Coordinate descent on the Stiefel manifold for deep neural network training

Estelle Massart[1] and Vinayak Abrol[2]

1- UCLouvain - ICTEAM
Avenue Georges Lemaître 4, B-1348 Louvain-la-Neuve - Belgium

2 - IIIT Delhi - CSE Department
Infosys Centre for AI, R&D Block, IIIT Delhi - India

Abstract. To alleviate the cost incurred by orthogonality constraints in optimization and model training, we propose a stochastic coordinate descent algorithm on the Stiefel manifold. We compute expressions for geodesics on the Stiefel manifold with initial velocity aligned with coordinates of the tangent space and show that, analogously to the orthogonal group, iterate updates of coordinate descent methods can be efficiently implemented in terms of multiplications by Givens matrices. We illustrate our proposed algorithm on deep neural network training.

1 Introduction

While orthogonality constraints have initially been proposed for recurrent neural networks [1], they have been extended to a wide range of other architectures, including fully-connected, convolutional, and residual neural networks [2, 3, 4, 5]. Imposing orthogonality constraints (either hard or soft, i.e., regularizers) is known to improve training stability and therefore reduce the risk of encountering exploding and vanishing gradients. Orthogonal initializations have been shown to lead for some architectures to dynamical isometry; a regime where the singular values of the input-output Jacobian concentrate around one, leading to faster training [2]; the benefits of orthogonal initialization on the speed of training deep linear neural networks have been proven in [6]. These benefits are partly explained in [7], where a connection is made between the maximum curvature of the optimization landscape as measured by the Fisher information matrix and the spectral radius of the input-output Jacobian. Finally, let us mention that orthogonality constraints were also used in generative adversarial networks [8, 9].

The biggest limitation to the use of orthogonality constraints for deep neural networks is the additional training cost due to the constraints. When all constrained matrices are square (and not rectangular), these costs can be alleviated by relying to coordinate descent on the orthogonal group, for which the cost of iterate update (due to the computation of geodesics/retractions) scales linearly with the size of the matrix [10, 11], instead of cubically for the usual Riemannian gradient descent algorithm. Our goal with this paper is to extend this algorithm to the Stiefel manifold, i.e., the set of rectangular matrices with orthonormal columns. The only work, to our knowledge, addressing coordinate descent on the Stiefel manifold in the case $p < n$ is [12], but the authors do not

propose practical expressions for the iterations nor implementation/numerics, their contribution is theoretical.

2 Geometry of the Stiefel manifold

We first introduce the Stiefel manifold, namely, the set of matrices with orthonormal columns $\mathrm{St}(p,n) := \{Y \in \mathbb{R}^{n\times p} : Y^\top Y = I_p,\}$, with I_p the $p \times p$ identity matrix. This set is known to be a D-dimensional Riemannian manifold, with $D := np - \frac{p(p+1)}{2}$. A manifold is a set that *locally looks flat*, i.e., that is equipped with a collection of *charts* allowing local mappings to open sets of the Euclidean space. Manifolds admit around any point a first-order approximation, called *tangent space*. In Riemannian manifolds, tangent spaces are endowed with a smoothly varying inner product, referred to as *Riemannian metric*. We define below several geometrical tools required by our algorithm: tangent spaces, Riemannian partial derivatives, and geodesics. This section relies on [13, 11].

The tangent space of the Stiefel manifold at $Y \in \mathrm{St}(p,n)$ is:

$$\mathcal{T}_Y\mathrm{St}(p,n) = \{Y\Omega + Y_\perp K : \Omega = -\Omega^\top, K \in \mathbb{R}^{(n-p)\times p}\}. \tag{1}$$

The Riemannian metric is chosen as the Euclidean inner product (i.e., $\mathrm{St}(p,n)$ is seen as an embedded submanifold of $\mathbb{R}^{n\times p}$), so that the norm of any tangent vector $\xi_Y \in \mathcal{T}_Y\mathrm{St}(p,n)$ is $\|\xi_Y\| = (\mathrm{tr}\left(\xi_Y^\top \xi_Y\right))^{\frac{1}{2}}$. Our approach requires us to select an orthonormal basis for $\mathcal{T}_Y\mathrm{St}(p,n)$. We will rely on the orthonormal basis

$$\mathcal{B} = \{Y\Omega_{i,j}, Y_\perp K_{u,v}\}_{\substack{i,v=1,\dots,p,\\ j=i+1,\dots,p\\ u=1,..,n-p}}, \tag{2}$$

where

$$\Omega_{i,j} = \frac{e_i e_j^\top - e_j e_i^\top}{\sqrt{2}} \qquad \text{and} \qquad K_{u,v} = e_u e_v^\top$$

are orthonormal bases of the set of skew-symmetric $p\times p$ matrices, and the set of $(n-p)\times p$ matrices, respectively, and where e_i refers to the ith canonical vector of suitable dimension. For an arbitrary smooth function $h : \mathrm{St}(p,n) \to \mathbb{R}$, the choice of basis (2) induces a notion of Riemannian partial derivative. The latter can be easily computed from the Euclidean gradient $\nabla^e h(Y)$[1]. The Riemannian partial derivative associated with a basis vector $\eta \in \mathcal{B}$ is

$$\mathrm{D}h(Y)[\eta] = \begin{cases} \mathrm{tr}\left(\Omega_{i,j}^\top Y^\top \nabla^e h(Y)\right) & \text{if } \eta = Y\Omega_{i,j} \text{ for some } i,j \\ \mathrm{tr}\left(K_{u,v}^\top Y_\perp^\top \nabla^e h(Y)\right) & \text{if } \eta = Y_\perp K_{u,v} \text{ for some } u,v. \end{cases} \tag{3}$$

Finally, the geodesic starting at $Y(0) \in \mathrm{St}(p,n)$ with initial velocity $\dot{Y}(0) \in$

[1] We rely here on an expression of the Riemannian partial derivatives in terms of the Euclidean gradient, as (a stochastic estimate of) the latter is typically available for DNN training, via backpropagation. We emphasize that, in general, computing the entire Euclidean gradient is not needed for evaluating the Riemannian partial derivatives.

$\mathcal{T}_{Y(0)}\mathrm{St}(p,n)$ is the curve $Y : t \mapsto Y(t)$ with

$$\begin{aligned} Y(t) &= \begin{pmatrix} Y(0) & \dot{Y}(0) \end{pmatrix} C I_{2p,p} \exp(-tY(0)^\top \dot{Y}(0)), \\ \text{with} \quad C &:= \exp t \begin{pmatrix} Y(0)^\top \dot{Y}(0) & -\dot{Y}(0)^\top \dot{Y}(0) \\ I_p & Y(0)^\top \dot{Y}(0) \end{pmatrix}. \end{aligned} \tag{4}$$

3 Geodesics along coordinate directions

Our proposed algorithm relies on expressions for geodesics with initial velocity aligned along an axis of the basis (2) of the tangent space. It is well-known that, in the case of square matrices (the orthogonal group), geodesics with initial velocity restricted to one coordinate axis of the tangent space to the orthogonal group can be very efficiently computed in terms of multiplication by a Givens matrix [10]. In this section, we extend the results of [10] to the Stiefel manifold.

Theorem 1 *The geodesic emanating from $Y(0)$ with initial velocity $\eta := Y(0)\Omega_{i,j}$ is*

$$Y(t) = Y(0)\exp(t\Omega_{i,j}) = Y(0)G_{i,j}(\sqrt{2}t) \tag{5}$$

where $G_{i,j}(t)$ is the Givens matrix

$$G_{i,j}(t) := \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cos(t) & \cdots & \sin(t) & \vdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin(t) & \cdots & \cos(t) & \vdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}. \tag{6}$$

In particular, evaluating it at some t requires $\mathcal{O}(n)$ floating point operations, compared to $\mathcal{O}(np^2)$ for arbitrary initial velocities.

Proof 1 *Equation* (4) *becomes*

$$Y(t) = \begin{pmatrix} Y(0) & Y(0)\Omega_{i,j} \end{pmatrix} \exp t \begin{pmatrix} \Omega_{i,j} & \Omega_{i,j}^2 \\ I & \Omega_{i,j} \end{pmatrix} I_{2p,p} \exp(-t\Omega_{i,j}). \tag{7}$$

Letting $P := \begin{pmatrix} \Omega_{i,j} & \Omega_{i,j}^2 \\ I & \Omega_{i,j} \end{pmatrix}$ *we have* $P^k = 2^{k-1} \begin{pmatrix} \Omega_{i,j}^k & \Omega_{i,j}^{k+1} \\ \Omega_{i,j}^{k-1} & \Omega_{i,j}^k \end{pmatrix}$. *(This equality can be easily checked for $k = 2$ and provable via recurrence for $k \geq 2$). Let us now write the Taylor development of the matrix exponential:*

$$\begin{aligned} \exp(tP) &= \begin{pmatrix} I + t\Omega_{i,j} + t^2\Omega_{i,j}^2 + \frac{2}{3}t^3\Omega_{i,j}^3 + \dots & X \\ tI + t^2\Omega_{i,j} + \frac{2}{3}t^3\Omega_{i,j}^2 + \dots & Y \end{pmatrix} \\ &= \begin{pmatrix} \frac{I}{2} + \frac{1}{2}\exp(2t\Omega_{i,j}) & X \\ \Omega_{i,j}^{-1}\left[\frac{1}{2}\exp(2t\Omega_{i,j}) - \frac{I}{2}\right] & Y \end{pmatrix}, \end{aligned}$$

where the blocks X and Y are not required, hence, not computed (note that the two diagonal blocks are equal so that $Y = \frac{I}{2} + \frac{1}{2}\exp(2t\Omega_{i,j})$). Inserting this expression for $\exp(tP)$ in (7), we get the desired result.

Theorem 2 *The geodesic emanating from $Y(0)$ with initial velocity $\eta := Y(0)_\perp K_{u,v}$ is*

$$Y_{(:,k)}(t) = \begin{cases} Y(0)_{(:,k)} & \text{if } k \neq v \\ Y(0)_{(:,v)}\cos(t) + Y(0)_{\perp_{(:,u)}}\sin(t) & \text{if } k = v. \end{cases} \tag{8}$$

In particular, evaluating it at some t requires $\mathcal{O}(n)$ floating point operations.

Proof 2 *Note that (4) becomes*

$$Y(t) = \begin{pmatrix} Y(0) & Y(0)_\perp K_{u,v} \end{pmatrix} \exp t \begin{pmatrix} 0 & -E_v \\ I & 0 \end{pmatrix} I_{2p,p}, \tag{9}$$

where $E_v = e_v e_v^\top$. Let $P = \begin{pmatrix} 0 & -E_v \\ I & 0 \end{pmatrix}$, we have

$$P^{2k} = \begin{pmatrix} (-E_v)^k & 0 \\ 0 & (-E_v)^k \end{pmatrix} \text{ and } P^{2k+1} = \begin{pmatrix} 0 & (-E_v)^{k+1} \\ (-E_v)^k & 0 \end{pmatrix}.$$

Using again the Taylor development of the matrix exponential, we get:

$$\exp(tP) = \begin{pmatrix} I - \frac{1}{2}t^2E_v + \frac{1}{24}t^4E_v + \dots & X \\ tI - \frac{1}{6}t^3E_v + \frac{1}{120}t^5E_v + \dots & Y \end{pmatrix} = \begin{pmatrix} I + (\cos(t) - 1)E_v & X \\ tI + (\sin(t) - t)E_v & Y \end{pmatrix},$$

where again we do not compute the blocks X and Y as they do not appear in the final expression due to the right multiplication by $I_{2p,p}$. The geodesic (9) is :

$$\begin{aligned} Y(t) &= Y(0)[I + (\cos(t) - 1)e_v e_v^\top] + Y(0)_\perp e_u e_v^\top [tI + (\sin(t) - t)e_v e_v^\top] \\ &= Y(0) + (\cos(t) - 1)Y(0)e_v e_v^\top + Y(0)_\perp \sin(t) e_u e_v^\top, \end{aligned}$$

which can be written column-wise as (8).

4 Stochastic coordinate descent on the Stiefel manifold

We consider without loss of generality the optimization problem

$$\min_{X \in \mathbb{R}^{l \times m}, Y \in \mathrm{St}(p,n)} f(X, Y), \tag{P}$$

in which some parameters are restricted to the Stiefel manifold while others are unconstrained, as often in DNN training. Algorithm 1 presents our proposed Riemannian coordinate descent (St-SRCD) algorithm on the Stiefel manifold. Similarly as in [11], we use the notation $\tilde{g}_X^k$ for the stochastic Euclidean gradient of f with respect to the unconstrained variable X at iteration k, and $\tilde{g}_{Y,i_k}^k$ for the i_k stochastic Riemannian partial derivative of f with respect to Y, obtained by inserting a stochastic estimate of the Euclidean gradient in (3).

Algorithm 1 St-SRCD: Stochastic RCD on the Stiefel manifold

Let $\{\alpha^k\}$ be a sequence of stepsizes. Set $k = 0$, and initialize the unconstrained and orthogonal variables $X^0 \in \mathbb{R}^{l\times m}, Y^0 \in \mathrm{St}(p,n)$.
while not converged **do**
  Compute the (stochastic) gradient $\tilde{g}_X^k$
  Update the unconstrained variable: $X^{k+1} = X^k - \alpha^k \tilde{g}_X^k$
  Select a coordinate $i^k \in \{1,\dots,D\}$ of the tangent space $\mathcal{T}_{Y^k}\mathrm{St}(p,n)$
  Compute the (stochastic) Riemannian partial derivative $\tilde{g}_{Y,i_k}^k$ using (3)
  Update the orthogonal variable by letting Y^{k+1} be the point of the geodesic starting from Y^k, with initial velocity $-\alpha^k \tilde{g}_{Y,i_k}^k$, at time $t = 1$ (This geodesic is given by either (5) or (8), depending on the coordinate).
  $k := k + 1$
end while

5 Numerical experiments

We used our proposed optimizer for training a VGG network on CIFAR-10 and CIFAR-100[2]. All CNN kernels were reshaped into matrices constrained to belong to the Stiefel manifold, and reshaped back to the appropriate size afterwards; the other model parameters (e.g., linear classification layers) were kept unconstrained. As a comparison point, we considered SGD with momentum, ADAM, stochastic Riemannian gradient descent, Cayley SGD, and Cayley ADAM [14]. These two first optimizers are fully unconstrained (no orthogonality constraints on the CNN kernels), while the three last impose the same constraints as our approach. All methods were trained with learning rates 0.001 and 0.01 for unconstrained and constrained parameters, respectively. During training, a learning rate scheduler was used with a period (number of epochs before learning rate decrease) of 40 and multiplicative factor .2. We trained all models on the same experimental setup on a single A100 GPU for a fair comparison and benchmarking. It can be observed from results reported in Table 1 that the proposed St-SRCD optimizer consistently achieves better than or comparable performance to the baselines. In particular, we remind that our proposed algorithm has a stricly lower complexity per iteration than RGD.

Table 1: Classification errors(%) on CIFAR-10 and CIFAR-100

Optimizer	**CIFAR-10**	**CIFAR-100**
SGD	6.32	26.84
ADAM	6.61	27.10
CAYLEY SGD	5.77	25.48
CAYLEY ADAM	5.88	25.61
RGD	**5.63**	25.49
St-SRCD (ours)	5.66	**25.47**

[2] `https://www.cs.toronto.edu/~kriz/cifar.html`

Acknowledgments

Massart is supported by the National Physical Laboratory (UK), and the F.R.S.-FNRS (Belgium). Abrol is supported by IIITD-IITD joint research grant (MFIRP-233) and Infosys Foundation via Infosys Centre for AI, IIIT Delhi.

References

[1] M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. In *33rd Int. Conf. on Machine Learning (ICML)*, New York, NY, USA, 2016.

[2] M. Murray, V. Abrol, and J. Tanner. Activation function design for deep networks: linearity and effective initialisation. *Applied and Computational Harmonic Analysis*, 59:117–154, 2022.

[3] N. Bansal, X. Chen, and Z. Wang. Can we gain more from orthogonality regularizations in training deep CNNs? In *Conf. on Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2018.

[4] L. Huang, L. Liu, F. Zhu, D. Wan, Z. Yuan, B. Li, and L. Shao. Controllable Orthogonalization in Training DNNs. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, virtual, 2020.

[5] K. Jia, S. Li, Y. Wen, T. Liu, and D. Tao. Orthogonal deep neural networks. *IEEE Tr. on Pattern Analysis and Machine Intelligence*, 43(4):1352 – 1368, 2020.

[6] W. Hu, L. Xiao, and J. Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks. In *8th Int. Conf. on Learning Representations (ICLR)*, virtual, 2020.

[7] P. A. Sokòl and I. M. Park. Information geometry of orthogonal initializations and training. In *Int. Conf. on Learning Representations (ICLR)*, virtual, 2020.

[8] J. Muller, R. Klein, and M. Weinmann. Orthogonal Wasserstein GANs. arxiv:1911.13060, 2019.

[9] A. Pandey, M. Fanuel, J. Schreurs, and J. A. K. Suykens. Disentangled Representation Learning and Generation With Manifold Optimization. *Neural Computation*, 34(10):2009–2036, 2022.

[10] U. Shalit and G. Chechik. Coordinate-descent for learning orthogonal matrices through Givens rotations. In *Int. Conf. on Machine Learning (ICML)*, Beijing, China, 2014.

[11] E. Massart and V. Abrol. Coordinate descent on the orthogonal group for recurrent neural network training. In *36th AAAI Conf. on Artificial Intelligence (AAAI)*, virtual, 2022.

[12] D. H. Gutman and N. Ho-Nguyen. Coordinate descent without coordinates: Tangent subspace descent on Riemannian manifolds. arxiv preprint 1912.10627, 2020.

[13] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[14] J. Li, F. Li, and S. Todorovic. Efficient Riemannian Optimization on the Stiefel Manifold via the Cayley Transform. In *Int. Conf. on Learning Representations (ICLR)*, virtual, 2020.