Computer Aided Geometric Design 79 (2020) 101864

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Laplacian-optimized diffusion for semi-supervised learning

Max Budninskiy [a,b], Ameera Abdelaziz [a], Yiying Tong [c], Mathieu Desbrun [d,a,*]

[a] Caltech, Pasadena, CA, USA
[b] true[X], Los Angeles, CA, USA
[c] Michigan State University, East Lansing, MI, USA
[d] ShanghaiTech University, Shanghai, China

ARTICLE INFO

Article history:
Available online 20 April 2020

Keywords:
Semi-supervised learning
Graph Laplacian
Biconvex optimization
High-dimensional geometry

ABSTRACT

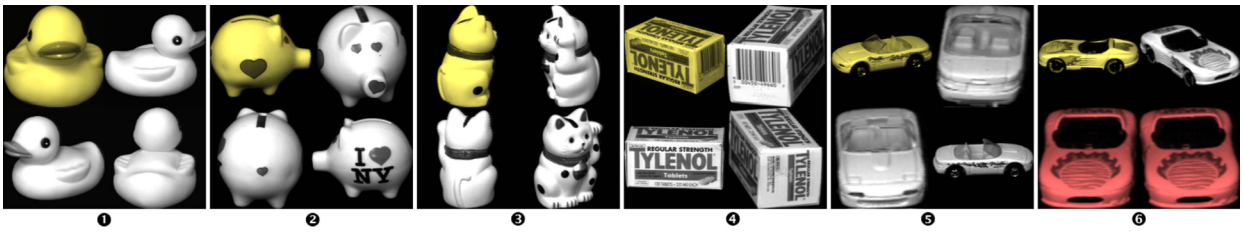
Semi-supervised learning (SSL) is fundamentally a geometric task: in order to classify high-dimensional point sets when only a small fraction of data points are labeled, the geometry of the unlabeled data points is exploited to gain better classifying accuracy. A number of state-of-the-art SSL techniques rely on label propagation through graph-based diffusion, with edge weights that are evaluated either analytically from the data or through compute-intensive training based on nonlinear and nonconvex optimization. In this paper, we bring discrete differential geometry to bear on this problem by introducing a graph-based SSL approach where label diffusion uses a Laplacian operator learned from the geometry of the input data. From a data-dependent graph of the input, we formulate a biconvex loss function in terms of graph edge weights and inferred labels. Its minimization is achieved through alternating rounds of optimization of the Laplacian and diffusion-based inference of labels. The resulting optimized Laplacian diffusion directionally adapts to the intrinsic geometric structure of the data which often concentrates in clusters or around low-dimensional manifolds within the high-dimensional representation space. We show on a range of classical datasets that our variational classification is more accurate than current graph-based SSL techniques. The algorithmic simplicity and efficiency of our discrete differential geometric approach (limited to basic linear algebra operations) also make it attractive, despite the seemingly complex task of optimizing all the edge weights of a graph.

1. Introduction

Discrete differential geometry (DDG) has been very successful in expressing geometric concepts using complementary computational and mathematical points of view, for example using Discrete Exterior Calculus—see, for instance, Crane et al. (2013). This dual perspective has led to the development of countless practical algorithms for real-world 3D geometric data, ranging from curvature approximation, shape smoothing, surface parameterization, vector field design, to the computation of geodesic distances. While the geometric concepts at the core of all these developments are actually valid in arbitrary dimensions, the vast majority of DDG developments focused on surfaces embedded in 3D, or on 3D objects directly, with very few exceptions such as Budninskiy et al. (2017, 2019). Yet, geometry is relevant in a number of non-graphics areas.

* Corresponding author at: Caltech, Pasadena, CA, USA.
E-mail addresses: mbudnin@caltech.edu (M. Budninskiy), aabdelaz@caltech.edu (A. Abdelaziz), ytong@msu.edu (Y. Tong), mathieu@caltech.edu (M. Desbrun).

https://doi.org/10.1016/j.cagd.2020.101864
0167-8396/



Error	LNP		HGF		LLGC		AEW-HGF		AEW-LLGC		Ours	
%Labels	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
1%	0.165	0.017	0.201	0.017	0.186	0.016	0.185	0.018	0.150	0.015	**0.0039**	0.0105
5%	0.113	0.009	0.148	0.012	0.145	0.013	0.116	0.014	0.099	0.013	**0.0021**	0.0016
10%	0.094	0.009	0.116	0.007	0.127	0.008	0.074	0.011	0.079	0.017	**0.0013**	0.0007
20%	0.080	0.010	0.089	0.006	0.122	0.008	0.040	0.009	0.073	0.019	**0.0010**	0.0010
40%	0.066	0.011	0.051	0.006	0.124	0.007	0.012	0.005	0.063	0.016	**0.0004**	0.0006

Fig. 1. Laplacian-optimized Semi-supervised Classification. Exploiting the *geometry of datasets* can help with image recognition: from the COIL-20 library of 1400 greyscale images spanning 20 classes (6 examples of class shown above) and knowing only in which class 1% of the images belong to (example of labeled images shown in yellow), our method learns a *Laplacian operator* over these images to infer the class of each unlabeled image. Our Laplacian-optimized approach produces a smaller error (proportion of mislabeled data points, computed as a mean and standard deviation over 20 runs with randomly-chosen labels) than existing semi-supervised learning techniques, even for a higher amount of labeled images (see table). For 10% of labels or more, only two images get sometimes misclassified (show in red) as class 5 instead of class 6. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Over the last ten years, the advancements of machine learning have had a significant impact in science: learning algorithms such as convolutional neural networks, generative adversarial networks, decision trees, or autoencoders offer opportunities to construct *data-driven* approaches to scientific problems instead of the traditional *model-based* approaches, improving efficiency and/or predictability in the process. As machine learning evolves, the importance of geometry and topology of data is increasingly acknowledged, see Sindhwani et al. (2010). For instance, a neural network classifier can be understood (or even visualized as in Playground (2018)) as applying a deformation of the input data to make it more easily linearly-separable: each layer nonlinearly deforms the embedding space where the data lies through an affine transform based on the weights of the network followed by a pointwise application of a monotone activation function as described in Olah (2014). Similarly, key in inferential tasks is the "manifold hypothesis", see Zhu and Goldberg (2009), stating that data tend not to be uniformly distributed in their embedding space, but instead, lie approximately on manifolds of much lower dimensions than the embedding space. Despite the clear geometric nature of many learning approaches, limited efforts have been dedicated to extend discrete differential geometry techniques to high-dimensional spaces in order to develop novel, more powerful learning tools.

In this paper, we bring DDG to bear on SSL, and introduce a geometric approach to graph-based semi-supervised learning. By optimizing the edge weights of a sparse graph to customize a Laplacian operator adapted to the data, we show that labeling via the resulting anisotropic diffusion outperforms current approaches.

1.1. Semi-supervised learning

Semi-supervised learning (SSL, Chapelle et al. (2010)) consists in automatically classifying data (i.e., to give each data point a "label"—class, category, type, etc.) based on a few known labeled data points. Labeling data is often a time consuming task for humans, requiring the efforts of expert annotators or expensive computational probing. Unlabeled samples, on the other hand, are typically easy to collect, and often offer hints on the actual underlying spatial distribution of the data. Semi-supervised learning tackles classification (i.e., inference of labels) by considering both unlabeled and labeled data: making use of this combined information often surpasses the classification performance that would have been obtained either by discarding the unlabeled data and doing fully supervised learning, or by discarding labels and doing fully unsupervised learning.

While deep learning methods can achieve high accuracy in classification problems by training neural networks on large datasets, the typical setup of SSL renders deep networks inappropriate in many contexts: there are often too few labeled data and/or too few unlabeled data to directly infer classification with decent accuracy. Note that recent results demonstrate that deep networks can be successfully applied for SSL when large datasets are available; for instance, Rasmus et al. (2015) used a combination of supervised neural network classification with unsupervised nested denoising autoencoders to perform SSL; deep self-training (where more labels are gradually inferred and added) has also been proven feasible in specific areas such as natural language processing or face recognition, see Vesely et al. (2013); multi-layer graph convolutional networks (see Kipf and Welling (2017)) have also shown promise recently. But when datasets of only moderate size need to be classified, popular semi-supervised learning approaches have involved, instead, mixture models, co-training and multiview learning, as well as semi-supervised support vector machines as in Zhu (2007). The success of these methods

depends critically on the validity of the underlying assumptions they make on the input data. In this paper, we focus on a particular class of SSL methods, so called *graph-based SSL*, that has consistently outperformed other methods and that offers *interpretability*, a feature lacking in deep learning.

1.2. Graph-based approaches to SSL

Foundations. A commonly-used assumption for input data stored in a meaningful embedding space is that two data points are likely to share the same label if they are "close" to each other, or if they lie on a same low-dimensional sub-manifold (i.e., if points of a same class form low dimensional subspaces locally, which can be seen as tangent spaces of a low-dimensional manifold in the high-dimensional embedding space); see, e.g., Zhu and Goldberg (2009), Chapelle et al. (2010). These simple geometric assumptions, that we will respectively refer to as the *cluster assumption* and the *manifold assumption* as various authors have done in the past (see Sindhwani et al. (2010)), are at the root of a large family of "graph-based methods", including Zhu et al. (2003); Zhou et al. (2003); Wang et al. (2006); Karasuyama and Mamitsuka (2017), which are considered state-of-the-art methods in graph-based SSL.
Methodology. Graph-based methods begin with the creation of a sparse graph whose vertices are input (labeled and unlabeled) data points. Typically, two data points are connected with an edge if either the distance between them is less than some threshold in the embedding space or if one of them is in the set of the K nearest neighbors of the other. This graph thus connects nearby points, and offers a discrete representation of the "geometry" of the data, which includes any sub-manifold around which the data concentrate. Each edge of this graph is then assigned a weight based on either its length in the embedding space as in Belkin and Niyogi (2004) (sometimes using a data-dependent rescaling of the coordinates, see Karasuyama and Mamitsuka (2017)), or on the distribution of the nearest data points around as in Wang et al. (2006). Equipped with this weighted graph, classification is then performed typically through propagation of the known labels to the rest of the graph, using diffusion based on the edge weights or through some form of entropy minimization.
Strengths and limitations. One advantage of semi-supervised learning compared to deep learning is that the graph is entirely defined by the data: it therefore dramatically reduces the number of hyperparameters like number of layers, neuron count, connectivity type, etc. While SSL involves a weighted graph over which classification is performed, an obvious limitation is that the edge weights are rarely globally trained (i.e., optimized): they are usually chosen to be Gaussian weights à la Belkin and Niyogi (2004) or Locally Linear Embedding (LLE) weights à la Roweis and Saul (2000). Since these "pre-canned" weights strongly influence the behavior of the classifier as they indicate the similarity between data points, graph-based approaches to SSL also depend critically on how well the input data fits the underlying geometric assumptions in the actual embedding (feature) space. The few methods that try to partially optimize weights based on the input data do rely on nonlinear, nonconvex minimization, which hurts their scalability in practice. Note that many methods have built over this graph-based diffusion approach, introducing various additional terms such as iterated Laplacians (Zhou and Belkin (2011)), Tikhonov (as in Belkin et al. (2006)) or ℓ_1 (as in Rustamov and Klosowski (2018)) regularization, and diffusion-inspired convolutional filters, see Kipf and Welling (2017).

1.3. Contributions

In this paper, we approach the design of a new Laplacian-optimized diffusion for semi-supervised learning using discrete differential geometry. To exploit the intrinsic structure of the data which is often spatially distributed in clusters and/or on low-dimensional manifolds within the high-dimensional representation space, we propose a biconvex minimization where the edge weights of a data-dependent graph are optimized in alternation with an inference of labels over the unlabeled data points. This edge weight optimization constructs a Laplacian operator whose quadratic form represents a smoothness energy in a metric adapted to the intrinsic distribution of the input in the embedding space. The resulting Laplacian-optimized graph diffusion of known labels exploits the geometry of the input, classifying unlabeled data more accurately than current SSL techniques. The algorithmic simplicity and efficiency of our approach also make it attractive, despite the seemingly complex task of optimizing all the graph weights.

2. Review of graph-based SSL methods

Before introducing our approach, we first delve into the existing graph-based methods to gather some of the key concepts we will leverage in our work, but also to point out how usual differential geometry notions (from Laplacian and Dirichlet energy to optimal transport) play a central role throughout this literature.

2.1. Notations

Semi-supervised learning considers an input dataset $\mathscr{D} = \{\mathbf{x}_i\}_{i=1..n}$ of n points distributed in the D-dimensional Euclidean space $\mathbb{R}^D$. The first ℓ input points are labeled, so that the input set can be written as $\mathscr{D} = \mathscr{D}_L \cup \mathscr{D}_U$, where $\mathscr{D}_L = \{\mathbf{x}_i\}_{i=1...\ell}$ are the labeled points and $\mathscr{D}_U = \{\mathbf{x}_i\}_{i=\ell+1...n}$ the unlabeled points, with $\ell \ll n$. Assuming there are C categories of labels (defined to be the integers $\{1, 2, ..., C\}$ for simplicity), we store label information in an $n \times C$ matrix $\mathbf{P}$, such that

$$\mathbf{P}_{ik} = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \mathscr{D}_L \text{ and label}(\mathbf{x}_i) = k, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the Euclidean norm of vectors will be denoted by $\|.\|$, while the Frobenius norm of a matrix will be represented as $\|.\|_F$, with $\|\mathbf{M}\|_F^2 \equiv \text{Tr}(\mathbf{M}\mathbf{M}^t)$. We will also denote by diag(.) the operator that turns a vector $\mathbf{v}$ of size k into a diagonal $k \times k$ matrix diag($\mathbf{v}$) such that $[\text{diag}(\mathbf{v})]_{ii} = [\mathbf{v}]_i$.

All graph-based methods proceed quite similarly. They begin by defining a connected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ with the nodes $\mathscr{V}$ corresponding to the n input data points, and where an edge $\mathbf{e}_{ij} \in \mathscr{E}$ connecting points $\mathbf{x}_i$ and $\mathbf{x}_j$ is created iff $\mathbf{x}_j$ is among the K nearest neighbors of $\mathbf{x}_i$ or vice-versa. With this data representation, they proceed to propagate known labels through the graph. This label propagation is achieved through various means designed to exploit the geometric assumptions of the data, as we review next.

2.2. Harmonic Gaussian fields

Harmonic Gaussian fields (HGF, Zhu et al. (2003)), one of the early approaches in graph-based SSL, proposed to use a discrete Laplacian operator over the graph to induce diffusion of the labels. For every graph edge $\mathbf{e}_{ij} \in \mathscr{E}$, they assign a positive Gaussian weight $\mathbf{w}_{ij}$ of the form:

$$\mathbf{w}_{ij} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / \sigma^2\right), \tag{1}$$

where σ is a length scale hyperparameter, typically picked based on the average distance between data points connected by an edge. Since these weights are known to give rise to a discrete approximation of the Laplace operator, see Belkin and Niyogi (2004), the inherent smoothness of the data can be leveraged by forming a harmonic interpolation of the labels based on these weight expressions. More precisely, let's define the matrix $\mathbf{Q}$ of size $n \times C$, where $\mathbf{Q}_i$ denotes its i-th row with C values, representing the **likelihood scores** for node $\mathbf{x}_i$ to be in each of the C classes. The HGF approach computes matrix $\mathbf{Q}$ as the minimizer of the following discrete Dirichlet energy:

$$\arg\min_{\mathbf{Q}} \sum_{\mathbf{e}_{ij} \in \mathscr{E}} \mathbf{w}_{ij} \|\mathbf{Q}_i - \mathbf{Q}_j\|^2 \quad \text{s.t.} \mathbf{Q}_i = \mathbf{P}_i \ \forall \mathbf{x}_i \in \mathscr{D}_L. \tag{2}$$

This minimization calculates C harmonic functions (one per class, stored in columns of $\mathbf{Q}$) subject to label constraints; the inferred label of a given unlabeled node is then defined as the index of the largest element in $\mathbf{Q}$'s i-th row:

$$\text{label}(\mathbf{x}_i) = \arg\max_{k=1\ldots C} \quad \mathbf{Q}_{ik}. \tag{3}$$

This approach can be interpreted as a Gaussian regression, since it corresponds to the conditional expectation of label assignment assuming that the latter is a realization of a Gaussian process with a covariance function given by the Laplace operator; it also shares obvious connections to the graph mincut approach of Blum and Chawla (2001). Note that the optimal $\mathbf{Q}$ is easily solved for via a sparse linear system, and by definition of Eq. (2), this harmonic label propagation is interpolatory, i.e., known input labels are kept unaltered. Prior knowledge of class sizes can also be accommodated through post-processing using class mass normalization as in de Sousa (2015).

2.3. Learning with local and global consistency

Learning with Local and Global Consistency method (LLGC, Zhou et al. (2003)) relies on the same Gaussian edge weights, but modifies the variational problem for $\mathbf{Q}$ to mimic spreading activation networks: first, they replace the interpolation of $\mathbf{P}$ with a fitting penalty, which only *favors* the initial labeling (in lieu of enforcing it exactly) to result in a globally smoother assignment; second, they propose to rescale the value of $\mathbf{Q}$ at a node $\mathbf{x}_i$ in the Dirichlet energy by dividing by the sum of the edge weights emanating from that node in order to compensate for unavoidable density variation in real datasets. Classification is therefore achieved by solving:

$$\arg\min_{\mathbf{Q}} \ \|\mathbf{Q} - \mathbf{P}\|_F^2 + \eta \sum_{\mathbf{e}_{ij} \in \mathscr{E}} \mathbf{w}_{ij} \| \frac{\mathbf{Q}_i}{\sqrt{\sum_k \mathbf{w}_{ik}}} - \frac{\mathbf{Q}_j}{\sqrt{\sum_k \mathbf{w}_{jk}}} \|^2. \tag{4}$$

Note that this normalization of the Dirichlet energy corresponds precisely to what is called the "normalized Laplacian" quadratic form ($\mathbf{L}^{\text{norm}}$) in Normalized Cuts of Shi and Malik (2000), a well-known approach to spectral clustering. The trade-off between the two competing energy terms in Eq. (4) is captured by a positive hyperparameter η. Finally, minimizing the resulting energy turns out to be an implicit integration step of diffusion as in Desbrun et al. (1999); indeed, differentiating the objective function shows that optimization of Eq. (4) amounts to solving:

$$\left(\mathbf{Id} + \eta \mathbf{L}^{\text{norm}}\right) \mathbf{Q} = \mathbf{P}. \tag{5}$$

Unlike HGF, LLGC does *not* guarantee label preservation, and mislabeling happens in practice for most choices of size η. Still, the normalization of the Dirichlet energy allows for better results overall compared to previous methods.

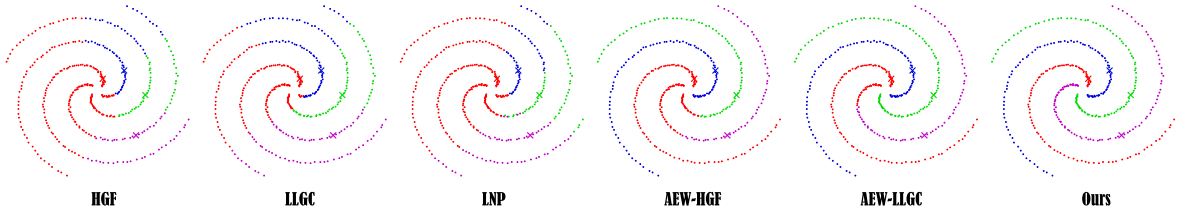


Fig. 2. Swirls. For a 2D dataset with points on 4 spiraling arms with 100 points and one label (marked as a colored cross) per spiral, previous SSL approaches fail to capture the obvious manifold structure of the data (each point is colored based on the inferred labeling). The three leftmost results (HGF from Zhu et al. (2003), LLGC from Zhou et al. (2003)) and LNP from Wang et al. (2006)) rely on local choices of edge weights, with the leftmost two overemphasizing label proximity and the third one focusing on local linearity instead. While the AEW approach from Karasuyama and Mamitsuka (2017) optimizes the weights globally (before using label propagation through HGF or LLGC), it properly discovers only two spirals at best. Our approach distinguishes the four spirals even with only one label per class.

2.4. Linear neighborhood propagation

The Linear Neighborhood Propagation (LNP, Wang and Zhang (2008)) departs from previous methods by rejecting the use of Gaussian weights since the Laplacian is certainly not the only operator to capture smoothness of label assignments. Instead, they take a cue from work on dimensionality reduction (namely, Locally Linear Embedding (LLE), Roweis and Saul (2000)), and propose to locally optimize edge weights to exploit the local linearity assumption. For a given node $\mathbf{x}_i$, weights of emanating edges $\mathbf{e}_{ij}$ are found through constrained quadratic optimization:

$$\underset{\{\mathbf{w}_{ij}\}_{j\in N(i)}}{\arg\min} \|\mathbf{x}_i - \sum_{j\in N(i)} \mathbf{w}_{ij}\mathbf{x}_j\|^2 \ \text{s.t.}\ \mathbf{w}_{ij} \geq 0 \ \text{and} \sum_{j\in N(i)} \mathbf{w}_{ij} = 1, \tag{6}$$

where $\mathcal{N}(i)$ represents the indices of the vertices connected to $\mathbf{x}_i$ in the graph $\mathcal{G}$. Since this optimization is performed independently for each node, note that $\mathbf{w}_{ij} \neq \mathbf{w}_{ji}$, and we should consider the graph as directed in this case. The interpretation of this optimization, also adopted in de Sousa (2015), is clear: a weight $\mathbf{w}_{ij}$ measures (between 0 and 1) how similar $\mathbf{x}_j$ is to $\mathbf{x}_i$, independent of the actual distances unlike what is used for Gaussian weights. It is thus more likely to build strong edges between quite separate manifolds, whereas Gaussian weights would just assume they are only weakly linked simply based on large distances.

With these (directed) edge weights, the labels are inferred like in LLGC, by solving for the matrix $\mathbf{Q}$ that minimizes:

$$\|\mathbf{Q} - \mathbf{P}\|_F^2 + \eta \sum_{i,j} \mathbf{w}_{ij} \left|\left|\mathbf{Q}_i - \mathbf{Q}_j\right|\right|^2. \tag{7}$$

The resulting implicit diffusion thus uses a locally-optimized "composite" Laplacian, and this labeling was shown to improve accuracy sometimes.

2.5. Measure-based classifiers

Previous classifiers are, in essence, using C different scalar functions, the largest of which indicates the predicted label for a given node. Subramanya et al. Subramanya and Bilmes (2011) noted that these C values can also be seen as providing a histogram (or probability distribution) for each data point, measuring the probability for this point to be of class c. As a consequence, they proposed to consider the smoothness of an assignment of probability distributions over the graph (instead of the smoothness of the numerical labels) to further improve accuracy. They formulated a loss function based on Kullback-Leibler (KL) divergence[1] which, after several tweaks, results in a convex programming problem. This problem was then solved via an alternating minimization (AM) approach, akin to the traditional Expectation-Maximization of Dempster et al. (1977) method, with closed-form updates. Improvements in classification accuracy were demonstrated from this measure-based approach. Further noting that the KL divergence cannot capture interactions between histogram bins, Solomon et al. (2014) suggested using optimal transport instead, and proposed a generalized Dirichlet energy over distribution-valued nodes of a graph to induce information propagation. However, they neither discussed computational complexity nor demonstrated improvement in classification accuracy.

2.6. Learning with weight optimization

Finally, the best results to date in graph-based SSL involve optimizing the edge weights *globally*, to best adapt them to the data. In fact, the idea dates back to HGF, where the authors noted that the Gaussian weights from Eq. (1) can be extended to read:

[1] KL divergence is also called "relative entropy" or "information gain', depending on the scientific field where it is invoked.

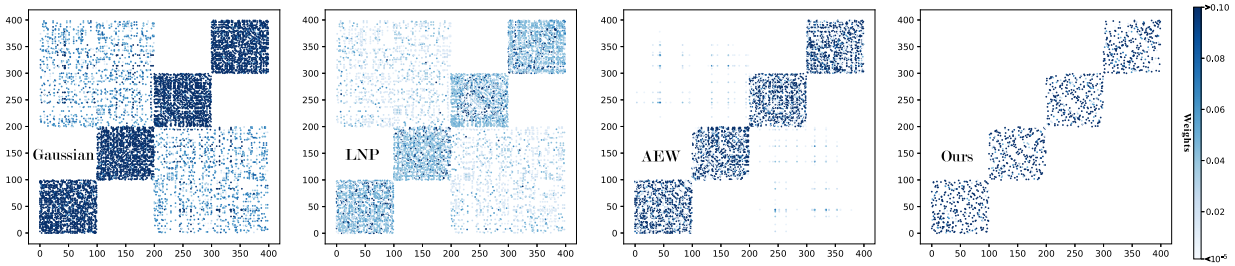


Fig. 3. Weighted graph. We visualize the different weighted graphs of the Swirls example in Fig. 2 as 2D matrices where element (i, j) is $\mathbf{w}_{ij}$ (for LNP, $(\mathbf{w}_{ij}+\mathbf{w}_{ji})/2$). Points are arranged by class (first 100 for class 1, ..., last 100 for class 4), but shuffled within each class. Since weights measure similarity between vertices of a common edge, one should expect a block diagonal structure, with weights corresponding to interclass edges near zero. Weights below 10^{-5} are not shown, and values above 0.1 are all displayed in a dark blue color.

$$\mathbf{w}_{ij}^{\sigma} = \exp\left(\sum_{d=1}^{D} \frac{(x_{i,d} - x_{j,d})^2}{\sigma_d^2}\right) \tag{8}$$

where now the D scaling hyperparameters $\sigma = (\sigma_1, \ldots \sigma_D)$ can be optimized based on the data to "learn" the relevant dimensions of the dataset. Unfortunately, Zhu et al. (2003) proposed to minimize label entropy to learn these hyperparameters, which led to very limited improvements: first, their optimization required regularization to prevent the hyperparameters from going to 0, otherwise it would end up selecting non-zero weights only between labels and their spatially closest unlabeled nodes; second, their entropy-based functional is highly nonlinear and nonconvex, hampering its efficient minimization.

Recently, Karasuyama and Mamitsuka Karasuyama and Mamitsuka (2017) revisited this idea by optimizing parameters $\sigma = (\sigma_1, \ldots \sigma_D)$ so as to minimize an LLE-type energy instead: they compute σ through

$$\underset{\{\sigma_1,\ldots,\sigma_D\}}{\arg\min} \sum_i \|\mathbf{x}_i - \frac{1}{\sum_k \mathbf{w}_{ik}^{\sigma}} \sum_{j\in N(i)} \mathbf{w}_{ij}^{\sigma}\, \mathbf{x}_j\|^2.$$

That is, they mixed the LLE weights Roweis and Saul (2000) advocated in LNP with the weight normalization proposed in Normalized Cuts Shi and Malik (2000), and used the above "normalized LLE" energy to find the hyperparameters $\{\sigma_d\}_{d=1..D}$ that make the weighted graph best fit the input data manifolds (i.e., for which every point can be best represented as a linear combination of its neighbors). The authors dubbed their approach Adaptive Edge Weighting (AEW). Unfortunately, the energy they use to optimize the weights is not convex. Minimizing it requires some form of gradient descent, and each gradient evaluation can be quite costly for high-dimensional input spaces since it is of complexity $\mathcal{O}(nKD^2)$. Yet, diffusing using HGF or LLGC the known labels through the graph with AEW-optimized weights was shown to beat previous methods on many datasets.

2.7. Discussion

The best current diffusion-based SSL approaches rely on graphs with learned edge weights (and their corresponding Laplacian operators) to proceed with label propagation in a way that most effectively leverages oft-present geometric properties of large data sets. However, they are often hampered by highly nonlinear optimizations of the weights, preventing their scalability and reliability; moreover, the loss function used to learn the weights is totally independent of the subsequent diffusion or even of the labels that are known. Instead, we propose to construct a consistent optimization of edge weights which, with the corresponding anisotropic diffusion they define, allows to adapt to the intrinsic geometric structure of the data in order to capture the cluster and manifold assumptions. Our variational approach thus improves on previous state-of-the-art semi-supervised learning methods in a number of ways:

- Edge weights are globally optimized in order to give rise to a (possibly anisotropic) diffusion of labels that best adapts to the intrinsic geometry of the data;
- Our loss function is biconvex in edge weights and label likelihood scores, rendering its optimization via alternating minimization particularly efficient;
- The resulting label propagation is interpolatory (i.e., is guaranteed to keep the known labels intact) and not biased by large density variations in the input data.

More importantly, our approach outperforms previous methods on typical datasets used to evaluate labeling accuracy.

3. Our Laplacian-optimized classifier

In this section, we present our variational approach to design a classifier through optimization of the Laplace operator.

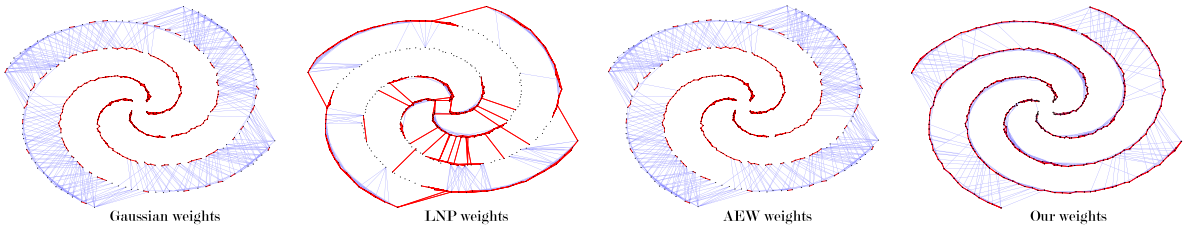


Fig. 4. Weights visualization. For the swirls dataset (see Fig. 2), we display the top 10% (in red) and the bottom 10% (in blue) of edge weights in terms of their magnitude, for the various types of Laplacian used in SSL. The graph was constructed with each node connected to its 20 nearest neighbors. Optimally, edges connecting nearby points of the same class should have larger weights, while edges connecting different classes should have vanishing weights; our approach fares much better than the other SSL techniques.

3.1. Notations

We will use the same notations as defined earlier in Sec. 2.1. We also assemble the coordinates of the input data points into a $n \times D$ matrix $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_n)$, and we denote by $\mathbf{S}$ the *label selection matrix*, that is, the $n \times n$ diagonal matrix such that $\mathbf{S}_{ii} = 1$ if $\mathbf{x}_i \in \mathscr{D}_L$ (i.e., if $i \leq \ell$), and $\mathbf{S}_{ii} = 0$ otherwise. Based on the data-dependent graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}\}$ of a set of n vertices $\mathscr{V}$, edges $\mathscr{E}$ and a (fixed, but arbitrary) orientation of its edges, we also assemble the "adjacency" matrix $\mathbf{d}$ of size $|\mathscr{E}| \times n$ containing ± 1 for the endpoints of a given edge with the sign based on edge orientation; that is, $\mathbf{d}_{ev} = 1$ if v refers to the index of the target vertex of the edge with index e, $\mathbf{d}_{ev} = -1$ if v refers to the index of the source vertex of the edge with index e, and $\mathbf{d}_{ev} = 0$ otherwise. This matrix is the classical *discrete exterior derivative on 0-forms* (see Desbrun et al. (2008)), i.e., a sparse matrix that is completely determined by the graph topology and the chosen edge orientations. With a slight notation abuse, we will also denote by $|\mathbf{d}|$ the same matrix but where the absolute value of each element is used instead; i.e., this matrix contains as many 1's on the row i as there are neighbors of the corresponding vertex $\mathbf{x}_i$ in $\mathscr{G}$. Finally, we will use the $n \times C$ matrix $\mathbf{Q}$ already mentioned in Sec. 2.2 to store C discrete functions on graph $\mathscr{G}$ that indicate the likelihood score for each node to be of a given class.

3.2. Parameterization of the Laplacian

A reasonable assumption for SSL is that the data in the embedding space is concentrated around a manifold or a set of manifolds. For example, scans of handwritten digits possess certainly less possible variations in style than the dimensionality of the ambient representation space which is the number of pixels the scans have. Exploiting the geometric structures of input datasets is thus crucial: in effect, it brings down the dimensionality of the problem at hand. Since the most successful approaches to graph-based SSL thus far have been based on some form of label diffusion to best leverage the structure of data, we adopt a similar approach here.

Failures of previous Laplacians. However, selecting a good Laplace operator to induce label propagation is a difficult task. As we reviewed in Sec. 2, the use of "pre-canned" weights on edges such as the Gaussian weights from Eq. (1) relies too much on pure distances in the embedding space to be useful, strongly biasing diffusion in high density regions; even variants taking local densities into account such as Zelnik-Manor and Perona (2004) to try to decrease this high dependence to irregular sampling suffer from this issue. Locally-optimized weights like in Eq. (6) fare sometimes better, but double the number of variables needed and totally ignore distances within kNN neighborhoods. Other data-dependent choices of weights such as AEW showed improved results by globally optimizing a more general parameterization of the weights, using the form given in Eq. (8). However, note that the number of hyperparameters (D) is potentially very different from the number of data points n: this means that one does not have much degrees of freedom to optimize if $D \ll n$ (they only rescale each axis individually); or too many degrees of freedom if $n \ll D$ (for example, images with D pixels) which could lead to overfitting. Moreover, their loss function is highly nonlinear and nonconvex in the edge weights, making them difficult to optimize. Finally, it is interesting to notice that none of the existing works have exploited the presence of labels to further improve weights.

DEC Laplacian. In this work, we parameterize the Laplacian by the values of its *positive edge weights* on the graph $\mathscr{G}$ using the notation adopted in Discrete Exterior Calculus (DEC, Desbrun et al. (2008)). If we denote the vector of all positive edge weights by $\mathbf{w}$ (given in the order of the set of edges $\mathscr{E}$) such that $\mathbf{w}_{ij}$ is the weight associated with edge e_{ij} between vertices i and j, the DEC Laplacian (see, for instance, Crane et al. (2013)) is expressed as:

$$\mathbf{L}(\mathbf{w}) = \mathbf{d}^t \operatorname{diag}(\mathbf{w})\, \mathbf{d}. \tag{9}$$

Note that this expression is strictly equivalent to the usual unnormalized graph Laplacian used in SSL: indeed, if we instead assemble the $|\mathscr{V}| \times |\mathscr{V}|$ matrix $\mathbb{W}$ from the vector of edge weights $\mathbf{w}$ through $\mathbb{W}_{ij} = \mathbf{w}_{ij}$ and call $\mathbb{D}$ the diagonal matrix such that $\mathbb{D}_{ii} = \sum_j W_{ij}$, one has: $\mathbf{L}(\mathbf{w}) = \mathbb{D} - \mathbb{W}$. The graph-based Laplacian is guaranteed to be symmetric positive semi-definite (with a kernel purely dependent on the topology of the weighted graph) if only non-negative weights are present, and it

does not depend on the choice of edge orientations. We adopt the DEC notation in this paper for one key reason: it emphasizes that the edge weights are the only metric dependence of the graph Laplacian, since the matrix $\mathbf{d}$ is purely topological. In fact, the diagonal matrix $\mathrm{diag}(\mathbf{w})$ can be interpreted as a *discrete "diagonal" Hodge star* $\star_1$ (see Bossavit (1998); Desbrun et al. (2008)) for differential 1-forms. Given an intrinsic metric m on a smooth manifold that the graph $\mathscr{G}$ represents, the edge weights thus approximate the continuous Hodge star of 1-forms induced by m Abraham et al. (1988). Therefore, the (positive) edge weights are the degrees of freedom to *explore all possible intrinsic metrics* in which to compute the Laplacian, instead of blindly sticking to the metric induced by the embedding space. This important geometric fact underlies our work.

3.3. Normalized Laplacian

Normalized Laplacians have been proposed in the past: to bring all the eigenvalues of the Laplacian between zero and one in mesh fairing, see Desbrun et al. (1999), to balance intercluster and intracluster similarity in spectral clustering, see Shi and Malik (2000), etc. In all these instances, the Laplace operator is altered by making the diagonal entries of its matrix representation *unit*. In our context of label propagation, existing (unnormalized) Laplacians have the issue that dense regions overpower sparse regions, as input labels in denser regions propagate much further than they should. Normalized Laplacians prevent that from happening by *equalizing* the local diffusivity on a per-vertex basis. To achieve this goal and render our Laplacian insensitive to large density variations, we propose to simply *constrain the sum of the edge weights at each vertex of* $\mathscr{G}$ *to be* 1, which has the effect of forcing all the diagonal entries of $\mathbf{L}$ to be 1. Notice that such linear constraints are much simpler to deal with numerically than relying on an "analytically-normalized" Laplacian with highly-nonlinear dependence on $\mathbf{w}$ as in Eq. (4) for instance. Since the operator $|\mathbf{d}|^t$ defined in Sec. 3.1 amounts to summing edge weights around each vertex, we can thus write that our choice of normalization verifies, for $\mathbf{1}$ denoting the vector of all ones,

$$|\mathbf{d}|^t \mathbf{w} = \mathbf{1}. \tag{10}$$

3.4. Variational classification principle

Based on our choice of Laplacian parameterization, we now define our classification energy $\mathbf{E}$, function of both the label likelihood scores matrix $\mathbf{Q}$ and the vector of all positive edge weights $\mathbf{w}$, as follows:

$$\mathbf{E}(\mathbf{Q},\mathbf{w}) = \underbrace{\mu \,\|\mathbf{S}\mathbf{Q}-\mathbf{P}\|_F^2 + \nu\,\mathrm{Tr}\big[\mathbf{Q}^t\mathbf{L}(\mathbf{w})\,\mathbf{Q}\big]}_{\textit{(diffusion)}} + \underbrace{\alpha\,\|\,\mathbf{L}(\mathbf{w})\,\mathbf{X}\,\|_F^2 + \gamma\,\mathbf{w}^t\overline{\mathbf{w}}}_{\textit{(data priors)}} + \underbrace{\beta\,\|1-|\mathbf{d}|^t\mathbf{w}\|^2}_{\textit{(normalization)}}. \tag{11}$$

Classification will be performed by *minimizing* this energy over all positive edge weights $\mathbf{w}\in\mathbb{R}^{|\mathscr{E}|}$ and matrices $\mathbf{Q}\in\mathbb{R}^{n\times C}$, that is, by solving:

$$\underset{\mathbf{Q},\mathbf{w}}{\arg\min}\ \mathbf{E}(\mathbf{Q},\mathbf{w}) \quad \text{s.t. } \mathbf{w}\succ\mathbf{0}, \tag{12}$$

from which all labels are derived using Eq. (3); i.e., the index of the largest value in the C components of row $\mathbf{Q}_i$ is the inferred label for $\mathbf{x}_i$. We postpone discussing how to perform this constrained minimization of $\mathbf{E}$ and how to select the various parameters involved $(\alpha,\beta,\gamma,\mu,\nu)$, focusing first on explaining each of the terms of Eq. (11) to understand their respective geometric meanings.

Diffusion. The *first term* simply forces the label likelihood scores $\mathbf{Q}_i$ for each *labeled* data point i to be equal to its proper given label $\mathbf{P}_i$; we will show that with appropriate parameter choice, it promotes an interpolatory label propagation. Notice our use of the label selection matrix $\mathbf{S}$ compared to Eq. (4), so as not to add any side effects for unlabeled data. The *second term* is the Dirichlet energy of $\mathbf{Q}$ based on the Laplacian $\mathbf{L}(\mathbf{w})$. Its presence enforces label coherency (i.e., smoothness) with respect to the metric defined by the current weights $\mathbf{w}$; note the actual diffusion may be strongly anisotropic if the geometry of data in the embedding space satisfies the manifold assumption such as in Fig. 2. The sum of these two terms is responsible for label diffusion: we will see that minimizing these two terms with respect to $\mathbf{Q}$ will lead to an interpolatory variant of the implicit diffusion given in Eq. (5).

Data priors. The *third term* of the energy $\mathbf{E}$ exploits the *local linearity assumption* of manifold learning: this biharmonic energy of the data (in the metric induced by the edge weights) promotes larger weights between nearby data points that are sitting on the same linear subspace. We thus let the metric vary to improve the square of the Dirichlet energy of the embedding of the data, thus capturing the smoothness present in the intrinsic geometry of the input. Note that using this biharmonic term is akin to the use of iterated Laplacians for SSL in Zhou and Belkin (2011) to prevent the typical spikes in point-constrained harmonic solutions. The *fourth term* encodes prior knowledge on weights through the use of a data-dependent, user-selected vector $\overline{\mathbf{w}}$ of edge values: a large value $\overline{\mathbf{w}}_{ij}$ for a particular edge $\mathbf{e}_{ij}$ forces the corresponding edge weight $\mathbf{w}_{ij}$ to be small. This term can thus encode the usual "cluster" assumption, i.e., nearby points are likely to have similar labels. This can be achieved, for instance, by setting $\overline{\mathbf{w}}_{ij}\sim||\mathbf{x}_i-\mathbf{x}_j||^p$ as we will discuss later. Together, these energy terms regularize weights and prevent overfitting.

Normalization. Finally, the *fifth term* serves as a penalty for deviating from a normalized Laplacian. It avoids having to enforce the linear constraints of Eq. (10) exactly, and maintains the simplicity of the energy since only quadratic terms are used. Note that enforcing exact normalization is not particularly vital: we only need to roughly equalize the various scales of the Laplacian, for which a penalty energy amply suffices.

3.5. Optimality conditions

Our particular choice of energy does not only exploit the traditional (cluster and manifold) assumptions of SSL: our energy also has, by design, the nice property of being biconvex, i.e., it is convex in $\mathbf{w}$ for a fixed matrix $\mathbf{Q}$, and in $\mathbf{Q}$ for a fixed set of weights $\mathbf{w}$. We now describe the conditions of optimality for these two convex energies.

Label optimality. For a fixed set of weights $\mathbf{w}$, the optimal label matrix $\mathbf{Q}^*$ is given as:

$$\mathbf{Q}^* = \underset{\mathbf{Q}\in\mathbb{R}^{n\times C}}{\arg\min}\ \mathbf{E}(\mathbf{Q},\mathbf{w}).$$

Setting the derivative of $\mathbf{E}$ with respect to $\mathbf{Q}$, $\partial\mathbf{E}/\partial\mathbf{Q}=2\mu\mathbf{S}^t(\mathbf{SQ}-\mathbf{P})+2\nu\mathbf{L}(\mathbf{w})\mathbf{Q}$, to zero yields the necessary optimality condition (since $\mathbf{SP}=\mathbf{P}$ and $\mathbf{SS}=\mathbf{S}$ by definition):

$$\left(\mathbf{S}+\frac{\nu}{\mu}\mathbf{L}(\mathbf{w})\right)\mathbf{Q}^* = \mathbf{P}. \tag{13}$$

Observe that it corresponds to an implicit integration step of graph diffusion of $\mathbf{P}$ based on the metric imposed by the fixed weights, matching the diffusion from Eq. (5) but with the added use of our selection matrix—which only adds extra terms on labeled points. Assuming the diagonal terms of the Laplacian are unit, these extra terms guarantee that labeled nodes will have a likelihood score above $\mu/(\nu+\mu)$ for their original label, and below $\nu/(\nu+\mu)$ for all their other $(C\text{-}1)$ label scores. As a consequence, selecting $\nu/\mu<1$ will guarantee *interpolatory diffusion*. In practice, we will use $\nu/\mu=\frac{1}{2}$ to allow slack since weights are only weakly constrained to sum to one per data point.

Weight optimality. If we now fix matrix $\mathbf{Q}$, the optimal weights $\mathbf{w}^*$ are found through a *convex quadratic programming (QP) problem*:

$$\underset{\mathbf{w}\in\mathbb{R}^{|\mathscr{E}|}}{\arg\min}\ \mathbf{E}(\mathbf{Q},\mathbf{w}) \quad \text{s.t. } \mathbf{w}\succ\mathbf{0}. \tag{14}$$

For computational convenience, our energy can be rewritten (proof omitted as being an unenlightening exercise in linear algebra) as a quadratic form of $\mathbf{w}$ as follows:

$$\mathbf{E}(\mathbf{Q},\mathbf{w}) = \tfrac{1}{2}\mathbf{w}^t\mathbf{A}\mathbf{w} + \mathbf{w}^t\overline{\mathbf{b}} + \mathbf{c} \tag{15}$$

where the sparse matrix $\mathbf{A}\in\mathbb{R}^{|\mathscr{E}|}$ is assembled once and for all as:

$$\mathbf{A} = 2\left(\alpha\,(\mathbf{d}\mathbf{d}^t)\odot(\mathbf{d}\mathbf{X}\mathbf{X}^t\mathbf{d}^t)+\beta\,|\mathbf{d}||\mathbf{d}|^t\right)$$

with $\odot$ denoting the Hadamard (elementwise) product, the vector $\overline{\mathbf{b}}\in\mathbf{R}^{|\mathscr{E}|}$ is easily evaluated based on $\mathbf{Q}$ through:

$$\overline{\mathbf{b}}_{ij} = \gamma\,\overline{\mathbf{w}}_{ij} + \nu\|\mathbf{Q}_i-\mathbf{Q}_j\|^2 - 4\beta,$$

and the vector $\mathbf{c}$ does not depend on $\mathbf{w}$. Note that the matrix $\mathbf{A}$ is positive-definite, sparse and easy to assemble, making the underlying QP problem particularly simple to solve.

3.6. Optimization

The constrained minimization of Eq. (12) can be efficiently performed by exploiting its biconvexity: it lends itself naturally to *alternating minimization*, see Csiszár and Tusnády (1984), where the energy is minimized with respect to each of its two variables in alternation, guaranteeing the decrease of the energy as iterations go.

Algorithm 1 Alternating optimization of weights and labels.

```
Initialize Q = 0
repeat
    Update weights w via sparse QP solve          ▷ Eq. (15)
    Update Q via sparse linear solve              ▷ Eq. (13)
until convergence
Output: Q as the matrix of label likelihoods
```

Alternating steps. We proceed with the minimization of $\mathbf{E}$ as outlined in Algorithm 1: starting from a matrix $\mathbf{Q}$ set to *zero*, we repeatedly solve for optimal weights and optimal label likelihoods $\mathbf{Q}$ until convergence, i.e., until both variables do not

change more than a prescribed threshold per step. This simple procedure was found to perform more efficiently than other strategies: we tried for instance to start with Gaussian weights (Eq. (1)) with σ based on the median edge length of the graph, but found it was actually increasing computational time to reach convergence, while leading to very similar optimal label likelihood scores.

Fast solves. While optimizing $\mathbf{Q}$ corresponds to a simple sparse linear solve using Eq. (13), optimizing $\mathbf{w}$ is more expensive: solving the QP problem in Eq. (14) with existing solvers is significantly slower than a linear solve for large graphs. We instead solve this QP problem by exploiting the simplicity of the inequality constraints and the sparsity of matrix $\mathbf{A}$: we employ a variation of Entropic Mirror Descent (EMD, Beck and Teboulle (2003)), which amounts to iteratively solving for the optimal weights through:

$$\mathbf{w}^{k+1} = \mathbf{w}^k \odot \operatorname{ExpClip}\left(-\left(\mathbf{A}\mathbf{w}^k + \overline{\mathbf{b}}\right)\right),$$

where ExpClip denotes the exponential function applied to each component of its vector argument, with the final values being clipped if they are outside of the range $[1-h, 1+h]$ to prevent too large updates of weights. The value of h is initialized to $\frac{1}{2}$, and reduced by a factor 0.8 each time $\mathbf{E}(\mathbf{w}^{k+1}) > \mathbf{E}(\mathbf{w}^k)$ to force smaller update sizes. This choice is particularly appropriate because of two key factors. First, the weights are constrained to be strictly positive, so this multiplicative update automatically maintains this property (by working in log space). Second, it bypasses the need for an involved line search by exploiting the convexity of the function and by implicitly using entropic regularization (preventing the new weights from moving far-away from the previous weights based on generalized KL divergence, see Beck and Teboulle (2003)): adaptive time stepping along the gradient $\mathbf{A}\mathbf{w}+\overline{\mathbf{b}}$ of the energy is thus parameter-free, and particularly efficient since $\mathbf{A}$ is sparse and updates are achieved in parallel.

3.7. Implementation details

We now go over our classification procedure in more details. In particular, we review how the graph is created and how the various hyperparameters are adapted to the input data.

Graph assembly. From the input (labeled and unlabeled) data points, we form the graph $\mathscr{G}$ by creating an edge $\mathbf{e}_{ij} \in \mathscr{E}$ between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ iff $\mathbf{x}_j$ is among the K nearest neighbors of $\mathbf{x}_i$ or vice-versa. This graph construction is common to all previous works, and we use a value for $K=10$ in all our examples (with the exception of Swirls (Figs. 2&4) where we use 20 neighbors to better demonstrate visually the behavior of weight optimization).

Insertion of interclass edges. Moreover, we insert additional edges between *labels*: for each labeled node $\mathbf{x}_i$ of class c, we add edges to 10% (at least 5 for small datasets) of the other labeled nodes of the same class, picked randomly within the input. These additional "randomized" label-to-label edges help making sure that labels have a big role to play in deciding edge weights: they allow for long edges acting as *bridges* between different spatial components of the same class. Previous methods did not tailor the graph based on labels as it may create more valence imbalance; instead, our Laplacian parameterization exploits the presence of these added label-to-label edges to find a more appropriate Laplacian to perform the label diffusion with, while normalization prevents the typical issues with high valences.

Choice of weight priors. In order to constrain the range search of weights and prevent overfitting, we use a vector $\overline{\mathbf{w}}$ to act as a prior on weights: for every edge $\mathbf{e}_{ij}$ that is *not* linking two labeled nodes of the same class, we set: $\overline{\mathbf{w}}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^4$, to induce smaller weights on long edges; all other values of $\overline{\mathbf{w}}$ are set to zero. The Euclidean distance to the power 4 was chosen after several experiments, which showed in particular that squared distances or inverse Gaussian weights did not perform as well. Other priors could be used, based on geometric quantities such as distances between tangent spaces as in Wang et al. (2011) and accumulated path curvatures, or even non-geometric data specific choices such as enforcing sparsity through ℓ_1 regularization as proposed in Rustamov and Klosowski (2018). We only used the distance to power $p=4$ in all our tests not to obfuscate the interpretation of our results.

Fixing hyperparameters. Hyperparameter selection can be performed using cross-validation. However, when very few labels are known (e.g., COIL-20 dataset with 1 label per class, see Fig. 1), cross-validation is inadequate. In order to overcome this issue, we apply a set of simple heuristics for parameter selection that can be used for *any* input dataset. While our classification energy $\mathbf{E}$ in Eq. (11) uses five hyperparameters, we actually keep $\gamma=1$ as a gauge, and compute data-dependent values for most of the others as follows. We begin with a "canonical" energy using $\mu=2$ and $\nu=\alpha=\beta=\gamma=1$, corresponding to a diffusion step size of $1/2$. After the first round of optimizations (one step for $\mathbf{Q}$, one step for $\mathbf{w}$, see Algorithm 1), we adjust the hyperparameters as follows:

- if $1000\gamma\overline{\mathbf{w}}^t\mathbf{w} \le \nu \operatorname{Tr}\left[\mathbf{Q}^t\mathbf{L}(\mathbf{w})\mathbf{Q}\right]$, we change the value of ν to:

$$\nu = \frac{\gamma}{1000}\overline{\mathbf{w}}^t\mathbf{w} / \operatorname{Tr}\left[\mathbf{Q}^t\mathbf{L}(\mathbf{w})\mathbf{Q}\right].$$

  The rationale behind this choice comes from the fact that the trace term is the only "cross term" for which $\mathbf{Q}$ and $\mathbf{w}$ interact, penalizing edge weights between nodes with strongly varying likelihoods. While this cross term penalizes diffusion across class borders, it may overpower prior-based contribution $\overline{\mathbf{w}}^t\mathbf{w}$ if ν and γ are not chosen carefully to balance the contributions of these terms. In practice, we found that choosing a value for ν such that the second term

Table 1
Datasets tested in this paper.

	Parameters		*Data characteristics*			*Relevant figures*
Dataset	K	α	D	n	C	
Swirls	20	0.5	2	400	4	Figs. 2, 3, 4
COIL-20	10	1	16384	1440	20	Figs. 1, 8(b)
MNIST-1K	10	3	784	1000	10	Figs. 5(left), 9(left)
MNIST-5K	10	3	784	5000	10	Figs. 5(right), 9(right)K
USPS-1K	10	3	256	1000	10	Figs. 6(left), 9(right)
USPS-5K	10	3	256	5000	10	Figs. 6(right), 9(right)
Vowels	10	3	10	990	11	Fig. 10(left)
Seeds	10	3	1024	1990	38	Fig. 10(middle)
Yale	10	3	7	210	3	Fig. 10(right)

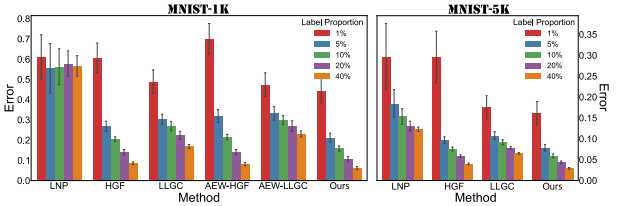


Fig. 5. MNIST. For the MNIST-1K (left) and MNIST-5K (right), Laplacian-optimized classification outperforms all other methods, independent of the number of labels used. Note that AEW cannot handle the 5K dataset in a reasonable amount of time, hence its omission.

is about 1000 times smaller than the fifth energy term after the first iteration of Algorithm 1 leads to the best results, hence this parameter selection.
- if ν is changed, μ is set to 2ν to keep a diffusion step size of $1/2$.
- if the sum of the adjacent edge weights of a node is above 2 or below $1/2$, we increase β by 1 in order to tighten the normalization constraint on the Laplacian.

While these updates of the parameters could be done after each round of optimizations, we found that ν rarely changes after the first time, so we opted to only update ν (and consequently, μ) once. We do keep tracking the normalization term though, increasing β as needed. In practice, we never found a case where β goes over 22, so this adaptation of β is rare and minor in magnitude. The *only* hyperparameter that is not set automatically is α. Our approach thus has the same number of hyperparameters as HGF or LLGC (which use η to adapt to the data). Additionally, our tests show that the optimal choice of α covers only a small range: it is always between 0.5 and 3 in all our examples (see Table 1). Therefore, as usual in SSL, we run our classification in parallel using values of α in the set $\{0.5, 1, 2, 3\}$ on a random subset of the dataset and pick the best of these four values. From our tests, it seems that α has to be raised when the data does *not* have a clear manifold structure (explaining why the swirl dataset uses $\alpha = 0.5$ while the speech vowel, which is more spread out in high dimension, requires $\alpha = 3$ to perform best), since the ratio α/γ indicates how much the manifold assumption is treated over other priors. There may thus be a way to set α directly through a finer evaluation of the input dataset; but tests on many more sets should be performed to validate this assumption. Finally, note that this simple hyperparameter selection is already enough to outperform other graph-based SSL methods as we will demonstrate shortly; but cross-validation can be employed instead (when appropriate) to further improve performance as it can lead to better data-adapted hyperparameter values.

Energy minimization. Alternate optimizations of the two arguments of the energy are performed as explained in Sec. 3.6. For the optimization of $\mathbf{Q}$, we add the identity matrix times 10^{-8} to the Laplacian to remove its kernel and prevent numerical issues; we use the SuperLU solver from scipy.sparse to solve the resulting sparse linear system. For the optimization of the weights $\mathbf{w}$, EMD iterations are performed until the relative change of energy becomes smaller than 10^{-7}. Note that this EMD approach scales very well: as we will demonstrate in Sec. 4.6, the computational gain we obtain compared to a QP solver like OSQP from Stellato et al. (2017) is already a factor 10 on small datasets ($n = 1000$), and grows quickly to multiple orders of magnitude with the size of the dataset.

3.8. Discussion

We conclude this section with a few comments on our geometric approach to SSL and possible extensions.

Importance of embedding space. Because we optimize the edge weights to capture the best metric in which to exploit the geometry of the input data, our approach is less sensitive to the choice of embedding space than methods based on pre-canned weights. However, the cluster and manifold assumptions are all the more valid if data is captured in a space where coordinates are meaningful. While our results show good behavior on typical datasets, finding a *better* feature space for a given dataset would improve the results further.

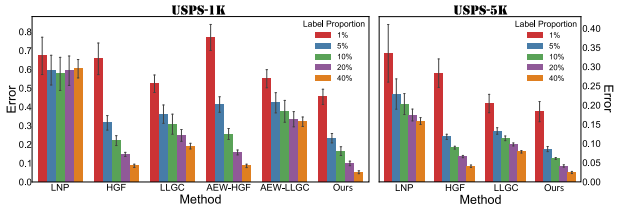


Fig. 6. USPS. For the USPS-1K (left) and USPS-5K (right), Laplacian-optimized classification outperforms all other methods, independent of the number of labels used. Note that AEW cannot handle the 5K dataset in a reasonable amount of time, hence its omission.

Importance of graph connectivity. Our graph connectivity is entirely determined (aside from the few randomized edges between labels of identical classes) by the Euclidean distance between nodes. Using other notions of proximity graphs (e.g. mutual kNN or ε-ball NN) and other distances (or kernels) for this construction could also be considered. The cosine distance $d(\mathbf{x},\mathbf{y}) = 1 - \mathbf{x}^t\mathbf{y}/(\|\mathbf{x}\|\|\mathbf{y}\|)$ is another common notion of distance, which indeed leads to marginal improvements to our results on a few datasets. We kept the Euclidean distance in all of our tests to allow for fair comparisons across the various datasets.

Post-processing. While we directly infer labels based on the maximum values of the label likelihood scores in $\mathbf{Q}$ for simplicity, more involved techniques could be used, using either k-means clustering or entropy minimization (in which case, a three-way alternating minimization approach could be performed). Moreover, class mass normalization could be used if class sizes are known as priors, as in Zhu et al. (2003).

Frobenius norm vs. cross-entropy. A sum-of-squares error function is not the most appropriate for classification problems: it corresponds to a maximum likelihood estimation with the assumption of Gaussian distributed target data. Using an entropy-based energy in lieu of $\|\mathbf{SQ} - \mathbf{P}\|_F^2$ is theoretically preferable. However, it would make the (still convex) minimization with respect to $\mathbf{Q}$ no longer as simple as a sparse linear solve. So any labeling benefit derived from this change in our energy would suffer increased computational times.

Interpretation of edge weight optimization. As expected due to our used of geometric data priors, we observe that our SSL implicitly learns an intrinsic metric for the data in which an anisotropic label diffusion is performed. Indeed, Fig. 4 demonstrates that the larger the edge weight is, the more likely the associated edge vertices are from the same class. Diffusivity is thus strong within each class, but weak across classes.

Transductive vs. inductive learning. In most graph-based frameworks, semi-supervised learning is often only transductive, i.e., the classification is only done for the unlabeled points in the training set. LNP Wang et al. (2006) suggested that inductive learning can also be performed, i.e., the classification of a new point which was not an unlabeled point in the training set can be performed as well. We can employ the same strategy by estimating the weights for the neighborhood of the new point locally once we added it to the graph without retraining the whole set, and then use diffusion of class labels to perform class inference for the new point.

4. Results and comparisons

In this section, we evaluate our Laplacian-optimized SSL method on a number of datasets with various characteristics (data size, dimension of the embedding space, number of classes, etc) to study its efficiency and classification accuracy. We also compare our results to previous methods in order to understand the pros and cons of our approach.

4.1. SSL methods

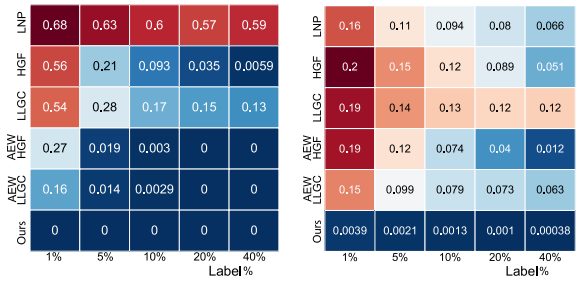
In all our tests, we compare our method to HGF from Zhu et al. (2003), LLGC from Zhou et al. (2003), LNP from Wang et al. (2006), as well as the most recent AEW-HGF and AEW-LLGC (i.e., the AEW-based optimization of edge weights from Karasuyama and Mamitsuka (2017), followed by either HGF-based or LLGC-based label diffusion as recommended by the authors). These methods were shown to outperform all basic classification techniques (such a nearest-neighbor voting or SVM), so we did not include earlier SSL techniques for conciseness.

For HGF and LLGC that use Gaussian weights, we use a standard deviation σ equal to the median edge length and use the diffusion parameters recommended in the original papers. Given its heavy computational requirements, we implemented AEW with TensorFlow (Abadi et al. (2015)) to exploit parallelism. Finally, OSQP (Stellato et al. (2017)) is used to solve the QP problems arising in LNP, as it systematically outperformed CVX (Grant and Boyd (2014)).

4.2. Datasets

In order to test our technique on very different types of data, we selected the following well-known datasets as they cover geometric, image, sound and biology datasets and they have been extensively tested in previous SSL papers. Each of these datasets contains all the labels for all the data points; we can thus test classification methods using only a percentage (varying from 1% to 40%) of randomly selected labels, and compare the results to the ground-truth labels.

Fig. 7. Mislabeled images. For MNIST (left) and USPS (right), four examples of images that our approach misclassifies for 1% of labels.



(a) Error proportions for Swirls (b) Error proportions for COIL

Fig. 8. Heatmaps. Laplacian-optimized classification performs systematically better than existing SSL algorithms for various percentage of labels in the swirls and COIL-20 datasets. The other trained approach, AEW, often performs 10 times worse, while predefined-weights approaches lead to even worse results.

Swirls. This pointset in $D=2$ dimensions contains 4 classes (one spiraling arm per class) of 100 points each (thus, $n=400$). This simplistic test (inspired by a similar dataset in Karasuyama and Mamitsuka (2017)) already highlights how well a variational SSL technique can leverage the manifold structure of data, compared to deciding labeling purely based on shortest distances; see Fig. 2. Despite its low-dimensionality, symmetry and obvious manifold structure, other methods surprisingly struggle with this example. This indicates that previous SSL approaches could easily fail even if applied to the output of one of the many existing dimensionality reduction algorithms (e.g., Isomap and LLE).

COIL. The Columbia University Image Library (COIL-20 from Nene et al. (1996)) dataset contains $n=1440$ grayscale images of 128×128 pixels (hence, $D=16384$). There are 72 images per class, and 20 classes (see class examples in Fig. 1). Each class consists in an object (toy car, sandwich, mug, etc) photographed from different angles.

MNIST. The MNIST dataset is a very common collection of grayscale images of handwritten digits of size 28×28 pixels (thus, $D=784$), with 10 classes (digits 0 to 9). We consider several subsets to test the influence of the size of the input, with $n=500, 1000, 2000, 5000$ and 10000.

USPS. The USPS dataset is similar to the MNIST dataset (grayscale images of digits as well), but with a coarser resolution (16×16 pixels, $D=256$). Just like for MNIST, we also consider several subsets with n varying from 500 to 10000 to test scalability and accuracy.

Vowels. This dataset of Dheeru and Taniskidou (2017) uses the *Deterding numerical description* of $n=990$ British-English vowel sounds pronounced by different people (10 scores per sound, $D=10$), for 11 different vowels.

Yale faces. The Yale Face Dataset B of Georghiades et al. (2001) consists of 32×32 grayscale images ($D=1024$) of 38 people with different lighting conditions and facial expressions. There are 50 images per person, leading to a total number of points of $n=1990$.

Seeds. The Wheat Seeds dataset of Dheeru and Taniskidou (2017) contains descriptions of seed kernels belonging to three different types of wheat, with 70 observations for each (hence, $n=210$). Each seed kernel is represented with $D=7$ geometric features (area, perimeter, compactness, etc).

4.3. Measuring accuracy

To obtain a reasonable estimate of a method's classification performance and to adhere to common practices of the SSL community, each of our results is obtained by running 20 labeling processes with a fixed amount of randomly-selected input labels. For each run, we compute the 0/1 error, i.e., the proportion of mislabeled data points:

$$\text{Error} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\text{label}(i)\neq\text{ground\_truth}(i)}. \tag{16}$$

We then report the *mean* and *standard deviation* of this error over the 20 runs. Observe that random guessing of labels over C classes would result in an error of $1-1/C$, so any mean error below this value indicates classifying power.

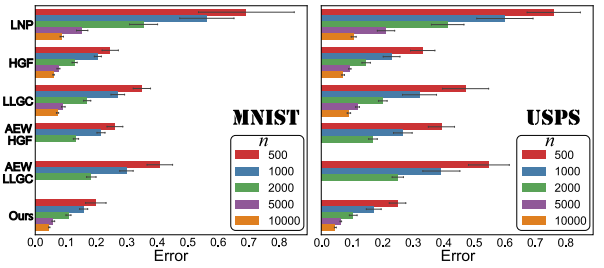


Fig. 9. Varying data sizes. By randomly subsampling MNIST (left) / USPS (right), we can test the various SSL algorithms for varying data sizes (from $n=500$ to $10,000$) and 10% of labels. Our Laplacian-optimized classifier exceeds all other approaches in accuracy, as demonstrated through barplots using the 0/1 error (proportion of mislabeled points) averaged over 20 runs with randomly selected labels.

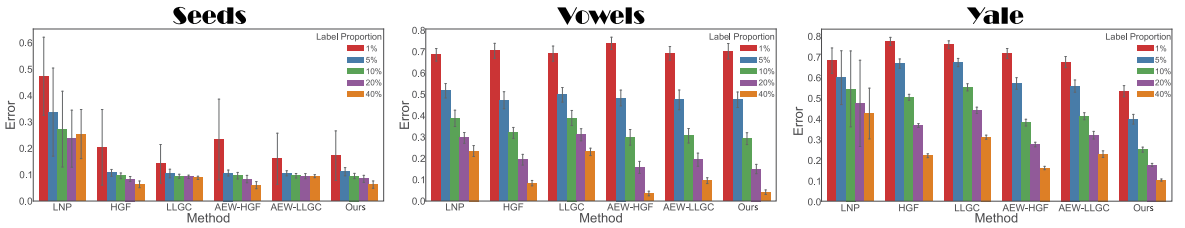


Fig. 10. Varying label proportions. On all the datasets we tried (here, seeds/vowels/Yale), classification performs better as the number of labels grows. While a low percentage of labels is common, we note that the accuracy of different methods increases at different rates with the percentage of labels. Our Laplacian-optimized approach behaves consistently on all datasets, quickly increasing accuracy as the number of labels increases.

4.4. Varying number of input labels

We tested our approach and other SSL techniques for varying proportions of labeled points: 1%, 5%, 10%, 20% and 40%. While real datasets cannot be reasonably expected to have more than 10% of labeled points, this test is important to prove "convergence" as the number of labels increases.

For Swirls, our Laplacian-optimized approach is the only one that recognized the four spiral-shaped classes perfectly at all label proportions, as seen in Fig. 8(a). Surprisingly, LNP improves only marginally with the number of labels, pointing to the need of having some notion of distance for the weights to be efficient at labeling. The AEW-optimized weights can only reconstruct the four classes perfectly when there is more than 20% of labels: this poor performance is probably due to the fact that having only one coefficient per dimension in $\mathbb{R}^2$ is not enough to adapt to the data.

For COIL, our approach significantly outperforms all other methods for all label proportions as well, see Fig. 8(b). LNP fares better than on swirls, but is only marginally better than HGF. Notice also that the AEW-optimized weights perform quite differently if they are used with HGF or LLGC for label propagation, which makes sense since the functional being minimized is not really related to either of these diffusion processes, unlike in our framework. Starting at 10% of labels, our technique only misclassifies a maximum of two images (see Fig. 1), which is 10 to 25 times better than other methods, and these two images are arguably taken from an angle that makes them similar to another class (class 6 in the figure, instead of class 5).

For the MNIST-1K, MNIST-5K, USPS-1K, and USPS-5K datasets, we also outperform the other methods systematically as demonstrated by the barplots in Figs. 5 and 6. Here again, LNP fails to show much improvement as the number of labels increases. While HGF performs very poorly for 1% of labels, it produces surprisingly good results for higher percentages. As expected, HGF and our approach are always interpolatory, while LNP, AEW-LLGC and LLGC quickly start violating input information in almost every tested dataset; for MNIST-1K, they mislabel 1-2 input data points when only 5% labels are given, and for 40% labels, they make respectively 182, 51 and 16 mistakes on average. Note that we did not include AEW results for the 5K datasets: the optimization required by AEW did not converge even after six hours using TensorFlow; but even for 1K datasets, AEW-optimized weights actually perform worse than Gaussian weights.

The other datasets offer more uniform results, see Fig. 10. In particular, all the methods perform about the same for the vowel dataset, which suggests that either there are not enough points, or there is a lack of well-defined geometric structures in this dataset. Still, our diffusion-optimized labeling performs nearly best on all these examples for all label proportions, with a noticeable improvement over previous methods for the Yale dataset. Table 2 recaps our results.

4.5. Varying dataset sizes

For our next set of experiments, we kept the proportion of labels fixed, but increased the size of the dataset instead. Intuitively, the more populated the dataset is, the more clear the geometric information we rely on should become, since the presence of clusters or manifolds ought to be more obvious as the number of unlabeled data points increases.

Table 2
Varying proportion of labels (with mean and standard deviation of 0/1 error, Eq. (16)).

Error		LNP		HGF		LLGC		AEW-HGF		AEW-LLGC		Ours	
Dataset	%labels	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
	1	0.676	0.056	0.564	0.057	0.538	0.061	0.273	0.122	0.157	0.076	**0.000**	0.000
	5	0.633	0.073	0.213	0.056	0.281	0.064	0.019	0.031	0.014	0.022	**0.000**	0.000
Swirls	10	0.596	0.096	0.093	0.039	0.173	0.045	0.003	0.013	0.003	0.007	**0.000**	0.000
	20	0.568	0.057	0.035	0.031	0.147	0.045	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	40	0.589	0.038	0.006	0.014	0.128	0.055	**0.000**	0.000	**0.000**	0.000	**0.000**	0.000
	1	0.612	0.111	0.606	0.076	0.489	0.060	0.701	0.078	0.474	0.060	**0.444**	0.062
	5	0.556	0.125	0.268	0.026	0.304	0.025	0.320	0.032	0.332	0.034	**0.212**	0.023
MNIST-1K	10	0.562	0.091	0.205	0.012	0.270	0.023	0.215	0.013	0.299	0.023	**0.158**	0.014
	20	0.577	0.065	0.141	0.014	0.224	0.020	0.141	0.013	0.269	0.027	**0.107**	0.012
	40	0.567	0.052	0.086	0.007	0.168	0.010	0.081	0.008	0.230	0.015	**0.062**	0.007
	1	0.672	0.102	0.656	0.086	0.524	0.048	0.770	0.071	0.550	0.050	**0.454**	0.042
	5	0.596	0.081	0.315	0.040	0.362	0.050	0.413	0.042	0.422	0.055	**0.233**	0.026
USPS-1K	10	0.576	0.091	0.220	0.027	0.308	0.055	0.256	0.030	0.377	0.060	**0.164**	0.024
	20	0.593	0.080	0.147	0.011	0.248	0.032	0.157	0.013	0.334	0.042	**0.099**	0.012
	40	0.604	0.050	0.086	0.009	0.191	0.016	0.086	0.009	0.321	0.026	**0.052**	0.008
	1	0.297	0.081	0.296	0.063	0.176	0.028	Not converged				**0.162**	0.028
	5	0.184	0.035	0.097	0.008	0.108	0.009	Not converged				**0.078**	0.008
MNIST-5K	10	0.154	0.019	0.076	0.005	0.092	0.006	Not converged				**0.059**	0.005
	20	0.130	0.012	0.059	0.003	0.078	0.003	Not converged				**0.044**	0.003
	40	0.123	0.005	0.040	0.002	0.065	0.002	Not converged				**0.029**	0.002
	1	0.335	0.077	0.283	0.038	0.204	0.025	Not converged				**0.183**	0.027
	5	0.229	0.040	0.117	0.007	0.133	0.009	Not converged				**0.086**	0.007
USPS-5K	10	0.203	0.028	0.089	0.004	0.114	0.006	Not converged				**0.061**	0.003
	20	0.174	0.016	0.066	0.003	0.098	0.004	Not converged				**0.042**	0.003
	40	0.159	0.009	0.041	0.003	0.078	0.003	Not converged				**0.025**	0.003
	1	0.685	0.030	0.704	0.036	**0.691**	0.036	0.739	0.029	0.692	0.033	0.702	0.037
	5	0.517	0.035	**0.472**	0.041	0.498	0.035	0.483	0.038	0.474	0.047	0.475	0.037
Vowels	10	0.388	0.039	0.319	0.026	0.389	0.036	0.299	0.038	0.306	0.035	**0.292**	0.028
	20	0.296	0.026	0.194	0.026	0.311	0.029	0.158	0.029	0.194	0.032	**0.150**	0.022
	40	0.234	0.026	0.084	0.013	0.230	0.018	**0.035**	0.011	0.096	0.014	0.042	0.011
	1	0.474	0.152	0.204	0.147	**0.141**	0.075	0.233	0.157	0.160	0.100	0.172	0.096
	5	0.337	0.171	0.107	0.013	**0.105**	0.017	0.106	0.011	**0.105**	0.010	0.111	0.016
Seeds	10	0.273	0.147	0.096	0.011	**0.094**	0.008	0.096	0.013	0.096	0.009	0.095	0.010
	20	0.237	0.111	0.084	0.010	0.092	0.007	**0.083**	0.014	0.094	0.010	0.084	0.014
	40	0.254	0.096	0.063	0.013	0.088	0.006	**0.060**	0.014	0.094	0.006	0.062	0.014
	1	0.684	0.064	0.778	0.020	0.761	0.020	0.718	0.025	0.677	0.028	**0.535**	0.028
	5	0.601	0.134	0.670	0.023	0.675	0.021	0.573	0.029	0.560	0.031	**0.399**	0.024
Yale	10	0.546	0.190	0.505	0.015	0.554	0.018	0.382	0.017	0.413	0.017	**0.251**	0.013
	20	0.476	0.215	0.368	0.010	0.442	0.016	0.276	0.011	0.321	0.019	**0.177**	0.008
	40	0.427	0.127	0.221	0.010	0.312	0.010	0.161	0.009	0.229	0.016	**0.103**	0.005

We used the MNIST and USPS datasets for this test, and randomly downsampled them to $n = 10000, 5000, 2000, 1000$, and 500 to have five testing sizes. Once again, AEW was only performed for sizes below 2000, as the optimization would not finish in less than six hours for the largest size. As demonstrated in Fig. 9 through horizontal barplots, all methods benefit from an increase in data points, giving credence to the geometric hypothesis that datasets are spatially distributed in a fashion that can be exploited for classification. Moreover, Laplacian-optimized weights beat all other methods for all five dataset sizes, be it for MNIST or for USPS. HGF is surprisingly accurate on these datasets (being slightly better than AEW), but fails to match our results. See also Fig. 7 to check examples of images (representing, arguably, very distorted digits) that fail to be properly classified. Table 3 recaps all the results.

4.6. Further evaluation of our approach

Finally, we conducted more tests on our approach to justify some of our previous claims and to offer a better understanding of what makes it competitive.

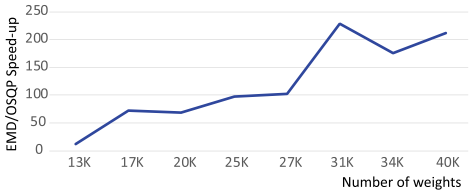
Optimized weights. Even the simplest dataset offers some interesting insights on our approach: Fig. 4 displays the top and bottom 10% of weights ordered by magnitude for the swirls dataset (for $K = 20$) to show where the diffusion is most preferred, and most avoided. While LNP clearly fails to capture the data by overemphasizing local linearity, Gaussian weights only partially capture the manifolds (see the large gaps between red edges) and assign very small weights almost only to edges across spirals (see blue edges). Instead, our Laplacian optimization consolidates the manifolds by assigning all manifold edges with high weights (red edges), while blue edges (i.e., small weights) are not just between spirals, but also for

Table 3
Varying data size with 10% of labels (with mean and standard deviation of 0/1 error, Eq. (16)).

Error		LNP		HGF		LLGC		AEW-HGF		AEW-LLGC		Ours	
Dataset	N	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
MNIST	500	0.692	0.161	0.245	0.027	0.349	0.029	0.261	0.027	0.408	0.043	**0.198**	0.034
	1000	0.562	0.091	0.205	0.012	0.270	0.023	0.215	0.013	0.299	0.023	**0.158**	0.014
	2000	0.355	0.047	0.129	0.008	0.170	0.012	0.133	0.009	0.183	0.016	**0.108**	0.008
	5000	0.154	0.019	0.076	0.005	0.092	0.006	Not converged				**0.059**	0.005
	10000	0.088	0.007	0.059	0.002	0.073	0.004	Not converged				**0.045**	0.002
USPS	500	0.729	0.086	0.318	0.039	0.453	0.074	0.377	0.042	0.526	0.066	**0.239**	0.027
	1000	0.576	0.091	0.220	0.027	0.308	0.055	0.256	0.030	0.377	0.060	**0.164**	0.024
	2000	0.396	0.052	0.141	0.014	0.193	0.014	0.162	0.014	0.240	0.018	**0.100**	0.012
	5000	0.203	0.028	0.089	0.004	0.114	0.006	Not converged				**0.061**	0.003
	10000	0.101	0.008	0.068	0.004	0.086	0.005	Not converged				**0.044**	0.003

geodesically-distant points, which is the right choice to create a data-adapted Laplacian. In this example, AEW shows nearly no improvement over the Gaussian weights: optimizing the only two scaling hyperparameters does not make much of a difference since the data is not homogeneous in either coordinate and does not exhibit global anisotropy. Another visualization is also instructive: Fig. 3 shows a matrix-form display of the weighted graphs generated by the various methods, for the same graph based on swirls. Clearly, the Gaussian-based and LLE-based weighted graphs connect a lot of nodes in different classes; AEW manages to partially improve the situation, but it still connects different classes. Our Laplacian-optimized approach eliminates undesirable edges, and the 4-class structure becomes very clear, leading to perfect labeling prediction when label diffusion is performed.

EMD vs. SQP. As part of our biconvex optimization, we use an iterative EMD approach instead of a classical QP solver. In order to verify that our alternate solver leads to speed improvement, we ran weights optimizations for various sizes of graphs (we used MNIST-5K with an increasing amount of nearest neighbors K to obtain examples with a growing amount of edges). As the inset indicates, we see several orders of magnitude improvements compared to OSQP Stellato et al. (2017), already for sizes of the order of 40K.



Alternating optimization. One could think that the single "cross term" using both $\mathbf{Q}$ and $\mathbf{w}$ is not enough to play a big role in our energy. Yet, on COIL for 1% of labeled points, using just the first optimization round of weights and labels leads to a classifying error of 1%, while after 5 rounds of alternating minimization, the error goes down to 0.39%. Even for 5% of labeled data, the error after one round of optimization is 0.35%, and goes to 0.21% after convergence. Our alternating optimization can thus have a significant effect on the classifier's accuracy. We note also that the optimization of $\mathbf{w}$ becomes faster as the solver goes on: while the first weight optimization takes 370 EMD iterations on the COIL-20 dataset, the second optimization round only takes 170 EMD iterations, and all subsequent rounds take less than 30 EMD iterations.

Timings. All methods first involve the same graph construction (which can be implemented in $\mathcal{O}(n \log n)$ with a dedicated nearest-neighbor data structure for high dimensions), so we disregard this preprocessing step in our discussion of timings. HGF and LLGC are the fastest, as they only involve a sparse linear solve. LNP is more time consuming due to the fact that finding optimal weights requires solving $2n$ QP solves and a non-symmetric matrix in the diffusion solve. Our algorithm is in between these methods: as a concrete example, on the same laptop and using COIL-20 with 1% of labels, our approach takes 4 seconds (excluding the offline computation of the matrix A, which has complexity of $\mathcal{O}(nD)$ and can be assembled efficiently through parallelization), while LNP took 14 seconds, and HGF and LLGC took 0.1 seconds. Comparatively, AEW took 137 seconds, that is, over thirty times slower than our approach. Note finally that based on our tests, the empirical complexity of our EMD-based biconvex optimization for classification is $\mathcal{O}(n^{1.6})$, although a more efficient implementation may further reduce this empirical computational complexity.

5. Conclusions

In this paper, we showed that the geometry of complex data can be leveraged through a sparse, nearest-neighbor graph and an optimized Laplacian over this graph, in order for a label diffusion to directly infer classification of all unlabeled nodes. This simple algorithm matches or exceeds the accuracy of all previous diffusion-based SSL techniques on a variety of datasets. At its core is a biconvex energy which optimizes the edge weights to directionally adapt a graph diffusion process to propagate labels over the dataset by exploiting its intrinsic geometric structure.

Future work. Our work raises a number of interesting research questions mixing machine learning and geometry in high dimensional spaces. Still in the context of SSL, inductive learning is trivial to incorporate in our framework as we briefly discussed earlier: for any new data point, we can recompute optimal weights in its two-ring neighbors and use the optimized weights to label it. However, this approach will become inefficient if a significant number of new points are added. Finding ways to update weights and likelihood scores efficiently could be useful in online applications for instance. Applying the graph convolutional network method of Kipf and Welling (2017) to our optimized Laplacian is also an intriguing direction, which could allow the method to scale to large datasets more efficiently. Our geometric approach to classification could also be used for geometric datasets: meshes or pointsets could be encoded via eigenvalues or feature vectors, and classification in one of these embedding spaces could be performed as explained in this document. However, it would be interesting to study which embedding space would be more likely to result in high accuracy classification from labels for this specific type of data. Even for $D=3$, our approach could detect noise and outliers vs. surface samples for dense pointsets: if a user can label a few parts of the pointsets into one of these two classes, inference on the rest of the data could be performed through our approach. Whether this approach would compete against random forests for this type of low-dimensional data remains to be seen. Clustering is another learning task that may benefit from geometric insights. In this unsupervised case, an input must be grouped into sets in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups. The absence of labels renders our approach inadequate, obviously. However, our optimization of the Hodge star can still be very relevant, as it can be made to identify clusters and manifolds like we did for SSL. Adapting our work to current state-of-the-art k-means methods for clustering like Shah and Koltun (2017) or spectral-based approaches that rely on edge weights may prove very fruitful. More generally, the computational efficiency of our optimized Laplacian offers insight on the viability of machine learning on point sets as the design of convolutional pointsets has been a particularly active research topic of late. Developing additional discrete differential geometry tools that apply to high dimensions could also be a rich source of new algorithms for data analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Rick Szeliski for very early feedback, and Tong Zhao and Florent Lafarge for proof-reading our first draft. This work was partially supported by NSF grants DMS-1721024 and IIS-1900473.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/.

Abraham, R., Marsden, J., Ratiu, T., 1988. Manifolds, Tensor Analysis, and Applications. Applied Mathematical Sciences, vol. 75. Springer-Verlag.

Beck, A., Teboulle, M., 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. Oper. Res. Lett. 31, 167–175.

Belkin, M., Niyogi, P., 2004. Semi-supervised learning on Riemannian manifolds. Mach. Learn. 56, 209–239.

Belkin, M., Niyogi, P., Sindhwani, V., 2006. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. 7, 2399–2434.

Blum, A., Chawla, S., 2001. Learning from labeled and unlabeled data using graph mincuts. In: ICML, pp. 19–26.

Bossavit, A., 1998. Computational Electromagnetism. Academic Press.

Budninskiy, M., Liu, B., Tong, Y., Desbrun, M., 2017. Spectral affine-kernel embeddings. Comput. Graph. Forum 36, 117–129.

Budninskiy, M., Yin, G., Feng, L., Tong, Y., Desbrun, M., 2019. Parallel transport unfolding: a connection-based manifold learning approach. SIAM J. Appl. Algebra Geom. 3, 266–291.

Chapelle, O., Schlkopf, B., Zien, A., 2010. Semi-Supervised Learning. MIT Press.

Crane, K., de Goes, F., Desbrun, M., Schröder, P., 2013. Digital geometry processing with discrete exterior calculus. In: ACM SIGGRAPH 2013 Course Notes.

Csiszár, I., Tusnády, G., 1984. Information geometry and alternating minimization procedures. Stat. Decis. 1 (Supplement Issue), 205–237.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the em algorithm. J. R. Stat. Soc., Ser. B, Methodol. 39, 1–38.

Desbrun, M., Kanso, E., Tong, Y., 2008. Discrete differential forms for computational modeling. In: Bobenko, et al. (Eds.), Discrete Differential Geometry. Birkhäuser, Basel, pp. 287–324.

Desbrun, M., Meyer, M., Schröder, P., Barr, A., 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In: ACM SIGGRAPH, pp. 317–324.

Dheeru, D., Taniskidou, E.K., 2017. UCI Machine Learning Repository. U. of California, Irvine. http://archive.ics.uci.edu/ml.

Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J., 2001. From few to many. IEEE Trans. Pattern Anal. Mach. Intell. 23, 643–660.

Grant, M., Boyd, S., 2014. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx.

Karasuyama, M., Mamitsuka, H., 2017. Adaptive edge weighting for graph-based learning algorithms. Mach. Learn. 106, 307–335.

Kipf, T., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations, pp. 1–14. OpenReview.Net.

Nene, S.A., Nayar, S.K., Murase, H., 1996. Columbia Object Image Library (COIL-20). Technical Report CUCS-005-96. Columbia University.

Olah, C., 2014. Neural networks, manifolds, and topology from Colah's blog. http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/.

Playground, T., 2018. Playground. http://playground.tensorflow.org.

Rasmus, A., Valpola, H., Honkala, M., Berglund, M., Raiko, T., 2015. Semi-supervised learning with ladder networks. In: NIPS, pp. 3546–3554.

Roweis, S.T., Saul, L.K., 2000. Nonlinear dimensionality reduction by locally linear embedding. Science 290, 2323–2326.

Rustamov, R.M., Klosowski, J.T., 2018. Interpretable graph-based semi-supervised learning via flows. In: AAAI Conference on Artificial Intelligence, pp. 3976–3983.
Shah, S.A., Koltun, V., 2017. Robust continuous clustering. Proc. Natl. Acad. Sci. 114, 9814–9819.
Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 22, 888–905.
Sindhwani, V., Belkin, M., Niyogi, P., 2010. Semi-supervised Learning, pp. 35–54. Chapter: The geometric basis of SSL. In: Chapelle et al. (2010).
Solomon, J., Rustamov, R.M., Guibas, L., Butscher, A., 2014. Wasserstein propagation for semi-supervised learning. In: ICML, pp. 306–314.
de Sousa, C.A.R., 2015. An overview on the Gaussian fields and harmonic functions method for semi-supervised learning. In: Int. Joint Conf. on Neural Networks, pp. 1–8.
Stellato, B., Banjac, G., Goulart, P., Bemporad, A., Boyd, S., 2017. OSQP: an operator splitting solver for quadratic programs. ArXiv e-prints arXiv:1711.08013.
Subramanya, A., Bilmes, J., 2011. Semi-supervised learning with measure propagation. J. Mach. Learn. Res. 12, 3311–3370.
Vesely, K., Hannemann, M., Burget, L., 2013. Semi-supervised training of deep neural networks. In: IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 267–272.
Wang, F., Wang, J., Zhang, C., Shen, H.C., 2006. Semi-supervised classification using linear neighborhood propagation. In: CVPR, pp. 160–167.
Wang, F., Zhang, C., 2008. Label propagation through linear neighborhoods. IEEE Trans. Knowl. Data Eng. 20, 55–67.
Wang, Y., Jiang, Y., Wu, Y., Zhou, Z.H., 2011. Spectral clustering on multiple manifolds. IEEE Trans. Neural Netw. 22, 1149–1161.
Zelnik-Manor, L., Perona, P., 2004. Self-tuning spectral clustering. In: NIPS, pp. 1601–1608.
Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B., 2003. Learning with local and global consistency. In: NIPS, pp. 321–328.
Zhou, X., Belkin, M., 2011. Semi-supervised learning by higher order regularization. In: International Conference on Artificial Intelligence and Statistics, vol. 15, pp. 892–900.
Zhu, X., 2007. Semi-Supervised Learning Literature Survey. Technical Report 1530. Computer Sciences, U. of Wisconsin–Madison.
Zhu, X., Ghahramani, Z., Lafferty, J., 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML, pp. 912–919.
Zhu, X., Goldberg, A.B., 2009. Introduction to semi-supervised learning. Synth. Lect. Artif. Intell. Mach. Learn. 3, 1–130.