A COMPUTATIONALLY EFFICIENT SPARSIFIED ONLINE NEWTON METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

Second-order methods have huge potential in improving the convergence of deep neural network (DNN) training, but are prohibitive due to their large memory and compute requirements. Furthermore, computing the matrix inverse or the Newton direction, which is needed in second-order methods, requires high precision computation for stable training as the preconditioner could have a large condition number. This paper provides a first attempt at developing computationally efficient sparse preconditioners for DNN training which can also tolerate low precision computation. Our new Sparsified Online Newton (SONew) algorithm emerges from the novel use of the so-called LogDet matrix divergence measure; we combine it with sparsity constraints to minimize the regret in the online convex optimization framework. Our mathematical analysis allows us to reduce the condition number of our sparse preconditioning matrix, thus improving the stability of training with low precision. We conduct experiments on a feed-forward neural-network autoencoder benchmark, where we compare training loss of optimizers when run for a fixed number of epochs. In the float32 experiments, our methods outperform the best-performing first-order optimizers and perform comparably to Shampoo, a state-of-the-art second-order optimizer. However, our method is even more effective in low-precision, where SONew finishes training considerably faster while performing comparably with Shampoo on training loss.

1 INTRODUCTION

Stochastic first order methods which use the negative gradient direction to update parameters have become the standard for training deep neural networks (DNNs). Gradient-based preconditioning involves finding an update direction by multiplying the gradient with a preconditioner matrix carefully chosen from gradients observed in previous iterations, to improve convergence. (Full-matrix) Adagrad (Duchi et al., 2011b), online Newton method (Hazan et al., 2007) and natural gradient descent (Amari, 1998) use a full-matrix preconditioner, but computing and storing the full matrix is infeasible when there are millions of parameters. Thus, diagonal versions such as diagonal Adagrad, Adam (Kingma & Ba, 2014), and RMSprop (Hinton et al., 2012) are now widely used to train DNNs due to their scalability.

Several higher-order methods have previously been applied to deep learning (Gupta et al., 2018; Anil et al., 2020; Goldfarb et al., 2020; Martens & Grosse, 2015). All these methods use Kronecker products that reduce computational and storage costs to make them feasible for training neural networks. However, these methods rely on matrix inverses or pth-roots that require high precision arithmetic as the matrices they deal with can have large condition numbers (Anil et al., 2020; 2022). Meanwhile, deep learning hardware accelerators have evolved towards using lower precision (bfloat16, float16, int8) (Henry et al., 2019; Jouppi et al., 2017) to reduce overall computational and memory costs and improve training performance. This calls for further research in developing efficient optimization techniques that work with low precision. Indeed there is recent work along these directions, from careful quantization of Adam (Dettmers et al., 2021) to 8-bits to optimizer agnostic local loss optimization (Amid et al., 2022) that leverage first-order methods to match higher-order methods.

In this paper, we present a first attempt towards computationally efficient *sparse* preconditioners for DNN training. Regret analysis when using a preconditioner reveals that the error is bounded by the sum of two terms (see (3) below); the first term depends on the change in the preconditioning matrix, while the second depends on the generalized gradient norm. We take the approach of minimizing

the second term while regularizing two successive preconditioners to be close in the LogDet matrix divergence measure (Kulis et al., 2009). This technique gives us an Online Newton method (Hazan et al., 2007). To make it computationally efficient, we further sparsify the preconditioner by finding a sparse approximation that is close in LogDet divergence. Thus we are consistent in using the same measure (LogDet divergence) in both the regularization and sparsification steps. This gives us our Sparsified Online Newton (SONew) method, which only requires $\mathcal{O}(n)$ time and memory complexity per iteration. We achieve this by imposing structured sparsity, such as tridiagonal and banded sparsity patterns, in the preconditioner. This is unlike most existing online Newton methods that require at least $\mathcal{O}(n^2)$ space and time complexity. By making each step linear time, the SONew method can be applied to training modern DNNs. Further, for some sparsity structures, our method is easily parallelized thus making negligible the overhead of computing the preconditioner. We also show that introducing sparsity allows us to reduce the condition number of our problem; as a consequence we show that our preconditioner can train DNNs even with low-precision arithmetic.

We establish regret bound guarantees of our algorithm in the online convex optimization framework. This involves using various properties about LogDet divergence and connections to other Bregman matrix divergences (Bregman, 1967), such as the von Neumann matrix divergence (Kulis et al., 2009). Our bound reveals an interesting behaviour about regret, where it decreases with increased correlation among gradients which are connected in the sparsity pattern/graph. We conduct experiments on an MLP Autoencoder, where we obtain better training loss compared to first order methods. We also compare our method in limited precision arithmetic (bfloat16) where we perform comparably with second-order methods, while being considerably faster.

2 BACKGROUND

The inner product between matrices is defined as $\langle A, B\rangle = \mathrm{Tr}(A^T B)$, where $\mathrm{Tr}(.)$ denotes the matrix trace. The Frobenius norm of a matrix A is $\|A\|_F = \sqrt{\mathrm{Tr}(A^T A)}$, while its spectral norm is $\|A\|_2 = \max_x \|Ax\|_2 / \|x\|_2$. We use S_n, S_n^+ S_n^{++} to denote the set of symmetric, positive semi-definite, and positive definite matrices respectively. The generalized norm of a vector $x \in \mathbb{R}^n$ with respect to matrix $A \in S_n^{++}$ is defined as $\|x\|_A = \sqrt{x^T A x}$. We use $\mathcal{G}$ and $\tilde{\mathcal{G}}$ to denote a graph and its sub-graph with a vertex set $[n]$. Let $E_{\mathcal{G}}$ denote set of edges in graph $\mathcal{G}$, and $\mathrm{neig}_{\mathcal{G}}(i)$ denote neighbours of vertex i in graph $\mathcal{G}$. The sparsity graph/pattern of a matrix $M \in \mathbb{R}^{n\times n}$ is a graph $\mathcal{G}$ with edges $E_{\mathcal{G}} = \{(i,j) : M_{ij} \neq 0\}$ corresponding to the non-zero entries in M. We extensively use $S_n(\mathcal{G})^{++}$ to denote the set of positive definite matrices with sparsity graph $\mathcal{G}$. Given an index set $I = \{i_1, i_2, .., i_n\}$, we denote $M_{I,I}$ to describe the corresponding principal sub-matrix of matrix M. $\mathrm{diag}(M)$ sets all off-diagonal entries in M to 0. $\det(M)$ is determinant of matrix M. Let $P_{\mathcal{G}} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ be defined as $P_{\mathcal{G}}(M)_{i,j} = M_{i,j}$ if $(i,j) \in E_{\mathcal{G}}$ else 0.

2.1 LOGDET MATRIX DIVERGENCE

Let $\phi : S_n^{++} \to \mathbb{R}$ be a strictly convex, differentiable function. Then the Bregman matrix divergence between $X, Y \in S_n^{++}$ is defined as (Kulis et al., 2009):

$$\mathrm{D}_\phi(X, Y) = \phi(X) - \phi(Y) - \mathrm{Tr}(\nabla\phi(Y)^T (X - Y)).$$

Since ϕ is convex, $\mathrm{D}_\phi(X, U) \geq 0$ for all $X, Y \succ 0$. A well known example is when $\phi(X) = \|X\|_F^2$, the corresponding Bregman divergence $\mathrm{D}_\phi(X, Y) = \|X - Y\|_F^2$ is the squared Frobenius norm. In this paper, we extensively use the divergence when the convex function is $\phi(X) = -\log\det(X)$; the corresponding divergence measure $\mathrm{D}_{\ell\mathrm{d}}(X, Y)$ is called the *LogDet matrix divergence*:

$$\mathrm{D}_{\ell\mathrm{d}}(X, Y) = -\log\det\left(XY^{-1}\right) + \mathrm{Tr}(XY^{-1}) - n. \tag{1}$$

Unlike the squared Frobenius norm, the LogDet divergence is scale invariant to invertible matrices M, i.e. $\mathrm{D}_{\ell\mathrm{d}}(M^T XM, M^T YM) = \mathrm{D}_{\ell\mathrm{d}}(X, Y)$, and can be written in the following revealing form in terms of eigendecompositions of $X = V\Lambda V^T$ and $Y = U\Theta U^T$ (Kulis et al., 2009):

$$\mathrm{D}_{\ell\mathrm{d}}(X, Y) = \sum_i \sum_j (v_i^T u_j)^2 (\lambda_i/\theta_j - \log(\lambda_i/\theta_j) - 1). \tag{2}$$

2.2 REGRET BOUND IN ONLINE CONVEX OPTIMIZATION

We set up our problem under the online convex optimization framework (OCO) (Hazan et al., 2016), where at each round the learner makes a prediction w_t in an online fashion and receives a loss $f_t(w_t)$ and gradient $g_t = \Delta f_t(w_t)$ as feedback. The goal of the learner is to reduce regret R_T by predicting w_t such that a low aggregate loss $\sum_{t=1}^{T} f_t(w_t)$ is achieved compared to the minima $w^* = \arg\min_w \sum_{t=1}^{T} f_t(w)$. Formally, the regret is given by

$$R_T(w_1, \ldots, w_T) = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(w^*).$$

3 SONEW: SPARSIFIED ONLINE NEWTON METHOD

We now present our proposed SONew algorithm.

3.1 REGRET MINIMIZATION VIA LOGDET DIVERGENCE

We proceed as in Hazan et al. (2016) by analyzing the error of the iterates (where the minima is $w^* \in \mathbb{R}^n$) for the update $w_{t+1} := w_t - \eta X_t g_t$, where $X_t \in \mathbb{R}^{n \times n}$ is a preconditioner matrix. Then $\|w_{t+1} - w^*\|^2_{X_t^{-1}} = \|w_t - \eta X_t g_t - w^*\|^2_{X_t^{-1}} = \|w_t - w^*\|^2_{X_t^{-1}} + \eta^2 g_t^T X_t g_t - 2\eta(w_t - w^*)^T g_t$. The convexity of f_t, $f_t(w_t) - f_t(w^*) \leq (w_t - w^*)^T g_t$ implies that $f_t(w_t) - f_t(w^*) \leq \|w_t - w^*\|^2_{X_t^{-1}} - \|w_{t+1} - w^*\|^2_{X_t^{-1}} + \eta^2 g_t^T X_t g_t$. Summing over all $t \in [T]$ and rearranging reveals the following upper bound on overall regret:

$$R_T \leq \frac{1}{2\eta} \cdot \left(\|w_1 - w^*\|^2_{X_1^{-1}} + \sum_{t=2}^{T} \|w_t - w^*\|^2_{X_t^{-1} - X_{t-1}^{-1}} \right) + \frac{\eta}{2} \sum_{t=1}^{T} g_t^T X_t g_t. \tag{3}$$

Since w^* is unknown, finding the X_t which minimizes the first term in equation 3 is infeasible. So to minimize regret, we attempt to minimize the last term in equation 3 while regularizing X_t^{-1} to be "close" to X_{t-1}^{-1}. The nearness measure we choose is the LogDet matrix divergence, thus leading to the following objective

$$X_t = \arg\min_{X \in S_n^{++}} g_t^T X g_t, \quad \text{such that} \quad \mathrm{D}_{\ell\mathrm{d}}\left(X, X_{t-1}\right) \leq c_t. \tag{4}$$

Why do we use the LogDet divergence? From equation 2, due to the term λ_i/θ_j, $\mathrm{D}_{\ell\mathrm{d}}(X, Y)$ prioritizes matching the smaller eigenvalues of X with those of Y, which in turn are the larger eigenvalues of X^{-1} and Y^{-1}. As a consequence, LogDet divergence regularizes Y^{-1} and X^{-1} by matching up their large eigenvalues. This intuition leads us to formulate equation 4 as our objective. There is precedence of using the LogDet divergence in the optimization literature; indeed the celebrated BFGS algorithm (Broyden, 1967; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970) can be shown to be the unique solution obtained when the LogDet divergence between successive preconditioners, subject to a secant constraint, is minimized (as shown in the beautiful 4-page paper Fletcher (1991)).

The problem in Equation (4) is convex in X since the LogDet divergence is convex in its first argument. So the Lagrange function $\mathcal{L}(X, \lambda_t) = g_t^T X g_t + \lambda_t(\mathrm{D}_{\ell\mathrm{d}}(X, X_{t-1}) - c_t) = \mathrm{Tr}(X g_t g_t^T) + \lambda_t(-\log\det\left(X X_{t-1}^{-1}\right) + \mathrm{Tr}(X X_{t-1}^{-1}) - d))$ and the KKT condition $\nabla\mathcal{L}(X, \lambda_t) = 0$ gives the following update rule for the preconditioner:

$$X_t^{-1} = X_{t-1}^{-1} + g_t g_t^T / \lambda_t. \tag{5}$$

Setting $c_t = 0$ in Equation 4 when $X_1 = I_n$ (equivalent to not changing the preconditioner in any round), will result in online gradient descent (Zinkevich, 2003). Setting $\lambda_t = 1$ gives the update rule of the online Newton method. However for large n, maintaining and updating X_t^{-1} as in (5) takes $\mathcal{O}(n^2)$ storage and time complexity, which is impractical when n is in the order of millions which is typically the case in DNNs.

3.2 SPARSE PRECONDITIONING WITH SONEW

To reduce the memory required to store X_t we explore sparse alternatives to solving the subproblem in Equation (4), by introducing additional sparsity constraints. We could add an L_0-norm penalty $\lambda \|X\|_0$ to the objective, but the resulting subproblem is non-convex, which can be difficult to solve exactly. The L_1-norm penalty serves as a convex relaxation for the L_0 norm penalty, and results in the sparse inverse covariance estimation (also known as graphical lasso) problem (Friedman et al., 2008; Hsieh et al., 2014):

$$X_t = \underset{X \in S_n^{++}}{\arg\min}\, \mathrm{D}_{\ell\mathrm{d}}\left(X, (X_{t-1}^{-1} + g_t g_t^T/\lambda_t)^{-1}\right) + \lambda \|X\|_1 .$$

The sparsity introduced by L_1 penalty will reduce the memory usage. However, the time taken to solve a graphical lasso estimator, even with the current best methods (Hsieh et al., 2013; Fattahi & Sojoudi, 2019; Zhang et al., 2018), can still be too large (several minutes for a matrix of size million) making it impractical to embed in DNN training since preconditioning happens immediately after processing every mini-batch.

In this paper, we take a different direction where we use use sparsity pattern constraints. Our sparsity constraints are represented by a fixed undirected graph $\mathcal{G}$. The sub-problem will be as follows

$$X_t = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min}\, \mathrm{D}_{\ell\mathrm{d}}\left(X, (X_{t-1}^{-1} + g_t g_t^T/\lambda_t)^{-1}\right), \tag{6}$$

where $S_n(\mathcal{G})^{++}$ denotes the set of positive definite matrices with the fixed sparsity pattern given by graph $\mathcal{G}$. Solving the subproblem in Equation (6) is seemingly impractical since X_{t-1}^{-1} can be a dense matrix. However, the structure of the LogDet divergence comes to the rescue; Equation 6 can be expanded as follows:

$$\begin{aligned} X_t &= \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} -\log\det(X) + \mathrm{Tr}(X(X_{t-1}^{-1} + g_t g_t^T/\lambda_t)), \\ &= \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} -\log\det(X) + \mathrm{Tr}(X P_{\mathcal{G}}(X_{t-1}^{-1} + g_t g_t^T/\lambda_t)). \end{aligned} \tag{7}$$

The main observation to make is $\mathrm{Tr}(X(X_{t-1}^{-1} + g_t g_t^T/\lambda_t)) = \mathrm{Tr}(X P_{\mathcal{G}}(X_{t-1}^{-1} + g_t g_t^T/\lambda_t))$, since $X \in S_n(\mathcal{G})^{++}$, where $P_{\mathcal{G}} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is defined as follows:

$$P_{\mathcal{G}}(M)_{i,j} = \begin{cases} M_{i,j} & \text{if } (i,j) \in E_{\mathcal{G}} \\ 0 & \text{otherwise} \end{cases} . \tag{8}$$

We show that the entire matrix X_{t-1}^{-1} need not be maintained, by analyzing the optimality condition of Equation 7, which is $\nabla(-\log\det X + \mathrm{Tr}(X P_{\mathcal{G}}(X_{t-1}^{-1} + g_t g_t^T/\lambda_t))) = 0$,

$$\begin{aligned} &P_{\mathcal{G}}(X_t^{-1}) = P_{\mathcal{G}}(X_{t-1}^{-1} + g_t g_t^T/\lambda_t) = P_{\mathcal{G}}(X_{t-1}^{-1}) + P_{\mathcal{G}}(g_t g_t^T/\lambda_t), \\ &\implies H_t = H_{t-1} + P_{\mathcal{G}}(g_t g_t^T/\lambda_t). \end{aligned} \tag{9}$$

Thus we only need to maintain $H_t \in S_n(\mathcal{G})^{++}$, which is $H_t = P_{\mathcal{G}}(X_t^{-1})$ and update it as $H_t = H_{t-1} + P_{\mathcal{G}}(g_t g_t^T/\lambda_t)$. This update can be done in $\mathcal{O}(|E_{\mathcal{G}}|)$ memory and time, while storing the entire matrix X_t^{-1} would have cost $\mathcal{O}(n^2)$. In SONew, this key observation is used to maintain H_t. Now that H_t is known at every round t, we can replace $P_{\mathcal{G}}(X_{t-1}^{-1} + g_t g_t^T/\lambda_t)$ in (7) with H_t as:

$$X_t = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} -\log\det(X) + \mathrm{Tr}(X H_t) = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min}\, \mathrm{D}_{\ell\mathrm{d}}(X, H_t^{-1}). \tag{10}$$

For an arbitrary graph $\mathcal{G}$, solving this subproblem might be difficult. However, it can be solved efficiently for banded and tridiagonal sparsity graphs. A banded graph of bandsize b consists of b subdiagonals apart from the main diagonal. A tridiagonal/chain graph is a banded graph with bandsize 1. The following theorem shows *embarrassingly parallelizable* explicit solutions to the subproblem 10 for tridiagonal and banded sparsity patterns.

Algorithm 1 SONew algorithm

Inputs: λ_t,
$\mathcal{G} \coloneqq$ banded/tridiagonal sparsity graph
$\epsilon \coloneqq$ damping parameter,
$T \coloneqq$ total number of rounds/iterations,
$\eta_t \coloneqq$ step size/learning rate,
Outputs: w_{T+1}
$H_0 = \epsilon I_d$, $w_1 = 0$
for $t \in \{1, \ldots, T\}$ **do**
  compute $g_t = \nabla f_t(w_t)$
  $H_t \coloneqq H_{t-1} + P_{\mathcal{G}}(g_t g_t^T/\lambda_t)$, with $P_{\mathcal{G}}$ as in (8) – can be computed in $\mathcal{O}(n)$ time & memory.
  Invoke Algorithm 2, which returns L, D such that $X_t = LDL^T$
  where $X_t = \arg\min_{X \in S_n(\mathcal{G})^{++}} -\log\det(X) + \mathrm{Tr}(XH_t)$ as defined in (10).
  Compute update $u_t = LDL^T g_t$,
  $w_{t+1} = w_t - \eta_t u_t$
end for
return w_{T+1}

Theorem 1 (Explicit solution for LogDet subproblem using $\mathcal{O}(n)$ flops)**.** *Let the sparsity pattern $\mathcal{G}$ be a banded matrix of band size b, i.e. for every vertex j, let $I_j = \{j+1, \ldots, j+b\}$, then edges $E_{\mathcal{G}} = (\bigcup_{j=1}^{n} \{j\} \times I_j) \cap \{(i,j) : i \leq n, j \leq n\}$.*

$$\hat{X}^{(b)} = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} \ \mathrm{D}_{\ell\mathrm{d}}(X, H_t^{-1}).$$

Let the Cholesky decomposition of the solution $\hat{X}^{(b)} = LDL^T$, then D is a diagonal matrix and L is a lower triangular sparse matrix with non-zero entries defined as follows:

$$L_{jj} = 1, \ L_{I_j j} = -H_{I_j I_j}^{-1} H_{I_j j}, \ \ D_{jj} = 1/(H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j}). \tag{11}$$

Algorithm 2 Explicit solution for LogDet subproblem using $\mathcal{O}(n)$ flops

Inputs: $\mathcal{G} \coloneqq$ the banded graph of band size b, typically small.
$H \in S_n(\mathcal{G})^{++}$, as defined in Equation 9.
Output: lower triangular sparse $L \in \mathbb{R}^{n \times n}$ and diagonal matrix D, such that $\hat{X}^{(b)} = LDL^T \in S_n(\mathcal{G})^{++}$
$L \coloneqq 0$, $D \coloneqq 0$
$L_{jj} \coloneqq 1, \forall j \in [n]$
Let $I_j = \{j+1, \ldots, j+b\}$ and let $H_{I_j j} = [H_{j+1,j}, H_{j+2,j}, \ldots, H_{j+b,j}] \in \mathbb{R}^b$, and $H_{I_j I_j} \in \mathbb{R}^{b \times b}$ be principal submatrix of H corresponding to indices I_j
Find $L_{I_j j} \in \mathbb{R}^n$ by solving linear systems $H_{I_j, I_j} L_{I_j j} = -H_{I_j, j} \ \forall j \in [n]$. This operation has a complexity $\mathcal{O}(nb^3) = \mathcal{O}(n)$ (since b is small), but is embarassingly parallelizable over n
$D_{j,j} \coloneqq 1/(H_{jj} + H_{I_j j}^T L_{I_j j})$
return L, D

Finding the above solution requires solving n linear systems of size b (which is small) as shown in Algorithm 2, which can cost $\mathcal{O}((n-b+1)b^3)$ flops, however it can be parallelized across the parameter dimension, giving a parallel complexity of $\mathcal{O}(b^3)$. When $b = 1$, the resulting tridiagonal sparsity pattern admits a simple solution requiring $\mathcal{O}(n)$ flops and a parallel runtime complexity of $\mathcal{O}(1)$. For $i < j$

$$(\hat{X}^{(1)})_{i,j} = \begin{cases} 0 & \text{if } j - i > 1 \\ \frac{-H_{i,i+1}}{(H_{ii}H_{i+1,i+1} - H_{i+1,i+1}^2)} & \text{if } j = i+1 \\ \frac{1}{H_{ii}} \left(1 + \sum_{j \in \mathrm{neig}_{\mathcal{G}}(i)} \frac{H_{ij}^2}{H_{ii}H_{jj} - H_{ij}^2}\right) & \text{if } i = j \end{cases}, \tag{12}$$

where, $\mathrm{neig}_{\mathcal{G}}(i)$ denotes the neighbours of vertex i in the chain graph $\mathcal{G}$.

3.3 ANALYSIS OF SONEW

The following theorem establishes regret guarantees of SONew in online convex optimization framework mentioned in Section 2.2.

Theorem 2. *When $\mathcal{G}$ = tridiagona/chain graph, then Algorithm 1 incurs the following regret under convex losses f_t*

$$R_T \leq \mathcal{O}\left(C^{1/2} \cdot T^{3/4} \cdot \left(\frac{1+\beta}{1-\beta}\right)^{1/2} \cdot \left(\sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1-\beta_i^2\right)\right)^{3/4}\right),$$

where $g_{1:T}^{(i)} = [(g_1)_i, \dots (g_T)_i]$ denotes gradient history of ith variable/parameter, ,$G_\infty^{(i)} = \|g_{1:T}^{(i)}\|_\infty$, $C = \max_t(\sum_{i=1}^{n}(w_t - w^*)_i^2 (G_\infty^{(i)})^2)$, $\beta_i = \frac{\left\langle g_{1:T}^{(i)}, g_{1:T}^{(i+1)}\right\rangle}{\sqrt{(\epsilon + \left\|g_{1:T}^{(i)}\right\|_2^2)\cdot(\epsilon + \left\|g_{1:T}^{(i+1)}\right\|_2^2)}}$ denotes dot-product between gradient histories of connected parameters in the chain graph, $\beta = \max_{i\in[n-1]} \beta_i$.

SONew is *scale-invariant* to diagonal transformation of gradients; if the gradients sent as feedback to the OCO learner are $\bar{g}_t = \Lambda g_t$, then the iterates for the transformed problem will be $\bar{x}_t = \Lambda^{-1} w_t = \Lambda^{-1} x_{t-1} - \eta(\Lambda^{-1} X_{t-1} \Lambda^{-1})\Lambda(g_{t-1})$, this is due to scale-invariance property of LogDet divergence in Section 2.1,i.e, the transformed preconditioner is $\bar{X}_t = \Lambda^{-1} X_{t-1} \Lambda^{-1}$. Furthermore, the regret bound derived in Theorem 2 for transformed and original problem is approximately same. Our regret bound is data-dependent, since if β_i is nearer to 1, then the regret is smaller due to the $\log(1-\beta_i^2)$ term. This effect is more amplified when $\log(1 + \|g_{1:T}^{(i)}\|_2^2/\epsilon)$ is relatively low compared to $\log(1-\beta_i^2)$. The proof for Theorem 2 is deferred to Appendix A.2.

4 NUMERICAL STABILITY OF SONEW

In finite precision computation, an operation with condition number κ can lose upto $\log_2(\kappa)$ bits of accuracy in its result. This issue manifests in a naive implementation of Algorithm 2, where for some of the vertices in the sparsity graph, in line 7 of Algorithm 3, $(H_{jj} + H_{I_j,j}^T L_{I_j,j}) = (H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j})$ becomes zero and sometimes negative. This is due to illconditioning of submatrices $H_{J_j J_j}$ and $H_{I_j I_j}$ as a result either the Schur complement $H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j}$ can be low or computing $H_{I_j I_j}^{-1} H_{I_j j}$ can be error prone, where $J_j = I_j \cup \{j\}$.

We conduct perturbation analysis of a normalized form of tridiagonal LogDet subproblem to establish a component-wise condition number (Higham, 2002).

Theorem 3 (Condition number of normalized tridiagonal LogDet subproblem)**.** *Let $H \in S_n^{++}$ such that $H_{i,i} = 1$ for $i \in [n]$, and let ΔH be a symmetric perturbation such that $\Delta H_{i,i} = 0$ for $i \in [n]$, and $H + \Delta H \in S_n^{++}$. Then*

$$\kappa_\infty^{\ell d} \leq \max_{i\in[n-1]} 2/(1-\beta_i^2) = \hat{\kappa}_\infty^{\ell d}, \tag{13}$$

where, $\beta_i = H_{i,i+1}$,$\kappa_\infty^{\ell d}$:= componentwise condition number of tridiagonal LogDet solution in (12) [1]

The following observation describes cases when the preconditioner cannot be derived using (14).

Lemma 1 (Degenerate inputs to banded LogDet subproblem)**.** *Let $H = GG^T$, where $G \in \mathbb{R}^{n\times T}$ and let $g_{1:T}^{(i)}$ be i^{th} row of G, equivalent to history of gradients of parameter i, then $H_{i,j} = \left\langle g_{1:T}^{(i)}, g_{1:T}^{(j)}\right\rangle$.*

- *For $b = 1$ in Equation (14),i.e., tridiagonal sparsity pattern is used: if $g_{1:T}^{(j)} = g_{1:T}^{(j+1)}$, then $H_{jj} - H_{j,j+1}^2 / H_{j+1,j+1} = 0$ and D_{jj} is undefined*
- *For $b > 1$: If $\operatorname{rank}(H_{J_j,J_j}) = \operatorname{rank}(H_{I_j,I_j}) = b$, then $(H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j}) = 0$ and D_{jj} is undefined. If $\operatorname{rank}(H_{I_j,I_j}) < b$ then $(H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j})$ is undefined.*

[1] The full version of Theorem 3 can be referred from Appendix A.3.

If H is a PSD matrix, then solution to the LogDet problem might not be well-defined since rank of submatrices involved in the explicit solution computation in Equation (14) might be low as shown in the Lemma 1. For instance, Case 1 can occur when preconditioning the input layer of an image-based DNN, where j^{th} and $(j+1)^{th}$ pixel can be equal or highly correlated throughout the dataset. Case 2 can occur in the first b iterations in Algorithm 1 when the rank of submatrices $\mathrm{rank}(H_{I_j,I_j}) < b$.[2]

We develop Algorithm 3 to handle PSD inputs H to banded LogDet problem, given that $H_{ii} > 0$. The algorithm finds a subgraph $\tilde{\mathcal{G}}$ of $\mathcal{G}$ which is not degenerate. This is done by removing edges which causes inverse $H^{-1}_{I_jI_j}$ to be undefined or $H_{jj} - H^T_{I_j,j} H^{-1}_{I_jI_j} H_{I_jj})$ to be low. Our ablation study in Table 4 demonstrates noticeable difference in performance with and without using Algorithm 3.

Theorem 4 (Numerically stable algorithm)**.** *Algorithm 3 finds a subgraph $\tilde{\mathcal{G}}$ of $\mathcal{G}$, such that explicit solution for $\tilde{\mathcal{G}}$ in (14) is well-defined. Furthermore, when $\mathcal{G}$ is a tridiagonal/chain graph, the component-wise condition number upperbound in (13) is reduced upon using Algorithm 3, $\hat{\kappa}^{\tilde{\mathcal{G}}}_{\ell d} < \hat{\kappa}^{\mathcal{G}}_{\ell d}$, where $\hat{\kappa}^{\tilde{\mathcal{G}}}_{\ell d}$, $\hat{\kappa}^{\mathcal{G}}_{\ell d}$ are defined as in Theorem 3 for graphs $\tilde{\mathcal{G}}$ and $\mathcal{G}$ respectively.*

Algorithm 3 Numerically stable banded LogDet solution

1: Input: $\mathcal{G}-$ tridiagonal or banded graph, $H-$ PSD matrix with $H_{ii} > 0$, $\gamma-$ threshold
2: Output: Finds subgraph $\tilde{\mathcal{G}}$ of $\mathcal{G}$ without any degenerate cases from Lemma 1 and finds preconditioner $\hat{X}$ corresponding to the subgraph
3: Let $E_i = \{(i,j) : (i,j) \in E_{\mathcal{G}}\}$ be edges from vertex i to its neighbours in graph $\mathcal{G}$.
4: Let $V_i^+ = \{j : i < j, (i,j) \in E_{\mathcal{G}}\}$ and $V_i^- = \{j : i > j, (i,j) \in E_{\mathcal{G}}\}$, denote positive and negative neighbourhood of vertex i.
5: Let $K = \left\{i : H_{ii} - H^T_{I_ii} H^{-1}_{I_iI_i} H_{I_ii} \text{ is undefined or } \leq \gamma, i \in [n]\right\}$
6: Consider a new subgraph $\tilde{\mathcal{G}}$ with edges $E_{\tilde{\mathcal{G}}} = E_{\mathcal{G}} \setminus ((\bigcup_{i \in K} E_i) \cup (V_i^+ \times V_i^-))$
7: return $\hat{X} = \arg\min_{X \in S_n(\tilde{\mathcal{G}})^{++}} \mathrm{D}_{\ell \mathrm{d}}\left(X, H_t^{-1}\right)$ solved via Equation (14)

5 Related Work

Online Newton method is a second order method in online convex optimization framework with properties such as scale invariance (Luo et al., 2016) and logarithmic regrets in exp-concave and strongly convex functions (Hazan et al., 2007; 2016). However, it has a time complexity of $\mathcal{O}(n^2)$, making it infeasible for large n. A diagonal version of this method SC-Adagrad (Mukkamala & Hein, 2017) was proposed to make the online Newton method scalable for deep-learning. SC-Adagrad is equivalent to setting the sparsity pattern $\mathcal{G}$ in equation 10 as a null graph, furthermore, setting $\mathcal{G}$ as a complete graph will result in online Newton method. However, introduction of LogDet divergence measure in SONew allows us to set different sparsity graphs as $\mathcal{G}$ such as banded graph with band-size b, for which our preconditioning process is more computationally efficient with a time complexity of $\mathcal{O}(b^3(n-b+1))$ compared to online-newton method $\mathcal{O}(n^2)$. As discussed in Theorem 2, SONew also utilizes correlation among gradients of parameters to improve convergence, in contrast to diagonal preconditioners such as SC-Adagrad.

Shampoo (Gupta et al., 2018; Anil et al., 2020) uses Kronecker factored preconditioners to reduce the memory and time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(d_1^2 + d_2^2)$ and $\mathcal{O}(d_1^3 + d_2^3)$ respectively, where $n = d_1 d_2$ denotes number of parameters for a linear layer of dimensions $d_1 \times d_2$. The time complexity of matrix inversion takes a heavy toll in Shampoo's compute time even with the Kronecker product assumption on the preconditioner, whereas, our method has a time complexity of $\mathcal{O}(b^3 d_1 d_2)$ quadratic in dimensions of the linear layer (note that $b = 1$ for tridiagonal structure). Furthermore, precision lost in matrix inversion can be large due to high condition number matrices occurring in training (Anil et al., 2022; 2020).

Solving the LogDet problem in equation 10 is closely related to the Maximum Determinant Matrix Completion (MDMC) problem (Andersen et al., 2013; Vandenberghe et al., 2015): $\hat{X} = \arg\min_{\Pi \in \bar{\mathcal{G}}, H+\Pi \succ 0} \log\det(H + \Pi)$, where $\bar{\mathcal{G}}$ is the complement of graph $\mathcal{G}$. It can be shown that MDMC is the dual of the LogDet problem, and has explicit solutions for chordal graphs (Andersen et al., 2013). Thus the explicit solutions in Equation 14 for the primal problem are the same as the ones proved in Andersen et al. (2013). Also, we noticed that the tridiagonal explicit solution

[2]This is proved in Appendix A.3

has been used previously in KFAC (Martens & Grosse, 2015) in the context of a gaussian graphical model interpretation of gradients. However, we also study numerical stability and degenerate cases of these explicit solutions which can occur frequently in DNN training and derive algorithms to mitigate such issues. In addition, we also have regret guarantees connecting these explicit solutions to regret in online convex optimization problem setup.

6 EXPERIMENTAL RESULTS

We conduct experiments on the Autoencoder benchmark (Schmidhuber, 2015) using MNIST dataset (Deng, 2012). We compare SONew against commonly used first order methods including SGD (Kiefer & Wolfowitz, 1952)), SGD with Momentum (Qian, 1999), Nesterov (Nesterov, 1983), Adagrad (Duchi et al., 2011a), Adam (Kingma & Ba, 2014), and Rmsprop (Tieleman & Hinton, 2012). Among second-order optimizers, we benchmark against Shampoo (Gupta et al., 2018), a state of the art optimizer used in practice. Computing the preconditioner in shampoo at every training step could be infeasible, we employ a commonly used technique to compute the preconditioner every t steps - referred to as Shampoo(t) in the experiments.

All experiments for Autoencoder are conducted on one V100 GPU. For SONew, we use exponentially moving average (referred to as momentum in the literature) for both first order and second order statistics. Concretely, let $\beta_1, \beta_2 \in [0, 1]$ be two hyperparameters. Let $g_t \in \mathbb{R}^n$ be the gradient in the tth step. Let $\mu_t \in \mathbb{R}^n$ be the first order gradient statistic, and $H_t \in S_n(\mathcal{G})^{++}$ be the second order gradient statistic as in Equation 9. Then, the following new update rules are used to maintain μ_t and H_t in the SONew implementation.

$$\mu_t = \beta_1 \mu_{t-1} + (1 - \beta_1) g_t, \quad H_t = \beta_2 H_{t-1} + (1 - \beta_2) g_t g_t^T.$$

Furthermore, the first-order statistic/momentum μ_t is used in $w_{t+1} = w_t - \eta_t X_t \mu_t$, replacing the gradient g_t. As discussed in Section 3.2 we need to store the values of H_t only corresponding to the sparsity pattern, hence SONew uses $\mathcal{O}(n)$ space. The updates above get computed in parallel in $\mathcal{O}(1)$ time. Moreover, we use Algorithm 3 to make SONew numerically stable. We also use grafting (Agarwal et al., 2022), a technique used to transfer step size between optimization algorithms. Specifically, given an update v_1 of Optimizer-1 and v_2 of Optimizer-2, grafting allows to use the direction suggested by Optimizer-2 with step size suggested by Optimizer-1. The final update is given by $\frac{\|v_1\|}{\|v_2\|} \cdot v_2$. Grafting has been shown to take advantage of a tuned optimizer step size and improve performance. In our case, we use Adam grafting - using Adam optimizer step size $\|v_1\|$ with SONew direction $v_2/\|v_2\|$.

We use three sparsity patterns for SONew - a) diagonal sparsity, resulting in a diagonal preconditioner similar to adaptive first order methods like Adam and Adagrad; b) tridiagonal sparsity, corresponding to a chain graph; and c) banded sparsity, represented by "band-k" in tables and figures for band size of k.

All the baselines and SONew are trained for 100 epochs and use the train set of MNIST dataset containing $60k$ points. We use a standard sized (2.72M parameters) Autoencoder with layer sizes: [1000, 500, 250, 30, 250, 500, 1000] and tanh non-linearity. Batch size is fixed at 1000 and we use learning rate schedule with a linear warmup of 5 epochs followed by linear decay towards 0. Over $2k$ hyperparameters are searched over for each experiment using a Bayesian optimization package. The search space for each optimizer is mentioned in Appendix A.4.

From experiments using float32 precision in Table 1 we observe that among first order methods, diag-SONew performs the best, followed by RMSprop. Increasing the number of edges in the sparsity graph to tridiag or banded with band size 4 enhances the performance further. We observe that diag-SONew takes same time as first order methods like Adam and RMSProp. tridiag-SONew performs significantly better (1.6 points better loss) with marginal overhead. Compared to the second order method Shampoo, tridiag-SONew perform $4\times$ faster at a marginal cost to the loss - even when Shampoo updates preconditioner once every 20 steps. When dealing with large scale models like Resnet50 (He et al., 2015), Shampoo requires updating preconditioner much more often (Anil et al., 2019), whereas our method is also promising in such scenarios. We leave testing on such large benchmarks as a future work.

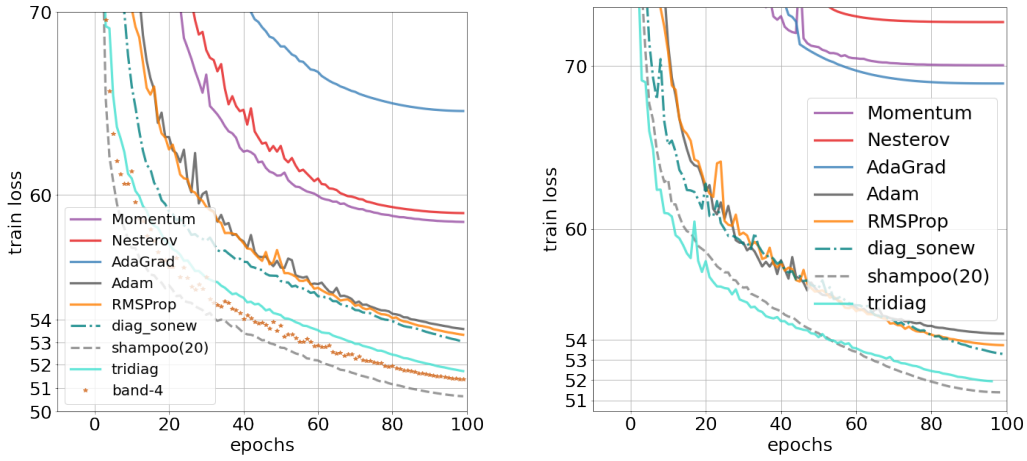
To show efficacy of SONew at lower precision, we conduct bfloat16 experiments. We notice in Table 2 that diag-SONew performs the best in first order methods, just like in float32. Moreover

Table 1: **float32 experiments on Autoencoder benchmark.** "tridiag" repersents tridiag-SONew, and "band-4" represents banded-SONew with band size 4. We observe that diag-SONew performs the best among all first order methods while taking similar time. tridiag and band-4 perform significantly better than first order methods while requiring similar linear space and being marginally slower. Shampoo performs best but takes $\mathcal{O}(d^3)$ time for computing preconditioner of a linear layer of size $d \times d$, whereas our methods take $\mathcal{O}(d^2)$ to find the precondioner.

Optimizer	First Order Methods							Second Order Methods		
	SGD	Nesterov	Adagrad	Momentum	RMSProp	Adam	diag-SONew	Shampoo(20)	tridiag	band-4
Train CE loss	67.654	59.087	54.393	58.651	53.330	53.591	53.025	50.702	51.723	51.357
Time(s)	62	102	62	67	62	62	63	371	87	251

Table 2: **bfloat16 experiments on Autoencoder benchmark.** diag-SONew performs the best among all first order methods, while degrading only marginally (0.26 absolute difference) compared to float32 performance. tridiag-SONew holds similar observation as well, with only 0.35 increase in loss. Shampoo performs the best but has a considerable drop (0.70) in performance compared to float32 due to using matrix inverse, and is slower due to its cubic time complexity for computing preconditioners. Shampoo implementation uses 16-bit quantization to make it work in 16-bit setting, leading to further slowdowns. Hence the running time in bfloat16 is even more than in float32.

Optimizer	First Order Methods							Second Order Methods	
	SGD	Nesterov	Adagrad	Momentum	RMSProp	Adam	diag-SONew	Shampoo(20)	tridiag
Train CE loss	80.454	72.975	68.854	70.053	53.743	54.328	53.29	51.401	51.937
Train time(s)	36	43	37	36	37	38	44	1245	70



(a) float32 (b) bfloat16

Figure 1: (a) Comparison of first order methods, second order method Shampoo, and our algorithm SONew on Autoencodeer benchmark trained on MNIST. Left (a) uses float32 training and right (b) uses bfloat16 training. We observe that SONew method like tridiag has early epochs advantage over shampoo in lower precision, and performs better than all first order methods in float32 and bfloat16.

SONew undergoes the least degradation in performance compared to all other optimizers. We also provide an ablation study on effect of using Algorithm 3 on training loss in the appendix.

7 Conclusions and Future Work

In this paper we have introduced a computationally efficient sparse preconditioner based second order optimizer. Our algorithm arises from a novel regret bound analysis using LogDet Divergence, and furthermore we make it numerically stable. Experimental results on the Autoencoder benchmark confirm the effectiveness of SONew in both float32 as well as bfloat16 precision.

In the future, we will test SONew on larger benchmarks like DLRM (Naumov et al., 2019) and ResNet50 (He et al., 2015), and explore different sparsity graphs for which efficient solutions exist for the LogDet subproblem (10).

REFERENCES

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Learning rate grafting: Transferability of optimizer tuning, 2022. URL https://openreview.net/forum?id=FpKgG31Z_i9.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Ehsan Amid, Rohan Anil, and Manfred Warmuth. Locoprop: Enhancing backprop via local loss optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 9626–9642. PMLR, 2022.

Martin S Andersen, Joachim Dahl, and Lieven Vandenberghe. Logarithmic barriers for sparse matrix cones. *Optimization Methods and Software*, 28(3):396–423, 2013.

Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory-efficient adaptive optimization, 2019. URL https://arxiv.org/abs/1901.11150.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

Rohan Anil, Sandra Gadanho, Da Huang, Nijith Jacob, Zhuoshu Li, Dong Lin, Todd Phillips, Cristina Pop, Kevin Regan, Gil I Shamir, et al. On the factory floor: ML engineering for industrial-scale ads recommendation models. *arXiv preprint arXiv:2209.05310*, 2022.

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

Charles G Broyden. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967.

Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011a. URL http://jmlr.org/papers/v12/duchi11a.html.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011b.

Salar Fattahi and Somayeh Sojoudi. Graphical lasso and thresholding: Equivalence and closed-form solutions. *Journal of machine learning research*, 2019.

Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

Roger Fletcher. A new variational result for quasi-Newton formulae. *SIAM Journal on Optimization*, 1(1):18–21, 1991.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-Newton methods for training deep neural networks. *Advances in Neural Information Processing Systems*, 33:2386–2396, 2020.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learningfor image recognition. *ComputerScience*, 2015.

Greg Henry, Ping Tak Peter Tang, and Alexander Heinecke. Leveraging the bfloat16 artificial intelligence datatype for higher-precision computations. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pp. 69–76. IEEE, 2019.

Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

Cho-Jui Hsieh, Mátyás A Sustik, Inderjit S Dhillon, Pradeep K Ravikumar, and Russell Poldrack. BIG & QUIC: Sparse inverse covariance estimation for a million variables. *Advances in neural information processing systems*, 26, 2013.

Cho-Jui Hsieh, Mátyás A Sustik, Inderjit S Dhillon, Pradeep Ravikumar, et al. QUIC: quadratic approximation for sparse inverse covariance estimation. *J. Mach. Learn. Res.*, 15(1):2911–2947, 2014.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pp. 1–12, 2017.

J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952. doi: 10.1214/aoms/1177729392. URL https://doi.org/10.1214/aoms/1177729392.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Brian Kulis, Mátyás A Sustik, and Inderjit S Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 10(2), 2009.

G Lindblad. Completely positive maps and entropy inequalities. 1975.

Haipeng Luo, Alekh Agarwal, Nicolo Cesa-Bianchi, and John Langford. Efficient second order online learning by sketching. *Advances in Neural Information Processing Systems*, 29, 2016.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Mahesh Chandra Mukkamala and Matthias Hein. Variants of rmsprop and adagrad with logarithmic regret bounds. In *International conference on machine learning*, pp. 2545–2553. PMLR, 2017.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o(1/k^2). 1983.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(98)00116-6. URL https://www.sciencedirect.com/science/article/pii/S0893608098001166.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, jan 2015. doi: 10.1016/j.neunet.2014.09.003. URL https://doi.org/10.1016%2Fj.neunet.2014.09.003.

David F Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

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.

Lieven Vandenberghe, Martin S Andersen, et al. Chordal graphs and semidefinite optimization. *Foundations and Trends® in Optimization*, 1(4):241–433, 2015.

Richard Zhang, Salar Fattahi, and Somayeh Sojoudi. Large-scale sparse inverse covariance estimation via thresholding and max-det matrix completion. In *International Conference on Machine Learning*, pp. 5766–5775. PMLR, 2018.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

A Appendix

A.1 Properties of LogDet subproblem

Theorem 5 (Explicit solution for LogDet subproblem using $\mathcal{O}(n)$ flops)**.** *Let the sparsity pattern $\mathcal{G}$ be a banded matrix of band size b, i.e. for every vertex j, let $I_j = \{j+1, \ldots, j+b\}$, then edges $E_{\mathcal{G}} = (\bigcup_{j=1}^{n} \{j\} \times I_j) \cap \{(i,j) : i \leq n, j \leq n\}$.*

$$\hat{X}^{(b)} = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} \; \mathrm{D}_{\ell\mathrm{d}}\left(X, H_t^{-1}\right)$$

Let the Cholesky decomposition of the solution $\hat{X}^{(b)} = LDL^T$, then D is a diagonal matrix and L is a lower triangular sparse matrix with non-zero entries defined as follows:

$$L_{jj} = 1, \; L_{I_j j} = -H_{I_j I_j}^{-1} H_{I_j j}, \;\; D_{jj} = 1/(H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j}) \tag{14}$$

Proof. The optimality condition of the above problem is

$$P_{\mathcal{G}}(X^{-1}) = P_{\mathcal{G}}(H) \tag{15}$$

Let $Z = L^{-T} D^{-1} L^{-1}$, then $P_{\mathcal{G}}(Z) = H$

$$ZL = L^{-T} D^{-1} \implies ZLe_j = L^{-T} D^{-1} e_j \tag{16}$$

Let $J_j = I_j \cup j$, then select J_j indices of vectors on both sides of the second equality.

$$\begin{bmatrix} Z_{jj} & Z_{jI_j} \\ Z_{I_j j} & Z_{J_j J_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{I_j} \end{bmatrix} = \begin{bmatrix} 1/d_{jj} \\ 0 \end{bmatrix} \tag{17}$$

Note that L^{-T} is an upper triangular matrix with ones in the diagonal hence J_j^{th} block of $L^{-T} e_j$ will be $[1, 0, 0, \ldots]$. Also, since $P_{\mathcal{G}}(Z) = H$

$$\begin{bmatrix} Z_{jj} & Z_{jI_j} \\ Z_{I_j j} & Z_{J_j J_j} \end{bmatrix} = \begin{bmatrix} H_{jj} & H_{jI_j} \\ H_{I_j j} & H_{J_j J_j} \end{bmatrix} \tag{18}$$

Substituting this in the linear equation 17

$$\begin{bmatrix} H_{jj} & H_{jI_j} \\ H_{I_jj} & H_{J_jJ_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{I_j} \end{bmatrix} = \begin{bmatrix} 1/d_{jj} \\ 0 \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} H_{jj} & H_{jI_j} \\ H_{I_jj} & H_{J_jJ_j} \end{bmatrix} \begin{bmatrix} d_{jj} \\ d_{jj} \cdot L_{I_j} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{20}$$

$$H_{jj}d_{jj} + d_{jj}H_{I_jj}^T L_{I_jj} = 1 \tag{21}$$

$$H_{I_jj}d_{jj} + d_{jj}H_{I_jI_j}L_{I_jj} = 0 \tag{22}$$

The lemma follows from solving the above equations. Note that here we used that lower triangular halves of matrices L and H have the same sparsity patterns, which follows from the fact that banded graph is chordal and has a perfect elimination order $[1, 2, \ldots, n]$.

□

A.2 Regret bound analysis

Proof of Theorem 2.

To upperbound the regret we prove the following lemma about regret decompsition

Lemma 2 (Hazan et al. (2016))**.** *In the OCO problem setup, if a prediction $w_t \in \mathbb{R}^n$ is made at round t and is updated as $w_{t+1} \coloneqq w_t - \eta X_t g_t$ using a preconditioner matrix $X_t \in S_n^{++}$*

$$R_T \leq \frac{1}{2\eta} \cdot (\|w_1 - w^*\|^2_{X_1^{-1}} - \|w_{T+1} - w^*\|_{X_T^{-1}}) \tag{23}$$

$$+ \frac{1}{2\eta} \cdot \sum_{t=1}^{T-1} \|w_{t+1} - w^*\|^2_{X_{t+1}^{-1} - X_t^{-1}} \tag{24}$$

$$+ \sum_{t=1}^{T} \frac{\eta}{2} \cdot g_t^T X_t g_t \tag{25}$$

Proof.

$$\begin{aligned} \|w_{t+1} - w^*\|^2_{X_t^{-1}} &= \|w_t - \eta X_t g_t - w^*\|^2_{X_t^{-1}} \\ &= \|w_t - w^*\|^2_{X_t^{-1}} + \eta^2 g_t^T X_t g_t - 2\eta (w_t - w^*)^T g_t \\ \implies 2\eta (w_t - w^*)^T g_t &= \|w_t - w^*\|^2_{X_t^{-1}} - \|w_{t+1} - w^*\|^2_{X_t^{-1}} + \eta^2 g_t^T X_t g_t \end{aligned}$$

□

Using the convexity of f_t, $f_t(w_t) - f_t(w^*) \leq (w_t - w^*)^T g_t$, where $g_t = \Delta f_t(w_t)$ and summing over $t \in [T]$

$$R_T \leq \sum_{t=1}^{T} \frac{1}{2\eta} \cdot \left(\|w_t - w^*\|^2_{X_t^{-1}} - \|w_{t+1} - w^*\|^2_{X_t^{-1}} \right) + \frac{\eta}{2} \cdot g_t^T X_t g_t \tag{26}$$

The first summation can be decomposed as follows

$$\sum_{t=1}^{T} \left(\|w_t - w^*\|^2_{X_t^{-1}} - \|w_{t+1} - w^*\|^2_{X_t^{-1}} \right) = \left(\|w_1 - w^*\|^2_{X_1^{-1}} - \|w_{T+1} - w^*\|^2_{X_T^{-1}} \right) + \sum_{t=1}^{T-1} \|w_{t+1} - w^*\|^2_{X_{t+1}^{-1} - X_t^{-1}}$$

Substituting the above identity in the Equation (26) proves the lemma.

Let $R_T \leq T_1 + T_2 + T_3$, where

- $T_1 = \frac{1}{2\eta} \cdot (\|w_1 - w^*\|^2_{X_1^{-1}} - \|w_{T+1} - w^*\|_{X_T^{-1}})$
- $T_2 = \frac{1}{2\eta} \cdot \sum_{t=1}^{T-1} \|w_{t+1} - w^*\|^2_{X_{t+1}^{-1} - X_t^{-1}}$
- $T_3 = \sum_{t=1}^{T} \frac{\eta}{2} \cdot g_t^T X_t g_t$

The following lemmas will be used to bound T_1, T_2, T_3.

Lemma 3. *If $\mathcal{G}$ = chain/tridiagonal graph and $\hat{X} = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}(X, H^{-1})$, then the inverse $\hat{X}^{-1}$ takes the following expression*

$$(\hat{X}^{-1})_{i,j} = \begin{cases} H_{i,j} & |i-j| \leq 1 \\ \frac{H_{i,i+1}H_{i+1,i+2}\ldots H_{j-1,j}}{H_{i+1,i+1}\ldots H_{j-1,j-1}} & \end{cases} \tag{27}$$

Proof.

$$\hat{X}^{-1}\hat{X}^{(j)} = e_j$$

Where $\hat{X}^{(j)}$ is the jth column of $\hat{X}$. Let $\hat{Y}$ denote the right hand side of Equation (27).

$$\begin{aligned}(\hat{Y}\hat{X})_{jj} &= \hat{X}_{jj}\hat{Y}_{jj} + \hat{X}_{j-1j}\hat{Y}_{j-1j} + \hat{X}_{jj+1}\hat{Y}_{jj+1} \\ &= \hat{X}_{jj}H_{jj} + \hat{X}_{j-1j}H_{j-1j} + \hat{X}_{jj+1}H_{jj+1} \\ &= 1\end{aligned}$$

The third equality is by using Equation (12).

Similarly, the offdiagonals of $\hat{Y}\hat{X}$ can be evaluated to be zero as follows.

$$(\hat{Y}\hat{X})_{ij} = \hat{Y}_{ij}\hat{X_j}j + \hat{Y}_{ij-1}\hat{X}_{j-1j} + \hat{Y}_{ij+1}\hat{X}_{j+1j} \tag{28}$$

$$= \hat{Y}_{ij}\hat{X}_{jj} + \hat{Y}_{ij}\frac{H_{j-1j-1}}{H_{j-1j}} + \hat{Y}_{ij}\frac{H_{jj+1}}{H_{jj}}\hat{X}_{j+1j} \tag{29}$$

$$= 0 \tag{30}$$

□

Lemma 4. *Let $y \in \mathbb{R}^n$, $\beta = \max_t \max_{i\in[n-1]} (H_t)_{ii+1}/\sqrt{(H_t)_{ii}(H_t)_{i+1i+1}} < 1$, $C = \sum_{i=1}^n y_i^2 (G_\infty^{(i)})^2$, where $g_{1:T}^{(i)} = [(g_1)_i, \ldots (g_T)_i]$ and $G_\infty^{(i)} = \|g_{1:T}^{(i)}\|_\infty$, then*

$$y^T X_t^{-1} y \leq (Ct + \epsilon \|y\|_2^2) \left(\frac{1+\beta}{1-\beta}\right) \tag{31}$$

Proof. Let $\tilde{X}_t^{-1} = \mathrm{diag}(H_t)^{-1/2}\hat{X}_t\,\mathrm{diag}(H_t)^{-1/2}$

$$y^T X_t^{-1} y \leq \left\|\mathrm{diag}(H_t)^{1/2} y\right\|_2^2 \left\|\tilde{X}_t^{-1}\right\|_2 \tag{32}$$

Using the identity of spectral radius $\rho(X) \leq \|X\|_\infty$ and since $\tilde{X}$ is positive definite, $\left\|\tilde{X}_t^{-1}\right\|_2 \leq \|\tilde{X}_t^{-1}\|_\infty$

$$\left\|\tilde{X}_t^{-1}\right\|_2 \leq \max_i \left\{\sum_j \left|(\tilde{X}_t^{-1})_{i,j}\right|\right\} \tag{33}$$

$$\leq 1 + 2(\beta + \beta^2 + \ldots) \tag{34}$$

$$\leq \frac{1+\beta}{1-\beta} \tag{35}$$

The second inequality is using Lemma 3. Using $(H_t)_{i,i} = \left\|g_{1:T}^{(i)}\right\|_2^2 + \epsilon$ in Equation (32) will give the lemma. □

Lemma 5 (Upperbound of T_1)**.** $T_1 \leq \frac{(C+\epsilon D_2^2)}{2\eta} \cdot \frac{1+\beta}{1-\beta}$, *where* $D_2 = \max_{t\in[T]} \|w_t - w^*\|_2$, *where* $C = \max_t(\sum_{i=1}^n (y_i^{(t)})^2 (G_\infty^{(i)})^2)$.

Proof. Since X_T is positive definite

$$T_1 \leq \frac{\|w_1 - w^*\|^2_{X_1^{-1}}}{2\eta} \tag{36}$$

$$\leq \frac{(y^{(1)})^T X_1^{-1} y^{(1)}}{2\eta} \tag{37}$$

Using Lemma 4 proves the lemma. □

Upperbounding T_2
Let $y_t = w_t - w^*$, $P_t = y_t^T X_t^{-1} y_t$, and $Q_t = y_t^T X_{t-1}^{-1} y_t$. Let $P = [P_2, \cdots, P_T]$ and $Q = [Q_2, \cdots, Q_T]$ be the two array representations. Then,

$$2\eta T_2 = \sum_{i=1}^{T-1} y_{t+1}^T \left(X_{t+1}^{-1} - X_t^{-1}\right) y_{t+1} \tag{38}$$

$$= \sum_{i=2}^{T} y_t^T \left(X_t^{-1} - X_{t-1}^{-1}\right) y_t \tag{39}$$

$$= \sum_{i=2}^{T} y_t^T X_t^{-1} y_t - y_t^T X_{t-1}^{-1} y_t \tag{40}$$

$$= \sum_{i=2}^{T} P_t - Q_t \tag{41}$$

$$= \|P - Q\|_1 \tag{42}$$

In order to upper bound this, we derive here a generalized version of Pinsker's inequality for our setting.

Lemma 6. *Let* $A = [A_1, \cdots, A_T]$ *and* $B = [B_1, \cdots, B_T]$ *be two* T*-length arrays. Let* $B_i, A_i > 0 \; \forall i \in [1, T+1]$. *Then,*

$$\|A - B\|_1^2 \leq \frac{2}{3}\left(\sum_{i=1}^{T}(2A_i + B_i)\right) D_{KL}(B, A) \tag{43}$$

where the generalized KL-Divergence $D_{KL}(B, A) = \sum_{i=1}^T \left(B_i log\left(\frac{B_i}{A_i}\right) - B_i + A_i\right)$.

Proof. This proof is a mock up of Pinsker's inequality for probability spaces. First step involves using the following identity, for $t \geq -1$

$$(1+t)log(1+t) - t \geq \frac{1}{2} \cdot \frac{t^2}{1 + \frac{t}{3}} \tag{44}$$

Let $t_i = \frac{B_i}{A_i} - 1$. Then, $-1 \leq t_i$. Therefore,

$$
\begin{aligned}
D_{KL}(B, A) &= \sum_{i=1}^{T} \left(B_i \log\left(\frac{B_i}{A_i}\right) - B_i + A_i \right) \\
&= \sum_{i=1}^{T} A_i \left(\frac{B_i}{A_i} \log\left(\frac{B_i}{A_i}\right) - \frac{B_i}{A_i} + 1 \right) \\
&= \sum_{i=1}^{T} A_i \left((t_i + 1)\log(t_i + 1) - t_i \right) \\
&\geq \sum_{i=1}^{T} \frac{A_i t_i^2}{2\left(1 + \frac{t_i}{3}\right)}
\end{aligned}
$$

$$
\begin{aligned}
D_{KL}(B, A) &= \sum_{i=1}^{T} \frac{A_i t_i^2}{2\left(1 + \frac{t_i}{3}\right)} \cdot \frac{\sum_{j=1}^{T} A_j (1 + \frac{t_j}{3})}{\sum_{k=1}^{T} A_k (1 + \frac{t_k}{3})} \\
&\geq \frac{\left(\sum_{i=1}^{T} A_i |t_i|\right)^2}{2\sum_{k=1}^{T} A_k (1 + \frac{t_k}{3})} && \text{(Using Cauchy-Schwartz)} \\
&= \frac{\left(\sum_{i=1}^{T} |A_i - B_i|\right)^2}{\frac{2}{3}\sum_{k=1}^{T} (2A_k + B_k)} \\
&= \frac{\|A - B\|_1^2}{\frac{2}{3}\sum_{j=1}^{T} (2A_i + B_i)}
\end{aligned}
$$

□

Using the above to bound Equation (42) with $B := P$ and $A := Q$, we get

$$
\begin{aligned}
2\eta T_2 = \|P - Q\|_1 &\leq \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) D_{KL}(P, Q)} && (45) \\
&= \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \sum_{t=2}^{T} D_{KL}(P_i, Q_i)} && (46) \\
&= \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \sum_{t=2}^{T} D_{KL}(y_t^T X_t^{-1} y_t, y_t^T X_{t-1}^{-1} y_t)} && (47) \\
&= \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \sum_{t=2}^{T} D_{KL}(\bar{y}_t{}^T \bar{X}_t^{-1} \bar{y}_t, \bar{y}_t{}^T \bar{X}_{t-1}^{-1} \bar{y}_t)} && (48) \\
&\leq \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \cdot \sum_{t=2}^{T} \|\bar{y}_t\|_2^2 \cdot D_{vN}(\bar{X}_t^{-1}, \bar{X}_{t-1}^{-1})} && (49)
\end{aligned}
$$

Here, $\bar{y}_t = \Lambda_t y_t$, $\bar{X}_t = \Lambda_t X_t \Lambda_t$, $\bar{X}_{t-1} = \Lambda_t X_{t-1} \Lambda_t$ D_{vN} is von-Neuman Divergence, and last inequality is from Lindblad (1975). We upper bound Equation (49) using the following lemma.

Lemma 7. *Let A, B be two PD matrices. Then,*

$$
D_{vN}(A, B) \leq \lambda_{max}(A) \cdot \mathrm{D}_{\ell\mathrm{d}}(B, A) \tag{50}
$$

Proof. Let $A = V\Lambda V^T$ and $B = U\Theta U^T$

$$\begin{aligned} D_{vN}(A,B) &= \sum_{i,j}(v_i^T u_j)\left(\lambda_i \log\left(\frac{\lambda_i}{\theta_j}\right) - \lambda_i + \theta_j\right) \\ &= \sum_{i,j}(v_i^T u_j)\lambda_i\left(\log\left(\frac{\lambda_i}{\theta_j}\right) - 1 + \frac{\theta_j}{\lambda_i}\right) \\ &\leq \max_i \lambda_i \sum_{i,j}(v_i^T u_j)\left(-\log\left(\frac{\theta_j}{\lambda_i}\right) - 1 + \frac{\theta_j}{\lambda_i}\right) \\ &= \lambda_{max}(A)\cdot \mathrm{D}_{\ell\mathrm{d}}(B,A) \end{aligned}$$

□

Combining the above result, we get

$$\begin{aligned} T_2 &= \frac{1}{2\eta}\cdot\|P-Q\|_1 \\ &\leq \frac{1}{2\eta}\cdot\sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t+P_t)\right)\cdot\sum_{t=2}^{T}\|\bar{y}_t\|_2^2\left\|\bar{X}_t^{-1}\right\|_2\cdot \mathrm{D}_{\ell\mathrm{d}}(\bar{X}_{t-1}^{-1},\bar{X}_t^{-1})} \\ &\leq \frac{1}{2\eta}\cdot\sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t+P_t)\right)\cdot\sum_{t=2}^{T}\|\bar{y}_t\|_2^2\left\|\bar{X}_t^{-1}\right\|_2\cdot \mathrm{D}_{\ell\mathrm{d}}(\bar{X}_{t-1}^{-1},\bar{X}_t^{-1})} \end{aligned}$$

The second inequality is due to scale invariance of LogDet divergence. If we set $\Lambda_t = \mathrm{diag}(H_t)^{1/2}$, then using Lemma 4

$$T_2 \leq \frac{1}{2\eta}\cdot\sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t+P_t)\right)\cdot\sum_{t=2}^{T}(Ct+\epsilon\|y_t\|_2^2)\left(\frac{1+\beta}{1-\beta}\right)\cdot \mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1},X_t^{-1})} \tag{51}$$

$$\leq \frac{1}{2\eta}\cdot\sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t+P_t)\right)\cdot(CT+\epsilon D_2^2)\left(\frac{1+\beta}{1-\beta}\right)\cdot\sum_{t=2}^{T}\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1},X_t^{-1})} \tag{52}$$

$$\leq \frac{1}{2\eta}\cdot(CT+\epsilon D_2^2)\left(\frac{1+\beta}{1-\beta}\right)\cdot\sqrt{\frac{2}{3}\cdot T\cdot\sum_{t=2}^{T}\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1},X_t^{-1})} \tag{53}$$

where, $D_2 = \max_t \|y_t\|_2$, the second and third inequality is using Lemma 4. Now to bound the $\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1},X_t^{-1})$ term in the above equation, we develop the following lemma.

Lemma 8. $\mathrm{Tr}(X_tH_t) = n$, $\forall t \in \{2,\ldots,T\}$ *in Algorithm 1 with any* $\mathcal{G}$

Proof. $P_{\mathcal{G}}(X_t^{-1}) = H_t$ is an optimality condition of LogDet subproblem Equation (10)

$$\mathrm{Tr}(X_tH_t) = \mathrm{Tr}(X_tP_{\mathcal{G}}(X_t^{-1})) \tag{54}$$

$$= \mathrm{Tr}(X_tX_t^{-1}) \tag{55}$$

$$= n \tag{56}$$

The second equality is because X_t follows the sparsity graph $\mathcal{G}$. □

Lemma 9.

$$\sum_{t=2}^{T}\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1},X_t^{-1}) \leq \log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_1^{-1}\right)}\right) \tag{57}$$

$$\leq \sum_{i=1}^{n}\log\left(1+\frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1}\log\left(1-\beta_i^2\right) \tag{58}$$

where $\beta_i = \frac{\left\langle g_{1:T}^{(i)}, g_{1:T}^{(i+1)} \right\rangle}{\sqrt{(\epsilon + \left\| g_{1:T}^{(i)} \right\|_2^2) \cdot (\epsilon + \left\| g_{1:T}^{(i+1)} \right\|_2^2)}}$

Proof. LogDet divergence is defined as follows

$$\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1}) = -\log\left(\frac{\det\left(X_{t-1}^{-1}\right)}{\det\left(X_t^{-1}\right)}\right) + \mathrm{Tr}(X_{t-1}^{-1}X_t) - n \tag{59}$$

The second two terms on right hand side can be simplified analyzed as follows.

$$\begin{aligned}
\mathrm{Tr}(X_{t-1}^{-1}X_t) - n &= \mathrm{Tr}(X_{t-1}^{-1}X_t - X_t^{-1}X_t) && (60)\\
&= \mathrm{Tr}((X_{t-1}^{-1} - X_t^{-1})X_t) && (61)\\
&= -g_t^T X_t g_t && (62)\\
&< 0 && (63)
\end{aligned}$$

Thus

$$\begin{aligned}
\sum_{t=2}^{T} \mathrm{D}_{\ell\mathrm{d}}\left(X_{t-1}^{-1}, X_t^{-1}\right) &\leq \sum_{t=2}^{T} -\log\left(\frac{\det\left(X_{t-1}^{-1}\right)}{\det\left(X_t^{-1}\right)}\right) && (64)\\
&\leq \log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_1^{-1}\right)}\right) && (65)
\end{aligned}$$

Since LogDet divergence is positive

$$\begin{aligned}
\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1}) &= -\log\left(\frac{\det\left(X_{t-1}^{-1}\right)}{\det\left(X_t^{-1}\right)}\right) + \mathrm{Tr}(X_{t-1}^{-1}X_t) - n && (66)\\
&\geq 0 && (67)\\
\implies \log\left(\frac{\det\left(X_t^{-1}\right)}{\det\left(X_{t-1}^{-1}\right)}\right) &\geq 0 && (68)
\end{aligned}$$

The second inequality uses Equation (63). Now, using Equation (14), which is in cholesky decomposition format,

$$\begin{aligned}
\log(\det\left(X_T^{-1}\right)) &= \sum_{i=1}^{n-1} \log\left((H_T)_{ii} - (H_T)_{ii+1}^2/(H_T)_{i+1i+1}\right) + \log((H_T)_{nn}) && (69)\\
&\leq \sum_{i=1}^{n-1} \log(1-\beta_i^2) + \sum_{i=1}^{n} \log((H_T)_{ii}) && (70)
\end{aligned}$$

Using the above, we can expand the log deteriminatn difference.

$$\begin{aligned}
\log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_1^{-1}\right)}\right) &\leq \log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_0^{-1}\right)}\right) && (71)\\
&\leq \sum_{i=1}^{n} \log\left(1 + \frac{\left\| g_{1:T}^{(i)} \right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1-\beta_i^2\right) && (72)
\end{aligned}$$

Let $X_0 = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}(X, H_0^{-1})$, the above inequality is since $\log\left(\frac{\det\left(X_1^{-1}\right)}{\det\left(X_0^{-1}\right)}\right) \geq 0$ using Equation (68) □

Lemma 10 (Upperbound of T_2)**.**

$$T_2 \leq \frac{1}{2\eta} \cdot (CT + \epsilon D_2^2)\left(\frac{1+\beta}{1-\beta}\right) \cdot \sqrt{\frac{2}{3} \cdot T \cdot \left(\sum_{i=1}^{n} \log\left(1 + \frac{\left\| g_{1:T}^{(i)} \right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1-\beta_i^2\right)\right)} \tag{73}$$

Proof. The proof directly follows from Equation (53) and Lemma 9 □

Lemma 11 (Upperbound of T_3)**.** *Let* $X_0 = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}(X, H_0^{-1})$

$$T_3 = \sum_{t=1}^{T} \frac{\eta}{2} \cdot g_t^T X_t g_t \tag{74}$$

$$\leq \frac{\eta}{2} \cdot \log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_0^{-1}\right)}\right) \tag{75}$$

$$\leq \frac{\eta}{2} \cdot \sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right) \tag{76}$$

where $\beta_i = \frac{((g_{1:T}^{(i)})^T g_{1:T}^{(i+1)})^2}{(\epsilon + \left\|g_{1:T}^{(i)}\right\|_2^2) \cdot (\epsilon + \left\|g_{1:T}^{(i+1)}\right\|_2^2)}$

Proof.

$$g_t^T X_t g_t = \mathrm{Tr}(X_t g_t g_t^T) \tag{77}$$

$$= \mathrm{Tr}(X_t(X_t^{-1} - X_{t-1}^{-1})) \tag{78}$$

$$= n - \mathrm{Tr}(X_t X_{t-1}^{-1}) \tag{79}$$

$$\leq -\log\left(\frac{\det\left(X_{t-1}^{-1}\right)}{\det\left(X_t^{-1}\right)}\right) \tag{80}$$

The second equality is using Lemma 3 and that X_t's sparsity graph is tridiagonal. The first inequality is using the property $\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1}) \geq 0$ of LogDet divergence. Thus summing up and using Equation (72) will give the lemma. □

Putting together T_1, T_2 and T_3 from Lemma 5, Lemma 10 and Lemma 11

$T_1 \leq \frac{(C + \epsilon D_2^2)}{2\eta} \cdot \frac{1+\beta}{1-\beta}$,

$T_2 \leq \frac{1}{2\eta} \cdot (CT + \epsilon D_2^2)\left(\frac{1+\beta}{1-\beta}\right) \cdot \sqrt{\frac{2}{3} \cdot T \cdot \left(\sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right)\right)}$,

$T_3 \leq \frac{\eta}{2} \cdot \sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right)$

Theorem 6 (Regret upperbound)**.** *If we set* $\eta = \frac{C^{1/2} T^{3/4} \cdot (\frac{1+\beta}{1-\beta})^{1/2}}{\left(\sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right)\right)^{1/4}}$*, then*

$$R_T \leq T_1 + T_2 + T_3 \tag{81}$$

$$\leq \mathcal{O}\left(C^{1/2} \cdot T^{3/4} \cdot \left(\frac{1+\beta}{1-\beta}\right)^{1/2} \cdot \left(\sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right)\right)^{3/4}\right) \tag{82}$$

A.3 Numerical stability

Theorem 7 (*Condition number of tridiagonal LogDet subproblem*)**.** *Let* $H \in S_n^{++}$ *such that* $H_{i,i} = 1$*, for* $i \in [n]$ *and a symmetric perturbation* ΔH *such that* $\Delta H_{i,i} = 0$*, for* $i \in [n]$ *and* $H + \Delta H \succ 0$*. Let* $\hat{X} = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}\left(X, H^{-1}\right)$ *and* $\hat{X} + \Delta\hat{X} =$

$\arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}\left(X, (H+\Delta H)^{-1}\right)$, *here* $\mathcal{G} \coloneqq$ *chain/tridiagonal sparsity graph and* $S_n(\mathcal{G})^{++}$ *denotes positive definite matrices which follows the sparsity pattern* $\mathcal{G}$.

$$\kappa_{\ell d} = \max_{|i-j|\leq 1} \limsup_{\epsilon \to 0} \left\{ \frac{\left|\Delta \hat{X}_{i,j}\right|}{\epsilon \left|\hat{X}_{i,j}\right|} : |\Delta H_{k,l}| \leq |\epsilon H_{k,l}|, (k,l) \in E_{\mathcal{G}} \right\} \tag{83}$$

$$\leq \max_{i \in [n-1]} 1/(1-\beta_i^2) \tag{84}$$

where, $\kappa_{\ell d} \coloneqq$ *condition number of the LogDet subproblem,* $\kappa_2(.) \coloneqq$ *condition number of a matrix in* ℓ_2 *norm,* $\beta_i = H_{i,i+1}/\sqrt{H_{i,i}H_{i+1,i+1}}$

Proof. Consider the offdiagonals for which $(\hat{X} + \Delta\hat{X})_{ii+1} = -H_{ii+1}/(1-H_{i,i+1}^2) = f(H_{i,i+1})$,where $f(x) = -x/(1-x^2)$. Let $y = f(x)$, $\hat{y} = f(x+\Delta x)$ and $|\Delta x/x| \leq \epsilon$ then using Taylor series

$$\left|\frac{(\hat{y}-y)}{y}\right| = \left|\frac{xf'(x)}{f(x)}\right| \left|\frac{\Delta x}{x}\right| + O((\Delta x)^2) \tag{85}$$

$$\implies \lim_{\epsilon \to 0} \left|\frac{(\hat{y}-y)}{\epsilon y}\right| \leq \frac{xf'(x)}{f(x)} \tag{86}$$

Using the above inequality, with $x \coloneqq H_{i,i+1}$ and $y \coloneqq \hat{X}_{i,i+1}$,

$$\lim_{\epsilon \to 0} \left|\frac{\Delta \hat{X}_{i,i+1}}{\epsilon \hat{X}_{i,i+1}}\right| \leq \frac{1+H_{i,i+1}^2}{1-H_{i,i+1}^2} \tag{87}$$

$$\leq \frac{2}{1-H_{i,i+1}^2} \tag{88}$$

Let $g(x) = x^2/(1-x^2)$, let $y_1 = g(w_1)$, $y_2 = g(x_2)$, $\hat{y}_1 = g(w_1+\Delta x)$, $\hat{y}_2 = g(x_2+\Delta x)$. Using Taylor series

$$\left|\frac{(\hat{y}_1-y_1)}{y_1}\right| = \left|\frac{x_1 f'(x_1)}{f(x_1)}\right| \left|\frac{\Delta x_1}{x_1}\right| + O((\Delta x_1)^2) \tag{89}$$

$$\left|\frac{(\hat{y}_2-y_2)}{y_2}\right| = \left|\frac{x_2 f'(x_2)}{f(x_2)}\right| \left|\frac{\Delta x_2}{x_2}\right| + O((\Delta x_2)^2) \tag{90}$$

$$\implies \lim_{\epsilon \to 0} \frac{\Delta y_1 + \Delta y_2}{\epsilon(1+y_1+y_2)} \leq \max\left(\frac{2}{1-x_1^2}, \frac{2}{1-x_2^2}\right) \tag{91}$$

Putting $x_1 \coloneqq H_{i,i+1}$, $x_2 \coloneqq H_{i,i-1}$ and analyzing $y_1 \coloneqq H_{i,i+1}^2/(1-H_{i,i+1}^2)$ and $y_2 \coloneqq H_{i,i-1}^2/(1-H_{i,i-1}^2)$ will result in the following

$$\lim_{\epsilon \to 0} \left|\frac{\Delta \hat{X}_{i,i}}{\hat{X}_{i,i}}\right| \leq \max\left(\frac{2}{1-H_{i,i+1}^2}, \frac{2}{1-H_{i,i-1}^2}\right) \tag{92}$$

Since $\hat{X}_{i,i} = 1 + H_{i,i+1}^2/(1-H_{i,i+1}^2) + H_{i,i-1}^2/(1-H_{i,i-1}^2)$. Putting together Equation (92) and Equation (87), the theorem is proved. □

Lemma 12 (Degenerate inputs to banded LogDet subproblem)**.** *Let* $H = GG^T$*, where* $G \in \mathbb{R}^{n \times T}$ *and* $g_{1:T}^{(i)} \coloneqq$ *corresponds to a row in* G*, equivalent to history of gradients of parameter* i*, then* $H_{i,j} = \left\langle g_{1:T}^{(i)}, g_{1:T}^{(j)} \right\rangle$.

- *For* $b = 1$ *in Equation* (14)*,i.e., tridiagonal sparsity pattern is used: if* $g_{1:T}^{(j)} = g_{1:T}^{(j+1)}$*, then* $H_{jj} - H_{j,j+1}^2/H_{j+1,j+1} = 0$ *and* D_{jj} *is undefined*
- *For* $b > 1$*: If* $\mathrm{rank}(H_{J_j,J_j}) = \mathrm{rank}(H_{I_j,I_j}) = b$*, then* $(H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j}) = 0$ *and* D_{jj} *is undefined. If* $\mathrm{rank}(H_{I_j,I_j}) < b$ *then* $(H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j})$ *is undefined.*

Proof. For $b = 1$, if $g_{1:T}^{(j)} = g_{1:T}^{(j+1)}$, then $H_{j,j+1} = H_{j,j} = H_{j+1,j+1} = \left\|g_{1:T}^{(j)}\right\|_2^2$, thus $H_{jj} - H_{j,j+1}^2/H_{j+1,j+1} = 0$.
For $b > 1$, since H_{I_j,I_j}, using Guttman rank additivity formula, $\text{rank}(H_{jj} - H_{j,j+1}^2/H_{j+1,j+1}) = \text{rank}(H_{J_jJ_j}) - rank(H_{I_jI_j}) = 0$, thus $H_{jj} - H_{j,j+1}^2/H_{j+1,j+1} = 0$.
Furthermore, if $\text{rank}(H) \leq b$, then all $b+1 \times b+1$ principal submatrices of H have rank b, thus $\forall j$, $H_{J_jJ_j}$ have a rank b, thus D_{jj} for all j are undefined.

□

Algorithm 4 Numerically stable banded LogDet solution

1: Input: $\mathcal{G}-$ tridiagonal or banded graph ,$H-$ PSD matrix with $H_{ii} > 0$, $\gamma-$ threshold
2: Output: Finds subgraph $\tilde{\mathcal{G}}$ of $\mathcal{G}$ without any degenerate cases from Lemma 1 and finds preconditioner $\hat{X}$ corresponding to the subgraph
3: Let $E_i = \{(i,j) : (i,j) \in E_{\mathcal{G}}\}$ be edges from vertex i to its neighbours in graph $\mathcal{G}$.
4: Let $V_i^+ = \{j : i < j, (i,j) \in E_{\mathcal{G}}\}$ and $V_i^- = \{j : i > j, (i,j) \in E_{\mathcal{G}}\}$, denote positive and negative neighbourhood of vertex i.
5: Let $K = \left\{i : H_{ii} - H_{I_ii}^T H_{I_iI_i}^{-1} H_{I_ii} \text{ is undefined or } \leq \gamma, i \in [n]\right\}$
6: Consider a new subgraph $\tilde{\mathcal{G}}$ with edges $E_{\tilde{\mathcal{G}}} = E_{\mathcal{G}} \setminus ((\bigcup_{i \in K} E_i) \cup (V_i^+ \times V_i^-))$
7: return $\hat{X} = \arg\min_{X \in S_n(\tilde{\mathcal{G}})^{++}} \mathrm{D}_{\ell\mathrm{d}}\left(X, H_t^{-1}\right)$ solved via Equation (14)

Theorem 8 (Numerically stable algorithm)**.** *Algorithm 3 (repeated above as Algorithm 4) finds a subgraph $\tilde{\mathcal{G}}$ of $\mathcal{G}$, such that explicit solution for $\tilde{\mathcal{G}}$ in Equation* (14) *is well-defined. Furthermore, when $\mathcal{G}$ is a tridiagonal/chain graph, the component-wise condition number in Equation* (13) *is reduced upon using Algorithm 3,*

$$\kappa_{\ell d}^{\tilde{\mathcal{G}}} < \kappa_{\ell d}^{\mathcal{G}} \tag{93}$$

$\kappa_{\ell d}^{\tilde{\mathcal{G}}}$, $\kappa_{\ell d}^{\mathcal{G}}$ are the condition numbers defined for the LogDet problem with graphs $\tilde{\mathcal{G}}$ and $\mathcal{G}$ respectively.

Proof. Let $I_i = \{j : i < j, (i,j) \in E_{\mathcal{G}}\}$ and $I_i' = \left\{j : i < j, (i,j) \in E_{\tilde{\mathcal{G}}}\right\}$ Let $K = \left\{i : H_{ii} - H_{I_ii}^T H_{I_iI_i}^{-1} H_{I_ii} \text{ is undefined or } 0, i \in [n]\right\}$ denote vertices which are getting removed by the algorithm, then for the new graph $\tilde{\mathcal{G}}$, $D_{ii} = 1/H_{i,i}, \forall i \in K$ since $H_{i,i} > 0$.
Let $\bar{K} = \left\{i : H_{ii} - H_{I_ii}^T H_{I_iI_i}^{-1} H_{I_ii} > 0, i \in [n]\right\}$. Let for some $j \in \bar{K}$, if

$$l = \arg\min \{i : j < i, i \in K \cap I_j\}, \tag{94}$$

denotes the nearest connected vertex higher than j for which D_{ll} is undefined or zero, then according to the definition $E_{\tilde{\mathcal{G}}}$ in Algorithm 3, $I_j' = \{j+1, \ldots l-1\} \subset I_j$, since D_{jj} is well-defined, $H_{I_jI_j}$ is invertible, which makes it a positive definite matrix (since H is PSD). Since $H_{jj} - H_{I_jj}^T H_{I_jI_j}^{-1} H_{I_jj} > 0$, using Guttman rank additivity formula $H_{J_jJ_j} \succ 0$, where $J_j = I_j \cup j$. Since $H_{J_j'J_j'}$ is a submatrix of $H_{J_jJ_j}$, it is positive definite and hence its schur complement $H_{jj} - H_{I_j'j}^T H_{I_j'I_j'}^{-1} H_{I_j'j} > 0$. Thus for all $j \in [n]$, the corresponding D_{jj}'s are well-defined in the new graph $\tilde{\mathcal{G}}$.

Note that $\kappa_{\ell d}^{\tilde{\mathcal{G}}} = \max_{i \in [n-1]} 1/(1-\beta_i^2) < \max_{i \in \bar{K}} 1/(1-\beta_i^2) = \kappa_{\ell d}^{\mathcal{G}}$, for tridiagonal graph, where $\beta_i = H_{i,i+1}$, in the case where $H_{ii} = 1$. This is because the $\arg\max_{i \in [n-1]} 1/(1-\beta_i^2) \in K$.

□

A.4 Experiment ablations and details

Affect of band size in banded-SONew Increasing band size in banded-SONew captures more correlation between parameters, hence should expectedly lead to better preconditioners. We confirm this through experiments on the Autoencoder benchmark where we take band size = 0 (diag-SONew), 1 (tridiag-SONew), 4, and 10.

Table 3: **float32 experiments on Autoencoder benchmark using different band sizes.** Band size 0 corresponds to diag-SONew and 1 corresponds to tridiag-SONew. We see the training loss getting better as we increase band size

Band size	0 (diag-SONew)	1 (tridiag-SONew)	4	10
Train CE loss	53.025	51.723	51.357	51.226

Table 4: **float32 and bfloat16 experiments on Autoencoder benchmark with and without Algorithm 3.** We observe gains in training loss when using Algorithm 3

Optimizer	**Training precision**	**Train CE loss - without Algorithm 3**	**Train CE loss - with Algorithm 3**
tridiag-SONew	bloat16	53.150	51.936
band-4-SONew	float32	51.534	51.357

Hyperparaeter search space We provide the hyperparamter search space for experiments presented in Section 6. We search over $2k$ hyperparameters for each experiment using a Bayesian Optimization package. The search ranges are: first order momentum term $\beta_1 \in [1e-1, 0.999]$, second order momentum term $\beta_2 \in [1e-1, 0.999]$, learning rate $\in [1e-7, 1e-1]$, $\epsilon \in [1e-10, 1e-1]$.

Affect of Numerical Stability Algorithm 3 On tridiag-SONew and banded-4-SONew, we observe that using Algorithm 3 improves training loss. We present in Table 4 results where we observed significant performance improvements.