# Noise-Aware Stochastic Optimization

**Dissertation**
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Lukas Balles, M.Sc.
aus Hardheim

Tübingen
2021

*To Maren and Selma.*

# *Acknowledgements*

I want to thank Philipp Hennig for the many things he has taught me and for his contagious passion for research. And for allowing me to put that research to the side when other things mattered more. I am grateful to Michael Black and Georg Martius for serving on my thesis advisory committee, and to Matthias Hein, Jakob Macke, and Michael Black for serving on my examination committee. I also want to thank Leila Masri, the coordinator of the International Max Planck Research School for Intelligent Systems, for her support and encouragement.

I had the privilege of working with many great colleagues at the Max Planck Institute for Intelligent Systems and the University of Tübingen. I cannot name all of them here, but I would be amiss not to thank Maren Mahsereci, Frederik Kunstner, Frank Schneider, Hans Kersting, Damien Garreau, Michael Schober, Anurag Ranjan, Javier Romero, Alexandra Gessner, Filip de Roos, Motonobu Kanagawa, Felix Dangel and Agustinus Kristiadi.

Finally, I thank my wonderful wife, Maren Balles, for her strength and unrelenting support.

Berlin, August 2021

# *Abstract*

First-order stochastic optimization algorithms like stochastic gradient descent (SGD) are the workhorse of modern machine learning. With their simplicity and low per-iteration cost, they have powered the immense success of deep artificial neural network models. Surprisingly, these stochastic optimization methods are essentially *unaware* of stochasticity. Neither do they collect information about the stochastic noise associated with their gradient evaluations, nor do they have explicit mechanisms to adjust their behavior accordingly. This thesis presents approaches to make stochastic optimization methods *noise-aware* using estimates of the (co-)variance of stochastic gradients.

First, we show how such variance estimates can be used to automatically adapt the minibatch size for SGD, i.e., the number of data points sampled in each iteration. This can replace the usual decreasing step size schedule required for convergence, which is much more challenging to automate. We highlight that both approaches can be viewed through the same lens of reducing the mean squared error of the gradient estimate.

Next, we identify an implicit variance adaptation mechanism in the ubiquitous ADAM method. In particular, we show that it can be seen as a version of SIGN-SGD with a coordinatewise "damping" based on the stochastic gradient's signal-to-noise ratio. We make this variance adaptation mechanism explicit, formalize it, and transfer it from SIGN-SGD to SGD.

Finally, we critically discuss a family of methods that preconditions stochastic gradient descent updates with the so-called "empirical Fisher" matrix, which is closely related to the stochastic gradient covariance matrix. This is usually motivated from information-geometric considerations as an approximation to the Fisher information matrix. We caution against this argument and show that the empirical Fisher approximation has fundamental theoretical flaws. We argue that preconditioning with the empirical Fisher is better understood as a form of variance adaptation.

# *Zusammenfassung*

Sochastische Optimierungsverfahren erster Ordnung, wie zum Beispiel das stochastische Gradientenverfahren (stochastic gradient descent, SGD), sind das Arbeitstier des modernen maschinellen Lernens. Mit ihrer Einfachheit und niedrigen Kosten pro Iteration haben sie den immensen Erfolg künstlicher neuronaler Netze maßgeblich vorangetrieben. Überraschender Weise sind diese stochastischen Optimierungsmethoden blind gegenüber Stochastizität. Weder sammeln sie Informationen über das stochastische Rauschen der verwendeten Gradientenauswertungen, noch verfügen sie über explizite Mechanismen, um ihr Verhalten an dieses Rauschen anzupassen. Diese Arbeit präsentiert Ansätze, stochastischen Optimierungsverfahren mittels Schätzung der (Ko-)Varianz der stochastischen Gradienten ein "Bewusstsein" für dieses Rauschen zu geben.

Zuerst zeigen wir, wie solche Varianzschätzungen genutzt werden können, um die sogenannte Minibatchgröße bei SGD automatisch anzupassen. Dies kann eine üblicherweise verwendete abnehmende Schrittweite ersetzten, welche ihrerseits sehr viel schwerer zu automatisieren ist. Wir stellen heraus, dass beide Herangehensweisen aus einer gemeinsamen Perspektive betrachtet werden können, nämlich als Reduktion der mittleren quadratischen Abweichung des Gradientenschätzers.

Als nächstes identifizieren wir in der außergewöhnlich populären ADAM-Methode einen impliziten Varianzadaptierungsmechanismus. Wir betrachten ADAM als eine Version von SIGN-SGD mit koordinatenweiser "Dämpfung" auf Basis des Signal-zu-Rausch-Verhältnisses des stochastischen Gradienten. Wir machen diesen Mechanismus explizit, formalisieren ihn, und übertragen ihn von SIGN-SGD zu SGD.

Abschließend folgt eine kritische Diskussion einer Methodenfamilie, welche SGD mit der sogenannten "empirischen Fisher-Matrix" präkonditionert. Diese Matrix ist eng mit der Kovarianzmatrix des stochastischen Gradienten verwandt. Die empirische Fisher-Matrix wird üblicherweise als Approximation für die Fisher-Matrix und somit aus informationsgeometrischen Überlegungen motiviert. Wir kritisieren dieses Argument und zeigen, dass diese Approximation fundamentale theoretische Schwächen hat. Wir argumentieren, dass die Präkonditionierung mit der empirischen Fisher-Matrix besser als eine Form von Varianzadaptierung gesehen werden sollte.

# *Contents*

# 1
# Introduction

Optimization lies at the heart of machine learning. After training data has been collected and a suitable model has been devised, we usually want to find the model parameters that provide the *best* explanation for the observed data. We have burdened ourselves with an optimization problem.

This is most prominently reflected in empirical risk minimization, the principal paradigm of statistical learning. Consider the supervised task of learning to predict a target $y \in \mathbb{Y}$ from input $x \in \mathbb{X}$ based on a training set $(x_1, y_1), \ldots, (x_N, y_N)$. Given a model $h_\theta \colon \mathbb{X} \to \mathbb{Y}$ with parameters $\theta$ and a loss function $\ell \colon \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}$, we want to set $\theta$ such as to minimize the so-called *empirical risk*

$$R(\theta) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} \ell_n(\theta), \quad \ell_n(\theta) \stackrel{\text{def}}{=} \ell(h_\theta(x_n), y_n). \tag{1.1}$$

For all but the most basic machine learning models, a minimizer of $R(\theta)$ cannot be found in closed form and we have to rely on numerical optimization methods. Such an algorithm repeatedly evaluates the model and generates increasingly better choices for its parameters. This procedure is where the model meets the data, where the numbers are crunched and knowledge is extracted.

In recent years, increasingly complex machine learning models—in particular deep artificial neural networks—have been remarkably successful in learning from ever larger datasets. This makes optimization a key computational challenge. As you read this, tens of thousands of processing units in data centers all over the world are performing iteration after iteration of an optimization algorithms. They are training machine learning models to answer search queries, translate texts, predict stock prices, recommend products to customers shopping online, or track moving objects in video scenes.

## 1.1 Stochastic Optimization

One of the most basic methods, but still a cornerstone of mathematical optimization, is gradient descent (GD), which can be traced back to Cauchy [1847]. It minimizes Eq. (1.1) with updates of the

form

$$\theta_{t+1} = \theta_t - \alpha \nabla R(\theta_t), \tag{1.2}$$

where $\alpha$ is a scalar, positive step size. The negative gradient $-R(\theta_t)$ is the direction of steepest descent and—with an appropriate step size—each iteration achieves a decrease in function value.

The gradient of the empirical risk takes the form

$$\nabla R(\theta) = \frac{1}{N} \sum_{n=1}^{N} \nabla \ell_n(\theta) \tag{1.3}$$

and, thus, requires the computation of $N$ individual per-example gradients $\nabla \ell_n(\theta)$. When using a computationally complex model to learn from a large training set, evaluating this full gradient in each iteration becomes inefficient or even prohibitively expensive.

In that case, we can resort to *stochastic* gradient descent (SGD)[1], which uses a cheap stochastic estimate of the gradient for each update. In its simplest form, at iteration $t$, SGD samples a data point $n^{(t)}$ uniformly at random from $\{1, \ldots, N\}$ and updates

$$\theta_{t+1} = \theta_t - \alpha_t g_t, \quad g_t \stackrel{\text{def}}{=} \nabla \ell_{n^{(t)}}(\theta_t). \tag{1.4}$$

The so-called stochastic gradient $g_t$ provides an unbiased estimate of the true gradient,

$$\mathbf{E}[g_t] = \sum_{n=1}^{N} \underbrace{\mathbf{P}[n^{(t)} = n]}_{=1/N} \nabla \ell_n(\theta_t) = \nabla R(\theta_t), \tag{1.5}$$

while drastically reducing the computational cost per iteration, since we compute only one instead of $N$ individual gradients.

[1] Stochastic gradient descent dates back to Robbins and Monro [1951], who proposed a "Stochastic Approximation Method" for finding the root of a function given access to noisy measurements of its function value. Kiefer et al. [1952] first applied it specifically to gradient-based optimization.

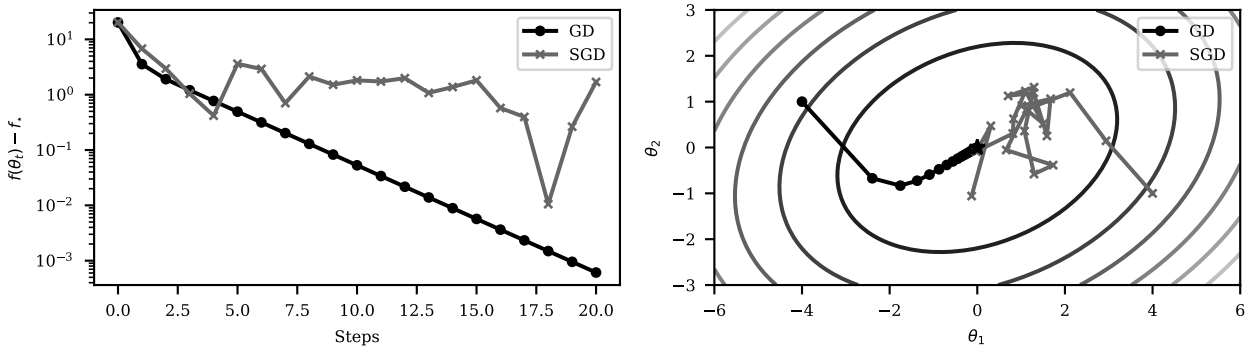


Figure 1.1: Gradient descent (GD) and stochastic gradient descent (SGD) on a two-dimensional quadratic objective. Both methods use the same constant step size. The plots depict the suboptimality in function value (left) and the iterates (right). While gradient descent converges, SGD goes into "diffusion" once it is near the optimum.

However, the optimization algorithm now has to make do with inexact gradient evaluations, subject to stochastic noise, which brings about new challenges. Most prominently, the stochasticity in the updates prevents SGD from converging to a solution with a fixed step size, see Figure 1.1. As shown by Robbins and Monro [1951], convergence of SGD requires us to set a decreasing step size schedule. In practice, these schedules are usually tuned manually

by trial and error. In the deterministic setting, step sizes are routinely adapted automatically with subroutines called *line searches*. Despite efforts of the research community and partial successes [Mahsereci and Hennig, 2015, 2017, Vaswani et al., 2019, Paquette and Scheinberg, 2020], robust and general line search methods for the stochastic setting are still lacking.

The stochastic regime also complicates the use of other tools and techniques from classical, deterministic optimization, such as automatic stopping criteria or the use of preconditioners that estimate the Hessian based on the observed changes in gradient (quasi-Newton methods).

## 1.2 Towards Noise-Aware Algorithms

Most popular stochastic optimization methods are more or less *oblivious* to the fact that they are operating in a stochastic regime. Stochastic gradient descent (SGD) is just the gradient descent method using the inexact gradient estimate in lieu of the true gradient. If the properties of that gradient estimate were to change—say it suddenly gets much less accurate—SGD would not change its behavior. The same can be said of most other currently-used stochastic optimization algorithms.

This thesis presents several approaches to make stochastic optimization methods *noise-aware*. The overarching goal is to design stochastic optimization algorithms that actively estimate properties of the stochastic error and adapt their behavior accordingly. A central object of interest which captures many interesting properties of the gradient estimate is its covariance matrix

$$\mathbf{Cov}[g_t] = \frac{1}{N}\sum_{n=1}^{N}\left[\nabla\ell_n(\theta_t) - \nabla R(\theta_t)\right]\left[\nabla\ell_n(\theta_t) - \nabla R(\theta_t)\right]^{\mathsf{T}}. \quad (1.6)$$

It describes the scale and "geometry" of the stochastic error associated with $g_t$. The exact covariance matrix will, of course, be unknown in practice since it depends on all $N$ individual gradients—the absence of which defines the stochastic optimization regime. But, as we will see shortly, it can be estimated in the common setting of minibatch SGD, where a small number $m \ll N$ of data points are sampled for each SGD iteration.

The following chapters present various ways in which stochastic optimization methods may utilize some of the information encoded in this gradient covariance matrix.

## 1.3 Overview

Chapter 2 will briefly review some preliminaries from machine learning and deterministic optimization. Chapter 3 will set the scene by formally introducing the stochastic optimization setting and discussing the basic stochastic gradient descent algorithm.

In Chapter 4, we discuss how the convergence-harming effect of gradient noise can be counteracted either by decreasing the step size (the standard approach) or, alternatively, by increasing the minibatch size, i.e., the number of data points sampled in each iteration. In fact, both approaches can be viewed through the same lens: reducing the mean squared error of the local gradient estimate. We show that, while the optimal greedy choice is to use a batch size of 1 and decrease the step size, there are practical arguments in favor of an increasing batch size. In particular, it lends itself more easily to automation based on gradient variance estimates; we present CABS, a heuristic for automatic batch size adaptation.

Chapter 5 investigates the very popular ADAM method [Kingma and Ba, 2015] and shows that it combines two aspects. Firstly, it can be seen as a version of SIGN-SGD, that is, the method that updates along the vector of coordinatewise signs of the negative stochastic gradient. Secondly, on top of this sign-based direction, ADAM employs a coordinatewise "damping" based on an estimate of the signal-to-noise ratio of the stochastic gradient in each coordinate. We defer a study of the first aspect to Chapter 6 and investigate the second aspect, which we term *variance adaptation*. We formalize the idea and transfer it from SIGN-SGD to SGD, which results in the *stochastic variance-adapted gradient* (SVAG) method. Furthermore, we perform an ablation study of the two aspects, which shows that the sign aspect accounts for most of the difference between ADAM and SGD.

Inspired by this empirical significance of the sign aspect, we take to investigating it theoretically in Chapter 6. Focusing on the curvature aspects, we study the noise-free version, sign gradient descent, as steepest descent with respect to the maximum norm. We show that sign gradient descent can outperform gradient descent for objectives whose Hessian is characterized by two properties: (i) a few dominant outlier eigenvalues, and (ii) some degree of concentration on its diagonal. Interestingly, these theoretical findings are matched by recent empirical studies of properties of the Hessian in neural network training objectives.

Finally, in Chapter 7, we discuss a family of methods that preconditions stochastic gradient descent with the so-called "empirical Fisher" matrix, which is closely related to the stochastic gradient covariance matrix. This is usually motivated from information-geometric considerations as an approximation to the Fisher information matrix. We caution against this argument and show that the empirical Fisher approximation has fundamental theoretical flaws. We argue that preconditioning with the empirical Fisher might better be understood as a form of variance adaptation.

# 2
# *Preliminaries*

This manuscript makes extensive use of linear algebra, multivariate calculus, and basic probability theory. General familiarity with these topics is assumed; more specific concepts and results will be introduced where needed. This chapter contains a short primer on basic concepts in machine learning in Section 2.1 and (deterministic) mathematical optimization in Section 2.2. Readers familiar with these topics may safely skip this chapter.

## *2.1 Machine Learning*

Machine learning is the study of algorithms that learn to solve certain tasks by inspecting *examples* in the form of training data. It has grown into a broad field, using a wide range of tools to tackle various kinds of tasks with different assumptions on the available data.

A common scenario, which will be our primary interest, is that of *supervised learning*, where we want to learn to predict a target quantity $y \in \mathbb{Y}$ given an input $x \in \mathbb{X}$. We distinguish between regression problems, where $y$ is a continuous quantity, and classification problems, where $y$ is categorical. For example, in a regression problem, we may want to predict the price $y \in \mathbb{R}$ of a house based on a vector $x \in \mathbb{R}^p$ describing some of its features, e.g., its square footage, the number of rooms, or the crime rate of the neighborhood. As an example for a classification problem, assume you are given a digital image of a handwritten digit and want to identify the digit. The goal of a supervised learning method is to construct a predictor $h\colon \mathbb{X} \to \mathbb{Y}$ based on a training set of examples of $(x, y)$ pairs.

In this section, we are going to describe a few key concepts of statistical machine learning, highlighting its close connection to optimization.

### *2.1.1 Empirical Risk Minimization*

Empirical risk minimization (ERM) is the principal paradigm of statistical machine learning. The relationship between the target $y \in \mathbb{Y}$ and the input $x \in \mathbb{X}$ is modeled using joint probability distri-

bution $p(x, y)$. In the ERM paradigm we choose a hypothesis space of possible predictors $h\colon \mathbb{X} \to \mathbb{Y}$, which usually takes the form of a parametrized family $\mathcal{H} = \{h_\theta \mid \theta \in \mathbb{R}^d\}$. The hypothesis class is also referred to as a *model*, because it posits a certain relationship between $x$ and $y$, and $\theta$ are the model parameters. Furthermore, we specify a loss function $\ell\colon \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}$ such that $\ell(y, \hat{y})$ quantifies our "discontent" with a prediction $\hat{y}$ when the true target is $y$. Our goal is to minimize the expected loss,

$$\mathcal{R}(\theta) = \mathbf{E}_{x,y}[\ell(y, h_\theta(x))], \tag{2.1}$$

which is also referred to as the *risk* of the predictor $h_\theta$. Of course, the true data distribution $p(x, y)$ is usually unknown and/or the expectation in Eq. (2.1) is intractable. Instead, we rely on training set

$$D = \{(x_1, y_1), \ldots, (x_N, y_N)\}, \tag{2.2}$$

which is assumed to consist of iid samples from the data-generating distribution, and minimize the so-called *empirical risk*

$$R_D(\theta) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, h_\theta(x_n)). \tag{2.3}$$

The expectation over $p(x, y)$ in Eq. (2.1) is approximated with the sample average over the training set to construct a tractable surrogate for the risk. Fitting model parameters by minimizing the empirical risk is called empirical risk minimization. (We drop the subscript $D$ in the empirical risk whenever the dependence on the training set need not be made explicit.)

*Example: Linear Least-Squares Regression.* Let $\mathbb{X} = \mathbb{R}^p$ and $\mathbb{Y} = \mathbb{R}$. We choose the hypothesis space of affine maps, $h_{w,b}(x) = w^\mathsf{T} x + b$, parametrized by $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$. We use the squared loss $\ell(y, \hat{y}) = \frac{1}{2}(\hat{y} - y)^2$. The resulting empirical risk minimization problem,

$$R(\theta) = \frac{1}{2N} \sum_{n=1}^{N} (w^\mathsf{T} x_n + b - y_n)^2, \tag{2.4}$$

is known as (linear) least-squares regression.

Least-squares regression, a method which dates back at least to Legendre [1805] and Gauss [1809], is probably the earliest application of the ERM principle. The key idea has since featured in many works in statistics and the burgeoning field of (statistical) machine learning. The empirical risk minimization principle has been summarized and formalized in the modern machine learning context by Vapnik [1992].

### 2.1.2 *Generalization*

The idea underlying the ERM paradigm is that the empirical risk $R(\theta)$ will approximate the true risk $\mathcal{R}(\theta)$ sufficiently well such that

a solution minimizing the former will also achieve a low value in the latter. This is the question of *generalization*: Does a model fit to a training set *generalize* to previously unseen data drawn from the same distribution? We will briefly discuss this question with a special emphasis on the implications for optimization.

In a nutshell, the generalization behavior depends on the size of the training set and the *capacity*[1] of the chosen model class. Denote by

$$\begin{aligned} \theta_D^\star &= \arg\min_\theta R_D(\theta), \\ \theta^\star &= \arg\min_\theta \mathcal{R}(\theta) \end{aligned} \tag{2.5}$$

the solutions found by minimizing the empirical risk and the true risk, respectively. Let

$$\mathcal{R}_{\min} = \min_{f\colon \mathbb{X}\to\mathbb{Y}} \mathbf{E}_{x,y}[\ell(y, f(x))] \tag{2.6}$$

be the lowest risk achievable by any possible predictor.[2] With that, we can decompose the excess risk

$$\mathcal{R}(\theta_D^\star) - \mathcal{R}_{\min} = \underbrace{\mathcal{R}(\theta_D^\star) - \mathcal{R}(\theta^\star)}_{=\varepsilon_{\text{est}}} + \underbrace{\mathcal{R}(\theta^\star) - \mathcal{R}_{\min}}_{=\varepsilon_{\text{app}}} \tag{2.7}$$

into an *estimation error* $\varepsilon_{\text{est}}$ and an *approximation error* $\varepsilon_{\text{app}}$. The estimation error arises from optimizing the empirical risk instead of the true risk. The approximation error is due to a restricted model class, which usually will not contain the optimal predictor and therefore not achieve the minimal risk. Increasing model capacity will decrease the approximation error, since we may get closer to the optimal predictor. But it will tend to increase the estimation error, since a high-capacity model class will tend to *overfit* to the particular training set. A larger training set size will bring the empirical risk closer to the true risk, thereby decreasing the estimation error; the approximation error is unaffected by the training set size.

From the optimization perspective, an interesting aspect is that overfitting happens over the course of the optimization procedure. Let $\theta_t$, $t \in \mathbb{N}$, be a sequence of iterates generated by an optimization algorithm which minimizes the empirical risk $R(\theta)$. If the optimization procedure is doing its job, we expect $R(\theta_t)$ to be decreasing over time. But what about $\mathcal{R}(\theta_t)$? A typical behavior is depicted in Figure 2.1. Early on, $\mathcal{R}(\theta_t) \approx R(\theta_t)$ and the true risk decreases alongside the empirical risk as desired. As the optimization progresses, $\theta_t$ gets more and more adapted to the training set and the true risk begins to systematically exceed the empirical risk. At some point, the true risk plateaus and can even start to increase, as $\theta_t$ *overfits* to the training set. This effect can be detected and counteracted by keeping a separate sample of data points, a so-called validation set. Since the validation set is statistically independent of $\theta_t$, the average loss computed on the validation set is an unbiased estimate of the true risk. Monitoring the validation loss lets us detect overfitting and stop the optimization process when it occurs.

[1] A formal definition "capacity" of a model class is beyond the scope of this brief introduction; there is not even a unique definition. For our purposes, we can think of it as the *size* of the hypothesis space $\mathcal{H}$. A large hypothesis space contains many different predictions and has the capacity to fit many different datasets. A small hypothesis class is more restricted.

[2] This is referred to as the *irreducible risk* or *Bayes risk*.

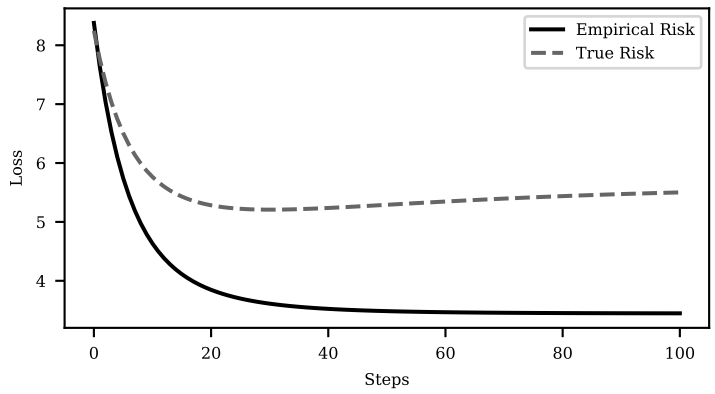


Figure 2.1: Overfitting occurs gradually during empirical risk minimization. Early on, $\mathcal{R}(\theta_t) \approx R(\theta_t)$ and the true risk decreases alongside the empirical risk as desired. As the optimization progresses, $\theta_t$ gets more and more adapted to the training set and the true risk begins to systematically exceed the empirical risk. At some point, the true risk plateaus and can even start to increase.

This plot was created by running gradient descent on an artificially generated linear least-squares regression problem as described in Section 2.1.1.

The phenomenon of overfitting hints at a larger point: The best optimization algorithm is not necessarily the best learning algorithm. After all, the optimizer's objective (the empirical risk) is not the quantity we are ultimately interested in (the true risk). Loosely speaking, an inexact optimization algorithm that gets us reasonably close to the empirical risk minimum very quickly might be preferable to an optimization algorithm that converges faster asymptotically but may be slow initially. At some point, a further reduction in the empirical risk will not pay off in terms of the true risk and investing additional computational resources into the optimization will be wasteful. This was highlighted and formalized in a landmark paper by Bottou and Bousquet [2008], who amend the approximation-estimation trade-off (Eq. 2.7) by accounting for the effect of inexact optimization of the empirical risk. Let $\hat{\theta}_D$ be the (inexact) solution returned by an optimization method applied to the empirical risk. We can decompose its excess risk as

$$\mathcal{R}(\hat{\theta}_D) - \mathcal{R}_{\min} = \underbrace{\mathcal{R}(\hat{\theta}_D) - \mathcal{R}(\theta^\star_D)}_{=\varepsilon_{\text{opt}}} + \underbrace{\mathcal{R}(\theta^\star_D) - \mathcal{R}(\theta^\star)}_{=\varepsilon_{\text{est}}} + \underbrace{\mathcal{R}(\theta^\star) - \mathcal{R}_{\min}}_{=\varepsilon_{\text{app}}}. \tag{2.8}$$

Investing computational resources into the optimization method will only reduce the *optimization error* $\varepsilon_{\text{opt}}$. As Bottou and Bousquet [2008] show, in the regime of large training sets, this investment might not pay off and would be better allocated to incorporate more data (if the estimation error is large) or to increase the capacity of the model class (if the approximation error is large).

Beyond that, if the empirical risk has multiple (local) minima, the optimization can have an effect as to which of these minima is found. This has found to be the case empirically in neural networks [e.g., Wilson et al., 2017] and studied theoretically in simpler problems [e.g., Soudry et al., 2018].

### 2.1.3 *Logistic Regression*

Before introducing artificial neural networks, the model class of primary interest for the research presented in this thesis, we introduce the method of *logistic regression*, developed and popularized

by Berkson [1944]. Despite its name, it is a method for classification problems, more specifically, binary classification where $y \in \{0, 1\}$. As in linear regression, we use affine maps of the form $w^\mathsf{T} x + b$. The output is then passed through the so-called logistic (or sigmoid) function $\sigma \colon \mathbb{R} \to [0, 1]$,

$$\sigma(t) = \frac{1}{1 + e^{-t}}, \tag{2.9}$$

which squashes it to a value between 0 and 1. This results in a predictor

$$h_{w,b}(x) = \sigma(w^\mathsf{T} x + b) = \frac{1}{1 + e^{-w^\mathsf{T} x - b}}, \tag{2.10}$$

which we interpret as the predictive probability for $y = 1$, i.e., $h_{w,b}(x) \approx p(y = 1|x)$. As a loss function, the probabilistic formulation suggests to use the negative log likelihood of our prediction, meaning that predicting a probability of $\pi \in [0, 1]$ for $y = 1$ incurs a loss of

$$\ell(y, \pi) = -y \log \pi - (1 - y) \log(1 - \pi). \tag{2.11}$$

This results in an empirical risk of the form

$$R(w, b) = -\frac{1}{N} \sum_{n=1}^{N} \left( y_n \log h_{w,b}(x_n) + (1 - y_n) \log(1 - h_{w,b}(x_n)) \right). \tag{2.12}$$

Logistic regression is possibly the simplest parametric classification method and an interesting model problem for (stochastic) optimization. While linear regression poses *quadratic* optimization problems, for which there are specialized algorithms, logistic regression requires a general purpose non-linear optimization method. At the same time, it is a "well-behaved" optimization problem, in particular it can be shown to be smooth and convex (see Section 2.2 for an introduction to these concepts). It can also be seen as the simplest possible neural network model.

### 2.1.4 *Artificial Neural Networks*

Artificial neural networks are a broad family of machine learning models and associated learning algorithms. Historically, they have evolved from computational models of the brain but this biological analogy only serves as a loose inspiration in most contemporary uses of these models. Many different forms of neural network models have emerged, but they are all built on the same basic building block: the *artificial neuron*.

An artificial neuron can be described as a function which receives a vector of input signals $z \in \mathbb{R}^n$ and computes a single scalar output as

$$a(w^\mathsf{T} z + b), \tag{2.13}$$

where $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are the parameters of the neuron (called its weight vector and bias, respectively) and $a \colon \mathbb{R} \to \mathbb{R}$ is a nonlinear *activation function*. This basic structure was first proposed under

the name *perceptron* by Rosenblatt [1958], who used a threshold activation function,

$$a(t) = \begin{cases} 1 & \text{if } t > 0, \\ 0 & \text{if } t \leq 0, \end{cases} \tag{2.14}$$

inspired by biological neurons. More recent choices for the activation function include the sigmoid function (Eq. 2.9), the hyperbolic tangent, or the so-called rectified linear unit [ReLU, Nair and Hinton, 2010],

$$a(t) = \begin{cases} t & \text{if } t > 0, \\ 0 & \text{if } t \leq 0, \end{cases} \tag{2.15}$$

which has become ubiquitous in modern deep learning.

*Fully-Connected Neural Networks* Neural network models are composed of multiple layers of artificial neurons, where the outputs of neurons in one layer form the input of neurons in subsequent layers. Networks with many such layers are called *deep* and the term *deep learning* has been coined to describe machine learning with deep artificial neural networks. The simplest and historically earliest form of artificial neural networks are so-called fully-connected networks or multilayer perceptrons (MLPs). In a fully-connected network, the concatenated outputs of all neurons in one layer form the input vector for the neurons in the consecutive layer. Hence, the neurons in two consecutive layers are *fully* interconnected. Mathematically, we can describe a layer as a function $h^{(\ell)} \colon \mathbb{R}^{n_{\ell-1}} \to \mathbb{R}^{n_\ell}$,

$$h^{(\ell)}(z) = \sigma\left(W^{(\ell)} z + b^{(\ell)}\right), \tag{2.16}$$

parametrized by a *weight matrix* $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and a *bias vector* $b^{(\ell)} \in \mathbb{R}^{n_\ell}$. The application of the nonlinearity $\sigma$ has to be understood elementwise. The fully-connected network is simply a composition of such layer functions, $h_\theta = h^{(L)} \circ \ldots \circ h^{(1)}$ with compatible input and output sizes. The network is parametrized by the the collection of weights and biases of all layers, $\theta = (W^{(1)}, b^{(1)}, \ldots, W^{(L)}, b^{(L)})$. To construct a fully-connected neural network that maps and $n$-dimensional input to an $m$-dimensional output, we choose the input dimension of the first layer (input layer) to be $n_0 = n$ and the output dimension of the final final layer (output layer) to be $n_L = m$. Modelling choices include the number of intermediate ("hidden") layers and the number of neurons in each of them, as well as the choice of activation function. These aspects are jointly referred to as the *architecture* of the neural network model.

Fully-connected neural networks are a modular class of nonlinear predictors. They are very flexible in the types of functions they can approximate. In fact, they have been shown [e.g. Hornik, 1991] to be universal function approximators: With an adequate number of hidden layers and neurons therein, they can approximate any continuous function over a compact domain to any desired accuracy.

### 2.1.5 Modern Deep Learning Architectures

Many variants of the basic fully-connected neural network have been developed over time, a few of which we want to mention here without diving into any details. Rather than giving a full understanding of each of these models and techniques, the purpose of this excursion is to highlight the versatility of neural networks.

*Convolutional Neural Networks* Some neural network variants are designed to facilitate the processing of different data modalities. E.g., convolutional neural networks [CNNs, Fukushima and Miyake, 1982, LeCun et al., 1989] were designed to process image data. In images, information is encoded in the patterns formed by a local neighborhood of pixels. MLPs do not by design consider this spatial nature of image data, since they treat each feature as "independent" from the others. Moreover, the number of pixels in a digital image is usually very large, leading to large fully-connected layers, which are computationally expensive and prone to overfitting. A *convolution layer* retains and leverages the spatial nature of image data by replacing the matrix multiplication in a fully-connected layer (Eq. 2.16) with a convolution operation. The "convolution kernel" acts as a (learnable) filter that scans an image for certain patterns and outputs an image-like feature map. This is much more efficient operation and, at the same time, encodes prior structural knowledge about image data, which acts as an inductive bias and facilitates learning. Note that, like a matrix multiplication, a convolution is a linear operation. Convolution layers are usually interspersed with so-called pooling layers, which reduce the spatial dimension of the feature map before passing it to the next layer. Variants of convolutional neural networks constitute the state of the art in computer vision tasks.

*Autoencoders* Other architectures, such as autoencoders, are designed for unsupervised learning. An autoencoder can be seen as the concatentation of two neural nets. An *encoder* $h_\theta(x)$ maps the input $x \in \mathbb{R}^d$ to an encoding $h_\theta(x) \in \mathbb{R}^p$ with $p < d$. A *decoder* maps the encoding $z \in \mathbb{R}^p$ back to the input space, $g_\omega(z) \in \mathbb{R}^d$. The goal is minimize a reconstruction error for an (unlabeled) training dataset, such as

$$\min_{\theta,\omega} \frac{1}{N} \sum_{n=1}^{N} \|x_n - g_\omega(h_\theta(x_n)\|^2. \tag{2.17}$$

The $p$-dimensional encoding serves as a bottleneck; to achieve a low reconstruction error, the autoencoder has to learn a compressed representation of the data, which can be used for dimensionality reduction [Kramer, 1991] or representation learning [Vincent et al., 2010].

*Residual Networks* Residual networks [ResNets He et al., 2016] are a class of neural network architectures achieving state-of-the-art re-

sults in computer vision. They introduce so-called residual (or skip) connections to standard feed-forward architectures. Instead of each layer's output feeding only into the immediately following layer, a residual connection feeds the output of a layer directly into a layer a few steps ahead. While the skip connections do not fundamentally alter the expressiveness of the neural network architecture, it has been shown empirically to aid gradient-based training compared to a comparable (non-residual) architecture. It is, thus, a remarkable example where a model is altered purely to make it more amenable to gradient-based optimization algorithms. Another recent development that is mostly motivated by facilitating gradient-based training—and a key feature of ResNets—are normalization layers, the most popular technique being *Batch Normalization* [Ioffe and Szegedy, 2015].

This list is by no means exhaustive, as the field of deep learning has developed rapidly over the last decade and continues to innovate at a high pace. So-called recurrent neural networks [e.g., LSTMs Hochreiter and Schmidhuber, 1997] are designed to operate on sequences or time series and have driven many advances in natural language processing. Generative adversarial networks [GANs Goodfellow et al., 2014] are combinations of two neural networks trained in a "competing" manner for unsupervised learning of generative models. Not to mention countless small "tricks of the trade", such as new activation functions [e.g. Ramachandran et al., 2017], parameter initialization schemes [e.g. Glorot and Bengio, 2010], or regularization methods such as "dropout" [Srivastava et al., 2014].

Practical deep learning has become an engineering discipline, with many different tools, best practices and rules of thumb. However, in the big picture—and in particular from the perspective of optimization—all variants have a lot in common: They are parametrized functions composed of multiple layers of (learnable) affine maps followed by a nonlinear activation function.

### 2.1.6 *Gradient-Based Learning in Neural Nets: Backpropagation*

Modern neural networks are trained using empirical risk minimization as described in Section 2.1.1. Minimizing the empirical risk with numerical optimization algorithms requires the computation of its gradient with respect to the parameters of the neural network. These computations can be implemented efficiently and the resulting algorithm is called the method of *backpropagation of errors* or simply *backpropagation*. The attribution of the "invention" of backpropagation is disputed, but its popularization for the training of neural networks is due to Rumelhart et al. [1986]. Backpropagation in neural networks is a special case of reverse-mode automatic differentiation, which implements the chain rule of calculus for functions specified by general computational graphs.

Computational graphs are directed acyclic graphs, where each node represents the output of a computation. The inputs to the computation performed by an individual node are given by its parents and the result of the computation is fed as input to the children. In that way, a computational graph can represent complex computations as a composition of basic computational modules. The computation graph also has "constant" nodes, who do not perform computations but simply represent inputs to the computation. In a machine learning context, these constant nodes may represent input data or model parameters.

With a slight abuse of notation, we can state the chain rule of calculus in the following way: Assume we have a composition $z(y(x))$ with $x \in \mathbb{R}^n$, $y\colon \mathbb{R}^n \to \mathbb{R}^m$ and $z\colon \mathbb{R}^m \to \mathbb{R}$. Then the gradient of $z$ w.r.t. $x$ can be written as

$$\nabla_x z = \frac{\partial y}{\partial x}^\mathsf{T} \nabla_y z, \tag{2.18}$$

where $\partial y / \partial x \in \mathbb{R}^{m \times n}$ is the Jacobian matrix of $y$ w.r.t. $x$. That is, we can compute the gradient of $z$ w.r.t. $x$ by multiplying the gradient of $z$ w.r.t. to the output of an intermediate computation $y(x)$ with the Jacobian of $y$ w.r.t. $x$.

This gives us a recipe to compute the gradients of a computation graph's output w.r.t. any (intermediate or input) node of a computation graph. We start with the output node, which trivially has a gradient w.r.t. itself of 1. We then find the parent node and "backpropagate" the gradient by multiplying with the Jacobian of the node. All that is required is for each node, whose forward operation represents a basic computational building block, to be able to multiply with the Jacobian of that operation. This is what lies at the heart of modern automatic differentation frameworks such as TensorFlow [Abadi et al., 2015] or PyTorch [Paszke et al., 2019], which power much of contemporary machine learning.

## 2.2 *Mathematical Optimization*

We briefly recall some fundamental concepts from (deterministic) mathematical optimization, loosely following the textbook of Nesterov [1983]. The treatment of stochastic optimization algorithms is deferred to the next section. Mathematical optimization is concerned with finding extremal values of an objective function $f\colon D \to \mathbb{R}$. We assume the domain $D$ to be a subset of $\mathbb{R}^d$ which puts us in the realm of so-called *continuous* optimization (as opposed to *discrete* optimization, where the domain is partly discrete, e.g., a set of integers). Optimization problems are canonically formulated as minimization problems[3] and written as

$$\min_{\theta \in D} f(\theta). \tag{2.19}$$

[3] Maximization is equivalent to minimization of $-f(\theta)$.

We ideally want to find a *global minimum* of $f$, that is, $\theta_\star \in D$ such that $f(\theta_\star) \leq f(\theta)$ for all $\theta \in D$. We assume that $f$ is lower-bounded

and that the lower bound is attained so that there will be at least one global minimum. We denote the minimal value as $f_\star = f(\theta_\star)$. If $f(\theta_\star) < f(\theta)$ for all $\theta \in D \backslash \{\theta_\star\}$ we call $f(\theta_\star)$ a strict (global) minimum.

Sometimes we are satisfied (or forced to content ourselves) with finding a *local minimum*, i.e., $\theta_\star$ for which there exists a neighborhood such that $f(\theta_\star) \leq f(\theta)$ for all $\theta$ in that neighborhood. We call $\theta_\star$ a strict local minimum if there is a neighborhood such that $f(\theta_\star) < f(\theta)$ for all $\theta \neq \theta_\star$ in that neighborhood.

Optimization problems are commonly classified based on properties of $f$ and $D$. In this work, we are mostly concerned with unconstrained, smooth, non-linear optimization problems.

- In *unconstrained* optimization problems the domain is $D = \mathbb{R}^d$ as opposed to constrained optimization where $D$ is a strict subset of $\mathbb{R}^d$, usually given in the form of multiple functional inequality and equality constraints $g_i(\theta) \leq 0$ and $h_j(\theta) = 0$.
- In *smooth* problems the objective $f$—and, if applicable, all constraint functions—are differentiable. This allows the optimization method to rely on gradients and, possibly, higher-order derivatives. Note that the word "smooth" can have other meanings depending on the context, notably the notion of Lipschitz smoothness of a function, which will be introduced below.
- The objectives we consider are *nonlinear*. There are specialized algorithms for *linear* optimization problems, where $f$ is a linear function and the domain is a convex polytope given by linear equality and inequality constraints.

### 2.2.1 *Optimality Conditions*

The definition of a minimum is descriptive rather than constructive and does not directly inspire methods to find such a point. Differentiable objective functions allow for a characterization of minima in terms of their derivatives.

We start by establishing a necessary condition for optimality in terms of the first-order derivative.

**Theorem 2.1.** *Let $\theta_\star$ be a local minimum of a differentiable function $f\colon \mathbb{R}^d \to \mathbb{R}$. Then*

$$\nabla f(\theta_\star) = 0. \tag{2.20}$$

*Proof.* See, e.g., Theorem 1.2.1 in Nesterov [2018]. □

We call a point with zero gradient a *stationary point* of $f$. Theorem 2.1 says that every local minimum is also a stationary point or, in other words, that stationarity is a necessary condition for a miminum. The intuitive reason is that, if there were a nonzero gradient, taking a small step into the direction of the negative gradient would lead to a further decrease in function value, contradicting the presence of a minimum. Note that not every stationary point is a

local minimum; stationary points can also be maxima or so-called saddle points.

For a twice differentiable function, another necessary optimality condition is that its second derivative (its curvature) is "non-negative". In the case of a multivariate objective, this means that its Hessian matrix is positive semi-definite[4]:

[4] A symmetric matrix $A \in \mathbb{R}^{d\times d}$ is positive semi-definite if $x^\mathsf{T} Ax \geq 0$ for all $x \in \mathbb{R}^d$. It is called positive definite if $x^\mathsf{T} Ax > 0$ for all nonzero $x \in \mathbb{R}^d$. We denote these properties as $A \succeq 0$ and $A \succ 0$, respectively.

**Theorem 2.2.** *Let $\theta_\star$ be a local minimum of a twice differentiable function $f\colon \mathbb{R}^d \to \mathbb{R}$. Then*

$$\nabla f(\theta_\star) = 0 \quad \textit{and} \quad \nabla^2 f(\theta_\star) \succeq 0. \tag{2.21}$$

*Proof.* See, e.g., Theorem 1.2.2 in Nesterov [2018]. □

Again, this is not a sufficient condition; such a point could also be a saddle point, see for example $f(\theta) = \theta^3$ at $\theta = 0$.

Let us, finally, discuss a sufficient condition.

**Theorem 2.3.** *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be twice differentiable. If $\theta_\star \in \mathbb{R}^d$ satisfies*

$$\nabla f(\theta_\star) \quad \textit{and} \quad \nabla^2 f(\theta_\star) \succ 0, \tag{2.22}$$

*then $\theta_\star$ is a strict local minimum of $f$.*

*Proof.* See, e.g., Theorem 1.2.3 in Nesterov [2018]. □

A stationary point with strictly positive curvature is a strict local minimum. Note that this condition is slightly stronger than necessary; an example is, again, $f(\theta) = \theta^4$, which has a strict local minimum at $\theta = 0$ despite having zero curvature at that point. Nevertheless, this criterion will be convenient to work with.

### 2.2.2 *Regularity Assumptions*

So far, we have only assumed the objective function to be differentiable and to have at least one local minimum. This encompasses a great many objective functions and it seems like a monstrous task to find optimization algorithms that work for *all* of them. To find efficient algorithms, we will have to make additional assumptions that the objective is, in some sense, well-behaved. These are often called regularity assumptions. We will discuss two of them here, Lipschitz smoothness and convexity, which are of paramount importance in mathematical optimization.

*Lipschitz Smoothness* One of the key assumption in numerical optimization is smoothness, which in this context means that the gradient function is Lipschitz.

**Definition 2.1** (Smoothness)**.** *A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is $L$-smooth if*

$$\|\nabla f(\theta') - \nabla f(\theta)\|_2 \leq L\|\theta' - \theta\|_2 \tag{2.23}$$

*for all $\theta, \theta' \in \mathbb{R}^d$.*

Smoothness controls how fast the gradient function changes as we move in its input space. It is commonly formulated with respect to the Euclidean norm, as we just did, but can equivalently be formulated for other norms. This will play an important role in Chapter 6 and will be discussed there. A crucial consequence of smoothness is that the function can be bounded by quadratic functions:

**Lemma 2.1.** *If $f\colon \mathbb{R}^d \to \mathbb{R}$ is L-smooth, then*

$$f(\theta') \leq f(\theta) + \nabla f(\theta)^\mathsf{T}(\theta' - \theta) + \frac{L}{2}\|\theta' - \theta\|_2^2 \tag{2.24}$$

*and*

$$f(\theta') \geq f(\theta) + \nabla f(\theta)^\mathsf{T}(\theta' - \theta) - \frac{L}{2}\|\theta' - \theta\|_2^2 \tag{2.25}$$

*for all $\theta, \theta' \in \mathbb{R}^d$.*

*Proof.* See, e.g., Lemma 1.2.3 in Nesterov [2018]. □

Bounds of this form are crucial for first-order optimization. In fact, gradient descent—the most basic first-order optimization method, which we will discuss shortly—can be seen as iteratively minimizing the upper bound, centered around the previous iterate:

$$\begin{aligned}\theta_{t+1} &= \arg\min_{\theta'} \left( f(\theta_t) + \nabla f(\theta_t)^\mathsf{T}(\theta' - \theta_t) + \frac{L}{2}\|\theta' - \theta_t\|_2^2 \right) \\ &= \theta_t - \frac{1}{L}\nabla f(\theta_t).\end{aligned} \tag{2.26}$$

For twice differentiable functions, smoothness arises from a bound on the Hessian matrix.

**Lemma 2.2.** *A twice differentiable function $f\colon \mathbb{R}^d \to \mathbb{R}$ is L-smooth if and only if*

$$\|\nabla^2 f(\theta)\|_2 \leq L. \tag{2.27}$$

*Proof.* See, e.g., Lemma 1.2.2 in Nesterov [2018]. □

Here, $\|A\|_2$ for a matrix $A$ refers to its spectral norm which—for symmetric matrices like Hessians—is given by the largest absolute eigenvalue. We can therefore also write this bound as $-L \cdot I_d \preceq \nabla^2 f(x) \preceq L \cdot I_d$, where $A \preceq B$ means that $B - A$ is positive semi-definite.

*Convexity* Another important regularity assumption is *convexity*. We call a set $D \subset \mathbb{R}^d$ convex if for any $\theta, \theta' \in D$ and $\lambda \in [0,1]$, $\lambda\theta + (1-\lambda)\theta'$ is also in $D$. Pictorially, for two points in a convex set, the set also contains all points on the straight line between the two. Examples for convex sets are $D \equiv \mathbb{R}^d$ or balls $B_\varepsilon(\theta_0) = \{\theta \in \mathbb{R}^d \mid \|\theta - \theta_0\|_2 \leq \varepsilon\}$ for $\varepsilon > 0$.

Next, we define a convex function:

**Definition 2.2** (Convex function)**.** *Let $D \subset \mathbb{R}^d$ be a convex set. A function $f\colon D \to \mathbb{R}$ is convex if, for any $\theta, \theta' \in D$ and $\lambda \in [0,1]$,*

$$f(\lambda\theta + (1-\lambda)\theta') \leq \lambda f(\theta) + (1-\lambda) f(\theta'). \tag{2.28}$$

*The function is strictly convex if the inequality holds strictly for any* $\lambda \in (0, 1)$.

Pictorially, the line interpolating between two points on the graph of a convex function lies above the graph.

For differentiable functions there is a useful equivalent characterization of convexity:

**Lemma 2.3.** *Let* $D \subset \mathbb{R}^d$ *be a convex set and* $f \colon D \to \mathbb{R}$ *be continuously differentiable. Then* $f$ *is convex if and only if, for any* $\theta, \theta' \in \mathbb{R}^d$,

$$f(\theta') \geq f(\theta) + \nabla f(\theta)^\mathsf{T}(\theta' - \theta). \tag{2.29}$$

*Strict convexity holds if the inequality is strict.*

*Proof.* See, e.g., Theorem 2.1.2 in Nesterov [2018]. □

The latter condition has the pictorial interpretation that the tangent which touches the graph of $f$ at $\theta$ lies below that graph. An immediate consequence of this characterization—which is key to optimization—is that any stationary point of a convex function $f$ is a global minimum: If $\nabla f(\theta_\star) = 0$, then Eq. (2.29) implies $f(\theta') \geq f(\theta_\star)$ for all $\theta'$. This also means that any local minimum of a convex function is also a global minimum.

Intuitively, convexity means that the function "bends upwards" everywhere, which relates to the curvature of the function. For a twice continuously differentiable function, this translates directly into a requirement on the Hessian of the function:

**Lemma 2.4.** *Let* $D \subset \mathbb{R}^d$ *be an open convex set and* $f \colon D \to \mathbb{R}$ *be twice continuously differentiable. Then* $f$ *is convex if and only if*

$$\nabla^2 f(\theta) \succeq 0 \quad \forall \theta \in D, \tag{2.30}$$

*i.e., the Hessian is positive semi-definite on* $D$*. Strict convexity holds if the Hessian is positive definite.*

*Proof.* See, e.g., Theorem 2.1.4 in Nesterov [2018]. □

*Strong Convexity* There is an even stronger version of convexity, aptly named *strong convexity*, which plays an important role in optimization.

**Definition 2.3** (Strongly convex function)**.** *Let* $D \subset \mathbb{R}^d$ *be a convex set. A continuously differentiable function* $f \colon D \to \mathbb{R}$ *is called* $\mu$*-strongly convex for* $\mu > 0$ *if, for any* $\theta, \theta'$,

$$f(\theta') \geq f(\theta) + \nabla f(\theta)^\mathsf{T}(\theta' - \theta) + \frac{\mu}{2}\|\theta' - \theta\|_2^2. \tag{2.31}$$

Convexity, in the form of Eq. (2.29) lower-bounded the function with the tangent touching $f$ at $\theta$, that is, with a linear function. Strong convexity is a stronger requirement in that it demands $f$ to be lower-bounded by a *quadratic* function touching $f$ at $\theta$.

As one might expect, we can formulate strong convexity in terms of the Hessian of a twice differentiable function:

**Lemma 2.5.** *Let $D \subset \mathbb{R}^d$ be an open convex set and $f\colon D \to \mathbb{R}$ be twice continuously differentiable. Then $f$ is $\mu$-strongly convex if and only if*

$$\nabla^2 f(\theta) \succeq \mu I_d \quad \forall \theta \in D, \tag{2.32}$$

*i.e., $\nabla^2 f - \mu I_d$ is positive semi-definite on $D$.*

*Proof.* See, e.g., Theorem 2.1.11 in Nesterov [2018]. □

Of course, strong convexity with any $\mu > 0$ implies strict convexity. The reverse is not true. In the second-order formulation of Lemmata 2.4 and 2.5, respectively, strict convexity requires a strictly positive definite Hessian everywhere, whereas strong convexity requires the Hessian to be "bounded away" from zero. As an example, $f(\theta) = e^\theta$ is strictly convex but not strongly convex, since its second derivative $f''(\theta)$ approaches zero for $\theta \to -\infty$. Pictorially, the exponential function gets flatter and flatter for negative input values and, thus, can not be lower-bounded by quadratic functions with a uniform curvature of $\mu > 0$.

An important property of strongly-convex function is given by the following Lemma.

**Lemma 2.6.** *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be $\mu$-strongly convex. Then*

$$\|\nabla f(\theta)\|_2^2 \geq 2\mu(f(\theta) - f_\star). \tag{2.33}$$

*Proof.* Special case of Theorem 2.1.10 in Nesterov [2018]. □

Note that we defined strong convexity with respect to the Euclidean norm in Definition 2.3. It can also be formulated with respect to other norms, which may change the strong convexity parameter $\mu$. This will become important in Chapter 6 and will be discussed there.

*Smooth and Strongly Convex Functions* As a consequence of Lemmata 2.5 and 2.2, a twice differentiable function is $L$-smooth and $\mu$-strongly convex if and only if

$$\mu I_d \preceq \nabla^2 f(\theta) \preceq L I_d \quad \forall \theta \in D. \tag{2.34}$$

Functions of this class can be lower- and upper-bounded by quadratic functions and, thus, behave "almost" like quadratic functions. As we will see, this makes them particularly well-behaved from an optimization perspective and, therefore, an important class to study.

### 2.2.3 Convergence of Optimization Algorithms

Practical optimization algorithms for general continuous optimization are *approximate* and *iterative* in nature. They are *approximate* in that they do not attempt to find the *exact* minimizer $\theta_\star$. Instead, they start from a (user-provided) starting point $\theta_0 \in \mathbb{R}^d$ and *iteratively* produce a sequence of iterates, $\theta_0, \theta_1, \theta_2, \ldots$ which converges to a minimum. The pure fact that an algorithm will converge *eventually* is reassuring but we are usually also interested in its *speed*. Before proceeding to the discussion of specific algorithms, we briefly review common notions of convergence speed.

*Convergence Bounds* Convergence bounds are always formulated for classes of functions, i.e., the set of functions satisfying certain assumptions. In a setting where convergence to a global minimum can be guaranteed, e.g., for convex objectives, we are usually interested in bounds of the form[5]

$$f_t - f_\star \leq h(t) \quad \text{or} \quad \|\theta_t - \theta_\star\| \leq h(t). \tag{2.35}$$

[5] Here, and throughout the remainder of this thesis, we denote $f_t = f(\theta_t)$, $\nabla f_t = \nabla f(\theta_t)$, et cetera.

That is, we bound the so-called *suboptimality* $(f_t - f_\star)$ or the distance to the minimizer $(\|\theta_t - \theta_\star\|)$ as a function of the number of iterations of the algorithm. Both notions of convergence are related and we know that

$$f_t - f_\star \to 0 \Leftrightarrow \|\theta_t - \theta_\star\| \to 0 \tag{2.36}$$

for sufficiently regular functions, but there can be a nontrivial relationship between the respective right-hand sides of the bounds.

A few remarks on such notions of convergence speed are in order:

- The guarantees take the form of upper bounds. We usually desire these bounds to be tight, meaning that it holds with equality for some worst-case situation, e.g., for a specific function from the class of functions under consideration.
- The bounds are formulated as a function of the number of iterations. Of course, this is merely a proxy for other quantities, which would be more informative but harder to measure and compare impartially. Eventually, we are interested in which algorithm is faster in actual wall-clock time but this depends, among other things, on the programming language used to implement it and the hardware that the resulting program is run on. Alternatively, we might want to quantify convergence speed as a function of the computational cost, but this is not a perfect measure either. Some algorithms have high computational cost, but lend themselves to easy parallelization, which makes them fast in wall-clock time. Others have low computational complexity, but require large amounts of additional memory. Formulating convergence bounds by number of iterations makes for easy comparability and relegates the considerations hinted at above to a later point in time. Bounds of this form usually make the assumption that $f$ is a black box, and that the algorithm requires a single evaluation (e.g., of the gradient and/or the Hessian) per iteration.
- Instead of giving an exact expression for the right-hand side $h(t)$, we often use "big $O$ notation" to specify the asymptotic *order of convergence*. That is, we write $f_t - f_\star \in O(h(t))$, meaning that there exists $M > 0, t_0 \in \mathbb{N}$ such that $f_t - f_\leq Mh(t)$ for all $t \geq t_0$. This hides all constants and lower-order terms, but is often easier to parse and simplifies proofs. However, it is important to understand that these constants and lower-order terms can very well matter in practice!

- Convergence bounds are sometimes equivalently expressed as the number of iterations needed to reach an "accuracy" of $\varepsilon > 0$, e.g., an algorithm needs at most $T(\varepsilon)$ iterations to achieve a suboptimality of $\varepsilon$.

*Example* As an example, we will shortly see that, for $L$-smooth and $\mu$-strongly convex functions, the gradient descent algorithm satisfies

$$f_t - f_\star \leq \left(1 - \frac{\mu}{L}\right)^t (f_0 - f_\star). \tag{2.37}$$

Such convergence of order $O(c^t)$ with $c < 1$ is called *linear convergence*. It implies that at most $O(\log \varepsilon)$ iterations are needed to guarantee $f_t - f_\star \leq \varepsilon$.

*Weaker Convergence Statements* In settings where convergence to a global minimum may not be guaranteed, similar bounds may be proven for convergence to a local minimum or to a stationary point. The latter can take the form of a bound

$$\|\nabla f_t\| \leq h(t). \tag{2.38}$$

### 2.2.4 *Gradient Descent*

We will now discuss the basic first-order optimization algorithm, gradient descent. It performs iterations of the form

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f_t, \tag{2.39}$$

where $\alpha_t > 0$ is a sequence of step sizes. The underlying idea is that the negative gradient points in the direction of steepest descent. Assume we are at $\theta \in \mathbb{R}^d$ and take a step of size $\varepsilon > 0$ in a (normalized) direction $\delta \in \mathbb{R}^d, \|\delta\|_2 = 1$. The directional derivative of $f$ along $\delta$, that is, the rate of change as we move infinitesimally along direction $\delta$, is given by $\lim_{\varepsilon \to 0} \varepsilon^{-1}(f(\theta + \varepsilon\delta) - f(\theta))$ and can be shown to evaluate to $\nabla f(\theta)^\mathsf{T} \delta$. The direction of steepest descent is defined as the direction with smallest (most negative) directional derivative

$$\min_{\delta \in \mathbb{R}^d} \nabla f(\theta)^\mathsf{T} \delta \text{ s.t. } \|\delta\|_2^2 = 1, \tag{2.40}$$

and is given by $\delta \propto -\nabla f(\theta)$. With a suitably chosen step size, gradient descent decreases the function value at each iteration.

Gradient descent can be traced back to Cauchy [1847] who applied the method to astronomical calculations.

*Choosing Step Sizes* Choosing a step size sequence $\alpha_t > 0$ in Eq. (2.39) is not a trivial task. As we will see below, gradient descent will diverge for step sizes that are too large. On the other hand, choosing excessively small step sizes will lead to painfully slow convergence. In practice, one of two options is typically used: Either the step size is set to a constant, chosen by trial and error or based on prior knowledge about the problem at hand. Or the

step size is set automatically at each iteration by a subroutine called a *line search*. Line searches attempt to solve the one-dimensional optimization problem

$$\min_{\alpha>0} f(\theta_t - \alpha \nabla f_t). \tag{2.41}$$

Practical line search routines search for approximate solutions that are "good enough" to guarantee fast convergence of the algorithm. Line searches increase the computational cost, since they require additional evaluations of function value and/or the gradient in order to test multiple step sizes before finding a satisfactory choice. But they relieve us of the burden of having to choose a step size manually.

*Convergence Results* For illustration, we will now prove two convergence results for gradient descent. For the first result, the only assumption we make is that $f$ is $L$-smooth.

**Theorem 2.4.** *Let $f$ be $L$-smooth and bounded from below. Then gradient descent with a constant step size $\alpha < 2/L$ satisfies*

$$\sum_{t=0}^{T} \|\nabla f_t\|_2^2 \leq C \tag{2.42}$$

*for some constant $C > 0$.*

This bound on the sum of squared gradients implies that $\|\nabla f_t\| \to 0$, that is, gradient descent converges to a stationary point.

*Proof.* By smoothness and the resulting bound in Lemma 2.1 we get

$$f_{t+1} \leq f_t - \alpha \|\nabla f_t\|_2^2 + \frac{L\alpha^2}{2} \|\nabla f_t\|_2^2 \tag{2.43}$$

or, after rearranging,

$$\|\nabla f_t\|_2^2 \leq \left(\alpha - \frac{L\alpha^2}{2}\right)^{-1} (f_t - f_{t+1}), \tag{2.44}$$

where we used the fact that $\alpha - L\alpha^2/2$ is positive due to our assumption that $\alpha < 2/L$. Summing this inequality from $t = 0$ to $t = T$ yields

$$\begin{aligned} \sum_{t=0}^{T} \|\nabla f_t\|_2^2 &\leq \left(\alpha - \frac{L\alpha^2}{2}\right)^{-1} \sum_{t=0}^{T} (f_t - f_{t+1}) \\ &= \left(\alpha - \frac{L\alpha^2}{2}\right)^{-1} (f_0 - f_{T+1}) \\ &\leq \left(\alpha - \frac{L\alpha^2}{2}\right)^{-1} (f_0 - f_\star). \end{aligned} \tag{2.45}$$

The last step is simply by definition of the lower bound $f_\star$. □

For smooth and strongly convex functions gradient descent achieves linear convergence in suboptimality.

**Theorem 2.5.** *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be $L$-smooth and $\mu$-strongly convex. Then gradient descent with step size $\alpha = 1/L$ achieves*

$$f_t - f_\star \leq \left(1 - \frac{\mu}{L}\right)^t (f_0 - f_\star). \tag{2.46}$$

*Proof.* By smoothness and the resulting bound in Lemma 2.1 we get

$$\begin{aligned} f_t &\leq f_{t-1} - \alpha \|\nabla f_{t-1}\|_2^2 + \frac{L\alpha^2}{2}\|\nabla f_{t-1}\|_2^2 \\ &= f_{t-1} - \frac{1}{2L}\|\nabla f_{t-1}\|_2^2. \end{aligned} \tag{2.47}$$

Subtracting $f_\star$ from both sides and using Lemma 2.6 yields

$$f_t - f_\star \leq \left(1 - \frac{\mu}{L}\right)(f_{t-1} - f_\star). \tag{2.48}$$

Applying this inequality recursively implies the desired result. □

The quantity $L/\mu$ is called the *condition number* of the problem. If this number is very large, then the constant in the linear convergence of gradient descent is very close to 1 and the algorithm can take a long time to converge. An pictorial example can be given with a two-dimensional quadratic function $f\colon \mathbb{R}^2 \to \mathbb{R}$,

$$f(\theta) = \frac{1}{2}\theta^\mathsf{T} H\theta, \quad \mathbb{R}^{2\times 2} \ni H \succ 0, \tag{2.49}$$

whose contour lines can be depicted, see Figure 2.2. By Lemmata 2.2 and 2.5, the smoothness constant and the strong convexity constant will be given by the largest and the smallest eigenvalue of $H$, respectively. If the condition number is small (close to 1), the contour lines will be close to circular and the negative gradient will point towards the minimum; gradient descent will converge quickly. If the condition number is large, the contour lines are elongated and and the gradient can be a bad indicator of the location of the minimum. Gradient descent exhibits very slow convergence.

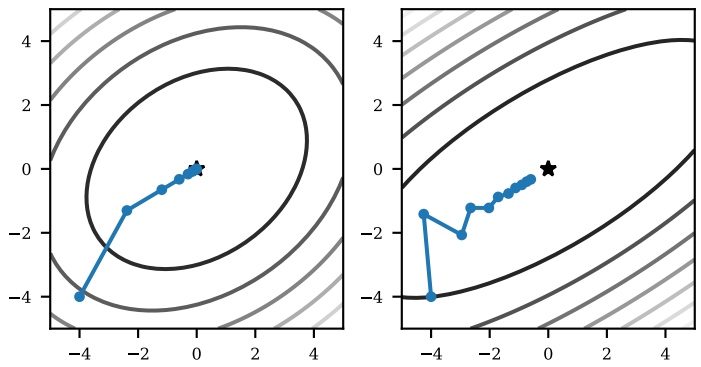


Figure 2.2: The convergence speed of gradient descent depends on the condition number of the problem. The figure depicts the contour lines of two-dimensional quadratic objectives with condition numbers of $L/\mu = 2$ (left) and $L/\mu = 8$ (right) and the trajectory of 10 gradient descent steps.

### 2.2.5 *Newton's Method*

Gradient descent is a so-called first-order method, since it only utilizes first-order derivatives of the objective function. Gradient

descent can be seen as iteratively minimizing a local linear model of its objective function plus a penalty term that keeps the new iterate close to the current one. The first-order Taylor expansion of $f$ around a point $\bar{\theta}$ is

$$f^{\text{lin}}_{\bar{\theta}}(\theta) \stackrel{\text{def}}{=} f(\bar{\theta}) + \nabla f(\bar{\theta})^{\mathsf{T}}(\theta - \bar{\theta}), \tag{2.50}$$

and a gradient descent update can be defined as

$$\theta_{t+1} = \arg\min_{\theta \in \mathbb{R}^d} \left( f^{\text{lin}}_{\theta_t}(\theta) + \frac{1}{2\alpha_t} \|\theta - \theta_t\|_2^2 \right). \tag{2.51}$$

In contrast to that, second-order methods utilize second-order derivatives of the objective to minimize a local *quadratic* model. The prototypical second-order method is *Newton's method*, which goes back to Sir Isaac Newton, who described a particular instance of the method in his 1711 work *De analysi per aequationes numero terminorum infinitas*.

Newton's method minimizes the second-order Taylor expansion of its objective,

$$f^{\text{quad}}_{\bar{\theta}}(\theta) \stackrel{\text{def}}{=} f(\bar{\theta}) + \nabla f(\bar{\theta})^{\mathsf{T}}(\theta - \bar{\theta}) + \frac{1}{2}(\theta - \bar{\theta})^{\mathsf{T}} \nabla^2 f(\theta)(\theta - \bar{\theta}), \tag{2.52}$$

which leads to updates of the form

$$\theta_{t+1} = \arg\min_{\theta \in \mathbb{R}^d} f^{\text{quad}}_{\theta_t}(\theta) = \theta_t - \nabla^2 f(\theta_t)^{-1} \nabla f(\theta_t). \tag{2.53}$$

We assume $f$ to be strictly convex, which makes the Hessian positive definite and, thus, invertible.

Incorporating second-order (curvature) information makes Newton's method a potentially much more powerful method than gradient descent. Taking the local geometry of the obective into account counteracts gradient descent's zig-zagging behavior described above. Newton's method finds the solution of a quadratic optimization problem in a single step, whereas gradient descent can take a very long time to converge on poorly-conditioned quadratic problems as shown in Figure 2.2.

Of course, these advantages come with some drawbacks. First, we had to make the additional assumption of a twice differentiable and convex objective. In a non-convex setting, the Hessian might be negative definite, which makes the local quadratic model unbounded from below and renders the Newton update meaningless. Even if these assumptions are fulfilled, we need computational access to the Hessian, which may be cumbersome from an implementation point of view and might cause considerable computational cost. These computational aspects have inspired a multitude of approximate Newton methods, which employ various approximations to the Hessian matrix, e.g., by computing only its diagonal. We will revisit one such approximation, the so-called generalized Gauss-Newton matrix in Chapter 7.

# 3
# Stochastic Optimization

We now turn to the stochastic optimization regime, which we already briefly introduced in Chapter 1. We give a general, formal problem statement and establish notation for the remainder of this manuscript. We then discuss stochastic gradient descent, the prototypical stochastic optimization algorithm.

## 3.1 Problem Statement

It will be convenient to adopt a more abstract formulation of stochastic optimization than the one given in Chapter 1. In a stochastic optimization problem, we want to minimize an objective of the form

$$f(\theta) = \mathbf{E}_{\xi}[f(\theta; \xi)] \tag{3.1}$$

but only have access to samples from $p(\xi)$, while the distribution itself is unknown.

An element $\xi$ can be considered an abstract notion of a data point. Eq. (3.1) includes the empirical risk minimization problem from Eq. (1.1) if we set $\xi = (x, y)$, define the function $f(\theta; \xi) := \ell(h_\theta(x), y)$, and let $p(\xi)$ be the empirical measure of the training set. But Eq. (3.1) also includes the "infinite data" regime as in the true risk (Eq. 2.1). While not stated explicitly, the distribution $p(\xi)$ could also change with $\theta$, which incorporates objectives that include (intractable) expectations over distributions defined by the model itself, such as in the training of variational autoencoders [Kingma and Welling, 2014].

By definition of the objective Eq. (3.1)—and the linearity of the expectation and the gradient—we know that

$$\mathbf{E}[\nabla_\theta f(\theta, \xi)] = \nabla_\theta \mathbf{E}[f(\theta, \xi)] = \nabla f(\theta). \tag{3.2}$$

We can obtain an unbiased estimate of $\nabla f(\theta)$ by sampling $\xi \sim p(\xi)$ and using $\nabla_\theta f(\theta, \xi)$. From now on, we will drop the subscript and write $\nabla f(\theta, \xi)$ instead of $\nabla_\theta f(\theta, \xi)$, since gradients will always be taken with respect to the parameters $\theta$.

The stochastic properties of the gradient estimate are encoded in

its covariance matrix

$$\begin{aligned}\Sigma(\theta) &\overset{\text{def}}{=} \mathbf{Cov}[\nabla f(\theta;\xi)] \\ &= \mathbf{E}\left[(\nabla f(\theta;\xi) - \nabla f(\theta))(\nabla f(\theta;\xi) - \nabla f(\theta))^\mathsf{T}\right].\end{aligned} \tag{3.3}$$

This covariance matrix will be a key object of interest in the remaineder of this thesis.

Instead of using a single sample $\xi$, we can obtain multiple iid samples and compute the average gradient over this so-called *minibatch*.

$$g \overset{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} \nabla f(\theta;\xi_i), \quad \xi_1,\dots,\xi_m \overset{\text{iid}}{\sim} p(\xi). \tag{3.4}$$

The (co-)variance of the minibatch gradient scales inversely proportional with the batch size,[1]

$$\mathbf{Cov}[g] = \frac{\Sigma(\theta)}{m}. \tag{3.5}$$

We can therefore decrease the stochastic error of our gradient estimate by using a larger minibatch. Of course, this requires sampling $m$ data points and computing $m$ individual gradients $\nabla f(\theta, \xi_i)$. The trade-offs involved in minibatching will be discussed in detail in Chapter 4. For now, we simply assume $m$ to be a given constant.

[1] For any set of random variables, $X_1, \dots, X_m$, the bilinearity of the covariance yields

$$\mathbf{Cov}[\tfrac{1}{m}\textstyle\sum_{i=1}^m X_i] = \sum_{i,j} \frac{1}{m^2}\mathbf{Cov}(X_i, X_j).$$

If the $X_i$ are iid with $\mathbf{Cov}[X_i] = \Sigma$, then $\mathbf{Cov}(X_i, X_j) = \delta_{ij}\Sigma$ and

$$\mathbf{Cov}[\tfrac{1}{m}\textstyle\sum_{i=1}^m X_i] = \frac{\Sigma}{m}.$$

## 3.2 *Stochastic Gradient Descent*

The prototypical stochastic optimziation algorithm is stochastic gradient descent, which dates back to Robbins and Monro [1951], who proposed a "Stochastic Approximation Method" for finding the root of a function given access to noisy measurements of its function value. Kiefer et al. [1952] first applied it specifically to gradient-based optimization.

At each iteration, stochastic gradient descent computes a (minibatch) stochastic gradient,

$$g_t \overset{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} \nabla f(\theta_t;\xi_i^{(t)}), \quad \xi_1^{(t)},\dots,\xi_m^{(t)} \overset{\text{iid}}{\sim} p(\xi), \tag{3.6}$$

and updates

$$\theta_{t+1} = \theta_t - \alpha_t g_t. \quad \text{(SGD)} \tag{3.7}$$

We will now proceed to a convergence analysis of stochastic gradient descent as a means of introducing some basic tools used in later chapters. We loosely follow Bottou et al. [2018].

### 3.2.1 *Conditional and Total Expectation*

The iterates $(\theta_t)_{t\in\mathbb{N}}$ generated by SGD—or any other stochastic optimization algorithm—are a stochastic process, since randomness is injected at every iteration via the random sampling of data points. The same hold for associated sequences like $f(\theta_t)$, $\nabla f(\theta_t)$, or $g_t$. We need to distinguish between a conditional expectation given

$\theta_t$, denoted by $\mathbf{E}_t[\cdot]$ and the "total" expectation, denoted by $\mathbf{E}[\cdot]$.[2] Unfortunately, this details is often glossed over in the machine learning literature. For example, we know that $\mathbf{E}_t[g_t] = \nabla f(\theta_t)$. But a statement like $\mathbf{E}[g_t] = \nabla f(\theta_t)$—often found in the literature—is nonsensical, since the left-hand side is a deterministic quantity whereas the right-hand side is a random variable.

[2] For readers familiar with measure theory, $\mathbf{E}_t$ is defined as the conditional expectation with respect to the $\sigma$-field

$$\mathcal{F}_t \stackrel{\text{def}}{=} \sigma(\theta_1, \ldots, \theta_t) = \sigma(g_0, \ldots, g_{t-1}). \tag{3.8}$$

### 3.2.2 *A Descent Lemma*

Most convergence results for SGD start from the following lemma, which shows that an SGD update achieves an expected decrease in function value for a sufficiently small step size.

**Lemma 3.1.** *Let $f$ be $L$-smooth. The iterates generated by* SGD *(Eq. 3.7) satisfy*

$$\begin{aligned}\mathbf{E}_t[f(\theta_{t+1})] &\leq f_t - \alpha_t \|\nabla f_t\|_2^2 + \frac{L\alpha_t^2}{2}\mathbf{E}_t[\|g_t\|^2] \\ &= f_t - \alpha_t \|\nabla f_t\|_2^2 + \frac{L\alpha_t^2}{2}\left(\|\nabla f_t\|^2 + \frac{\operatorname{tr}(\Sigma(\theta_t))}{m}\right).\end{aligned} \tag{3.9}$$

If the variance is bounded, we can always choose $\alpha_t$ small enough to decrease the function value in expectation.

*Proof.* Using the smoothness-based upper bound of Lemma 2.1, we have

$$\mathbf{E}_t[f(\theta_{t+1})] \leq \mathbf{E}_t\left[f(\theta_t) + \nabla f(\theta_t)^\mathsf{T}(\theta_{t+1} - \theta_t) + \frac{L}{2}\|\theta_{t+1} - \theta_t\|_2^2\right]. \tag{3.10}$$

Plugging in $\theta_{t+1} - \theta_t = -\alpha_t g_t$ and using the linearity of the expectation (note that $f(\theta_t), \nabla f(\theta_t) \in \mathcal{F}_t$), this yields

$$\mathbf{E}_t[f(\theta_{t+1})] \leq f(\theta_t) - \alpha_t \nabla f(\theta_t)^\mathsf{T}\mathbf{E}_t[g_t] + \frac{L\alpha_t^2}{2}\mathbf{E}_t[\|g_t\|_2^2]. \tag{3.11}$$

Since, $\mathbf{E}_t[g_t] = \nabla f(\theta_t)$, this proves the first inequality.

The second inequality results from the fact that, for any random variable $X \in \mathbb{R}^d$ with mean $\mu$ and covariance $\Sigma$, we have

$$\operatorname{tr}(\Sigma) = \sum_{i=1}^{d}\mathbf{E}[(X_i - \mu_i)^2] = \mathbf{E}\left[\sum_{i=1}^{d}(X_i - \mu_i)^2\right] = \mathbf{E}[\|X - \mu\|_2^2] \tag{3.12}$$

and

$$\mathbf{E}[\|X - \mu\|_2^2] = \mathbf{E}[\|X\|_2^2 - 2X^\mathsf{T}\mu + \|\mu\|_2^2] = \mathbf{E}[\|X\|_2^2] - \|\mu\|_2^2, \tag{3.13}$$

and therefore $\mathbf{E}[\|X\|_2^2] = \|\mu\|_2^2 + \operatorname{tr}(\Sigma)$. Note that $\mathbf{Cov}_t[g_t] = \Sigma(\theta_t)/m$. □

The descent lemma shows that the performance of SGD depends on the gradient covariance matrix. Therefore, we have to make additional assumptions on the stochastic properties of the gradient estimates $g_t$. Different assumptions can be found in the stochastic optimization literature. Some of them are overly simplifying

and unrealistic, such as bounding the covariance by a constant, $\mathrm{tr}(\Sigma(\theta)) \leq C$, or even bounding the stochastic gradient norm itself ($\|g_t\| \leq C$ for all $t$). These assumptions are broken even in simple toy examples, e.g., when $f(\theta, \xi) = \|\theta - \xi\|^2$. A more realistic assumption that has been commonly used, and is adequate for our purposes, is the following.

**Assumption 3.1.** *There are constants $C_0, C_1 \geq 0$ such that*

$$\mathrm{tr}(\Sigma(\theta)) \leq C_0 + C_1 \|\nabla f(\theta)\|_2^2, \tag{3.14}$$

*for all $\theta \in \mathbb{R}^d$.*

This allows the covariance to be non-zero everywhere (including at stationary points) and to grow quadratically in the gradient norm. Plugging Assumption 3.1 into the descent lemma reads:

**Lemma 3.2.** *Let $f$ be $L$-smooth and assume the variance bound in Assumption 3.1 holds. Then the iterates generated by* SGD *satisfy*

$$\mathbf{E}_t[f_{t+1}] \leq f_t - \left(\alpha_t - \left(1 + \frac{C_1}{m}\right)\frac{L\alpha_t^2}{2}\right)\|\nabla f_t\|_2^2 + \frac{L\alpha_t^2 C_0}{2m} \tag{3.15}$$

*Proof.* Use Assumption 3.1 in Lemma 3.1. □

### 3.2.3 *Convergence for Strongly-Convex Objectives*

We give a simple covergence result for SGD on strongly convex objectives. It is particularly illustrative to first consider the behavior of SGD with a constant step size.

**Theorem 3.1.** *Let $f$ be $L$-smooth and $\mu$-strongly convex and assume the gradient distribution satisfies Assumption 3.1. Then* SGD *with a constant step size*

$$\alpha_t \equiv \alpha \leq \frac{1}{L(1 + C_1/m)} \tag{3.16}$$

*satisfies*

$$\mathbf{E}[f_t - f_\star] \leq \frac{\alpha L C_0}{2m\mu} + (1 - \alpha\mu)^t \left(f_0 - f_\star + \frac{\alpha L C_0}{2m\mu}\right). \tag{3.17}$$

Before we give the proof, let's understand the implications of this result. First, note how it incorporates the result for noise-free gradient descent: If the gradient estimate is exact, we will have $C_0 = C_1 = 0$ and recover the linear convergence from Theorem 2.5. In the stochastic regime, constant step size SGD will *not* converge to the optimum While the second term on the right-hand side of Eq. (3.17) still decreases linearly, the first term defines an irreducible "plateau level",

$$\mathbf{E}[f_t - f_\star] \to \frac{\alpha L C_0}{2m\mu}, \tag{3.18}$$

which depends on the step size $\alpha$, the batch size $m$, the geometry of the problem (through $L$ and $\mu$) as well as the noise level at the minimum $C_0$. This can be seen as linear convergence to a neighborhood

of the optimum, defined by a plateau level of the suboptimality. Note the effect of the step size: Choosing a smaller step size leads to a lower plateau level but, at the same time, reduces the linear convergence speed in the second term. A larger batch size enables a larger step size (Eq. 3.16) and reduces the plateau level, at the expense of a higher per-iteration cost.

*Proof of Theorem 3.1.* Using the bound on the step size in Lemma 3.2 yields

$$\mathbf{E}_t[f_{t+1}] \leq f_t - \frac{\alpha}{2}\|\nabla f_t\|_2^2 + \frac{L\alpha^2 C_0}{2m}. \tag{3.19}$$

Strong convexity implies $\|\nabla f_t\| \geq 2\mu(f_t - f_\star)$ (Lemma 2.6). Plugging that in, subtracting $f_\star$ from both sides, and rearranging results in

$$\mathbf{E}_t[f_{t+1}] - f_\star \leq (1 - \mu\alpha)(f_t - f_\star) + \frac{L\alpha^2 C_0}{2m}. \tag{3.20}$$

Taking the total expectation and subtracting the constant $\alpha L C_0/(2m\mu)$ from both sides, yields

$$\mathbf{E}[f_{t+1} - f_\star] - \frac{\alpha L C_0}{2m\mu} \leq (1 - \mu\alpha)\left(\mathbf{E}[f_t - f_\star] - \frac{\alpha L C_0}{2m\mu}\right). \tag{3.21}$$

The result follows from iterating this contraction inequality backwards. □

The dependency on the step size in Theorem 3.1 makes clear that, for SGD to converge, we need to let the step size go to zero over time. A natural strategy would be to run SGD with a constant step size until the plateau level is reached, then decrease the step size and repeat. Such strategies are indeed used successfully in practice, but they are not very amenable to theoretical analysis.

The question is: At what speed should $\alpha_t$ decrease? This has already been answered in the seminal paper by Robbins and Monro [1951]. They show that a step size schedule for SGD has to satisfy

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty. \tag{3.22}$$

For a strongly convex function, this guarantees a sublinear convergence rate of

$$\mathbf{E}[f(\theta_t) - f_\star] \in O\left(\frac{1}{t}\right), \tag{3.23}$$

see, e.g., Theorem 4.7 in Bottou et al. [2018].

In Chapter 4, we will discuss locally optimal step sizes as well as an alternative strategy, which keeps the step size constant and instead increases the batch size.

## 3.3 *Stochastic Optimization for Deep Learning*

The training of deep neural networks involves large datasets, extremely high-dimensional parameter spaces, and non-convex

empirical risk landscapes. These characteristics favor cheap first-order over more involved methods like, for example, quasi-Newton methods. They also make it difficult to establish strong theoretical guarantees for any optimization method. As a consequence, the development of optimization methods for deep learning has been driven by empirical comparisons and many methods are heuristic in nature. This section describes the most popular methods.

*Momentum* SGD with momentum updates

$$v_t = \beta v_{t-1} - \alpha_t g_t, \quad \theta_{t+1} = \theta_t + v_t. \tag{3.24}$$

In deterministic optimization, this method has been shown to improve the dependency on the conditioning of the problem [Polyak, 1964, Nesterov, 1983]. Such an effect has not been proven to exist in the stochastic regime, but practitioners have found momentum to greatly improve stability and speed in deep learning optimization [e.g. Sutskever et al., 2013]. This is often attributed to the fact that momentum effectively averages stochastic gradients over time and, thus, has the effect of smoothing the gradient estimate. This is why a slight variation of the momentum method is often used, which can be written as

$$m_t = \beta m_{t-1} + (1-\beta) g_t, \quad \theta_{t+1} = \theta_t - \alpha_t m_t, \tag{3.25}$$

and explicitly maintains an exponential moving average of stochastic gradients.

*Adaptive gradient methods* are a family of stochastic optimization methods which employ an elementwise rescaling of the stochastic gradient, based on gradient information aggregated over the trajectory of the optimizer. This line of research was spawned by Duchi et al. [2011], who proposed ADAGRAD, which updates

$$\theta_{t+1} = \theta_t - \alpha_t \frac{g_t}{\sqrt{v_t + \varepsilon}}, \quad v_t = v_{t-1} + g_t^2 = \sum_{\tau=0}^{t} g_\tau^2, \tag{3.26}$$

where all operations on vectors (division, square-root, square) are to be understood elementwise, and $\varepsilon > 0$ is a small constant.

A number of variants and extensions of ADAGRAD have been proposed over time [Zeiler, 2012, Tieleman and Hinton, 2012]. The most popular adaptive gradient method, ADAM [Kingma and Ba, 2015], will be the subject of Chapter 5 and will be discussed in detail there.

## 3.4 Estimating the Gradient Variance

As we have seen, the (co-)variance of stochastic gradients plays a crucial role in stochastic optimization. This thesis presents different approaches of explicitly using (co-)variance information in stochastic optimization methods. Of course, the true covariance matrix,

$$\Sigma(\theta) = \mathbf{E}\left[(\nabla f(\theta;\xi) - \nabla f(\theta))(\nabla f(\theta;\xi) - \nabla f(\theta))^\mathsf{T}\right], \tag{3.27}$$

is unknown since it requires the gradients of *all* data points, the absence of which is exactly what defines the stochastic optimization regime.

However, we can *estimate* the covariance matrix from a minibatch of datapoints, $\xi_1, \ldots, \xi_m$. Just like we compute a stochastic gradient as the empirical mean of the individual gradients,

$$g \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} \nabla f(\theta;\xi_i), \tag{3.28}$$

we can compute their empirical covariance matrix as

$$S \stackrel{\text{def}}{=} \frac{1}{m-1}\sum_{i=1}^{m}(\nabla f(\theta;\xi_i) - g)(\nabla f(\theta;\xi_i) - g)^\mathsf{T} \in \mathbb{R}^{d\times d}. \tag{3.29}$$

For high-dimensional problems, this matrix will be expensive to compute and even store, but we can still compute its diagonal efficiently,

$$s \stackrel{\text{def}}{=} \frac{1}{m-1}\sum_{i=1}^{m}(\nabla f(\theta;\xi_i) - g)^2 \in \mathbb{R}^{d}, \tag{3.30}$$

where the square is to be understood elementwise. Such estimates will feature throughout this manuscript.

# 4
# *Variance-Based Step Size and Batch Size*

As we have seen, the stochasticity of the gradient estimates prevents SGD from converging with a constant step size. We now want to understand this behavior in more detail and discuss how both a decreasing step size and an increasing batch size can force SGD into convergence. This will lead us to propose the *Coupled Adaptive Batch Size* (CABS), a heuristic for automatic batch size adaptation.

The contents of this chapter, in particular those of Section 4.4, are based on the following publication:

> Lukas Balles, Javier Romero, and Philipp Hennig. Coupling adaptive batch sizes with learning rates. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 410–419, 2017b.

Coauthor contributions:

| | Sc. Ideas | Experiments | Interpretation | Writing |
|---|---|---|---|---|
| **Lukas Balles** | 70% | 100% | 70% | 70% |
| Javier Romero | 15% | 0% | 15% | 15% |
| Philipp Hennig | 15% | 0% | 15% | 15% |

## *4.1 Bias-Variance Trade-Off in Stochastic Optimization*

For the remainder of this chapter, we will mostly be concerned with the effect of a single SGD step, starting from a fixed but arbitrary point $\theta \in \mathbb{R}^d$. We will therefore occasionally drop $\theta$ from the notation, e.g., $\nabla f = \nabla f(\theta)$, $\Sigma = \Sigma(\theta)$, et cetera. Let $\tilde{g}$ be any estimate of $\nabla f$. It need not necessarily be of the form of Eq. (3.4) and we do not require it to be unbiased; we merely assume that it has a finite mean squared error (MSE), $\mathbf{E}\left[\|\tilde{g} - \nabla f\|_2^2\right] < \infty$. If we assume $f$ to be $L$-smooth then, by Lemma 2.1, an SGD step with $\tilde{g}$ as the

gradient estimate and step size $\alpha \leq 1/L$ yields

$$\begin{aligned}\mathbf{E}\left[f\left(\theta - \alpha\tilde{g}\right)\right] &\leq f(\theta) - \alpha\nabla f^{\mathsf{T}}\mathbf{E}[\tilde{g}] + \frac{L}{2}\mathbf{E}[\|\alpha\tilde{g}\|_2^2] \\ &\leq f(\theta) - \alpha\nabla f^{\mathsf{T}}\mathbf{E}[\tilde{g}] + \frac{\alpha}{2}\mathbf{E}[\|\tilde{g}\|_2^2] \\ &= f(\theta) - \frac{\alpha}{2}\left(\|\nabla f\|_2^2 - \mathbf{E}\left[\|\tilde{g}\|_2^2 - 2\nabla f^{\mathsf{T}}\tilde{g} + \|\nabla f\|_2^2\right]\right) \\ &= f(\theta) - \frac{\alpha}{2}\left(\|\nabla f\|_2^2 - \mathbf{E}\left[\|\tilde{g} - \nabla f\|_2^2\right]\right).\end{aligned} \tag{4.1}$$

Hence, the expected change in function value from a single SGD step depends on the gradient estimate $\tilde{g}$ exclusively through its MSE. To achieve a decrease in expectation, we require

$$\mathbf{E}\left[\|\tilde{g} - \nabla f\|_2^2\right] < \|\nabla f\|_2^2. \tag{4.2}$$

In particular, there is absolutely no need for $\tilde{g}$ to be unbiased. By a standard bias-variance decomposition, we know that

$$\underbrace{\mathbf{E}\left[\|\tilde{g} - \nabla f\|_2^2\right]}_{\text{MSE}} = \underbrace{\|\mathbf{E}[\tilde{g}] - \nabla f\|_2^2}_{\text{bias}} + \underbrace{\mathbf{E}\left[\|\tilde{g} - \mathbf{E}[\tilde{g}]\|_2^2\right]}_{\text{variance}}. \tag{4.3}$$

A certain degree of bias in the gradient estimate is, thus, absolutely acceptable if it leads to lower variance.

While not commonly viewed through this lens, a decreasing step size sequence can be seen as a way of trading a bias for lower variance to enforce the condition given in Eq. (4.2). A minibatch gradient (Eq. 3.4) is unbiased and has an MSE of

$$\mathbf{E}\left[\left\|g^{(m)} - \nabla f\right\|_2^2\right] = \frac{\operatorname{tr}(\Sigma)}{m}. \tag{4.4}$$

It will thus fulfill the requirement of Eq. (4.2) only up until the point where $\|\nabla f\|_2$ becomes too small compared to the variance. Now define

$$\tilde{g}^{(m)} = \gamma g^{(m)}, \quad \gamma \in [0, 1]. \tag{4.5}$$

Since $\mathbf{E}[\tilde{g}^{(m)}] = \gamma\nabla f$, this will be a biased estimate of $\nabla f$. Its MSE is

$$\mathbf{E}[\|\gamma g^{(m)} - \nabla f\|_2^2] = \|\nabla f\|_2^2 - 2\gamma\|\nabla f\|_2^2 + \gamma^2\left(\|\nabla f\|_2^2 + \frac{\operatorname{tr}(\Sigma)}{m}\right). \tag{4.6}$$

If $\|\nabla f\|^2$ is small compared to $\operatorname{tr}(\Sigma)/m$, choosing a small $\gamma$ leads to a better MSE. In fact, we will shortly see that the optimal $\gamma$ is a function of the "noise-to-signal ratio" $\operatorname{tr}(\Sigma)/\|\nabla f\|_2^2$.

We can thus think of the decreasing step size $\alpha_t$ in SGD (Eq. 1.4) as the product of two terms, $\alpha_t = \alpha\gamma_t$.

- The constant *geometric* step size of $\alpha \approx 1/L$ adjusts to the curvature of the problem. This is the optimal step size in the noise-free setting.
- A decreasing sequence $\gamma_t \in [0, 1]$ adjusts for *stochasticity* by increasingly shrinking the gradient estimate towards zero. This reduces its variance at the expense of a bias towards zero.

## 4.2 Optimal Step Size and Batch Size

For simplicity, assume we know $L$ and can set $\alpha$ to the optimal noise-free step size of $\alpha = 1/L$. Combining Eqs. (4.1) and (4.6), the *expected decrease* in function value from the step $\theta - \frac{\gamma}{L} g^{(m)}$ is

$$f(\theta) - \mathbf{E}[f(\theta - \frac{\gamma}{L} g^{(m)})] \geq \frac{\|\nabla f\|_2^2}{2L} \left( 2\gamma - \gamma^2 - \frac{\mathrm{tr}(\Sigma)}{\|\nabla f\|_2^2} \frac{\gamma^2}{m} \right). \quad (4.7)$$

The first factor, $\|\nabla f\|_2^2 / 2L$ is the decrease we could achieve with a noise-free gradient step ($\Sigma = 0, \gamma = 1$). The second factor is

$$\mathcal{G}(\gamma, m) \stackrel{\text{def}}{=} 2\gamma - \gamma^2 - \eta^2 \frac{\gamma^2}{m}, \quad (4.8)$$

where we define the noise-to-signal ratio

$$\eta^2 \stackrel{\text{def}}{=} \frac{\mathrm{tr}(\Sigma)}{\|\nabla f\|_2^2}. \quad (4.9)$$

The factor $\mathcal{G}(\gamma, m)$ is smaller than 1 and is a function of step size and batch size. It can be seen as the proportion of the noise-free decrease that can be realized in the stochastic setting.

Naturally, we want to set the step size and/or batch size such as to maximize the expected decrease. We will now discuss these optimal settings. With regards to the batch size, $\mathcal{G}(\gamma, m)$ is obviously maximized for $m \to \infty$. However, increasing the batch size also linearly[1] increases the computational cost of the stochastic gradient evaluation. We therefore consider a cost-normalized version

$$\frac{\mathcal{G}(\gamma, m)}{m} = 2\frac{\gamma}{m} - \frac{\gamma^2}{m} - \eta^2 \frac{\gamma^2}{m^2}, \quad (4.10)$$

which quantifies the expected decrease per "unit" of computational cost. The cost normalization does not alter the optimal setting of $\gamma$.

[1] In practice, the relationship is not perfectly linear due to potential for parallelization. This will be discussed further in Section 4.3.

### 4.2.1 Optimal Step Size

First, we find the step size that maximizes Eq. (4.10) given a batch size.

**Proposition 4.1.** *For a given batch size $m$, the step size which maximizes $\mathcal{G}(\gamma, m)/m$ is*

$$\gamma_\star = \left( 1 + \frac{\eta^2}{m} \right)^{-1} \quad (4.11)$$

*and yields*

$$\frac{\mathcal{G}(\gamma_\star, m)}{m} = \frac{1}{m + \eta^2}. \quad (4.12)$$

*Proof.* The proof of this and all other results in this chapter may be found in Appendix A. □

The step size factor $\gamma_\star$ is in $[0, 1]$ and adapts the step size to the noise-to-signal ratio: The larger the noise-to-signal ratio, the smaller the step size.

### 4.2.2 Optimal Batch Size

Next, we we optimize the batch size while treating the step size as given. For the purpose of this theoretical consideration, we treat the batch size as a continuous quantity; in practice it would have to be rounded to the nearest integer.

**Proposition 4.2.** *For a given step size $\gamma$, the batch size that maximizes $\mathcal{G}(\gamma, m)/m$ is given by*

$$m_\star = \frac{2\gamma}{2-\gamma}\eta^2. \tag{4.13}$$

*and yields*

$$\frac{\mathcal{G}(\gamma, m_\star)}{m_\star} = \frac{(2-\gamma)^2}{4\eta^2}. \tag{4.14}$$

Again, the optimal batch size adapts to the noise-to-signal ratio. Here, $m_\star$ is directly proportional to $\eta^2$. This means that the *effective* noise-to-signal ratio of the mini-batch gradient estimate, $\eta^2/m_\star$, is held *constant*.

### 4.2.3 A Batch Size of One is Theoretically Optimal

What is the jointly optimal setting of batch size $m$ and step size $\gamma$? It is easy to observe, e.g. from Eq. (4.12), that the batch size should be chosen as small as possible. Since the batch size is a discrete quantity, this means that $m = 1$ is the optimal choice, combined with a step size of

$$\gamma = \left(1 + \frac{\operatorname{tr}(\Sigma)}{\|\nabla f\|^2}\right)^{-1} \tag{4.15}$$

Interestingly, this simple finding is not discussed in the literature. Various works have argued for small batch sizes from different angles, but our reasoning above is much more fundamental.

- In their seminal paper, Bottou and Bousquet [2008] argued for stochastic gradient descent (which they understand to mean batch size 1) over full-batch gradient descent from the perspective of the overall tradeoff between total computational cost of the learning procedure and the *generalization* error. Their rationale is that—under a finite computational budget and in the presence of other sources of generalization error, such as approximation[2] and estimation[3] error—"it should not be necessary to carry out [the] minimization with great accuracy." Similar points have been made by Bottou and LeCun [2004], Bottou et al. [2018], and other authors.

- Another line of research is based on the (largely empirical) observation that in neural network training, small-batch SGD seems to find parameter settings with better generalization performance compared to large-batch SGD; even if both methods converge to similar levels of training loss [e.g., LeCun et al., 1998, Keskar et al., 2017].

[2] Approximation error stems from choosing a restricted class of prediction functions, even the "best" of which can only *approximate* the true function.

[3] Estimation error arises because, given a finite data set, we can only *estimate* the "best" function within our class of possible prediction functions.

In both cases, the arguments concern the generalization performance of the obtained solution. Our reasoning is distinct from that. We show that, even if we ignore the generalization aspect and just want to minimize the training loss as quickly as possible, SGD with a batch size of one is the (theoretically) most efficient greedy choice.

## 4.3 The Case for Adaptive Batch Size Methods

Of course, the previous section was a purely theoretical exercise, since the noise-to-signal ratio $\eta$ that determines the optimal step size and batch size is not known in practice. We would like to estimate it and use those estimates to automate SGD optimization. From this perspective, an adaptively increasing batch size is a valid practical choice for two reasons:

- The assumed linear relationship between batch size and computational cost does not hold if parallel computing resources are available.
- The noise-to-signal ratio—and therefore the optimal step size (Eq. 4.11)—is fundamentally hard to estimate at small batch sizes. Therefore, the optimal decreasing step size schedule does not lend itself to automation.

*Nonlinear Relationship of Batch Size and Computational Cost* When optimizing $G(\gamma, m)/m$, we assumed that the cost is linear in $m$, i.e., that evaluating a minibatch stochastic gradient on ten data points is ten times more expensive than evaluating it on a single data point. This assumption was an oversimplification. The forward and backward pass in neural nets consists largely of matrix multiplication operations. Hardware accelerators, such as GPUs or TPUs, are designed to execute such operations in a massively parallel fashion. Therefore, small batch sizes that do not fully leverage the computational power of the available hardware are wasteful. This holds even more so, if we have access to a distributed system of workers. The gradient computation can be trivially parallelized by splitting the minibatch and letting each worker compute the gradient on a subset.[4]

[4] This will involve communication overhead, which can be a bottleneck in practice, but this is beyond the scope of our considerations here.

*Estimating the Noise-to-Signal Ratio is Hard* The noise-to-signal ratio is fundamentally hard to estimate at small batch sizes. The obvious extreme case is $m = 1$, in which case no empirical variance estimate can be obtained. But even if we fix $m$ to a constant, such as $m = 10$ or $m = 100$, estimating $\eta^2$ is fundamentally ill-posed for large values of $\eta^2$. We can illustrate this with the following toy experiment: Assume we have $m$ iid samples $x_1, \dots, x_m \sim \mathcal{N}(\mu, \sigma^2)$, and we estimate $\eta^2 = \sigma^2/\mu^2$ based on the standard empirical moments:

$$\hat{\eta}^2 = \frac{\hat{\sigma}^2}{\hat{\mu}^2}, \quad \hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \hat{\sigma}^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \hat{\mu})^2.$$

Figure 4.1 shows histograms of the distribution of $\hat{\eta}^2$ for varying sample size $m$ and true values for $\eta^2$. While small noise-to-signal ratios ($\eta = 0.25$) can be estimated relatively accurately with as little as $m = 10$ samples, the estimates start to deteriorate as $\eta$ grows.

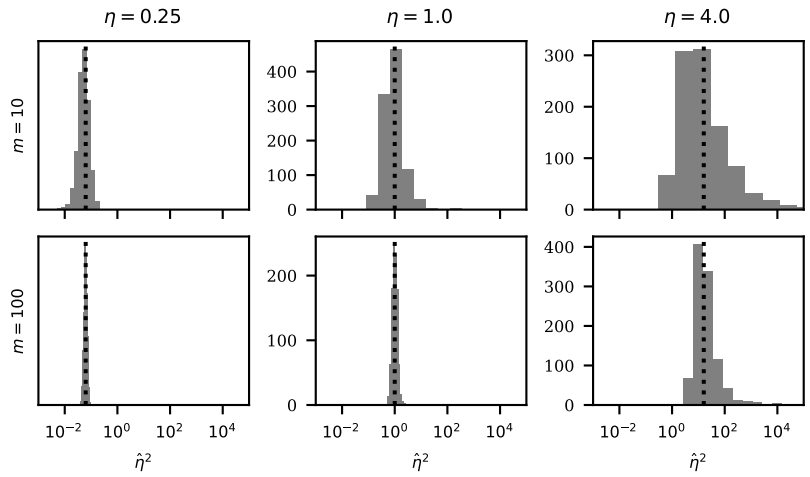


Figure 4.1: Estimating large noise-to-signal ratios is fundamentally ill-posed. Based on $m$ samples from a one-dimensional Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, we estimate the noise-to-signal ratio with the standard empirical moments. The histograms show the distribution of $\hat{\eta}^2 = \hat{\sigma}^2/\hat{\mu}^2$ over 1000 independent trials for varying sample size $m$ and true values for $\eta^2$. While small noise-to-signal ratios ($\eta = 0.25$) can be estimated relatively accurately with as little as $m = 10$ samples, the estimates start to deteriorate as $\eta$ grows. For $\eta = 4.0$, the estimate $\hat{\eta}^2$ scatters over roughly three orders of magnitude even for $m = 100$ samples.

Part of the problem is that $\hat{\mu}^2$ is a *biased* estimate of $\mu^2$,

$$\mathbf{E}[\hat{\mu}^2] = \mu^2 + \frac{\sigma^2}{m}. \tag{4.16}$$

This bias becomes more severe as $\sigma^2 \gg \mu$, i.e., in the high noise-to-signal ratio regime.

Operating in the optimal batch size regime can ameliorate this problem. Since the optimal batch size (Eq. 4.13) is proportional to $\eta^2$, an adaptive batch size scheme will keep the effective noise-to-signal ratio $\eta^2/m$ roughly constant over time. This will allow us to estimate $\eta^2$ with some fixed relative accuracy, even as $\eta^2$ grows to very large values.

## 4.4 *CABS—A Practical Adaptive Batch Size Method*

Based on these considerations, in this section, we propose a practical method for dynamic batch size adaptation. We emphasize that the method is a heuristic without guarantees and that it is only one way of leveraging the insights of the preceding sections. The method, called the *Coupled Adaptive Batch Size* (CABS), estimates the stochastic gradient variance and adapts the batch size to decrease the variance proportionally to the value of the objective function. In contrast to recent work, our algorithm couples the batch size to the learning rate, reflecting the known relationship between the two. We show experimentally that our batch size adaptation yields faster convergence and simplifies learning rate tuning.

### 4.4.1 Related Work

Increasing batch size schedules have received attention in recent works, e.g., by Friedlander and Schmidt [2012], who propose to increase the batch size by a pre-specified constant factor in each iteration. However, this strategy does not attempt to estimate the optimal batch size Eq. (4.13) and, therefore, is not adaptive to the specific problem.

Byrd et al. [2012] and De et al. [2017] have proposed to adapt the batch size based on variance estimates. Their criterion is based on the observation that $-g^{(m)}$ is a descent direction if

$$\|g^{(m)} - \nabla f\| \leq \theta \|g^{(m)}\|, \quad \text{with} \quad 0 \leq \theta < 1 \tag{4.17}$$

(proof in Appendix A). The squared expectation of the left-hand side evaluates to $\operatorname{tr}(\Sigma)/m$, which inspires a criterion of the form

$$m = \frac{1}{\theta^2} \frac{\operatorname{tr}(\Sigma)}{\|g^{(m)}\|^2}, \tag{4.18}$$

which resembles the optimal batch size (Eq. 4.13) with an empirical estimate of the noise-to-signal ratio. As we have just discussed, such estimates have fundamental limitations.

### 4.4.2 The Coupled Adaptive Batch Size (CABS)

Our method is based on the theoretically optimal batch size given in Proposition 4.2. Since we want to avoid a decreasing step size schedule, we set $\gamma = 1$, in which case the optimal batch size is simply

$$m_\star = 2\eta^2 = 2\frac{\operatorname{tr}(\Sigma)}{\|\nabla f\|^2}. \tag{4.19}$$

As discussed in Section 4.3, this noise-to-signal ratio is difficult to reliably and robustly estimate. To address this practical problem we propose to use the following rule, which we term the *Coupled Adaptive Batch Size* (CABS):

$$m = \alpha \frac{\operatorname{tr}(\Sigma)}{f}. \tag{4.20}$$

A formal justification for this simplification will be given shortly, but first we want to highlight some intuitive benefits of this batch size adaptation scheme.

Most importantly, by replacing the squared gradient norm with the function value, the quantity becomes easier to estimate. The function value computed on a minibatch, $|\mathcal{B}|^{-1} \sum_{i \in \mathcal{B}} f(\theta, \xi_i)$, is an unbiased estimate of the true quantity $f(\theta)$. That is in contrast to the gradient norm, where $\|g^{(m)}\|^2$ is a biased estimate of $\|\nabla f\|^2$, which causes the problems described in Section 4.3.

Another advantage of the CABS rule is the direct *coupling* of learning rate and batch size, which reflects the known relationship between the two. Using CABS can thus be seen as "tailoring" the noise level to the chosen learning rate. We show experimentally, see

Section 4.4.4, that this makes finding a well-performing learning rate easier.

*Mathematical Motivation for CABS* CABS was largely motivated as a heuristic based on experimental success. However, one can show that the CABS rule matches the optimal batch size for a very simple optimization problem of the form

$$f(\theta) = \frac{h}{2}\|\theta - \theta_\star\|^2. \tag{4.21}$$

For this function, we have

$$\|\nabla f(\theta)\|^2 = 2h(f(\theta) - f(\theta_\star)) = 2hf(\theta). \tag{4.22}$$

Plugging this into the optimal batch size formula (Eq. 4.19) and assuming a well-tuned learning rate, $\alpha \approx 1/L = 1/h$, yields the CABS rule.

This is, of course, a substantial simplification. The result can partly be generalized to the less restrictive assumption of $\mu$-strong convexity, under which we still have $\|\nabla f(\theta)\| \geq 2\mu(f(\theta) - f_\star)$ (see Lemma B.2). This gives us a proportionality of the squared gradient norm and the function value, if we assume $f_\star \approx 0$, which holds in most practical deep learning problems, where the empirical risk can typically reach zero [Zhang et al., 2017].

### 4.4.3 *Practical Implementation*

Obviously, neither $f$ nor $\mathrm{tr}(\Sigma)$ are known exactly at each individual SGD step, but unbiased estimates of both quantities can be obtained from a minibatch, as described in Chapter 3.

We realize the CABS criterion in a predictive manner, meaning that we do *not* find the exact batch size that satisfies Eq. (4.20) in each single optimization step.[5] Instead, we leverage the observation that gradient variance and function value change only slowly from one optimization step to the next, which allows us to use our current estimates of $f$ and $\mathrm{tr}(\Sigma)$ to set the batch size used for the *next* optimization step. It also allows for a smoothing of both quantities over multiple optimization steps.

The resulting batch size is rounded to the nearest integer and clipped at minimal and maximal batch sizes. A minimal batch size avoids under-utilization of the computational resources with very small batches and provides additional stability of the algorithm in the small-batch regime. A maximal batch size is necessary due to hardware limitations: In contemporary deep learning, GPU memory limits the number of samples that can be processed at once. Our implementation has such a limit but it was never reached in our experiments.[6]

Algorithm 1 provides pseudo-code. EVALUATE($\theta$, $\mathcal{B}$) denotes an evaluation of function value $f(\theta)$, stochastic gradient $g(\theta)$ and variance estimate $s(\theta)$ (Eq. 3.30) using mini-batch $\mathcal{B}$. ROUND_CLIP($m$,

[5] Byrd et al. [2012] and De et al. [2017] increase the batch size by a small increment whenever the criterion is not satisfied, and only then perform the update. This incremental computation introduces an overhead and, when the increment is small, can lead to under-utilization of computing resources.

[6] Note that *algorithmic* batch size (the number of training samples used to compute a gradient estimate before updating the parameters) and *computational* batch size (the number of training samples that are processed simultaneously) are in principle independent and one could split an algorithmic batch into feasible computational batches when necessary, freeing the algorithm of hardware-specific constraints.

$m_{\min}$, $m_{\max}$) rounds $m$ to the nearest integer and clips it at the provided minimal and maximal values.

**Require:** Learning rate $\alpha$, initial parameters $\theta_0$, number of steps $T$, batch size bounds $(m_{\min}, m_{\max})$, running average constant $\mu$
$\theta \leftarrow \theta_0$, $m \leftarrow m_{\min}$, $f_{\text{avg}} \leftarrow 0$, $\hat{\sigma}^2 \leftarrow 0$
**for** $t = 1, \ldots, T$ **do**
Draw a mini-batch $\mathcal{B}$ of size $m$
$f, g, s \leftarrow \textsc{Evaluate}(\theta, \mathcal{B})$
$\theta \leftarrow \theta - \alpha g$
$\hat{\sigma}^2 \leftarrow \mu\hat{\sigma}^2 + (1-\mu)\|s\|_1$
$f_{\text{avg}} \leftarrow \mu f_{\text{avg}} + (1-\mu) f$
$m \leftarrow \textsc{Round\_Clip}(\alpha\hat{\sigma}^2 / f_{\text{avg}}, m_{\min}, m_{\max})$
**end for**

**Algorithm 1:** SGD with Coupled Adaptive Batch Size

### 4.4.4 *Experiments*

We evaluate the proposed batch size adaptation method by training convolutional neural networks (CNNs) on four popular image classification benchmark data sets: MNIST [LeCun et al., 1989], Street View House Numbers (SVHN) [Netzer et al., 2011], as well as CIFAR-10 and CIFAR-100 [Krizhevsky, 2009]. A description of the neural network architectures and further details can be found in Appendix A.

We compared against constant batch sizes 32, 128, and 512. We also compare against a batch size adaptation based on the criterion given by Eq. (4.18) used by Byrd et al. [2012] and De et al. [2017]. Since implementation details differ between these two works, and both combine batch size adaptation with other techniques (Byrd et al. [2012] use it in a Newton-CG method, De et al. [2017] use a backtracking line search), we resort to a custom implementation of said criterion. For a fair comparison, we realize it in a similar manner as CABS. That is, we use the criterion of Eq. (4.18), while keeping the predictive update mechanism for the batch size, the smoothing via exponential moving averages, rounding and clipping exactly as in our CABS implementation described in Section 4.4.3 and Algorithm 1. This method will simply be referred to as *Competitor* in the remainder of this section.

Performance is tracked as a function of the number of accessed training examples, instead of the number of optimization steps. The (constant) learning rate for each batch size method was tuned for maximum test accuracy given the fixed budget of accessed training examples. We tried six candidates $\alpha \in \{0.3, 0.1, 0.06, 0.03, 0.01, 0.006\}$; this relevant range has been determined with a few exploratory experiments. In addition to the learning rate, the competitor method has a free parameter $\theta$. De et al. [2017] suggest setting it to 1.0, the highest possible noise tolerance. In our experiments, we found the performance of the method to be fairly sensitive to the choice of

$\theta$. We thus tried $\theta \in \{0.6, 0.8, 1.0\}$ and report results for the best-performing choice. For CABS, there is no such parameter to tune.

*Results and Discussion* Results are depicted in Figure 4.2. On SVHN, CIFAR-10 and CIFAR-100, CABS yields significantly faster decrease in training loss with the curve continuously lying below all others. It also achieves the best test set accuracy of all methods on all three problems. While the margin over the second-best method is very small on SVHN, it amounts to a noticeable 0.4 percentage points on CIFAR-10 and even 1.4 points on CIFAR-100.

Surprisingly, on MNIST—the *least* complex of the benchmark problems we investigated—our method is outperformed by the small constant batch size of 32 and the competitor method, which also chooses small batch sizes throughout. CABS makes rapid progress initially, but seems to choose unnecessarily large batch sizes later on. We conjecture that CABS overestimates the gradient variance due to the homogeneous structure of the MNIST data set; if the distribution of gradients is very closely-centered, outliers in a few coordinate directions lead to comparably high variance estimates.

Overall, CABS outperforms alternative batch size schemes on three out of the four benchmark problems we investigated and the benefits seem to increase with the complexity of the problem (MNIST $\rightarrow$ SVHN $\rightarrow$ CIFAR-10 $\rightarrow$ CIFAR-100).

*Learning Rate Tuning* We also present results regarding the sensitivity to the choice of learning rate when using CABS. As detailed above, the coupling of learning rate and batch size in CABS can be seen as tailoring the noise level to the chosen learning rate. This suggests that the performance of the optimizer should be less sensitive to the choice of learning rate when adapting the batch size with our method. Indeed, this can be observed in Figure 4.3. CABS significantly reduces the dependency of the performance on the learning rate compared to both the constant batch size and the competing adaptive method.

## 4.5 Conclusion

In this chapter, we saw that the gradient estimate affects the performance of SGD exclusively through its mean squared error; in particular, it need not be unbiased. A decreasing step size can be seen as a way of enforcing a sufficiently small MSE at the expense of a bias towards zero.

An alternative way to decrease the MSE is to increase the batch size. While a decreasing step size is theoretically more efficient, multiple practical considerations favor an increasing batch size. In particular, an increasing batch size is more amenable to adaptive automation.

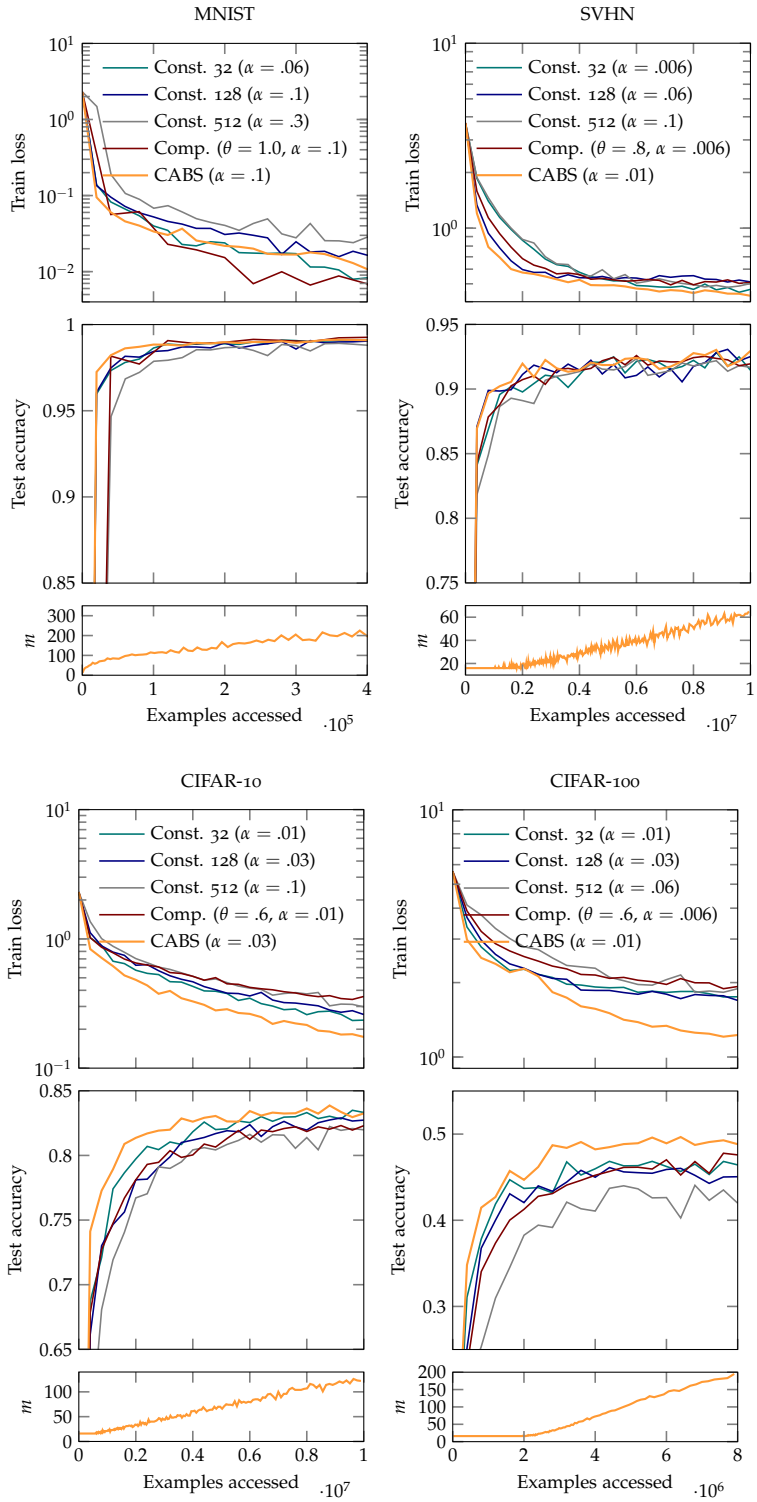


Figure 4.2: Experimental results for different batch size strategy. The (shared) horizontal axis indicates the number of examples used for training. The top and middle panel depict evolution of training loss and test accuracy, respectively, color-coded for different batch size methods, each with its optimal learning rate. The bottom panel shows the batch size chosen by CABS. CABS outperforms competing method on three out of the four problems.

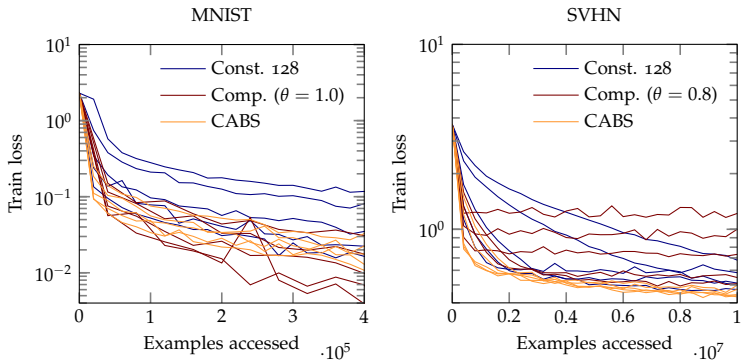


Figure 4.3: Learning rate sensitivity. Families of training loss curves for CABS, *Competitor* and a constant batch size (color-coded). Each individual curve corresponds to a learning rate $\alpha \in \{.1, .06, .03, .01, .006\}$. The curves for different step sizes cluster much more closely together when adapting the batch size with CABS.

We proposed CABS, a practical rule for dynamic batch size adaptation based on estimates of the gradient variance and coupled to the chosen learning rate. In our experiments, CABS was able to speed up SGD training in neural networks and simplify the tuning of the learning rate.

# 5
# *Dissecting Adam*

The deep learning community has originated a number of stochastic optimization algorithms that achieve great practical performance—often outperforming basic SGD significantly—but are of somewhat heuristic nature. Arguably the most popular among those is the ADAM optimizer [Kingma and Ba, 2015].[1] In this chapter, we want to gain a better understanding of this method. More specifically, we interpret ADAM as a combination of two aspects: for each coordinate the update direction is determined by the *sign* of stochastic gradients, whereas the update magnitude is determined by an estimate of their *relative variance*. We disentangle these two aspects and consider them in isolation, gaining insight into the mechanisms underlying ADAM. This analysis also extends recent results on adverse effects of ADAM on generalization performance, isolating the sign aspect as the problematic one.

[1] Some of our considerations naturally extend to ADAM's relatives, like RMSPROP [Tieleman and Hinton, 2012], but we restrict our attention to ADAM to keep the presentation concise.

The contents of this chapter are based on:

Lukas Balles and Philipp Hennig. Dissecting Adam: The sign, magnitude and variance of stochastic gradients. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 413–422. PMLR, 2018.

Coauthor contributions:

| | Sc. Ideas | Experiments | Interpretation | Writing |
|---|---|---|---|---|
| **Lukas Balles** | 80% | 100% | 80% | 70% |
| Philipp Hennig | 20% | 0% | 20% | 30% |

## *5.1 Introduction*

Let us start by introducing the ADAM method. Assuming access to stochastic gradient estimates $g_t$ at $\theta_t$, ADAM maintains exponential moving averages of observed stochastic gradients,

$$\tilde{m}_t = \beta_1 \tilde{m}_{t-1} + (1-\beta_1) g_t, \tag{5.1}$$

and their elementwise squares,

$$\tilde{v}_t = \beta_2 \tilde{v}_{t-1} + (1-\beta_2) g_t^2, \tag{5.2}$$

with $\beta_1, \beta_2 \in (0, 1)$. These are initialized as $\tilde{m}_0 = \tilde{v}_0 = 0$ and thus are biased towards small values in early iterations. ADAM "bias-corrects" these moving averages,

$$m_t = \frac{\tilde{m}_t}{1 - \beta_1^{t+1}}, \quad v_t = \frac{\tilde{v}_t}{1 - \beta_2^{t+1}}, \tag{5.3}$$

making sure that $m_t$ and $v_t$ are convex combinations of (squared) stochastic gradients. With that, the ADAM update reads

$$\theta_{t+1} = \theta_t - \alpha_t \frac{m_t}{\sqrt{v_t} + \varepsilon} \tag{5.4}$$

with a small (default: $10^{-8}$) constant $\varepsilon > 0$ preventing division by zero. Here and throughout the rest of this chapter, all operations on vectors—in particular multiplication, division, squares, square-roots, and the sign operator—have to be understood elementwise.

### 5.1.1 A New Perspective on Adam

Denote $m_{t,i} := [m_t]_i$, $v_{t,i} := [v_t]_i$, et cetera. Ignoring $\varepsilon$ and assuming $|v_{t,i}|, |m_{t,i}| > 0$ for the moment, we can rewrite the ADAM update as

$$\frac{m_t}{\sqrt{v_t}} = \left(\frac{v_t}{m_t^2}\right)^{-1/2} \operatorname{sign}(m_t) = \left(1 + \frac{v_t - m_t^2}{m_t^2}\right)^{-1/2} \operatorname{sign}(m_t), \tag{5.5}$$

where the sign is to be understood elementwise.

The name ADAM stands (loosely) for *adaptive moment estimation*. Kingma and Ba [2015] argue that $m_t$ is an estimate of the first moment (mean) of $g_t$, i.e., $m_t \approx \mathbf{E}[g_t]$, whereas $v_t$ is an estimate of its (elementwise) non-central second moment, $v_t \approx \mathbf{E}[g_t^2]$. If we adopt that notion—which we will discuss further in Section 5.4.1—the quantity $(v_t - m_t^2)$ can be seen as an estimate of the vector of elementwise variances of a stochastic gradient, $v_t - m_t^2 = \mathbf{E}[g_t^2] - \mathbf{E}[g_t]^2 =: \sigma_t^2 \in \mathbb{R}^d$. That is the diagonal of its covariance matrix. The magnitude of $m_t$ is effectively cancelled out of the update direction; it *only* appears in the ratio $(v_t - m_t^2)/m_t^2$. Hence, ADAM can be interpreted as a combination of two aspects:

- The update *direction* for the $i$-th coordinate is given by the *sign* of $m_{t,i}$, whereas
- the update *magnitude* for the $i$-th coordinate is solely determined by the global step size $\alpha$ and the factor

$$\gamma_{t,i} := \left(1 + \hat{\eta}_{t,i}^2\right)^{-1/2}, \tag{5.6}$$

where $\hat{\eta}_{t,i}^2$ estimates the *relative variance* or *noise-to-signal ratio*,

$$\hat{\eta}_{t,i}^2 := \frac{v_{t,i} - m_{t,i}^2}{m_{t,i}^2} \approx \frac{\mathbf{Var}[g_{t,i}]}{\mathbf{E}[g_{t,i}]^2} =: \eta_{t,i}^2. \tag{5.7}$$

We will refer to the second aspect as *variance adaptation*. The variance adaptation factors shorten the update in directions of high

relative variance, adapting for varying reliability of the stochastic gradient estimate in different coordinates.

The above interpretation of ADAM's update rule has to be viewed in contrast to existing ones. A motivation given by Kingma and Ba [2015] is that $v_t$ is a diagonal approximation to the empirical Fisher information matrix, making ADAM an approximation to natural gradient descent [Amari, 1998]. However, there are fundamental reservations towards the *empirical* Fisher, which have been discussed, among others, by Martens [2020] and will be the main topic of Chapter 7 of this thesis. Furthermore, even if we accepted $\operatorname{diag}(v_t)$ as an approximation of the Fisher, ADAM preconditions with its square root, which is entirely uncalled for from the natural gradient perspective.

Another possible motivation—which is not found in peer-reviewed publications but circulates the community as "conventional wisdom"—is that ADAM performs an approximate *whitening* of stochastic gradients. However, this view hinges on the fact that ADAM divides by the square-root of the *non-central* second moment and not by the standard deviation.

### 5.1.2 Overview

Both aspects of ADAM—taking the sign and variance adaptation—are briefly mentioned in Kingma and Ba [2015], who note that "[t]he effective stepsize [...] is also invariant to the scale of the gradients" and refer to $m_t/\sqrt{v_t}$ as a "signal-to-noise ratio". The purpose of this work is to disentangle these two aspects in order to discuss and analyze them in isolation. This naturally suggests an ablation study by incorporating one of the aspects while excluding the other. Taking the sign of a stochastic gradient without any further modification gives rise to SIGN-SGD[2]. On the other hand, *Stochastic Variance-Adapted Gradient* (SVAG)—to be derived in §5.3.2—applies variance adaptation directly to the stochastic gradient instead of its sign. Together with ADAM, the momentum variants of SGD, SIGN-SGD, and SVAG constitute the four possible recombinations of the sign aspect and the variance adaptation, see Fig. 5.1.

[2] In the original publication [Balles and Hennig, 2018], we referred to this method as *Stochastic Sign Descent (SSD)*. It was concurrently studied by Bernstein et al. [2018] under the name SIGN-SGD, which has been used more frequently since, and has therefore been adopted for this manuscript.

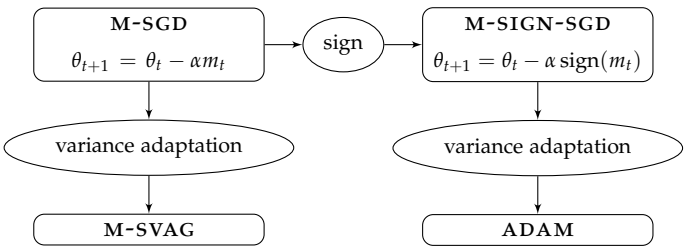


Figure 5.1: The methods under consideration in this paper. "M-" refers to the use of $m_t$ in place of $g_t$, which we colloquially refer to as the *momentum variant*. M-SVAG will be derived below.

The original publication on which the current chapter is based [Balles and Hennig, 2018] contained a brief discussion of SIGN-SGD, which has since been refined in a separate paper [Balles et al., 2020]. An analysis of the sign aspect is thus deferred to a sepa-

rate Chapter 6 and we proceed as follows: After discussing related work, Section 5.3 presents a principled derivation of elementwise variance adaptation factors and introduces SVAG. Subsequently, we discuss the practical implementation of variance-adapted methods (Section 5.4). Section 5.5 draws a connection to recent work on ADAM's effect on generalization. Finally, Section 5.6 presents experimental results.

## 5.2 *Related Work*

Sign-based optimization algorithms have received some attention in the past. RPROP [Riedmiller and Braun, 1993] is based on gradient signs and adapts per-element update magnitudes based on observed sign changes. Karimi et al. [2016] prove convergence results for a variant of RPROP in the *non-stochastic* case. Seide et al. [2014] empirically investigate the use of stochastic gradient signs in a distributed setting with the goal of reducing communication cost. SIGN-SGD has been studied by Bernstein et al. [2018] concurrently with the original paper on which this chapter is based. Refinements of the theoretical analysis by Bernstein et al. [2019] and Safaryan and Richtárik [2019] appeared thereafter. These works will be discussed in detail in Chapter 6, which is based on our own follow-up work [Balles et al., 2020].

Variance-based update directions have been proposed before, e.g., by Schaul et al. [2013], where the variance appears together with curvature estimates in a diagonal preconditioner for SGD. Their variance-dependent terms resemble the variance adaptation factors we will derive in Section 5.3. The corresponding parts of our work complement that of Schaul et al. [2013] in various ways. Most notably, we provide a principled motivation for variance adaptation that is independent of the update direction and use that to extend the variance adaptation to the momentum case.

Research on *variance-reduced* stochastic optimization methods [e.g., Le Roux et al., 2012, Johnson and Zhang, 2013] is related but largely orthogonal to our notion of variance adaptation: The former aims to construct gradient estimates with lower variance whereas the latter adapts the search direction to mitigate adverse effects of the (remaining) variance.

## 5.3 *Variance Adaptation*

In our decomposition of the ADAM update (Eq. 5.5), we observed that the update magnitude in each coordinate is scaled by $\gamma_{t,i} = (1 + \hat{\eta}_{t,i}^2)^{-1/2}$, where $\hat{\eta}_{t,i}$ is an estimate of the relative variance, $\hat{\eta}_{t,i} \approx \mathbf{Var}[g_{t,i}] / \mathbf{E}[g_{t,i}]^2$. This *shortens* the update in directions of high relative variance, adapting for varying reliability of the stochastic gradient estimate in different coordinates. Considering this general idea in isolation from the sign aspect naturally suggests to employ it on other update directions, for example directly on the stochastic

gradient instead of its sign.

A principled motivation arises from the following consideration: Assume we want to update in a direction $p \in \mathbb{R}^d$, but only have access to an estimate $\hat{p}$ with $\mathbf{E}[\hat{p}] = p$. We allow elementwise factors $\gamma \in \mathbb{R}^d$ and update $\gamma \odot \hat{p}$, where $\odot$ denotes elementwise multiplication for emphasis. One way to make "optimal" use of these factors is to choose them such as to minimize the expected squared distance to the desired update direction.

**Lemma 5.1.** *Let $\hat{p} \in \mathbb{R}^d$ be a random variable with $\mathbf{E}[\hat{p}] = p$ and $\mathbf{var}[p_i] = \sigma_i^2$. Then $\mathbf{E}[\|\gamma \odot \hat{p} - p\|_2^2]$ is minimized by*

$$\gamma_i = \frac{\mathbf{E}[\hat{p}_i]^2}{\mathbf{E}[\hat{p}_i^2]} = \frac{p_i^2}{p_i^2 + \sigma_i^2} = \frac{1}{1 + \sigma_i^2 / p_i^2}. \tag{5.8}$$

The factor $\gamma_i$ is determined by the relative variance of $\hat{p}_i$ with a maximum of 1 for vanishing variance and approaching 0 for $\sigma_i \gg p_i^2$.

The expression in Eq. (5.8) already looks remarkably similar to the variance adaptation factors in our decomposition of the ADAM update (Eq. 5.5). However, we have to slightly adapt the argument to handle the nonlinear sign operator. Assume we want to update in a direction $\mathrm{sign}(p)$, but only have access to $\mathrm{sign}(\hat{p})$. The following Lemma gives the optimal (i.e., achieving minimal mean-squared distances) variance adaptation factors for this case.

**Lemma 5.2.** *Let $\hat{p} \in \mathbb{R}^d$ be a random variable with $\mathbf{E}[\hat{p}] = p$ and $\mathbf{var}[p_i] = \sigma_i^2$. Then $\mathbf{E}[\|\gamma \odot \mathrm{sign}(\hat{p}) - \mathrm{sign}(p)\|_2^2]$ is minimized by*

$$\gamma_i = (2\rho_i - 1), \tag{5.9}$$

*where $\rho_i := \mathbf{P}[\mathrm{sign}(\hat{p}_i) = \mathrm{sign}(p_i)]$.*

The quantity $\rho_i$ is very intuitive; it is simply the probability of "guessing the sign correctly" when using the noisy estimate $\hat{p}_i$ instead of $p_i$. This means that $\gamma_i$ is proportional to the success probability with a maximum of 1 when we are perfectly certain about the sign of the gradient ($\rho_i = 1$) and a minimum of 0 in the absence of information ($\rho_i = 0.5$, i.e., we are randomly guessing the sign).

### 5.3.1 *Adam as Variance-Adapted Sign-SGD*

How does this general notion of variance adaptation relate to ADAM? We argue that ADAM implements an approximation to the optimal variance adaptation factors for SIGN-SGD, as given by Lemma 5.2. Applied to $p = \nabla f_t$, $\hat{p} = g_t$, the Lemma suggests to use variance adaptation factors $\gamma_{t,i}^{\mathrm{opt}} = 2\rho_{t,i} - 1$ with $\rho_{t,i} = \mathbf{P}\left[\mathrm{sign}(g_{t,i}) = \mathrm{sign}(\nabla f_{t,i})\right]$. The success probability $\rho_{t,i}$ generally depends on the distribution of $g_t$. If we assume $g_t$ to be normally distributed,[3] we have

$$\gamma_{t,i}^{\mathrm{opt}} = 2\rho_{t,i} - 1 = \mathrm{erf}\left(\frac{|\nabla f_{t,i}|}{\sqrt{2}\sigma_{t,i}}\right) = \mathrm{erf}\left(\frac{1}{\sqrt{2}\eta_{t,i}}\right), \tag{5.10}$$

[3] Since a mini-batch gradient $g_t$ is the sum of iid terms, this is supported by a central limit argument for large batch sizes.

see Appendix B.2.1. Note that it is uniquely determined by $\eta_{t,i}$, that is, the relative variance of $g_{t,i}$.

Figure 5.2 shows that these optimal variance adaptation factors in Eq. (5.10) are closely approximated by $(1+\eta_{t,i}^2)^{-1/2}$, the variance adaptation terms we identified in ADAM (see Eq. 5.5). Hence, ADAM can be regarded as an approximate realization of this optimal variance adaptation scheme. This comes with the caveat that ADAM applies these factors to $\operatorname{sign}(m_t)$ instead of $\operatorname{sign}(g_t)$. Variance adaptation for $m_t$ will be discussed further in §5.4.3 and in Appendix B.2.4.

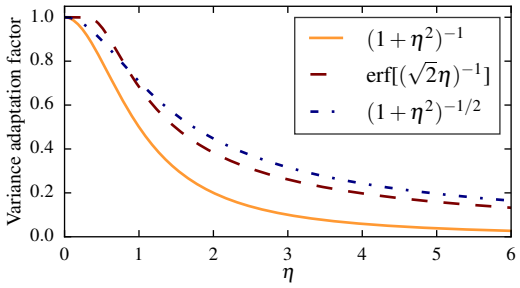


Figure 5.2: Variance adaptation factors as functions of the relative standard deviation $\eta$. The optimal factor for the sign of a (Gaussian) stochastic gradient is $\operatorname{erf}[(\sqrt{2}\eta)^{-1}]$, which is closely approximated by $(1+\eta^2)^{-1/2}$, the factor implicitly employed by ADAM. $(1+\eta^2)^{-1}$ is the optimal factor for a stochastic gradient.

### 5.3.2 *Stochastic Variance-Adapted Gradient (SVAG)*

Applying Lemma 5.1 to $p = \nabla f_t, \hat{p} = g_t$, the optimal variance adaptation factors for a stochastic gradient are found to be

$$\gamma_{t,i} = \frac{\nabla f_i^2}{\nabla f_i^2 + \sigma_i^2} = \frac{1}{1+\sigma_i^2/\nabla f_i^2} = \frac{1}{1+\eta_i^2}. \tag{5.11}$$

A term of this form also appears, together with diagonal curvature estimates, in Schaul et al. [2013]. We refer to the method updating along $\gamma \odot g$ as *Stochastic Variance-Adapted Gradient* (SVAG). To support intuition, Fig. 5.3 shows a conceptual sketch of this variance adaptation scheme.

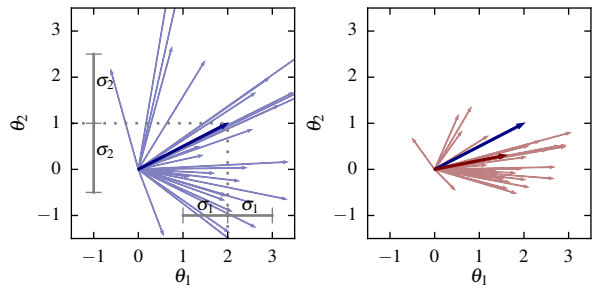


Figure 5.3: Conceptual sketch of variance-adapted stochastic gradients. The top panel shows the true gradient $\nabla f = (2,1)$ and stochastic gradients scattered around it with $(\sigma_1,\sigma_2) = (1,1.5)$. In the bottom panel, we apply variance adaptation with factors Eq. (5.11). In this example, the $\theta_2$-coordinate has much higher relative variance ($\eta_2^2 = 2.25$) than the $\theta_1$-coordinate ($\eta_1^2 = 0.25$) and is thus scaled back more. This reduces the variance of the update direction at the expense of biasing it away from the true gradient in expectation.

Variance adaptation of this form guarantees convergence *without* manually decreasing the global step size. We recover the $\mathcal{O}(1/t)$ rate of SGD for smooth, strongly convex functions. We emphasize that this result considers an *idealized* version of SVAG with exact variance adaptation factors. It should be considered as a motivation for this variance adaptation strategy, not a statement about its performance with estimated variance adaptation factors.

**Theorem 5.1.** *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be $\mu$-strongly convex and $L$-smooth. We update $\theta_{t+1} = \theta_t - \alpha(\gamma_t \odot g_t)$, with stochastic gradients $\mathbf{E}_t[g_t] = \nabla f_t$, $\mathbf{Var}_t[g_{t,i}] = \sigma_{t,i}^2$, variance adaptation factors $\gamma_{t,i} = \nabla f_{t,i}^2 / (\nabla f_{t,i}^2 + \sigma_{t,i}^2)$, and a global step size $\alpha = 1/L$. Assume that there are constants $c_v, M_v > 0$ such that $\sum_{i=1}^d \sigma_{t,i}^2 \leq c_v \|\nabla f_t\|^2 + M_v$ (cf. Assumption 3.1). Then*

$$\mathbf{E}[f(\theta_t) - f_\star] \in \mathcal{O}\left(\frac{1}{t}\right), \tag{5.12}$$

*where $f_\star$ is the minimum value of $f$.*

The assumption on the variance is relatively mild, allowing it to be non-zero everywhere and to grow quadratically in the gradient norm.

## 5.4 Practical Implementation of M-SVAG

Section 5.3 has introduced the general idea of variance adaptation; we now discuss its practical implementation. For the sake of a concise presentation, we focus on one particular variance-adapted method, M-SVAG, which applies variance adaptation to the update direction $m_t$. This method is of particular interest due to its relationship to ADAM outlined in Figure 5.1. Many of the following considerations correspondingly apply to other variance-adapted methods, e.g., SVAG and variants of ADAM, some of which are discussed and evaluated in Appendix B.3.

### 5.4.1 Estimating Gradient Variance

In practice, the optimal variance adaptation factors are unknown and have to be estimated. A key ingredient is an estimate of the stochastic gradient variance. We have argued in the introduction that ADAM obtains such an estimate from moving averages, $\sigma_{t,i}^2 \approx v_{t,i} - m_{t,i}^2$. The underlying assumption is that the distribution of stochastic gradients is approximately constant over the effective time horizon of the exponential moving average, making $m_t$ and $v_t$ estimates of the first and second moment of $g_t$, respectively:

**Assumption 5.1.** *At step $t$, assume*

$$\mathbf{E}[m_{t,i}] \approx \nabla f_{t,i}, \quad \mathbf{E}[v_{t,i}] \approx \nabla f_{t,i}^2 + \sigma_{t,i}^2. \tag{5.13}$$

While this can only ever hold approximately, Assumption 5.1 is the tool we need to obtain gradient variance estimates from past gradient observations. It will be more realistic in the case of high noise and small step size, where the variation between successive stochastic gradients is dominated by stochasticity rather than change in the true gradient.

We make two modifications to ADAM's variance estimate. First, we will use the same moving average constant $\beta_1 = \beta_2 = \beta$ for $m_t$ and $v_t$. This constant should define the effective range for which we implicitly assume the stochastic gradients to come from the same

distribution, making different constants for the first and second moment implausible.

Secondly, we account for a systematic bias in the variance estimate. As we show in Appendix B.2.4, under Assumption 1,

$$\mathbf{E}[m_{t,i}^2] \approx \nabla f_{t,i}^2 + \rho(\beta, t)\sigma_{t,i}^2, \tag{5.14}$$

with

$$\rho(\beta, t) \stackrel{\text{def}}{=} \frac{(1-\beta)(1+\beta^{t+1})}{(1+\beta)(1-\beta^{t+1})}, \tag{5.15}$$

and consequently $\mathbf{E}[v_{t,i} - m_{t,i}^2] \approx (1 - \rho(\beta, t))\, \sigma_{t,i}^2$. We correct for this bias and use the variance estimate

$$\hat{s}_t := \frac{1}{1-\rho(\beta,t)}(v_t - m_t^2). \tag{5.16}$$

*Mini-Batch Gradient Variance Estimates:* An alternative variance estimate can be computed locally "within" a single minibatch, see Section B.4 of the appendix. We have experimented with both estimators and found the resulting methods to have similar performance. For the main paper, we stick to the moving average variant for its ease of implementation and direct correspondence with ADAM. We present experiments with the minibatch variant in the supplementary material. These demonstrate the merit of variance adaptation *irrespective* of how the variance is estimated.

### 5.4.2 Estimating the Variance Adaptation Factors

The gradient variance itself is not of primary interest; we have to estimate the variance adaptation factors, given by Eq. (5.11) in the case of SVAG. We propose to use the estimate

$$\hat{\gamma}_t^g = \frac{1}{1 + \hat{s}_t / m_t^2} = \frac{m_t^2}{m_t^2 + \hat{s}_t}. \tag{5.17}$$

While $\hat{\gamma}_t^g$ is an intuitive quantity, it is *not* an unbiased estimate of the exact variance adaptation factors as defined in Eq. (5.11). To our knowledge, unbiased estimation of the exact factors is intractable. We have experimented with several partial bias correction terms but found them to have destabilizing effects.

### 5.4.3 Incorporating Momentum

So far, we have considered variance adaptation for the update direction $g_t$. In practice, we may want to update in the direction of $m_t$ to incorporate momentum.[4] According to Lemma 5.1, the variance adaptation factors should then be determined by the relative of variance of $m_t$.

Once more adopting Assumption 5.1, we have $\mathbf{E}[m_t] \approx \nabla f_t$ and $\mathbf{Var}[m_{t,i}] \approx \rho(\beta, t)\sigma_{t,i}^2$, the latter being due to Eq. (5.14). Hence, the relative variance of $m_t$ is $\rho(\beta, t)$ times that of $g_t$, such that the

[4] Our use of the term *momentum* is somewhat colloquial. To highlight the relationship with ADAM (Fig. 5.1), we have defined M-SGD as the method using the update direction $m_t$, which is a rescaled version of SGD with momentum. M-SVAG applies variance adaptation to $m_t$. This is not to be confused with the application of momentum acceleration [Polyak, 1964, Nesterov, 1983] *on top* of a SVAG update.

optimal variance adaptation factors for the update direction $m_t$ according to Lemma 5.1 are

$$\gamma_{t,i}^m = \frac{1}{1 + \rho(\beta, t)\sigma_{t,i}^2 / \nabla f_{t,i}^2}. \tag{5.18}$$

We use the following estimate thereof:

$$\hat{\gamma}_t^m = \frac{1}{1 + \rho(\beta, t)\, \hat{s}_t / m_t^2} = \frac{m_t^2}{m_t^2 + \rho(\beta, t)\, \hat{s}_t}. \tag{5.19}$$

Note that $m_t$ now serves a double purpose: It determines the base update direction and, at the same time, is used to obtain an estimate of the gradient variance.

### 5.4.4 *Details*

Note that Eq. (5.16) is ill-defined for $t = 0$, since $\rho(\beta, 0) = 0$. We use $\hat{s}_0 = 0$ for the first iteration, resulting in an initial SGD-step. One final detail concerns a possible division by zero in Eq. (5.19). Unlike ADAM, we do not add a constant offset $\varepsilon$ in the denominator. A division by zero only occurs when $m_{t,i} = v_{t,i} = 0$; we check for this case and perform no update in the respective coordinates.

This completes the description of our implementation of M-SVAG. Alg. 2 provides pseudo-code (ignoring the details discussed in §5.4.4 for readability).

**Input:** $\theta_0 \in \mathbb{R}^d$, $\alpha > 0$, $\beta \in [0, 1]$, $T \in \mathbb{N}$
Initialize $\theta \leftarrow \theta_0$, $\tilde{m} \leftarrow 0$, $\tilde{v} \leftarrow 0$
**for** $t = 0, \dots, T-1$ **do**
  $\tilde{m} \leftarrow \beta\tilde{m} + (1-\beta)g(\theta), \quad \tilde{v} \leftarrow \beta\tilde{v} + (1-\beta)g(\theta)^2$
  $m \leftarrow (1-\beta^{t+1})^{-1}\tilde{m}, \quad v \leftarrow (1-\beta^{t+1})^{-1}\tilde{v}$
  $s \leftarrow (1-\rho(\beta,t))^{-1}(v - m^2)$
  $\gamma \leftarrow m^2/(m^2 + \rho(\beta,t)s)$
  $\theta \leftarrow \theta - \alpha(\gamma \odot m)$
**end for**

**Algorithm 2:** M-SVAG

## 5.5 *Connection to Generalization*

In deep learning, different solutions with comparable training loss can have significantly varying generalization performance. Recently, the question of the effect of the optimization algorithm on *generalization* has received increased attention. In particular, Wilson et al. [2017] have argued that "adaptive methods" (referring to AdaGrad, RMSProp, and ADAM) have adverse effects on generalization compared to "non-adaptive methods" (gradient descent, SGD, and their momentum variants). In addition to an extensive empirical validation of that claim, the authors make a theoretical

argument using a binary least-squares classification problem,

$$R(\theta) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(x_i^\mathsf{T}\theta - y_i)^2 = \frac{1}{2n}\|X\theta - y\|^2, \tag{5.20}$$

with $n$ data points $(x_i, y_i) \in \mathbb{R}^d \times \{\pm 1\}$, stacked in a matrix $X \in \mathbb{R}^{n\times d}$ and a label vector $y \in \{\pm 1\}^n$. For this problem class, the non-adaptive methods provably converge to the max-margin solution, which we expect to have favorable generalization properties. In contrast to that, Wilson et al. [2017] show that—for *some* instances of this problem class—the adaptive methods converge to solutions with arbitrarily bad generalization to unseen data. The authors construct such problematic instances using the following Lemma.

**Lemma 5.3** (Lemma 3.1 in Wilson et al. [2017]). *Suppose* $[X^\mathsf{T}y]_i \neq 0$ *for* $i = 1, \dots, d$, *and there exists* $c \in \mathbb{R}$ *such that* $X\operatorname{sign}(X^\mathsf{T}y) = cy$. *Then, when initialized at* $\theta_0 = 0$, *the iterates generated by full-batch* ADAGRAD, ADAM, *and* RMSPROP *on the objective* (5.20) *satisfy* $\theta_t \propto \operatorname{sign}(X^\mathsf{T}y)$.

Intriguingly, as we show in Appendix B.2.5, this statement easily extends to sign descent, i.e., the method updating $\theta_{t+1} = \theta_t - \alpha\operatorname{sign}(\nabla R(\theta_t))$.

**Lemma 5.4.** *Under the assumptions of Lemma 5.3, the iterates generated by sign descent satisfy* $\theta_t \propto \operatorname{sign}(X^\mathsf{T}y)$.

On the other hand, this does *not* extend to M-SVAG,[5] which is an *adaptive* method by any standard. While this does by no means imply that it converges to the max-margin solution or has otherwise favorable generalization properties, the construction of Wilson et al. [2017] does *not* apply to M-SVAG.

[5] This follows easily from the fact that the first step of M-SVAG coincides with a gradient descent step.

This suggests that it is the *sign* aspect that impedes generalization in the examples constructed by Wilson et al. [2017], rather than the elementwise adaptivity as such. Our experiments substantiate this suspicion. The fact that all currently popular adaptive methods are also sign-based has led to a conflation of these two aspects. The main motivation for this work was to disentangle them.

## 5.6 Experiments

We experimentally compare M-SVAG and ADAM to their non-variance-adapted counterparts M-SGD and M-SIGN-SGD (Alg. 3). Since these are the four possible recombinations of the sign and the variance adaptation (Fig. 5.1), this comparison allows us to separate the effects of the two aspects.

### 5.6.1 Experimental Set-Up

We evaluated the four methods on the following problems:

P1 A vanilla convolutional neural network (CNN) with two convolutional and two fully-connected layers on the Fashion-MNIST dataset [Xiao et al., 2017].

**Input:** $\theta_0 \in \mathbb{R}^d$, $\alpha > 0$, $\beta \in [0, 1]$, $T \in \mathbb{N}$
Initialize $\theta \leftarrow \theta_0$, $\tilde{m} \leftarrow 0$
**for** $t = 0, \ldots, T-1$ **do**
  $\tilde{m} \leftarrow \beta\tilde{m} + (1-\beta)g(\theta)$
  $\boxed{m \leftarrow (1-\beta^{t+1})^{-1}\tilde{m}}$
  $\boxed{\theta \leftarrow \theta - \alpha m}$ $\theta \leftarrow \theta - \alpha\,\mathrm{sign}(\tilde{m})$
**end for**

**Algorithm 3:** M-SGD and M-SIGN-SGD

P2 A vanilla CNN with three convolutional and three fully-connected layers on CIFAR-10 [Krizhevsky, 2009].

P3 The wide residual network WRN-40-4 architecture of Zagoruyko and Komodakis [2016] on CIFAR-100.

P4 A two-layer LSTM [Hochreiter and Schmidhuber, 1997] for character-level language modelling on Tolstoy's *War and Peace*.

A detailed description of all network architectures has been moved to Section B.1 of the Appendix.

For all experiments, we used $\beta = 0.9$ for M-SGD, M-SSD and M-SVAG and default parameters ($\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$) for ADAM. The global step size $\alpha$ was tuned for each method individually by first finding the maximal stable step size by trial and error, then searching downwards. We selected the one that yielded maximal test accuracy within a fixed number of training steps; a scenario close to an actual application of the methods by a practitioner. (Loss and accuracy have been evaluated at a fixed interval on the full test set as well as on an equally-sized portion of the training set). Experiments with the best step size have been replicated ten times with different random seeds. While (P1) and (P2) were trained with constant $\alpha$, we used a decreasing schedule for (P3) and (P4), which was fixed in advance for all methods. Full details can be found in Appendix B.1.

### 5.6.2 *Results*

The results are depicted in Fig. 5.4. We make four main observations.

*1) The sign aspect dominates.* With the exception of (P4), the performance of the four methods distinctly clusters into sign-based and non-sign-based methods. Of the two components of ADAM identified in §5.1.1, the sign aspect seems to be by far the dominant one, accounting for most of the difference between ADAM and M-SGD. ADAM and M-SIGN-SGD display surprisingly similar performance; an observation that might inform practitioners' choice of algorithm, especially for very high-dimensional problems, where ADAM's additional memory requirements are an issue.

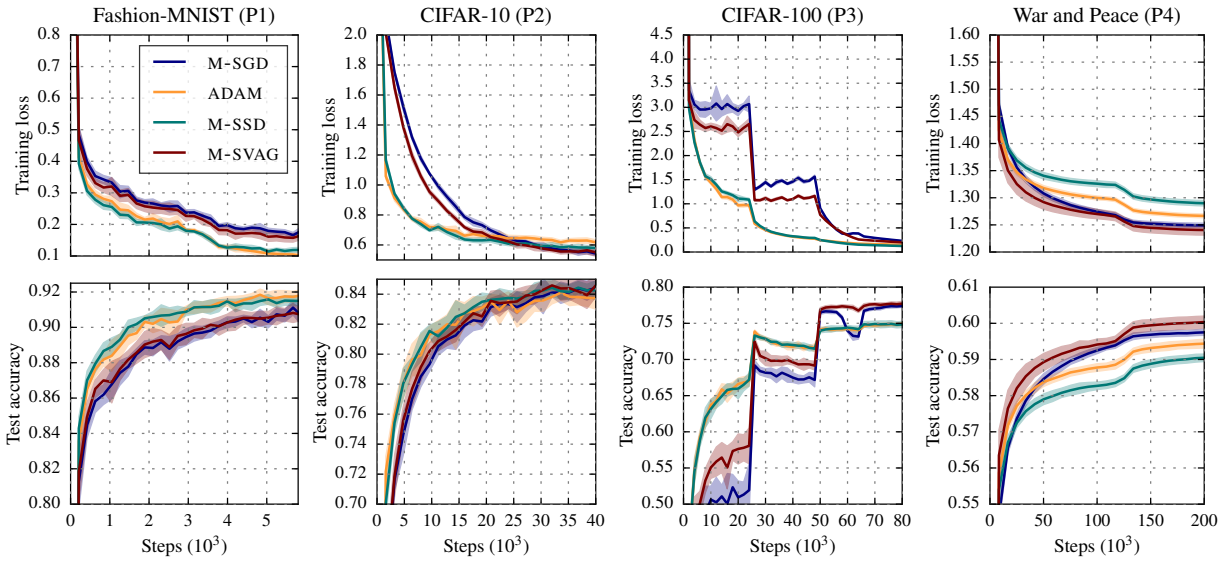


Figure 5.4: Experimental results on the four test problems. Plots display training loss and test accuracy over the number of steps. Curves for the different optimization methods are color-coded. The shaded area spans one standard deviation, obtained from ten replications with different random seeds. M-SIGN-SGD is referred to as M-SSD in the legend due to a difference in terminology between the original paper and this manuscript.

*2) The usefulness of the sign is problem-dependent.* Considering only training loss, the two sign-based methods clearly outperform the two non-sign-based methods on problems (P1) and (P3). On (P2), ADAM and M-SIGN-SGD make rapid initial progress, but later plateau and are undercut by M-SGD and M-SVAG. On the language modelling task (P4) the non-sign-based methods show superior performance. This shows that the usefulness of sign-based methods depends on the particular problem at hand.

*3) Variance adaptation helps.* In all experiments, the variance-adapted variants perform at least as good as, and often better than, their "base algorithms". The magnitude of the effect varies. For example, ADAM and M-SIGN-SGD have identical performance on (P3), but M-SVAG significantly outperforms M-SGD on (P3) as well as (P4).

*4) Generalization effects are caused by the sign.* The CIFAR-100 example (P3) displays similar effects as reported by Wilson et al. [2017]: ADAM vastly outperforms M-SGD in training loss, but has significantly worse test performance. Observe that M-SIGN-SGD behaves almost identical to ADAM in both train and test and, thus, displays the same generalization-harming effects. M-SVAG, on the other hand, improves upon M-SGD and, in particular, does not display any adverse effects on generalization. This corroborates the suspicion raised in §5.5 that the generalization-harming effects of ADAM are caused by the sign aspect rather than the elementwise adaptive step sizes.

## 5.7 Conclusion

We have argued that ADAM combines two components: taking signs and applying variance adaptation. Our experiments show

that the sign aspect is by far the dominant one, but its usefulness is problem-dependent. Chapter 6 will discuss the sign aspect in more detail and, in short, will show that the usefuless of sign gradient descent depends on the conditioning of the problem as well as its *axis-alignment*. Sign-based methods also seem to have an adverse effect on the generalization performance of the obtained solution; a possible starting point for further research into the generalization effects of optimization algorithms.

The second aspect, variance adaptation, is not restricted to ADAM but can be applied to any update direction. We have provided a general motivation for variance adaptation factors that is independent of the update direction. In particular, we introduced M-SVAG, a variance-adapted variant of momentum SGD, which is a useful addition to the practitioner's toolbox for problems where sign-based methods like ADAM fail. A TensorFlow [Abadi et al., 2015] implementation can be found at `https://github.com/lballes/msvag`.

# 6
# *The Geometry of Sign Gradient Descent*

In Chapter 5 we have seen that the exceedingly popular ADAM method is closely related to SIGN-SGD. Beyond this connection, sign-based optimization methods themselves have become popular in machine learning due to their favorable communication cost in distributed optimization and their surprisingly good performance in neural network training. In this chapter, we review and expand the analysis of sign-based optimization methods. Recent works analyzing SIGN-SGD have used a non-standard "separable smoothness" assumption, whereas some older works study sign gradient descent as steepest descent with respect to the $\ell_\infty$-norm. In this work, we unify these existing results by showing a close connection between separable smoothness and $\ell_\infty$-smoothness and argue that the latter is the weaker and more natural assumption. We then proceed to study the smoothness constant with respect to the $\ell_\infty$-norm and thereby isolate geometric properties of the objective function which affect the performance of sign-based methods. In short, we find sign-based methods to be preferable over gradient descent if (i) the Hessian is to some degree concentrated on its diagonal, and (ii) its maximal eigenvalue is much larger than the average eigenvalue. Both properties are common in deep networks.

The contents of this chapter are based on the preprint:

Lukas Balles, Fabian Pedregosa, and Nicolas Le Roux. The geometry of sign gradient descent. *arXiv preprint arXiv:2002.08056*, 2020

Coauthor contributions:

| | Sc. Ideas | Experiments | Interpretation | Writing |
|---|---|---|---|---|
| **Lukas Balles** | 60% | 70% | 60% | 60% |
| Fabian Pedregosa | 20% | 30% | 20% | 10% |
| Nicolas Le Roux | 20% | 0% | 20% | 30% |

## 6.1 Introduction

Several recent works have considered the sign gradient descent (SIGN-GD) method and its stochastic counterpart (SIGN-SGD)

$$\theta_{t+1} = \theta_t - \alpha_t \operatorname{sign}(\nabla f_t), \tag{6.1}$$

$$\theta_{t+1} = \theta_t - \alpha_t \operatorname{sign}(g_t), \tag{6.2}$$

where the sign is applied elementwise. In particular, these methods have been studied in the context of distributed optimization where they conveniently reduce the communication cost to a single bit per gradient coordinate [e.g., Seide et al., 2014, Bernstein et al., 2018, Karimireddy et al., 2019]. SIGN-SGD is also of interest due to a connection to the popular ADAM method [Kingma and Ba, 2015] as discussed in Chapter 5.

*Analysis of Sign-Based Methods* Multiple authors [Kelner et al., 2014, Carlson et al., 2015, Karimi et al., 2016] have analyzed variants of sign-based methods under the assumption of smoothness with respect to the $\ell_\infty$-norm (maximum norm), i.e.,

$$\|\nabla f(\theta') - \nabla f(\theta)\|_1 \leq L_\infty \|\theta' - \theta\|_\infty \tag{6.3}$$

for all $\theta, \theta' \in \mathbb{R}^d$ with smoothness constant $L_\infty > 0$. On the other hand, Bernstein et al. [2018] have analyzed SIGN-SGD under a non-standard smoothness assumption that there are constants $l_1 \ldots, l_d > 0$ such that

$$f(\theta') \leq f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{1}{2} \sum_i l_i (\theta'_i - \theta_i)^2 \tag{6.4}$$

for all $\theta, \theta' \in \mathbb{R}^d$. (In a slight overload of notation, $\theta_i$ refers to the $i$-th dimension of the vector $\theta$ and *not* to the $i$-th iterate of an optimization procedure.) Follow-up works refining the analysis also adopted this assumption [Bernstein et al., 2019, Safaryan and Richtárik, 2019], which we will refer to as *separable* smoothness to emphasize that the quadratic term *separates* over coordinates with individual constants $l_i$.

Reiterating all convergence results from these works would be beyond the scope of this chapter, but it is crucial to understand that the convergence rates of sign-based methods are governed by $L_\infty$ in the papers based on $\ell_\infty$-smoothness and by $\sum_i l_i$ in those based on the separable smoothness assumption.

*Contributions* The separable smoothness assumption seems to add an unnecessary level of granularity since neither the algorithm itself nor its analysis uses the individual values $l_i$ but only their sum. This work clarifies the relationship between separable smoothness and $\ell_\infty$-smoothness. We show that the convergence results based on separable smoothness also hold under $\ell_\infty$-smoothness if $L_\infty = \sum_i l_i$ and that the latter is a strictly weaker assumption. This unifies all

existing results on sign-based methods under the umbrella of $\ell_\infty$-smoothness.

We then proceed to analyze the geometric meaning of $\ell_\infty$-smoothness. We tie the corresponding smoothness constant $L_\infty$ to properties of the Hessian and show that it is favorable if the Hessian fulfills two conditions: (i) some degree of "diagonal concentration" and (ii) the maximal eigenvalue being much larger than the average eigenvalue. Notably, these properties have repeatedly been observed in deep learning training tasks. Our analysis thus provides a possible explanation of the empirical success of sign-based methods—and, by extension, ADAM—in deep learning. The dependence on the diagonal concentration of the Hessian, which relates to the axis alignment of the objective, is in stark contrast to the Euclidean smoothness constant $L_2$, which controls the convergence speed of (stochastic) gradient descent.

## 6.2 Smoothness and Steepest Descent

Before proceeding, we briefly review the concept of smoothness with respect to arbitrary norms and the associated steepest descent methods, which will be crucial the remainder of this work.

### 6.2.1 Smoothness w.r.t. Arbitrary Norms

Smoothness is a standard assumption in optimization and means that the gradient function is Lipschitz, i.e., $\|\nabla f(\theta') - \nabla f(\theta)\|_2 \leq L_2\|\theta' - \theta\|_2$ for some positive scalar $L_2$. The crucial significance of this assumption is that it gives rise to local quadratic bounds on $f$:

$$f(\theta') \leq f(\theta') + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{L_2}{2}\|\theta' - \theta\|_2^2. \tag{6.5}$$

This bound motivates gradient descent; fixing $\theta$ and minimizing w.r.t. $\theta'$ yields the update $\theta' = \theta - L_2^{-1}\nabla f(\theta)$.

This notion of smoothness can be generalized to arbitrary norms. We say $f$ is $L$-smooth w.r.t. a norm $\|\cdot\|$ if

$$\|\nabla f(\theta') - \nabla f(\theta)\|_* \leq L\|\theta' - \theta\| \tag{6.6}$$

for all $\theta, \theta' \in \mathbb{R}^d$. Here, $\|\cdot\|_*$ denotes the dual norm of $\|\cdot\|$, defined as $\|\omega\|_* := \max_{\|\theta\|\leq 1}\langle \omega, \theta \rangle$. Table 6.1 lists the dual norm pairs for the methods under consideration in this chapter. The standard case of Euclidean smoothness falls under this definition since the Euclidean norm is dual to itself.

Due to the equivalence of norms on $\mathbb{R}^d$, a function that is smooth with respect to one norm is also smooth with respect to any other norm. However, the tightest possible smoothness constant,

$$L \stackrel{\text{def}}{=} \sup_{\theta \neq \theta'} \frac{\|\nabla f(\theta') - \nabla f(\theta)\|_*}{\|\theta' - \theta\|}, \tag{6.7}$$

will depend on the choice of norm. In the following, we will always assume $L$ to be given by Eq. (6.7). This constant governs the conver-

| Method | Norm | Dual | Update direction |
|---|---|---|---|
| Gradient descent | $\|\cdot\|_2$ | $\|\cdot\|_2$ | $\nabla f$ |
| Sign gradient descent | $\|\cdot\|_\infty$ | $\|\cdot\|_1$ | $\|\nabla f\|_1 \operatorname{sign}(\nabla f)$ |
| Coordinate descent | $\|\cdot\|_1$ | $\|\cdot\|_\infty$ | $\nabla f_{i_{\max}} e^{(i_{\max})}$ |
| Block-normalized GD | $\|\cdot\|_\infty^{\mathcal{B}}$ | $\|\cdot\|_1^{\mathcal{B}}$ | see Appendix C.1 |

Table 6.1: A few steepest descent methods. The table lists the used norm $\|\cdot\|$, its dual $\|\cdot\|_*$, and the resulting update direction. Coordinate descent and block-normalized gradient descent are discussed in Appendix C.1.

gence speed of the corresponding steepest descent method, which we will define next.

### 6.2.2 *Steepest Descent*

As in the Euclidean case, smoothness gives rise to a local quadratic bound on the function around a point $\theta$:

**Lemma 6.1.** *If $f$ is $L$-smooth w.r.t. $\|\cdot\|$, then*

$$f(\theta') \leq f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{L}{2}\|\theta' - \theta\|^2 \tag{6.8}$$

*for all $\theta, \theta' \in \mathbb{R}^d$.*

*Steepest descent* with respect to the norm $\|\cdot\|$ iteratively minimizes this upper bound:

$$\theta_{t+1} \in \arg\min_{\theta \in \mathbb{R}^d} \left( \langle \nabla f_t, \theta - \theta_t \rangle + \frac{L}{2}\|\theta - \theta_t\|^2 \right). \tag{6.9}$$

This minimizer need not be unique, in which case steepest descent is to be understood as choosing *any* solution.

We have already seen that gradient descent is steepest descent with respect to the Euclidean norm. As noted by Kelner et al. [2014] and Carlson et al. [2015], steepest descent with respect to the maximum norm gives rise to a version of sign gradient descent, namely

$$\theta_{t+1} = \theta_t - \frac{1}{L_\infty}\|\nabla f_t\|_1 \operatorname{sign}(\nabla f_t). \tag{6.10}$$

This is equivalent to Eq. (6.1) up to the scaling with the gradient norm which may be subsumed in the step size.

*Side note* The steepest descent framework encompasses many other well-known methods, see Table 6.1. For example, steepest descent w.r.t. the $\ell_1$-norm yields a version of coordinate descent. An interesting observation is that a block-wise maximum norm gives rise to a "block-normalized" gradient descent, which may be seen as a block-wise extension of sign gradient descent. Variants of block-wise normalization have recently found application in deep learning [Yu et al., 2017, Ginsburg et al., 2019] with blocks corresponding to layers. We discuss this further in Appendix C.1.

### 6.2.3 *Convergence of Steepest Descent*

The convergence of steepest descent methods bases upon the following Lemma, which guarantees an improvement in function value in each step.

**Lemma 6.2.** *Let $f$ be $L$-smooth w.r.t. $\|\cdot\|$. Then steepest descent (Eq. 6.9) satisfies*

$$f(\theta_{t+1}) \leq f(\theta_t) - \frac{1}{2L}\|\nabla f(\theta_t)\|_*^2. \tag{6.11}$$

This implies various convergence results, which we discuss in Appendix C.1. Generally, all steepest descent methods will enjoy the same rate of convergence, but Lemma 6.2 shows the significance of (i) the smoothness constant, which we want to be small, and (ii) the dual gradient norm, which we want to be large. These two aspects will play a role when we compare sign gradient descent and gradient descent in Section 6.5.

## 6.3 Separable Smoothness and $\ell_\infty$-Smoothness

This section clarifies the relationship between separable smoothness and $\ell_\infty$-smoothness in order to unify existing convergence results.

### 6.3.1 $\ell_\infty$-Smoothness Can Replace Separable Smoothness

We now show that $\ell_\infty$-smoothness can replace separable smoothness for the analysis of sign-based methods by showing that

(i) separable smoothness with constants $l_1, \ldots, l_d$ implies $\ell_\infty$-smoothness with constant $L_\infty = \sum_i l_i$, and

(ii) convergence results based on separable smoothness also hold under the latter, weaker assumption.

While separable smoothness is directly defined by Eq. (6.4) in existing works, it is easily embedded in the framework of Section 6.2 as 1-smoothness w.r.t. the norm $\|\cdot\|_{\mathbf{L}}$ where $\mathbf{L} = \operatorname{diag}(l_1, \ldots, l_d)$ and $\|\omega\|_{\mathbf{L}}^2 := \left(\sum_i l_i \omega_i^2\right)$. The bound given by Lemma 6.1 then coincides with Eq. (6.4). With that, we can establish statement (i).

**Proposition 6.1.** *If $f$ is 1-smooth w.r.t. $\|\cdot\|_{\mathbf{L}}$, then $f$ is $(\sum_i l_i)$-smooth w.r.t. the maximum norm.*

Regarding statement (ii), we note that the separable smoothness assumption in form of Eq. (6.4) enters existing convergence proofs exclusively with $\theta = \theta_t, \theta' = \theta_{t+1}$. The simple but essential observation is that, since sign-based updates have the same magnitude in each coordinate, $(\sum_i l_i)$-smoothness w.r.t. the maximum norm yields the exact same bound.

**Proposition 6.2.** *Let $\theta \in \mathbb{R}^d$, $\delta \in \{-1, 1\}^d$ and $\alpha > 0$. Both separable smoothness with constants $l_1, \ldots, l_d$ and $\ell_\infty$-smoothness with constant $L_\infty = \sum_i l_i$ imply*

$$f(\theta + \alpha\delta) \leq f(\theta) + \alpha\langle \nabla f(\theta), \delta\rangle + \frac{\alpha^2}{2}\sum_i l_i. \tag{6.12}$$

Since all existing convergence results start from exactly this bound they hold under either of the two assumptions.

In summary, separable smoothness adds a level of granularity that sign-based algorithms cannot exploit and that is not necessary for their analysis. Max-norm smoothness removes this unnecessary granularity, is a strictly weaker assumption, and is naturally tied to sign-based methods via the steepest descent formalism. We thus argue that $\ell_\infty$-smoothness should be used for the analysis of sign-based methods.

### 6.3.2 *Consequences of the Unification*

The results established in the previous subsection relax the assumptions of some previous works and allow for a better comparability of existing results. We can adapt results based on separable smoothness to use $\ell_\infty$-smoothness simply by replacing $\sum_i l_i$ with $L_\infty$. As an example, Theorem 1 of Bernstein et al. [2018] shows convergence to a stationary point for SIGN-SGD on non-convex problems in the "large-batch setting", where a mini-batch size of $T$ is used to perform $T$ iterations. Adapted to $\ell_\infty$-smoothness, the rate reads

$$\mathbf{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f_t\|_1\right]^2 \leq \frac{1}{T}\left[\sqrt{L_\infty}\left(f_0 - f^\star + \frac{1}{2}\right) + \text{const}\right]^2. \tag{6.13}$$

Bernstein et al. [2018] contrast this with a rate for SGD achieved under similar assumptions, namely

$$\mathbf{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f_t\|_2^2\right] \leq \frac{1}{T}\left[2L_2\left(f_0 - f^\star\right) + \text{const}\right]. \tag{6.14}$$

For details refer to Bernstein et al. [2018].

This can now easily be compared with other results, for example, those arising from the steepest descent framework. In the deterministic setting, norm-scaled sign gradient descent (Eq. 6.10) achieves a non-convex rate of

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f_t\|_1^2 \leq \frac{2L_\infty(f_0 - f^\star)}{T}, \tag{6.15}$$

whereas gradient descent achieves

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f_t\|_2^2 \leq \frac{2L_2(f_0 - f^\star)}{T}, \tag{6.16}$$

see Proposition C.2 in Appendix C.1.

Our unification makes clear that the max-norm smoothness constant $L_\infty$ crucially affects the convergence speed of sign-based methods, irrespective of the setting (stochastic vs. deterministic) and the precise version (Eq. 6.1 vs. Eq. 6.10). It is thus critical to establish a better understanding of the geometric meaning of $\ell_\infty$-smoothness, which we will tackle in the next section.

## 6.4 Understanding $\ell_\infty$-Smoothness

The previous sections have established the significance of smoothness w.r.t. the maximum norm for sign-based optimization methods, showing that the corresponding smoothness constant $L_\infty$ crucially affects their convergence speed. We now turn our attention to the "meaning" of this constant. While we have a good intuition for the Euclidean smoothness constant—an upper bound on the eigenvalues of the Hessian—this is lacking for smoothness w.r.t. the maximum norm. Our goal in this section is to understand which properties of the objective function affect the constant $L_\infty$. We first introduce a Hessian-based formulation of general smoothness constants, generalizing the aforementioned result for Euclidean smoothness. We then show that $L_\infty$ depends on both the eigenvalues of the Hessian as well as its degree of "diagonal concentration", which corresponds to the axis alignment of the objective. Next, we contrast this more explicitly with Euclidean smoothness, pinpointing conditions under which the objective function has a favorable $L_\infty$ constant *relative* to $L_2$, suggesting a susceptibility to sign-based optimization methods. Finally, we discuss how these insights relate back to the separable smoothness condition.

### 6.4.1 Smoothness as a Bound on the Hessian

The definition of smoothness in Eq. (6.6) is unwieldy. For the Euclidean norm, a more intuitive characterization is widely known: A twice-differentiable function $f$ is $L_2$-smooth w.r.t. the Euclidean norm if and only if $\|\nabla^2 f(\theta)\|_2 \leq L_2$ for all $\theta \in \mathbb{R}^d$. Here, $\|\cdot\|_2$ for matrices denotes the spectral norm, given by the largest-magnitude eigenvalue for symmetric matrices. To facilitate our discussion of the $\ell_\infty$-smoothness constant, the following proposition generalizes this Hessian-based formulation of smoothness constants. It shows that smoothness with respect to any other norm likewise arises from a bound on the Hessian, but in different matrix norms.

**Proposition 6.3.** *For any norm $\|\cdot\|$ on $\mathbb{R}^d$, we define the matrix norm induced by $\|\cdot\|$ (also denoted by $\|\cdot\|$) as*

$$\|\mathbf{H}\| \overset{def}{=} \max_{\|\theta\| \leq 1} \|\mathbf{H}\theta\|_* . \tag{6.17}$$

*A twice-differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is $L$-smooth w.r.t. $\|\cdot\|$ if and only if $\|\nabla^2 f(\theta)\| \leq L$ for all $\theta$.*

We are not aware of prior works showing this for the general case. For the Euclidean norm, Proposition 6.3 gives back the familiar spectral norm bound.

### 6.4.2 $L_\infty$ is Sensitive to Axis-Alignment

By Proposition 6.3, $L_\infty$ is determined by

$$L_\infty = \sup_{\theta \in \mathbb{R}^d} \|\nabla^2 f(\theta)\|_{\infty,1}, \quad \|\mathbf{H}\|_{\infty,1} \overset{\text{def}}{=} \max_{\|\theta\|_\infty \leq 1} \|\mathbf{H}\theta\|_1. \tag{6.18}$$

Unfortunately, computing $\|\mathbf{H}\|_{\infty,1}$ is NP-hard [Rohn, 2000], but we can gain some intuition on a two-dimensional quadratic where all relevant quantities are easily found in closed form (see Appendix C.2 for details). As depicted in Fig. 6.1, $L_\infty$ does not only depend on the eigenvalues, but also on the axis alignment of the objective, which is encoded in the diagonal concentration of the Hessian.

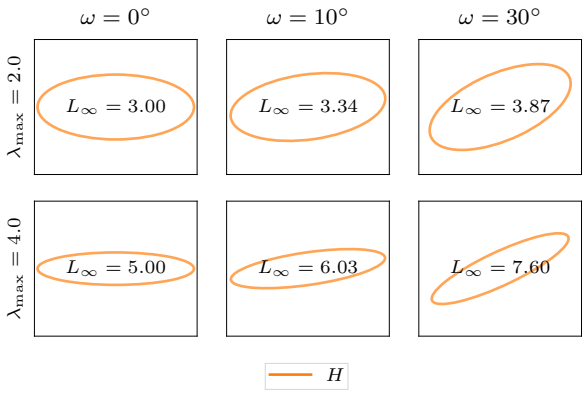


Figure 6.1: For $\mathbf{H} \in \mathbb{R}^{2\times 2}$, we plot a contour line of $f(\theta) = \frac{1}{2}\theta^\mathsf{T}\mathbf{H}\theta$, which forms an ellipse with principal axes given by the eigenvectors of $\mathbf{H}$ and axis lengths given by the inverse eigenvalues. We fix $\lambda_{\min} = 1$ and vary $\lambda_{\max} > 1$ as well as the angle $\omega$ between the principal axes of the ellipse and the coordinate axes. (Mathematical details can be found in Appendix C.2.) The $\ell_\infty$-smoothness constant $L_\infty = \|\mathbf{H}\|_{\infty,1}$ is sensitive to the axis alignment of the objective. This is in contrast to the Euclidean smoothness constant, which is simply given by $L_2 = \lambda_{\max}$.

We can generalize this insight through the following upper bound.

**Proposition 6.4.** *Let $\mathbf{H} \in \mathbb{R}^{d\times d}$ be nonzero, positive semi-definite with eigenvalues $\lambda_1, \ldots, \lambda_d$. Then*

$$\|\mathbf{H}\|_{\infty,1} \leq \rho_{diag}(\mathbf{H})^{-1} \sum_{i=1}^{d} \lambda_i \tag{6.19}$$

*with $\rho_{diag}(\mathbf{H}) := \sum_i |H_{ii}| / \sum_{i,j} |H_{ij}|$.*

The quantity $\rho_{\text{diag}}(\mathbf{H}) \in [d^{-1}, 1]$ measures the degree of diagonal concentration. Proposition 6.4 thus indicates that $L_\infty$ will depend both on the axis alignment of the Hessian as well as its eigenvalues. This is in stark contrast to Euclidean smoothness, for which the relevant matrix norm is invariant to rotations. We will make this comparison more explicit in the next subsection.

Proposition 6.4 only applies to positive semi-definite matrices. We provide a similar, albeit slightly looser, bound for all symmetric matrices:

**Proposition 6.5.** *Let $\mathbf{H} \in \mathbb{R}^{d\times d}$ be symmetric with eigenvalues $\lambda_1, \ldots, \lambda_d$ and orthonormal eigenvectors $v^{(1)}, \ldots, v^{(d)}$. Then*

$$\|\mathbf{H}\|_{\infty,1} \leq \sum_{i=1}^{d} |\lambda_i| \, \|v^{(i)}\|_1^2 \,. \tag{6.20}$$

In this bound, the dependency on the axis alignment appears through $\|v^{(i)}\|_1^2$. Since the eigenvectors have unit $\ell_2$-norm, their squared $\ell_1$-norm is bounded by $1 \leq \|v^{(i)}\|_1^2 \leq d$. The lower bound is attained if $v^{(i)}$ is perfectly axis-aligned. The upper bound is attained if all elements of $v^{(i)}$ are of equal magnitude. Due to the

weighting by $|\lambda_i|$, we see that the axis alignment particularly matters for eigenvectors associated with eigenvalues of large magnitude. Both bounds are tight for axis-aligned matrices but can be substantially loose for very non-axis-aligned ones.

These *upper* bounds on $\|\mathbf{H}\|_{\infty,1}$ identify *sufficient* conditions which favor sign gradient descent. We note that the sensitivity to axis alignment also manifests itself in a lower bound. Indeed, for any pair $(\lambda_i, v^{(i)})$ we have

$$\|\mathbf{H}\|_{\infty,1} \geq \frac{\|\mathbf{H}v^{(i)}\|_1}{\|v^{(i)}\|_\infty} = |\lambda_i| \frac{\|v^{(i)}\|_1}{\|v^{(i)}\|_\infty}. \tag{6.21}$$

Again, the ratio $\|v^{(i)}\|_1 / \|v^{(i)}\|_\infty \in [1, d]$ can be seen as a measure of axis alignment of $v^{(i)}$.

### 6.4.3 *Comparing $L_\infty$ and $L_2$*

We have seen that the $\ell_\infty$-smoothness constant governs the convergence of sign gradient descent (and related methods) whereas the Euclidean smoothness constant governs the convergence of gradient descent. We now want to compare these two constants. Assume $f$ to be $L_2$-smooth w.r.t. $\|\cdot\|_2$ and $L_\infty$-smooth w.r.t. $\|\cdot\|_\infty$ in the "tight" sense of Eq. (6.7). Basic inequalities between the norms yield (see Appendix C.5)

$$L_2 \leq L_\infty \leq dL_2 \tag{6.22}$$

irrespective of $f$. The smoothness constant governing the convergence speed of sign gradient descent will always be larger (worse) than the one pertinent to gradient descent. This does *not* mean that SIGN-GD is always worse than GD. The smoothness constant is only one factor influencing the convergence speed; Lemma 6.2 shows that the dual gradient norm plays a role as well. As we will discuss in Section 6.5, this norm is larger for SIGN-GD which can make up for the disadvantage of a larger smoothness constant.

Here, we want to characterize under which circumstances $L_\infty$ *tends* to be small—in particular much smaller than its worst case of $dL_2$—such that sign-based methods can be competitive with gradient descent. Propositons 6.4 and 6.5 imply the following two conditions:

1. We need the Hessian to exhibit some degree of diagonal concentration or, equivalently, to have somewhat axis-aligned eigenvectors.

2. The sum of absolute eigenvalues should be much smaller than its worst case of $d\lambda_{\max}$ or, equivalently, the average absolute eigenvalue should be much smaller than the maximal one:

$$\bar{\lambda} \stackrel{\text{def}}{=} \frac{1}{d} \sum_i |\lambda_i| \ll \max_i |\lambda_i| =: \lambda_{\max}. \tag{6.23}$$

Notably, both properties have independently been found to be present in neural network training. Multiple papers studying the Hessian in neural network training objectives [Chaudhari et al., 2017, Papyan, 2018, Ghorbani et al., 2019, Li et al., 2020] find that the spectrum is dominated by a small number of *outlier eigenvalues* that far exceed the bulk of eigenvalues, which are concentrated close to zero. This is exactly the setting that will lead to $\bar{\lambda} \ll \lambda_{\max}$. The question of the diagonal concentration of the Hessian has been studied as early as 1988 by Becker and LeCun. More recently, Adolphs et al. [2019] have confirmed some degree of diagonal concentration in contemporary neural architectures. In light of our analysis above, these observations suggest that the geometry of optimization problems encountered in deep learning lends itself to sign-based optimization methods (and its relatives such as ADAM).

### 6.4.4 *Separable Smoothness in the Hessian-Based View*

A natural question is how separable smoothness fits into this Hessian-based formulation and—given its close connection to $\ell_\infty$-smoothness—how it reflects the sensitivity to axis alignment. Bernstein et al. [2018] note that, for twice-differentiable $f$, separable smoothness results from a diagonal bound on the Hessian, i.e., $-\mathbf{L} \preceq \nabla^2 f(\theta) \preceq \mathbf{L}$ for all $x \in \mathbb{R}^d$ with $\mathbf{L} = \operatorname{diag}(l_1, \dots, l_d)$. It is tempting to think of the values $l_1, \dots, l_d$ merely as bounds on the eigenvalues of $\nabla^2 f(\theta)$, but we will see in the following that these values also depend on the axis alignment of the Hessian.

With "$\preceq$" being only a partial ordering there is no clear definition of the *tightest* diagonal bound. Since the performance of sign-based methods depends on $\sum_i l_i$, we will choose the bound which minimizes this sum. With that, we can establish the following result:

**Proposition 6.6.** *Let $\mathbf{H} \in \mathbb{R}^{d \times d}$ be positive semi-definite with eigenvalues $\lambda_1 \dots, \lambda_d$ and define*

$$L_\infty(\mathbf{H}) \overset{def}{=} \max_{\|\theta\|_\infty \leq 1} \|\mathbf{H}\theta\|_1, \tag{6.24}$$

$$L_{sep}(\mathbf{H}) \overset{def}{=} \min_{l_i \geq 0} \sum_i l_i \quad s.t. \quad \mathbf{H} \preceq \operatorname{diag}(l_i). \tag{6.25}$$

*Then*

$$L_\infty(\mathbf{H}) \leq L_{sep}(\mathbf{H}) \leq \rho_{diag}(\mathbf{H})^{-1} \sum_i \lambda_i. \tag{6.26}$$

Hence, $L_{\text{sep}}$ upper bounds $L_\infty$, reflecting again that $\ell_\infty$-smoothness is the weaker of the two conditions. At the same time, $L_{\text{sep}}$ is upper-bounded by the same quantity that appears in Proposition 6.4, reflecting the sensitivity to axis alignment. Figure 6.2 illustrates this on a two-dimensional quadratic example.

*Side note.* The two-dimensional example also reveals another shortcoming of the analysis based on separable smoothness. To contrast the performance of sign-based methods with that of

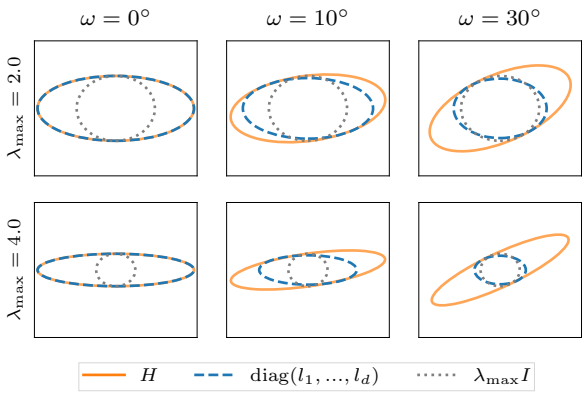


Figure 6.2: For $\mathbf{H} \in \mathbb{R}^{2\times 2}$, we plot a contour line of $f(\theta) = \frac{1}{2}\theta^\mathsf{T}\mathbf{H}\theta$, which forms an ellipse with principal axes given by the eigenvectors of $\mathbf{H}$ and axis lengths given by the inverse eigenvalues. We fix $\lambda_{\min} = 1$ and vary $\lambda_{\max} > 1$ as well as the angle $\omega$ between the principal axes of the ellipse and the coordinate axes. (Mathematical details can be found in Appendix C.2.) Separable smoothness bounds $\mathbf{H}$ by a diagonal matrix, $\mathrm{diag}(l_1, \ldots, l_d) \succeq \mathbf{H}$, corresponding to an axis-aligned ellipse that lies fully within the $\mathbf{H}$-ellipse. The "best" bounding ellipse is given by Eq. (6.25). This bound changes with the axis alignment, becoming both smaller and more circular (i.e., larger and more similar $l_i$) as we rotate further from the coordinate axes. In contrast to that, Euclidean smoothness bounds $\mathbf{H}$ by $\lambda_{\max}\mathbf{I} \succeq \mathbf{H}$, i.e., a rotation-invariant circle.

(stochastic) gradient descent, it is assumed that the convergence speed of the latter is controlled by $l_{\max}$, which implicitly assumes $l_{\max} = \lambda_{\max}$. This may be misleading since $l_{\max}$ can exceed $\lambda_{\max}$, see Appendix C.2. Of course, we could deviate from the definition in Eq. (6.25) and choose $l_i \equiv \lambda_{\max}$ to guarantee $l_{\max} = \lambda_{\max}$, but the resulting $L_{\text{sep}}$ would be very unfavorable for SIGN-SGD.

## 6.5 *Gradient Descent vs Sign Gradient Descent*

We found in Section 6.4 that $L_\infty$ always exceeds $L_2$ but identified conditions under which $L_\infty \ll dL_2$. In this section, we want to explicitly compare gradient descent and sign gradient descent. To facilitate this comparison, we make two restrictions. First, we consider the non-stochastic setting in order to isolate the dependency of the two methods on the *geometry* of the objective rather than the stochasticity. Second, we consider the norm-scaled version of sign gradient descent (Eq. 6.10). This variant is not usually applied to, say, deep learning tasks, for multiple possible reasons which we discuss in Appendix C.3. We argue that, in the smooth, non-stochastic setting, the norm-scaled version is natural and preferable since (i) it arises as steepest descent w.r.t. the maximum norm, and (ii) unlike the version in Eq. (6.1), it converges with a constant step size.

Hence, we are comparing two steepest descent methods. By Lemma 6.2, their guaranteed improvement in function value at $\theta \in \mathbb{R}^d$ is given by

$$\mathcal{I}_{\text{GD}}(\theta) \stackrel{\text{def}}{=} \frac{\|\nabla f(\theta)\|_2^2}{L_2}, \tag{6.27}$$

$$\mathcal{I}_{\text{SIGN-GD}}(\theta) \stackrel{\text{def}}{=} \frac{\|\nabla f(\theta)\|_1^2}{L_\infty}. \tag{6.28}$$

We now compare the two methods based on these improvement guarantees, thus taking into account the dual gradient norm in addition to the smoothness constant. Basic norm inequalities, $d\|\nabla f(\theta)\|_2^2 \geq \|\nabla f(\theta)\|_1^2 \geq \|\nabla f(\theta)\|_2^2$, show that this potentially favors sign gradient descent.

Following Bernstein et al. [2018] we define

$$\phi(\omega) \stackrel{\text{def}}{=} \frac{\|\omega\|_1^2}{d\|\omega\|_2^2} \in [d^{-1}, 1], \tag{6.29}$$

which measures the density of the vector $\omega$, i.e., how evenly its mass is distributed across coordinates. With that, we can write

$$\mathcal{R}(\theta) \stackrel{\text{def}}{=} \frac{\mathcal{I}_{\text{SIGN-GD}}(\theta)}{\mathcal{I}_{\text{GD}}(\theta)} = \phi(\nabla f(\theta)) \frac{dL_2}{L_\infty}. \tag{6.30}$$

If in addition to $L_\infty \ll dL_2$, for which we have identified conditions in Section 6.4, we encounter dense gradients, then $\mathcal{R}(\theta)$ is large and sign gradient descent makes faster progress than gradient descent.

We emphasize that this is a *local* comparison at $\theta \in \mathbb{R}^d$ due to the dependence on the gradient $\nabla f(\theta)$. It is tempting to try and lower-bound the gradient density—that is, to assume $\phi(\nabla f(\theta)) \geq \phi_g \gg d^{-1}$ for all $\theta$—in order to make global statements. Bernstein et al. [2018] assume such a lower bound when contrasting SIGN-GD and GD in their analysis. However, $\phi(\nabla f(\theta))$ can easily be shown to attain $d^{-1}$ even on quadratics. Any non-trivial lower bound would thus have to be restricted to the trajectory of the optimizer and take into account the initialization, which seems out of reach at the moment. The effect of the gradient norm thus remains an empirical question which we now address.

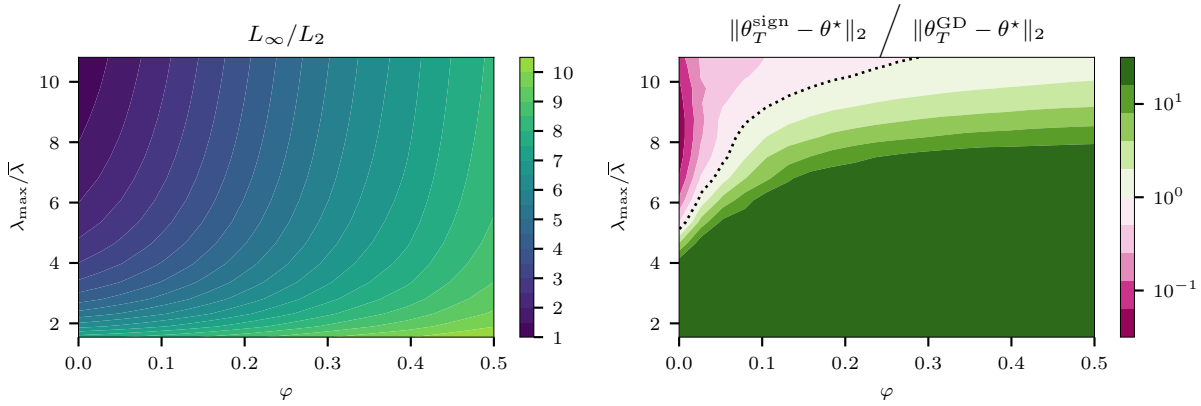


Figure 6.3: We consider quadratic objectives varying across two axes: $\lambda_{\max}/\bar{\lambda}$ as well as a rotation value $\theta$. The left plot depicts the ratio of the two relevant smoothness constants. $L_\infty$ is sensitive to $\varphi$ and grows relative to $L_2 = \lambda_{\max}$ as the problem becomes less axis-aligned. The right plot depicts the relative performance of gradient descent and sign gradient descent on these problems. GD drastically (the colormap is clipped) outperforms SIGN-GD for mildly-conditioned (small $\lambda_{\max}/\bar{\lambda}$) and non-axis-aligned (large $\varphi$) problems. However, sign gradient descent is preferable for problems with high $\lambda_{\max}/\bar{\lambda}$, given that they have some degree of axis alignment (small $\varphi$). The dashed line represents equal performance of both algorithms.

*Quadratic Experiments* We give a simple toy experiment to illustrate and verify the conditions under which SIGN-GD outperforms gradient descent. Synthetic quadratic problems of moderate dimension allow us to compute and control all relevant quantities. To generate Hessians with varying $\bar{\lambda}/\lambda_{\max}$ and axis alignment, we set the eigenvalues as $\mathbf{\Lambda} = \text{diag}(1, 1, \ldots, 1, \lambda_{\max})$ and rotate the eigenvectors by some ratio $\varphi$ in the direction prescribed by a randomly-drawn rotation matrix. One may think of $\varphi$ as a degree of rotation; the technical details can be found in Appendix C.4. For each Hessian, we compute the two smoothness constants $L_2 = \lambda_{\max}$

and $L_\infty = \|\mathbf{H}\|_{\infty,1}$. We then run $T = 100$ iterations of gradient descent (with $\alpha = 1/L_2$) and sign gradient descent (with $\alpha = 1/L_\infty$) and compute the distance to the optimum $\|\theta_T - \theta^*\|_2^2$ as a scale-invariant performance measure. Averaging over repeated runs with $\theta_0 \sim \mathcal{N}(0, \mathbf{I})$ marginalizes out the effect of initialization. The results are depicted in Fig. 6.3 and confirm the findings of Sections 6.4 and 6.5. The $L_\infty$ constant—and consequently the performance of SIGN-GD—is sensitive to the axis alignment of $\mathbf{H}$ and suffers as we increase $\theta$. For problems with $\lambda_{\max} \gg \bar{\lambda}$ that are somewhat axis-aligned, sign gradient descent outperforms gradient descent, even on these simple quadratic problems.

## 6.6 Conclusion

In this chapter, we made two main contributions. First, we unified the assumptions of existing works on both batch and stochastic sign-based methods by clarifying the relationship between separable smoothness and $\ell_\infty$-smoothness. The latter is the less restrictive, more natural assumption, and sufficient to support all existing results. Second, by studying the corresponding smoothness constant $L_\infty$ we give a clear characterization of properties of the objective function which drive the performance of SIGN-(S)GD and related methods, e.g., ADAM. We showed that SIGN-(S)GD may be preferable to (stochastic) gradient descent in the presence of *outlier eigenvalues* ($\lambda_{\max} \gg \bar{\lambda}$) given that the objective is somewhat axis-aligned. Notably, these properties have independently been found to be present in neural network training tasks. Hence, SIGN-(S)GD need not necessarily be seen as a gradient compression scheme for distributed optimization, i.e., as an approximation of (S)GD, but it is a well-motivated optimization method in its own right.

Our approach of understanding SIGN-(S)GD via the corresponding smoothness constant can be transferred to other methods. In Appendix C.1, we briefly discuss the case of block-normalized gradient descent, which can be seen as a block-wise generalization of SIGN-GD.

Finally, we want to mention two other interesting aspects of sign-based methods, which we sidestepped in this work. First, we focused on the geometric implications of sign-based methods, leaving aside the impact of gradient noise. Since the magnitude of sign-based updates is bounded, one might hope to gain additional robustness to noise. This aspect features somewhat implicitly in the work of Bernstein et al. [2018, 2019] and Safaryan and Richtárik [2019]. Relatedly, Zhang et al. [2019b] suggest that ADAM has a certain robustness to heavy-tailed gradient noise.

Second, in practice, SIGN-(S)GD is often used in the form of Eq. (6.1) and not in the norm-scaled version arising in the steepest descent framework (Eq. 6.10). There are various possible reasons for this discrepancy, which we discuss in Appendix C.3. In particular, we expand on the work of Zhang et al. [2019a] who recently

suggested that *normalized* gradient descent is actually adapted to a certain "relaxed smoothness" condition. We generalize this argument to SIGN-GD, providing a possible explanation for the aforementioned discrepancy.

# 7
# *Natural Gradient Descent and the "Empirical Fisher"*

Natural gradient descent, which preconditions a gradient descent update with the Fisher information matrix of the underlying statistical model, is a way to capture partial second-order information. Several highly visible works have advocated an approximation known as the empirical Fisher, drawing connections between approximate second-order methods and heuristics like ADAM. We dispute this argument by showing that the empirical Fisher—unlike the Fisher—does not generally capture second-order information. We further argue that the conditions under which the empirical Fisher approaches the Fisher (and the Hessian) are unlikely to be met in practice, and that, even on simple optimization problems, the pathologies of the empirical Fisher can have undesirable effects.

The empirical Fisher *does*, however, contain useful information as it is the non-central covariance matrix of the stochastic gradient. We point to the concept of variance adaptation as a possible alternative explanation for the empirical success of methods that precondition with the empirical Fisher.

This chapter is based on the conference paper:

> Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical Fisher approximation for natural gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4156–4167. Curran Associates, Inc., 2019

Coauthor contributions:

| | Sc. Ideas | Experiments | Interpretation | Writing |
|---|---|---|---|---|
| Frederik Kunstner | 50% | 80% | 50% | 50% |
| **Lukas Balles** | 40% | 20% | 40% | 40% |
| Philipp Hennig | 10% | 0% | 10% | 10% |

## *7.1 Introduction*

In this chapter, we will make additional assumptions about the optimization problem compared to the generic form given in Eq. (3.1). We assume that $\theta \in \mathbb{R}^d$ parametrizes a probabilistic model for the conditional distribution of an output $y$ given an input $x$, i.e., $p_\theta(y|x)$. We further assume that the loss is the negative log likeli-

hood thereof, $\ell_n(\theta) := -\log p_\theta(y_n|x_n)$ and that the probabilistic model is of the form $p_\theta(y|x) = p(y|f(x,\theta))$, where $p(y|\cdot)$ is an exponential family with natural parameters in $\mathbb{F}$ and $f\colon \mathbb{X} \times \mathbb{R}^d \to \mathbb{F}$ is a prediction function parameterized by $\theta \in \mathbb{R}^d$. (The prediction function could be anything from a simple linear map to a complex neural network architecture.) Our objective thus reads

$$R(\theta) \stackrel{\text{def}}{=} -\sum_n \log p_\theta(y_n|x_n) = -\sum_n \log p(y_n|f(x_n,\theta)). \tag{7.1}$$

This framework covers many common scenarios such as least-squares regression ($\mathbb{Y} = \mathbb{F} = \mathbb{R}$ and $p(y|f) = \mathcal{N}(y; f, 1)$) or $C$-class classification with cross-entropy loss ($\mathbb{Y} = \{1,\dots,C\}$, $\mathbb{F} = \mathbb{R}^C$ and $p(y=c|f) = \exp(f_c)/\sum_i \exp(f_i)$).

Gradient descent updates of the form $\theta_{t+1} = \theta_t - \alpha_t \nabla R(\theta_t)$ can be *preconditioned* with a matrix $B_t$ that incorporates additional information, such as local curvature, leading to updates of the form $\theta_{t+1} = \theta_t - \gamma_t B_t^{-1} \nabla R(\theta_t)$. Choosing $B_t$ to be the Hessian yields Newton's method, but its computation is often burdensome and might not be desirable for non-convex problems. A prominent variant in machine learning is *natural gradient descent* [NGD; Amari, 1998]. It adapts to the *information geometry* of the problem by measuring the distance between parameters with the Kullback–Leibler divergence between the resulting distributions rather than their Euclidean distance, using the Fisher information matrix (or simply "Fisher") of the model as a preconditioner,

$$F(\theta) \stackrel{\text{def}}{=} \sum_n \mathbf{E}_{p_\theta(y|x_n)} \left[ \nabla_\theta \log p_\theta(y|x_n)\, \nabla_\theta \log p_\theta(y|x_n)^\mathsf{T} \right]. \tag{7.2}$$

While this motivation is conceptually distinct from approximating the Hessian, the Fisher coincides with a generalized Gauss-Newton [Schraudolph, 2002] approximation of the Hessian for objectives of the form Eq. (7.1). This gives NGD theoretical grounding as an approximate second-order method.

A number of recent works in machine learning have relied on a certain approximation of the Fisher, which is often called the *empirical Fisher (EF)* and is defined as

$$\tilde{F}(\theta) := \sum_n \nabla_\theta \log p_\theta(y_n|x_n)\, \nabla_\theta \log p_\theta(y_n|x_n)^\mathsf{T}. \tag{7.3}$$

At first glance, this approximation is merely replacing the expectation over $y$ in Eq. (7.2) with a sample $y_n$. However, $y_n$ is a training label and *not* a sample from the model's predictive distribution $p_\theta(y|x_n)$. Therefore, and contrary to what its name suggests, the empirical Fisher is *not* an empirical (i.e. Monte Carlo) estimate of the Fisher. Due to the unclear relationship between the model distribution and the data distribution, the theoretical grounding of the empirical Fisher approximation is dubious.

Adding to the confusion, the term "empirical Fisher" is used by different communities to refer to different quantities. Authors closer to statistics tend to use "empirical Fisher" for Eq. (7.2), while many

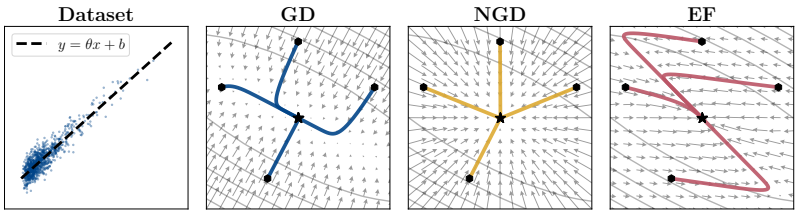


Figure 7.1: Fisher vs. empirical Fisher as preconditioners for linear least-squares regression on the data shown in the left-most panel. The second plot shows the gradient vector field of the (quadratic) loss function and sample trajectories for gradient descent. The remaining plots depict the vector fields of the natural gradient and the "EF-preconditioned" gradient, respectively. NGD successfully adapts to the curvature whereas preconditioning with the empirical Fisher results in a distorted gradient field.

works in machine learning, some listed in Section 7.2, use "empirical Fisher" for Eq. (7.3). While the statistical terminology is more accurate, we adopt the term "Fisher" for Eq. (7.2) and "empirical Fisher" for Eq. (7.3), which is the subject of this work, to be accessible to readers more familiar with this convention. We elaborate on the different uses of the terminology in Section 7.3.1.

The main purpose of this work is to provide **a detailed critical discussion of the empirical Fisher approximation.** While the discrepancy between the empirical Fisher and the Fisher has been mentioned in the literature before [Pascanu and Bengio, 2014, Martens, 2020], we see the need for a detailed elaboration of the subtleties of this important issue. The intricacies of the relationship between the empirical Fisher and the Fisher remain opaque from the current literature. Not all authors using the EF seem to be fully aware of the heuristic nature of this approximation and overlook its shortcomings, which can be seen clearly even on simple linear regression problems, see Fig. 7.1. Natural gradients adapt to the curvature of the function using the Fisher while the empirical Fisher distorts the gradient field in a way that lead to worse updates than gradient descent.

The empirical Fisher approximation is so ubiquitous that it is sometimes just called the Fisher [e.g., Chaudhari et al., 2017, Wen et al., 2020]. Possibly as a result of this, there are examples of algorithms involving the Fisher, such as Elastic Weight Consolidation [Kirkpatrick et al., 2017] and KFAC [Martens and Grosse, 2015], which have been re-implemented by third parties using the empirical Fisher. Interestingly, there is also at least one example of an algorithm that was originally developed using the empirical Fisher and later found to work better with the Fisher [Wierstra et al., 2008, Sun et al., 2009]. As the empirical Fisher is now used beyond optimization, for example as an approximation of the Hessian in empirical works studying properties of neural networks [Chaudhari et al., 2017, Jastrzębski et al., 2018], the pathologies of the EF approximation may lead the community to erroneous conclusions—an arguably more worrysome outcome than a suboptimal preconditioner.

The poor theoretical grounding stands in stark contrast to the practical success that empirical Fisher-based methods have seen. This paper is in no way meant to negate these practical advances but rather points out that the existing justifications for the approx-

imation are insufficient and do not stand the test of simple examples. This indicates that there are effects at play that currently elude our understanding, which is not only unsatisfying, but might also prevent advancement of these methods. We hope that this paper helps spark interest in understanding these effects; our final section explores a possible direction.

### 7.1.1 *Overview and Contributions*

We first provide a short but complete overview of the natural gradient and the closely related generalized Gauss-Newton method. Our main contribution is a critical discussion of the *specific arguments* used to advocate the empirical Fisher approximation. A principal conclusion is that, while the empirical Fisher follows the formal definition of a generalized Gauss-Newton matrix, it is not guaranteed to capture any useful second-order information. We propose a clarifying amendment to the definition of a generalized Gauss-Newton to ensure that all matrices satisfying it have useful approximation properties. Furthermore, while there are conditions under which the empirical Fisher approaches the true Fisher, we argue that these are unlikely to be met in practice. We illustrate that using the empirical Fisher can lead to highly undesirable effects; Fig. 7.1 shows a first example.

This raises the question: Why are methods based on the empirical Fisher practically successful? We point to an alternative explanation, as an adaptation to gradient noise in *stochastic* optimization instead of an adaptation to curvature.

## 7.2 *Related Work*

The generalized Gauss-Newton [Schraudolph, 2002] and natural gradient descent [Amari, 1998] methods have inspired a line of work on approximate second-order optimization [Martens, 2010, Botev et al., 2017, Park et al., 2000, Pascanu and Bengio, 2014, Ollivier, 2015]. A successful example in modern deep learning is the KFAC algorithm [Martens and Grosse, 2015], which uses a computationally efficient structural approximation to the Fisher.

Numerous papers have relied on the empirical Fisher approximation for preconditioning and other purposes. Our critical discussion is in no way intended as an invalidation of these works. All of them provide important insights and the use of the empirical Fisher is usually not essential to the main contribution. However, there is a certain degree of vagueness regarding the relationship between the Fisher, the EF, Gauss-Newton matrices and the Hessian. Oftentimes, only limited attention is devoted to possible implications of the empirical Fisher approximation.

The most prominent example of preconditioning with the EF is ADAM, which uses a moving average of squared gradients as "an approximation to the diagonal of the Fisher information matrix"

[Kingma and Ba, 2015]. The EF has been used in the context of variational inference by various authors [Graves, 2011, Zhang et al., 2018, Salas et al., 2018, Khan et al., 2018, Mishkin et al., 2018], some of which have drawn further connections between NGD and ADAM. There are also several works building upon which substitute the EF for the Fisher [George et al., 2018, Osawa et al., 2019].

The empirical Fisher has also been used as an approximation of the Hessian for purposes other than preconditioning. Chaudhari et al. [2017] use it to investigate curvature properties of deep learning training objectives. It has also been employed to explain certain characteristics of SGD [Zhu et al., 2019, Jastrzębski et al., 2018] or as a diagnostic tool during training [Liao et al., 2020].

Le Roux et al. [2008] and Roux and Fitzgibbon [2010] have considered the empirical Fisher in its interpretation as the (non-central) covariance matrix of stochastic gradients. While they refer to their method as "Online Natural Gradient", their goal is explicitly to adapt the update to the *stochasticity* of the gradient estimate, *not to curvature*. We will return to this perspective in Section 7.5.

Before moving on, we want to re-emphasize that other authors have previously raised concerns about the empirical Fisher approximation [e.g., Pascanu and Bengio, 2014, Martens, 2020]. This paper is meant as a detailed elaboration of this known but subtle issue, with novel results and insights. Concurrent to our work, Thomas et al. [2020] investigated similar issues in the context of estimating the generalization gap using information criteria.

## 7.3 Generalized Gauss-Newton and Natural Gradient Descent

This section briefly introduces natural gradient descent, addresses the difference in terminology for the quantities of interest across fields, introduces the generalized Gauss-Newton (GGN) and reviews the connections between the Fisher, the GGN, and the Hessian.

### 7.3.1 Natural Gradient Descent

Gradient descent follows the direction of "steepest descent", the negative gradient. But the definition of *steepest* depends on a notion of distance and the gradient is defined with respect to the Euclidean distance. The natural gradient is a concept from information geometry [Amari, 1998] and applies when the gradient is taken w.r.t. the parameters $\theta$ of a family of probability distributions $p_\theta$. Instead of measuring the distance between parameters $\theta$ and $\theta'$ with the Euclidean distance, we use the Kullback–Leibler (KL) divergence between the distributions $p_\theta$ and $p_{\theta'}$. The resulting steepest descent direction is the negative gradient preconditioned with the Hessian of the KL divergence, which is exactly the *Fisher*

*information matrix* of $p_\theta$,

$$\begin{aligned} F(\theta) &\stackrel{\text{def}}{=} \mathbf{E}_{p_\theta(z)}\left[\nabla_\theta \log p_\theta(z) \nabla_\theta \log p_\theta(z)^\mathsf{T}\right] \\ &= \mathbf{E}_{p_\theta(z)}\left[-\nabla_\theta^2 \log_\theta p_\theta(z)\right]. \end{aligned} \tag{7.4}$$

The second equality may seem counterintuitive, but the difference between the outer product of gradients and the Hessian cancels out in expectation with respect to the model distribution at $\theta$, see Appendix D.1. This equivalence highlights the relationship of the Fisher to the Hessian.

### 7.3.2 *Difference in Terminology Across Fields*

In our setting, we only model the conditional distribution $p_\theta(y|x)$ of the joint distribution $p_\theta(x, y) = p(x)p_\theta(y|x)$. The Fisher information of $\theta$ for $N$ samples from the joint distribution $p_\theta(x, y)$ is

$$F_{\prod_n p_\theta(x,y)}(\theta) = N\mathbf{E}_{x,y\sim p(x)p_\theta(y|x)}\left[\nabla_\theta \log p_\theta(y|x) \nabla_\theta \log p_\theta(y|x)^\mathsf{T}\right], \tag{7.5}$$

This is what statisticians would call the "Fisher information" of the model $p_\theta(x, y)$. However, we typically do not know the distribution over inputs $p(x)$, so we use the empirical distribution over $x$ instead and compute the Fisher information of the conditional distribution $\prod_n p_\theta(y|x_n)$;

$$F_{\prod_n p_\theta(y|x_n)}(\theta) = \sum_n \mathbf{E}_{y\sim p_\theta(y|x_n)}\left[\nabla_\theta \log p_\theta(y|x_n) \nabla_\theta \log p_\theta(y|x_n)^\mathsf{T}\right]. \tag{7.6}$$

This is Eq. (7.2), which we call the "Fisher". This is the terminology used by work on the application of natural gradient methods in machine learning, such as Martens [2020] and Pascanu and Bengio [2014], as it is the Fisher information for the distribution we are optimizing, $\prod_n p_\theta(y|x_n)$. Work closer to the statistics literature, following the seminal paper of Amari [1998], such as Park et al. [2000] and Karakida et al. [2019], call this quantity the "empirical Fisher" due to the usage of the empirical samples for the inputs. In contrast, we call Eq. (7.3) the "empirical Fisher", restated here,

$$\tilde{F}(\theta) = \sum_n \nabla_\theta \log p_\theta(y_n|x_n) \nabla_\theta \log p_\theta(y_n|x_n)^\mathsf{T}, \tag{7.7}$$

where "empirical" describes the use of samples for both the inputs and the outputs. This expression, however, does not have a direct interpretation as a Fisher information as it does not sample the output according to the distribution defined by the model. Neither is it a Monte Carlo approximation of Eq. (7.6), as the samples $y_n$ do not come from $p_\theta(y|x_n)$ but from the data distribution $p(y|x_n)$. How close the empirical Fisher (Eq. 7.7) is to the Fisher (Eq. 7.6) depends on how close the model $p_\theta(y|x_n)$ is to the true data-generating distribution $p(y|x_n)$.

| Quantity | | Statistics terminology | ML terminology |
|---|---|---|---|
| $F_{\prod_n p_\theta(x,y)}$ | Eq. (7.5) | Fisher | |
| $F_{\prod_n p_\theta(y\|x_n)}$ | Eq. (7.6) | empirical Fisher | Fisher |
| $\tilde{F}$ | Eq. (7.7) | | empirical Fisher |

Table 7.1: Common terminology for the Fisher information and related matrices by authors closely aligned with statistics, such as Amari [1998], Park et al. [2000], and Karakida et al. [2019], or machine learning, such as Martens [2010], Schaul et al. [2013], and Pascanu and Bengio [2014].

### 7.3.3 *Generalized Gauss-Newton*

One line of argument justifying the use of the empirical Fisher approximation uses the connection between the Hessian and the Fisher through the generalized Gauss-Newton (GGN) matrix [Schraudolph, 2002]. Here, we give a condensed overview of the definition and properties of the GGN.

The original Gauss-Newton algorithm is an approximation to Newton's method for nonlinear least squares problems, $R(\theta) = \frac{1}{2}\sum_n (f(x_n,\theta) - y_n)^2$. By the chain rule, the Hessian can be written as

$$\nabla^2 R(\theta) = \underbrace{\sum_n \nabla_\theta f(x_n,\theta) \nabla_\theta f(x_n,\theta)^\mathsf{T}}_{=:\mathrm{G}(\theta)} + \underbrace{\sum_n r_n \nabla^2_\theta f(x_n,\theta)}_{=:\Gamma(\theta)}, \tag{7.8}$$

where $r_n = f(x_n,\theta) - y_n$ are the residuals. The first part, $G(\theta)$, is the Gauss-Newton matrix. For small residuals, $\Gamma(\theta)$ will be small and $G(\theta)$ will approximate the Hessian. In particular, when the model perfectly fits the data, the Gauss-Newton is equal to the Hessian.

Schraudolph [2002] generalized this idea to objectives of the form $R(\theta) = \sum_n a_n(b_n(\theta))$, with $b_n \colon \mathbb{R}^d \to \mathbb{R}^M$ and $a_n \colon \mathbb{R}^M \to \mathbb{R}$, for which the Hessian can be written as[1]

$$\begin{aligned}\nabla^2 R(\theta) = &\sum_n (\mathrm{J}_\theta b_n(\theta))^\mathsf{T} \ \nabla^2_b a_n(b_n(\theta)) \ (\mathrm{J}_\theta b_n(\theta)) \\ &+ \sum_{n,m} [\nabla_b a_n(b_n(\theta))]_m \nabla^2_\theta b_n^{(m)}(\theta).\end{aligned} \tag{7.9}$$

[1] $\mathrm{J}_\theta b_n(\theta) \in \mathbb{R}^{M\times d}$ is the Jacobian of $b_n$; we use the shortened notation $\nabla^2_b a_n(b_n(\theta)) := \nabla^2_b a_n(b)|_{b=b_n(\theta)}$; $[\cdot]_m$ selects the $m$-th component of a vector; and $b_n^{(m)}$ denotes the $m$-th component function of $b_n$.

The generalized Gauss-Newton matrix (GGN) is defined as the part of the Hessian that ignores the second-order information of $b_n$,

$$G(\theta) := \sum_n [\mathrm{J}_\theta b_n(\theta)]^\mathsf{T} \ \nabla^2_b a_n(b_n(\theta)) \ [\mathrm{J}_\theta b_n(\theta)]. \tag{7.10}$$

If $a_n$ is convex, as is customary, the GGN is positive (semi-)definite even if the Hessian itself is not, making it a popular curvature matrix in non-convex problems such as neural network training. The GGN is ambiguous as it crucially depends on the "split" given by $a_n$ and $b_n$. As an example, consider the two following possible splits for the least-squares problem from above:

$$a_n(b) = \frac{1}{2}(b - y_n)^2, \qquad b_n(\theta) = f(x_n,\theta), \qquad \text{or} \tag{7.11}$$

$$a_n(b) = \frac{1}{2}(f(x_n,b) - y_n)^2, \qquad b_n(\theta) = \theta. \tag{7.12}$$

The first recovers the classical Gauss-Newton, while in the second case, the GGN equals the Hessian. While this is an extreme example, the split will be important for our discussion.

### 7.3.4 *Connections Between the Fisher, the GGN and the Hessian*

While NGD is not explicitly motivated as an approximate second-order method, the following result, noted by several authors,[2] shows that the Fisher captures partial curvature information about the problem defined in Eq. (7.1).

[2] Heskes [2000] showed this for regression with squared loss, Pascanu and Bengio [2014] for classification with cross-entropy loss, and Martens [2020] for general exponential families. However, this has been known earlier in the statistics literature in the context of "Fisher Scoring" (see Wang [2010] for a review).

**Proposition 7.1** (Martens [2020], §9.2)**.** *If $p(y|f)$ is an exponential family distribution with natural parameters $f \in \mathbb{F}$, then the Fisher information matrix coincides with the GGN of the objective in Eq.* (7.1) *using the split*

$$a_n(b) = -\log p(y_n|b), \qquad b_n(\theta) = f(x_n, \theta), \tag{7.13}$$

*and reads*

$$F(\theta) = G(\theta) = -\sum_n [\mathrm{J}_\theta f(x_n, \theta)]^\mathsf{T} \, \nabla_f^2 \log p(y_n|f(x_n, \theta)) \, [\mathrm{J}_\theta f(x_n, \theta)]. \tag{7.14}$$

For completeness, a proof can be found in Appendix D.2. The key insight is that $\nabla_f^2 \log p(y|f)$ does not depend on $y$ for exponential families. One can see Eq. (7.13) as the "canonical" split, since it matches the classical Gauss-Newton for the probabilistic interpretation of least-squares. From now on, when referencing "the GGN" without further specification, we mean this particular split.

The GGN, and under the assumptions of Proposition 7.1 also the Fisher, are well-justified approximations of the Hessian and we can bound their approximation error in terms of the (generalized) residuals, mirroring the motivation behind the classical Gauss-Newton (Proof in Appendix D.2.2).

**Proposition 7.2.** *Let $R(\theta)$ be defined as in Eq.* (7.1) *with $\mathbb{F} = \mathbb{R}^M$. Denote by $f_n^{(m)}$ the m-th component of $f(x_n, \cdot) \colon \mathbb{R}^D \to \mathbb{R}^M$ and assume each $f_n^{(m)}$ is $\beta$-smooth. Let $G(\theta)$ be the GGN (Eq. 7.10). Then,*

$$\|\nabla^2 R(\theta) - G(\theta)\|_2^2 \leq r(\theta)\beta, \tag{7.15}$$

*where $\|\cdot\|_2$ denotes the spectral norm for matrices and*

$$r(\theta) \overset{def}{=} \sum_{n=1}^{N} \|\nabla_f \log p(y_n|f(x_n, \theta))\|_1. \tag{7.16}$$

The approximation improves as the residuals in $r(\theta)$ diminish, and is exact if the data is perfectly fit.

## 7.4 *Critical Discussion of the Empirical Fisher*

Two arguments have been put forward to advocate the empirical Fisher approximation. Firstly, it has been argued that it follows

the definition of a generalized Gauss-Newton matrix, making it an approximate curvature matrix in its own right. We examine this relation in §7.4.1 and show that, while technically correct, it does not entail the approximation guarantee usually associated with the GGN.

Secondly, a popular argument is that the empirical Fisher approaches the Fisher at a minimum if the model "is a good fit for the data". We discuss this argument in §7.4.2 and point out that it requires strong additional assumptions, which are unlikely to be met in practical scenarios. In addition, this argument only applies close to a minimum, which calls into question the usefulness of the empirical Fisher as a preconditioner in the early phase of optimization. We discuss this in §7.4.3, showing that preconditioning with the empirical Fisher leads to adverse effects on the scaling and the direction of the updates far from an optimum.

We use simple examples to illustrate our arguments. We want to emphasize that—as *counter-examples* to arguments found in the existing literature—they are designed to be as simple as possible, and deliberately do not involve intricate state-of-the art models that would complicate analysis. On a related note, while contemporary machine learning often relies on *stochastic* optimization, we restrict our considerations to the deterministic (full-batch) setting to focus on the adaptation to curvature.

### 7.4.1 *The Empirical Fisher as a Generalized Gauss-Newton Matrix*

The first justification for the empirical Fisher is that it matches the construction of a generalized Gauss-Newton (Eq. 7.10) using the split [Bottou et al., 2018]

$$a_n(b) = -\log b, \qquad b_n(\theta) = p(y_n|f(x_n,\theta)). \tag{7.17}$$

Although technically correct,[3] we argue that this split does not provide a reasonable approximation.

[3] The equality can easily be verified by plugging the split (7.17) into the definition of the GGN (Eq. 7.10) and observing that $\nabla_b^2 a_n(b) = \nabla_b a_n(b)\, \nabla_b a_n(b)^\mathsf{T}$ as a special property of the choice $a_n(b) = -\log(b)$.

For example, consider a least-squares problem which corresponds to the log-likelihood $\log p(y|f) = \log \exp[-\frac{1}{2}(y-f)^2]$. In this case, Eq. (7.17) splits the identity function, $\log\exp(\cdot)$, and takes into account the curvature from the log while ignoring that of exp. This questionable split runs counter to the basic motivation behind the classical Gauss-Newton matrix, that small residuals lead to a good approximation to the Hessian: The empirical Fisher

$$\begin{aligned}\tilde{F}(\theta) &= \sum_n \nabla_\theta \log p_\theta(y_n|x_n)\, \nabla_\theta \log p_\theta(y_n|x_n)^\mathsf{T} \\ &= \sum_n r_n^2\, \nabla_\theta f(x_n,\theta)\, \nabla_\theta f(x_n,\theta)^\mathsf{T},\end{aligned} \tag{7.18}$$

approaches zero as the residuals $r_n = f(x_n,\theta) - y_n$ become small. In that same limit, the Fisher $F(\theta) = \sum_n \nabla f(x_n,\theta) \nabla f(x_n,\theta)^\mathsf{T}$ does approach the Hessian, which we recall from Eq. (7.8) to be given by $\nabla^2 R(\theta) = F(\theta) + \sum_n r_n \nabla_\theta^2 f(x_n,\theta)$. This argument generally applies for problems where we can fit all training samples such

that $\nabla_\theta \log p_\theta(y_n|x_n) = 0$ for all $n$. In such cases, the EF goes to zero while the Fisher (and the corresponding GGN) approaches the Hessian (Prop. 7.2), which will generally be non-zero.

For the *generalized* Gauss-Newton, the role of the "residual" is played by the gradient $\nabla_b a_n(b)$; compare Equations (7.8) and (7.9). To retain the motivation behind the classical Gauss-Newton, the split should be chosen such that this gradient can in principle attain zero, in which case the residual curvature not captured by the GGN in (7.9) vanishes. The EF split (Eq. 7.17) does not satisfy this property, as $\nabla_b \log b$ can never go to zero for a probability $b \in [0, 1]$. It might be desirable to amend the definition of a generalized Gauss-Newton to enforce this property (addition in **bold**):

**Definition 7.1** (Generalized Gauss-Newton)**.** *A split $R(\theta) = \sum_n a_n(b_n(\theta))$ with convex $a_n$, leads to a generalized Gauss-Newton matrix of $R$, defined as*

$$G(\theta) = \sum_n G_n(\theta), \quad G_n(\theta) := [\mathrm{J}_\theta b_n(\theta)]^\mathsf{T} \ \nabla_b^2 a_n(b_n(\theta)) \ [\mathrm{J}_\theta b_n(\theta)], \tag{7.19}$$

***if the split $a_n, b_n$ is chosen such that there is $b_n^* \in \mathbf{Im}(b_n)$ such that*** $\nabla_b a_n(b)|_{b=b_n^*} = 0$**.**

Under suitable smoothness conditions, a split satisfying this condition will have a meaningful error bound akin to Proposition 7.2. To avoid confusion, we want to note that this condition does not assume the existence of $\theta^*$ such that $b_n(\theta^*) = b_n^*$ for all $n$; only that the residual gradient for each data point can, in principle, go to zero.

### 7.4.2 *The Empirical Fisher Near a Minimum*

An often repeated argument is that the empirical Fisher converges to the true Fisher when the model is a good fit for the data [e.g., Jastrzębski et al., 2018, Zhu et al., 2019]. Unfortunately, this is often misunderstood to simply mean "near the minimum". The above statement has to be carefully formalized and requires additional assumptions, which we detail in the following.

Assume that the training data consists of iid samples from some data-generating distribution $p_{\text{true}}(x, y) = p_{\text{true}}(y|x)p_{\text{true}}(x)$. If the model is realizable, i.e., there exists a parameter setting $\theta_\mathrm{T}$ such that $p_{\theta_\mathrm{T}}(y|x) = p_{\text{true}}(y|x)$, then clearly by a Monte Carlo sampling argument,

$$\tilde{F}(\theta_\mathrm{T})/N \to F(\theta_\mathrm{T})/N, \quad N \to \infty. \tag{7.20}$$

Additionally, if the maximum likelihood estimate for $N$ samples $\theta_N^\star$ is consistent in the sense that $p_{\theta_N^\star}(y|x)$ converges to $p_{\text{true}}(y|x)$ as $N \to \infty$,

$$\frac{1}{N}\tilde{F}(\theta_N^\star) \overset{N\to\infty}{\longrightarrow} \frac{1}{N}F(\theta_N^\star). \tag{7.21}$$

That is, the empirical Fisher converges to the Fisher *at* the minimum as the number of data points grows. (Both approach the Hessian, as can be seen from the second equality in Eq. 7.4 and detailed in Appendix D.2.2.) For the EF to be a useful approximation, we thus need

(i) a "correctly-specified" model in the sense of the realizability condition, and

(ii) enough data to recover the true parameters.

These two requirements are often at odds with each other.

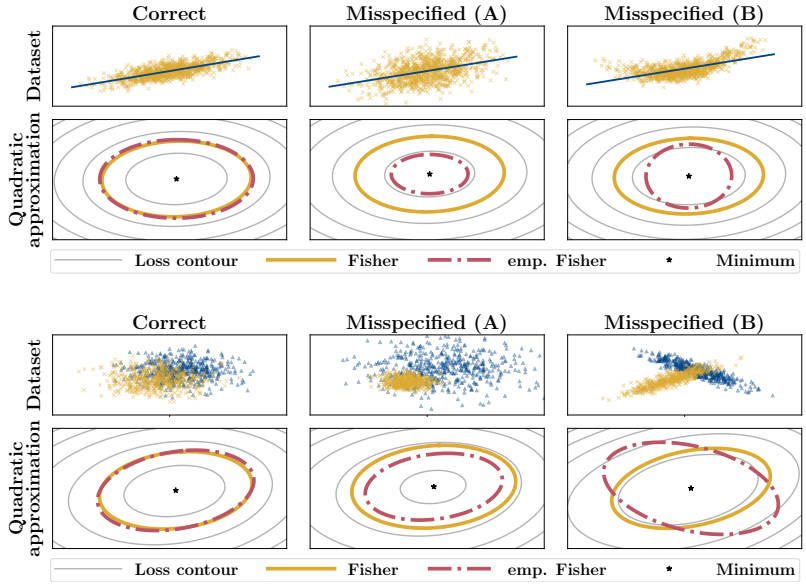


Figure 7.2: Quadratic approximations of the loss function at the minimum using the Fisher and the empirical Fisher on linear regression (top) and logistic regression (bottom) problems. The true Fisher coincides with the GGN in all cases and is a decent approximation of the true loss. The EF closely approximates the Fisher if the model captures the true data-generating process, but can be arbitrarily wrong if that assumption is violated, even though we are at the exact minimum and use a very large number of data points.

*Correctly-Specified Model* Condition (i) requires that the likelihood $p(y|f)$ is well-specified and that the prediction function $f(x, \theta)$ captures all relevant information. This is possible in classical statistical modeling of, say, scientific phenomena where the effect of $x$ on $y$ is modeled based on domain knowledge. But it is unlikely to hold when the model is only approximate, as is most often the case in machine learning. We will now illustrate this with various examples of model misspecification on two-dimensional toy problems for linear and logistic regression. In both cases the Fisher coincides with the GGN (Section 7.3.4) making it a well-grounded approximation of the Hessian at the minimum (and even exact for linear regression). We compare the empirical and the true Fisher at the exact minimum and use a very large number of data points, making model misspecification the only significant source of deviation between the two. Figure 7.2 depicts the examples:

- The top row shows a linear regression model, which assumes the data to be generated as $y|x \sim \mathcal{N}(x\theta^* + b^*, 1)$. If this is fulfilled (left), the EF is a good approximation of the Fisher and, thus, the

Hessian. The middle panel (Misspecified (A)) shows a situation where the data is actually generated by $y|x \sim \mathcal{N}(x\theta^* + b^*, 2)$, that is, the observation noise is higher than the model assumes. The EF "underestimates" the scale of the Fisher in this situation. Finally, in the right panel (Misspecified (B)), the true data has a quadratic relationship that is not captured by the linear model. The EF deviates substantially from the Fisher both in scale and shape.

- The plot on the bottom shows logistic regression experiments. Logistic regression assumes a linear relationship between the features and the logits [Hastie et al., 2009, §4.4.5]. This assumption is fulfilled if the features are generated by two class-conditional Gaussians with identical covariance matrices (left panel), which results in a good correspondence between the EF and the true Fisher. Using different class-conditional covariances (middle and right panels) violates the logistic regression assumption and can cause the EF to be arbitrarily wrong. Note: we achieve classification accuracies of $\geq 85\%$ in the misspecified cases compared to 73% in the well-specified case, which shows that a well-performing model is not necessarily a well-specified one.

*Enough Data* It is, of course, possible to satisfy the realizability condition by using a very flexible prediction function $f(x, \theta)$, such as a deep network. However, this will typically be at odds with requirement (ii), since "enough" data has to be seen relative to the model capacity. The higher the model capacity, the more data is necessary to recover a parameter setting that replicates the true data distribution. The massively overparameterized models typically used in deep learning are able to fit the training data almost perfectly, even when regularized [Zhang et al., 2017]. As discussed in 7.4.1, this will lead to the individual gradients, and thus the EF, being close to zero at a minimum, whereas the Hessian will generally be nonzero.

### 7.4.3 *Preconditioning with the Empirical Fisher Far from an Optimum*

The relationship discussed in §7.4.2 only holds close to the minimum. Any similarity between $p_\theta(y|x)$ and $p_{\text{true}}(y|x)$ is *very* unlikely when $\theta$ has not been adapted to the data, for example, at the beginning of an optimization procedure. This makes the empirical Fisher a questionable choice as a preconditioner, in particular in such early phases.

In fact, the empirical Fisher can cause severe, adverse distortions of the gradient field far from the optimum, as evident even on the elementary linear regression problem of Fig. 7.1. As a consequence, EF-preconditioned gradient descent compares unfavorably to NGD even on simple linear regression and classification tasks, as shown in Fig. 7.3. The cosine similarity plotted in Fig. 7.3 shows that the

empirical Fisher can be arbitrarily far from the Fisher in that the two preconditioned updates point in almost opposite directions.

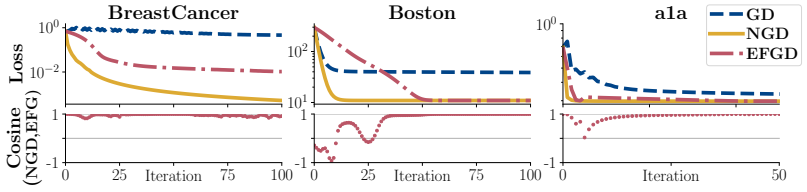


Figure 7.3: Fisher (NGD) vs. empirical Fisher (EFGD) as preconditioners (with damping) on linear classification (BreastCancer, a1a) and regression (Boston). While the EF *can* be a good approximation for preconditioning on some problems (e.g., a1a), it is not guaranteed to be. The second row shows the cosine similarity between the EF direction and the natural gradient, over the path taken by EFGD, showing that the EF can lead to update directions that are opposite to the natural gradient (see Boston). Even when the direction is correct, the magnitude of the steps can lead to poor performance (see BreastCancer). See Appendix D.3 for details and additional experiments.

One particular issue is the scaling of EF-preconditioned updates. As the empirical Fisher is the sum of "squared" gradients (Eq. 7.3), multiplying the gradient by the inverse of the EF leads to updates of magnitude almost inversely proportional to that of the gradient, at least far from the optimum. This effect has to be counteracted by adapting the step size, which requires manual tuning and makes the selected step size dependent on the starting point. Fig. 7.4 shows the linear regression problem on the Boston dataset, originally shown in Fig. 7.3, where each line is a different starting point, using the same hyperparameters as in Fig. 7.3. The starting points are selected from $[-\theta^\star, \theta^\star]$, where $\theta^\star$ is the optimum. When the optimization starts close to the minimum (low loss), the empirical Fisher is a good approximation to the Fisher and there are very few differences with NGD. However, when the optimization starts far from the minimum (high loss), the individual gradients, and thus the sum of outer product gradients, are large, which leads to very small steps, regardless of curvature, and slow convergence. While this could be counteracted with a larger step size in the beginning, this large step size would not work close to the minimum where it would cause oscillations. The selection of the step size therefore depends on the starting point, and would ideally follow a decreasing schedule. NGD does not suffer from these issues.

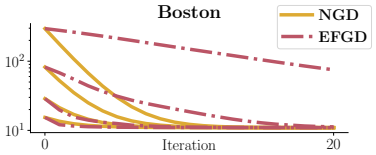


Figure 7.4: Linear regression on the Boston dataset with different starting points (each line is a different initialization). When the optimization starts close to the minimum (low initial loss), the empirical Fisher is a good approximation to the Fisher and there are very few differences with NGD, but the performance degrades as the optimization procedure starts farther away (large initial loss).

## 7.5 *Variance Adaptation*

The previous sections have shown that, interpreted as a *curvature* matrix, the empirical Fisher is a questionable choice at best. Another perspective on the empirical Fisher is that, in contrast to the Fisher, it contains useful information to adapt to the gradient noise in *stochastic* optimization.

In stochastic gradient descent [SGD; Robbins and Monro, 1951], we sample $n \in [N]$ uniformly at random and use a stochastic gradient $g(\theta) = -N \nabla_\theta \log p_\theta(y_n|x_n)$ as an inexpensive but noisy estimate of $\nabla R(\theta)$. The empirical Fisher, as a sum of outer products of individual gradients, coincides with the non-central second moment of this estimate and can be written as

$$N\tilde{F}(\theta) = \Sigma(\theta) + \nabla R(\theta)\, \nabla R(\theta)^\mathsf{T}, \qquad \Sigma(\theta) := \mathbf{Cov}[g(\theta)]. \tag{7.22}$$

Gradient noise is a major hindrance to SGD and the covariance information encoded in the EF may be used to attenuate its harmful effects, e.g., by scaling back the update in high-noise directions.

A small number of works have explored this idea before. Le Roux et al. [2008] showed that the update direction $\Sigma(\theta)^{-1}g(\theta)$ maximizes the probability of achieving a decrease in function value, while Schaul et al. [2013] proposed a diagonal rescaling based on the signal-to-noise ratio of each coordinate, $D_{ii} := [\nabla R(\theta)]_i^2 \,/\, ([\nabla R(\theta)]_i^2 + \Sigma(\theta)_{ii})$. Balles and Hennig [2018] identified these factors as *optimal* in that they minimize the expected error $\mathbf{E}\left[\|Dg(\theta) - \nabla R(\theta)\|_2^2\right]$ for a diagonal matrix $D$.

A straightforward extension of this argument to full matrices yields the variance adaptation matrix

$$\begin{aligned} M &= \left(\Sigma(\theta) + \nabla R(\theta)\,\nabla R(\theta)^\mathsf{T}\right)^{-1} \nabla R(\theta)\,\nabla R(\theta)^\mathsf{T} \\ &= (N\tilde{F}(\theta))^{-1} \nabla R(\theta)\,\nabla R(\theta)^\mathsf{T}. \end{aligned} \tag{7.23}$$

In that sense, preconditioning with the empirical Fisher can be understood as an adaptation to gradient noise instead of an adaptation to curvature. The multiplication with $\nabla R(\theta)\nabla R(\theta)^\mathsf{T}$ in Eq. (7.23) will counteract the poor scaling discussed in §7.4.3.

This perspective on the empirical Fisher is currently not well studied. Of course, there are obvious difficulties ahead: Computing the matrix in Eq. (7.23) requires the evaluation of all gradients, which defeats its purpose. It is not obvious how to obtain meaningful estimates of this matrix from, say, a mini-batch of gradients, that would provably attenuate the effects of gradient noise. Nevertheless, we believe that variance adaptation is a possible explanation for the practical success of existing methods using the EF and an interesting avenue for future research. To put it bluntly: it may just be that the name "empirical Fisher" is a fateful historical misnomer, and the quantity should instead just be described as the gradient's non-central second moment.

As a final comment, it is worth pointing out that some methods precondition with the *square-root* of the EF, the prime example being ADAM. While this avoids the "inverse gradient" scaling discussed in §7.4.3, it further widens the conceptual gap between those methods and natural gradient. In fact, such a preconditioning effectively cancels out the gradient magnitude, which has recently been examined more closely as "sign gradient descent" [Balles and Hennig, 2018, Bernstein et al., 2018].

## *7.6 Computational Aspects*

The empirical Fisher approximation is often motivated as an easier-to-compute alternative to the Fisher. While there is some merit to this argument, we argued above that it computes the wrong quantity. A Monte Carlo approximation to the Fisher has the same computational complexity and a similar implementation: sample

one output $\tilde{y}_n$ from the model distribution $p(y|f(x_n,\theta))$ for each input $x_n$ and compute the outer product of the gradients

$$\sum_n \nabla_\theta \log p(\tilde{y}_n|f(x_n,\theta)) \nabla_\theta \log p(\tilde{y}_n|f(x_n,\theta))^\mathsf{T}. \tag{7.24}$$

While noisy, this one-sample estimate is unbiased and does not suffer from the problems discussed above. This is the approach used by Martens and Grosse [2015] and Zhang et al. [2018].

As a side note, some implementations use a biased estimate by using the most likely output $\hat{y}_n = \arg\max_y p(y|f(x_n,\theta))$ instead of sampling $\tilde{y}_n$ from $p(y|f(x_n,\theta))$. This scheme could be beneficial in some circumstances as it reduces variance, but it can backfire by increasing the bias. For the least-squares loss, $p(y|f(x_n,\theta))$ is a Gaussian distribution centered at $f(x_n,\theta)$ and the most likely output is $f(x_n,\theta)$. The gradient $\nabla_\theta \log p(y|f(x_n,\theta))|_{y=f(x_n,\theta)}$ is then always zero.

For high quality estimates, sampling additional outputs and averaging the results is inefficient. If $M$ Monte Carlo samples $\tilde{y}_1,\dots,\tilde{y}_M$ per input $x_n$ are used to compute the gradients $g_m = \nabla \log p(\tilde{y}_m|f(x_n,\theta))$, most of the computation is repeated. The gradient $g_m$ is

$$g_m = -\nabla_\theta \log p(\tilde{y}_m|f(x_n,\theta)) = -(\mathrm{J}_\theta f(x_n,\theta))^\mathsf{T} \nabla_f \log p(\tilde{y}_m|f), \tag{7.25}$$

where the Jacobian of the model output, $\mathrm{J}_\theta f$, does not depend on $\tilde{y}_m$. The Jacobian of the model is typically more expensive to compute than the gradient of the log-likelihood w.r.t. the model output, especially when the model is a neural network. This approach repeats the difficult part of the computation $M$ times. The expectation can instead be computed in closed form using the generalized Gauss-Newton equation (Eq. D.7, or Eq. 7.10 in the main text), which requires the computation of the Jacobian only once per sample $x_n$.

The main issue with this approach is that computing Jacobians is currently not well supported by deep learning auto-differentiation libraries, such as TensorFlow or Pytorch. However, the current implementations relying on the empirical Fisher also suffer from this lack of support, as they need access to the individual gradients to compute their outer product. Access to the individual gradients is equivalent to computing the Jacobian of the vector ,

$$[-\log p(y_1|f(x_1,\theta)), ..., -\log p(y_N|f(x_N,\theta)]^\mathsf{T}. \tag{7.26}$$

The ability to efficiently compute Jacobians and/or individual gradients in parallel would drastically improve the practical performance of methods based on the Fisher and empirical Fisher, as most of the computation of the backward pass can be shared between samples.

## 7.7 Conclusions

We offered a critical discussion of the empirical Fisher approximation, summarized as follows:

- While the empirical Fisher follows the formal definition of a generalized Gauss-Newton matrix, we argued that the underlying split of the objective function does not retain useful second-order information. We proposed a clarifying amendment to the definition of the GGN.
- A clear relationship between the empirical Fisher and the Fisher only exists at a minimum under strong additional assumptions: (i) a realizable model and (ii) enough data relative to that model's capacity. These two conditions are at odds with each other and unlikely to be met in practice, especially when using vastly overparametrized general function approximators and settling for approximate minima, as is common practice in deep learning.
- Far from an optimum, preconditioning with the empirical Fisher leads to update magnitudes which are inversely proportional to that of the gradient, complicating step size tuning and often leading to poor performance even for linear models.
- As a possible alternative explanation of the practical success of EF preconditioning, and an interesting avenue for future research, we have pointed to the concept of variance adaptation.

The existing arguments do not justify the empirical Fisher as a reasonable approximation to the Fisher or the Hessian. Of course, this does not rule out the existence of certain model classes for which the EF might give reasonable approximations. However, as long as we have not clearly identified and understood these cases, the true Fisher is the "safer" choice as a curvature matrix and should be preferred in virtually all cases.

Contrary to conventional wisdom, the Fisher is not inherently harder to compute than the EF. As shown by Martens and Grosse [2015], an unbiased estimate of the true Fisher can be obtained at the same computational cost as the empirical Fisher by replacing the expectation in Eq. (7.2) with a single sample $\tilde{y}_n$ from the model's predictive distribution $p_\theta(y|x_n)$; even exact computation of the Fisher is feasible in many cases, as discussed in Section 7.6. The apparent reluctance to compute the Fisher might have more to do with the current lack of convenient implementations in deep learning libraries. We believe that it is misguided—and potentially dangerous—to accept the poor theoretical grounding of the EF approximation purely for implementational convenience.

# 8
# Conclusion

Stochastic optimization algorithms have become *the* computational workhorse of contemporary machine learning and, in particular, deep learning. Custom computer processors have been designed to compute gradients of deep learning models faster and more efficiently. Technology companies are housing thousands of them in their data centers to power machine learning-based services from speech recognition to object tracking in videos. In light of that, it is astonishing that we seem to be stuck with the most simplistic optimization algorithms. With the rise of deep learning, we have shelved half a century worth of research on deterministic optimization—quasi-Newton methods, line searches, automatic stopping criteria, to name just a few aspects—and reverted to basic first-order methods and manual hyperparameter tuning.

There are various reasons for this setback. The training of deep neural networks poses highly non-convex optimization problems, which are well known to be fundamentally more challenging. Tricks and heuristics like rectified linear units [Nair and Hinton, 2010] or batch normalization [Ioffe and Szegedy, 2015] introduce additional pathologies from the optimization perspective.

This thesis suggests that another major reason is that current algorithms are *unaware* of the stochastic error in the evaluations they receive. They are unable to obtain any information about the characteristics of the error (e.g., properties such as its covariance) and they lack any mechanism that would allow them to adapt their behavior accordingly. In the preceding chapters we have presented various advances to make stochastic optimization methods *noise-aware*.

## 8.1 Summary

Chapter 4 discussed how batch size and step size influence the behavior of stochastic gradient descent. We showed that a batch size of 1 is cost-efficient in a greedy sense, but demands a step size schedule that decreases with the signal-to-noise ratio of the stochastic gradient. Accurately estimating that signal-to-noise ratio is fundamentally ill-posed and, therefore, the optimal step size schedule is not amenable to automation. An increasing step size

schedule offers an alternative that can be automated in a principled way. Doing so requires an estimate of the trace of the gradient covariance matrix, which can be obtained from the evaluation of a minibatch of gradients.

In Chapter 5, we identified a variance-based adaptation mechanism in the the ADAM optimizer by Kingma and Ba [2015], likely the most widely-used optimization method in deep learning. We interpreted ADAM as a combination of two aspects: SIGN-SGD and elementwise variance adaptation. We formalized the variance adaptation aspect, which can be seen as an elementwise extension of the optimal variance-based (scalar) step size identified in Chapter 4. We transferred it from SIGN-SGD to SGD to facilitate an ablation study of the two aspects. Our experiments showed that the sign aspect accounts for most of the empirical difference in performance between ADAM and SGD.

Chapter 6 studied that sign aspect in greater detail. Ignoring stochasticity, we studied sign gradient descent as steepest descent in the non-Euclidean geometry induced by the maximum norm. This unified and simplified various existing results on sign-based optimization methods. We show that an objective function is most amenable to sign-based methods if its Hessian is characterized by outlier eigenvalues and a certain degree of diagonal concentration, which is remarkably in line with recent experimental studies on the Hessian in deep learning training tasks.

Finally, Chapter 7 examined methods that use the (noncentral) gradient covariance matrix as a preconditioner for (stochastic) gradient descent. These methods are usually motivated as approximate versions of natural gradient descent, arguing that the (noncentral) gradient covariance matrix approximates the Fisher information and dubbing it the "empirical Fisher". We showed that this approximation has fundamental theoretical and practical flaws. We argued that the empirical success of methods that precondition with the empirical Fisher might better be understood as a form of variance adaptation, akin to the mechanism discussed in Chapter 5.

## *8.2 Further Research*

The overarching goal of the work presented in this thesis is to devise methods that act like little intelligent agents themselves: They should autonomously collect information about the stochastic noise and actively adapt their behavior accordingly. While a lot more work will be necessary, hopefully, some elements and ideas that might enable such methods have been provided here.

There are several avenues for further research, most of which have already been highlighted in the respective chapters. A broader point that briefly surfaced in multiple chapters is the interplay between geometry and stochasticity. Most research—including that presented here—tackles one of these aspects while the other one is ignored or hidden away in crude bounds that often rely on overly

simplifying assumptions. However, there are many connections between the geometry of an empirical risk minimization problem and its stochastic properties. A more comprehensive approach that considers both aspects jointly will very likely be fruitful.

Finally, it is the author's strong conviction that there is an urgent need for more bold and creative research to take us beyond simple first-order methods in deep learning. Deep learning frameworks, such as TensorFlow and PyTorch, have clearly helped the community make enormous progress in recent years. But their dominance also channels research efforts into ideas that are easy to realize within those frameworks. With regards to optimization, this favors simple methods that use exactly one stochastic gradient evaluation per iteration. Thankfully, we have recently seen some progress with the emergence of new and more flexible frameworks [Bradbury et al., 2018] and community efforts to improve existing ones [Dangel et al., 2020].

# 9
# *Bibliography*


Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Leonard Adolphs, Jonas Kohler, and Aurelien Lucchi. Ellipsoidal trust region methods and the marginal value of Hessian information for neural network training. *arXiv preprint arXiv:1905.09201*, 2019.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Lukas Balles and Philipp Hennig. Dissecting Adam: The sign, magnitude and variance of stochastic gradients. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 413–422. PMLR, 2018.

Lukas Balles, Maren Mahsereci, and Philipp Hennig. Automizing stochastic optimization with gradient variance estimates. In *Automatic Machine Learning Workshop at ICML 2017*, 2017a.

Lukas Balles, Javier Romero, and Philipp Hennig. Coupling adaptive batch sizes with learning rates. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 410–419, 2017b.

Lukas Balles, Fabian Pedregosa, and Nicolas Le Roux. The geometry of sign gradient descent. *arXiv preprint arXiv:2002.08056*, 2020.

S Becker and Yann LeCun. Improving the convergence of back-propagation learning with second-order methods. i-voc. 1988 connectionist models summer school, 1988.

Joseph Berkson. Application of the logistic function to bio-assay. *Journal of the American statistical association*, 39(227):357–365, 1944.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. SignSGD: compressed optimisation for non-convex problems. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 559–568. PMLR, 2018.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. SignSGD with majority vote is communication efficient and fault tolerant. In *7th International Conference on Learning Representations (ICLR)*, 2019.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 557–565. PMLR, 06–11 Aug 2017.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.

Léon Bottou and Yann LeCun. Large scale online learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2): 223–311, 2018. DOI: 10.1137/16M1080173.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155, 2012.

David Carlson, Ya-Ping Hsieh, Edo Collins, Lawrence Carin, and Volkan Cevher. Stochastic spectral descent for discrete graphical models. *IEEE Journal of Selected Topics in Signal Processing*, 10(2), 2015.

Augustin Louis Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25 (1847):536–538, 1847.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Automated Inference with Adaptive Batches. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1504–1513, Fort Lauderdale, Florida, USA, 20–22 Apr 2017. PMLR.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

Michael P Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

C.F. Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Hamburgi sumptibus Frid. Perthes et I.H. Besser, 1809.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a Kronecker-factored eigenbasis. In Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9573–9583, Montréal, Canada, 3–8 Dec 2018.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via Hessian eigenvalue density. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2232–2241. PMLR, 9–15 Jun 2019.

Boris Ginsburg, Patrice Castonguay, Oleksii Hrinchuk, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, Huyen

Nguyen, and Jonathan M Cohen. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. *arXiv preprint arXiv:1905.11286*, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Sardinia, Italy, 13–15 May 2010. PMLR.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Alex Graves. Practical variational inference for neural networks. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356, Granada, Spain, 12–14 Dec 2011.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Verlag, 2009.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. DOI: 10.1109/CVPR.2016.90.

Tom Heskes. On "natural" learning and pruning in multilayered perceptrons. *Neural Computation*, 12(4):881–901, 2000.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

Stanisław Jastrzębski, Zac Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Amos Storkey, and Yoshua Bengio. Three factors influencing minima in SGD. In *International Conference on Artificial Neural Networks*, 2018.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL `http://www.scipy.org/`.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of Fisher information in deep neural networks: Mean field approach. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1032–1041. PMLR, 2019.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 2019.

Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms*. SIAM, 2014.

N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P.T. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of

*Proceedings of Machine Learning Research*, pages 2616–2625. PMLR, 2018.

Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations (ICLR)*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical Fisher approximation for natural gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4156–4167. Curran Associates, Inc., 2019.

Nicolas Le Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.

Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 1998.

A.M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. 1805.

Kfir Y Levy. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016.

Xinyan Li, Qilong Gu, Yingxue Zhou, Tiancong Chen, and Arindam Banerjee. Hessian based analysis of SGD for deep nets: Dynamics and generalization. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 190–198. SIAM, 2020.

Zhibin Liao, Tom Drummond, Ian Reid, and Gustavo Carneiro. Approximate Fisher information matrix to characterise the training of deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):15–26, 2020.

Maren Mahsereci and Philipp Hennig. Probabilistic line searches for stochastic optimization. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 181–189. Curran Associates, Inc., 2015.

Maren Mahsereci and Philipp Hennig. Probabilistic line searches for stochastic optimization. *Journal of Machine Learning Research*, 18(119):1–59, 2017.

James Martens. Deep learning via Hessian-free optimization. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 735–742. Omnipress, 2010.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146): 1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In Francis Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2408–2417, 2015.

Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. SLANG: Fast structured covariance approximations for Bayesian deep learning with natural gradient. In Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6248–6258. Curran Associates, Inc., Montréal, Canada, 3–8 Dec 2018.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814. Omnipress, 2010.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

Yurii Nesterov. *Lectures on Convex Optimization*. Springer Publishing Company, Incorporated, 2nd edition, 2018. ISBN 3319915770.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Isaac Newton. *De analysi per aequationes numero terminorum infinitas*. 1711.

Yann Ollivier. Riemannian metrics for neural networks I: feedforward networks. *Information and Inference: A Journal of the IMA*, 4 (2):108–153, 2015.

Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Large-scale distributed second-order optimization using Kronecker-factored approximate curvature for deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12359–12367. Computer Vision Foundation / IEEE, June 2019.

Vardan Papyan. The full spectrum of deep net Hessians at scale: Dynamics with sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Courtney Paquette and Katya Scheinberg. A stochastic line search method with expected complexity analysis. *SIAM Journal on Optimization*, 30(1):349–376, 2020. DOI: 10.1137/18M1216250.

Hyeyoung Park, Shun-ichi Amari, and Kenji Fukumizu. Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks*, 13(7):755–764, 2000.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations (ICLR)*, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors,

*Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Jiri Rohn. Computing the norm $\|A\|_{\infty,1}$ is NP-hard. *Linear & Multilinear Algebra*, 47, 2000.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Nicolas Le Roux and Andrew Fitzgibbon. A fast natural Newton method. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning*, pages 623–630, Haifa, Israel, June 2010. Omnipress.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323: 533–536, 1986.

Mher Safaryan and Peter Richtárik. On stochastic sign descent methods. *arXiv preprint arXiv:1905.12938*, 2019.

Arnold Salas, Stefan Zohren, and Stephen Roberts. Practical Bayesian learning of neural networks via adaptive subgradient methods. *arXiv preprint arXiv:1811.03679*, 2018.

Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 343–351, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Yi Sun, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber. Efficient natural evolution strategies. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 539–546, 2009.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Valentin Thomas, Fabian Pedregosa, Bart van Merriënboer, Pierre-Antoine Manzagol, Yoshua Bengio, and Nicolas Le Roux. On the interplay between noise and curvature and its effect on optimization and generalization. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3503–3513. PMLR, 26–28 Aug 2020.

Tijmen Tieleman and Geoffrey Hinton. RMSPROP: Divide the gradient by a running average of its recent magnitude. Coursera Online Course: Neural networks for machine learning, Lecture 6.5, 2012.

Vladimir Vapnik. Principles of risk minimization for learning theory. In J. Moody, S. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1992.

Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110): 3371–3408, 2010.

Yong Wang. Fisher scoring: An interpolation family and its Monte Carlo implementations. *Computational Statistics & Data Analysis*, 54(7):1744–1755, 2010.

Yeming Wen, Kevin Luk, Maxime Gazeau, Guodong Zhang, Harris Chan, and Jimmy Ba. An empirical study of stochastic gradient descent with structured covariance noise. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3621–3631. PMLR, 26–28 Aug 2020.

Daan Wierstra, Tom Schaul, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3381–3387, 2008.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Adams Wei Yu, Lei Huang, Qihang Lin, Ruslan Salakhutdinov, and Jaime Carbonell. Block-normalized gradient method: An empirical study for training deep neural network. *arXiv preprint arXiv:1707.04822*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12, September 2016.

Matthew D Zeiler. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5847–5856. PMLR, 2018.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Analysis of gradient clipping and adaptive scaling with a relaxed smoothness condition. *arXiv preprint arXiv:1905.11881*, 2019a.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J Reddi, Sanjiv Kumar, and Suvrit Sra. Why Adam beats SGD for attention models. *arXiv preprint arXiv:1912.03194*, 2019b.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7654–7663. PMLR, 2019.

# A
# Appendix to Chapter 4

## A.1 Proofs

*Proof of Proposition 4.1.* The maximizer is easily found by setting the partial derivative w.r.t. $\gamma$ to zero:

$$0 \stackrel{!}{=} \frac{\mathrm{d}}{\mathrm{d}\gamma}\left[\frac{\mathcal{G}(\gamma, m)}{m}\right]_{\gamma=\gamma_\star} = \frac{2}{m} - \frac{2\gamma_\star}{m} - \frac{2\eta^2\gamma_\star}{m^2}. \tag{A.1}$$

If we plug $\gamma_\star$ back in, we get

$$\begin{aligned}
\frac{\mathcal{G}(\gamma_\star, m)}{m} &= \frac{\gamma_\star}{m}\left(2 - \gamma_\star - \frac{\eta^2\gamma_\star}{m}\right) \\
&= \frac{1}{m(1+\eta^2/m)}\left(2 - \frac{1}{1+\eta^2/m} - \frac{\eta^2}{m(1+\eta^2/m)}\right) \\
&= \frac{1}{m+\eta^2}\left(2 - \frac{m}{m+\eta^2} - \frac{\eta^2}{m+\eta^2}\right) \\
&= \frac{1}{m+\eta^2}
\end{aligned} \tag{A.2}$$

□

*Proof of Proposition 4.2.* The maximizer is easily found by setting the partial derivative w.r.t. $m$ to zero:

$$0 \stackrel{!}{=} \frac{\mathrm{d}}{\mathrm{d}m}\left[\frac{\mathcal{G}(\gamma, m)}{m}\right]_{m=m_\star} = -2\frac{\gamma}{m_\star^2} + \frac{\gamma^2}{m_\star^2} + 2\frac{\eta^2\gamma^2}{m_\star^3}. \tag{A.3}$$

Multiplying by $m_\star^3/\gamma$ results in

$$0 = (-2+\gamma)m_\star + 2\eta^2\gamma. \tag{A.4}$$

Plugging $m_\star$ back in yields

$$\begin{aligned}
\frac{\mathcal{G}(\gamma, m_\star)}{m_\star} &= \frac{\gamma}{m_\star}\left(2 - \gamma - \frac{\eta^2\gamma}{m_\star}\right) \\
&= \gamma\frac{2-\gamma}{2\gamma\eta^2}\left(2 - \gamma - \gamma\eta^2\frac{2-\gamma}{2\gamma\eta^2}\right) \\
&= \frac{2-\gamma}{2\eta^2}\left(2 - \gamma - \frac{2-\gamma}{2}\right) \\
&= \frac{(2-\gamma)^2}{4\eta^2}
\end{aligned} \tag{A.5}$$

and solving for $m_\star$ yields the desired solution. □

*Proof of Equation (4.17).* By the Cauchy-Schwarz inequality, $\langle g, \nabla f \rangle = \langle g, g \rangle - \langle g, g - \nabla f \rangle \geq \|g\|^2 - \|g\| \|g - \nabla f\| = \|g\|(\|g\| - \|g - \nabla f\|)$. This becomes positive if $\|g - \nabla f\| < \|g\|$. □

## A.2 Experimental Details

*MNIST* Our network has two convolutional layers with 5×5 filters (32 and 64 filters, respectively) and subsequent max-pooling over 2×2 windows. This is followed by a fully-connected layer with 1024 units. The activation function is ReLU for all layers. The output layer has 10 units with softmax activation and we use cross-entropy loss.

*SVHN* We train a CNN with two convolutional layers, each with 64 filters of size 5×5 and subsequent max-pooling over 3×3 windows with stride 2. They are followed by two fully-connected layers with 256 and 128 units, respectively. The activation function is ReLU for all layers. The output layer has 10 units with softmax activation and we use cross-entropy loss. We apply $L_2$-regularization and perform data augmentation operations (random cropping of 24×24 pixel subimages, left-right mirroring, color distortion) on the training inputs.

*CIFAR-10 and CIFAR-100* For CIFAR-10, we crop the images to 24×24 pixels and train a CNN with two convolutional layers, each with 64 filters of size 5x5 and subsequent max-pooling over 3x3 windows with stride 2. They are followed by two fully-connected layers with 384 and 192 units, respectively. The activation function is ReLU for all layers. The output layer has 10 units with softmax activation and we use cross-entropy loss. We perform data augmentation operations (random cropping, left-right mirroring, color distortion) on the training set.

For CIFAR-100, we use the full 32×32 image and add a third convolutional layer (64 filters of size $5 \times 5$ followed by max pooling). The fully-connected layers have 512 and 256 units, respectively, and the output layer has 100 units. We add $L_2$-regularization.

# B
# Appendix to Chapter 5

## B.1 Experimental Details

### B.1.1 Network Architectures

*Fashion-MNIST* We trained a simple convolutional neural network with two convolutional layers (size 5×5, 32 and 64 filters, respectively), each followed by max-pooling over 3×3 areas with stride 2, and a fully-connected layer with 1024 units. ReLU activation was used for all layers. The output layer has 10 units with softmax activation. We used cross-entropy loss, without any additional regularization, and a minibatch size of 64. We trained for a total of 6000 steps with a constant global step size $\alpha$.

*CIFAR-10* We trained a CNN with three convolutional layers (64 filters of size 5×5, 96 filters of size 3×3, and 128 filters of size 3×3) interspersed with max-pooling over 3×3 areas with stride 2 and followed by two fully-connected layers with 512 and 256 units. ReLU activation was used for all layers. The output layer has 10 units with softmax activation. We used cross-entropy loss function and applied $L_2$-regularization on all weights, but not the biases. During training we performed some standard data augmentation operations (random cropping of sub-images, left-right mirroring, color distortion) on the input images. We used a batch size of 128 and trained for a total of 40k steps with a constant global step size $\alpha$.

*CIFAR-100* We use the WRN-40-4 architecture of Zagoruyko and Komodakis [2016]; details can be found in the original paper. We used cross-entropy loss and applied $L_2$-regularization on all weights, but not the biases. We used the same data augmentation operations as for CIFAR-10, a batch size of 128, and trained for 80k steps. For the global step size $\alpha$, we used the decrease schedule suggested by Zagoruyko and Komodakis [2016], which amounts to multiplying with a factor of 0.2 after 24k, 48k, and 64k steps. TensorFlow code was adapted from `https://github.com/dalgu90/wrn-tensorflow`.

*War and Peace* We preprocessed *War and Peace*, extracting a vocabulary of 83 characters. The language model is a two-layer LSTM with 128 hidden units each. We used a sequence length of 50 characters and a batch size of 50. Drop-out regularization was applied during training. We trained for 200k steps; the global step size $\alpha$ was multiplied with a factor of 0.1 after 125k steps. TensorFlow code was adapted from `https://github.com/sherjilozair/char-rnn-tensorflow`.

### *B.1.2 Step Size Tuning*

Step sizes $\alpha$ (initial step sizes for the experiments with a step size decrease schedule) for each optimizer have been tuned by first finding the maximal stable step size by trial and error and then searching downwards over multiple orders of magnitude, testing $6 \cdot 10^m$, $3 \cdot 10^m$, and $1 \cdot 10^m$ for order of magnitude $m$. We evaluated loss and accuracy on the full test set (as well as on an equally-sized portion of the training set) at a constant interval and selected the best-performing step size for each method in terms of maximally reached test accuracy. Using the best choice, we replicated the experiment ten times with different random seeds, randomizing the parameter initialization, data set shuffling, drop-out, et cetera. In some rare cases where the accuracies for two different step sizes were very close, we replicated both and then chose the one with the higher maximum mean accuracy.

The following list shows all explored step sizes, with the "winner" in boldface.

**Problem 1: Fashion-MNIST**

M-SGD:

$3, 1, 6 \cdot 10^{-1}, 3 \cdot 10^{-1}, \mathbf{1 \cdot 10^{-1}}, 6 \cdot 10^{-2}, 3 \cdot 10^{-2}, 1 \cdot 10^{-2}, 6 \cdot 10^{-3}, 3 \cdot 10^{-3}$

ADAM:

$3 \cdot 10^{-2}, 10^{-2}, 6 \cdot 10^{-3}, 3 \cdot 10^{-3}, \mathbf{1 \cdot 10^{-3}}, 6 \cdot 10^{-4}, 3 \cdot 10^{-4}, 1 \cdot 10^{-4}$

M-SSD:

$10^{-2}, 6 \cdot 10^{-3}, 3 \cdot 10^{-3}, 1 \cdot 10^{-3}, 6 \cdot 10^{-4}, \mathbf{3 \cdot 10^{-4}}, 1 \cdot 10^{-4}$

M-SVAG:

$3, 1, 6 \cdot 10^{-1}, \mathbf{3 \cdot 10^{-1}}, 1 \cdot 10^{-1}, 6 \cdot 10^{-2}, 3 \cdot 10^{-2}, 1 \cdot 10^{-2}, 6 \cdot 10^{-3}, 3 \cdot 10^{-3}$

**Problem 2: CIFAR-10**

M-SGD:

$6 \cdot 10^{-1}, 3 \cdot 10^{-1}, 1 \cdot 10^{-1}, 6 \cdot 10^{-2}, \mathbf{3 \cdot 10^{-2}}, 1 \cdot 10^{-2}, 6 \cdot 10^{-3}, 3 \cdot 10^{-3}$

ADAM:

$6 \cdot 10^{-3}, 3 \cdot 10^{-3}, 1 \cdot 10^{-3}, \mathbf{6 \cdot 10^{-4}}, 3 \cdot 10^{-4}, 1 \cdot 10^{-4}, 6 \cdot 10^{-5}$

M-SSD:

$6 \cdot 10^{-3}, 3 \cdot 10^{-3}, 1 \cdot 10^{-3}, 6 \cdot 10^{-4}, 3 \cdot 10^{-4}, \mathbf{1 \cdot 10^{-4}}, 6 \cdot 10^{-5}, 3 \cdot 10^{-5}$

M-SVAG:

$1, 6 \cdot 10^{-1}, 3 \cdot 10^{-1}, 1 \cdot 10^{-1}, \mathbf{6 \cdot 10^{-2}}, 3 \cdot 10^{-2}, 1 \cdot 10^{-2}, 6 \cdot 10^{-3}$

**Problem 3: CIFAR-100**

M-SGD:

$6, \mathbf{3}, 1, 6 \cdot 10^{-1}, 3 \cdot 10^{-1}, 1 \cdot 10^{-1}, 6 \cdot 10^{-2}, \mathbf{3 \cdot 10^{-2}}, 1 \cdot 10^{-2}$

ADAM:

$1 \cdot 10^{-2}, 6 \cdot 10^{-3}, 3 \cdot 10^{-3}, 1 \cdot 10^{-3}, 6 \cdot 10^{-4}, \mathbf{3 \cdot 10^{-4}}, 1 \cdot 10^{-4}, 6 \cdot 10^{-5}, 3 \cdot 10^{-5}$

M-SSD:

$1 \cdot 10^{-2}, 6 \cdot 10^{-3}, 3 \cdot 10^{-3}, 1 \cdot 10^{-3}, 6 \cdot 10^{-4}, 3 \cdot 10^{-4}, \mathbf{1 \cdot 10^{-4}}, 6 \cdot 10^{-5}, 3 \cdot 10^{-5}$

M-SVAG:

$6, \mathbf{3}, 1, 6 \cdot 10^{-1}, 3 \cdot 10^{-1}, 1 \cdot 10^{-1}, 6 \cdot 10^{-2}, \mathbf{3 \cdot 10^{-2}}, 1 \cdot 10^{-2}$

**Problem 4: War and Peace**

M-SGD:

$10, 6, \mathbf{3}, 1, 6 \cdot 10^{-1}, 3 \cdot 10^{-1}, 1 \cdot 10^{-1}, 6 \cdot 10^{-2}$

ADAM:

$1 \cdot 10^{-2}, 6 \cdot 10^{-3}, \mathbf{3 \cdot 10^{-3}}, 1 \cdot 10^{-3}, 6 \cdot 10^{-4}, 3 \cdot 10^{-4}, 1 \cdot 10^{-4}, 6 \cdot 10^{-5}$

M-SSD:

$1 \cdot 10^{-2}, 6 \cdot 10^{-3}, 3 \cdot 10^{-3}, \mathbf{1 \cdot 10^{-3}}, 6 \cdot 10^{-4}, 3 \cdot 10^{-4}, 1 \cdot 10^{-4}, 6 \cdot 10^{-5}$

M-SVAG:

$30, \mathbf{10}, 6, 3, 1, 6 \cdot 10^{-1}, 3 \cdot 10^{-1}, 1 \cdot 10^{-1}$

## *B.2 Mathematical Details*

### *B.2.1 The Sign of a Stochastic Gradient*

We have stated in the main text that the sign of a stochastic gradient, $s := \text{sign}(g)$, has success probabilities

$$\rho_i := \mathbf{P}[s_i = \text{sign}(\nabla f_i)] = \frac{1}{2} + \frac{1}{2} \operatorname{erf}\left(\frac{|\nabla f_i|}{\sqrt{2}\sigma_i}\right) \tag{B.1}$$

under the assumption that $g \sim \mathcal{N}(\nabla f, \Sigma)$ and $\sigma_i^2 = \Sigma_{ii}$. The following Lemma formally proves this statement and Figure B.1 provides a pictorial illustration.

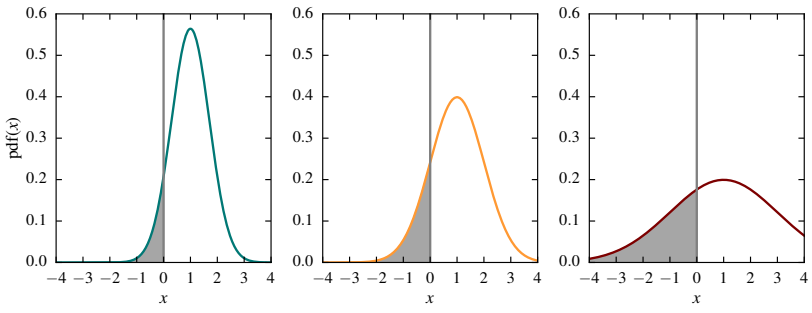


Figure B.1: Probability density functions (pdf) of three Gaussian distributions, all with $\mu = 1$, but different variances $\sigma^2 = 0.5$ (left), $\sigma^2 = 1.0$ (middle), $\sigma^2 = 4.0$ (right). The shaded area under the curve corresponds to the probability that a sample from the distribution has the opposite sign than its mean. For the Gaussian distribution, this probability is uniquely determined by the fraction $\sigma/|\mu|$, as shown in Lemma B.1.

**Lemma B.1.** *If* $X \sim \mathcal{N}(\mu, \sigma^2)$ *then*

$$\mathbf{P}[\text{sign}(X) = \text{sign}(\mu)] = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{|\mu|}{\sqrt{2}\sigma}\right)\right). \tag{B.2}$$

*Proof.* Define $\rho := \mathbf{P}[\text{sign}(X) = \text{sign}(\mu)]$. The cumulative density function (cdf) of $X \sim \mathcal{N}(\mu, \sigma^2)$ is $\mathbf{P}[X \leq x] = \Phi((x - \mu)/\sigma)$,

where $\Phi(z) = 0.5(1 + \text{erf}(z/\sqrt{2}))$ is the cdf of the standard normal distribution. If $\mu < 0$, then

$$\rho = \mathbf{P}[X < 0] = \Phi\left(\frac{0-\mu}{\sigma}\right) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{-\mu}{\sqrt{2}\sigma}\right)\right). \tag{B.3}$$

If $\mu > 0$, then

$$\begin{aligned}\rho = \mathbf{P}[X > 0] &= 1 - \mathbf{P}[X \leq 0] = 1 - \Phi\left(\frac{0-\mu}{\sigma}\right) \\ &= 1 - \frac{1}{2}\left(1 + \text{erf}\left(\frac{-\mu}{\sqrt{2}\sigma}\right)\right) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{\mu}{\sqrt{2}\sigma}\right)\right),\end{aligned} \tag{B.4}$$

where the last step used the anti-symmetry of the error function.

□

### *B.2.2 Variance Adaptation Factors*

*Proof of Lemma 5.1.* Using $\mathbf{E}[\hat{p}_i] = p_i$ and $\mathbf{E}[\hat{p}_i^2] = p_i^2 + \sigma_i^2$, we get

$$\begin{aligned}\mathbf{E}[\|\gamma \odot \hat{p} - p\|_2^2] &= \sum_{i=1}^{d} \mathbf{E}[(\gamma_i \hat{p}_i - p_i)^2] \\ &= \sum_{i=1}^{d} \left(\gamma_i^2 \mathbf{E}[\hat{p}_i^2] - 2\gamma_i p_i \mathbf{E}[\hat{p}_i] + p_i^2\right) \\ &= \sum_{i=1}^{d} \left(\gamma_i^2 (p_i^2 + \sigma_i^2) - 2\gamma_i p_i^2 + p_i^2\right).\end{aligned} \tag{B.5}$$

Setting the derivative w.r.t. $\gamma_i$ to zero, we find the optimal choice

$$\gamma_i = \frac{p_i^2}{p_i^2 + \sigma_i^2}. \tag{B.6}$$

□

*Proof of Lemma 5.2.* Using $\mathbf{E}[\text{sign}(\hat{p}_i)] = (2\rho_i - 1)\,\text{sign}(p_i)$ and $\text{sign}(\cdot)^2 = 1$, we get

$$\begin{aligned}\mathbf{E}[\|\gamma \odot \text{sign}(\hat{p}) - \text{sign}(p)\|_2^2] &= \sum_{i=1}^{d} \mathbf{E}\left[(\gamma_i \,\text{sign}(\hat{p}_i) - \text{sign}(p_i))^2\right] \\ &= \sum_{i=1}^{d} \left(\gamma_i^2 - 2\gamma_i \,\text{sign}(p_i)\mathbf{E}[\text{sign}(\hat{p}_i)] + 1\right) \\ &= \sum_{i=1}^{d} \left(\gamma_i^2 - 2\gamma_i(2\rho_i - 1) + 1\right)\end{aligned} \tag{B.7}$$

and easily find the optimal choice

$$\gamma_i = 2\rho_i - 1. \tag{B.8}$$

by setting the derivative to zero. □

### B.2.3 Convergence of Idealized SVAG

We prove the convergence results for idealized variance-adapted stochastic gradient descent (Theorem 5.1). The stochastic optimizer generates a discrete stochastic process $\{\theta_t\}_{t\in\mathbb{N}_0}$. We denote as $\mathbf{E}_t[\cdot] = \mathbf{E}[\cdot|\theta_t]$ the conditional expectation given a realization of that process up to time step $t$. Recall that $\mathbf{E}[\mathbf{E}_t[\cdot]] = \mathbf{E}[\cdot]$.

We first show the following Lemma.

**Lemma B.2.** *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be $\mu$-strongly convex and $L$-smooth. Denote as $\theta_* := \arg\min_{\theta\in\mathbb{R}^d} f(\theta)$ the unique minimizer and $f_* = f(\theta_*)$. Then, for any $\theta \in \mathbb{R}^d$,*

$$\frac{2L^2}{\mu}(f(\theta) - f_*) \geq \|\nabla f(\theta)\|^2 \geq 2\mu(f(\theta) - f_*). \tag{B.9}$$

*Proof.* Regarding the first inequality, we use $\nabla f(\theta_*) = 0$ and the Lipschitz continuity of $\nabla f(\cdot)$ to get $\|\nabla f(\theta)\|^2 = \|\nabla f(\theta) - \nabla f(\theta_*)\|^2 \leq L^2\|\theta - \theta_*\|^2$. Using strong convexity, we have $f(\theta) \geq f_* + \nabla f(\theta_*)^\mathsf{T}(\theta - \theta_*) + (\mu/2)\|\theta - \theta_*\|^2 = f_* + (\mu/2)\|\theta - \theta_*\|^2$. Plugging the two inequalities together yields the desired inequality.

The second inequality arises from strong convexity, by minimizing both sides of

$$f(\theta') \geq f(\theta) + \nabla f(\theta)^\mathsf{T}(\theta' - \theta) + \frac{\mu}{2}\|\theta' - \theta\|^2 \tag{B.10}$$

w.r.t. $\theta'$. The left-hand side obviously has minimal value $f_*$. For the right-hand side, we set its derivative, $\nabla f(\theta) + \mu(\theta' - \theta)$, to zero to find the minimizer $\theta' = \theta - \nabla f(\theta)/\mu$. Plugging that back in yields the minimal value $f(\theta) - \|\nabla f(\theta)\|/(2\mu)$. □

*Proof of Theorem 5.1.* Using the Lipschitz continuity of $\nabla f$, we can bound $f(\theta + \Delta\theta) \leq f(\theta) + \nabla f(\theta)^\mathsf{T}\Delta\theta + \frac{L}{2}\|\Delta\theta\|^2$. Hence,

$$\begin{aligned}
\mathbf{E}_t[f_{t+1}] &\leq f_t - \alpha\mathbf{E}_t[\nabla f_t^\mathsf{T}(\gamma_t \odot g_t)] + \frac{L\alpha^2}{2}\mathbf{E}_t[\|\gamma_t \odot g_t\|^2] \\
&= f_t - \frac{1}{L}\sum_{i=1}^d \gamma_{t,i}\nabla f_{t,i}\mathbf{E}_t[g_{t,i}] + \frac{1}{2L}\sum_{i=1}^d \gamma_{t,i}^2\mathbf{E}_t[g_{t,i}^2] \\
&= f_t - \frac{1}{L}\sum_{i=1}^d \gamma_{t,i}\nabla f_{t,i}^2 + \frac{1}{2L}\sum_{i=1}^d \gamma_{t,i}^2(\nabla f_{t,i}^2 + \sigma_{t,i}^2).
\end{aligned} \tag{B.11}$$

Plugging in the definition $\gamma_{t,i} = \nabla f_{t,i}^2/(\nabla f_{t,i}^2 + \sigma_{t,i}^2)$ and simplifying, we get

$$\mathbf{E}_t[f_{t+1}] \leq f_t - \frac{1}{2L}\sum_{i=1}^d \frac{\nabla f_{t,i}^4}{\nabla f_{t,i}^2 + \sigma_{t,i}^2}. \tag{B.12}$$

This shows that $\mathbf{E}_t[f_{t+1}] \leq f_t$. Defining $e_t := f_t - f_*$, this implies

$$\mathbf{E}[e_{t+1}] = \mathbf{E}[\mathbf{E}_t[e_{t+1}]] \leq \mathbf{E}[e_t] \tag{B.13}$$

and consequently, by iterating backwards, $\mathbf{E}[e_t] \leq \mathbf{E}[e_0] = e_0$ for all $t$. Next, using the discrete version of Jensen's inequality[1] we find

$$\sum_{i=1}^d \frac{\nabla f_{t,i}^4}{\nabla f_{t,i}^2 + \sigma_{t,i}^2} \geq \frac{\|\nabla f_t\|^4}{\|\nabla f_t\|^2 + \sum_{i=1}^d \sigma_{t,i}^2}. \tag{B.14}$$

[1] Jensen's inequality states that, for a real convex function $\phi$, numbers $x_i \in \mathbb{R}$, and positive weights $a_i \in \mathbb{R}_+$ with $\sum_i a_i = 1$, we have $\sum_i a_i\phi(x_i) \geq \phi\left(\sum_i a_i x_i\right)$. We apply it here to the convex function $\phi(x) = 1/x, x > 0$, with $x_i := \frac{\nabla f_{t,i}^2 + \sigma_{t,i}^2}{\nabla f_{t,i}^2}$ and $a_i := \frac{\nabla f_{t,i}^2}{\|\nabla f_t\|^2}$.

Using the assumption $\sum_{i=1}^{d} \sigma_{t,i}^2 \leq c_v \|\nabla f_t\|^2 + M_v$ in the denominator, we obtain

$$\frac{\|\nabla f_t\|^4}{\|\nabla f_t\|^2 + \sum_{i=1}^{d} \sigma_{t,i}^2} \geq \frac{\|\nabla f_t\|^4}{(1+c_v)\|\nabla f_t\|^2 + M_v}. \tag{B.15}$$

Using Lemma B.2, we have

$$\frac{2L^2}{\mu} e_t \geq \|\nabla f_t\|^2 \geq 2\mu e_t \tag{B.16}$$

and can further bound

$$\frac{\|\nabla f_t\|^4}{(1+c_v)\|\nabla f_t\|^2 + M_v} \geq \frac{4\mu^2 e_t^2}{\frac{2(1+c_v)L^2}{\mu} e_t + M_v} \stackrel{\text{def}}{=} \frac{c_1 e_t^2}{c_2 e_t + c_3}, \tag{B.17}$$

where the last equality defines the (positive) constants $c_1, c_2$ and $c_3$. Combining Eqs. (B.14), (B.15) and (B.17), inserting in (B.12), and subtracting $f_*$ from both sides, we obtain

$$\mathbf{E}_t[e_{t+1}] \leq e_t - \frac{1}{2L} \frac{c_1 e_t^2}{c_2 e_t + c_3}, \tag{B.18}$$

and, consequently, by taking expectations on both sides,

$$\begin{aligned} \mathbf{E}[e_{t+1}] &\leq \mathbf{E}[e_t] - \frac{1}{2L} \mathbf{E}\left[\frac{c_1 e_t^2}{c_2 e_t + c_3}\right] \\ &\leq \mathbf{E}[e_t] - \frac{1}{2L} \frac{c_1 \mathbf{E}[e_t]^2}{c_2 \mathbf{E}[e_t] + c_3} \end{aligned} \tag{B.19}$$

where the last step is due to Jensen's inequality applied to the convex function $\phi(x) = \frac{c_1 x^2}{c_2 x + c_3}$. Using $\mathbf{E}[e_t] \leq e_0$ in the denominator and introducing the shorthand $\bar{e}_t := \mathbf{E}[e_t]$, we get

$$\bar{e}_{t+1} \leq \bar{e}_t - c\bar{e}_t^2 = \bar{e}_t(1 - c\bar{e}_t), \tag{B.20}$$

with $c := c_1/(2L(c_2 e_0 + c_3)) > 0$. To conclude the proof, we will show that this implies $\bar{e}_t \in \mathcal{O}(\frac{1}{t})$. Without loss of generality, we assume $\bar{e}_{t+1} > 0$ and obtain

$$\bar{e}_{t+1}^{-1} \geq \bar{e}_t^{-1} (1 - c\bar{e}_t)^{-1} \geq \bar{e}_t^{-1} (1 + c\bar{e}_t) = \bar{e}_t^{-1} + c, \tag{B.21}$$

where the second step is due to the simple fact that $(1-x)^{-1} \geq (1+x)$ for any $x \in [0,1)$. Summing this inequality over $t = 0, \ldots, T-1$ yields $\bar{e}_T^{-1} \geq e_0^{-1} + Tc$ and, thus,

$$T\bar{e}_T \leq \left(\frac{1}{Te_0} + c\right)^{-1} \stackrel{T\to\infty}{\longrightarrow} \frac{1}{c} < \infty, \tag{B.22}$$

which shows that $\bar{e}_t \in \mathcal{O}(\frac{1}{t})$. □

### *B.2.4 Gradient Variance Estimates via Moving Averages*

We proof Eq. (5.14). Iterating the recursive formula for $\tilde{m}_t$ backwards, we get

$$m_t = \sum_{s=0}^{t} \underbrace{\frac{1-\beta_1}{1-\beta_1^{t+1}} \beta_1^{t-s}}_{=:c(\beta_1,t,s)} g_s, \tag{B.23}$$

with coefficients $c(\beta_1, t, s)$ summing to one by the geometric sum formula, making $m_t$ a convex combination of stochastic gradients. Likewise, $v_t = \sum_{s=0}^{t} c(\beta_2, t, s) g_s^2$ is a convex combination of squared stochastic gradients. Hence,

$$\begin{aligned} \mathbf{E}[m_{t,i}] &= \sum c(\beta, t, s) \mathbf{E}[g_{s,i}], \\ \mathbf{E}[v_{t,i}] &= \sum c(\beta, t, s) \mathbf{E}[g_{s,i}^2]. \end{aligned} \tag{B.24}$$

Assumption 5.1 thus necessarily implies $\mathbf{E}[g_{s,i}] \approx \nabla f_{t,i}$ and $\mathbf{E}[g_{s,i}^2] \approx \nabla f_{t,i}^2 + \sigma_{t,i}^2$. (This will of course be utterly wrong for gradient observations that are far in the past, but these won't contribute significantly to the moving average.) It follows that

$$\begin{aligned} \mathbf{E}[m_{t,i}^2] &= \mathbf{E}[m_{t,i}]^2 + \mathbf{Var}[m_{t,i}] \\ &= \nabla f_{t,i}^2 + \sum_{s=0}^{t} c(\beta, t, s)^2 \, \mathbf{Var}[g_{s,i}] \\ &= \nabla f_{t,i}^2 + \sigma_{t,i}^2 \sum_{s=0}^{t} c(\beta, t, s)^2, \end{aligned} \tag{B.25}$$

where the second step is due to the fact that $g_s$ and $g_{s'}$ are stochastically independent for $s \neq s'$. The last term evaluates to

$$\begin{aligned} \rho(\beta, t) &:= \sum_{s=0}^{t} c(\beta, t, s)^2 = \sum_{s=0}^{t} \left( \frac{1-\beta}{1-\beta^{t+1}} \beta^{t-s} \right)^2 \\ &= \frac{(1-\beta)^2}{(1-\beta^{t+1})^2} \sum_{k=0}^{t} (\beta^2)^k \\ &= \frac{(1-\beta)^2}{(1-\beta^{t+1})^2} \, \frac{1-(\beta^2)^{t+1}}{1-\beta^2} \\ &= \frac{(1-\beta)(1-\beta)}{(1-\beta^{t+1})(1-\beta^{t+1})} \, \frac{(1-\beta^{t+1})(1+\beta^{t+1})}{(1-\beta)(1+\beta)} \\ &= \frac{(1-\beta)(1+\beta^{t+1})}{(1+\beta)(1-\beta^{t+1})}, \end{aligned} \tag{B.26}$$

where the fourth step is another application of the geometric sum formula, and the fifth step uses $1 - x^2 = (1-x)(1+x)$. Note that

$$\rho(\beta, t) \to \frac{1-\beta}{1+\beta} \quad (t \to \infty), \tag{B.27}$$

such that $\rho(\beta, t)$ is uniquely defined by $\beta$ in the long term.

As an interesting side note, the division by $1 - \rho(\beta, t)$ in Eq. (5.16) is analogous to Bessel's correction (the use of $n-1$ instead of $n$ in the classical sample variance) for the case where we use moving averages instead of arithmetic means.

### *B.2.5 Connection to Generalization*

*Proof of Lemma 5.4.* Like in the proof of Lemma 3.1 in Wilson et al. [2017], we inductively show that $\theta_t = \lambda_t \operatorname{sign}(X^\mathsf{T} y)$ with a scalar $\lambda_t$. This trivially holds for $\theta_0 = 0$. Assume that the assertion holds for

all $s \leq t$. Then

$$\begin{aligned}\nabla R(\theta_t) &= \frac{1}{n} X^\mathsf{T}(X\theta_t - y) \\ &= \frac{1}{n} X^\mathsf{T}(\lambda_t X \operatorname{sign}(X^\mathsf{T} y) - y) \\ &= \frac{1}{n} X^\mathsf{T}(\lambda_t c y - y) = \frac{1}{n}(\lambda_t c - 1) X^\mathsf{T} y,\end{aligned} \tag{B.28}$$

where the first step is the gradient of the objective (Eq. 5.20), the second step uses the inductive assumption, and the third step uses the assumption $X \operatorname{sign}(X^\mathsf{T} y) = cy$. Now, plugging Eq. (B.28) into the update rule, we find

$$\begin{aligned}\theta_{t+1} &= \theta_t - \alpha \operatorname{sign}(\nabla R(\theta_t)) \\ &= \lambda_t \operatorname{sign}(X^\mathsf{T} y) - \alpha \operatorname{sign}((\lambda_t c - 1) X^\mathsf{T} y) \\ &= (\lambda_t - \alpha \operatorname{sign}(\lambda_t c - 1)) \operatorname{sign}(X^\mathsf{T} y).\end{aligned} \tag{B.29}$$

Hence, the assertion holds for $t + 1$. □

## *B.3 Alternative Methods*

### *B.3.1 SVAG*

M-SVAG applies variance adaptation to the update direction $m_t$, resulting in the variance adaptation factors Eq. 5.19. We can also update in direction $g_t$ and choose the appropriate estimated variance adaptation factors, resulting in an implementation of SVAG without momentum. We have already derived the necessary variance adaptation factors en route to those for the momentum variant, see Eq. (5.17) in §5.4.2. Pseudo-code is provided in Alg. 4. It differs from M-SVAG only in the last two lines.

**Input:** $\theta_0 \in \mathbb{R}^d$, $\alpha > 0$, $\beta \in [0, 1]$, $T \in \mathbb{N}$
Initialize $\theta \leftarrow \theta_0$, $\tilde{m} \leftarrow 0$, $\tilde{v} \leftarrow 0$
**for** $t = 0, \dots, T-1$ **do**
  $\tilde{m} \leftarrow \beta\tilde{m} + (1-\beta) g(\theta), \quad \tilde{v} \leftarrow \beta\tilde{v} + (1-\beta) g(\theta)^2$
  $m \leftarrow (1-\beta^{t+1})^{-1}\tilde{m}, \quad v \leftarrow (1-\beta^{t+1})^{-1}\tilde{v}$
  $s \leftarrow (1 - \rho(\beta, t))^{-1}(v - m^2)$
  $\gamma \leftarrow m^2/(m^2 + s)$
  $\theta \leftarrow \theta - \alpha(\gamma \odot g)$
**end for**

**Algorithm 4:** SVAG

### *B.3.2 Variants of Adam*

This paper interpreted ADAM as variance-adapted M-SSD. The experiments in the main paper used a standard implementation of ADAM as described by Kingma and Ba [2015]. However, in the derivation of our implementation of M-SVAG, we have made multiple adjustments regarding the estimation of variance adaptation

factors which correspondingly apply to the sign case. Specifically, this concerns:

- The use of the same moving average constant for the first and second moment ($\beta_1 = \beta_2 = \beta$).
- The bias correction in the gradient variance estimate, see Eq. (5.16).
- The adjustment of the variance adaptation factors for the momentum case, see §5.4.3.
- The omission of a constant offset $\varepsilon$ in the denominator.

Applying these adjustment to the sign case gives rise to a variant of the original ADAM algorithm, which we will refer to as ADAM*. Pseudo-code is provided in Alg. 5. Note that we use the variance adaptation factors $(1+\eta)^{-1/2}$ and *not* the optimal ones derived in §5.3.1, which would under the Gaussian assumption be $\mathrm{erf}[(\sqrt{2}\eta)^{-1}]$. We initially experimented with both variants and found them to perform almost identically, which is not surprising given how similar the two are (see Fig. 5.2). We thus stuck with the first option for direct correspondence with the original ADAM and to avoid the cumbersome error function.

In analogy to SVAG versus M-SVAG, we could also define a variance-adapted version stochastic sign descent *without* momentum, i.e., using the base update direction $\mathrm{sign}(g_t)$. We did not explore this further in this work.

**Input:** $\theta_0 \in \mathbb{R}^d$, $\alpha > 0$, $\beta \in [0,1]$, $T \in \mathbb{N}$
Initialize $\theta \leftarrow \theta_0$, $\tilde{m} \leftarrow 0$, $\tilde{v} \leftarrow 0$
**for** $t = 0, \dots, T-1$ **do**
  $\tilde{m} \leftarrow \beta\tilde{m} + (1-\beta)g(\theta), \quad \tilde{v} \leftarrow \beta\tilde{v} + (1-\beta)g(\theta)^2$
  $m \leftarrow (1-\beta^{t+1})^{-1}\tilde{m}, \quad v \leftarrow (1-\beta^{t+1})^{-1}\tilde{v}$
  $s \leftarrow (1-\rho(\beta,t))^{-1}(v-m^2)$
  $\gamma \leftarrow \sqrt{m^2/(m^2+\rho(\beta,t)s)}$
  $\theta \leftarrow \theta - \alpha(\gamma \odot \mathrm{sign}(m))$
**end for**

**Algorithm 5:** ADAM*

### *B.3.3 Experiments*

We tested SVAG as well as ADAM* with and without momentum on the problems (P2) and (P3) from the main paper. Results are shown in Figure B.2.

We observe that SVAG performs better than M-SVAG on (P2). On (P3), it makes faster initial progress but later plateaus, leading to slightly worse outcomes in both training loss and test accuracy. SVAG is a viable alternative. In future work, it will be interesting to apply SVAG to problems where SGD outperforms M-SGD.

Next, we compare ADAM* to the original ADAM algorithm. In the CIFAR-100 example (P3) the two methods are on par. On (P2),

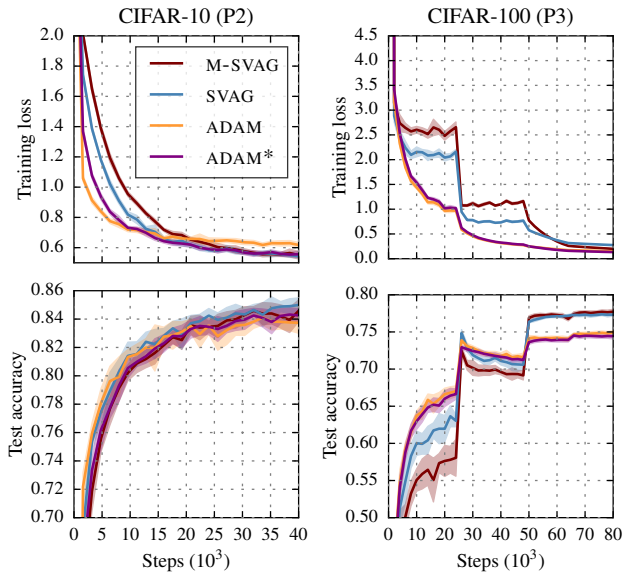


Figure B.2: Experimental results for SVAG and ADAM*. The plot is set-up like Fig. 5.4.

ADAM is marginally faster in the early stages of the the optimization process. ADAM* quickly catches up and reaches lower minimal training loss values. We conclude that the adjustments to the variance adaptation factors derived in §5.4 do have a positive effect.

## *B.4 Minibatch Gradient Variance Estimates*

In the main text, we have discussed estimation of gradient variances via moving averages of the past gradient observations. An alternative gradient variance estimate can be obtained locally, within a single minibatch. Given $\theta_t$, the individual gradients $\nabla f(\theta_t; \xi_i^{(t)})$, $i = 1, \dots m$, in a minibatch are iid random variables with covariance matrix $\Sigma(\theta_t)$ and, thus,

$$\mathbf{Cov}_t[g_t] = \frac{\Sigma(\theta_t)}{m}. \tag{B.30}$$

We can thus estimate the covariance of $g_t$ based on the sample covariance of the $\{\nabla f(\theta; \xi_i^{(t)})\}_{i=1}^m$. We are only interested in its diagonal, $\mathbf{Var}[X] := \mathrm{diag}(\mathbf{Cov}[X]) \in \mathbb{R}^d$, i.e., the vector containing its elementwise variances. We can estimate it via

$$\hat{s}^{\mathrm{mb}}(\theta_t) \stackrel{\mathrm{def}}{=} \frac{1}{m}\left(\frac{1}{m-1}\sum_{i=1}^{m} \nabla f(\theta; \xi_i)^2 - g_t^2\right). \tag{B.31}$$

Several recent papers [Mahsereci and Hennig, 2015, Balles et al., 2017a] have used this variance estimate for other aspects of stochastic optimization. In contrast to the moving average-based estimators, this is an unbiased estimate of the *local* gradient variance. The (non-trivial) implementation of this estimator for neural networks is described in Balles et al. [2017a].

### B.4.1 M-SVAG with Mini-Batch Estimates

We explored a variant of M-SVAG which use minibatch gradient variance estimates. The local variance estimation allows for a theoretically more pleasing treatment of the variance of the update direction $m_t$. Starting from the formulation of $m_t$ in Eq. (B.23) and considering that $g_s$ and $g_{s'}$ are stochastically independent for $s \neq s'$, we have

$$\mathbf{Var}[m_t] = \sum_{s=0}^{t} \left( \frac{1-\beta}{1-\beta^{t+1}} \beta^{t-s} \right)^2 \mathbf{Var}[g_s]. \tag{B.32}$$

Given that we now have access to a true, local, unbiased estimate of $\mathbf{Var}[g_s]$, we can estimate $\mathbf{Var}[m_t]$ by

$$\bar{s}_t := \sum_{s=0}^{t} \left( \frac{1-\beta}{1-\beta^{t+1}} \beta^{t-s} \right)^2 \hat{s}^{\mathrm{mb}}(\theta_s). \tag{B.33}$$

It turns out that we can track this quantity with another exponential moving average: It is $\bar{s}_t = \rho(\beta, t) r_t$ with

$$\tilde{r}_t = \beta^2 \tilde{r}_{t-1} + (1-\beta^2)\hat{s}_t^{\mathrm{mb}}, \quad r_t = \frac{\tilde{r}_t}{1-(\beta^2)^{t+1}}. \tag{B.34}$$

This can be shown by iterating Eq. (B.34) backwards and comparing coefficients with Eq. (B.33). The resulting minibatch variant of M-SVAG is presented in Algorithm 6.

Note that minibatch gradient variance estimates could likewise be used for the alternative methods discussed in §B.3. We do not explore this further in this work.

### B.4.2 Experiments

We tested the minibatch variant of M-SVAG on the problems (P1) and (P2) from the main text and compared it to the moving average version. Results are shown in Figure B.3. The two algorithms have almost identical performance.

**Input:** $\theta_0 \in \mathbb{R}^d$, $\alpha > 0$, $\beta \in [0,1]$, $T \in \mathbb{N}$
Initialize $\theta \leftarrow \theta_0$, $\tilde{m} \leftarrow 0$, $\tilde{r} \leftarrow 0$
**for** $t = 0, \dots, T-1$ **do**
  Compute minibatch gradient $g(\theta)$ and variance $\hat{s}^{\mathrm{mb}}(\theta)$
  $\tilde{m} \leftarrow \beta\tilde{m} + (1-\beta)g(\theta), \quad \tilde{r} \leftarrow \beta^2\tilde{r} + (1-\beta^2)\hat{s}^{\mathrm{mb}}(\theta)$
  $m \leftarrow (1-\beta^{t+1})^{-1}\tilde{m}, \quad r \leftarrow (1-\beta^{2(t+1)})^{-1}\tilde{r}$
  $\gamma \leftarrow m^2/(m^2 + \rho(\beta,t)r)$
  $\theta \leftarrow \theta - \alpha(\gamma \odot m)$
**end for**
**Algorithm 6:** M-SVAG with minibatch variance estimate

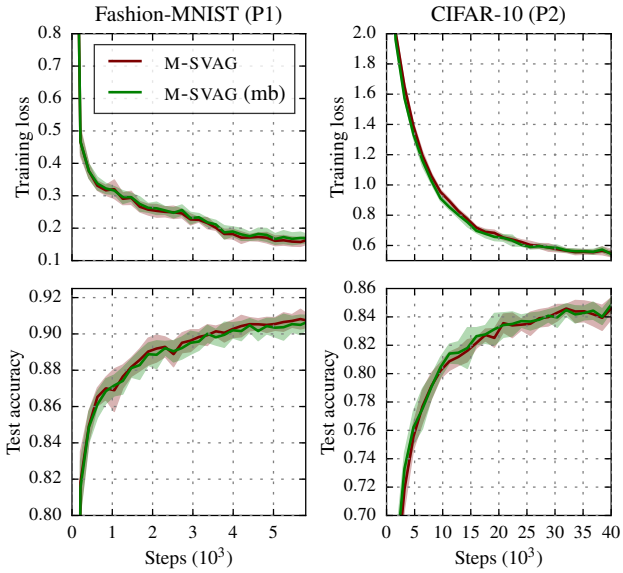


Figure B.3: Experimental results for the minibatch variant of M-SVAG (marked "mb" in the legend). The plot is set-up like Fig. 5.4.

# C
# Appendix to Chapter 6

- Appendix C.1 extends the discussion of steepest descent methods.
- Appendix C.2 gives details on the two-dimensional quadratic example used in Section 6.4.
- Appendix C.3 discusses normalized steepest descent methods as a possible explanation for the discrepancy between Eq. (6.1) and Eq. (6.10).
- Appendix C.4 provides details on the experiments.
- All proofs, including those of results in the appendices, can be found in Appendix C.5.

## C.1 Details on Steepest Descent

In this section, we provide a more comprehensive overview of steepest descent methods including various convergence results.

### C.1.1 The Steepest Descent Operator

Following earlier works on steepest descent methods [e.g. Kelner et al., 2014], it will be useful to re-write the steepest descent update (Eq. 6.9) as

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f_t^{\|\cdot\|} \quad \text{with} \quad \omega^{\|\cdot\|} \in \arg\max_{\theta' \in \mathbb{R}^d} \left( \langle \omega, \theta' \rangle - \frac{1}{2} \|\theta'\|^2 \right) . \tag{C.1}$$

The equivalence arises from substituting $\theta' = -\frac{1}{\alpha_t}(\theta - \theta_t)$ in Eq. (6.9). This allows to concisely express the steepest descent update direction.

### C.1.2 Additional Examples for Steepest Descent Methods

We give two additional examples for steepest descent methods to demonstrate the versatility of this framework.

*Coordinate Descent* Steepest descent with respect to the $L^1$-norm yields coordinate descent,

$$\nabla f^{\|\cdot\|_1} = |\nabla f_{i_{\max}}| e^{(i_{\max})} \tag{C.2}$$

where the selected coordinate is chosen as $i_{\max} \in \arg\max_{i\in[d]} |\nabla f_{t,i}|$ and $e^{(i)}$ denotes the $i$-th coordinate vector. The corresponding smoothness assumption is

$$\|\nabla f(\theta') - \nabla f(\theta)\|_\infty \leq L_1 \|\theta' - \theta\|_1. \tag{C.3}$$

In fact, this assumption implies the coordinate-wise Lipschitz smoothness assumption,

$$|\nabla f(\theta + he^{(i)})_i - \nabla f(\theta)_i| \leq L_1 |h| \quad \forall i \in [d], \tag{C.4}$$

which is widely-used in the literature on coordinate descent, since

$$|\nabla f(\theta + he^{(i)})_i - \nabla f(\theta)_i| \leq \|\nabla f(\theta + he^{(i)}) - \nabla f(\theta)\|_\infty \overset{(C.3)}{\leq} L_1 \|\theta + he^{(i)} - \theta\|_1 = L_1 |h|. \tag{C.5}$$

*Block-Normalized Gradient Descent* Assume a block structure on $\mathbb{R}^d$ given by a partitioning $\mathcal{B} = \{B_1, \ldots, B_b\}$ of $[d]$, with $B_k \subset [d]$, $B_k \cap B_l = \emptyset$ for $k \neq l$, and $\bigcup_k B_k = [d]$. For $B \subset [d]$, define $\theta_B \in \mathbb{R}^{|B|}$ to be the vector consisting of $(\theta_i)_{i\in B}$. We can now define norms with respect to this block structure, such as

$$\|\theta\|_\infty^{\mathcal{B}} = \max_{B\in\mathcal{B}} \|\theta_B\|_2 \quad \text{with dual norm} \quad \|\theta\|_1^{\mathcal{B}} = \sum_{B\in\mathcal{B}} \|\theta_B\|_2. \tag{C.6}$$

Steepest descent w.r.t. $\|\cdot\|_\infty^{\mathcal{B}}$ results in *block-normalized gradient descent*,

$$\nabla f^{\|\cdot\|_\infty^{\mathcal{B}}} = \|\nabla f\|_1^{\mathcal{B}} \mathrm{norm}_{\mathcal{B}}(\nabla f), \quad \mathrm{norm}_{\mathcal{B}}(\omega) = \left( \frac{\omega_{B_1}^\mathsf{T}}{\|\omega_{B_1}\|_2}, \ldots, \frac{\omega_{B_b}^\mathsf{T}}{\|\omega_{B_b}\|_2} \right)^\mathsf{T}. \tag{C.7}$$

This method is a block-wise equivalent of sign gradient descent, normalizing the update magnitude over blocks instead of elementwise. Variants of this method have recently been studied empirically for neural network training [Yu et al., 2017, Ginsburg et al., 2019] with the blocks corresponding to the weights and biases of individual layers.

We can analyze this method in a similar fashion as we did for signGD in the main paper. The matrix norm determining the corresponding smoothness constant is given by

$$\|\mathbf{H}\|_{\infty,1}^{\mathcal{B}} = \max_{\|\theta\|_\infty^{\mathcal{B}} \leq 1} \|\mathbf{H}\theta\|_1^{\mathcal{B}}. \tag{C.8}$$

We can provide an upper bound for this matrix norm analogous to Proposition 6.4.

**Proposition C.1.** *Let $\mathbf{H} \in \mathbb{R}^{d\times d}$ be nonzero, positive semi-definite and let a block structure be given by a partitioning $\mathcal{B}$. Then*

$$\|\mathbf{H}\|_{\infty,1}^{\mathcal{B}} \leq \rho_{diag}^{\mathcal{B}}(\mathbf{H})^{-1} \sum_{B\in\mathcal{B}} \lambda_{max}(\mathbf{H}_{BB}) \tag{C.9}$$

*with* $\rho^{\mathcal{B}}_{diag}(\mathbf{H}) := \frac{\sum_{BB'} \|\mathbf{H}_{BB'}\|_2}{\sum_B \|\mathbf{H}_{BB}\|_2}$.

The quantity $\rho^{\mathcal{B}}_{\text{diag}}(\mathbf{H})$ measures the degree of concentration of $\mathbf{H}$ on its block diagonal. Without going into details, this allows us to reason about block-normalized gradient descent in a similar way as we did for sign gradient descent in the main paper. In particular, we can identify conditions which favor block-normalized GD. Firstly, this will require $\rho^{\mathcal{B}}_{\text{diag}}$ to be sufficiently large, i.e., some degree of concentration of the Hessian on the diagonal blocks. Secondly, it requires a certain structure among the eigenvalues of the blocks. With a slight abuse of notation, denote $\lambda_B = \lambda_{\max}(\mathbf{H}_{BB})$ and note that $\lambda_{\max}(\mathbf{H}) \geq \max_B \lambda_B$. Then block-normalized GD will be favored relative to GD if

$$\max_B \lambda_B \gg \frac{1}{|\mathcal{B}|} \sum_{B \in \mathcal{B}} \lambda_B \tag{C.10}$$

given that $\rho^{\mathcal{B}}_{\text{diag}}(\mathbf{H})$ is not too small.

### *C.1.3 Convergence Results for Steepest Descent*

Without further assumptions, smoothness guarantees convergence to a first-order stationary point.

**Proposition C.2.** *If $f$ is $L$-smooth w.r.t. $\|\cdot\|$, then steepest descent (Eq. 6.9) satisfies*

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f_t\|_*^2 \leq \frac{2L(f_0 - f^\star)}{T}. \tag{C.11}$$

For smooth and convex functions, Kelner et al. [2014] showed $O(1/T)$ convergence in suboptimality. We restate this result here for completeness.

**Theorem C.1** (Theorem 1 in Kelner et al. [2014])**.** *If $f$ is $L$-smooth w.r.t. $\|\cdot\|$ and convex, then steepest descent (Eq. 6.9) satisfies*

$$f_T - f^\star \leq \frac{2LR^2}{T+4} \quad \textit{with} \quad R \overset{def}{=} \max_{\theta \text{ s.t. } f(\theta) \leq f(\theta_0)} \min_{\theta^\star \text{ s.t. } f(\theta^\star) = f^\star} \|\theta - \theta^\star\|. \tag{C.12}$$

The rate has a dependence on the initial distance to the nearest minimizer measured in the respective norm.

It is also straight-forward to show linear convergence in suboptimality under an additional assumption, known as the Polyak-Łojasiewicz (PL) condition. It is usually given for the Euclidean case as $\|\nabla f(\theta)\|_2^2 \geq 2\mu(f(\theta) - f^\star)$ but can likewise be formulated for arbitrary norms.

**Definition C.1.** *A function $f\colon \mathbb{R}^d \to \mathbb{R}$ satisfies the PL condition with constant $\mu$ w.r.t. a norm $\|\cdot\|$ if $\|\nabla f(\theta)\|_*^2 \geq 2\mu(f(\theta) - f^\star)$ for all $\theta \in \mathbb{R}^d$.*

We refer to this as PL with respect to $\|\cdot\|$ even though only the dual norm appears in the definition, since it is the natural counterpart to smoothness w.r.t. $\|\cdot\|$ and used to prove linear convergence for steepest descent w.r.t. $\|\cdot\|$. As with smoothness, we have equivalence of the PL condition for all norms, but constants may differ. Note that strong convexity implies the PL condition, but the class of PL functions also covers some non-convex functions.

**Proposition C.3.** *If $f$ is $L$-smooth and fulfills the PL condition with constant $\mu$ w.r.t $\|\cdot\|$, then steepest descent (Eq. 6.9) satisfies*

$$f_T - f^\star \leq \left(1 - \frac{\mu}{L}\right)^{\mathsf{T}} (f_0 - f^\star). \tag{C.13}$$

We are not aware of any published work showing this simple result in its general form.

## C.2 Two-Dimensional Quadratic Example

We give details on the two-dimensional quadratic example used in Section 6.4. We consider $f(\theta) = \frac{1}{2}\theta^{\mathsf{T}}\mathbf{H}\theta$ with positive definite Hessian $\nabla^2 f(\theta) \equiv \mathbf{H} = \left[\begin{smallmatrix} a & b \\ b & d \end{smallmatrix}\right]$. In this case, we can easily find closed-form expressions to $L_\infty(\mathbf{H})$ and $L_{\text{sep}}(\mathbf{H})$ in Proposition 6.6.

For $L_\infty$, we have $L_\infty(\mathbf{H}) = \|\mathbf{H}\|_{\infty,1} = \max_{\|\theta\|_\infty \leq 1} \|\mathbf{H}\theta\|_1 = \max_{\|\theta\|_\infty \leq 1} (|ax_1 + bx_2| + |bx_1 + dx_2|)$. Recall that $a, d > 0$ by positive definiteness of $\mathbf{H}$. By separating the two cases $b \geq 0$ and $b < 0$ we find

$$L_\infty(\mathbf{H}) = a + d + 2|b|. \tag{C.14}$$

Hence, $L_\infty$ and $L_{\text{sep}}$ coincide in this case. Furthermore, the upper-bound in Proposition 6.6 is also tight,

$$\rho_{\text{diag}}(\mathbf{H})^{-1} \sum_i \lambda_i = a + d + 2|b|, \tag{C.15}$$

since $\sum_i \lambda_i = \text{tr}(\mathbf{H}) = a + d$ and $\rho_{\text{diag}}(\mathbf{H}) = (a + d)/(a + d + 2|b|)$.

*Side note* This example also reveals another problem with the separable smoothness condition. In the literature using the separable smoothness condition it is assumed that the convergence of GD is determined by $l_{\max}$, which implicitly assumes $l_{\max} = \lambda_{\max}$. The example above shows that this may be misleading since The eigenvalues of $\mathbf{H}$ evaluate to

$$\lambda_{1/2} = \frac{a + d}{2} \pm \sqrt{\frac{(a - d)^2}{4} + b^2}. \tag{C.16}$$

W.l.o.g. assume $d \geq a$. Then we have

$$l_{\max} = d + |b| \quad \text{and} \quad \lambda_{\max} = \frac{a + d}{2} + \sqrt{\frac{(a - d)^2}{4} + b^2} \tag{C.17}$$

We see that $\lambda_{\max}$ can easily exceed $l_{\max}$. Of course, we could deviate from the definition of Eq. (6.25) and choose $l_i \equiv \lambda_{\max}$ to guarantee $l_{\max} = \lambda_{\max}$. However, this bound would very unfavorable for sign(S)GD, whose performance depends on $\sum_i l_i$.

## C.3 On Normalized Steepest Descent

There is a discrepancy between the version of sign gradient descent arising as steepest descent w.r.t. maximum norm, $\theta_{t+1} = \theta_t - \alpha \|\nabla f_t\|_1 \operatorname{sign}(\nabla f_t)$, and signSGD as used in neural network training, $\theta_{t+1} = \theta_t - \alpha_t \operatorname{sign}(g_t)$, with a constant or manually decreasing step size sequence $\alpha_t$. The norm-scaled version has actually been shown to be useful in the smooth, convex, non-stochastic setting; e.g., Kelner et al. [2014] successfully apply it to solve max-flow problems in graphs. Without the scaling by the gradient norm, we have a *normalized* method with an update magnitude determined solely by the step size, independent of the gradient magnitude. There are various possible reasons why this might be beneficial in settings like neural network training.

One possible explanation is that normalized methods address certain challenges of non-convex problems, e.g., by escaping from saddle points faster [Levy, 2016] or converging to global minima for quasi-convex functions [Hazan et al., 2015].

Another rationale is that, in the stochastic optimization setting, a decreasing step size is needed anyway to enforce convergence. Since $\|\nabla f_t\|_1$ is not available in the stochastic setting—and $\|g_t\|_1$ may be a poor estimate—it might be easier to subsume a similar scaling effect in the manually-tuned step size schedule.

We want to add to this discussion in two ways. First, we provide a basic convergence result for non-stochastic normalized steepest descent methods with a decreasing step size under the classical smoothness assumption, which we could not find in the literature. While normalized methods are clearly suboptimal under that assumption, this provides at least a basic convergence guarantee.

Second, we expand on the work of Zhang et al. [2019a], who show that normalized gradient descent is adapted to a certain relaxed smoothness condition which might be a better description of the regularity exhibited by neural network training objectives, which are not smooth in the sense of Eq. (6.6). By extending their reasoning to arbitrary norms, we provide a possible explanation for the success of normalized signGD (Eq. 6.1).

### C.3.1 Convergence of Normalized Steepest Descent under Classical Smoothness

We show convergence to a first-order stationary point for normalized steepest descent

$$\theta_{t+1} = \theta_t - \alpha_t \frac{\nabla f_t^{\|\cdot\|}}{\|\nabla f_t\|_*} \tag{C.18}$$

with a decreasing step size schedule. To see that this is indeed a *normalized* method, recall from Lemma C.2 that $\|\omega^{\|\cdot\|}\| = \|\omega\|_*$. Normalized steepest descent w.r.t. the maximum norm thus is sign gradient descent in the version of Eq. (6.1).

**Proposition C.4.** *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be L-smooth w.r.t. $\|\cdot\|$ and assume we perform normalized steepest descent updates*

$$\theta_{t+1} = \theta_t - \frac{\alpha_t}{L} \frac{\nabla f_t^{\|\cdot\|}}{\|\nabla f_t\|_*} \tag{C.19}$$

*with $\alpha_t = \frac{1}{\sqrt{t+1}}$. Then*

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f_t\|_* \leq \frac{L(f_0 - f_\star)}{\sqrt{T}} + \frac{\log(T+1)}{2\sqrt{T}} \overset{T\to\infty}{\longrightarrow} 0. \tag{C.20}$$

The proof may be found in Appendix C.5.

### C.3.2 Relaxed Smoothness

Zhang et al. [2019a] discuss a "soft" version of normalized gradient descent method,

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{\|\nabla f_t\|_2 + \beta} \nabla f_t. \tag{C.21}$$

and show that it is better geared towards the type of regularity exhibited by neural network training objectives, which are not smooth in the sense of Eq. (6.6). In particular, they show that this method is optimal under a certain "relaxed" smoothness assumption, which allows the curvature to grow with the gradient norm instead of bounding it globally as in classical smoothness.

Since they consider normalized gradient descent, their discussion is based on Euclidean geometry. We show here that their reasoning can be generalized to arbitrary norms, providing another possible explanation of the practical success of *normalized* steepest descent methods, e.g., sign gradient descent without the scaling by $\|\nabla f_t\|_1$.

*Results of Zhang et al. [2019a]* The relaxed smoothness condition proposed by Zhang et al. [2019a] reads

$$\|\nabla^2 f(\theta)\|_2 \leq L^{(0)} + L^{(1)} \|\nabla f(\theta)\|_2, \tag{C.22}$$

where $\|\cdot\|_2$ for matrices denotes the spectral norm. This allows the curvature to grow with the gradient norm, in contrast to classical smoothness, which demands a global bound on the Hessian.

This relaxed smoothness gives rise to normalized gradient descent since, as we will see later, it provides local quadratic bounds of the form

$$f_{t+1} \leq f_t + \langle \nabla f_t, \theta_{t+1} - \theta_t \rangle + \frac{1}{2}(A + B\|\nabla f_t\|_2)\|\theta_{t+1} - \theta_t\|_2^2, \quad A, B \geq 0. \tag{C.23}$$

This resembles the bound of Lemma 6.1, but the quadratic term now scales with the gradient norm. It is minimized by a normalized gradient descent update (Eq. C.21) with appropriately chosen $\alpha$ and $\beta$.

The main finding of Zhang et al. [2019a] is that gradient descent can become arbitrarily slow for the class of functions satisfying

this relaxed smoothness, whereas normalized gradient descent (Eq. C.21) retains an $O(1/\varepsilon^2)$ rate of convergence to an $\varepsilon$-stationary point.

*Generalization to Arbitrary Norms* In this section, we generalize the concept of relaxed smoothness to arbitrary norms, which will give rise to general normalized steepest descent methods. We define relaxed smoothness w.r.t. to some norm $\|\cdot\|$ analogously to the Euclidean case (Eq. C.22), but use the dual norm for the gradient and the induced matrix norm of Proposition 6.3 for the Hessian.

**Definition C.2.** *A function $f$ is called $(L^{(0)}, L^{(1)})$-smooth with respect to some norm $\|\cdot\|$ if*

$$\|\nabla^2 f(\theta)\| \leq L^{(0)} + L^{(1)}\|\nabla f(\theta)\|_*, \tag{C.24}$$

*where $\|\cdot\|$ for matrices is the norm defined in Eq.* (6.17)*.*

Under this smoothness assumption, we have the following local quadratic bound:

**Lemma C.1.** *Assume $f$ is $(L^{(0)}, L^{(1)})$-smooth with respect to a norm $\|\cdot\|$. Then for $\theta, \theta' \in \mathbb{R}^d$ with $\|\theta' - \theta\| \leq \frac{1}{L^{(1)}}$,*

$$f(\theta') \leq f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{1}{2}(5L^{(0)} + 4L^{(1)}\|\nabla f(\theta)\|_*)\|\theta' - \theta\|^2. \tag{C.25}$$

This resembles the bound in Lemma 6.1, but the quadratic term now scales with the gradient norm. In analogy to steepest descent, we can now construct an optimization method that minimizes this bound in each step. Using Lemma C.1 with $\theta = \theta_t$ and minimizing w.r.t. $\theta'$ yields

$$\theta_{t+1} = \theta_t - \frac{1}{(5L^{(0)} + 4L^{(1)}\|\nabla f_t\|_*)} \nabla f_t^{\|\cdot\|}. \tag{C.26}$$

This can be seen as a "soft" version of normalized steepest descent which reverts back to steepest descent in the vicinity of a stationary point, i.e., when $\|\nabla f_t\|_*$ becomes small.

We now show that the convergence theorem of Zhang et al. [2019a] carries over to this generalized setting. The proofs (see Appendix C.5) are straight-forward adaptations of that in Zhang et al. [2019a], with a little bit of extra care with regards to the norms.

**Theorem C.2.** *Assume $f$ is $(L^{(0)}, L^{(1)})$-smooth with respect to a norm $\|\cdot\|$. Then normalized steepest descent (Eq. C.26) converges to an $\varepsilon$-stationary point, $\|\nabla f\|_* \leq \varepsilon$, in at most*

$$T_\varepsilon = 18(f_0 - f^\star) \max\left(\frac{L^{(0)}}{\varepsilon^2}, \frac{(L^{(1)})^2}{L^{(0)}}\right) \tag{C.27}$$

*iterations.*

## C.4 Experimental Details

### C.4.1 Quadratic Experiments

*Generating Hessians.* We draw a random rotation matrix $\boldsymbol{R}$ from the Haar distribution[1] and set the Hessian to be $\mathbf{H} = \boldsymbol{R}^{\varphi}\boldsymbol{\Lambda}(\boldsymbol{R}^{\varphi})^*$, where $\boldsymbol{R}^{\varphi}$ for $\varphi \in [0,1]$ is a non-integer matrix power and $\boldsymbol{A}^*$ denotes the conjugate transpose matrix. We can think of this as rotating the eigenvectors of the Hessian by a fraction of $\varphi$ in the direction prescribed by $\boldsymbol{R}$. The non-integer matrix power $\boldsymbol{R}^{\varphi}$, is computed via the eigendecomposition $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^*$ as $\boldsymbol{R}^{\varphi} = \boldsymbol{U}\boldsymbol{D}^{\varphi}\boldsymbol{U}^*$ where $\boldsymbol{D}^{\varphi}$ for the diagonal matrix $\boldsymbol{D}$ is obtained by raising its elements to the power $\varphi$.

[1] The uniform distribution on the special orthogonal group $SO(d)$ of $d$-dimensional rotation matrices. We used the `special_ortho_group` function provided by the `scipy.stats` package [Jones et al., 2001].

*Computing $L_\infty$.* To compute the smoothness constant w.r.t. the maximum norm, we have to compute the matrix norm $\|\mathbf{H}\|_{\infty,1} = \max_{\|\theta\|_\infty \leq 1} \|\mathbf{H}\theta\|_1$. We use the fact that the solution is attained at $\theta \in \{-1,1\}^d$ [see Rohn, 2000] and brute-force search for the maximum $\|\mathbf{H}\theta\|_1$ in this set. Since there are $2^d$ vectors in $\{-1,1\}^d$, this is only possible for relatively small dimension.

*On the performance measure.* When comparing gradient descent and sign gradient descent on these quadratic problems, we use the distance to the optimum as a performance measure. The reason is that we are interested in a comparison over a range of different quadratics with varying $\lambda_{\max}$. The function value, which scales with $\lambda_{\max}$ would not be suitable for such a comparison. Since we are comparing optimization methods which are adapted to different norms, it might make a difference which norm we choose to compute the distance to the optimum. We opted for the Euclidean norm to benefit the baseline method (gradient descent) as the lesser of two evils.

## C.5 Proofs

This section contains proofs for all statements in the main text as well in the appendix. We proceed by order of appearance.

The proofs relating to steepest descent methods make use of the formulation introduced in Eq. (C.1) with the steepest descent operator $\omega^{\|\cdot\|}$. For later use, we establish the following Lemma connecting this steepest descent operator to the dual norm.

**Lemma C.2.** *For all $\theta, \omega \in \mathbb{R}^d$, we have*

$$\text{(a)} \quad \langle \theta, \omega \rangle \leq \|\theta\| \|\omega\|_* \tag{C.28}$$

$$\text{(b)} \quad \|\omega^{\|\cdot\|}\|^2 = \langle \omega, \omega^{\|\cdot\|} \rangle \tag{C.29}$$

$$\text{(c)} \quad \|\omega^{\|\cdot\|}\| = \|\omega\|_* \tag{C.30}$$

*Proof of Lemma C.2.* Statement (a) follows immediately from the definition of the dual norm.

Regarding (b), by definition of $\omega^{\|\cdot\|}$, we know that $\langle \omega, c\omega^{\|\cdot\|}\rangle - \frac{1}{2}\|c\omega^{\|\cdot\|}\|^2$ is maximized by $c = 1$. Hence the derivative w.r.t. $c$,

$$\frac{d}{dc}\left[\langle \omega, c\omega^{\|\cdot\|}\rangle - \frac{1}{2}\|c\omega^{\|\cdot\|}\|^2\right] = \langle \omega, \omega^{\|\cdot\|}\rangle - c\|\omega^{\|\cdot\|}\|^2, \tag{C.31}$$

must evaluate to 0 at $c = 1$, which proves (b).

For $(c)$, we use the equivalent definition

$$\|\omega\|_* = \max_{\theta\neq 0} \frac{\langle \theta, \omega\rangle}{\|\theta\|}. \tag{C.32}$$

Assume w.l.o.g. that $\omega^{\|\cdot\|} \neq 0$. Then

$$\|\omega\|_* = \max_{\theta\neq 0} \frac{\langle \theta, \omega\rangle}{\|\theta\|} \geq \frac{\langle \omega^{\|\cdot\|}, \omega\rangle}{\|\omega^{\|\cdot\|}\|} \overset{\text{(b)}}{=} \|\omega^{\|\cdot\|}\|. \tag{C.33}$$

Conversely, Let $\theta' \in \arg\max_{\|\theta\|=1}\langle \theta, \omega\rangle$, such that $\langle \omega, \theta'\rangle = \|\omega\|_*$. Then

$$\frac{1}{2}\|\omega\|_*^2 = \langle \omega, \|\omega\|_*\theta'\rangle - \frac{1}{2}\|\|\omega\|_*\theta'\|^2 \leq \langle \omega, \omega^{\|\cdot\|}\rangle - \frac{1}{2}\|\omega^{\|\cdot\|}\|^2 \overset{\text{(b)}}{=} \frac{1}{2}\|\omega^{\|\cdot\|}\|^2, \tag{C.34}$$

where the inequality is by definition of $\omega^{\|\cdot\|}$. □

### C.5.1 *Proofs for Section 6.3*

*Proof.* Proof of Lemma 6.1 Define $g(\tau) = f(\theta + \tau(\theta' - \theta))$ for $\tau \in [0, 1]$ with $g'(\tau) = \langle \nabla f(\theta + \tau(\theta' - \theta)), \theta' - \theta\rangle$. Then

$$\begin{aligned}
f(\theta') &= f(\theta) + \int_0^1 g'(\tau)\,\mathrm{d}\tau \\
&= f(\theta) + \int_0^1 \langle \nabla f(\theta + \tau(\theta' - \theta)), \theta' - \theta\rangle\,\mathrm{d}\tau \\
&= f(\theta) + \langle \nabla f(\theta), \theta' - \theta\rangle + \int_0^1 \langle \nabla f(\theta + \tau(\theta' - \theta)) - \nabla f(\theta), \theta' - \theta\rangle\,\mathrm{d}\tau \\
&\overset{\text{(C.28)}}{\leq} f(\theta) + \langle \nabla f(\theta), \theta' - \theta\rangle + \int_0^1 \|\nabla f(\theta + \tau(\theta' - \theta)) - \nabla f(\theta)\|_*\|\theta' - \theta\|\,\mathrm{d}\tau \\
&\overset{\text{(6.6)}}{\leq} f(\theta) + \langle \nabla f(\theta), \theta' - \theta\rangle + \int_0^1 L\|\tau(\theta' - \theta)\|\|\theta' - \theta\|\,\mathrm{d}\tau \\
&= f(\theta) + \langle \nabla f(\theta), \theta' - \theta\rangle + L\|\theta' - \theta\|^2\int_0^1 \tau\,\mathrm{d}\tau \\
&= f(\theta) + \langle \nabla f(\theta), \theta' - \theta\rangle + \frac{L}{2}\|\theta' - \theta\|^2.
\end{aligned} \tag{C.35}$$

The first inequality is due to Lemma C.2(a) and the second inequality uses the $L$-smoothness. □

*Proof of Lemma 6.2.* We apply Lemma 6.1 with $\theta' = \theta^+ = \theta - \frac{1}{L}\nabla f(\theta)^{\|\cdot\|}$

$$\begin{aligned}
f(\theta^+) &\leq f(\theta) + \langle \nabla f(\theta), \theta^+ - \theta\rangle + \frac{L}{2}\|\theta^+ - \theta\|^2 \\
&= f(\theta) + \langle \nabla f(\theta), -\frac{1}{L}\nabla f(\theta)^{\|\cdot\|}\rangle + \frac{L}{2}\left\|-\frac{1}{L}\nabla f(\theta)^{\|\cdot\|}\right\|^2 \\
&= f(\theta) - \frac{1}{L}\left(\langle \nabla f(\theta), \nabla f(\theta)^{\|\cdot\|}\rangle - \frac{1}{2}\|\nabla f(\theta)^{\|\cdot\|}\|^2\right).
\end{aligned} \tag{C.36}$$

By Lemma C.2, we have $\langle \nabla f(\theta), \nabla f(\theta)^{\|\cdot\|} \rangle = \|\nabla f(\theta)^{\|\cdot\|}\|^2 = \|\nabla f(\theta)\|_*^2$. Substituting this in yields the desired bound. □

*Proof of Proposition 6.1.* The dual norm of $\|\cdot\|_{\mathbf{L}}$ is $\|\cdot\|_{\mathbf{L}^{-1}}$, such that the assumption of 1-smoothness w.r.t. $\|\cdot\|_{\mathbf{L}}$ amounts to

$$\|\nabla f(\theta') - \nabla f(\theta)\|_{\mathbf{L}^{-1}} \leq \|\theta' - \theta\|_{\mathbf{L}} \quad \forall\, \theta, \theta' \in \mathbb{R}^d. \tag{C.37}$$

First, by definition of the maximum norm, we get

$$\|\omega\|_{\mathbf{L}} = \sqrt{\sum_i l_i \omega_i^2} \leq \sqrt{\sum_i l_i \|\omega\|_\infty^2} = \sqrt{\sum_i l_i}\|\omega\|_\infty. \tag{C.38}$$

Secondly, using Cauchy-Schwarz,

$$\|\omega\|_1 = \sum_i |\omega_i| = \sum_i \frac{|\omega_i|}{\sqrt{l_i}}\sqrt{l_i} \leq \sqrt{\sum_i \frac{\omega_i^2}{l_i}}\sqrt{\sum_i l_i} = \sqrt{\sum_i l_i}\|\omega\|_{\mathbf{L}^{-1}}. \tag{C.39}$$

Combining these two inequalities with the assumption yields the assertion:

$$\begin{aligned}\|\nabla f(\theta') - \nabla f(\theta)\|_1 &\overset{(C.39)}{\leq} \sqrt{\sum_i l_i}\|\nabla f(\theta') - \nabla f(\theta)\|_{\mathbf{L}^{-1}} \overset{(C.37)}{\leq} \sqrt{\sum_i l_i}\|\theta' - \theta\|_{\mathbf{L}} \\ &\overset{(C.38)}{\leq} \sqrt{\sum_i l_i}\sqrt{\sum_i l_i}\|\theta' - \theta\|_\infty = \left(\sum_i l_i\right)\|\theta' - \theta\|_\infty.\end{aligned} \tag{C.40}$$

□

*Proof of Proposition 6.2.* For separable smoothness, we use the definition (Eq. 6.4) with $\theta' = \theta + \alpha\delta$ and get

$$f(\theta + \alpha\delta) \leq f(\theta) + \langle \nabla f(\theta), \delta\rangle + \frac{\alpha^2}{2}\sum_i l_i \delta_i^2 = f(\theta) + \langle \nabla f(\theta), \delta\rangle + \frac{\alpha^2}{2}\sum_i l_i. \tag{C.41}$$

due to $\delta_i^2 = 1$ for all $i$.

For $\ell_\infty$-smoothness, we use Lemma 6.1 with $\theta' = \theta + \alpha\delta$ and get

$$f(\theta + \alpha\delta) \leq f(\theta) + \langle \nabla f(\theta), \delta\rangle + \frac{L_\infty}{2}\alpha^2\|\delta\|_\infty^2 = f(\theta) + \langle \nabla f(\theta), \delta\rangle + \frac{\alpha^2}{2}\sum_i l_i. \tag{C.42}$$

where the second step is due to $\|\delta\|_\infty = 1$ and the assumption that $L_\infty = \sum_i l_i$. □

### C.5.2 Proofs for Section 6.4

*Proof of Proposition 6.3.* Note that the matrix norm by construction satisfies

$$\|\mathbf{H}\theta\|_* \leq \|\mathbf{H}\|\|\theta\|. \tag{C.43}$$

We first show that Eq. (6.17) defines a matrix norm. Clearly $\|\mathbf{H}\| \geq 0$ and $\|\mathbf{H}\| = 0$ iff $\mathbf{H} = 0$. Furthermore, $\|\lambda\mathbf{H}\| = |\lambda|\|\mathbf{H}\|$. It remains to show subadditivity. Let $\mathbf{H}, \mathbf{H}' \in \mathbb{R}^{d\times d}$

$$\begin{aligned}\|\mathbf{H} + \mathbf{H}'\| &= \max_{\|\theta\|\leq 1}\|(\mathbf{H} + \mathbf{H}')\theta\|_* \leq \max_{\|\theta\|\leq 1}\left(\|\mathbf{H}\theta\|_* + \|\mathbf{H}'\theta\|_*\right) \\ &\leq \max_{\|\theta\|\leq 1}\|\mathbf{H}\theta\|_* + \max_{\|\theta'\|\leq 1}\|\mathbf{H}'\theta'\|_* = \|\mathbf{H}\| + \|\mathbf{H}'\|.\end{aligned} \tag{C.44}$$

Now assume $\|\nabla^2 f(\theta)\| \leq L$ for all $x \in \mathbb{R}^d$. Let $\theta, \theta' \in \mathbb{R}^d$ and define $g(\tau) = \nabla f(\theta + \tau(\theta' - \theta))$ for $\tau \in [0,1]$.

$$\begin{aligned}
\|\nabla f(\theta') - \nabla f(\theta)\|_* &= \left\| \int_0^1 \frac{d}{\mathrm{d}\tau} g(\tau)\, \mathrm{d}\tau \right\|_* \\
&= \left\| \int_0^1 \nabla^2 f(\theta + \tau(\theta' - \theta))(\theta' - \theta)\, \mathrm{d}\tau \right\|_* \\
&\leq \int_0^1 \left\| \nabla^2 f(\theta + \tau(\theta' - \theta))(\theta' - \theta) \right\|_* \mathrm{d}\tau \\
&\overset{(C.43)}{\leq} \int_0^1 \|\nabla^2 f(\theta + \tau(\theta' - \theta))\| \|\theta' - \theta\|\, \mathrm{d}\tau \\
&\leq L\|\theta' - \theta\| \int_0^1 1\, \mathrm{d}\tau = L\|\theta' - \theta\|.
\end{aligned} \tag{C.45}$$

Conversely, assume $L$-smoothness and fix $\theta \in \mathbb{R}^d$. For any $\|\delta\| \leq 1$ and $\varepsilon > 0$,

$$\left\| \left( \int_0^\varepsilon \nabla^2 f(\theta + \tau\delta)\, \mathrm{d}\tau \right) \delta \right\|_* = \|\nabla f(\theta + \varepsilon\delta) - \nabla f(\theta)\|_* \leq \varepsilon L \|\delta\| \leq \varepsilon L. \tag{C.46}$$

Dividing by $\varepsilon$ and letting $\varepsilon \to 0$, we get

$$\begin{aligned}
\|\nabla^2 f(\theta)\, \delta\|_* &= \left\| \lim_{\varepsilon \to 0} \left( \frac{1}{\varepsilon} \int_0^\varepsilon \nabla^2 f(\theta + \tau\delta)\, \mathrm{d}\tau \right) \delta \right\|_* \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left\| \left( \int_0^\varepsilon \nabla^2 f(\theta + \tau\delta)\, \mathrm{d}\tau \right) \delta \right\|_* \leq L.
\end{aligned} \tag{C.47}$$

This implies $\|\nabla^2 f(\theta)\| = \sup_{\|\delta\| \leq 1} \|\nabla^2 f(\theta)\delta\|_* \leq L$. □

*Proof of Proposition 6.4.* First note that

$$\|\mathbf{H}\|_{\infty,1} \overset{\text{def}}{=} \max_{\|\theta\|_\infty \leq 1} \|\mathbf{H}\theta\|_1 = \max_{\|\theta\|_\infty \leq 1} \sum_i \left| \sum_j H_{ij} x_j \right| \leq \sum_{i,j} |H_{ij}|. \tag{C.48}$$

Recall that $\sum_i |H_{ii}| = \sum_i H_{ii} = \sum_i \lambda_i$ for positive definite matrices. Then

$$\|\mathbf{H}\|_{\infty,1} \leq \sum_{i,j} |H_{ij}| = \frac{\sum_{i,j} |H_{ij}|}{\sum_i |H_{ii}|} \sum_i \lambda_i = \rho_{\text{diag}}(\mathbf{H})^{-1} \sum_i \lambda_i. \tag{C.49}$$

□

*Proof of Proposition 6.5.* Since $\mathbf{H}$ is a real symmetric matrix, it has a system of orthonormal eigenvectors and can be written as

$$\mathbf{H} = \sum_i \lambda_i v^{(i)} (v^{(i)})^\mathsf{T}. \tag{C.50}$$

With that we find

$$\begin{aligned}
\|\mathbf{H}\|_{\infty,1} &= \max_{\|\theta\|_\infty \leq 1} \|\mathbf{H}\theta\|_1 = \max_{\|\theta\|_\infty \leq 1} \left\| \sum_i \lambda_i (\theta^\mathsf{T} v^{(i)}) v^{(i)} \right\|_1 \\
&\leq \max_{\|\theta\|_\infty \leq 1} \sum_i |\lambda_i| \underbrace{|\theta^\mathsf{T} v^{(i)}|}_{\leq \|v^{(i)}\|_1} \|v^{(i)}\|_1 \leq \sum_i |\lambda_i| \, \|v^{(i)}\|_1^2.
\end{aligned} \tag{C.51}$$

□

*Proof of Eq.* (6.22). The fact that $\|\omega\|_\infty \le \|\omega\|_2 \le \|\omega\|_1$ implies

$$L_2 = \sup_{\theta\neq\theta'} \frac{\|\nabla f(\theta') - \nabla f(\theta)\|_2}{\|\theta' - \theta\|_2} \le \sup_{\theta\neq\theta'} \frac{\|\nabla f(\theta') - \nabla f(\theta)\|_1}{\|\theta' - \theta\|_\infty} = L_\infty. \tag{C.52}$$

Conversely, using $\frac{1}{\sqrt{d}}\|\omega\|_1 \le \|\omega\|_2 \le \sqrt{d}\|\omega\|_\infty$

$$L_\infty = \sup_{\theta\neq\theta'} \frac{\|\nabla f(\theta') - \nabla f(\theta)\|_1}{\|\theta' - \theta\|_\infty} \le \sup_{\theta\neq\theta'} \frac{\sqrt{d}\|\nabla f(\theta') - \nabla f(\theta)\|_2}{\frac{1}{\sqrt{d}}\|\theta' - \theta\|_2} = dL_2. \tag{C.53}$$

□

*Proof of Proposition 6.6.* First inequality: First, let $\hat{l}_1, \dots, \hat{l}_d \ge 0$ be the minimizer in the definition of $L_{\text{sep}}$. For any $\omega$ with $\|\omega\|_\infty \le 1$, we have

$$\omega^\mathsf{T}\mathbf{H}\omega \le \omega^\mathsf{T} \operatorname{diag}(\hat{l}_1, \dots, \hat{l}_d)\omega = \sum_i \hat{l}_i\omega_i^2 \le \sum_i \hat{l}_i. \tag{C.54}$$

Next, we rewrite the definition of $L_\infty$ as

$$L_\infty = \max_{\|\theta\|_\infty, \|\theta'\|_\infty \le 1} \theta^\mathsf{T}\mathbf{H}\theta'. \tag{C.55}$$

and let $(\hat{\theta}, \hat{\theta}')$ be the maximizer. Then due to $\mathbf{H}$ being psd, we have

$$0 \le (\hat{\theta} - \hat{\theta}')^\mathsf{T}\mathbf{H}(\hat{\theta} - \hat{\theta}') = \hat{\theta}^\mathsf{T}\mathbf{H}\hat{\theta} - 2\hat{\theta}^\mathsf{T}\mathbf{H}\hat{\theta}' + \hat{\theta}'^\mathsf{T}\mathbf{H}\hat{\theta}' \le 2\sum_i \hat{l}_i - 2\hat{\theta}^\mathsf{T}\mathbf{H}\hat{\theta}', \tag{C.56}$$

where the last inequality is due to Eq. (C.54) applied to $\hat{\theta}$ and $\hat{\theta}'$. This proves the assertion, since $\sum_i \hat{l}_i = L_{\text{sep}}$ and $\hat{\theta}^\mathsf{T}\mathbf{H}\hat{\theta}' = L_\infty$ by definition.

Second inequality: We set $l_i = \sum_j |H_{ij}|$ and denote $\mathbf{L} = \operatorname{diag}(l_1, \dots, l_d)$. Then

$$[\mathbf{L} - \mathbf{H}]_{ii} \ge \sum_{j\neq i} |H_{ij}| \tag{C.57}$$

$$\sum_{j\neq i} |[\mathbf{L} - \mathbf{H}]_{ij}| = \sum_{j\neq i} |H_{ij}| \tag{C.58}$$

making $\mathbf{L} - \mathbf{H}$ diagonally dominant with non-negative diagonal elements, hence positive semi-definite. Therefore, $\mathbf{L}$ is admissible in the definition of $L_{\text{sep}}$. Now,

$$\sum_i l_i = \sum_{i,j} |H_{ij}| = \left(\frac{\sum_i |H_{ii}|}{\sum_{i,j} |H_{ij}|}\right)^{-1} \sum_i \lambda_i, \tag{C.59}$$

where we used $\sum_i |H_{ii}| = \sum_i H_{ii} = \sum_i \lambda_i$. □

### C.5.3 *Proofs for Appendix C.1*

*Proof of Proposition C.1.* First note that

$$\begin{aligned}\|\mathbf{H}\|_{\infty,1}^{\mathcal{B}} &\overset{\text{def}}{=} \max_{\|\theta\|_\infty^{\mathcal{B}} \le 1} \|\mathbf{H}\theta\|_1^{\mathcal{B}} = \max_{\|\theta\|_\infty^{\mathcal{B}} \le 1} \sum_{B\in\mathcal{B}} \|[\mathbf{H}\theta]_B\|_2 = \max_{\|\theta\|_\infty^{\mathcal{B}} \le 1} \sum_{B\in\mathcal{B}} \left\| \sum_{B'\in\mathcal{B}} \mathbf{H}_{BB'}\theta_{B'} \right\|_2 \\ &\le \max_{\|\theta\|_\infty^{\mathcal{B}} \le 1} \sum_{B,B'\in\mathcal{B}} \|\mathbf{H}_{BB'}\theta_{B'}\|_2 \le \sum_{B,B'\in\mathcal{B}} \max_{\omega\in\mathbb{R}^{|B'|}, \|\omega\|_2\le 1} \|\mathbf{H}_{BB'}\omega\|_2 = \sum_{B,B'\in\mathcal{B}} \|\mathbf{H}_{BB'}\|_2\end{aligned} \tag{C.60}$$

Recall that the diagonal blocks $\mathbf{H}_{BB}$ are positive definite since $\mathbf{H}$ is positive definite and thus $\|\mathbf{H}_{BB}\|_2 = \lambda_{\max}(\mathbf{H}_{BB})$ and

$$\|\mathbf{H}\|_{\infty,1}^{\mathcal{B}} \leq \sum_{BB'} \|\mathbf{H}_{BB'}\|_2 = \frac{\sum_{BB'} \|\mathbf{H}_{BB'}\|_2}{\sum_B \|\mathbf{H}_{BB}\|_2} \sum_B \lambda_{\max}(\mathbf{H}_{BB}) = \rho_{\text{diag}}^{\mathcal{B}}(\mathbf{H})^{-1} \sum_B \lambda_{\max}(\mathbf{H}_{BB}). \tag{C.61}$$

□

*Proof of Proposition C.2.* Lemma 6.2 gives

$$\|\nabla f_t\|_*^2 \leq 2L(f_t - f_{t+1}) \tag{C.62}$$

Rearranging and summing for $t = 0, \dots, T-1$ yields

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f_t\|_*^2 \leq \frac{2L}{T} \sum_{t=0}^{T-1} (f_t - f_{t+1}) = \frac{2L(f_0 - f_T)}{T} \leq \frac{2L(f_0 - f^\star)}{T}. \tag{C.63}$$

□

*Proof of Proposition C.3.* Combining Lemma 6.2 and the PL condition gives

$$f_{t+1} \leq f_t - \frac{1}{2L} \|\nabla f_t\|_*^2 \leq f_t - \frac{\mu}{L}(f_t - f^\star). \tag{C.64}$$

Subtracting $f^\star$ from both sides and iterating backwards yields the statement. □

### C.5.4 Proofs for Appendix C.3

We first prove Proposition C.4.

*Proof of Proposition C.4.* By Lemma 6.1 we have

$$f_{t+1} \leq f_t - \frac{\alpha_t}{L\|\nabla f_t\|_*} \langle \nabla f_t, \nabla f_t^{\|\cdot\|} \rangle + \frac{\alpha_t^2}{2L\|\nabla f_t\|_*^2} \|\nabla f_t^{\|\cdot\|}\|^2 \tag{C.65}$$

By Lemma C.2, we have $\langle \nabla f_t, \nabla f_t^{\|\cdot\|} \rangle = \|\nabla f_t^{\|\cdot\|}\|^2 = \|\nabla f_t\|_*^2$, which yields

$$f_{t+1} \leq f_t - \frac{1}{L} \left( \alpha_t \|\nabla f_t\|_* - \frac{\alpha_t^2}{2} \right). \tag{C.66}$$

With a telescopic sum, we get

$$f_0 - f^\star \geq f_0 - f_T = \sum_{t=0}^{T-1} (f_t - f_{t+1}) \geq \frac{1}{L} \sum_{t=0}^{T-1} \alpha_t \|\nabla f_t\|_* - \frac{1}{2L} \sum_{t=0}^{T-1} \alpha_t^2 \tag{C.67}$$

Now with $\alpha_t = 1/\sqrt{t+1} \geq 1/\sqrt{T}$, we have

$$L(f_0 - f^\star) \geq \frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} \|\nabla f_t\|_* - \frac{1}{2} \underbrace{\sum_{t=0}^{T-1} \frac{1}{t+1}}_{\leq \log(T+1)} \tag{C.68}$$

and thus

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f_t\|_* \leq \frac{L(f_0 - f^\star)}{\sqrt{T}} + \frac{\log(T+1)}{2\sqrt{T}} \tag{C.69}$$

□

We now proceed to the proofs for Subsection C.3.2 about the relaxed smoothness assumption. All proofs are closely following the ones given for the Euclidean norm in Zhang et al. [2019a].

To prove Lemma C.1, we first need the following Lemma, which allows us to control the growth of the gradient norm in the vicinity of a point $\theta \in \mathbb{R}^d$.

**Lemma C.3.** *Assume Eq.* (C.24) *holds and let* $\theta, \theta'$ *with* $\|\theta' - \theta\| \leq \frac{1}{L^{(1)}}$. *Then*

$$\|\nabla f(\theta')\|_* \leq 4\left(\frac{L^{(0)}}{L^{(1)}} + \|\nabla f(\theta)\|_*\right). \tag{C.70}$$

*Proof.* Define $\theta(\tau) = \theta + \tau(\theta' - \theta)$ as well as $g(\tau) = \nabla f(\theta(\tau))$ with $g'(\tau) = \nabla^2 f(\theta(\tau))(\theta' - \theta)$. Then

$$\begin{aligned}
\|\nabla f(\theta(t))\|_* &= \left\|\nabla f(\theta) + \int_0^t g'(\tau)\,\mathrm{d}\tau\right\|_* \\
&\leq \|\nabla f(\theta)\|_* + \int_0^t \|g'(\tau)\|_*\,\mathrm{d}\tau \\
&= \|\nabla f(\theta)\|_* + \int_0^t \|\nabla^2 f(\theta(\tau))(\theta' - \theta)\|_*\,\mathrm{d}\tau \\
&\overset{(C.43)}{\leq} \|\nabla f(\theta)\|_* + \underbrace{\|(\theta' - \theta)\|}_{\leq 1/L^{(1)}} \int_0^t \underbrace{\|\nabla^2 f(\theta(\tau))\|}_{\leq L^{(0)} + L^{(1)}\|\nabla f(\theta(\tau))\|_* \text{ (C.24)}}\,\mathrm{d}\tau \\
&\leq \|\nabla f(\theta)\|_* + \frac{1}{L^{(1)}}\int_0^t L^{(0)} + L^{(1)}\|\nabla f(\theta(\tau))\|_*\,\mathrm{d}\tau \\
&= \|\nabla f(\theta)\|_* + t\frac{L^{(0)}}{L^{(1)}} + \int_0^t \|\nabla f(\theta(\tau))\|_*\,\mathrm{d}\tau
\end{aligned} \tag{C.71}$$

Applying the integral form of Groenwall's inequality[2] yields

$$\|\nabla f(\theta(t))\|_* \leq \|\nabla f(\theta)\|_* + t\frac{L^{(0)}}{L^{(1)}} + \int_0^t \left(\|\nabla f(\theta)\|_* + \tau\frac{L^{(0)}}{L^{(1)}}\right)\exp(t-\tau)\,\mathrm{d}\tau. \tag{C.73}$$

We now specialize to $t = 1$ and upper-bound the integrand

$$\begin{aligned}
\|\nabla f(\theta')\|_* &= \|\nabla f(\theta(1))\|_* \\
&\leq \|\nabla f(\theta)\|_* + \frac{L^{(0)}}{L^{(1)}} + \int_0^1 \left(\|\nabla f(\theta)\|_* + \underbrace{\tau}_{\leq 1}\frac{L^{(0)}}{L^{(1)}}\right)\underbrace{\exp(1-\tau)}_{\leq \exp(1) < 3}\,\mathrm{d}\tau \\
&\leq \|\nabla f(\theta)\|_* + \frac{L^{(0)}}{L^{(1)}} + 3\left(\|\nabla f(\theta)\|_* + \frac{L^{(0)}}{L^{(1)}}\right)\int_0^1 \mathrm{d}\tau \\
&= 4\left(\frac{L^{(0)}}{L^{(1)}} + \|\nabla f(\theta)\|_*\right).
\end{aligned} \tag{C.74}$$

□

We can now approach the proof of Lemma C.1.

*Proof of Lemma C.1.* According to Taylor's theorem we have

$$f(\theta') = f(\theta) + \langle \nabla f(\theta), \theta' - \theta\rangle + \frac{1}{2}\langle \theta' - \theta, \nabla^2 f(\xi)(\theta' - \theta)\rangle \tag{C.75}$$

[2] Groenwall's inequality says that if $u(t) \leq \alpha(t) + \int_{t_0}^t \beta(\tau)u(\tau)\,\mathrm{d}\tau$ for continuous $u$ and $\beta$, then

$$u(t) \leq \alpha(t) + \int_{t_0}^t \alpha(\tau)\beta(\tau)\exp\left(\int_\tau^t \beta(r)dr\right)\mathrm{d}\tau. \tag{C.72}$$

We apply it here with $u(t) = \|\nabla f(\theta(t))\|_*$ and $\alpha(t) = \|\nabla f(\theta)\|_* + tL^{(0)}/L^{(1)}$ and $\beta(\tau) \equiv 1$.

with some $\xi \in \{\theta + \tau(\theta' - \theta) \mid \tau \in [0,1]\}$. We can bound the quadratic term as

$$\begin{aligned}\langle \theta' - \theta, \nabla^2 f(\xi)(\theta' - \theta)\rangle &\overset{(C.28)}{\leq} \|\theta' - \theta\| \|\nabla^2 f(\xi)(\theta' - \theta)\|_* \\ &\overset{(C.43)}{\leq} \|\theta' - \theta\|^2 \|\nabla^2 f(\xi)\| \\ &\overset{(C.24)}{\leq} (L^{(0)} + L^{(1)}\|\nabla f(\xi)\|_*)\|\theta' - \theta\|^2.\end{aligned} \tag{C.76}$$

The first inequality is by definition of the dual norm (see Lemma C.2). The second inequality is by construction of the induced matrix norm. The final inequality uses the relaxed smoothness assumption (Eq. C.24).

Next, since $\|\theta' - \theta\| \leq 1/L^{(1)}$ by assumption of Lemma C.1, we know $\|\xi - \theta\| \leq \frac{1}{L^{(1)}}$. Lemma C.3 thus gives us

$$\|\nabla f(\xi)\|_* \leq 4\left(\frac{L^{(0)}}{L^{(1)}} + \|\nabla f(\theta)\|_*\right). \tag{C.77}$$

Plugging this back into Eq. (C.76) yields

$$\langle \theta' - \theta, \nabla^2 f(\xi)(\theta' - \theta)\rangle \leq (5L^{(0)} + 4L^{(1)}\|\nabla f(\theta)\|_*)\|\theta' - \theta\|^2. \tag{C.78}$$

Using that in Eq. (C.75) proves the assertion. □

Finally, we prove Theorem C.2

*Proof of Theorem C.2.* Using Lemma C.1 with $\theta = \theta_t$ and $\theta' = \theta_{t+1} = \theta_t - \eta_t \nabla f_t^{\|\cdot\|}$ yields

$$f_{t+1} \leq f_t - \eta_t \langle \nabla f_t, \nabla f_t^{\|\cdot\|}\rangle + \frac{\eta_t^2}{2}(5L^{(0)} + 4L^{(1)}\|\nabla f_t\|_*)\|\nabla f_t^{\|\cdot\|}\|^2. \tag{C.79}$$

Recall from Lemma C.2 that $\langle z, z^{\|\cdot\|}\rangle = \|z^{\|\cdot\|}\|^2 = \|z\|_*^2$ and, hence,

$$\begin{aligned}f_{t+1} &\leq f_t - \eta_t \langle \nabla f_t, \nabla f_t^{\|\cdot\|}\rangle + \frac{\eta_t^2}{2}(5L^{(0)} + 4L^{(1)}\|\nabla f_t\|_*)\|\nabla f_t^{\|\cdot\|}\|^2 \\ &= f_t - \left(\eta_t - \frac{\eta_t^2}{2}(5L^{(0)} + 4L^{(1)}\|\nabla f_t\|_*)\right)\|\nabla f_t\|_*^2 \\ &= f_t - \frac{\|\nabla f_t\|_*^2}{2(5L^{(0)} + 4L^{(1)}\|\nabla f_t\|_*)}.\end{aligned} \tag{C.80}$$

If $\varepsilon \leq \|\nabla f_t\|_* \leq L^{(0)}/L^{(1)}$, we get

$$f_{t+1} \leq f_t - \frac{\varepsilon^2}{18L^{(0)}} \tag{C.81}$$

If $\|\nabla f_t\|_* \geq L^{(0)}/L^{(1)}$, we get

$$\begin{aligned} f_{t+1} &\leq f_t - \frac{\|\nabla f_t\|_*^2}{2(5L^{(0)} + 4L^{(1)}\|\nabla f_t\|_*)} \\ &= f_t - \frac{\|\nabla f_t\|_*}{10L^{(0)}/\|\nabla f_t\|_* + 8L^{(1)}} \\ &\leq f_t - \frac{\|\nabla f_t\|_*}{18L^{(1)}} \\ &\leq f_t - \frac{L^{(0)}}{18(L^{(1)})^2} \end{aligned} \tag{C.82}$$

Hence,

$$f_{t+1} \leq f_t - \min\left\{\frac{L^{(0)}}{18(L^{(1)})^2}, \frac{\varepsilon^2}{18L^{(0)}}\right\}. \tag{C.83}$$

Now assume that we have $T$ iterations with $\|\nabla f_t\|_* \geq \varepsilon$. Then

$$f_0 - f^\star \geq f_0 - f_T = \sum_{t=0}^{T-1}(f_t - f_{t+1}) \geq T\min\left\{\frac{L^{(0)}}{18(L^{(1)})^2}, \frac{\varepsilon^2}{18L^{(0)}}\right\}. \tag{C.84}$$

Rearranging yields

$$T \leq 18\frac{f_0 - f^\star}{\min\left\{\frac{L^{(0)}}{(L^{(1)})^2}, \frac{\varepsilon^2}{L^{(0)}}\right\}} = 18(f_0 - f^\star)\max\left(\frac{L^{(0)}}{\varepsilon^2}, \frac{(L^{(1)})^2}{L^{(0)}}\right). \tag{C.85}$$

□

# D
# Appendix to Chapter 7

*§D.1: Details on Natural Gradient Descent* provides additional exposition on the natural gradient and the generalized Gauss-Newton; its relation to distances in probability distribution space (§D.1.1), the expression of the Fisher for common loss functions (§D.1.2) and the view of the generalized Gauss-Newton as a linearization of the model (§D.1.3).

*§D.2: Proofs* provides proof of the propositions and statements skipped in the main paper; the relation between the expected Hessian and expected outer product of gradients (Eq. 7.4), the equivalence between the generalized Gauss-Newton (Prop. 7.1), and the bound on the difference between the generalized Gauss-Newton and the Hessian (Prop. 7.2).

*§D.3: Experimental Details* gives the necessary details to reproduce our experiments.

*§D.4: Additional Experimental Results* shows the experiments on different datasets.

## *D.1 Details on Natural Gradient Descent*

We give an expanded version of the introduction to natural gradient descent provided in Section 7.3.1

### *D.1.1 Measuring Distance in Kullback-Leibler Divergence*

Gradient descent minimizes the objective function by updating in the "direction of steepest descent". But what, precisely, is meant by the direction of steepest descent? Consider the following definition,

$$\lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left( \arg\min_{\delta} f(\theta + \delta) \right) \quad \text{s.t.} \quad d(\theta, \theta + \delta) \leq \varepsilon, \tag{D.1}$$

where $d(\cdot, \cdot)$ is some distance function. We are looking for the update step $\delta$ which minimizes $f$ within an $\varepsilon$ distance around $\theta$, and let the radius $\varepsilon$ go to zero (to make $\delta$ finite, we have to divide by $\varepsilon$). This definition makes clear that the direction of steepest descent

is intrinsically tied to the geometry we impose on the parameter space by the definition of the distance function. If we choose the Euclidean distance $d(\theta, \theta') = \|\theta - \theta'\|_2$, Eq. (D.1) reduces to the (normalized) negative gradient.

Now, assume that $\theta$ parameterizes a statistical model $p_\theta(z)$. The parameter vector $\theta$ is not the main quantity of interest; the distance between $\theta$ and $\theta'$ would be better measured in terms of distance between the distributions $p_\theta$ and $p_{\theta'}$. A common function to measure the difference between probability distributions is the Kullback–Leibler (KL) divergence. If we choose $d(\theta, \theta') = D_{KL}\left(p_{\theta'} \parallel p_\theta\right)$, the steepest descent direction becomes the natural gradient, $F(\theta)^{-1}\nabla R(\theta)$, where

$$F(\theta) = \nabla^2_{\theta'} D_{KL}\left(p_\theta \parallel p_{\theta'}\right)|_{\theta'=\theta} \tag{D.2}$$

the Hessian of the KL divergence, is the *Fisher information matrix* of the statistical model and

$$\begin{aligned} F(\theta) &= \mathbf{E}_{p_\theta(z)}\left[\nabla \log p_\theta(z) \nabla \log p_\theta(z)^\mathsf{T}\right] \\ &= \mathbf{E}_{p_\theta(z)}\left[-\nabla^2 \log p_\theta(z)\right]. \end{aligned} \tag{D.3}$$

To see why, apply the chain rule on the log to split the equation in terms of the Hessian and the outer product of the gradients of $p_\theta$ w.r.t. $\theta$,

$$\begin{aligned} \mathbf{E}_{p_\theta(z)}\left[-\nabla^2_\theta \log p_\theta(z)\right] &= \mathbf{E}_{p_\theta(z)}\left[-\frac{1}{p_\theta(z)}\nabla^2_\theta p_\theta(z)\right] \\ &\quad + \mathbf{E}_{p_\theta(z)}\left[\frac{1}{p_\theta(z)^2}\nabla_\theta p_\theta(z) \nabla_\theta p_\theta(z)^\mathsf{T}\right]. \end{aligned} \tag{D.4}$$

The first term on the right-hand side is zero, since

$$\begin{aligned} \mathbf{E}_{p_\theta(z)}\left[-\frac{1}{p_\theta(z)}\nabla^2_\theta p_\theta(z)\right] &= -\int_z \frac{1}{p_\theta(z)}\nabla^2_\theta p_\theta(z) p_\theta(z)\,\mathrm{d}z \\ &= \int_z \nabla^2_\theta p_\theta(z)\,\mathrm{d}z \\ &= \nabla^2_\theta \underbrace{\left[\int p_\theta(z)\,\mathrm{d}z\right]}_{=1} = 0. \end{aligned} \tag{D.5}$$

The second term is the expected outer-product of the gradients, as $\partial_\theta \log f(\theta) = \frac{1}{f(\theta)}\partial_\theta f(\theta)$,

$$\begin{aligned} \frac{1}{p_\theta(z)^2}\nabla_\theta p_\theta(z) \nabla_\theta p_\theta(z)^\mathsf{T} &= \left(\frac{1}{p_\theta(z)}\nabla_\theta p_\theta(z)\right)\left(\frac{1}{p_\theta(z)}\nabla_\theta p_\theta(z)\right)^\mathsf{T} \\ &= \nabla_\theta \log p_\theta(z)\, \nabla_\theta \log p_\theta(z)^\mathsf{T}. \end{aligned} \tag{D.6}$$

The same technique also shows that if the empirical distribution over the data is equal to the model distribution $p_\theta(y|f(x,\theta)$, then the Fisher, empirical Fisher and Hessian are all equal.

### D.1.2 The Fisher for Common Loss Functions

For a probabilistic conditional model of the form $p(y|f(x,\theta))$ where $p$ is an exponential family distribution, the equivalence between the Fisher and the generalized Gauss-Newton leads to a straightforward way to compute the Fisher without expectations, as

$$\begin{aligned} F(\theta) &= \sum_n (\mathrm{J}_\theta f(x_n,\theta))^\mathsf{T} (\nabla^2 \log p(y_n|f(x_n,\theta))) (\mathrm{J}_\theta f(x_n,\theta)) \\ &= \sum_n J_n^\mathsf{T} H_n J_n, \end{aligned} \tag{D.7}$$

where $J_n = \mathrm{J}_\theta f(x_n,\theta)$ and $H_n = \nabla^2 \log p(y_n|f(x_n,\theta))$ often has an exploitable structure.

**The squared-loss** used in regression, $\frac{1}{2}\sum_n \|y_n - f(x_n,\theta)\|^2$, can be cast in a probabilistic setting with a Gaussian distribution with unit variance,

$$p(y_n|f(x_n,\theta)) = \mathcal{N}(y_n; f(x_n,\theta), 1) \propto \exp\left(-\frac{1}{2}\|y_n - f(x_n,\theta)\|^2\right). \tag{D.8}$$

The Hessian of the negative log-likelihood w.r.t. $f$ is then

$$\begin{aligned} \nabla_f^2 \left[-\log p(y_n|f)\right] &= \nabla_f^2 \left[-\log \exp\left(-\frac{1}{2}\|y_n - f\|^2\right)\right] \\ &= \nabla_f^2 \left[\frac{1}{2}\|y_n - f\|^2\right] = 1. \end{aligned} \tag{D.9}$$

And as the function $f$ is scalar-valued, the Fisher reduces to an outer-products of gradients,

$$F(\theta) = \sum_n \nabla_\theta f(x_n,\theta) \nabla_\theta f(x_n,\theta)^\mathsf{T}. \tag{D.10}$$

We stress that this is different to the outer product of gradients of the overall loss;

$$F(\theta) \neq \sum_n \nabla_\theta \log p(y_n|f(x_n,\theta)) \nabla_\theta \log p(y_n|f(x_n,\theta))^\mathsf{T}. \tag{D.11}$$

**The cross-entropy loss** used in $C$-class classification can be cast as an exponential family distribution by using the softmax function on the mapping $f(x_n,\theta)$,

$$p(y_n = c|f(x_n,\theta)) = [\mathrm{softmax}(f)]_c = \frac{e^{f_c}}{\sum_i e^{f_i}} = \pi_c, \tag{D.12}$$

The Hessian of the negative log-likelihood w.r.t. $f$ is independent of the class label $c$,

$$\begin{aligned} \nabla_f^2(-\log p(y = c|f)) &= \nabla_f^2 \left[-f_c + \log\left(\sum_i e^{f_i}\right)\right] \\ &= \nabla_f^2 \left[\log\left(\sum_i e^{f_i}\right)\right]. \end{aligned} \tag{D.13}$$

A close look at the partial derivatives shows that

$$\frac{\partial^2}{\partial f_i^2} \log\left(\sum_c e^{f_c}\right) = \frac{e^{f_i}}{(\sum_c e^{f_c})} - \frac{{e^{f_i}}^2}{(\sum_c e^{f_c})^2}, \tag{D.14}$$

$$\frac{\partial^2}{\partial f_i \partial f_j} \log\left(\sum_c e^{f_c}\right) = -\frac{e^{f_i} e^{f_j}}{(\sum_c e^{f_c})^2}, \tag{D.15}$$

and the Hessian w.r.t. $f$ can be written in terms of the vector of predicted probabilities $\pi$ as

$$\nabla_f^2(-\log p(y|f)) = \operatorname{diag}(\pi) - \pi\pi^\mathsf{T}. \tag{D.16}$$

Writing $\pi_n$ the vector of probabilities associated with the $n$-th sample, the Fisher becomes

$$F(\theta) = \sum_n [\mathrm{J}_\theta f(x_n, \theta)]^\mathsf{T} (\operatorname{diag}(\pi_n) - \pi_n \pi_n^\mathsf{T})[\mathrm{J}_\theta f(x_n, \theta)]. \tag{D.17}$$

### *D.1.3 The Generalized Gauss-Newton as a Linear Approximation of the Model*

In Section 7.3.3, we mentioned that the generalized Gauss-Newton with a split $R(\theta) = \sum_n a_n(b_n(\theta))$ can be interpreted as an approximation of $R$ where the second-order information of $a_n$ is conserved but the second-order information of $b_n$ is ignored. To make this connection explicit, see that if $b_n$ is a linear function, the Hessian and the GGN are equal as the Hessian of $b_n$ w.r.t. to $\theta$ is zero,

$$\begin{aligned} \nabla^2 R(\theta) = &\underbrace{\sum_n (\mathrm{J}_\theta b_n(\theta))^\mathsf{T} \ \nabla_b^2 a_n(b_n(\theta)) \ (\mathrm{J}_\theta b_n(\theta))}_{\text{GGN}} \\ &+ \sum_{n,m} [\nabla_b a_n(b_n(\theta))]_m \underbrace{\nabla_\theta^2 b_n^{(m)}(\theta)}_{=0}. \end{aligned} \tag{D.18}$$

This corresponds to the Hessian of a local approximation of $R$ where the inner function $b$ is linearized. We write the first-order Taylor approximation of $b_n$ *around* $\theta$ as a function of $\theta'$,

$$\bar{b}_n(\theta, \theta') := b_n(\theta) + \mathrm{J}_\theta b_n(\theta)(\theta' - \theta),$$

and approximate $R(\theta')$ by replacing $b_n(\theta')$ by its linear approximation $\bar{b}_n(\theta, \theta')$. The generalized Gauss-Newton is the Hessian of this approximation, evaluated at $\theta' = \theta$,

$$\begin{aligned} G(\theta) &= \nabla_{\theta'}^2 \sum_n a_n(\bar{b}_n(\theta, \theta'))|_{\theta'=\theta} \\ &= \sum_n (\mathrm{J}_\theta b_n(\theta))^\mathsf{T} \nabla_b^2 a_n(b_n(\theta)) \ (\mathrm{J}_\theta b_n(\theta)). \end{aligned} \tag{D.19}$$

## *D.2 Proofs*

### *D.2.1 Proof of Propositon 7.1*

In Section 7.3.4, Proposition 7.1, we stated that the Fisher and the generalized Gauss-Newton are equivalent for the problems considered in the introduction;

**Proposition 7.1** (Martens [2020], §9.2). *If $p(y|f)$ is an exponential family distribution with natural parameters $f \in \mathbb{F}$, then the Fisher information matrix coincides with the GGN of the objective in Eq.* (7.1) *using the split*

$$a_n(b) = -\log p(y_n|b), \qquad b_n(\theta) = f(x_n, \theta), \tag{7.13}$$

*and reads*

$$F(\theta) = G(\theta) = -\sum_n [\mathrm{J}_\theta f(x_n, \theta)]^\mathsf{T} \, \nabla_f^2 \log p(y_n|f(x_n, \theta)) \, [\mathrm{J}_\theta f(x_n, \theta)]. \tag{7.14}$$

*Proof.* Plugging the split into the definition of the GGN (Eq. 7.10) yields $G(\theta)$, so we only need to show that the Fisher coincides with this GGN. By the chain rule, we have

$$\nabla_\theta \log p(y|f(x_n, \theta)) = \mathrm{J}_\theta f(x_n, \theta)^\mathsf{T} \, \nabla_f \log p(y|f(x_n, \theta)), \tag{D.20}$$

and we can then apply the following steps.

$$F(\theta) = \sum_n \mathbf{E}_{y \sim p_\theta(y|x_n)} \left[ \mathrm{J}_\theta f(x_n, \theta)^\mathsf{T} \, \nabla_f \log p(y|f_n) \, \nabla_f \log p(y|f_n)^\mathsf{T} \mathrm{J}_\theta f(x_n, \theta) \right], \tag{D.21}$$

$$= \sum_n \mathrm{J}_\theta f(x_n, \theta)^\mathsf{T} \mathbf{E}_{y \sim p_\theta(y|x_n)} \left[ \nabla_f \log p(y|f_n) \, \nabla_f \log p(y|f_n)^\mathsf{T} \right] \mathrm{J}_\theta f(x_n, \theta), \tag{D.22}$$

$$= \sum_n \mathrm{J}_\theta f(x_n, \theta)^\mathsf{T} \mathbf{E}_{y \sim p_\theta(y|x_n)} \left[ -\nabla_f^2 \log p(y|f_n) \right] \mathrm{J}_\theta f(x_n, \theta), \tag{D.23}$$

Eq. (D.21) rewrites the Fisher using the chain rule, Eq. (D.22) take the Jacobians out of the expectation as they do not depend on $y$ and Eq. (D.23) is due to the equivalence between the expected outer product of gradients and expected Hessian shown in the last section.

If $p$ is an exponential family distribution with natural parameters (a linear combination of) $f$, its log density has the form $\log p(y|f) = f^\mathsf{T} T(y) - A(f) + \log h(y)$ where $T$ are the sufficient statistics, $A$ is the cumulant function, and $h$ is the base measure. Its Hessian w.r.t. $f$ is independent of $y$. We can thus drop the expectation and find

$$F(\theta) = \sum_n \mathrm{J}_\theta f(x_n, \theta)^\mathsf{T} \nabla_f^2 (-\log p(y_n|f_n)) \mathrm{J}_\theta f(x_n, \theta). \tag{D.24}$$

□

### *D.2.2 Proof of Proposition 7.2*

In §7.3.4, Prop. 7.2, we show that the difference between the Fisher (or the GNN) and the Hessian can be bounded by the residuals and the smoothness constant of the model $f$;

**Proposition 7.2.** *Let $R(\theta)$ be defined as in Eq. (7.1) with $\mathbb{F} = \mathbb{R}^M$. Denote by $f_n^{(m)}$ the m-th component of $f(x_n, \cdot)\colon \mathbb{R}^D \to \mathbb{R}^M$ and assume each $f_n^{(m)}$ is $\beta$-smooth. Let $G(\theta)$ be the GGN (Eq. 7.10). Then,*

$$\|\nabla^2 R(\theta) - G(\theta)\|_2^2 \leq r(\theta)\beta, \tag{7.15}$$

*where $\|\cdot\|_2$ denotes the spectral norm for matrices and*

$$r(\theta) \stackrel{def}{=} \sum_{n=1}^{N} \|\nabla_f \log p(y_n|f(x_n,\theta))\|_1. \tag{7.16}$$

*Proof.* Dropping $\theta$ from the notation for brevity, the Hessian can be expressed as

$$\nabla^2 R = G + \sum_{n=1}^{N}\sum_{m=1}^{M} r_n^{(m)}\, \nabla_\theta^2 f_n^{(m)}, \tag{D.25}$$

where

$$r_n^{(m)} = \left.\frac{\partial \log p(y_n|f)}{\partial f^{(m)}}\right|_{f=f_n(\theta)} \tag{D.26}$$

is the derivative of $-\log p(y|f)$ w.r.t. the $m$-th component of $f$, evaluated at $f = f_n(\theta)$.

If all $f_n^{(m)}$ are $\beta$-smooth, their Hessians are bounded by $-\beta I \preceq \nabla_\theta^2 f_n^{(m)} \preceq \beta I$ and

$$-\left|\sum_{n,m} r_n^{(m)}\right| \beta I \preceq \nabla^2 R - G \preceq \left|\sum_{n,m} r_n^{(m)}\right| \beta I. \tag{D.27}$$

Pulling the absolute value inside the double sum gives the upper bound

$$\begin{aligned}\left|\sum_{n,m} r_n^{(m)}\right| &\leq \sum_n \sum_m \left|\left.\frac{\partial \log p(y_n|f)}{\partial f^{(m)}}\right|_{f=f_n(\theta)}\right| \\ &= \sum_n \|\nabla_f \log p(y_n|f_n(\theta))\|_1\end{aligned} \tag{D.28}$$

and the statement about the spectral norm (the largest singular value of the matrix) follows. □

## D.3 Experimental Details

In contrast to the main text of the paper, which uses the sum formulation of the loss function,

$$R(\theta) = \sum_n \log p(y_n|f(x_n,\theta)),$$

the implementation—and thus the reported step sizes and damping parameters—apply to the average,

$$R(\theta) = \frac{1}{N}\sum_n \log p(y_n|f(x_n,\theta)).$$

The Fisher and empirical Fisher are accordingly rescaled by a $1/N$ factor.

### D.3.1 Vector field of the empirical Fisher preconditioning

The problem used for Fig. 7.1 is a linear regression on $N = 1000$ samples from

$$x_i \sim \text{Lognormal}\,(0, 3/4)\,, \quad \epsilon_i \sim \mathcal{N}(0,1), \quad y_i = 2 + 2x_i + \epsilon_i. \tag{D.29}$$

To be visible and of a similar scale, the gradient, natural gradient and empirical Fisher-preconditioned gradient were relatively rescaled by $1/3$, 1 and 3, respectively. The trajectories of each method is computed by running each update,

$$\text{GD:} \qquad \theta_{t+1} = \theta_t - \gamma \nabla R(\theta_t). \tag{D.30}$$

$$\text{NGD:} \qquad \theta_{t+1} = \theta_t - \gamma (F(\theta_t) + \lambda I)^{-1} \nabla R(\theta_t), \tag{D.31}$$

$$\text{EFGD:} \qquad \theta_{t+1} = \theta_t - \gamma (\tilde{F}(\theta_t) + \lambda I)^{-1} \nabla R(\theta_t), \tag{D.32}$$

using a step size of $\gamma = 10^{-4}$ and a damping parameter of $\lambda = 10^{-8}$ to ensure stability for $50'000$ iterations. The vector field is computed using the same damping parameter. The starting points are

$$\begin{bmatrix} 2 & 4.5 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad \begin{bmatrix} 4.5 & 3 \end{bmatrix}, \qquad \begin{bmatrix} -0.5 & 3 \end{bmatrix}.$$

### D.3.2 EF as a Quadratic Approximation at the Minimum for Misspecified Models

The problems are optimized using using the Scipy [Jones et al., 2001] implementation of BFGS[1]. The quadratic approximation of the loss function using the matrix $M$ (the Fisher or empirical Fisher) used is $R(\theta) \approx \frac{1}{2}(\theta - \theta^\star)M(\theta - \theta^\star)$, for $\|\theta - \theta^\star\|^2 = 1$. The datasets used for the logistic regression problem of Fig. 7.2 are described in Table D.1.

[1] https://docs.scipy.org/doc/scipy/reference/optimize.minimize-bfgs.html

| Model | $p(x\|y=0)$ | $p(x\|y=1)$ |
|---|---|---|
| Correct model: | $\mathcal{N}\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}\right)$ | $\mathcal{N}\left(\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}\right)$ |
| Misspecified (A): | $\mathcal{N}\left(\begin{bmatrix}1.5\\1.5\end{bmatrix}, \begin{bmatrix}3 & 0\\0 & 3\end{bmatrix}\right)$ | $\mathcal{N}\left(\begin{bmatrix}-1.5\\-1.5\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$ |
| Misspecified (B): | $\mathcal{N}\left(\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}1.5 & -0.9\\-0.9 & 1.5\end{bmatrix}\right)$ | $\mathcal{N}\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1.5 & 0.9\\0.9 & 1.5\end{bmatrix}\right)$ |

Table D.1: Logistic regression datasets used for Fig. 7.2. For all datasets, $p(y=0) = p(y=1) = 1/2$.

The dataset used for the linear regression problem in Fig. 7.2 is described in Table D.2. All experiments used $N = 1000$ samples.

| Model | $y$ | $\epsilon$ |
|---|---|---|
| Correct model: | $y = x + \epsilon$ | $\epsilon \sim \mathcal{N}(0,1)$ |
| Misspecified (A): | $y = x + \epsilon$ | $\epsilon \sim \mathcal{N}(0,2)$ |
| Misspecified (B): | $y = x + \frac{1}{2}x^2 + \epsilon$ | $\epsilon \sim \mathcal{N}(0,1)$ |

Table D.2: Linear regression datasets used for Fig. 7.2. For all datasets, $x \sim \mathcal{N}(0,1)$.

### D.3.3 Preconditioning with the Empirical Fisher

The optimization experiment uses the update rules described in §D.3.1 by Equations (D.30), (D.31), and (D.32), respectively.

Table D.3 describes the datasets used in the experiments. These datasets are available at:

Table D.3: Datasets

| Dataset | # Features | # Samples | Type | Figure |
|---|---|---|---|---|
| a1a | 1′605 | 123 | Classification | Fig. 7.3 |
| BreastCancer | 683 | 10 | Classification | Fig. 7.3 |
| Boston Housing | 506 | 13 | Regression | Fig. 7.3 |
| Yacht Hydrodynamics | 308 | 7 | Regression | Fig. D.1 |
| Powerplant | 9′568 | 4 | Regression | Fig. D.1 |
| Wine | 178 | 13 | Regression | Fig. D.1 |
| Energy | 768 | 8 | Regression | Fig. D.1 |

- `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#a1a`
- `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#breast-cancer`
- `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html`
- `https://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics`
- `https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant`
- `https://archive.ics.uci.edu/ml/datasets/Wine`
- `https://archive.ics.uci.edu/ml/datasets/Energy+efficiency`

The step size and damping hyperparameters are selected by a gridsearch, selecting for each optimizer the run with the minimal loss after 100 iterations. The grid used is described in Table D.4 as a log-space.

Table D.4: Grid used for the hyperparameter search for the optimization experiments, in $\log_{10}$. The number of samples to generate was selected as to generate a smooth grid in base 10, e.g., $10^0, 10^{.25}, 10^{.5}, 10^{.75}, 10^1, 10^{1.25}, \ldots$

| Parameter | | Grid |
|---|---|---|
| Step size | $\gamma$ | `logspace(start=-20, stop=10, num=241)` |
| Damping | $\lambda$ | `logspace(start=-10, stop=10, num=41)` |

Table D.5 shows the hyperparameters selected by the gridsearch.

The cosine similarity is computed between the gradient preconditioned with the empirical Fisher and the Fisher, without damping, at each step along the path taken by the empirical Fisher optimizer.

The problems are initialized at $\theta_0 = 0$ and run for 100 iterations. This initialization is favorable to the empirical Fisher for the logistic regression problems. Not only is it guaranteed to not be arbitrarily wrong, but the empirical Fisher and the Fisher coincide when the predicted probabilities are uniform. For the sigmoid activation of the output of the linear mapping, $\sigma(f)$, the gradient and Hessian are

$$-\frac{\partial}{\partial f}\log p(y|f) = \sigma(f) \quad -\frac{\partial^2}{\partial f^2}\log p(y|f) = \sigma(f)(1-\sigma(f)). \tag{D.33}$$

They coincide when $\sigma(f) = \frac{1}{2}$, at $\theta = 0$, or when $\sigma(f) \in \{0, 1\}$, which require infinite weights.

| Dataset | Algorithm | $\gamma$ | $\lambda$ |
|---|---|---|---|
| Boston | GD | −5.250 | |
| | NGD | 0.125 | −10.0 |
| | EFGD | −1.250 | −8.0 |
| BreastCancer | GD | −5.125 | |
| | NGD | 0.125 | −10.0 |
| | EFGD | −1.250 | −10.0 |
| a1a | GD | 0.250 | |
| | NGD | 0.250 | −10.0 |
| | EFGD | −0.375 | −8.0 |
| Wine | GD | −5.625 | |
| | NGD | 0.000 | −8.5 |
| | EFGD | −1.375 | −6.0 |
| Energy | GD | −5.500 | |
| | NGD | 0.000 | −7.5 |
| | EFGD | 0.875 | −3.0 |
| Powerplant | GD | −5.750 | |
| | NGD | −0.625 | −8.0 |
| | EFGD | 3.375 | −1.0 |
| Yacht | GD | −1.500 | |
| | NGD | −0.750 | −7.5 |
| | EFGD | 1.625 | −6.5 |

Table D.5: Selected hyperparameters, given in $\log_{10}$.

## *D.4 Additional Experimental Results*

Fig. D.1 repeats the experiment described in Fig. 7.3 (§7.4.3) on additional linear regression problems. Those additional examples show that the poor performance of empirical Fisher-preconditioned updates compared to NGD is not isolated to the examples shown in the main text.

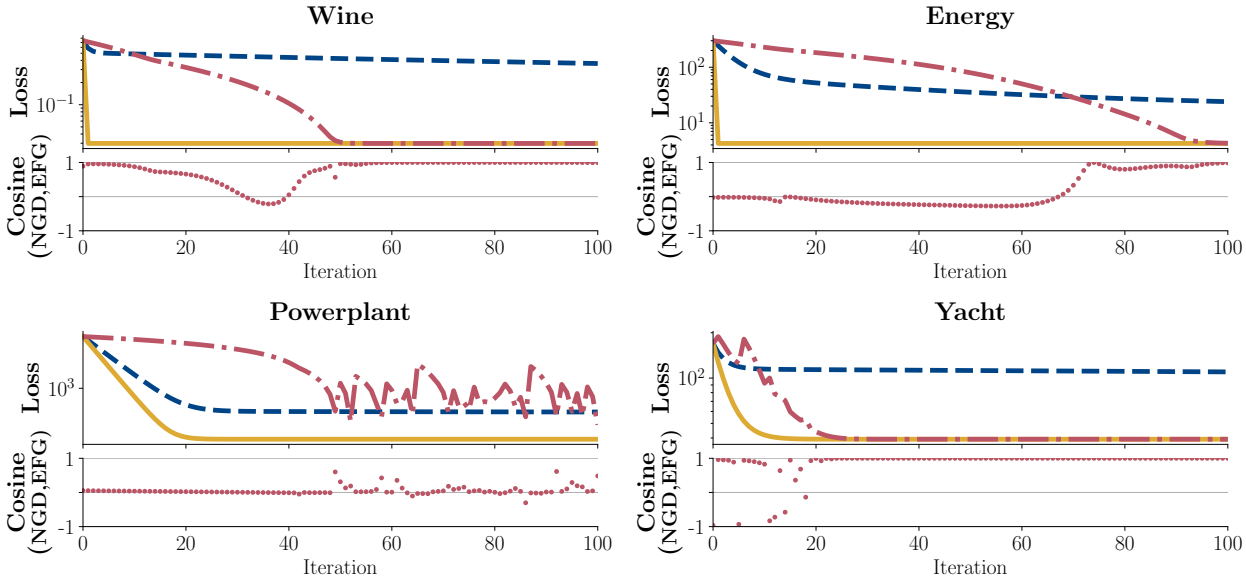


Figure D.1: Comparison of the Fisher (NGD) and the empirical Fisher (EFGD) as preconditioners on additional linear regression problems. The second row shows the cosine similarity between the EF-preconditioned gradient and the natural gradient at each step on the path taken by EFGD.