Journal of Mathematical Imaging and Vision (2023) 65:164–184
https://doi.org/10.1007/s10851-022-01132-9

Learning Linearized Assignment Flows for Image Labeling

Alexander Zeilmann[1] · Stefania Petra[2] · Christoph Schnörr[1]

Received: 1 August 2021 / Accepted: 12 May 2022 / Published online: 17 January 2023

Abstract
We introduce a novel algorithm for estimating optimal parameters of linearized assignment flows for image labeling. An exact formula is derived for the parameter gradient of any loss function that is constrained by the linear system of ODEs determining the linearized assignment flow. We show how to efficiently evaluate this formula using a Krylov subspace and a low-rank approximation. This enables us to perform parameter learning by Riemannian gradient descent in the parameter space, without the need to backpropagate errors or to solve an adjoint equation. Experiments demonstrate that our method performs as good as highly-tuned machine learning software using automatic differentiation. Unlike methods employing automatic differentiation, our approach yields a low-dimensional representation of internal parameters and their dynamics which helps to understand how assignment flows and more generally neural networks work and perform.

Keywords Assignment flows · Image labeling · Parameter learning · Exponential integration · Low-rank approximation

Mathematics Subject Classification 34C40 · 62H35 · 68U10 · 68T05 · 91A22

Contents

✉ Alexander Zeilmann
alexander.zeilmann@iwr.uni-heidelberg.de
https://ipa.math.uni-heidelberg.de

Stefania Petra
https://www.stpetra.com

1 Image and Pattern Analysis Group, Heidelberg University, Heidelberg, Germany

2 Mathematical Imaging Group, Heidelberg University, Heidelberg, Germany

1 Introduction

1.1 Overview, Motivation

Learning the parameters of large neural networks from training data constitutes a basic problem in imaging science, machine learning and other fields. The prevailing approach utilizes gradient descent or approximations thereof based on automatic differentiation [5] and corresponding software tools, like PyTorch [19] and TensorFlow [1]. This kind of software support has been spurring research in imaging science and machine learning dramatically. However, merely relying on numerical schemes and their automatic differentiation tends to thwart attempts to shed light on the often-criticized black-box behavior of deep networks and to better understand the internal representation and function of parameters and their adaptive dynamics.

In this paper, we explore a different route. Adopting the *linearized assignment flow* approach introduced by [27], we focus on a corresponding large system of linear ODEs of the form

$$\dot{V} = A(\Omega)V + B, \tag{1.1}$$

and study a geometric approach to learning the regularization parameters Ω by Riemannian gradient descent of a loss function

$$\Omega \mapsto \mathcal{L}(V(T;\Omega)) \tag{1.2}$$

constrained by the dynamical system (1.1). Here, we exploit the crucial property that the solution to (1.1) can be specified in closed form (2.24) and can be computed efficiently using exponential integration ([27] and Sect. 2.4). Matrix $V \in \mathbb{R}^{|I|\times c}$ represents a tangent vector of the so-called assignment manifold, $|I|$ is the number of nodes $i \in I$ of the underlying graph, and c is the number of labels (classes) that have to be assigned to data observed at nodes $i \in I$. Specifically,

- we derive a formula—see Theorem 3.8—for the Euclidean parameter gradient $\partial_\Omega \mathcal{L}(V(T;\Omega))$ in closed form;
- we show that a low-rank representation of this gradient can be used to efficiently and accurately approximate this closed-form gradient; neither backpropagation, nor automatic differentiation or solving adjoint equations are required;
- we highlight that the resulting parameter estimation algorithm, in terms of a Riemannian gradient descent iteration (3.7) on the parameter manifold, can be implemented without any specialized software support with modest computational resources;

The significance of our work reported in this paper arises in a broader context. The linearized assignment flow approach also comprises the equation

$$W(T) = \mathrm{Exp}_{\mathbb{1}_{\mathcal{W}}}(V(T)) \tag{1.3}$$

that yields the labeling in terms of almost integral assignment vectors $W_i \in \mathbb{R}^c_+,\ i \in I$ that form the rows of the matrix W, depending on the solution $V(t)$ of (1.1) for a sufficiently large time $t = T$. Both Eqs. (1.3) and (1.1) together constitute a linearization of the full nonlinear assignment flow [3]

$$\dot{W} = R_W S(W) \tag{1.4}$$

at the barycenter $\mathbb{1}_{\mathcal{W}}$ of the assignment manifold. Choosing an arbitrary sequence of time intervals (step sizes) $h_1, h_2, \ldots$ and setting

$$W^{(0)} = \mathbb{1}_{\mathcal{W}}, \qquad W^{(k)} = W(h_k), \qquad k \in \mathbb{N}, \tag{1.5}$$

a sequence of linearized assignment flows

$$W^{(k+1)} = \mathrm{Exp}_{\mathbb{1}_{\mathcal{W}}}(V^{(k)}), \tag{1.6a}$$

$$V^{(k+1)} = V^{(k)} + V\big(h_k; \Omega^{(k)}, W^{(k)}\big), \quad k = 0, 1, 2, \ldots \tag{1.6b}$$

can be computed in order to approximate (1.4) more closely, where $V\big(h_k; \Omega, W^{(k)}\big)$ solves the corresponding updated ODE (1.1) of the form

$$\dot{V} = A(\Omega^{(k)}; W^{(k)})V + \Pi_0 S(W^{(k)}). \tag{1.6c}$$

The time-discrete equations (1.6) reveal two basic ingredients of deep networks (or neural ODEs) which the full assignment flow (1.4) embodies in a continuous-time manner: coupling a pointwise nonlinearity (1.6a) and diffusion (1.6b), (1.6c) enhances the expressivity of network models for data analysis.

The key point motivating the work reported in this paper is that our results apply to learning the parameters Ω^k in each step of the iterative scheme (1.6). We expect that the gradient, and its low-dimensional subspace representations, will help the further study of how each ingredient of (1.6) impacts the predictive power of assignment flows. Furthermore, 'deep' extensions of (1.4) and (1.6) are equally feasible within the *same* mathematical framework (cf. Sect. 5.2).

1.2 Related Work

Assignment flows were introduced by [3]. For a survey of prior and recent related work, we refer to [23]. *Linearized* assignment flows were introduced by [27] as part of a comprehensive study of numerical schemes for the geometric integration of the assignment flow equation (1.4).

While the bulk of these schemes are based on a Lie group action (cf. [14]) on the assignment manifold, which enables to apply established theory and algorithms for the numerical integration of ODEs that evolve in an Euclidean space [11], the linearity of the ODE (1.1) specifically allows to represent its solution in *closed form* by the Duhamel (or variation-of-constants) formula [24]. Corresponding extensions to *nonlinear* ODEs rely on exponential integration [12,13]. Iteration (1.6) combines a corresponding iterative scheme and the tangent-space based parametrization (1.3) of the linearized assignment flow.

A key computational step of the latter class of methods requires to evaluate an analytical matrix-valued function, like the matrix exponential and similar functions [8, Section 10]. While basic methods [17] only work for problem of small and medium size, dedicated methods using Krylov subspaces [2,10] and established numerical linear algebra [20,21] can be applied to larger problems. The algorithm that results from our approach employs such methods.

Machine learning requires to compute gradients of loss functions that take solutions of ODEs as argument. This defines an enormous computational task and explains why automatic differentiation and corresponding software tools are almost exclusively applied. Alternative dedicated recent methods like [16] focus on a special problem structure, viz. the action of the differential of the matrix exponential on a rank-one matrix. Our closed-form formula for the parameter gradient also involves the differential of a matrix exponential. Yet, we wish to evaluate the gradient itself rather than its action on another matrix. The special problem structure that we can exploit is the Kronecker sum of matrices. Accordingly, our approach is based on the recent corresponding work [6] and an additional subsequent low-rank approximation.

1.3 Contribution, Organization

We derive a closed-form expression of the gradient of any C^1 loss function of the form (1.2) that depends on the solution $V(t)$ of the linear system of ODEs (1.1) at some arbitrary but fixed time $t = T$. In addition, we develop a numerical method that enables to evaluate the gradient efficiently for the common large sizes of image labeling problems. We apply the method to optimal parameter estimation by Riemannian gradient descent and validate our approach by a series of proof-of-concept experiments. This includes a comparison with automatic differentiation applied to two numerical schemes for integrating the linearized assignment flow: geometric explicit Euler and exponential integration. It turns out that our method is as accurate and efficient as the highly optimized automatic differentiation software, like PyTorch [19] and TensorFlow [1]. We point out that to our knowledge, automatic differentiation has not been applied to exponential integration, so far.

This paper extends the conference paper [26] in that all parameter dependencies of the loss function, constrained by the linearized assignment flow, are taken into account (cf. diagram (3.15)). In addition, a complete proof of the corresponding main result (Theorem 3.8) is provided. The space complexity of various gradient approximations are specified in a series of Remarks. The approach is validated numerically and more comprehensively by comparing to automatic differentiation and by examining the influence of all parameters.

The plan for this paper is as follows. Section 2 summarizes the assignment flow approach, the linearized assignment flow and exponential integration for integrating the latter flow. Section 3 details the derivation of the exact gradient of any loss function of the flow with respect to the weight parameters that regularize the flow. Furthermore, a low-rank approximation of the gradient is developed for evaluating the gradient efficiently. We also sketch how automatic derivation is applied to two numerical schemes in order to solve the parameter estimation problem in alternative ways. Numerical experiments are reported in Sect. 4 for comparing the methods and for inspecting quantitatively the gradient approximation and properties of the estimated weight patches that parametrize the linearized assignment flow. We conclude in Sect. 5 and point out further directions of research.

2 Preliminaries

2.1 Basic Notation

We set $[n] = \{1, 2, \dots, n\}$ for $n \in \mathbb{N}$. The cardinality of a finite set S is denoted by $|S|$, e.g., $|[n]| = n$. $\mathbb{R}^n_+$ denotes the positive orthant and $\mathbb{R}^n_>$ its interior. $\mathbb{1} = (1, 1, \dots, 1)^\top$ has dimension depending on the context that we specify sometimes by a subscript, e.g., $\mathbb{1}_n \in \mathbb{R}^n$. Similarly, we set $0_n = (0, 0, \dots, 0)^\top \in \mathbb{R}^n$. $\{e_i : i \in [n]\}$ is the canonical basis of $\mathbb{R}^n$ and $I_n = (e_1, \dots, e_n) \in \mathbb{R}^{n\times n}$ the identity matrix.

The support of a vector $x \in \mathbb{R}^n$ is denoted by $\operatorname{supp}(x) = \{i \in [n] : x_i \neq 0\}$. $\Delta_n = \{p \in \mathbb{R}^n_+ : \langle \mathbb{1}_n, p\rangle = 1\}$ is the probability simplex whose points represent discrete distributions on $[n]$. Distributions with full support $[n]$ form the relative interior $\mathring{\Delta}_n = \Delta_n \cap \mathbb{R}^n_>$. $\langle\cdot,\cdot\rangle$ is the Euclidean inner product of vectors and matrices. In the latter case, this reads $\langle A, B\rangle = \operatorname{tr}(A^\top B)$ with the trace $\operatorname{tr}(A) = \sum_i A_{ii}$. The induced Frobenius norm is denoted by $\|A\| = \sqrt{\langle A, A\rangle}$, and other matrix norms like the spectral norm $\|A\|_2$ are indicated by subscripts. The mapping $\operatorname{Diag}\colon \mathbb{R}^n \to \mathbb{R}^{n\times n}$ sends a vector x to the diagonal matrix $\operatorname{Diag}(x)$ with entries x. $A \otimes B$ denotes the Kronecker *product* of matrices A and B [7,25] and $\oplus$ the Kronecker *sum*

$$A \oplus B = A \otimes I_n + I_m \otimes B \in \mathbb{R}^{mn\times mn}, \qquad A \in \mathbb{R}^{m\times m}, \quad B \in \mathbb{R}^{n\times n}. \tag{2.1}$$

We have

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) \tag{2.2}$$

for matrices of compatible dimensions. The operator vec_r turns a matrix into the vector by stacking the row vectors. It satisfies

$$\operatorname{vec}_r(ABC) = (A \otimes C^\top)\operatorname{vec}_r(B). \tag{2.3}$$

The Kronecker product $v \otimes w \in \mathbb{R}^{mn}$ of two vectors $v \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$ is defined by viewing the vectors as matrices with only one column and applying the definition of Kronecker products for matrices. We have

$$v \otimes w = \mathrm{vec}_r(vw^\top). \tag{2.4}$$

The matrix exponential of a square matrix A is given by [8, Ch. 10]

$$\mathrm{expm}(A) = \sum_{k\geq 0} \frac{A^k}{k!}. \tag{2.5}$$

$L(\mathcal{E}_1, \mathcal{E}_2)$ denotes the space of all linear bounded mappings from $\mathcal{E}_1$ to $\mathcal{E}_2$.

2.2 Assignment Flow

Let $G = (I, E)$ be a given undirected graph with vertices $i \in I$ indexing data

$$\mathcal{F}_I = \{f_i : i \in I\} \subset \mathcal{F} \tag{2.6}$$

given in a metric space $(\mathcal{F}, d)$. In this paper, we focus primarily on the application of image labeling in which the graph G is a grid graph equipped with a 3×3 or larger neighborhood $\mathcal{N}_i = \{k \in I : ik = ki \in E\} \cup \{i\}$ at each pixel $i \in I$. The linearized assignment flow and the learning approach in this paper can, however, also be applied to the case of data labeling on arbitrary graphs.

Along with $\mathcal{F}_I$, *prototypical data (labels)* $\mathcal{L}_J = \{l_j \in \mathcal{F} : j \in J\}$ are given that represent classes $j = 1, \dots, |J|$. *Supervised image labeling* denotes the task to assign precisely one prototype l_j to each datum f_i at every vertex i in a coherent way, depending on the label assignments in the neighborhoods $\mathcal{N}_i$. These assignments at i are represented by probability vectors

$$W_i \in \mathring{\Delta}_{|J|}, \quad i \in I. \tag{2.7}$$

The set $\mathring{\Delta}_{|J|}$ becomes a Riemannian manifold denoted by $\mathcal{S} := (\mathring{\Delta}_{|J|}, g_{FR})$ when endowed with the Fisher–Rao metric g_{FR}. Collecting all assignment vectors as *rows* defines the strictly positive row-stochastic *assignment matrix*

$$W = (W_1, \dots, W_{|I|})^\top \in \mathcal{W} = \mathcal{S} \times \cdots \times \mathcal{S} \subset \mathbb{R}^{|I|\times|J|}, \tag{2.8}$$

that we regard as point on the product *assignment manifold* $\mathcal{W}$. Image labeling is accomplished by geometrically integrating the *assignment flow* $W(t)$ solving

$$\dot{W} = R_W\big(S(W)\big), \qquad W(0) = \mathbb{1}_{\mathcal{W}} := \frac{1}{|J|}\mathbb{1}_{|I|}\mathbb{1}_{|J|}^\top \quad \text{(barycenter)}, \tag{2.9}$$

where R_W and $S(W)$ are defined in (2.11b) resp. (2.17). The assignment flow provably converges toward a binary matrix [28], i.e., $\lim_{t\to\infty} W_i(t) = e_{j(i)}$, for every $i \in I$ and some $j(i) \in J$, which yields the label assignment $f_i \mapsto l_{j(i)}$. In practice, geometric integration is terminated when $W(t)$ is ε-close to an integral point using the entropy criterion from [3], followed by trivial rounding, due to the existence of basins of attraction around each integral point [28].

We specify the right-hand side of the differential equation in (2.9)—see (2.14) and (2.17) below—and refer to [3,23] for more details and the background. With the tangent space

$$T_0 = T_p\mathcal{S} = \{v \in \mathbb{R}^{|J|} : \langle \mathbb{1}, v\rangle = 0\}, \qquad \forall p \in \mathcal{S}, \tag{2.10}$$

that does not depend on the base point $p \in \mathcal{S}$, we define

$$\Pi_0 : \mathbb{R}^{|J|} \to T_0, \quad z \mapsto I_{|J|} - \frac{1}{|J|}\mathbb{1}_{|J|}\mathbb{1}_{|J|}^\top, \tag{2.11a}$$

$$R_p : \mathbb{R}^{|J|} \to T_0, \quad z \mapsto R_p(z) = \big(\mathrm{Diag}(p) - pp^\top\big)z, \tag{2.11b}$$

$$\mathrm{Exp} : \mathcal{S} \times T_0 \to \mathcal{S}, \quad (p, v) \mapsto \mathrm{Exp}_p(v) = \frac{e^{\frac{v}{p}}}{\langle p, e^{\frac{v}{p}}\rangle}p, \tag{2.11c}$$

$$\mathrm{Exp}^{-1} : \mathcal{S} \times \mathcal{S} \to T_0, \quad (p, q) \mapsto \mathrm{Exp}_p^{-1}(q) = R_p \log\frac{q}{p}, \tag{2.11d}$$

$$\mathrm{exp} : \mathcal{S} \times \mathbb{R}^{|J|} \to \mathcal{S}, (p, z) \mapsto \mathrm{exp}_p(z) = \mathrm{Exp}_p \circ R_p(z) = \frac{pe^z}{\langle p, e^z\rangle}, \tag{2.11e}$$

where multiplication, division, exponentiation $e^{(\cdot)}$ and $\log(\cdot)$ apply *component-wise* to vectors. Corresponding maps

$$R_W, \qquad \mathrm{Exp}_W, \qquad \mathrm{exp}_W \tag{2.12}$$

in connection with the product manifold (2.8) are defined analogously, and likewise the tangent space

$$\mathcal{T}_0 = T_0 \times \cdots \times T_0 = T_W\mathcal{W}, \qquad \forall W \in \mathcal{W} \tag{2.13}$$

and the extension of the orthogonal projection (2.11a) onto $\mathcal{T}_0$, again denoted by Π_0. For example, regarding (2.9), with $W \in \mathcal{W}$ and $S(W) \in \mathcal{W}$ (or more generally $S \in \mathbb{R}^{|I|\times|J|}$), we have

$$\begin{aligned} R_W S(W) &= \big(R_{W_1}S_1(W), \dots, R_{W_{|I|}}S_{|I|}(W)\big)^\top \\ &= \mathrm{vec}_r^{-1}\Big(\mathrm{Diag}(R_W)\,\mathrm{vec}_r\big(S(W)\big)\Big) \end{aligned} \tag{2.14a}$$

with

$$\mathrm{Diag}(R_W) := \begin{pmatrix} R_{W_1} & 0 & \cdots & 0 \\ 0 & R_{W_2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & & R_{W_{|I|}} \end{pmatrix}. \tag{2.14b}$$

Given data $\mathcal{F}_I$ are taken into account as distance vectors

$$D_i = \big(d(f_i, l_1), \dots, d(f_i, l_{|J|})\big)^\top, \quad i \in I \tag{2.15}$$

and mapped to $\mathcal{W}$ by

$$\begin{aligned} L(W) &= \exp_W(-\tfrac{1}{\rho}D) \in \mathcal{W}, \\ L_i(W_i) &= \exp_{W_i}(-\tfrac{1}{\rho}D_i) \\ &= \frac{W_i e^{-\frac{1}{\rho}D_i}}{\langle W_i, e^{-\frac{1}{\rho}D_i}\rangle}, \end{aligned} \tag{2.16}$$

where $\rho > 0$ is a user parameter for normalizing the scale of the data. These *likelihood vectors* represent data terms in conventional variational approaches: Each individual flow $\dot{W}_i = R_{W_i} L_i(W_i)$, $W_i(0) = \mathbb{1}_{\mathcal{S}}$ converges to $e_{j(i)}$ with $j(i) = \arg\min_{j\in J} D_{ij}$ and in this sense maximizes the local data likelihood.

The vector field defining the assignment flow (2.9) arises through *coupling* flows for individual pixels through *geometric averaging* within the neighborhoods $\mathcal{N}_i$, $i \in I$, conforming to the underlying Fisher–Rao geometry

$$S(W) = \begin{pmatrix} \vdots \\ S_i(W)^\top \\ \vdots \end{pmatrix} = \mathcal{G}^\Omega\big(L(W)\big) \in \mathcal{W}, \tag{2.17a}$$

$$\begin{aligned} S_i(W) &= \mathcal{G}_i^\Omega\big(L(W)\big) \\ &= \mathrm{Exp}_{W_i}\left(\sum_{k\in\mathcal{N}_i} \omega_{ik}\, \mathrm{Exp}_{W_i}^{-1}\big(L_k(W_k)\big)\right), \quad i \in I. \end{aligned} \tag{2.17b}$$

The *similarity vectors* $S_i(W)$ are parametrized by strictly positive *weight patches* $(\omega_{ik})_{k\in\mathcal{N}_i}$, centered at $i \in I$ and indexed by local neighborhoods $\mathcal{N}_i \subset I$, that in turn define the *weight parameter matrix*

$$\begin{aligned} \Omega &= (\Omega_i)_{i\in I} \in \mathbb{R}_+^{|I|\times|I|}, \\ \Omega_i|_{\mathcal{N}_i} &= (\omega_{ik})_{k\in\mathcal{N}_i} \in \mathring{\Delta}_{|\mathcal{N}_i|}, \\ \sum_{k\in\mathcal{N}_i} \omega_{ik} &= 1, \ \forall i \in I. \end{aligned} \tag{2.18}$$

The matrix Ω comprises all *regularization parameters* satisfying the latter linear constraints. Flattening these weight patches defines row vectors $\Omega_i|_{\mathcal{N}_i}$, $i \in I$ and, by complementing with 0, entries of the *sparse* row vectors Ω_i of the matrix Ω. Note that the positivity assumption $\omega_{ik} > 0$ is reflected by the membership $\Omega_i|_{\mathcal{N}_i} \in \mathring{\Delta}_{|\mathcal{N}_i|}$. Throughout this paper, we assume that all pixels have neighborhoods of equal size

$$|\mathcal{N}| := |\mathcal{N}_i|, \quad \forall i \in I \tag{2.19}$$

and therefore simply write $\Omega_i|_{\mathcal{N}} = \Omega_i|_{\mathcal{N}_i}$. These parameters are used in the linearized assignment flow, to be introduced next. We explain a corresponding parameter estimation approach in Sect. 3 and a parameter predictor in Sect. 4.4.

2.3 Linearized Assignment Flow

The *linearized assignment flow*, introduced by [27], approximates (2.9) by

$$\dot{W} = R_W\left(S(W_0) + dS_{W_0} R_{W_0} \log\frac{W}{W_0}\right), \ W(0) = W_0 \in \mathcal{W} \tag{2.20}$$

around any point W_0. In what follows, we only consider the barycenter

$$W_0 = \mathbb{1}_{\mathcal{W}} \tag{2.21}$$

which is the initial point of (2.9). The differential equation (2.20) is still *nonlinear* but can be parametrized by a *linear* ODE on the tangent space

$$W(t) = \mathrm{Exp}_{W_0}\big(V(t)\big), \tag{2.22a}$$

$$\begin{aligned} \dot{V} &= R_{W_0}\big(S(W_0) + dS_{W_0}V\big) =: B_{W_0} + A(\Omega)V, \\ V(0) &= 0, \end{aligned} \tag{2.22b}$$

where matrix $A(\Omega)$ linearly depends on the parameters Ω of (2.17). The action of $A(\Omega)$ on V is explicitly given by [27, Prop. 4.4]

$$\begin{aligned} A(\Omega)V &= -R_{W_0} dS_{W_0} V = R_{S(W_0)}\Omega V \\ &= -\operatorname{vec}_r^{-1}\big(\mathrm{Diag}(R_{S(W_0)})\operatorname{vec}_r(\Omega V)\big) \end{aligned} \tag{2.23a}$$

$$= \left(R_{S_1(W_0)}\sum_{k\in\mathcal{N}_1}\omega_{1k}V_k, \dots, R_{S_{|I|}(W_0)}\sum_{k\in\mathcal{N}_{|I|}}\omega_{|I|k}V_k\right)^\top, \tag{2.23b}$$

where $\mathrm{Diag}(R_{S(W_0)})$ is defined by (2.14b) and we took into account (2.21). The linear ODE (2.22b) admits a closed-form solution which in turn enables a different numerical approach (Sect. 2.4) and a novel approach to parameter learning (Sect. 3).

2.4 Exponential Integration

The solution to (2.22b) is given by a high-dimensional integral (Duhamel's formula) whose value in closed form is given by

$$V(t;\Omega) = t\varphi\big(tA(\Omega)\big)B_{W_0}, \varphi(x) = \frac{e^x - 1}{x} = \sum_{k=0}^{\infty}\frac{x^k}{(k+1)!}, \tag{2.24}$$

where the entire function φ is extended to matrix arguments as the limit of an absolutely convergent power series in the matrix space [9, Theorem 6.2.8]. As the matrix A is already very large even for medium-sized images, however, it is not feasible in practice to compute $\varphi(tA)$ in this way. Exponential integration [10,18], therefore, was used by [27] for approximately evaluating (2.24), as sketched next.

Applying the row-stacking operator (2.3) to both sides of (2.22b) and (2.24), respectively, yields with

$$v = \mathrm{vec}_r(V) \tag{2.25}$$

the ODE (2.22b) in the form

$$\dot{v} = b + A^J(\Omega)v,\ v(0) = 0, \quad b = b(\Omega) = \mathrm{vec}_r(B_{W_0}) \in \mathbb{R}^n, \tag{2.26a}$$

$$\begin{aligned} A^J(\Omega) &= \big(A^J_{ik}(\Omega)\big)_{i,k\in I} \in \mathbb{R}^{n\times n}, \\ A^J_{ik}(\Omega) &= \begin{cases} \omega_{ik} R_{S_i(W_0)}, & k \in \mathcal{N}_i, \\ 0, & k \notin \mathcal{N}_i. \end{cases} \end{aligned} \tag{2.26b}$$

$$\begin{aligned} v(t;\Omega) &= t\varphi\big(tA^J(\Omega)\big)b, \\ n &:= \dim v(t;\Omega) = |I||J|, \end{aligned} \tag{2.26c}$$

where $A^J(\Omega)$ results from

$$\begin{aligned} \mathrm{vec}_r\big(A(\Omega)V\big) &\overset{(2.23)}{=} \mathrm{Diag}(R_{S(W_0)})\,\mathrm{vec}_r(\Omega V) \\ &= \mathrm{Diag}(R_{S(W_0)})(\Omega \otimes I_{|J|})v & (2.27a) \\ &= A^J(\Omega)v. & (2.27b) \end{aligned}$$

Using the Arnoldi iteration [21] with initial vector $q_1 = b/\|b\|$, we determine an orthonormal basis $Q_m = (q_1,\dots,q_m) \in \mathbb{R}^{n\times m}$ of the Krylov space $\mathcal{K}_m(A^J, b)$ of dimension m. As will be validated in Sect. 4, choosing $m \le 10$ yields sufficiently accurate approximations of the actions of the matrix exponential expm and the φ operator on a vector, respectively, that are given by

$$\begin{aligned} \mathrm{expm}\big(tA^J(\Omega)\big)b &\approx \|b\|Q_m\,\mathrm{expm}(tH_m)e_1, \\ H_m &= -Q_m^\top A^J(\Omega)Q_m, & (2.28a) \\ t\varphi\big(tA^J(\Omega)\big)b &\approx t\|b\|Q_m\varphi(tH_m)e_1. & (2.28b) \end{aligned}$$

The expression $\varphi(tH_m)e_1$ results from computing the left-hand side of the relation [8, Section 10.7.4]

$$\mathrm{expm}\begin{pmatrix} tH_m & e_1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \mathrm{expm}(tH_m) & \varphi(tH_m)e_1 \\ 0 & 1 \end{pmatrix} \tag{2.29}$$

and extracting the upper-right vector. Since H_m is a small matrix, any standard method [17] can be used for computing the matrix exponential on the left-hand side.

3 Parameter Estimation

Section 3.1 details our approach for learning optimal weight parameters for a given image and ground truth labeling: Riemannian gradient descent is performed with respect to a loss function that depends on the solution of the linearized assignment flow. A closed-form expression of this gradient is derived in Sect. 3.2 along with a low-rank approximation in Sect. 3.3 that can be computed efficiently. As an alternative and baseline, we outline in Sect. 3.4 two gradient approximations based on numerical schemes for integrating the linearized assignment flow and automatic differentiation.

3.1 Learning Procedure

Let

$$\mathcal{P}_\Omega = \{\Omega \in \mathbb{R}_+^{|I|\times|I|} : \Omega \text{ satisfies } (2.18)\} \tag{3.1}$$

denote the space of weight parameter matrices that parametrize the similarity mapping (2.17). Due to (2.18) and (2.19), the restrictions $\Omega_i|_{\mathcal{N}}$ are strictly positive probability vectors, as are the assignment vectors W_i defined by (2.7). Therefore, similar to $W_i \in \mathcal{S}$, we consider each $\Omega_i|_{\mathcal{N}}$ as point on a corresponding manifold $(\Delta_{|\mathcal{N}|}, g_{FR})$ equipped with the Fisher–Rao metric and—in this sense—regard $\mathcal{P}_\Omega$ in (3.1) as corresponding product manifold.

Let $W^* \in \mathcal{W}$ denote the ground truth labeling for a given image, and let $V^* = \Pi_0 W^* \in \mathcal{T}_0$ be a tangent vector such that $\lim_{s\to\infty} \mathrm{Exp}_{\mathbb{1}_\mathcal{W}}(sV^*) = W^*$. Our objective is to determine Ω such that, for some specified time $T > 0$, the vector

$$V_T(\Omega) := V(T;\Omega), \tag{3.2}$$

given by (2.24) and corresponding to the linearized assignment flow, approximates the *direction* of V^* and hence

$$\lim_{s\to\infty} \mathrm{Exp}_{\mathbb{1}_\mathcal{W}}\big(sV_T(\Omega)\big) = W^*. \tag{3.3}$$

In this formula the direction of the vector $V_T(\Omega)$ only is relevant, but not its magnitude. A distance function that also satisfies these properties is given by

$$f_\mathcal{L}\colon \mathcal{T}_0 \to \mathbb{R}, \qquad V \mapsto 1 - \frac{\langle V^*, V\rangle}{\|V^*\|\|V\|}. \tag{3.4}$$

In addition, we consider a regularizer

$$\begin{aligned} \mathcal{R}\colon \mathcal{P}_\Omega \to \mathbb{R}, \quad \Omega &\mapsto \frac{\tau}{2}\sum_{i\in I}\|t_i(\Omega)\|^2, \\ t_i(\Omega) &= \exp^{-1}_{\mathbb{1}_\Omega}(\Omega_i|_\mathcal{N}), \qquad \tau > 0 \end{aligned} \tag{3.5}$$

and define the loss function

$$\mathcal{L}\colon P_\Omega \to \mathbb{R}, \qquad \mathcal{L}(\Omega) = f_{\mathcal{L}}\big(V_T(\Omega)\big) + \mathcal{R}(\Omega), \tag{3.6}$$

with $V_T(\Omega)$ from (3.2). Ω is determined by the Riemannian gradient descent sequence

$$\begin{aligned} \Omega^{(k+1)} &= \exp_{\Omega^{(k)}}\big(-h\nabla\mathcal{L}(\Omega^{(k)})\big), \quad k \geq 0, \\ \Omega_i^{(0)}|_{\mathcal{N}} &= \mathbb{1}_{|\mathcal{N}|}, \quad i \in I \end{aligned} \tag{3.7}$$

with step size $h > 0$. Here

$$\nabla\mathcal{L}(\Omega) = R_\Omega \partial\mathcal{L}(\Omega) \tag{3.8}$$

is the Riemannian gradient with respect to the Fisher–Rao metric. R_Ω is given by (2.12) and (2.11b) and effectively applies to the restrictions $\Omega_i|_{\mathcal{N}}$ of the row vectors with all remaining components equal to 0. It remains to compute the Euclidean gradient $\partial\mathcal{L}(\Omega)$ of the loss function (3.6) which is presented in the subsequent Sect. 3.2.

3.2 Loss Function Gradient

In Sect. 3.2.2, we derive a closed-form expression for the loss function gradient (Theorem 3.8), after introducing some basic calculus rules for representing and computing differentials of matrix-valued mappings in Sect. 3.2.1.

3.2.1 Matrix Differentials

Let $F\colon \mathbb{R}^{m_1\times m_2} \to \mathbb{R}^{n_1\times n_2}$ be a smooth mapping. Using the canonical identification $T\mathcal{E} \cong \mathcal{E}$ of the tangent spaces of any Euclidean space $\mathcal{E}$ with $\mathcal{E}$ itself, we both represent and compute the differential

$$dF\colon \mathbb{R}^{m_1\times m_2} \to L(\mathbb{R}^{m_1\times m_2}, \mathbb{R}^{n_1\times n_2}) \tag{3.9}$$

in terms of a vector-valued mapping f, which is defined by F according to the commutative diagram

$$\begin{tikzcd} \mathbb{R}^{m_1\times m_2} \arrow[rr, "F"] \arrow[dr, "dF"] \arrow[dd, "\mathrm{vec}_r"'] & & \mathbb{R}^{n_1\times n_2} \arrow[dd, "\mathrm{vec}_r"] \\ & L(\mathbb{R}^{m_1\times m_2}, \mathbb{R}^{n_1\times n_2}) \cong \mathbb{R}^{n_1 n_2\times m_1 m_2} & \\ \mathbb{R}^{m_1 m_2} \arrow[rr, "f"] \arrow[ur, "df"] & & \mathbb{R}^{n_1 n_2} \end{tikzcd} \tag{3.10}$$

In formulas, this means that based on the equation

$$\mathrm{vec}_r\big(F(X)\big) = f\big(\mathrm{vec}_r(X)\big), \quad \forall X \in \mathbb{R}^{m_1\times m_2}, \tag{3.11}$$

we set

$$\begin{aligned} &\mathrm{vec}_r\big(dF(X)Y\big) = df\big(\mathrm{vec}_r(X)\big)\,\mathrm{vec}_r(Y), \\ &\quad \forall X, Y \in \mathbb{R}^{m_1\times m_2} \end{aligned} \tag{3.12}$$

and hence *define* and compute the differential (3.9) as matrix-valued mapping

$$dF := df \circ \mathrm{vec}_r\,. \tag{3.13}$$

The corresponding linear actions on $Y \in \mathbb{R}^{m_1\times m_2}$ and $\mathrm{vec}_r(Y) \in \mathbb{R}^{m_1 m_2}$, respectively, are given by (3.12). We state an auxiliary result required in the next subsection, which also provides a first concrete instance of the general relation (3.12).

Lemma 3.1 (differential of the matrix exponential) *If* $F = \mathrm{expm}\colon \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$*, then* (3.12) *reads*

$$\begin{aligned} \mathrm{vec}_r\big(d\,\mathrm{expm}(X)Y\big) &= \big(\mathrm{expm}(X) \otimes I_n\big)\varphi(-X \oplus X^\top) \\ &= \mathrm{vec}_r(Y),\ Y \in \mathbb{R}^{n\times n}, \end{aligned} \tag{3.14}$$

with φ *given by* (2.24).

Proof The result follows from [8, Thm. 10.13] where columnwise vectorization is used, after rearranging so as to conform to the row-stacking mapping vec_r used in this paper. □

3.2.2 Closed-Form Gradient Expression

We separate the computation of $\mathcal{L}(\Omega)$ and the gradient $\partial\mathcal{L}(\Omega)$ into several operations that were introduced in Sects. 2 and 3.1. We illustrate their composition and accordingly the process from parameters Ω to a loss $\mathcal{L}(\Omega)$ in the following flow diagram that refers to quantities in (2.26) and (2.27) related to the linearized assignment flow, after vectorization.

$$\begin{tikzcd}
\Omega \arrow[r, "(M1)"] \arrow[dr, "(M3)"] \arrow[ddr, bend right, "(M5)"'] & S(W_0) = \exp_{\mathbb{1}_{\mathcal{W}}}\left(-\frac{1}{\rho}\Omega D\right) \arrow[r, "(M2)"] \arrow[d] & b(\Omega) = \mathrm{vec}_r(R_{W_0} S(W_0)) \arrow[d, "(M4)"] \\
& A^J(\Omega) = \mathrm{Diag}(R_{S(W_0)})(\Omega \otimes I_{|J|}) \arrow[r, "(M4)"] & v_T(\Omega) = T\varphi\left(TA^J(\Omega)\right) b(\Omega) \arrow[d] \\
& \mathcal{R}(\Omega) \arrow[r] & \mathcal{L}(\Omega) = f_{\mathcal{L}}(v_T(\Omega)) + \mathcal{R}(\Omega)
\end{tikzcd} \tag{3.15}$$

In what follows, we traverse this diagram from top-left to bottom-right and collect each partial result by a corresponding lemma or proposition. Theorem 3.8 assembles all results and provides a closed-form expression of the loss function gradient $\partial\mathcal{L}(\Omega)$. To enhance readability, the proofs of most lemmata are listed in Appendix A.1.

We focus on mapping (M1) in diagram (3.15).

Lemma 3.2 *The differential of the function*

$$\begin{aligned} &f_1\colon \mathbb{R}^{|I|\times|I|} \to \mathbb{R}^{|I|\times|J|}, \\ &\quad \Omega \mapsto f_1(\Omega) := S(W_0) = \exp_{\mathbb{1}_{\mathcal{W}}}\Big(-\frac{1}{\rho}\Omega D\Big), \\ &\quad D \in \mathbb{R}^{|I|\times|J|} \end{aligned} \tag{3.16}$$

and its transpose are given by

$$df_1(\Omega)Y = -\frac{1}{\rho} R_{f_1(\Omega)}(YD), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|}, \tag{3.17a}$$

$$df_1(\Omega)^\top Z = -\frac{1}{\rho} R_{f_1(\Omega)}(Z)D^\top, \quad \forall Z \in \mathbb{R}^{|I|\times|J|}, \tag{3.17b}$$

with $R_{f_1(\Omega)}$ defined by (2.14).

Proof see Appendix A.1.

We consider mapping (M2) of diagram (3.15), taking into account mapping (M4) and notation (3.16).

Lemma 3.3 *The differential of the function*

$$\begin{aligned} &f_2\colon \mathbb{R}^{|I|\times|I|} \to \mathbb{R}^{|I|^2}, \\ &\Omega \mapsto f_2(\Omega) := b(\Omega) = \mathrm{vec}_r\left(R_{W_0} f_1(\Omega)\right) \end{aligned} \tag{3.18}$$

and its transpose are given by

$$df_2(\Omega)Y = \mathrm{vec}_r\left(R_{W_0} df_1(\Omega)Y\right), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|} \tag{3.19a}$$

$$df_2(\Omega)^\top Z = df_1(\Omega)^\top (R_{W_0} Z), \qquad \forall Z \in \mathbb{R}^{|I|\times|I|}. \tag{3.19b}$$

Proof see Appendix A.1.

We note that $df_2(\Omega)^\top$ should act on a vector $\mathrm{vec}_r(Z) \in \mathbb{R}^{|I|^2}$. We prefer the more compact and equivalent non-vectorized expression (3.19b).

We turn to mapping (M3) of diagram (3.15) and use (3.15).

Lemma 3.4 *The differential of the mapping*

$$\begin{aligned} &f_3\colon \mathbb{R}^{|I|\times|I|} \to \mathbb{R}^{n\times n}, \\ &\quad \Omega \mapsto f_3(\Omega) := A^J(\Omega) = \mathrm{Diag}(R_{f_1(\Omega)})(\Omega \otimes I_{|J|}), n = |I||J| \end{aligned} \tag{3.20}$$

is given by

$$\begin{aligned} df_3(\Omega)Y = {} & \mathrm{Diag}(dR_{f_1(\Omega)}Y)(\Omega \otimes I_{|J|}) \\ & + \mathrm{Diag}(R_{f_1(\Omega)})(Y \otimes I_{|J|}), \\ & \forall Y \in \mathbb{R}^{|I|\times|I|}. \end{aligned} \tag{3.21a}$$

Here, $\mathrm{Diag}(dR_{f_1(\Omega)}Y) \in \mathbb{R}^{n\times n}$ *is defined by* (2.14b) *and* $|I|$ *block matrices of size* $|J| \times |J|$ *on the diagonal of the form*

$$\begin{aligned} dR_{f_{1i}(\Omega)}Y = {} & \mathrm{Diag}\left(df_{1i}(\Omega)Y\right) - \left(df_{1i}(\Omega)Y\right) f_{1i}(\Omega)^\top \\ & - f_{1i}(\Omega)\left(df_{1i}(\Omega)Y\right)^\top, \quad i \in I, \end{aligned} \tag{3.21b}$$

where $df_{1i}(\Omega)Y$ *is given by*

$$(dR_{f_{1i}(\Omega)}Y)S_i = \left((dR_{f_1(\Omega)}Y)S\right)_i, \quad i \in I \tag{3.21c}$$

for any $S = (\dots, S_i, \dots)^\top \in \mathbb{R}^{|I|\times|J|}$ *and by* (3.17a).

Proof see Appendix A.1.

We focus on the differential of the vector-valued mapping $v_T(\Omega) \in \mathbb{R}^n$ of (3.15) with n given by (2.26c). We utilize the fact that analogous to (2.29), the vector

$$v_T(\Omega) = T\varphi(TA^J(\Omega))b(\Omega) = (I_n, 0_n)\,\mathrm{expm}\left(\mathcal{A}(\Omega)\right)e_{n+1} \tag{3.22a}$$

can be extracted from the last column of the matrix

$$\mathrm{expm}\left(\mathcal{A}(\Omega)\right) = \begin{pmatrix} \mathrm{expm}\left(TA^J(\Omega)\right) & v_T(\Omega) \\ 0_n^\top & 1 \end{pmatrix},$$

$$\mathcal{A}(\Omega) = \begin{pmatrix} TA^J(\Omega) & Tb(\Omega) \\ 0_n^\top & 0 \end{pmatrix}. \tag{3.22b}$$

By means of relation (3.11), we associate a vector-valued function $f_{\mathcal{A}}$ with the matrix-valued mapping $\mathcal{A}$ through

$$\operatorname{vec}_r\big(\mathcal{A}(\Omega)\big) = f_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big) \tag{3.23}$$

and record for later that, for any matrix $Y \in \mathbb{R}^{|I|\times|I|}$, Eq. (3.12) implies

$$\operatorname{vec}_r\big(d\mathcal{A}(\Omega)Y\big) = df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)\operatorname{vec}_r(Y). \tag{3.24}$$

Lemma 3.5 *The differential of the mapping $\mathcal{A}$ in (3.22b) is given by*

$$d\mathcal{A}(\Omega)Y = T\begin{pmatrix} df_3(\Omega) & df_2(\Omega) \\ 0_n^\top & 0 \end{pmatrix}\left(\begin{pmatrix}1\\1\end{pmatrix}\otimes Y\right), \quad \forall Y \in \mathbb{R}^{|I|\times|I|}. \tag{3.25}$$

Proof Equation (3.25) is immediate due to

$$d\mathcal{A}(\Omega) = \begin{pmatrix} T\,dA^J(\Omega)Y & T\,db(\Omega)Y \\ 0_n^\top & 0 \end{pmatrix} \tag{3.26}$$

and Lemmata 3.3 and 3.4. □

Now we are in the position to specify the differential of the solution to the linearized assignment flow with respect to the regularizing weight parameters.

Proposition 3.6 *Let*

$$f_4(\Omega) := v_T(\Omega) := v(T;\Omega) \tag{3.27}$$

denote the solution (2.26c) in vectorized form to the ODE (2.22b). Then, the differential is given according to the convention (3.13) by

$$df_4(\Omega)Y = T\Big(d\big(\varphi(TA^J(\Omega))b(\Omega)\big) + \varphi\big(TA^J(\Omega)\big)df_2(\Omega)\Big)Y \tag{3.28a}$$

where

$$d\big(\varphi(TA^J(\Omega))b(\Omega)\big)Y \tag{3.28b}$$
$$= \Big(\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big)\otimes e_{n+1}^\top\Big)$$
$$\varphi\big(-\mathcal{A}(\Omega)\oplus\mathcal{A}(\Omega)^\top\big)\cdot df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)\operatorname{vec}_r(Y), \tag{3.28c}$$
$$\forall Y \in \mathbb{R}^{|I|\times|I|}, \tag{3.28d}$$

where $A^J(\Omega)$ is given by (2.26b), $\mathcal{A}(\Omega)$ by (3.22b), $df_{\mathcal{A}}$ by (3.24) and Lemma 3.5, and df_2 by Lemma 3.3.

Proof Equation (3.28a) follows directly from Eq. (2.26c) and Lemma 3.3 makes explicit the second summand on the right-hand side. It remains to compute the first summand. Using (3.22) and the chain rule, we have for any $Y \in \mathbb{R}^{|I|\times|I|}$,

$$d\big(T\varphi(TA^J(\Omega))b(\Omega)\big)Y = (I_n, 0_n)\,d\operatorname{expm}\big(\mathcal{A}(\Omega)\big)\big(d\mathcal{A}(\Omega)Y\big)e_{n+1}. \tag{3.29a}$$

Applying vec_r to both sides which does not change the vector on the left-hand side, yields by (2.3)

$$d\big(T\varphi(TA^J(\Omega))b(\Omega)\big)Y = \big((I_n, 0_n)\otimes e_{n+1}^\top\big)\operatorname{vec}_r\Big(d\operatorname{expm}\big(\mathcal{A}(\Omega)\big)\big(d\mathcal{A}(\Omega)Y\big)\Big). \tag{3.29b}$$

Applying Lemma 3.1 and (3.24), we obtain

$$d\big(T\varphi(TA^J(\Omega))b(\Omega)\big)Y = \big((I_n, 0_n)\otimes e_{n+1}^\top\big)$$
$$\big(\operatorname{expm}\big(\mathcal{A}(\Omega)\big)\otimes I_{n+1}\big)\varphi\big(-\mathcal{A}(\Omega)\oplus\mathcal{A}(\Omega)^\top\big) \tag{3.29c}$$
$$\cdot\, df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)\operatorname{vec}_r(Y) \tag{3.29d}$$

and using (2.2) and (3.22b)

$$= \Big(\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big)\otimes e_{n+1}^\top\Big)$$
$$\varphi\big(-\mathcal{A}(\Omega)\oplus\mathcal{A}(\Omega)^\top\big)\cdot df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)\operatorname{vec}_r(Y). \tag{3.29e}$$

□

We finally consider the regularizing mapping $\mathcal{R}(\Omega)$, defined by (3.5) and corresponding to mapping (M5) in diagram (3.15). Here, we have to take into account the constraints (2.18) imposed on Ω. Accordingly, we define the corresponding set of tangent matrices

$$\mathcal{Y}_\Omega = \big\{Y \in \mathbb{R}^{|I|\times|I|} \colon \langle \mathbb{1}_{\mathcal{N}}, Y_i|_{\mathcal{N}}\rangle = 0,\ \forall i \in I\big\}. \tag{3.30}$$

Lemma 3.7 *The differential of the mapping $\mathcal{R}$ in (3.5) is given by*

$$d\mathcal{R}(\Omega)Y = \tau\sum_{i\in I}\Big\langle t_i(\Omega), \Pi_0\Big(\frac{Y_i}{\Omega_i}\Big)\Big|_{\mathcal{N}}\Big\rangle, \quad \forall Y \in \mathcal{Y}_\Omega. \tag{3.31}$$

Proof see Appendix A.1.

Putting all results together, we state the main result of this section.

Theorem 3.8 (loss function gradient) *Let*

$$\mathcal{L}(\Omega) = f_{\mathcal{L}}\big(v_T(\Omega)\big) + \mathcal{R}(\Omega) \tag{3.32}$$

be a continuously differentiable loss function, where $v_T(\Omega)$ given by (2.26c) is the vectorized solution to the linearized assignment flow (2.22b) at time $t = T$. Then, its gradient $\partial\mathcal{L}(\Omega)$ is given by

$$\langle \partial\mathcal{L}(\Omega), Y\rangle = d\mathcal{L}(\Omega)Y, \qquad \forall Y \in \mathcal{Y}_\Omega \tag{3.33a}$$

with

$$d\mathcal{L}(\Omega)Y = \big\langle \partial f_{\mathcal{L}}\big(v_T(\Omega)\big), df_4(\Omega)Y\big\rangle + d\mathcal{R}(\Omega)Y \tag{3.33b}$$

and $df_4(\Omega)$ given by (3.28), and with $d\mathcal{R}(\Omega)Y$ given by Lemma 3.7.

Proof The claim (3.33) follows from applying the definition of the gradient in (3.33a) and evaluating the right-hand side using the chain rule and Proposition 3.6, to obtain (3.33b). □

3.3 Gradient Approximation

In this section, we discuss the complexity of the evaluation of the loss function gradient $\partial\mathcal{L}(\Omega)$ as given by (3.33), and we develop a low-rank approximation (3.47) that is computationally feasible and efficient.

3.3.1 Motivation

We reconsider the gradient $\partial\mathcal{L}$ given by (3.33). The gradient involves the term $df_4(\Omega)Y$, given by (3.28), which comprises two summands. We focus on the computationally expensive first summand on the right-hand side of (3.28a) given by (3.28b)-(3.28c), i.e., the term

$$\underbrace{\Big(\big(\mathrm{expm}(TA^J(\Omega)), v_T(\Omega)\big)\otimes e_{n+1}^\top\Big)\varphi\big(-\mathcal{A}(\Omega)\oplus\mathcal{A}(\Omega)^\top\big)\cdot df_{\mathcal{A}}\big(\mathrm{vec}_r(\Omega)\big)}_{=:C(\Omega)}\mathrm{vec}_r(Y). \tag{3.34}$$

In order to evaluate the corresponding component of $\partial\mathcal{L}(\Omega)$ based on (3.33b), the matrix $C(\Omega)$ is transposed and multiplied with $\partial f_{\mathcal{L}}(v_T(\Omega))$,

$$\begin{aligned}
&C(\Omega)^\top\partial f_{\mathcal{L}}(v_T(\Omega)) && (3.35a)\\
&\quad = df_{\mathcal{A}}\big(\mathrm{vec}_r(\Omega)\big)^\top\varphi\big(-\mathcal{A}(\Omega)^\top\oplus\mathcal{A}(\Omega)\big)\\
&\qquad\cdot\Big(\big(\mathrm{expm}(TA^J(\Omega)), v_T(\Omega)\big)^\top\otimes e_{n+1}\Big)\partial f_{\mathcal{L}}(v_T(\Omega)) && (3.35b)\\
&\quad = df_{\mathcal{A}}\big(\mathrm{vec}_r(\Omega)\big)^\top\varphi\big(-\mathcal{A}(\Omega)^\top\oplus\mathcal{A}(\Omega)\big)\\
&\qquad\cdot\Big(\big(\mathrm{expm}(TA^J(\Omega)), v_T(\Omega)\big)^\top\otimes e_{n+1}\Big)\\
&\qquad\cdot\big(\partial f_{\mathcal{L}}(v_T(\Omega))\otimes(1)\big) && (3.35c)\\
&\quad \overset{(2.2)}{=} df_{\mathcal{A}}\big(\mathrm{vec}_r(\Omega)\big)^\top\varphi\big(-\mathcal{A}(\Omega)^\top\oplus\mathcal{A}(\Omega)\big)\\
&\qquad\cdot\Big(\big(\mathrm{expm}(TA^J(\Omega)), v_T(\Omega)\big)^\top\partial f_{\mathcal{L}}(v_T(\Omega))\otimes e_{n+1}\Big). && (3.35d)
\end{aligned}$$

Thus, the matrix-valued function φ defined by (2.24) has to be evaluates at a Kronecker sum of matrices and then multiplied by a vector. The structure of this expression has the general form

$$f(M_1\oplus M_2)(b_1\otimes b_2), \qquad M_1, M_2\in\mathbb{R}^{k\times k}, \quad b_1, b_2\in\mathbb{R}^k, \tag{3.36}$$

where in our case we have

$$M_1 = -\mathcal{A}(\Omega)^\top,\ M_2 = \mathcal{A}(\Omega),\ k = n+1 = |I||J|+1, \tag{3.37a}$$

$$b_1 = \big(\mathrm{expm}(TA^J(\Omega)),\ v_T(\Omega)\big)^\top\partial f_{\mathcal{L}}\big(v_T(\Omega)\big), \tag{3.37b}$$

$$b_2 = e_{n+1},\ f = \varphi. \tag{3.37c}$$

As the following discussions also hold in the general setting (3.36), we derive our gradient approximation in this full generality. Afterward, we apply our setting to the gradient approximation (3.47). First, we discuss two ways to compute (3.36):

Direct Computation Compute the Kronecker sum $M_1\oplus M_2$, evaluate the matrix function φ and multiply the vector $b_1\otimes b_2$. This approach has space and time complexity of at least $\mathcal{O}(k^4)$, with k given by (3.37a). The complexity might be even higher depending on how the function f is evaluated.

Krylov Subspace Approximation Use the Krylov space $\mathcal{K}_m(M_1\oplus M_2, b_1\otimes b_2)$ for approximating (3.36), as explained in Sect. 2.4. This approach has space complexity $\mathcal{O}(k^2m^2)$ and time complexity $\mathcal{O}(k^2(m+1))$ [22, p. 132].

Remark 3.9 (space complexity) Consider an image with 512×512 pixels ($|I| = 262\,144$), $|J| = 10$ labels (i.e., $k = |I||J|+1 = 2\,621\,441$) and using 8 bytes per number. Then the direct computation requires to store more than 10^{14} terabytes of data. The Krylov subspace approximation (with $m = 10$) is significantly cheaper, but still requires to store more than 5000 terabytes. Hence both methods are computationally infeasible especially in view of the fact that (3.36) has to be recomputed in every step of the gradient descent procedure (3.7).

3.3.2 An Approximation by Benzi and Simoncini

To reduce the memory footprint, we employ an approximation for computing (3.36), first discussed by Benzi and

Simoncini [6], and refine it using a new additional approximation in Sect. 3.3.3. In the following, the notation from Benzi and Simoncini is slightly adapted to our definition (2.1) of the Kronecker sum that differs from Benzi and Simoncini's definition of the Kronecker sum ($A \oplus B = B \otimes I + I \otimes A$).

The approach uses the Arnoldi iteration [21] to determine orthonormal bases P_m, Q_m and the corresponding Hessenberg matrices T_1 and T_2 of the two Krylov subspaces $\mathcal{K}(M_1, b_1), \mathcal{K}(M_2, b_2)$. The matrices are connected by a standard relation of Krylov subspaces [8, Section 13.2.1],

$$M_1 P_m = P_m T_1 + t_1 p_{m+1} e_m^\top, \tag{3.38a}$$
$$M_2 Q_m = Q_m T_2 + t_2 q_{m+1} e_m^\top, \tag{3.38b}$$

where $t_1 \in \mathbb{R}$, $p_{m+1} \in \mathbb{R}^n$ (resp. $t_2 \in \mathbb{R}$, $q_{m+1} \in \mathbb{R}^n$) refer to the entries of the Hessenberg matrices and the orthonormal bases in the next step of the Arnoldi iteration. With these formulas, we deduce

$$\begin{aligned}
&(M_1 \oplus M_2)(P_m \otimes Q_m) \\
&\overset{(2.1)}{=} (M_1 P_m \otimes Q_m) + (P_m \otimes M_2 Q_m) && (3.39a)\\
&\overset{(3.38)}{=} (P_m T_1 + t_1 p_{m+1} e_m^\top \otimes Q_m) \\
&\qquad +(P_m \otimes Q_m T_2 + P_m \otimes t_2 q_{m+1} e_m^\top) && (3.39b)\\
&= (P_m \otimes Q_m)(T_1 \oplus T_2) + (t_1 p_{m+1} e_m^\top \otimes Q_m) \\
&\quad +(P_m \otimes t_2 q_{m+1} e_m^\top). && (3.39c)
\end{aligned}$$

Ignoring the last two summands and multiplying by $(P_m \otimes Q_m)^\top$ yields the approximation

$$(M_1 \oplus M_2) \approx (P_m \otimes Q_m)(T_1 \oplus T_2)(P_m \otimes Q_m)^\top, \tag{3.40}$$

which after applying f and multiplying $b_1 \otimes b_2$ leads to the approximation

$$\begin{aligned}
&f(M_1 \oplus M_2)(b_1 \otimes b_2) \\
&\quad \approx (P_m \otimes Q_m) f(T_1 \oplus T_2)(P_m \otimes Q_m)^\top (b_1 \otimes b_2) && (3.41)
\end{aligned}$$

of the expression (3.36) as proposed by Benzi and Simoncini. We note that, due to the orthonormality of the bases P_m and Q_m and their relation to the vectors b_1, b_2 that generate the subspaces $\mathcal{K}(M_1, b_1), \mathcal{K}(M_2, b_2)$, the approximation simplifies to

$$\begin{aligned}
&f(M_1 \oplus M_2)(b_1 \otimes b_2) \\
&\quad \approx \|b_1\|\|b_2\|(P_m \otimes Q_m) f(T_1 \oplus T_2) e_1 && (3.42a)\\
&\quad = \|b_1\|\|b_2\| \operatorname{vec}_r \left(P_m \ \operatorname{vec}_r^{-1}\left(f(T_1 \oplus T_2)e_1\right) Q_m^\top\right), && (3.42b)
\end{aligned}$$

where $e_1 \in \mathbb{R}^{m^2}$ denotes the first unit vector.

Remark 3.10 (complexity of the approximation (3.42b)) Computing and storing the matrices P_m, Q_m, T_1 and T_2 has space complexity $\mathcal{O}(2km^2)$ and a time complexity of $\mathcal{O}(2k(m+1))$ [22, p. 132]. Storing the matrices $T_1 \oplus T_2$ and $f(T_1 \oplus T_2)$ has complexity $\mathcal{O}(m^4)$. Finally, multiplying the three matrices $P_m \in \mathbb{R}^{k \times m}$, $\operatorname{vec}_r^{-1}(f(T_1 \oplus T_2)e_1) \in \mathbb{R}^{m \times m}$ and $Q_m^\top \in \mathbb{R}^{m \times k}$ has time complexity $\mathcal{O}(k^2 m + km^2)$ and space complexity $\mathcal{O}(k^2 + km)$.

Ignoring negligible terms (recall $m \ll k$), the entire approximation has computational complexity $\mathcal{O}(k^2 m)$ and storage complexity $\mathcal{O}(k^2)$. Compared to the Krylov subspace approximation of (3.36) discussed in the preceding section, this is a reduction of space complexity by a factor m^2.

Consider an image with 512×512 pixels ($|I| = 262\,144$) and $|J| = 10$ labels as in Remark 3.9. Then the approximation (3.42b) requires to store a bit more than 50 terabytes. While this is a huge improvement compared to the 5000 terabytes from the Krylov approximation (see Remark 3.9), using this approximation is still computationally infeasible. This motivates why we introduce below an additional low-rank approximation that yields a computationally feasible and efficient gradient approximation.

3.3.3 Low-Rank Approximation

We consider again the approximation (3.42b)

$$\begin{aligned}
&f(M_1 \oplus M_2)(b_1 \otimes b_2) \\
&\quad \approx \|b_1\|\|b_2\| \operatorname{vec}_r \left(P_m \ \operatorname{vec}_r^{-1}\left(f(T_1 \oplus T_2)e_1\right) Q_m^\top\right) && (3.43)
\end{aligned}$$

and decompose the matrix $\operatorname{vec}_r^{-1}\left(f(T_1 \oplus T_2)e_1\right) \in \mathbb{R}^{m \times m}$ using the singular value decomposition (SVD)

$$\operatorname{vec}_r^{-1}\left(f(T_1 \oplus T_2)e_1\right) = \sum_{i \in [m]} \sigma_i y_i \otimes z_i^\top, \tag{3.44}$$

with $y_i, z_i \in \mathbb{R}^m$ and the singular values $\sigma_i \in \mathbb{R}$, $i \in [m]$. As m is generally quite small, computing the SVD is neither computationally nor storage-wise expensive. We accordingly rewrite the approximation in the form

$$\begin{aligned}
&\|b_1\|\|b_2\| \operatorname{vec}_r \left(P_m \ \operatorname{vec}_r^{-1}\left(f(T_1 \oplus T_2)e_1\right) Q_m^\top\right) && (3.45a)\\
&\quad = \|b_1\|\|b_2\| \operatorname{vec}_r \left(P_m \Big(\sum_{i \in [m]} \sigma_i y_i \otimes z_i^\top\Big) Q_m^\top\right) && (3.45b)\\
&\quad = \|b_1\|\|b_2\| \sum_{i \in [m]} \sigma_i (P_m y_i) \otimes (Q_m z_i). && (3.45c)
\end{aligned}$$

Remark 3.11 (space complexity) While the factorized form (3.45c) is equal to the approximation (3.42b), it requires only a fraction of the storage space: The intermediate results require storing m singular values and k numbers for each $P_m y_i$ and $Q_m z_i$, and the final approximation has an additional storage requirement of $\mathcal{O}(2km)$. In total $\mathcal{O}(4km)$ numbers need to be stored.

For a 512×512 pixels image with 10 labels (see Remark 3.9), storing this approximation requires at most a gigabyte of memory.

In practice, this can be further improved: Numerical experiments show that the singular values decay very rapidly, such that just the first singular value can be used to obtain the gradient approximation

$$f(M_1 \oplus M_2)(b_1 \otimes b_2) \approx \|b_1\|\|b_2\|\sigma_1(P_m y_1) \otimes (Q_m z_1). \tag{3.46}$$

Numerical results in Sect. 4 demonstrate that this approximation is sufficiently accurate.

Remark 3.12 (space complexity) The term $\|b_1\|\|b_2\|\sigma_1(P_m y_1) \otimes (Q_m z_1)$ requires to store $\mathcal{O}(2k)$ numbers, i.e., about twice as much storage space as the original image. In total, we need to store $\mathcal{O}(2k+2km)$ numbers. The required storage for the running example (see Remark 3.9) now adds up to less than 500 megabytes.

We conclude this section by returning to our problem using the notation (3.37) and state the proposed *low-rank approximation of the loss function gradient*. By (3.33), (3.35), (3.37) and (3.46), we have

$$\partial\mathcal{L}(\Omega) \approx c(\Omega) \cdot \operatorname{vec}_r^{-1}\Big(df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)^\top\big(\sigma_1(P_m y_1) \otimes (Q_m z_1)\big)\tag{3.47a}$$

where

$$c(\Omega) = \Big\|\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big)^\top \partial f_{\mathcal{L}}\big(v_T(\Omega)\big)\Big\|, \tag{3.47b}$$

$$v_T(\Omega) = v(T;\Omega) \qquad \text{(cf. (2.26c))} \tag{3.47c}$$

$$\sigma_1 y_1 \otimes z_1^\top \approx \operatorname{vec}_r^{-1}\big(\varphi(T_1 \otimes T_2)e_1\big). \quad \text{(largest singular value and vectors)} \tag{3.47d}$$

Here, the matrices P_m, Q_m, T_1, T_2 result from the Arnoldi iteration, cf. (3.38), that returns the two Krylov subspaces used to approximate the matrix vector product $\varphi(-\mathcal{A}(\Omega)^\top \oplus \mathcal{A}(\Omega))b_1$, with b_1 given by (3.37b).

3.4 Computing the Gradient Using Automatic Differentiation

An entirely different approach to computing the gradient $\partial\mathcal{L}(\Omega)$ of the loss function (3.6) is to not use an approximation of the exact gradient given in closed form by (3.8), but to replace the solution $v_T(\Omega)$ to the linearized assignment flow in (3.33b) by an approximation determined by a numerical integration scheme and to compute the exact gradient therefrom. Thus, one replaces a *differentiate-then-approximate* approach by an *approximate-then-differentiate* alternative. We numerically compare these two approaches in Sect. 4.

We sketch the latter alternative. Consider again the loss function (3.6) evaluated at the linearized assignment flow integrated up to time T

$$\mathcal{L}(\Omega) = f_{\mathcal{L}}\big(v_T(\Omega)\big). \tag{3.48}$$

Gradient approximations determined by automatic differentiation depend on what numerical scheme is used. We pick out two basic choices out of a broad range of proper schemes studied in [27]. In both cases, we implemented the loss function $f_{\mathcal{L}}$ in PyTorch together with the functions $\Omega \mapsto A^J(\Omega)$ and $\Omega \mapsto b(\Omega)$ given by (2.26). Now two approximations can be distinguished depending on how the mappings $(A^J(\Omega), b(\Omega)) \mapsto v_T(\Omega) = v(T;\Omega)$ are implemented.

Automatic Differentiation Based on the Explicit Euler Scheme We partition the interval $[0, T]$ into T/h subintervals with some step size $h > 0$ and use the iterative scheme

$$v^{(k+1)} = v^{(k+1)} + h\big(A^J(\Omega)v^{(k)} + b(\Omega)\big), \qquad v^{(0)} = 0, \tag{3.49}$$

in order to approximate $v_T(\Omega) \approx v^{(T/h)}$ the solution to the linearized assignment flow ODE (2.26a). As the computations only involve basic linear algebra, PyTorch is able to compute the gradient using automatic differentiation.

Automatic Differentiation Based on Exponential Integration The second approximation utilizes the numerical integration scheme developed in Sect. 2.4. Again, only basic operations of linear algebra are involved so that PyTorch can compute the gradient using automatic differentiation. The more special matrix exponential (2.29) is computed by PyTorch using a Taylor polynomial approximation [4].

Both approaches determine an approximation of the Euclidean gradient $\partial\mathcal{L}(\Omega)$ which we subsequently convert into an approximation of the Riemannian gradient using Eq. (3.8).

4 Experiments

In this section, we report and discuss a series of experiments illustrating our novel gradient approximation (3.47) and the applicability of the linearized assignment flow to the image labeling problem.

We start with a discussion of the data generation (Sect. 4.1) and the general experimental setup (Sect. 4.2), before discussing properties of the gradient approximation (Sect. 4.3). In order to illustrate a complete pipeline that can also label previously unseen images, we trained a simple parameter predictor and report its application in Sect. 4.4.

4.1 Data Generation

As for the experiments, we focused on the image labeling scenarios depicted in Fig. 1a, b. Each scenario consists of a set containing five 128×128 pixel images with *random* Voronoi structure, in order to mimic low-dimensional structure that has to be separated in noisy data from the background. This task occurs frequently in applications and cannot be solved without *adaptive* regularization.

For the design of the parameter predictor (Sect. 4.4), we used all patches of five additional unseen images for validation. In all cases we report the mean over all labeled pixels of 5 training and validation images, respectively. In order to test the resilience to noise, we added Gaussian noise to the images. The ground truth labeling is, in both labeling scenarios, given by the noiseless version of the images.

In the first scenario illustrated in Fig. 1a, we want to separate the boundary of the cells (black label) from their interior (white label). The main difficulty here is to preserve the thin line structures even in the presence of image noise. Weight patches with uniform (uninformed) weights average out most of the lines as Fig. 1c shows.

In the second scenario illustrated in Fig. 1b, we label the Voronoi cells according to their color represented by 8 labels. Due to superimposed noise, a pixelwise local rounding to the nearest label yields about 50% wrongly labeled pixels, see Fig. 1d.

4.2 Experimental Setup

Features and Parametrization For simplicity, we used the raw image data in a 3×3 window around each pixel as feature (2.6) for this pixel. Weight patches $(\omega_{ik})_{k \in \mathcal{N}_i}$ in the Ω-matrix (2.18) also had the size of 3×3 pixels in all experiments. While the linearized assignment flow works with arbitrary features and also with larger neighborhood sizes for the weight parameters, the above setup suffices to illustrate and substantiate the contribution of this paper.

Performance Measure All labelings were evaluated on the tangent space of the assignment manifold using the loss function $f_{\mathcal{L}}$ given by (3.4). Since the values of this function are rather abstract, however, we report the percentage of wrongly labeled pixels in all performance plots.

Gradient Computation We evaluated the loss function and approximated its Riemannian gradient in three different ways, as further detailed in Sect. 4.3, throughout using uniform (uninformed) weight patches as initialization. In particular, other common ways to update the parameters, like Adam or AdaMax [15], are possible as well, in conjunction with our approach. Therefore, we also compared gradient approximations based on our approach with the results of automatic differentiation, as implemented by PyTorch [19].

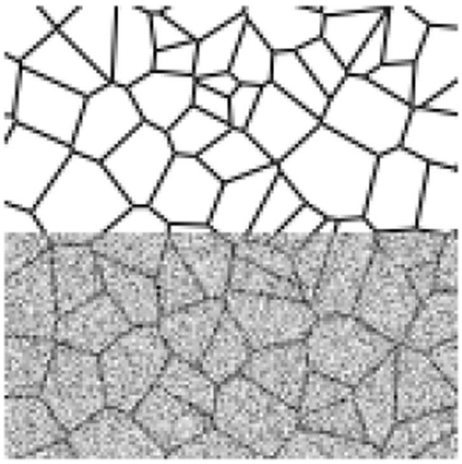
Parameter Prediction Parameter prediction for labeling novel data relies on the relation of features extracted from training data to corresponding parameters estimated by the Riemannian gradient descent (3.7). For any feature extracted from novel data, the predictor specifies the parameters, to be used for labeling the data by integrating the linearized assignment flow after substituting the predicted parameters. Details are provided in Sect. 4.4.

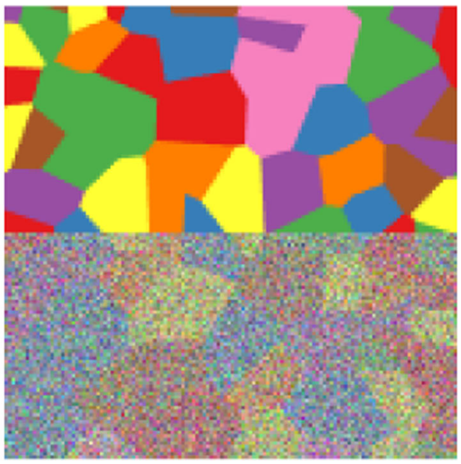
4.3 Properties of the Gradient Approximation

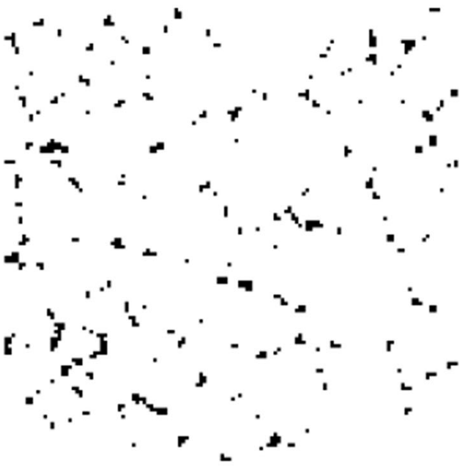
In this section, we report results that empirically validate our novel gradient approximation (3.47) by means of parameter estimation for the linearized assignment flow.

First, we compared our gradient approximation with two methods based on automatic differentiation (backpropagation), see also Sect. 3.4. To this end, we implemented in PyTorch [19] the simple explicit Euler scheme (3.49) for integrating the linearized assignment flow and computed the gradient of the loss function $\mathcal{L}(\Omega)$ (3.6) with respect to Ω using automatic differentiation. Similarly, the Krylov subspace approximation (2.28b) of the solution of the linearized assignment flow was implemented in PyTorch. As all involved computations in this approximation are basic linear algebra operations, PyTorch is able to apply automatic differentiation for evaluating the gradient.

These gradients are used for carrying out the gradient descent iteration (3.7) in order to optimize the weight parameters. Figure 2 illustrates the comparison of the three approaches. Although they rely on quite different principles, we observe a remarkable comparability of the three approaches with respect to the reduction of the percentage of wrongly labeled pixels per training iteration, for both noisy and noiseless images. In particular, our low-rank approximation based on the closed-form loss function gradient expression is competitive. In view of the minor differences between the curves, we point out that changing hyperparameters, like the step size in the gradient descent or the scale parameter τ of the regularizer $\mathcal{R}$ in (3.5), have a greater effect on the training performance than the choice of either of the three approaches. Overall, these results validate the closed-form formulas in Sect. 3.2 and, in particular, Theorem 3.8, and the subsequent low-rank approximation in Sect. 3.3. We point out, however, that our approach only reveals data-dependent low-dimensional subspaces where the essential parameters of the linearized assignment flow reside.


(a) Random Voronoi line structure to be labeled from noisy input data.


(b) Random colored Voronoi regions to be labeled from noisy input data.


(c) Labeling with uniform weights, that is without weight adaption, cannot separate line structure from the background in noisy data.

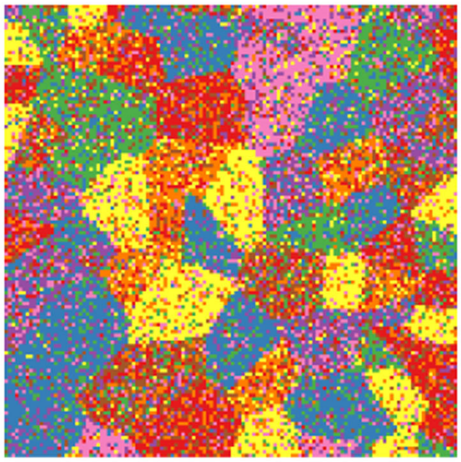

(d) Pixelwise individual nearest label assignments produce an error rate larger than 50%.

Fig. 1 Randomized scenarios for training and testing. Two randomly generated images for the two respective scenarios that were used to evaluate weight parameter estimation and prediction. **a** Random line structure whose accurate labeling requires to adapt weight parameters. **b** Random Voronoi cells to be labeled by pixelwise assignment of one of the colors (🟥, 🟦,🟩,🟪,🟧,🟨,🟫,🩷). In both cases, Gaussian noise was added. The resulting noisy images are shown in the lower part of either panel (rescaled in the color channels to avoid color clipping). **c** The amount of noise is chosen quite large such that a labeling with uniform ("uninformed non-adaptive") weights completely destroys the thin line structure in **a**. **d** A pixelwise local nearest label assignment yields around 50% wrongly labeled pixels for the labeling scenario depicted in **b**. Both of these naive parameter settings indicate the need for a more structured choice of the weight patches, by taking into account local image features in a local spatial neighborhood

Next, we compared our gradient approximation to the exact gradient on a per-pixel basis. However, as the exact gradient is computationally infeasible, we used the gradient produced by automatic differentiation of the explicit Euler scheme with a very small step size as surrogate. Figure 3a demonstrates the high accuracy of our gradient approximation. A pixelwise illustration of the gradient approximation, at the initial step of the training procedure for adapting the parameters, is provided in Fig. 3b. The set of pixels with nonzero loss function gradient concentrate around the line structure since here weight adaption is required to achieve a proper labeling.

Our last three experiments regarding the gradient approximation, illustrated in Fig. 4, concern

- the influence of the Krylov dimension m,
- the rank of our approximation, and
- the time T up to which the linearized assignment flow is integrated.

We observe according to Fig. 4a that already Krylov subspace of small dimension $m \approx 10$ suffice for computing linearized assignment flows and learning their parameters. Similarly, the final rank-one gradient approximation of the gradient according to Eq. (3.46) suffices for parameter estimation, as illustrated in Fig. 4b. These experiments show that quite *low-dimensional* representations suffice for representing the information required for optimal regularization of dynamic image labeling. We point out that such insights cannot be gained from automatic differentiation.

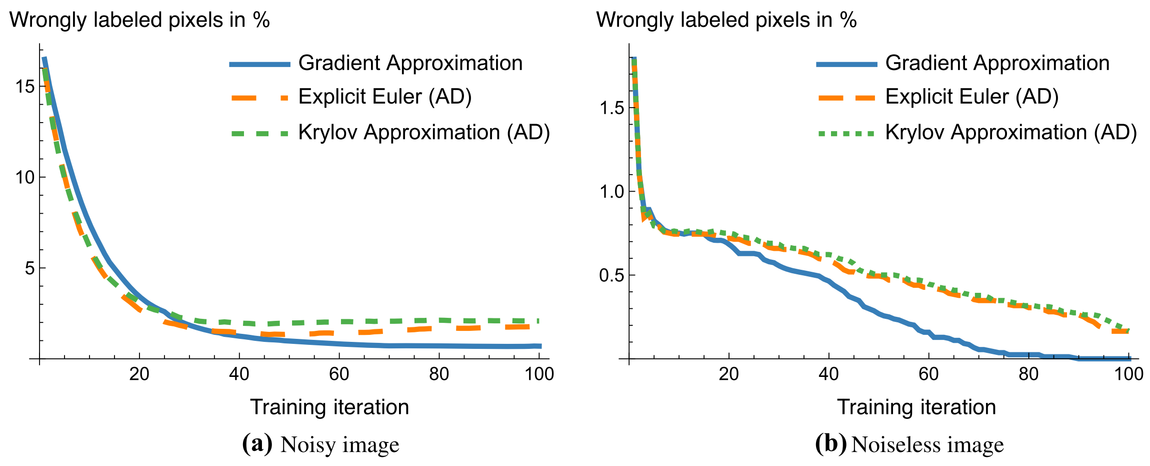


Fig. 2 Comparing gradient approximation and automatic differentiation. Both figures show, for the second scenario depicted in Fig. 1b, the effect of parameter learning in terms of the labeling error during the training procedure (3.7). **a** Shows the result for noisy input data, **b** or noiseless input data. Note the different scales of the two ordinates. As is exemplarily shown here by both figures, we generally observed very similar results for all three algorithms which validates the closed-form formulas in Sect. 3.2 and the subsequent subspace approximation in Sect. 3.3

The influence of the time T used for integrating the linearized assignment flow on parameter learning is illustrated in Fig. 4c. For the considered parameter estimation setup, we observe that already small integration times T yield good training results, whereas large times T yield slower convergence. A possible explanation is that, in the latter case, the linearized assignment flow is close to an integral solution which, when erroneous, is more difficult to correct.

4.4 Parameter Prediction

Besides parameter learning, parameter *prediction* for unseen test data defines another important task. This task amounts to model and represent the relation of local features and optimal weight parameters, as basis to predict proper weights in unseen test data as a function of corresponding local features.

We illustrate this for the scenario depicted in Fig. 1a using the following simple end-to-end learned approach to parameter prediction. We trained a predictor that produces a weight patch $\widehat{\Omega}_i$ given the features f_i at vertex i of novel unseen data. The predictor is parameterized with $N = 50$ by

$$p_j \in \mathbb{R}^{3|\mathcal{N}|}, \quad j \in [N] \quad \text{feature prototypes}, \tag{4.1a}$$

$$\nu_j \in T_0, \quad j \in [N] \quad \text{tangent vectors representing prototypical weight patches}, \tag{4.1b}$$

and a scale parameter $\sigma \in \mathbb{R}$. Similar to the assignment vectors (2.7), the to-be-predicted weight patches $\widehat{\Omega}_i$ are elements of the probability simplex $\mathring{\Delta}_{|\mathcal{N}_i|}$, see (2.18). Accordingly, use tangent vector $\nu_j \in T_0$ to represent weight patches. In particular, tangent vector of *predicted* weight patches result from weight averaging of vectors $\{\nu_j\}_{j\in[N]}$, and the predicted weight patch by lifting, see (4.4).

We initialize $\sigma = 1$ and initialize the p_j, $j \in [N]$ by clustering noise-free patches extracted from of training images. Given p_j, we initialize ν_j such that it is directed toward the label of the corresponding prototypical patch,

$$\nu_j = \Pi_0 \left(e^{-\|p_{j,1}-p_{j,\text{center pixel}}\|}, \dots, e^{-\|p_{j,|\mathcal{N}|}-p_{j,\text{center pixel}}\|}\right)^\top, \quad j \in [N]. \tag{4.2}$$

The predictor is trained by the following gradient descent iteration. As the change in the number of wrongly labeled pixels was small, we stopped the iteration after 100 steps, see Fig. 5c.

(1) We compute the similarities

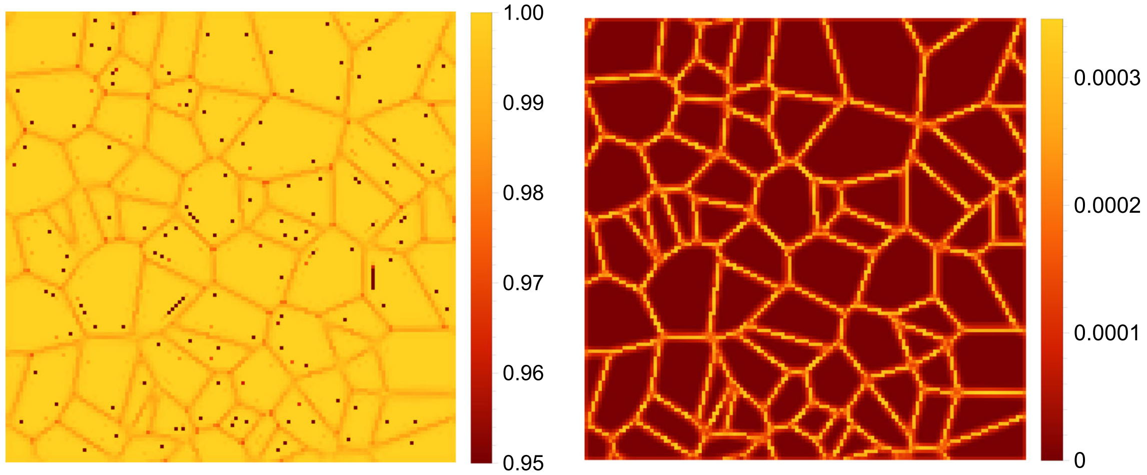
$$s_{ij} = e^{-\sigma\|f_i - p_j\|}, \qquad j \in [N] \tag{4.3}$$

for each f_i and pixels i in all training images.

(2) We predict the corresponding weight patches as lifted weighted average of the tangent vectors ν_j

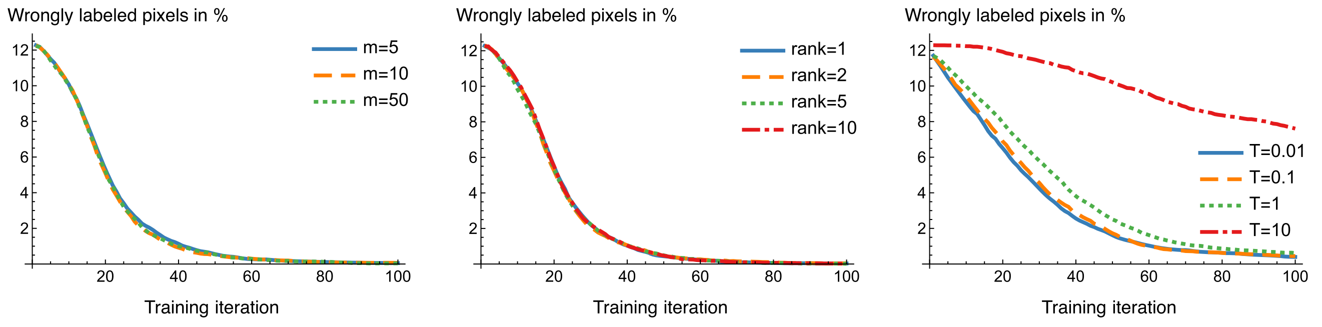
$$\widehat{\Omega}_i(\nu, p, \sigma) = \exp_{\mathbb{1}_\Omega}\left(\sum_{j\in[N]} \frac{s_{ij}}{\sum_{k\in[N]} s_{ik}} \nu_j\right). \tag{4.4}$$

(3) Substituting $\widehat{\Omega}$ for Ω, we run the linearized assignment flow and evaluate the distance function (3.4).



(a) Gradient directions **(b)** Norm of gradients

Fig. 3 Checking the gradient approximation at each pixel. We evaluated our gradient approximation (3.47), at the first step of the training iteration and at each pixel, for the scenario depicted in Fig. 1a. As a proxy for the exact but computationally infeasible gradient, we used the gradient produced by automatic differentiation of the explicit Euler scheme with a very small step size. Then, we compared both gradients at each pixel using the cosine similarity, i.e., the value 1 means that the gradients point exactly in the same direction, whereas 0 signals orthogonality and −1 means that they point in opposite directions. **a** More than 99% of the pixels have a value of 0.9 or more, corresponding to an angle of 26° or less between the gradient directions. This illustrates excellent agreement between our gradient approximation and the exact gradient. Disagreements with the exact gradient occur rarely and randomly at isolated pixels throughout the image. **b** Norm of the gradients are displayed at each pixel. Non-vanishing norms indicate where parameter learning (adaption) occurs. Since the initial weight parameter patches are uniform, no adaption—corresponding to zero norms of gradients—occurs in the interior of each Voronoi cell, because parameters are already optimal in such homogeneous regions



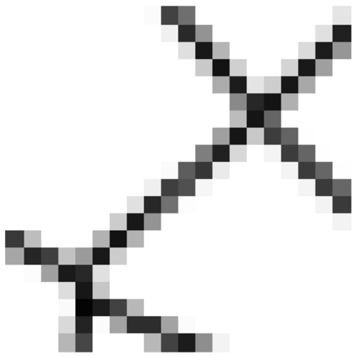
(a) Krylov subspace dimension m **(b)** Rank of the approximation **(c)** Integration time T

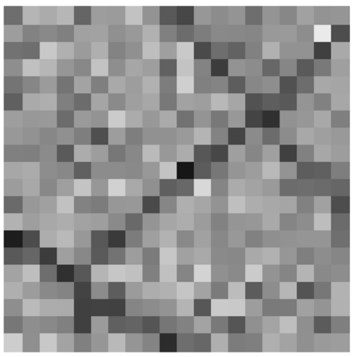
Fig. 4 Influence of Krylov subspace dimension, rank of the gradient approximation and integration time. The setup of Fig. 2 was used to demonstrate the influence of the Krylov subspace dimension, the low-rank approximation and the integration time T on our gradient approximation for parameter learning. **a** In general, we observed that Krylov dimensions of 5 to 10 are sufficient for most experiments. Larger Krylov dimensions only increase the computation time without any noticeable improvement of accuracy. **b** Training curves for different low-rank approximations coincide. This illustrates that just selecting the largest singular value and vectors in (3.47), according to the final rank-one approximation (3.46), suffices for parameter learning. **c** For small integration times T, the convergence rates of training do not much differ. Only for larger time points T, we observe slower convergence of training, presumably because almost hard decisions are more difficult to correct by changing the parameters of the underlying dynamical system

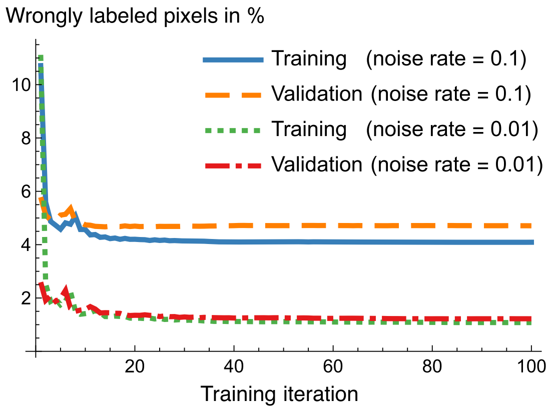
(4) The gradient of this function with respect to the predictor parameters (ν, p, σ) results from composing the differential due to Theorem 3.8 and the differential of (4.4).
(5) The gradient is used to update the predictor parameters, and all steps are repeated.

During training, the accuracy of the predictor is monitored, as illustrated in Fig. 5c. The iteration terminates when the slope of the validation curve, which measures label changes, are sufficiently flat.

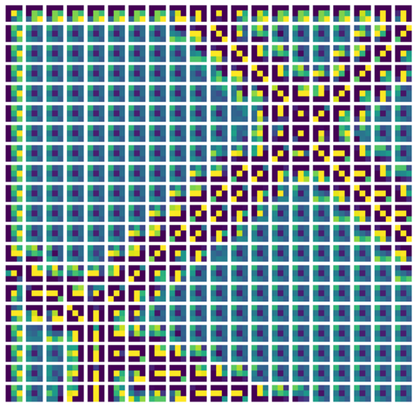
After the training of the predictor, the linearized assignment flow is parametrized in a data-driven way so as to separate reliably line structure in noisy data for arbitrary ran-

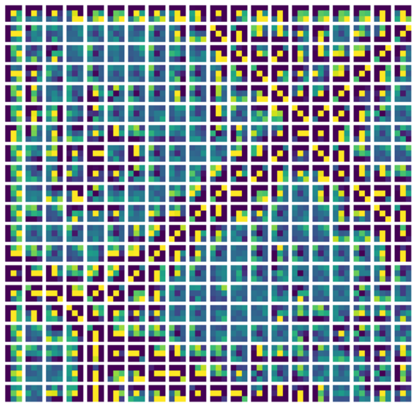

(a) Section of noise-free image

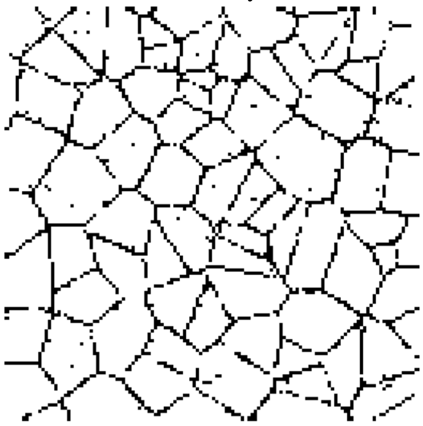

(b) Section of noisy image



(c) Predictor accuracy


(d) Predicted Ω-weight patches for noiseless input


(e) Predicted Ω-weight patches for noisy input


(f) Labeling of the noisy image with predicted weights

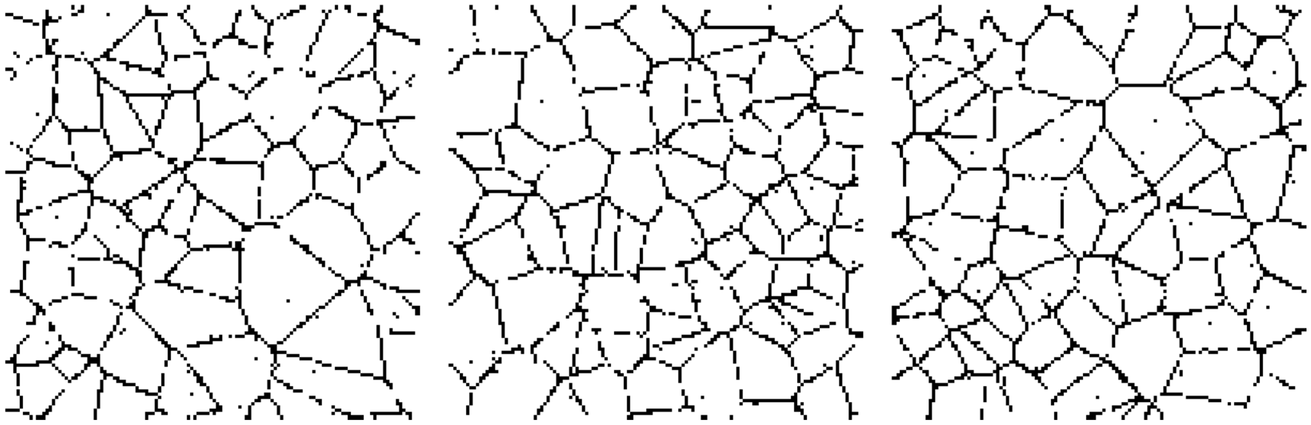


Fig. 5 Parameter predictor. We learned a weight patch predictor as described in Sect. 4.4 for the scenario depicted in Fig. 1a. In order to assess the predicted parameters by comparison, we also estimated weights patches for the *noise-free* test data in the same way as for the training data. **a** Section of a noise-free test image. **b** The corresponding section of the noisy test image that is used as input data for prediction. **c** The training and validation accuracy during the training of the predictor. **d** Weight patches estimated for the *noise-free* data (**a**). **e** Predicted weight patches based on the *noisy* data (**b**). **(f)** The labeled (section of the) test image using the predicted weight patches (**d**). Comparing this result to the result depicted in Fig. 1c shows the effect of *predicted* parameter adaption. **Last row:** Further labelings on unseen noisy random test images

dom instances, as depicted in Fig. 5: panel (f) and last row. This result should be compared to the non-adaptive labeling result in Fig. 1c.

5 Conclusion and Further Work

5.1 Conclusion

We presented a novel approach for learning the parameters of the linearized assignment flow for image labeling. Based on the exact formula of the parameter gradient of a loss function subject to the ODE-constraint, an approximation of the gradient was derived using exponential integration and a Krylov subspace based low-rank approximation, that is memory efficient and sufficiently accurate. Experiments demonstrate that our research implementation is on par with highly tuned-machine learning toolboxes. Unlike the latter, however, our approach additionally returns the essential information for image labeling in terms of a low-dimensional parameter subspace.

5.2 Future Work

Our future work will study generalizations of the linearized assignment flow. Since this can be done within the overall mathematical framework of the assignment flow approach, the result presented in this paper is applicable. We briefly indicate this for the continuous-time ODE (1.1) that we write down here again with an index 0,

$$\dot{V}_0 = A_0(\Omega_0)V_0 + B_0. \tag{5.1}$$

Recall that B_0, given by B_{W_0} of (2.22b), represents the input data (2.15) via the mappings (2.16) and (2.17). Now suppose the data are represented in another way and denoted by B_1. Then, consider the additional system

$$\dot{V}_1 = A_1(\Omega_1)V_1 + B_1 + V_0(T)L, \tag{5.2}$$

where the solution $V_0(T_0)$ to (5.1) at time $t = T_0$, possibly transformed to a tangent subspace by a linear mapping L, modifies the data term B_1 of (5.2). Applying (2.24) to (5.1) at time $t = T_0$ and to (5.2) at time $t = T_1$ yields the solution

$$V_1(T_1) = T_1\varphi\big(T_1 A_1(\Omega)\big)\Big(B_1 + T_0\varphi\big(T_0 A_0(\Omega_0)\big)B_0 L\Big), \tag{5.3}$$

which is a *composition* of linearized assignment flows and hence linear too, due to the *sequential* coupling of (5.1) and (5.2). *Parallel* coupling of the dynamical systems is feasible as well and leads to larger matrix φ that is *structured* and linearly depends on the components $A_0(\Omega_0)$, $A_1(\Omega_1)$, L. Designing larger networks of this sort by repeating these steps is straightforward.

In either case, the overall basic structure of (1.1), (1.3) is preserved. This enables us to broaden the scope of assignment flows for applications and to study, in a controlled manner, various mathematical aspects of deep networks in terms of sequences of generalized linearized assignment flow, analogous to (1.6).

Acknowledgements This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2181/1 - 390900948 (the Heidelberg STRUCTURES Excellence Cluster), and within the DFG priority programme 2298 on the "Theoretical Foundations of Deep Learning," grant SCHN 457/17-1.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix A. Proofs

A.1. Proofs of Sect. 3.2.2

Proof of Lemma 3.2 Regarding the differential of the mapping (2.11e) with respect to its second argument, we have $d\exp_p(u)v = R_{\exp_p(u)}v$ by [27, Lemma 4.5], with R given by (2.11b). Applying this relation to (3.16) where $\exp_{\mathbb{1}_{\mathcal{W}}}$ acts row-wise analogous to the mapping R_W as explained by (2.12) and (2.14), yields

$$\begin{aligned} df_1(\Omega)Y &= R_{\exp_{\mathbb{1}_{\mathcal{W}}}(-\frac{1}{\rho}\Omega D)}\Big(-\frac{1}{\rho}YD\Big) \\ &= R_{f_1(\Omega)}\Big(-\frac{1}{\rho}YD\Big), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|}, \end{aligned} \tag{A.1}$$

which is (3.17a). As for the transpose, we vectorize both sides using again (2.14),

$$\begin{aligned} \mathrm{vec}_r\,\big(df_1(\Omega)Y\big) &= \mathrm{Diag}(R_{f_1(\Omega)})\,\mathrm{vec}_r\,\Big(-\frac{1}{\rho}YD\Big) \\ &= -\frac{1}{\rho}\,\mathrm{Diag}(R_{f_1(\Omega)})(I_{|I|}\otimes D^\top)\,\mathrm{vec}_r(Y). \end{aligned} \tag{A.2}$$

Applying the transposed matrix to any vector $\operatorname{vec}_r(Z)$ with $Z \in \mathbb{R}^{|I|\times|J|}$ and taking into account the symmetry of the matrix $\operatorname{Diag}(R_{f_1(\Omega)})$, yields

$$df_1(\Omega)^\top Z = -\frac{1}{\rho}\operatorname{vec}_r^{-1}\big((I_{|I|}\otimes D)\operatorname{Diag}(R_{f_1(\Omega)})\operatorname{vec}_r(Z)\big) \tag{A.3a}$$

$$\overset{(2.14)}{=} -\frac{1}{\rho}\operatorname{vec}_r^{-1}\big((I_{|I|}\otimes D)\operatorname{vec}_r(R_{f_1(\Omega)}Z)\big)$$
$$= -\frac{1}{\rho}R_{f_1(\Omega)}(Z)D^\top. \tag{A.3b}$$

Proof of Lemma 3.3 Since R_{W_0} does not depend on Ω and vec_r is linear, we directly obtain (3.19a). Regarding the transpose map, we expand the right-hand side of (3.19a),

$$df_2(\Omega)Y \overset{(2.14)}{=} \operatorname{Diag}(R_{W_0})\operatorname{vec}_r(df_1(\Omega)Y)$$
$$\overset{(A.2)}{=} -\frac{1}{\rho}\operatorname{Diag}(R_{W_0})\operatorname{Diag}(R_{f_1(\Omega)})(I_{|I|}\otimes D^\top)\operatorname{vec}_r(Y). \tag{A.4}$$

Applying the transposed matrix to any vector $\operatorname{vec}_r(Z) \in \mathbb{R}^{|I|^2}$ yields (recall that the matrices $\operatorname{Diag}(R_{W_0}), \operatorname{Diag}(R_{f_1(\Omega)})$ are symmetric)

$$df_2(\Omega)^\top Z = -\frac{1}{\rho}\operatorname{vec}_r^{-1}\big((I_{|I|}\otimes D)$$
$$\operatorname{Diag}(R_{f_1(\Omega)})\operatorname{Diag}(R_{W_0})\operatorname{vec}_r(Z)\big) \tag{A.5a}$$
$$\overset{(2.14)}{=} -\frac{1}{\rho}\operatorname{vec}_r^{-1}\big((I_{|I|}\otimes D)\operatorname{Diag}(R_{f_1(\Omega)})$$
$$\operatorname{vec}_r(R_{W_0}Z)\big) \tag{A.5b}$$
$$\overset{(2.14)}{=} -\frac{1}{\rho}\operatorname{vec}_r^{-1}\Big((I_{|I|}\otimes D)\operatorname{vec}_r\big(R_{f_1(\Omega)}(R_{W_0}Z)\big)\Big)$$
$$= -\frac{1}{\rho}R_{f_1(\Omega)}(R_{W_0}Z)D^\top \tag{A.5c}$$
$$\overset{(3.17b)}{=} df_1(\Omega)^\top(R_{W_0}Z). \tag{A.5d}$$

□

Proof of Lemma 3.4 We have

$$df_3(\Omega)Y = \big(d\operatorname{Diag}(R_{f_1(\Omega)})Y\big)(\Omega\otimes I_{|J|})$$
$$+\operatorname{Diag}(R_{f_1(\Omega)})(Y\otimes I_{|J|}),\quad \forall Y\in\mathbb{R}^{|I|\times|I|} \tag{A.6}$$

and have to the differential in the first summand on the right-hand side. By (2.14),

$$\operatorname{Diag}(R_{f_1(\Omega)})\operatorname{vec}_r(S) = \operatorname{vec}_r(R_{f_1(\Omega)}S),\quad \forall S\in\mathbb{R}^{|I|\times|J|} \tag{A.7}$$

and hence $d\operatorname{Diag}(R_{f_1(\Omega)})$ is given by

$$\big(d\operatorname{Diag}(R_{f_1(\Omega)})Y\big)\operatorname{vec}_r(S) = \operatorname{vec}_r\big((dR_{f_1(\Omega)}Y)S\big),$$
$$\forall Y\in\mathbb{R}^{|I|\times|I|},\quad \forall S\in\mathbb{R}^{|I|\times|J|}. \tag{A.8}$$

It remains to compute $dR_{f_1(\Omega)}$ and to evaluate the defining right-hand side, to obtain the left-hand side in explicit form. Focusing on a single component $R_{f_{1i}}(\Omega)$ of the mapping $R_{f_1(\Omega)}$, we have by (2.11b)

$$R_{f_{1i}}(\Omega) = \operatorname{Diag}\big(f_{1i}(\Omega)\big) - f_{1i}(\Omega)f_{1i}(\Omega)^\top \tag{A.9a}$$
$$dR_{f_{1i}(\Omega)}Y = \operatorname{Diag}\big(df_{1i}(\Omega)Y\big)$$
$$-\big(df_{1i}(\Omega)Y\big)f_{1i}(\Omega)^\top - f_{1i}(\Omega)\big(df_{1i}(\Omega)Y\big)^\top \tag{A.9b}$$

and hence for any $S_i \in \mathbb{R}^{|J|}$ and $S = (\dots, S_i, \dots)^\top \in \mathbb{R}^{|I|\times|J|}$

$$(dR_{f_{1i}(\Omega)}Y)S_i = \big((dR_{f_1(\Omega)}Y)S\big)_i,\quad i\in I. \tag{A.9c}$$

Thus, analogous to (2.14), we obtain

$$(dR_{f_1(\Omega)}Y)S = \big(\dots, (dR_{f_{1i}(\Omega)}Y)S_i, \dots\big)^\top$$
$$= \operatorname{vec}_r^{-1}\Big(\big(\operatorname{Diag}(dR_{f_1(\Omega)})Y\big)\operatorname{vec}_r(S)\Big). \tag{A.9d}$$

Applying vec_r to both sides and comparing with (A.8), we conclude

$$d\operatorname{Diag}(R_{f_1(\Omega)})Y = \operatorname{Diag}(dR_{f_1(\Omega)}Y) \tag{A.9e}$$

which proves (3.21). □

Proof of Lemma 3.7 The mapping $\exp_p$ specified by (2.11e) satisfies $\exp_p = \exp_p\circ\Pi_0$ and a short computation [3, Appendix]) shows that the restriction $\exp_p|_{\mathcal{T}_0}$, again denoted by $\exp_p$, has the inverse

$$\exp_p^{-1}\colon \mathcal{S}\to\mathcal{T}_0,\qquad q\mapsto \Pi_0(\log q - \log p) \tag{A.10}$$

and consequently the differential

$$d\exp_p^{-1}(q)u = \Pi_0\Big(\frac{u}{q}\Big),\quad u\in\mathcal{T}_0. \tag{A.11}$$

For $W, \widetilde{W} \in \mathcal{W}$ and $V \in \mathcal{T}_0$, this differential applies componentwise, i.e.,

$$\big(d\exp_W^{-1}(\widetilde{W})V\big)_i = \Pi_0\Big(\frac{V_i}{\widetilde{W}_i}\Big),\quad i\in I. \tag{A.12}$$

Application to (3.5) yields for any $Y \in \mathcal{Y}_\Omega$ Eq. (3.31). □

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. OSDI (2016)
2. Al-Mohy, A.H., Higham, N.J.: Computing the action of the matrix exponential, with an application to exponential integrators. SIAM J. Sci. Comput. **33**(2), 488–511 (2011)
3. Åström, F., Petra, S., Schmitzer, B., Schnörr, C.: Image labeling by assignment. J. Math. Imaging Vis. **58**(2), 211–238 (2017)
4. Bader, P., Blanes, S., Casas, F.: Computing the matrix exponential with an optimized Taylor polynomial approximation. Mathematics **7**(12), 1174 (2019)
5. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey. J. Mach. Learn. Res. **18**, 1–43 (2018)
6. Benzi, M., Simoncini, V.: Approximation of functions of large matrices with Kronecker structure. Numer. Math. **135**(1), 1–26 (2017)
7. Graham, A.: Kronecker Products and Matrix Calculus: with Applications. Ellis Horwood Limited, New York (1981)
8. Higham, N.J.: Functions of Matrices: Theory and Computation. SIAM, Philadelphia (2008)
9. Horn, R.A., Johnson, C.R.: Topics in Matrix Analysis. Cambridge University Press, Cambridge (1991)
10. Hochbruck, M., Lubich, C.: On Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **34**(5), 1911–1925 (1997)
11. Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I, 3rd edn. Springer, London (2008)
12. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2010)
13. Hochbruck, M., Ostermann, A., Schweitzer, J.: Exponential Rosenbrock-type methods. SIAM J. Numer. Anal. **47**(1), 786–803 (2009)
14. Iserles, A., Munthe-Kaas, H.Z., Nørsett, S.P., Zanna, A.: Lie-group methods. Acta Numer. **09**, 215–365 (2000)
15. Kingma, D.P. , Ba, J.: Adam: a method for stochastic optimization (2015). arXiv:1412.6980
16. Kandolf, P., Koskela, A., Relton, S.D., Schweitzer, M.: Computing low-rank approximations of the Fréchet derivative of a matrix function using Krylov subspace methods. Numer. Linear Algebra Appl. **28**, e2401 (2021)
17. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. **45**(1), 3–49 (2003)
18. Niesen, J., Wright, W.M. : Algorithm 919: a Krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators. ACM Trans. Math. Softw. **38**(3), Article 22 (2012)
19. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, NIPS, vol. 32. Curran Associates Inc., Red Hook (2019)
20. Saad, Y.: Analysis of some Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **29**(1), 209–228 (1992)
21. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM, Philadelphia (2003)
22. Saad, Y.: Numerical Methods for Large Eigenvalue Problems, Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia (2011)
23. Schnörr, C.: Assignment flows. In: Grohs, P., Holler, M., Weinmann, A. (eds.) Variational Methods for Nonlinear Geometric Data and Applications, pp. 235–260. Springer, Berlin (2020)
24. Teschl, G.: Ordinary Differential Equations and Dynamical Systems, Graduate Studies in Mathematics, vol. 140. Amer. Math. Soc., Ann Arbor (2012)
25. Van Loan, C.F.: The ubiquitous Kronecker product. J. Comput. Appl. Math. **123**, 85–100 (2000)
26. Zeilmann, A., Petra, S., Schnörr, C.: Learning linear assignment flows for image labeling via exponential integration. In: Elmoataz, A., Fadili, J., Quéau, Y., Rabin, J., Simon, L. (eds.) Scale Space and Variational Methods in Computer Vision (SSVM), vol. 12679, pp. 385–397. LNCS (2021)
27. Zeilmann, A., Savarino, F., Petra, S., Schnörr, C.: Geometric numerical integration of the assignment flow. Inverse Probl. **36**(3), 034004 (2020)
28. Zern, A., Zeilmann, A., Schnörr, C.: Assignment Flows for Data Labeling on Graphs: Convergence and Stability. Inf. Geom. **5**, 355–404 (2022). https://doi.org/10.1007/s41884-021-00060-8

Alexander Zeilmann received his B.Sc. degree (2015) from TU Munich, Germany, with one semester at Utrecht University, Netherlands, and his M.Sc. degree in Mathematics (2017) from Heidelberg University, Germany. Currently, he is a Ph.D. student at Heidelberg University. His research interests include high-dimensional numerical analysis, dynamical systems on manifolds, machine learning and their applications to image analysis and mathematical modeling for medicine.

Stefania Petra received her B.Sc. degree in Mathematics and Computer Science in 2001 and her M.Sc. in Mathematics in 2003 from the Babes-Bolyai University of Cluj-Napoca. In 2006, she received her Ph.D. degree from the University of Würzburg in the field of numerical optimization. She continued working as a research fellow in the University of Mannheim and Heidelberg in the field of mathematical image processing. During 2013–2015, she was a Margarete von Wrangell Fellow within the postdoctoral lecture qualification program of the Ministry of Science, Research and Arts of the state of Baden Württemberg in Germany. Since 2015, she is an Assistant Professor at

Heidelberg University where she is leading the Mathematical Imaging Group at the Institute of Applied Mathematics. Her research interests include mathematical models and computational approaches for data analysis and machine learning using variational methods, information geometry, sparsity representation and numerical optimization with an emphasis on imaging applications.

Christoph Schnörr received his degrees from the Technical University of Karlsruhe (1991) and the University of Hamburg (1998), respectively. He became full professor at the University of Mannheim in 1998. In 2008, he joined the Heidelberg University where he is heading the Image and Pattern Analysis Group at the Institute of Applied Mathematics. From 2005–2014, he was co-Editor in Chief of the International Journal of Computer Vision (Springer). At present, he is member of the Editorial Boards of the Journal of Mathematical Imaging and Vision (JMIV, Springer) and the SIAM Journal of Imaging Sciences (SIIMS, SIAM). He serves as member on the steering board of the cluster of excellence STRUCTURES and in the Interdisciplinary Center for Scientific Computing (IWR), both at the Heidelberg University. His research focuses on mathematical models of image analysis and machine learning, based on differential geometry, dynamical systems on manifolds and numerical optimization.