Stochastic Flows and Geometric Optimization on the Orthogonal Group

Krzysztof Choromanski [1 2] **David Cheikhi** [* 3] **Jared Davis** [* 1] **Valerii Likhosherstov** [* 4] **Achille Nazaret** [* 3] **Achraf Bahamou** [* 2] **Xingyou Song** [* 1] **Mrugank Akarte** [2] **Jack Parker-Holder** [5] **Jacob Bergquist** [2] **Yuan Gao** [2] **Aldo Pacchiano** [6] **Tamas Sarlos** [7] **Adrian Weller** [4 8] **Vikas Sindhwani** [1]

Abstract

We present a new class of stochastic, geometrically-driven optimization algorithms on the orthogonal group $O(d)$ and naturally reductive homogeneous manifolds obtained from the action of the rotation group $SO(d)$. We theoretically and experimentally demonstrate that our methods can be applied in various fields of machine learning including deep, convolutional and recurrent neural networks, reinforcement learning, normalizing flows and metric learning. We show an intriguing connection between efficient stochastic optimization on the orthogonal group and graph theory (e.g. matching problem, partition functions over graphs, graph-coloring). We leverage the theory of Lie groups and provide theoretical results for the designed class of algorithms. We demonstrate broad applicability of our methods by showing strong performance on the seemingly unrelated tasks of learning world models to obtain stable policies for the most difficult $\mathrm{Humanoid}$ agent from $\mathrm{OpenAI\ Gym}$ and improving convolutional neural networks.

1. Introduction and Related Work

Constrained optimization is at the heart of modern machine learning (ML) (Kavis et al., 2019; Vieillard et al., 2019) as many fields of ML benefit from methods such as linear, quadratic, convex or general nonlinear constrained optimization. Other applications include reinforcement learning with safety constraints (Chow et al., 2018). Our focus in this paper is *on-manifold* optimization, where constraints require solutions to belong to certain matrix manifolds. We are mainly interested in three spaces: the orthogonal group $\mathcal{O}(d) = \{\mathbf{M} \in \mathbb{R}^{d \times d} : \mathbf{M}^\top \mathbf{M} = \mathbf{I}_d\}$, its generalization, the *Stiefel manifold* $\mathcal{ST}(d,k) = \{\mathbf{M} \in \mathbb{R}^{d \times k} : \mathbf{M}^\top \mathbf{M} = \mathbf{I}_k\}$ for $k \leq d$ and its subgroup $\mathcal{SO}(d) = \{\mathbf{M} \in \mathbb{R}^{d \times d} : \mathbf{M}^\top \mathbf{M} = \mathbf{I}_d, \det(\mathbf{M}) = 1\}$ of d-dimensional rotations.

Interestingly, a wide variety of constrained optimization problems can be rewritten as optimization on matrix manifolds defined by orthogonality constraints. These include:

- **metric learning:** optimization leveraging the decomposition of k-rank Mahalanobis matrices $\mathbf{M} = \mathbf{U}\mathbf{D}_+\mathbf{U}^\top$, where: $\mathbf{U} \in \mathcal{ST}(d,k)$ and $\mathbf{D}_+$ is diagonal with positive nonzero diagonal entries (Shukla & Anand, 2015);

- **synchronization over the special Euclidean group**, where its elements $g_i \in \mathcal{SE}(d) = \mathbb{R}^d \times \mathcal{SO}(d)$ need to be retrieved from noisy pairwise measurements encoding the elements $g_j g_i^{-1}$ that transform g_i to g_j; these algorithms find applications particularly in vision and robotics (SLAM) (Rosen et al., 2019);

- **PSD programs with block-diagonal constraints:** optimization on $\mathcal{ST}(d,k)$ can be elegantly applied for finding bounded-rank solutions to positive semidefinite (PSD) programs, where the PSD matrix $\mathbf{Z}$ can be decomposed as: $\mathbf{Z} = \mathbf{R}^\top \mathbf{R}$ for a low-rank matrix $\mathbf{R} \in \mathbb{R}^{r \times d}$ ($r \ll d$) and where r can be systematically increased leading effectively to the so-called *Riemannian staircase* method;

- **combinatorial optimization:** combinatorial problems involving object-rearrangements such as sorting or graph isomorphism (Zavlanos & Pappas, 2008) can be cast as optimization on the orthogonal group $O(d)$ which is a relaxation of the permutation group $\mathrm{Perm}(d)$.

Orthogonal groups are deeply rooted in the theory of manifolds. As noted in (Gallier, 2011) " *...most familiar spaces are naturally reductive manifolds. Remarkably, they all arise from some suitable action of the rotation group* $\mathcal{SO}(d)$,

[*]Equal contribution [1]Google Brain Robotics, New York, USA [2]Department of Industrial Engineering and Operations Research, Columbia University, New York, USA [3]Department of Computer Science, Columbia University, New York, USA [4]Department of Engineering, University of Cambridge, Cambridge, United Kingdom [5]Department of Engineering, University of Oxford, Oxford, United Kingdom [6]Department of Computer Science, University of California, Berkeley, USA [7]Google Research, Mountain View, USA [8]The Alan Turing Institute, London, United Kingdom. Correspondence to: Krzysztof Choromanski <kchoro@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

a Lie group, which emerges as the master player." These include in particular the Stiefel and Grassmann manifolds.

It is striking that the group $O(d)$ and its "relatives" provide solutions to many challenging problems in machine learning, where unstructured (i.e. unconstrained) algorithms do exist. Important examples include the training of recurrent and convolutional neural networks, and normalizing flows. *Sylvester normalizing flows* are used to generate complex probabilistic distributions and encode invertible transformations between density functions using matrices taken from $\mathcal{ST}(d,k)$ (van den Berg et al., 2018). Orthogonal convolutional layers were recently shown to improve training models for vision data (Wang et al., 2019; Bansal et al., 2018). Finally, the learning of deep/recurrent neural network models is notoriously difficult due to the *vanishing/exploding gradient* problem (Bengio et al., 1993). To address this challenge, several architectures such as LSTMs and GRUs were proposed (Greff et al., 2015; Cho et al., 2014) but don't provide guarantees that gradients' norms would stabilize. More recently orthogonal RNNs (ORNN) that do provide such guarantees were introduced. ORNNs impose orthogonality constraints on hidden state transition matrices (Arjovsky et al., 2016; Henaff et al., 2016; Helfrich et al., 2018). Training RNNs can now be seen as optimization on matrix manifolds with orthogonality constraints.

However, training orthogonal RNNs is not easy and reflects challenges common for general on-manifold optimization (Absil et al., 2008; Hairer, 2005; Edelman et al., 1998). Projection methods that work by mapping the unstructured gradient step back into the manifold have expensive, cubic time complexity. Standard geometric *Riemannian-gradient* methods relying on steps conducted in the space tangent to the manifold at the point of interest and translating updates back to the manifold via *exponential* or *Cayley* mapping (Helfrich et al., 2018) still require cubic time for those transformations. Hence, in practice orthogonal optimization becomes problematic in higher-dimensional settings.

Another problem with these transforms is that apart from computational challenges, they also incur numerical instabilities ultimately leading to solutions diverging from the manifold. We demonstrate this in Sec. 6 and provide additional evidence in the Appendix (Sec. 9.1.2, 9.3), in particular on a simple 16-dimensional combinatorial optimization task.

An alternative approach is to parameterize orthogonal matrices by unconstrained parameters and conduct standard gradient step in the parameter space. Examples include decompositions into short-length products of Householder reflections and more (Mhammedi et al., 2017; Jing et al., 2017). Such methods enable fast updates, but inherently lack representational capacity by restricting the class of representable matrices and thus are not a subject of our work.

In this paper we propose a new class of optimization algorithms on matrix manifolds defined by orthogonality constraints, in particular: the orthogonal group $\mathcal{O}(d)$, its subgroup of rotations $\mathcal{SO}(d)$ and Stiefel manifold $\mathcal{ST}(d,k)$.

We highlight the following contributions:

1. **Fast Optimization:** We present the first on-manifold stochastic gradient flow optimization algorithms for ML with **sub-cubic** time complexity per step as opposed to **cubic** characterizing SOTA (Sec. 3.3) that do not constrain optimization to strictly lower-dimensional subspaces. That enables us to improve training speed without compromising the representational capacity of the original Riemannian methods. We obtain additional computational gains (Sec. 3.1) by parallelizing our algorithms.

2. **Graphs vs. Orthogonal Optimization:** To obtain these, we explore intriguing connections between stochastic optimization on the orthogonal group and graph theory (in particular the matching problem, its generalizations, partition functions over graphs and graph coloring). By leveraging structure of the Lie algebra of the orthogonal group, we map points from the rotation group to weighted graphs (Sec. 3).

3. **Convergence of Stochastic Optimizers:** We provide rigorous theoretical guarantees showing that our methods converge for a wide class of functions $F : \mathcal{M} \to \mathbb{R}$ defined on $\mathcal{O}(d)$ under moderate regularity assumptions (Sec. 5).

4. **Wide Range of Applications:** We confirm our theoretical results by conducting a broad set of experiments ranging from training RL policies and RNN-based world models (Ha & Schmidhuber, 2018) (Sec. 6.1) to improving vision CNN-models (Sec. 6.2). In the former, our method is the only one that trains stable and effective policies for the high-dimensional Humanoid agent from OpenAI Gym. We carefully quantified the impact of our algorithms. The RL experiments involved running 240 training jobs, each distributed over hundreds of machines. We also demonstrated numerical instabilities of standard non-stochastic techniques (Sec. 9.1.2 and Sec. 9.3).

Our algorithm can be applied in particular in the black-box setting (Salimans et al., 2017), where function to be optimized is accessible only through potentially expensive querying process. We demonstrate it in Sec. 6.1, where we simultaneously train the RNN-model and RL-policy taking advantage of it through its latent states. Full proofs of our theoretical results are in the Appendix.

2. The Geometry of the Orthogonal Group

In this section we provide the reader with technical content used throughout the paper. The theory of matrix manifolds is vast so we focus on concepts needed for the exposition of our results. We refer to Lee (2012) for a more thorough introduction to the theory of smooth manifolds.

Definition 2.1 (manifold embedded in $\mathbb{R}^n$)**.** *Given $n, d \in \mathbb{Z}_{\geq 1}$ with $n \geq d$, a d-dimensional smooth manifold in $\mathbb{R}^n$ is a nonempty set $\mathcal{M} \subseteq \mathbb{R}^n$ such that for every $\mathbf{p} \in \mathcal{M}$ there are two open sets $\Omega \subseteq \mathbb{R}^d$ and $U \subseteq \mathcal{M}$ with $\mathbf{p} \in U$ and a smooth function $\phi : \Omega \to \mathbb{R}^n$ (called local parametrization of $\mathcal{M}$ at $\mathbf{p}$) such that ϕ is a homeomorphism between Ω and $U = \phi(\Omega)$, and $\mathbf{d}\phi(t_0)$ is injective, where $t_0 = \phi^{-1}(\mathbf{p})$.*

Matrix manifolds $\mathcal{M} \subseteq \mathbb{R}^{N \times N}$ are included in the above as embedded in $\mathbb{R}^{N \times N}$ after vectorization. Furthermore, they are usually equipped with additional natural group structure. This enables us to think about them as *Lie groups* (see Lee, 2012) that are smooth manifolds with smooth group operation. For the matrix manifolds considered here, the group operation is always standard matrix multiplication.

One of the key geometric concepts for on-manifold optimization is the notion of the tangent space.

Definition 2.2 (tangent space $\mathcal{T}_{\mathbf{p}}(\mathcal{M})$)**.** *The tangent space $\mathcal{T}_{\mathbf{p}}(\mathcal{M})$ to a smooth manifold $\mathcal{M} \subseteq \mathbb{R}^n$ at point $\mathbf{p} \in \mathcal{M}$ is a space of all vectors $\mathbf{v} = \gamma'(0)$, where $\gamma : (-1, 1) \to \mathcal{M}$ is a smooth curve on $\mathcal{M}$ such that $\mathbf{p} = \gamma(0)$.*

We refer to Sec. 9.4.1 for a rigorous definition of smooth curves on manifolds. It is not hard to see that $\mathcal{T}_{\mathbf{p}}(\mathcal{M})$ is a vector space of the same dimensionality as $\mathcal{M}$ (from the injectivity of $\mathbf{d}\phi$ in $\phi^{-1}(\mathbf{p})$) that can be interpreted as a local linearization of $\mathcal{M}$. Thus it is not surprising that mapping from tangent spaces back to $\mathcal{M}$ will play an important role in on-manifold optimization (see: Subsection 2.1.1).

For a Lie group $\mathcal{M} \subseteq \mathbb{R}^{N \times N}$, we call the tangent space at $\mathbf{I}_N$ the corresponding *Lie algebra*.

2.1. Manifold $\mathcal{O}(d)$ and $\mathcal{ST}(d, k)$

For every $\mathbf{X} \in \mathcal{ST}(d, k)$, there exists a matrix, which we call $\mathbf{X}_\perp$, such that $[\mathbf{X}\mathbf{X}_\perp] \in \mathcal{O}(d)$, where $[]$ stands for matrix concatenation. In fact such a matrix can be chosen in more than one way if $k < d$ (for $k = d$ we take $\mathbf{X}_\perp = \emptyset$).

Lemma 2.3. *For every $\mathbf{X} \in \mathcal{ST}(d, k)$, the tangent space $\mathcal{T}_{\mathbf{X}}(\mathcal{M})$, where $\mathcal{M} = \mathcal{ST}(d, k)$ and $\mathrm{Sk}(k)$ stands for skew-symmetric (antisymmetric) matrices, satisfies (Lee, 2012):*

$$\mathcal{T}_{\mathbf{X}}(\mathcal{M}) = \{\mathbf{X}\mathbf{A} + \mathbf{X}_\perp\mathbf{B} : \mathbf{A} \in \mathrm{Sk}(k), \mathbf{B} \in \mathbb{R}^{(d-k)\times k}\}.$$

We conclude that tangent spaces for the orthogonal group $\mathcal{O}(d)$ are of the form: $\{\mathbf{U}\Omega : \Omega \in \mathrm{Sk}(d)\}$, where $\mathbf{U} \in \mathcal{O}(d)$ and $\mathrm{Sk}(d)$ constitutes its Lie algebra with basis $\mathcal{H}_d$ consisting of matrices $(\mathbf{H}_{i,j})_{1 \leq i < j \leq d}$ s.t. $\mathbf{H}_{i,j}[i, j] = -\mathbf{H}_{i,j}[j, i] = 1$ and $\mathbf{H}_{i,j}[k, l] = 0$ if $\{k, l\} \neq \{i, j\}$.

Smooth manifolds whose tangent spaces are equipped with inner products (see: Sec. 9.4.2 for more details) are called *Riemannian manifolds* (Lee, 2012). Inner products provide a means to define a metric and talk about non-Euclidean distances between points on the manifold (or equivalently: lengths of geodesic lines connecting points on the manifold).

2.1.1. Exponential mapping & Cayley transform

Points $\mathbf{U}\Omega$ of a tangent space $\mathcal{T}_{\mathbf{U}}(\mathcal{O}(d))$ can be mapped back to $\mathcal{O}(d)$ via matrix exponentials by the mapping $\mathbf{U}\exp(t\Omega)$ for $t \in \mathbb{R}$. Curves $\gamma_\Omega : \mathbb{R} \to \mathcal{O}(d)$ defined as $\gamma_\Omega(t) = \mathbf{U}\exp(t\Omega)$ are in fact geodesics on $\mathcal{O}(d)$ tangent to $\mathbf{U}\Omega$ in $\mathbf{U}$. The analogue of the set of canonical axes are the geodesics induced by the canonical basis $\mathcal{H}_d$. The following lemma describes these geodesics.

Lemma 2.4 (Givens rotations and geodesics)**.** *The geodesics on $\mathcal{O}(d)$ induced by matrices $\mathbf{H}_{i,j}$ are of the form $\mathbf{U}\mathbf{G}^t_{i,j}$, where $\mathbf{G}^t_{i,j}$ is a t-angle Givens rotation in the 2-dimensional space $\mathrm{Span}\{\mathbf{e}_i, \mathbf{e}_j\}$ defined as: $\mathbf{G}^t_{i,j}[i, i] = \mathbf{G}^t_{i,j}[j, j] = \cos(t), \mathbf{G}^t_{i,j}[i, j] = -\mathbf{G}^t_{i,j}[j, i] = \sin(t)$ and $\mathbf{G}^t_{i,j}[k, l] = \mathbf{I}_d[k, l]$ for $(k, l) \notin \{(i, i), (i, j), (j, i), (j, j)\}$.*

All introduced geometric concepts are illustrated in Fig. 1.

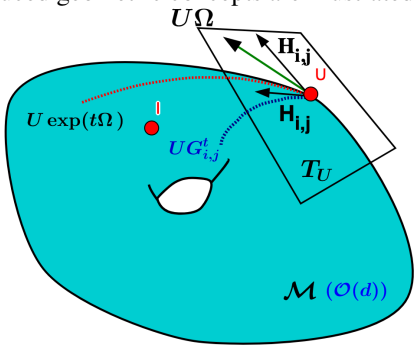

Figure 1. Illustration of basic geometric manifold concepts: tangent vector space $\mathcal{T}_{\mathbf{U}}$ at point $\mathbf{U} \in \mathcal{M}$. If $\mathcal{M} = \mathcal{O}(d)$ then $\mathcal{T}_{\mathbf{U}} = \{\mathbf{U}\Omega : \Omega \in \mathrm{Sk}(d)\}$ and geodesics are of the form $\mathbf{U}\exp(t\Omega)$ for $t \in \mathbb{R}$. Geodesics tangent to points $\mathbf{U}\mathbf{H}_{i,j}$ are of the form $\mathbf{U}\mathbf{G}^t_{i,j}$.

For the general Stiefel manifold $\mathcal{ST}(d, k)$, exponentials can also be used to map skew-symmetric matrices to curves on $\mathcal{ST}(d, k)$ as follows: $\gamma_\Omega(t) = \exp(t\Omega)\mathbf{X}$, where $\mathbf{X} \in \mathcal{ST}(d, k)$ and exp is mapping $\mathrm{Sk}(d)$ to $\mathcal{O}(d)$. Analogously, curves γ_Ω are tangent to $\Omega\mathbf{X}$ in $\mathbf{X}$. The exponential map is sometimes replaced by the *Cayley transform* defined as: $Y(\Omega) = \left(\mathbf{I} + \frac{\Omega}{2}\right)^{-1}\left(\mathbf{I} - \frac{\Omega}{2}\right)$. And again, it can be shown that curves γ_Ω defined as: $\gamma_\Omega(t) = Y(t\Omega)\mathbf{X}$ are tangent to $\Omega\mathbf{X}$ in $\mathbf{X}$ (since: $\gamma'(0) = -\Omega\mathbf{X}$), even though they are no longer geodesics.

2.1.2. Riemannian optimization on $\mathcal{ST}(d, k)$

Consider an optimization problem of the form:

$$\max_{\mathbf{X} \in \mathbb{R}^{d \times k}, \mathbf{X}^\top\mathbf{X} = \mathbf{I}_k} F(\mathbf{X}) \tag{1}$$

for a differentiable function: $F : \mathbb{R}^{d \times k} \to \mathbb{R}$.

Notice that the standard directional derivative functional $DF_{\mathbf{X}} : \mathbb{R}^{d\times k} \rightarrow \mathbb{R}$ satisfies: $DF_{\mathbf{X}}(\mathbf{Z}) = \sum_{i,j} \frac{\partial F}{\partial \mathbf{X}[i,j]} \mathbf{Z}_{i,j} = \mathrm{tr}(\mathbf{G}^\top \mathbf{Z})$, where $\mathbf{G}$ is a standard gradient matrix. To extend gradient-based techniques to on-manifold optimization, we need to:

1. compute the projection of $\mathbf{G}$ into the tangent space $\mathcal{T}_{\mathbf{X}}(\mathcal{ST}(d,k))$, which we call *Riemannian gradient*,
2. use curves on $\mathcal{ST}(d,k)$ tangent to the Riemannian gradient to make updates on the manifold.

It turns out that to make updates on the manifold, it will suffice to use the curves γ_Ω defined above. It remains to compute Riemannian gradients and corresponding skew-symmetric matrices Ω. Using standard representation theorems (see: 9.4.3), it can be proven (under canonical inner product, see: Sec. 9.4.2) that the Riemannian gradient is of the form $\mathcal{R} = \Omega\mathbf{X}$, where $\Omega \in \mathrm{Sk}(d)$ satisfies:

$$\Omega = \Omega(\mathbf{X}, \mathbf{G}) = \mathbf{G}\mathbf{X}^\top - \mathbf{X}\mathbf{G}^\top. \tag{2}$$

We are ready to conclude that for $\mathbf{X}(t) \in \mathcal{ST}(d,k)$, a differential equation (DE):

$$\dot{\mathbf{X}}(t) = \Omega(t)\mathbf{X}(t), \tag{3}$$

where $\Omega(t) = \mathbf{G}(t)\mathbf{X}(t)^\top - \mathbf{X}(t)\mathbf{G}(t)^\top$ and $\mathbf{G}(t)$ is a standard gradient matrix (for unconstrained optimization) in $\mathbf{X}(t)$ for a given function $F : \mathcal{ST}(d,k) \rightarrow \mathbb{R}$ to be maximized, defines a gradient-flow on $\mathcal{ST}(d,k)$. The flow can be discretized and integrated with the use of either the exponential map or the Cayley transform Y as follows:

$$\mathbf{X}_{i+1} = \Gamma(\eta\Omega(\mathbf{X}_i, \mathbf{G}_i))\mathbf{X}_i, \tag{4}$$

where $\mathbf{G}_i$ is a standard gradient matrix in $\mathbf{X}_i$, $\Gamma = Y$ or $\Gamma = \exp$ and $\eta > 0$ is a step size.

Eq. 4 leads to an optimization algorithm with **cubic** time per optimization step (both exponential and Cayley transforms require cubic time), prohibitive in higher-dimensional applications. We propose our solution in the next section.

Interestingly, if $\mathcal{ST}(d,k) = \mathcal{O}(d)$ (i.e. $d = k$), the order of terms in the RHS of Equation 3 can be reversed and such DE also defines a gradient flow for $\Omega(\mathbf{X}, \mathbf{G}) = \mathbf{X}^\top\mathbf{G} - \mathbf{G}^\top\mathbf{X}$ that can be integrated via Equation 4, with the order of terms in RHS reversed. We refer to it as a *reverse form*.

3. Graph-Driven On-Manifold Optimization

Our algorithms provide an efficient way of computing discretized flows given by Eq. 3, by proposing graph-based techniques for time-efficient stochastic approximations of updates from Eq. 4. We focus on Γ being an exponential map since it has several advantages over Cayley transform (in particular, it is surjective onto the space of all rotations).

We aim to replace $\Omega(\mathbf{X}_i, \mathbf{G}_i)$ with an unbiased sparser low-variance stochastic structured estimate $\widehat{\Omega}$. This will enable us to replace cubic time complexity of an update with sub-cubic. Importantly, an unbiased estimator of Ω does not lead to an unbiased estimator of $\mathbf{X}_{i+1}$. However, as seen from Taylor expansion, the bias can be controlled by the step size η and in practice will not hurt accuracy. We verify this claim both theoretically (Sec. 5) by providing strong convergence results, and empirically (Sec. 6).

A key observation linking our setting with graph theory is that elements of the Lie algebra of $\mathcal{O}(d)$ can be interpreted as adjacency matrices of *weighted tournaments*. Then, specific subsamplings of subtournaments lead to efficient updates. To explain this we need the following definitions.

Definition 3.1 (weighted tournament)**.** *For $\mathbf{W} \in \mathrm{Sk}(d)$, the weighted tournament $T(\mathbf{W})$ is a directed graph with vertices $\{0, 1, ..., d-1\}$ and edges $E(\mathbf{W})$, where $(i,j) \in E(\mathbf{W})$ iff $\mathbf{W}[i,j] > 0$. $\mathbf{W}$ is called an adjacency matrix of $T(\mathbf{W})$. The induced undirected graph is denoted $G_{T(\mathbf{W})}$ and its set of edges $E(G_{T(\mathbf{W})})$. A subtournament S of $T(\mathbf{W})$ is obtained by deleting selected edges of $T(\mathbf{W})$ and zeroing corresponding entries in the adjacency matrix $\mathbf{W}$, to form an adjacency matrix of S (denoted $\mathbf{W}[S]$).*

Assume we can rewrite $\Omega \in \mathbb{R}^{d\times d}$ as:

$$\Omega = \sum_{T\in\mathcal{T}} p_T \Omega_T, \tag{5}$$

where: $\mathcal{T}$ is a family of subtournaments of $T(\Omega)$, $\Omega_T \in \mathrm{Sk}(d)$ and $\{p_T\}_{T\in\mathcal{T}}$ is a discrete probability distribution on $\mathcal{T}$. Then Eq. 5 leads to the unbiased estimator $\widehat{\Omega}$ of Ω, when $\widehat{\Omega} = \Omega_T$ with probability p_T. Such a decomposition is possible only if each edge in $E(\Omega)$ appears at least in one T. We call this a *covering family*.

For a covering family $\mathcal{T}$, the matrices Ω_T can be defined as:

$$\Omega_T = \frac{1}{p_T}\mathbf{M}_{\mathcal{T}} \odot \Omega[T] \tag{6}$$

where $\mathbf{M}_{\mathcal{T}}[i,j] > 0$ is the inverse of the number of $T \in \mathcal{T}$ such that $\{i,j\} \in E(G_T)$ and $\odot$ stands for the Hadamard product. We propose to replace the update from Eq. 4 with the following stochastic variant (with probability p_T):

$$\mathbf{X}_{i+1} = \Gamma(\eta\Omega_T(\mathbf{X}_i, \mathbf{G}_i))\mathbf{X}_i \tag{7}$$

We focus on the family of unbiased estimators $\widehat{\Omega}$ given by equations 5 and 6. In order to efficiently apply the above stochastic approach for on-manifold optimization, we need to construct a family of subtournaments $\mathcal{T}$ and a corresponding probabilistic distribution $\{p_T\}_{T\in\mathcal{T}}$ such that: **(1)** update of $\mathbf{X}_i$ given $\Omega_T(\mathbf{X}_i, \mathbf{G}_i)$ can be computed in sub-cubic time, **(2)** sampling Ω_T via $\mathcal{P} = \{p_T\}$ can be conducted in sub-cubic time, **(3)** the stochastic update is accurate enough.

3.1. Multi-connected-components graphs

To achieve **(1)**, we consider a structured family of subtournaments $\mathcal{T}$. For each $T \in \mathcal{T}$, the corresponding undirected graph G_T consists of several connected components. Then the RHS of Eq. 7 can be factorized leading to an update:

$$\mathbf{X}_{i+1} = \left(\prod_{k=1,...,l} \Gamma(\eta \Omega_{T_k}(\mathbf{X}_i, \mathbf{G}_i)) \right) \mathbf{X}_i, \qquad (8)$$

where l stands for the number of connected components and T_k is a subtournament of T such that G_{T_k} is G_T's k^{th} connected component. We can factorize because matrices $\{\Omega_{T_k}\}_{k=1,...,l}$ all commute since they correspond to different connected components. We will construct the connected components to have the same number of vertices $s = \frac{d}{l}$ (w.l.o.g we can assume that $l \mid d$) such that the corresponding matrices Ω_{T_k} have at most s non-zero columns/rows.

Time complexity of right-multiplying $\Gamma(\eta \Omega_{T_k}(\mathbf{X}_i, \mathbf{G}_i))$ by a $d \times d$ matrix is $O(ds^2)$ thus total time complexity of the update from Eq. 8 is $T_{\text{update}} = O(ds^2 l) = O(d^2 s)$. Thus if $s = o(d)$, we have: $T_{\text{update}} = o(d^3)$. In fact the update from Eq. 8 can be further GPU-parallelized across l threads (since different matrices $\Gamma(\eta \Omega_{T_k}(\mathbf{X}_i, \mathbf{G}_i))$ modify disjoint subsets of entries of each column of the matrix that $\Gamma(\eta \Omega_{T_k}(\mathbf{X}_i, \mathbf{G}_i))$ is right-multiplied by) leading to total time complexity $T^{\text{parallel}}_{\text{update}} = O(ds^2)$ after GPU-parallelization.

Let $\mathcal{T}_s$ be the combinatorial space of all subtournaments T of $T(\Omega)$ such that G_T has s-size connected components. Note that for $2 \leq s < d$, $\mathcal{T}_s$ is of size exponential in d.

3.2. Sampling subtournaments

Points **(2)** and **(3)** are related to each other thus we address them simultaneously. We observe that sampling uniformly at random $\mathcal{T} \sim \text{Unif}(\mathcal{T}_s)$ is trivial: we randomly permute vertices $\{0, 1, ..., d-1\}$ (this can be done in linear time, see: Python Fisher-Yates shuffle), then take first s to form first connected component, next s to form next one, etc. For the uniform distribution over $\mathcal{T}_s$ we have: $\frac{1}{p_T}\mathbf{M}_{\mathcal{T}_s} = \frac{d-1}{s-1}\mathbf{J}_d$, where $\mathbf{J}_d \in \mathbb{R}^{d \times d}$ is all-one matrix (see Lemma 9.3 in the Appendix), thus we have: $\Omega_T = \frac{d-1}{s-1} G_T$. We conclude that sampling Ω_T can be done in time $O(\frac{d}{s}\binom{s}{2}) = O(ds)$.

Instead of $\mathcal{T}_s$, one can consider a much smaller family $\mathcal{T}$, where graphs G_T for different $T \in \mathcal{T}$ consist of disjoint sets of edges (and every edge belongs to one G_T). We call such $\mathcal{T}$ *non-intersecting*. It is not hard to see that in that setting $\mathbf{M}_{\mathcal{T}} = \mathbf{J}_d$ (since edges are not shared across different graphs G_T). Also, the size of $\mathcal{T}$ is at most quadratic in d. Thus sampling Ω_T can be conducted in time $O(d^2)$.

In both cases, $\mathcal{T} = \mathcal{T}_s$ and $\mathcal{T}$ *non-intersecting*, we notice that $\mathbf{M}_{\mathcal{T}} \propto \mathbf{J}_{\mathbf{d}}$. We call such a $\mathcal{T}$ a *homogeneous family*.

3.2.1. Non-uniform distributions

To reduce variance of $\widehat{\Omega}$, uniform distributions should be replaced by non-uniform ones (detailed analysis of the variance of different methods proposed in this paper is given in the Appendix (Sec. 9.6, 9.7)). Intuitively, matrices Ω_T for which graphs G_T have large absolute edge-weights should be prioritized and given larger probabilities p_T. We denote by $\|\|_{\mathcal{F}}$ the Frobenius norm and by w_e the weight of edge e.

Lemma 3.2 (importance sampling)**.** *Given $\mathcal{T}$, a distribution $\mathcal{P}^{\text{opt}}_{\mathcal{T}}$ over $\mathcal{T}$ producing unbiased $\widehat{\Omega}$ and minimizing variance $\text{Var}(\widehat{\Omega}) = \mathbb{E}[\|\widehat{\Omega} - \Omega\|^2_{\mathcal{F}}]$ satisfies: $p^{\text{opt}}_T \sim \sqrt{\sum_{(i,j) \in E(G_T)} (M_{\mathcal{T}} \odot \Omega)[i,j]^2}$. For homogeneous families, it simplifies to $p^{\text{opt}}_T \sim \sqrt{\sum_{e \in E(G_T)} w_e^2}$.*

From now on, we consider homogeneous families. Sampling from optimal $\mathcal{P}$ is straightforward if $\mathcal{T}$ is non-intersecting and can be conducted in time $O(d^2)$, but becomes problematic for families $\mathcal{T}$ of exponential sizes. We now introduce a rich family of distributions for which sampling can be conducted efficiently for homogeneous $\mathcal{T}$.

Definition 3.3 (h-regular distributions)**.** *For even function $h : \mathbb{R} \to \mathbb{R}_+$ such that $h(0) = 0$, we say that distribution $\mathcal{P}^h(\mathcal{T})$ over $\mathcal{T}$ is h-regular if $p^h_T \sim \sum_{e \in E(G_T)} h(w_e)$.*

The core idea is to sample uniformly at random, which we can do for $\mathcal{T}_s$, but then accept sampled T with certain easily-computable probability q^h_T. If T is not accepted, the procedure is repeated. Probability q^h_T should satisfy $q^h_T = \lambda p^h_T$, for a renormalization term $\lambda > 0$.

Algorithm 1 Constructing tournament $T \sim \mathcal{P}^h(\mathcal{T})$ and Ω_T

Hyperparameters: $0 < \alpha, \beta < 1$ (only in version II)
Input: $\Omega \in \text{Sk}(d)$
Output: subtournament T and corresponding Ω_T
Preprocessing: Compute $\tau = \rho \|h(\Omega)\|_1$, where $\rho = 1$ (version I) or $\rho = \frac{s}{d\alpha\beta}$ (version II).
while True **do**
1. sample $T \sim \text{Unif}(\mathcal{T}_s)$ (see: Sec. 3.2),
2. compute $q^h_T = \frac{2h(G_T)}{\tau}$ ($h(G_T)$ as in Lemma 3.5),
3. with probability q^h_T return $\left(T, \frac{\|h(\Omega)\|_1}{2h(G_T)} G_T\right)$.

The larger λ, the smaller the expected number of trials needed to accept sampled T. Indeed, the following is true:

Lemma 3.4. *The expected number of trials before sampled T is accepted is: $\frac{|\mathcal{T}_s|}{\lambda}$, where $|\mathcal{X}|$ stands for the size of $\mathcal{X}$.*

On the other hand, we must have: $\lambda \leq \frac{1}{\max_T p^h_T}$. The following result enables us to choose large enough $\lambda > 0$, so that the expected number of trials remains small.

Lemma 3.5. *For an h-regular distribution on $\mathcal{T}_s$, probability p^h_T is given as: $p^h_T = \frac{2h(G_T)}{W \|h(\Omega)\|_1} \leq$*

$\frac{1}{W}$, *where* $h(G_T) = \sum_{e \in E(G_T)} h(w_e)$, $h(\Omega) = [h(\Omega[i,j])]_{i,j \in \{0,1,\ldots,d-1\}}$, $\|h(\Omega)\|_1 = \sum_{i,j} h(\Omega[i,j])$ *and* $W = (d-2)!/\left((s-2)!(s!)^{\frac{d-s}{s}}(\frac{d-s}{s})!\right)$.

Lemma 3.5 enables us to choose $\lambda = W$ and consequently: $q_T^h = \frac{2h(G_T)}{\|h(\Omega)\|_1}$. We can try to do even better, by taking: $\lambda = \frac{W\|h(\Omega)\|_1}{\tau}$ for any τ such that $\tau^* = 2\max_{T \in \mathcal{T}_s} h(G_T) \leq \tau < \|h(\Omega)\|_1$, leading to: $q_T^h = \frac{2h(G_T)}{\tau}$.

Efficiently finding a nontrivial (i.e. smaller than $\|h(\Omega)\|_1$) upper bounds τ on τ^* is not always possible, but can be trivially done for (α, β, h)-balanced matrices Ω, i.e. Ω such that at least an α-fraction of all entries $\Omega[i,j]$ of Ω satisfy: $h(\Omega[i,j]) \geq \beta \max_{a,b} h(\Omega[a,b])$. It is not hard to see that for such Ω one can take: $\tau = O(\frac{s}{d\alpha\beta}\|h(\Omega)\|_1)$ or $\tau = O(\frac{s}{d}\|h(\Omega)\|_1)$ for $\alpha^{-1}, \beta^{-1} = O(1)$. We observed (see: Section 9.2) that in practice one can often take τ of that order. Our general method for sampling $T \in \mathcal{T}_s$ and the corresponding Ω_T is given in Alg. 1. We have:

Theorem 3.6. *Alg. 1 outputs $T \in \mathcal{T}_s$ from distribution $\mathcal{P}^h(\mathcal{T}_s)$ and corresponding Ω_T. Its expected time complexity is $O(\frac{d^2\gamma\tau}{\|h(\Omega)\|_1}) + \xi$, where γ is time complexity for computing a fixed entry of Ω and ξ - for computing $\|h(\Omega)\|_1$.*

3.2.2. Extensions

Dynamic domains $\mathcal{T}$: One can consider changing sampling domains $\mathcal{T}$ across iterations of the optimization algorithm. If non-intersecting families are used (see: Sec. 3.2), $\mathcal{T}$ can be chosen at each step (or periodically) to minimize the optimal variance given by Lemma 3.2. Optimizing $\mathcal{T}$ is almost always a nontrivial combinatorial problem. We shed light on these additional intrinsic connections between combinatorics and on-manifold optimization in Sec. 9.7.

Partition functions: Our sampling scheme can be modified to apply to distributions from Lemma 3.2. The difficulty lies in obtaining a renormalization factor for probabilities $p_T^h \sim \sqrt{\sum_{e \in E(G_T)} w_e^2}$ which is a nontrivial graph partition function. However it can often be approximated by Monte Carlo methods (Jain et al., 2017). In practice we do not need to use variance-optimal optimizers to obtain good results.

3.3. Time Complexity of the stochastic algorithm

To summarize, we can conduct single optimization step given a sampled T and Ω_T in time $O(d^2 s)$ or even $O(ds^2)$ after further GPU-parallelization. If Ω is given then, due to Theorem 3.6 and by previous analysis, entire sampling procedure can be done in $O(d^2)$ time leading to sub-cubic complexity of the entire optimization step. If Ω is not given, then notice first that computing a fixed entry of Ω takes time $O(k)$ for $\mathcal{ST}(d,k)$ and $O(d)$ for $\mathcal{O}(d)$ or $\mathcal{SO}(d)$. If sampling is conducted uniformly, then it invokes getting $\binom{s}{2}\frac{d}{s} = O(ds)$ edges of the graph (i.e. entries of Ω) and thus total complexity is still sub-cubic. Furthermore, for $\mathcal{ST}(d,k)$ and $k = o(d)$ that remains true even for non-uniform sampling due to Theorem 3.6 since Alg. 1 runs in $O(d^2 k)$ as opposed to $O(d^3)$ time.

Finally, for $\mathcal{O}(d)$, $\mathcal{SO}(d)$ and when $d = O(k)$, by Theorem 3.6, non-uniform sampling can be conducted in time $O(\frac{d^2 s}{\alpha\beta}) + \xi$ for (α, β, h)-balanced Ω. The first term is sub-cubic when $\alpha^{-1}\beta^{-1} = o(\frac{d}{s})$ as we confirm in practical applications (see: Sec. 9.2 and our discussion above). In practice for such Ω, the value $\|h(\Omega)\|_1$ can be accurately estimated for arbitrarily small relative error $\epsilon > 0$ with probability $p = 1 - O(\exp(-(\frac{\epsilon\alpha\beta r}{3d})^2))$ by simple Monte Carlo procedure that approximates $h(\Omega)$ with its sub-sampled version of only $O(r)$ nonzero entries (see: Sec. 9.5.5). In practice one can choose $r = o(d^2)$ resulting in $\xi = o(d^3)$ and sub-cubic complexity of the entire step.

4. Optimizing with Graph Matchings

Note that if $\mathcal{T}$ is chosen in such a way that every connected component of G_T consists of two vertices then G_T is simply a *matching* (Diestel, 2012), i.e. a graph, where every vertex is connected to at most one more vertex. In particular, if $s = 2$, then $\{G_T : T \in \mathcal{T}_s\}$ is a collection of all *perfect matchings* of $G_{T(\Omega)}$, i.e. matchings, where every vertex belongs to some edge. For such $\mathcal{T}$s, the update rule given by Eq. 8 has particularly elegant form, namely:

$$\mathbf{X}_{i+1} = \prod_{k=1,\ldots,l} \mathbf{G}^{\theta_k}_{i_k,j_k} \mathbf{X}_i, \tag{9}$$

for some $\theta_1, \ldots, \theta_l \in [0, 2\pi]$ and $i_1, j_1, \ldots, i_k, j_k$. In other words, the update is encoded by the product of Givens rotations (see: Section 2.1.1). This sheds new light on recently proposed algorithms using products of Givens rotations to learn neural networks for RL (Choromanski et al., 2019) or to approximate Haar measure on $\mathcal{O}(d)$, (see: Sec. 9.9).

Constructing a good non-intersecting family $\mathcal{T}$ in this setting is intrinsically related to graph matching/edge coloring optimization problems since $\mathcal{T}$ can be obtained by iteratively finding heavy matchings (i.e. monochromatic classes of valid edge colorings) of G_T. In Sec. 9.7 we explain these connections in more detail. Even though not central for our main argument, they provide additional deeper context.

5. Convergence Results

We show that our stochastic optimizers have similar convergence rates as deterministic ones. The difference is quantified by the term σ related to the variance of the estimator of Ω (see below). Without loss of generality we consider optimization on $\mathcal{O}(d)$. Analogous results hold for $\mathcal{ST}(d,k)$.

Theorem 5.1. *Let $F : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}$ be such that standard gradient ∇F is defined on $\mathcal{O}(d)$ and for all $\mathbf{M}, \mathbf{N} \in \mathcal{O}(d)$,*

$$\|\nabla F(\mathbf{M}) - \nabla F(\mathbf{N})\|_{\mathcal{F}} \leq L\|\mathbf{M} - \mathbf{N}\|_{\mathcal{F}}, \tag{10}$$

where $L > 0$ [1] *and* $\|\cdot\|_{\mathcal{F}}$ *is a Frobenius norm. Let* $\{\mathbf{X}_i\}_{i\geq 0}$ *be the sequence generated by the proposed stochastic update.* $\mathbf{X}_0 \in \mathcal{O}(d)$ *is fixed and* $\mathbf{X}_{i+1} := \exp(\eta_i \widehat{\Omega}_i)\mathbf{X}_i$, *where* $\widehat{\Omega}_i$ *is drawn from distribution* $\mathbb{P}(\widehat{\Omega}_i)$ *defined on* $\mathrm{Sk}(d)$ *s.t.* $\mathbb{E}\widehat{\Omega}_i = \Omega_i$ *and* $\Omega_i := \nabla F(\mathbf{X}_i)\mathbf{X}_i^\top - \mathbf{X}_i \nabla F(\mathbf{X}_i)^\top$. *Here* $\{\eta_i > 0\}_{i\geq 0}$ *is a sequence of step sizes. Then*

$$\min_{i=\overline{0..T}} \mathbb{E}\|\nabla_{\mathcal{O}} F(\mathbf{X}_i)\|_{\mathcal{F}}^2 \leq 2\frac{F^* - F(\mathbf{X}_0)}{\sum_{i=0}^T \eta_i} + \Sigma_T$$

where $\Sigma_T = \sigma^2\left((2\sqrt{d}+1)L + \|\nabla F(\mathbf{I}_d)\|_{\mathcal{F}}\right)\frac{\sum_{i=0}^T \eta_i^2}{\sum_{i=0}^T \eta_i}$, $\nabla_{\mathcal{O}} F$ *denotes a Riemannian gradient (see: Sec. 2.1.2),* $F^* = \sup_{\mathbf{X}\in\mathcal{O}(d)} F(\mathbf{X})$ *and* $\sigma^2 > 0$ *is chosen so that* $\forall \Omega \in \{\nabla_{\mathcal{O}} F(\mathbf{X})\mathbf{X}^\top | \mathbf{X} \in \mathcal{O}(d)\} : \sigma^2 \geq \mathbb{E}\|\widehat{\Omega}\|_{\mathcal{F}}^2$.

6. Experiments

6.1. RL with Evolution Strategies and RNNs

Here we demonstrate the effectiveness of our approach in optimizing policies for a variety of continuous RL tasks from the OpenAI Gym (Humanoid, Walker2d, HalfCheetah) and DM Control Suite (Reacher : Hard, HopperStand and Swimmer : 15).

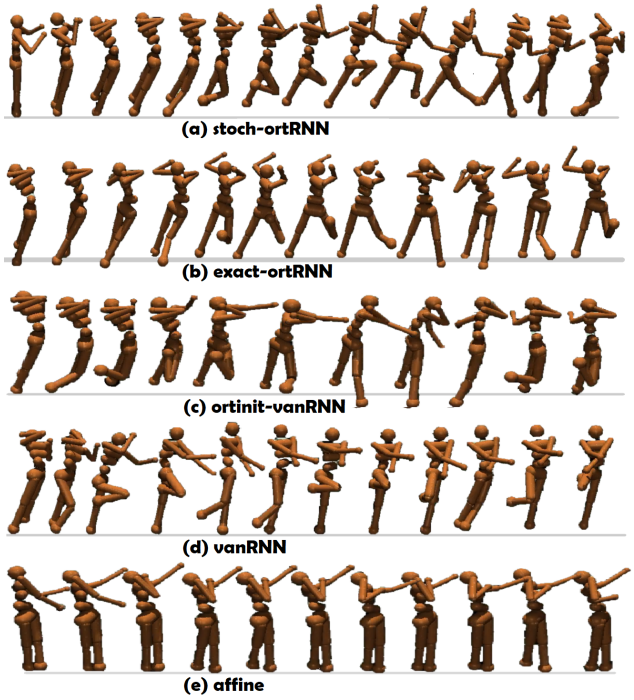

Figure 2. Visualizations of policies learned by different algorithms for Humanoid from OpenAI Gym.

We aim at jointly learn the RNN-based world model (Ha & Schmidhuber, 2018) and a policy affine in the latent (but not original) state, using evolution strategy optimization methods, recently proven to match or outperform SOTA policy gradient algorithms (Salimans et al., 2017). We conduct stochastic optimization on $\mathcal{O}(d)$ by constraining transition-matrices of RNNs to be orthogonal (see: Sec. 1).

Compared methods: Our algorithm (stoch-ortRNN) applying matching-based sampling with $h : x \to |x|$ is compared with three other methods: **(1)** its deterministic variant where exponentials are explicitly computed (exact-ortRNN), **(2)** unstructured vanilla RNNs (vanRNN), **(3)** vanilla RNN with orthogonal initialization of the transition matrix and periodic projections back into orthogonal group (ortinit-vanRNN). For the most challenging Humanoid task we also trained purely affine policies.

Comparing with **(3)** enables us to measure incremental gain coming from the orthogonal optimization as opposed to just orthogonal initialization which was proven to increase performance of neural networks (Saxe et al., 2014). We observed that just orthogonal initialization does not work well (is comparable to vanilla RNN) thus in ortinit-vanRNN we also periodically every p iterations project back onto orthogonal group. We tested different p and observed that in practice it needs to satisfy $p \leq 20$ to provide training improvements. We present variant with $p = 20$ in Fig. 3 since it is the fastest. Detailed ablation studies with different values of p are given in Table 1.

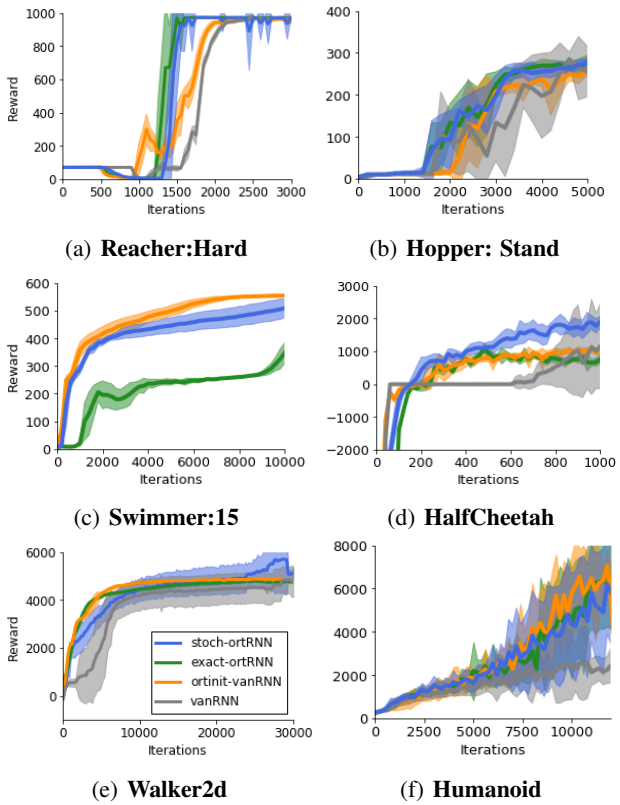

Figure 3. Comparison of all RNN-based algorithms on DM Control Suite (a-c) and OpenAI Gym (d-f) tasks: Each plot shows the mean +- one stdev across $s = 10$ seeds. For Swimmer : 15 we present only three curves since vanRNN did not train at all.

[1] As noted by Shalit & Chechik (2014), because $\mathcal{O}(d)$ is compact, any F with a continuous second derivative will obey (10).

Experiment with affine policies helps us to illustrate that training nontrivial hidden state dynamics is indeed crucial. Purely affine policies were recently showed to provide high rewards for OpenAI environments (Mania et al., 2018), but we demonstrate that they lead to inferior agent's behaviors, thus their rewards are deceptive. To do that, we simulated all learned policies for the most challenging 376-dimensional Humanoid environment (see video library in Appendix).

Setting: For each method we run optimization for $s = 10$ random seeds (resulting in 240 experiments) and hidden state of sizes $h = 200, 400$ (similar results in both settings). For all environments distributed optimization is conducted on 800 machines.

Results: In Fig. 2 we show most common policies learned by all four RNN-based algorithms and trained affine policy for the most challenging Humanoid task. Our method was the only one that consistently produced good walking/running behaviors. Surprisingly, most popular policies produced by exact-ortRNN while still effective, were clearly less efficient. The behavior further deteriorated if orthogonal optimization was replaced by ortinit-vanRNN. Pure vanilla RNNs produced unnatural movements, where agent started spinning and jumping on one leg. For affine policies forward movement was almost nonexistent. We do believe we are the first to apply and show gains offered by orthogonal RNN architectures for RL.

We conjecture that the superiority of our method over exact-ortRNN is partially due to numerical instabilities of standard Riemannian optimization on $\mathcal{O}(d)$ (see: Appendix, Sec 9.1.2). In the Appendix (Sec. 9.3) we demonstrate that it is the case even for very low-dimensional tasks ($d = 16$).

In Fig. 3 we present corresponding training curves for all analyzed RNN-based methods. For all environments stoch-ortRNN method did well, in particular in comparison to exact-ortRNN. In Sec. 6.3 we show that stoc-ortRNN is much faster than other methods using ortho-constraints.

6.2. Optimization on $\mathcal{ST}(d, k)$ for Vision-Based Tasks

Here we apply our methods to orthogonal convolutional neural networks (Jia et al., 2019; Huang et al., 2018; Xie et al., 2017; Bansal et al., 2018; Wang et al., 2019).

Setting: For MNIST, we used a 2-layer MLP with each layer of width 100, and tanh activations specifically to provide a simple example of the vanishing gradient problem, and to benchmark our stochastic orthogonal integrator. The matching-based variants of our algorithm with uniform sampling was accurate enough. For CIFAR10, we used a PlainNet-110 (ResNet-110 (He et al., 2016) without residual layers), similar to the experimental setting in (Xie et al., 2017). For a convolutional kernel of shape $[H, W, C_{in}, C_{out}]$ (denoting respectively [`height, width, in-channel, out-channel`]), we impose orthogonality on the flattened 2-D matrix of shape $[H * W * C_{in}, C_{out}]$ as commonly used in (Jia et al., 2019; Huang et al., 2018; Xie et al., 2017; Bansal et al., 2018; Wang et al., 2019). We applied our algorithm for $s \geq 2$ and uniform sampling. Further training details are in the Appendix 9.1.1.

Results: In Fig. 4, we find that the stochastic optimizer (when $s = 2$) is competitive with the exact orthogonal variant and outperforms vanilla training (SGD + Momentum). In Fig. 5, we present the best test accuracy curve found using a vanilla optimizer (SGD+Momentum), comparable to the best one from (Xie et al., 2017) (66% test accuracy). As in (Xie et al., 2017), we found training PlainNet-110 with vanilla Adam challenging. We observe that orthogonal optimization improves training. Even though stochastic optimizers for $s = 2$ are too noisy for this task, taking $s > 2$ (we used $s = \lceil \frac{d}{\log(d)} \rceil$) provides optimizers competitive with exact orthogonal. Next we present computational advantages of our algorithms over other methods.

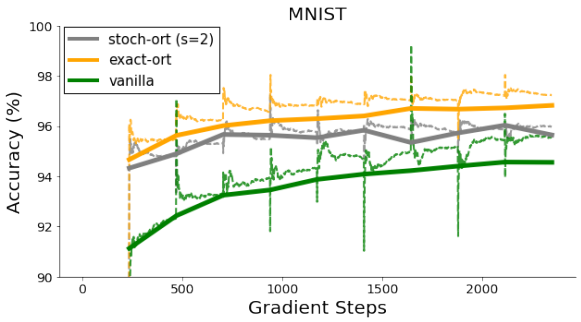

Figure 4. Performance of different methods on MNIST. Thin/Bold curves denote Training/Test accuracy respectively.

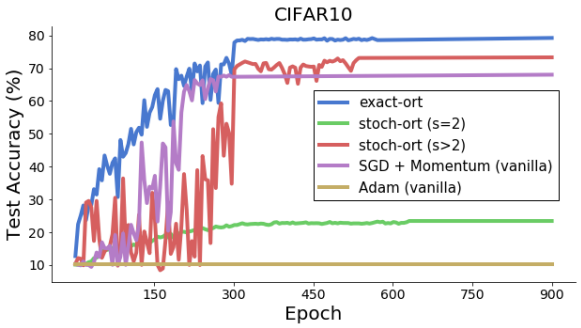

Figure 5. Performance of PlainNet-110 with stochastic optimizer providing improvements over vanilla SGD+Momentum.

6.3. Time Complexity

	ES-200	ES-400	MNIST	CIFAR
exact-ort	> 1600	> 12800	> 200	> 49K
stoch-ort ($s = 2$)	$< \mathbf{68}$	$< \mathbf{272}$	$< \mathbf{8.5}$	< 2K
ortinit-vanRNN ($p = 20$)	> 84	> 656	N/A	N/A
ortinit-vanRNN ($p = 10$)	> 164	> 1296	N/A	N/A
ortinit-vanRNN ($p = 8$)	> 204	> 1616	N/A	N/A
ortinit-vanRNN ($p = 5$)	> 324	> 2576	N/A	N/A
ortinit-vanRNN ($p = 4$)	> 404	> 3216	N/A	N/A
stoch-ort ($s = r^*$)	N/A	N/A	< 43	$< \mathbf{14K}$

Table 1: Comparison of average no of FLOPS [in 10^4] for or-

thogonal matrices updates per step for different methods. ES-h stands for the setting from Sec. 6.1 with hidden state of size h and $r^* = \lceil \frac{d}{\log(d)} \rceil$. Fastest successful runs are bolded.

To abstract from specific implementations, we compare number of FLOPS per iteration for updates of orthogonal matrices in different methods. See results in Table 1. We see that our optimizers outperform other orthogonal optimization methods, and preserve accuracy, as discussed above.

7. Conclusion

We introduced the first stochastic gradient flows algorithms for ML to optimize on orthogonal manifolds that are characterized by sub-cubic time complexity and maintain representational capacity of standard cubic methods. We provide strong connection with graph theory and show broad spectrum of applications ranging from CNN training for vision to learning RNN-based world models for RL.

8. Acknowledgements

Adrian Weller acknowledges support from the David MacKay Newton research fellowship at Darwin College, The Alan Turing Institute under EPSRC grant EP/N510129/1 and U/B/000074, and the Leverhulme Trust via CFI.

References

Absil, P., Mahony, R. E., and Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008. ISBN 978-0-691-13298-3. URL http://press.princeton.edu/titles/8586.html.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 1120–1128, 2016. URL http://proceedings.mlr.press/v48/arjovsky16.html.

Assadi, S., Bateni, M., and Mirrokni, V. S. Distributed weighted matching via randomized composable coresets. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 333–343, 2019. URL http://proceedings.mlr.press/v97/assadi19a.html.

Bansal, N., Chen, X., and Wang, Z. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 4266–4276, 2018.

Bengio, Y., Frasconi, P., and Simard, P. Y. The problem of learning long-term dependencies in recurrent networks. In *Proceedings of International Conference on Neural Networks (ICNN'88), San Francisco, CA, USA, March 28 - April 1, 1993*, pp. 1183–1188, 1993. doi: 10.1109/ICNN.1993.298725. URL https://doi.org/10.1109/ICNN.1993.298725.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734, 2014. URL https://www.aclweb.org/anthology/D14-1179/.

Choromanski, K., Rowland, M., Sindhwani, V., Turner, R. E., and Weller, A. Structured evolution with compact architectures for scalable policy optimization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 969–977, 2018. URL http://proceedings.mlr.press/v80/choromanski18a.html.

Choromanski, K., Pacchiano, A., Pennington, J., and Tang, Y. Kama-nns: Low-dimensional rotation based neural networks. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pp. 236–245, 2019. URL http://proceedings.mlr.press/v89/choromanski19a.html.

Chow, Y., Nachum, O., Duéñez-Guzmán, E. A., and Ghavamzadeh, M. A lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 8103–8112, 2018.

Chu, T., Gao, Y., Peng, R., Sachdeva, S., Sawlani, S., and Wang, J. Graph sparsification, spectral sketches, and faster resistance computation, via short cycle decompositions. *CoRR*, abs/1805.12051, 2018. URL http://arxiv.org/abs/1805.12051.

Czumaj, A., Lacki, J., Madry, A., Mitrovic, S., Onak, K., and Sankowski, P. Round compression for parallel matching algorithms. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pp. 471–484, 2018. doi: 10.1145/3188745.3188764. URL https://doi.org/10.1145/3188745.3188764.

Diestel, R. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. ISBN 978-3-642-14278-9.

Edelman, A., Arias, T. A., and Smith, S. T. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Analysis Applications*, 20(2):303–353, 1998. doi: 10.1137/S0895479895290954. URL https://doi.org/10.1137/S0895479895290954.

Gallier, J. *Geometric Methods and Applications: For Computer Science and Engineering*. Texts in Applied Mathematics. Springer New York, 2011. ISBN 9781441999610. URL https://books.google.co.uk/books?id=4v5VOTZ-vMcC.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. LSTM: A search space odyssey. *CoRR*, abs/1503.04069, 2015. URL http://arxiv.org/abs/1503.04069.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 2455–2467, 2018.

Hairer, E. Important aspects of geometric numerical integration. *J. Sci. Comput.*, 25(1):67–81, 2005. doi: 10.1007/s10915-004-4633-7. URL https://doi.org/10.1007/s10915-004-4633-7.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Helfrich, K., Willmott, D., and Ye, Q. Orthogonal recurrent neural networks with scaled cayley transform. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 1974–1983, 2018. URL http://proceedings.mlr.press/v80/helfrich18a.html.

Henaff, M., Szlam, A., and LeCun, Y. Recurrent orthogonal networks and long-memory tasks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2034–2042, 2016. URL http://proceedings.mlr.press/v48/henaff16.html.

Higham, N. J. The scaling and squaring method for the matrix exponential revisited. *SIAM Review*, 51(4):747–764, 2009. doi: 10.1137/090768539. URL https://doi.org/10.1137/090768539.

Holyer, I. The np-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981. doi: 10.1137/0210055. URL https://doi.org/10.1137/0210055.

Huang, L., Liu, X., Lang, B., Yu, A. W., Wang, Y., and Li, B. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3271–3278, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17072.

Jain, V., Koehler, F., and Mossel, E. Approximating partition functions in constant time. *CoRR*, abs/1711.01655, 2017. URL http://arxiv.org/abs/1711.01655.

Jia, K., Li, S., Wen, Y., Liu, T., and Tao, D. Orthogonal deep neural networks. *CoRR*, abs/1905.05929, 2019. URL http://arxiv.org/abs/1905.05929.

Jing, L., Shen, Y., Dubcek, T., Peurifoy, J., Skirlo, S. A., LeCun, Y., Tegmark, M., and Soljacic, M. Tunable efficient unitary neural networks (EUNN) and their application to rnns. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1733–1741, 2017. URL http://proceedings.mlr.press/v70/jing17a.html.

Kavis, A., Levy, K. Y., Bach, F., and Cevher, V. Unixgrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 6257–6266, 2019.

Lattanzi, S., Moseley, B., Suri, S., and Vassilvitskii, S. Filtering: a method for solving graph problems in MapReduce. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Colocated with FCRC 2011)*, pp. 85–94, 2011. doi: 10.1145/1989493.1989505. URL https://doi.org/10.1145/1989493.1989505.

Lee, J. *Introduction to smooth manifolds. 2nd revised ed*, volume 218. 01 2012. doi: 10.1007/978-1-4419-9982-5.

Mania, H., Guy, A., and Recht, B. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 1805–1814, 2018.

Mhammedi, Z., Hellicar, A. D., Rahman, A., and Bailey, J. Efficient orthogonal parametrisation of recurrent neural networks using Householder reflections. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 2401–2409, 2017. URL http://proceedings.mlr.press/v70/mhammedi17a.html.

Micali, S. and Vazirani, V. V. An o(sqrt(|v|) |e|) algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pp. 17–27, 1980. doi: 10.1109/SFCS.1980.12. URL https://doi.org/10.1109/SFCS.1980.12.

Rosen, D. M., Carlone, L., Bandeira, A. S., and Leonard, J. J. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *I. J. Robotics Res.*, 38(2-3), 2019. doi: 10.1177/0278364918784361. URL https://doi.org/10.1177/0278364918784361.

Rowland, M., Choromanski, K., Chalus, F., Pacchiano, A., Sarlós, T., Turner, R. E., and Weller, A. Geometrically coupled monte carlo sampling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 195–205, 2018.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017. URL http://arxiv.org/abs/1703.03864.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6120.

Shalit, U. and Chechik, G. Coordinate-descent for learning orthogonal matrices through Givens rotations. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 548–556, 2014. URL http://proceedings.mlr.press/v32/shalit14.html.

Shukla, A. and Anand, S. Distance metric learning by optimization on the stiefel manifold. pp. 7.1–7.10, 01 2015. doi: 10.5244/C.29.DIFFCV.7.

van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 393–402, 2018. URL http://auai.org/uai2018/proceedings/papers/156.pdf.

Vieillard, N., Pietquin, O., and Geist, M. On connections between constrained optimization and reinforcement learning. *CoRR*, abs/1910.08476, 2019. URL http://arxiv.org/abs/1910.08476.

Wang, J., Chen, Y., Chakraborty, R., and Yu, S. X. Orthogonal convolutional neural networks. In *arXiv:1911.12207*, 2019.

Xie, D., Xiong, J., and Pu, S. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5075–5084, 2017. doi: 10.1109/CVPR.2017.539. URL https://doi.org/10.1109/CVPR.2017.539.

Zavlanos, M. M. and Pappas, G. J. A dynamical systems approach to weighted graph matching. *Automatica*, 44(11):2817–2824, 2008. doi: 10.1016/j.automatica.2008.04.009. URL https://doi.org/10.1016/j.automatica.2008.04.009.