sqrt{\frac{\beta\kappa}{2\pi}}e^{-\frac{1}{2}\beta\kappa(x-\theta)^2}. \tag{2.101}$$

We proceed to calculate the four elements, beginning with $\zeta_{11} = \zeta_{\kappa\kappa}$:

$$\begin{aligned}\zeta_{\kappa\kappa} = -\int dx \int dy \sqrt{\frac{\beta\kappa}{2\pi}}e^{-\frac{1}{2}\beta\kappa(y-\theta)^2}\left(\frac{1}{2\kappa} - \frac{\beta(x-\theta)^2}{2}\right)\left(\frac{1}{2\kappa} - \frac{\beta(y-\theta)^2}{2}\right)\\ \sum_{n\neq 0} -\frac{\gamma}{\kappa n}\frac{1}{2^n n!}\sqrt{\frac{\beta\kappa}{2\pi}}e^{-\frac{1}{2}\beta\kappa(x-\theta)^2}H_n\left(\sqrt{\frac{\beta\kappa}{2}}(x-\theta)\right)H_n\left(\sqrt{\frac{\beta\kappa}{2}}(y-\theta)\right).\end{aligned} \tag{2.102}$$

Transforming to the variables $x' = \sqrt{\beta\kappa/2}(x-\theta), y' = \sqrt{\beta\kappa/2}(y-\theta)$, and using $\frac{1}{2} - x'^2 = -\frac{1}{4}H_2(x')$, this is

$$\zeta_{\kappa\kappa} = \frac{1}{\pi}\frac{\gamma}{\kappa^3}\frac{1}{16}\sum_{n\neq 0}\frac{1}{n 2^n n!}\left(\int dx'\, e^{-x'^2}H_2(x')H_n(x')\right)^2. \tag{2.103}$$

Applying the orthogonality property

$$\int dx'\, e^{-x'^2}H_m(x')H_n(x') = \delta_{mn}2^n n!\sqrt{\pi}, \tag{2.104}$$

we have

$$\zeta_{\kappa\kappa} = \frac{\gamma}{4\kappa^3}. \tag{2.105}$$

Similarly, the elements $\zeta_{\theta\kappa}$ and $\zeta_{\kappa\theta}$ are proportional to the product

$$\int dx'\, e^{-x'^2}\frac{1}{4}H_2(x')H_n(x')\int dy'\, e^{-y'^2}\frac{1}{2}H_1(y')H_n(y'), \tag{2.106}$$

which evaluates to zero for all n. Finally

$$\zeta_{\theta\theta} = \frac{2\beta\gamma}{\pi}\sum_{n\neq 0}\frac{1}{n2^n n!}\left(\int dx'\,\frac{1}{2}H_1(x')H_n(x')\right)^2 = \beta\gamma. \tag{2.107}$$

Gathering elements, we have

$$\zeta = \gamma\begin{bmatrix}(4\kappa^3)^{-1} & 0\\ 0 & \beta\end{bmatrix}. \tag{2.108}$$

As mentioned in the previous section, the same result is obtained by evaluating Eq. 2.84 for this system. Eq. 2.108 is also identical to the result obtained by evaluating the formula for a thermodynamic metric given in [127] for a harmonic potential with time-varying spring constant and trap center.

We can now calculate optimal protocols for the harmonic oscillator. For the metric Eq. 2.108, Eq. 2.71 takes the form

$$\int_0^\Omega dt\,\gamma\left(\frac{\dot\kappa^2}{4\kappa^3}+\beta\dot\theta^2\right) = \int_0^\Omega dt\,\gamma\left(\dot\mu^2+\beta\dot\theta^2\right). \tag{2.109}$$

In the second equality above we have made the change of variables $\mu = \sqrt{\kappa}$. This is a diffeomorphism for $\kappa > 0$. From Eq. 2.109 it is clear that the potential Eq. 2.98 gives rise to a flat geometry in (μ,θ)-space. However, the protocols have a nontrivial structure in the physical control parameter space (κ,θ) due to the existence of the forbidden region $\kappa \le 0$. The Euler-Lagrange equations corresponding to Eq. 2.109 are $\ddot\mu = \ddot\theta = 0$. The solutions are straight lines in the (μ,θ) plane. Given initial and final values of the physical parameters—κ_Ω and κ_0, and similarly for θ—the protocol that minimizes Eq. 2.71 is

$$\theta^{\rm opt}(t) = \frac{\theta_\Omega-\theta_0}{\Omega}t+\theta_0 \tag{2.110a}$$

$$\kappa^{\rm opt}(t) = \left(\frac{\sqrt{\kappa_\Omega}-\sqrt{\kappa_0}}{\Omega}t+\sqrt{\kappa_0}\right)^2. \tag{2.110b}$$

The optimal protocol demands a constant rate of change for θ and $\sqrt{\kappa}$.

In this example, we can explicitly check the consistency conditions of Section 2.1. To do so, it is convenient to rescale the optimal control problem so that all control parameters are dimensionless. This is easily done by first rescaling $x \to x/\ell$ where the length measure ℓ is defined by $\beta\ell^2\kappa = 1 = E\ell\beta$, as discussed at the end of Section 2.2, and then multiplying the potential (Eq. 2.98) by β. These rescalings do not disturb the optimal control problem.

We have the following optimal protocols for the dimensionless control parameters $\left(\tilde{\mu}, \tilde{\theta}\right) = \left(\sqrt{\beta\kappa\ell^2}, E/\kappa\ell\right)$:

$$\tilde{\theta}^{\text{opt}}(t) = \frac{\tilde{\theta}_\Omega - \tilde{\theta}_0}{\Omega} t + \tilde{\theta}_0 \tag{2.111a}$$

$$\tilde{\mu}^{\text{opt}}(t) = \frac{\tilde{\mu}_\Omega - \tilde{\mu}_0}{\Omega} t + \tilde{\mu}_0. \tag{2.111b}$$

These are of precisely the same form as Eq. 2.110. In terms of the dimensionless control parameters, the eigenvalues of the Fokker-Planck operator for the harmonic oscillator are given by $-\tilde{\mu}^2 n/\beta\ell^2\gamma$. Therefore, under the optimal protocol, the relaxation time of the Brownian system is $\tau_{\alpha_1} = \beta\ell^2\gamma/\left(\tilde{\mu}^{\text{opt}}\right)^2$.

Without loss of generality, we can assume $\dot{\tilde{\mu}}^{\text{opt}} \geq \dot{\tilde{\theta}}^{\text{opt}}$. For ease of notation in what follows, we write the difference $\tilde{\mu}_\Omega - \tilde{\mu}_0$ as $\Delta\tilde{\mu}$. The longest driving timescale set by the optimal protocol is then given by $\tau_\lambda = 1/\dot{\tilde{\mu}}^{\text{opt}} = \Omega/\Delta\tilde{\mu}$.

Therefore we have

$$\nu = \frac{\tau_{\alpha_1}}{\tau_\lambda} = \frac{\beta\ell^2\gamma}{\left(\tilde{\mu}^{\text{opt}}\right)^2}\frac{\Delta\tilde{\mu}}{\Omega} = \mathcal{O}\left(\frac{1}{\Omega}\right). \tag{2.112}$$

ν can be made small by choosing Ω, the duration of the protocol, to be sufficiently long.

From Eq. 2.111, we see that $\dot{\tilde{\mu}}^{\text{opt}}$ is of order $1/\Omega$. The rate of change of the spectrum of the Fokker-Planck operator, too, goes as $1/\Omega$. To see this, note that $|\dot{\alpha}_1| = 1/\tau_{\alpha_1}$. Differentiating this with respect to time, we find $|\dot{\alpha}_1| = 2\dot{\tilde{\mu}}^{\text{opt}}\tilde{\mu}^{\text{opt}}/\beta\ell^2\gamma = \mathcal{O}(1/\Omega)$ since $\dot{\tilde{\mu}}^{\text{opt}}$ is $\mathcal{O}(1/\Omega)$. Thus, both the control parameters and the spectrum of the Fokker-Planck operator vary appreciably only on the timescale of the control parameters, and are roughly constant on the timescale of the system if Ω is chosen to be large.

Lastly, differentiating Eq. 2.112 with respect to time, we find that $\dot{\nu}$ is of order $1/\Omega^2$, i.e., $\mathcal{O}(\nu^2)$, and is therefore suppressed on the control timescale.

2.4 Summary and discussion

We have developed a precise perturbative solution to Eq. 2.1 and used it to calculate the heat generated in the environment when the external parameters of a small stochastic system are varied in time. In so doing, we derived a new formula for the thermodynamic metric and all correction terms at the same order in the perturbation theory.

Both [127] and [154] propose formulae for thermodynamic metrics but do not establish the class of potentials for which those formulae are valid. The formula we have derived, Eq. 2.72, holds for potentials $V_{\boldsymbol{\lambda}(t)}$ such that both $V_{\boldsymbol{\lambda}(t)}$ and the associated Schrödinger potential $U_{\boldsymbol{\lambda}(t)}$ are confining. We have shown that for a subset of such potentials, namely, those in Eq. 2.85, the formula Eq.2.84 of [154] is approximately valid.

We found that the expansion in ν has an emergent local diffeomorphism symmetry not present in the original formula, Eq.2.55, for average heat production. Every term of this expansion transforms as a tensor of this diffeomorphism symmetry. Restricting to the symmetric 2-tensor (metric) in the expansion, we explicitly worked out the equations for an optimal protocol. These equations of motion describe geodesics in the space of control parameters.

Additional directions for future research include extending the perturbation theory to underdamped systems, and to higher spatial dimensions. For the latter, much of the formalism developed here will be applicable but it will be necessary to study the spectral properties of the Schrödinger operator in higher dimensions.

In this paper we derived a formula for the thermodynamic metric corresponding to the confining potential $U_{\boldsymbol{\lambda}(t)}$. This invites the following question: given a metric, what is the class of potentials that give rise to it? This may be especially interesting and tractable in the case of two-dimensional Riemannian geometries.

Chapter 3

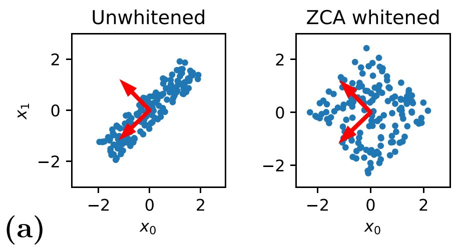
On whitening, exact second-order optimization, and generalization

In the following, we study the generalization properties of models trained with first-order optimizers on whitened data. We find that whitening negatively impacts generalization in a dimension-dependent manner, with the effect worsening in high dimensional settings. Exploiting the equivalence of training with a first-order method on whitened data and training with an exact second-order method on unwhitened data in certain model classes, we then study the generalization properties of models trained with exact second-order optimizers. We find that such models also exhibit reduced generalization compared with the same models trained with first-order optimizers, but that this effect can be countered by appropriately regularizing the optimizer.

What is whitening?

Whitening is a data preprocessing step that removes correlations between input features (see Fig. 3.1). It is used across many scientific disciplines, including geology [53], physics [68], machine learning [76], linguistics [1], and chemistry [29]. It has a particularly rich history in neuroscience, where it has been proposed as a mechanism by which biological visual systems realize Barlow's redundancy reduction hypothesis [14, 18, 13, 40, 125].

Whitening is often recommended since, by standardizing the variances in each direction in feature space, it typically speeds up the convergence of learning algorithms [76, 143], and causes models to better capture contributions from low variance feature directions. Whitening can also encourage models to focus on more fundamental higher-order statistics in data, by removing second-order statistics [65]. Whitening has further been a direct inspiration for deep learning techniques such as batch normalization [66] and dynamical isometry [98, 147].



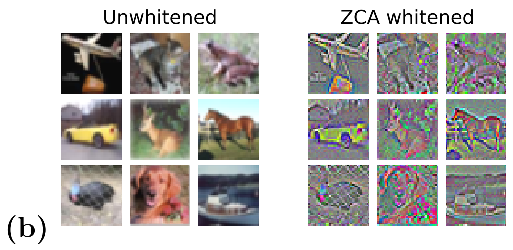


Figure 3.1: **Whitening removes correlations between feature dimensions in a dataset.** Whitening is a linear transformation of a dataset that sets all non-zero eigenvalues of the covariance matrix to 1. ZCA whitening is a specific choice of the linear transformation that rescales the data in the directions given by the eigenvectors of the covariance matrix, but without additional rotations or flips. *(a)* A toy 2d dataset before and after ZCA whitening. Red arrows indicate the eigenvectors of the covariance matrix of the unwhitened data. *(b)* ZCA whitening of CIFAR-10 images preserves spatial and chromatic structure, while equalizing the variance across all feature directions.

Whitening destroys information useful for generalization

Our argument proceeds in two parts: First, we prove that when a model with an isotropically initialized, fully connected first layer is trained with either gradient descent or stochastic gradient descent (SGD), information in the data covariance matrix is the *only* information that can be used to generalize. This result is agnostic to the choice of loss function and to the architecture of the model after the first layer. Second, we show that whitening always destroys information in the data covariance matrix.

Whitening the data and then training with gradient descent or SGD therefore results in either diminished or nonexistent generalization properties compared to the same model trained on unwhitened data. The seriousness of the effect varies with the difference between the number of datapoints n and the number of features d, worsening as $n - d$ gets smaller.

Empirically, we find that this effect holds even when the first layer is not fully connected and when its weight initialization is not isotropic - for example, in a convolutional network trained from a Xavier initialization.

Second-order optimization harms generalization similarly to whitening

Second-order optimization algorithms take advantage of information about the curvature of the loss landscape to take a more direct route to a minimum [28, 27]. There are many approaches to second-order or quasi second-order optimization [31, 50, 54, 122, 43, 81, 114, 79, 131, 86, 34, 137, 47, 132, 150, 61, 33, 69, 129, 44, 88, 55, 4, 153, 26, 87, 52, 82, 25, 56, 123, 23, 11, 5, 96], and there is active debate over whether second-order optimization harms generalization [144, 152, 151, 8, 136]. The measure of curvature used in these algorithms is often related to feature-feature covariance matrices of the input or of intermediate activations

[88]. In some situations second-order optimization is equivalent to steepest descent training on whitened data [128, 88].

We take advantage of the similarities between whitening and second-order optimization to argue that exact second-order optimization also prevents information about the input distribution from being leveraged during training, and can harm generalization (see Figs. 3.4, 3.6). Our results are strongest for unregularized, exact second-order optimizers and for the large width limit of neural networks. We do find, however, that when strongly regularized and carefully tuned, second-order methods can lead to superior performance (Fig. 3.7).

3.1 On whitening and generalization

Consider a dataset $X \in \mathbb{R}^{d\times n}$ consisting of n independent d-dimensional examples. X consists of samples from an underlying data distribution to which we do not have access. We write F for the feature-feature second moment matrix and K for the sample-sample second moment matrix:

$$F = XX^\top \in \mathbb{R}^{d\times d}\,, \quad K = X^\top X \in \mathbb{R}^{n\times n}\,. \tag{3.1}$$

We assume that at least one of F or K is full rank. We omit normalization factors of $1/n$ and $1/d$ in the definitions of F and K, respectively, for notational simplicity in later sections. As defined, K is also the Gram matrix of X.

We are interested in understanding the effect of whitening on the performance of a trained model when evaluated on a test set. We begin by proving the general result that for any model with a dense, isotropically initialized first layer, the trained model depends on the training inputs only through K. This is Part A. Then, in Part B, we show that whitening reduces the information in K. Together, these two results lead to the conclusion that whitening limits generalization. We discuss the resulting predictions in various cases depending on the dimensionality of the data and how the whitening transform is computed. Lastly, in Part C, we discuss all our results in the context of linear least squares models. These models can be exactly solved, and are productive of insight into why whitening might be expected to negatively affect generalization.

Part A: Data dependence of training dynamics and test predictions

Training dynamics depend on the training data only through its second moments

Consider a model f with a dense first layer Z:

$$f(X) = g_\theta(Z)\,, \quad Z = WX\,, \tag{3.2}$$

where W denotes the first layer weights and θ denotes all remaining parameters (see Fig. 3.3(a)). The structure of $g_\theta\left(\cdot\right)$ is unrestricted. W is initialized from an isotropic distribution. We

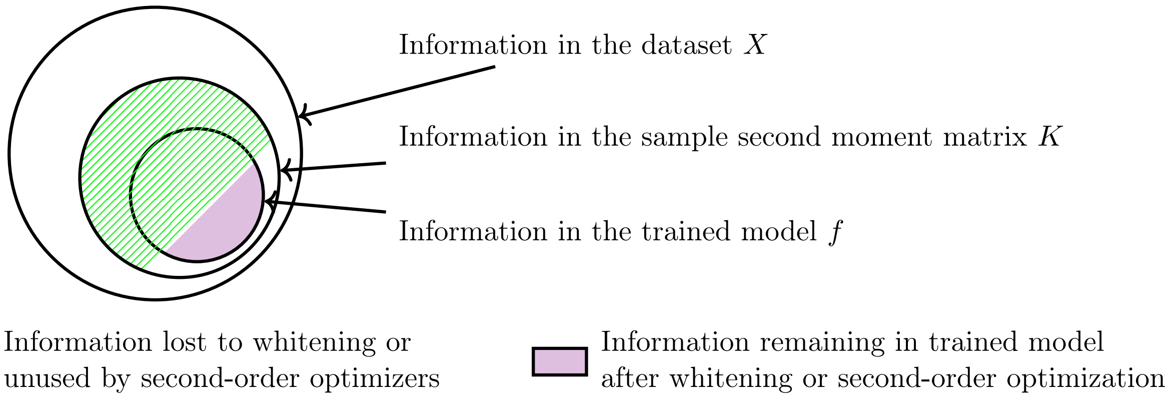

Information in the dataset X
Information in the sample second moment matrix K
Information in the trained model f
▨ Information lost to whitening or unused by second-order optimizers
🟪 Information remaining in trained model after whitening or second-order optimization

Figure 3.2: **A Venn diagram summarizing the core result.** For any model with a dense, isotropically initialized first layer, only information contained in the sample second moment matrix K of the data is available to inform model predictions. Trained models learn a subset of the information contained in K. Both data whitening and second-order optimization make information in K unavailable. This reduces the information about the dataset available to the model, and often harms generalization. In some cases all information in K is rendered unavailable, in which case the trained model is completely unable to generalize.

study a supervised learning problem, in which each vector X_i corresponds to a label Y_i.[1] We adopt the notation $X_{\text{train}} \in \mathbb{R}^{d \times n_{\text{train}}}$ and Y_{train} for the training inputs and labels, and write the corresponding second moment matrices as F_{train} and K_{train}. We consider models with loss $L(f(X); Y)$ trained by SGD. The update rules are

$$\theta^{t+1} = \theta^t - \eta \frac{\partial L^t}{\partial \theta^t}, \tag{3.3a}$$

$$W^{t+1} = W^t - \eta \frac{\partial L^t}{\partial W^t} = W^t - \eta \frac{\partial L^t}{\partial Z^t_{\text{train}}} X^\top_{\text{train}}, \tag{3.3b}$$

where t denotes the current training step, η is the learning rate, and L^t is the loss evaluated only on the minibatch used at step t. As a result, the activations $Z_{\text{train}} = W X_{\text{train}}$ evolve as

$$Z^{t+1}_{\text{train}} = Z^t_{\text{train}} - \eta \frac{\partial L^t}{\partial Z^t_{\text{train}}} K_{\text{train}}. \tag{3.4}$$

Treating the weights, activations, and function predictions as random variables, with distributions induced by the initial distribution over W^0, the update rules (Eqs. 3.3-3.4) can be represented by the causal diagram in Fig. 3.3(b). We can now state one of our main results.

[1]Our results also apply to unsupervised learning, which can be viewed as a special case of supervised learning where Y_i contains no information.

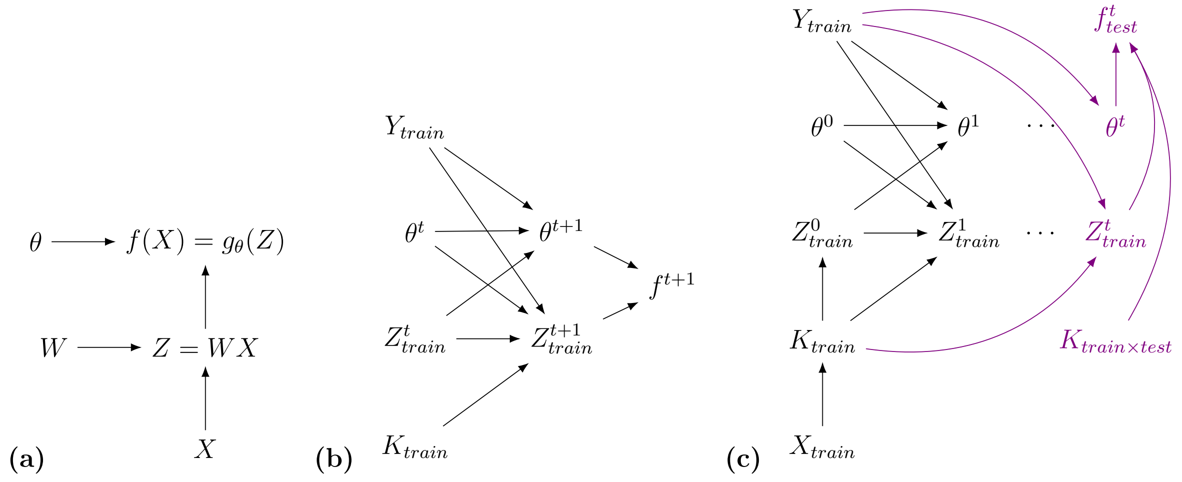


Figure 3.3: **Model activations and parameters depend on the training data only through second moments.** *(a)* Our model class consists of a linear transformation $Z = WX$, followed by a nonlinear map $g_\theta(Z)$ with parameters θ. Note that this model class includes fully connected neural networks, among other common machine learning models. *(b)* Causal dependencies for a single gradient descent update. The changes in weights, activations, and model output depend on the training data through the training sample second moment matrix, K_{train}, and the targets, Y_{train}. *(c)* Causal structure for the entire training trajectory. The final weights and training activations only depend on the training data through the training sample second moment matrix K_{train}, and the targets Y_{train}, while the test predictions (in purple) also depend on the mixed second moment matrix, $K_{\text{train}\times\text{test}}$.

Theorem 3.1.0.1. *Let $f(X)$ be a function as in Eq. 3.2, with linear first layer $Z = WX$, and additional parameters θ. Let W be initialized from an isotropic distribution. Further, let $f(X)$ be trained via gradient descent on a training dataset X_{train}. Then, the learned weights θ^t and first layer activations Z^t_{train} are independent of X_{train} conditioned on K_{train} and Y_{train}. In terms of mutual information $\mathcal{I}$, we have*

$$\mathcal{I}(Z^t_{train}, \theta^t; X_{train} \mid K_{train}, Y_{train}) = 0 \;\forall t. \tag{3.5}$$

Proof. To begin, we note that the first layer activation at initialization, Z^0_{train}, is a random variable due to random weight initialization, and only depends on X_{train} through K_{train}:

$$\mathcal{I}(Z^0_{\text{train}}; X_{\text{train}} \mid K_{\text{train}}) = 0. \tag{3.6}$$

This is a consequence of the isotropy of the initial weight distribution, and we will prove it later. Note also that the deeper layer weights at initialization are independent of X_{train}:

$$\mathcal{I}(\theta^0; X_{\text{train}}) = 0. \tag{3.7}$$

Using these two facts and the update rules Eqs. 3.3-3.4, the causal diagram for all of training is given by (the black part of) Fig. 3.3(c). It is clear that the conditional independence of Eq. 3.5 follows from this diagram, since the only additional data dependence of Z^t_{train} and θ^t over their counterparts at $t = 0$ is through Y_{train}. We provide an explicit argument by induction.

We have already established the base case. Assume Eq. 3.5 holds for $t = i$. By the chain rule,

$$\begin{aligned}\mathcal{I}(Z^{i+1}_{\text{train}},&\theta^{i+1}; X_{\text{train}} \mid K_{\text{train}}, Y_{\text{train}}) \\ &= \mathcal{I}(Z^{i+1}_{\text{train}}; X_{\text{train}} \mid K_{\text{train}}, Y_{\text{train}}) + \mathcal{I}(\theta^{i+1}; X_{\text{train}} \mid K_{\text{train}}, Y_{\text{train}}, Z^{i+1}_{\text{train}}).\end{aligned} \tag{3.8}$$

When K_{train} and Y_{train} are held fixed, the only sources of randomness in Z^{i+1}_{train} and θ^{i+1} are Z^i_{train} and θ^i. This can be seen by examining the update rules for Z_{train} and θ, Eqs. 3.4 and 3.3, in which we can write the loss function at time i as $L^i = L(g_{\theta^i}(Z^i_{\text{train}}); Y_{\text{train}})$. By assumption, Z^i_{train} and θ^i are conditionally independent of X_{train} given K_{train} and Y_{train}, and therefore both terms on the right-hand side of Eq. 3.8 evaluate to zero.

It remains to establish Eq. 3.6.

Let $\mathbb{P}(W^t)$ denote the probability distribution of the first layer weights at step t. We have assumed that W^0 is isotropically initialized: for any $R \in O(d)$ where $O(d)$ is the set of d-dimensional orthogonal matrices, we have

$$\mathbb{P}(W^0 R) = \mathbb{P}(W^0). \tag{3.9}$$

Now consider the conditional probability $\mathbb{P}(Z^0_{\text{train}}|X_{\text{train}})$. We can write it in terms of the distribution over initial weights, $P(Z^0_{\text{train}}|X_{\text{train}}) = \int DW^0 P(W^0)\delta(Z^0_{\text{train}} - W^0 X_{\text{train}})$, where DW^0 is the uniform measure over the components of W^0. Then, for any $R \in O(d)$, we have

$$\begin{aligned}\mathbb{P}(Z^0_{\text{train}}|RX_{\text{train}}) &= \int DW^0 \; \mathbb{P}(W^0)\delta(Z^0_{\text{train}} - W^0 R X_{\text{train}}) \\ &= \int D\tilde{W}^0 \; \mathbb{P}(\tilde{W}^0 R^\top)\delta(Z^0_{\text{train}} - \tilde{W}^0 X_{\text{train}}) \\ &= \int D\tilde{W}^0 \; \mathbb{P}(\tilde{W}^0)\delta(Z^0_{\text{train}} - \tilde{W}^0 X_{\text{train}}) \\ &= \mathbb{P}(Z^0_{\text{train}}|X_{\text{train}})\,.\end{aligned} \tag{3.10}$$

Here, δ denotes the Dirac delta function. To arrive at the second line we defined $\tilde{W}^0 := W^0 R$ and used the invariance of the measure DW^0. The third line follows from the $O(d)$ invariance of the initial weight distribution. Thus, the rotational invariance of the distribution over first layer weights leads to rotational invariance of the conditional distribution over first layer activations.

By the first fundamental theorem of invariant theory [73], the only $O(d)$ invariant functions of n vectors in d dimensions are the n^2 inner products $K_{\text{train}} = X^\top_{\text{train}} X_{\text{train}}$. Thus

$\mathbb{P}(Z^0_{\text{train}}|X_{\text{train}}) = h(K_{\text{train}})$ for some function h, and $\mathbb{P}(Z^0_{\text{train}}|X_{\text{train}}, K_{\text{train}}) = \mathbb{P}(Z^0_{\text{train}}|K_{\text{train}})$. Eq. 3.6 then follows from the definition of conditional mutual information.

$$\begin{aligned} I(Z^0_{\text{train}}&;X_{\text{train}} \mid K_{\text{train}}) \\ &\equiv \mathbb{E}_{K_{\text{train}}}\left[D_{\text{KL}}(\mathbb{P}(Z^0_{\text{train}}, X_{\text{train}}|K_{\text{train}})||\mathbb{P}(Z^0_{\text{train}}|K_{\text{train}})\mathbb{P}(X_{\text{train}}|K_{\text{train}}))\right] \\ &= \mathbb{E}_{K_{\text{train}}}\left[\mathbb{P}(X_{\text{train}}|K_{\text{train}})D_{\text{KL}}(\mathbb{P}(Z^0_{\text{train}}|X_{\text{train}}, K_{\text{train}})||\mathbb{P}(Z^0_{\text{train}}|K_{\text{train}}))\right] \\ &= 0. \end{aligned} \tag{3.11}$$

□

Test set predictions depend on train and test inputs only through their second moments

Let $X_{\text{test}} \in \mathbb{R}^{d\times n_{\text{test}}}$ and Y_{test} be the test data. The test predictions $f_{\text{test}} = f(X_{\text{test}})$ are determined by $Z^t_{\text{test}} = W^t X_{\text{test}}$ and θ^t. To identify sources of data dependence, we can write the evolution of the test set predictions Z_{test} over the course of training in a manner similar to Eq. 3.4:

$$Z^{t+1}_{\text{test}} = Z^t_{\text{test}} - \eta \frac{\partial L^t}{\partial Z^t_{\text{train}}} K_{\text{train}\times\text{test}}, \tag{3.12}$$

where $K_{\text{train}\times\text{test}} = X^\top_{\text{train}} X_{\text{test}} \in \mathbb{R}^{n_{\text{train}}\times n_{\text{test}}}$. The initial first layer activations are independent of the training data, and depend on X_{test} only through K_{test}. Therefore we have

$$\mathcal{I}(Z^0_{\text{test}}; X \mid K_{\text{test}}) = 0, \tag{3.13}$$

where X is the combined training and test data. If we denote the second moment matrix over this combined set by K, then the evolution of the test predictions is described by the (purple part of the) causal diagram in Fig. 3.3(c), from which we conclude the following.

Theorem 3.1.0.2. *For a function $f(X)$ as in Eq. 3.2, with first-layer weights initialized isotropically, trained with the update rules Eqs. 3.3-3.4, test predictions depend on the training data only through K and Y_{train}. This is summarized in the mutual information statement*

$$\mathcal{I}(f_{test}; X \mid K, Y_{train}) = 0\,. \tag{3.14}$$

Part B: Whitening and generalization

We have established that trained models with fully connected, isotropically initialized first layers depend on the input data only through K. Now we show that by removing information from F, whitening removes information in K that could otherwise be used to generalize. In the extreme case $n \le d$, K is trivialized, and we show that any generalization ability in a model trained in this regime relies solely on linear interpolation between inputs. We offer a detailed theoretical study of these effects in a linear model, and we use this example to make a connection with unregularized second-order optimization.

We begin with the definition of whitening.

Definition 3.1.0.1 (Whitening). *Any linear transformation M s.t. $\hat{X} = MX$ maps the eigenspectrum of F to ones and zeros, with the multiplicity of ones given by* $\operatorname{rank}(F)$*.*

It is natural to consider the two cases $n \leq d$ and $n \geq d$ (when $n = d$ both cases apply).

$$\begin{aligned} n \geq d: \;\; \hat{F} = I^{d \times d}, \quad \hat{K} &= \sum_{i=1}^{d} \hat{u}_i \hat{u}_i^\top . \\ n \leq d: \;\; \hat{F} = \sum_{j=1}^{n} \hat{v}_j \hat{v}_j^\top , \quad \hat{K} &= I^{n \times n}. \end{aligned} \tag{3.15}$$

Here, $\hat{F}$ and $\hat{K}$ denote the whitened second moment matrices, and the vectors $\hat{u}_i$ and $\hat{v}_j$ are orthogonal unit vectors of dimension n and d respectively. Eq. 3.15 follows directly from the fact that $X^\top X$ and $XX^\top$ share nonzero eigenvalues.

Full data whitening of a high dimensional dataset

We first consider a simplified setup: computing the whitening transform using the combined training and test data. We refer to this as "full-whitening". We consider the large feature count ($d \geq n$) regime.

Corollary 3.1.0.2.1. *When $d \geq n$, and when the whitening transform is computed on the full input dataset X (including both train and test points), then the whitened input data $\hat{X}$ provides no information about the predictions f_{test} of the model on test points. That is,*

$$\mathcal{I}(f_{test}; \hat{X} \mid Y_{train}) = 0 \,. \tag{3.16}$$

Proof. By Eq. 3.15 we have $\hat{K} = I$. Since $\hat{K}$ is now a constant rather than a random variable, Eq. 3.14 simplifies directly to Eq. 3.16. □

To further clarify this prediction, note that Eq. 3.16 implies $\mathcal{I}(f_{\text{test}}; Y_{\text{test}} \mid Y_{\text{train}}) = 0$ for fully-whitened data because the true test labels are solely determined by X_{test}. Therefore *knowing the model prediction on a test point in this setting gives no information about the true test label.*

Training data whitening of a high dimensional dataset

In practice, we are more interested in the common setting of computing a whitening transform based only on the training data. We call data whitened in this way "train-whitened". As mentioned above, the test predictions of a model are entirely determined by the first layer activations Z^t_{test} and the deeper layer weights θ^t. From Theorem 3.1.0.1 we see that the learned weights θ^t depend on the training data only through K_{train}, and are thus independent of the training data for whitened data:

$$\mathcal{I}(\theta^t; \hat{X}_{\text{train}} \mid Y_{\text{train}}) = 0 \,. \tag{3.17}$$

It is worth emphasizing this point because in most realistic networks the majority of model parameters are contained in these deeper weights θ^t.

Despite the deep layer weights, θ^t, being unable to extract information from the training distribution, the model is not entirely incapable of generalizing to test inputs. This is because the test activations Z_{test} will interpolate between training examples, using the information in $\hat{K}_{\text{train}\times\text{test}}$. More precisely,

$$Z^t_{\text{test}} = Z^0_{\text{test}} + \left(Z^t_{\text{train}} - Z^0_{\text{train}}\right)\hat{K}_{\text{train}\times\text{test}}. \tag{3.18}$$

This interpolation in Z is the *only* way in which structure in the inputs X_{train} can drive generalization. This should be contrasted with the case of full-whitening, discussed above, where $\hat{K}_{\text{train}\times\text{test}} = 0$. We therefore predict that when whitening is performed only on the training data, there will be some generalization, but it will be much more limited than can be achieved without whitening.

Full data whitening of lower dimensional datasets

When the dataset size is larger than the data dimensionality, whitening continues to remove information which could otherwise be used for generalization, but it no longer removes *all* of the information in the training inputs. In this regime, by mapping the feature-feature second moment matrix F to the identity matrix, whitening also reduces the degrees of freedom in the sample-sample second moment matrix K. Because information about the training dataset is available to the model only through K (Fig. 3.3(c)), reducing the degrees of freedom of K also reduces the information available to the model about the training inputs.

Theorem 3.1.0.3. *Consider a dataset $X \in \mathbb{R}^{d\times n}$, with $n > d$, and where all submatrices formed from d columns of X are full rank (this condition holds in the generic case). Consider the same model class and training procedure as in Theorem 3.1.0.1. Any dataset X can be compressed to $c \leq nd$ scalar values without losing any of the information that determines the distribution over the test set predictions f_{test} of the trained model. When models are trained on unwhitened data, then $c = \min\left(nd - {}^{(d^2-d)}\!/_2, {}^{(n^2+n)}\!/_2\right)$. However, when models are trained on whitened data, then the whitened dataset can be further compressed to $\hat{c} \leq (n-d)d$ scalars.*

Data whitening therefore reduces the amount of information about the input data that can be used to generate model predictions.

Proof. By Theorem 3.1.0.2 and the causal diagram in Fig. 3.3(c), we know that the distribution over test set predictions depends on model inputs only through K. Since K is positive semidefinite, it is fully specified by ${}^{(n^2+n)}\!/_2$ entries. For d < n these entries are not independent. K encodes the inner products between n vectors in d dimensions, These are specified by n magnitudes and $n(d-1) - (d(d-1))/2$ independent angles.

Thus, for a model trained on unwhitened data,

$$c = \min\left(nd - \frac{d^2-d}{2}, \frac{n^2+n}{2}\right). \tag{3.19}$$

Next we consider the case that full data whitening has been performed, such that $\hat{F} = I$. Recall the following identity, where $^+$ indicates the pseudoinverse:

$$\hat{X}^\top = \hat{X}^+ \hat{X} \hat{X}^\top. \tag{3.20}$$

Using this, we can rewrite $\hat{K}$ as

$$\begin{aligned}\hat{K} &= \hat{X}^\top \hat{X} = \hat{X}^+ \hat{X} \hat{X}^\top \hat{X} = \hat{X}^+ \hat{F} \hat{X} \\ &= \hat{X}^+ \hat{X}.\end{aligned} \tag{3.21}$$

Next consider a modified dataset

$$\widetilde{X} = Q\hat{X} = [I \cdots], \tag{3.22}$$

where the matrix $Q \in \mathbb{R}^{d \times d}$ has been chosen such that the first d columns of $\widetilde{X}$ correspond to the identity matrix (ie Q is the inverse of the submatrix formed by the first d columns of $\hat{X}$). Because its first d columns are deterministic, $\widetilde{X}$ can be stored using $(n-d)\,d$ real values. Despite being represented by d^2 fewer values, this compressed dataset can still be used to reconstruct $\hat{K}$,

$$\begin{aligned}\hat{K} &= \hat{X}^+ \hat{X} = \hat{X}^+ Q^{-1} Q \hat{X} = \left(Q\hat{X}\right)^+ Q\hat{X} \\ &= \widetilde{X}^+ \widetilde{X}.\end{aligned} \tag{3.23}$$

We further observe that when $n > d$, $(n-d)\,d < {(n^2+n)}/{2}$. This is enough to establish

$$\hat{c} = (n-d)\,d < c. \tag{3.24}$$

□

Summary of predictions and a note

In a model with a fully connected first layer, with first layer weights initialized from an isotropic distribution, whitening the data before training with SGD is expected to result in reduced generalization ability compared to the same model trained on unwhitened data. The severity of the effect varies with the relationship of n to d.

Full data whitening when $n < d$ is a limiting case in which generalization is expected to be completely destroyed. When $n \leq d$ and the data is train-whitened, generalization is forced to rely solely on interpolation and is expected to be poor. When $n > d$ and the data is either fully or train-whitened, model predictions still depend on strictly less information than would be available had the data not been whitened, and once again generalization is expected to suffer. For $n \gg d$, the effect of whitening on generalization is expected to be minimal.

As we discuss later, these same predictions apply to second-order optimization of linear models and of overparameterized networks (with d corresponding the number of parameters rather than the number of input dimensions).

In this work, the measure of generalization we are concerned with is the performance of a trained model on a test set. Although standard in the neural network community, this is different from the measure of generalization usually studied in statistical learning theory, which is the expected difference between the empirical and population risks. Interestingly, when this latter measure of generalization is considered, there are results establishing precisely the opposite of our results here, namely that both a reduction in the mutual information between the dataset and model output [148] and a greater compression of the dataset [42] are *beneficial* for generalization, albeit with different assumptions on the model class than the ones we have here. How to reconcile these two kinds of results is a question for future research.

Part C: Whitening in linear least squares models

Linear models are widely used for regression and prediction tasks and provide an instructive laboratory to understand the effects of data whitening. Studying them also provides intuition for why whitening is harmful.

Consider a linear model with mean squared error loss,

$$f(X) = W^\top X\,, \quad L = \frac{1}{2}||f(X) - Y||^2\,. \tag{3.25}$$

This loss function is convex. We begin by discussing the low dimensional case, $d < n$, where the loss has a unique global optimum $W^\star = F_{\text{train}}^{-1} X_{\text{train}} Y_{\text{train}}$. The model predictions at this global optimum, $f_\star(X) = W^{\star\top} X$, are invariant under any whitening transform (3.1.0.1). As a result, any quality metric (loss, accuracy, etc...) for this global minimum is the same for whitened and unwhitened data.

The story is more interesting, however, during training. Consider a model trained via gradient flow (similar statements can be made for gradient descent or stochastic gradient descent). The dynamics of the weights are given by

$$\frac{dW}{dt} = -\frac{\partial L}{\partial W}\,, \quad W(t) = e^{-tF_{\text{train}}} W(0) + (1 - e^{-tF_{\text{train}}}) W^*\,. \tag{3.26}$$

The evolution in Eq. 3.26 implies that the information contained in the trained weights $W(t)$ about the training data X is entirely determined by F_{train} and $W^\star$. In terms of mutual information, we have

$$\mathcal{I}(W(t); X | F_{\text{train}}, W^\star) = 0\,. \tag{3.27}$$

As whitening sets $\hat{F}_{\text{train}} = I$, a linear model trained on whitened data does not benefit from the information in F_{train}.

At a more microscopic level, we can decompose Eq. 3.26 in terms of the eigenvectors, v_i, of F:

$$W(t) = \sum_{i=1}^{d} v_i w_i(t),\ \ w_i(t) = e^{-t\lambda_i} w_i(0) + (1 - e^{-\lambda_i t}) w_i^\star\,. \tag{3.28}$$

We see that for unwhitened data the eigenmodes with larger eigenvalues converge more quickly towards the global optimum, while the small eigendirections converge relatively slowly. For centered X, F is the feature covariance and these eigendirections are exactly the principle components of the data. As a result, training on unwhitened data is biased towards learning the top principal directions at early times. This bias is often beneficial for generalization. Similar simplicity biases have been found empirically in deep linear networks [111] and in deep networks trained via SGD [100, 104] where networks learn low frequency modes before high. In contrast, for whitened data, $\hat{F}_{\text{train}} = I$ and the evolution of the weights takes the form

$$\hat{w}_i(t) = e^{-t}\hat{w}_i(0) + (1 - e^{-t})\hat{w}_i^\star \,. \tag{3.29}$$

All hierarchy between the principle directions has been removed, thus training overfits immediately. For this reason linear models trained on unwhitened data can generalize significantly better at finite times than the same models trained on whitened data. Empirical support for this in a linear image classification task with random features is shown in Fig. 3.4(a).

At the global optimum, $W^\star = F_{\text{train}}^{-1} X_{\text{train}} Y_{\text{train}}$, the network predictions on test points can be written in a few equivalent ways,

$$f_\star(X_{\text{test}}) = Y_{\text{train}}^\top X_{\text{train}}^\top F_{\text{train}}^{-1} X_{\text{test}} = Y_{\text{train}}^\top K_{\text{train}}^{+} K_{\text{train}\times\text{test}} = Y_{\text{train}}^\top \hat{K}_{\text{train}\times\text{test}} \,. \tag{3.30}$$

Here, the $+$ superscript is the pseudoinverse, and $\hat{K}_{\text{train}\times\text{test}}$ is the whitened train-test sample-sample second moment matrix. These expressions make manifest that the test predictions at the global optimum only depend on the training data through K_{train} and $K_{\text{train}\times\text{test}}$, and that the global optimum is the same regardless of whether the data is whitened. This can also be seen in Figs. 3.4(a) and 3.8.

The discussion is very similar in the high dimensional case, $d > n$. In this case, there is no longer a unique solution to the optimization problem, but there is still a unique optimum within the span of the data.

$$W_\parallel^\star = \left(F_{\text{train}}^\parallel\right)^{-1} X_{\text{train}}^\parallel Y_{\text{train}} \,, \quad W_\perp^\star = W_\perp(0) \,. \tag{3.31}$$

Here, we have introduced the notation $\parallel$ for directions in the span of the training data and $\perp$ for orthogonal directions. Explicitly, if we denote by $V^\parallel = \{v_1, v_2, \ldots, v_n\} \in \mathbb{R}^{n\times d}$ the non-null eigenvectors of F_{train} and $V^\perp = \{v_{n+1}, v_{n+2}, \ldots, v_d\} \in \mathbb{R}^{(d-n)\times d}$ the null eigenvectors, then $X_{\text{train}}^\parallel := V^\parallel X_{\text{train}}$, $W_\parallel := W V^\parallel$, $W_\perp := W V^\perp$, and $F_{\text{train}}^\parallel := V^\parallel F_{\text{train}} (V^\parallel)^\top$.

The model predictions at this optimum can be written as

$$f_\star(X_{\text{test}}) = f^0(X_{\text{test}}) - \left(f^0(X_{\text{train}}) - Y_{\text{train}}\right)^\top K_{\text{train}}^{-1} K_{\text{train}\times\text{test}} \,. \tag{3.32}$$

This is the solution found by GD, SGD, and projected Newton's method. The evolution approaching this optimum can be written (again assuming gradient flow for simplicity) as

$$W_\parallel(t) = e^{-tF_{\text{train}}^\parallel} W_\parallel(0) + (1 - e^{-tF_{\text{train}}^\parallel}) W_\parallel^* \,, \quad W_\perp(t) = W_\perp(0). \tag{3.33}$$

In terms of the individual components, $[W_\parallel(t)]_i = e^{-t\lambda_i}[W_\parallel(0)]_i + (1 - e^{-t\lambda_i})[W^*_\parallel]_i$.

As in the case $d < n$, the hierarchy in the spectrum allows for the possibility of beneficial early stopping, while whitening the data results in immediate overfitting.[2]

Let us briefly consider the case where $n \leq d$ with full data whitening. As discussed, the global optimum $W^\star$ is still unique up to a pseudoinverse: $W^\star = F^+_{\text{train}} X_{\text{train}} Y_{\text{train}}$. When full data whitening is performed, we have $\hat{K} = I$ from Eq. 3.15, and so the mixed second moment matrix $\hat{K}_{\text{train}\times\text{test}}$, which is an off-diagonal block of $\hat{K}$, is a zero matrix. Therefore $f_\star(X_{\text{test}}) = Y^\top_{\text{train}} \hat{K}_{\text{train}\times\text{test}} = 0$ for all the test points, and it is particularly simple to see how whitening can destroy generalization.

3.2 On exact second-order optimization and generalization by analogy with whitening

In Part A of Section 3.1, we argued that for any model with a dense, isotropically initialized first layer, the trained model depends on the training inputs only through K. In Part B, we demonstrated that whitening reduces the information in K and therefore negatively impacts generalization. Here, we show that exact second-order optimizers, typified by Newton's method, have reduced access to the information in K for certain model classes. Although in practice unregularized Newton's method is rarely used as an optimization algorithm due to its computational complexity, a poorly conditioned Hessian, or poor generalization performance, it is still important to study because it serves as the basis of and as a limiting case for most second-order methods.

It is known that in linear least squares models, training with Newton's method is exactly equivalent to training on whitened data with a first-order optimizer. We reproduce the argument that established this equivalence below. We further argue that highly overparameterized neural networks trained with Newton's method evolve as linear models trained with the same optimizer. By analogy with whitening, then, exact second-order optimizers are expected to produce linear models and overparameterized neural networks that do not generalize as well as the same models trained with first-order optimizers. Thus, we are able to give an explanation for why unregularized second-order methods have poor generalization performance. We find that our conclusions also hold empirically in a deep CNN (see Figs. 3.4, 3.6).

Linear least squares

We compare a pure Newton update step on unwhitened data with a gradient descent update step on whitened data in a linear least squares model. The Newton update step uses the

[2]The result discussed here, that whitening eliminates the benefits of early stopping, if there are any, should be distinguished from the phenomenon of benign overfitting, in which the optimal solution to a training problem still generalizes well, in defiance of the bias-variance tradeoff of classical statistics. Benign overfitting is a property of $W^\star$, and not of any solution to the training problem obtained by early stopping. See [21, 19, 59] for studies of benign overfitting in linear least squares problems.

Hessian H of the model as a preconditioner for the gradient:

$$W_{\text{Newton}}^{t+1} = W_{\text{Newton}}^{t} - \eta H^{-1} \frac{\partial L^t}{\partial W^t} \,. \tag{3.34}$$

We allow for a general step size η, with $\eta = 1$ giving the canonical Newton update. When H is rank deficient, we take H^{-1} to be a pseudoinverse. For a linear model with mean squared error (MSE) loss, the Hessian equals the second moment matrix F_{train}, and the model output evolves as

$$f_{\text{Newton}}^{t+1}(X) = f_{\text{Newton}}^{t}(X) - \eta \frac{\partial L^t}{\partial f_{\text{Newton}}^t} X_{\text{train}}^\top F_{\text{train}}^{-1} X \,. \tag{3.35}$$

We can compare this with the evolution of a linear model $\hat{f}(X) = \hat{W} M X$ trained via gradient descent on whitened data $\hat{X} = MX$ with a mean squared loss:

$$\hat{f}^{t+1}(X) = \hat{f}^{t}(X) - \eta \frac{\partial L^t}{\partial \hat{f}^t} X_{\text{train}}^\top M^\top M X. \tag{3.36}$$

Noting that $M^\top M = F_{\text{train}}^{-1}$, Eqs. 3.35 and 3.36 give identical update rules. Thus if both functions are initialized to have the same output, Newton updates give the same predictions as gradient descent on whitened data. While this correspondence is known in the literature, we can now use it to say something further, namely that by applying the argument in Section 3.1, Part B, we expect Newton's method to produce linear models that generalize poorly. This result assumes a mean squared loss, but we find experimentally that generalization is also harmed with a cross entropy loss in Fig. 3.4(d).

Overparameterized neural networks

In recent years much progress has been made in understanding the dynamics of wide neural networks [67]. In the large width limit, which occurs when the number of units or channels in intermediate network layers grows towards infinity, many neural network architectures, including fully connected and convolutional architectures, behave as linear models in their parameters throughout training. In particular, it has been realized that wide networks trained via GD, SGD, or gradient flow evolve as linear models with static, nonlinear features given by the derivative of the network map at initialization [78]. Here we extend the connection between linear models and wide networks to second-order methods. We argue that wide networks trained with a regularized Newton's method evolve as linear models trained with the same second-order optimizer.

In the wide network limit, d corresponds to the number of features rather than the number of input dimensions, and the number of features is equal to the number of parameters. Second-order optimization is therefore predicted to be harmful for much larger dataset sizes when optimizing overparameterized neural networks.

We consider a regularized Newton update step,

$$\theta^{t+1} = \theta^t - \eta \left(\epsilon \mathbf{1} + H\right)^{-1} \frac{\partial L^t}{\partial \theta} \,. \tag{3.37}$$

This diagonal regularization is a common generalization of Newton's method. One motivation for such an update rule in the case of very wide neural networks is that the Hessian is necessarily rank deficient, and so some form of regularization is needed.

For a linear model, $f_{\text{linear}}(x) = \theta^\top \cdot g(x)$, with fixed nonlinear features, $g(x)$, the regularized Newton update rule in weight space leads to the function space update

$$f^{t+1}_{\text{linear}}(x) = f^t_{\text{linear}}(x) - \eta \sum_{\substack{x_a, x_b \\ \in X_{\text{train}}}} \Theta_{\text{linear}}(x, x_a) \left(\epsilon \mathbf{1} + \Theta_{\text{linear}}\right)^{-1}_{ab} \frac{\partial L_b}{\partial f_{\text{linear}}} \,. \tag{3.38}$$

Here, Θ_{linear} is a constant kernel, $\Theta_{\text{linear}}(x, x') = \frac{\partial f}{\partial \theta}^\top \cdot \frac{\partial f}{\partial \theta} = g^\top(x) \cdot g(x')$.

For a deep neural network, the function space update takes the form

$$f^{t+1}(x) = f^t(x) - \eta \sum_{\substack{x_a, x_b \\ \in X_{\text{train}}}} \Theta(x, x_a) \left(\epsilon \mathbf{1} + \Theta\right)^{-1}_{ab} \frac{\partial L_b}{\partial f} + \frac{\eta^2}{2} \sum_{\mu,\nu=1}^{P} \frac{\partial^2 f}{\partial \theta_\mu \partial \theta_\nu} \Delta\theta^t_\mu \Delta\theta^t_\nu + \cdots \,. \tag{3.39}$$

Here we have indexed the model weights by $\mu = 1 \ldots P$, denoted the change in weights by $\Delta\theta^t$ and introduced the neural tangent kernel (NTK), $\Theta(x, x') = \frac{\partial f}{\partial \theta}^\top \cdot \frac{\partial f}{\partial \theta}$.

In general, Eqs. 3.38 and 3.39 lead to different network evolution due to the non-constancy of the NTK and the higher-order terms in the learning rate. For wide neural networks, however, it was realized that the NTK is constant [67] and the higher-order terms in η appearing in Eq. 3.39 vanish at large width [48, 63, 80, 10, 6].[3]

With these simplifications, the large width limit of Eq. 3.39 describes the same evolution as a linear model trained with fixed features $g(x) = \frac{\partial f(x)}{\partial \theta}|_{\theta=\theta_0}$ trained via a regularized Newton update.

3.3 Experiments

Model and Task Descriptions

We describe our basic experiment structure, and provide brief descriptions of the four types of models we studied and associated experimental variations here. Detailed methods are given in Section 3.5.

[3]These simplifications were originally derived for gradient flow, gradient descent and stochastic gradient descent, but hold equally well for the regularized Newton updates considered here. This can be seen, for example, by applying Theorem 1 of [48].

The kernel of all our experiments is as follows: From a dataset, we draw a number of subsets, tiling a range of dataset sizes. Each subset is divided into train, test, and validation examples, and three copies of the subset are made, two of which are whitened. In one case the whitening transform is computed using only the training examples (train-whitening), and in the other using the training, test, and validation examples (full-whitening). Note that the test set size must be reduced in order to run experiments on small datasets, since the test set is considered part of the dataset for full whitening. Models are trained from random initialization on each of the three copies of the data using the same training algorithm and stopping criterion. Test errors and the number of training epochs are recorded. We emphasize that in any single experiment in which whitening is performed, the same whitening transform is always applied to train, test, and validation data. Experiments differ in the specific subset of the data (train only or train + test + validation) on which the whitening transform is computed.

Linear models and MLPs. To experimentally demonstrate theoretical results, we study CIFAR-10 classification in linear models and CIFAR-10 and MNIST classification in three-layer, fully connected multilayer perceptrons (MLPs). Linear models were trained by optimizing mean squared error loss, where the model outputs were a linear map between the 512-dimensional outputs of a four layer convolutional network at random initialization on CIFAR-10, and their 10-dimensional one-hot labels. This setup is in part motivated by analogy to training the last linear readout layer of a deep neural network. We solved the gradient flow equation for the time at which the MSE on the validation set is lowest, and report the test error at that time. The experiment was repeated using continuous-time Newton's method, consisting of continuous-time gradient descent using an inverse Hessian preconditioner. MLPs were trained using SGD with constant step size until the training accuracy reaches a fixed cutoff threshold, at which point test accuracy was measured.

Convolutional networks. Since our theoretical results on the effect of whitening apply only to models with a fully connected and isotropically initialized first layer, we test whether the same qualitative behavior is observed in CNNs trained from a Xavier initialization. We chose the popular wide residual (WRN) architecture [149], trained on CIFAR-10. Training was performed using full batch gradient descent with a cosine learning rate schedule for a fixed number of epochs. Full batch training was used to remove experimental confounds from choosing minibatch sizes at different dataset sizes. A validation set was split from the CIFAR-10 training set. Test error corresponding to the parameter values with the lowest validation error was reported.

We also trained a smaller CNN (a ResNet-50 convolutional block followed by an average pooling layer and a dense linear layer) on unwhitened data with full batch gradient descent and with the Gauss-Newton method (with and without a scaled identity regularizer) to compare their respective generalization performances. A grid search was performed over learning rate, and step sizes were chosen using a backoff line search initialized at that learning

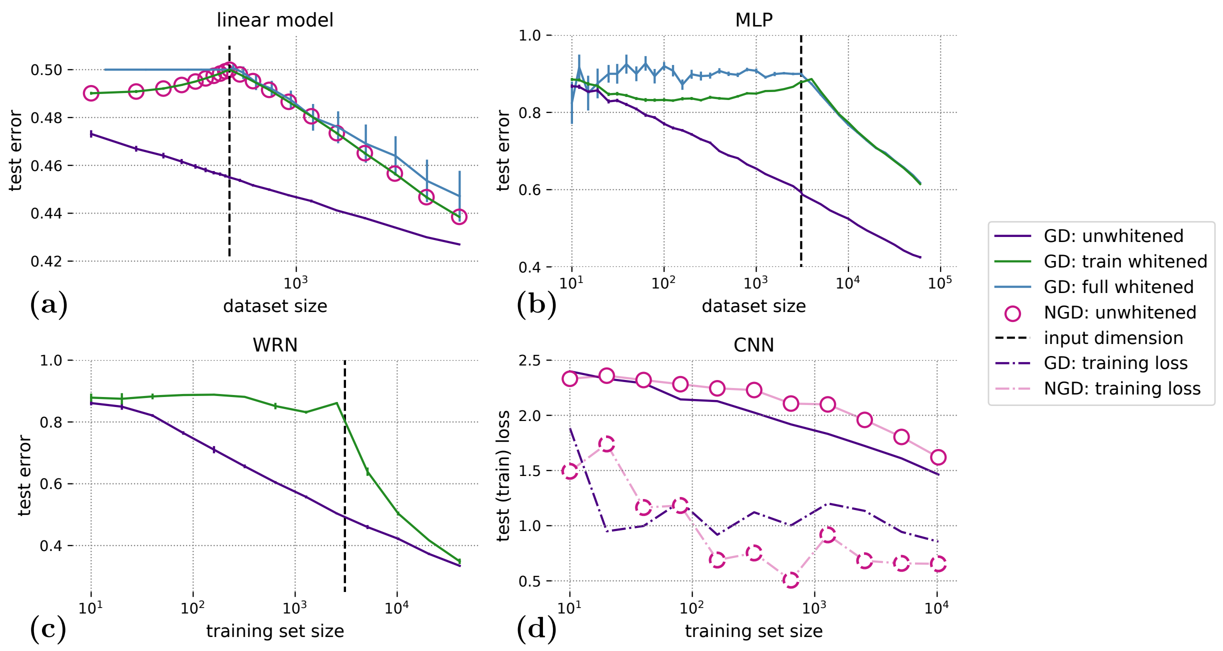


Figure 3.4: **Whitening and second-order optimization reduce or prevent generalization.** *(a)-(c)* Models trained on both full-whitened data (blue; panes a,b) and train-whitened data (green; panes a-c) consistently underperform models trained by gradient descent on unwhitened data (purple; all panes). In (a), Newton's method on unwhitened data (pink circles) behaves identically to gradient descent on whitened data. *(d)* second-order optimization in a convolutional network results in poorer generalization properties than steepest descent. Points plotted correspond to the learning rate and training step with the best validation loss for each method; data for this experiment was unwhitened. CIFAR-10 is used for all experiments (see Fig. 3.5 for experiments on MNIST). In (c) and (d) we use a cross entropy loss (see Section 3.5 for details).

rate. Test and training losses corresponding to the best achieved validation loss were reported. Note that this experiment is relatively large scale; because we perform full second-order optimization to avoid confounds due to choosing a quasi-Newton approximation, iterations are cubic in the number of model parameters.

Results

Whitening and second-order optimization impair generalization. In agreement with theory, in Figs. 3.4(a) and (b), linear models and MLPs trained on fully whitened data generalize at chance levels (indicated by test errors of 0.5 and 0.9, respectively) when the size of the dataset is smaller than the dimensionality of the data, and models trained on train-whitened data perform strictly worse than those trained on unwhitened data. Furthermore,

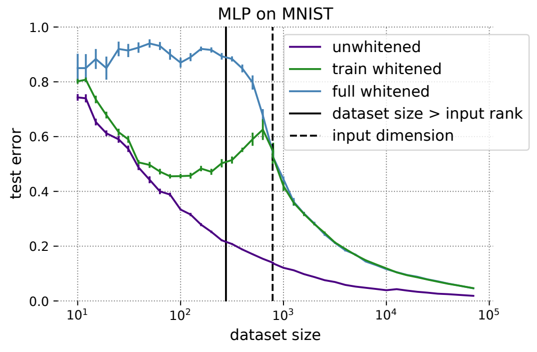


Figure 3.5: **Whitening MNIST before training negatively impacts generalization in MLPs.** Models trained on fully whitened data (in blue) are unable to generalize until the size of the dataset exceeds its maximal input rank of 276, indicated by the solid black vertical line. Regardless of how the whitening transform is computed, models trained on whitened data (blue and green) consistently underperform those trained on unwhitened data (in purple).

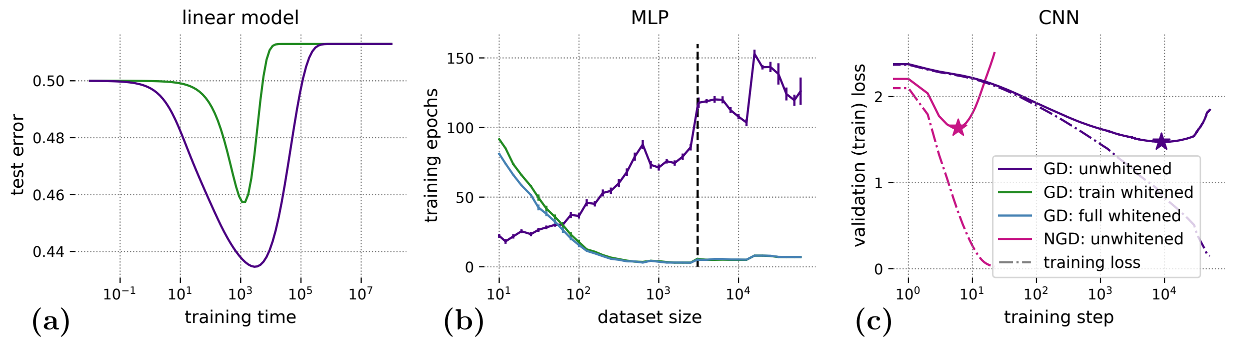


Figure 3.6: **Models trained on whitened data or with second-order optimizers converge faster.** *(a)* Linear models trained on whitened data optimize faster, but their best test accuracy was always worse. Data plotted here is for a training set of size 2560. Similar results for smaller training set sizes are given in Fig. 3.8. *(b)* Whitening the data significantly lowers the number of epochs needed to train an MLP to a fixed cutoff in training accuracy, when the learning rate and all other training parameters are kept constant. Discrete jumps in the plot data correspond to points at which the (constant) learning rate was changed. The dashed vertical line indicates the input dimensionality of the data. See Section 3.5 for details. *(c)* second-order optimization accelerates training on unwhitened data in a convolutional network, compared to gradient descent. Data shown is for a training set of size 10240. Stars correspond to values of the validation loss at which test and training losses are plotted in Fig. 3.4(d).

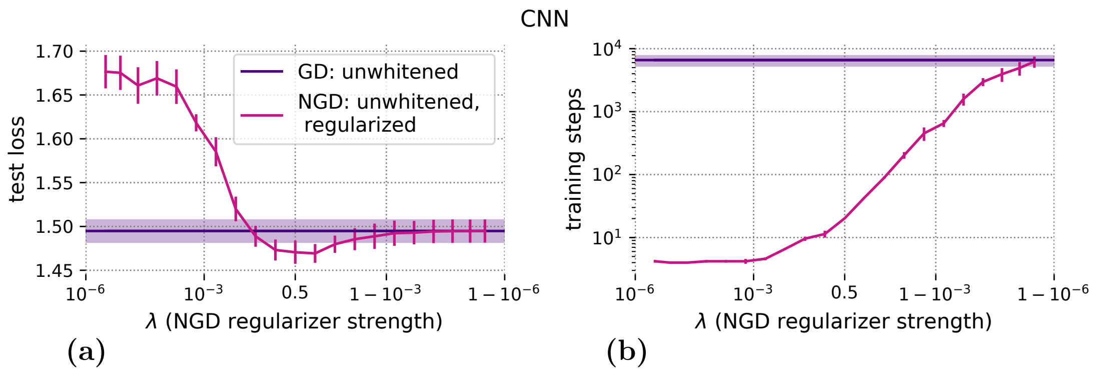


Figure 3.7: **Regularized second-order methods can train faster than gradient descent, with minimal or even positive impact on generalization.** Models were trained on a size 10240 subset of CIFAR-10 by minimizing a cross entropy loss. Error bars indicate twice the standard error in the mean. *(a)* Test loss as a function of regularizer strength. At intermediate values of λ, the second-order optimizer produces *lower* values of the test loss than gradient descent. Test loss is measured at the training step corresponding to the best validation performance for both algorithms. See text for further discussion. *(b)* At all values of $\lambda < 1$, the second-order optimizer requires fewer training steps to achieve its best validation performance.

the generalization ability of these models recovers only *gradually* as the dataset grows. On CIFAR-10, a 20% gap in performance between MLPs trained on whitened and unwhitened data persists even at the largest dataset size, suggesting that whitening can remain detrimental even when the number of training examples exceeds the number of features by an order of magnitude. Similar results are seen in Fig. 3.5 on MNIST, except that the generalization gap at the largest dataset size is much smaller than for CIFAR-10. Because MNIST is highly rank deficient, the high dimensional regime for this dataset is defined by $n < r$, where r is the maximal input rank of the dataset ($r < d$).

In Fig. 3.4(c) we see a generalization gap in the high dimensional regime between WRNs trained on train-whitened versus unwhitened data, which persists when the size of the dataset grows beyond its dimensionality. This is despite the fact that the convolutional input layer violates the theoretical requirement of a fully connected first layer, and that we used a Xavier initialization scheme, therefore also violating the theoretical requirement for an isotropic first-layer weight initialization. We note that these results are consistent with the whitening experiments in the original WRN paper [149]. Generalization ability begins to recover before the size of the training set reaches its input dimensionality, suggesting that the effect of whitening can be countered by engineering knowledge of the data statistics into the model architecture.

In Fig. 3.4(a), we demonstrate experimentally the correspondence we proved in the discussion on linear least squares models in Section 3.2. In Fig. 3.4(d), we observe that pure second-order optimization similarly harms generalization even in a convolutional network. Despite training to lower values of the training loss, a CNN trained with an unregularized Gauss-Newton method exhibits higher test loss (at the training step with best validation loss) than the same model trained with gradient descent.

Whitening and second-order optimization accelerate training. In Figs. 3.6(a) and 3.8, linear models trained on whitened data or with a second-order optimizer converge to their final loss faster than models trained on unwhitened data, but their best test performance is always worse. In Fig. 3.6(b), MLPs trained on whitened CIFAR-10 data take fewer epochs to reach the same training accuracy cutoff than models trained on unwhitened data, except at very small (< 50) dataset sizes. The effect is stark at large dataset sizes, where the gap in the number of training epochs is two orders of magnitude large. second-order optimization similarly speeds up training in a convolutional network. In Fig. 3.6(c), unregularized Gauss-Newton descent achieves its best validation loss two orders of magnitude faster (as measured in the number of training steps) than gradient descent.

Regularized second-order optimization can simultaneously accelerate training and improve generalization. In Fig. 3.7 we perform full batch second-order optimization with preconditioner $((1-\lambda)B + \lambda I)^{-1}$, where $\lambda \in [0,1]$ is a regularization coefficient, and B^{-1} is the unregularized Gauss-Newton preconditioner. $\lambda = 0$ corresponds to unregularized Gauss-Newton descent, while $\lambda = 1$ corresponds to full batch steepest descent. At all values of λ, regularized Gauss-Newton achieves its lowest validation loss in fewer training steps than steepest descent (Fig. 3.7(b)). For some values of λ, the regularized Gauss-Newton method additionally produces lower test loss values than steepest descent (Fig. 3.7(a)).

Writing the preconditioner in terms of the eigenvectors, $\hat{e}_i$, and eigenvalues, μ_i, of B,

$$((1-\lambda)B + \lambda I)^{-1} = \sum_i \frac{1}{(1-\lambda)\mu_i + \lambda} \hat{e}_i \hat{e}_i^\top , \tag{3.40}$$

we see that regularized Gauss-Newton optimization acts similarly to unregularized Gauss-Newton in the subspace spanned by eigenvectors with eigenvalues larger than $\lambda/(1-\lambda)$, and similarly to steepest descent in the subspace spanned by eigenvectors with eigenvalues smaller than $\lambda/(1-\lambda)$. We therefore suggest that regularized Gauss-Newton should be viewed as discarding information in the large-eigenvector subspace, though our theory does not formally address this case. As λ increases from zero to one, the ratio $\lambda/(1-\lambda)$ increases from zero to infinity. Regularized Gauss-Newton method therefore has access to information about the relative magnitudes of more and more of the principal components in the data as λ grows larger. We interpret the improved test performance with regularized Gauss-Newton at about $\lambda = 0.5$ in Fig. 3.7(a) as suggesting that this loss of information within the leading subspace is actually beneficial for the model on this dataset, likely due to aspects of the model's inductive bias which are actively harmful on this task.

Supplementary experiments with linear least squares

In Fig. 3.8 we present the same experiment as in Fig. 3.6(a) at three additional dataset sizes. In all cases, the best test performance achievable by early stopping on whitened data was worse than on unwhitened data.

In Fig. 3.9, we study the effect on generalization of using the entire dataset of 60000 CIFAR-10 images to compute the whitening transform regardless of training set size. We call this type of whitening "distribution whitening" to indicate that we are interested in what happens when the whitening matrix approaches its ensemble limit. We find that distribution whitening does not close the generalization gap with unwhitened models, but it generally does better than train-whitened and fully-whitened models at small and intermediate dataset sizes.

In Fig. 3.10, we compare generalization performance of models trained on whitened versus unwhitened data from two different parameter initializations. Initial distributions with larger variance produce models that generalize worse, but for a fixed initial distribution, models trained on whitened data generally underperform models trained on unwhitened data.

3.4 Discussion

Are whitening and second-order optimization a good idea? Our work suggests that whitening and second-order optimization come with costs – a likely reduction in the best achievable generalization. However, both can drastically decrease training time – an effect we also see in our experiments. As compute is often a limiting factor on performance [121], there are many scenarios where faster training may be worth the reduction in generalization. Additionally, the negative effects may be largely resolved if the whitening transform or second-order preconditioner are regularized, as is often done in practice [55]. We observe benefits from regularized second-order optimization in Fig. 3.7, and similar results have been observed for whitening [77].

Directions for future work. The practice of whitening has, in the machine learning community, largely been replaced by batch normalization, for which it served as inspiration [66]. Studying connections between whitening and batch normalization, and especially understanding the degree to which batch normalization destroys information about the data distribution, may be particularly fruitful. Indeed, some results already exist in this direction [64].

Most second-order optimization algorithms involve regularization, structured approximations to the Hessian, and often non-stationary online approximations to curvature. Understanding the implications of our theory results for practical second-order optimization algorithms should prove to be an extremely fruitful direction for future work. It is our suspicion that milder loss of information about the training inputs will occur for many of

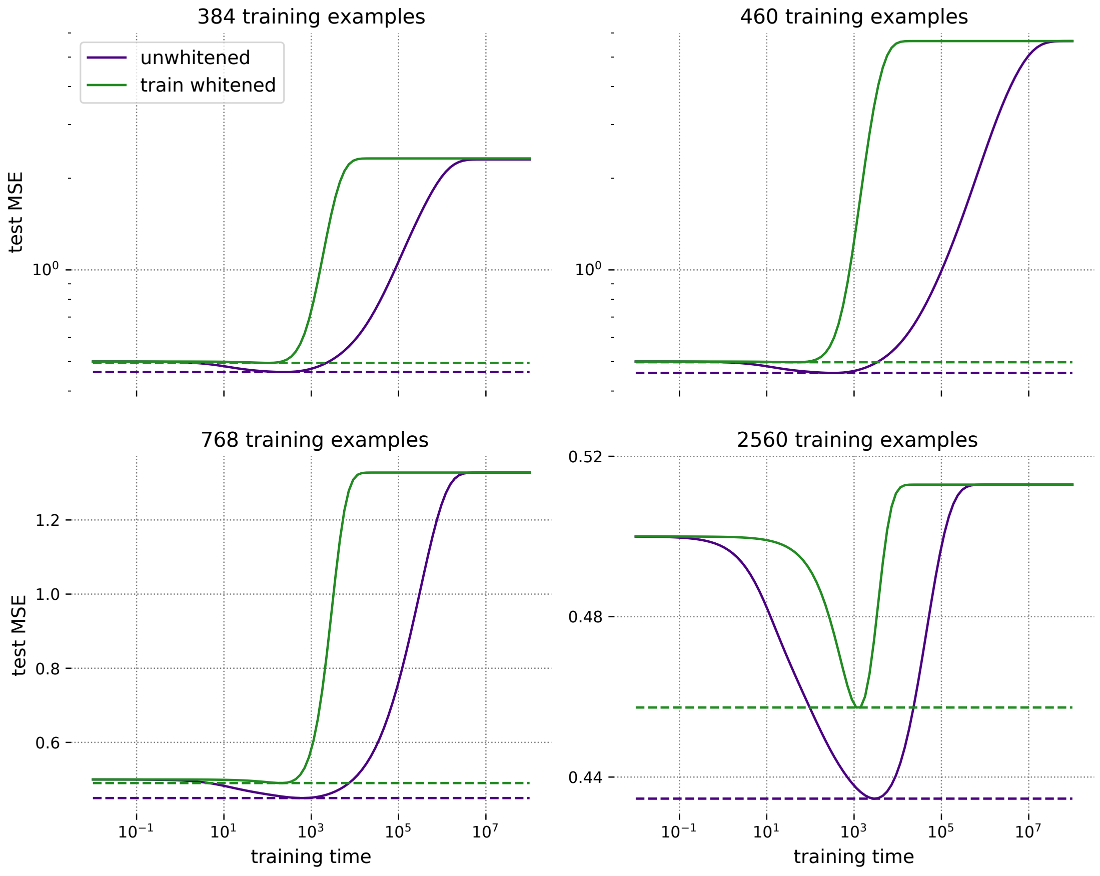


Figure 3.8: **Whitening data speeds up training but reduces generalization in linear models.** Here we show representative examples of the evolution of test error with training time in a linear least-squares model where the training set consists of $384, 460, 768, 2560$ examples, as labeled. In all cases, while models trained on train-whitened data (in green) reach their optimal mean squared errors in a smaller number of epochs, they do no better than models trained on unwhitened data (in purple). In the large time limit of training, the two kinds of models are indistinguishable as measured by test error. The y-axis in the top row of plots is in log scale for clarity. In all cases, the input dimensionality of the data was 512.

these algorithms. In addition, it would be interesting to understand how to relax the large width requirement in our theoretical analysis.

Recent work analyzes deep neural networks through the lens of information theory [17, 133, 7, 20, 124, 2, 72, 9, 3, 110, 116], often computing measures of mutual information similar to those we discuss. Our result that the only usable information in a dataset is contained in

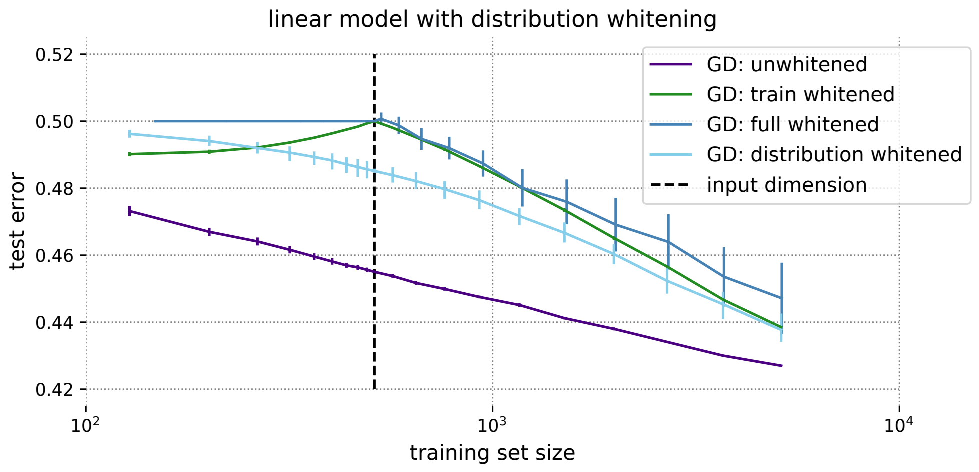


Figure 3.9: **Whitening using the entire dataset behaves similarly to conventional whitening, with only a slight improvement in performance.** Whitening using a whitening transform computed on the entire CIFAR-10 dataset of 50000 training and 10000 test images (distribution whitening) improves performance over train- and full whitening, but does not close the performance gap with unwhitened data. With the exception of the "distribution whitened" line, gradient descent data in this plot is identical to Fig. 3.4(a).

its sample-sample second moment matrix K may inform or constrain this type of analysis.

3.5 Methods

Whitening Methods

PCA Whitening

Consider a dataset $X \in \mathbb{R}^{d \times n}$. PCA whitening can be viewed as a two-step operation involving rotation of X into the PCA basis, followed by the normalization of all PCA components to unity. We implement this transformation as follows. First, we compute the the singular value decomposition of the unnormalized feature-feature second moment matrix $XX^\top$:

$$XX^\top = U\Sigma V^\top, \tag{3.41}$$

where the rank of Σ is $\min(n, d)$. The PCA whitening transform is then computed as $M = \Sigma^{-1/2} \cdot V^\top$, where the dot signifies element-wise multiplication between the whitening coefficients, $\Sigma^{-1/2}$, and their corresponding singular vectors. When Σ is rank deficient $(n < d)$, we use one of two methods to stabilize the computation of the inverse square root:

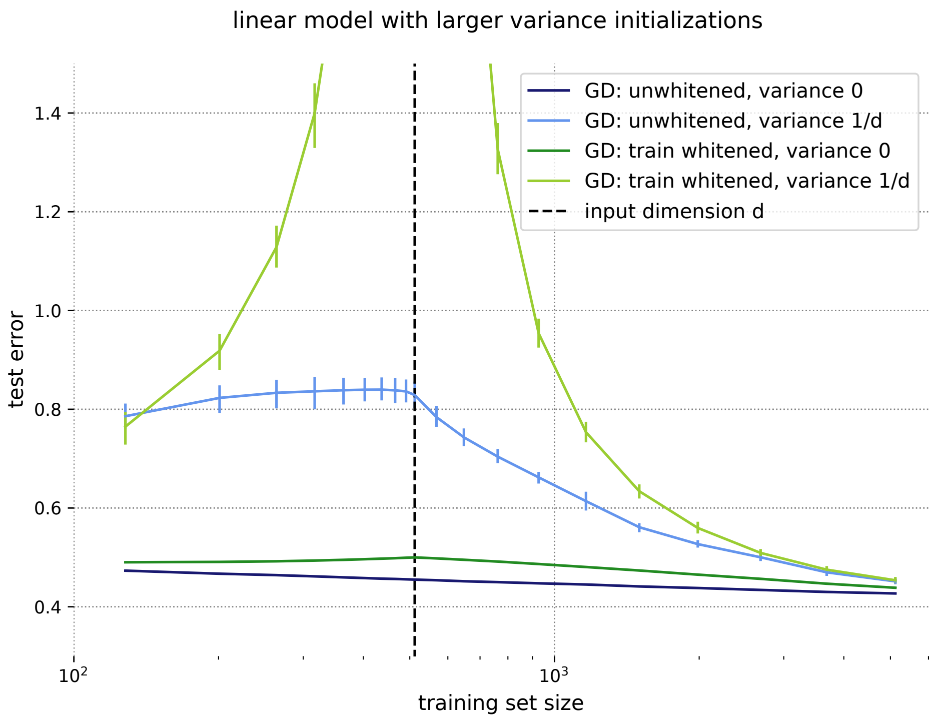


Figure 3.10: **The effect of whitening on linear models with non-zero parameter initialization.** Linear models are initialized with parameter variances of 0 or $1/d$. In all cases the test loss is reported for the time during gradient flow training when the model achieves the lowest validation loss. Unwhitened data was scaled to have the same norm accumulated over all samples in the training set as whitened data, for each training set size, to avoid artifacts due to overall input scale. A model output of zero corresponds to a test loss of 0.5. All configurations with loss greater than 0.5 are doing *worse* than an uninformative prediction of 0. At both initialization scales, the model trained on whitened data performs worse than the model trained on unwhitened data for almost all dataset sizes, while for one dataset size they perform similarly. Data for the variance 0 initialization is identical to Fig. 3.4(a).

the addition of noise, or manual rank control. In the former, a small jitter is added to the diagonal elements of Σ before inverting it. This was implemented in the experiments in linear models. In the latter, the last $d - n$ diagonal elements of $\Sigma^{-1/2}$ are explicitly set to unity. This method was implemented in the MLP experiments.

ZCA Whitening

ZCA (short for zero-phase components analysis) [22] can be thought of as PCA whitening followed by a rotation back into the original basis. The ZCA whitening transform is $M = U\Sigma^{-1/2} \cdot V^{\top}$. ZCA whitening produces images that look like real images, preserving local structure. For this reason, it is used in the CNN experiments.

Linear Model

Dataset composition. The dataset for this experiment was a modified version of CIFAR-10, where the images were first processed by putting them through an off-the-shelf (untrained) four layer convolutional network with tanh nonlinearities and collecting the outputs of the penultimate layer. This resulted in a dataset of 512-dimensional images and their associated labels. Both the CIFAR-10 training and test sets were processed in this way. The linear estimator was trained to predict one-hot (ten dimensional) labels.

Training set sizes ranged from 128 to 5120 examples, randomly sampled from the preprocessed CIFAR-10 data. For experiments on unwhitened and train-whitened data, a validation set of 10000 images was split from the CIFAR-10 training set, and test error was measured on the CIFAR-10 test set. For experiments on fully whitened data, validation and test sets of 10 images each were split from the CIFAR-10 training and test sets, respectively.

Whitening. At each training set size, four copies of the data were made, and three were whitened using the PCA whitening method. For train-whitened data, the whitening transform was computed using only the training examples. For fully whitened data, the twenty validation and test images were included in the computation of the whitening transform. For distribution whitened data (Fig. 3.9), the entire CIFAR-10 dataset of 60000 images (train as well as test) was used to compute the whitening transform.

Training and Measurements. We used a mean squared error loss function. Weights were initialized to all-zeros, except for the data in Fig. 3.10, for which initial weights were drawn from a Gaussian with variance $1/d$. At each training set size, fifty models (initialized with different random seeds) were trained with full-batch gradient descent, with the optimization path defined by the gradient flow equation. Writing the model parameters as ϕ, this equation is

$$\phi(t) = \phi^* + e^{-tCB}(\phi^* - \phi^{(0)})$$

for preconditioner B, feature-feature correlation matrix C, infinite-time solution θ^*, and initial iterate $\theta^{(0)}$. In the case of gradient descent, the preconditioner B is simply the identity matrix.

In order to generate the plot data for Fig. 3.4(a), we solved the gradient flow equation for the parameters ϕ that achieved the lowest validation error, and calculated the test error achieved by those parameters. Mean test errors and their inner 80th percentiles across the twenty different initializations and across whitening states were computed and plotted. To make the plots in Fig. 3.6(a) and 3.8, we tracked test performance over the course of training on unwhitened and train-whitened data.

On train-whitened datasets, we also implemented Newton's Method. This was done by putting the preconditioner B in the gradient flow equation equal to the inverse Hessian, i.e. $\left(XX^\top\right)^{-1}$. The preconditioner was computed once using the whole training set, and remained constant over the course of training. For experiments on whitened data, the data was whitened before computing the preconditioner.

Multilayer Perceptron

On MNIST

Architecture. We used a $784 \times 512 \times 512 \times 10$ fully connected network with a rectified linear nonlinearity in the hidden layers and a softmax function at the output layer. Initial weights were sampled from a normal distribution with variance 10^{-4}.

Dataset composition. The term "dataset size" here refers to the total size of the dataset, i.e. it counts the training as well as test examples. We did not use validation sets in the MLP experiments. Datasets of varying sizes were randomly sampled from the MNIST training and test sets. Dataset sizes were chosen to tile the available range (0 to 70000) evenly in log space. The smallest dataset size was 10 and the two largest were 50118 and 70000. For all but the largest size, the ratio of training to test examples was 8 : 2. The largest dataset size corresponded to the full MNIST dataset, with its training set of 60000 images and test set of 10000 images.

The only data preprocessing step (apart from whitening) that we performed was to normalize all pixel values to lie in the range $[0, 1]$.

Whitening. At each dataset size, three copies of the dataset were made and two were whitened. Of these, one was train-whitened and the other fully whitened. PCA whitening was performed. The same whitening transform was always applied to both the training and test sets.

Training and Measurements. We used sum of squares loss function. Initial weights were drawn from a Gaussian with mean zero and variance 10^{-4}. Training was performed with SGD using a constant learning rate and batch size, though these were both modulated according to dataset size. Between a minimum of 2 and a maximum of 200, batch size was chosen to be a hundredth of the number of training examples. We chose a learning rate of 0.1 if the number of training examples was ≤ 50, 0.001 if the number of training examples was ≥ 10000, and 0.01 otherwise. Models were trained to 0.999 training accuracy, at which point the test accuracy was measured, along with the number of training epochs, and the accuracy on the full MNIST test set of 10000 images. This procedure was repeated twenty times, using twenty different random seeds, for each dataset. Means and standard errors across random seeds were calculated and are plotted in Fig. 3.5.

For example, at the smallest dataset size of 10, the workflow was as follows. Eight training images were drawn from the MNIST training and two as test images were drawn from the MNIST test set. From this dataset, two more datasets were constructed by whitening the images. In one case the whitening transform was computed using only the eight training examples, and in another by using all ten images. Three copies of the MLP were initialized and trained on the eight training examples of each of the three datasets to a training accuracy of 0.999. Once this training accuracy was achieved, the test accuracy of each model on the two test examples, and on the full MNIST test set, was recorded, along with the number of training epochs. This entire procedure was repeated twenty times.

Computation of the input rank of MNIST data. MNIST images are encoded by 784 pixel values. However, the input rank of MNIST is much smaller than this. To estimate the maximal input rank of MNIST, at each dataset size (call it n) we constructed twenty samples of n images from MNIST. For each sample, we computed the 784×784 feature-feature second moment matrix F and its singular value decomposition, and counted the number of singular values larger than some cutoff. The cutoff was 10^{-5} times the largest singular value of F for that sample. We averaged the resulting number, call it r, over the twenty samples. If $r = n$, we increased n and repeated the procedure, until we arrived at the smallest value of n where $r > n$, which was 276. This is what we call the maximal input rank of MNIST, and is indicated by the solid black line in Fig. 3.5.

On CIFAR-10

The procedure for the CIFAR-10 experiments was almost identical to the MNIST experiments described above. The differences are given here.

The classifier was of shape $3072 \times 2000 \times 2000 \times 10$ - slightly larger in the hidden layers and of necessity in the input layer.

The learning rate schedule was as follows: 0.01 if the number of training examples was ≤ 504, then dropped to 0.005 until the number of training examples exceeded 2008, then dropped to 0.001 until the number of training examples exceeded 10071, and 0.0005 thereafter.

The CIFAR-10 dataset is full rank in the sense that the input rank of any subset of the data is equal to the dimensionality, 3072, of the images.

Fig. 3.4(b), 3.5 plot details

In Figs. 3.4(b) and 3.5, for models trained on unwhitened data and train-whitened data, we have plotted test error evaluated on the full CIFAR-10 and MNIST test sets of 10000 images. For models trained on fully whitened data, we have plotted the errors on the test examples that were included in the computation of the whitening transform.

Convolutional Networks

WRN

Architecture. We use the ubiquitous Wide ResNet 28-10 architecture from [149]. This architecture starts with a convolutional embedding layer that applies a 3×3 convolution with 16 channels. This is followed by three "groups", each with four residual blocks. Each residual block features two instances of a batch normalization layer, a convolution, and a

ReLU activation. The three block groups feature convolutions of 160 channels, 320 channels, and 640 channels, respectively. Between each group, a convolution with stride 2 is used to downsample the spatial dimensions. Finally, global-average pooling is applied before a fully connected readout layer.

Dataset composition. We constructed thirteen datasets from subsets of CIFAR-10. The thirteen training sets ranged in size from 10 to 40960, and consisted of between 2^0 and 2^{12} examples per class. In addition, we constructed a validation set of 5000 images taken from the CIFAR-10 training set which we used for early stopping. Finally, we use the standard CIFAR-10 test set to report performance.

Whitening. We performed ZCA whitening using only the training examples to compute the whitening transform.

Training and Measurements. We used a cross entropy loss and the Xavier weight initialization. We performed full-batch gradient descent training with a learning rate of 10^{-3} until the training error was less than 10^{-3}. We use TPUv2 accelerators for these experiments and assign one image class to each chip. Care must be taken to aggregate batch normalization statistics across devices during training. After training, the test accuracy at the training step corresponding to the highest validation accuracy was reported. At each dataset size, this procedure was repeated twice, using two different random seeds. Means and standard errors across seeds were calculated and are plotted in Fig. 3.4(c).

CNN

Architecture. The network consisted of a single ResNet-50 convolutional block followed by a flattening operation and two fully connected layers of sizes 100 and 200, successively. Each dense layer had a *tanh* nonlinearity and was followed by a layer norm operation.

Dataset composition. Training sets were of eleven sizes: 10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, and 10240 examples. A validation set of 10000 images was split from the CIFAR-10 training set.

Training and Measurements. We used a cross entropy loss. Initial weights were drawn from a Gaussian with mean zero and variance 10^{-4}. Training was accomplished once with the Gauss-Newton method (see [26] for details), once with full batch gradient descent, and once with a regularized Gauss-Newton method. With a regularizer $\lambda \in [0, 1]$, the usual preconditioning matrix B of the Gauss-Newton update was modified as $((1-\lambda)B + \lambda I)^{-1}$. This method interpolates between pure Gauss-Newton ($\lambda = 0$) and gradient descent ($\lambda = 1$). In the Gauss-Newton experiments, we used conjugate gradients to solve for update directions; the sum of residuals of the conjugate gradients solution was required to be at most 10^{-5}.

For the gradient descent and unregularized Gauss-Newton experiments, at each training set size, ten CNNs were trained beginning with seven different initial learning rates: $2^{-8}, 2^{-6}, 2^{-4}, 2^{-2}, 1, 4,$ and 16. After the initial learning rate, backtracking line search was used to choose subsequent step sizes. Models were trained until they achieved 100% training accuracy. The model with the initial learning rate that achieved the best validation performance of the seven was then selected. Its test performance on the CIFAR-10 test set was evaluated at the training step corresponding to its best validation performance. The entire procedure was repeated for five random seeds. In Fig. 3.4(d), we have plotted average test and validation losses over the random seeds as functions of dataset size and training algorithm. In Fig. 3.6(c), we have plotted an example of the validation and training performance trajectories over the course of training for a training set of size 10240.

For the regularized Gauss-Newton experiment, the only difference is that we trained one CNN at each initial learning rate per random seed, and then selected the model with the best validation performance. In Fig. 3.7, we have plotted average metrics over the five random seeds. Errorbars and shaded regions indicate twice the standard error in the mean.

Chapter 4

Adaptive step size selection from a dynamical systems perspective

Many tasks in machine learning and statistics can be formulated as optimization problems. First-order optimization algorithms, such as gradient descent, are commonly used due to their simplicity and due to the fact that their complexity scales mildly in the number of decision variables.

The computational complexity of these algorithms is typically measured by how the number of iterations grows as a function of accuracy (in terms of function value or distance to the optimizer) over a given class of functions and over a set of initial conditions. While this notion of complexity often neglects constants, it has been successful at characterizing fundamental performance limits [see, e.g., 93, 35, 91] and deriving optimal algorithms. However, when facing a practical problem instance, the constants do matter and determine the actual number of iterations needed. This motivates the line of research that we present here, which leverages tools from numerical analysis in order to improve performance on practical problem instances.

By exploiting analogies between first-order optimization algorithms and dynamical systems, we analyze the use of adaptive step size methods from numerical analysis in optimization. These routines are efficient at computing solutions to differential equations and are used in most software packages related to ordinary and partial differential equations [57]. Our empirical studies reveal that these techniques can also be applied in optimization and often significantly reduce the number of iterations needed for convergence. Key benefits are: 1) the proposed approach applies to a wide range of algorithms,[1] and 2) the proposed approach does not disturb the computational complexity per iteration; i.e., the complexity remains $\mathcal{O}(d)$, where d refers to the problem dimension. Thus, compared to a fixed step size routine, an adaptive routine has the potential to speed up convergence at essentially no additional computational cost.

[1]We focus on first-order algorithms, but second-order algorithms, such as the Newton method, can be treated in a similar way.

In mathematical optimization, there is a very long tradition of using steps of variable length. The methods typically fall into two categories: line search and trust region strategies. The former aims to choose a step size that sufficiently reduces the objective function at each iteration. The difficulty lies in balancing the amount of decrease with the resulting computational effort [94]. Well-known strategies with convergence guarantees are the Wolfe [146] or the Armijo-Goldstein conditions [12]. In contrast to line search, where the search direction is fixed and only the step size is varied, trust region strategies optimize over both the step size and the step direction. This is typically done by constructing a local approximation of the objective function about the current iterate. Furthermore, the step size is often restricted, ensuring that the approximation remains valid. Excellent texts on this subject include [51] and [94], for example.

In contrast to these techniques, our approach has its origins in the numerical analysis of differential equations. We draw on recent results that view optimization algorithms as continuous-time dynamical systems [see, e.g., 74, 130, 142, 45, 90] and apply adaptive step size techniques as a means to efficiently discretize the continuous-time equations. This provides an efficient alternative to line search and trust region methods. Our discussion is not limited to a single algorithm but provides a blueprint that can be applied to any algorithm that has a meaningful continuous-time representation as an ordinary differential equation (such as Newton-type methods, gradient descent, accelerated gradient descent, etc.).

Related work. A vast literature exists on adaptive step size methods. These are often introduced as approximations to the Newton method and include, for example, the popular Broyden-Fletcher-Goldfarb-Shanno algorithm [see, for example, 38, 39, for more recent accounts and variants] or the Davidon-Fletcher-Powell method [15]. Important work has also been done in the machine learning community with the aim of generalizing line search strategies to a stochastic setting [see, e.g., the recent work of 84, 103], or [135]. Moreover, many popular optimization algorithms include adaptive elements, such as Adam [70], ADAGrad [46], or RMSProp. Understanding the benefits and drawbacks of these enhancements compared to standard stochastic gradient descent is an active area of research [41, 145]. The purpose of the worked presented here is to introduce adaptive step size methods from a perspective rooted in numerical analysis to the toolbox that is available to design and analyze practical optimization algorithms.

Notation. We consider an objective function, $f : \mathbb{R}^d \to \mathbb{R}$, which has a Lipschitz-continuous gradient. In order to simplify our exposition, we further assume f has a single, isolated minimum x^* at the origin with $f(0) = 0$, and that the Lipschitz constant of the gradient is unity. We write the smallest and largest eigenvalues of the Hessian of f at x^* as μ and L, respectively, and introduce the condition number as $\kappa = L/\mu$.

Outline. Section 4.1 introduces an adaptive step size routine and illustrates its use with the gradient method. This is followed by a discussion of how to apply the method to accelerated algorithms, and then a meta-algorithm that applies to any optimizer with an ODE representation. Section 4.2 evaluates the performance of the adaptive step size routine on strongly convex functions. We study non-accelerated methods (gradient descent), as well as accelerated methods (heavy ball). We randomize over different problem instances and vary the condition number. Section 4.3 investigates the performance on a nonconvex problem. In particular, we study principal components analysis (PCA) as our benchmark, due to its practical importance, the fact that the optimal solution can be accurately determined, and because the problem size can be easily varied. We again randomize over different problem instances. We conclude with a short discussion in Section 4.4.

4.1 Adaptive step size routine

Gradient descent

Gradient descent has a well-defined continuous-time representation in the form of gradient flow,

$$\dot{x}(t) = -f'(x(t)). \tag{4.1}$$

We use dots to write time derivatives and primes for derivatives with respect to x. Gradient descent is the Euler discretization of Eq. 4.1. Using x_n and h_n to write the n^{th} discrete-time update and step size, respectively, this is

$$x_{n+1} = x_n - h_n f'(x_n). \tag{4.2}$$

The Euler discretization is a common method of discretizing ordinary differential equations. However, there are many other discretization schemes for Eq. 4.2. These differ, for example, in the accuracy with which they track the underlying continuous dynamics. More accurate discretization schemes afford the ability to pick larger step sizes [57]. The central idea of our work is that we can use this fact to guide adaptation of the step size during optimization. Two ingredients are required to implement this idea: a way to measure the accuracy of the Euler step Eq. 4.2, and a mechanism for adapting the step size accordingly. These two ingredients are described next.

Measuring accuracy. The error of a discretization scheme is given by the difference between the continuous-time and discrete-time trajectories. Starting from $x(t) = x_n$, an application of Taylor's theorem reveals that

$$x(t+h_n) - x_{n+1} = \frac{1}{2} f''(x(\xi)) f'(x(\xi)) h_n^2 = C(x_n, h_n)\, h_n^2, \tag{4.3}$$

for some $\xi \in (t, t+h_n)$, provided that $f'(x)$ is continuously differentiable. The function $C(x_n, h_n)$ is roughly constant for small h_n, which shows that the local error of gradient descent is quadratic in h_n.

We compare this to Heun's method, which is a more accurate discretization of Eq. 4.2. Using the notation x_n^{H} for the n^{th} Heun update, this is

$$x_{n+1}^{\mathrm{H}} = x_n - \frac{1}{2}h_n\left(f'(x_n) + f'(x_{n+1})\right), \tag{4.4}$$

where x_{n+1} is given by Eq. 4.2. Heun's method is also called the "midpoint" rule, because the increment to x_n is the average of (is "midway" between) $-f'$ evaluated at x_n and at x_{n+1}. Again, by applying Taylor's theorem we find

$$x(t + h_n) - x_{n+1}^{\mathrm{H}} = C^{\mathrm{H}}(x_n, h_n)h_n^3, \tag{4.5}$$

where the function $C^{\mathrm{H}}(x_n, h_n)$ is roughly constant for small h_n. The local error of Heun's method is cubic in h_n, and so it is more accurate than Eq. 4.2.

For small h_n, we can therefore estimate the accuracy of gradient descent at each iteration by comparing it to the Heun update:

$$||x(t + h_n) - x_{n+1}|| = \delta_n(x_n, h_n) + \mathcal{O}(h_n^3), \tag{4.6}$$

where $\delta_n(x_n, h_n) = ||x_{n+1}^{\mathrm{H}} - x_{n+1}||$. We use the notation $||\cdot||$ for the Euclidean norm. We emphasize that using the Heun update to estimate the accuracy of gradient descent is a particular choice. We could well choose any other more accurate discretization of Eq. 4.1 than Eq. 4.2.

The error analysis just presented is valid, strictly speaking, only for small h_n. Nevertheless, we observe in our experiments that the resulting adaptive step size routine is still effective at step sizes of order one.

A note on complexity. The Heun discretization has the advantage of being computationally efficient. Each evaluation of f' is a single oracle query for a gradient. The gradient evaluation of $f'(x_{n+1})$ in Eq. 4.4 can be reused in Eq. 4.2 for the next iteration. This means that δ_n can be computed without additional evaluation of the gradient. The step size control mechanism we propose in the following therefore comes with almost no additional computational cost; in fact, it requires precisely *one* additional gradient computation in total, over all the iterations.

Adapting step size: proportional control. We use a proportional controller ("P control") to set step sizes based on the value of δ_n. P control is the simplest version of a family of control mechanisms called PID (proportional integral derivative) control. These find use as automated mechanisms for controlling a user-specified quantity in a dynamical system. In our case, the quantity we wish to control is δ_n. The idea of using PID control to set step sizes for a discretization scheme for differential equations is not new [57, 58]. Our innovation here is to apply this idea to optimization.

The P control mechanism introduces two hyperparameters: r, the desired error between the Heun and gradient descent updates on any given iteration, and θ, which can be interpreted as a gain. The control mechanism gives the following prescription for modulating the step size:

$$h_{n+1} = \left(\frac{r}{\delta_n}\right)^{\theta/2} h_n. \tag{4.7}$$

The form of Eq. 4.7 is motivated by the fact that, after taking logarithms, neglecting $\mathcal{O}(h_n^3)$ terms, and using Eq. 4.6 and Eq. 4.3, we have

$$\log h_{n+1} = \log h_n + \frac{\theta}{2}\left(\log r - \log \delta_n\right) = (1-\theta)\log h_n + \frac{\theta}{2}\left(\log r - \log C\right).$$

This means that $\log h_n$ evolves roughly as a linear system, and is expected to converge to its steady state, given by $\delta_n = r$, with a rate of $1-\theta$. The feedback law Eq. 4.7 is called "proportional" control because the quantity $\log h_{n+1} - \log h_n$ is proportional to the error signal $\log r - \log \delta_n$.

At each iteration during optimization, we compute the gradient descent and Heun updates, compute δ_n, and calculate the step size h_{n+1} for the *next* iteration using Eq. 4.7. We then apply the gradient descent update with the step size h_n recommended by the controller on the previous iteration.

We emphasize that the Heun update Eq. 4.4 is never actually applied to an iterate during optimization. Its sole purpose lies in the computation of δ_n.

Setting θ and r. Formally, we have $\theta \in [0, 2]$ [58]. Note that $\theta = 0$ results in a constant step size routine. Increasing θ turns up the gain on the step size control, allowing larger changes between iterations, and potentially lowering the number of iterations required to converge.

We use small values of θ, i.e., in the set $\{10^{-4}, 10^{-3}, 10^{-2}\}$. These values of θ work for functions with a large range of condition numbers. At larger values of θ we sometimes observe oscillatory behavior in the optimizer. Values of θ smaller than 10^{-4} lead to slow adaptation of the step size, which forfeits the benefit of the control mechanism.

The value of r specifies how closely we want the Euler discretization to approximate the underlying gradient flow trajectories. We set $r = 0.5$ in experiments on strongly convex functions, forcing the P controller to set step sizes such that δ_n is no greater than 0.5. We find the controller is less sensitive to changes in r. Smaller values also work well. In experiments with PCA, we use $r = 10^{-4}$.

In practice we find it necessary to place a few additional constraints on the P control mechanism Eq. 4.7. If the factor $(r/\delta_n)^{\theta/2}$ is ≥ 10 or ≤ 0.1, we set it equal to 10 or 0.1, respectively. If, furthermore, $h_n \geq 2$ or ≤ 0.01, we set it equal to 2 or 0.01, respectively. The upper bound of 2 on the step size is motivated by the fact that this is the stability boundary for gradient descent on a 1-Lipschitz convex function. It is possible to remove these constraints by fine-tuning r and θ to specific problem instances, i.e., to specific ranges of condition numbers, but too much tuning defeats the purpose of the adaptive algorithm, so we feel it is reasonable to introduce the constraints. With these constraints, we are able to

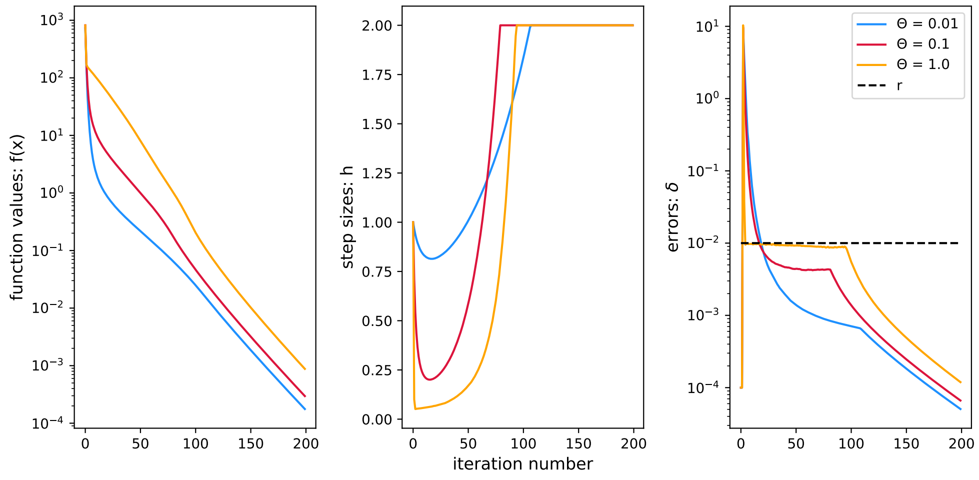


Figure 4.1: **Gradient descent with adaptive P control.** Example trajectories of the function value (left panel), the step size (middle panel), and the error δ_n (right panel) for gradient descent with step sizes set using proportional control are shown for three different values of θ. Here, $f : \mathbb{R}^{500} \to \mathbb{R}$ is a randomly generated strongly convex function with a condition number of 100. See Section 4.2 for a description of how the function f is generated. Each value of θ produces a different trajectory. After a few tens of iterations, the control mechanism is able to force δ_n below the specified value of r. Large values of θ produce the largest differences between successive step sizes and vice versa.

fix r and θ and optimize strongly convex functions with a broad range of condition numbers (1.1 to about 1100).

In Fig. 4.1 we present example trajectories of gradient descent with adaptive P control on a strongly convex function at a range of values of θ. The condition number in this example is 100. Each value of θ produces an algorithm that converges, but takes a different trajectory than the others. In all cases we see that the function value and the error δ_n decrease rapidly at about the same time that the step size becomes large. This is typical behavior.

Second and higher-order algorithms: the heavy ball method

We construct adaptive step size mechanisms similar to Eq. 4.7 for optimizers with higher order continuous-time representations using a simple trick. If we rewrite the higher order differential equation as a system of (coupled) first-order equations, the method of the previous section goes through almost without modification. We illustrate this here with the example of the heavy ball method.

We work with the following continuous-time limit of the heavy ball algorithm:

$$\dot{x}(t) = p(t) \tag{4.8a}$$

$$\dot{p}(t) = -\frac{2}{\sqrt{\kappa}} p(t) - f'(x(t)). \tag{4.8b}$$

Here we have introduced the (momentum) variable $p = \dot{x}$ to write the dynamics as a set of coupled first-order differential equations in time.

Symplectic discretization. We have seen that discretization schemes differ in the accuracy with which they track the underlying continuous system. They also differ in the subsets of properties of the original continuous-time system they preserve. Due to the analogy between Eq. 4.8 and the damped motion of a massive particle through a fluid, it is advantageous to use the *semi-implicit* instead of the *standard* Euler discretization. The advantage is particularly apparent for large values of κ. In the limit as $\kappa \to \infty$, the standard Euler discretization becomes unstable even for arbitrarily small step sizes, whereas the semi-implicit Euler discretization remains stable up to a large step size [92]. The semi-implicit Euler discretization of Eq. 4.8 is

$$x_{n+1} = x_n + h_n p_{n+1} \tag{4.9a}$$

$$p_{n+1} = p_n + h_n \left(-\frac{2}{\sqrt{\kappa}} p_n - f'(x_n) \right). \tag{4.9b}$$

Compared with a vanilla Euler update, the semi-implicit Euler method replaces p_n with p_{n+1} in the update for x_n.

In order to calculate an error δ_n at each step, we require a more accurate discretization of Eq. 4.8 than Eq. 4.9. In analogy with the previous section, we use the following Heun discretization:

$$x^{\mathrm{H}}_{n+1} = x_n + \frac{1}{2} h_n \left(p_{n+1} + p_{n+2} \right) \tag{4.10a}$$

$$p^{\mathrm{H}}_{n+1} = p_n + \frac{1}{2} h_n \bigg(-\frac{2}{\sqrt{\kappa}} (p_n + p_{n+1}) - (f'(x_n) + f'(x_{n+1})) \bigg), \tag{4.10b}$$

where x_{n+1} and p_{n+1} are given by Eq. 4.9 and p_{n+2} is the semi-implicit Euler update to p_{n+1}. We calculate $\delta_n = ||(x_{n+1}, p_{n+1}) - (x^{\mathrm{H}}_{n+1}, p^{\mathrm{H}}_{n+1})||$ where (x_n, p_n) is the vector formed by concatenating x_n and p_n. Now step sizes are simply given by the P control mechanism Eq. 4.7.

As for gradient descent, we place some extra constraints on the P controller. If the factor $(r/\delta_n)^{\theta/2}$ is ≥ 5 or ≤ 0.05, we set it equal to 5 or 0.05, respectively. And if $h_n \geq 0.8$ or ≤ 0.01, we set it equal to 0.8 or 0.01, respectively. The upper bound of 0.8 on the step size is motivated by the fact that this is roughly the stability boundary for the heavy ball algorithm on quadratic functions. Also similar to gradient descent, we set $r = 0.5$, and use θ in the set $\{10^{-4}, 10^{-3}, 10^{-2}\}$.

Meta algorithm

So far we have discussed how to construct an adaptive mechanism for step size control for gradient descent and the heavy ball algorithm. Here we generalize the ideas to any algorithm that has a well-defined continuous-time representation.

Given a discrete-time algorithm of interest (call it algorithm A), the construction of a P-control-like mechanism for step size selection requires the following steps:

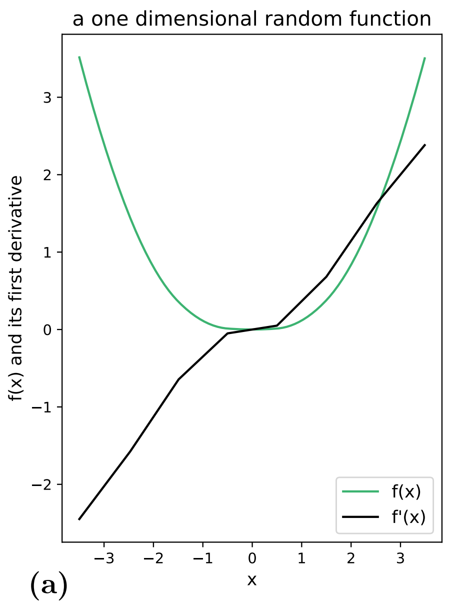
1. Find the continuous-time counterpart of algorithm A. If this is a second or higher order differential equation with respect to time, rewrite it as a system of coupled first-order equations.
2. Identify a higher order discretization (call this algorithm B) of the continuous-time system. Typically, computational efficiency will guide this choice.
3. At each iteration during optimization, compute the difference between the updates of algorithms A and B. Feed this difference to a proportional controller with appropriately set hyperparameters.
4. Apply the update of algorithm A with the step size recommended by the controller.

4.2 Benchmark: Strongly convex functions

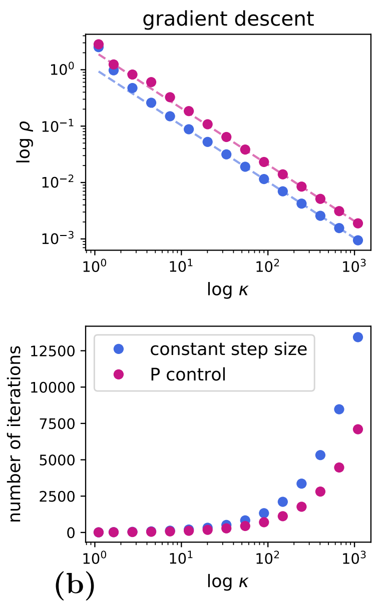
In this section we demonstrate the use of a P controller for adaptive step size selection when optimizing random strongly convex functions with gradient descent and the heavy ball method. We compare the performance of the P controller to a constant step size scheme and show that the former is able to achieve the relevant lower bounds and can also improve on the constant in many cases. We find that the P controller performs particularly well on ill-conditioned problems.

Random generation of strongly convex functions

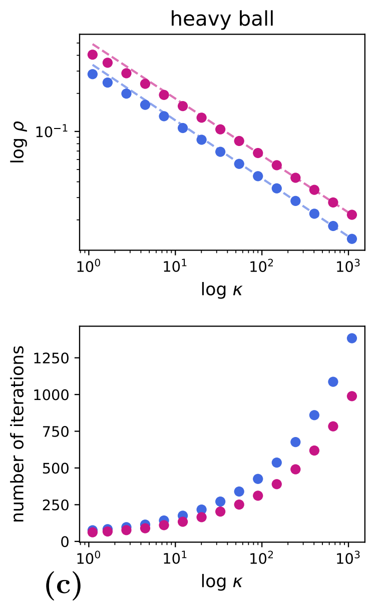
We divide the real line into a finite number N of discrete intervals. For a given value of the condition number κ, we sample a set of N numbers $\kappa^{-1} + z(1 - \kappa^{-1})$, where $z \sim \text{Unif}[0, 1)$, that serve as the piecewise-constant second derivatives of a strongly convex function. The function is then constructed numerically from the second derivative and the conditions $f(0) = 0$, $f'(0) = 0$. Multidimensional functions $f : \mathbb{R}^d \to \mathbb{R}$ are obtained by summing d one-dimensional functions. We ensure that at $x^* = (0, ..., 0)$, the smallest eigenvalue of the Hessian of f is $1/\kappa$, and the largest eigenvalue is 1 (recall that we fix $L = 1$). An example of such a function is given in Fig. 4.2(a) with $N = 7$ and $\kappa = 10$. We fix $N = 7$ in all our experiments.



(a)



(b)



(c)

Figure 4.2: **(a)** An example of a strongly convex function f generated from a randomly sampled set of second derivatives. Note that $f(0) = 0$ and $f'(0) = 0$. In this example, the condition number is 10. **(b)** (Top panel) Gradient descent with P control produces the same convergence behavior as gradient descent with a constant step size of 1. The asymptotic performance of the latter matches its theoretical lower bound. The dashed lines are a linear fit of $\log \rho$ against $\log \kappa$ for large values of κ. The slopes of these fits are the same, but P control has an improved constant. Data shown here is for $(r, \theta) = (0.5, 0.01)$. (Bottom panel) For large condition numbers κ, P control converges in a smaller number of iterations, up to a factor of 2 less than the constant step size routine. **(c)** (Top panel) The heavy ball method with P control performs similarly to the same algorithm with a constant step size of 0.5. Data shown here is for $(r, \theta) = (0.5, 0.01)$. (Bottom panel) On average, P control converges in a smaller number of iterations for almost all values of κ. We emphasize that in both (b) and (c), r and θ are set to 0.5 and 0.01, respectively. For each value of κ, we plot means over fifty repeats of each algorithm. Standard deviations are also plotted, but are too small to be visible.

Gradient descent

We choose a range of condition numbers evenly tiling the log scale between 1.1 and 1100. For each value of κ, we generate fifty strongly convex functions $f : \mathbb{R}^{500} \to \mathbb{R}$. Each function is optimized once using gradient descent with P control to set step sizes, and once with a constant step size of 1, both times beginning with a different random initialization $x^0 \sim 5 * \text{Unif}[0, 1)^{500}$. The starting step size for the adaptive method is also 1 for even comparison. Optimization halts when the norm of the difference between successive updates drops below 10^{-8}.

We characterize the convergence of gradient descent using

$$||x_n|| \sim B||x_0||e^{\rho n}, \tag{4.11}$$

where B is a positive constant and ρ is the convergence rate, which depends on the condition number κ. To probe this dependency, for each trajectory, we perform a linear fit of the log norm of the iterate (scaled by the norm of the initial iterate) to the iteration number:

$$\log \frac{||x_n||}{||x_0||} = \rho n + \tilde{B}, \tag{4.12}$$

where $\tilde{B}$ is a constant. The slope ρ produced by this fit is recorded, along with the number of iterations required to reach the stopping criterion. This data is graphed in Fig. 4.2(b) (top panel) as a function of condition number for $(r, \theta) = (0.5, 0.01)$. We see that $\log \rho$ exhibits roughly linear behavior with $\log \kappa$. It is therefore natural to estimate the asymptotic dependence of ρ on κ for each algorithm with a linear fit to the data in Fig. (4.2)(b) (top panel) for large κ.

For gradient descent, the complexity lower bound for ρ is $1/\kappa$, i.e., $\rho \sim \mathcal{O}(1/\kappa)$ [93]. A constant step size of $1/L$ (1 here, as $L = 1$) achieves this bound. Therefore we expect the slope of the linear fit to the constant step size data in Fig. 4.2(b) (top panel) to be roughly -1.

Results. The P controller achieves the same asymptotic convergence behavior as a constant step size routine with step size 1, and for appropriate values of θ, produces a better (larger) constant. In Fig. 4.2(b) (top panel), at large values of κ, the slope and constant of a linear fit of $\log \rho$ against $\log \kappa$ for the constant step size method are $-0.999(1)$ and $0.013(3)$, respectively. (We report standard deviations in brackets; for example, writing 0.013(3) indicates a parameter of 0.013 with a standard deviation of 0.003.) For gradient descent with P control, with $\theta = 0.01$, the corresponding fit parameters are $-1.000(1)$ and $0.317(3)$. The slopes of the two fits are similar. The larger constant produced by the P control algorithm indicates convergence with a smaller number of iterations. This is demonstrated visually in Fig. 4.2(b) (bottom panel), where for moderate to large values of κ (order 10 onward), P control requires fewer iterations to converge than the constant step size routine. For κ of the order of 10^2, this difference can be a factor of 2. At small values of κ, P control confers little advantage in convergence rate over the constant step size method.

Heavy ball method

We repeat the experiments and analysis of the previous section with the heavy ball algorithm in place of gradient descent. As before, we compare the performance of a constant step size routine to adaptive P control on random strongly convex functions with a range of condition numbers.

For the constant step size routine, we use a step size of 0.5. This is motivated by the fact that the algorithm achieves a rate $\rho \sim 1/(2\sqrt{\kappa})$ on quadratic functions, and step sizes larger than $2(\sqrt{2}-1) \approx 0.828$ lead to instability on quadratics. All other experimental details are similar to those in Section 4.2. The momentum p is initialized to the zero vector.

Results. In Fig. 4.2(c) (top panel) we present convergence rates ρ against condition numbers κ for the heavy ball method with constant step size and with P control. Once again we observe a linear relationship between these variables, and perform a linear fit at large values of κ to estimate the asymptotic behavior of ρ with κ. The slope and constant of the linear fit to the constant step size algorithm are $-0.459(3)$ and $-0.453(8)$, respectively. The corresponding numbers for P control with $\theta = 0.01$ are $-0.450(2)$ and $-0.290(6)$.

Continuous-time analysis of the heavy ball equation Eq. 4.8 establishes a complexity lower bound of $1/\sqrt{\kappa}$ for ρ [91]. A linear fit of $\log \rho$ against $\log \kappa$ for an algorithm that achieves this lower bound should have a slope of $-1/2$. The same holds true in discrete time for large κ [93]. Both the constant and adaptive step size methods perform close to this lower bound, i.e., they perform reasonably well, with asymptotic convergence rates of about -0.45.

Once again, as with gradient descent, P control has a better constant, which manifests as a smaller number of iterations to convergence. This is seen in Fig. 4.2(c) (bottom panel). For large κ the performance gains increase with κ. For $\kappa = 1000$ the adaptive step size method reduces the number of iterations roughly by 30%.

4.3 A nonconvex problem: principal components analysis

We demonstrate the use of P control to set step sizes for a simple instance of the PCA problem: given a positive semidefinite symmetric matrix $A \in \mathbb{R}^{d \times d}$ with distinct eigenvalues, the goal is to learn its first principal component (eigenvector) . The corresponding optimization problem

$$\arg\min_{||x||=1} -x^\top A x, \tag{4.13}$$

where $x \in \mathbb{R}^d$, is nonconvex but still admits a unique computable solution.

A well-established algorithm that solves Eq. 4.13 is Oja's rule [95], given by the update

$$x_{n+1} = \frac{(I + h_n A)x_n}{||(I + h_n A)x_n||}, \tag{4.14}$$

where I is the identity matrix in d dimensions and h_n is the n^{th} step size.

Let the eigengap of A, $\mathrm{gap}(A)$, be defined as the difference between the largest and second-largest eigenvalues of A. Define a positive constant c. A recent convergence proof guarantees that the error of Eq. 4.14, measured by the quantity $1 - (x_n^\top x^*)^2$ where x^* is the principal eigenvector of A, is $\mathcal{O}(1/n)$ for a step size routine of c/n if $c \geq 1/(2 * \mathrm{gap}(A))$ [16]. However, in most real-world problems, $\mathrm{gap}(A)$ would not normally be known ahead of time, complicating the choice of c. Indeed, in [16], this is posed as a problem for practitioners. We are able to offer a solution in the form of P control, which avoids the problem entirely because it requires no knowledge of the gap. We compare Oja's rule with a c/n step size scheme with different values of c to P control (with fixed r and θ) across a range of eigengaps and find that the latter performs competitively.

We note that the convergence result of [16] is proved in the streaming setting, where instead of a fixed matrix A we have a collection of vectors y_i, sampled i.i.d. from a mean-zero distribution with covariance A, that we access one at a time. In such a scenario, A is replaced by the rank one matrices $y_n y_n^\top$ in Eq. 4.14. Since our goal here is simply to evaluate the adaptive step size routine on a nonconvex optimization problem, we confine ourselves to the simplest possible, and hence deterministic, setting even though P control could also be applied in the streaming setting.

P control for Oja's rule. The continuous-time system corresponding to Oja's rule has a particularly simple form. It is gradient flow on the unit sphere in $d-1$ dimensions, which we can write as the differential equation

$$\dot{x}(t) = \left(A - \frac{x(t)^\top A x(t)}{x(t)^\top x(t)} I \right) x(t). \tag{4.15}$$

We note that the original convergence proof of Eq. 4.14 was done in continuous time by analyzing the behavior of Eq. 4.15 [95].

It is not difficult to see that the local error of Eq. 4.14 is quadratic in h_n (see the discussion around Eq. 4.3). In order to compute δ_n for Eq. 4.7, we need a more accurate discretization of Eq. 4.15 than Eq. 4.14. We use a Heun update $x_{n+1}^{\mathrm{H}} = x_n + 0.5h_n(g(x_n) + g(x_{n+1}))$, where g is given by the right-hand side of Eq. 4.15 and x_{n+1} is given by Eq. 4.14. Then $\delta_n = ||x_{n+1} - x_{n+1}^{\mathrm{H}}||$, as before. To calculate the step size for Oja's rule, we use this value of δ_n in the P controller Eq. 4.7 at each iteration.

As before, the Heun method allows us to reuse the computation of x_{n+1} at the next iteration. Hence the adaptive step size routine does not alter the computational cost per iteration.

We interrogate a range of eigengaps between 10^{-5} and 10^{-1}. At each value of the eigengap we generate twenty diagonal matrices $A \in \mathbb{R}^{100\times 100}$, and apply random orthogonal transformations to introduce off-diagonal elements. On each matrix, we run Oja's rule Eq. 4.14 four times: once with P control, and thrice with a c/n step size routine with $c = 0.5/\mathrm{gap}(A), 5/\mathrm{gap}(A)$, and $50/\mathrm{gap}(A)$. We parametrize the P controller with $(r, \theta) = (10^{-4}, 10^{-4})$. Iterates are initialized from the standard uniform distribution. The stop criterion is either $||x_{n+1} - x_n|| \leq 10^{-8}$ or $n = 2.5 \times 10^5$, whichever occurs first.

Results. The P control mechanism produces an algorithm that is competitive with a c/n step size routine for moderate and large values of c, and that outperforms the smallest allowed value of c, as measured by the number of iterations required to converge (see Fig. 4.3). As mentioned previously, the utility of this result derives from the fact that the eigengap is not normally known ahead of time, making it difficult in practice to set a good value of c for the c/n step size routine such that the algorithm achieves its $\mathcal{O}(1/n)$ rate of convergence.

Thus we see that P control can readily be adapted to problems that are not strongly convex, and can produce algorithms that perform well compared to the state of the art.

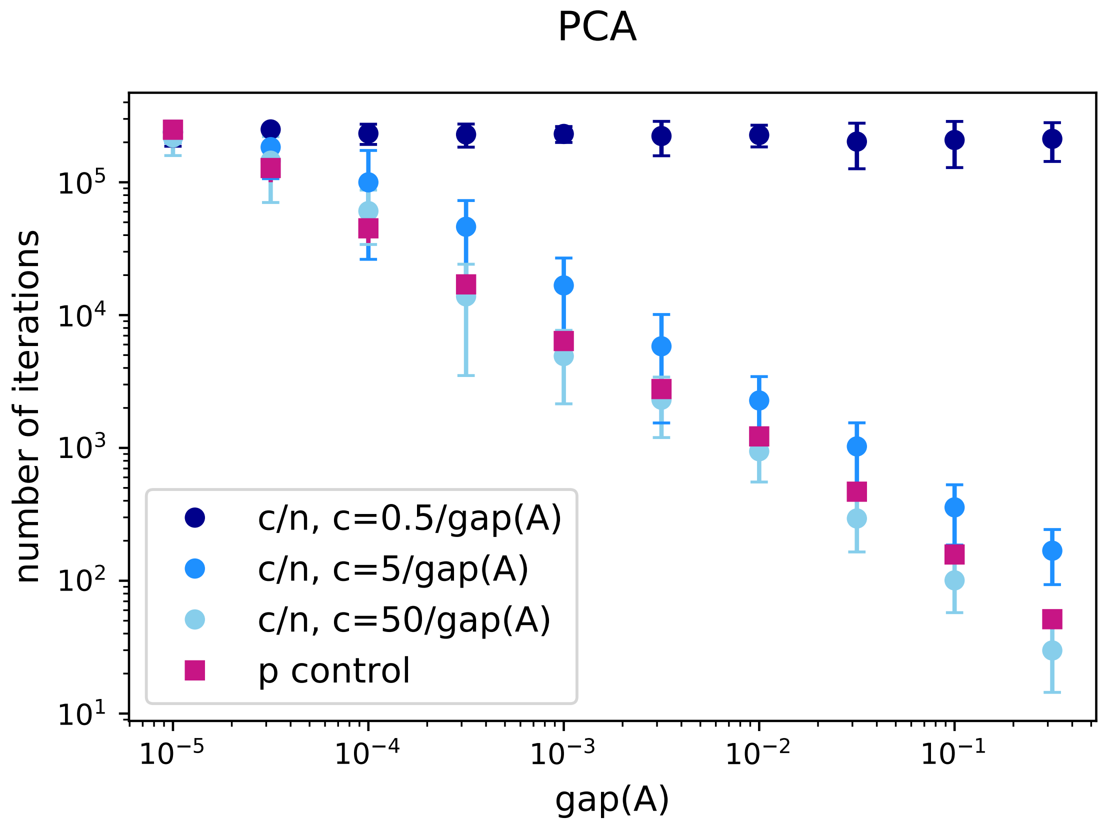


Figure 4.3: **Oja's rule with P control performs competitively with a $1/n$ step size routine.** Setting step sizes in Oja's rule using P control with r and θ both set to 10^{-4} produces an algorithm that converges, on average, in about as many iterations as a $1/n$ step size routine with a moderate to large premultiplier c. On the y-axis we have the mean number of iterations to convergence over twenty repeats, and on the x-axis we have gap(A), the difference between the two largest eigenvalues of the matrix A. Error bars indicate one standard deviation. The error bars for P control are too small for visibility. See text for experimental details.

4.4 Discussion

We have shown how to construct an adaptive step size control mechanism for optimization rooted in ideas from the numerical analysis of differential equations, and we have discussed the specific examples of gradient descent and the heavy ball method. Our method has the advantage of being computationally cheap and easy to tune. Experiments on strongly convex functions indicate that for ill-conditioned problems our approach is able to reduce the number of iterations compared to the state of the art, and is competitive across all problem instances. Experiments on principal components analysis further highlight the potential of the approach even for nonconvex optimization problems.

The adaptive step size method applies to any algorithm with a well-defined continuous-time representation and is flexible in that it easily admits replacements of all its building blocks. For example, in any of the problems we discussed, we could replace the Heun method with a different discretization scheme or use a different controller. There are a wide range of possibilities for further theoretical and practical research.

Bibliography

[1] Steven Abney. *Semisupervised learning for computational linguistics*. Chapman and Hall/CRC, 2007.

[2] A. Achille and S. Soatto. "Emergence of Invariance and Disentangling in Deep Representations". In: *Proceedings of the ICML Workshop on Principled Approaches to Deep Learning* (2017).

[3] Alessandro Achille and Stefano Soatto. "Where is the Information in a Deep Neural Network?" In: *arXiv preprint arXiv:1905.12213* (2019).

[4] Naman Agarwal, Brian Bullins, and Elad Hazan. "Second-Order Stochastic Optimization for Machine Learning in Linear Time". In: *arXiv preprint arXiv:1602.03943* (2016).

[5] Naman Agarwal et al. "Efficient Full-Matrix Adaptive Regularization". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, June 2019, pp. 102–110.

[6] Kyle Aitken and Guy Gur-Ari. "On the asymptotics of wide networks with polynomial activations". To appear.

[7] Alexander A Alemi et al. "Deep Variational Information Bottleneck". In: *arXiv:1612.00410* (2016).

[8] Shun-ichi Amari et al. *When Does Preconditioning Help or Hurt Generalization?* 2020. arXiv: `2006.10732 [stat.ML]`.

[9] Rana Ali Amjad and Bernhard C Geiger. "How (not) to train your neural network using the information bottleneck principle". In: *arXiv preprint arXiv:1802.09766* (2018).

[10] Anders Andreassen and Ethan Dyer. "Asymptotics of Wide Convolutional Neural Networks". To appear.

[11] Rohan Anil et al. "Memory-Efficient Adaptive Optimization for Large-Scale Learning". In: *arXiv preprint arXiv:1901.11150* (2019).

[12] Larry Armijo. "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives". In: *Pacific Journal of Mathematics* 16.1 (1966), pp. 1–3.

[13] Joseph J. Atick and A. Norman Redlich. “What Does the Retina Know About Natural Scenes?” In: *Neural Comput.* 4.2 (Mar. 1992), pp. 196–210.

[14] Fred Attneave. “Some informational aspects of visual perception”. In: *Psychol. Rev* (1954), pp. 183–193.

[15] Saman Babaie-Kafaki. “On Optimality of the Parameters of Self-Scaling Memoryless Quasi-Newton Updating Formulae”. In: *Journal of Optimization Theory and Applications* 167 (2015), pp. 91–101.

[16] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. “The Fast Convergence of Incremental PCA”. In: *Advances in Neural Information Processing Systems*. Ed. by C. J. C. Burges et al. Vol. 26. Curran Associates, Inc., 2013, pp. 3174–3182.

[17] Arindam Banerjee. “On bayesian bounds”. In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 81–88.

[18] Horace Barlow. “Possible Principles Underlying the Transformations of Sensory Messages”. In: *Sensory Communication* 1 (Jan. 1961).

[19] Peter L. Bartlett et al. “Benign overfitting in linear regression”. In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30063–30070.

[20] Raef Bassily et al. “Learners that Use Little Information”. In: *arXiv preprint arXiv:1710.05233* (2017).

[21] Mikhail Belkin, Daniel Hsu, and Ji Xu. “Two Models of Double Descent for Weak Features”. In: *SIAM Journal on Mathematics of Data Science* 2.4 (2020), pp. 1167–1180.

[22] Anthony J. Bell and Terrence J. Sejnowski. “The “independent components” of natural scenes are edge filters”. In: *Vision Research* 37.23 (1997), pp. 3327–3338.

[23] Albert S. Berahas, Majid Jahani, and Martin Takáč. “Quasi-Newton Methods for Deep Learning: Forget the Past, Just Sample”. In: *arXiv preprint arXiv:1901.09997* (2019).

[24] Valentin Blickle and Clemens Bechinger. “Realization of a micrometre-sized stochastic heat engine”. In: *Nat Phys* 8.2 (Feb. 2012), pp. 143–146.

[25] Raghu Bollapragada et al. “A Progressive Batching L-BFGS Method for Machine Learning”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, July 2018, pp. 620–629.

[26] Aleksandar Botev, Hippolyt Ritter, and David Barber. “Practical Gauss-Newton Optimisation for Deep Learning”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 557–565.

[27] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. “Optimization Methods for Large-Scale Machine Learning”. In: *SIAM Review* 60.2 (2018), pp. 223–311.

[28] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* USA: Cambridge University Press, 2004. ISBN: 0521833787.

[29] Rasmus Bro and Age K Smilde. “Principal component analysis”. In: *Analytical Methods* 6.9 (2014), pp. 2812–2831.

[30] Dorje Brody and Nicolas Rivier. “Geometrical aspects of statistical mechanics”. In: *Phys. Rev. E* 51 (2 Feb. 1995), pp. 1006–1011.

[31] CG Broyden. “The convergence of a class of double-rank minimization algorithms 2. The new algorithm”. In: *IMA Journal of Applied Mathematics* (1970).

[32] Jacob Burbea and C Radhakrishna Rao. “Entropy differential metric, distance and divergence measures in probability spaces: A unified approach”. In: *Journal of Multivariate Analysis* 12.4 (1982), pp. 575–596.

[33] RH Byrd et al. “A Stochastic Quasi-Newton Method for Large-Scale Optimization”. In: *arXiv preprint arXiv:1401.7020* (2014).

[34] Richard H Byrd et al. “On the use of stochastic hessian information in optimization methods for machine learning”. In: *SIAM Journal on Optimization* 21.3 (2011), pp. 977–995.

[35] Yair Carmon et al. “Lower bounds for finding stationary points II: first-order methods”. In: *Mathematical Programming* (2019). Preprint available online.

[36] Sadi Carnot. *Refléxions sur la puissance motrice du feu et sur les machines propres à développer cette puissance.* Paris: Chez Bachelier, Libraire, 1824.

[37] Gavin E. Crooks. “Measuring Thermodynamic Length”. In: *Phys. Rev. Lett.* 99 (10 Sept. 2007), p. 100602.

[38] Frank E. Curtis and Xiaocun Que. “A quasi-Newton algorithm for nonconvex, nonsmooth optimization with global convergence guarantees”. In: *Mathematical Programming Computation* 7 (2015), pp. 399–428.

[39] Frank E. Curtis, Daniel P. Robinson, and Baoyu Zhou. “A self-correcting variable-metric algorithm framework for nonsmooth optimization”. In: *IMA Journal of Numerical Analysis* 40 (2020), pp. 1154–1187.

[40] Yang Dan, Joseph J. Atick, and R. Clay Reid. “Efficient Coding of Natural Scenes in the Lateral Geniculate Nucleus: Experimental Test of a Computational Theory”. In: *Journal of Neuroscience* 16.10 (1996), pp. 3351–3362.

[41] Yann N. Dauphin et al. “Equilibrated adaptive learning rates for non-convex optimization”. In: *Advances in Neural Information Processing Systems 28* (2015), pp. 1–9.

[42] Ofir David, Shay Moran, and Amir Yehudayoff. “On Statistical Learning via the Lens of Compression”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems.* NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 2792–2800.

[43] John E Dennis Jr and Jorge J Moré. “Quasi-Newton methods, motivation and theory”. In: *SIAM review* 19.1 (1977), pp. 46–89.

[44] Guillaume Desjardins et al. “Natural Neural Networks”. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 2071–2079.

[45] Jelena Diakonikolas and Michael I. Jordan. “Generalized Momentum-Based Methods: A Hamiltonian Perspective”. In: *arXiv:1906.00436 [math.OC]* (2019), pp. 1–30.

[46] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159.

[47] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[48] Ethan Dyer and Guy Gur-Ari. “Asymptotics of Wide Networks from Feynman Diagrams”. In: *ArXiv* abs/1909.11304 (2020).

[49] R. P. Feynman. “Forces in Molecules”. In: *Phys. Rev.* 56 (4 Aug. 1939), pp. 340–343.

[50] R Fletcher. “A new approach to variable metric algorithms”. In: *The computer journal* (1970).

[51] R. Fletcher. *Practical Methods of Optimization*. second. John Wiley & Sons, 1987.

[52] Thomas George et al. “Fast Approximate Natural Gradient Descent in a Kronecker-Factored Eigenbasis”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Montréal, Canada: Curran Associates Inc., 2018, pp. 9573–9583.

[53] Alan R Gillespie, Anne B Kahle, and Richard E Walker. “Color enhancement of highly correlated images. I. Decorrelation and HSI contrast stretches”. In: *Remote Sensing of Environment* 20.3 (1986), pp. 209–235.

[54] D Goldfarb. “A family of variable-metric methods derived by variational means”. In: *Mathematics of computation* (1970).

[55] Roger Grosse and James Martens. “A Kronecker-factored approximate Fisher matrix for convolution layers”. In: *arXiv preprint arXiv:1602.01407* (2016).

[56] Vineet Gupta, Tomer Koren, and Yoram Singer. “Shampoo: Preconditioned Stochastic Tensor Optimization”. In: *CoRR* abs/1802.09568 (2018). arXiv: `1802.09568`.

[57] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I*. second. Springer, 1993.

[58] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II*. second. Springer, 1996.

[59] Trevor Hastie et al. "Surprises in High-Dimensional Ridgeless Least Squares Interpolation". In: *Annals of Statistics* 50 (2 2022), pp. 949–986.

[60] Takahiro Hatano and S. I. Sasa. "Steady-State Thermodynamics of Langevin Systems". In: *Phys. Rev. Lett.* 86 (16 Apr. 2001), pp. 3463–3466.

[61] P Hennig. "Fast probabilistic optimization from noisy gradients". In: *International Conference on Machine Learning* (2013).

[62] P. D. Hislop and I. M. Sigal. *Introduction to Spectral Theory with Applications to Schrödinger Operators*. New York: Springer, 1996.

[63] Jiaoyang Huang and H B Yau. "Dynamics of Deep Neural Networks and Neural Tangent Hierarchy". In: *ArXiv* abs/1909.08156 (2019).

[64] Lei Huang et al. "Decorrelated Batch Normalization". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 791–800.

[65] Aapo Hyvärinen, Jarmo Hurri, and Patrick O. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*. 1st. Springer Publishing Company, Incorporated, 2009.

[66] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456.

[67] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems*. 2018, pp. 8571–8580.

[68] Fredrick A Jenet et al. "Detecting the stochastic gravitational wave background using pulsar timing". In: *The Astrophysical Journal Letters* 625.2 (2005), p. L123.

[69] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[70] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs.LG]* (2014), pp. 1–9.

[71] T Koide. "Perturbative expansion of irreversible work in Fokker–Planck equationà laquantum mechanics". In: *Journal of Physics A: Mathematical and Theoretical* 50.32 (July 2017), p. 325001.

[72] Artemy Kolchinsky, Brendan D Tracey, and Steven Van Kuyk. "Caveats for information bottleneck in deterministic scenarios". In: *arXiv preprint arXiv:1808.07593* (2018).

[73] Hanspeter Kraft and C. Procesi. *Classical invariant theory: a primer*. 1996.

[74] Walid Krichene, Alexandre M. Bayen, and Peter L. Bartlett. "Accelerated Mirror Descent in Continuous and Discrete Time". In: *Advances in Neural Information Processing Systems 28* (2015), pp. 2845–2853.

[75] L. D. Landau and E. M. Lifshitz. *Quantum Mechanics (Non-relativistic theory), Third Edition.* Oxford, UK: Butterworth-Heinemann, 1977.

[76] Yann Le Cun et al. "Efficient Backprop". In: *Neural Networks, Tricks of the Trade.* Lecture Notes in Computer Science LNCS 1524. Springer Verlag, 1998.

[77] Jaehoon Lee et al. "Finite Versus Infinite Neural Networks:an Empirical Study". In: *in preparation* (2020).

[78] Jaehoon Lee et al. "Wide neural networks of any depth evolve as linear models under gradient descent". In: *Advances in neural information processing systems.* 2019, pp. 8570–8581.

[79] Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. "Trust region newton method for logistic regression". In: *The Journal of Machine Learning Research* 9 (2008), pp. 627–650.

[80] Etai Littwin, Tomer Galanti, and L. Wolf. "On the Optimization Dynamics of Wide Hypernetworks". In: *ArXiv* abs/2003.12193 (2020).

[81] Dong C DC Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1-3 (1989), pp. 503–528.

[82] Yao Lu et al. "Block Mean Approximation for Efficient Second Order Optimization". In: *ArXiv* abs/1804.05484 (2018).

[83] Christian Maes and Karel Netočný. "A Nonequilibrium Extension of the Clausius Heat Theorem". In: *Journal of Statistical Physics* 154.1 (2014), pp. 188–203.

[84] Maren Mahsereci and Philipp Hennig. "Probabilistic Line Searches for Stochastic Optimization". In: *Journal of Machine Learning Research* 18 (2017), pp. 1–59.

[85] Dibyendu Mandal and Christopher Jarzynski. "Analysis of slow transitions between nonequilibrium steady states". In: *Journal of Statistical Mechanics: Theory and Experiment* 2016.6 (2016), p. 063204.

[86] James Martens. "Deep learning via Hessian-free optimization". In: *Proceedings of the 27th International Conference on Machine Learning (ICML).* Vol. 951. 2010.

[87] James Martens, Jimmy Ba, and Matt Johnson. "Kronecker-factored Curvature Approximations for Recurrent Neural Networks". In: *International Conference on Learning Representations.* 2018.

[88] James Martens and Roger Grosse. "Optimizing neural networks with kronecker-factored approximate curvature". In: *International conference on machine learning.* 2015, pp. 2408–2417.

[89] I. A. Martínez et al. "Brownian Carnot engine". In: *Nat Phys* 12 (Jan. 2016), pp. 67–70.

[90] Michael Muehlebach and Michael I. Jordan. "A Dynamical Systems Perspective on Nesterov Acceleration". In: *Proceedings of the International Conference on Machine Learning* (2019), pp. 1–7.

[91] Michael Muehlebach and Michael I. Jordan. "Continuous-time Lower Bounds for Gradient-based Algorithms". In: *Proceedings of the International Conference on Machine Learning* (2020), pp. 1–13.

[92] Michael Muehlebach and Michael I. Jordan. "Optimization with Momentum: Dynamical, Control-Theoretic, and Symplectic Perspectives". In: *arXiv: 2002.12493* (2020).

[93] Yurii Nesterov. *Introductory Lectures on Convex Optimization*. Springer, 2004.

[94] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. second. Springer Science and Business Media, 2006.

[95] Erkki Oja and Juha Karhunen. "On Stochastic Approximation of the Eigenvectors and Eigenvalues of the Expectation of a Random Matrix". In: *Journal of Mathematical Analysis and Applications* 106 (1985).

[96] K. Osawa et al. "Scalable and Practical Natural Gradient for Large-Scale Deep Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1.

[97] G. A. Pavliotis. *Stochastic Processes and Applications*. Vol. 60. Texts in Applied Mathematics. New York: Springer-Verlag, 2014.

[98] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice". In: *Advances in neural information processing systems*. 2017, pp. 4785–4795.

[99] Eric Poisson. *A Relativist's Toolkit: The Mathematics of Black-Hole Mechanics*. Cambridge University Press, 2004.

[100] Nasim Rahaman et al. "On the spectral bias of neural networks". In: *arXiv preprint arXiv:1806.08734* (2018).

[101] Hannes Risken. *Fokker-Planck Equation*. Berlin: Springer, 1984. ISBN: 978-3-642-96807-5.

[102] Hannes Risken. "Solutions of the Fokker-Planck Equation in Detailed Balance". In: *Z. Physik* 251 (1972), pp. 231–243.

[103] Michal Rolínek and Georg Martius. "L4: Practical loss-based stepsize adaptation for deep learning". In: *Advances in Neural Information Processing Systems 31* (2018), pp. 1–11.

[104] Basri Ronen et al. "The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 4761–4771.

[105] Grant M. Rotskoff and Gavin E. Crooks. "Optimal control in nonequilibrium systems: Dynamic Riemannian geometry of the Ising model". In: *Phys. Rev. E* 92 (6 2015), 060102(R).

[106] Grant M. Rotskoff, Gavin E. Crooks, and Eric Vanden-Eijnden. "Geometric approach to optimal nonequilibrium control: Minimizing dissipation in nanomagnetic spin systems". In: *Phys. Rev. E* 95 (1 2017), p. 012148.

[107] George Ruppeiner. "Thermodynamics: A Riemannian geometric model". In: *Phys. Rev. A* 20 (4 Oct. 1979), pp. 1608–1613.

[108] P Salamon, J Nulton, and E Ihrig. "On the relation between entropy and energy versions of thermodynamic length". In: *The Journal of Chemical Physics* 80.1 (1984), pp. 436–437.

[109] Peter Salamon and R Stephen Berry. "Thermodynamic length and dissipated availability". In: *Physical Review Letters* 51.13 (1983), p. 1127.

[110] Andrew M Saxe et al. "On the information bottleneck theory of deep learning". In: *Journal of Statistical Mechanics: Theory and Experiment* 2019.12 (Dec. 2019), p. 124020.

[111] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[112] F Schlögl. "Thermodynamic metric and stochastic measures". In: *Zeitschrift für Physik B Condensed Matter* 59.4 (1985), pp. 449–454.

[113] Tim Schmiedl and Udo Seifert. "Optimal Finite-Time Processes In Stochastic Thermodynamics". In: *Phys. Rev. Lett.* 98 (10 Mar. 2007), p. 108301.

[114] Nicol Schraudolph, Jin Yu, and Simon Günter. "A stochastic quasi-Newton method for online convex optimization". In: *AIstats* (2007).

[115] Daniel V. Schroeder. *An Introduction to Thermal Physics*. Oxford: Oxford University Press, 2021.

[116] Ravid Schwartz-Ziv and Alexander A Alemi. "Information in Infinite Ensembles of Infinitely-Wide Neural Networks". In: *arXiv preprint arXiv:1911.09189* (2019).

[117] Udo Seifert. "Stochastic thermodynamics, fluctuation theorems and molecular machines". In: *Reports on Progress in Physics* 75 (Dec. 2012), p. 126001.

[118] Ken Sekimoto. "Kinetic Characterization of Heat Bath and the Energetics of Thermal Ratchet Models". In: *Journal of the Physical Society of Japan* 66.5 (1997), pp. 1234–1237.

[119] Ken Sekimoto. *Stochastic Energetics*. Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-05411-2.

[120] Ken Sekimoto and Shin-ichi Sasa. "Complementarity Relation for Irreversible Process Derived from Stochastic Energetics". In: *Journal of the Physical Society of Japan* 66.11 (Nov. 1997), pp. 3326–3328.

[121] Christopher J Shallue et al. “Measuring the effects of data parallelism on neural network training”. In: *arXiv preprint arXiv:1811.03600* (2018).

[122] DF Shanno. “Conditioning of quasi-Newton methods for function minimization”. In: *Mathematics of computation* (1970).

[123] Noam Shazeer and Mitchell Stern. “Adafactor: Adaptive learning rates with sublinear memory cost”. In: *arXiv preprint arXiv:1804.04235* (2018).

[124] Ravid Shwartz-Ziv and Naftali Tishby. “Opening the black box of deep neural networks via information”. In: *arXiv preprint arXiv:1703.00810* (2017).

[125] Eero P Simoncelli and Bruno A Olshausen. “Natural Image Statistics and Neural Representation”. In: *Annual Review of Neuroscience* 24.1 (2001), pp. 1193–1216.

[126] David A. Sivak and Gavin E. Crooks. “Thermodynamic geometry of minimum-dissipation driven barrier crossing”. In: *Phys. Rev. E* 94 (5 Nov. 2016), p. 052106.

[127] David A. Sivak and Gavin E. Crooks. “Thermodynamic Metrics and Optimal Paths”. In: *Phys. Rev. Lett.* 108 (19 May 2012), p. 190602.

[128] Jascha Sohl-Dickstein. “The natural gradient by analogy to signal whitening, and recipes and tricks for its use”. In: *arXiv preprint arXiv:1205.1828* (2012).

[129] Jascha Sohl-Dickstein, Ben Poole, and Surya Ganguli. “Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods”. In: *International Conference on Machine Learning*. 2014, pp. 604–612.

[130] Weijie Su, Stephen Boyd, and Emmanuel J. Candès. “A Differential Equation for Modeling Nesterov’s Accelerated Gradient Method: Theory and Insights”. In: *Journal of Machine Learning Research* 17.153 (2016), pp. 1–43.

[131] Peter Sunehag et al. “Variable metric stochastic approximation theory”. In: *arXiv preprint arXiv:0908.3529* (Aug. 2009). arXiv: `0908.3529`.

[132] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude.* COURSERA: Neural Networks for Machine Learning. 2012.

[133] Naftali Tishby and Noga Zaslavsky. “Deep learning and the information bottleneck principle”. In: *2015 IEEE Information Theory Workshop (ITW)*. IEEE. 2015, pp. 1–5.

[134] C. Van den Broeck, S. I. Sasa, and U. Seifert. “Focus on stochastic thermodynamics”. In: *New Journal of Physics* 18 (2016), p. 020401.

[135] Sharan Vaswani et al. “Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates”. In: *Advances in Neural Information Processing Systems 32* (2019), pp. 1–14.

[136] Sharan Vaswani et al. *To Each Optimizer a Norm, To Each Norm its Generalization.* 2020. arXiv: `2006.06821 [cs.LG]`.

[137] Oriol Vinyals and Daniel Povey. "Krylov subspace descent for deep learning". In: *arXiv preprint arXiv:1111.4259* (2011).

[138] Neha S. Wadia, Michael I. Jordan, and Michael Muehlebach. "Optimization with Adapative Step Size Selection from a Dynamical Systems Perspective". In: *Proceedings of the 35th Conference on Neural Information Processing Systems. NeurIPS Workshop on Optimization for Machine Learning.* 2021.

[139] Neha S. Wadia, Ryan V. Zarcone, and Michael R. DeWeese. "Solution to the Fokker-Planck Equation for Slowly Driven Brownian Motion: Emergent Geometry and a Formula for the Corresponding Thermodynamic Metric". In: *Physical Review E* 105 (2022), p. 034130.

[140] Neha S. Wadia et al. "Whitening and Second Order Optimization Both Make Information in the Dataset Unusable During Training, and Can Reduce or Prevent Generalization". In: *ICML*. 2021.

[141] Frank Weinhold. "Metric geometry of equilibrium thermodynamics". In: *The Journal of Chemical Physics* 63.6 (1975), pp. 2479–2483.

[142] Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. "A variational perspective on accelerated methods in optimization". In: *Proceedings of the National Academy of Sciences* 113.47 (2016), E7351–E7358.

[143] Simon Wiesler and Hermann Ney. "A Convergence Analysis of Log-linear Training". In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*. NIPS'11. Granada, Spain: Curran Associates Inc., 2011, pp. 657–665.

[144] Ashia C Wilson et al. "The marginal value of adaptive gradient methods in machine learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4148–4158.

[145] Ashia C. Wilson et al. "The Marginal Value of Adaptive Gradient Methods in Machine Learning". In: *Advances in Neural Information Processing Systems 30* (2017), pp. 1–11.

[146] Philip Wolfe. "Convergence Conditions for Ascent Methods". In: *SIAM Review* 11.2 (1969), pp. 226–235.

[147] Lechao Xiao et al. "Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks". In: *arXiv preprint arXiv:1806.05393* (2018).

[148] Aolin Xu and Maxim Raginsky. "Information-theoretic analysis of generalization capability of learning algorithms". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 2524–2533.

[149] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *CoRR* abs/1605.07146 (2016). arXiv: `1605.07146`.

[150] Matthew D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. In: *CoRR* abs/1212.5701 (2012). arXiv: `1212.5701`.

[151] Guodong Zhang, James Martens, and Roger B Grosse. “Fast Convergence of Natural Gradient Descent for Over-Parameterized Neural Networks”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8082–8093.

[152] Guodong Zhang et al. “Three mechanisms of weight decay regularization”. In: *arXiv preprint arXiv:1810.12281* (2018).

[153] Huishuai Zhang et al. “Block-diagonal Hessian-free Optimization for Training Neural Networks”. In: *CoRR* abs/1712.07296 (2017). arXiv: `1712.07296`.

[154] Patrick R. Zulkowski and Michael R. DeWeese. “Optimal control of overdamped systems”. In: *Phys. Rev. E* 92 (3 2015), p. 032117.

[155] Patrick R. Zulkowski et al. “Geometry of thermodynamic control”. In: *Phys. Rev. E* 86 (4 Oct. 2012), p. 041148.