TOWARDS STOCHASTIC CONJUGATE GRADIENT METHODS

Nicol N. Schraudolph *Thore Graepel*

schraudo@inf.ethz.ch graepel@inf.ethz.ch

Institute of Computational Science
ETH Zürich, Switzerland
http://www.icos.ethz.ch/

To appear in *Proc. 9th Intl. Conf. Neural Information Processing*, Singapore 2002

ABSTRACT

The method of conjugate gradients provides a very effective way to optimize large, deterministic systems by gradient descent. In its standard form, however, it is not amenable to stochastic approximation of the gradient. Here we explore a number of ways to adopt ideas from conjugate gradient in the stochastic setting, using fast Hessian-vector products to obtain curvature information cheaply. In our benchmark experiments the resulting highly scalable algorithms converge about an order of magnitude faster than ordinary stochastic gradient descent.

1. INTRODUCTION

For the optimization of large, differentiable systems, methods that require the inversion of a curvature matrix (Levenberg-Marquardt [1, 2]), or the storage of an iterative approximation of that inverse (quasi-Newton methods such as BFGS), are prohibitively expensive. Conjugate gradient techniques [3], which are capable of exactly minimizing a d-dimensional unconstrained quadratic problem in d iterations without requiring explicit knowledge of the curvature matrix, have become the method of choice for such cases. Their rapid convergence, however, breaks down when the function to be optimized is noisy, as occurs in online (stochastic) gradient descent problems. Here the state of the art is simple gradient descent, coupled with adaptation of local step size parameters. The most advanced of these methods, SMD [4, 5], uses curvature matrix-vector products that can be obtained efficiently and automatically [5, 6]. Here we use the same curvature matrix-vector products to adopt some ideas from conjugate gradient in the stochastic setting.

2. CONJUGATE GRADIENT

Basic concepts. The fundamental idea of conjugate gradient methods is to successively minimize the parameters $\vec{w}$ of a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ along a set of d non-interfering search directions. Two directions $\vec{v}_i$ and $\vec{v}_j$ are non-interfering if they are *conjugate*, that is,

$$i \neq j \;\Rightarrow\; \vec{v}_i^T H \vec{v}_j \;=\; 0\,, \tag{1}$$

where $H = \nabla^2 f$ is the Hessian of the system. A set of mutually conjugate directions is produced by the iteration

$$\vec{v}_t \;=\; \alpha\, \vec{v}_{t-1} \,-\, \vec{g}_t\,, \tag{2}$$

where $\vec{g}_t = \nabla f(\vec{w}_t)$ denotes the gradient at the t-th iteration, and

$$\alpha \;=\; \frac{||\,\vec{g}_t||^2}{||\,\vec{g}_{t-1}||^2}\,. \tag{3}$$

A d-dimensional quadratic bowl is minimized exactly after d iterations of (2), starting with $\vec{v}_0 = -\vec{g}_0$. This process depends upon the function being minimized exactly along each given search direction. In a quadratic bowl, this is achieved by adjusting the parameters $\vec{w}$ via

$$\vec{w}_{t+1} \;=\; \vec{w}_t \,-\, \beta\, \vec{v}_t\,, \tag{4}$$

$$\text{where} \quad \beta \;=\; \frac{\vec{g}_t^T \vec{v}_t}{\vec{v}_t^T H \vec{v}_t}\,. \tag{5}$$

Nonlinear problems. For conjugate gradient on nonlinear optimization problems, an explicit line minimization is usually employed to set the step size β. However, it has been shown that a local quadratic approximation (*i.e.*, the β given in (5) above) can be very effective here, provided that suitable trust-region modifications are made (scaled conjugate gradient [7]). Of course a nonlinear problem will not be minimized exactly after d iterations, but the method can be restarted at that point by resetting $\vec{v}$ to the current gradient.

Fast curvature matrix-vector products. It should be noted that the calculation of β in (5) does *not* require explicit storage of the Hessian, which would be $O(d^2)$. There are ways to calculate the product of the Hessian with an arbitrary vector at a cost comparable to that of obtaining the gradient, which typically is $O(d)$ [6]. The same goes for other measures of curvature, such as the Gauss-Newton approximation of the Hessian, and the Fisher information matrix [5]. *Algorithmic differentiation* software[1] provides generic implementations of the building blocks from which these algorithms are constructed.

3. STOCHASTIC GRADIENT DESCENT

Empirical loss functions are often minimized using noisy measurements of gradient (and, if applicable, curvature) obtained on small, random (hence "stochastic") subsamples ("batches") of data, or even individual data points (batch size = 1). This is done for reasons of computational efficiency on large, redundant data sets, and out of necessity when adapting online to a continual stream of noisy, potentially non-stationary data.

Simple stochastic gradient. Unfortunately conjugate gradient techniques are not designed to cope with noisy gradient and curvature information, and hence tend to diverge in stochastic settings. Occasional reports to the contrary invariably concern regimes that we would term near-deterministic (i.e., large batch sizes). For relatively small systems, the extended Kalman filter is a robust alternative, albeit at a cost of $O(d^2)$ per iteration. Large stochastic systems are therefore still often optimized using simple (first-order) gradient descent.

Local step size adaptation. The convergence of simple stochastic gradient descent can be improved by adjusting a local step size for each system parameter. The most advanced of these algorithms, *stochastic meta-descent* (SMD), adapts local step sizes by a dual gradient descent procedure that uses fast curvature matrix-vector products [4, 5].

Adaptive momentum. This method adapts the system parameters $\vec{w}$ by an iterative stochastic approximation of second-order gradient steps. As formulated originally [8] it was unfortunately stable only for linear systems. We have recently succeeded in making it robust for nonlinear optimization problems by incorporating a trust-region modification [9]. This algorithm also uses fast curvature matrix-vector products.

4. TOWARDS STOCHASTIC CONJUGATE GRADIENT: THREE ALGORITHMS

We now seek to improve upon the above stochastic gradient methods by adopting certain ideas from conjugate gradient. The resulting three new algorithms all use fast Hessian-gradient products.

[1] See `http://www-unix.mcs.anl.gov/autodiff/`

Pairwise conjugation of gradients. One of the reasons why conjugate gradient breaks down in a stochastic setting is that the noise makes it impossible to maintain the conjugacy of search directions over multiple iterations. Instead of trying to construct a set of conjugate directions, then, let us just move down the gradient:

$$\vec{w}_{t+1} \;=\; \vec{w}_t \;-\; \beta\, \vec{g}_t \tag{6}$$

with a step size aimed at conjugating successive steps:

$$\beta \;=\; \frac{\vec{g}_t^{\,T} H \vec{g}_t}{||H\vec{g}_t||^2} \tag{7}$$

This choice of β achieves pairwise conjugation of gradients (*i.e.*, $\vec{g}_{t+1}^{T} H \vec{g}_t \;=\; 0$) in a deterministic quadratic bowl, as can easily be verified using the identity $\vec{g} = H\vec{w}$.

Diagonal Conditioner. In systems where the Hessian has some sparsity, it is advantageous to be able to *decouple* the step sizes for individual subspaces. To this end we construct a diagonal conditioner from (7) by replacing the inner products by Hadamard (component-wise) products and division. In order to remove the poles resulting from zero components in the denominator, we then average both numerator and denominator over the trajectory, obtaining

$$\beta \;=\; \mathrm{diag}\left(\frac{\langle \mathrm{diag}(\vec{g}_t) H \vec{g}_t\rangle_\gamma}{\langle \mathrm{diag}(H\vec{g}_t) H \vec{g}_t\rangle_\gamma}\right) \vec{g}_t\,, \tag{8}$$

where $\langle\cdot\rangle$ denotes exponential averaging, *i.e.*,

$$\langle u_t\rangle_\gamma \;=\; u_t \;+\; \gamma\, \langle u_{t-1}\rangle_\gamma\,. \tag{9}$$

Compared to (7) above, we have effectively replaced averaging over search space dimensions with leaky averaging over the optimization trajectory.

Stabilized conjugate gradient. Our third algorithm seeks to preserve as much of the conjugate gradient machinery as possible while stabilizing it for use in a stochastic setting. We focus on the calculation of α in (3), clearly a weak point: when gradients are noisy and line minimizations inaccurate, we could easily have $||\vec{g}_t|| \gg ||\vec{g}_{t-1}||$, resulting in inordinately large values of α. We guard against this by adding the correction term $|\,\vec{g}_t^{\,T}\vec{v}_{t-1}|$ to the denominator, replacing (3) with

$$\alpha \;=\; \frac{||\,\vec{g}_t||^2}{||\,\vec{g}_{t-1}||^2 \;+\; |\,\vec{g}_t^{\,T}\vec{v}_{t-1}|} \tag{10}$$

Note that an accurate line minimization along $\vec{v}_{t-1}$ ensures that $\vec{g}_t^{\,T}\vec{v}_{t-1} = 0$, so in that case the above reduces to standard conjugate gradient. The correction term comes into play only to the extent that the line minimization fails and the gradient grows in magnitude, reducing the impact on α of this pathological situation.

5. QUADRATIC BOWL EXPERIMENTS

Deterministic bowl. The d-dimensional quadratic bowl provides us with a simplified test setting in which every aspect of the optimization can be controlled. It is defined by the unconstrained problem of minimizing with respect to d parameters $\vec{w}$ the function

$$f(\vec{w}) \;=\; \frac{1}{2}\,\vec{w}^T J J^T\,\vec{w}\,, \tag{11}$$

where the Jacobian J is a $d \times d$ matrix of our choosing. Note that by definition the Hessian $H = JJ^T$ is positive semidefinite and constant with respect to the parameters $\vec{w}$; these are the two crucial simplifications compared to more realistic, nonlinear problems. The gradient here is

$$\vec{g} \;=\; \nabla f(\vec{w}) \;=\; H\vec{w}\,. \tag{12}$$

Stochastic bowl. The stochastic optimization problem analogous to the deterministic one above is the minimization (again with respect to $\vec{w}$) of the function

$$f(\vec{w}, X) \;=\; \frac{1}{2\,b}\,\vec{w}^T J\,X X^T J^T \vec{w}\,, \tag{13}$$

where $X = [\vec{x}_1, \vec{x}_2, \ldots \vec{x}_b]$ is a $d \times b$ matrix collecting a *batch* of b random input vectors to the system, each drawn i.i.d. from a normal distribution: $\vec{x}_i \sim N(0, I)$. This means that $E(XX^T) = b\,I$, so that in expectation this is identical to the deterministic formulation:

$$E_X[f(\vec{w}, X)] \;=\; \frac{1}{2\,b}\,\vec{w}^T J\,E(XX^T) J^T \vec{w} \;=\; f(\vec{w})\,. \tag{14}$$

The optimization problem is harder here since the objective can only be probed by supplying stochastic inputs to the system, giving rise to the noisy estimates

$$\tilde{H} \;=\; b^{-1} J\,X X^T J^T \tag{15}$$

$$\text{and} \quad \tilde{\vec{g}} = \nabla_{\vec{w}} f(\vec{w}, X) = \tilde{H}\vec{w} \tag{16}$$

of Hessian and gradient, respectively. The degree of stochasticity is determined by the batch size b; the system becomes deterministic in the limit as $b \to \infty$.

Choice of Jacobian. We choose J so as to make the quadratic bowl a) very ill-conditioned, and b) partially decoupled. In other words, the Hessian should have a) eigenvalues of widely differing magnitude, and b) eigenvectors of intermediate sparsity. These conditions are intended to model the mixture of axis-aligned and oblique "narrow valleys" that is characteristic of multi-layer perceptrons. We achieve them by imposing some sparsity on the notoriously ill-conditioned *Hilbert matrix*, defining

$$(J)_{ij} \;=\; \begin{cases} \frac{1}{i+j-1} & \text{if } i \bmod j \;=\; 0 \;\vee\; j \bmod i \;=\; 0\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{17}$$

We call the optimization problem resulting from setting J to this matrix the *modified Hilbert bowl*.

Experiments. We tested the three proposed new algorithms in deterministic, intermediate (batch size 5), and fully stochastic (batch size 1) settings on the modified Hilbert bowls of dimension 5 and 20, whose Hessians have condition numbers of $4.9 \cdot 10^3$ and $4.2 \cdot 10^5$, respectively. For comparison, we also tested standard conjugate gradient—equations (2), (3), (4), and (5)—and simple gradient descent with the global step size set to the inverse of the largest eigenvalue of the Hessian. We repeated each experiment for 100 different random starting points on the unit hypersphere, and report the geometric mean loss over those repetitions.

Results (Fig. 1). Unsurprisingly, both conjugate gradient algorithms show identical, strong performance in the deterministic setting but diverge in the fully stochastic one. In the intermediate regime, however, standard conjugate gradient gets stuck early on (flat curves), whereas our stabilized variant continues to perform, converging an order of magnitude faster than simple gradient descent.

The pairwise conjugation method is the best algorithm in the fully stochastic setting, where it converges an order of magnitude faster than simple gradient descent ($d = 5$), and gives reliable performance where the latter diverges ($d = 20$). We note that its performance in fact *improves* with increasing stochasticity (from $b = 5$ to $b = 1$)—a remarkable feat. Finally, our diagonally conditioned method (with $\gamma = 0.1$) also converges reliably, if somewhat more slowly than pairwise conjugation. We expect that the decoupling will make diagonal conditioning advantageous on sparser problems.

6. SUMMARY AND CONCLUSIONS

Adopting ideas from conjugate gradient methods, we have proposed three new stochastic gradient descent techniques, and tested them in very ill-conditioned quadratic bowls. The results are quite promising in that one of our techniques outperforms simple stochastic gradient descent by an order of magnitude in all settings. We are now investigating the adaptation of our algorithms to nonlinear optimization problems, such as online learning in multi-layer perceptrons. We also hope to further improve upon these techniques through a systematic rederivation of the conjugate gradient method in the linear stochastic setting [10].

7. REFERENCES

[1] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathematics*, II(2):164–168, 1944.

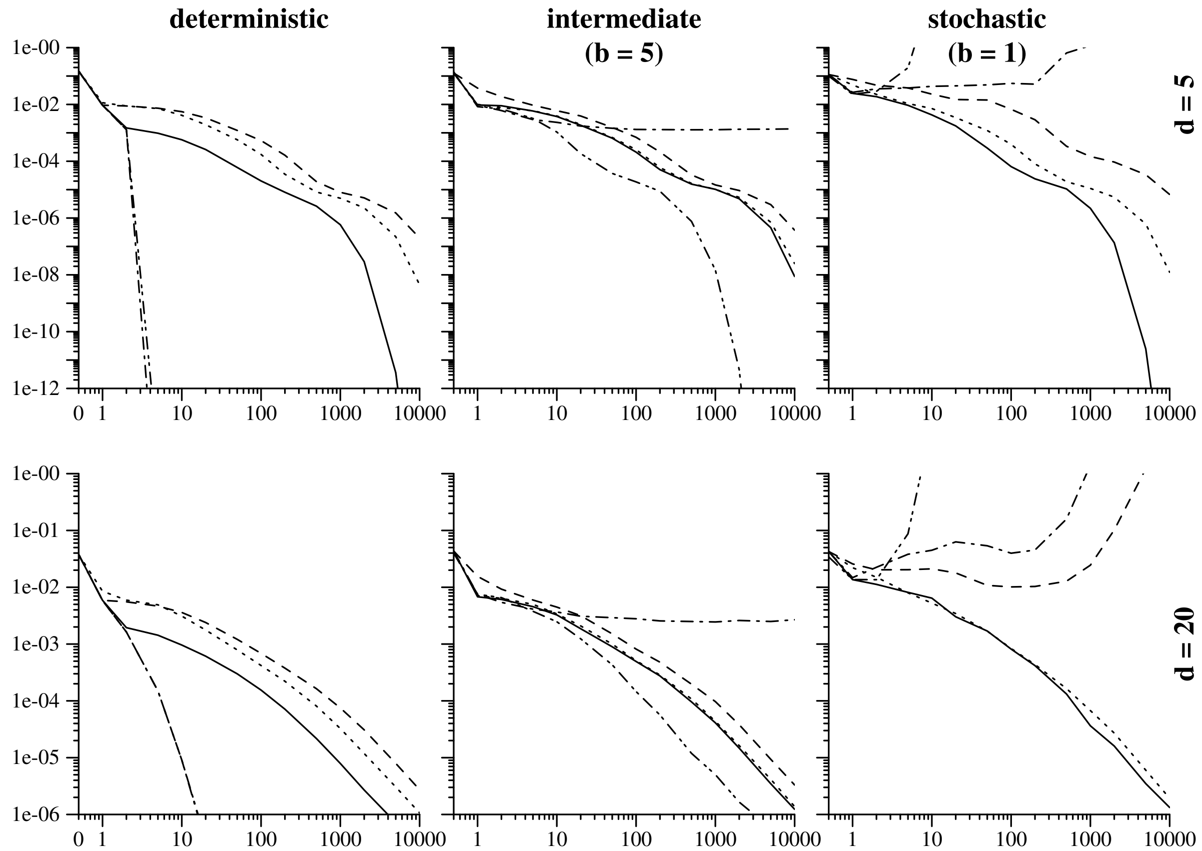

Figure 1: Log-log plots of average loss (geometric mean over 100 repetitions) *vs.* iterations of training in modified Hilbert bowls of dimension $d = 5$ (top) and $d = 20$ (bottom), for conjugate gradient (dot-dashed), stabilized conjugate gradient (dot-dot-dashed), pairwise conjugation (solid), diagonal conditioner ($\gamma = 0.1$, dotted), and simple gradient descent (dashed).

[2] D. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[3] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[4] N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, pages 569–574, Edinburgh, Scotland, 1999. IEE, London. `http://www.inf.ethz.ch/~schraudo/pubs/smd.ps.gz`.

[5] N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

[6] B. A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

[7] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[8] G. B. Orr. *Dynamics and Algorithms for Stochastic Learning*. PhD thesis, Department of Computer Science and Engineering, Oregon Graduate Institute, Beaverton, OR 97006, 1995. `ftp://neural.cse.ogi.edu/pub/neural/papers/orrPhDch1-5.ps.Z`, `orrPhDch6-9.ps.Z`.

[9] T. Graepel and N. N. Schraudolph. Stable adaptive momentum for rapid online learning in nonlinear systems. In Dorronsoro [11]. `http://www.inf.ethz.ch/~schraudo/pubs/sam.ps.gz`.

[10] N. N. Schraudolph and T. Graepel. Conjugate directions for stochastic gradient descent. In Dorronsoro [11]. `http://www.inf.ethz.ch/~schraudo/pubs/hg.ps.gz`.

[11] J. R. Dorronsoro, editor. *Proceedings of the International Conference on Artificial Neural Networks*, volume 2415 of *Lecture Notes in Computer Science*, Madrid, Spain, 2002. Springer Verlag, Berlin.