Biological Learning of Irreducible Representations of Commuting Transformations

Anonymous Author(s)
Affiliation
Address
email

Abstract

A challenge in neuroscience is to understand the neural mechanisms underlying the brain's remarkable ability to learn and detect transformations of objects. Cohen and Welling proposed a mathematical framework for learning efficient representations of transformations based on commutative Lie groups. In particular, they used a decomposition of a commutative Lie group into 2-dimensional irreducible representations. We explore the possibility that the brain uses a similar learning paradigm. Specifically, we propose bio-inspired mechanisms for learning these irreducible representations. We derive two algorithms for learning commutative groups from sequences of transformed images. The two approaches differ in their algorithmic foundations — one is based on SVD of the anti-symmetrized outer product of consecutive frames and the other based on PCA of the difference between consecutive frames. The two algorithms lead to different neural network realizations — the former uses a one layer network while the latter uses a two layer network. Both networks work in an online setting and are able to recover the results from Cohen and Welling, which focus on rotations, as well as learn more generic transformation groups. Our circuits suggest patterns that can be searched for in nascent connectomics and physiology datasets.

1 Introduction

The brains of humans and other animals effortlessly learn and detect transformations of stimuli due to their physical movement and changes of perspective. For example, we can identify faces regardless of their location and orientation in the visual field. Similarly, flies, to maintain stable flight, detect relative translational and rotational motion of their surroundings. Thus, an important problem in neuroscience is to understand how the brain decomposes complex transformations into elementary ones, e.g., translations and rotations, and measures their speed or magnitude.

An influential approach to motion detection and learning is due to Rao and Ruderman [9] and is built around the idea of infinitesimal transformations, also known as Lie group generators. This approach separates the transformation identity (e.g., rotation around a given center, translation in a given direction) from its velocity. One can imagine a neural mechanism implementing a generator, which then will be able to detect a transformation of this kind and measure its velocity [1].

While a single generator describes a one parameter subgroup of transformations, Cohen and Welling [4] considered multi-parameter groups, which in the case of rotations are so-called toroidal subgroups. Toroidal groups arise from decomposing every rotation into 'irreducible representations', which are 2D rotations that commute. Irreducible components can then be combined to describe more complex rotations. Such decomposition was earlier used by Bethge et al. [2], albeit without accounting for varying velocity. The ideas of irreducible representations and toroidal groups jointly open a theoretical possibility for neural mechanisms implementing a larger class of transformations consistently.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Irreducible representations also offer another advantage for neural implementation compared to an approach based on individual generators. Mathematically, a generator is a matrix where each entry corresponds to a pair of pixels of the input image or patch; in a neural implementation, this would correspond to a synapse multiplying the inputs from such a pair, one of them with a delay. Also, a neuron implementing a generator would be required to have a number of multiplicative units that scales quadratically with the number of pixels in the patch, which can be prohibitive for efficient data driven learning. (For the case of translations in a 1D image the latter issue was addressed in [1]). When using the irreducible-representation approach, we will show that the need for multiplicative units does not arise, and only a linear number of synaptic connections will be needed per neuron.

An algorithm for learning toroidal groups from sequences of transformed images was developed by Cohen and Welling [4], but without a focus on biological plausibility. Here, we propose two biological solutions to learning toroidal groups and their irreducible representations. Using the setup from [4], these neural networks receive pairs of transformed images, extract 2D subspaces, and output the estimate of the velocity of rotations within each subspace. Both implementations operate in the online setting, therefore satisfying a basic constraint of biological systems. The first proposal is based on averaging of bivectors [2] and uses singular value decomposition (SVD) to extract the subspace of maximal average rotation. We derive a novel algorithm for online SVD: cross-power iteration. The network finds subsequent rotation subspaces using deflation. The deflation part requires connections between neurons that are not biologically plausible. Our second proposal, based on PCA of time difference vectors, uses local learning rules, which makes it biologically plausible [3]. This approach introduces a two-layer network that decomposes the transformations into their irreducible 2D rotations in the first layer, and outputs the angle of each 2D rotation in the second layer. Having such concrete circuit proposals should allow searching for them in the nascent connectomics and physiology datasets.

2 Setup

Consider a stream of consecutive video frames (Fig. 1) vectorized into d-dimensional pixel-intensity vectors $\{\mathbf{x}_t\}$ satisfying linear relations

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1}. \tag{1}$$

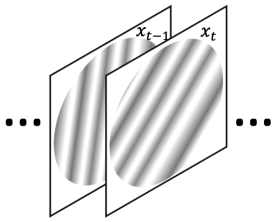


Figure 1: Pairs of consecutive vectorized video frames $(\mathbf{x}_{t-1}, \mathbf{x}_t)$.

Provided the time-step is small, the norm of $\mathbf{x}_t$ is approximately constant between consecutive frames, so we can model $\mathbf{A}_t$ as an orthogonal matrix. In this case, we can decompose $\mathbf{A}_t$ into independent rotations within orthogonal 2D subspaces, as follows

$$\mathbf{A}_t = \mathbf{Q}_t \mathbf{\Gamma}_t \mathbf{Q}_t^\top, \quad \mathbf{\Gamma}_t = \text{blockdiag}(\mathbf{R}(\theta_{t,1}), \ldots, \mathbf{R}(\theta_{t,k})), \quad \mathbf{R}(\theta) := \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{2}$$

Here $\mathbf{Q}_t$ is a $d \times d$ orthogonal matrix, $\mathbf{\Gamma}_t$ is a $d \times d$ block-diagonal matrix whose blocks are 2×2 rotation matrices indexed by $\theta_{t,1}, \ldots, \theta_{t,k}$, and we have assumed, for simplicity, that d is even and $k := d/2$. Pairs of consecutive column vectors $(\mathbf{q}_{t,2i-1}, \mathbf{q}_{t,2i})$ of $\mathbf{Q}_t$ define orthogonal 2D subspaces and $\theta_{t,i}$ represents the angle of rotation in that 2D subspace. As desired, this canonical separates the transformation identity (the 2D subspace) from the velocity (the angle of rotation).

Following Cohen and Welling [4], we consider the problem of learning a representation for the k-dimensional commutative subgroup of rotations of the form $\mathbf{A}_t = \mathbf{Q}\mathbf{\Gamma}_t\mathbf{Q}^\top$, where $\mathbf{Q}$ is a *fixed* orthogonal matrix and $\mathbf{\Gamma}_t$ is any block-diagonal matrix of the form in equation (2). The subgroup is commonly referred to as a *toroidal subgroup* because it is parameterized by the k-dimensional torus $\{(\theta_1, \ldots, \theta_k) : \theta_j \in [0, 2\pi)\}$. With this simplification, the k mutually orthogonal 2D subspaces constitute an *irreducible representation* of the subgroup [4].

The focus of this work is to derive a neurally plausible algorithm for (A) learning the irreducible representation from a stream of pairs of vectorized video frames $\{(\mathbf{x}_{t-1}, \mathbf{x}_t)\}$, and (B) outputting the angle of rotation $\theta_{t,i}$ within each 2D subspace.

3 Learning the irreducible representation from data

We consider learning the irreducible representations from data in the offline setting using 2 approaches. The first approach is based on an SVD of the average bivector between consecutive frames. The second appraoch is based on PCA of the covariance of the difference between consecutive frames. The analysis here will be the starting point for the derivations of the online algorithms. Throughout this section, we assume the sequence $\{\mathbf{x}_t\}$ is centered and whitened so that $\langle \mathbf{x}_t \mathbf{x}_t^\top \rangle_t = \mathbf{I}_d$. In addition, we assume that $\mathbf{A}_t$ is independent of $\mathbf{x}_t$.

3.1 Learning on SVD of bivectors of consecutive frames

Given a pair of consecutive frames $(\mathbf{x}_{t-1}, \mathbf{x}_t)$, define the *bivector*

$$\mathbf{B}_t := \mathbf{x}_t \mathbf{x}_{t-1}^\top - \mathbf{x}_{t-1} \mathbf{x}_t^\top = \mathbf{A}_t \mathbf{x}_t \mathbf{x}_t^\top - \mathbf{x}_t \mathbf{x}_t^\top \mathbf{A}_t^\top \,, \tag{3}$$

where we have used the relation $\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1}$. The bivector is an anti-symmetric matrix, which is also commonly referred to as the 'wedge product' or 'exterior product'. Under our assumptions that $\mathbf{x}_t$ and $\mathbf{A}_t$ are independent and $\mathbf{x}_t$ is whitened, we have

$$\mathbf{B} := \langle \mathbf{B}_t \rangle_t = \langle \mathbf{A}_t - \mathbf{A}_t^\top \rangle_t = \mathbf{Q} \mathbf{G} \mathbf{Q}^\top, \tag{4}$$

where

$$\mathbf{G} := \langle \mathbf{\Gamma}_t - \mathbf{\Gamma}_t^\top \rangle_t = \text{blockdiag}(\mathbf{G}_1, \ldots, \mathbf{G}_k), \qquad \mathbf{G}_i := \begin{pmatrix} 0 & -g_i \\ g_i & 0 \end{pmatrix}, \qquad g_i := 2\langle \sin \theta_i \rangle_t.$$

Note that g_i can be zero even if $\theta_{t,i}$ is not identically zero. For example, because sine is an odd function, if the distribution of $\theta_{t,i}$ is symmetric about zero, then $g_i = 0$. Therefore, to use this approach, we assume that $g_i \neq 0$ for $i = 1, \ldots, k$.

To learn the column vectors of $\mathbf{Q}$, it will be convenient to write the SVD of $\mathbf{B}$ in terms of $\mathbf{Q}$. To this end, we assume, without loss of generality, that $g_1 \geq \cdots \geq g_k \geq 0$. Then we have the following lemma, whose proof is deferred to the appendix.

Lemma 1. *The SVD of* $\mathbf{B}$ *can be written as* $\mathbf{B} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top$*, where*

$$\mathbf{U} := \mathbf{Q}, \qquad \mathbf{\Sigma} := \text{diag}(g_1, g_1, \ldots, g_k, g_k), \qquad \mathbf{V} := [\mathbf{V}_1, \ldots, \mathbf{V}_d],$$

and $(\mathbf{V}_{2i-1}, \mathbf{V}_{2i}) = (\mathbf{Q}_{2i}, -\mathbf{Q}_{2i-1})$ *for* $i = 1, \ldots, k$.

In other words, each pair of left- and right-singular vectors of $\mathbf{B}$, $(\mathbf{U}_i, \mathbf{V}_i)$, spans one of the 2D mutually orthogonal subspaces. Therefore, to learn the irreducible representation, it suffices to learn pairs of left- and right-singular vectors of $\mathbf{B}$.

Note that the first pair of left- and right-singular vectors span the same subspace as the second pair of left- and right-singular vectors (assuming $g_1 > g_2$), and same with the third and fourth pairs, and so on. Therefore, with this approach, care needs to be taken to ensure one is learning distinct 2D subspaces.

3.2 Learning on PCA of differences between consecutive frames

We now present an alternative method for learning the column vectors of $\mathbf{Q}$, by performing PCA on the covariance of the discrete time derivatives defined by $\dot{\mathbf{x}}_t := \mathbf{x}_t - \mathbf{x}_{t-1}$. Under our assumptions that $\mathbf{A}_t$ is independent of $\mathbf{x}_t$ and the vectors $\mathbf{x}_t$ are whitened, we have

$$\langle \dot{\mathbf{x}}_t \dot{\mathbf{x}}_t^\top \rangle_t = 2\mathbf{I}_d - \langle \mathbf{A}_t + \mathbf{A}_t^\top \rangle_t = \mathbf{Q}(2\mathbf{I}_d - \langle \mathbf{\Gamma}_t + \mathbf{\Gamma}_t^\top \rangle_t)\mathbf{Q}^\top = \mathbf{Q} \hat{\mathbf{\Gamma}} \mathbf{Q}^\top, \tag{5}$$

where $\hat{\mathbf{\Gamma}} = \text{diag}(\gamma_1, \gamma_1, \ldots, \gamma_k, \gamma_k)$ and $\gamma_i := 2 - 2\langle \cos \theta_{t,i} \rangle_t$. Importantly, the column vectors of $\mathbf{Q}$ are eigenvectors of $\langle \dot{\mathbf{x}}_t \dot{\mathbf{x}}_t^\top \rangle_t$ with associated eigenvalues γ_i and, assuming that $\gamma_1 > \cdots > \gamma_k$, each γ_i corresponds to a unique 2D subspace. Therefore the transformations with the largest angles can be found by performing PCA on $\langle \dot{\mathbf{x}}_t \dot{\mathbf{x}}_t^\top \rangle_t$. Note that under this approach, $\gamma_i = 0$ if and only if $\theta_{t,i}$ is identically zero.

4 Biological algorithms based on SVD of the average bivector

To compute the SVD of $\mathbf{B}$, we consider the following optimization problem:

$$\max_{\mathbf{U},\mathbf{V}} \mathrm{Tr}(\mathbf{U}^\top \mathbf{B}\mathbf{V}) \qquad \text{s.t.} \qquad \mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}, \tag{6}$$

whose solution is the matrices of left- and right-singular vectors of $\mathbf{B}$.

4.1 Finding the subspace with largest rotation velocity

We start with an optimization problem for finding the top pair of singular vectors:

$$\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^\top \mathbf{B}\mathbf{v} \qquad \text{s.t.} \qquad \|\mathbf{u}\| = \|\mathbf{v}\| = 1\,. \tag{7}$$

To optimize in the online setting, we perform projected stochastic gradient descent, which we refer to as *cross-power iteration*:

$$\mathbf{u} \leftarrow \frac{\mathbf{u} + \eta_t \mathbf{B}_t \mathbf{v}}{\|\mathbf{u} + \eta_t \mathbf{B}_t \mathbf{v}\|}, \qquad \mathbf{v} \leftarrow \frac{\mathbf{v} - \eta_t \mathbf{B}_t \mathbf{u}}{\|\mathbf{v} - \eta_t \mathbf{B}_t \mathbf{u}\|}, \tag{8}$$

where $\eta_t > 0$ is the learning rate and we have used the anti-symmetric property $\mathbf{B}^\top = -\mathbf{B}$.

Normalization steps are hard to map onto a biological circuit, so we follow the idea from Oja's algorithm [6]: assuming small enough learning rates, update equations can be expanded using a Taylor series in η_t. In this case, the iteration takes the form:

$$\mathbf{u} \leftarrow \mathbf{u} + \eta_t\big(\mathbf{B}_t\mathbf{v} - (\mathbf{u}^\top\mathbf{B}_t\mathbf{v})\mathbf{u}\big)\,, \tag{9a}$$

$$\mathbf{v} \leftarrow \mathbf{v} + \eta_t\big(-\mathbf{B}_t\mathbf{u} + (\mathbf{v}^\top\mathbf{B}_t\mathbf{u})\mathbf{v}\big)\,. \tag{9b}$$

Convergence proof for this algorithm is given in the appendix.

Since the matrix $\mathbf{B}$ is antisymmetric, the singular vectors resulting from this optimization, $\mathbf{u}$ and $\mathbf{v}$, are orthogonal, and define the 2D subspace with the largest average angle of rotation. Now each input vector $\mathbf{x}_t$ can be projected on the subspace, and we denote the projected coordinates as $a_t := \mathbf{u}^\top\mathbf{x}_t, b_t := \mathbf{v}^\top\mathbf{x}_t$. Using the definition of $\mathbf{B}_t$ from Eq. (3) and substituting a_t, b_t into the objective:

$$\mathbf{u}^\top\mathbf{B}\mathbf{v} = \mathbf{u}^\top\langle\mathbf{B}_t\rangle_t\mathbf{v} = \langle a_t b_{t-1} - a_{t-1}b_t\rangle_t = \langle y_t\rangle_t, \tag{10}$$

where $y_t := a_t b_{t-1} - a_{t-1}b_t$. This suggests two linear neurons with common input and weights $\mathbf{u}$ and $\mathbf{v}$ generating output streams a_t and b_t, followed by a circuit known as a Hassenstein-Reichardt detector [10]; see Fig. 2(a). Now the angle of rotation θ_t can be recovered:

$$\theta_t = \arcsin\left(\frac{y_t}{\sqrt{a_t^2 + b_t^2}\sqrt{a_{t-1}^2 + b_{t-1}^2}}\right)$$

4.2 Full dimension: finding all subspaces

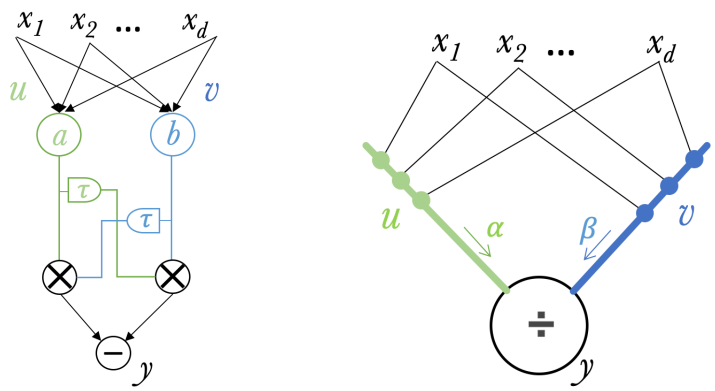
Subsequent singular vector pairs can be found by deflation. Recalling the identities for anti-symmetric SVD ($\mathbf{U}_{2i-1} = \mathbf{V}_{2i}, \mathbf{U}_{2i} = -\mathbf{V}_{2i-1}, \sigma_{2i-1,2i-1} = \sigma_{2i,2i}, \mathbf{B}\mathbf{V}_i = \sigma_i\mathbf{U}_i, \mathbf{B}\mathbf{U}_i = -\sigma_i\mathbf{V}_i$), we can rewrite:

$$\mathbf{B} = \sum_i \sigma_i \mathbf{U}_i\mathbf{V}_i^\top = \sum_{j:\text{odd}} (\mathbf{U}_j\mathbf{V}_j^\top - \mathbf{V}_j\mathbf{U}_j^\top)\sigma_j = \sum_{j:\text{odd}} (\mathbf{U}_j\mathbf{U}_j^\top + \mathbf{V}_j\mathbf{V}_j^\top)\mathbf{B}\,. \tag{11}$$

Then $\mathbf{U}_j\mathbf{V}_j$ for j odd, which is all we need, can be found as solution of an optimization problem similar to Eq. (7), but for the "deflated" matrix:

$$\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^\top\Big(\mathbf{I} - \sum_{j:\text{odd}, j<k} (\mathbf{U}_j\mathbf{U}_j^\top + \mathbf{V}_j\mathbf{V}_j^\top)\Big)\mathbf{B}\mathbf{v} \qquad \text{s.t.} \quad \|\mathbf{u}\| = \|\mathbf{v}\| = 1\,. \tag{12}$$

A straightforward deflation algorithm would find singular vector pairs one by one, corresponding to decreasing singular values. Our goal however is to find all of them in parallel, in an online and neurally plausible manner. The idea is, on every time step, to perform one step of the iterations from Eqns. (9a) and (9b) in a loop on sequentially deflated matrices, using current estimates of singular vectors. This yields Algorithm 1:



(a) (b)

Figure 2: (a) Hassenstein-Reichardt circuit realizing bivector calculation. Nodes a and b project the input vector onto the weight vectors $\mathbf{u}$ and $\mathbf{v}$, then the signal lines split, with delays, combine to compute multiplications, and ultimately are subtracted, as in Eq. (10). (b) Neuron realizing biological cross-power iteration. Synapses on dendritic branches with weights $\mathbf{u}$ and $\mathbf{v}$ also compute time differentiation and integration, forming dendritic currents α and β, followed by division in the soma.

Algorithm 1: Cross-power iteration with deflation

for $i = 1$ to k **do**
  $\mathbf{u}_i, \mathbf{v}_i \leftarrow$ random, $\|\mathbf{u}_i\| = \|\mathbf{v}_i\| = 1$ {Initialize}
end for
for $t = 1, 2, \ldots$ **do**
  $\mathbf{D} \leftarrow 0$ {current deflation estimate}
  for $i = 1$ to k **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \eta_t[\mathbf{B}_t\mathbf{v}_i - \mathbf{u}_i\mathbf{u}_i^\top\mathbf{B}_t\mathbf{v}_i - \mathbf{D}\mathbf{B}_t\mathbf{v}_i]$
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + \eta_t[-\mathbf{B}_t\mathbf{u}_i + \mathbf{v}_i\mathbf{v}_i^\top\mathbf{B}_t\mathbf{u}_i + \mathbf{D}\mathbf{B}_t\mathbf{u}_i]$
    $\mathbf{D} \leftarrow \mathbf{D} + \mathbf{u}_i\mathbf{u}_i^\top + \mathbf{v}_i\mathbf{v}_i^\top$
  end for
end for

The algorithm bears similarity to Generalized Hebbian Algorithm (GHA) for PCA [11], which is also based on deflation. Proof of convergence can be found in the Appendix.

4.3 Biological SVD

The circuit in Fig. 2(a) above is not realizable inside a neuron because of the crossing pathways. It also cannot be realized with two neurons, as the original Hassenstein-Reichardt detector, because then the synaptic weight updates in one neuron would require information only available in the other neuron, violating the principle of locality. To overcome these difficulties we propose a modification of the bivector expression:

$$\tilde{\mathbf{B}}_t = (\mathbf{x}_t - \mathbf{x}_{t-1})(\mathbf{x}_t + \mathbf{x}_{t-1})^\top . \tag{13}$$

which is similar to what was suggested by Bahroun et al. [1]. Individual matrices are no longer anti-symmetric; however, the expectation is anti-symmetric, and is the same as before under our assumptions. Projected on the 2D subspace of interest are now the difference and sum of vectors: $\alpha_t := \mathbf{u}^\top(\mathbf{x}_t - \mathbf{x}_{t-1})$ and $\beta_t := \mathbf{v}^\top(\mathbf{x}_t + \mathbf{x}_{t-1})$. Then substituting in the objective we can rewrite: $\mathbf{u}^\top\tilde{\mathbf{B}}_t\mathbf{v} = \alpha_t\beta_t$.

This suggests a neuron with two dendritic branches both connecting to the same set of inputs, Fig. 2(b). The first has synapses with weights $\mathbf{u}$ that also calculate temporal differences (respond to time derivative); the second one has synapses with weights $\mathbf{v}$ that also integrate over short time periods. The neuron performs division in the soma and outputs $y_t = \alpha_t/\beta_t = \tan\theta_t$, from which the angle can be read. We assume the activity is suppressed when β_t is too small for reliable calculation

of the ratio. That means the neuron is going to have a "blind spot" when $(\mathbf{x}_t + \mathbf{x}_{t-1})$ is close to orthogonal to $\mathbf{v}$.

The neural dynamics are described by the equations:

$$\begin{aligned} \alpha_t &\leftarrow \mathbf{u}^\top(\mathbf{x}_t - \mathbf{x}_{t-1}) \\ \beta_t &\leftarrow \mathbf{v}^\top(\mathbf{x}_t + \mathbf{x}_{t-1}) \\ y_t &\leftarrow \alpha_t/\beta_t\,, \end{aligned} \tag{14}$$

and synaptic plasticity rules are derived from the cross-power iteration algorithm using a Taylor approximation in the learning rate, as in Eqns. (9a)–(9b):

$$\begin{aligned} \mathbf{u} &\leftarrow \mathbf{u} + \eta_t\big(\beta_t(\mathbf{x}_t - \mathbf{x}_{t-1}) - \alpha_t\beta_t\mathbf{u}\big)\,, \\ \mathbf{v} &\leftarrow \mathbf{v} + \eta_t\big(\alpha_t(\mathbf{x}_t + \mathbf{x}_{t-1}) + \alpha_t\beta_t\mathbf{v}\big)\,. \end{aligned} \tag{15}$$

5 Biological algorithm based on PCA of time differences

We construct a two-layer network for outputting the angles $\theta_{t,i}$. The first layer of our network projects the inputs onto the columns of $\mathbf{Q}$, and the second layer computes the angles. We adopt the similarity matching approach [7] to derive a network that projects the inputs onto the 2D subspaces defined by pairs of column vectors of $\mathbf{Q}$. The 'usual' similarity matching approach [8] projects the inputs onto the principal subspace. To ensure that the inputs are projected onto the correct 2D subspace, we adopt the approach in [5] to construct the first layer of our two-layer network, which projects the inputs onto the 2D subspaces spanned by pairs of column vectors of $\mathbf{Q}$ associated with the same eigenvalue.

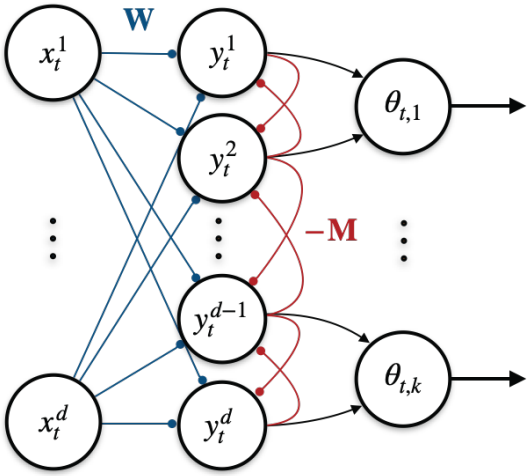


Figure 3: 2-layer network.

5.1 Derivation of the first layer

We assume the first layer of the network has d neurons whose activities at time t are encoded in the vector $\mathbf{y}_t$. Define the data matrices $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_T]$, $\mathbf{Y} := [\mathbf{y}_1, \ldots, \mathbf{y}_T]$, $\dot{\mathbf{X}} := [\dot{\mathbf{x}}_1, \ldots, \dot{\mathbf{x}}_T]$ and $\dot{\mathbf{Y}} := [\dot{\mathbf{y}}_1, \ldots, \dot{\mathbf{y}}_T]$, where recall that the dot represents a discrete time derivative, e.g., $\dot{\mathbf{x}}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$. Let $\dot{\mathbf{X}} = \mathbf{Q}\mathbf{\Sigma}\mathbf{P}^\top$ denote the SVD of the data matrix. We start with the PCA objective function [5]

$$\min_{\mathbf{Y}} -2\,\mathrm{Tr}(\dot{\mathbf{X}}^\top\dot{\mathbf{X}}\dot{\mathbf{Y}}^\top\dot{\mathbf{Y}}) + \mathrm{Tr}(\dot{\mathbf{Y}}^\top\mathbf{\Lambda}^{-1}\dot{\mathbf{Y}}\dot{\mathbf{Y}}^\top\mathbf{\Lambda}^{-1}\dot{\mathbf{Y}}), \tag{16}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_1, \ldots, \lambda_k, \lambda_k)$ is any fixed diagonal matrix with $\lambda_1 > \cdots > \lambda_k$. Suppose that the diagonal elements γ_i of the matrix $\hat{\mathbf{\Gamma}}$ (see Eq. (16)) obey $\gamma_1 > \gamma_2 > \cdots > \gamma_k$. Then the optimal solutions to (16) are given by

$$\mathbf{Y}_{\text{opt}} = \mathbf{\Lambda}\mathbf{R}\mathbf{Q}^\top\mathbf{X}, \tag{17}$$

where $\mathbf{R}$ is any block-diagonal matrix with 2×2 blocks that are orthogonal matrices, see [5, Lemma 1]. In particular, the optimal solution is the projection of the data matrix $\mathbf{X}$ onto the 2D subspaces defined by the pairs of column vectors of $\mathbf{Q}$.

Starting from (16), we derive a minimax objective. First, we substitute $d \times d$ matrices $\mathbf{W}$ and $\mathbf{M}$ in place of $\dot{\mathbf{X}}\dot{\mathbf{Y}}^\top$ and $\dot{\mathbf{Y}}\dot{\mathbf{Y}}^\top$, respectively, to obtain

$$\min_{\mathbf{Y}} \min_{\mathbf{W}} \max_{\mathbf{M}} L(\mathbf{W}, \mathbf{M}, \mathbf{Y}), \tag{18}$$

where

$$L(\mathbf{W}, \mathbf{M}, \mathbf{Y}) = -4\,\mathrm{Tr}(\dot{\mathbf{Y}}^\top \mathbf{W} \dot{\mathbf{X}}) + 2\,\mathrm{Tr}(\mathbf{W}^\top \mathbf{W}) + 2\,\mathrm{Tr}(\dot{\mathbf{Y}}^\top \mathbf{M} \dot{\mathbf{Y}}) - \mathrm{Tr}(\mathbf{\Lambda} \mathbf{M}^\top \mathbf{\Lambda} \mathbf{M}). \tag{19}$$

After interchanging the order of optimization, we arrive at the objective

$$\min_{\mathbf{W}} \max_{\mathbf{M}} \min_{\mathbf{Y}} L(\mathbf{W}, \mathbf{M}, \mathbf{Y}). \tag{20}$$

In the offline setting, we first optimize over $\dot{\mathbf{Y}}$ and find that $\dot{\mathbf{Y}} = \mathbf{M}^{-1}\mathbf{W}\dot{\mathbf{X}}$, or equivalently, $\mathbf{Y} = \mathbf{M}^{-1}\mathbf{W}$. We then take a gradient descent ascent step with respect to $(\mathbf{W}, \mathbf{M})$:

$$\mathbf{W} \leftarrow \mathbf{W} + \eta(\dot{\mathbf{Y}}\dot{\mathbf{X}}^\top - \mathbf{W}), \qquad \mathbf{M} \leftarrow \mathbf{M} + \eta(\dot{\mathbf{Y}}\dot{\mathbf{Y}}^\top - \mathbf{\Lambda}\mathbf{M}\mathbf{\Lambda}). \tag{21}$$

In the online setting, at each time step t, the network receives the input $\mathbf{x}_t$ and computes the output $\mathbf{y}_t$ by running the recurrent neural dynamics to equilibrium:

$$\frac{d\mathbf{y}_t(\gamma)}{d\gamma} = \mathbf{W}\mathbf{x}_t - \mathbf{M}\mathbf{y}_t(\gamma) \qquad \Rightarrow \qquad \mathbf{y}_t = \mathbf{M}^{-1}\mathbf{W}\mathbf{x}_t. \tag{22}$$

The network then computes $\dot{\mathbf{y}}_t := \mathbf{y}_t - \mathbf{y}_{t-1}$ and the synaptic weights are updated according to the Hebbian rules

$$\mathbf{W} \leftarrow \mathbf{W} + \eta(\dot{\mathbf{y}}_t\dot{\mathbf{x}}_t^\top - \mathbf{W}), \qquad \mathbf{M} \leftarrow \mathbf{M} + \eta(\dot{\mathbf{y}}_t\dot{\mathbf{y}}_t^\top - \mathbf{\Lambda}\mathbf{M}\mathbf{\Lambda}). \tag{23}$$

5.2 Derivation of the second layer

The first layer of the network consists of d neurons and it transforms the inputs so that the 2D subspaces represented by pairs of neurons. The second layer of the network consists of k neurons with each neuron computing an angle $\theta_{i,t}$. The ith neuron of the second layer receives inputs $(y_{t-1,2i-1}, y_{t,2i-1}, y_{t-1,2i}, y_{t,2i})$ from the $(2i-1)^{\text{st}}$ and $2i^{\text{th}}$ neurons of the first layer, which satisfy the relation

$$\begin{bmatrix} y_{t,2i-1} \\ y_{t,2i} \end{bmatrix} = \begin{bmatrix} \cos\theta_{t,i} & -\sin\theta_{t,i} \\ \sin\theta_{t,i} & \cos\theta_{t,i} \end{bmatrix} \begin{bmatrix} y_{t-1,2i-1} \\ y_{t-1,2i} \end{bmatrix}. \tag{24}$$

Therefore,

$$\cos\theta_{t,i} = \frac{y_{2i-1,t-1}y_{2i,t} + y_{2i,t-1}y_{2i-1,t}}{y_{2i-1,t-1}^2 + y_{2i,t-1}^2}, \qquad \sin\theta_{t,i} = \frac{y_{2i-1,t-1}y_{2i,t} - y_{2i,t-1}y_{2i-1,t}}{y_{2i-1,t-1}^2 + y_{2i,t-1}^2}, \tag{25}$$

and so

$$\theta_{t,i} = \arctan\left(\frac{y_{2i-1,t-1}y_{2i,t} - y_{2i,t-1}y_{2i-1,t}}{y_{2i-1,t-1}y_{2i,t} + y_{2i,t-1}y_{2i-1,t}}\right), \tag{26}$$

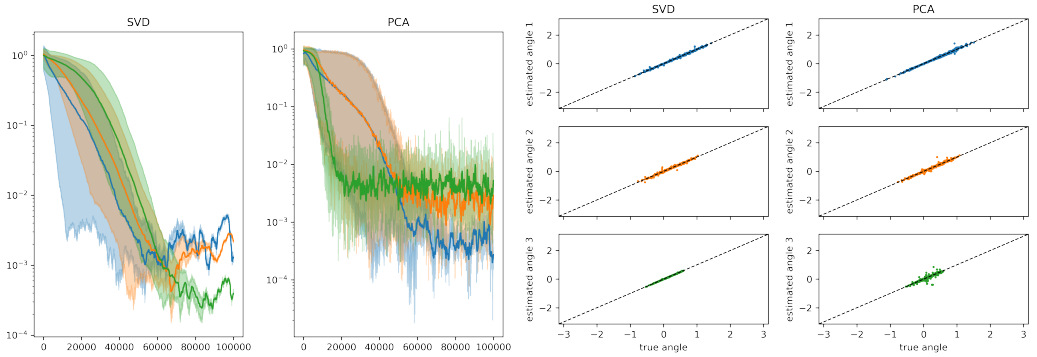
which we define to be the output of the ith neuron in the second layer.

Algorithm 2: Two-layer network (PCA based)

initialize $\mathbf{W}$, $\mathbf{M}$ positive definite, $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{y}_0 = \mathbf{0}$
for $t = 1, 2 \ldots$ **do**
  $\mathbf{y}_t = \mathbf{M}^{-1}\mathbf{W}\mathbf{x}_t$
  $\mathbf{W} \leftarrow \mathbf{W} + \eta((\mathbf{y}_t - \mathbf{y}_{t-1})(\mathbf{x}_t - \mathbf{x}_{t-1})^\top - \mathbf{W})$
  $\mathbf{M} \leftarrow \mathbf{M} + \eta((\mathbf{y}_t - \mathbf{y}_{t-1})(\mathbf{y}_t - \mathbf{y}_{t-1})^\top - \mathbf{\Lambda}\mathbf{M}\mathbf{\Lambda})$
  for $i = 1, \ldots, k$ **do**
    $\theta_{t,i} = \arctan\left(\frac{y_{2i-1,t-1}y_{2i,t} - y_{2i,t-1}y_{2i-1,t}}{y_{2i-1,t-1}y_{2i,t} + y_{2i,t-1}y_{2i-1,t}}\right)$
  end for
end for

6 Simulations

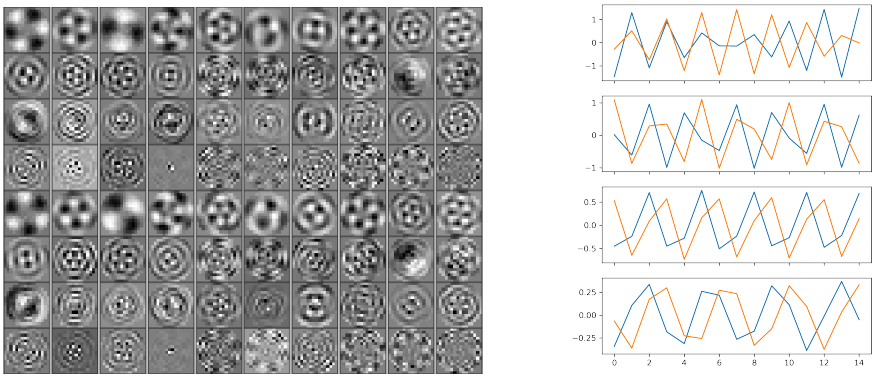
Simulations here are intended to demonstrate the ability of our algorithms to learn arbitrary toroidal groups from data, i.e. recover 2D subspaces and estimate rotation angles, and also generate reasonable filters. For that purpose we created a toroidal group with three 2D subspaces in 10-dimensional space. An orthogonal matrix $\mathbf{Q}$ was generated randomly, then rotations angles in each subspace were generated with independent normal distributions with means 0.3, 0.2, 0.1, and corresponding standard deviations 0.4, 0.3, 0.2. These rotations were applied to vectors generated from a standard normal distribution in 10 dimensions. Simulation results are shown in Fig. 4. In total we generated 10^6 samples and rotations that were fed to both algorithms. Learning rates were manually selected to be $5 \cdot 10^{-4}$ for both algorithms. This was repeated 10 times with random initialization, which took 14 minutes total on a MacBook Pro with 3.5 GHz Dual-Core Intel Core i7 processor. Subspace fit loss is evaluated as one minus cosine between the true and estimated planes. Angle estimates are depicted for the latest 1000 iterations and pooled randomly from all 10 runs.



(a) (b) (c) (d)

Figure 4: Simulations on random vectors. (a) and (c): SVD of bivector; (b) and (d): PCA of vector differences. (a) and (b): Convergence of subspace fit by iteration. Blended area shows the range between maxima and minima across all 10 runs; dense line shows mean across runs, median filtered over 500 iterations. (c) and (d): Angle estimation for each subspace.

Next we look at the filters generated by our algorithms, i.e., pairs of vectors found for each 2D



(a) (b)

Figure 5: Filters obtained from (a) rotations of 2D images, (b) translations of 1D images.

subspace. Rotations of images around the canter is an example that both [2] and [4] looked at. We used random images of size 16×16 and rotated each by random angles around the center. Rotations were performed using bi-linear approximation, ignoring pixels outside the central circle. We ran by SVD algorithm on a total of 2×10^6 images and rotations with the learning rate equal to $2 \cdot 10^{-4}$. Figure 5(a) shows the 40 top filters obtained; corresponding pairs of vectors defining subspaces are on the upper and lower half of the chart. The results are similar to those presented in [2] and [4]. Running time was 40 min on MacBook Pro with 3.5 GHz Dual-Core Intel Core i7 processor.

Sub-pixel translations of 1D images can be approximated by cyclic shifts, which are rotations of the image vector. We generated 10^5 15-pixel 1D images from the standard normal distribution. Top 4 filter pairs obtained by PCA algorithm are shown in Fig. 5(b); they look like Fourier pairs, as expected. Execution took 5 min.

7 Conclusion

In this work, we developed two biologically plausible neural algorithms that can learn toroidal groups of transformations, detect transformations and measure their velocity. Our approach is based on irreducible representations of rotations, as used in [2; 4]. This allowed us to suggest neural implementations that avoid unrealistic multiplicative units encountered in the designs based on Lie group generators.

Our first algorithm is based on SVD of the average bivector, derived from two images under transformation. For that we have developed a novel online SVD algorithm. The second algorithm is based on PCA of the difference between vectorized images under transformation. Arising two-layer network fully adheres to the principles of biological plausibility outlined in [3]. Thus, our work generates experimentally testable hypotheses that can be searched for in nascent connectomics and physiological datasets from various species and brain regions.

References

[1] Yanis Bahroun, Dmitri Chklovskii, and Anirvan Sengupta. A similarity-preserving network trained on transformed images recapitulates salient features of the fly motion detection circuit. In *Advances in neural information processing systems*, pages 14201–14212, 2019.

[2] Matthias Bethge, Sebastian Gerwinn, and Jakob H. Macke. Unsupervised learning of a steerable basis for invariant image representations. *Human Vision and Electronic Imaging XII. Edited by Rogowitz*, 6492:64920C–64920C–12, 2007. doi: 10.1117/12.711119.

[3] Dmitri Chklovskii. The search for biologically plausible neural computation: The conventional approach, 2016. URL http://www.offconvex.org/2016/11/03/MityaNN1/.

[4] Taco Cohen and Max Welling. Learning the irreducible representations of commutative lie groups. In *International Conference on Machine Learning*, pages 1755–1763. PMLR, 2014.

[5] Victor Minden, Cengiz Pehlevan, and Dmitri B Chklovskii. Biologically plausible online principal component analysis without recurrent neural dynamics. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 104–111. IEEE, 2018.

[6] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[7] Cengiz Pehlevan and Dmitri B Chklovskii. Neuroscience-inspired online unsupervised learning algorithms: Artificial neural networks. *IEEE Signal Processing Magazine*, 36(6):88–96, 2019.

[8] Cengiz Pehlevan, Tao Hu, and Dmitri B Chklovskii. A hebbian/anti-hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *Neural Computation*, 27(7):1461–1495, 2015.

[9] Rajesh P. N. Rao and Daniel L. Ruderman. Learning lie groups for invariant visual perception. In *Advances in neural information processing systems*, pages 810—-816, 1999.

[10] Werner Reichardt. Autocorrelation, a principle for evaluation of sensory information by the central nervous system. In *Symposium on Principles of Sensory Communication 1959*, pages 303–317. MIT press, 1961.

[11] Terence D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90044-0. URL https://www.sciencedirect.com/science/article/pii/0893608089900440.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section Simulations
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section Simulations
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section Simulations
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Lemma 1

Define the $d \times d$ orthogonal matrix $\mathbf{P}$ by

$$\mathbf{P} := \text{blockdiag}\left(\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \ldots, \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\right).$$

Since $\mathbf{P}$ is orthogonal, we have

$$\mathbf{B} = \mathbf{Q}\mathbf{G}\mathbf{Q}^\top = \mathbf{Q}(\mathbf{G}\mathbf{P})(\mathbf{Q}\mathbf{P})^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top,$$

where $\mathbf{U} := \mathbf{Q}$, $\mathbf{\Sigma} := \text{diag}(g_1, g_1, \ldots, g_k, g_k)$ and $\mathbf{V} := \mathbf{Q}\mathbf{P}$. We assume, without loss of generality, that $g_1 \geq \cdots \geq g_k \geq 0$. By definition, the $(2i-1)^{\text{st}}$ column vectors of $\mathbf{U}$ and $\mathbf{V}$ and the $2i^{\text{th}}$ column vectors of $\mathbf{U}$ and $\mathbf{V}$ satisfy the relations

$$(\mathbf{u}_{2i-1}, \mathbf{v}_{2i-1}) = (\mathbf{q}_{2i-1}, -\mathbf{q}_{2i}), \qquad (\mathbf{u}_{2i}, \mathbf{v}_{2i}) = (\mathbf{q}_{2i}, \mathbf{q}_{2i+1}), \qquad i = 1, \ldots, k.$$

B Convergence proof for cross-power iteration with deflation

Assume $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ is SVD, singular values in Σ are non-increasing, and $\mathbf{B}$ is full rank. Singular values of anti-symmetric matrix come in pairs, but we will assume for simplicity that there are no two equal pairs. We prove the convergence by induction, similar to the proof for GHA [11], so we start from the top singular vector pair. Differential equations for the iterations in Eqns.8 are

$$\frac{d\mathbf{u}}{dt} = \mathbf{B}\mathbf{v} - (\mathbf{u}^\top\mathbf{B}\mathbf{v})\mathbf{u}, \qquad \frac{d\mathbf{v}}{dt} = -\mathbf{B}\mathbf{u} + (\mathbf{v}^\top\mathbf{B}\mathbf{u})\mathbf{v}. \tag{27}$$

The variables $\mathbf{u}, \mathbf{v}$ can be expanded in the column spaces of orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$: $\mathbf{u} = \mathbf{U}\mathbf{p}, \mathbf{v} = \mathbf{V}\mathbf{q}$. Substituting into Eqn.27, obtain differential equations for the new variables:

$$\frac{d\mathbf{p}}{dt} = \mathbf{\Sigma}\mathbf{q} - \mathbf{p}\mathbf{p}^\top\mathbf{\Sigma}\mathbf{q} \tag{28}$$

$$\frac{d\mathbf{q}}{dt} = \mathbf{\Sigma}\mathbf{p} - \mathbf{q}\mathbf{q}^\top\mathbf{\Sigma}\mathbf{p} \tag{29}$$

For vector $\mathbf{p}$ we follow the evolution of the ratio of the component p_1 corresponding to the top singular value σ_1 and some other p_k such that $\sigma_k < \sigma_1$. From the equations above:

$$\frac{d(p_k/p_1)}{dt} = \frac{p_k}{p_1}\left(\sigma_k\frac{q_k}{p_k} - \sigma_1\frac{q_1}{p_1}\right). \tag{30}$$

For every k-th component of vectors $\mathbf{p}, \mathbf{q}$ we can calculate:

$$\frac{d(q_k/p_k)}{dt} = \sigma_k\left(1 - (q_k/p_k)^2\right). \tag{31}$$

The solution is

$$\frac{q_k(t)}{p_k(t)} = -\frac{1 + C\exp(2\sigma_k t)}{1 - C\exp(2\sigma_k t)}, \tag{32}$$

and converges to 1 when t goes to infinity since $\sigma_k > 0$. Using this, the expression in parenthesis in Eq.30 converges to $\sigma_k - \sigma_1$, which is negative, so p_k/p_1 goes to zero for every k where $\sigma_k < \sigma_1$. We are left to show that the norm of the vector is maintained at 1. For that we follow the evolution of $z := \mathbf{u}^\top\mathbf{u}$:

$$\frac{dz}{dt} = 2\mathbf{p}^\top\frac{d\mathbf{p}}{dt} = 2\mathbf{p}^\top(\mathbf{I} - \mathbf{p}\mathbf{p}^\top)\mathbf{B}\mathbf{q} = 2(1-z)\mathbf{p}^\top\mathbf{B}\mathbf{q} = 0$$

since we initialized $z = 1$. We conclude that $\mathbf{u}$ converges to $\mathbf{U}_1$, top left singular vector. Similarly, $\mathbf{v}$ converges to $\mathbf{V}_1$, top right singular vector.

Having the base for the induction, for the induction step i we assume that for all $k < i$ the variables $\mathbf{u}_k, \mathbf{v}_k$ converge correspondingly to $\mathbf{U}_{2k-1}, \mathbf{V}_{2k-1}$, columns of $\mathbf{U}$ and $\mathbf{V}$. Now need to prove that $\mathbf{u}_i, \mathbf{v}_i$ converge to $\mathbf{U}_{2i-1}, \mathbf{V}_{2i-1}$. Differential equations corresponding to Algorithm 1 are:

$$\frac{d\mathbf{u}_i}{dt} = \mathbf{B}\mathbf{v}_i - \mathbf{u}_i\mathbf{u}_i^\top\mathbf{B}\mathbf{v}_i - \sum_{k<i}\left(\mathbf{u}_k\mathbf{u}_k^\top + \mathbf{v}_k\mathbf{v}_k^\top\right)\mathbf{B}\mathbf{v}_i \tag{33}$$

$$\frac{d\mathbf{v}_i}{dt} = -\mathbf{B}\mathbf{u}_i + \mathbf{v}_i\mathbf{v}_i^\top\mathbf{B}\mathbf{u}_i + \sum_{k<i}\left(\mathbf{u}_k\mathbf{u}_k^\top + \mathbf{v}_k\mathbf{v}_k^\top\right)\mathbf{B}\mathbf{u}_i \tag{34}$$

Substituting as before $\mathbf{u}_i = \mathbf{U}\mathbf{p}_i, \mathbf{v}_i = \mathbf{V}\mathbf{q}_i$ and multiplying from the left the first equation by $\mathbf{U}^\top$ and the second – by $\mathbf{V}^\top$ gives:

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{\Sigma}\mathbf{q}_i - \mathbf{p}_i\mathbf{p}_i^\top\mathbf{\Sigma}\mathbf{q}_i - \sum_{k<i}\left(\mathbf{p}_k\mathbf{p}_k^\top + \mathbf{J}\mathbf{q}_k\mathbf{q}_k^\top\mathbf{J}^\top\right)\mathbf{\Sigma}\mathbf{q}_i \tag{35}$$

$$\frac{d\mathbf{q}_i}{dt} = \mathbf{\Sigma}\mathbf{p}_i - \mathbf{q}_i\mathbf{q}_i^\top\mathbf{\Sigma}\mathbf{p}_i - \sum_{k<i}\left(\mathbf{q}_k\mathbf{q}_k^\top + \mathbf{J}^\top\mathbf{p}_k\mathbf{p}_k^\top\mathbf{J}\right)\mathbf{\Sigma}\mathbf{p}_i \tag{36}$$

where $\mathbf{J} := \mathbf{U}^\top\mathbf{V}$ is a block-diagonal matrix with blocks: $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$. The task now is to prove that p_i and q_i converge to unit vector with 1 at $2i-1$-st component, the rest being zeros.

By the assumption of the induction step, $\mathbf{p}_k, \mathbf{q}_k$ for $k < i$ converge to corresponding unit vectors. Then simple matrix calculation show, that we have the following convergence, as fast as the slowest of $\mathbf{p}_k, \mathbf{q}_k$:

$$\sum_{k<i} \left(\mathbf{p}_k \mathbf{p}_k^\top + \mathbf{J}\mathbf{q}_k \mathbf{q}_k^\top \mathbf{J}^\top\right) \to \mathbf{\Gamma}, \quad \sum_{k<i} \left(\mathbf{q}_k \mathbf{q}_k^\top + \mathbf{J}^\top \mathbf{p}_k \mathbf{p}_k^\top \mathbf{J}\right) \to \mathbf{\Gamma}, \tag{37}$$

where $\mathbf{\Gamma}$ is diagonal matrix, and on the diagonal values 1 occupy first $2(i-1)$ positions, the rest are zeros. Differential equations now take form:

$$\begin{aligned} \frac{d\mathbf{p}_i}{dt} &= \mathbf{\Sigma}(\mathbf{I} - \mathbf{\Gamma})\mathbf{q}_i - \mathbf{p}_i \mathbf{p}_i^\top \mathbf{\Sigma}\mathbf{q}_i \\ \frac{d\mathbf{q}_i}{dt} &= \mathbf{\Sigma}(\mathbf{I} - \mathbf{\Gamma})\mathbf{p}_i - \mathbf{q}_i \mathbf{q}_i^\top \mathbf{\Sigma}\mathbf{p}_i \end{aligned} \tag{38}$$

These are somewhat similar to 28, 29, and we proceed similarly to observe:

$$\begin{aligned} \frac{d(q_{ik}/p_{ik})}{dt} &= \sigma_k(1-\gamma_k)\left(1 - (q_{ik}/p_{ik})^2\right) \\ \frac{d(p_{ik}/p_{im})}{dt} &= \frac{p_{ik}}{p_{im}}\left(\sigma_k(1-\gamma_k)\frac{q_{ik}}{p_{ik}} - \sigma_m(1-\gamma_m)\frac{q_{im}}{p_{im}}\right). \end{aligned} \tag{39}$$

From the first of these equations we conclude that q_{ik}/p_{ik} converges to 1 when $k \geq 2i-1$, i.e. $\gamma_k = 0$. Then for $k > 2i-1$ from the second equation:

$$\frac{d(p_{ik}/p_{i,2i-1})}{dt} = \frac{p_{ik}}{p_{i,2i-1}}\left(\sigma_k - \sigma_{2i-1}\right), \tag{40}$$

which converges to zero because of the ordering of singular values. For $k < 2i-1$ we have:

$$\frac{d(p_{ik}/p_{i,2i-1})}{dt} = -\sigma_{2i-1}\frac{p_{ik}}{p_{i,2i-1}}, \tag{41}$$

which converges to zero because singular values are positive. As before, we can show that the norm of vector $\mathbf{p}_i$ is maintained at 1, so $p_{i,2i-1}$ converges to 1. The proof for $\mathbf{q}_i$ goes along the same lines.