Journal of Machine Learning Research 21 (2020) 1-76 Submitted 11/17; Revised 6/20; Published 8/20

New Insights and Perspectives on the Natural Gradient Method

James Martens JAMES.MARTENS@GMAIL.COM
DeepMind
London, United Kingdom

Editor: Léon Bottou

Abstract

Natural gradient descent is an optimization method traditionally motivated from the perspective of information geometry, and works well for many applications as an alternative to stochastic gradient descent. In this paper we critically analyze this method and its properties, and show how it can be viewed as a type of 2nd-order optimization method, with the Fisher information matrix acting as a substitute for the Hessian. In many important cases, the Fisher information matrix is shown to be equivalent to the Generalized Gauss-Newton matrix, which both approximates the Hessian, but also has certain properties that favor its use over the Hessian. This perspective turns out to have significant implications for the design of a practical and robust natural gradient optimizer, as it motivates the use of techniques like trust regions and Tikhonov regularization. Additionally, we make a series of contributions to the understanding of natural gradient and 2nd-order methods, including: a thorough analysis of the convergence speed of stochastic natural gradient descent (and more general stochastic 2nd-order methods) as applied to convex quadratics, a critical examination of the oft-used "empirical" approximation of the Fisher matrix, and an analysis of the (approximate) parameterization invariance property possessed by natural gradient methods (which we show also holds for certain other curvature matrices, but notably not the Hessian).

Keywords: natural gradient methods, 2nd-order optimization, neural networks, convergence rate, parameterization invariance

Contents

1. Introduction and Overview

The natural gradient descent approach, pioneered by Amari and collaborators (e.g. Amari, 1998), is a popular alternative to traditional gradient descent methods which has received a lot of attention over the past several decades, motivating many new and related approaches. It has been successfully applied to a variety of problems such as blind source separation (Amari and Cichocki, 1998), reinforcement learning (Peters and Schaal, 2008), and neural network training (e.g. Park et al., 2000; Martens and Grosse, 2015; Desjardins et al., 2015).

Natural gradient descent is generally applicable to the optimization of probabilistic models[1], and involves the use of the so-called "natural gradient", which is defined as the gradient times the inverse of the model's Fisher information matrix (aka "the Fisher" ; see **Section 5**), in place of the standard gradient. In many applications, natural gradient descent seems to require far fewer iterations than gradient descent, making it a potentially attractive alternative method. Unfortunately, for models with very many parameters such as large neural networks, computing the natural gradient is impractical due to the extreme size of the Fisher matrix. This problem can be addressed through the use of various approximations to the Fisher (e.g Le Roux et al., 2008; Ollivier, 2015; Grosse and Salakhudinov, 2015; Martens and Grosse, 2015) that are designed to be easier to compute, to store and finally to invert, than the exact Fisher.

Natural gradient descent is classically motivated as a way of implementing steepest descent[2] in the space of realizable distributions[3] instead of the space of parameters, where distance in the distribution space is measured with a special "Riemannian metric" (Amari and Nagaoka, 2007). This metric depends only on the properties of the distributions themselves and not their parameters, and in particular is defined so that it approximates the square root of the Kullback–Leibler divergence within small neighborhoods. Under this interpretation (discussed in detail in **Section 6**), natural gradient descent is invariant to any smooth and invertible reparameterization of the model, putting it in stark contrast to gradient descent, whose performance is parameterization dependent.

In practice however, natural gradient descent still operates within the default parameter space, and works by computing directions in the space of distributions and then translating them back to the default space before taking a step. Because of this, the above discussed interpretation breaks down unless the step-size becomes arbitrarily small, and as discussed in **Section 10**, this breakdown has important implications for designing a natural gradient method that can work well in practice. Another problem with this interpretation is that it doesn't provide any obvious reason why a step of natural gradient descent should make more progress reducing the objective than a step of standard gradient descent (assuming well-chosen step-sizes for both). Moreover, given a large step-size one also loses the param-

1. This includes neural networks, which can be cast as conditional models.
2. "Steepest descent" is a common synonym for gradient descent, which emphasizes the interpretation of the gradient as being the direction that descends down the loss surface most "steeply". Here "steepness" is measured as the amount loss reduction per unit of distance traveled, where distance is measured according to some given metric. For standard gradient descent this metric is the Euclidean distance on the default parameter space.
3. "Realizable distributions" means distributions which correspond to some setting of the model's parameters.

eterization invariance property of the natural gradient method, although it will still hold *approximately* under certain conditions which are described in **Section 12**.

In **Section 10** we argue for an alternative view of natural gradient descent: as a type of 2nd-order method[4] which utilizes the Fisher as an alternative to the Hessian. As discussed in **Section 7**, 2nd-order methods work by forming a local quadratic approximation to the objective around the current iterate, and produce the next iterate by optimizing this approximation within some region where the approximation is believed to be accurate. According to this view, natural gradient descent ought to make more progress per step than gradient descent because it uses a local quadratic model/approximation of the objective function which is more detailed (which allows it to be less conservative) than the one implicitly used by gradient descent.

In support of this view is the fact that the Fisher can be cast as an approximation of the Hessian in at least two different ways (provided the objective has the form discussed in **Section 4**). First, as discussed in **Section 5**, it corresponds to the expected Hessian of the loss under the model's distribution over predicted outputs (instead of the usual empirical one used to compute the exact Hessian). Second, as we establish in **Section 9**, it is very often equivalent to the so-called "Generalized Gauss-Newton matrix" (GGN) (discussed in **Section 8**), a generalization of the classical Gauss-Newton matrix (e.g. Dennis Jr and Schnabel, 1996; Ortega and Rheinboldt, 2000; Nocedal and Wright, 2006), which is a popular alternative/approximation to the Hessian that has been used in various practical 2nd-order optimization methods designed specifically for neural networks (e.g Schraudolph, 2002; Martens, 2010; Vinyals and Povey, 2012), and which may actually be a *better* choice than the Hessian in the context of neural net training (see **Section 8.1**).

Viewing natural gradient descent as a 2nd-order method is also *prescriptive*, since it suggests the use of various damping/regularization techniques often used in the optimization literature to account for the limited accuracy of local quadratic approximations (especially over long distances). Indeed, such techniques have been successfully applied in 2nd-order methods designed for neural networks (e.g. Martens, 2010; Martens and Grosse, 2015), where they proved crucial in achieving fast and robust performance in practice. And before that have had a long history of application in the context of practical non-linear regression procedures (Tikhonov, 1943; Levenberg, 1944; Marquardt, 1963; Moré, 1978).

The "empirical Fisher", which is discussed in **Section 11**, is an approximation to the Fisher whose computation is easier to implement in practice using standard automatic-differentiation libraries. The empirical Fisher differs from the usual Fisher in subtle but important ways, which as we show in **Section 11.1**, make it considerably less useful as an approximation to the Fisher, or as a curvature matrix to be used in 2nd-order methods. Using the empirical Fisher also breaks some of the theory justifying natural gradient descent, although it nonetheless preserves its (approximate) parameterization invariance (as we show in **Section 12**). Despite these objections, the empirical Fisher has been used in many approaches, such as TONGA (Le Roux et al., 2008), and the recent spate of methods that

4. By "2nd-order method" we mean any iterative optimization method which generates updates as the (possibly approximate) solution of a non-trivial local quadratic model of the objective function. This extends well beyond the classical Newton's method, and includes approaches like (L-)BFGS, and methods based on the Gauss-Newton matrix.

use the diagonal of this matrix such as RMSprop (Tieleman and Hinton, 2012) and Adam (Ba and Kingma, 2015) (which we examine in **Section 11.2**).

A well-known and oft quoted result about stochastic natural gradient descent is that it is asymptotically "Fisher efficient" (Amari, 1998). Roughly speaking, this means that it provides an asymptotically unbiased estimate of the parameters with the lowest possible variance among all unbiased estimators (that see the same amount of data), thus achieving the best possible expected objective function value. Unfortunately, as discussed in **Section 14.1**, this result comes with several important caveats which significantly limit its applicability. Moreover, even when it is applicable, it only provides an *asymptotically* accurate characterization of the method, which may not usefully describe its behavior given a finite number of iterations.

To address these issues we build on the work of Murata (1998) in **Section 14.2** and **Section 14.3** to develop a more powerful convergence theory for stochastic 2nd-order methods (including natural gradient descent) as applied to convex *quadratic objectives*. Our results provide a more precise expression for the convergence speed of such methods than existing results do[5], and properly account for the effect of the starting point. And as we discuss in **Section 14.2.1** and **Section 14.3.1** they imply various interesting consequences about the relative performance of various 1st and 2nd-order stochastic optimization methods. Perhaps the most interesting conclusion of this analysis is that, while stochastic gradient descent with Polyak-style parameter averaging achieves the same *asymptotic* convergence speed as stochastic natural gradient descent (and is thus also "Fisher efficient", as was first shown by Polyak and Juditsky (1992)), stochastic 2nd-order methods can possess a much more favorable dependence on the starting point, which means that they can make much more progress given a limited iteration budget. Another interesting observation made in our analysis is that stochastic 2nd-order methods that use a decaying learning rate (of the form $\alpha_k = 1/(k+a+1)$) can, for certain problems, achieve an asymptotic objective function value that is better than that achieved in the same number of iterations by stochastic gradient descent (with a similar decaying learning rate), by a large constant factor.

Unfortunately, these convergence theory results fail to explain why 2nd-order optimization with the GGN/Fisher works so much better than classical 2nd-order schemes based on the Hessian *for neural network training* (Schraudolph, 2002; Martens, 2010; Vinyals and Povey, 2012). In **Section 15** we propose several important open questions in this direction that we leave for future work.

5. Alternate versions of several of our results have appeared in the literature before (e.g. Polyak and Juditsky, 1992; Bordes et al., 2009; Moulines and Bach, 2011). These formulations tend to be more general than ours (we restrict to the quadratic case), but also less precise and interpretable. In particular, previous results tend to omit asymptotically negligible terms that are nonetheless important pre-asymptotically. Or when they do include these terms, their bounds are simultaneously looser and more complicated than our own, perhaps owing to their increased generality. We discuss connections to some prior work on convergence bounds in Sections 14.2.2 and 14.3.2.

Table of Notation

Notation	Description
$[v]_i$	i-th entry of a vector v
$[A]_{i,j}$	(i,j)-th entry a matrix A
$\nabla\gamma$	gradient of a scalar function γ
J_γ	Jacobian of a vector-valued function γ
H_γ	Hessian of a scalar function γ (typically taken with respect to θ unless otherwise specified)
H	Hessian of the objective function h w.r.t. θ (i.e. H_h)
θ	vector of all the network's parameters
W_i	weight matrix at layer i
ϕ_i	activation function for layer i
s_i	unit inputs at layer i
a_i	unit activities at layer i
ℓ	number of layers
m	dimension of the network's output $f(x,\theta)$
m_i	number of units in i-th layer of the network
$f(x,\theta)$	function mapping the neural network's inputs to its output
$L(y,z)$	loss function
h	objective function
S	training set
k	current iteration
n	dimension of θ
$M_k(\delta)$	local quadratic approximation of h at θ_k
λ	strength constant for penalty-based damping
$\lambda_j(A)$	j-th largest eigenvalue of a symmetric matrix A
G	generalized Gauss-Newton matrix (GGN)
$P_{x,y}(\theta)$	model's distribution
$Q_{x,y}$	data distribution
$\hat{Q}_{x,y}$	training/empirical distribution
$R_{y\|z}$	predictive distribution used at network's output (so $P_{y\|x}(\theta) = R_{y\|f(x,\theta)}$)
p, q, r	density functions associated with above P, Q, and R (resp.)
F	Fisher information matrix (typically associated with $P_{x,y}$)
F_D	Fisher information matrix associated with parameterized distribution D
$\tilde{\nabla}h$	The natural gradient for objective h.

Table 1: A table listing some of the notation used throughout this document.

2. Neural Networks

Feed-forward neural networks are structured similarly to classical circuits. They typically consist of a sequence of ℓ "layers" of units, where each unit in a given layer receive inputs from the units in the previous layer, and computes an affine function of these followed by a scalar non-linear function called an "activation function". The input vector to the network, denoted by x, is given by the units of the first layer, which is called the "input layer" (and is not counted towards the total ℓ). The output vector of the network is given by the units of the network's last layer (called the "output layer"). The other layers are referred to as the network's "hidden layers".

Formally, given input $x \in \mathbb{R}^{m_0}$, and parameters $\theta \in \mathbb{R}^n$ which determine weight matrices $W_1 \in \mathbb{R}^{m_1 \times m_0}, W_2 \in \mathbb{R}^{m_2 \times m_1}, \ldots, W_\ell \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ and biases $b_1 \in \mathbb{R}^{m_1}, b_2 \in \mathbb{R}^{m_2}, \ldots, b_\ell \in \mathbb{R}^{m_\ell}$, the network computes its output $f(x, \theta) = a_\ell$ according to

$$\begin{aligned} s_i &= W_i a_{i-1} + b_i \\ a_i &= \phi_i(s_i)\,, \end{aligned}$$

where $a_0 = x$. Here, a_i is the vector of values ("activities") of the network's i-th layer, and $\phi_i(\cdot)$ is the vector-valued non-linear "activation function" computed at layer i, and is often given by some simple scalar function applied coordinate-wise.

Note that most of the results discussed in this document will apply to the more general setting where $f(x, \theta)$ is an arbitrary differentiable function (in both x and θ).

3. Supervised Learning Framework

The goal of optimization in the context of supervised learning is to find some setting of θ so that, for each input x in the training set, the output of the network (which we will sometimes call its "prediction") matches the given target outputs as closely as possible, as measured by some loss. In particular, given a training set S consisting of pairs (x, y), we wish to minimize the objective function

$$h(\theta) \equiv \frac{1}{|S|} \sum_{(x,y) \in S} L(y, f(x, \theta)), \tag{1}$$

where $L(y, z)$ is a "loss function" which measures the disagreement between y and z.

The prediction $f(x, \theta)$ may be a guess for y, in which case L might measure the inaccuracy of this guess (e.g. using the familiar squared error $\frac{1}{2}\|y - z\|^2$). Or $f(x, \theta)$ could encode the parameters of some simple predictive distribution. For example, $f(x, \theta)$ could be the set of probabilities which parameterize a multinomial distribution over the possible discrete values of y, with $L(y, f(x, \theta))$ being the negative log probability of y under this distribution.

4. KL Divergence Objectives

The natural gradient method of Amari (1998) can potentially be applied to any objective function which measures the performance of some statistical model. However, it enjoys richer theoretical properties when applied to objective functions based on the KL divergence between the model's distribution and the target distribution, or certain approximations/surrogates of these. In this section we will establish the basic notation and properties

of these objective functions, and discuss the various ways in which they can be formulated. Each of these formulations will be analogous to a particular formulation of the Fisher information matrix and natural gradient (as defined in Section 5), which will differ in subtle but important ways.

In the idealized setting, input vectors x are drawn independently from a *target* distribution Q_x with density function $q(x)$, and the corresponding (target) outputs y from a *conditional target* distribution $Q_{y|x}$ with density function $q(y|x)$.

We define the goal of learning as the minimization of the KL divergence from the target joint distribution $Q_{x,y}$, whose density is $q(y,x) = q(y|x)q(x)$, to the learned distribution $P_{x,y}(\theta)$, whose density is $p(x,y|\theta) = p(y|x,\theta)q(x)$. (Note that the second $q(x)$ is not a typo here, since we are not learning the distribution over x, only the conditional distribution of y given x.) Our objective function is thus

$$\mathrm{KL}(Q_{x,y} \| P_{x,y}(\theta)) = \int q(x,y) \log \frac{q(x,y)}{p(x,y|\theta)} dxdy \,.$$

This is equivalent to the expected KL divergence $\mathrm{E}_{Q_x}[\mathrm{KL}(Q_{y|x} \| P_{y|x}(\theta))]$ as can be seen by

$$\begin{aligned}
\mathrm{KL}(Q_{x,y} \| P_{x,y}(\theta)) &= \int q(x,y) \log \frac{q(y|x)q(x)}{p(y|x,\theta)q(x)} dxdy \\
&= \int q(x) \int q(y|x) \log \frac{q(y|x)}{p(y|x,\theta)} dydx \\
&= \mathrm{E}_{Q_x} \left[\mathrm{KL}(Q_{y|x} \| P_{y|x}(\theta)) \right] \qquad (2)
\end{aligned}$$

It is often the case that we only have samples from Q_x and no direct knowledge of its density function. Or the expectation w.r.t. Q_x in eqn. 2 may be too difficult to compute. In such cases, we can substitute an empirical *training* distribution $\hat{Q}_x$ in for Q_x, which is given by a set S_x of samples from Q_x. This results in the objective

$$\mathrm{E}_{\hat{Q}_x} \left[\mathrm{KL}(Q_{y|x} \| P_{y|x}(\theta)) \right] = \frac{1}{|S|} \sum_{x \in S_x} \mathrm{KL}(Q_{y|x} \| P_{y|x}(\theta)) \,.$$

Provided that $q(y|x)$ is known for each x in S_x, and that $\mathrm{KL}(Q_{y|x} \| P_{y|x}(\theta))$ can be efficiently computed, we can use the above expression as our objective. Otherwise, as is often the case, we might only have access to a single sample y from $Q_{y|x}$ for each $x \in S_x$, giving an empirical *training* distribution $\hat{Q}_{y|x}$. Substituting this in for $Q_{y|x}$ gives the objective function

$$\mathrm{E}_{\hat{Q}_x} \left[\mathrm{KL}(\hat{Q}_{y|x} \| P_{y|x}(\theta)) \right] \propto -\frac{1}{|S|} \sum_{(x,y) \in S} \log p(y|x,\theta) \,,$$

where we have extended S_x to a set S of the (x,y) pairs (which agrees with how S was defined in Section 3). Here, the proportionality is with respect to θ, and it hides an additive constant which is technically infinity[6]. This is *effectively* the same objective that is minimized in standard maximum likelihood learning.

6. The constant corresponds to the differential entropy of the Dirac delta distribution centered at y. One can think of this as approaching infinity under the limit-based definition of the Dirac.

This kind of objective function fits into the general supervised learning framework described in Section 3 as follows. We define the learned conditional distribution $P_{y|x}(\theta)$ to be the composition of the deterministic prediction function $f(x, \theta)$ (which may be a neural network), and an "output" conditional distribution $R_{y|z}$ (with associated density function $r(y|z)$), so that

$$P_{y|x}(\theta) = R_{y|f(x,\theta)}\,.$$

We then define the loss function as $L(y, z) = -\log r(y|z)$.

Given a loss function L which is not explicitly defined this way one can typically still find a corresponding R to make the definition apply. In particular, if $\exp(-L(y, z))$ has the same finite integral w.r.t. y for each z, then one can define R by taking $r(y|z) \propto \exp(-L(y, z))$, where the proportion is w.r.t. both y and z.

5. Various Definitions of the Natural Gradient and the Fisher Information Matrix

The Fisher information matrix F of $P_{x,y}(\theta)$ w.r.t. θ (aka the "Fisher") is given by

$$F = \mathrm{E}_{P_{x,y}}\left[\nabla \log p(x, y|\theta) \nabla \log p(x, y|\theta)^\top\right] \tag{3}$$

$$= -\mathrm{E}_{P_{x,y}}\left[H_{\log p(x,y|\theta)}\right]\,. \tag{4}$$

where gradients and Hessians are taken w.r.t. θ. It can be immediately seen from the first of these expressions for F that it is positive semi-definite (PSD) (since it's the expectation of something which is trivially PSD, a vector outer-product). And from the second expression we can see that it also has the interpretation of being the negative expected Hessian of $\log p(x, y|\theta)$.

The usual definition of the natural gradient (Amari, 1998) which appears in the literature is

$$\tilde{\nabla} h = F^{-1} \nabla h\,,$$

where F is the Fisher and h is the objective function.

Because $p(x, y|\theta) = p(y|x, \theta)q(x)$, where $q(x)$ doesn't depend on θ, we have

$$\nabla \log p(x, y|\theta) = \nabla \log p(y|x, \theta) + \nabla \log q(x) = \nabla \log p(y|x, \theta)\,,$$

and so F can also be written as the expectation (w.r.t. Q_x) of the Fisher information matrix of $P_{y|x}(\theta)$ as follows:

$$F = \mathrm{E}_{Q_x}\left[\mathrm{E}_{P_{y|x}}\left[\nabla \log p(y|x, \theta) \nabla \log p(y|x, \theta)^\top\right]\right] \quad \text{or} \quad F = -\mathrm{E}_{Q_x}\left[\mathrm{E}_{P_{y|x}}\left[H_{\log p(y|x,\theta)}\right]\right]\,.$$

In Amari (1998), this version of F is computed explicitly for a basic perceptron model (basically a neural network with 0 hidden layers) in the case where $Q_x = N(0, I)$. However, in practice the real $q(x)$ may not be directly available, or it may be difficult to integrate $H_{\log p(y|x,\theta)}$ over Q_x. For example, the conditional Hessian $H_{\log p(y|x,\theta)}$ corresponding to a

multi-layer neural network may be far too complicated to be analytically integrated, even for a very simple Q_x. In such situations Q_x may be replaced with its empirical version $\hat{Q}_x$, giving

$$F = \frac{1}{|S|} \sum_{x \in S_x} \mathrm{E}_{P_{y|x}} \left[\nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^\top \right] \quad \text{or} \quad F = -\frac{1}{|S|} \sum_{x \in S_x} \mathrm{E}_{P_{y|x}} \left[H_{\log p(y|x,\theta)} \right] .$$

This is the version of F considered in Park et al. (2000).

From these expressions we can see that when $L(y,z) = -\log r(y|z)$ (as in Section 4), the Fisher has the interpretation of being the expectation under $P_{x,y}$ of the Hessian of $L(y, f(x,\theta))$:

$$F = \frac{1}{|S|} \sum_{x \in S_x} \mathrm{E}_{P_{y|x}} \left[H_{L(y,f(x,\theta))} \right] .$$

Meanwhile, the Hessian H of h is also given by the expected value of the Hessian of $L(y, f(x,\theta))$, except under the distribution $\hat{Q}_{x,y}$ instead of $P_{x,y}$ (where $\hat{Q}_{x,y}$ is given by the density function $\hat{q}(x,y) = \hat{q}(y|x)\hat{q}(x)$). In other words

$$H = \frac{1}{|S|} \sum_{x \in S_x} \mathrm{E}_{\hat{Q}_{x,y}} \left[H_{L(y,f(x,\theta))} \right] .$$

Thus F and H can be seen as approximations of each other in some sense.

6. Geometric Interpretation

The negative gradient $-\nabla h$ can be interpreted as the steepest descent direction for h in the sense that it yields the greatest instantaneous rate of reduction in h per unit of change in θ, where change in θ is measured using the standard Euclidean norm $\|\cdot\|$. More formally we have

$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg\min_{d : \|d\| \leq \epsilon} h(\theta + d) .$$

This interpretation highlights the strong dependence of the gradient on the Euclidean geometry of the parameter space (as defined by the norm $\|\cdot\|$).

One way to motivate the natural gradient is to show that it (or more precisely its negation) can be viewed as a steepest descent direction, much like the negative gradient can be, except with respect to a metric that is intrinsic to the distributions being modeled, as opposed to the default Euclidean metric which is tied to the given parameterization. In particular, the natural gradient can be derived by adapting the steepest descent formulation to use an alternative definition of (local) distance based on the "information geometry" (Amari and Nagaoka, 2000) of the space of probability distributions. The particular distance function[7] which gives rise to the natural gradient turns out to be

$$\mathrm{KL}(P_{x,y}(\theta + d) \| P_{x,y}(\theta)) .$$

7. Note that this is not a formal "distance" function in the usual sense since it is not symmetric.

To formalize this, one can use the well-known connection between the KL divergence and the Fisher, given by the Taylor series approximation

$$\mathrm{KL}(P_{x,y}(\theta + d) \| P_{x,y}(\theta)) = \frac{1}{2} d^\top F d + O(d^3)\,,$$

where "$O(d^3)$" is short-hand to mean terms that are order 3 or higher in the entries of d. Thus, F defines the local quadratic approximation of this distance, and so gives the mechanism of *local* translation between the geometry of the space of distributions, and that of the original parameter space with its default Euclidean geometry.

To make use of this connection, Arnold et al. (2011) proves for general PSD matrices A that

$$\frac{-A^{-1}\nabla h}{\|\nabla h\|_{A^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,min}_{d : \|d\|_A \leq \epsilon} h(\theta + d)\,,$$

where the notation $\|v\|_B$ is defined by $\|v\|_B = \sqrt{v^\top B v}$. Taking $A = \frac{1}{2}F$ and using the above Taylor series approximation to establish that

$$\mathrm{KL}(P_{x,y}(\theta + d) || P_{x,y}(\theta)) \to \frac{1}{2} d^\top F d = \frac{1}{2} \|d\|_F^2$$

as $\epsilon \to 0$, (Arnold et al., 2011) then proceed to show that

$$-\sqrt{2}\frac{\tilde{\nabla} h}{\|\nabla h\|_{F^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,min}_{d\,:\,\mathrm{KL}(P_{x,y}(\theta+d) \| P_{x,y}(\theta)) \leq \epsilon^2} h(\theta + d)\,,$$

(where we recall the notation $\tilde{\nabla} h = F^{-1} \nabla h$).

Thus the negative natural gradient is indeed the steepest descent direction in the space of distributions where distance is measured in small local neighborhoods by the KL divergence.

Note that both F and $\tilde{\nabla} h$ are defined in terms of the standard basis in θ-space, and so obviously depend on the parameterization of h. But the KL divergence does not, and instead only depends on the form of the predictive distribution $P_{y|x}$. Thus, the direction in distribution space defined implicitly by $\tilde{\nabla} h$ will be invariant to our choice of parameterization (whereas the direction defined by ∇h will not be, in general).

By using the smoothly varying PSD matrix F to locally define a metric tensor at every point in parameter space, a Riemannian manifold can be generated over the space of distributions. Note that the associated metric of this space won't be the square root of the KL divergence (this isn't even a valid metric), although it will be "locally equivalent" to it in the sense that the two functions will approximate each other within a small local neighborhood.

7. 2nd-order Optimization

The basic idea in 2nd-order optimization is to compute the update δ to $\theta \in \mathbb{R}^n$ by minimizing some local quadratic approximation or "model" $M_k(\delta)$ of $h(\theta_k + \delta)$ centered around the

current iterate θ_k. That is, we compute $\delta_k^* = \arg\min_\delta M_k(\delta)$ and then update θ according to $\theta_{k+1} = \theta_k + \alpha_k \delta_k^*$, where $M_k(\delta)$ is defined by

$$M_k(\delta) = \frac{1}{2}\delta^\top B_k \delta + \nabla h(\theta_k)^\top \delta + h(\theta_k)\,,$$

and where $B_k \in \mathbb{R}^{n \times n}$ is the "curvature matrix", which is symmetric. The "sub-problem" of minimizing $M_k(\delta)$ can be solved exactly by solving the $n \times n$ dimensional linear system $B_k \delta = -\nabla h$, whose solution is $\delta^* = -B_k^{-1} \nabla h$ when B_k is positive definite.

Gradient descent, the canonical 1st-order method, can be viewed in the framework of 2nd-order methods as making the choice $B_k = \beta I$ for some β, resulting in the update $\delta_k^* = -\frac{1}{\beta}\nabla h(\theta_k)$. In the case where h is convex and Lipschitz-smooth[8] with constant $\mathcal{L}$, a safe/conservative choice that will ensure convergence with $\alpha_k = 1$ is $\beta = \mathcal{L}$ (e.g. Nesterov, 2013). The intuition behind this choice is that B will act as a *global* upper bound on the curvature of h, in the sense that $B_k = \mathcal{L} I \succeq H(\theta)$[9] for all θ, so that δ_k^* never extends past the point that would be safe in the worst-case scenario where the curvature is at its upper bound $\mathcal{L}$ the entire way along δ^*. More concretely, one can show that given this choice of β, $M_k(\delta)$ upper bounds $h(\theta_k + \delta)$, and will therefore never predict a reduction in $h(\theta_k + \delta)$ where there is actually a sharp increase (e.g. due to h curving unexpectedly upward on the path from θ_k to $\theta_k + \delta$). Minimizing $M_k(\delta)$ is therefore guaranteed not to increase $h(\theta_k + \delta)$ beyond the current value $h(\theta_k)$ since $M_k(0) = h(\theta_k)$. But despite these nice properties, this choice will almost always overestimate the curvature in most directions, leading to updates that move unnecessarily slowly along directions of consistent low curvature.

While neural networks haven't been closely studied by optimization researchers until somewhat recently, many of the local optimization issues related to neural network learning can be seen as special cases of problems which arise more generally in continuous optimization. For example, tightly coupled parameters with strong local dependencies, and large variations in scale along different directions in parameter space (which may arise due to the "vanishing gradient" phenomenon (Hochreiter et al., 2000)), are precisely the sorts of issues for which 2nd-order optimization is well suited. Gradient descent on the other hand is well known to be very sensitive to such issues, and in order to avoid large oscillations and instability must use a learning rate which is inversely proportional to $\mathcal{L}$. 2nd-order optimization methods provide a much more powerful and elegant solution to the problem of variations in scale/curvature along different directions by selectively re-scaling the gradient along different eigen-directions of the curvature matrix B_k according to their associated curvature (eigenvalue), instead of employing a one-size-fits-all curvature estimate.

In the classical Newton's method we take $B_k = H(\theta_k)$, in which case $M_k(\delta)$ becomes the 2nd-order Taylor-series approximation of h centered at θ_k. This choice gives us the most accurate *local* model of the curvature possible, and allows for rapid exploration of low-curvature directions and thus faster convergence.

Unfortunately, naive implementations of Newton's method can run into numerous problems when applied to neural network training objectives, such as H being sometimes indefinite (and thus $M_k(\delta)$ being unbounded below in directions of negative curvature) and related issues of "model trust", where the method implicitly trusts its own local quadratic

8. By this we mean that $\|\nabla h(\theta) - \nabla h(\theta')\| \leq \mathcal{L}\|\theta - \theta'\|$ for all θ and θ'.
9. Here we define $A \succeq C$ to mean that $A - C$ is PSD.

model of the objective too much, causing it to propose very large updates that may actually increase the h. These problems are usually not encountered with first order methods, but only because they use a very conservative local model that is intrinsically incapable of generating large updates. Fortunately, using the Gauss-Newton approximation to the Hessian (as discussed in Section 8), and/or applying various update damping/trust-region techniques (as discussed in Section 10), the issue of model trust issue in 2nd-order methods can be mostly overcome.

Another important issue preventing the naive application of 2nd-order methods to neural networks is the typically very high dimensionality of the parameter space (n), which prohibits the calculation/storage/inversion of the n^2-entry curvature matrix B_k. To address this, various approximate Newton methods have been developed within the optimization and machine learning communities. These methods work by approximating B_k with something easier to compute/store/invert such as a low-rank or diagonal matrix, or by performing only approximate/incomplete optimization of $M_k(\delta)$. A survey of such methods is outside the scope of this report, but many good references and reviews are available (e.g. Nocedal and Wright, 2006; Fletcher, 2013). Martens (2016) reviews these approaches specifically in the context of neural networks.

Finally, it is worth observing that the *local* optimality of the Hessian-based 2nd-order Taylor series approximation to h won't necessarily yield the fastest possible optimization procedure, as it is possible to imagine quadratic models that take a "longer view" of the objective. (As an extreme example, given knowledge of a global minimizer θ^* of h, one could construct a quadratic model whose minimizer is exactly θ^* but which is a very poor local approximation to $h(\theta)$.) It is possible that the Fisher might give rise to such quadratic models, which would help explain its observed superiority to the Hessian in neural network optimization (Schraudolph, 2002; Martens, 2010; Vinyals and Povey, 2012). We elaborate more on this speculative theory in Section 8.1.

8. The Generalized Gauss-Newton Matrix

This section discusses the Generalized Gauss-Newton matrix of Schraudolph (2002), and justifies its use as an alternative to the Hessian. Its relevance to our discussion of natural gradient methods will be made clear later in Section 9, where we establish a correspondence between this matrix and the Fisher.

The classical Gauss-Newton matrix (or more simply the Gauss-Newton matrix) is the curvature matrix G which arises in the Gauss-Newton method for non-linear least squares problems (e.g. Dennis Jr and Schnabel, 1996; Ortega and Rheinboldt, 2000; Nocedal and Wright, 2006). It is applicable to our standard neural network training objective h in the case where $L(y, z) = \frac{1}{2}\|y - z\|^2$, and is given by

$$G = \frac{1}{|S|} \sum_{(x,y) \in S} J_f^\top J_f \,,$$

where J_f is the Jacobian of $f(x, \theta)$ w.r.t. the parameters θ. It is usually defined as a modified version of the Hessian H of h (w.r.t. θ), obtained by dropping the second term inside the

sum in the following expression for H:

$$H = \frac{1}{|S|} \sum_{(x,y) \in S} \left(J_f^\top J_f - \sum_{j=1}^{m} [y - f(x,\theta)]_j H_{[f]_j} \right) ,$$

where $H_{[f]_j}$ is the Hessian (w.r.t. θ) of the j-th component of $f(x,\theta)$. We can see from this expression that $G = H$ when $y = f(x,\theta)$. And more generally, if the y's are well-described by the model $f(x,\theta) + \epsilon$ for i.i.d. noise ϵ then $G = H$ will hold approximately.

An alternative way to derive the classical Gauss-Newton is to simply replace the non-linear function $f(x,\theta)$ by its own local linear approximation, centered at the current iterate θ_k. In particular, we replace f by $\tilde{f}(x,\theta) = J_f \cdot (\theta - \theta_k) + f(x,\theta_k)$ so that h becomes a quadratic function of θ, with derivative $\nabla h(\theta_k)$ and Hessian given by G.

Beyond the fact that the resulting matrix is PSD and has other nice properties discussed below, there doesn't seem to be any obvious justification for linearizing f (or equivalently, dropping the corresponding term from the Hessian). It's likely that the reasonableness of doing this depends on problem-specific details about L and f, and how the curvature matrix will be used by the optimizer. In Subsection 8.1.3 we discuss how linearizing f might be justified, specifically for wide neural networks, by some recent theoretical analyses.

Schraudolph (2002) showed how the idea of the Gauss-Newton matrix can be generalized to the situation where $L(y,z)$ is *any* loss function which is convex in z. The generalized formula for G is

$$G = \frac{1}{|S|} \sum_{(x,y) \in S} J_f^\top H_L J_f \,, \tag{5}$$

where H_L is the Hessian of $L(y,z)$ w.r.t. z, evaluated at $z = f(x,\theta)$. Because $L(y,z)$ is convex in z, H_L will be PSD for each (x,y), and thus so will G. We will call this G the Generalized Gauss-Newton matrix (GGN). (Note that this definition is sensitive to where we draw the dividing line between the loss function L and the network itself (i.e. z), in contrast to the definition of the Fisher, which is invariant to this choice.)

Analogously to the case of the classical Gauss-Newton matrix (which assumed $L(y,z) = \frac{1}{2}\|y - z\|^2$), the GGN can be obtained by dropping the second term inside the sum of the following expression for the Hessian H:

$$H = \frac{1}{|S|} \sum_{(x,y) \in S} \left(J_f^\top H_L J_f + \sum_{j=1}^{m} \left[\nabla_z L(y,z)|_{z=f(x,\theta)} \right]_j H_{[f]_j} \right) . \tag{6}$$

Here $\nabla_z L(y,z)|_{z=f(x,\theta)}$ is the gradient of $L(y,z)$ w.r.t. z, evaluated at $z = f(x,\theta)$. Note if we have for some local optimum θ^* that $\left[\nabla_z L(y,z)|_{z=f(x,\theta^*)} \right]_j \approx 0$ for each (x,y) and j, which corresponds to the network making an optimal prediction for each training case over each dimension, then $G(\theta^*) = H(\theta^*)$. In such a case, the behavior of a 2nd-order optimizer using G will approach the behavior of standard Newton's method as it converges to θ^*. A weaker condition implying equivalence is that $\frac{1}{S_y(x)} \sum_{y \in S_y(x)} \left[\nabla_z L(y,z)|_{z=f(x,\theta^*)} \right]_j \approx 0$ for all $x \in S_x$ and j, where $S_y(x)$ denotes the set of y's s.t. $(x,y) \in S$, which corresponds to the

network making an optimal prediction for each x in the presence of intrinsic uncertainty about the target y. (This can be seen by noting that $H_{[f]_j}$ doesn't depend on y.)

Like the Hessian, the GGN can be used to define a local quadratic model of h, as given by:

$$M_k(\delta) = \frac{1}{2}\delta^\top G(\theta_k)\delta + \nabla h(\theta_k)^\top \delta + h(\theta_k)\,.$$

In 2nd-order methods based on the GGN, parameter updates are computed by minimizing $M_k(\delta)$ w.r.t. δ. The exact minimizer[10] $\delta^* = -G(\theta_k)^{-1}\nabla h(\theta_k)$ is often too difficult to compute, and so practical methods will often only approximately minimize $M_k(\delta)$ (e.g Dembo et al., 1982; Steihaug, 1983; Dennis Jr and Schnabel, 1996; Martens, 2010; Vinyals and Povey, 2012).

Since computing the whole matrix explicitly is usually too expensive, the GGN is typically accessed via matrix-vector products. To compute such products efficiently one can use the method of Schraudolph (2002), which is a generalization of the well-known method for computing such products with the classical Gauss-Newton (and is also related to the TangentProp method of Simard et al. (1992)). The method is similar in cost and structure to standard backpropagation, although it can sometimes be tricky to implement (see Martens and Sutskever (2012)).

As pointed out in Martens and Sutskever (2011), the GGN can also be derived by a generalization of the previously described derivation of the classical Gauss-Newton matrix to the situation where L is an arbitrary convex loss. In particular, if we substitute the linearization $\tilde{f}$ for f in h as before (where $\tilde{f}(x,\theta) = J_f \cdot (\theta - \theta_i) + f(x,\theta_i)$ is the linearization of f), it is not difficult to see that the Hessian of the resulting modified h will be equal to the GGN.

Schraudolph (2002) advocated that when computing the GGN, L and f be redefined so that as much as possible of the network's computation is performed within L instead of f, while maintaining the convexity of L. This is because, unlike f, L is not linearly approximated in the GGN, and so its associated second-order derivative terms are faithfully captured. What this almost always means in practice is that what is usually thought of as the final non-linearity of the network (i.e. ϕ_ℓ) is folded into L, and the network itself just computes the identity function at its final layer. Interestingly, in many natural situations which occur in practice, doing this gives a much simpler and more elegant expression for H_L. Exactly when and why this happens will be made clear in Section 9.

8.1 Speculation on Possible Advantages of the GGN Over the Hessian

8.1.1 Qualitative Observations

Unlike the Hessian, the GGN is positive semi-definite (PSD). This means that it never models the curvature as negative in any direction. The most obvious problem with negative curvature is that the quadratic model will predict an unbounded improvement in the objective for moving in the associated directions. Indeed, without the use of some kind of trust-region or damping technique (as discussed in Section 10), or pruning/modification

10. This formula assumes $G(\theta_k)$ is invertible. If it's not, the problem will either be unbounded, or the solution can be computed using the pseudo-inverse instead.

of negative curvature directions (Vinyals and Povey, 2012; Dauphin et al., 2014), or self-terminating Newton-CG scheme (Steihaug, 1983), the update produced by minimizing the quadratic model will be infinitely large in such directions.

However, attempts to use such methods in combination with the Hessian have yielded lackluster results for neural network optimization compared to methods based on the GGN (Martens, 2010; Martens and Sutskever, 2012; Vinyals and Povey, 2012). So what might be going on here? While the true curvature of $h(\theta)$ can indeed be negative in a local neighborhood (as measured by the Hessian), we know it must quickly become non-negative as we travel along any particular direction, given that our loss $L(y, z)$ is convex in z and bounded below. Meanwhile, positive curvature predicts a quadratic penalty, and in the worst case merely underestimates how badly the objective will eventually increase along a particular direction. We can thus say that negative curvature is somewhat less "trustworthy" than positive curvature for this reason, and speculate that a 2nd-order method based on the GGN won't have to rely as much on trust-regions etc (which restrict the size of the update and slow down performance) to produce reliable updates.

There is also the issue of estimation from limited data. Because contributions made to the GGN for each training case and each individual component of $f(x, \theta)$ are PSD, there can be no cancellation between positive and negative/indefinite contributions. This means that the GGN can be more robustly estimated from subsets of the training data than the Hessian. (By analogy, consider how much harder it is to estimate the scale of the mean value of a variable when that variable can take on both positive and negative values, and has a mean close to 0.) This property also means that positive curvature from one case or component will never be cancelled out by negative curvature from another case or component. And if we believe that negative curvature is less trustworthy than positive curvature over larger distances, this is probably a good thing.

Despite these nice properties, the GGN is notably *not* an upper bound on the Hessian (in the PSD sense), as it fails to model *all* of the positive curvature contained in the latter. But crucially, it only fails to model the (positive or negative) curvature coming from the network function $f(x, \theta)$, as opposed to the curvature coming from the loss function $L(y, z)$. (To see this, recall the decomposition of the Hessian from eqn. 6, noting that the term dropped from the Hessian depends only on the gradients of L and the Hessian of components of f.) Curvature coming from f, whether it is positive or negative, is arguably less trustworthy/stable across long distance than curvature coming from L, as argued below.

8.1.2 A More Detailed View of the Hessian vs the GGN

Consider the following decomposition of the Hessian, which is a generalization of the one given in eqn. 6:

$$H = \frac{1}{|S|} \sum_{(x,y) \in S} \left(J_f^\top H_L J_f + C + C^\top + \sum_{i=1}^{\ell} \sum_{j=1}^{m_i} \left[\nabla_{a_i} L(y, f)\right]_j J_{s_i}^\top H_{[\phi_i(s_i)]_j} J_{s_i} \right) .$$

Here, a_i, ϕ_i and s_i are defined as in Section 2, $\nabla_{a_i} L(y, f)$ is the gradient of $L(y, f)$ w.r.t. a_i, $H_{[\phi_i(s_i)]_j}$ is the Hessian of $\phi_i(s_i)$ (i.e. the function which computes a_i) w.r.t. s_i, J_{s_i} is the

Jacobian of s_i (viewed as a function of θ and x) w.r.t. θ, and C is given by

$$C = \begin{bmatrix} J_{a_0} \otimes \nabla_{s_1} L(y,f) \\ J_{a_1} \otimes \nabla_{s_2} L(y,f) \\ \vdots \\ J_{a_\ell} \otimes \nabla_{s_{\ell-1}} L(y,f) \end{bmatrix},$$

where $\otimes$ denotes the Kronecker product.

The $C + C^\top$ term represents the contribution to the curvature resulting from the interaction of the different layers. Even in a linear network, where each ϕ_i computes the identity function, this is non-zero, since f will be an order ℓ multilinear function of the parameters. We note that since a_i doesn't depend on W_j for $j > i$, C will be block lower-triangular, with blocks corresponding to the W_i's.

Aside from $C + C^\top$, the contribution to the Hessian made by f comes from a sum of terms of the form $[\nabla_{a_i} L(y,f)]_j J_{s_i}^\top H_{[\phi_i(s_i)]_j} J_{s_i}$. These terms represent the curvature of the activation functions, and will be zero in a linear network (since we would have $H_{[\phi_i(s_i)]_j} = 0$). It seems reasonable to suspect that the sign of these terms will be subject to rapid and unpredictable change, resulting from sign changes in both $H_{[\phi_i(s_i)]_j}$ and $[\nabla_{a_i} L(y,f)]_j$. The former is the "local Hessian" of ϕ_i, and will change signs as the function ϕ_i enters its different convex and concave regions (ϕ_i is typically non-convex). $[\nabla_{a_i} L(y,f)]_j$ meanwhile is the loss derivative w.r.t. that unit's output, and depends on the behavior of all of the layers above a_i, and on which "side" of the training target the network's current prediction is (which may flip back and forth at each iteration).

This is to be contrasted with the term $J_f^\top H_L J_f$, which represents the curvature of the loss function L, and which remains PSD everywhere (and for each individual training case). Arguably, this term will be more stable w.r.t. changes in the parameters, especially when averaged over the training set.

8.1.3 Some Insights From Other Works

In the case of the squared error loss $L(y,z) = \frac{1}{2}\|y-z\|^2$ (which means that the GGN reduces to the standard Gauss-Newton matrix) with $m = 1$, Chen (2011) established that the GGN is the unique matrix which gives rise to a local quadratic approximation of $L(y, f(x,\theta))$ which is both non-negative (as L itself is), and vanishing on a subspace of dimension $n-1$ (which is the dimension of the *manifold* on which L itself vanishes). Notably, the quadratic approximation produced using the Hessian need not have either of these properties. By summing over output components and averaging over the training set S, one should be able to generalize this result to the entire objective $h(\theta)$ with $m \geq 1$. Thus, we see that the GGN gives rise to a quadratic approximation which shares certain global characteristics with the true $h(\theta)$ that the 2nd-order Taylor series doesn't, despite being a less precise approximation to $h(\theta)$ in a strictly *local* sense.

Botev et al. (2017) observed that for networks with piece-wise linear activation functions, such as the popular RELUs (given by $[\phi_i(s_i)]_j = \max([s_i]_j, 0)$), the GGN and the Hessian will coincide on the diagonal blocks whenever the latter is well-defined. This can be seen from the above decomposition of H by noting that $C + C^\top$ is zero on the diagonal blocks,

and that for piece-wise linear activation functions we have $H_{[\phi_i(s_i)]_j} = 0$ everywhere that this quantity exists (i.e. everywhere except the "kinks" in the activation functions).

Finally, under certain realistic assumptions on the network architecture and initialization point, and a lower bound on the width of the layers, recent results have shown that the f function for a neural network behaves very similarly to its local linear approximation (taken at the initial parameters) throughout the entirety of optimization. This happens both for gradient descent (Du et al., 2018; Jacot et al., 2018; Lee et al., 2019), and natural gradient descent / GGN-based methods (Zhang et al., 2019; Cai et al., 2019), applied to certain choices for L. Not only does this allow one to prove strong global convergence guarantees for these algorithms, it lends support to the idea that modeling the curvature in f (which is precisely the part of the Hessian that the GGN throws out) may be pointless for the purposes of optimization in neural networks, and perhaps even counter-productive.

9. Computational Aspects of the Natural Gradient and Connections to the Generalized Gauss-Newton Matrix

9.1 Computing the Fisher (and Matrix-Vector Products With It)

Note that

$$\nabla \log p(y|x,\theta) = J_f^\top \nabla_z \log r(y|z)\,,$$

where J_f is the Jacobian of $f(x,\theta)$ w.r.t. θ, and $\nabla_z \log r(y|z)$ is the gradient of $\log r(y|z)$ w.r.t. z, evaluated at $z = f(x,\theta)$ (with r defined as near the end of Section 4).

As was first shown by Park et al. (2000), the Fisher information matrix is thus given by

$$\begin{aligned} F &= \mathrm{E}_{Q_x}\left[\mathrm{E}_{P_{y|x}}\left[\nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^\top\right]\right] \\ &= \mathrm{E}_{Q_x}[\mathrm{E}_{P_{y|x}}[J_f^\top \nabla_z \log r(y|z) \nabla_z \log r(y|z)^\top J_f]] \\ &= \mathrm{E}_{Q_x}[J_f^\top \,\mathrm{E}_{P_{y|x}}[\nabla_z \log r(y|z) \nabla_z \log r(y|z)^\top] J_f] = \mathrm{E}_{Q_x}[J_f^\top F_R J_f]\,, \end{aligned}$$

where F_R is the Fisher information matrix of the predictive distribution $R_{y|z}$ at $z = f(x,\theta)$. F_R is itself given by

$$F_R = \mathrm{E}_{P_{y|x}}[\nabla_z \log r(y|z) \nabla_z \log r(y|z)^\top] = \mathrm{E}_{R_{y|f(x,\theta)}}[\nabla_z \log r(y|z) \nabla_z \log r(y|z)^\top]$$

or

$$F_R = -\,\mathrm{E}_{R_{y|f(x,\theta)}}[H_{\log r(y|z)}]\,,$$

where $H_{\log r(y|z)}$ is the Hessian of $\log r(y|z)$ w.r.t. z, evaluated at $z = f(x,\theta)$.

Note that even if Q_x's density function $q(x)$ is known, and is relatively simple, only for certain choices of $R_{y|z}$ and $f(x,\theta)$ will it be possible to analytically evaluate the expectation w.r.t. Q_x in the above expression for F. For example, if we take $Q_x = \mathcal{N}(0, I)$, $R_{y|z} = \mathcal{N}(z, \sigma^2)$, and f to be a simple neural network with no hidden units and a single tan-sigmoid output unit, then both F and its inverse can be computed efficiently (Amari, 1998). This situation is exceptional however, and for even slightly more complex models, such as neural

networks with one or more hidden layers, it has never been demonstrated how to make such computations feasible in high dimensions.

Fortunately the situation improves significantly if Q_x is replaced by $\hat{Q}_x$, as this gives

$$F = \mathrm{E}_{\hat{Q}_x}[J_f^\top F_R J_f] = \frac{1}{|S|} \sum_{x \in S_x} J_f^\top F_R J_f \,, \tag{7}$$

which is easy to compute assuming F_R is easy to compute. Moreover, this is essentially equivalent to the expression in eqn. 5 for the generalized Gauss-Newton matrix (GGN), except that we have the Fisher F_R of the predictive distribution ($R_{y|z}$) instead of Hessian H_L of the loss (L) as the "inner" matrix.

Eqn. 7 also suggests a straightforward and efficient way of computing matrix-vector products with F, using an approach similar to the one in Schraudolph (2002) for computing matrix-vector products with the GGN. In particular, one can multiply by J_f using a linearized forward pass (aka forward-mode automatic differentiation), then multiply by F_R (which will be easy if $R_{y|z}$ is sufficiently simple), and then finally multiply by $J_f^\top$ using standard backprop.

9.2 Qualified Equivalence of the GNN and the Fisher

As we shall see in this subsection, the connections between the GGN and Fisher run deeper than just similar expressions and algorithms for computing matrix-vector products.

In Park et al. (2000) it was shown that if the density function of $R_{y|z}$ has the form $r(y|z) = \prod_{j=1}^m c(y_j - z_j)$ where $c(a)$ is some univariate density function over $\mathbb{R}$, then F is equal to a re-scaled[11] version of the classical Gauss-Newton matrix for non-linear least squares, with regression function given by f. And in particular, the choice $c(a) = \exp(-a^2/2)$ turns the learning problem into non-linear least squares, and F into the classical Gauss-Newton matrix.

Heskes (2000) showed that the Fisher and the classical Gauss-Newton matrix are equivalent in the case of the squared error loss, and proposed using the Fisher as an alternative to the Hessian in more general contexts. Concurrently with this work, Pascanu and Bengio (2014) showed that for several common loss functions like cross-entropy and squared error, the GGN and Fisher are equivalent.

We will show that in fact there is a much more general equivalence between the two matrices, starting from the observation that the expressions for the GGN in eqn. 5 and Fisher in eqn. 7 are identical up to the equivalence of H_L and F_R.

First, note that $L(y, z)$ might not even be convex in z, so that it wouldn't define a valid GGN matrix. But even if $L(y, z)$ is convex in z, it won't be true in general that $F_R = H_L$, and so the GGN and Fisher will differ. However, there is an important class of $R_{y|z}$'s for which $F_R = H_L$ will hold, provided that we have $L(y, z) = -\log r(y|z)$ (putting us in the framework of Section 4).

Notice that $F_R = -\mathrm{E}_{R_{y|f(x,\theta)}}[H_{\log r(y|z)}]$, and $H_L = -H_{\log r(y|z)}$ (which follows from $L(y, z) = -\log r(y|z)$). Thus, the two matrices being equal is equivalent to the condition

$$\mathrm{E}_{R_{y|f(x,\theta)}}[H_{\log r(y|z)}] = H_{\log r(y|z)} \,. \tag{8}$$

11. The re-scaling constant will be determined by the properties of $c(a)$.

While this condition may seem arbitrary, it is actually very natural and holds in the important case where $R_{y|z}$ corresponds to an exponential family model with "natural" parameters given by z. Stated in terms of equations this condition is

$$\log r(y|z) = z^\top T(y) - \log Z(z)$$

for some function $T(y)$, where $Z(z)$ is the normalizing constant/partition function. In this case we have $H_{\log r(y|z)} = -H_{\log Z}$ (which doesn't depend on y), and so eqn. 8 holds trivially.

Examples of such $R_{y|z}$'s include:

- multivariate normal distributions where z parameterizes only the mean μ
- multivariate normal distributions where z is the concatenation of $\Sigma^{-1}\mu$ and the vectorization of Σ^{-1}
- multinomial distributions where the softmax of z is the vector of probabilities for each class

Note that the loss function L corresponding to the multivariate normal is the familiar squared error, and the loss corresponding to the multinomial distribution is the familiar cross-entropy.

Interestingly, the relationship observed by Ollivier et al. (2018) between natural gradient descent and methods based on the extended Kalman filter for neural network training relies on precisely the same condition on $R_{y|z}$. This makes intuitive sense, since the extended Kalman filter is derived by approximating f as affine and then applying the standard Kalman filter for linear/Gaussian systems (which implicitly involves computing a Hessian of a linear model under a squared loss), which is the same approximation that can be used to derive the GGN from the Hessian (see Section 8).

As discussed in Section 8, when constructing the GGN one must pay attention to how f and L are defined with regards to what parts of the neural network's computation are performed by each function (this choice is irrelevant to the Fisher). For example, the softmax computation performed at the final layer of a classification network is usually considered to be part of the network itself and hence to be part of f. The output $f(x, \theta)$ of this computation are normalized probabilities, which are then fed into a cross-entropy loss of the form $L(y, z) = -\sum_j y_j \log z_j$. But the other way of doing it, which Schraudolph (2002) recommends, is to have the softmax function be part of L instead of f, which results in a GGN which is slightly closer to the Hessian due to "less" of the computational pipeline being linearized before taking the 2nd-order Taylor series approximation. The corresponding loss function is $L(y, z) = -\sum_j y_j z_j + \log(\sum_j \exp(z_j))$ in this case. As we have established above, doing it this way also has the nice side effect of making the GGN equivalent to the Fisher, provided that $R_{y|z}$ is an exponential family model with z as its natural parameters.

This (qualified) equivalence between the Fisher and the GGN suggests how the GGN can be generalized to cases where it might not otherwise be well-defined. In particular, it suggests formulating the loss as the negative log density for some distribution, and then taking the Fisher of this distribution. Sometimes, this might be as simple as defining $r(y|z) \propto \exp(-L(y, z))$ as per the discussion at the end of Section 4.

For example, suppose our loss is defined as the negative log probability of a multi-variate normal distribution $R_{y|z} = N(\mu, \sigma^2)$ parameterized by μ and $\gamma = \log \sigma^2$ (so that $z = \begin{bmatrix} \mu \\ \gamma \end{bmatrix}$). In other words, suppose that

$$L(y, z) = -\log r(y|z) \propto \frac{1}{2}\gamma + \frac{1}{2\exp(\gamma)}(x - \mu)^2 \,.$$

In this case the loss Hessian is equal to

$$H_L = \frac{1}{\exp(\gamma)} \begin{bmatrix} 1 & x - \mu \\ x - \mu & \frac{1}{2}(x - \mu)^2 \end{bmatrix} \,.$$

It is not hard to verify that this matrix is indefinite for certain settings of x and z (e.g. $x = 2$, $\mu = \gamma = 0$). Therefore, L is not convex in z and we cannot define a valid GGN matrix from it.

To resolve this problem we can use the Fisher F_R in place of H_L in the formula for the GGN, which by eqn. 7 yields F. Alternatively, we can insert reparameterization operations into our network to transform μ and γ into the natural parameters $\frac{\mu}{\sigma^2} = \frac{\mu}{\exp(\gamma)}$ and $-\frac{1}{2\sigma^2} = -\frac{1}{2\exp(\gamma)}$, and then proceed to compute the GGN as usual, noting that $H_L = F_R$ in this case, so that H_L will be PSD. Either way will yield the same curvature matrix, due to the above discussed equivalence of the Fisher and GGN matrix for natural parameterizations.

10. Constructing Practical Natural Gradient Methods, and the Critical Role of Damping

Assuming that it is easy to compute, the simplest way to use the natural gradient in optimization is to substitute it in place of the standard gradient within a basic gradient descent approach. This gives the iteration

$$\theta_{k+1} = \theta_k - \alpha_k \tilde{\nabla} h(\theta_k) \,, \tag{9}$$

where $\{\alpha_k\}_k$ is a schedule of step-sizes/learning-rates.

Choosing the step-size schedule can be difficult. There are adaptive schemes which are largely heuristic in nature (Amari, 1998) and some non-adaptive prescriptions such as $\alpha_k = \rho/k$ for some constant ρ, which have certain theoretical convergence guarantees in the stochastic setting, but which won't necessarily work well in practice.

In principle, we could apply the natural gradient method with infinitesimally small steps and produce a smooth idealized path through the space of realizable distributions. But since this is usually impossible in practice, and we don't have access to any other simple description of the class of distributions parameterized by θ that we could work with more directly, our only option is to take non-negligible discrete steps in the given parameter space[12].

12. In principle, we could move to a much more general class of distributions, such as those given by some non-parametric formulation, where we could work directly with the distributions themselves. But even assuming such an approach would be practical from a computational efficiency standpoint, we would lose the various advantages that we get from working with powerful parametric models like neural networks. In particular, we would lose their ability to generalize to unseen data by modeling the "computational process" which explains the data, instead of merely using smoothness and locality to generalize.

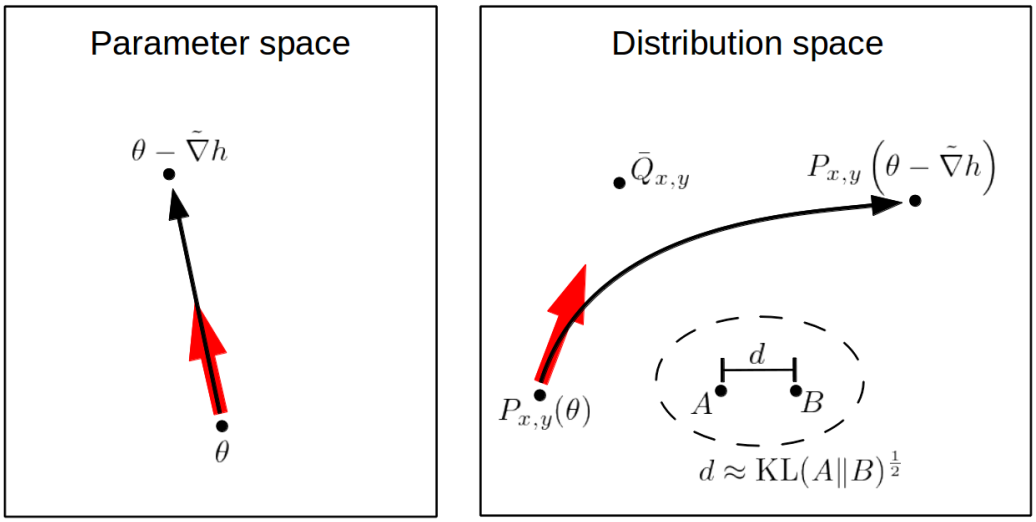

Figure 1: A typical situation encountered when performing large discrete updates in the original parameter space. The red arrow is the natural gradient direction (given by the vector $\tilde{\nabla} h$ in parameter space) and the black arrow is the path generated by taking $\theta - \alpha \tilde{\nabla} h$ for $\alpha \in [0, 1]$.

The fundamental problem with simple schemes such as the one in eqn. 9 is that they implicitly assume that the natural gradient is a good direction to follow over non-negligible distances in the original parameter space, which will not be true in general. Traveling along a straight line in the original parameter space will not yield a straight line in distribution space, and so the resulting path may instead veer far away from the target that the natural gradient originally pointed towards. This is illustrated in Figure 1.

Fortunately, we can exploit the (qualified) equivalence between the Fisher and the GGN in order to produce natural gradient-like updates which will often be appropriate to take with $\alpha_k = 1$. In particular, we know from the discussion in Section 8 that the GGN matrix G can serve as a reasonable proxy for the Hessian H of h, and may even be superior in certain contexts. Meanwhile, the update δ produced by minimizing the GGN-based local quadratic model $M_k(\delta) = \frac{1}{2}\delta^\top G(\theta_k)\delta + \nabla h(\theta_k)^\top \delta + h(\theta_k)$ is given by $-G(\theta_k)^{-1}\nabla h(\theta_k)$, which will be equal to the negative natural gradient when $F = G$. Thus, the (negative) natural gradient, with scaling factor $\alpha = 1$, can be seen as the optimal update according to a particular local 2nd-order approximation of h. And just as in the case of other 2nd-order methods, the break-down in the accuracy of this quadratic approximation over long distances, combined with the potential for the natural gradient to be very large (e.g. when F contains some very small eigenvalues), can often lead to very large and very poor update proposals. Simply re-scaling the update by reducing α may be too crude a mechanism to deal with this subtle problem, as it will affect all eigen-directions (of F) equally, including those in which the natural gradient is already sensible, or even overly conservative.

Instead, the connection between natural gradient descent and 2nd-order methods motivates the use of "update damping" techniques that have been developed for the latter, which work by constraining or penalizing the solution for δ in various ways during the optimization

of $M_k(\delta)$. Examples include Tikhonov regularization/damping and the closely related trust-region method (e.g. Tikhonov, 1943; Moré and Sorensen, 1983; Conn et al., 2000; Nocedal and Wright, 2006), and other ones such as the "structural damping" approach of Martens and Sutskever (2011), or the approach present in Krylov Subspace Descent (Vinyals and Povey, 2012). See Martens and Sutskever (2012) for an in-depth discussion of these and other damping techniques in the context of neural network optimization.

This idea is supported by practical experience in neural network optimization. For example, the Hessian-free optimization approach of Martens (2010) generates its updates using a Tikhonov damping scheme applied to the exact GGN matrix (which was equivalent to the Fisher in that work). These updates, which can be applied with a step-size of 1, make a lot more progress optimizing the objective than updates computed without any damping (which must instead rely on a carefully chosen step-size to even be feasible).

It is worth pointing out that other interpretations of natural gradient descent can also motivate the use of damping/regularization terms. In particular, Ollivier et al. (2018) has shown that online natural gradient descent, with a particular flavor of Tikhonov regularization, closely resembles a certain type of extended Kalman filter-based training algorithm for neural networks (Singhal and Wu, 1989; Ruck et al., 1992), where θ is treated as an evolving hidden state that is estimated by the filter (using training targets as noisy observations and inputs as control signals).

11. The Empirical Fisher

An approximation of the Fisher known as the "empirical Fisher" (Schraudolph, 2002), which we denote by $\bar{F}$, is commonly used in practical natural gradient methods. It is obtained by taking the inner expectation of eqn. 3 over the target distribution $Q_{x,y}$ (or its empirical surrogate $\hat{Q}_{x,y}$) instead of the model's distribution $P_{x,y}$.

In the case where one uses $\hat{Q}_{x,y}$, this yields the following simple form:

$$\begin{aligned}\bar{F} &= \mathrm{E}_{\hat{Q}_{x,y}}\left[\nabla \log p(x,y|\theta) \nabla \log p(x,y|\theta)^\top\right] \\ &= \mathrm{E}_{\hat{Q}_x}\left[\mathrm{E}_{\hat{Q}_{y|x}}\left[\nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^\top\right]\right] \\ &= \frac{1}{|S|}\sum_{(x,y)\in S} \nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^\top .\end{aligned}$$

This matrix is often incorrectly referred to as the Fisher, or even the Gauss-Newton, even though it is not equivalent to either of these matrices in general.

11.1 Comparisons to the Standard Fisher

Like the Fisher F, the empirical Fisher $\bar{F}$ is PSD. But unlike F, it is essentially free to compute, provided that one is already computing the gradient of h. And it can also be applied to objective functions which might not involve a probabilistic model in any obvious way.

Compared to F, which is of rank $\leq |S| \operatorname{rank}(F_R)$, $\bar{F}$ has a rank of $\leq |S|$, which can make it easier to work with in practice. For example, the problem of computing the diagonal (or various blocks) is easier for the empirical Fisher than it is for higher rank matrices like the

standard Fisher (Martens et al., 2012). This has motivated its use in optimization methods such as TONGA (Le Roux et al., 2008), and as the diagonal preconditioner of choice in the Hessian-free optimization method (Martens, 2010). Interestingly however, there are stochastic estimation methods (Chapelle and Erhan, 2011; Martens et al., 2012) which can be used to efficiently estimate the diagonal (or various blocks) of the standard Fisher F, and these work quite well in practice. (These include the obvious method of sampling y's from the model's conditional distribution and computing gradients from them, but also includes methods based on matrix factorization and random signs. See Martens et al. (2012) for comparative analysis of the variance of these methods.)

Despite the various practical advantages of using $\bar{F}$, there are good reasons to use true Fisher F instead of $\bar{F}$ whenever possible. In addition to Amari's extensive theory developed for the exact natural gradient (which uses F), perhaps the best reason for using F over $\bar{F}$ is that F turns out to be a reasonable approximation/substitute to the Hessian H of h in certain important special cases, which is a property that $\bar{F}$ lacks in general.

For example, as discussed in Section 5, when the loss is given by $-\log p(y|x)$ (as in Section 4), F can be seen as an approximation of H, because both matrices have the interpretation of being the expected Hessian of the loss under some distribution. Due to the similarity of the expression for F in eqn. 3 and the one above for $\bar{F}$, it might be tempting to think that $\bar{F}$ is given by the expected Hessian of the loss under $\hat{Q}_{x,y}$ (which is actually the formula for H) in the same way that F is given by eqn. 4. But this is not the case in general.

And as we saw in Section 9, given certain assumptions about how the GGN is computed, and some additional assumptions about the form of the loss function L, F turns out to be equivalent to the GGN. This is very useful since the GGN can be used to define a local quadratic approximation of h, whereas F normally doesn't have such an interpretation. Moreover, Schraudolph (2002) and later Martens (2010) compared $\bar{F}$ to the GGN and observed that the latter performed much better as a curvature matrix within various neural network optimization methods.

As concrete evidence for why the empirical Fisher is, at best, a questionable choice for the curvature matrix, we will consider the following example. Set $n = 1$, $f(x, \theta) = \theta$, $R_{y|z} = \mathcal{N}(z, 1)$, and $S = \{(0, 0)\}$, so that $h(\theta)$ is a simple convex quadratic function of θ, given by $h(\theta) = \frac{1}{2}\theta^2$. In this example we have that $\nabla h = \theta$, $\bar{F} = \theta^2$, while $F = 1$. If we use $\bar{F}^\xi$ as our curvature matrix for some exponent $\frac{1}{2} \leq \xi \leq 1$, then it is easy to see that an iteration of the form

$$\theta_{k+1} = \theta_k - \alpha_k(\bar{F}(\theta_k)^\xi)^{-1}\nabla h(\theta_k) = \theta_k - \alpha_k(\theta_k^2)^{-\xi}\theta_k = (1 - \alpha_k|\theta_k|^{-2\xi})\theta_k$$

will fail to converge to the minimizer (at $\theta = 0$) unless $\xi < 1$ and the step-size α_k goes to 0 sufficiently fast. And even when it does converge, it will only be at a rate comparable to the speed at which α_k goes to 0, which in typical situations will be either $\mathcal{O}(1/k)$ or $\mathcal{O}(1/\sqrt{k})$. Meanwhile, a similar iteration of the form

$$\theta_{k+1} = \theta_k - \alpha_k F^{-1}\nabla h(\theta_k) = \theta_k - \alpha_k\theta_k = (1 - \alpha_k)\theta_k\,,$$

which uses the exact Fisher F as the curvature matrix, will experience very fast linear convergence[13] with rate $|1 - \alpha|$, for any fixed step-size $\alpha_k = \alpha$ satisfying $0 < \alpha < 2$.

13. Here we mean "linear" in the classical sense that $|\theta_k - 0| \leq |\theta_0 - 0||1 - \alpha|^k$.

It is important to note that this example uses a noise-free version of the gradient, and that this kind of linear convergence is (provably) impossible in most realistic stochastic/online settings. Nevertheless, we would argue that a highly desirable property of any stochastic optimization method should be that it can, in principle, revert to an optimal (or nearly optimal) behavior in the deterministic setting. This might matter a lot in practice, since the gradient may end up being sufficiently well estimated in earlier stages of optimization from only a small amount of data (which is a common occurrence in our experience), or in later stages provided that larger mini-batches or other variance-reducing procedures are employed (e.g. Le Roux et al., 2012; Johnson and Zhang, 2013). More concretely, the pre-asymptotic convergence rate of stochastic 2nd-order optimizers can still depend strongly on the choice of the curvature matrix, as we will show in Section 14.

11.2 A Discussion of Recent Diagonal Methods Based on the Empirical Fisher

Recently, a spate of stochastic optimization methods have been proposed that are all based on diagonal approximations of the empirical Fisher $\bar{F}$. These include the diagonal version of AdaGrad (Duchi et al., 2011), RMSProp (Tieleman and Hinton, 2012), Adam (Ba and Kingma, 2015), etc. Such methods use iterations of the following form (possibly with some slight modifications):

$$\theta_{k+1} = \theta_k - \alpha_k (B_k + \lambda I)^{-\xi} g_k(\theta_k)\,, \tag{10}$$

where the curvature matrix B_k is taken to be a diagonal matrix $\mathrm{diag}(u_k)$ with u_k adapted to maintain some kind of estimate of the diagonal of $\bar{F}$ (possibly using information from previous iterates/mini-batches), $g_k(\theta_k)$ is an estimate of $\nabla h(\theta_k)$ produced from the current mini-batch, α_{kk} is a schedule of step-sizes, and $0 < \lambda$ and $0 < \xi \leq 1$ are hyperparameters (discussed later in this section).

There are also slightly more sophisticated methods (Schaul et al., 2013; Zeiler, 2013) which use preconditioners that combine the diagonal of $\bar{F}$ with other quantities (such as an approximation of the diagonal of the Gauss-Newton/Fisher in the case of Schaul et al. (2013)) in order to correct for how the empirical Fisher doesn't have the right "scale" (which is ultimately the reason why it does poorly in the example given at the end of Section 11.1).

A diagonal preconditioner (Nash, 1985) of the form used in eqn. 10 was also used by (Martens, 2010) to accelerate the conjugate gradient (CG) sub-optimizations performed within a truncated-Newton method (using the GGN matrix). In the context of CG, the improper scale of $\bar{F}$ is not as serious an issue due to the fact that CG is invariant to the overall scale of its preconditioner (since it computes an optimal "step-size" at each step which automatically adjusts for the scale). However, it still makes more sense to use the diagonal of the true Fisher F as a preconditioner, and thanks to the method proposed by Chapelle and Erhan (2011), this can be estimated efficiently and accurately.

The idea of using the diagonal of F, $\bar{F}$, or the Gauss-Newton as a preconditioner for stochastic gradient descent (SGD) and was likely first applied to neural networks with the work of Lecun and collaborators (Becker and LeCun, 1989; LeCun et al., 1998), who proposed an iteration of the form in eqn. 10 with $\xi = 1$ where u_k approximates the diagonal of the Hessian or the Gauss-Newton matrix (which as shown in Section 9, is actually equivalent to F for the common squared-error loss). Following this work, various neural network

optimization methods have been developed over the last couple of decades that use diagonal, block-diagonal, low-rank, or Krylov-subspace based approximations of F or $\bar{F}$ as a curvature matrix/preconditioner. In addition to methods based on diagonal approximations already mentioned, some methods based on non-diagonal approximations include the method of Park et al. (2000), TONGA (Le Roux et al., 2008), Natural Newton (Le Roux and Fitzgibbon, 2010), HF (Martens, 2010), KSD (Vinyals and Povey, 2012) and many more.

The idea of computing an estimate of the (empirical) Fisher using a history of previous iterates/mini-batches also appeared in various early works. The particular way of doing this proposed Duchi et al. (2011), which is to use an equally weighted average of all past gradients, was motivated from a regret-based asymptotic convergence analysis and tends not to work well in practice (Tieleman and Hinton, 2012). The traditional and more intuitive approach of using an exponentially decayed running average (e.g. LeCun et al., 1998; Park et al., 2000) works better, at least pre-asymptotically, as it is able to naturally "forget" very old contributions to the estimate (which are based on stale parameter values).

It is important to observe that the way $\bar{F}$ is estimated can affect the convergence characteristics of an iteration like eqn. 10 in subtle but important ways. For example, if $\bar{F}$ is estimated using gradients from previous iterations, and especially if it is the average of *all* past gradients (as in AdaGrad), it may shrink sufficiently slowly that the convergence issues seen in the example at the end of Section 11.1 are avoided. Moreover, for reasons related to this phenomenon, it seems likely that the proofs of regret bounds in Duchi et al. (2011) and the related work of Hazan et al. (2007) could *not* be modified to work if the exact $\bar{F}$, computed only at the current θ, were used. Developing a better understanding of this issue, and the relationship between methods developed in the online learning literature (such as AdaGrad), and classical stochastic 2nd-order methods based on notions of curvature, remains an interesting direction for future research.

11.3 The Constants λ and ξ

The constants λ and ξ present in eqn. 10 are often thought of as fudge factors designed to correct for the "poor conditioning" (Becker and LeCun, 1989) of the curvature matrix, or to guarantee boundedness of the updates and prevent the optimizer from "blowing up" (LeCun et al., 1998). However, these explanations are oversimplifications that reference the symptoms instead of the cause. A more compelling and functional explanation, at least in the case of λ, comes from viewing the update in eqn. 10 as being the minimizer of a local quadratic approximation $M_k(\delta) = \frac{1}{2}\delta^\top B_k \delta + \nabla h(\theta_k)^\top \delta + h(\theta_k)$ to $h(\theta_k + \delta)$, as discussed in Section 10. In this view, λ plays the role of a Tikhonov damping parameter (Tikhonov, 1943; Conn et al., 2000; Nocedal and Wright, 2006; Martens and Sutskever, 2012) which is added to B_k in order to ensure that the proposed update stays within a certain region around zero in which $M_k(\delta)$ remains a reasonable approximation to $h(\theta_k + \delta)$. Note that this explanation implies that no single fixed value of λ will be appropriate throughout the entire course of optimization, since the local properties of the objective will change, and so an adaptive adjustment scheme, such as the one present in HF (Martens, 2010) (which is based on the Levenberg-Marquardt method), should be used.

The use of the exponent $\xi = 3/4$ first appeared in HF as part of its diagonal preconditioner for CG, and was justified as a way of making the curvature estimate "more conservative" by making it closer to a multiple of the identity, to compensate for the diagonal approximation being made (among other things). Around the same time, Duchi et al. (2011) proposed to use $\xi = 1/2$ within an update of the form of eqn. 10, which was required in order to prove certain regret bounds for non-strongly-convex objectives.

To shed some light on the question of ξ, we can consider the work of Hazan et al. (2007), who like Duchi et al. (2011), developed and analyzed an online approximate Newton method within the framework of online convex optimization. Like the non-diagonal version of AdaGrad, the method proposed by Hazan et al. (2007) uses an estimate of the empirical Fisher $\bar{F}$ computed as the average of gradients from all previous iterations. While impractical for high dimensional problems like any non-diagonal method is (or at least, one that doesn't make some other strong approximation of the curvature matrix), this method achieves a better bound on the regret to what Duchi et al. (2011) was able to show for AdaGrad ($\mathcal{O}(\log(k))$ instead of $\mathcal{O}(\sqrt{k})$, where k is the total number of iterations), which was possible in part due to the use of stronger hypotheses about the properties of h (e.g. that for each x and y, $L(y, f(x, \theta))$ is a strongly convex function of θ). Notably, this method uses $\xi = 1$, just as in standard natural gradient descent, which provides support for such a choice, especially since the h used in neural networks will typically satisfy these stronger assumptions in a local neighborhood of the optimum, at least when standard ℓ_2 regularization is used.

However, it is important to note that Hazan et al. (2007) also proves a $\mathcal{O}(\log(k))$ bound on the regret for a basic version of SGD, and that what actually differentiates the various methods they analyze is the constant hidden in the big-O notation, which is much larger for the version of SGD they consider than for their approximate Newton method. In particular, the former depends on a quantity which grows with the condition number of the Hessian H at θ^* while the latter does not, in a way that echos the various analyses performed on stochastic gradient descent and stochastic approximations of Newton's method in the more classical "local-convergence" setting (e.g. Murata, 1998; Bottou and LeCun, 2005).

12. A Critical Analysis of Parameterization Invariance

One of the main selling points of the natural gradient method is its invariance to reparameterizations of the model. In particular, the smooth path through the space of distributions generated by the idealized natural gradient method with infinitesimally small steps will be invariant to any smooth invertible reparameterization of the f.

More precisely, it can be said that this path will be the same whether we use the default parameterization (given by $P_{y|x}(\theta)$), or parameterize our model as $P_{y|x}(\zeta(\gamma))$, where $\zeta : \mathbb{R}^n \to \mathbb{R}^n$ is a smooth invertible "reparameterization function" which relates θ to γ as $\theta = \zeta(\gamma)$.

In this section we will examine this "smooth path parameterization invariance" property more closely in order to answer the following questions:

- How can we characterize it using only basic properties of the curvature matrix?
- Is there an elementary proof that can be applied in a variety of settings?

- What other kinds of curvature matrices give rise to it, and is the Hessian included among these?
- Will this invariance property imply that *practical* optimization algorithms based on the natural gradient (i.e. those that use large steps) will behave in a way that is invariant to the parameterization?

Let ζ be as above, and let d_θ and d_γ be updates given in θ-space and γ-space (resp.). Additively updating γ by d_γ and translating it back to θ-space via ζ gives $\zeta(\gamma + d_\gamma)$. Measured by some non-specific norm $\|\cdot\|$, this differs from $\theta + d_\theta$ by:

$$\|\zeta(\gamma + d_\gamma) - (\theta + d_\theta)\| .$$

This can be rewritten and bounded as

$$\|(\zeta(\gamma + d_\gamma) - (\zeta(\gamma) + J_\zeta d_\gamma)) + (J_\zeta d_\gamma - d_\theta)\| \leq \|\zeta(\gamma + d_\gamma) - (\zeta(\gamma) + J_\zeta d_\gamma)\| + \|J_\zeta d_\gamma - d_\theta\| , \tag{11}$$

where J_ζ is the Jacobian of ζ, and we have used $\theta = \zeta(\gamma)$.

The first term on the RHS of eqn. 11 measures the extent to which $\zeta(\gamma + d_\gamma)$ fails to be predicted by the first-order Taylor series approximation of ζ centered at γ (i.e. the local affine approximation of ζ at γ). This quantity will depend on the size of d_γ, and the amount of curvature in γ. In the case where ζ is affine, it will be exactly 0. We can further bound it by applying Taylor's theorem for each component of ζ, which gives

$$\|\zeta(\gamma + d_\gamma) - (\zeta(\gamma) + J_\zeta d_\gamma)\| \leq \frac{1}{2} \left\| \begin{bmatrix} d_\gamma^\top H_{[\zeta]_1}(\gamma + c_1 d_\gamma) d_\gamma \\ d_\gamma^\top H_{[\zeta]_2}(\gamma + c_2 d_\gamma) d_\gamma \\ \vdots \\ d_\gamma^\top H_{[\zeta]_n}(\gamma + c_n d_\gamma) d_\gamma \end{bmatrix} \right\| \tag{12}$$

for some $c_i \in (0, 1)$. If we assume that there is some $C > 0$ so that for all i and γ, $\|H_{[\zeta]_i}(\gamma)\|_2 \leq C$, then using the fact that $|d_\gamma^\top H_{[\zeta]_i}(\gamma + c_n d_\gamma) d_\gamma| \leq \frac{1}{2}\|H_{[\zeta]_i}(\gamma + c_i d_\gamma)\|_2 \|d_\gamma\|^2$, we can further upper bound this by $\frac{1}{2} C \sqrt{n} \|d_\gamma\|^2$.

The second term on the RHS of eqn. 11 will be zero when

$$J_\zeta d_\gamma = d_\theta , \tag{13}$$

which (as we will see) is a condition that is satisfied in certain natural situations. A slightly weakened version of this condition is that $J_\zeta d_\gamma \propto d_\theta$. Because we have

$$\lim_{\epsilon \to 0} \frac{\zeta(\gamma + \epsilon d_\gamma) - \zeta(\gamma)}{\epsilon} = J_\zeta d_\gamma$$

this condition can thus be interpreted as saying that d_γ, when translated appropriately via ζ, points in the same direction away from θ that d_θ does. In the smooth path case, where the optimizer only moves an infinitesimally small distance in the direction of d_γ (or d_θ) at each iteration before recomputing it at the new γ (or θ), this condition is sufficient to establish that the path in γ space, when mapped back to θ space via the ζ function, will be

the same as the path which would have been taken if the optimizer had worked directly in θ space.

However, for a practical update scheme where we move the entire distance of d_γ or d_θ before recomputing the update vector, such as the one in eqn. 9, this kind of invariance will not strictly hold even when $J_\zeta d_\gamma = d_\theta$. But given that $J_\zeta d_\gamma = d_\theta$, the per-iteration error will be bounded by the first term on the RHS of eqn. 11, and will thus be small provided that d_γ is sufficiently small and ζ is sufficiently smooth (as shown above).

Now, suppose we generate the updates d_θ and d_γ from curvature matrices B_θ and B_γ according to $d_\theta = -\alpha B_\theta^{-1} \nabla h$ and $d_\gamma = -\alpha B_\gamma^{-1} \nabla_\gamma h$, where $\nabla_\gamma h$ is the gradient of $h(\zeta(\gamma))$ w.r.t. γ. Then noting that $\nabla_\gamma h = J_\zeta^\top \nabla h$, the condition in eqn. 13 becomes equivalent to

$$J_\zeta B_\gamma^{-1} J_\zeta^\top \nabla h = B_\theta^{-1} \nabla h \,.$$

For this to hold, a *sufficient* condition is that $B_\theta^{-1} = J_\zeta B_\gamma^{-1} J_\zeta^\top$. Since J_ζ is invertible (because ζ is) an equivalent condition is

$$J_\zeta^\top B_\theta J_\zeta = B_\gamma \,. \tag{14}$$

The following theorem summarizes our results so far.

Theorem 1. *Suppose that $\theta = \zeta(\gamma)$ and B_θ and B_γ are invertible matrices satisfying*

$$J_\zeta^\top B_\theta J_\zeta = B_\gamma$$

*Then we have that additively updating θ by $d_\theta = -\alpha B_\theta^{-1} \nabla h$ is **approximately** equivalent to additively updating γ by $d_\gamma = -\alpha B_\gamma^{-1} \nabla_\gamma h$, in the sense that $\zeta(\gamma + d_\gamma) \approx \theta + d_\theta$, with error bounded according to*

$$\|\zeta(\gamma + d_\gamma) - (\theta + d_\theta)\| \leq \|\zeta(\gamma + d_\gamma) - (\zeta(\gamma) + J_\zeta d_\gamma)\| \,.$$

Moreover, this error can be further bounded as in eqn. 12, and will be exactly 0 if ζ is affine. And if there is a $C \geq 0$ such that $\|H_{[\zeta]_i}(\gamma)\|_2 \leq C$ for all i and γ, then we can even further bound this as $\frac{1}{2} C \sqrt{n} \|d_\gamma\|^2$.

Because the error bound is zero when ζ is affine, this result will trivially extend to entire sequences of arbitrary number of steps for such ζ's. And in the more general case, since the error scales as α^2, we can obtain equivalence of sequences of T/α steps in the limit as $\alpha \to 0$. Because the length of the updates scale as α, and we have T/α of them, the sequences converges to smooth paths of fixed length in the limit. The following corollary establishes this result, under a few additional (mild) hypotheses. Its proof is in Appendix E.

Corollary 2. *Suppose that B_θ and B_γ are invertible matrices satisfying*

$$J_\zeta^\top B_\theta J_\zeta = B_\gamma$$

for all values of θ. Then the path followed by an iterative optimizer working in θ-space and using additive updates of the form $d_\theta = -\alpha B_\theta^{-1} \nabla h$ is the same as the path followed by an iterative optimizer working in γ-space and using additive updates of the form $d_\gamma = -\alpha B_\gamma^{-1} \nabla_\gamma h$, provided that the optimizers use equivalent starting points (i.e. $\theta_0 = \zeta(\gamma_0)$), and that either

- *ζ is affine,*
- *or d_θ/α is uniformly continuous as a function of θ, d_γ/α is uniformly bounded (in norm), there is a C as in the statement of Theorem 1, and $\alpha \to 0$.*

Note that in the second case we allow the number of steps in the sequences to grow proportionally to $1/\alpha$ so that the continuous paths they converge to have non-zero length as $\alpha \to 0$.

So from these results we see that natural gradient-based methods that take finite steps will *not* be invariant to smooth invertible reparameterizations ζ, although they will be *approximately* invariant, and in a way that depends on the degree of curvature of ζ and the size α of the step-size.

12.1 When is the Condition $J_\zeta^\top B_\theta J_\zeta = B_\gamma$ Satisfied?

Suppose the curvature matrix B_θ has the form

$$B_\theta = \mathrm{E}_{D_{x,y}}[J_f^\top A J_f]\,,$$

where $D_{x,y}$ is some arbitrary distribution over x and y (such as the training distribution), and $A \in \mathbb{R}^{m \times m}$ is an invertible matrix-valued function of x, y and θ, whose value is parameterization invariant (i.e. its value depends only on the value of θ that a given γ maps to under the γ parameterization). Note that this type of curvature matrix includes as special cases the Generalized Gauss-Newton (whether or not it's equivalent to the Fisher), the Fisher, and the empirical Fisher (discussed in Section 11).

To obtain the analogous curvature matrix B_γ for the γ parameterization we replace f by $f \circ \zeta$ which gives

$$B_\gamma = \mathrm{E}_{D_{x,y}}[J_{f\circ\zeta}^\top A\, J_{f\circ\zeta}]\,.$$

Then noting that $J_{f\circ\zeta} = J_f J_\zeta$, where J_ζ is the Jacobian of ζ, we have

$$B_\gamma = \mathrm{E}_{D_{x,y}}[(J_f J_\zeta)^\top A (J_f J_\zeta)] = J_\zeta^\top\, \mathrm{E}_{D_{x,y}}[J_f^\top A J_f] J_\zeta = J_\zeta^\top B_\theta J_\zeta\,.$$

(Here we have used the fact that the reparameterization function ζ is independent of x and y.) Thus, this type of curvature matrix satisfies the sufficient condition in eqn. 14.

The Hessian on the other hand does not satisfy this sufficient condition, except in certain special cases. To see this, note that taking the curvature matrix to be the Hessian gives

$$B_\gamma = J_\zeta^\top H J_\zeta + \frac{1}{|S|} \sum_{(x,y)\in S} \sum_{j=1}^{n} [\nabla h]_j H_{[\zeta]_j}\,,$$

where $H = B_\theta$ is the Hessian of h w.r.t. θ. Thus, when the curvature matrix is the Hessian, the sufficient condition $J_\zeta^\top B_\theta J_\zeta = J_\zeta^\top H J_\zeta \propto B_\gamma$ holds if and only if

$$\frac{1}{|S|} \sum_{(x,y)\in S} \sum_{j=1}^{n} [\nabla h]_j H_{[\zeta]_j} = J_\zeta^\top H J_\zeta\,,$$

where ∇L is the gradient of $L(y, z)$ w.r.t. z (evaluated at $z = f(x, \theta)$), and we allow a proportionality constant of 0. Rearranging this gives

$$\frac{1}{|S|} \sum_{(x,y)\in S} \sum_{j=1}^{n} [\nabla h]_j J_\zeta^{-\top} H_{[\zeta]_j} J_\zeta^{-1} = H \,.$$

This relation is unlikely to be satisfied unless the left hand side is equal to 0. One situation where this will occur is when $H_{[\zeta]_j} = 0$ for each j, which holds when $[\zeta]_j$ is an affine function of γ. Another situation is where we have $\nabla h = 0$ for each $(x, y) \in S$.

13. A New Interpretation of the Natural Gradient

As discussed in Section 10, the negative natural gradient is given by the minimizer of a local quadratic approximation $M(\delta)$ to h whose curvature matrix is the Fisher F. And if we have that the gradient ∇h and F are computed on the same set S of data points, $M(\delta)$ can be written as

$$\begin{aligned}
M(\delta) &= \frac{1}{2}\delta^\top F \delta + \nabla h^\top \delta + h(\theta) \\
&= \frac{1}{|S|} \sum_{(x,y)\in S} \left[\frac{1}{2}\delta^\top J_f^\top F_R J_f \delta + (J_f^\top \nabla_z \log r(y|z))^\top \delta \right] + h(\theta) \\
&= \frac{1}{|S|} \sum_{(x,y)\in S} \Bigg[\frac{1}{2}(J_f\delta)^\top F_R (J_f\delta) + \nabla_z \log r(y|z)^\top F_R{}^{-1} F_R (J_f\delta) \\
&\qquad\qquad + \frac{1}{2}(\nabla_z \log r(y|z))^\top F_R^{-1} F_R F_R^{-1} \nabla_z \log r(y|z) \\
&\qquad\qquad - \frac{1}{2}(\nabla_z \log r(y|z))^\top F_R^{-1} F_R F_R^{-1} \nabla_z \log r(y|z) \Bigg] + h(\theta) \\
&= \frac{1}{|S|} \sum_{(x,y)\in S} \frac{1}{2}(J_f\delta + F_R^{-1}\nabla_z \log r(y|z))^\top F_R (J_f\delta + F_R^{-1}\nabla_z \log r(y|z)) + c \\
&= \frac{1}{|S|} \sum_{(x,y)\in S} \frac{1}{2}\|J_f\delta + F_R^{-1}\nabla_z \log r(y|z)\|_{F_R}^2 + c \,,
\end{aligned}$$

where F_R is the Fisher of the predictive distribution $R_{y|z}$ (as originally defined in Section 9), $\|v\|_{F_R} = \sqrt{v^\top F_R v}$, and $c = h(\theta) - \frac{1}{2}(\sum_{(x,y)\in S} \nabla_z \log r(y|z)^\top F_R^{-1} \nabla_z \log r(y|z))/|S|$ is a constant (independent of δ).

Note that for a given $(x, y) \in S$, $F_R^{-1}\nabla_z \log r(y|z)$ can be interpreted as the natural gradient direction in z-space for an objective corresponding to the KL divergence between the predictive distribution $R_{y|z}$ and a delta distribution on the given y. In other words, it points in the direction which moves $R_{y|z}$ most quickly towards to said delta distribution, as measured by the KL divergence (see Section 6). And assuming that the GGN interpretation of F holds (as discussed in Section 9), we know that it also corresponds to the optimal change in z according to the 2nd-order Taylor series approximation of the loss function $L(y, z)$.

Thus, $M(\delta)$ can be interpreted as the sum of squared distances (as measured using the Fisher metric tensor) between these "optimal" changes in the z's, and the changes in the z's

which result from adding δ to θ, as predicted using 1st-order Taylor-series approximations to f.

In addition to giving us a new interpretation for the natural gradient, this expression also gives us an easy-to-compute bound on the largest possible improvement to h (as predicted by $M(\delta)$). In particular, since the squared error terms are non-negative, we have

$$M(\delta) - h(\theta) \geq -\frac{1}{2|S|} \sum_{(x,y) \in S} \nabla_z \log r(y|z)^\top F_R^{-1} \nabla_z \log r(y|z) \,.$$

Given $F_R = H_L$, this quantity has the simple interpretation of being the optimal improvement in h (as predicted by a 2nd-order order model of $L(y, z)$ for each case in S) achieved in the hypothetical scenario where we can change the z's independently for each case.

The existence of this bound shows that the natural gradient can be meaningfully defined even when F^{-1} may not exist, provided that we compute F and ∇h on the *same data*, and that each F_R is invertible. In particular, it can be defined as the minimizer of $M(\delta)$ that has minimum norm (which must exist since $M(\delta)$ is bounded below), which in practice could be computed by using the pseudo-inverse of F in place of F^{-1}. Other choices are possible, although care would have to be taken to ensure invariance of the choice with respect to parameterization.

14. Asymptotic Convergence Speed

14.1 Amari's Fisher Efficiency Result

A property of natural gradient descent which is frequently referenced in the literature is that it is "Fisher efficient". In particular, Amari (1998) showed that an iteration of the form

$$\theta_{k+1} = \theta_k - \alpha_k \tilde{g}_k(\theta_k) \tag{15}$$

when applied to an objective of the form discussed in Section 4, with α_k shrinking as $1/k$, and with $\tilde{g}_k(\theta_k) = F^{-1} g_k(\theta_k)$ where $g_k(\theta_k)$ is a stochastic estimate of $\nabla h(\theta_k)$ (from a single training case), will produce an estimator θ_k which is asymptotically "Fisher efficient". This means that θ_k will tend to an unbiased estimator of the global optimum θ^* of $h(\theta)$, and that its expected squared error matrix (which tends to its variance) will satisfy

$$\mathrm{E}[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top] = \frac{1}{k} F(\theta^*)^{-1} + \mathcal{O}\left(\frac{1}{k^2}\right) , \tag{16}$$

which is (asymptotically) the smallest[14] possible variance matrix that any unbiased estimator computed from k training cases can have, according to the Cramér-Rao lower bound[15].

14. With the usual definition of $\preceq$ for matrices: $A \preceq C$ iff $C - A$ is PSD.

15. Note that to apply the Cramér-Rao lower bound in this context one must assume that the training data set, on which we compute the objective (and which determines θ^*), is infinitely large, or more precisely that its conditional distribution over y has a density function. For finite training sets one can easily obtain an estimator with exactly zero error for a sufficiently large k (assuming a rich enough model class), and so these requirements are not surprising. If we believe that there is a true underlying distribution of the

This result can also be straightforwardly extended to handle the case where $g_k(\theta_k)$ is computed using a mini-batch of size m (which uses m independently sampled cases at each iteration), in which case the above asymptotic variance bound becomes

$$\frac{1}{mk}F(\theta^*)^{-1} + \mathcal{O}\left(\frac{1}{k^2}\right),$$

which again matches the Cramér-Rao lower bound.

Note that all expectations in this section will be taken with respect to all random variables, both present and historical (i.e. from previous iterations). So for example, $\mathrm{E}[\theta_k]$ is computed by marginalizing over the distribution of g_i for all $i < k$. If "θ" appears inside an expectation without a subscript then it is not an iterate of the optimizer and is instead just treated as fixed non-stochastic value.

This result applies to the version of natural gradient descent where F is computed using the training distribution $\hat{Q}_x$ and the model's conditional distribution $P_{y|x}$ (see Section 5). If we instead consider the version where F is computed using the true data distribution Q_x, then a similar result will still apply, provided that we sample x from Q_x and y from $Q_{y|x}$ when computing the stochastic gradient $g_k(\theta_k)$, and that θ^* is defined as the minimum of the idealized objective $\mathrm{KL}(Q_{x,y}\|P_{x,y}(\theta))$ (see Section 4).

While this Fisher efficiency result would seem to suggest that natural gradient descent is the best possible optimization method in the stochastic setting, it unfortunately comes with several important caveats and conditions, which we will discuss. (Moreover, as we will later, it is also possessed by much simpler methods, and so isn't a great justification for the use of natural gradient descent by itself.)

Firstly, the proof assumes that the iteration in eqn. 15 eventually converges to the global optimum θ^* (at an unspecified speed). While this assumption can be justified when the objective h is convex (provided that α_k is chosen appropriately), it won't be true in general for non-convex objectives, such as those encountered in neural network training. In practice however, a reasonable local optimum θ^* might be a good enough surrogate for the global optimum, in which case a property analogous to Fisher efficiency may still hold, at least approximately.

Secondly, it is assumed in Amari's proof that F is computed using the full training distribution $\hat{Q}_x$, which in the case of neural network optimization usually amounts to an entire pass over the training set S. So while the proof allows for the gradient ∇h to be stochastically estimated from a mini-batch, it doesn't allow this for the Fisher F. This is a serious challenge to the idea that (stochastic) natural gradient descent gives an estimator which makes optimal use of the training data that it sees. And note that while one can

data that has a density function, and from which the training set is just a finite collection of samples, then Cramér-Rao can be thought of as applying to the problem of estimating the true parameters of this distribution from said samples (with F computed using the true distribution), and will accurately bound the rate of convergence to the true parameters until we start to see samples repeat. After that point, convergence to the true parameters will slow down and eventually stop, while convergence on the training objective may start to beat the bound. This of course implies that any convergence on the training set that happens faster than the Cramér-Rao bound will necessarily correspond to over-fitting (since this faster convergence cannot happen for the test loss).

approximate F using minibatches from S, which is a solution that often works well in practice (especially when combined with a decayed-averaging scheme[16]), a Fisher efficiency result like the one proved by Amari (1998) will likely no longer hold. Investigating the manner and degree in which it may hold *approximately* when F is estimated in this way is an interesting direction for future research.

A third issue with Amari's result is that it is given in terms of the convergence of θ_k (as measured by the Euclidean norm) instead of the objective function value, which is arguably much more relevant. Fortunately, it is straightforward to obtain the former from the latter. In particular, by applying Taylor's theorem and using $\nabla h(\theta^*) = 0$ we have

$$\begin{aligned} h(\theta_k) - h(\theta^*) &= \frac{1}{2}(\theta_k - \theta^*)^\top H^*(\theta_k - \theta^*) + \nabla h(\theta^*)^\top(\theta_k - \theta^*) + \mathcal{O}\left((\theta_k - \theta^*)^3\right) \\ &= \frac{1}{2}(\theta_k - \theta^*)^\top H^*(\theta_k - \theta^*) + \mathcal{O}\left((\theta_k - \theta^*)^3\right) , \end{aligned} \tag{17}$$

where $H^* = H(\theta^*)$ and $\mathcal{O}\left((\theta_k - \theta^*)^3\right)$ is short-hand to mean a function which is cubic in the entries of $\theta_k - \theta^*$. From this it follows[17] that

$$\begin{aligned} \mathrm{E}[h(\theta_k)] - h(\theta^*) &= \frac{1}{2}\,\mathrm{E}\left[(\theta_k - \theta^*)^\top H^*(\theta_k - \theta^*)\right] + \mathrm{E}\left[\mathcal{O}\left((\theta_k - \theta^*)^3\right)\right] \\ &= \frac{1}{2}\,\mathrm{tr}\left(H^*\,\mathrm{E}\left[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top\right]\right) + \mathrm{E}\left[\mathcal{O}\left((\theta_k - \theta^*)^3\right)\right] \\ &= \frac{1}{2k}\,\mathrm{tr}\left(H^* F(\theta^*)^{-1}\right) + \mathrm{E}\left[\mathcal{O}\left((\theta_k - \theta^*)^3\right)\right] \\ &= \frac{n}{2k} + o\left(\frac{1}{k}\right) , \end{aligned} \tag{18}$$

where we have used $H^* = F(\theta^*)$, which follows from the "realizability" hypothesis used to prove the Fisher efficiency result (see below). Note that while this is the same convergence rate ($\mathcal{O}(1/k)$) as the one which appears in Hazan et al. (2007) (see our Section 11), the constant is much better. However, the comparison is slightly unfair, as Hazan et al. (2007) doesn't require that the curvature matrix be estimated on the entire data set (as discussed above).

The fourth and final caveat of Amari's Fisher efficiency result is that Amari's proof assumes that the training distribution $\hat{Q}_{x,y}$ and the optimal model distribution $P_{x,y}(\theta^*)$ coincide, a condition called "realizability" (which is also required in order for the Cramér-Rao lower bound to apply). This essentially means that the model perfectly captures the

16. By this we mean a scheme which maintains an estimate where past contributions decay exponentially at some fixed rate. In other words, we estimate F at each iteration as $(1-\beta)F_{\text{new}} + \beta F_{\text{old}}$ for some $0 < \beta < 1$ where F_{new} is the Fisher as computed on the current mini-batch (for the current setting of θ), and F_{old} is the old estimate (which will be based on stale θ values).

17. The last line of this derivation uses $\mathrm{E}\left[\mathcal{O}\left((\theta_k - \theta^*)^3\right)\right] = o(1/k)$, which is an (unjustified) assumption that is used in Amari's proof. This assumption has intuitive appeal since $\mathrm{E}\left[\mathcal{O}\left((\theta_k - \theta^*)^2\right)\right] = \mathcal{O}(1/k)$, and so it makes sense that $\mathrm{E}\left[\mathcal{O}\left((\theta_k - \theta^*)^3\right)\right]$ would shrink faster. However, extreme counterexamples are possible which involve very heavy-tailed distributions on θ_k over unbounded regions. By adding some mild hypotheses such as θ_k being restricted to some bounded region, which is an assumption frequently used in the convex optimization literature, it is possible to justify this assumption rigorously. Rather than linger on this issue we will refer the reader to Bottou and LeCun (2005), which provides a more rigorous treatment of these kinds of asymptotic results, using various generalizations of the big-O notation.

training distribution at $\theta = \theta^*$. This assumption is used in Amari's proof of the Fisher efficiency result to show that the Fisher F, when evaluated at $\theta = \theta^*$, is equal to both the empirical Fisher $\bar{F}$ and the Hessian H of h. (These equalities follow immediately from $\hat{Q}_{x,y} = P_{x,y}(\theta^*)$ using the forms of the Fisher presented in Section 5.) Note that realizability is a subtle condition. It can fail to hold if the model isn't powerful enough to capture the training distribution. But also if the training distribution is a finite set of pairs (x, y) and the model is powerful enough to perfectly capture this (as a Delta distribution), in which case $F(\theta^*)$ is no longer well-defined (because its associated density function isn't), and convergence faster than the Cramér-Rao bound becomes possible.

It is not clear from Amari's proof what happens when this correspondence fails to hold at $\theta = \theta^*$, and whether a (perhaps) weaker asymptotic upper bound on the variance might still be provable. Fortunately, various authors (Murata, 1998; Bottou and LeCun, 2005; Bordes et al., 2009) building on early work of Amari (1967), provide some further insight into this question by studying asymptotic behavior of general iterations of the form[18]

$$\theta_{k+1} = \theta_k - \alpha_k B_k^{-1} g_k(\theta_k)\,, \tag{19}$$

where $B_k = B$ is a fixed[19] curvature matrix (which is independent of θ_k and k), and where $g_k(\theta_k)$ is a stochastic estimate of $\nabla h(\theta_k)$.

In particular, Murata (1998) gives exact (although implicit) expressions for the asymptotic mean and variance of θ_k in the above iteration for the case where $\alpha_k = 1/(k+1)$ or α_k is constant. These expressions describe the (asymptotic) behavior of this iteration in cases where the curvature matrix B is not the Hessian H or the Fisher F, covering the non-realizable case, as well as the case where the curvature matrix is only an approximation of the Hessian or Fisher. Bordes et al. (2009) meanwhile gives expressions for $\mathrm{E}[h(\theta_k)]$ in the case where α_k shrinks as $1/k$, thus generalizing eqn. 18 in a similar manner.

In the following subsections we will examine these results in more depth, and improve on those of Bordes et al. (2009) (at least in the *quadratic* case) by giving an *exact* asymptotic expression for $\mathrm{E}[h(\theta_k)]$. We will also analyze iterate averaging (aka Polyak averaging ; see Section 14.3) in the same setting.

Some interesting consequences of this analysis are discussed in Sections 14.2.1 and 14.3.1. Of particular note is that for any choice of B, $\mathrm{E}[h(\theta_k)] - h(\theta_0)$ can be expressed as a sum of two terms: one that scales as $\mathcal{O}(1/k)$ and doesn't depend on the starting point θ_0, and one that does depend on the starting point and scales as $\mathcal{O}(1/k^2)$ or better. Moreover, the first term, which is asymptotically dominant, carries all the dependence on the noise covariance, and crucially isn't improved by the use of a non-trivial choices for B such as F or H, assuming the use of Polyak averaging. Indeed, if Polyak averaging is used, this term matches the Cramér-Rao lower bound, and thus even plain stochastic gradient descent becomes Fisher efficient! (This also follows from the analysis of Polyak and Juditsky (1992).) Meanwhile, if learning rate decay is used instead of Polyak averaging, one *can* improve the constant on this term by using 2nd-order methods, although not the overall $1/k$ rate.

18. Note that some authors define B_k to be the matrix that multiplies the gradient, instead of its inverse (as we do instead).

19. Note that for a non-constant B_k where B_k^{-1} converges sufficiently quickly to a fixed B^{-1} as θ_k converges to θ^*, these analyses will likely still apply, at least approximately.

While these results strongly suggest that 2nd-order methods like natural gradient descent won't be of much help *asymptotically* in the stochastic setting, we argue that the constant on the starting point dependent $\mathcal{O}(1/k^2)$ term can still be improved significantly through the use of such methods, and this term may matter more in practice given a limited iteration budget (despite being negligible for very large k).

14.2 Some New Results Concerning Asymptotic Convergence Speed of General Stochastic 2nd-order Methods

In this subsection we will give two results which characterize the asymptotic convergence of the stochastic iteration in eqn. 19 as applied to the convex quadratic objective $h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^*(\theta - \theta^*)$ (whose minimizer is θ^*). The proofs of both results are in Appendix B.

We begin by defining

$$\Sigma_g(\theta) = \mathrm{Var}(g(\theta)) = \mathrm{E}\left[(g(\theta) - \mathrm{E}[g(\theta)])(g(\theta) - \mathrm{E}[g(\theta)])^\top\right] ,$$

where $g(\theta)$ denotes the random variable whose distribution coincides with the conditional distribution of $g_k(\theta_k)$ given θ_k. Note that this notation is well-defined as long as $g_k(\theta_k)$ depends only on the value of θ_k, and not on k itself. (This will be true, for example, if the $g_k(\theta_k)$'s are generated by sampling a fixed-size mini-batch of iid training data.)

To simplify our analysis we will assume that $\Sigma_g(\theta)$ is *constant* with respect to θ, allowing us to write it simply as Σ_g. While somewhat unrealistic, one can reasonably argue that this assumption will become approximately true as θ converges to the optimum θ^*. It should be noted that convergence of stochastic optimization methods can happen faster than in our analysis, and indeed than is allowed by the Cramér-Rao lower bound, if $\Sigma_g(\theta)$ approaches 0 sufficiently fast as θ goes to θ^* . This can happen if the model can obtain zero error on all cases in the training distribution (e.g Loizou and Richtárik, 2017), a situation which is ruled out by the hypothesis that this distribution has a density function (as is required by Cramér-Rao). But it's worth observing that this kind of faster convergence can't happen on the test data distribution (assuming it satisfies Cramér-Rao's hypotheses), and thus will only correspond to faster over-fitting.

Before stating our first result we will define some additional notation. We denote the variance of θ_k by

$$V_k = \mathrm{Var}(\theta_k) = \mathrm{E}\left[(\theta_k - \mathrm{E}[\theta_k])(\theta_k - \mathrm{E}[\theta_k])^\top\right] .$$

And we define the following linear operators[20] that map square matrices to matrices of the same size:

$$\Xi(X) = B^{-1}H^*X + \left(B^{-1}H^*X\right)^\top = B^{-1}H^*X + XH^*B^{-1} \text{, and}$$
$$\Psi_\beta(X) = \left(I - \beta B^{-1}H^*\right) X \left(I - \beta B^{-1}H^*\right)^\top .$$

20. Note that these operators are *not* $n \times n$ matrices themselves, although they can be represented as $n^2 \times n^2$ matrices if we vectorize their $n \times n$ matrix arguments. Also note that such operators can be linearly combined and composed, where we will use the standard $\pm$ notation for linear combination, multiplication for composition, and where I will be the identity operator. So, for example, $(I + \Xi^2)(X) = X + \Xi(\Xi(X))$.

We also define

$$U = B^{-1}\Sigma_g B^{-1}$$

for notational brevity, as this is an expression which will appear frequently. Finally, for an n-dimensional symmetric matrix A we will denote its i-th largest eigenvalue by $\lambda_i(A)$, so that $\lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A)$.

The following theorem represents a more detailed and rigorous treatment of the type of asymptotic expressions for the mean and variance of θ_k given by Murata (1998), although specialized to the quadratic case. Note that symbols like V_∞ have a slightly different meaning here than in Murata (1998).

Theorem 3. *Suppose that θ_k is generated by the stochastic iteration in eqn. 19 while optimizing a quadratic objective $h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^*(\theta - \theta^*)$.*

If α_k is equal to a constant α satisfying $\alpha\lambda_1(B^{-1}H^*) \leq 1$, then the mean and variance of θ_k are given by

$$\mathrm{E}[\theta_k] = \theta^* + (I - \alpha B^{-1}H^*)^k(\theta_0 - \theta^*)$$
$$V_k = \left(I - \Lambda^k\right)(V_\infty)\,,$$

where $\Lambda = \Psi_\alpha$ and $V_\infty = \alpha^2\left(I - \Lambda\right)^{-1}(U)$.

If on the other hand we have $\alpha_k = 1/(k+a+1)$ for some $a \geq 1$, with $b \equiv \lambda_n\left(B^{-1}H^*\right) > \frac{1}{2}$ and $\lambda_1\left(B^{-1}H^*\right) \leq a+1$, then the mean and variance of θ_k are given by

$$\mathrm{E}[\theta_k] = \theta^* + \prod_{j=0}^{k-1}\left(I - \alpha_j B^{-1}H^*\right)(\theta_0 - \theta^*)$$
$$V_k = \frac{1}{k+a}\left(\Xi - I\right)^{-1}(U) + E_k\,,$$

where E_k is a matrix valued "error" that shrinks as $1/k^2$.

Remark 4. *Due to the properties of quadratic functions, and the assumptions of constant values for B and Σ_g, one could state and/or prove this theorem (and the ones that follow) while taking one of H^*, B, or Σ_g to be the identity matrix, without any loss of generality. This is due to the fact that optimizing with a preconditioner is equivalent to optimizing a linearly-reparameterized version of the objective with plain stochastic gradient descent.*

One interesting observation that we can immediately make from Theorem 3 is that, at least in the case where the objective is a convex quadratic, $\mathrm{E}[\theta_k]$ progresses in a way that is fully independent of the distribution of noise in the gradient estimate (which is captured by the Σ_g matrix). Indeed, it proceeds as θ_k itself would in the case of fully deterministic optimization. It is only the variance of θ_k around $\mathrm{E}[\theta_k]$ that depends on the gradient estimator's noise.

To see why this happens, note that if $h(\theta)$ is quadratic then $\nabla h(\theta)$ will be an affine function, and thus will commute with expectation. This allows us to write

$$\mathrm{E}[g(\theta_k)] = \mathrm{E}[\nabla h(\theta_k)] = \nabla h(\mathrm{E}[\theta_k])\,.$$

Provided that α_k doesn't depend on θ_k in any way (as we are implicitly assuming), we then have

$$\mathrm{E}[\theta_{k+1}] = \mathrm{E}[\theta_k - \alpha_k B^{-1} g(\theta_k)] = \mathrm{E}[\theta_k] - \alpha_k B^{-1} \nabla h(\mathrm{E}[\theta_k])\,,$$

which is precisely the deterministic version of eqn. 19, where we treat $\mathrm{E}[\theta_k]$ as the parameter vector being optimized].

While Theorem 3 provides a detailed picture of how well θ^* is estimated by θ_k, it doesn't tell us anything directly about how quickly progress is being made on the objective, which is arguably a much more relevant concern in practice. Fortunately, as observed by Murata (1998), we have the basic identity (proved for completeness in Appendix A):

$$\mathrm{E}\left[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top\right] = V_k + (\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\,.$$

And it thus follows that

$$\begin{aligned}
\mathrm{E}[h(\theta_k)] - h(\theta^*) &= \frac{1}{2}\,\mathrm{tr}\left(H^*\,\mathrm{E}\left[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top\right]\right) \\
&= \frac{1}{2}\,\mathrm{tr}\left(H^*\left(V_k + (\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right)\right) \\
&= \frac{1}{2}\,\mathrm{tr}\left(H^* V_k\right) + \frac{1}{2}\,\mathrm{tr}\left(H^*(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right) \qquad (20)
\end{aligned}$$

which allows us to relate the convergence of $E[\theta_k]$ (which behaves like θ_k in the deterministic version of the algorithm) and the size/shape of the variance of θ_k to the convergence of $\mathrm{E}[h(\theta_k)]$. In particular, we see that in this simple case where $h(\theta)$ is quadratic, $\mathrm{E}[h(\theta_k)] - h(\theta^*)$ neatly decomposes as the sum of two independent terms that quantify the roles of these respective factors in the convergence of $\mathrm{E}[h(\theta_k)]$ to $h(\theta^*)$.

In the proof of the following theorem (which is in the appendix), we will use the above expression and Theorem 3 to precisely characterize the asymptotic convergence of $\mathrm{E}[h(\theta_k)]$. Note that while Murata (1998) gives expressions for this as well, they cannot be directly evaluated except in certain special cases (such as when $B = H$), and only include the asymptotically dominant terms.

Theorem 5. *Suppose that θ_k is generated by the stochastic iteration in eqn. 19 while optimizing a quadratic objective $h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^*(\theta - \theta^*)$.*

If α_k is equal to a constant α satisfying $\alpha\lambda_1(B^{-1}H^*) \leq 1$, then the expected objective $\mathrm{E}[h(\theta_k)]$ satisfies

$$l(k) \leq \mathrm{E}[h(\theta_k)] - h(\theta^*) \leq u(k)\,,$$

where

$$u(k) = \left[1 - (1 - \epsilon_1)^{2k}\right]\frac{\alpha}{4}\,\mathrm{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right) + (1 - \epsilon_2)^{2k}\,h(\theta_0)$$

and

$$l(k) = \left[1 - (1 - \epsilon_2)^{2k}\right]\frac{\alpha}{4}\,\mathrm{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right) + (1 - \epsilon_1)^{2k}\,h(\theta_0)\,,$$

with $\epsilon_1 = \lambda_1(C) = \alpha\lambda_1\left(B^{-1}H^*\right)$ *and* $\epsilon_2 = \lambda_n(C) = \alpha\lambda_n\left(B^{-1}H^*\right)$.

If on the other hand we have $\alpha_k = 1/(k+a+1)$ *for some* $a \geq 1$, *with* $b \equiv \lambda_n\left(B^{-1}H^*\right) > \frac{1}{2}$ *and* $\lambda_1\left(B^{-1}H^*\right) \leq a+1$, *then the expected objective satisfies*

$$l(k) \leq \mathrm{E}[h(\theta_k)] - h(\theta^*) \leq u(k)\,,$$

where

$$u(k) = \frac{1}{4(k+a)}\operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1}U\right) + \frac{\nu(a)k}{4(k+a)^3}\operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1}\Psi_1(U)\right) + h(\theta_0)\left(\frac{1+a}{k+a}\right)^{2b}$$

and

$$l(k) = \frac{1}{4(k+a)}\operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1}U\right) - \frac{1}{4a}\left(\frac{1+a}{k+a}\right)^{2b}\operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1}U\right)\,,$$

where $\nu(a) = (a+2)^3/(a(a+1)^2)$.

14.2.1 Some Consequences of Theorem 5

In the case of a fixed step-size $\alpha_k = \alpha$, Theorem 5 shows that $\mathrm{E}[h(\theta_k)]$ will tend to the constant

$$h(\theta^*) + \frac{\alpha}{4}\operatorname{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right)\,.$$

The size of this extra additive factor is correlated with the step-size α and gradient noise covariance Σ_g. If the covariance or step-sizes are small enough, it may not be very large in practice.

Moreover, one can use the fact that the iterates $\{\theta_k\}_{k=1}^{\infty}$ are (non-independent) asymptotically unbiased estimators of θ^* to produce an asymptotically unbiased estimator with shrinking variance by averaging them together. This is done in the Polyak Averaging method (e.g. Polyak and Juditsky, 1992), which we analyze in Section 14.3.

In the scenario where $\alpha_k = 1/(k+a+1)$, if one performs stochastic 2nd-order optimization with $B = H^*$ (so that $b = 1$) and any $a \geq 1$, Theorem 5 gives that

$$\mathrm{E}[h(\theta_k)] - h(\theta^*) \leq \frac{1}{2(k+a)}\operatorname{tr}\left(H^{*-1}\Sigma_g\right) + h(\theta_0)\left(\frac{1+a}{k+a}\right)^2$$

(where we have used the fact that $\Psi_1 = 0$ when $B = H^*$). And if one considers the scenario corresponding to 1st-order optimization where we take $B = \lambda_n(H^*)I$ (so that $b = 1$) and $a = \kappa(H^*)$, where $\kappa(H^*) = \lambda_1(H^*)/\lambda_n(H^*)$ is the condition number of H^*, we get

$$\mathrm{E}[h(\theta_k)] - h(\theta^*) \leq \left(\frac{1}{4(k+\kappa(H^*))} + \mathcal{O}\left(\frac{1}{(k+\kappa(H^*))^2}\right)\right)\frac{1}{\lambda_n(H^*)} \cdot \operatorname{tr}\left(\left(I - \frac{\lambda_n(H^*)}{2}H^{*-1}\right)^{-1}\Sigma_g\right) + h(\theta_0)\left(\frac{1+\kappa(H^*)}{k+\kappa(H^*)}\right)^2\,.$$

In deriving this we have applied Lemma 20 while exploiting the fact that $\Psi_1(U) \preceq U$ (i.e. $U - \Psi_1(U)$ is PSD).

These two bounds can be made more similar looking by choosing $a = \kappa(H^*)$ in the first one. (However this is unlikely to be the optimal choice in general, and the extra freedom in choosing a seems to be one of the advantages of using $B = H^*$.) In this case, the starting-point dependent terms (which are noise independent) seem to exhibit the same asymptotics, although this is just an artifact of the analysis, and it is possible to obtain tighter bounds on these terms by considering the entire spectrum instead of just the most extreme eigenvalue (as was done in eqn. 37 while applying Lemma 18).

The noise-dependent terms, which are the ones that dominate asymptotically as $k \to \infty$ and were derived using a very tight analysis (and indeed we have a matching lower bound for these), exhibit a more obvious difference in the above expressions. To compare their sizes we can apply Lemma 18 to obtain the following bounds (see Appendix C):

$$\frac{1}{2\lambda_1\left(H^*\right)} \operatorname{tr}(\Sigma_g) \leq \frac{1}{2} \operatorname{tr}\left(H^{*-1}\Sigma_g\right) \leq \frac{1}{2\lambda_n\left(H^*\right)} \operatorname{tr}(\Sigma_g)$$

and

$$\frac{1}{4\lambda_n(H^*)} \operatorname{tr}(\Sigma_g) \leq \frac{1}{4\lambda_n(H^*)} \operatorname{tr}\left(\left(I - \frac{\lambda_n(H^*)}{2} H^{*-1}\right)^{-1} \Sigma_g\right) \leq \frac{1}{2\lambda_n(H^*)} \operatorname{tr}(\Sigma_g)\,.$$

Because the lower bound is much smaller in the $B = H^*$ case, these bounds thus allow for the possibility that the noise dependent term will be much smaller in that case. A necessary condition for this to happen is that H^* is ill-conditioned (so that $\lambda_1\left(H^*\right) \gg \lambda_n\left(H^*\right)$), although this alone is not sufficient.

To provide an actual concrete example where the noise-dependent term is smaller, we must make further assumptions about the nature of the gradient noise covariance matrix Σ_g. As an important example, we consider the scenario where the stochastic gradients are computed using (single) randomly sampled cases from the training set S, and where we are in the realizable regime (so that $H^* = \bar{F}^* = \Sigma_g$; see Section 14.1). When $B = H^*$, the constant on the noise dependent term will thus scale as

$$\frac{1}{2} \operatorname{tr}\left(H^{*-1}\Sigma_g\right) = \frac{1}{2} \operatorname{tr}\left(H^{*-1}H^*\right) = \frac{n}{2}\,,$$

while if $B = \lambda_n(H^*)I$ it will scale as

$$\begin{aligned}\frac{1}{4\lambda_n(H^*)} \operatorname{tr}\left(\left(I - \frac{\lambda_n(H^*)}{2} H^{*-1}\right)^{-1} \Sigma_g\right) &= \frac{1}{4\lambda_n(H^*)} \operatorname{tr}\left(\left(I - \frac{\lambda_n(H^*)}{2} H^{*-1}\right)^{-1} H^*\right) \\ &= \frac{1}{4\lambda_n(H^*)} \sum_{i=1}^{n} \frac{\lambda_i(H^*)}{1 - \frac{\lambda_n(H^*)}{2\lambda_i(H^*)}} = \frac{1}{4} \sum_{i=1}^{n} \frac{r_i}{1 - \frac{1}{2r_i}}\,,\end{aligned}$$

where we have defined $r_i = \lambda_i(H^*)/\lambda_n(H^*)$. (To go from the first to the second line we have used the general fact that for any rational function g the i-th eigenvalue of $g(H^*)$ is given by $g(\lambda_i)$. And also that the trace is the sum of the eigenvalues.)

Observing that $1 \leq r_i$, we thus have $r_i \leq r_i/(1 - 1/(2r_i))$, from which it also follows that

$$\frac{1}{4}\frac{\operatorname{tr}(H^*)}{\lambda_n(H^*)} \;=\; \frac{1}{4}\sum_{i=1}^{n} r_i \;\leq\; \frac{1}{4}\sum_{i=1}^{n} \frac{r_i}{1-\frac{1}{2r_i}}\,.$$

and

$$\frac{n}{2} \;\leq\; 2\cdot\frac{1}{4}\sum_{i=1}^{n} \frac{r_i}{1-\frac{1}{2r_i}}\,.$$

From these bounds we can see that noise scaling in the $B = H^*$ case is no worse than twice that of the $B = \lambda_n(H^*)I$ case. And it has the potential to be much smaller, such as when $\frac{\operatorname{tr}(H^*)}{\lambda_n(H^*)} \gg n$, or when the spectrum of H^* covers a large range. For example, if $\lambda_i(H^*) = n - i + 1$ then the noise scales as $\Omega(n^2/k)$ in the $B = \lambda_n(H^*)I$ case.

14.2.2 Related Results

The related result most directly comparable to Theorem 5 is Theorem 1 of Bordes et al. (2009), which provides upper and lower bounds for $\mathrm{E}[h(\theta_k)] - h(\theta^*)$ in the case where $\alpha_k = 1/(k + a + 1)$ and $\lambda_n\left(B^{-1}H^*\right) > 1/2$. In particular, using a different technique from our own, Bordes et al. (2009) show that[21]

$$\frac{1}{k}\cdot\frac{\operatorname{tr}(H^*U)}{4\left(\lambda_1(B^{-1}H^*)-\frac{1}{2}\right)} + o\left(\frac{h(\theta_0)}{k}\right) \;\leq\; \mathrm{E}[h(\theta_k)] - h(\theta^*) \;\leq\; \frac{1}{k}\cdot\frac{\operatorname{tr}(H^*U)}{4\left(\lambda_n(B^{-1}H^*)-\frac{1}{2}\right)} + o\left(\frac{h(\theta_0)}{k}\right).$$

This result is more general in the sense that it doesn't assume the objective is quadratic. However, it is also far less detailed than our result, and in particular doesn't describe the asymptotic value of $\mathrm{E}[h(\theta_k)] - h(\theta^*)$, instead only giving (fairly loose) upper and lower bounds on it. It can be obtained from our Theorem 5, in the quadratic case, by a straightforward application of Lemma 18.

There are other relevant results in the vast literature on general strongly convex functions, such as Kushner and Yin (2003) and the references therein, and Moulines and Bach (2011). These results, while usually only presented for standard stochastic gradient descent, can be applied to the same setting considered in Theorem 5 by performing a simple linear reparameterization. A comprehensive review of such results would be outside the scope of this report, but to the best of our knowledge there is no result which would totally subsume Theorem 5 for convex quadratics. The bounds in Moulines and Bach (2011) for example, are more general in a number of ways, but also appear to be less detailed than ours, and harder to interpret.

21. Note that the notation 'B' as it is used by Bordes et al. (2009) means the *inverse* of the matrix B as it appears in this paper. And while Bordes et al. (2009) presents their bounds with $\bar{F}$ in place of Σ_g, these are the same matrix when evaluated at $\theta = \theta^*$ as we have $\mathrm{E}[g(\theta^*)] = 0$ (since θ^* is a local optimum).

14.3 An Analysis of Averaging

In this subsection we will extend the analysis from Subsection 14.2 to incorporate basic iterate averaging of the standard type (e.g. Polyak and Juditsky, 1992). In particular, we will bound $\mathrm{E}[h(\bar{\theta}_k)]$ where

$$\bar{\theta}_k = \frac{1}{k+1} \sum_{i=0}^{k} \theta_i \,.$$

Note that while this type of averaging leads to elegant bounds (as we will see), a form of averaging based on an exponentially-decayed moving average typically works much better in practice. This is given by

$$\bar{\theta}_k = (1 - \beta_k)\theta_k + \beta_k \bar{\theta}_{k-1} \qquad\qquad \bar{\theta}_0 = \theta_0$$

for $\beta_k = \min\{1 - 1/k, \beta_{\max}\}$ with $0 < \beta_{\max} < 1$ close to 1 (e.g. $\beta_{\max} = 0.99$). This type of averaging has the advantage that it more quickly "forgets" the very early θ_i's, since their "weight" in the average decays exponentially. However, the cost of doing this is that the variance will never converge exactly to zero, which arguably matters more in theory than it does in practice.

The main result of this subsection, which is proved in Appendix B, is stated as follows:

Theorem 6. *Suppose that θ_k is generated by the stochastic iteration in eqn. 19 with constant step-size $\alpha_k = \alpha$ while optimizing a quadratic objective $h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^* (\theta - \theta^*)$. Further, suppose that $\alpha \lambda_1(B^{-1}H^*) < 1$, and define $\bar{\theta}_k = \frac{1}{k+1}\sum_{i=0}^{k} \theta_i$. Then we have the following bound:*

$$\begin{aligned}\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \min &\left\{ \frac{1}{2(k+1)} \operatorname{tr}\left(H^{*-1}\Sigma_g\right),\ \frac{\alpha}{4} \operatorname{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right)\right\} \\ &+ \min\left\{ \frac{1}{2(k+1)^2\alpha^2} \left\| H^{*-1/2} B(\theta_0 - \theta^*)\right\|^2, \right. \\ &\qquad \left. \frac{1}{2(k+1)\alpha} \left\| B^{1/2}(\theta_0 - \theta^*)\right\|^2,\ h(\theta_0)\right\} .\end{aligned}$$

Note that this bound can be written asymptotically ($k \to \infty$) as

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \mathcal{O}\left(\frac{1}{2(k+1)} \operatorname{tr}\left(H^{*-1}\Sigma_g\right)\right) ,$$

which notably doesn't depend on either α or B.

This is a somewhat surprising property of averaging to be sure, and can be intuitively explained as follows. Increasing the step-size along any direction d (as measured by $\alpha d^\top B^{-1} d$) will increase the variance in that direction for each iterate (since the step-size multiplies the noise in the stochastic gradient estimate), but will also cause the iterates to decorrelate faster in that direction (as can be seen from eqn. 39). Increased decorrelation in the iterates leads to lower variance in their average, which counteracts the aforementioned increase in the variance. As it turns out, these competing effects will exactly cancel in the limit, which the proof of Theorem 6 rigorously establishes.

14.3.1 Some Consequences of Theorem 6

In the case of stochastic 2nd-order optimization of a convex quadratic objective where we take $B = H^*$ (which allows us to use an α close to 1) this gives

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \frac{\mathrm{tr}\left(H^{*-1}\Sigma_g\right)}{2(k+1)} + \frac{h(\theta_0)}{(k+1)^2\alpha^2}\,.$$

Then choosing the maximum allowable value of α this becomes

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \frac{\mathrm{tr}\left(H^{*-1}\Sigma_g\right)}{2(k+1)} + \frac{h(\theta_0)}{(k+1)^2}\,,$$

which is a similar bound to the one described in Section 14.2.1 for stochastic 2nd-order optimization (with $B = H^*$) using an annealed step-size $\alpha_k = 1/(k+1)$.

For the sake of comparison, applying Theorem 6 with $B = I$ gives

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \frac{\mathrm{tr}\left(H^{*-1}\Sigma_g\right)}{2(k+1)} + \frac{\left\|H^{*-1/2}(\theta_0 - \theta^*)\right\|^2}{2(k+1)^2\alpha^2} \tag{21}$$

under the assumption that $\alpha\lambda_1(H^*) < 1$. For the maximum allowable value of α this becomes

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \frac{\mathrm{tr}\left(H^{*-1}\Sigma_g\right)}{2(k+1)} + \frac{\lambda_1(H^*)^2\left\|H^{*-1/2}(\theta_0 - \theta^*)\right\|^2}{2(k+1)^2}\,.$$

An interesting observation we can make about these bounds is that they *do not* demonstrate any improvement through the use of 2nd-order optimization on the *asymptotically dominant* noise-dependent term in the bound (a phenomenon first observed by Polyak and Juditsky (1992) in a more general setting). Moreover, in the case where the stochastic gradients (the $g_k(\theta_k)$'s) are sampled using random training cases in the usual way so that $\Sigma_g = \bar{F}(\theta)$, and the realizability hypothesis is satisfied so that $H^* = F(\theta^*) = \bar{F}(\theta^*)$ (see Section 14.1), we can see that simple stochastic gradient descent with averaging achieves a similar *asymptotic* convergence speed (given by $n/(k+1) + o(1/k)$) to that possessed by Fisher efficient methods like stochastic natural gradient descent (c.f. eqn. 18), despite not involving the use of curvature matrices!

Moreover, in the non-realizable case, $1/(2k)\,\mathrm{tr}\left(H^{*-1}\Sigma_g\right)$ turns out to be the same asymptotic rate as that achieved by the "empirical risk minimizer" (i.e. the estimator of θ that minimizes the expected loss over the training cases processed thus far) and is thus "optimal" in a certain strong sense for infinite data sets. See Frostig et al. (2014) for a good recent discussion of this.

However, despite these observations, these bounds *do* demonstrate an improvement to the noise-independent term (which depends on the starting point θ_0) through the use of 2nd-order optimization. When H^* is ill-conditioned and $\theta_0 - \theta^*$ has a large component in the direction of eigenvectors of H^* with small eigenvalues, we will have

$$\lambda_1(H^*)^2\|H^{*-1/2}(\theta_0 - \theta^*)\|^2 \gg h(\theta_0)\,.$$

Crucially, this noise-independent term may often matter more in practice (despite being *asymptotically* negligible), as the LHS expression may be very large compared to Σ_g, and we may be interested in stopping the optimization long before the more slowly shrinking noise-dependent term begins to dominate asymptotically (e.g. if we have a fixed iteration budget, or are employing early-stopping). This is especially likely to be the case if the gradient noise is low due to the use of large mini-batches.

It is also worth pointing out that compared to standard stochastic 2nd-order optimization with a fixed step-size (as considered by the first part of Theorem 5), the noise-independent term shrinks much more slowly when we use averaging (quadratically vs exponentially), or for that matter when we use an annealed step-size $\alpha_k = 1/(k+1)$ with $b > 1$. This seems to be the price one has to pay in order to ensure that the noise-dependent term shrinks as $1/k$. (Although in practice one can potentially obtain a more favorable dependence on the starting point by adopting the "forgetful" exponentially-decaying variant of averaging discussed previously.)

14.3.2 Related Results

Under weaker assumptions about the nature of the stochastic gradient noise, Polyak and Juditsky (1992) showed that

$$\mathrm{E}\left[(\bar{\theta}_k - \theta^*)(\bar{\theta}_k - \theta^*)^\top\right] = \frac{1}{k+1} H^{*-1/2} \Sigma_g H^{*-1/2} + o\left(\frac{1}{k}\right),$$

which using the first line of eqn. 20 yields

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) = \frac{\mathrm{tr}\left(H^{*-1}\Sigma_g\right)}{2(k+1)} + o\left(\frac{1}{k}\right).$$

While consistent with Theorem 6, this bound gives a less detailed picture of convergence, and in particular fails to quantify the relative contribution of the noise-dependent and independent terms, and thus doesn't properly distinguish between the behavior of stochastic 1st or 2nd-order optimization methods (i.e. $B = I$ vs $B = H^*$).

Assuming a model for the gradient noise which is consistent with linear least-squares regression and $B = I$, Défossez and Bach (2014) showed that

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \approx \frac{\mathrm{tr}\left(H^{*-1}\Sigma_g\right)}{k+1} + \frac{\left\|H^{*-1/2}(\theta_0 - \theta^*)\right\|^2}{(k+1)^2\alpha^2}$$

holds in the asymptotic limit as $\alpha \to 0$ and $k \to \infty$.

This expression is similar to the one generated by Theorem 6 (see eqn. 21), although it only holds in the asymptotic limit of small α and large k, and assumes a particular θ-dependent form for the noise (arising in least-squares linear regression), which represents neither a subset nor a super-set of our general θ-independent formulation. An interesting question for future research is whether Theorem 3 could be extended in way that would allow Σ_g to vary with θ, and whether this would allow us to prove a more general version of Theorem 6 that would cover the case of linear least-squares.

A result which is more directly comparable to our Theorem 6 is "Theorem 3" of Flammarion and Bach (2015), which when applied to the same general case considered in Theorem 6 gives the following upper bound (assuming that $B = I$[22] and $\alpha\lambda_1(H^*) \leq 1$):

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq 4\alpha \operatorname{tr}(\Sigma_g) + \frac{\|\theta_0 - \theta^*\|^2}{(k+1)\alpha}\,.$$

Unlike the bound proved in Theorem 6, this bound fails to establish that $\mathrm{E}[h(\bar{\theta}_k)]$ even converges, since the term $4\alpha \operatorname{tr}(\Sigma_g)$ is constant in k.

There are other older results in the literature analyzing averaging in general settings, such as those contain in Kushner and Yin (2003) and the references therein. However, to the best of our knowledge there is no result in the literature which would totally subsume Theorem 6 for the case of convex quadratics, particularly with regard to the level of detail in the non-asymptotically-dominant terms of the bound (which is very important for our purposes here).

15. Conclusions and Open Questions

In this report we have examined several aspects of the natural gradient method, such as its relationship to 2nd-order methods, its local convergence speed, and its invariance properties.

The link we have established between natural gradient descent and (stochastic) 2nd-order optimization with the Generalized Gauss-Newton matrix (GGN) provides intuition for why it might work well with large step-sizes, and gives prescriptions for how to make it work robustly in practice. In particular, we advocate viewing natural gradient descent as a GGN-based 2nd-order method in disguise (assuming the equivalence between the Fisher and GGN holds), and adopting standard practices from the optimization literature to ensure fast and robust performance, such as trust-region/damping/Tikhonov regularization, and Levenberg-Marquardt adjustment heuristics (Moré, 1978).

However, even in the case of squared loss, where the GGN becomes the standard Gauss-Newton matrix, we don't yet have a completely rigorous understanding of 2nd-order optimization with the GGN. A completely rigorous account of its global convergence remains elusive (even if we can assume convexity), and convergence rate bounds such as those proved in Section 14 don't even provide a complete picture of its *local* convergence properties.

Another issue with these kinds of local convergence bounds, which assume the objective is quadratic (or is well-approximated as such), is that they are always improved by using the Hessian instead of the GGN, and thus fail to explain the empirically observed superiority of the GGN over the Hessian for neural network training. This is because they assume that the objective function has constant curvature (given by the Hessian at the optimum), so that optimization could not possibly be helped by using an alternative curvature matrix like the GGN. And for this reason they also fail to explain why damping methods are so crucial in practice.

22. Note that the assumption that $B = I$ doesn't actually limit this result since stochastic 2nd-order optimization of a quadratic using a *fixed* B can be understood as stochastic gradient descent applied to a transformed version of the original quadratic (with an appropriately transformed gradient noise matrix Σ_g).

Moreover, even just interpreting the constants in these bounds can be hard when using the Fisher or GGN. For example, if we pay attention only to the noise-independent/starting point-dependent term in the bound from Theorem 6, which is given by

$$\frac{1}{2(k+1)^2\alpha^2}\left\|H^{*-1/2}B(\theta_0-\theta^*)\right\|^2,$$

and plug in $B=F$ and the maximum-allowable learning rate $\alpha=1/\lambda_1(B^{-1}H^*)$, we get the somewhat opaque expression

$$\frac{\lambda_1(F^{-1}H^*)^2}{2(k+1)^2}\left\|H^{*-1/2}F(\theta_0-\theta^*)\right\|^2.$$

It's not immediately obvious how we can further bound this expression in the non-realizable case (i.e. where we don't necessarily have $F=H^*$) using easily accessible/interpretable properties of the objective function. This is due to the complicated nature of the relationship between the GGN and Hessian, which we haven't explored in this report beyond the speculative discussion in Section 8.1.

Finally, we leave the reader with a few open questions.

- Can the observed advantages of the GGN vs the Hessian be rigorously justified for neural networks (assuming proper use of damping/trust-regions in both cases to deal with issues like negative curvature)?
- Are there situations where the Fisher and GGN are distinct and one is clearly preferable over the other?
- When will be pre-asymptotic advantage of stochastic 2nd-order methods vs SGD with Polyak averaging matter in practice? And can this be characterized rigorously using accessible properties of the target objective?

Acknowledgments

We gratefully acknowledge support from Google, DeepMind, and the University of Toronto. We would like to thank Léon Bottou, Guillaume Desjardins, Alex Botev, and especially the very thorough anonymous JMLR reviewers for their useful feedback on earlier versions of this manuscript.

Appendix A. Proof of Basic Identity from Murata (1998)

Proposition 7. *Given the definitions of Section 14 we have*

$$\mathrm{E}\left[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top\right] = V_k + (\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top .$$

Proof We have

$$\begin{aligned}
\mathrm{E}\left[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top\right] &= \mathrm{E}\left[(\theta_k - \mathrm{E}[\theta_k] + \mathrm{E}[\theta_k] - \theta^*)(\theta_k - \mathrm{E}[\theta_k] + \mathrm{E}[\theta_k] - \theta^*)^\top\right] \\
&= \mathrm{E}\left[(\theta_k - \mathrm{E}[\theta_k])(\theta_k - \mathrm{E}[\theta_k])^\top\right] + \mathrm{E}\left[(\mathrm{E}[\theta_k] - \theta^*)(\theta_k - \mathrm{E}[\theta_k])^\top\right] \\
&\quad + \mathrm{E}\left[(\theta_k - \mathrm{E}[\theta_k])(\mathrm{E}[\theta_k] - \theta^*)^\top\right] + \mathrm{E}\left[(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right] \\
&= V_k + (\mathrm{E}[\theta_k] - \theta^*)\,\mathrm{E}\left[(\theta_k - \mathrm{E}[\theta_k])\right]^\top \\
&\quad + \mathrm{E}\left[(\theta_k - \mathrm{E}[\theta_k])\right](\mathrm{E}[\theta_k] - \theta^*)^\top + (\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top \\
&= V_k + (\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top ,
\end{aligned}$$

where we have used $\mathrm{E}\left[(\theta_k - \mathrm{E}[\theta_k])\right] = \mathrm{E}[\theta_k] - \mathrm{E}[\theta_k] = 0$. ■

Appendix B. Proofs of Convergence Theorems

In this section we will prove Theorems 3, 5, and 6.

To begin, we recall the following linear operator definitions from the beginning of Section 14.2:

$$\begin{aligned}
\Xi(X) &= B^{-1}H^*X + \left(B^{-1}H^*X\right)^\top = B^{-1}H^*X + XH^*B^{-1} , \\
\Psi_\beta(X) &= \left(I - \beta B^{-1}H^*\right) X \left(I - \beta B^{-1}H^*\right)^\top = I - \beta\Xi(X) + \beta^2\Omega(X) ,
\end{aligned}$$

to which we add

$$\Omega(X) = B^{-1}H^*X\left(B^{-1}H^*\right)^\top = B^{-1}H^*XH^*B^{-1} .$$

We note that

$$\begin{aligned}
\Xi(\Omega(X)) &= B^{-1}H^*\left(B^{-1}H^*XH^*B^{-1}\right) + \left(B^{-1}H^*XH^*B^{-1}\right)H^*B^{-1} \\
&= B^{-1}H^*\left(B^{-1}H^*X + XH^*B^{-1}\right)H^*B^{-1} \\
&= \Omega(\Xi(X))
\end{aligned}$$

and so Ω and Ξ commute as operators. And because the remaining operators defined above are all linear combinations of these, it follows that they all commute with each other.

According to eqn. 19 we have

$$\begin{aligned}
\theta_{k+1} - \theta^* &= \theta_k - \alpha_k B^{-1} g_k(\theta_k) - \theta^* \\
&= \theta_k - \theta^* - \alpha_k B^{-1}(\nabla h(\theta_k) + \epsilon_k(\theta_k)) \\
&= \theta_k - \theta^* - \alpha_k B^{-1}H^*(\theta_k - \theta^*) - \alpha_k B^{-1}\epsilon_k(\theta_k) \\
&= (I - \alpha_k B^{-1}H^*)(\theta_k - \theta^*) - \alpha_k B^{-1}\epsilon_k(\theta_k) , \qquad (22)
\end{aligned}$$

where we have defined $\epsilon_k(\theta_k) = g_k(\theta_k) - \nabla h(\theta_k)$ and used $\nabla h(\theta_k) = H^*(\theta_k - \theta^*)$. Taking the expectation of both sides while using $\mathrm{E}\left[\epsilon_k(\theta_k)\right] = 0$ we get

$$\mathrm{E}[\theta_{k+1}] - \theta^* = (I - \alpha_k B^{-1} H^*)(\mathrm{E}[\theta_k] - \theta^*)\,.$$

Iterating this we get

$$\mathrm{E}[\theta_k] - \theta^* = \prod_{j=0}^{k-1} \left(I - \alpha_j B^{-1} H^*\right)(\theta_0 - \theta^*)\,. \tag{23}$$

Then observing that $\mathrm{Var}(\theta_k - \theta^*) = \mathrm{Var}(\theta_k) = V_k$ for all k, and exploiting the uncorrelatedness of θ_k and $\epsilon_k(\theta_k)$, we can take the variance of both sides of eqn. 22 to get that V_k evolves according to

$$\begin{aligned} V_{k+1} &= \mathrm{Var}((I - \alpha_k B^{-1} H^*)(\theta_k - \theta^*)) + \mathrm{Var}(\alpha_k B^{-1} \epsilon_k(\theta_k)) \\ &= (I - \alpha_k B^{-1} H^*) V_k (I - \alpha_k B^{-1} H^*)^\top + \alpha_k^2 B^{-1} \Sigma_g B^{-1} \\ &= \Lambda_k(V_k) + \alpha_k^2 U\,, \end{aligned}$$

where we have defined

$$\begin{aligned} U &= B^{-1} \Sigma_g B^{-1}\,, \text{ and} \\ \Lambda_k &= \Psi_{\alpha_k}\,. \end{aligned}$$

This recursion for V_k will be central in our analysis.

B.1 The Constant $\alpha_k = \alpha$ Case

In this subsection we will prove the claims made in Theorems 3 and 5 pertaining to the case where:

- $\alpha_k = \alpha$ for a constant α, and
- $\alpha \lambda_1(B^{-1} H^*) \leq 1$.

To begin, we define the notation $\Lambda = \Psi_\alpha$.

Corollary 8. *The operator $I - \Lambda$ has all positive eigenvalues and is thus invertible. Moreover, we have*

$$(I - \Lambda)^{-1} = \sum_{i=0}^{\infty} \Lambda^i\,.$$

And so if X is a PSD matrix then $(I - \Lambda)^{-1}(X)$ is as well.

Proof Let $D = I - \alpha B^{-1} H^*$, so that $\Lambda(X) = D X D^\top$.

The eigenvalues of $B^{-1} H^*$ are the same as those of $B^{-1/2} H^* B^{-1/2}$ (since the matrices differ by a similarity transform), and are thus real-valued and positive. Moreover, the minimum eigenvalue of D is $\lambda_n(D) = 1 - \alpha \lambda_1(B^{-1} H^*) \geq 0$, and the maximum eigenvalue is $\lambda_1(D) = 1 - \alpha \lambda_n(B^{-1} H^*) < 1$, where we have used $\lambda_n(B^{-1} H^*) = \lambda_n(B^{-1/2} H^* B^{-1/2}) > 0$ (both H^* and B are positive definite).

The claim then follows by an application of Lemma 21. ■

Lemma 9. *The variance matrix V_k is given by the formula*

$$V_k = \left(I - \Lambda^k\right)(V_\infty),$$

where $V_\infty = \alpha^2(I-\Lambda)^{-1}(U)$.

Proof By expanding the recursion for V_k we have

$$V_k = \alpha^2 \sum_{i=0}^{k-1} \Lambda^i(U).$$

And Corollary 8 tells us that

$$(I-\Lambda)^{-1} = \sum_{i=0}^{\infty} \Lambda^i.$$

Thus, we can write:

$$\begin{aligned}\sum_{i=0}^{k-1} \Lambda^i &= \sum_{i=0}^{\infty} \Lambda^i - \sum_{i=k}^{\infty} \Lambda^i \\ &= \sum_{i=0}^{\infty} \Lambda^i - \Lambda^k \sum_{i=0}^{\infty} \Lambda^i \\ &= (I-\Lambda^k)(I-\Lambda)^{-1}.\end{aligned}$$

We thus conclude that

$$V_k = \alpha^2 \sum_{i=0}^{k-1} \Lambda^i(U) = \alpha^2(I-\Lambda^k)(I-\Lambda)^{-1}(U) = (I-\Lambda^k)(V_\infty),$$

as claimed. ■

Proposition 10. *For any appropriately sized matrix X we have*

$$\operatorname{tr}\left(H^*\left(I-\Lambda\right)^{-1}(X)\right) = \frac{1}{2\alpha}\operatorname{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1} BXB\right).$$

Proof

Let $Y = (I-\Lambda)^{-1}(X)$, so that $(I-\Lambda)(Y) = X$. Written as a matrix equation this is

$$\alpha B^{-1}H^*Y + \alpha Y H^* B^{-1} - \alpha^2 B^{-1}H^*YH^*B^{-1} = X.$$

Left and right multiplying both sides by B we have

$$\alpha H^*YB + \alpha BYH^* - \alpha^2 H^*YH^* = BXB.$$

This can be written as $A^\top P + PA + Q = 0$, where

$$\begin{aligned} A &= YH^* \\ P &= \alpha\left(B - \frac{\alpha}{2}H^*\right) \\ Q &= -BXB\,. \end{aligned}$$

In order to compute $\mathrm{tr}(H^*Y)$ we can thus apply Lemma 17. However, we must first verify that our P is invertible. To this end we will show that $B - \frac{\alpha}{2}H^*$ is positive definite. This is equivalent to the condition that $B^{-1/2}(B - \frac{\alpha}{2}H^*)B^{-1/2} = I - \frac{\alpha}{2}B^{-1/2}H^*B^{-1/2}$ is positive definite, or in other words that $\lambda_1(B^{-1/2}H^*B^{-1/2}) = \lambda_1(B^{-1}H^*) < 2$. This is true by hypothesis (indeed, we are assuming $\alpha\lambda_1(B^{-1}H^*) < 1$).

By Lemma 17 it thus follows that

$$\mathrm{tr}(H^*Y) = \mathrm{tr}(YH^*) = \mathrm{tr}(A) = -\frac{1}{2}\,\mathrm{tr}(P^{-1}Q) = \frac{1}{2\alpha}\,\mathrm{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1} BXB\right)\,.$$

■

Lemma 11. *The expected objective value satisfies*

$$l(k) \leq \mathrm{E}[h(\theta_k)] - h(\theta^*) \leq u(k)\,,$$

where

$$u(k) = \left[1 - (1-\epsilon_1)^{2k}\right]\frac{\alpha}{4}\,\mathrm{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right) + (1-\epsilon_2)^{2k}\,h(\theta_0)$$

and

$$l(k) = \left[1 - (1-\epsilon_2)^{2k}\right]\frac{\alpha}{4}\,\mathrm{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right) + (1-\epsilon_1)^{2k}\,h(\theta_0)\,,$$

with $\epsilon_1 = \lambda_1(C) = \alpha\lambda_1\left(B^{-1}H^*\right)$ *and* $\epsilon_2 = \lambda_n(C) = \alpha\lambda_n\left(B^{-1}H^*\right)$.

Proof Note that for any appropriately sized matrix X we have

$$\begin{aligned} H^{*1/2}\Lambda(X)\,H^{*1/2} &= H^{*1/2}\left((I - \alpha B^{-1}H^*)X(I - \alpha H^*B^{-1})\right)\,H^{*1/2} \\ &= \left((I - \alpha H^{*1/2}B^{-1}H^{*1/2})H^{*1/2}XH^{*1/2}(I - \alpha H^{*1/2}B^{-1}H^{*1/2})\right) \\ &= \tilde{\Lambda}\left(H^{*1/2}XH^{*1/2}\right)\,, \end{aligned}$$

where we have defined

$$\tilde{\Lambda}(Y) = (I - C)Y(I - C)^\top = (I - C)Y(I - C)$$

with

$$C = \alpha H^{*1/2}B^{-1}H^{*1/2}\,.$$

Applying this repeatedly we obtain

$$H^{*1/2}\Lambda^k(X)\,H^{*1/2} = \tilde{\Lambda}^k(H^{*1/2}XH^{*1/2})\,.$$

Then, using the expression for V_k from Lemma 9, it follows that

$$\begin{aligned} H^{*1/2}V_kH^{*1/2} &= H^{*1/2}\left(V_\infty - \Lambda^k(V_\infty)\right)H^{*1/2} \\ &= H^{*1/2}\left(I - \Lambda^k\right)(V_\infty)H^{*1/2} \\ &= \left(I - \tilde{\Lambda}^k\right)(H^{*1/2}V_\infty H^{*1/2})\,. \end{aligned} \tag{24}$$

And thus

$$\begin{aligned} \frac{1}{2}\operatorname{tr}\left(H^*V_k\right) &= \frac{1}{2}\operatorname{tr}\left(H^{*1/2}V_kH^{*1/2}\right) \\ &= \frac{1}{2}\operatorname{tr}\left(H^{*1/2}V_\infty H^{*1/2}\right) - \frac{1}{2}\operatorname{tr}\left(\tilde{\Lambda}^k(H^{*1/2}V_\infty H^{*1/2})\right)\,. \end{aligned} \tag{25}$$

Next, we observe that for any matrix X

$$\operatorname{tr}\left(\tilde{\Lambda}^k(X)\right) = \operatorname{tr}\left((I-C)^kX(I-C)^k\right) = \operatorname{tr}\left((I-C)^{2k}X\right)\,. \tag{26}$$

Because the eigenvalues of a product of square matrices are invariant under cyclic permutation of those matrices, we have $\lambda_1(C) = \lambda_1(\alpha H^{*1/2}B^{-1}H^{*1/2}) = \alpha\lambda_1(B^{-1}H^*) \leq 1$ so that $I - C$ is PSD, and it thus follows that $\lambda_i((I-C)^{2k}) = (1-\lambda_{n-i+1}(C))^{2k}$. Then, assuming that X is also PSD, we can use Lemma 18 to get

$$(1-\lambda_1(C))^{2k}\operatorname{tr}(X) \leq \operatorname{tr}\left((I-C)^{2k}X\right) \leq (1-\lambda_n(C))^{2k}\operatorname{tr}(X)\,.$$

Applying this to eqn. 25 we thus have the upper bound

$$\frac{1}{2}\operatorname{tr}\left(H^*V_k\right) \leq \left(1-(1-\lambda_1(C))^{2k}\right)\frac{1}{2}\operatorname{tr}(H^*V_\infty)\,, \tag{27}$$

and the lower bound

$$\frac{1}{2}\operatorname{tr}\left(H^*V_k\right) \geq \left(1-(1-\lambda_n(C))^{2k}\right)\frac{1}{2}\operatorname{tr}(H^*V_\infty)\,. \tag{28}$$

From Lemma 9, V_∞ is given by $V_\infty = \alpha^2(I-\Lambda)^{-1}(B^{-1}\Sigma_gB^{-1})$. Thus, by Proposition 10 we have

$$\begin{aligned} \operatorname{tr}(H^*V_\infty) &= \alpha^2\frac{1}{2\alpha}\operatorname{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}BB^{-1}\Sigma_gB^{-1}B\right) \\ &= \frac{\alpha}{2}\operatorname{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right)\,. \end{aligned} \tag{29}$$

Next, we will compute/bound the term $\operatorname{tr}\left(H^*(\mathrm{E}[\theta_k]-\theta^*)(\mathrm{E}[\theta_k]-\theta^*)^\top\right)$.

From eqn. 23 we have

$$\mathrm{E}[\theta_k] - \theta^* = (I - \alpha B^{-1} H^*)^k (\theta_0 - \theta^*)\,.$$

Then observing

$$H^{*1/2}(I - \alpha B^{-1} H^*) = \left(I - \alpha H^{*1/2} B^{-1} H^{*1/2}\right) H^{*1/2} = (I - C)\, H^{*1/2}$$

and applying this recursively, it follows that

$$\begin{aligned} H^{*1/2}(\mathrm{E}[\theta_k] - \theta^*) &= H^{*1/2}(I - \alpha B^{-1} H^*)^k (\theta_0 - \theta^*) \\ &= (I - C)^k\, H^{*1/2}(\theta_0 - \theta^*)\,. \end{aligned} \tag{30}$$

Thus we have

$$\begin{aligned} \frac{1}{2}\operatorname{tr}\left(H^*(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right) &= \frac{1}{2}\operatorname{tr}\left(H^{*1/2}(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top H^{*1/2}\right) \\ &= \frac{1}{2}\operatorname{tr}\left((I - C)^k\, H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2}(I - C)^{k\top}\right) \\ &= \frac{1}{2}\operatorname{tr}\left((I - C)^{2k}\left(H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2}\right)\right)\,. \end{aligned}$$

Applying Lemma 18 in a similar manner to before, and using the fact that

$$h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^*(\theta - \theta^*) = \frac{1}{2}\operatorname{tr}\left(H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2}\right)\,,$$

we have the upper bound

$$\frac{1}{2}\operatorname{tr}\left(H^*(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right) \leq (1 - \lambda_n(C))^{2k}\, h(\theta_0)\,, \tag{31}$$

and the lower bound

$$\frac{1}{2}\operatorname{tr}\left(H^*(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right) \geq (1 - \lambda_1(C))^{2k}\, h(\theta_0)\,. \tag{32}$$

Combining eqn. 20, eqn. 27, eqn. 28, eqn. 29, eqn. 31, and eqn. 32 yields the claimed bound.

■

B.2 The $\alpha_k = 1/(k + a + 1)$ Case

In this subsection we will prove the claims made in Theorems 3 and 5 pertaining to the case where

- $\alpha_k = 1/(k + a + 1)$,
- $b = \lambda_n\left(B^{-1}H^*\right) > 1/2$, and
- $\lambda_1\left(B^{-1}H^*\right) \leq a + 1$.

Corollary 12. *The operator $\Xi - I$ has all positive eigenvalues and is thus invertible. Moreover, if X is a PSD matrix then $(\Xi - I)^{-1}(X)$ is as well.*

Proof We note that the operator $\Omega(X) = B^{-1}H^*XH^*B^{-1}$ is represented by the matrix $B^{-1}H^* \otimes B^{-1}H^*$. Because Kronecker products respect eigenvalue decompositions, the eigenvalues of this matrix are given by $\{\lambda_i(B^{-1}H^*)\lambda_j(B^{-1}H^*) \mid 0 \leq i, j \leq n\}$, and are thus all positive. (Because both H^* and B are positive definite, the eigenvalues of $B^{-1}H^*$ are all positive.) Thus, Ω^{-1} exists and has all positive eigenvalues. And observing that $(\Omega^{-1})(X) = H^{*-1}BXBH^{*-1}$, we see that Ω^{-1} preserves PSD matrices.

Define the operator $\Phi(X) = \Omega^{-1}((\Xi - I)(X))$. Because Ω^{-1} has all positive eigenvalues and commutes with Ξ, Φ will have all positive eigenvalues if and only if $\Xi - I$ does. Moreover, because Ω^{-1} is a bijection from the set of PSD matrices to itself, Φ^{-1} will map PSD matrices to PSD matrices if and only if $(\Xi - I)^{-1}$ does.

With these facts in hand it suffices to prove our various claims about the operator Φ.

Note that

$$\begin{aligned}
\Omega^{-1}((\Xi - I)(X)) &= \Omega^{-1}\left(B^{-1}H^*X + XH^*B^{-1} - X\right) \\
&= XBH^{*-1} + H^{*-1}BX - H^{*-1}BXBH^{*-1} \\
&= X - (I - H^{*-1}B)X(I - H^{*-1}B)^\top \\
&= X - DXD^\top,
\end{aligned}$$

where we have defined $D = I - H^{*-1}B$.

The eigenvalues of $H^{*-1}B$ are the same as those of $B^{1/2}H^{*-1}B^{1/2}$ (since the matrices differ by a similarity transform), and are thus real-valued and positive. Moreover, they are the inverse of those of $B^{-1}H^*$. So the minimum eigenvalue of D is $\lambda_n(D) = 1 - \lambda_1(H^{*-1}B) = 1 - 1/\lambda_n(B^{-1}H^*) \geq 1 - 1/0.5 = -1$. And the maximum eigenvalue of D is just $\lambda_n(D) = 1 - \lambda_n(H^{*-1}B) \leq 1$.

Lemma 21 is therefore applicable to Φ and the claim follows. ■

Lemma 13. *For all $k \geq 0$*

$$V_k = \frac{1}{k+a}(\Xi - I)^{-1}(U) + E_k,$$

where E_k is a matrix-valued "error" defined by the recursion

$$E_{k+1} = \Lambda_k(E_k) + \frac{1}{(k+a)(k+a+1)^2}Z \qquad E_0 = -\frac{1}{a}(\Xi - I)^{-1}(U),$$

with $Z = (\Xi - I)^{-1}\Psi_1(U)$.

Proof We will proceed by induction on k.

Observe that $V_0 = \mathrm{Var}(\theta_0) = 0$, and that this agrees with our claimed expression for V_k when evaluated at $k = 0$:

$$\begin{aligned}
V_0 &= \frac{1}{0+a}(\Xi - I)^{-1}(U) + E_0 \\
&= \frac{1}{a}(\Xi - I)^{-1}(U) - \frac{1}{a}(\Xi - I)^{-1}(U) = 0.
\end{aligned}$$

For the inductive case, suppose that $V_k = \frac{1}{k+a}\left(\Xi - I\right)^{-1}(U) + E_k$ for some k.

Observe that for any $i > 0$

$$\frac{1}{i} - \frac{1}{i(i+1)} = \frac{(i+1)-1}{i(i+1)} = \frac{i}{i(i+1)} = \frac{1}{i+1}\,,$$

from which it follows that

$$\frac{1}{k+a} - \frac{1}{(k+a)(k+a+1)} = \frac{1}{k+a+1}\,,$$

and thus also

$$\frac{1}{(k+a)(k+a+1)} - \frac{1}{(k+a)(k+a+1)^2} = \frac{1}{(k+a+1)^2}\,.$$

By definition, we have $\Psi_1 = I - \Xi + \Omega$, which implies

$$\left(\Xi - I\right)^{-1}\Psi_1 = \left(\Xi - I\right)^{-1}\Omega - I\,,$$

and so $Z = \left(\left(\Xi - I\right)^{-1}\Omega - I\right)(U)$.

Using the above expressions and the fact that the various linear operators commute we have

$$\begin{aligned}
V_{k+1} &= \Lambda_k(V_k) + \alpha_k^2 U \\
&= \Lambda_k(V_k) + \frac{1}{(k+a+1)^2}U \\
&= \Lambda_k\left(\frac{1}{k+a}\left(\Xi - I\right)^{-1}(U) + E_k\right) + \frac{1}{(k+a+1)^2}U \\
&= \Lambda_k\left(\frac{1}{k+a}\left(\Xi - I\right)^{-1}(U)\right) + \left(\frac{1}{(k+a)(k+a+1)} - \frac{1}{(k+a)(k+a+1)^2}\right)(U) + \Lambda_k(E_k) \\
&= \left(\Xi - I\right)^{-1}\left(\frac{1}{k+a}\Lambda_k + \frac{1}{(k+a)(k+a+1)}\left(\Xi - I\right) - \frac{1}{(k+a)(k+a+1)^2}\Omega\right)(U) \\
&\quad + \frac{1}{(k+a)(k+a+1)^2}\left(\left(\Xi - I\right)^{-1}\Omega - I\right)(U) + \Lambda_k(E_k) \\
&= \left(\Xi - I\right)^{-1}\left(\frac{1}{k+a}\Lambda_k + \frac{1}{(k+a)(k+a+1)}\left(\Xi - I\right) - \frac{1}{(k+a)(k+a+1)^2}\Omega\right)(U) + E_{k+1}\,.
\end{aligned}$$

Next, observing that

$$\begin{aligned}
&\frac{1}{k+a}\Lambda_k + \frac{1}{(k+a)(k+a+1)}\left(\Xi - I\right) - \frac{1}{(k+a)(k+a+1)^2}\Omega \\
&\qquad = \frac{1}{k+a}I - \frac{1}{(k+a)(k+a+1)}\Xi + \frac{1}{(k+a)(k+a+1)^2}\Omega \\
&\qquad\quad + \frac{1}{(k+a)(k+a+1)}\left(\Xi - I\right) - \frac{1}{(k+a)(k+a+1)^2}\Omega \\
&\qquad = \left(\frac{1}{k+a} - \frac{1}{(k+a)(k+a+1)}\right)I \\
&\qquad = \frac{1}{k+a+1}I\,,
\end{aligned}$$

our previous expression for V_{k+1} simplifies to

$$\begin{aligned} V_{k+1} &= (\Xi - I)^{-1} \left(\frac{1}{k+a+1} I \right) (U) + E_{k+1} \\ &= \frac{1}{k+a+1} (\Xi - I)^{-1} (U) + E_{k+1} \,. \end{aligned}$$

■

Proposition 14. *For any appropriately sized matrix X we have*

$$\operatorname{tr}\left(H^* (\Xi - I)^{-1} (X) \right) = \frac{1}{2} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2} H^{*-1} \right)^{-1} X \right) .$$

Proof

Let $Y = (\Xi - I)^{-1} (X)$, so that $(\Xi - I)(Y) = X$. Written as a matrix equation this is

$$B^{-1} H^* Y + Y H^* B^{-1} - Y = X \,.$$

And rearranging this becomes

$$Y H^* \left(B^{-1} - \frac{1}{2} H^{*-1} \right) + \left(B^{-1} - \frac{1}{2} H^{*-1} \right) H^* Y - X = 0 \,,$$

which is of the form $A^\top P + P A + Q = 0$ with

$$\begin{aligned} A &= H^* Y \\ P &= \left(B^{-1} - \frac{1}{2} H^{*-1} \right) \\ Q &= -X \,, \end{aligned}$$

In order to compute $\operatorname{tr}(H^* Y)$ we can thus apply Lemma 17. However, we must first verify that our P is invertible. To this end we will show that $B^{-1} - \frac{1}{2} H^{*-1}$ is positive definite. This is equivalent to the condition that $H^{*1/2}(B^{-1} - \frac{1}{2} H^{*-1}) H^{*1/2} = H^{*1/2} B^{-1} H^{*1/2} - \frac{1}{2} I$ is positive definite, or in other words that $\lambda_n(H^{*1/2} B^{-1} H^{*1/2}) = \lambda_n(B^{-1} H^*) > 1/2$, which is true by hypothesis.

Thus Lemma 17 is applicable, and yields

$$\operatorname{tr}(H^* Y) = \operatorname{tr}(A) = -\frac{1}{2} \operatorname{tr}(P^{-1} Q) = \frac{1}{2} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2} H^{*-1} \right)^{-1} X \right) .$$

■

Lemma 15. *For E_k as defined in Lemma 13 we have the following upper and lower bounds:*

$$\operatorname{tr}(H^* E_k) \leq \frac{\nu(a)k}{2(k+a)^3} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} \Psi_1(U)\right)$$

and

$$\operatorname{tr}(H^* E_k) \geq -\frac{1}{2a}\left(\frac{a+1}{k+a}\right)^{2b} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} U\right),$$

where $\nu(a) = (a+2)^3/(a(a+1)^2)$.

Proof

Recall that $E_0 = -\frac{1}{a}\left(\Xi - I\right)^{-1}(U)$ and $Z = \left(\Xi - I\right)^{-1}\Psi_1(U)$.

Observing that Ψ_1 preserves PSD matrices, that U is PSD, and that if a linear operator preserves PSD matrices it also preserves negative semi-definite (NSD) matrices (which follows directly from linearity), we can then apply Corollary 12 to get that E_0 is NSD and Z is PSD.

Define F_k by

$$F_{k+1} = \Lambda_k(F_k) + \frac{1}{(k+a)(k+a+1)^2} Z \qquad F_0 = 0\,,$$

and D_k by

$$D_{k+1} = \Lambda_k(D_k) \qquad D_0 = E_0\,.$$

Because Λ_k preserves PSD (and NSD) matrices like Ψ_1 does, and PSDness is preserved under non-negative linear combinations, it's straightforward to show that $D_k \preceq E_k \preceq F_k$ and $F_k \succeq 0 \succeq D_k$ for all k (where $X \preceq Y$ means that $Y - X$ is PSD). It thus follows that $\operatorname{tr}(H^* D_k) \leq \operatorname{tr}(H^* E_k) \leq \operatorname{tr}(H^* F_k)$.

Note that for any appropriately sized matrix X we have

$$\begin{aligned} H^{*1/2}\Lambda_k(X)\, H^{*1/2} &= H^{*1/2}\left((I - \alpha_k B^{-1}H^*)X(I - \alpha_k H^* B^{-1})\right) H^{*1/2} \\ &= \left((I - \alpha_k H^{*1/2}B^{-1}H^{*1/2})H^{*1/2}XH^{*1/2}(I - \alpha_k H^{*1/2}B^{-1}H^{*1/2})\right) \\ &= \tilde{\Lambda}_k\left(H^{*1/2}XH^{*1/2}\right), \end{aligned}$$

where we have defined

$$\tilde{\Lambda}(Y) = (I - C_k)Y(I - C_k)^\top = (I - C_k)Y(I - C_k)$$

with

$$C_k = \alpha_k H^{*1/2}B^{-1}H^{*1/2}\,.$$

It thus follows that

$$\begin{aligned}\operatorname{tr}(H^*\Lambda_k(X)) &= \operatorname{tr}\left(H^{*1/2}\Lambda_k(X)H^{*1/2}\right)\\ &= \operatorname{tr}\left((I-C_k)H^{*1/2}XH^{*1/2}(I-C_k)\right)\\ &= \operatorname{tr}\left((I-C_k)^2H^{*1/2}XH^{*1/2}\right).\end{aligned}$$

Because the eigenvalues of a product of square matrices are invariant under cyclic permutation of those matrices, we have $\lambda_1(C_k) = \lambda_1(\alpha_k H^{*1/2}B^{-1}H^{*1/2}) = \alpha_k\lambda_1(B^{-1}H^*) \leq \frac{1}{a+1}\lambda_1(B^{-1}H^*) \leq 1$ so that $I - C_k$ is PSD, and it thus follows that $\lambda_i((I-C_k)^2) = (1-\lambda_{n-i+1}(C_k))^2$. Then assuming X is also PSD we can use Lemma 18 to get

$$\begin{aligned}\operatorname{tr}\left((I-C_k)^2H^{*1/2}XH^{*1/2}\right) &\leq \lambda_1\left((I-C_k)^2\right)\operatorname{tr}(H^{*1/2}XH^{*1/2})\\ &= \left(1-\frac{b}{k+a+1}\right)^2\operatorname{tr}(H^*X),\end{aligned}$$

where we have defined $b = \lambda_n(B^{-1}H^*)$.

From this it follows that

$$\begin{aligned}\operatorname{tr}(H^*F_{k+1}) &= \operatorname{tr}\left(H^*\left(\Lambda_k(F_k) + \frac{1}{(k+a)(k+a+1)^2}Z\right)\right)\\ &= \operatorname{tr}(H^*\Lambda_k(F_k)) + \frac{1}{(k+a)(k+a+1)^2}\operatorname{tr}(H^*Z)\\ &\leq \left(1-\frac{b}{k+a+1}\right)^2\operatorname{tr}(H^*F_k) + \frac{1}{(k+a)(k+a+1)^2}\operatorname{tr}(H^*Z).\end{aligned}$$

Iterating this inequality and using the fact that $F_0 = 0$ we thus have

$$\begin{aligned}\operatorname{tr}(H^*F_k) &\leq \operatorname{tr}(H^*F_0)\prod_{j=0}^{k-1}\left(1-\frac{b}{j+a+1}\right)^2\\ &\quad + \operatorname{tr}(H^*Z)\sum_{i=0}^{k-1}\frac{1}{(i+a)(i+a+1)^2}\prod_{j=i+1}^{k-1}\left(1-\frac{b}{j+a+1}\right)^2\\ &= \operatorname{tr}(H^*Z)\sum_{i=0}^{k-1}\frac{1}{(i+a)(i+a+1)^2}\prod_{j=i+1}^{k-1}\left(1-\frac{b}{j+a+1}\right)^2.\end{aligned}$$

Then applying Proposition 23 we can further upper bound this by

$$\frac{\nu(a)k}{(k+a)^3}\operatorname{tr}(H^*Z),$$

where $\nu(a) = (a+2)^3/(a(a+1)^2)$.

Recalling the definition $Z = (\Xi - I)^{-1}\Psi_1(U)$ and applying Proposition 14, we have

$$\operatorname{tr}(H^*Z) = \operatorname{tr}\left(H^*(\Xi-I)^{-1}\Psi_1(U)\right) = \frac{1}{2}\operatorname{tr}\left(\left(B^{-1}-\frac{1}{2}H^{*-1}\right)^{-1}\Psi_1(U)\right).$$

And so in summary we have

$$\operatorname{tr}(H^* E_k) \leq \operatorname{tr}(H^* F_k) \leq \frac{\nu(a)k}{2(k+a)^3} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} \Psi_1(U)\right).$$

It remains to establish the claimed lower bound.
For NSD matrices X, Corollary 19 tells us that

$$\begin{aligned}\operatorname{tr}\left((I - C_k)^2 H^{*1/2} X H^{*1/2}\right) &\geq \lambda_1\left((I - C_k)^2\right) \operatorname{tr}(H^{*1/2} X H^{*1/2}) \\ &= \left(1 - \frac{b}{k+a+1}\right)^2 \operatorname{tr}(H^* X).\end{aligned}$$

From this it follows that

$$\operatorname{tr}(H^* D_{k+1}) = \operatorname{tr}\left(H^*\left(\Lambda_k(D_k)\right)\right) \geq \left(1 - \frac{b}{k+a+1}\right)^2 \operatorname{tr}(H^* D_k).$$

Iterating this inequality and using the fact that $D_0 = E_0$ we thus have

$$\begin{aligned}\operatorname{tr}(H^* D_k) &\geq \operatorname{tr}(H^* D_0) \prod_{j=0}^{k-1} \left(1 - \frac{b}{j+a+1}\right)^2 \\ &= \operatorname{tr}(H^* E_0) \prod_{j=0}^{k-1} \left(1 - \frac{b}{j+a+1}\right)^2.\end{aligned}$$

Then applying Proposition 23 we can further lower bound this by

$$\left(\frac{a+1}{k+a}\right)^{2b} \operatorname{tr}(H^* E_0).$$

By the definition of E_0 and Proposition 14 we have

$$\operatorname{tr}(H^* E_0) = -\frac{1}{a} \operatorname{tr}\left(H^*\left(\Xi - I\right)^{-1}(U)\right) = -\frac{1}{2a} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} U\right),$$

and so in summary our lower bound is

$$\operatorname{tr}(H^* E_k) \geq \operatorname{tr}(H^* D_k) \geq -\frac{1}{2a}\left(\frac{a+1}{k+a}\right)^{2b} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} U\right).$$

■

Lemma 16. *The expected objective value satisfies*

$$l(k) \leq \mathrm{E}[h(\theta_k)] - h(\theta^*) \leq u(k),$$

where

$$u(k) = \frac{1}{4(k+a)} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} U\right) + \frac{\nu(a)k}{4(k+a)^3} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} \Psi_1(U)\right) + h(\theta_0)\left(\frac{a+1}{k+a}\right)^{2b}$$

and

$$l(k) = \frac{1}{4(k+a)} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} U\right) - \frac{1}{4a}\left(\frac{a+1}{k+a}\right)^{2b} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} U\right),$$

where $\nu(a) = (a+2)^3/(a(a+1)^2)$.

Proof

From eqn. 20 we have

$$\mathrm{E}[h(\theta_k)] - h(\theta^*) = \frac{1}{2}\operatorname{tr}\left(H^* V_k\right) + \frac{1}{2}\operatorname{tr}\left(H^*(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right). \tag{33}$$

By the expression for V_k from Lemma 13 we have that

$$\begin{aligned}\frac{1}{2}\operatorname{tr}\left(H^* V_k\right) &= \frac{1}{2}\operatorname{tr}\left(H^*\left(\frac{1}{k+a}\left(\Xi - I\right)^{-1}(U) + E_k\right)\right) \\ &= \frac{1}{2(k+a)}\operatorname{tr}\left(H^*\left(\left(\Xi - I\right)^{-1}(U)\right)\right) + \frac{1}{2}\operatorname{tr}\left(H^* E_k\right).\end{aligned} \tag{34}$$

Applying Proposition 14 we have that the first term above is given by

$$\frac{1}{2(k+a)}\operatorname{tr}\left(H^*\left(\left(\Xi - I\right)^{-1}(U)\right)\right) = \frac{1}{4(k+a)}\operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} U\right). \tag{35}$$

And by Lemma 15, the second term is lower and upper bounded as follows:

$$\begin{aligned}-\frac{1}{4a}\left(\frac{a+1}{k+a}\right)^{2b} \operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} U\right) &\leq \frac{1}{2}\operatorname{tr}(H^* E_k) \\ &\leq \frac{\nu(a)k}{4(k+a)^3}\operatorname{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1}\Psi_1(U)\right),\end{aligned} \tag{36}$$

where $\nu(a) = (a+2)^3/(a(a+1)^2)$.

It remains to compute/bound the term $\operatorname{tr}\left(H^*(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right)$. From eqn. 23 we have

$$\mathrm{E}[\theta_k] - \theta^* = \prod_{j=0}^{k-1}\left(I - \alpha_j B^{-1} H^*\right)(\theta_0 - \theta^*).$$

Observing

$$H^{*1/2}(I - \alpha_i B^{-1}H^*) = (I - \alpha_i H^{*1/2}B^{-1}H^{*1/2})H^{*1/2}$$

it follows that

$$\begin{aligned}
H^{*1/2}(\mathrm{E}[\theta_k] - \theta^*) &= H^{*1/2}\prod_{j=0}^{k-1}\left(I - \alpha_j B^{-1}H^*\right)(\theta_0 - \theta^*) \\
&= \prod_{j=0}^{k-1}\left(I - \alpha_j H^{*1/2}B^{-1}H^{*1/2}\right)H^{*1/2}(\theta_0 - \theta^*) \\
&= \psi_k\left(H^{*1/2}B^{-1}H^{*1/2}\right)H^{*1/2}(\theta_0 - \theta^*)\,,
\end{aligned}$$

where ψ_k is a polynomial defined by

$$\psi_k(x) = \prod_{j=0}^{k-1}(1 - \alpha_j x) = \prod_{j=0}^{k-1}\left(1 - \frac{x}{j+a+1}\right)\,.$$

In the domain $0 \le x \le a+1$ we have that $\psi_k(x)$ is a decreasing function of x. Moreover, for x's in this domain Proposition 23 says that

$$\psi_k(x) \le \left(\frac{a+1}{k+a}\right)^x\,.$$

So because the eigenvalues of $\psi_k(X)$ for any matrix X are given by $\{\psi_k(\lambda_i(X))\}_i$, and $\lambda_1(B^{-1}H^*) \le a+1$ by hypothesis, it thus follows that

$$\lambda_1\left(\psi_k\left(H^{*1/2}B^{-1}H^{*1/2}\right)\right) = \psi_k\left(\lambda_n\left(H^{*1/2}B^{-1}H^{*1/2}\right)\right) = \psi_k\left(\lambda_n\left(B^{-1}H^*\right)\right) \le \left(\frac{a+1}{k+a}\right)^b\,.$$

And so by Lemma 18 we have that

$$\begin{aligned}
&\frac{1}{2}\operatorname{tr}\left(H^*(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top\right) = \frac{1}{2}\operatorname{tr}\left(H^{*1/2}(\mathrm{E}[\theta_k] - \theta^*)(\mathrm{E}[\theta_k] - \theta^*)^\top H^{*1/2}\right) \\
&\quad= \frac{1}{2}\operatorname{tr}\left(\psi_k\left(H^{*1/2}B^{-1}H^{*1/2}\right)^2\left(H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2}\right)\right) \\
&\quad\le \lambda_1\left(\psi_k\left(H^{*1/2}B^{-1}H^{*1/2}\right)\right)^2\frac{1}{2}\operatorname{tr}\left(H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2}\right) \qquad (37) \\
&\quad= \lambda_1\left(\psi_k\left(H^{*1/2}B^{-1}H^{*1/2}\right)\right)^2\frac{1}{2}(\theta_0 - \theta^*)^\top H^*(\theta_0 - \theta^*) \\
&\quad\le \left(\frac{a+1}{k+a}\right)^{2b} h(\theta_0)\,.
\end{aligned}$$

Combining eqn. 33, eqn. 34, eqn. 35, eqn. 36 and the above bound the claimed result follows.

■

B.3 Proof of Theorem 6

Theorem. *Suppose that θ_k is generated by the stochastic iteration in eqn. 19 with constant step-size $\alpha_k = \alpha$ while optimizing a quadratic objective $h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^*(\theta - \theta^*)$. Further more, suppose that $\alpha\lambda_1(B^{-1}H^*) < 1$, and define $\bar{\theta}_k = \frac{1}{k+1}\sum_{i=0}^{k}\theta_i$. Then we have the following bound:*

$$\begin{aligned}\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \min\left\{\frac{1}{2(k+1)}\operatorname{tr}\left(H^{*-1}\Sigma_g\right),\ \frac{\alpha}{4}\operatorname{tr}\left(\left(B - \frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right)\right\} \\ + \min\left\{\frac{1}{2(k+1)^2\alpha^2}\left\|H^{*-1/2}B(\theta_0 - \theta^*)\right\|^2, \right. \\ \left. \frac{1}{2(k+1)\alpha}\left\|B^{1/2}(\theta_0 - \theta^*)\right\|^2,\ h(\theta_0)\right\}.\end{aligned}$$

Proof

To begin, we observe that, analogously to eqn. 20,

$$\mathrm{E}[h(\bar{\theta}_k)] - h(\theta^*) = \frac{1}{2}\operatorname{tr}\left(H^*\bar{V}_k\right) + \frac{1}{2}\operatorname{tr}\left(H^*\left(\mathrm{E}[\bar{\theta}_k] - \theta^*\right)\left(\mathrm{E}[\bar{\theta}_k] - \theta^*\right)^\top\right), \tag{38}$$

where

$$\bar{V}_k = \operatorname{Var}(\bar{\theta}_k) = \operatorname{Cov}(\bar{\theta}_k, \bar{\theta}_k) = \mathrm{E}\left[\left(\bar{\theta}_k - \mathrm{E}[\bar{\theta}_k]\right)\left(\bar{\theta}_k - \mathrm{E}[\bar{\theta}_k]\right)^\top\right].$$

Our first major task is to find an expression for $\bar{V}_k$ in order to bound the term $\frac{1}{2}\operatorname{tr}\left(H^*\bar{V}_k\right)$. To this end we observe that

$$\bar{V}_k = \frac{1}{(k+1)^2}\sum_{i=0}^{k}\sum_{j=0}^{k}\operatorname{Cov}(\theta_i, \theta_j).$$

For $j > i$ we have

$$\operatorname{Cov}(\theta_i, \theta_j) = \operatorname{Cov}\left(\theta_i, \theta_{j-1} - \alpha B^{-1}g_{j-1}(\theta_{j-1})\right) = \operatorname{Cov}(\theta_i, \theta_{j-1}) - \alpha\operatorname{Cov}(\theta_i, g_{j-1}(\theta_{j-1}))B^{-1},$$

where

$$\begin{aligned}\operatorname{Cov}(\theta_i, g_{j-1}(\theta_{j-1})) &= \mathrm{E}\left[(\theta_i - \mathrm{E}[\theta_i])(g_{j-1}(\theta_{j-1}) - \mathrm{E}[g_{j-1}(\theta_{j-1})])^\top\right] \\ &= \mathrm{E}_{\theta_i,\theta_{j-1}}\left[\mathrm{E}_{g_{j-1}(\theta_{j-1})|\theta_{j-1}}\left[(\theta_i - \mathrm{E}[\theta_i])(g_{j-1}(\theta_{j-1}) - \mathrm{E}[g_{j-1}(\theta_{j-1})])^\top\right]\right] \\ &= \mathrm{E}_{\theta_i,\theta_{j-1}}\left[(\theta_i - \mathrm{E}[\theta_i])(\nabla h(\theta_{j-1}) - \mathrm{E}[g_{j-1}(\theta_{j-1})])^\top\right] \\ &= \mathrm{E}\left[(\theta_i - \mathrm{E}[\theta_i])(\nabla h(\theta_{j-1}) - \mathrm{E}[g_{j-1}(\theta_{j-1})])^\top\right].\end{aligned}$$

Here we have used the fact that $g_{j-1}(\theta_{j-1})$ is conditionally independent of θ_i given θ_{j-1} for $j - 1 \geq i$ (which allows us to take the conditional expectation over $g_{j-1}(\theta_{j-1})$ inside), and is an unbiased estimator of $\nabla h(\theta_{j-1})$.

Then, noting that $\mathrm{E}[g_{j-1}(\theta_{j-1})] = \mathrm{E}[\nabla h(\theta_{j-1})] = \mathrm{E}[H^*(\theta_{j-1} - \theta^*)] = H^*(\mathrm{E}[\theta_{j-1}] - \theta^*)$, we have

$$\begin{aligned}\nabla h(\theta_{j-1}) - \mathrm{E}[g_{j-1}(\theta_{j-1})] &= H^*(\theta_{j-1} - \theta^*) - H^*(\mathrm{E}[\theta_{j-1}] - \theta^*) \\ &= H^*(\theta_{j-1} - \mathrm{E}[\theta_{j-1}])\end{aligned}$$

so that

$$\begin{aligned}\mathrm{Cov}\left(\theta_i, g_{j-1}(\theta_{j-1})\right) &= \mathrm{E}\left[(\theta_i - \mathrm{E}[\theta_i])(\nabla h(\theta_{j-1}) - \mathrm{E}[g_{j-1}(\theta_{j-1})])^\top\right] \\ &= \mathrm{E}\left[(\theta_i - \mathrm{E}[\theta_i])(H^*(\theta_{j-1} - \mathrm{E}[\theta_{j-1}]))^\top\right] \\ &= \mathrm{E}\left[(\theta_i - \mathrm{E}[\theta_i])(\theta_{j-1} - \mathrm{E}[\theta_{j-1}])^\top\right] H^* = \mathrm{Cov}(\theta_i, \theta_{j-1}) H^* \,.\end{aligned}$$

From this we conclude that

$$\begin{aligned}\mathrm{Cov}\left(\theta_i, \theta_j\right) &= \mathrm{Cov}\left(\theta_i, \theta_{j-1}\right) - \alpha \,\mathrm{Cov}\left(\theta_i, g_{j-1}(\theta_{j-1})\right) B^{-1} \\ &= \mathrm{Cov}\left(\theta_i, \theta_{j-1}\right) - \alpha \,\mathrm{Cov}(\theta_i, \theta_{j-1}) H^* B^{-1} \\ &= \mathrm{Cov}\left(\theta_i, \theta_{j-1}\right)\left(I - \alpha B^{-1} H^*\right)^\top \,.\end{aligned}$$

Applying this recursively we have that for $j \geq i$

$$\mathrm{Cov}\left(\theta_i, \theta_j\right) = V_i\big(I - \alpha B^{-1} H^*\big)^{j-i\,\top} \,. \tag{39}$$

Taking transposes and switching the roles of i and j we similarly have for $i \geq j$ that

$$\mathrm{Cov}\left(\theta_i, \theta_j\right) = \left(I - \alpha B^{-1} H^*\right)^{i-j} V_j \,.$$

Thus, we have the following expression for the variance $\bar{V}_k$ of the averaged parameter $\bar{\theta}_k$:

$$\begin{aligned}\bar{V}_k &= \frac{1}{(k+1)^2} \sum_{i=0}^{k} \sum_{j=0}^{k} \mathrm{Cov}(\theta_i, \theta_j) \\ &= \frac{1}{(k+1)^2} \sum_{i=0}^{k} \left(\sum_{j=0}^{i} \left(I - \alpha B^{-1} H^*\right)^{i-j} V_j + \sum_{j=i+1}^{k} V_i \big(I - \alpha B^{-1} H^*\big)^{j-i\,\top} \right) ,\end{aligned}$$

which by reordering the sums and re-indexing can be written as

$$\bar{V}_k = \frac{1}{(k+1)^2} \sum_{i=0}^{k} \left(\sum_{j=0}^{i} \left(I - \alpha B^{-1} H^*\right)^{j} V_i + \sum_{j=1}^{k-i} V_i \big(I - \alpha B^{-1} H^*\big)^{j\,\top} \right) .$$

Having computed $\bar{V}_k$ we now deal with the term $\frac{1}{2}\,\mathrm{tr}\left(H^* \bar{V}_k\right)$. Observing that

$$H^{*1/2}(I - \alpha B^{-1} H^*) = \left(I - \alpha H^{*1/2} B^{-1} H^{*1/2}\right) H^{*1/2} = (I - C)\, H^{*1/2} \,,$$

where $C = \alpha H^{*1/2} B^{-1} H^{*1/2}$, we have

$$H^{*1/2} \bar{V}_k H^{*1/2} = \frac{1}{(k+1)^2} \sum_{i=0}^{k} \left(\sum_{j=0}^{i} (I-C)^j \left(H^{*1/2} V_i H^{*1/2} \right) + \sum_{j=1}^{k-i} (H^{*1/2} V_i H^{*1/2}) \left(I - C \right)^j \right) .$$

It thus follows that

$$\begin{aligned} \frac{1}{2} \operatorname{tr}\left(H^* \bar{V}_k \right) &= \frac{1}{2} \operatorname{tr}\left(H^{*1/2} \bar{V}_k H^{*1/2} \right) \\ &= \frac{1}{2(k+1)^2} \sum_{i=0}^{k} \operatorname{tr}\left(\left(\sum_{j=0}^{i} (I-C)^j + \sum_{j=1}^{k-i} (I-C)^j \right) H^{*1/2} V_i H^{*1/2} \right) . \end{aligned}$$

Recall that from eqn. 24 we have

$$H^{*1/2} V_i H^{*1/2} = \left(I - \tilde{\Lambda}^i \right) \left(H^{*1/2} V_\infty H^{*1/2} \right) ,$$

where $\tilde{\Lambda}(Y) = (I-C)Y(I-C)^\top = (I-C)Y(I-C)$.

Plugging this into the previous equation, using the definition of $\tilde{\Lambda}$, and the fact that various powers of C commute, we have

$$\begin{aligned} \frac{1}{2} \operatorname{tr}\left(H^* \bar{V}_k \right) &= \frac{1}{2(k+1)^2} \operatorname{tr}\left(\sum_{i=0}^{k} \left(\sum_{j=0}^{i} (I-C)^j + \sum_{j=1}^{k-i} (I-C)^j \right) \left(I - \tilde{\Lambda}^i \right) \left(H^{*1/2} V_\infty H^{*1/2} \right) \right) \\ &= \frac{1}{2(k+1)^2} \operatorname{tr}\left(\sum_{i=0}^{k} \left(\sum_{j=0}^{i} (I-C)^j + \sum_{j=1}^{k-i} (I-C)^j \right) \left(I - (I-C)^{2i} \right) H^{*1/2} V_\infty H^{*1/2} \right) . \end{aligned} \tag{40}$$

Because C and $I-C$ are PSD (which follows from the hypothesis $\lambda_1(C) = \alpha \lambda_1(B^{-1} H^*) < 1$), we have the following basic matrix inequalities:

$$\sum_{j=0}^{i} (I-C)^j + \sum_{j=1}^{k-i} (I-C)^j \preceq 2 \sum_{j=0}^{\infty} (I-C)^j - I = 2C^{-1} - I \tag{41}$$

$$\sum_{j=0}^{i} (I-C)^j + \sum_{j=1}^{k-i} (I-C)^j \preceq (k+1) I \tag{42}$$

$$\sum_{i=0}^{k} \left(I - (I-C)^{2i} \right) \preceq (k+1) I , \tag{43}$$

where $X \preceq Y$ means that $Y - X$ is PSD.

As the right and left side of all the previously stated matrix inequalities are commuting matrices (because they are all linear combinations of powers of C, and thus share their eigenvectors with C), we can apply Lemma 20 to eqn. 40 to obtain various simplifying upper bounds on $\frac{1}{2} \operatorname{tr}\left(H^* \bar{V}_k \right)$.

Applying Lemma 20 using eqn. 41 and then eqn. 43 gives the upper bound

$$\begin{aligned}\frac{1}{2}\operatorname{tr}\left(H^*\bar{V}_k\right) &\leq \frac{1}{2(k+1)^2}\operatorname{tr}\left(\left(2C^{-1}-I\right)\ (k+1)I\ H^{*1/2}V_\infty H^{*1/2}\right)\\ &= \frac{1}{k+1}\operatorname{tr}\left(\left(\frac{1}{\alpha}B-\frac{1}{2}H^*\right)\ V_\infty\right),\end{aligned}$$

where we have used $H^{*1/2}C^{-1}H^{*1/2} = H^{*1/2}\left(\alpha H^{*1/2}B^{-1}H^{*1/2}\right)^{-1}H^{*1/2} = \frac{1}{\alpha}B$.

Or we can apply the lemma using eqn. 42 and then eqn. 43, which gives a different upper bound of

$$\begin{aligned}\frac{1}{2}\operatorname{tr}\left(H^*\bar{V}_k\right) \leq \frac{1}{2(k+1)^2}\operatorname{tr}\left((k+1)I\ (k+1)I\ H^{*1/2}V_\infty H^{*1/2}\right) &\leq \frac{1}{2}\operatorname{tr}\left(H^*V_\infty\right)\\ &= \frac{\alpha}{4}\operatorname{tr}\left(\left(B-\frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right),\end{aligned}$$

where we have used eqn. 29 on the last line.

Applying these bounds to eqn. 40 yields

$$\frac{1}{2}\operatorname{tr}\left(H^*\bar{V}_k\right) \leq \min\left\{\frac{1}{k+1}\operatorname{tr}\left(\left(\frac{1}{\alpha}B-\frac{1}{2}H^*\right)\ V_\infty\right),\ \frac{\alpha}{4}\operatorname{tr}\left(\left(B-\frac{\alpha}{2}H^*\right)^{-1}\Sigma_g\right)\right\}. \quad (44)$$

To compute $\operatorname{tr}\left(\left(\frac{1}{\alpha}B-\frac{1}{2}H^*\right)V_\infty\right)$, we begin by recalling the definition $V_\infty = \alpha^2\left(I-\Lambda\right)^{-1}(U)$. Applying the operator $(I-\Lambda)$ to both sides gives $(I-\Lambda)\left(V_\infty\right) = \alpha^2(U)$, which corresponds to the matrix equation

$$\alpha B^{-1}H^*V_\infty + \alpha V_\infty H^*B^{-1} - \alpha^2 B^{-1}H^*V_\infty H^*B^{-1} = \alpha^2 U = \alpha^2 B^{-1}\Sigma_g B^{-1}.$$

Left and right multiplying both sides by $\frac{1}{\alpha}B$ gives

$$\frac{1}{\alpha}H^*V_\infty B + \frac{1}{\alpha}BV_\infty H^* - H^*V_\infty H^* = \Sigma_g,$$

which can be rewritten as

$$\left(\frac{1}{\alpha}B-\frac{1}{2}H^*\right)V_\infty H^* + H^*V_\infty\left(\frac{1}{\alpha}B-\frac{1}{2}H^*\right) = \Sigma_g.$$

This is of the form $A^\top P + PA + Q = 0$ where

$$\begin{aligned}A &= V_\infty\left(\frac{1}{\alpha}B-\frac{1}{2}H^*\right)\\ P &= H^*\\ Q &= -\Sigma_g.\end{aligned}$$

Applying Lemma 17 we get that

$$\operatorname{tr}\left(\left(\frac{1}{\alpha}B-\frac{1}{2}H^*\right)V_\infty\right) = \operatorname{tr}(A^\top) = \operatorname{tr}(A) = \frac{1}{2}\operatorname{tr}(P^{-1}Q) = \frac{1}{2}\operatorname{tr}\left(H^{*-1}\Sigma_g\right). \quad (45)$$

It remains to bound the term $\frac{1}{2}\operatorname{tr}\left(H^*(\mathrm{E}[\bar{\theta}_k]-\theta^*)(\mathrm{E}[\bar{\theta}_k]-\theta^*)^\top\right)$.

First, we observe that by Theorem 3

$$\mathrm{E}[\bar{\theta}_k]-\theta^* = \frac{1}{k+1}\sum_{i=0}^{k}(\mathrm{E}[\theta_i]-\theta^*) = \frac{1}{k+1}\sum_{i=0}^{k}\left(I-\alpha B^{-1}H^*\right)^i(\theta_0-\theta^*)\,.$$

Applying eqn. 30 then gives

$$H^{*1/2}\left(\mathrm{E}[\bar{\theta}_k]-\theta^*\right) = \frac{1}{k+1}\sum_{i=0}^{k}(I-C)^i H^{*1/2}(\theta_0-\theta^*)\,.$$

And thus we have

$$\begin{aligned}\frac{1}{2}\operatorname{tr}\left(H^*\left(\mathrm{E}[\bar{\theta}_k]-\theta^*\right)\left(\mathrm{E}[\bar{\theta}_k]-\theta^*\right)^\top\right) &= \frac{1}{2}\operatorname{tr}\left(H^{*1/2}\left(\mathrm{E}[\bar{\theta}_k]-\theta^*\right)\left(\mathrm{E}[\bar{\theta}_k]-\theta^*\right)^\top H^{*1/2}\right)\\ &= \frac{1}{2(k+1)^2}\operatorname{tr}\left(\left(\sum_{i=0}^{k}(I-C)^i\right)H^{*1/2}(\theta_0-\theta^*)(\theta_0-\theta^*)^\top H^{*1/2}\left(\sum_{i=0}^{k}(I-C)^i\right)\right)\,.\end{aligned} \tag{46}$$

Similarly to eqn. 41–43 we have the following matrix inequalities

$$\sum_{i=0}^{k}(I-C)^i \preceq \sum_{i=0}^{\infty}(I-C)^i = C^{-1} \tag{47}$$

$$\sum_{i=0}^{k}(I-C)^i \preceq (k+1)I\,. \tag{48}$$

Applying Lemma 20 using eqn. 47 twice we obtain an upper bound on the RHS of eqn. 46 of

$$\frac{1}{2(k+1)^2}\operatorname{tr}\left(C^{-1}\,H^{*1/2}(\theta_0-\theta^*)(\theta_0-\theta^*)^\top H^{*1/2}\,C^{-1}\right) = \frac{1}{2(k+1)^2\alpha^2}\left\|H^{*-1/2}B(\theta_0-\theta^*)\right\|^2\,.$$

Applying the lemma using eqn. 47 and eqn. 48 gives a different upper bound of

$$\frac{1}{2(k+1)^2}\operatorname{tr}\left(C^{-1}\,H^{*1/2}(\theta_0-\theta^*)(\theta_0-\theta^*)^\top H^{*1/2}\,(k+1)I\right) \;= \frac{1}{2(k+1)\alpha}\left\|B^{1/2}(\theta_0-\theta^*)\right\|^2\,.$$

And finally, applying the lemma using eqn. 48 twice gives an upper bound of

$$\frac{1}{2(k+1)^2}\operatorname{tr}\left((k+1)I\,H^{*1/2}(\theta_0-\theta^*)(\theta_0-\theta^*)^\top H^{*1/2}\,(k+1)I\right) = h(\theta_0)\,.$$

Combining these various upper bounds gives us

$$\begin{aligned}&\frac{1}{2}\operatorname{tr}\left(H^*(\mathrm{E}[\bar{\theta}_k]-\theta^*)(\mathrm{E}[\bar{\theta}_k]-\theta^*)^\top\right)\\ &\quad\le \min\left\{\frac{1}{2(k+1)^2\alpha^2}\left\|H^{*-1/2}B(\theta_0-\theta^*)\right\|^2,\;\frac{1}{2(k+1)\alpha}\left\|B^{1/2}(\theta_0-\theta^*)\right\|^2,\;h(\theta_0)\right\}\,.\end{aligned} \tag{49}$$

The result now follows from eqn. 38, eqn. 44, eqn. 45, and eqn. 49.

■

Appendix C. Derivations of Bounds for Section 14.2.1

By Lemma 18

$$\operatorname{tr}\left(H^{*-1}\Sigma_g\right) \geq \lambda_n\left(H^{*-1}\right)\operatorname{tr}(\Sigma_g) = \frac{1}{\lambda_1\left(H^*\right)}\operatorname{tr}(\Sigma_g)$$

and

$$\operatorname{tr}\left(H^{*-1}\Sigma_g\right) \leq \lambda_1\left(H^{*-1}\right)\operatorname{tr}(\Sigma_g) = \frac{1}{\lambda_n\left(H^*\right)}\operatorname{tr}(\Sigma_g)\,,$$

so that

$$\frac{1}{2\lambda_1\left(H^*\right)}\operatorname{tr}(\Sigma_g) \leq \frac{1}{2}\operatorname{tr}\left(H^{*-1}\Sigma_g\right) \leq \frac{1}{2\lambda_n\left(H^*\right)}\operatorname{tr}(\Sigma_g)\,.$$

Meanwhile, by Lemma 18

$$\begin{aligned}
\operatorname{tr}\left(\left(I - \frac{\lambda_n(H^*)}{2}H^{*-1}\right)^{-1}\Sigma_g\right) &\geq \lambda_n\left(\left(I - \frac{\lambda_n(H^*)}{2}H^{*-1}\right)^{-1}\right)\operatorname{tr}(\Sigma_g) \\
&= \frac{1}{\lambda_1\left(I - \frac{\lambda_n(H^*)}{2}H^{*-1}\right)}\operatorname{tr}(\Sigma_g) \\
&= \frac{1}{1 - \frac{\lambda_n(H^*)}{2}\lambda_n(H^{*-1})}\operatorname{tr}(\Sigma_g) \\
&= \frac{1}{1 - \frac{1}{2\kappa(H^*)}}\operatorname{tr}(\Sigma_g) \geq \operatorname{tr}(\Sigma_g)\,,
\end{aligned}$$

where $\kappa(H^*) = \lambda_1(H^*)/\lambda_n(H^*)$ is the condition number of H^*. Similarly, by Lemma 18 we have

$$\begin{aligned}
\operatorname{tr}\left(\left(I - \frac{\lambda_n(H^*)}{2}H^{*-1}\right)^{-1}\Sigma_g\right) &\leq \lambda_1\left(\left(I - \frac{\lambda_n(H^*)}{2}H^{*-1}\right)^{-1}\right)\operatorname{tr}(\Sigma_g) \\
&= \frac{1}{\lambda_n\left(I - \frac{\lambda_n(H^*)}{2}H^{*-1}\right)}\operatorname{tr}(\Sigma_g) \\
&= \frac{1}{1 - \frac{\lambda_n(H^*)}{2}\lambda_1(H^{*-1})}\operatorname{tr}(\Sigma_g) \\
&= \frac{1}{1 - \frac{\lambda_n(H^*)}{2\lambda_n(H^*)}}\operatorname{tr}(\Sigma_g) = 2\operatorname{tr}(\Sigma_g)\,,
\end{aligned}$$

and thus

$$\frac{1}{4\lambda_n(H^*)}\operatorname{tr}(\Sigma_g) \leq \frac{1}{4\lambda_n(H^*)}\operatorname{tr}\left(\left(I - \frac{\lambda_n(H^*)}{2}H^{*-1}\right)^{-1}\Sigma_g\right) \leq \frac{1}{2\lambda_n(H^*)}\operatorname{tr}(\Sigma_g)\,.$$

Appendix D. Some Self-contained Technical Results

Lemma 17. *Suppose $A^\top P + PA + Q = 0$ is a matrix equation where P is invertible. Then we have*

$$\operatorname{tr}(A) = -\frac{1}{2}\operatorname{tr}(P^{-1}Q)\,.$$

Proof Pre-multiplying both sides of $A^\top P + PA + Q = 0$ by P^{-1} and taking the trace yields $\operatorname{tr}(P^{-1}A^\top P) + \operatorname{tr}(A) + \operatorname{tr}(P^{-1}Q) = 0$. Then noting that $\operatorname{tr}(P^{-1}A^\top P) = \operatorname{tr}(PP^{-1}A^\top) = \operatorname{tr}(A^\top) = \operatorname{tr}(A)$ this becomes $2\operatorname{tr}(A) + \operatorname{tr}(P^{-1}Q) = 0$, from which the claim follows. ■

Lemma 18 (Adapted from Lemma 1 from Wang et al. (1986))**.** *Suppose X and S are $n \times n$ matrices such that S is symmetric and X is PSD. Then we have*

$$\lambda_n(S)\operatorname{tr}(X) \leq \operatorname{tr}(SX) \leq \lambda_1(S)\operatorname{tr}(X)\,.$$

Corollary 19. *Suppose X and S are $n \times n$ matrices such that S is symmetric and X is negative semi-definite (NSD). Then we have*

$$\lambda_1(S)\operatorname{tr}(X) \leq \operatorname{tr}(SX) \leq \lambda_n(S)\operatorname{tr}(X)\,.$$

Proof Because X is NSD, $-X$ is PSD. We can therefore apply Lemma 18 to get that

$$\lambda_n(S)\operatorname{tr}(-X) \leq \operatorname{tr}(S(-X)) \leq \lambda_1(S)\operatorname{tr}(-X)\,,$$

or in other words

$$-\lambda_n(S)\operatorname{tr}(X) \leq -\operatorname{tr}(SX) \leq -\lambda_1(S)\operatorname{tr}(X)\,.$$

Multiplying by -1 this becomes

$$\lambda_n(S)\operatorname{tr}(X) \geq \operatorname{tr}(SX) \geq \lambda_1(S)\operatorname{tr}(X)\,.$$

■

Lemma 20. *If A, S, T, and X are matrices such that A, S and T commute with each other, $S \preceq T$ (i.e. $T - S$ is PSD), and A and X are PSD, then we have*

$$\operatorname{tr}(ASX) \leq \operatorname{tr}(ATX)\,.$$

Proof Since A, S and T are commuting PSD matrices they have the same eigenvectors, as does $A^{1/2}$ (which thus also commutes).

Meanwhile, $S \preceq T$ means that $T - S$ is PSD, and thus so is $A^{1/2}(T-S)A^{1/2}$. Because the trace of the product of two PSD matrices is non-negative (e.g. by Lemma 18), it follows that $\operatorname{tr}((A^{1/2}(T-S)A^{1/2})X) \geq 0$. Adding $\operatorname{tr}(A^{1/2}SA^{1/2}X)$ to both sides of this we get $\operatorname{tr}(A^{1/2}TA^{1/2}X) \geq \operatorname{tr}(A^{1/2}SA^{1/2}X)$. Because $A^{1/2}$ commutes with T and S we have $\operatorname{tr}(A^{1/2}TA^{1/2}X) = \operatorname{tr}(ATX)$ and $\operatorname{tr}(A^{1/2}SA^{1/2}X) = \operatorname{tr}(ASX)$, and so the result follows. ■

Lemma 21. *Suppose D is a matrix with real eigenvalues bounded strictly between -1 and 1. Define the operator $\Phi(X) = X - DXD^\top$. Then Φ has positive eigenvalues and is thus invertible. Moreover, we have*

$$\Phi^{-1}(X) = \sum_{i=0}^{\infty} D^i X (D^i)^\top .$$

And so if X is a PSD matrix then $\Phi^{-1}(X)$ is as well.

Proof The linear operator Φ can be expressed as a matrix using Kronecker product notation as $I - D \otimes D$. See Van Loan (2000) for a discussion of Kronecker products and their properties.

Because Kronecker products respect eigenvalue decompositions, the eigenvalues of $D \otimes D$ are given by $\{\lambda_i(D)\lambda_j(D) \mid 0 \leq i, j \leq n\}$. By hypothesis, the eigenvalues of D are real and bounded strictly between -1 and 1, and it therefore follows that the eigenvalues of $D \otimes D$ have the same property. From this it immediately follows that the eigenvalues of $I - D \otimes D$ are all > 0, and thus $I - D \otimes D$ is invertible.

Moreover, because of these bounds on the eigenvalues for $D \otimes D$, we have

$$(I - D \otimes D)^{-1} = \sum_{i=0}^{\infty} (D \otimes D)^i = \sum_{i=0}^{\infty} (D^i \otimes D^i) .$$

Translating back to operator notation this is

$$\Phi^{-1}(X) = \sum_{i=0}^{\infty} D^i X (D^i)^\top .$$

For any PSD matrix X this is a sum (technically a convergent series) of self-evidently PSD matrices, and is therefore PSD itself.

■

Proposition 22. *Let H_n be the n-th Harmonic number, defined by $H_n = \sum_{i=1}^{n} \frac{1}{i}$. For any integers $n_1 \geq n_2 \geq 1$ we have*

$$H_{n_1} - H_{n_2} \geq \log(n_1) - \log(\min\{n_2 + 1, n_1\}) .$$

Proof An inequality for H_n due to Young (1991) is

$$\log(n) + \gamma + \frac{1}{2(n+1)} \leq H_n \leq \log(n) + \gamma + \frac{1}{2n} ,$$

where γ is the Euler-Mascheroni constant.

In particular, we have

$$H_{n_1} \geq \log(n_1) + \gamma + \frac{1}{2(n_1+1)} \geq \log(n_1) + \gamma ,$$

and

$$H_{n_2+1} \leq \log(n_2+1) + \gamma + \frac{1}{2(n_2+1)} \leq \log(n_2+1) + \gamma + \frac{1}{n_2+1}\,,$$

which implies

$$H_{n_2} \leq \log(n_2+1) + \gamma\,.$$

Taking the difference of the two inequalities yields

$$H_{n_1} - H_{n_2} \geq \log(n_1) - \log(n_2+1)\,.$$

Noting that $\min\{n_2+1, n_1\} = n_1$ if and only if $n_1 = n_2$, and that in such a case we have $H_{n_1} - H_{n_2} = 0$, the result follows. ■

Proposition 23. *Suppose $0 \leq i \leq k-1$ for integers i and k, and b is a non-negative real number.*

For any non-negative integer i such that $b \leq i+a+1$ we have

$$\prod_{j=i}^{k-1}\left(1 - \frac{b}{j+a+1}\right) \leq \left(\frac{i+a+1}{k+a}\right)^b\,.$$

And for $b \leq a+2$ we have

$$\sum_{i=0}^{k-1} \frac{1}{(i+a)(i+a+1)^2} \prod_{j=i+1}^{k-1}\left(1 - \frac{b}{j+a+1}\right)^2 \leq \frac{\nu(a)k}{(k+a)^3}\,,$$

where $\nu(a) = (a+2)^3/(a(a+1)^2)$.

Proof It is a well-known fact that for $0 \leq y \leq 1$

$$1 - y \leq \exp(-y)\,.$$

For all $j \geq i$ we have $0 \leq \frac{b}{j+a+1} \leq 1$ (since $0 \leq b \leq i+a+1 \leq j+a+1$), and so

$$1 - \frac{b}{j+a+1} \leq \exp\left(-\frac{b}{j+a+1}\right)\,.$$

From this inequality and Proposition 22 it follows that

$$\begin{aligned}
\prod_{j=i}^{k-1}\left(1-\frac{b}{j+a+1}\right) &\leq \prod_{j=i}^{k-1}\exp\left(-\frac{b}{j+a+1}\right)\\
&= \exp\left(-b\sum_{j=i}^{k-1}\frac{1}{j+a+1}\right)\\
&= \exp\left(-b(H_{k+a}-H_{i+a})\right)\\
&\leq \exp\left(-b\left(\log(k+a)-\log(\min\{i+a+1,k+a\})\right)\right)\\
&= \left(\frac{\min\{i+a+1,k+a\}}{k+a}\right)^b\\
&\leq \left(\frac{i+a+1}{k+a}\right)^b.
\end{aligned}$$

Squaring both sides of the penultimate version of the above inequality it follows that

$$\begin{aligned}
&\sum_{i=0}^{k-1}\frac{1}{(i+a)(i+a+1)^2}\prod_{j=i+1}^{k-1}\left(1-\frac{b}{j+a+1}\right)^2\\
&\qquad\leq \sum_{i=0}^{k-1}\frac{1}{(i+a)(i+a+1)^2}\left(\frac{\min\{i+a+2,k+a\}}{k+a}\right)^{2b}\\
&\qquad= \frac{1}{(k+a)^{2b}}\sum_{i=0}^{k-1}\frac{(\min\{i+a+2,k+a\})^3}{(i+a)(i+a+1)^2}\left(\min\{i+a+2,k+a\}\right)^{2b-3}\\
&\qquad\leq \frac{1}{(k+a)^{2b}}\sum_{i=0}^{k-1}\nu(a)\left(\min\{i+a+2,k+a\}\right)^{2b-3}\\
&\qquad\leq \frac{\nu(a)}{(k+a)^{2b}}\sum_{i=0}^{k-1}(k+a)^{2b-3}=\frac{\nu(a)}{(k+a)^{2b}}k\cdot(k+a)^{2b-3}=\frac{\nu(a)k}{(k+a)^3},
\end{aligned}$$

where $\nu(a)=(a+2)^3/(a(a+1)^2)$, which is the second claimed inequality. ■

Appendix E. Proof of Corollary 2

Corollary. *Suppose that B_θ and B_γ are invertible matrices satisfying*

$$J_\zeta^\top B_\theta J_\zeta = B_\gamma$$

for all values of θ. Then the path followed by an iterative optimizer working in θ-space and using additive updates of the form $d_\theta = -\alpha B_\theta^{-1}\nabla h$ is the same as the path followed by an iterative optimizer working in γ-space and using additive updates of the form $d_\gamma = -\alpha B_\gamma^{-1}\nabla_\gamma h$, provided that the optimizers use equivalent starting points (i.e. $\theta_0=\zeta(\gamma_0)$), and that either

- *ζ is affine,*
- *or d_θ/α is uniformly continuous as a function of θ, d_γ/α is uniformly bounded (in norm), there is a C as in the statement of Theorem 1, and $\alpha \to 0$.*

Note that in the second case we allow the number of steps in the sequences to grow proportionally to $1/\alpha$ so that the continuous paths they converge to have non-zero length as $\alpha \to 0$.

Proof In the case where ζ is affine the result follows immediately from Theorem 1, and so it suffices to prove the second case.

We will denote by $\theta_0, \theta_1, ...$, and $\gamma_0, \gamma_1, ...$ the sequences of iterates produced by each optimizer. Meanwhile, $d_{\theta_0}, d_{\theta_1}, ...$ and $d_{\gamma_0}, d_{\gamma_1}, ...$ will denote the sequences of their updates.

By the triangle inequality

$$\begin{aligned}
\|\zeta(\gamma_{k+1}) - \theta_{k+1}\| &= \|\zeta(\gamma_k + d_{\gamma_k}) - (\theta_k + d_{\theta_k})\| \\
&= \|\zeta(\gamma_k + d_{\gamma_k}) - (\zeta(\gamma_k) + d_{\zeta(\gamma_k)}) + (\zeta(\gamma_k) + d_{\zeta(\gamma_k)}) - (\theta_k + d_{\theta_k})\| \\
&= \|\zeta(\gamma_k + d_{\gamma_k}) - (\zeta(\gamma_k) + d_{\zeta(\gamma_k)}) + (\zeta(\gamma_k) - \theta_k) + (d_{\zeta(\gamma_k)} - d_{\theta_k})\| \\
&\leq \|\zeta(\gamma_k + d_{\gamma_k}) - (\zeta(\gamma_k) + d_{\zeta(\gamma_k)})\| + \|\zeta(\gamma_k) - \theta_k\| + \|d_{\zeta(\gamma_k)} - d_{\theta_k}\| \,.
\end{aligned}$$

By Theorem 1, we can upper bound the first term on the RHS by $\frac{1}{2}C\sqrt{n}\|d_{\gamma_k}\|^2$. Using the hypothesis $\|d_\gamma/\alpha\| \leq D$ for all γ for some universal constant D, this is further bounded by $\frac{1}{2}\alpha^2 C D^2 \sqrt{n} \equiv \alpha^2 E$, where E is a universal constant. And by the hypothesized uniform continuity of d_θ/α (as a function of θ) there exists a universal constant U such that $\|d_{\zeta(\gamma_k)}/\alpha - d_{\theta_k}/\alpha\| \leq U\|\zeta(\gamma_k) - \theta_k\|$, which gives a bound of $\alpha U\|\zeta(\gamma_k) - \theta_k\|$ on the third term.

In summary, we now have

$$\|\zeta(\gamma_{k+1}) - \theta_{k+1}\| \leq \alpha^2 E + \|\zeta(\gamma_k) - \theta_k\| + \alpha U\|\zeta(\gamma_k) - \theta_k\| = \alpha^2 E + (1 + \alpha U)\|\zeta(\gamma_k) - \theta_k\| \,. \tag{50}$$

Starting from $\|\zeta(\gamma_0) - \theta_0\| = 0$ (which is true by hypothesis) and applying this formula recursively, we end up with the geometric series formula

$$\|\zeta(\gamma_k) - \theta_k\| \leq \alpha^2 E \sum_{i=0}^{k-1} (1 + \alpha U)^i = \alpha^2 E \left(\frac{(1 + \alpha U)^k - 1}{\alpha U} \right) .$$

Because each step scales as α, sequences of length T/α will converge to continuous paths of a finite non-zero length (that depends on T) as $\alpha \to 0$. Noting that $\lim_{\alpha \to 0}(1 + \alpha U)^{T/\alpha} = \exp(UT)$ (which is a standard result), it follows that the RHS of eqn. 50 converges to zero as $\alpha \to 0$ for $k = T/\alpha$, and indeed for all natural numbers $k \leq T/\alpha$. Thus we have for all $k \leq T/\alpha$

$$\lim_{\alpha \to 0} \|\zeta(\gamma_k) - \theta_k\| = 0 \,,$$

which completes the proof.

■

References

S. Amari. Theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 16(3):299–307, 1967.

S. Amari and H. Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical monographs*. Oxford University Press, 2000.

S.-I. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2): 251–276, 1998.

S.-i. Amari and A. Cichocki. Adaptive blind signal processing-neural network approaches. *Proceedings of the IEEE*, 86(10):2026–2048, 1998.

S.-i. Amari and H. Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2007.

L. Arnold, A. Auger, N. Hansen, and Y. Ollivier. Information-geometric optimization algorithms: A unifying picture via invariance principles. 2011.

J. Ba and D. Kingma. Adam: A method for stochastic optimization. In *ICLR*, 2015.

S. Becker and Y. LeCun. Improving the convergence of back-propagation learning with second order methods. In D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 29–37. Morgan Kaufmann, 1989.

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

A. Botev, H. Ritter, and D. Barber. Practical gauss-newton optimisation for deep learning. *arXiv preprint arXiv:1706.03662*, 2017.

L. Bottou and Y. LeCun. On-line learning for very large data sets. *Appl. Stoch. Model. Bus. Ind.*, 21(2):137–151, Mar. 2005. ISSN 1524-1904.

T. Cai, R. Gao, J. Hou, S. Chen, D. Wang, D. He, Z. Zhang, and L. Wang. A gram-gauss-newton method learning overparameterized deep neural networks for regression problems. *arXiv preprint arXiv:1905.11675*, 2019.

O. Chapelle and D. Erhan. Improved Preconditioner for Hessian Free Optimization. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

P. Chen. Hessian matrix vs. gauss–newton hessian matrix. *SIAM Journal on Numerical Analysis*, 49(4):1417–1435, 2011.

A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods*. SIAM, 2000.

Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

A. Défossez and F. Bach. Constant step size least-mean-square: Bias-variance trade-offs and optimal sampling distributions. 2014.

R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact newton methods. *SIAM Journal on Numerical analysis*, 19(2):400–408, 1982.

J. E. Dennis Jr and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16. Siam, 1996.

G. Desjardins, K. Simonyan, R. Pascanu, and K. Kavukcuoglu. Natural neural networks. In *Advances in Neural Information Processing Systems*, pages 2071–2079, 2015.

S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.

J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

N. Flammarion and F. Bach. From averaging to acceleration, there is only a step-size. 2015.

R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

R. Frostig, R. Ge, S. M. Kakade, and A. Sidford. Competing with the empirical risk minimizer in a single pass. 2014.

R. Grosse and R. Salakhudinov. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2304–2313, 2015.

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Maching Learning*, 69(2-3):169–192, Dec. 2007.

T. Heskes. On "natural" learning and pruning in multilayered perceptrons. *Neural Computation*, 12(4):881–901, 2000.

S. Hochreiter, F. F. Informatik, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. In J. Kolen and S. Kremer, editors, *Field Guide to Dynamical Recurrent Networks*. IEEE Press, 2000.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, 2013.

H. Kushner and G. G. Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

N. Le Roux and A. Fitzgibbon. A fast natural Newton method. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

N. Le Roux, P.-a. Manzagol, and Y. Bengio. Topmoumoute online natural gradient algorithm. In *Advances in Neural Information Processing Systems 20*, pages 849–856. MIT Press, 2008.

N. Le Roux, M. W. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2012.

Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 546–546, 1998.

J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pages 8570–8581, 2019.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

N. Loizou and P. Richtárik. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *arXiv preprint arXiv:1712.09677*, 2017.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

J. Martens. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

J. Martens. *Second-order optimization for neural networks*. PhD thesis, University of Toronto, 2016.

J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1033–1040, 2011.

J. Martens and I. Sutskever. Training deep and recurrent networks with Hessian-free optimization. In *Neural Networks: Tricks of the Trade*, pages 479–535. Springer, 2012.

J. Martens, I. Sutskever, and K. Swersky. Estimating the hessian by backpropagating curvature. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. *Numerical analysis*, pages 105–116, 1978.

J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

E. Moulines and F. R. Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

N. Murata. A statistical study of on-line learning. In D. Saad, editor, *On-line Learning in Neural Networks*, pages 63–92. Cambridge University Press, 1998.

S. G. Nash. Preconditioning of truncated-newton methods. *SIAM Journal on Scientific and Statistical Computing*, 6(3):599–616, 1985.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 2. ed. edition, 2006.

Y. Ollivier. Riemannian metrics for neural networks i: feedforward networks. *Information and Inference*, 4(2):108–153, 2015.

Y. Ollivier et al. Online natural gradient as a kalman filter. *Electronic Journal of Statistics*, 12(2):2930–2961, 2018.

J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

H. Park, S.-I. Amari, and K. Fukumizu. Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks*, 13(7):755–764, Sept. 2000.

R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. In *ICLR*, 2014.

J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7):1180–1190, 2008.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4), July 1992.

D. W. Ruck, S. K. Rogers, M. Kabrisky, P. S. Maybeck, and M. E. Oxley. Comparative analysis of backpropagation and the extended kalman filter for training multilayer perceptrons. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):686–691, 1992.

T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14, 2002.

P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent prop-a formalism for specifying selected invariances in an adaptive network. In *Advances in neural information processing systems*, pages 895–903, 1992.

S. Singhal and L. Wu. Training multilayer perceptrons with the extended kalman algorithm. In *Advances in neural information processing systems*, pages 133–140, 1989.

T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

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

A. N. Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198, 1943.

C. F. Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1-2):85–100, 2000.

O. Vinyals and D. Povey. Krylov subspace descent for deep learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

S.-D. Wang, T.-S. Kuo, and C.-F. Hsu. Trace bounds on the solution of the algebraic matrix Riccati and Lyapunov equation. *Automatic Control, IEEE Transactions on*, 31 (7):654–656, 1986.

R. M. Young. Euler's constant. *The Mathematical Gazette*, 75(472):187–190, 1991.

M. D. Zeiler. ADADELTA: An adaptive learning rate method. 2013.

G. Zhang, J. Martens, and R. B. Grosse. Fast convergence of natural gradient descent for over-parameterized neural networks. In *Advances in Neural Information Processing Systems*, pages 8080–8091, 2019.