Scaling Up Natural Gradient by Sparsely Factorizing the Inverse Fisher Matrix

Roger B. Grosse RGROSSE@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Computer Science, University of Toronto

Abstract

Second-order optimization methods, such as natural gradient, are difficult to apply to high-dimensional problems, because they require approximately solving large linear systems. We present FActorized Natural Gradient (FANG), an approximation to natural gradient descent where the Fisher matrix is approximated with a Gaussian graphical model whose precision matrix can be computed efficiently. We analyze the Fisher matrix for a small RBM and derive an extremely sparse graphical model which is a good match to the covariance of the sufficient statistics. Our experiments indicate that FANG allows RBMs to be trained more efficiently compared with stochastic gradient descent. Additionally, our analysis yields insight into the surprisingly good performance of the "centering trick" for training RBMs.

1. Introduction

In the field of deep learning, stochastic gradient descent (SGD) has been the optimization workhorse in the large-data setting (Bottou & Bousquet, 2007). While it often works well in practice, it has been observed that deep networks suffer from severe curvature problems, and this observation has motivated work on second-order optimization methods (Martens, 2010; Martens & Sutskever, 2012). Second-order methods such as Newton-Rhapson and natural gradient (Amari, 1998) attenuate these problems by solving a linear system related to the curvature of the loss function. Unfortunately, the requirement of constructing and solving a large linear system makes the exact versions of these algorithms impractical for large models.

Various approximations have been proposed which avoid the cost of matrix inversion. The AdaGrad method (Duchi et al., 2011) typically assumes a diagonal approximation to the matrix, as diagonal matrices can be easily inverted. TONGA (Le Roux et al., 2008) is an approximation to natural gradient which assumes block diagonal structure, and low rank structure within each block. While these methods have the virtue that the curvature statistics can be estimated online, the approximation may be inaccurate. Hessian-free (H-F) optimization (Martens, 2010) uses an iterative method to approximately solve the linear system with a series of implicit matrix-vector products (Schraudolph, 2002). H-F accounts for the curvature more accurately than diagonal methods, but requires an expensive iterative procedure for each update and only uses curvature information associated with a single mini-batch.

Ideally, we would like an approximation to the curvature which can be estimated online, is cheap to invert, and captures the important interactions between different model parameters. In Section 4, we propose FActorized Natural Gradient (FANG), an approximation to natural gradient descent where the Fisher matrix $\mathbf{G}$ is approximated with a Gaussian graphical model over the sufficient statistics. By empirically analyzing the structure of $\mathbf{G}$ for a binary restricted Boltzmann machine (RBM; Smolensky, 1986), we derive an extremely sparse graphical model structure which is nonetheless a good fit to $\mathbf{G}$. We show how the approximate covariance can be efficiently inverted to compute the approximate natural gradient, as well as how the graphical model parameters can be efficiently estimated from data. In Section 5, we provide an alternative interpretation of FANG as an approximate whitening transformation, and use this interpretation to explain the surprising effectiveness of the recently discovered "centering trick" for training RBMs (Cho et al., 2011; Montavon & Muller, 2012), which has not previously been connected to natural gradient. In Section 6, we show that FANG outperforms SGD, both with and without the centering trick, for training binary RBMs. While we focus on RBMs, FANG should give a recipe more generally for designing second-order optimization methods which account for model structure.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

2. Background

2.1. Learning in MRFs

While our work focuses on RBMs, we first introduce the more general setting of exponential families because our proposed methods can be described more cleanly in this setting. An exponential family is a family of distributions which satisfy the following form:

$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}(\boldsymbol{\eta})} h(\mathbf{x}) \exp\left(\boldsymbol{\eta}^T \mathbf{g}(\mathbf{x})\right), \tag{1}$$

where $\boldsymbol{\eta}$ denotes the vector of natural parameters, $\mathbf{g}$ denotes the sufficient statistics of the distribution, and $\mathcal{Z}$ is the (often intractable) partition function. The representation in terms of natural parameters and sufficient statistics is unique up to affine transformation. Examples of exponential families include many commonly used distributions (e.g. Gaussians and multinomials) as well as Markov random field (MRF) models (Wainwright & Jordan, 2008).

Restricted Boltzmann machines (RBMs; Smolensky, 1986) are a type of MRF with a bipartite structure over two sets of variables: the visible units $\mathbf{v}$ (which are typically observed) and hidden units $\mathbf{h}$ (which are typically latent). For simplicity, we assume all of the units are binary-valued, but RBMs can be defined with other exponential families (Welling et al., 2004). In the binary case, the model distribution is given by:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}} \exp\left(\mathbf{a}^T \mathbf{v} + \mathbf{b}^T \mathbf{h} + \mathbf{v}^T \mathbf{W} \mathbf{h}\right), \tag{2}$$

where the parameters are the visible biases $\mathbf{a}$, the hidden biases $\mathbf{b}$, and the weights $\mathbf{W}$. This can be fit into the exponential family formalism using $\boldsymbol{\eta} = (\mathbf{a}, \mathbf{b}, \mathrm{vec}(\mathbf{W}))$ and $\mathbf{g}(\mathbf{v}, \mathbf{h}) = (\mathbf{v}, \mathbf{h}, \mathrm{vec}(\mathbf{v}\mathbf{h}^T))$, where vec stacks the columns of a matrix into a vector.

Conceptually, one can fit exponential family models to data using maximum likelihood. The log-likelihood gradient has a particularly simple form:

$$\nabla_{\boldsymbol{\eta}} \ell = \mathbb{E}_{\mathrm{data}}[\mathbf{g}(\mathbf{x})] - \mathbb{E}_{\mathrm{model}}[\mathbf{g}(\mathbf{x})], \tag{3}$$

where $\mathbb{E}_{\mathrm{data}}$ denotes expectations with respect to the data conditional distribution $p(\mathbf{h} \,|\, \mathbf{v})$, and $\mathbb{E}_{\mathrm{model}}$ denotes expectations with respect to the model distribution. We refer to the model statistics $\mathbf{s} = \mathbb{E}[\mathbf{g}(\mathbf{x})]$ as the *moments* of the model.

In the case of RBMs, the data conditional moments can be computed exactly because the data conditional distribution factorizes. However, computing the model moments is intractable because it is equivalent to marginalization, a problem which is #P-hard in the worst case. While much work has been devoted to accurately approximating the model moments, in this paper we use persistent contrastive divergence (PCD; Tieleman, 2008), the most commonly used procedure for training RBMs in a generative setting. In PCD, one approximates the model moments using a set of approximate samples called fantasy particles. After each SGD update, the particles are updated with one or more Gibbs steps in order to move them closer to the true model distribution.

2.2. Natural gradient

Natural gradient descent[1] (Amari, 1998) is a second-order optimization method for training statistical models. The update rule is given by

$$\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} + \alpha \mathbf{G}^{-1} \nabla_{\boldsymbol{\eta}} \ell, \tag{4}$$

where $\mathbf{G}$ is the matrix representation of a Riemannian metric, most commonly the Fisher metric. For exponential families, the Fisher metric is given by:

$$\mathbf{G} = \mathbb{E}_{p(\mathbf{x})}\left[\nabla_{\boldsymbol{\eta}} \ell (\nabla_{\boldsymbol{\eta}} \ell)^T\right] = \mathrm{Cov}_{p(\mathbf{x})}(\mathbf{g}). \tag{5}$$

Natural gradient descent with the Fisher metric is also known as Fisher scoring. One justification for the Fisher metric is that it is the second-order Taylor approximation to KL divergence; natural gradient descent can be interpreted as steepest descent in this metric. To a first order approximation, the update is invariant to reparameterization of the model. By contrast, standard SGD is steepest descent in the Euclidean norm, which results in different directions depending how the model is parameterized.

In maximum likelihood learning for fully observed exponential family models with natural parameterization, the Fisher matrix is the negative Hessian of the log-likelihood, so natural gradient is equivalent to Newton's method. This equivalence does not hold in our setting since RBMs have hidden variables, but it still provides a heuristic motivation for the algorithm.

Natural gradient is difficult to apply to high-dimensional models, since $\mathbf{G}$ is a $D \times D$ matrix, where D is the number of parameters; therefore, it is impractical even to compute $\mathbf{G}$ explicitly, much less to solve the linear system of Eqn 4 exactly. For instance, a typical binary RBM trained on MNIST may have 784 visible units and 500 hidden units, so $\mathbf{G}$ would have more than $(784 \times 500)^2$ entries. Various approximations have been proposed. TONGA (Le Roux et al., 2008) is as an approximation to NG which combines low-rank and block diagonal structure. Metric-free natural gradient (Desjardins et al., 2013), an approximation inspired by Hessian-free optimization (Martens, 2010), uses conjugate gradient combined with implicit computation of matrix-vector products involving $\mathbf{G}$. Kronecker-factored approximate curvature (Martens & Grosse, 2015) is a recent method, similar in spirit to our own, which uses a different tractable approximation to the Fisher matrix for training feed-forward neural nets.

[1] When referring to *descent* methods, we follow the machine learning convention of minimizing the negative log-likelihood.

2.3. Sparse approximate inverses

There are several methods for approximately solving large linear systems $\mathbf{Ax} = \mathbf{b}$ which are similar in spirit to our own. The incomplete Cholesky factorization (Meijerink & van der Vorst, 1977; Benzi, 2002) computes an approximate factorization $\mathbf{A} \approx \mathbf{LL}^T$ where the lower triangular factor $\mathbf{L}$ shares the sparsity structure of $\mathbf{A}$. The linear system is then approximately solved using backsubstitution. As we discuss in Section 5, FANG can be viewed as an approximate Cholesky factorization of $\mathbf{A}^{-1}$ (rather than $\mathbf{A}$). The method of sparse approximate inverses (Benzi et al., 1995; Benzi, 2002) attempts to find sparse matrices $\mathbf{M}$ and $\mathbf{N}$ such that $\mathbf{MN} \approx \mathbf{A}^{-1}$; FANG can be viewed as finding a factorization of this form (see Section 5). Both methods are used as preconditioners in conjugate gradient.

Neither method applies to our setting, since they both require the matrix $\mathbf{A}$ to be sparse. The Fisher matrix $\mathbf{G}$ is typically dense, and we don't believe it can be accurately approximated with a sparse matrix. Instead, FANG fits a sparse approximate factorization of $\mathbf{G}^{-1}$ directly without ever computing $\mathbf{G}$ as an intermediate step.

3. Notation

For the remainder of this paper, we focus on binary RBMs. We denote the natural parameters as $\boldsymbol{\eta} = (\mathbf{a}, \mathbf{b}, \mathrm{vec}(\mathbf{W}))$, the sufficient statistics as $\mathbf{g} = (\mathbf{v}, \mathbf{h}, \mathrm{vec}(\mathbf{vh}^T))$, the moments as $\mathbf{s} = \mathbb{E}[\mathbf{g}]$, and the Fisher matrix as $\mathbf{G}$. The visible activations, hidden activations, and pairwise statistics will be denoted v_i, h_j, and $m_{ij} = v_i h_j$, respectively.

Unfortunately, referring to vectors whose dimensions correspond to both unary and pairwise statistics leads to some awkward notation. We maintain the following indexing conventions: i always indexes the visible units, j always indexes the hidden units, and a is an "abstract index" for a sufficient statistic (and may refer to either a unary or pairwise statistic). In some contexts, we will specify that $a = (i, j)$ references a particular pairwise statistic over units i and j.

We will sometimes need to refer to particular blocks of a matrix $\mathbf{G}$ whose rows and columns correspond to sufficient statistics. In this case, $\mathbf{G}_{(i,j),(i,j)}$ will denote the 2×2 block for unary statistics i and j, while $\mathbf{G}_{a,a}$ will denote the single entry for the pairwise statistic $a = (i, j)$.

4. Factorized Natural Gradient

In this section, we motivate and present our proposed method, FActorized Natural Gradient (FANG). This method is based on approximating the Fisher matrix $\mathbf{G} = \mathrm{Cov}(\mathbf{g})$ as the covariance matrix for a sparse Gaussian graphical model over the sufficient statistics $\mathbf{g}$. As a reference point, observe that a diagonal approximation to $\mathbf{G}$ implicitly assumes that all of the sufficient statistics are independent, or equivalently, it assumes a fully disconnected graph structure (Figure 1). At the opposite extreme, exact natural gradient models $\mathbf{G}$ as a general covariance matrix, which corresponds to a fully connected graph. Ideally, we would like a compromise: a sparse graph structure which nonetheless captures the important interactions between the sufficient statistics.

4.1. Analyzing G for a small RBM

To motivate our graphical model approximation, we first analyze $\mathbf{G}$ for a small RBM where it can be computed exactly, in order to determine which interactions are important to model.

Many computations, such as marginalization and partition function computation, can be performed exactly on an RBM with only 20 hidden units (Salakhutdinov & Murray, 2008) by exhaustively summing over all 2^{20} hidden configurations. This approach works for computing $\mathbf{G}$ as well. In particular, $\mathbf{G} = \mathbb{E}[\mathbf{gg}^T] - \mathbb{E}[\mathbf{g}]\mathbb{E}[\mathbf{g}]^T$, where

$$\mathbb{E}[\mathbf{g}] = \sum_{\mathbf{h}} p(\mathbf{h})\, \mathbb{E}[\mathbf{g} \mid \mathbf{h}] \tag{6}$$

$$\mathbb{E}[\mathbf{gg}^T] = \sum_{\mathbf{h}} p(\mathbf{h}) \left(\mathbb{E}[\mathbf{g} \mid \mathbf{h}]\mathbb{E}[\mathbf{g} \mid \mathbf{h}]^T + \mathrm{Cov}[\mathbf{g} \mid \mathbf{h}] \right). \tag{7}$$

All of the conditional expectations and covariances can be computed efficiently because the conditional distribution factorizes over $\mathbf{v}$. In this section, we analyze $\mathbf{G}$ for a binary RBM with 20 hidden units and 196 visible units, trained on a subsampled version of the MNIST handwritten digit dataset (LeCun et al., 1998) using PCD. The matrix $\mathbf{G}$ is of size 4136×4136.

We analyze the eigendecomposition of $\mathbf{G}$. The eigenvalue spectrum and a visualization of the top eigenvector are shown in Figure 2. From the eigenspectrum, we see that there are about 10 particularly large eigenvalues, and many moderate-sized eigenvalues. To interpret the eigenvectors, we use the identity $\mathrm{d}\mathbf{s} = \mathbf{G}\mathrm{d}\boldsymbol{\eta}$ (Amari & Nagaoka, 2000). (Intuitively, $\mathbf{G}$ relates small changes in the two exponential family parameterizations.) This implies that the eigenvectors are the values of $\mathrm{d}\boldsymbol{\eta}$ which satisfy $\mathrm{d}\mathbf{s} = \lambda \mathrm{d}\boldsymbol{\eta}$ for some λ. Viewing the eigenvectors in the space of moments rather than in the space of natural parameters allows us to separate out two different effects. In particular,

$$\mathrm{d}\mathbb{E}[\mathbf{vh}^T] = \mathrm{d}\mathbb{E}[\mathbf{v}]\mathbb{E}[\mathbf{h}]^T + \mathbb{E}[\mathbf{v}]\mathrm{d}\mathbb{E}[\mathbf{h}]^T + \mathrm{d}\,\mathrm{Cov}(\mathbf{v}, \mathbf{h}). \tag{8}$$

The final term is the interesting one for representation learning, because changing the correlations corresponds to changing what a given hidden unit represents. The first two terms are less interesting because they simply correspond to shifting the average activations. We quantify the extent to which each eigenvector is explained in terms of changing

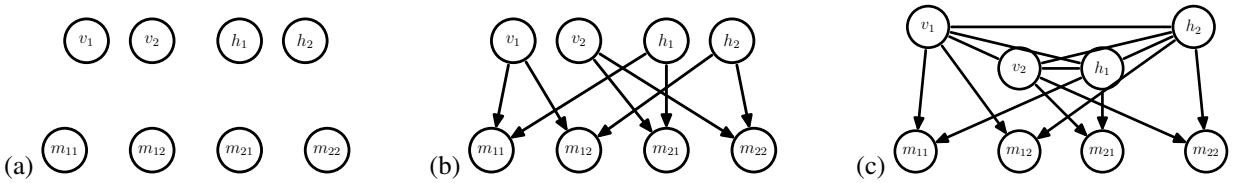

Figure 1. Graphical models for different approximations to the Fisher matrix. **Left:** diagonal approximation. **Middle:** centering trick (Section 5.1). **Right:** FANG.

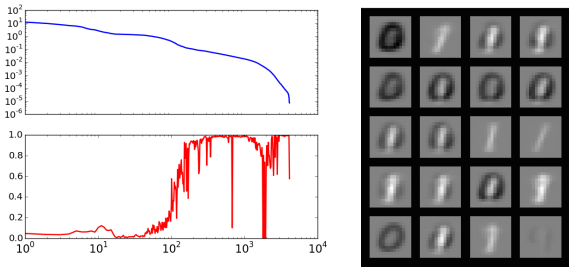

Figure 2. Visualization of the eigendecomposition of $\mathbf{G}$ for a small MNIST RBM. **Left top:** Eigenspectrum of $\mathbf{G}$. **Left bottom:** Fraction of each eigenvector explainable in terms of changes to the correlations. (See Section 4.1 for details.) **Right:** $\mathrm{d}\mathbf{W}$ component of the top eigenvector, which appears to capture the difference between 0's and 1's.

correlations with

$$f(\mathrm{d}\mathbf{s}) = \frac{\|\mathrm{d}\,\mathrm{Cov}(\mathbf{v},\mathbf{h})\|^2}{\|\mathrm{d}\,\mathrm{Cov}(\mathbf{v},\mathbf{h})\|^2 + \|\mathrm{d}\mathbb{E}[\mathbf{v}]\mathbb{E}[\mathbf{h}]^T + \mathbb{E}[\mathbf{v}]\mathrm{d}\mathbb{E}[\mathbf{h}]^T\|^2}. \tag{9}$$

Figure 2 shows f for each of the eigenvectors. For the first 100 eigenvectors, most of the change is explained in terms of changes to the average activations, rather than to the correlations. Figure 2 shows a visualization of the top eigenvector, which appears to reflect the difference between 0's and 1's. The first 30 eigenvectors are all qualitatively similar to this one.

To summarize, the largest eigenvectors of $\mathbf{G}$ (and hence the directions of highest curvature) correspond to changes to the unary statistics, rather than changes to the covariances. This is problematic from the standpoint of optimization, since the most interesting structure (the covariances) is buried in the smaller eigenvalues. This suggests that it is *essential* to accurately model the dependence between the unary statistics, as well as the way in which the pairwise statistics depend on the unary ones.

4.2. Gaussian graphical model

As argued in the preceding section, an accurate model must capture the covariance of the unary statistics, as well as model the manner in which the pairwise statistics depend on the unary ones. We now attempt to model this structure by approximating $\mathbf{G}$ as a Gaussian graphical model. In particular, consider the graphical model structure of Figure 1(c), which includes a fully connected clique over unary statistics v_i and h_j, as well as linear dependence of each pairwise statistic $m_{ij} = v_i h_j$ on the corresponding unary statistics. This corresponds to a factorization of the joint distribution as

$$p(\mathbf{g}) = p(\mathbf{v},\mathbf{h}) \prod_{i=1}^{N_v} \prod_{j=1}^{N_h} p(m_{ij} \mid v_i, h_j). \tag{10}$$

As a heuristic motivation for this factorization structure, observe that if the variables in the graphical model were binary, this would represent the joint distribution exactly. In particular, the fully connected clique over $(\mathbf{v},\mathbf{h})$ could represent the RBM's joint distribution, and each m_{ij} is a deterministic function of its parents. The approximation comes from modeling the joint distribution as Gaussian. Each of the CPDs for the pairwise variables is a regression model; while it can't represent the conditional distribution exactly, one hopes it might still be a reasonable proxy.

Specifying a Gaussian graphical model with this structure requires specifying:

- The mean vector $\boldsymbol{\mu} = \mathbb{E}[\mathbf{g}]$
- The covariance matrix $\boldsymbol{\Sigma}_{\mathrm{un}} = \mathbb{E}[\mathbf{g}_{\mathrm{un}}\mathbf{g}_{\mathrm{un}}^T]$ corresponding to the fully connected clique over unary statistics
- The conditional probability distributions for the pairwise potentials m_{ij}. This takes the form of a linear-Gaussian model,

$$p(m_{ij} \mid v_i, h_j) \propto \exp\left(-\frac{1}{2\sigma_{ij}^2}(m_{ij} - \alpha v_i - \beta h_j - \gamma)^2\right),$$

  which requires specifying the coefficients α and β, the offset γ, and the conditional variance σ_{ij}^2.

The graphical model defines a covariance matrix $\mathbf{G}_{\mathrm{fac}}$ which approximates the Fisher matrix $\mathbf{G}$. However, the graphical model structure reflects itself in the *inverse* covariance matrix, or precision matrix, $\boldsymbol{\Lambda} = \mathbf{G}_{\mathrm{fac}}^{-1}$. In particular, each entry $\boldsymbol{\Lambda}_{ab}$ is zero unless a and b are part of the

same clique. There are $(N_v + N_h)^2$ nonzero entries for the clique over unary statistics, plus an additional 5 nonzero entries for each of the $N_v N_h$ CPDs $p(m_{ij} \mid v_i, h_j)$. Therefore, the total number of nonzero entries is $O(N_v^2 + N_h^2)$, which is linear in the number of weights as long as $N_v \approx N_h$. This is in contrast with the unstructured Fisher matrix $\mathbf{G}$, which has $O(N_v^2 N_h^2)$ nonzero entries — quadratic in the number of weights! For instance, in a typical sized RBM with 784 visible units and 500 hidden units, $\mathbf{G}_{\text{fac}}^{-1}$ has only 3.6 million nonzero entries, compared with nearly 154 billion for $\mathbf{G}$ or $\mathbf{G}^{-1}$.

Computing the approximate natural gradient $\mathbf{r} = \mathbf{G}_{\text{fac}}^{-1} \nabla_{\boldsymbol{\eta}} \ell$ requires only computing a sparse matrix-vector product, so the running time is linear in the number of nonzero entries. Rather than computing $\mathbf{G}_{\text{fac}}^{-1}$ explicitly, it is more convenient to represent it implicitly in terms of the parameters of the graphical model. The resulting update is shown in Algorithm 1.

Algorithm 1 Factorized Natural Gradient (FANG) for binary RBMs

while stopping criterion not satisfied **do**
  Take a step of persistent contrastive divergence (PCD)
  $\mathbf{g}^{(s)} \leftarrow$ Sufficient statistics from the sth PCD particle
  $\nabla_{\boldsymbol{\eta}} \ell \leftarrow$ log-likelihood gradient as estimated by PCD

  {Update weights and biases}
  $\mathbf{r} \leftarrow \mathbf{0}$
  $\mathbf{r}_{\text{un}} \leftarrow \boldsymbol{\Lambda}_{\text{un}} [\nabla_{\boldsymbol{\eta}} \ell]_{\text{un}}$
  for each abstract index $a = (i, j)$ **do**
    $\mathbf{r}_{i,j,a} \leftarrow \mathbf{r}_{i,j,a} + \frac{1}{\sigma_{ij}^2} \begin{pmatrix} -\boldsymbol{\beta}_a \\ 1 \end{pmatrix} (-\boldsymbol{\beta}_a^T, 1) [\nabla_{\boldsymbol{\eta}} \ell]_{i,j,a}$
  end for
  $\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} + \alpha \mathbf{r}$

  {Update mean and covariance statistics}
  $\boldsymbol{\mu} \leftarrow (1 - \frac{1}{\tau}) \boldsymbol{\mu} + \frac{1}{\tau} \frac{1}{K} \sum_{s=1}^{K} \mathbf{g}^{(s)}$
  for each pair of indices a, b such that $\boldsymbol{\Sigma}_{ab}$ is needed to fit $\mathbf{G}_{\text{fac}}^{-1}$ **do**
    $\mathbf{S}_{ab} \leftarrow (1 - \frac{1}{\tau}) \mathbf{S}_{ab} + \frac{1}{\tau} \frac{1}{K} \sum_{s=1}^{K} \mathbf{g}_a^{(s)} \mathbf{g}_b^{(s)}$
    $\boldsymbol{\Sigma}_{ab} \leftarrow \mathbf{S}_{ab} - \boldsymbol{\mu}_a \boldsymbol{\mu}_b$
  end for

  if it has been long enough since the last update **then**
    {Update graphical model parameters}
    $\boldsymbol{\Lambda}_{\text{un}} \leftarrow \boldsymbol{\Sigma}_{\text{un}}^{-1}$
    for each visible index i and hidden index j, and corresponding pairwise index $a = (i, j)$ **do**
      $\boldsymbol{\beta}_a \leftarrow \boldsymbol{\Sigma}_{(i,j),(i,j)}^{-1} \boldsymbol{\Sigma}_{(i,j),a}$
      $\sigma_{ij}^2 \leftarrow \boldsymbol{\Sigma}_{a,a} - \boldsymbol{\Sigma}_{a,(i,j)} \boldsymbol{\Sigma}_{(i,j),(i,j)}^{-1} \boldsymbol{\Sigma}_{(i,j),a}$
    end for
  end if
end while

4.3. Fitting the graphical model parameters

It remains to describe how to fit the parameters of the graphical model, which are summarized in Section 4.2. Maximum likelihood estimates of the mean vector $\boldsymbol{\mu}$ and covariance matrix over unary statistics $\boldsymbol{\Sigma}_{\text{un}}$ can each be derived from the empirical first and second moments of the relevant variables.

The remaining parameters correspond to the directed edges. In general, maximum likelihood estimation for directed Gaussian graphical models reduces to independent linear regression problems, one for each CPD. In our case, this means there are $N_v \times N_h$ independent regression problems, each one predicting m_{ij} as a linear function of its two parents v_i and h_j. In order to solve these regression problems, we need only the covariance matrices $\boldsymbol{\Sigma}_{(i,j,a),(i,j,a)}$ over a pairwise statistic and its two parents. (The notation $\boldsymbol{\Sigma}_{(i,j,a),(i,j,a)}$ refers to the 3×3 block whose indices are the visible index i, the hidden index j, and the abstract index $a = (i, j)$.) Given this matrix, we can determine the regression weights $\boldsymbol{\beta}_a$ and conditional variance σ_{ij}^2 using the linear regression formula

$$\boldsymbol{\beta}_a = \boldsymbol{\Sigma}_{(i,j),(i,j)}^{-1} \boldsymbol{\Sigma}_{(i,j),a} \tag{11}$$

$$\sigma_{ij}^2 = \boldsymbol{\Sigma}_{a,a} - \boldsymbol{\Sigma}_{a,(i,j)} \boldsymbol{\Sigma}_{(i,j),(i,j)}^{-1} \boldsymbol{\Sigma}_{(i,j),a}. \tag{12}$$

Note that the offset parameter γ is not used in the FANG update, so we do not bother to compute it.

We estimate all of the empirical statistics in an online fashion from the PCD particles (which are hopefully a good proxy for the model distribution), using an exponentially decaying average with timescale $\tau = 100$. As a form of smoothing, we add $\lambda \mathbf{I}$ to all of the estimated covariance matrices, with $\lambda = 0.01$ fixed throughout all our experiments.

There are two components to the running time for fitting the graphical model. First, we need to estimate the statistics. Updating $\boldsymbol{\Sigma}_{\text{un}}$ requires $K(N_v + N_h)^2$ operations (where K is the number of PCD particles). Updating the $N_v N_h$ covariance matrices for the pairwise CPDs requires $9 K N_v N_h$ operations. Each of these costs is comparable to the cost of computing the SGD update itself.

The other significant cost is computing the graphical model parameters. Estimating the CPD parameters requires solving $N_v N_h$ regression problems in 2 variables, so it is an $O(N_v N_h)$ operation. Inverting the covariance matrix $\boldsymbol{\Sigma}_{\text{un}}$ requires $O(N_v + N_h)^3$ operations, which makes it asymptotically the most expensive operation required for FANG. Fortunately, the graphical model parameters do not need to be updated in every iteration, since $\mathbf{G}$ is unlikely to change very quickly during training. In our experiments, we re-estimated the graphical model parameters once every 100 mini-batches.

While Algorithm 1 is presented in terms of `for`-loops for clarity, all of the steps of the algorithm can be vectorized. Our implementation made use of the CUDAMat (Mnih, 2009) and Gnumpy (Tieleman, 2010) libraries for GPU lin-

ear algebra operations.[2]

5. Interpreting FANG as approximate whitening

In this section, we develop an alternative view of FANG as an approximate whitening transformation. This interpretation yields additional insights into the algorithm which are not apparent from the graphical model formulation. Importantly, it yields a novel interpretation of the centering trick of Cho et al. (2011) and Montavon & Muller (2012) as an approximation to natural gradient.

For notational convenience, we number the coordinates of all vectors and matrices in *decreasing* order, e.g.

$$\mathbf{g} = (g_D, g_{D-1}, \cdots, g_1)^T.$$

To motivate the use of linear reparameterizations, observe that the choice of sufficient statistics and natural parameters for an exponential family is unique only up to affine transformation. In particular, any invertible linear transformation $\mathbf{g} \mapsto \mathbf{A}\mathbf{g}$ and $\boldsymbol{\eta} \mapsto \mathbf{A}^{-T}\boldsymbol{\eta}$ results in an equivalent model. When computing gradient descent updates, it may be advantageous to choose a transformation $\mathbf{A}$ which decorrelates different statistics.

In particular, for each statistic g_a, choose a set of parent statistics $\mathrm{Pa}(a) \subset \{1, \ldots, a-1\}$. Consider the whitening transform which subtracts from each statistic the component which is linearly predictable from its parents, and normalizes the result to unit variance:

$$\bar{g}_a = g_a - \sum_{b \in \mathrm{Pa}(a)} \beta_{a,b} g_b \tag{13}$$

$$\tilde{g}_a = \bar{g}_a / \sqrt{\mathrm{Var}(\bar{g}_a)}, \tag{14}$$

where $\{\beta_{a,b}\}_{b \in \mathrm{Pa}(a)}$ denote the optimal linear regression weights for predicting g_a as a function of its parents. These two steps can be written as $\tilde{\mathbf{g}} = \mathbf{D}^{1/2}\mathbf{L}^T\mathbf{g}$, where

$$\mathbf{L} = \begin{pmatrix} 1 & & & \\ -\beta_{D,D-1} & 1 & & \\ \vdots & & \ddots & \\ -\beta_{D,1} & -\beta_{D-1,1} & \cdots & 1 \end{pmatrix} \tag{15}$$

$$\mathbf{D} = \mathrm{diag}\left(1/\mathrm{Var}(\bar{g}_D), \ldots, 1/\mathrm{Var}(\bar{g}_1)\right), \tag{16}$$

and we define $\beta_{a,b} = 0$ for $b \notin \mathrm{Pa}(a)$.

We now show that exact natural gradient corresponds to the special case where $\mathrm{Pa}(a) = \{1, \ldots, a-1\}$. In this case, $\mathbf{L}\mathbf{D}\mathbf{L}^T$ is the Cholesky factorization of $\mathbf{G}^{-1}$, and the transformation exactly whitens the sufficient statistics. The natural gradient descent update is equivalent to the SGD update in the whitened space.

Theorem 1. *If* $\mathrm{Pa}(a) = \{1, \ldots, a-1\}$ *for all* a*, then* $\mathrm{Cov}(\tilde{\mathbf{g}}) = \mathbf{I}$ *and* $\mathbf{G}^{-1} = \mathbf{L}\mathbf{D}\mathbf{L}^T$*. Natural gradient descent is equivalent to (Euclidean) gradient descent on the whitened parameters* $\tilde{\boldsymbol{\eta}} = \mathbf{D}^{1/2}\mathbf{L}^T\boldsymbol{\eta}$*.*

Proof. Since each $\bar{g}_a$ corresponds to the residuals in regression from $g_1, \ldots, g_{a-1}$, $\bar{g}_a$ must be uncorrelated with any linear combination of $g_1, \ldots, g_{a-1}$. In particular, $\bar{g}_b$ for $b < a$ is such a linear combination, so all of the statistics $\bar{g}_a$ are therefore uncorrelated. The next step renormalizes these statistics to unit variance. Therefore, $\mathrm{Cov}(\tilde{\mathbf{g}}) = \mathbf{I}$.

Now, since $\mathbf{g} = \mathbf{L}^{-T}\mathbf{D}^{-1/2}\tilde{\mathbf{g}}$,

$$\begin{aligned} \mathbf{G} = \mathrm{Cov}(\mathbf{g}) &= \mathbf{L}^{-T}\mathbf{D}^{-1/2}\,\mathrm{Cov}(\tilde{\mathbf{g}})\mathbf{D}^{-1/2}\mathbf{L}^{-1} \\ &= \mathbf{L}^{-T}\mathbf{D}^{-1/2}\mathbf{D}^{-1/2}\mathbf{L}^{-1} \\ &= (\mathbf{L}\mathbf{D}\mathbf{L}^T)^{-1}. \end{aligned}$$

Therefore $\mathbf{G}^{-1} = \mathbf{L}\mathbf{D}\mathbf{L}^T$.

Performing gradient descent on transformed parameters $\tilde{\boldsymbol{\eta}} = \mathbf{A}\boldsymbol{\eta}$ is equivalent to updating $\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} + \alpha\mathbf{A}^T\mathbf{A}\nabla_{\boldsymbol{\eta}}\ell$. Plugging in $\mathbf{A} = \mathbf{D}^{1/2}\mathbf{L}^T$ shows that natural gradient is equivalent to gradient descent in the whitened space. □

We now show that FANG is equivalent to the approximate whitening transformation where $\mathrm{Pa}(a)$ is chosen to be the set of parents of node a in the graphical model.[3] Eqn 13 computes the same optimal regression weights for each statistic as Eqn 11. The variances in Eqn 14 are the same as the conditional variance parameters computed by Eqn 12. The factorization $\mathbf{G}_{\mathrm{fac}} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ corresponds to the Cholesky factorization of the precision matrix of a directed Gaussian graphical model (Pourahmadi, 1999).

5.1. Explaining the centering trick

It has been observed that standard SGD training of RBMs is not invariant even to seemingly innocuous transformations of the data, such as inverting the pixel values (Cho et al., 2011). *E.g.*, SGD performs far worse at learning a generative model of MNIST when 0 is used as the foreground value and 1 as the background value, compared with the standard case where 1 is the foreground value.

Several researchers independently discovered an update rule for training RBMs and DBMs, known as the enhanced gradient (Cho et al., 2011; 2013), or centering trick (Montavon & Muller, 2012), which improves the stability of SGD and is invariant to inverting pixel values. The idea

[2] Because we were not aware of an existing GPU routine to solve many independent linear systems, the $\boldsymbol{\beta}_a$ parameters were computed on the CPU. In principle, however, this step ought to be highly parallelizable.

[3] In order to draw this equivalence, we must convert the hybrid graphical model into a fully directed one by assigning an arbitrary ordering to the unary statistics. The parent set for each unary statistic consists of all previous unary statistics.

is to center each of the units at zero by subtracting its empirical mean: $\tilde{\mathbf{v}} = \mathbf{v} - \boldsymbol{\mu}_{\mathbf{v}}$ and $\tilde{\mathbf{h}} = \mathbf{h} - \boldsymbol{\mu}_{\mathbf{h}}$. The model distribution is given in terms of the centered activations by

$$p(\mathbf{v}, \mathbf{h}) \propto \exp\left(\tilde{\mathbf{a}}^T \tilde{\mathbf{v}} + \tilde{\mathbf{b}}^T \tilde{\mathbf{h}} + \tilde{\mathbf{v}}^T \tilde{\mathbf{W}} \tilde{\mathbf{h}}\right), \tag{17}$$

where the centered parameters are given by $\tilde{\mathbf{a}} = \mathbf{a} + \mathbf{W}\boldsymbol{\mu}_{\mathbf{h}}$, $\tilde{\mathbf{b}} = \mathbf{b} + \mathbf{W}^T \boldsymbol{\mu}_{\mathbf{v}}$, and $\tilde{\mathbf{W}} = \mathbf{W}$. The centering trick updates the (non-centered) model parameters in a way which is equivalent to performing SGD on the centered parameters.

Montavon & Muller (2012) used the centering trick to train deep Boltzmann machines without a pre-training step, and Desjardins et al. (2013) found that it was critical to achieving good performance with metric-free natural gradient. Additionally, Montavon & Muller (2012) observed empirically that the trick improves the conditioning of the curvature matrix for some simple MRFs. However, the reason for the benefit appears to be poorly understood.

In our framework, the centering trick can be viewed as an approximation to natural gradient. In particular, the transformed RBM is an exponential family with the sufficient statistics:

$$\tilde{\mathbf{v}} = \mathbf{v} - \boldsymbol{\mu}_{\mathbf{v}} \tag{18}$$

$$\tilde{\mathbf{h}} = \mathbf{h} - \boldsymbol{\mu}_{\mathbf{h}} \tag{19}$$

$$\tilde{\mathbf{v}}\tilde{\mathbf{h}}^T = \mathbf{v}\mathbf{h}^T - \mathbf{v}\boldsymbol{\mu}_{\mathbf{h}}^T - \boldsymbol{\mu}_{\mathbf{v}}\mathbf{h}^T + \boldsymbol{\mu}_{\mathbf{v}}\boldsymbol{\mu}_{\mathbf{h}}^T. \tag{20}$$

The terms $\boldsymbol{\mu}_{\mathbf{v}}$, $\boldsymbol{\mu}_{\mathbf{h}}$, and $\boldsymbol{\mu}_{\mathbf{v}}\boldsymbol{\mu}_{\mathbf{h}}^T$ can be dropped since they are constant offsets and therefore do not affect the SGD update. This is a linear transformation of the original sufficient statistics, which can be written in the form Eqn 13 by setting $\beta_{a,i} = \mu_{hj}$ and $\beta_{a,j} = \mu_{vi}$ for each abstract index $a = (i, j)$, and setting all other weights to zero. Therefore, the centering trick can be viewed as an approximate whitening transformation where $\mathrm{Pa}(a) = \{i, j\}$, $\mathrm{Pa}(i) = \{\}$, and $\mathrm{Pa}(j) = \{\}$ for visible indices i, hidden indices j, and abstract indices $a = (i, j)$. The corresponding graphical model is shown in Figure 1.

In general, the values of $\beta_{a,b}$ defined above are not the optimal regression weights. However, if we are given two *independent* random variables v and h, the optimal regression weights for predicting $m = vh$ as a linear function of v and h are $\mathbb{E}[h]$ and $\mathbb{E}[v]$, respectively. Therefore, the centering trick can be viewed as an approximation to FANG where the visible and hidden units are assumed to be independent. This would be a reasonable approximation to the extent that any individual visible unit is only weakly correlated with any individual hidden unit. (Note that the assumption that v_i and h_j are independent is far more benign than the assumption made by the diagonal approximation to $\mathbf{G}$, namely that *all pairs of sufficient statistics* are uncorrelated, including v_i and m_{ij}.)

	$D(p\|q)$	# parameters
spherical	6469.8	1
diagonal	2290.4	4,136
block diagonal	2006.9	409,366
rank 50	4954.3	206,801
rank 200	3623.5	827,201
random connectivity	2243.3	35,196
centering	801.9	4,352
FANG	548.1	35,196

Table 1. Accuracy of various approximations to $\mathbf{G}$, evaluated in terms of the KL divergence $\mathrm{D}(p\|q)$, where p is a zero-centered Gaussian with covariance $\mathbf{G}$, and q is a zero-centered Gaussian with the approximate covariance. See Section 6.1 for details.

6. Experiments

In this section, we first evaluate FANG by comparing the accuracy of the approximation $\mathbf{G}_{\mathrm{fac}}$ with various generic approximations to PSD matrices. Next, we evaluate its ability to train binary restricted Boltzmann machines as generative models, compared with SGD, both with and without the centering trick.

6.1. Accuracy of approximation

We first compare various approximations to $\mathbf{G}$ in terms of the KL divergence $\mathrm{D}(p\|q)$, where p is a zero-centered Gaussian with covariance $\mathbf{G}$, and q is a zero-centered Gaussian with the approximate covariance. This comparison is done using the small RBM from Section 4.1. The KL divergence doesn't necessarily reflect the optimization performance, but is meant to give a rough measure of the accuracy of the approximations. Table 1 shows a comparison of our factorized approximation with diagonal, block diagonal and low-rank-plus-identity approximations of various sizes. The block diagonal approximation contains one block for each hidden unit, similarly to TONGA (Le Roux et al., 2008). FANG achieves far lower KL divergence than these alternatives, despite requiring far fewer parameters than the block diagonal or low rank approximations. We also include the graphical model corresponding to the centering trick (Section 5.1). Interestingly, the centering trick yields a more accurate approximation to $\mathbf{G}$ than the generic approximations, with fewer parameters, even though it was never intended as an approximation to natural gradient. As a control, we include a graphical model equivalent to FANG, but where each pairwise statistic has two *random* unary statistics as parents; this control is a very poor approximation, suggesting that the particular pattern of edges is important.

6.2. RBM training

Our RBM training experiments were conducted on two datasets: the MNIST handwritten digit dataset, which has

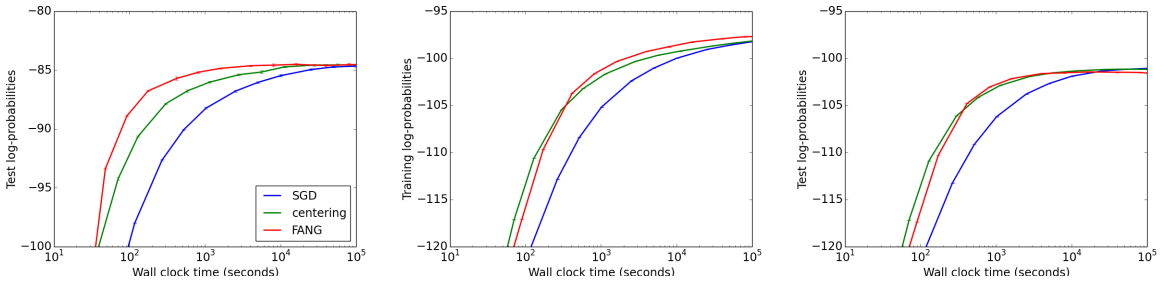

Figure 3. Comparison of standard SGD, the centering trick, and FANG for training RBMs as generative models. All three methods use PCD to estimate the log-likelihood gradient. **Left:** Log-probabilities of all three methods on the MNIST test set, as a function of running time. (Training set performance is roughly the same.) **Middle:** Omniglot training set. **Right:** Omniglot test set.

long served as a benchmark for representation learning algorithms (LeCun et al., 1998), and the more complex Omniglot dataset of handwritten characters in a variety of world languages (Lake et al., 2013). We compared FANG against SGD, both with and without the centering trick. All three methods used PCD to estimate the log-likelihood gradient.[4]

Accurately evaluating the test log-likelihood of an RBM is difficult because it requires the intractable partition function (Salakhutdinov & Murray, 2008). Similarly to prior work, we used annealed importance sampling (AIS; Neal, 2001) to estimate the partition function. Our annealing schedule used geometric averages with 50,000 distributions spaced linearly from 0 to 1. We computed 95% confidence intervals using the bootstrap, similarly to Burda et al. (2014).

Estimating RBM log-probabilities is notoriously tricky, since AIS can sometimes dramatically overestimate the log-likelihood (Grosse et al., 2013; Burda et al., 2014). To counter this problem, we re-estimated the final log-likelihoods using two alternative methods. First, we used AIS with moment averages (Grosse et al., 2013), which has been shown to yield more accurate estimates (at a larger computational cost). Second, we used the Reverse AIS Estimator (RAISE; Burda et al., 2014), a recently proposed estimator which errs on the side of underestimating, rather than overestimating, the log-probabilities. In all cases, all estimates agreed to within 0.5 nats, suggesting that the original AIS estimates are reliable.

Figure 3 shows the training and test log-likelihoods as a function of time. On the MNIST dataset, all three methods eventually reached a test log-likelihood of -84.6 nats, which is the state-of-the-art for RBMs. However, FANG approached this log-likelihood value much faster than SGD; for instance, it exceeded -85 nats after 27 minutes, compared with 1.5 hours for SGD with centering and 6.9 hours for SGD without centering. On the Omniglot dataset, FANG achieved a higher training log-likelihood than either version of SGD. However, as often happens with second-order optimization, the model overfit, and the test log-likelihoods were no higher than those obtained with centering. Despite this effect, both FANG and centering significantly outperformed plain SGD even on the test set.

[4]Details: Each of these datasets consists of pixel values between 0 and 1, and we followed the standard practice of using the continuous values during training (Hinton, 2010) and Bernoulli samples for evaluation (Salakhutdinov & Murray, 2008). The original Omniglot dataset consisted of 48×48 images, but we rescaled them to 28×28. We used 2000 PCD particles, mini-batches of size 2000, and a learning rate schedule of $\alpha\sqrt{\gamma/(\gamma+t)}$, where t is the update count, $\gamma = 1000$, and α was tuned separately for each algorithm.

7. Discussion

We have presented FANG, a second-order optimization algorithm which exploits the structure of the covariance over sufficient statistics. By approximating the covariance with a Gaussian graphical model, we obtain a compact approximation whose inverse can be efficiently computed. FANG performs well on training binary RBMs, and it helps explain the good performance of the centering trick. While our focus was on binary RBMs, this work suggests a recipe for constructing second-order optimization algorithms by investigating the structure of the curvature matrix.

We note that other second-order optimization methods besides natural gradient also require inverting covariance matrices. AdaGrad (Duchi et al., 2011) is typically used with a diagonal approximation to the covariance. Riemannian manifold Hamiltonian Monte Carlo (Girolami & Calderhead, 2011) is an MCMC analogue of natural gradient. The same structure exploited by FANG could potentially be used in conjunction with these other methods as well.

Acknowledgments

This research was supported by Samsung and NSERC. We thank Yuri Burda, Geoffrey Hinton, and James Martens for helpful discussions.

References

Amari, S. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

Amari, Shun-ichi and Nagaoka, Hiroshi. *Methods of Information Geometry*. Oxford University Press, 2000.

Benzi, M. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182: 418–477, 2002.

Benzi, M., Meyer, C. D., and Tuma, M. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal of Scientific Computing*, 17(5): 1135–1149, September 1995.

Bottou, L. and Bousquet, O. The tradeoffs of large scale learning. In *Neural Information Processing Systems*, 2007.

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Accurate and conservative estimates of MRF log-likelihood using reverse annealing. arXiv:1412.8566, 2014.

Cho, K., Raiko, T., and Ilin, A. Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. In *Int'l. Conf. on Machine Learning*, 2011.

Cho, K., Raiko, T., and Ilin, A. Enhanced gradient for training restricted Boltzmann machines. *Neural Computation*, 25:805–831, 2013.

Desjardins, G., Pascanu, R., Courville, A., and Bengio, Y. Metric-free natural gradient for joint-training of Boltzmann machines. arXiv:1301.3545, 2013.

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

Girolami, M. and Calderhead, B. Riemannian manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society B: Methodology*, 73 (2):123–214, 2011.

Grosse, R. B., Maddison, C. J., and Salakhutdinov, R. Annealing between distributions by averaging moments. In *Neural Information Processing Systems*, 2013.

Hinton, G. A practical guide to training restricted Boltzmann machines. Technical Report 003, University of Toronto, 2010.

Lake, Brenden M, Salakhutdinov, Ruslan, and Tenenbaum, Josh. One-shot learning by inverting a compositional causal process. In Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2526–2534. Curran Associates, Inc., 2013.

Le Roux, N., Manzagol, P., and Bengio, Y. Topmoumoute online natural gradient algorithm. In *Neural Information Processing Systems*, 2008.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Martens, J. Deep learning via Hessian-free optimization. In *Int'l. Conf. on Machine Learning*, 2010.

Martens, J. and Grosse, R. Optimizing neural networks with Kronecker-factored approximate curvature. In *Int'l. Conf. on Machine Learning*, 2015.

Martens, J. and Sutskever, I. Training deep and recurrent networks with Hessian-free optimization. In Montavon, G., Orr, G., and Muller, K. R. (eds.), *Neural Networks: Tricks of the Trade*. Springer, 2012.

Meijerink, J. A. and van der Vorst, H. A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, 31(137):148–162, January 1977.

Mnih, V. CUDAMat: a CUDA-based matrix class for Python. Technical report, University of Toronto, 2009.

Montavon, G. and Muller, K. R. Deep Boltzmann machines and the centering trick. In Montavon, G., Orr, G., and Muller, K. R. (eds.), *Neural Networks: Tricks of the Trade*. Springer, 2012.

Neal, R. M. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Pourahmadi, M. Joint mean-covariance models with applications to longitudinal data: unconstrained parameterization. *Biometrika*, 86(3):677–690, September 1999.

Salakhutdinov, Ruslan and Murray, Ian. On the quantitative analysis of deep belief networks. In *Int'l. Conf. on Machine Learning*, 2008.

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

Smolensky, P. Information processing in dynamical systems: foundations of harmony theory. In Rumelhart, D. E. and McClelland, J. L. (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, 1986.

Tieleman, T. Gnumpy: an easy way to use GPU boards in Python. Technical report, University of Toronto, 2010.

Tieleman, Tijmen. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Intl. Conf. on Machine Learning*, 2008.

Wainwright, Martin J. and Jordan, Michael I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Welling, M., Rosen-Zvi, M., and Hinton, G. E. Exponential family harmoniums with an application to information retrieval. In *Neural Information Processing Systems*, 2004.