Predictive Rescaling of Quasi-Newton Methods

Philippe Tillet PTILLET@G.HARVARD.EDU
Harvard University
John A. Paulson School of Engineering and Applied Science

Abstract

In this work, we provide a theoretical and empirical analysis for various statistical properties of Quasi-Newton methods. We prove that re-scaling Quasi-Newton inverse Hessian approximations can be seen as Maximum-A-Posteriori (MAP) estimation under a certain Bayesian Gamma-Wishart model. This observation leads to a refined version of the BFGS algorithm that outperforms the traditional formulation for various objective functions.

1. Introduction

Unconstrained non-linear optimization algorithms are ubiquitous in Science and Engineering, with applications ranging from physics to climate science. In order to achieve fast and proper convergence, most such algorithms rely on approximations often designed decades ago, which could hopefully be refurbished using modern statistical techniques.

Much work has been done in the field of stochastic optimization for noisy loss functions. For example, Duchi et al. proposed ADAGRAD (3) – an adaptive subgradient method for online learning – while Mahsereci et al. derived probabilistic line searches for stochastic optimization (2). Not to mention the significant amount of research aiming at designing adaptive schedules for Stochastic Gradient Descent (4; 5).

This being said, relatively little has been done to integrate concepts from statistics for the optimization of deterministic loss functions such as those encountered in many fields outside of Machine Learning.

This work explores the extent to which statistical techniques can be leveraged for the benefits of deterministic optimization. Section 2 recalls the fundamental principles underlying unconstrained optimization theory – and in particular Quasi-Newton methods. In Section 3, we prove that rescaling inverse Hessian approximations can be seen as MAP estimation under a certain Gamma-Wishart Bayesian model. This observation leads to a new BFGS algorithm that is stated formally in Section 4 and evaluated in Section 5, where we observe very encouraging results. Section 6 provides concluding remarks as well as possibilities of future work.

2. Unconstrained Optimization

The goal of unconstrained (nonlinear) optimization is to minimize some convex objective function f with respect to a vector $\mathbf{x} \in \mathbb{R}^N$ of unconstrained real variables:

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} f(\mathbf{x})$$

This problem can be solved iteratively by using so-called *line-search methods*. They work by picking – at each iteration k – a descent direction $\mathbf{p}_k$ along which to move, and jump by a factor α_k along $\mathbf{p}_k$ in order to build a new iterate $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$.

An effective choice for $\mathbf{p}_k$ can be derived from the second-order Taylor expansion of f at $\mathbf{x}_k$:

$$f(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + \mathbf{p}^T \nabla f(\mathbf{x}_k) + \mathbf{p}^T \nabla^2 f(\mathbf{x}_k)\mathbf{p}$$

Setting the gradient of this model to 0 leads to the so-called *Newton* direction:

$$\mathbf{p}_k = -\nabla^2 f(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k)$$

For the purpose of the subsequent analysis, it is fundamental to note that this direction is *well scaled* by construction: $\alpha_k = 1$.

Unfortunately, the computation of $\nabla^2 f(\mathbf{x}_k)$ – let alone its inversion – is often prohibitive in practice, hence the emergence of Quasi-Newton methods, which aim at building *approximations* B_k of $\nabla^2 f(\mathbf{x}_k)$ using first-order curvature information only. Specifically, B_k is

Final Project for the CS281 class

designed to mimic some important properties of the Hessian matrix, and should in satisfy the *secant equation*:

$$B_k \mathbf{s}_k = \mathbf{y}_k$$

Where

$$\mathbf{s}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$$
$$\mathbf{y}_k = \nabla f(x_k) - \nabla f(x_{k-1})$$

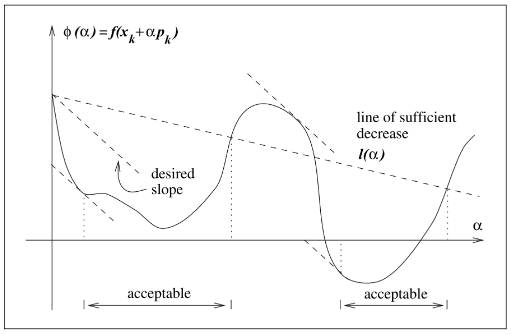
The BFGS formula, named after its inventors (Broyden, Fletcher, Goldfarb, Shanno), defines an update formula for B_{k+1} given B_k that preserves positive-definiteness and is guaranteed to satisfy the secant equation for all k:

$$B_{k+1} = B_k - \frac{B_k \mathbf{s}_k \mathbf{s}_k^T B_k}{\mathbf{s}_k^T B_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{s}_k^T \mathbf{s}_k} \tag{1}$$

This formula can be rearranged to update an inverse Hessian approximation $H_k = B_k^{-1}$ directly (using the Sherman - Morrison formula), or even to compute $\mathbf{p}_k = -H_k \nabla_f(\mathbf{x}_k)$ *directly* using the limited memory $\mathbf{y}_{k-m}, \cdots, \mathbf{y}_k$ and $\mathbf{s}_{k-m}, \cdots, \mathbf{s}_k$ [L-BFGS].

The resulting directions are usually *badly* scaled and often require carefully chosen initial Hessian approximations H_0 as well as $\alpha_k \neq 1$. Specifically, convergence is ensured if and only if α_k satisfies the so-called Wolfe-Powell conditions:

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(x_k)^T \mathbf{p}_k \tag{2}$$
$$|\nabla f(\mathbf{x}_k) + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| \leq c_2 |\nabla f(x_k)^T \mathbf{p}_k| \tag{3}$$
$$0 < c_1 < c_2 < 1$$



$\phi(\alpha) = f(x_k + \alpha p_k)$
line of sufficient decrease
$l(\alpha)$
desired slope
α
acceptable
acceptable

Figure 1. Graphical interpretation of the Wolfe-Powell conditions

Fig. 1 provides a graphical interpretation of the said conditions: (2) constrains f to decrease sufficiently along $\mathbf{p}_k$, while (3) requires the absolute slope of the line-search to be below a certain threshold.

3. MAP estimation of the Hessian Matrix

As mentioned previously, the true Newton direction is such that $\alpha_k = 1$ generally satisfies the Wolfe-Powell conditions directly (provided that a quadratic model is a good enough local approximation of f at $\mathbf{x}_k$). This suggests that observing of α_k — and more specifically how it differs from 1 — could provide some valuable information regarding the "quality" of the inverse Hessian approximation H_k.

In this section, we prove that this information can be leveraged through a Bayesian Gamma-Wishart model whose Maximum-A-Posteriori (MAP) has the simple form

$$\Sigma^{(\text{MAP})} = \gamma H \ (\gamma \in \mathbb{R})$$

where $\Sigma = \nabla f(\mathbf{x})^{-1}$ is a latent variable whose posterior likelihood is

$$P(\Sigma|\alpha, H) \propto P(\alpha|H, \Sigma) P(\Sigma|H)$$
$$\log P(\Sigma|\alpha, H) = \text{cst} + \log P(\alpha|H, \Sigma) + \log P(\Sigma|H)$$

3.1. Likelihood function

The distribution of the ideal step-size α given H and Σ can be modeled as:

$$P(\alpha|H, \Sigma) \sim \Gamma\Big(N, \frac{1}{\text{tr}(H\Sigma^{-1})}\Big) \tag{4}$$

Note that

$$\lim_{H \to \Sigma} P(\alpha|H, \Sigma) = \Gamma\Big(N, \frac{1}{N}\Big)$$

which has the following statistical properties:

$$\mathbb{E}[\alpha|H, \Sigma] = 1$$
$$\arg\max P(\alpha|H, \Sigma) = \frac{N-1}{N}$$
$$\text{Var}[\alpha|H, \Sigma] = \frac{1}{N}$$

In other words, this distribution converges to the deterministic choice $\alpha = 1$ as the approximation of the inverse Hessian becomes ideal ($H \to \Sigma$) and the dimension of the input space grows larger ($N \to \infty$).

3.2. Prior Distribution

Recall that the inverse Hessian matrix of the log-likelihood for a normally distributed random vector is simply given by the covariance matrix of its components. Guided by the popularity of inverse-Wishart priors for this covariance matrix in Bayesian inference, we have chosen

$$P(\Sigma|H) \sim W^{-1}(\lambda H, \nu), \quad \nu > N+1. \qquad (5)$$

When $\lambda = \nu + N + 1$, this prior has the following desirable properties:

$$\mathbb{E}[\Sigma|H] = \frac{\nu+N+1}{\nu-N-1}H$$
$$\arg\max p(\Sigma|H) = H$$

Which makes sense, as H is the best guess one can make for Σ prior to observing α.

3.3. Maximum A Posteriori (MAP) estimation

The log-likelihood is:

$$\log P(\alpha|H,\Sigma) = \text{cst} + N\log\text{tr}(H\Sigma^{-1}) - \alpha\text{tr}(H\Sigma^{-1})$$

and its gradient

$$\frac{\partial \log P(\alpha|H,\Sigma)}{\partial\Sigma} = \Big(\frac{p}{\text{tr}(H\Sigma^{-1})} - \alpha\Big)\Sigma^{-1}H\Sigma^{-1}$$

Note that the maximum likelihood is obtained for

$$\Sigma_{\text{ML}} = \alpha H$$

On the other hand, the log-prior is given by

$$\log P(\Sigma|H) = -\frac{1}{2}\Big((\nu+p+1)\log|\Sigma| + \text{tr}(\lambda H\Sigma^{-1})\Big)$$

And its gradient is:

$$\begin{aligned}\frac{\partial \log P(\Sigma|H)}{\partial\Sigma} &= -\frac{1}{2}\Big((\nu+p+1)\Sigma^{-1} - \lambda(\Sigma^{-1}H\Sigma^{-1})^T\Big)\\ &= -\frac{1}{2}\Big((\nu+p+1)\Sigma^{-1} - \lambda\Sigma^{-1}H\Sigma^{-1}\Big)\\ &= \frac{1}{2}\Sigma^{-1}(\lambda H - (\nu+p+1)\Sigma)\Sigma^{-1}\end{aligned}$$

The gradient of the posterior likelihood is therefore nullified if and only if

$$\Big(\frac{p}{\text{tr}(H\Sigma^{-1})} + \frac{\lambda}{2} - \alpha\Big)H = \frac{(\nu+p+1)}{2}\Sigma$$

which has a solution of the form

$$\Sigma_{MAP} = \gamma H \qquad (6)$$

Where

$$2\gamma + \lambda - 2\alpha = (\nu+N+1)\gamma$$
$$\gamma = \frac{\lambda - 2\alpha}{\nu+N-1}$$

Note that choosing $\lambda = \nu+N+1$ does not only lead to an intuitive prior on Σ but also to interesting posterior properties for Σ_{MAP}:

1. The MAP preserves positive-definiteness (in practice $\nu+N+1 > 2N+1 > 2\alpha$)
2. The MAP is equal to H when $\alpha = 1$, and increases (resp. decreases) in magnitude when $\alpha < 1$ (resp. $\alpha > 1$).
3. $\nu + N$ is an intuitive regularizer that prevents γ from becoming too big on large-scale problems.

4. Predictive Rescaling of BFGS

The previous analysis and its remarkably intuitive result suggest a simple refinement of the BFGS algorithm, whereby the new inverse Hessian estimate H_{k+1} would derived not from H_k but from

$$\Sigma_{MAP} = \frac{\nu+N+1-2\alpha_k}{\nu+N-1}H_k$$

Note that good values for $\nu > N+1$ are problem-specific, but in practice, choosing $\nu = N+2$ yields good results.

The update equation for the inverse Hessian approximation becomes:

$$\begin{aligned}H_{k+1} = \Sigma_{\text{MAP}} &+ \frac{(\mathbf{s}_k^T\mathbf{y}_k + \mathbf{y}_k^T\Sigma_{\text{MAP}}\mathbf{y}_k)\mathbf{s}_k\mathbf{s}_k^T}{(\mathbf{s}_k^T\mathbf{y}_k)^2}\\ &- \frac{\Sigma_{\text{MAP}}\mathbf{y}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{y}_k^T\Sigma_{\text{MAP}}}{\mathbf{s}_k^T\mathbf{y}_k}\end{aligned} \qquad (7)$$

This modification is remarkably simple and does not effectively increase the computational complexity of the BFGS update (it adds in exactly $2N^2$ memory operations and N^2 multiplications)

While $\Sigma_{\text{MAP}} \approx H_k$ for very large values of N, the changes made by our rescaled version of BFGS are multiplicative and may still accumulate over time to have meaningful consequences at later iterations.

The full algorithm is shown in Algorithm 1. It is essentially like the standard BFGS algorithm, apart from the fact that H_k is transparently replaced by Σ_k.

Various techniques can be used to implement the Line-Search procedure, and our implementation uses the cubic interpolation techniques described in (1).

5. Numerical Experiments

This section provides an empirical evaluation of the performance of Algorithm 1 on two non-trivial objec-

Algorithm 1 Predictively Rescaled BFGS

```
Input: Objective function $f$, initial guess $\mathbf{x}_0 \in \mathbb{R}^N$
Output: $\mathbf{x}^\star = \arg\min_{\mathbf{x}} f(\mathbf{x})$
$k \leftarrow 0$
$H_0 \leftarrow I$
$\Sigma_0 \leftarrow H_0$
for $k$=0 to maxit do
    if Convergence then
        break
    if $k > 0$ then
        $\Sigma_k \leftarrow \alpha_{k-1} H_k$
    $H_{k+1} \leftarrow \text{Update}(\Sigma_k)$                     (7)
    $\mathbf{p}_k \leftarrow -H_{k+1} \nabla f(\mathbf{x}_k)$
    $\alpha_k \leftarrow \text{LineSearch}(\mathbf{x}_k, \mathbf{p}_k)$        (2), (3)
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$
return $\mathbf{x}_k$
```

tive functions. We demonstrate that rescaling the inverse Hessian approximation using (6) leads to significantly better convergence.

5.1. Objective Functions

We considered two deterministic objective functions:

1. The generalized N-dimensional Rosenbrock function:

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$$

   And its gradient:

$$\frac{\partial f}{\partial x_1} = -400x_1(x_2 - x_1^2) - 2(1 - x_1)$$
$$\frac{\partial f}{\partial x_i} = 200(x_i - x_{i-1}^2) - 400x_i(x_{i+1} - x_i^2) - 2(1 - x_i)$$
$$\frac{\partial f}{\partial x_N} = -400x_1(x_2 - x_1^2) - 2(1 - x_1)$$

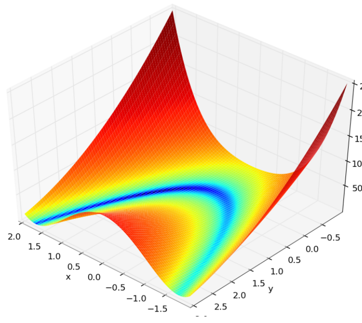


Figure 2. 2D Rosenbrock Function

   As shown in Fig. 2, it is characterized by a long and narrow valley which tends to make optimization difficult.

   In this work we use $N = 100$.

2. The (deterministic) loss of a deep but small MultiLayer Perceptron (MLP) on the MNIST dataset using batch processing. This Neural network – whose architecture is shown in Fig. 3 – is composed of 3 hidden layers (sigmoid activation functions) of 5 neurons each, supplemented by a softmax output layer.

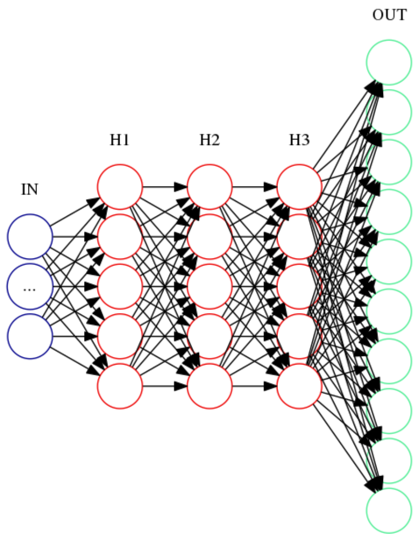


Figure 3. Architecture of the MLP considered in this paper

   We acknowledge that our algorithm will not offer a practical solution to the optimization neural network architectures, but the corresponding objective function is known to have a non-trivial landscape which would be interesting to leverage.

5.2. Implementation

We used a state-of-the-art line-search procedure using cubic interpolation (1) and the popular "soft" Wolfe-Powell constants:

$$c_1 = 10^{-4}$$
$$c_2 = 0.9$$

We integrated Algorithm 1 in the C++ Unconstrained Minimization Template Library using BLAS (UMinTL, Philippe Tillet, https://github.com/ptillet/umintl/tree/cs-281).

Note that this library was first written as part of my

MSc-Thesis, and not for the CS281 class. Necessary changes were made to the C++ codebase in order to incorporate the rescaling technique and generate both the Rosenbrock and the MNIST MLP data for the plots shown in section 5.3.

5.3. Results

We now compare the performance of our algorithm against that of a standard BFGS implementation for the two objective functions mentioned above. Specifically, we measure the evolution of the objective function's value with respect to the number of function evaluations.

5.3.1. Rosenbrock

Figure 4 shows the performance of the rescaled BFGS algorithm on the generalized Rosenbrock function starting from $\mathbf{x}_0 \in [0,1]^N$ for $N = 100$.

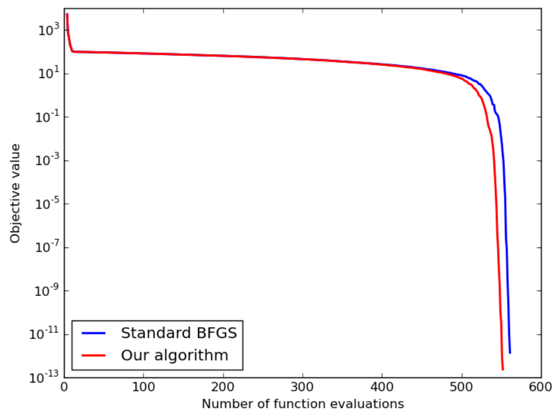


Figure 4. Training performance of our algorithm on the Generalized Rosenbrock Function.

Rescaling the inverse Hessian approximation does not seem to provide much benefit in the earlier stage of the optimization procedure. However, as the iterate approaches the true solution $\mathbf{x}^\star = [-1, 1, \cdots, 1]$, the MAP estimation becomes beneficial and our algorithms achieves faster convergence than the traditional approach.

Overall, our algorithm converges in 551 function evaluations (versus 561 for BFGS).

5.3.2. MLP

We now evaluate the performance of our algorithm for the network architecture presented in Figure 3 on the MNIST dataset for a cross-entropy loss function. The gradient was computed using backpropagation.

Figure 5 shows the evolution of the training objective (i.e., loss function). It appears that both algorithms still behave similarly at the beginning of the optimization process, before the benefits of the MAP estimation kicks in and provides a clear advantage to our algorithm.

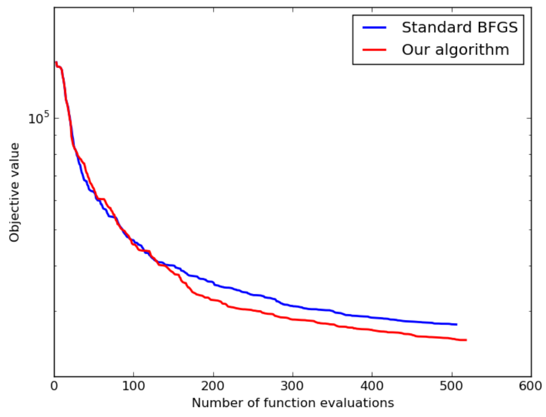


Figure 5. Training performance of our algorithm on MNIST using the above MLP architecture.

The same behavior is observed when measuring the classification accuracy of our MLP on the MNIST testing set (Figure 6): the better performance of our rescaled algorithm suggests that it does a better job at adapting the pecularities of the underlying landscape. After 500 forward-backward passes, our algorithm achieves a misclassification rate of 12.0% against 13.7% for BFGS.

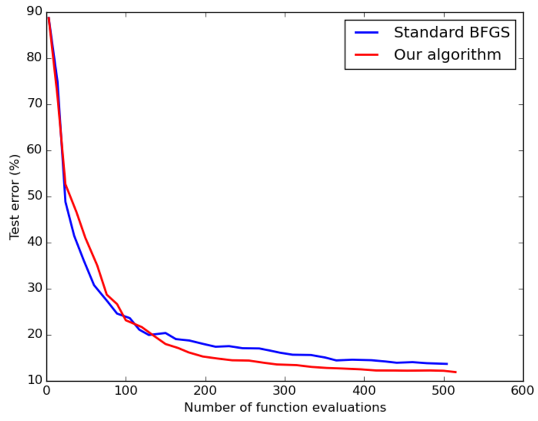


Figure 6. Accuracy of our algorithm on MNIST using the above MLP architecture.

6. Conclusions

To our knowledge, our work is the first to provide a theoretical ground for rescaling inverse Hessian approximations based on line-search information. We prove that such rescaling corresponds to MAP estimation under a certain Bayesian model, and show that the resulting algorithm outperforms both quantitatively and qualitatively traditional Quasi-Newton methods for two non-trivial objective functions.

A potential shortcoming of our approach is that the rescaling factor γ_k does not depend on the local curvature of f at $\mathbf{x}_k$. Equation (7) suggests that such curvature information could be partly captured by $\mathbf{s}_k$ and $\mathbf{y}_k$. The integration of these two vectors in the Bayesian model presented in Section 3 could result in even better performance.

Other possibilities of future work include extensions of our model to large-scale algorithms such as the Newton-CG method or L-BFGS.

References

[1] J. Nocedal and S. J. Wright, *Numerical Optimization* (2006)

[2] M. Mahsereci and P. Hennig, *Probabilistic Line Searches for Stochastic Optimization* (2015)

[3] J. Duchi, E. Hazan and Y. Singer *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization* (2011)

[4] V.P. Plagianakos *Learning rate adaptation in stochastic gradient descent* (2001)

[5] T. Schaul *No More Pesky Learning Rates* (2012)