Journal of Mathematical Imaging and Vision (2021) 63:186–215
https://doi.org/10.1007/s10851-020-00977-2

Learning Adaptive Regularization for Image Labeling Using Geometric Assignment

Ruben Hühnerbein[1] · Fabrizio Savarino[1] · Stefania Petra[2] · Christoph Schnörr[1]

Received: 22 October 2019 / Accepted: 18 June 2020 / Published online: 6 August 2020

Abstract
We study the inverse problem of model parameter learning for pixelwise image labeling, using the linear assignment flow and training data with ground truth. This is accomplished by a Riemannian gradient flow on the manifold of parameters that determines the regularization properties of the assignment flow. Using the symplectic partitioned Runge–Kutta method for numerical integration, it is shown that deriving the sensitivity conditions of the parameter learning problem and its discretization commute. A convenient property of our approach is that learning is based on exact inference. Carefully designed experiments demonstrate the performance of our approach, the expressiveness of the mathematical model as well as its limitations, from the viewpoint of statistical learning and optimal control.

Keywords Image labeling · Assignment manifold · Assignment flow · Dynamical systems · Replicator equation · Evolutionary dynamics · Sensitivity analysis · Parameter learning · Adaptive regularization

1 Introduction

1.1 Overview and Scope

The *image labeling problem*, i.e., the problem to classify images pixelwise depending on the spatial context, has been thoroughly investigated during the last two decades using discrete graphical models. While the evaluation (inference) of such models is well understood [15], *learning the parameters* of such models has remained elusive, in particular for models with higher connectivity of the underlying graph. Various sampling-based and other approximation methods exist (cf. [28] and references therein), but the *relation* between approximations of the *learning problem* on the one hand, and approximations of the subordinate *inference problem* on the other hand, is less understood [22].

In this paper, we focus on parameter learning for contextual pixelwise image labeling based on the *assignment flow* introduced by [2]. In comparison with discrete graphical models, an antipodal viewpoint was adopted by [2] for the design of the assignment flow approach: Rather than performing *non-smooth convex outer* relaxation and programming, followed by *subsequent* rounding to integral solutions that is common when working with large-scale discrete graphical models, the assignment flow provides a *smooth nonconvex interior* relaxation that performs rounding to integral solutions *simultaneously*. Convergence and stability of the assignment flow have been studied in [26], and extensions to unsupervised scenarios are reported in [27,29]. In [12], it was shown that the assignment flow can emulate a given discrete graphical model in terms of smoothed local Wasserstein distances that evaluate the edge-based parameters of the graphical model. In comparison with established belief propagation iterations [23,24], the assignment flow driven by 'Wasserstein messages' [12] continuously takes into account basic constraints, which enables to compute good suboptimal solutions just by numerically integrating the flow in a proper way [25]. We refer to [20] for summarizing recent work based on the assignment flow and a discussion of further aspects.

In this paper, we ignore the connection to discrete graphical models and focus on the parameter learning problem for the assignment flow directly. This problem is raised in [2, Section 5 and Fig. 14]. The present paper provides a detailed solution. Adopting the *linear* assignment flow as introduced

✉ Ruben Hühnerbein
ruben.huehnerbein@iwr.uni-heidelberg.de

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

[2] Mathematical Imaging Group, Heidelberg University, Heidelberg, Germany

by [25] enables to cast the parameter estimation problem into the general form

$$\min_{p\in\mathcal{P}} \quad \mathcal{C}\big(x(T,p)\big) \tag{1.1a}$$

$$\text{s.t.} \quad \dot{x}(t) = f(x(t),p,t), \quad t\in[0,T], \tag{1.1b}$$

$$x(0) = x_0, \tag{1.1c}$$

where the parameters p determine the vector field of the linear assignment flow (1.1b) whose unique solution is evaluated at some point of time T by a suitable loss function (1.1a). This problem formulation has a range of advantages.

- *Inference* (labeling) that always defines a subroutine of a learning procedure can be carried out *exactly* by means of numerically solving (1.1b). In other words, errors of approximate inference (e.g., as they occur with graphical models) are absent and cannot compromise the effectiveness of parameter learning.
- In addition, *discretization effects* can be handled in the most convenient way: We show the operations of (i) deriving the optimality conditions of (1.1) and (ii) problem discretization *commute* if a proper numerical scheme is used.

As a result, we obtain a well-defined and relatively simple algorithm for parameter learning that is easy to implement and enables reproducible research. We report the results of a careful numerical evaluation in order to highlight the scope of our approach and its limitations.

We discuss in Sect. 1.2 our specific contributions that elaborate a related conference paper [13] through the content of Sect. 2 (sensitivity analysis, commutativity of diagram 2, numerical schemes), Sect. 4 (parameter estimation, algorithm), Sect. 5 (a range of experiments) and the 'Appendix' (proofs).

1.2 Related Work, Contribution and Organization

The task to optimize parameters of a dynamical system (1.1) is a familiar one in the communities of scientific computing and optimal control [4,21], but may be less known to the imaging community. Therefore, we provide the necessary background in Sect. 2.1.

Geometric numerical integration of ODEs on manifolds is a mature field as well [8]. Here, we have to distinguish between the integration of the assignment flow [25] and integration schemes for numerically solving (1.1). The task to design the latter schemes faces the 'optimize-then-discretize' versus 'discretize-then-optimize' dilemma. Conditions and ways to resolve this dilemma have been studied in the optimal control literature [7,18]. See also the recent survey [19] and references therein. We provide the corresponding background in Sects. 2.2 and 2.3 including a detailed proof of Theorem 7 that is merely outlined in [19]. The application to the linear assignment flow (Sect. 3) requires considerable work, taking into account that the state equation (1.1b) derives from the full nonlinear geometric assignment flow (Sect. 4). Section 4 concludes with specifying Algorithms 1 and 2 whose implementation realizes our approach.

From a more distant viewpoint, our work ties in with research on networks from a *dynamical systems* point of view that emanated from [11] in computer science and has also been promoted recently in mathematics [5]. The recent work [6], for example, studied stability issues of discrete-time network dynamics using techniques of numerical ODE integration. The authors adopted the discretize-then-differentiate viewpoint on the parameter estimation problem and suggested symplectic numerical integration in order to achieve better stability. As mentioned above, our work contrasts in that inference is always *exact*[1] during learning, unlike the more involved architecture of [6] where learning is based on *approximate* inference. Furthermore, in our case, symplectic numerical integration is a *consequence* of making the diagram of Fig. 2 (page 7) *commute*. This property qualifies our approach as a proper (though rudimentary) method of *optimal control* (cf. [18]).

We numerically evaluate our approach in Sect. 5 using three different experiments. The first experiment considers a scenario of two labels and images of binary letters. The results discussed in Sect. 5.1 illustrate the *adaptivity* of regularization by using *non-uniform* weights that are predicted for *novel unseen* image data. The second experiment uses a class of computer-generated random images such that learning the regularization parameters is *necessary* for accurately labeling each image pixelwise. It is demonstrated in Sect. 5.2 that, for each given ground truth labeling, the parameter estimation problem can be solved *exactly*. As a consequence, the performance of the assignment flow solely depends on the prediction map, i.e., the ability to map features extracted from *novel* data to proper weights as regularization parameters, using as examples both features and *optimal* parameters computed during the training phase. For statistical reasons, this task becomes feasible if the domain of the prediction map is restricted to *local* contexts, in terms of features observed within local windows. We discuss consequences for future work in Sect. 6. Finally, in Sect. 5.3, we conduct an experiment that highlights the remarkable model expressiveness of the assignment flow as well as limitations that result from learning *constant* parameters.

We conclude in Sect. 6.

[1] 'Exact' means that T is chosen sufficiently large such that the assignment $W(T)$ is almost integral, i.e., a labeling, according to the entropy-based criterion of [2, Section 3.3.4].

1.3 Basic Notation

For the clarity of exposition, we use general mathematical notation in Sect. 2 that should be standard, whereas specific notation related to the assignment flow is introduced in Sect. 3.

We set $[n] = \{1, 2, \dots, n\}$ for $n \in \mathbb{N}$ and $\mathbb{1}_n = (1, 1, \dots, 1)^\top \in \mathbb{R}^n$. For a matrix $A \in \mathbb{R}^{m \times n}$, the ith row vector is denoted by A_i, $i \in [m]$ and its transpose by $A^\top \in \mathbb{R}^{n \times m}$. $\langle a, b \rangle$ denotes the Euclidean inner product of $a, b \in \mathbb{R}^n$ and $\langle A, B \rangle = \sum_{i \in [n]} \langle A_i, B_i \rangle$ the (Frobenius) inner product between two matrices $A, B \in \mathbb{R}^{m \times n}$. The probability simplex is denoted by $\Delta_n = \{p \in \mathbb{R}^n : p_i \geq 0,\ i \in [n],\ \langle \mathbb{1}_n, p \rangle = 1\}$. Various orthogonal projections onto a convex set are generally denoted by Π and distinguished by a corresponding subscript, like $\Pi_n, \Pi_{\mathcal{P}}, \cdots$, etc.

The functions exp, log apply *componentwise* to strictly positive vectors $x \in \mathbb{R}^n_{++}$, e.g., $e^x = (e^{x_1}, \dots, e^{x_n})$, and similarly for strictly positive matrices. Likewise, if $x, y \in \mathbb{R}^n_{++}$, then we simply write

$$xy = (x_1 y_1, \dots, x_n y_n), \quad \frac{x}{y} = \left(\frac{x_1}{y_1}, \dots, \frac{x_n}{y_n}\right) \tag{1.2}$$

for the *componentwise* multiplication and division.

We assume the reader to be familiar with elementary notions of Riemannian geometry as found, e.g., in [14,16]. Specifically, given a Riemannian manifold $(\mathcal{M}, g)$ with metric g and a smooth function $f : \mathcal{M} \to \mathbb{R}$, the Riemannian gradient of f is denoted by grad f and given by

$$\langle \operatorname{grad} f, X \rangle_g = \mathrm{d} f(X), \quad \forall X \tag{1.3}$$

where X denotes any smooth vector field on $\mathcal{M}$, that returns the tangent vector $X_p \in T_p\mathcal{M}$ when evaluated at $p \in \mathcal{M}$. The right-hand side of (1.3) denotes the differential df of f, acting on X. More generally, for a map $F : \mathcal{M} \to \mathcal{N}$ between manifolds, we write $dF(p)[v] \in T_{F(p)}\mathcal{N}$, $p \in \mathcal{M}$, $v \in T_p\mathcal{M}$, if the base point p matters.

In the Euclidean case $f : \mathbb{R}^n \to \mathbb{R}$, the gradient is a column vector and denoted by ∂f. For $F : \mathbb{R}^n \to \mathbb{R}^m$, we identify the differential $dF \in \mathbb{R}^{m \times n}$ with the Jacobian matrix. If $x = (x_1, x_2)^\top \in \mathbb{R}^n = \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ with $n = n_1 + n_2$, then the Jacobian of $F(x) = F(x_1, x_2)$ with respect to the parameter vector x_i is denoted by $d_{x_i} F$, for $i = 1, 2$.

2 Sensitivity Analysis for Dynamical Systems

In this section, we consider the constrained optimization problem (1.1) with a smooth objective function $\mathcal{C} : \mathbb{R}^{n_x} \to \mathbb{R}$. The constraints are given by a general *initial value problem (IVP)*, which consist of a system of ordinary differential equations (ODEs) (1.1b) that is parametrized by a vector $p \in \mathcal{P} \subset \mathbb{R}^{n_p}$ and an initial value $x_0 \in \mathbb{R}^{n_x}$. To ensure existence, uniqueness and continuous differentiability of the solution trajectory $x(t)$ on the whole time horizon $[0, T]$, we assume that $f(\cdot, p, \cdot)$ of (1.1b) is Lipschitz continuous on $\mathbb{R}^{n_x} \times [0, T]$, for any p.

Since we assume the initial value x_0 and the time horizon $[0, T]$ to be fixed, the objective function (1.1a)

$$\Phi(p) := \mathcal{C}(x(T, p)) \tag{2.1}$$

effectively is a function of parameter p, i.e., $\Phi : \mathbb{R}^{n_p} \to \mathbb{R}$. In order to solve (1.1) with a gradient-based method, we have to compute the *gradient*

$$\partial_p \Phi(p) = d_p x(T, p)^\top \partial_x \mathcal{C}(x(T, p)). \tag{2.2}$$

The term $d_p x(T, p)$—called *sensitivity*—measures the sensitivity of the solution trajectory $x(t)$ at time T with respect to changes in the parameter p. Two basic approaches for determining (2.2) are stated in Sect. 2.1, and we briefly highlight why using one of them, the *adjoint approach*, is advantageous for computing sensitivities. In Sect. 2.2, we recall symplectic Runge–Kutta methods and conditions for preserving quadratic invariants. The latter property relates to the derivation of a class of numerical methods such that evaluating (2.2), which derives from the time-continuous problem (1.1), is *identical* to first discretizing (1.1) followed by computing the corresponding derived expression (2.2). Two specific instances of the general numerical scheme are detailed in Sect. 2.4.

2.1 Sensitivity Analysis

In this section, we describe how the sensitivity $d_p x(T, p)$ can be determined by solving one of the two initial value problems defined below: the *variational system* and the *adjoint system*.

Theorem 1 (Variational system; [10, Ch. I.14, Thm. 14.1]) *Suppose the derivatives $d_x f$ and $d_p f$ exist and are continuous in the neighborhood of the solution $x(t)$ for $t \in [0, T]$. Then, the sensitivity with respect to the parameters*

$$d_p x(T, p) =: \delta(T) \tag{2.3}$$

exists, is continuous and satisfies the variational system

$$\dot{\delta}(t) = d_x f(x(t), p, t)\delta(t) + d_p f(x(t), p, t), \tag{2.4a}$$
$$\delta(0) = 0 \in \mathbb{R}^{n_x \times n_p}, \tag{2.4b}$$

with $t \in [0, T]$ and $\delta(t) \in \mathbb{R}^{n_x \times n_p}$. If the initial value $x(0)$ (1.1c) depends on the parameters p, the initial value (2.4b) has to be adjusted as $\delta(0) = d_p x(0)$.

Proof A detailed proof can be found in [10, Ch. I.14, Thm. 14.1]. In order to make this paper self-contained, a sketch of the argument follows.

The integral representation of the solution to (1.1b) is given by $x(t, p) = x_0 + \int_0^t f(x(s), p, s)\mathrm{d}s$. Differentiating with respect to p and exchanging integration and differentiation by the theorem of Lebesgue yields

$$d_p x(t, p) = d_p x_0 + \int_0^t d_p\big(f(x(s), p, s)\big)\mathrm{d}s \tag{2.5a}$$

$$= d_p x_0 + \int_0^t \Big(d_x f(x(s), p, s) d_p x(s, p) + d_p f(x(s), p, s)\Big)\,\mathrm{d}s. \tag{2.5b}$$

Substituting $\delta(t) = d_p x(t, p)$ gives

$$\delta(t) = \delta_0 + \int_0^t d_x f(x(s), p, s)\delta(s) + d_p f(x(s), p, s)\mathrm{d}s, \tag{2.6}$$

which is the integral representation of the trajectory $\delta(t)$ solving (2.4). □

For the computation of the variational system (2.4), the solution $x(t)$ is required. Since the variational system (2.4) is a matrix-valued system of dimension $n_x \times n_p$, the size of the system grows with the number of parameters n_p. For small n_p, solving the variational system is efficient. In practice, it can be simultaneously integrated numerically together with system (1.1b).

Theorem 2 (Adjoint system) *Suppose that the derivatives $d_x f$ and $d_p f$ exist and are continuous in the neighborhood of the solution $x(t)$ for $t \in [0, T]$. Then, the sensitivity with respect to the parameters is given by*

$$d_p x(T, p)^\top = \int_0^T d_p f(x(t), p, t)^\top \lambda(t) dt, \tag{2.7}$$

where $\lambda(t) \in \mathbb{R}^{n_x \times n_x}$ solves the adjoint system

$$\dot{\lambda}(t) = -d_x f(x(t), p, t)^\top \lambda(t), \quad t \in [0, T], \tag{2.8a}$$

$$\lambda(T) = I \in \mathbb{R}^{n_x \times n_x}. \tag{2.8b}$$

Proof This proof is elaborated on in a broader context in Sect. 2.3. □

Similar to the variational system of Theorem 1, solving the adjoint system (2.8) requires the solution $x(t)$. The adjoint system is matrix-valued of dimension $n_x \times n_x$, in contrast to the variational system which is of dimension $n_x \times n_p$. Thus, if $n_p \gg n_x$ as will be the case in our scenario, it is more efficient to solve (2.8) instead of (2.4). Another major difference is that the adjoint system is defined *backwards* in time, starting from the endpoint T. This has important computational advantages for our setting. In view of the required gradient (2.2), we are not interested in the full sensitivity but rather in the derivative along the direction $\eta := \partial_x \mathcal{C}(x(T, p))$, i.e., $d_p x(T, p)^\top \eta$. This can be achieved by exploiting the structure of the adjoint system, by multiplying (2.8) from the right by η and setting $\bar{\lambda}(t) := \lambda(t)\eta$. The resulting IVP is again an adjoint system, no longer being matrix-valued but vector-valued $\bar{\lambda}(t) \in \mathbb{R}^{n_x}$, with $\bar{\lambda}(T) = \eta \in \mathbb{R}^{n_x}$. Thus, from now on, we consider the latter case and denote $\bar{\lambda}(t)$ again by $\lambda(t)$, which is vector-valued.

As a consequence, we will focus on the adjoint system (2.8) in the remainder of this paper. In particular, (2.7) will be used to estimate parameters p by solving (1.1) using a gradient descent flow. This requires to solve the adjoint system numerically. However, a viable alternative to this 'optimize-then-discretize' approach is to reverse this order, that is, to discretize problem (1.1) first and then to derive a corresponding *time-discrete* adjoint system. It turns out that both ways are equivalent if a proper class of numerical integration scheme is chosen for discretizing the system in time. This will be shown in Sect. 2.3 after collecting required background material in Sect. 2.2.

2.2 Symplectic Partitioned Runge–Kutta Methods

In this section, we recall basic concepts of numerical integration from [8,19] in order to prepare Sect. 2.3. Symplectic schemes are typically applied to Hamiltonian systems in order to conserve certain quantities, often with a physical background. The pseudo-Hamiltonian defined below by (2.19) will play a similar role, albeit there is no physical background for our concrete scenario to be studied in subsequent sections.

A general *s-stage Runge–Kutta (RK) method* with $s \in \mathbb{N}$ is given by [9]

$$x_{n+1} = x_n + h_n \sum_{i=1}^{s} b_i k_{n,i}, \tag{2.9a}$$

$$k_{n,i} = f(X_{n,i}, p, t_n + c_i h_n), \tag{2.9b}$$

$$X_{n,i} = x_n + h_n \sum_{j=1}^{s} a_{ij} k_{n,j}, \tag{2.9c}$$

where $h_n = t_{n+1} - t_n$ in (2.9a) denotes a step size. The coefficients $a_{ij}, b_i, c_i \in \mathbb{R}$ can be arranged in a so-called

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} = \begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss-1} & \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s \end{array}$$

Fig. 1 ABOVE: Butcher tableau of a general s-stage Runge–Kutta method. BELOW: Butcher tableau of a s-stage explicit Runge–Kutta method

Butcher tableau (Fig. 1), with entries a_{ij} defining the Runge–Kutta matrix A.

Lower-triangular Runge–Kutta matrices A, i.e.,

$$a_{ij} = 0 \quad \text{for} \quad j \geq i, \tag{2.10}$$

result in *explicit* RK schemes and in *implicit* RK schemes otherwise. Implicit Runge–Kutta methods are well suited for integrating numerically stiff ODEs, but are also significantly more complex than explicit ones. Since (2.9b) cannot be solved explicitly, a system of algebraic equations has to be solved. The following theorem specifies the conditions under which a solution for these equations exists.

Theorem 3 (Existence of a numerical solution; [9, Ch. II, Thm. 7.2]) *For any $p \in \mathbb{R}^{n_p}$ let $f(\cdot, p, \cdot)$ of (1.1b) be continuous and satisfy a Lipschitz condition on $\mathbb{R}^{n_x} \times [0, T]$ with constant L, independent of p. If*

$$h < \frac{1}{L \max_{i=1,\dots,s} \sum_{j=1}^{s} |a_{ij}|} \tag{2.11}$$

there exists a unique solution of (2.9), which can be obtained by fixed-point iteration. If $f(x, p, t)$ is q times differentiable, the functions $k_{n,i}$ (as functions of h) are also in C^q.

Proof A detailed proof can be found in [9, Ch. II, Thm. 7.2]. □

Suppose that the given system (1.1b) is *partitioned* into two parts with $x = (y^\top, z^\top)^\top$, $f = (f_1^\top, f_2^\top)^\top$ and

$$\dot{y} = f_1(y, z, t), \tag{2.12a}$$
$$\dot{z} = f_2(y, z, t). \tag{2.12b}$$

Partitioned Runge–Kutta (PRK) methods integrate (2.12) using two different sets of coefficients

$$a_{ij}, b_i, c_i \in \mathbb{R} \ \text{ for } \ (2.12a), \tag{2.13a}$$
$$\overline{a}_{ij}, \overline{b}_i, \overline{c}_i \in \mathbb{R} \ \text{ for } \ (2.12b). \tag{2.13b}$$

The following theorems state conditions under which RK methods preserve certain quantities that should be invariant under the flow of the system that is integrated numerically. In this sense, such RK schemes are called *symplectic*.

Theorem 4 (Symplectic Runge–Kutta method; [8, Ch. VI, Thm. 7.6 and 7.10]) *Assume that the system (1.1b) has a quadratic invariant I, i.e., $I(\cdot, \cdot)$ is a real-valued bilinear mapping such that $(d/dt)I(x(t), x(t)) = 0$, for each t and x_0. If the coefficients of a Runge–Kutta method (2.9) satisfy*

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \tag{2.14}$$

then the value $I(x_n, x_n)$ does not depend on n.

Theorem 5 *(*Symplectic PRK method; [19, Thm. 2.4 and 2.6]*) Assume that $S(\cdot, \cdot)$ is a real-valued bilinear mapping such that $(d/dt)S(y(t), z(t)) = 0$ for each t and x_0 of the solution $x(t) = [y(t)^\top, z(t)^\top]^\top$ of (2.12). If the coefficients of the partitioned Runge–Kutta method (2.13) satisfy*

$$b_i \overline{a}_{ij} - b_i \overline{b}_j + \overline{b}_j a_{ji} = 0, \quad \overline{b}_i = b_i, \quad \overline{c}_i = c_i, \tag{2.15}$$

then the value $S(y_n, z_n)$ does not depend on n.

Remark 1 Assume the first set of Runge–Kutta coefficients are given and denoted by a_{ij}, b_i, c_i with indices $i, j \in [s]$. This method is used for the first n-variables (2.12a). Furthermore, let $b_i \neq 0$ for all stages $i \in [s]$. In view of condition (2.15), we can construct a *symplectic* PRK method by choosing

$$\overline{a}_{ij} := b_j - b_j a_{ji}/b_i, \quad \overline{b}_i := b_i, \quad \overline{c}_i := c_i, \tag{2.16}$$

as coefficients for the second n-variables (2.12b). This construction results in an overall *symplectic PRK method* of the partitioned system (2.12).

2.3 Computing Adjoint Sensitivities

There are two basic approaches for computing (2.2), the *differentiate-then-discretize* approach and the *discretize-then-differentiate* approach. Figure 2 illustrates both approaches by paths colored with blue and violet, respectively. Details are worked out in this section. Our main objective is to make this diagram *commutative* by adopting a class of numerical schemes as outlined in the preceding section.

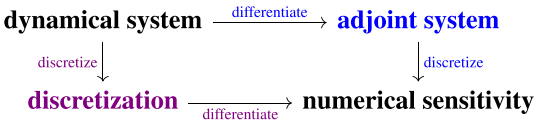


Fig. 2 Illustration of the methodological part of this section. Our approach satisfies the commuting diagram, i.e., *identical* results are obtained either if the continuous problem is differentiated first and then discretized (blue path), or the other way around (violet path) (Color figure online)

In the following, we drop the dependency of $x(t)$ on the parameter p, to simplify notation by just writing $x(t)$. The following theorem details the blue path of Fig. 2.

Theorem 6 (Adjoint sensitivity: differentiate-then-discretize) *The gradient* (2.2) *of the objective function* (2.1) $\Phi(p) = C(x(T))$ *of problem* (1.1) *with respect to the parameter* p *is given by*

$$\partial\Phi(p) = \int_0^T d_p f(x(t), p, t)^\top \lambda(t) dt, \tag{2.17}$$

where $x(t), \lambda(t)$ *solve the two-point boundary value problem*

$$\dot{x}(t) = f(x(t), p, t), \qquad x(0) = x_0, \tag{2.18a}$$
$$\dot{\lambda}(t) = -d_x f(x(t), p, t)^\top \lambda(t), \qquad \lambda(T) = \partial\mathcal{C}(x(T)). \tag{2.18b}$$

In terms of the pseudo-Hamiltonian

$$H(x, \lambda, p, t) = \langle f(x, p, t), \lambda \rangle, \tag{2.19}$$

the system has the following form

$$\dot{x}(t) = d_\lambda H(x, \lambda, p, t), \qquad x(0) = x_0, \tag{2.20a}$$
$$\dot{\lambda}(t) = -d_x H(x, \lambda, p, t), \qquad \lambda(T) = \partial\mathcal{C}(x(T)). \tag{2.20b}$$

Proof See Appendix 1. □

Remark 2 The presence of the pseudo-Hamiltonian (2.19) suggests to use either a *symplectic* RK method or a *symplectic* PRK method to integrate the boundary value problem (2.18). In view of Remark 1, we can use a general RK method with coefficients a_{ij}, b_i, c_i for $i, j \in [s]$ for the first variables (2.18a), and another RK method with $\overline{a}_{ij}, \overline{b}_i, \overline{c}_i$ for $i, j \in [s]$ satisfying (2.16) for the second variables (2.18b). Again, this construction results in an overall *symplectic PRK method* of the boundary problem (2.18). Note that (2.18a) is independent of variable λ. Due to this property, we can solve (2.18) sequentially in practice, i.e., we first integrate (2.18a) and afterward (2.18b).

Now, we consider the alternative violet path of Fig. 2. Applying a RK method with step-sizes $h_n = t_{n+1} - t_n > 0$ to problem (1.1) results in the *nonlinear optimization problem*

$$\min_{p\in\mathcal{P}} \mathcal{C}\big(x_N(p)\big) \tag{2.21a}$$
$$\text{s.t.} \quad x_{n+1} = x_n + h_n \sum_{i=1}^s b_i k_{n,i}, \tag{2.21b}$$
$$k_{n,i} = f(X_{n,i}, p, t_n + c_i h_n), \quad i \in [s], \tag{2.21c}$$
$$X_{n,i} = x_n + h_n \sum_{j=1}^s a_{ij} k_{n,j}, \quad i \in [s], \tag{2.21d}$$
$$x_0 = x(0), \tag{2.21e}$$

with $n = 0, \ldots, N-1$.

Next, we *differentiate* this problem and state the result in the following theorem.

Theorem 7 (Adjoint sensitivity: discretize-then-differentiate) *Suppose the step-size* h_n *satisfies condition* (2.11). *Then, the gradient of the objective function* $\Phi(p) = \mathcal{C}(x_N(p))$ *from* (2.21) *with respect to parameter* p *is given by*

$$\partial\Phi(p) = \sum_{n=0}^{N-1} h_n \sum_{i=1}^s \overline{b}_i \left(d_p f(X_{n,i}, p, t_n + \overline{c}_i h_n)\right)^\top \Lambda_{n,i}, \tag{2.22}$$

where the discrete adjoint variables are given by

$$\lambda_{n+1} = \lambda_n + h_n \sum_{i=1}^s \overline{b}_i \ell_{n,i}, \tag{2.23a}$$
$$\ell_{n,i} = -d_x f(X_{n,i}, p, t_n + \overline{c}_i h_n)^\top \Lambda_{n,i}, \tag{2.23b}$$
$$\Lambda_{n,i} = \lambda_n + h_n \sum_{j=1}^s \overline{a}_{ij} \ell_{n,j}, \tag{2.23c}$$

with $n = 0, \ldots, N-1$, $i \in [s]$ *and step-size* $h_n = t_{n+1} - t_n$. *The internal stages* $X_{n,i}$ *are given by* (2.21d). *This scheme is a general RK method* (2.9) *applied to the adjoint system* (2.18b) *with coefficients*

$$\overline{a}_{ij} = b_j - \frac{a_{ji} b_j}{b_i}, \quad \overline{b}_i = b_i, \quad \overline{c}_i = c_i, \tag{2.24}$$

for $b_i \neq 0$ *and* $i, j = [s]$.

Proof An outline of the proof can be found in [19, Thm. 3.6]. Following the suggested outline, we provide a detailed proof in Appendix 2. □

Remark 3 Comparing the statements of Theorem 6 and Theorem 7, we see that the formula of the discrete sensitivity (2.22) is an approximation of the integral (2.17) with quadrature

Table 1 Symplectic PRK coefficients induced by the explicit Euler method

$$\begin{array}{c|c} c_1 & a_{11} \\ \hline & b_1 \end{array} = \begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

forward coefficients

$$\begin{array}{c|c} \overline{c}_1 & \overline{a}_{11} \\ \hline & \overline{b}_1 \end{array} = \begin{array}{c|c} 0 & 1 \\ \hline & 1 \end{array}$$

backward coefficients

weights b_i. Furthermore, we observe that the coefficients of the constructed PRK method (2.16) coincide with the derived coefficients (2.24). Thus, by restricting the class of numerical schemes to *symplectic PRK methods* satisfying (2.15), the approaches due to the Theorem 6 (and Remark 2) and Theorem 7 are mathematically identical, and the diagram depicted in Fig. 2 *commutes*.

2.4 Two Specific Numerical Schemes

We complement and illustrate the general result of the preceding section by specifying two numerical RK schemes of different order, the basic explicit Euler method and Heun's method, respectively.

2.4.1 Adjoint Sensitivity: Explicit Euler Method

We integrate the forward dynamic (2.18a) with the explicit Euler method [10]. The straightforward use of (2.16) leads to another Runge–Kutta method for integrating the adjoint system (2.18b). The forward and backward coefficients of this overall symplectic partitioned Runge–Kutta method are then given by Table 1.

By substituting the backward coefficients $\overline{a}_{11}, \overline{b}_1$ and $\overline{c}_1$ into (2.23), we derive the concrete formulas of the discrete adjoint method

$$\lambda_{n+1} = \lambda_n + h_n \ell_{n,1} \tag{2.25a}$$
$$\ell_{n,1} = -\partial_x f(X_{n,1}, t_n)^\top \Lambda_{n,1} \tag{2.25b}$$
$$\Lambda_{n,1} = \lambda_n + h_n \ell_{n,1}. \tag{2.25c}$$

Note that (2.25c) coincides with (2.25a), that is by traversing from $n+1$ to n, we can rewrite (2.25) in the form

$$\lambda_n = \lambda_{n+1} + h_n d_x f(X_{n,1}, t_n)^\top \lambda_{n+1}. \tag{2.26}$$

Formula (2.22) for the gradient of $\Phi(p) = \mathcal{C}(x_N(p))$ from (2.21) reads

$$\partial\Phi(p) = \sum_{n=0}^{N-1} h_n d_p f(X_{n,1}, t_n)^\top \lambda_{n+1}. \tag{2.27}$$

Table 2 Symplectic PRK coefficients induced by Heun's method

$$\begin{array}{c|cc} c_1 & a_{11} & a_{12} \\ c_2 & a_{21} & a_{22} \\ \hline & b_1 & b_2 \end{array} = \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

forward coefficients

$$\begin{array}{c|cc} \overline{c}_1 & \overline{a}_{11} & \overline{a}_{12} \\ \overline{c}_2 & \overline{a}_{21} & \overline{a}_{22} \\ \hline & \overline{b}_1 & \overline{b}_2 \end{array} = \begin{array}{c|cc} 0 & 1/2 & -1/2 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

backward coefficients

2.4.2 Adjoint Sensitivity: Heun's Method

We integrate the forward dynamic (2.18a) with Heun's method [10]. The straightforward use of (2.16) leads to another Runge–Kutta method for integrating the adjoint system (2.18b). The forward and backward coefficients of this overall symplectic partitioned Runge–Kutta method are then given by Table 2

Although the butcher tableau of the backward coefficients (see Table 2, right matrix) is completely dense, the final update formulas are *explicit* by traversing backward in time, as we will show below. Again, we derive the concrete formulas of the discrete adjoint method by substituting the backward coefficients into (2.23)

$$\lambda_{n+1} = \lambda_n + h_n\big(\tfrac{1}{2}\ell_{n,1} + \tfrac{1}{2}\ell_{n,2}\big) \tag{2.28a}$$
$$\ell_{n,1} = -d_x f(X_{n,1}, t_n)^\top \Lambda_{n,1} \tag{2.28b}$$
$$\ell_{n,2} = -d_x f(X_{n,2}, t_n + h_n)^\top \Lambda_{n,2} \tag{2.28c}$$
$$\Lambda_{n,1} = \lambda_n + h_n\big(\tfrac{1}{2}\ell_{n,1} - \tfrac{1}{2}\ell_{n,2}\big) \tag{2.28d}$$
$$\Lambda_{n,2} = \lambda_n + h_n\big(\tfrac{1}{2}\ell_{n,1} + \tfrac{1}{2}\ell_{n,2}\big). \tag{2.28e}$$

Note that (2.28e) coincides with (2.28a), which implies the equations

$$\lambda_{n+1} = \Lambda_{n,2}, \quad \text{and} \tag{2.29a}$$
$$\ell_{n,2} = -d_x f(X_{n,2}, t_n + h_n)^\top \lambda_{n+1}. \tag{2.29b}$$

Using (2.29), we reformulate (2.28d)

$$\Lambda_{n,1} = \lambda_n + h_n\big(\tfrac{1}{2}\ell_{n,1} - \tfrac{1}{2}\ell_{n,2}\big) \tag{2.30a}$$
$$= \lambda_n + h_n\big(\tfrac{1}{2}\ell_{n,1} - \tfrac{1}{2}\ell_{n,2}\big) + (h_n\ell_{n,2} - h_n\ell_{n,2}) \tag{2.30b}$$
$$= \lambda_n + h_n\big(\tfrac{1}{2}\ell_{n,1} + \tfrac{1}{2}\ell_{n,2}\big) - h_n\ell_{n,2} \tag{2.30c}$$
$$\overset{(2.28a)}{=} \lambda_{n+1} - h_n\ell_{n,2} \tag{2.30d}$$
$$\overset{(2.29)}{=} \lambda_{n+1} + h_n d_x f(X_{n,2}, t_n + h_n)^\top \lambda_{n+1}. \tag{2.30e}$$

Formula (2.30e) is an explicit Euler step traversing backward from $n+1$ to n. Thus, we can rewrite the overall scheme (2.28) as

$$\tilde{\lambda}_n = \lambda_{n+1} + h_n d_x f(X_{n,2}, t_n + h_n)^\top \lambda_{n+1} \tag{2.31a}$$

$$\lambda_n = \lambda_{n+1} + \frac{h_n}{2}\Big(d_x f(X_{n,1}, t_n)^\top \tilde{\lambda}_n + d_x f(X_{n,2}, t_n + h_n)^\top \lambda_{n+1}\Big). \tag{2.31b}$$

Again, this is an *explicit method traversing backward from* $n+1$ to n. Formula (2.22) for the gradient of $\Phi(p) = \mathcal{C}(x_N(p))$ from (2.21) has the form

$$\partial_p \mathcal{C}(x_N) = \sum_{n=0}^{N-1} \frac{h_n}{2}\Big(d_p f(X_{n,1}, t_n)^\top \tilde{\lambda}_n + d_p f(X_{n,2}, t_n + h_n)^\top \lambda_{n+1}\Big). \tag{2.32}$$

Remark 4 Both example schemes (explicit Euler & Heun's method) have in common that the final update schemes of the adjoint integration can be solved *explicitly* by traversing backward from $n+1 \to n$. Note that this holds for these two specific numerical schemes, but may not hold in general for other higher-order schemes.

3 Image Labeling Using Geometric Assignment

In this section, we summarize material from [2,25] required in the remainder of this paper. See also [20] for a discussion of the assignment flow approach in a broader context.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a given undirected graph with $m := |\mathcal{V}|$ vertices and let

$$f\colon \mathcal{V} \to \mathcal{F}, \quad i \mapsto f_i \in \mathcal{F} \quad \text{with} \quad f(\mathcal{V}) =: \mathcal{F}_\mathcal{V} \subset \mathcal{F} \tag{3.1}$$

be data on the graph given in a metric space $(\mathcal{F}, d)$. We call $\mathcal{F}_\mathcal{V}$ *image data* given by *features* f_i extracted from a raw image at pixel $i \in \mathcal{V}$ in a preprocessing step. Along with f, we assume prototypical data

$$\mathcal{X} = \{\ell_1, \ldots, \ell_n\} \subset \mathcal{F} \tag{3.2}$$

to be given, henceforth called *labels*. Each label ℓ_j represents the data of class j. *Image labeling* denotes the problem of finding an assignment $\mathcal{V} \to \mathcal{X}$ assigning class labels to nodes depending on the image data $\mathcal{F}_\mathcal{V}$ and the local context encoded by the graph structure $\mathcal{G}$. We refer to [12] for more details and background on the image labeling problem.

$\mathcal{G}$ may be a grid graph (with self-loops) as in low-level image processing or a less structured graph, with arbitrary connectivity in terms of the neighborhoods

$$\mathcal{N}_i = \{k \in \mathcal{V}\colon ik = ki \in \mathcal{E}\} \cup \{i\}, \quad i \in \mathcal{V}, \tag{3.3}$$

where ik is a shorthand for the undirected edge $\{i,k\} \in \mathcal{E}$. We require these neighborhoods to satisfy the relations

$$k \in \mathcal{N}_i \quad \Leftrightarrow \quad i \in \mathcal{N}_k, \qquad \forall i, k \in \mathcal{I}. \tag{3.4}$$

We associate with each neighborhood $\mathcal{N}_i$ from (3.3) weights $\omega_{ik} \in \mathbb{R}$ for all $k \in \mathcal{N}_i$, satisfying

$$\omega_{ik} > 0 \quad \text{and} \quad \sum_{k \in \mathcal{N}_i} \omega_{ik} = 1, \quad \text{for all } i \in \mathcal{V}. \tag{3.5}$$

These weights parametrize the regularization property of the assignment flow below. Learning these weights from the given data is the subject of the remainder of this paper.

3.1 Assignment Manifold

The probabilistic assignment of labels $\mathcal{X}$ at one node $i \in \mathcal{V}$ is represented by the manifold of discrete probability distributions with full support

$$\mathcal{S}_n := \{p \in \Delta_n\colon p > 0\} \tag{3.6}$$

with constant tangent space for all $p \in \mathcal{S}_n$

$$T_p\mathcal{S}_n = \{v \in \mathbb{R}^n\colon \langle \mathbb{1}, v\rangle = 0\} =: T_n. \tag{3.7}$$

Throughout this paper, we only work with T_n. The probability space $\mathcal{S}$ is turned into a Riemannian manifold $(\mathcal{S}_n, g)$ by equipping it with the Fisher–Rao (information) metric

$$g_p(u,v) := \sum_{j \in [n]} \frac{u_j v_j}{p_j}, \tag{3.8}$$

with $u, v \in T_n$ and $p \in \mathcal{S}_n$. Furthermore, we have the uniform distribution of labels

$$\mathbb{1}_{\mathcal{S}_n} := \frac{1}{n}\mathbb{1}_n \in \mathcal{S}_n, \qquad (\textit{barycenter}) \tag{3.9}$$

and the orthogonal projection onto the tangent space with respect to the standard Euclidean structure of $\mathbb{R}^n$

$$\Pi_n\colon \mathbb{R}^n \to T_n, \quad \Pi_n := I - \mathbb{1}_{\mathcal{S}_n}\mathbb{1}^\top \tag{3.10}$$

with $\ker(\Pi_n) = \mathbb{R}\mathbb{1}_n$. The *replicator operator* for $p \in \mathcal{S}_n$ is given by the linear map

$$R_p\colon \mathbb{R}^n \to T_n, \quad R_p := \operatorname{Diag}(p) - pp^\top, \tag{3.11}$$

satisfying

$$R_p = R_p \Pi_n = \Pi_n R_p. \tag{3.12}$$

The *Riemannian gradient* of a smooth function $f : \mathcal{S}_n \to \mathbb{R}$ is denoted by $\operatorname{grad} f : \mathcal{S}_n \to T_n$ and relates to the Euclidean gradient ∂f by [2, Prop. 1] as

$$\operatorname{grad} f(p) = R_p \partial f(p) \quad \text{for } p \in \mathcal{S}_n. \tag{3.13}$$

Adopting the α-connection with $\alpha = 1$, also called *e-connection*, from information geometry [1, Section 2.3], [3], the exponential map based on the corresponding affine geodesics reads

$$\begin{aligned} &\operatorname{Exp} : \mathcal{S}_n \times T_n \to \mathcal{S}_n, \\ &(p, v) \mapsto \operatorname{Exp}_p(v) = \frac{p e^{\frac{v}{p}}}{\langle p, e^{\frac{v}{p}} \rangle} \end{aligned} \tag{3.14a}$$

$$\begin{aligned} &\operatorname{Exp}^{-1} : \mathcal{S}_n \times \mathcal{S}_n \to T_n, \\ &(p, q) \mapsto \operatorname{Exp}_p^{-1}(q) = R_p \log \frac{q}{p}. \end{aligned} \tag{3.14b}$$

Specifically, we define for all $p \in \mathcal{S}_n$

$$\begin{aligned} &\exp_p : T_n \to \mathcal{S}_n, \\ &z \mapsto \operatorname{Exp}_p \circ R_p(z) = \frac{p e^z}{\langle p, e^z \rangle}, \end{aligned} \tag{3.15a}$$

$$\exp_p^{-1} : \mathcal{S}_n \to T_n, \quad q \mapsto \Pi_n \log \frac{q}{p}. \tag{3.15b}$$

Applying the map $\exp_p$ to a vector in $\mathbb{R}^n = T_n \oplus \mathbb{R}\mathbb{1}_n$ does not depend on the constant component of the argument, due to (3.12).

Remark 5 The map Exp corresponds to the e-connection of information geometry [1], rather than to the exponential map of the Riemannian connection. Accordingly, the affine geodesics (3.14a) are not length-minimizing with respect to the Riemannian structure. But locally, they provide a close approximation [2, Prop. 3] and are more convenient for numerical computations.

Global label assignments on the whole set of nodes $\mathcal{V}$ are represented as points on the *assignment manifold*, given by the product

$$\mathcal{W} := \mathcal{S}_n \times \cdots \times \mathcal{S}_n \qquad (m = |\mathcal{V}| \text{ times}) \tag{3.16}$$

with constant *tangent space*

$$\mathcal{T}_{\mathcal{W}} := T_n \times \cdots \times T_n \quad (m = |\mathcal{V}| \text{ times}) \tag{3.17}$$

and Riemannian structure $(\mathcal{W}, g)$ given by the Riemannian product metric. We identify $\mathcal{W}$ with the embedding into $\mathbb{R}^{m \times n}$

$$\begin{aligned} \mathcal{W} = \{ W \in \mathbb{R}^{m \times n} : \, & W \mathbb{1}_n = \mathbb{1}_m \text{ and } W_{ij} > 0 \\ & \text{for all } i \in [m], \;\; j \in [n] \}. \end{aligned} \tag{3.18}$$

Thus, points $W \in \mathcal{W}$ are row-stochastic matrices $W \in \mathbb{R}^{m \times n}$ with row vectors $W_i \in \mathcal{S}_n$, $i \in \mathcal{V}$ representing the label assignments for every $i \in \mathcal{V}$. Due to this embedding of $\mathcal{W}$, the tangent space $\mathcal{T}_{\mathcal{W}}$ can be identified with

$$\mathcal{T}_{\mathcal{W}} = \{ V \in \mathbb{R}^{m \times n} : \, V \mathbb{1}_n = 0 \} \tag{3.19}$$

and therefore for $V \in \mathcal{T}_{\mathcal{W}}$ every row vector V_i is contained in T_n for every $i \in \mathcal{V}$. The global uniform distribution, given by the uniform distribution in every row, again called *barycenter*, is denoted by

$$\mathbb{1}_{\mathcal{W}} := (\mathbb{1}_{\mathcal{S}_n}, \ldots, \mathbb{1}_{\mathcal{S}_n}) = \mathbb{1}_m \mathbb{1}_{\mathcal{S}_n}{}^{\top} \in \mathcal{W}, \tag{3.20}$$

where the second equality is due to the embedding (3.18). The mappings (3.10)–(3.14a) naturally extend to the assignment manifold $\mathcal{W}$

$$\Pi[Z] = \big(\Pi_n[Z_1], \ldots, \Pi_n[Z_m] \big)^{\top} \in \mathcal{T}_{\mathcal{W}}, \tag{3.21a}$$

$$R_W[Z] = \big(R_{W_1}[Z_1], \ldots, R_{W_m}[Z_m] \big)^{\top} \in \mathcal{T}_{\mathcal{W}}, \tag{3.21b}$$

$$\operatorname{Exp}_W(V) = \big(\operatorname{Exp}_{W_1}(V_1), \ldots, \operatorname{Exp}_{W_m}(V_m) \big)^{\top} \in \mathcal{W}, \tag{3.21c}$$

with $W \in \mathcal{W}$, $Z \in \mathbb{R}^{m \times n}$ and $V \in \mathcal{T}_{\mathcal{W}}$. The maps $\exp_W$, $\operatorname{Exp}_W^{-1}$, $\exp_W^{-1}$ are similarly defined based on (3.15a), (3.14b) and (3.15b). Due to (3.13) , the Riemannian gradient and the Euclidean gradient of a smooth function $f : \mathcal{W} \to \mathbb{R}$ are also related by

$$\operatorname{grad} f(W) = R_W[\partial f(W)] \quad \text{for } W \in \mathcal{W}. \tag{3.22}$$

3.2 Assignment Flow

Based on the given data (3.1) and labels (3.2), the ith row of the *distance matrix* $D \in \mathbb{R}^{m \times n}$ is defined by

$$D_i := \big(d(f_i, \ell_1), \ldots, d(f_i, \ell_n) \big)^{\top} \in \mathbb{R}^n, \tag{3.23}$$

for all $i \in \mathcal{V}$. This distance information is *lifted* onto the manifold by the following *likelihood matrix*

$$L(W) := \exp_W(-D/\rho) \in \mathcal{W}, \tag{3.24a}$$

$$L_i(W_i) = \frac{W_i e^{-\frac{1}{\rho} D_i}}{\langle W_i, e^{-\frac{1}{\rho} D_i} \rangle}, \quad \rho > 0, \quad i \in \mathcal{V}, \tag{3.24b}$$

where $\rho > 0$ is a scaling parameter to normalize the a priori unknown scale of the distances induced by the features f_i depending on the application at hand. This representation of the data is regularized by weighted geometric averaging in the local neighborhoods (3.3) using the weights (3.5), to obtain the *similarity matrix* $S(W) \in \mathcal{W}$, with ith row defined by

$$\begin{aligned} &S_i : \mathcal{W} \to \mathcal{S}_n, \\ &S_i(W) := \operatorname{Exp}_{W_i}\Big(\sum_{k \in \mathcal{N}_i} w_{ik} \operatorname{Exp}_{W_i}^{-1}(L_k(W_k)) \Big). \end{aligned} \tag{3.25}$$

If Exp_{W_i} were the exponential map of the Riemannian (Levi-Civita) connection, then the sum inside the outer brackets of the right-hand side in (3.25) would just be the negative Riemannian gradient with respect to W_i of the objective function used to define the Riemannian center of mass, i.e., the weighted sum of the squared Riemannian distances between W_i and L_k [14, Lemma 6.9.4]. In view of Remark 5, this interpretation is only approximately true mathematically, but still correct informally: $S_i(W)$ moves W_i toward the normalized geometric mean of the likelihood vectors L_k, $k \in \mathcal{N}_i$.

The similarity matrix induces the *assignment flow* through a system of spatially coupled nonlinear ODEs which evolves the assignment vectors

$$\dot{W} = R_W S(W), \quad W(0) = \mathbb{1}_{\mathcal{W}}, \tag{3.26a}$$

$$\dot{W}_i = R_{W_i} S_i(W), \quad W_i(0) = \mathbb{1}_{\mathcal{S}_n}. \quad i \in \mathcal{V}. \tag{3.26b}$$

Integrating this flow numerically [25] yields curves $W_i(t) \in \mathcal{S}_n$ for every pixel $i \in \mathcal{V}$ emanating from $W_i(0) = \mathbb{1}_{\mathcal{S}_n}$, which approach some vertex (unit vector) of $\overline{\mathcal{S}_n} = \Delta_n$ and hence a unique label assignment after a trivial rounding $W_i(t)$ for sufficiently large $t > 0$.

3.3 Linear Assignment Flow

The *linear assignment flow*, introduced by [25], uses the exponential map with respect to the e-connection (3.14a) in order to approximate the mapping (3.25) as part of the assignment flow (3.26a) by

$$\dot{W} = R_W\Big[S(W_0) + dS(W_0)\big[\operatorname{Exp}_{W_0}^{-1}(W) \big] \Big], \tag{3.27a}$$

$$W_0 = W(0) = \mathbb{1}_{\mathcal{W}} \in \mathcal{W}. \tag{3.27b}$$

This linear assignment flow (3.27) is still *nonlinear* but admits the following parametrization [25, Prop. 4.2]:

$$W(t) = \operatorname{Exp}_{W_0}\big(V(t)\big), \tag{3.28a}$$

$$\dot{V}(t) = R_{W_0}\big[S(W_0) + dS(W_0)[V(t)] \big], \tag{3.28b}$$

$$V(0) = 0, \tag{3.28c}$$

where the latter ODE is *linear* and defined on the vector space $\mathcal{T}_{\mathcal{W}}$. Fixing $S(W_0)$ in the following, (3.28) is *linear* with respect to both the tangent vector V and the parameters ω_{ik} in the differential $dS(W_0)$ (see (3.30) and Remark 7), that makes this approach attractive for parameter estimation.

It can be shown that $S_i(W)$ from (3.25) can equivalently be expressed with $\exp_{\mathbb{1}_{\mathcal{S}_n}}$ as

$$S_i(W) = \exp_{\mathbb{1}_{\mathcal{S}_n}}\Big(\sum_{k \in \mathcal{N}_i} \omega_{ik} \Big(\exp_{\mathbb{1}_{\mathcal{S}_n}}^{-1}(W_k) - \frac{1}{\rho} D_k \Big) \Big) \tag{3.29}$$

for all $i \in \mathcal{V}$, $W \in \mathcal{W}$. A standard calculation shows that the ith component of the differential $dS(W) : \mathcal{T}_{\mathcal{W}} \to \mathcal{T}_{\mathcal{W}}$ is given by

$$\begin{aligned} &dS_i(W) : \mathcal{T}_{\mathcal{W}} \to T_n, \\ &dS_i(W)[V] = \sum_{k \in \mathcal{N}_i} \omega_{ik} R_{S_i(W)} \left[\frac{V_k}{W_k} \right] \end{aligned} \tag{3.30}$$

for all $V \in \mathcal{T}_0$, $i \in \mathcal{V}$.

3.4 Numerical Integration of the Flow

Setting $\Lambda(V, W) := \exp_W(V)$ gives an action $\Lambda : \mathcal{T}_{\mathcal{W}} \times \mathcal{W} \to \mathcal{W}$ of the vector space $\mathcal{T}_{\mathcal{W}}$ viewed as an additive group on the assignment manifold $\mathcal{W}$. In [25], this action is used to numerically integrate the assignment flow by applying geometric Runge–Kutta methods. The resulting method for an arbitrary vector field $F : \mathcal{W} \to \mathcal{T}_{\mathcal{W}}$ is as follows. Suppose the ODE

$$\dot{W}(t) = R_{W(t)}[F(W(t))], \quad W(0) = \mathbb{1}_{\mathcal{W}} \tag{3.31}$$

on the assignment manifold is given. Then, the parametrization $W(t) = \exp_{\mathbb{1}_{\mathcal{W}}}(V(t))$ yields an equivalent reparametrized ODE

$$\dot{V}(t) = F(W(t)) = F\big(\exp_{\mathbb{1}_{\mathcal{W}}}(V(t)) \big), \tag{3.32a}$$

$$V(0) = 0 \tag{3.32b}$$

purely evolving on the vector space $\mathcal{T}_{\mathcal{W}}$, where standard Runge–Kutta methods (cf. Sect. 2) can now be used for numerical integration. Translating these update schemes back onto $\mathcal{W}$ yields geometric Runge–Kutta methods on $\mathcal{W}$ induced by the Lie-group action $\Lambda = \exp$.

Remark 6 Notice that the assumption $F(W) \in \mathcal{T}_{\mathcal{W}}$ is crucial because the transformation of the ODE (3.31) onto $\mathcal{T}_{\mathcal{W}}$ in (3.32) uses the inverse of R_W, which only exists for elements of $\mathcal{T}_{\mathcal{W}}$ but not for $\mathbb{R}^{m \times n}$. However, there is no limitation.

Suppose any vector field $\tilde{F}\colon \mathcal{W} \to \mathbb{R}^{m\times n}$ is given. Due to $R_W = R_W \circ \Pi$ by (3.12), we may consider $F(W) := \Pi[\tilde{F}(W)] \in \mathcal{T}_{\mathcal{W}}$ instead, without changing the underlying ODE (3.31) for $W(t)$.

In the following, we mainly use the Euler method to numerically integrate the flow (3.32) on the vector space $\mathcal{T}_{\mathcal{W}}$, i.e.,

$$\begin{aligned} V^{(k+1)} &= V^{(k)} + h_k F\big(W^{(k)}\big), \quad V^{(0)} = 0, \\ W^{(k)} &= \exp_{\mathbb{1}_{\mathcal{W}}}\big(V^{(k)}\big) \end{aligned} \tag{3.33}$$

with step-size $h_k > 0$. Due to the Lie-group action, this update scheme translates to the geometric Euler integration on $\mathcal{W}$ given by

$$W^{(k+1)} = \exp_{W^{(k)}}\Big(h_k F\big(W^{(k)}\big)\Big), \tag{3.34a}$$

$$W^{(0)} = \mathbb{1}_{\mathcal{W}}, \tag{3.34b}$$

with step-size $h_k > 0$.

4 Learning Adaptive Regularization Parameters

In this section, we study the parameter learning approach (4.1), which is a specific instance of the general formulation (1.1). The goal is to adapt the regularization of the linear assignment flow (3.27) on the fixed time horizon $[0, T]$ controlled by the weights (3.5), in the following collectively denoted by Ω, so as to preserve important image structure in a supervised manner. During learning, the image structure is prescribed by given ground truth labeling information W^*, where every row W_i^* is some unit basis vector e_{k_i} of $\mathbb{R}^n$ representing the ground truth label l_{k_i} at node $i \in \mathcal{V}$. The adaptivity of the weights with respect to the desired image structure is measured by $\mathcal{C}$ in terms of the discrepancy between ground truth W^* and the labeling induced by $V(T) = V(T, \Omega)$ at fixed time T. The corresponding optimization problem reads

$$\min_{\Omega\in\mathcal{P}} \quad \mathcal{C}\big(V(T,\Omega)\big) \tag{4.1a}$$

$$\text{s.t.} \quad \dot{V}(t) = F(V(t),\Omega), \quad t \in [0,T], \tag{4.1b}$$

$$V(0) = 0, \tag{4.1c}$$

with components

- $\mathcal{P}$ parameter manifold, representing the weights ω_{ik} from (3.5); see Sect. 4.1.
- $F(V,\Omega)$ modified version of the linear assignment flow (4.11); see Sect. 4.2.
- $\mathcal{C}$ a objective function measuring the discrepancy to the ground truth; see Sect. 4.3.

It is important to note that the dependency of $\mathcal{C}(V(T,\Omega))$ on the weights Ω is only *implicitly* given through the solution $V(T) = V(T,\Omega)$ of (4.1b). In Sect. 4.4, we therefore present a numerical first-order scheme for optimizing (4.1) where the gradient of $\mathcal{C}(V(T,\Omega))$ with respect to the parameter Ω is calculated using the sensitivity analysis from Sect. 2.

4.1 Parameter Manifold

In the following, we define the parameter manifold representing the weights ω_{ik} from (3.5) associated with the neighborhood $\mathcal{N}_i$, $i \in \mathcal{V}$. Based on this parametrization, we can compute the differential $dS(W_0)$ and thus describe the linear assignment flow (3.28) on the tangent space by a corresponding expression in Lemma 2.

To simplify the exposition, we assume that all neighborhoods $\mathcal{N}_i$ have the same size

$$N := |\mathcal{N}_i| \quad \text{for all } i \in \mathcal{V}. \tag{4.2}$$

Due to the constraints (3.5), the weight vector $\Omega_i := (\omega_{i1}, \ldots, \omega_{iN})^\top$ can be viewed as a point in $\mathcal{S}_N$. Accordingly, we define the *parameter manifold*

$$\mathcal{P} := \mathcal{S}_N \times \ldots \times \mathcal{S}_N \qquad (m = |\mathcal{V}| \text{ times}) \tag{4.3}$$

as feasible set for learning the weights, which has the form of an assignment manifold and thus also has a Riemannian structure $(\mathcal{P}, g)$, given by the Fisher-Rao metric. We use the identification

$$\begin{aligned} \mathcal{P} = \{\Omega \in \mathbb{R}^{m\times N} &: \Omega \mathbb{1}_N = \mathbb{1}_m \text{ and } \Omega_{ik} > 0 \\ &\text{for all } i \in [m], k \in [N]\}. \end{aligned} \tag{4.4}$$

Points $\Omega \in \mathcal{P}$ now represent the global choice of weights with Ω_i representing the weights ω_{ik} associated with the neighborhood $\mathcal{N}_i$ in (3.5). The constant tangent space of $\mathcal{P}$ is denoted by $\mathcal{T}_{\mathcal{P}}$ and the corresponding orthogonal projection by

$$\begin{aligned} &\Pi_{\mathcal{P}}\colon \mathbb{R}^{m\times N} \to \mathcal{T}_{\mathcal{P}}, \\ &M \mapsto \Pi_{\mathcal{P}}[M] = (\Pi_N[M_1], \ldots, \Pi_N[M_m])^\top. \end{aligned} \tag{4.5}$$

Next, we give a global expression for the differential $dS(W)$ which will simplify the following formulas and calculations. For this, we define the *averaging matrix* $A_\Omega \in \mathbb{R}^{m\times m}$ with weights $\Omega \in \mathcal{P}$ by

$$(A_\Omega)_{ik} := \delta_{k\in\mathcal{N}_i}\Omega_{ik} = \begin{cases} \Omega_{ik}, & \text{for } k \in \mathcal{N}_i \\ 0, & \text{else,} \end{cases} \tag{4.6}$$

where $\delta_{k\in\mathcal{N}_i}$ is the Kronecker delta with value 1 if $k \in \mathcal{N}_i$ and 0 otherwise. We observe that the averaging matrix A_Ω *linearly* depends on the weight parameters.

Thus, A_Ω parametrizes averages depending on the corresponding weights Ω with respect to the underlying graph structure, given by the neighborhoods (3.3). For a matrix $M \in \mathbb{R}^{m\times n}$, the averages of the row vectors with weights Ω are then just given by the matrix multiplication $A_\Omega M$, with the ith row vector given by

$$(A_\Omega M)_i = \sum_{k\in\mathcal{N}_i} \omega_{ik} M_k. \quad \text{for all } i \in \mathcal{V}. \tag{4.7}$$

For later use, we record the following formula for the adjoint of A_Ω as a linear map with respect to Ω.

Lemma 1 *If the averaging matrix is viewed as a linear map $A\colon \mathbb{R}^{m\times N} \to \mathbb{R}^{m\times m}$, $\Omega \mapsto A_\Omega$, then the adjoint map $A^\top\colon \mathbb{R}^{m\times m} \to \mathbb{R}^{m\times N}$, $B \mapsto A_B^\top$ is given by*

$$\big(A_B^\top\big)_{ij} = B_{ij} \quad \text{for } i \in \mathcal{V},\, j \in \mathcal{N}_i. \tag{4.8}$$

Proof For arbitrary $B \in \mathbb{R}^{m\times m}$ and $\Omega \in \mathbb{R}^{m\times N}$, we obtain $\langle A_\Omega, B\rangle = \sum_{i,j\in\mathcal{V}} \delta_{j\in\mathcal{N}_i} \Omega_{ik} B_{ik} = \langle \Omega, A_B^\top\rangle$ due to (4.6). □

Using A_Ω with $\Omega \in \mathcal{P}$, it follows from (3.30) that $dS(W)$ can be expressed as

$$dS(W)[V] = R_{S(W)}\left[A_\Omega\left(\frac{V}{W}\right)\right], \tag{4.9}$$

for all $V \in \mathcal{T}_\mathcal{W}$, $W \in \mathcal{W}$. As a result, the linear assignment flow (3.28) on the vector space $\mathcal{T}_\mathcal{W}$ can be parametrized as follows.

Lemma 2 *Using the parametrization $\overline{V} := nV$, the linear assignment flow (3.28) takes the form*

$$W(t) = \exp_{\mathbb{1}_\mathcal{W}}(\overline{V}(t)), \tag{4.10a}$$

$$\dot{\overline{V}}(t) = \Pi[S(W_0)] + R_{S(W_0)}[A_\Omega \overline{V}(t)], \tag{4.10b}$$

$$\overline{V}(0) = 0. \tag{4.10c}$$

Proof At $p = \mathbb{1}_{\mathcal{S}_n}$, the linear map (3.12) takes the form

$$R_{\mathbb{1}_{\mathcal{S}_n}} = \mathrm{Diag}(\mathbb{1}_{\mathcal{S}_n}) - \mathbb{1}_{\mathcal{S}_n}\mathbb{1}_{\mathcal{S}_n}{}^\top = \frac{1}{n}\big(I - \mathbb{1}_{\mathcal{S}_n}\mathbb{1}^\top\big) = \frac{1}{n}\Pi_n,$$

where $I \in \mathbb{R}^{n\times n}$ denotes the identity matrix. Because of $W_0 = \mathbb{1}_\mathcal{W}$, $R_{W_0} = \frac{1}{n}\Pi$ follows. Therefore, $V = \frac{1}{n}\overline{V} = R_{\mathbb{1}_\mathcal{W}}\overline{V}$ which directly yields

$$W = \mathrm{Exp}_{\mathbb{1}_\mathcal{W}}(V) = \mathrm{Exp}_{\mathbb{1}_\mathcal{W}}(R_{\mathbb{1}_\mathcal{W}}[\overline{V}]) = \exp_{\mathbb{1}_\mathcal{W}}(\overline{V}).$$

Using (4.9) together with $\frac{V}{W_0} = nV = \overline{V}$, the linear assignment flow (3.28) takes the form

$$\begin{aligned}\dot{V}(t) &= R_{W_0}\left[S(W_0) + R_{S(W_0)}\left[A_\Omega\left(\frac{V(t)}{W_0}\right)\right]\right] \\ &= \frac{1}{n}\Pi\big[S(W_0) + R_{S(W_0)}[A_\Omega \overline{V}(t)]\big].\end{aligned}$$

As a consequence of $\Pi R_{S(W_0)} = R_{S(W_0)}$ by (3.12), the right-hand side of (4.10) follows after multiplying the equation by n and using $n\dot{V} = \dot{\overline{V}}$. □

Remark 7 To simplify notation, we will write V for $\overline{V}$ below. Equation (4.10) highlights the importance to fix $S(W_0)$ in order to obtain a model that is linear in both the state vector V and the parameters Ω.

4.2 Modified Linear Assignment Flow

We now return to our objective to estimate the weight parameters $\Omega \in \mathcal{P}$ controlling the linear assignment flow on the fixed time interval $[0, T]$, in the supervised scenario (4.1). In this formulation, the data represented by the likelihood matrix (3.24) only influence the linear assignment flow (3.27), or equivalently (4.10), through the constant similarity matrix $S(W_0)$ that comprises *averaged* data information depending on the initial choice of the weights Ω_0. However, since the *initial* weights are in general *not* adapted to any specific image structure, this can lead to a loss of desired structural information through $S(W_0)$ at the outset, that cannot be recovered afterward.

To avoid this problem, we slightly modify the linear assignment flow in (4.10) to obtain an explicit data term that *does not depend on the choice of the initial* weights. This is done through replacing the constant term $S(W_0)$ by the lifted distances $L(W_0)$, which results in the *modified linear assignment flow*

$$W(t) = \exp_{W_0}\big(V(t)\big) \tag{4.11a}$$

$$\begin{aligned}\dot{V}(t) &= \Pi[L(W_0)] + R_{S(W_0)}[A_\Omega V(t)] \\ &=: F(V(t), \Omega),\end{aligned} \tag{4.11b}$$

$$V(0) = 0. \tag{4.11c}$$

Remark 8 We point out that, strictly speaking, the similarity matrix $S(W_0)$ is involved in two ways, in the constant term of (4.10) and in the expression $R_{S(W_0)}$ of the differential $dS(W_0)$ (cf. (4.9)). However, the effect of the latter with respect to the initial weights is negligible, and the former appearance only causes the above-mentioned loss of initial data information.

We note again that (4.11) is *linear* with respect to both the tangent vector V and the parameters Ω only if $S(W_0)$ is kept constant.

Proposition 1 *The differential of the map $F: \mathcal{T}_\mathcal{W} \times \mathcal{P} \to \mathcal{T}_\mathcal{W}$ on the right-hand side of (4.11) with respect to the first and second argument are given by*

$$d_V F(V,\Omega): \mathcal{T}_\mathcal{W} \to \mathcal{T}_\mathcal{W}, \quad X \mapsto d_V F(V,\Omega)[X] = R_{S(W_0)}[A_\Omega X], \tag{4.12a}$$

$$d_\Omega F(V,\Omega): \mathcal{T}_\mathcal{P} \to \mathcal{T}_\mathcal{W}, \quad \Psi \mapsto d_\Omega F(V,\Omega)[\Psi] = R_{S(W_0)}[A_\Psi V]. \tag{4.12b}$$

The corresponding adjoint mappings with respect to the standard Euclidean structure of $\mathbb{R}^{m\times n}$ are

$$d_V F(V,\Omega)^\top: \mathcal{T}_\mathcal{W} \to \mathcal{T}_\mathcal{W}, \quad X \mapsto d_V F(V,\Omega)^\top[X] = A_\Omega^\top R_{S(W_0)}[X], \tag{4.13a}$$

$$d_\Omega F(V,\Omega)^\top: \mathcal{T}_\mathcal{W} \to \mathcal{T}_\mathcal{P}, \quad X \mapsto d_\Omega F(V,\Omega)^\top[X] = \Pi_\mathcal{P}\big[A^\top_{(R_{S(W_0)}[X])V^\top}\big], \tag{4.13b}$$

with the adjoint $A^\top_{(\cdot)}$ from Lemma 1.

Proof Let $V, X \in \mathcal{T}_\mathcal{W}$ and set $\gamma(t) := V + tX \in \mathcal{T}_0$ for all $t \in \mathbb{R}$. Then,

$$\begin{aligned} d_V F(V,\Omega)[X] &= \frac{\mathrm{d}}{\mathrm{d}t} F(\gamma(t),\Omega)\big|_{t=0} \\ &= R_{S(W_0)}[A_\Omega \dot\gamma(0)] = R_{S(W_0)}[A_\Omega X]. \end{aligned}$$

Similarly, for $\Omega \in \mathcal{P}$ and $\Psi \in \mathcal{T}_\mathcal{P}$, let $\eta(t) := \Omega + t\Psi \in \mathcal{P}$ be a curve with $t \in (-\varepsilon, \varepsilon)$ for sufficiently small $\varepsilon > 0$. The linearity of the averaging operator A_Ω with respect to Ω gives

$$\begin{aligned} d_\Omega F(V,\Omega)[X] &= \frac{\mathrm{d}}{\mathrm{d}t} F(V,\eta(t))\big|_{t=0} \\ &= \frac{\mathrm{d}}{\mathrm{d}t} R_{S(W_0)}[A_{\eta(t)} V]\big|_{t=0} = R_{S(W_0)} A_\Psi[V]. \end{aligned}$$

We now determine the adjoint differentials. Consider arbitrary $X, Y \in \mathcal{T}_\mathcal{W}$ and note that the linear map $R_S(W_0)$ is symmetric, since every component map $R_{S_i(W_0)}$ is symmetric by (3.11). Thus,

$$\begin{aligned} \langle d_V F(V,\Omega)[Y], X\rangle &= \langle R_{S(W_0)}[A_\Omega Y], X\rangle \\ &= \langle Y, A_\Omega^\top R_{S(W_0)}[X]\rangle \end{aligned}$$

and therefore $d_V F(V,\Omega)^\top[X] = A_\Omega^\top R_{S(W_0)}[X]$.

Now, let arbitrary $\Psi \in \mathcal{T}_\mathcal{P}$ and $X \in \mathcal{T}_\mathcal{P}$ be given. Then,

$$\begin{aligned} \langle d_\Omega F(V,\Omega)[\Psi], X\rangle &= \langle R_{S(W_0)}[A_\Psi V], X\rangle \\ &= \langle A_\Psi, (R_{S(W_0)}[X])V^\top\rangle \\ &= \langle \Psi, A^\top_{(R_{S(W_0)}[X])V^\top}\rangle \\ &= \langle \Psi, \Pi_\mathcal{P}\big[A^\top_{(R_{S(W_0)}[X])V^\top}\big]\rangle, \end{aligned}$$

which proves the expression for the corresponding adjoint.□

4.3 Objective Function

Let $W = \exp_{\mathbb{1}_\mathcal{W}}(V) \in \mathcal{W}$ be an assignment induced by $V \in \mathcal{T}_\mathcal{W}$. Accumulating the KL-divergence between the ground truth W_i^* and W_i for every node $i \in \mathcal{V}$,

$$\begin{aligned} \mathrm{KL}(W_i^*, W_i) &= \sum_{j\in[n]} W_{ij}^* \log\left(\frac{W_{ij}^*}{W_{ij}}\right) \\ &= \langle W_i^*, \log(W_i^*)\rangle - \langle W_i^*, \log(W_i)\rangle, \end{aligned} \tag{4.14}$$

results in a measure of the global deviation between W induced by V and the ground truth W^*

$$\begin{aligned} \mathcal{C}(V) &:= \sum_{i\in\mathcal{V}} \mathrm{KL}(W_i^*, \exp_{\mathbb{1}_{\mathcal{S}_n}}(V_i)) \\ &= \langle W^*, \log(W^*)\rangle - \langle W^*, \log\big(\exp_{\mathbb{1}_\mathcal{W}}(V)\big)\rangle. \end{aligned} \tag{4.15}$$

Remark 9 It is important to note that $\mathcal{C}$ does *not explicitly* depend on the weights $\Omega \in \mathcal{P}$. In problem formulation (4.1a), this dependency is only given *implicitly* through the evaluation of $\mathcal{C}$ at $V(T,\Omega)$, where $V(t,\Omega)$ is the object depending on the parameter Ω as solution of the modified linear assignment flow (4.11).

Proposition 2 *The Euclidean gradient of objective (4.15) for fixed $W^* \in \mathcal{W}$ is given by*

$$\partial\mathcal{C}(V) = \exp_{\mathbb{1}_\mathcal{W}}(V) - W^* \quad \textit{for} \quad V \in \mathcal{T}_\mathcal{W}. \tag{4.16}$$

Proof Let $V \in \mathcal{T}_\mathcal{W}$. Note that for every $i \in \mathcal{V}$

$$\begin{aligned} \langle W_i^*, \log\big(\exp_{\mathbb{1}_{\mathcal{S}_n}}(V_i)\big)\rangle &= \langle W_i^*, V_i - \log(\langle \mathbb{1}, e^{V_i}\rangle)\mathbb{1}\rangle \\ &= \langle W_i^*, V_i\rangle + \log(\langle \mathbb{1}, e^{V_i}\rangle). \end{aligned}$$

Hence, the KL-divergence between W_i^* and the induced assignment $W_i = \exp_{\mathbb{1}_{\mathcal{S}_n}}(V_i)$ takes the form

$$\mathrm{KL}\left(W_i^*, W_i\right) = \langle W_i^*, \log(W_i^*)\rangle - \langle W_i^*, V_i\rangle + \log(\langle \mathbb{1}, e^{V_i}\rangle)$$

and results in the following expression for $\mathcal{C}$ from (4.15),

$$\mathcal{C}(V) = \langle W^*, \log(W^*)\rangle - \langle W^*, V\rangle + \sum_{i\in[m]} \log(\langle \mathbb{1}, e^{V_i}\rangle).$$

Take $X \in \mathbb{R}^{m\times n}$ and set $\gamma(t) := V + tX$ for $t \in \mathbb{R}$. The above formula for $\mathcal{C}$ then implies

$$\begin{aligned}\langle \partial C(V), X\rangle &= \frac{\mathrm{d}}{\mathrm{d}t} C(\gamma(t))\big|_{t=0} \\ &= -\langle W^*, X\rangle + \sum_{i\in[m]} \frac{1}{\langle \mathbb{1}, e^{V_i}\rangle} \langle e^{V_i}, X_i\rangle \\ &= \langle \exp_{\mathbb{1}_{\mathcal{W}}}(V) - W^*, X\rangle.\end{aligned}$$

Since $X \in \mathbb{R}^{m\times n}$ was arbitrary, expression (4.16) follows. □

4.4 Numerical Optimization

With the above definitions of $\mathcal{C}$ and F, the optimization problem (4.1) for adapting the weights of the modified linear assignment flow (4.11) takes the form

$$\min_{\Omega\in\mathcal{P}} \sum_{i\in\mathcal{V}} \mathrm{KL}(W_i^*, W_i(T,\Omega)) \tag{4.17a}$$

s.t.

$$\dot V(t) = \Pi[L(W_0)] + R_{S(W_0)}[A_\Omega[V(t)]], \tag{4.17b}$$

$$V(0) = 0, \tag{4.17c}$$

$$W(T,\Omega) = \exp_{\mathbb{1}_{\mathcal{W}}}(V(T,\Omega)), \tag{4.17d}$$

with $t \in [0, T]$. Our strategy for *parameter learning* is to follow the *Riemannian gradient descent flow* on the parameter manifold induced by the potential

$$\begin{aligned}&\Phi : \mathcal{P} \to \mathbb{R}, \\ &\Omega \mapsto \Phi(\Omega) := \sum_{i\in\mathcal{V}} \mathrm{KL}(W_i^*, W_i(T,\Omega)).\end{aligned} \tag{4.18}$$

Due to (3.22), this Riemannian gradient flow on $\mathcal{P}$ takes the form

$$\begin{aligned}\dot\Omega(t) &= -\operatorname{grad}_{\mathcal{P}} \Phi\big(\Omega(t)\big) \\ &= -R_\Omega\big[\partial\Phi\big(\Omega(t)\big)\big],\end{aligned} \tag{4.19a}$$

$$\Omega(0) = \mathbb{1}_{\mathcal{P}}, \tag{4.19b}$$

where R_Ω is given by (3.21b) on $\mathcal{P}$ and initial value (4.19b) represents an *unbiased* initialization, i.e., *uniform* weights at every patch $\mathcal{N}_i$ at $i \in \mathcal{V}$.

We discretize (4.19) using the geometric explicit Euler scheme (3.34) from Sect. 3.4 with constant step-size $h' > 0$, which results in Algorithm 1.

Algorithm 1: Discretized *Riemannian flow* (4.19).

Data: Initial weights $\Omega^{(0)} = \mathbb{1}_{\mathcal{P}}$, objective function $\Phi(\Omega) = \mathcal{C}\big(V(T,\Omega)\big)$
Result: Weight parameter estimates Ω^*

```
// geometric Euler integration
for k = 0, ..., K do
    compute ∂Φ(Ω^(k)) ;                    // Algorithm 2
    Ω^(k+1) = exp_{Ω^(k)} ( − h' R_{Ω^(k)} [∂Φ(Ω^(k))] );
```

Algorithm 1 calls Algorithm 2 that we explain next. As pointed out in Remark 9, the dependency of $\Phi(\Omega) = \mathcal{C}(V(T,\Omega))$ on Ω is only implicitly given through the solution $V(t,\Omega)$ of the modified linear assignment flow (4.11), evaluated at time T. According to (2.2), the gradient of Φ decomposes as

$$\partial\Phi(\Omega) = d_\Omega V(T,\Omega)^\top \big[\partial\mathcal{C}(V(T,\Omega))\big], \tag{4.20}$$

where $d_\Omega V(T,\Omega)^\top$ is the sensitivity of the solution $V(T,\Omega)$ with respect to Ω. Thus, the major task is to determine the sensitivity of $V(T,\Omega)$ in order to obtain the gradient $\partial\Phi(\Omega)$, which in turn drives the Riemannian gradient descent flow and adapts the weights Ω. To this end, we choose the discretize-then-differentiate approach (2.22)—recall the commutative diagram of Fig. 2 and relations summarized as Remark 3—with the explicit Euler method and constant step-size $h > 0$, which results in Algorithm 2.

Algorithm 2: Computation of the Euclidean gradient $\partial\Phi(\Omega^{(k)})$ (4.20).

Data: Current weights $\Omega^{(k)}$
Result: Objective value $\Phi(\Omega^{(k)}) = \mathcal{C}(V^{(N)}(\Omega^{(k)}))$, adjoint sensitivity $\partial\Phi(\Omega^{(k)})$

```
// forward Euler integration
for j = 0, ..., N − 1 do
    V^(j+1) = V^(j) + h F(V^(j), Ω^(k));
compute λ^(N) = ∂C(V^(N)(Ω^(k)));
set ∂Φ(Ω) = 0;
// backward Euler integration
for j = N − 1, ..., 0 do
    λ^(j) = λ^(j+1) + h d_V F(V^(j), Ω^(k))^⊤ λ^(j+1);
    ∂Φ(Ω) += h d_Ω F(V^(j−1), Ω^(k))^⊤ λ^(j);
    // summand of (4.20)
```

5 Experiments

In this section, we demonstrate and evaluate our approach. We start in Sect. 5.1 with a scenario of two labels and images of binary letters. We show that an adaptive regularizer, which is trained on letters with *vertical and horizontal* structures only, effectively labels *curvilinear* letters. This result illustrates the *adaptivity* of regularization by using *non-uniform* weights that are predicted for *novel unseen* image data.

In Sect. 5.2, we consider a scenario with three labels and curvilinear line structure, that has to be detected and labeled explicitly in noisy data. Just using uniform weights for regularization must fail. In addition to the noise, the actual image structure is randomly generated as well and defines a class of images. We demonstrate empirically that learning the weights to adapt within local neighborhoods from example data solves this problem.

In Sect. 5.3, we adopt a different viewpoint and focus on pattern formation, rather than on pattern detection and recovery. We demonstrate the modeling expressiveness of the assignment flow with respect to pattern formation. In fact, even when using the *linear* assignment flow as in the present paper, label information can be flexibly transported across the image domain under certain conditions. The experiments just indicate what can be done, in principle, in order to stimulate future work. We return to this point in Sect. 6.

Regarding parameter learning, all experiments were conducted using the Euler scheme of Sect. 2.4.1 for solving the adjoint system. Section 2.4.2 provides a slightly more advanced alternative. While the latter method integrates more accurately, the resulting overall costs depend on further factors whose evaluation is beyond the scope of this paper. For an in-depth study of numerical schemes in connection with geometric integration of the assignment flow, we refer to [25].

5.1 Adaptive Regularization of Binary Letters

In this experiment, we consider binary images of letters. The goal is to label a given letter image into *foreground* and *background* regions. In Fig. 3, these labels are encoded by $\{\square, \blacksquare\} = \{\textit{background}, \textit{foreground}\}$. First, we apply our approach during a *training phase* in order to learn weight adaptivity for letters consisting of *vertical and horizontal* structures (Fig. 3a). Afterward, we evaluate the approach in a *test phase* using letters consisting of *curvilinear* structures (Fig. 3g).

5.1.1 Training Phase

Figure 3a shows the binary images of letters which we used as *training data*. Hereby, a given binary image served as input image and as ground truth as well. By using these data and solving problem (4.17), we *learn how to adapt the regularization parameter* of the modified linear assignment flow (4.11).

Optimization For each binary letter image, we solved problem (4.17) using Algorithms 1 and 2and the following parameter values: $|\mathcal{N}_i| = 7 \times 7$ (size of local neighborhoods, for every i), the Hamming distance (for the computation of the distance matrix (3.23)), $\rho = 0.5$ (scaling parameter for distance matrix, cf. (3.24)), $h = 0.1$ (constant step size for computing the gradient with Algorithm 2), and $T = 6$ (end of time horizon). As for optimization on the parameter manifold $\mathcal{P}$ through the Riemannian gradient flow (Algorithm 1), we used an initial value of $h' = 0.005$ together with backtracking for adapting the step size. We terminated the iteration once the relative change

$$\frac{|\Phi(\Omega^{(k)}) - \Phi(\Omega^{(k-1)})|}{h'|\Phi(\Omega^{(k)})|} \tag{5.1}$$

of the objective function $\Phi\big(\Omega^{(k)}\big) = \mathcal{C}\big(V^{(N)}(\Omega^{(k)})\big)$ dropped below 0.01 or the maximum number of 50 iterations was reached.

Results The left column of Figure 3 shows the results obtained during the training phase. Using *uniform* weights fails completely to detect and label the letter structures (panel b). In contrast, the *adapted* regularizer preserves the structure perfectly (panel c), i.e., the *optimal* weights steered the linear assignment flow toward the given ground-truth labeling. Panel (d) visualizes the weight *adaptivity* at each pixel in terms of the KL-divergence of the *learned* weight to the *uniform* weight patch.

5.1.2 Test Phase

During the training phase, optimal weights were associated with all training features through optimization, based on ground truth and a corresponding objective function. In the test phase with novel data and features, appropriate weights have to be *predicted* because ground truth no longer is available. This was done by extracting a *coreset* [17] from the output generated by Algorithm 1 during the training phase and constructing a map from novel features to weights, as described next.

Coreset Let Ω^* denote the set of optimal weight patches generated by Algorithm 1. As features, we used 7×7 patches extracted from the training images. Let P^* denote all *feature vectors* f_i, $i \in \mathcal{V}$ (dimension $7 \times 7 = 49$) that were given as a point set in the Euclidean feature space $\mathcal{F} = \mathbb{R}^{49}$. We partitioned P^* into two classes: *foreground* and *background*. Each class is represented by 156 prototypical patches extracted from the binary images. To each of these patches, a *prototypical weight patch* was assigned, namely the geometric mean of all *optimal* weight patches in Ω^* belonging to that patch.

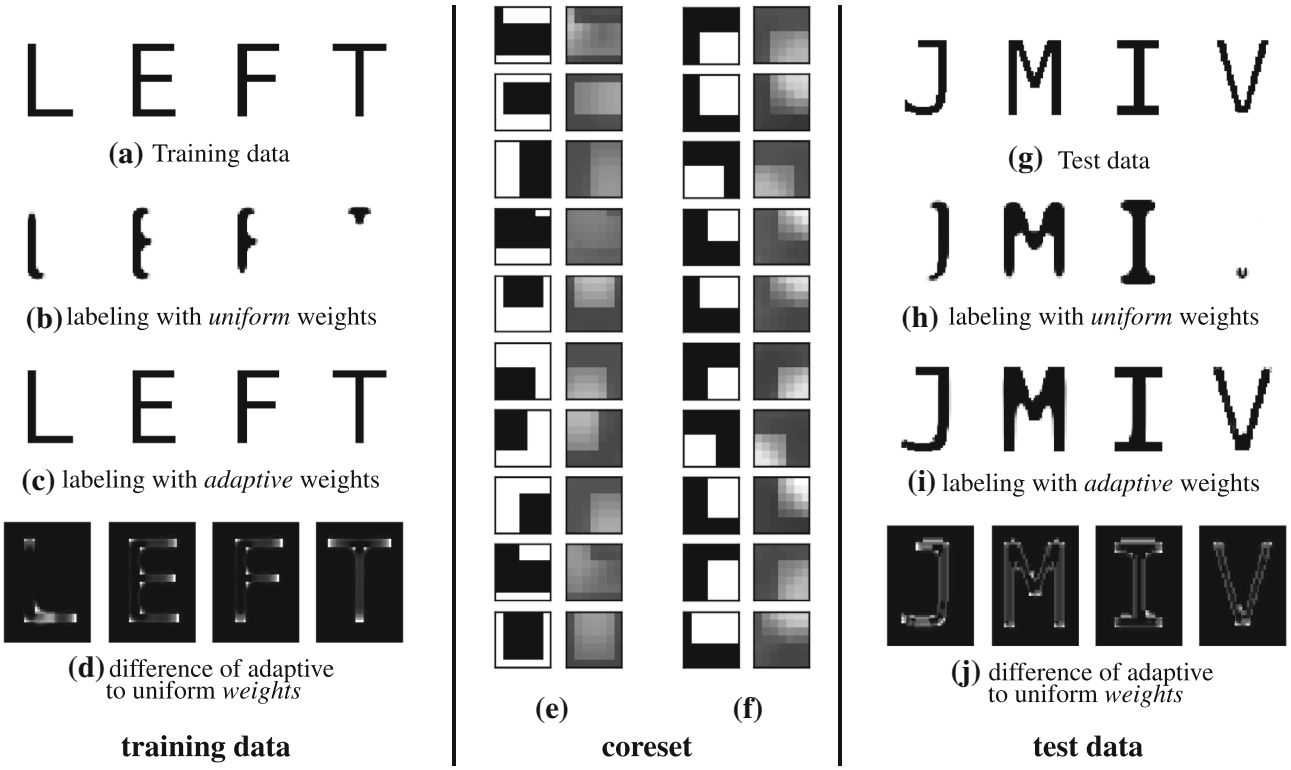


Fig. 3 Adaptive regularization of binary letters. LEFT COLUMN: **a** Training data. **b** Labeling with uniform regularization fails. **c** Perfect adaptive reconstruction (sanity check). **d** illustrates weight adaptivity at each pixel in terms of the KL-divergence of the *learned* weight to the *uniform* weight patch. MIDDLE COLUMN: This column illustrates 10 pairs of image patches and corresponding weights for each class separately. The image patches with the corresponding optimal weight patches are illustrated by **e** for the *foreground* class and by **f** for the *background* class. We observe that the regularizer increases the influence of neighbors on the geometric averaging which belong to the respective class. RIGHT COLUMN: **g** Novel test data to be labeled using the regularizer trained on **a**. **h** Uniform regularization fails **i** Adaptive reconstruction predicts *curvilinear* structures of **e** using 'knowledge' based on **a** where *only vertical and horizontal* structures occur. **j** illustrates weights adaptivity at each pixel (cf. **d**)

The middle column of Fig. 3 illustrates 10 pairs of image patches and corresponding weights for each class separately. By comparing the image patches with the corresponding optimal weight patches (cf. (e) *foreground*, (f) *background*), we observe that the regularizer increases the influence of neighbors on the geometric averaging which belong to the respective class.

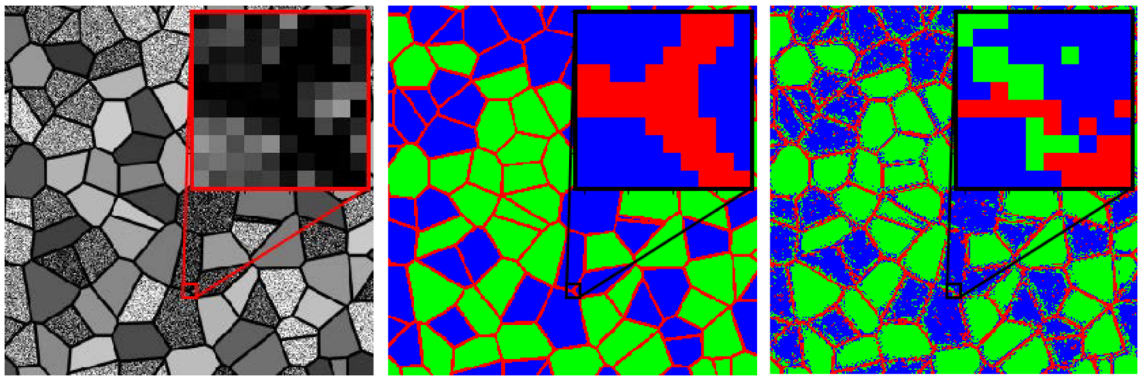
Mapping features to weights For a given *novel* test image, we extracted 7×7 patches from the image, determined the closest image patch of the *coreset* and assigned the corresponding weight patch to pixel i. Note that we used the same size 7×7 for the image patches and for the neighborhood size of geometric averaging.

Inference (labeling novel data) In the test phase, we used the modified linear assignment flow and all parameter values in the same way, as was done during training. The only difference is that *predicted* weight patches were used for regularization, as described above.

Results The right column of Figure 3 shows the results obtained during the test phase. Panel (g) depicts the *novel* (unseen) binary images. The next two panels show the labeling results using uniform weights (h) and using adaptive weights (i). Panel (j) illustrates the weight *adaptivity* by showing the difference of predicted to uniform weights.

5.2 Adaptive Regularization of Curvilinear Line Structures

We consider a collection of images containing line structures induced by random Voronoi diagrams (Fig. 4a). The goal is pixel accurate labeling of any given image with three labels representing: thin *line* structure, *homogeneous* region and *texture*. In the figures below, these labels are encoded by the three colors {■, ■, ■} = {*line*, *homogeneous*, *texture*}. As usual in *supervised machine learning*, our approach is first applied during a *training phase* in order to learn weight adaptivity from ground truth labelings and subsequently evaluated in a *test phase* using *novel unseen data*.



(a) input scene **(b) ground truth** **(c) local rounding of distances** (5.2)

Fig. 4 Training data and local label assignments. The training data consist of 20 pairs of randomly generated images: **a** shows a randomly generated *input image* from which features are extracted, as described in the text, and **b** the corresponding *ground truth*. The ground truth images encode the labels with colors {, , ,} = {line, homogeneous, texture}. Even though the global image structure can be easily assessed by the human eye, assigning correct labels pixelwise by an algorithm requires context-sensitive decisions, as the close-up view illustrates. **c** illustrates the quality of the distances (5.2) between extracted feature vectors. The panel shows the labeling obtained by local rounding, i.e., by assigning to each pixel the label minimizing the corresponding distance. Comparing the close-up views of panel **b**, **c** shows that label assignments to individual pixels are noisy and incomplete (Color figure online)

5.2.1 Training Phase

We used 20 randomly generated images together with ground truth as *training data*: Fig. 4a shows one of these images and Fig. 4b the corresponding ground truth. By following the same procedure as in Sect. 5.1.1, we use these data in order to *adapt the regularization parameter* of the modified linear assignment flow (4.11) by solving problem (4.1), with the specific form given by (4.17).

Feature Vectors The basis of our feature vectors is the outputs of simple 7×7 first- and second-order derivative filters, which are tuned to orientations at $0, 15, \ldots, 180$ degrees. (We took absolute values of filter outputs to eliminate the $180 \sim 360$ degree symmetry.) We reduced the dimension of the resulting feature vectors from 24 to 12 by taking the maximum of the first-order and second-order filter outputs, for each orientation. To incorporate more spatial information, we extracted 3×3 patches from this 12-dimensional feature vector field. Thus, our *feature vectors* f_i, $i \in \mathcal{V}$ had dimension $3 \times 3 \times 12 = 108$ and were given as a point set in the Euclidean feature space $\mathcal{F} = \mathbb{R}^{108}$.

Label Extraction Using ground truth information, we divided all feature vectors extracted from the training data into three classes: thin *line* structure, *homogeneous* region and *texture*. We computed 200 prototypical feature vectors $l_{jc} \in \mathcal{F}$, $j \in [200]$, in each class $c \in$ {line, homogeneous, texture} by k-means clustering. Thus, each *label* (line, homogeneous, texture) was represented by 200 feature vectors in $\mathcal{F}$.

Distance Matrix Even though in the original formulation (3.2) labels are represented by a single feature vector, multiple representatives can be taken into account as well by modifying the distance matrix (3.23) accordingly. With the identification

$$c \in \{\text{line, homogeneous, texture}\} = \{1, 2, 3\},$$

we defined the entries of the distance matrix D_{ic}, for every $i \in \mathcal{V}$, as the distance between f_i and the best fitting representative l_{jc} for class c, i.e.,

$$D_{ic} := \min_{j \in [200]} \| f_i - l_{jc} \|_2. \tag{5.2}$$

The quality of this distance information is illustrated in Fig. 4c that shows the labeling obtained by *local rounding*, i.e., by assigning to each pixel i the label $c = \min_{\tilde{c}} D_{i\tilde{c}}$. Although the result looks similar to the ground truth (cf. Fig. 4b), it is actually quite noisy when looking to single pixels in the close-up view of Fig. 4c.

Optimization For each input image of the training set, we solved problem (4.1) using Algorithms 1 and 2 and the following parameter values: $|\mathcal{N}_i| = 9 \times 9$ (size of local neighborhoods, for every i), $\rho = 1$ (scaling parameter for distance matrix, cf. (3.24)), $h = 0.5$ (constant step-size for computing the gradient with Algorithm 2), and $T = 6$ (end of time horizon). As for optimization on the parameter manifold $\mathcal{P}$ through the Riemannian gradient flow (Algorithm 1), we used an initial value of $h' = 0.0125$ together with backtracking for adapting the step-size. We terminated the iteration once the relative change (5.1) of the objective function dropped below 0.001 or the maximum number of 100 iterations was reached.

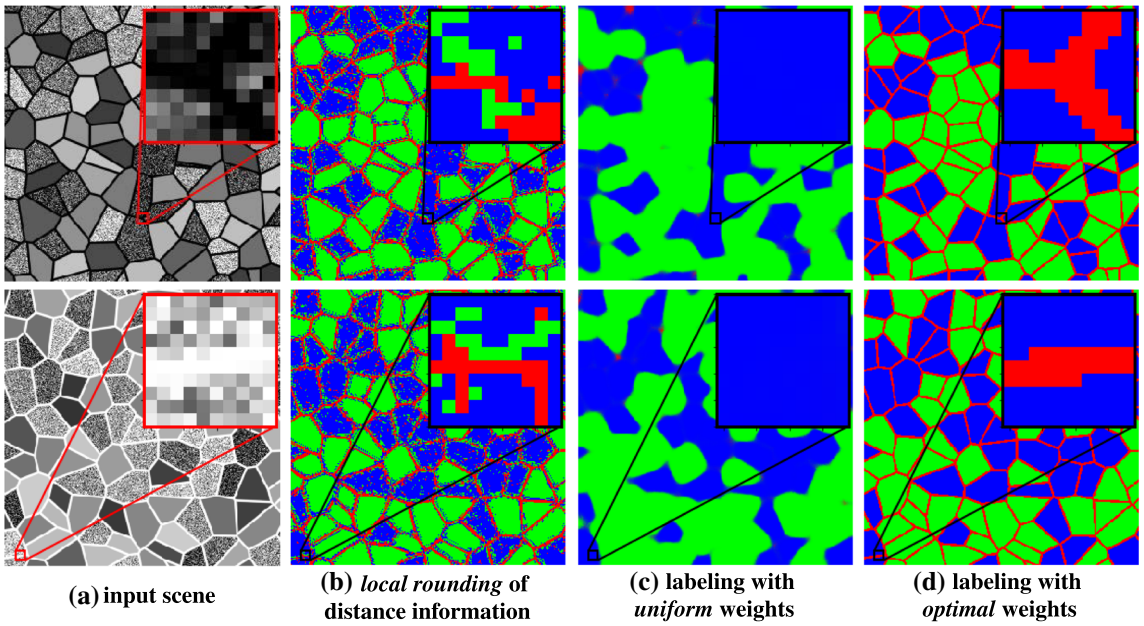


Fig. 5 Training phase: labeling results. This figure shows results of the training phase. Panel **a** shows the given input scene and panel **b** the corresponding *locally rounded* distance information. The labeling with *uniform* regularization (panel **c**) returns *smoothed over* regions and completely fails to preserve the line structures. The *adaptive* regularizer preserves the line structure perfectly (panel **d**), i.e., the *optimal weights* are able to steer the linear assignment flow successfully toward the given ground-truth labeling. ({■, ■, ■} = {line, homogeneous, texture}) (Color figure online)

Results Figure 5 shows two results obtained during the training phase. They illustrate *non-adaptive* regularization using *uniform* weights, which results in blurred partitions and fails completely to detect and label the line structures (panel c). On the other hand, the *adapted* regularizer preserves and restores the structure perfectly (panel d), i.e., the *optimal* weights steered the linear assignment flow toward the given ground-truth labeling.

Figure 6 shows a close-up view of a 10×10 pixel region together with the corresponding 10×10 *optimal weight patches*, extracted from Ω^*. The top row depicts (a) the training data, (b) the corresponding ground truth, (c) the local label assignments, and (d) the labeling obtained when using the learned weights Ω^*. Plot (e) shows the corresponding optimal weight patches $\Omega_i^* = (\omega_{i1}, \ldots, \omega_{i\mathcal{N}})^\top$ associated with every pixel i in the 10×10 pixel region, where small and large weights are indicated by dark and bright gray values, respectively. These weight patches illustrate the result of the *learning process* for adapting the weights. Close to the line structure, the regularizer *increases* the influence (with larger weights) of neighbors whose distance information matches the prescribed ground truth label. Away from the line structure, the regularizer has learned to *suppress* (with small weights) neighbors that belong to a line structure.

5.2.2 Test Phase

As already explained in Sect. 5.1.2, we have to *predict* appropriate weights for novel data and features. We proceed as done before by extracting a *coreset* from the output generated by Algorithm 1 and constructing a map from novel features to weights.

Coreset Let Ω^* denote the set of optimal weight patches generated by Algorithm 1, and let P^* denote the set of all 15×15 patches of local label assignments based on the corresponding training features and distance (5.2). We partitioned P^* into three classes: thin *line structures*, *homogeneous* regions and *texture*, and extracted for each class separately 225 prototypical patches by k-means clustering. To each of these patches and the corresponding cluster, a *prototypical weight patch* was assigned, namely the weighted geometric mean of all *optimal* weight patches in Ω^* belonging to that cluster. As weights for the averaging, we used the Euclidean distance between the respective patches of local label assignments and the corresponding cluster centroid.

Figure 7 depicts 10 pairs of patches of prototypical label assignments and weights, for each of the three classes: line, homogenous and texture. Comparing these weight patches with the optimal patches depicted in Fig. 6, we observe that the former are regularized (smoothed) by geometric averaging and, in this sense, summarize and represent all optimal weights computed during the training phase.

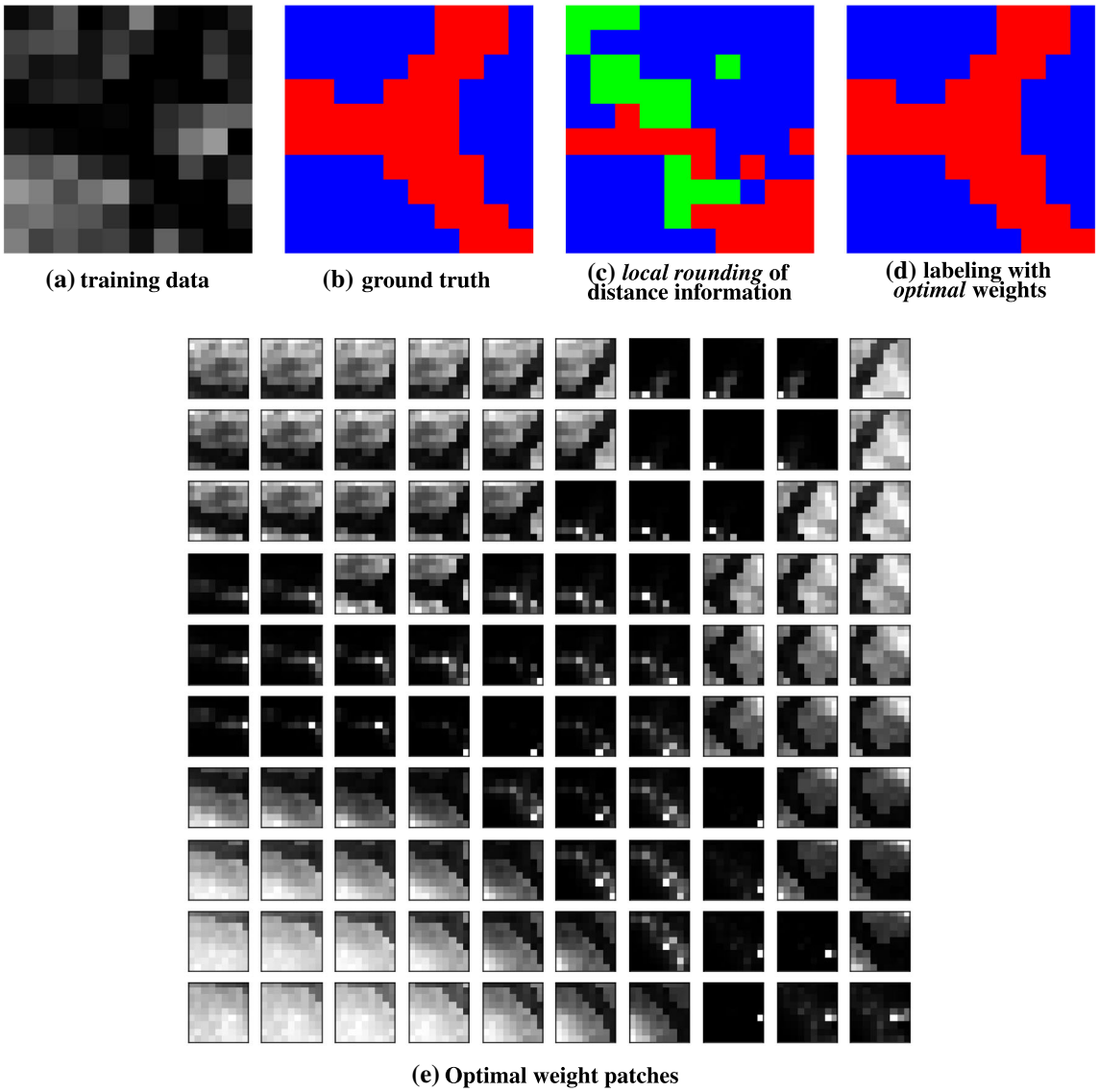


Fig. 6 Training phase: optimal weight patches. TOP ROW: **a** Close-up view of training data (10 × 10 pixel region). **b** The corresponding ground truth section. **c** Local label assignments. **d** Correct labeling using adapted optimal weights. BOTTOM ROW: **e** The corresponding optimal weight patches (10 × 10 grid), one patch for each pixel. Close to the line structure, the regularizer increases the influence of neighbors on the geometric averaging of assignments whose distances match the prescribed ground truth labels. Away from the line structure, the regularizer has learned to *suppress* with small weights neighbors belonging to a line structure

Mapping features to weights For each *novel* test image, we extracted features using the same procedure as done in the *training phase* and computed at each pixel i the patch of local label assignments. For the latter patch, the closest patch of local label assignments of the *coreset* was determined, and the corresponding weight patch was assigned to pixel i.

Note that the patch size 15×15 of local label assignments was chosen larger as the patch size 9×9 of the weights that was used both during training and for testing. The former larger neighborhood defines the local 'feature context' that is used to predict weights for novel data.

Inference (labeling novel data) In the test phase, we used the modified linear assignment flow and all parameter values in the same way, as was done during training. The only difference is that *predicted* weight patches were used for regularization, as described above.

Results Figure 8 shows a result of the test phase. Since all data are *randomly* generated, this result is representative for the entire image class. The panels (a) and (g) show the input data, whereas ground truth (b) is only shown for visual comparison. Panel (c) shows the labeling obtained using uniform weights and (d) illustrates the difference of (c) to the

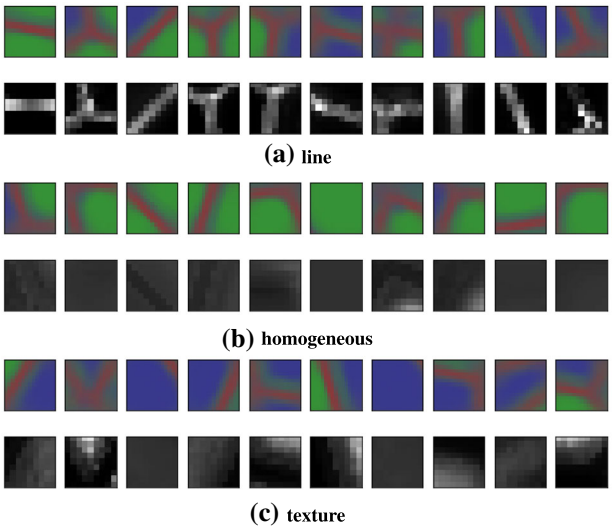


Fig. 7 Coreset visualization. This plot shows 3×10 prototypical patches of local label assignments and the corresponding weight patches of the coreset, for each of the three classes. **a** 10 prototypical pairs of the class *line*. Weight patches 'know' to which neighbors large weights have to be assigned, such that the local line structure is labeled correctly. **b** Weight patches of the *homogeneous* label class are almost uniform, which is plausible, because the noisy assignments can be filtered most effectively. **c** The weight patches of the *texture* label are comparable to the *homogeneous* ones and almost uniform, for the same reason. (Color code {■, ■, ■} = {line, homogeneous, texture}) (Color figure online)

ground truth (b). Panel (e) shows the labeling obtained using adaptive weights, and (f) the corresponding difference of (e) to the ground truth (b). The labeling result clearly demonstrated the impact of weight *adaptivity*. This aspect is further illustrated in panel (h).

Figure 9 shows *predicted* weight patches for novel test data in the same format as Fig. 6 depicts *optimal* weight patches computed during training. The similarity of the behavior of predicted and optimal weights for pixels close and away from local line structure demonstrates that the approach generalizes well to novel data. Since these data are randomly generated, this performance is achieved for any image data in this class.

5.3 Pattern Formation by Label Transport

In this section, we illustrate the *model expressiveness* of the assignment flow. Specifically, we choose an input image and a target labeling which patterns are *quite different*. The task is to estimate weights in order to steer the assignment flow to the target labeling. We show that our learning approach can determine the weights that 'connect' these patterns by the assignment flow. This shows that the weights which determine the regularization properties of the assignment flow actually encode information for *pattern formation*. Finally,

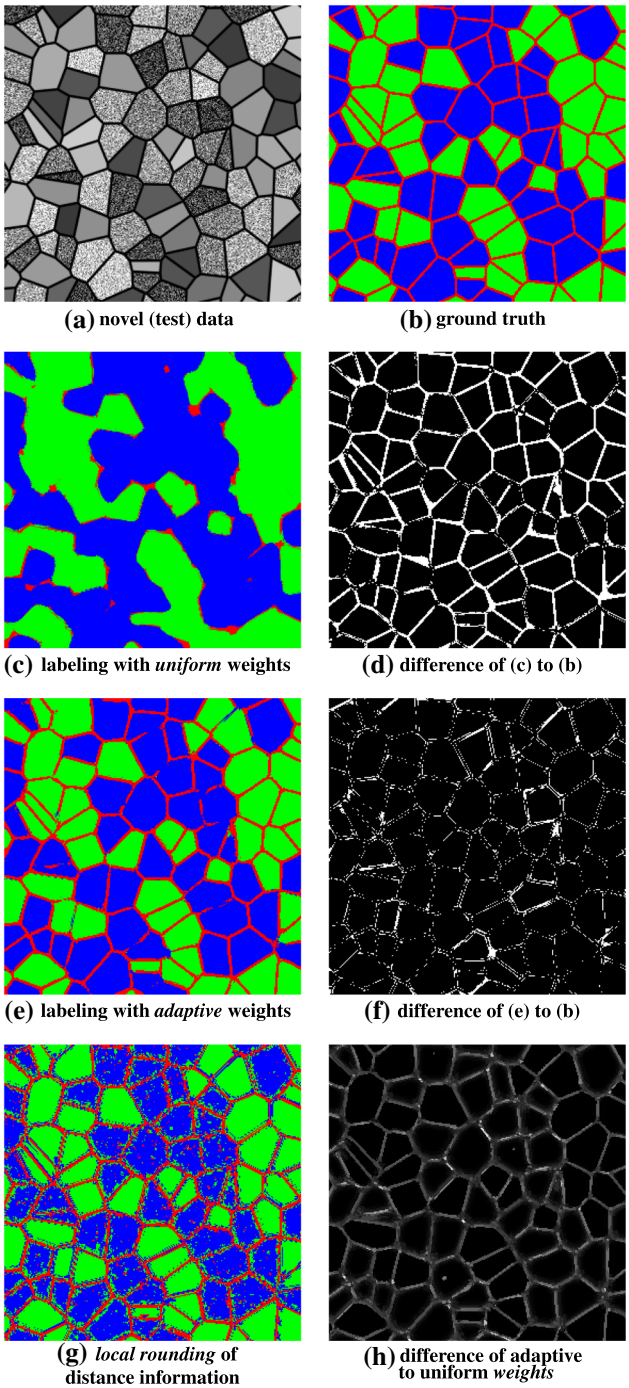


Fig. 8 Test phase: labeling results. **a** Randomly generated novel input data, **b** the corresponding ground truth. **c** Labeling using *uniform* weights fails to detect and label line structures. **d** illustrates the difference of **c** to the ground truth (**b**). **e** *Adaptive* regularizer based on *predicted* weights yields a result that largely agrees with ground truth. **f** shows the difference of **e** to the ground truth (**b**). **g** shows the corresponding *locally rounded* distance information extracted from the image data (**a**). Panel **h** illustrates weights adaptivity at each pixel in terms of the distance of the *predicted* weight patch to the *uniform* weight

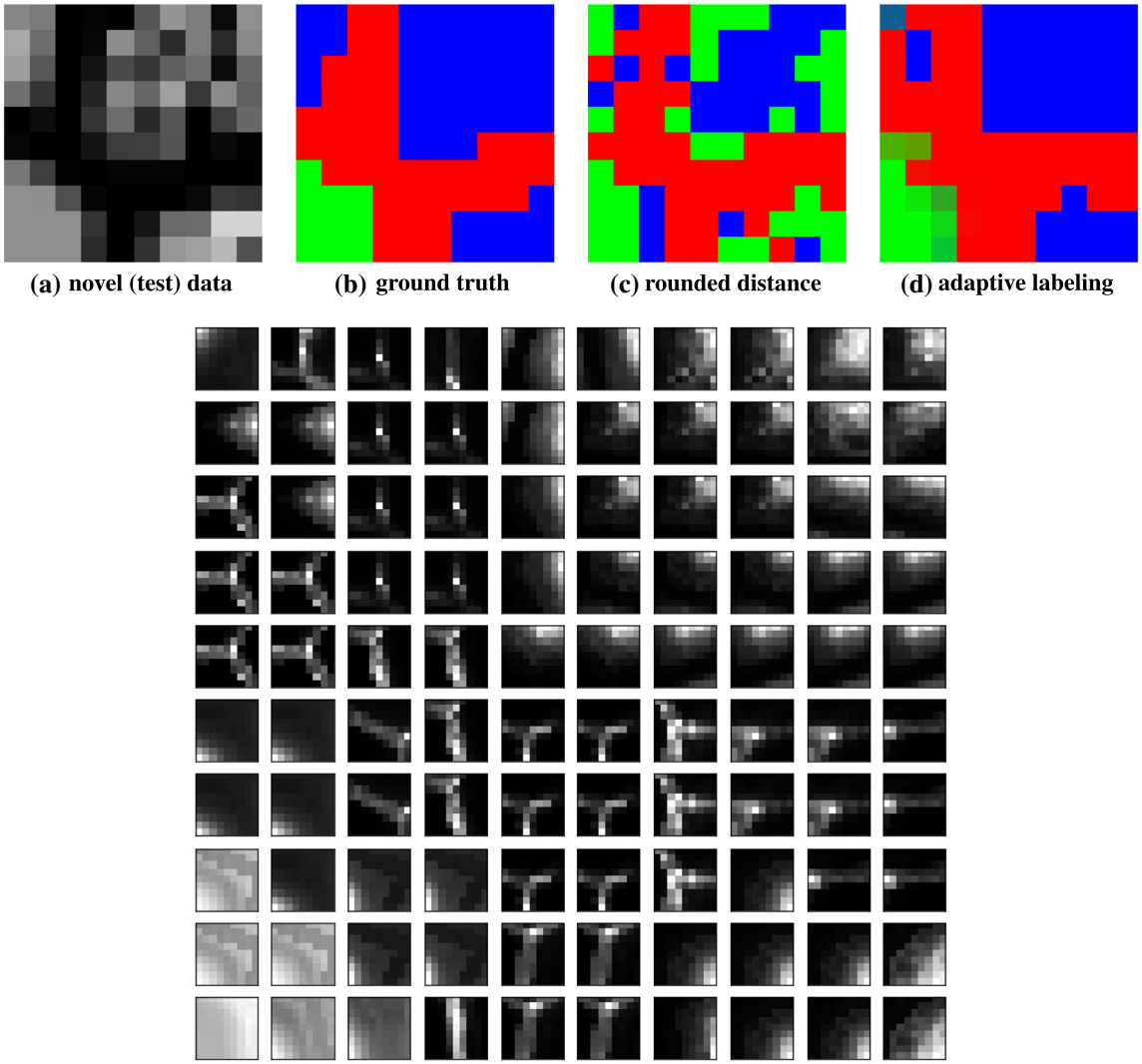


Fig. 9 Test phase: predicted weight patches. TOP ROW: **a** Close-up view of *novel* data (10×10 pixel window). **b** Corresponding ground truth section (just for visual comparison, *not* used in the experiment). **c** Local label assignment. **d** Labeling result using adaptive regularization with predicted weights. BOTTOM: **e** Corresponding predicted weight patches (10×10 grid), one patch for each pixel of the test data (**a**). The *predicted* weight patches behave similar to the *optimal* weight patches depicted in Fig. 6 that were computed during the training phase (for different data). This shows that our approach generalizes to novel data

we briefly point out and illustrate in Sect. 5.3.3 limitations of the current version of our approach.

For the patterns below, we used $\mathcal{X} = \{\square, \blacksquare\} = \{\textit{background}, \textit{foreground}\}$ as labels and the Hamming distance for the computation of the distance matrix (3.23).

5.3.1 Pattern Completion

The *top row* of Fig. 10 shows the *input image* and the *target labeling*. The second row illustrates the evolution of the *linear* assignment flow using optimal weight parameters. These optimal parameters were obtained by the Riemannian gradient flow on the parameter manifold in order to solve problem (4.17), which effectively steers the assignment flow to the target labeling.

Having obtained the optimal weights Ω^* after convergence, we inserted them into the original *nonlinear* assignment flow. The evolution of corresponding label assignments is shown by the third row of Fig. 10. The fact that the label assignment at the final time T is close to the target labeling which the linear assignment flow reaches exactly confirms the remarkably close approximation of the nonlinear flow by the linear assignment flow, as already demonstrated in [25] in a completely different way.

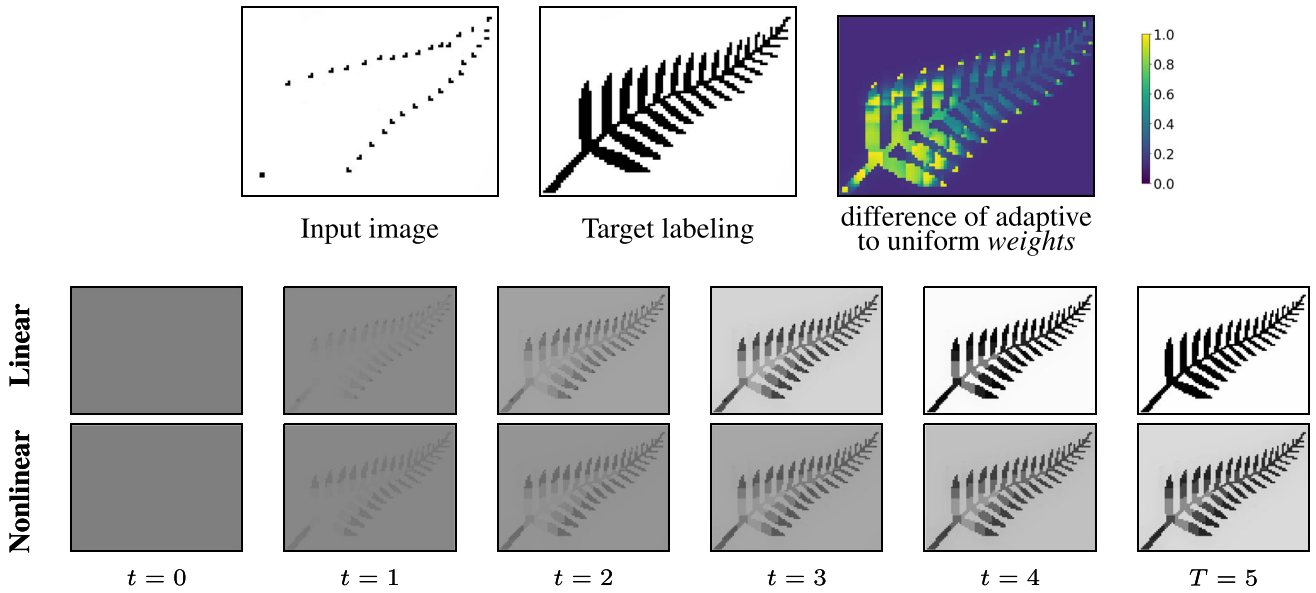


Fig. 10 Pattern completion. This figure illustrates the model expressiveness of the assignment flow. TOP ROW: Input image and target labeling. The task was to estimate weights in order to steer the assignment flow to the target labeling. The rightmost panel illustrates, for each pixel, the distance of *uniform* weights from the *optimal* estimated weight patch. MIDDLE ROW: Label assignments of the linear assignment flow using optimal weights obtained by solving (4.1). The Riemannian gradient flow on the parameter manifold effectively steers the flow to the target labeling. BOTTOM ROW: Label assignments of the *nonlinear* assignment flow using the optimal weights that were estimated using the *linear* assignment flow. Closeness of both labeling patterns at the final point of time $T = 5$ demonstrates that the linear assignment flow provides a good approximation of the full nonlinear flow

The rightmost panel in the top row of Fig. 10 shows, for each pixel, the deviation of the optimal weight patch that forms uniform weights. While it is obvious that the 'source labeling' of the input data receives large weights, the spatial arrangement of weights at all other locations is hard to predict beforehand by humans. This is why *learning* them is necessary.

5.3.2 Transporting and Enlarging Label Assignments

We repeated the experiment of the previous section using the academic scenario depicted in Fig. 11. A major difference is that locations of the input image *do not* form a subset of the locations of the target labeling. As a consequence, the corresponding 'mass' of assignments has to be both *transported and enlarged*.

The results shown in Fig. 11 closely resemble those of Fig. 10, such that the corresponding comments apply likewise. We just point out again the following: Looking at the optimal weight patches in terms of their deviation from uniform weights, as depicted in the rightmost panel in the top row of Fig. 11, it is both interesting and not too difficult to understand—after convergence and informally by visual inspection—how these weights encode this particular 'label transport.' However, predicting these weights and certifying their optimality *beforehand* seems to be an infeasible task. For example, it is hard to predict that the creation of intermediate locations where assignment mass temporarily accumulates (clearly visible in Fig. 11) effectively optimizes the constrained functional (4.1). *Learning* these weights, on the other hand, just requires to apply our approach.

5.3.3 Parameter Learning Versus Optimal Control

Figure 12 illustrates limitations of our parameter learning approach. In this experiment, we simply *exceeded* the time horizon in order to inspect labelings induced by the linear assignment flow *after* the point of time T, that was used for determining optimal weights in the training phase. Starting with T, Fig. 12 shows these labelings for both experiments corresponding to Figs. 10 and 11.

Unlike the fern pattern (top row) where the initial label locations formed a subset of the target locations, the 'moving mass pattern' (bottom row) is *unsteady* in the following quite natural sense: The linear assignment flow simply continues transporting mass beyond time T. As a result, assignments to the white label are transported to locations of the black target pattern. Hence, the target pattern is first created up to time T and destroyed afterward.

This behavior is not really a limitation, but a consequence of merely learning *constant* weight parameters. Due to the formulation of the optimization problem (4.1), opti-

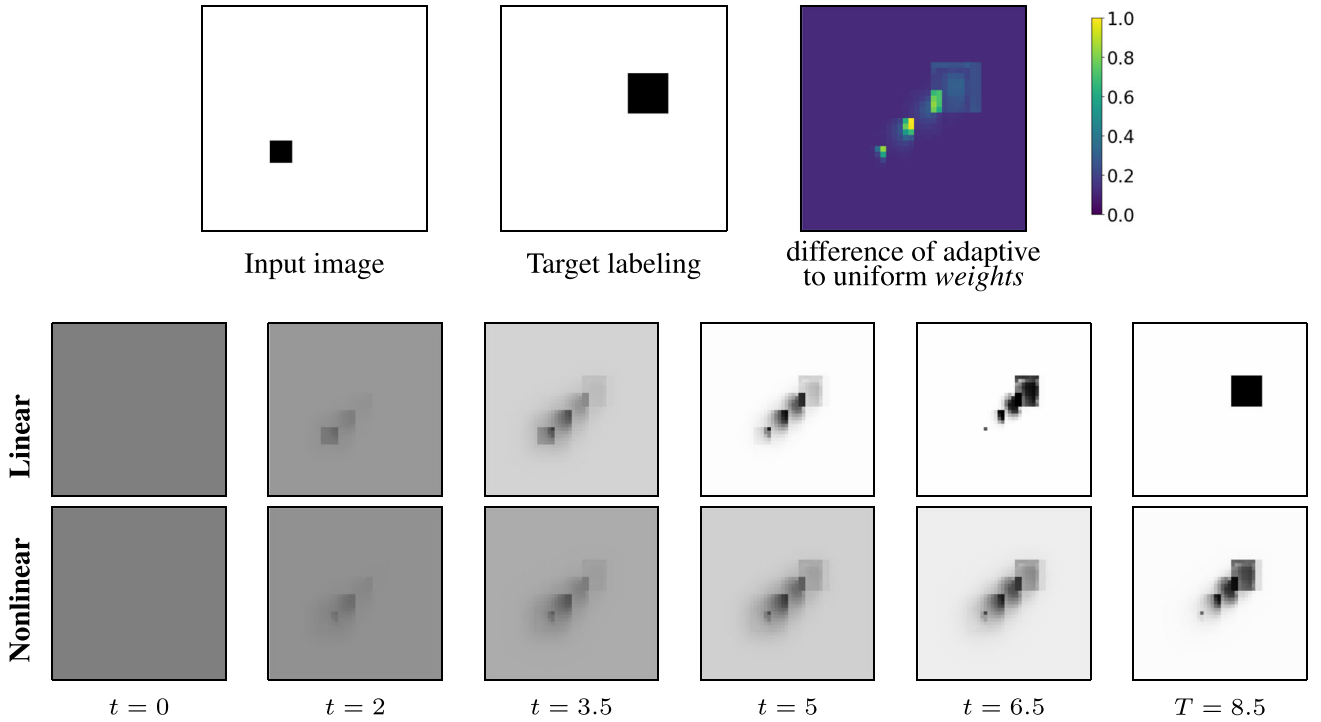


Fig. 11 Transporting and Enlarging Label Assignments. See Fig. 10 for the setup. TOP ROW: Label locations of the input data *do not* form a subset of the target locations. Thus, 'mass' of label assignments has to be both transported and enlarged. Rightmost panel: Distance of the *optimal* weight patch from uniform weights, for every pixel. MIDDLE ROW: Applying our approach to (4.1) effectively solves the problem. BOTTOM ROW: Inserting the optimal weights that are computed using the *linear* assignment flow into the *nonlinear* assignment flow gives a similar result and underlines the good approximation property of the linear assignment flow. It is interesting to observe that computing the Riemannian gradient flow on the parameter manifold entails 'intermediate locations' where assignment mass accumulates temporarily. This underlines the necessity of learning, since it seems hard to predict such an *optimal* regularization strategy beforehand

mal weights not only encode the 'knowledge' how to steer the assignment flow in order to solve the problem, but also the *time period* after which the task has to be completed. Fixing this issue requires a higher-level of adaptivity: Weight *functions* depending on time and the current state of assignments would have to be estimated, that may be adjusted online through feedback in order to *control* the assignment flow in a more flexible way.

6 Conclusion

We introduced a parameter learning approach for image labeling based on the assignment flow. During the training phase, weights for geometric averaging of label assignments are estimated from ground truth labelings, in order to steer the flow to prescribed labelings. Using the linearized assignment flow, we showed that, by using a class of symplectic partitioned Runge–Kutta methods, this task can be accomplished by numerically integrating the adjoint system in a consistent way. Consistent means that discretization and differentiation for the training problem *commute*. An additional convenient property of our approach is that the parameter manifold has mathematically the structure of an assignment manifold, such that Riemannian gradient descent can be used for effectively solving the training problem.

The output of the training phase is a database containing features extracted from training data, together with the respective optimal weights. In order to complete the parameter learning task, a mapping has to be specified that predicts optimal weights for novel unseen data. We solved this task simply by nearest-neighbor prediction after partitioning the database using k-means clustering and geometric averaging of the weights, separately for each cluster. We evaluated this approach for a binary label scenario consisting of letters and a 3-label scenario involving line structures where just using uniform weights inevitably fails in both cases. We additionally conducted experiments that highlight the model expressiveness of the assignment flow and also limitations caused by merely learning *constant* parameters.

Our main insights include the following. Regarding numerical optimization for parameter learning in connection with image labeling, our approach is more satisfying than working with discrete graphical models, where parameter

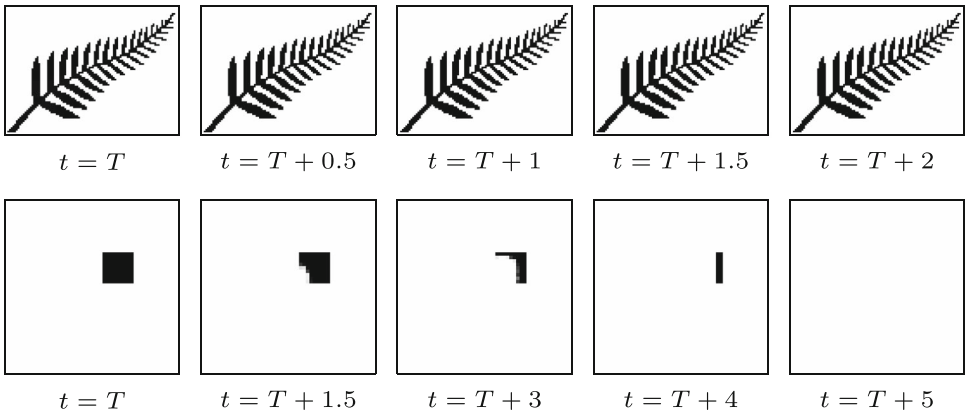


Fig. 12 Parameter learning vs. optimal control. The plots show label assignments by computing the assignment flow *beyond* the final point of time T used during training, for the experiments corresponding to Figs. 10 and 11. Unlike the pattern completion experiment (top row) where few locations of initial data formed a subset of the target pattern, the target pattern (bottom row, at time T) of the moving-mass experiment is *unsteady* in the following sense: At time T, the flow continues to transport mass which eventually erases the target pattern with assignments of the white background label. The reason is that *constant* parameters are only learned that not only encode the 'knowledge' how to steer the flow to the target pattern but also the time period $[0, T]$ for accomplishing this task. In order to remedy this limitation, weight *functions* depending on the assignments (state of the assignment flow) would have to be estimated by applying techniques of optimal control

learning requires to evaluate the partition function, which is a much more involved task when working with cyclic grid graphs. The latter problem of computational *statistics* shows up in our scenario in similar form as the problem to design the prediction map from features to weight parameters. A key difference of these two scenarios is that by restricting the scope to statistical predictions at a *local* scale, i.e., only within small windows, the prediction task becomes *manageable*, since regarding numerical optimization, no further approximations are involved at all.

Regarding future work, we mention two directions. The natural way for broadening the scope of the prediction map and the class of images that the assignment flow can represent is the composition of two or several assignment flows in a hierarchical fashion. This puts our work closer to current mainstream research on deep networks, whose parametrizations and internal representations are not fully understood, however. We hope that using the assignment flow can help to understand hierarchical architectures better.

The second line of research concerns the learning of weight *functions*, rather than constant parameters, as motivated in Sect. 5.3.3, since this would also enhance model expressiveness and adaptivity considerably. A key problem then is to clarify the role of these functions and the choice of an appropriate time scale, as part of an hierarchical composition of assignment flows.

Acknowledgements Open Access funding provided by Projekt DEAL. Part of this research was performed while R. Hühnerbein was visiting the Institute for Pure and Applied Mathematics (IPAM) at UCLA, which is supported by the National Science Foundation (Grant No. DMS-1440415). Financial support by the German Science Foundation (DFG), Grant GRK 1653, is gratefully acknowledged. This work has also been stimulated by the Heidelberg Excellence Cluster STRUCTURES, funded by the DFG under Germany's Excellence Strategy EXC-2181/1 - 390900948.

Proofs of Section 2

Proof of Theorem 6

A proof can be found, e.g., in [4]. However, in order to make this paper self-contained, we include a proof here.

Proof Setting up the Lagrangian

$$\mathcal{L}(x, p, \lambda) = \mathcal{C}(x(T)) - \int_0^T \big\langle \lambda, F(\dot{x}, x, p, t) \big\rangle dt \qquad (A.1)$$

with multiplier $\lambda(t)$ and $F(\dot{x}, x, p, t) := \dot{x} - f(x, p, t) \equiv 0$, we get with $\Phi(p) = \mathcal{C}(x(T))$ from (2.1)

$$\begin{aligned}\partial\Phi = \partial_p\mathcal{L} = d_p x(T)^\top \partial\mathcal{C}\big(x(T)\big) - \int_0^T \Big(d_{\dot{x}} F d_p \dot{x} \\ + d_x F d_p x + d_p F\Big)^\top \lambda \,\mathrm{d}t,\end{aligned} \tag{A.2}$$

where integration applies component-wise. By using $d_{\dot{x}} F = I$, where I denotes the identity matrix, we partially integrate the first term under the integral,

$$\int_0^T d_p \dot{x}^\top \lambda \, dt = d_p x^\top \lambda \Big|_{t=0}^T - \int_0^T d_p x^\top \dot{\lambda} \,\mathrm{d}t. \tag{A.3}$$

We further obtain with $d_p F = -d_p f$ and $d_x F = -d_x f$

$$\begin{aligned}\partial\Phi &= d_p x(T)^\top \partial\mathcal{C}(x(T)) - d_p x^\top \lambda \Big|_{t=0}^T \\ &\quad + \int_0^T d_p x^\top \dot{\lambda} \,\mathrm{d}t \qquad \text{(A.4a)} \\ &\quad + \int_0^T \Big(d_x f d_p x + d_p f\Big)^\top \lambda \,\mathrm{d}t \\ &= d_p x(T)^\top \partial\mathcal{C}(x(T)) - d_p x(T)^\top \lambda(T) \\ &\quad + d_p x(0)^\top \lambda(0) \qquad \text{(A.4b)} \\ &\quad + \int_0^T d_p x^\top \dot{\lambda} + d_p x^\top d_x f^\top \lambda + d_p f^\top \lambda \,\mathrm{d}t.\end{aligned}$$

We consider systems where the initial value x_0 is independent of the parameter p, i.e., $d_p x(0) = 0$. Additionally factoring out the unknown Jacobian $d_p x$, we obtain

$$\begin{aligned}&= d_p x(T)^\top \Big(\partial\mathcal{C}(x(T)) - \lambda(T)\Big) \\ &\quad + \int_0^T d_p x^\top \Big(\dot{\lambda} + d_x f^\top \lambda\Big) + d_p f^\top \lambda \,\mathrm{d}t.\end{aligned} \tag{A.4c}$$

Now, by choosing $\lambda(t)$ such that conditions (2.18b) are fulfilled, i.e.,

$$\dot{\lambda}(t) = -d_x f^\top \lambda(t), \quad \lambda(T) = \partial_x \mathcal{C}(x(T)),$$

we finally obtain

$$\partial\Phi = \int_0^T d_p f^\top \lambda(t) \,\mathrm{d}t. \tag{A.5}$$

□

Proof of Theorem 7

For the proof of this theorem, we follow the suggested outline of [19]: State the Lagrangian of the nonlinear problem (2.21) and apply the following lemma, which is a slightly different version of Lemma 3.5 in [19].

Lemma 3 *Suppose that the mapping $\phi\colon \mathbb{R}^{n_p \times d'} \to \mathbb{R}^{d'}$ is such that the Jacobian matrix $d_\gamma \phi$ is invertible at a point $(p_0, \gamma_0) \in \mathbb{R}^{n_p} \times \mathbb{R}^{d'}$, that is in the neighborhood of p_0, the equation $\phi(p, \gamma) = 0$ defines γ as a function of p. For some given function $\mathcal{C}\colon \mathbb{R}^{n_p \times d'} \to \mathbb{R}$ consider the induced function of the form $\Phi\colon \mathbb{R}^{n_p} \to \mathbb{R}$, defined by $\Phi(p) := \mathcal{C}(p, \gamma(p))$. We introduce the Lagrangian*

$$\mathcal{L}(p, \gamma, \lambda) = \mathcal{C}(p, \gamma) + \langle \phi(p, \gamma), \lambda \rangle. \tag{A.6}$$

Then, the Euclidean gradient of Φ with respect to p at p_0 is given by

$$\partial\Phi(p_0) = \partial_p \mathcal{L}(p_0, \gamma_0, \lambda_0), \tag{A.7}$$

where the vectors $\gamma_0 = \gamma(p_0) \in \mathbb{R}^{d'}$ and $\lambda_0 \in \mathbb{R}^{d'}$ are uniquely determined by

$$0 = \partial_\lambda \mathcal{L}(p_0, \gamma_0, \lambda_0) = \phi(p_0, \gamma_0), \tag{A.8a}$$

$$\begin{aligned}&0 = \partial_\gamma \mathcal{L}(p_0, \gamma_0, \lambda_0) \\ &\iff \quad \partial_\gamma \mathcal{C}(p_0, \gamma_0) = -d_\gamma \phi(p_0, \gamma_0)^\top \lambda_0.\end{aligned} \tag{A.8b}$$

Proof Since we evaluate all occurring functions and their derivatives at the same points p_0, γ_0 and λ_0, we drop them as arguments in the following, to simplify notation.

(i) Equation (A.8a) directly follows by differentiating $\mathcal{L}$ with respect to λ at $(p_0, \gamma_0, \lambda_0)$.

(ii) Equation (A.8b) is immediately obtained by differentiating $\mathcal{L}$ with respect to γ at $(p_0, \gamma_0, \lambda_0)$. Since $d_\gamma \phi$ is invertible at (p_0, γ_0), the resulting linear system uniquely determines the vector λ_0.

(iii) Next, we show that this λ_0 also satisfies the first equation (A.7). By differentiating $\phi(p, \gamma) = 0$ with respect to p at (p_0, γ_0), we obtain

$$\begin{aligned}&d_\gamma \phi d_p \gamma_0 + d_p \phi = 0 \\ &\overset{d_\gamma\phi \text{ is invertible}}{\iff} \quad d_p \gamma_0 = -(d_\gamma \phi)^{-1} d_p \phi.\end{aligned} \tag{A.9}$$

We will make use of this identity for $d_p \gamma_0$ in the following. Differentiating Φ with respect to p at p_0 and by the chain rule, we obtain

$$\partial\Phi = \partial_p \mathcal{C} + d_p \gamma_0^\top \partial_\gamma \mathcal{C} \tag{A.10a}$$

$$\overset{(A.8b)}{=} \partial_p \mathcal{C} - d_p \gamma_0^\top d_\gamma \phi^\top \lambda_0 \tag{A.10b}$$

$$\overset{(A.9)}{=} \partial_p \mathcal{C} + \big((d_\gamma \phi)^{-1} d_p \phi\big)^\top d_\gamma \phi^\top \lambda_0 \tag{A.10c}$$

$$= \partial_p \mathcal{C} + d_p \phi^\top \lambda_0 = \partial\mathcal{L}, \tag{A.10d}$$

which shows (A.7). □

Proof of Theorem 7 We begin by stating the Lagrangian of problem (2.21)

$$\begin{aligned}\mathcal{L}(x,\,p,\,\lambda) &= \mathcal{C}\big(x_N\big) - \lambda_0^\top (x_0 - x(0)) \\ &\quad - \sum_{n=0}^{N-1} \lambda_{n+1}^\top \Big[x_{n+1} - x_n - h_n \sum_{i=1}^{s} b_i k_{n,i}\Big] \\ &\quad - \sum_{n=0}^{N-1} h_n \sum_{i=1}^{s} b_i \Lambda_{n,i}^\top \Big[k_{n,i} - f(X_{n,i},\,p,\,t_n + c_i h_n)\Big].\end{aligned} \tag{A.11}$$

In order to apply Lemma 3, we explain which role the variables γ, λ, ϕ play in this situation:

1. *Intermediate stages:* The vector γ represents all intermediate stages related to the evaluation of the function $\Phi(p) = \mathcal{C}(x_N(p))$, i.e., all intermediate values x_i and stages k_i of the Runge–Kutta method. These variables are stacked and arranged as follows:

$$= \begin{bmatrix} x_0 \\ \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{N-1} \end{bmatrix} \in \mathbb{R}^{d'}, \quad \gamma_n = \begin{bmatrix} k_n \\ x_{n+1} \end{bmatrix} \in \mathbb{R}^{(s+1)n_x}, \tag{A.12}$$

$$\text{and} \quad k_n = \begin{bmatrix} k_{n,1} \\ \vdots \\ k_{n,s} \end{bmatrix} \in \mathbb{R}^{sn_x}.$$

2. *Lagrange multiplier* The vector λ contains all Lagrange multipliers in (A.11) belonging to the constraints (2.21b)–(2.21d). The multipliers are stacked and arranged as follows:

$$\lambda = \begin{bmatrix} -\lambda_0 \\ -\Lambda_0 \\ \vdots \\ -\lambda_{N-1} \\ -\Lambda_{N-1} \\ -\lambda_N \end{bmatrix} \in \mathbb{R}^{d'}, \quad \Lambda_n = \begin{bmatrix} h_n b_1 \Lambda_{n,1} \\ \vdots \\ h_n b_s \Lambda_{n,s} \end{bmatrix} \in \mathbb{R}^{sn_x}. \tag{A.13}$$

3. *Intermediate mappings* Analogously, the vector ϕ contains all intermediate mappings ϕ_n of the computation of $\Phi(p) = \mathcal{C}(x_N(p))$ with $n = 1, \ldots, N-1$. In our situation, ϕ is the concatenation of the *forward* Runge–Kutta evaluation, which we express using the Kronecker product as

$$\begin{aligned}\phi &= \begin{bmatrix} x_0 - x(0) \\ \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_{N-1} \end{bmatrix} \in \mathbb{R}^{d'}, \\ \Psi_n &= \begin{bmatrix} k_n - F_n(X_n,\,p) \\ x_{n+1} - x_n - h_n (b^\top \otimes I_{n_x}) k_n \end{bmatrix} \\ &= \begin{bmatrix} \Psi_{n,1} \\ \Psi_{n,2} \end{bmatrix} \in \mathbb{R}^{(s+1)n_x},\end{aligned} \tag{A.14}$$

where $\Psi_{n,1} \in \mathbb{R}^{sn_x}$ and $\Psi_{n,2} \in \mathbb{R}^{n_x}$, as well as

$$\begin{aligned}F_n(X_n,\,p) &= \begin{bmatrix} f(X_{n,1},\,p,\,t_n + c_1 h_n) \\ \vdots \\ f(X_{n,s},\,p,\,t_n + c_s h_n) \end{bmatrix}, \\ X_n &= \mathbb{1}_s \otimes x_n + h_n (A \otimes I_{n_x}) k_n = \begin{bmatrix} X_{n,1} \\ \vdots \\ X_{n,s} \end{bmatrix}.\end{aligned} \tag{A.15}$$

We proceed by computing the Jacobian $d_\gamma \phi$. Note that the intermediate variables γ_n (A.12) are only contained in the intermediate mappings Ψ_n (A.14), which results in a sparse block structure of the overall Jacobian $d_\gamma \phi$.

1. *Small block matrices* Each small block matrix represents the derivative of the nth iteration step Ψ_n and is given by

$$\begin{aligned}d_{(x_n,k_n,x_{n+1})} \Psi_n &= \begin{bmatrix} d_{x_n}\Psi_{n,1} & d_{k_n}\Psi_{n,1} & d_{x_{n+1}}\Psi_{n,1} \\ d_{x_n}\Psi_{n,2} & d_{k_n}\Psi_{n,2} & d_{x_{n+1}}\Psi_{n,2} \end{bmatrix} \\ &= \begin{bmatrix} D_n & A_n \\ -I_{n_x} & B_n^\top & I_{n_x} \end{bmatrix},\end{aligned} \tag{A.16}$$

with

$$A_n = I_{sn_x} - h_n d_x F_n(X_n,\,p)(A \otimes I_{n_x}), \tag{A.17a}$$

$$B_n^\top = -h_n b^\top \otimes I_{n_x}, \tag{A.17b}$$

$$D_n = -d_x F_n(X_n,\,p)(\mathbb{1}_s \otimes I_{n_x}), \tag{A.17c}$$

where A and b are the Runge–Kutta coefficients given by the above tableau of Fig. 1.

2. *Sparse block structure* The overall Jacobian $d_\gamma \phi$ consists of $N-1$ blocks (one for each iteration) of the form (A.16)

and is given by

$$d_\gamma \phi = \begin{bmatrix} I_{n_x} & & & & & & & \\ D_1 & A_1 & & & & & & \\ -I_{n_x} & B_1^\top & I_{n_x} & & & & & \\ & & D_2 & A_2 & & & & \\ & & -I_{n_x} & B_2^\top & I_{n_x} & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & & D_{N-1} & A_{N-1} & \\ & & & & & -I_{n_x} & B_{N-1}^\top & I_{n_x} \end{bmatrix}. \tag{A.18}$$

3. *Invertibility of* $d_\gamma \phi$: A matrix M is invertible if $\det M \neq 0$. Since the matrix $d_\gamma \phi$ is lower block diagonal, its determinant is given by

$$\det d_\gamma \phi = \det A_1 \cdot \ldots \cdot \det A_{N-1} \tag{A.19}$$

Thus, we only need to show that $\det A_n \neq 0$ for all $n = 1, \ldots, N-1$. Equation (A.17a) reads in a more compact form

$$\begin{aligned} &A_n = (I_{sn_x} - h_n M), \text{ with} \\ &M := d_x F_n(X_n, p)(A \otimes I_{n_x}). \end{aligned} \tag{A.20}$$

We show $\det A_n \neq 0$ by using the equivalent statement $\ker(A_n) = \{0\}$. Now, let $x \in \mathbb{R}^{sn_x} \backslash \{0\}$, then

$$\begin{aligned} \|x\| &\leq \|x - h_n M x\| + \|h_n M x\| \\ &\leq \|A_n x\| + \|h_n M\| \|x\|. \end{aligned} \tag{A.21}$$

By using the row-sum norm $\|\cdot\|_\infty$, we have

$$\begin{aligned} \|h_n M\|_\infty &\overset{(A.20)}{=} h_n \|d_x F_n(X_n, p)(A \otimes I_{n_x})\|_\infty \\ &\leq h_n \|d_x F_n(X_n, p)\|_\infty \|(A \otimes I_{n_x})\|_\infty \\ &< h_n L \max_{i=1,\ldots,s} \sum_{j=1}^{s} |a_{ij}| \overset{(2.11)}{<} 1, \end{aligned} \tag{A.22}$$

where L denotes the Lipschitz constant of f and the step-size h_n satisfies the assumption (2.11). Substituting (A.22) into (A.21) gives

$$\begin{aligned} \|x\| < \|A_n x\| + \|x\| &\iff 0 < \|A_n x\| \\ &\iff x \notin \ker(A_n). \end{aligned} \tag{A.23}$$

Since, the kernel of A_n is trivial, A_n is invertible and consequently the overall Jacobian $d_\gamma \phi$ as well.

Now, we are in a position to apply Lemma 3. More precisely, (A.8b) tells us that the vector λ is uniquely determined by the linear system $d_\gamma \phi^\top \lambda = -\partial_\gamma \mathcal{C}$. In our situation, this system is given by

$$\begin{bmatrix} I_{n_x} & D_1^\top & -I_{n_x} & & & & & \\ & A_1^\top & B_1 & & & & & \\ & & I_{n_x} & D_2^\top & -I_{n_x} & & & \\ & & & A_2^\top & B_2 & \ddots & & \\ & & & & I_{n_x} & \ddots & & \\ & & & & & \ddots & D_{N-1}^\top & -I_{n_x} \\ & & & & & & A_{N-1}^\top & B_{N-1} \\ & & & & & & & I_{n_x} \end{bmatrix} \begin{bmatrix} -\lambda_0 \\ -\Lambda_0 \\ -\lambda_1 \\ -\Lambda_1 \\ \vdots \\ -\lambda_{N-1} \\ -\Lambda_{N-1} \\ -\lambda_N \end{bmatrix}$$

$$= - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \partial_x \mathcal{C}(x_N) \end{bmatrix}. \tag{A.24}$$

We determine the exact identity of λ by backward substitution:

1. From the last row of (A.24), we immediately obtain

$$\lambda_N = \partial_x \mathcal{C}(x_N). \tag{A.25}$$

2. Next, we prove equation (2.23a). For each $n = 0, \ldots, N-1$, we obtain

$$\begin{aligned} 0 &= \begin{bmatrix} I_{n_x} & D_n^\top & -I_{n_x} \end{bmatrix} \begin{bmatrix} \lambda_n \\ \Lambda_n \\ \lambda_{n+1} \end{bmatrix} = \lambda_n + D_n^\top \Lambda_n - \lambda_{n+1} \\ &= \lambda_n - (d_x F_n(X_n, p)(\mathbb{1}_s \otimes I_{n_x}))^\top \begin{bmatrix} h_n b_1 \Lambda_{n,1} \\ \vdots \\ h_n b_s \Lambda_{n,s} \end{bmatrix} \\ &\quad - \lambda_{n+1} \\ &= \lambda_n - (\mathbb{1}_s^\top \otimes I_{n_x}) d_x F_n(X_n, p)^\top \begin{bmatrix} h_n b_1 \Lambda_{n,1} \\ \vdots \\ h_n b_s \Lambda_{n,s} \end{bmatrix} \\ &\quad - \lambda_{n+1} \\ &= \lambda_n - h_n \sum_{i=1}^{s} b_i d_x f(X_{n,i}, p, t_n + c_i h_n)^\top \Lambda_{n,i} \\ &\quad - \lambda_{n+1}. \end{aligned}$$

$$\lambda_{n+1} = \lambda_n + h_n \sum_{i=1}^{s} b_i \ell_{n,i}, \tag{A.26}$$

with $\ell_{n,i} = -d_x f(X_{n,i}, p, t_n + c_i h_n)^\top \Lambda_{n,i}$.

3. Equation (2.23c) follows by

$$\begin{aligned}
0 &= \begin{bmatrix} 0 & A_n^\top & B_n \end{bmatrix} \begin{bmatrix} \lambda_n \\ \Lambda_n \\ \lambda_{n+1} \end{bmatrix} \\
&= A_n^\top \Lambda_n + B_n \lambda_{n+1} \\
&= \left(I_{sn_x} - h_n d_x F_n(X_n, p)(A \otimes I_{n_x})\right)^\top \Lambda_n \\
&\quad - h_n (b \otimes I_{n_x}) \lambda_{n+1} \\
&= \left(I_{sn_x} - h_n (A^\top \otimes I_{n_x}) d_x F_n(X_n, p)^\top\right) \Lambda_n \\
&\quad - h_n (b \otimes I_{n_x}) \lambda_{n+1}.
\end{aligned}$$

In the following, we consider the ith entry of the previous equation, i.e., $h_n b_i \Lambda_{n,i}$ of Λ_n with $i = 1, \ldots, s$.

$$\begin{aligned}
0 &= h_n b_i \Lambda_{n,i} - h_n^2 \sum_{j=1}^{s} a_{ji} b_j \partial_x f(X_{n,j}, p, t_n + c_j h_n)^\top \Lambda_{n,j} \\
&\quad - h_n b_i \lambda_{n+1} \\
\Lambda_{n,i} &= \lambda_{n+1} + h_n \sum_{j=1}^{s} \frac{a_{ji} b_j}{b_i} d_x f(X_{n,j}, p, t_n + c_j h_n)^\top \Lambda_{n,j} \\
&\overset{(A.26)}{=} \lambda_n + h_n \sum_{i=1}^{s} b_i \ell_{n,i} - h_n \sum_{j=1}^{s} \frac{a_{ji} b_j}{b_i} \ell_{n,j} \\
&= \lambda_n + h_n \sum_{j=1}^{s} \left(b_j - \frac{a_{ji} b_j}{b_i}\right) \ell_{n,j},
\end{aligned}$$

with $\ell_{n,j} = -d_x f(X_{n,j}, p, t_n + c_j h_n)^\top \Lambda_{n,j}$.
Finally, we show the formula of the gradient (2.22), which is given by (A.7)

$$\partial \Phi = \partial_p \mathcal{C} + d_p \phi^\top \lambda_0 \overset{\partial_p \mathcal{C} = 0}{=} d_p \phi^\top \lambda_0. \tag{A.27}$$

The Jacobian $d_p \phi^\top$ consists of the following building blocks: For the nth iteration step Ψ_n, the *local* Jacobian with respect to parameter p reads

$$\begin{aligned}
d_p \Psi_n &= \begin{bmatrix} d_p \Psi_{n,1} \\ d_p \Psi_{n,2} \end{bmatrix} = \begin{bmatrix} \bar{D}_n \\ 0 \end{bmatrix}, \\
\bar{D}_n &= -d_p F_n(X_n, p)(\mathbb{1}_s \otimes I_{n_p}).
\end{aligned} \tag{A.28}$$

By concatenating $N-1$ of these blocks (one for each iteration $n = 1, \ldots, N-1$) of (A.28), the overall Jacobian is given by

$$d_p \phi^\top = \begin{bmatrix} 0 & \bar{D}_0^\top & 0 & \bar{D}_1^\top & 0 \ldots 0 & \bar{D}_{N-1}^\top & 0 \end{bmatrix}. \tag{A.29}$$

Now, formula (2.22) is explicitly given by

$$\begin{aligned}
\partial \Phi &= d_p \phi^\top \lambda_0 \\
&= \begin{bmatrix} 0 & \bar{D}_0^\top & 0 \ldots 0 & \bar{D}_{N-1}^\top & 0 \end{bmatrix} \begin{bmatrix} -\lambda_0 \\ -\Lambda_0 \\ -\lambda_1 \\ \vdots \\ -\lambda_{N-1} \\ -\Lambda_{N-1} \\ -\lambda_N \end{bmatrix} \\
&= -\sum_{n=0}^{N-1} \bar{D}_n^\top \Lambda_n \\
&= \sum_{n=0}^{N-1} (d_p F_n(X_n, p)(\mathbb{1}_s \otimes I_{n_p}))^\top \Lambda_n \\
&= \sum_{n=0}^{N-1} ((\mathbb{1}_s \otimes I_{n_p})) d_p F_n(X_n, p)^\top \Lambda_n \\
&\overset{(A.13)}{=} \sum_{n=0}^{N-1} h_n \sum_{i=1}^{s} b_i d_p f(X_{n,i}, p, t_n + c_i h_n)^\top \Lambda_{n,i}.
\end{aligned}$$

□

References

1. Amari, S.I., Nagaoka, H.: Methods of Information Geometry. Oxford University Press, New York (2000)
2. Åström, F., Petra, S., Schmitzer, B., Schnörr, C.: Image labeling by assignment. J. Math. Imaging Vis. **58**(2), 211–238 (2017)
3. Ay, N., Jost, J., Lê, H.V., Schwachhöfer, L.: Information Geometry. Springer, Berlin (2017)
4. Cao, Y., Li, S., Petzold, L., Serban, R.: Adjoint sensitivity analysis for differential-algebraic equations: the adjoint DAE system and its numerical solution. SIAM J. Sci. Comput. **24**(3), 1076–1089 (2003)
5. Weinan, E.: A proposal on machine learning via dynamical systems. Commun. Math. Stat. **5**(1), 1–11 (2017)
6. Haber, E., Ruthotto, L.: Stable architectures for deep neural networks. Inverse Probab. **34**(1), 014004 (2017)
7. Hager, W.W.: Runge–Kutta methods in optimal control and the transformed adjoint system. Numer. Math. **87**(2), 247–282 (2000)
8. Hairer, E., Lubich, C., Wanner, G.: Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations, 2nd edn. Springer, Berlin (2006)
9. Hairer, E., Nørsett, S., Wanner, G.: Solving Ordinary Differential Equations I: Nonstiff Problems, 2nd edn. Springer, Berlin (1993)
10. Hairer, E., Nørsett, S., Wanner, G.: Solving Ordinary Differential Equations I, 3rd edn. Springer, Berlin (2008)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of CVPR (2016)
12. Hühnerbein, R., Savarino, F., Åström, F., Schnörr, C.: Image labeling based on graphical models using Wasserstein messages and geometric assignment. SIAM J. Imaging Sci. **11**(2), 1317–1362 (2018)

13. Hühnerbein, R., Savarino, F., Petra, S., Schnörr, C.: Learning adaptive regularization for image labeling using geometric assignment. In: Proceedings of SSVM. Springer (2019)
14. Jost, J.: Riemannian Geometry and Geometric Analysis, 7th edn. Springer, Berlin (2017)
15. Kappes, J., Andres, B., Hamprecht, F., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., Rother, C.: A comparative study of modern inference techniques for structured discrete energy minimization problems. Int. J. Comput. Vis. **115**(2), 155–184 (2015)
16. Lee, J.M.: Introduction to Smooth Manifolds. Springer, Berlin (2013)
17. Phillips, J.: Coresets and sketches. In: Goodman, J.E. (ed.) Handbook of Discrete and Computational Geometry. CRC Press, Boca Raton (2016)
18. Ross, I.M.: A roadmap for optimal control: the right way to commute. Ann. N. Y. Acad. Sci. **1065**(1), 210–231 (2006)
19. Sanz-Serna, J.: Symplectic Runge–Kutta schemes for adjoint equations, automatic differentiation, optimal control, and more. SIAM Rev. **58**(1), 3–33 (2016)
20. 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)
21. Tomović, R., Vukobratovíc, M.: General Sensitivity Theory. Modern Analytic and Computational Methods in Science and Mathematics. Elsevier, Amsterdam (1972)
22. Wainwright, M.J.: Estimating the "wrong" graphical model: benefits in the computation-limited setting. J. Mach. Learn. Res. **7**, 1829–1859 (2006)
23. Wainwright, M.J., Jaakola, T.S., Willsky, A.S.: MAP estimation via agreement on trees: message-passing and linear programming. IEEE Trans. Inf. Theory **51**(11), 3697–3717 (2005)
24. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Trans. Inf. Theory **51**(7), 2282–2312 (2005)
25. Zeilmann, A., Savarino, F., Petra, S., Schnörr, C.: Geometric numerical integration of the assignment flow. Inverse Probl. **36**(3), 034004 (2020)
26. Zern, A., Zeilmann, A., Schnörr, C.: Assignment flows for data labeling on graphs: convergence and stability. CoRR arXiv:2002.11571 (2020)
27. Zern, A., Zisler, M., Petra, S., Schnörr, C.: Unsupervised assignment flow: label learning on feature manifolds by spatially regularized geometric assignment. J. Math. Imaging Vis. (2020). https://doi.org/10.1007/s10851-019-00935-7d
28. Zhu, S.C., Liu, X.: Learning in Gibbsian fields: how accurate and how fast can it be? IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 1001–1006 (2002)
29. Zisler, M., Zern, A., Petra, S., Schnörr, C.: Self-assignment flows for unsupervised data labeling on graphs. SIAM J. Imaging Sci. (2020) **(in Press)**

Ruben Hühnerbein received his M.Sc. degree in Mathematics (2016), and his Ph.D. degree in Applied Mathematics (2020) from Heidelberg University, Germany. He is currently a postdoctoral researcher in the Image and Pattern Analysis group of Christoph Schnörr. His research interests include mathematical models, numerical optimization, machine learning and their applications in image analysis.

Fabrizio Savarino received his M.Sc. degree in Mathematics (2016) and his Ph.D. degree in Applied Mathematics (2020) from Heidelberg University, Germany. Currently, he is a postdoctoral research fellow of the excellence cluster 'Structures' at the Heidelberg University. His research interests include variational methods, optimization and differential geometric methods in machine learning and their application to image analysis.

Stefania Petra received her B.Sc. degree in Mathematics and Computer Science in 2001 and her M.Sc. in Mathematics in 2003 from the Babe s-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 Wrangel-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 the Heidelberg University where she is leading the Mathematical Imaging Group at the Institute of Applied Mathematics. Her research interests include compressed sensing, mathematical models of image analysis with an emphasis on tomography and numerical optimization.

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. Together with colleagues, he has set up and is co-directing the Heidelberg Collaboratory for Image Processing funded by industrial partners. He also serves on the steering board of the cluster of excellence STRUCTURES at the Heidelberg University. His research focuses on mathematical models of image analysis and corresponding aspects of numerical algorithm design and optimization.