Graph-based Semi-Supervised Learning

Ashwin De Silva* **Jeremy Welland*** **Sai Koukuntla*** **Zhenghan Fang***
Dept. of Biomedical Engineering
Johns Hopkins University

1 Introduction

In this work, we study graph-based methods for semi-supervised learning (SSL), presenting the theoretical motivations of these methods, and experimental results on benchmark datasets. Our code is available at `https://github.com/Laknath1996/semi-supervised-learning`.

Problem Statement Suppose we have a large number of samples $x_1, \ldots, x_n \in \mathcal{X} \subseteq \mathbb{R}^d$, where only the first $m << n$ such samples (which we can assume, without loss of generality, to be the first m such samples) have labels $y_i = f(x_i) \in \mathcal{Y} = \{1, \ldots, C\}$. Here, we have assumed that there are C classes and $f : \mathcal{X} \mapsto \mathcal{Y}$ is some labeling function that assigns one of these class labels to to each sample x_i. In semi-supervised learning, we wish to learn a function $\hat{f}$ using the entirety of n samples such that $\hat{f}$ may achieve a low misclassification error on the unlabeled samples:

$$err(\hat{f}) = \frac{1}{n-m} \sum_{i=m+1}^{n} \mathbb{1}\{\hat{f}(x_i) \neq f(x_i)\}.$$

The main idea here is that we can exploit the total amount of samples we have to understand the underlying geometry of the data, and leverage this information to learn an appropriate classifier $\hat{f}$.

While there are a number of different ways of tackling the semi-supervised learning problem such as generative, consistency-regularization, and pseudo-labeling methods (Yang et al., 2022), in this work we restrict our focus to graph-based semi-supervised learning.

Graph-based semi-supervised learning We can represent each sample by a vertex in a graph that measures some degree of similarity between the samples. Such a graph may be able to represent the underlying structure of the data and this information can be exploited in the context of the semi-supervised learning problem. After constructing a similarity graph using both labeled and unlabeled samples, the labeled information can be propagated to the unlabeled samples through the learned graph. Note that the large amount of unlabeled samples allows us to construct a rich graph that encodes the geometric structure of the data. This forms the basis for graph-based semi-supervised learning techniques. In this work, we consider 4 such methods and discuss their theoretical motivations, algorithmic implementations, experimental results on several benchmarks, and limitations in detail.

2 Laplacian Eigenmaps-based SSL

2.1 Overview

Under the assumption that the data resides on a submanifold within a high diemsional space, Belkin & Niyogi (2002) aim to develop an algorithmic framework that can classify a partially labeled dataset. The main idea of this framework is that the classification functions f are naturally defined only on the said submanifold instead of the total ambient space of the data. They argue that one can produce a basis for the Hilbert space of square-integrable functions on the submanifold using the Laplace-Beltrami operator. This basis can be extracted using only the unlabeled samples, and once it is recovered, a small number of labeled samples can be used to construct a classifier defined on the manifold. The algorithm models the manifold structure using a graph constructed using the nearest neighbors of each sample and approximates the Laplace-Beltrami operator using the graph Laplacian matrix.

*All members contributed equally. See section 7 for a summary of individual contributions.

2.2 Algorithmic Description

For simplicity, suppose that we are solving a binary semi-supervised classification problem with labels $y_i \in \{-1, 1\}; i = 1, \ldots, m$.

The first step of the algorithm is to construct a graph G where each sample in the dataset is represented by a corresponding vertex. An edge exists between vertex i and j only if x_i is among the k nearest neighbors of x_j or x_j is among the k nearest neighbors of x_i. We use the standard Euclidean distance in $\mathbb{R}^d$ when computing the nearest neighbors of each sample. The computation of the adjacency matrix $W_{n\times n}$ of G can be summarized as,

$$W_{ij} = \begin{cases} 1, & \text{if } x_i \in k\text{-nearest neighbors}(x_j) \text{ or } x_j \in k\text{-nearest neighbors}(x_i), \\ 0, & \text{otherwise.} \end{cases}$$

The unnormalized graph Laplacian matrix L of G is given by $L = D - W$, where D is the degree matrix of G i.e. a diagonal matrix that satisfies $D_{ii} = \sum_j W_{ij}$. It is important to note that L is a positive semi-definite matrix and it can be thought of as an operator on functions defined on the vertices of G. In the second step, we compute the p eigenvectors $e_1, \ldots, e_p$ corresponding to the p smallest eigenvalues of L.

Next, we build a simple linear classifier on the space spanned by $e_1, \ldots, e_p$ by performing the following ordinary least squared regression in the p-dimensional space obtained by the mapping $x_i \mapsto (e_1(i), \ldots, e_p(i))$.

$$a^* = \underset{a}{\operatorname{argmin}} \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{p} a_j e_j(i) \right)^2 \tag{1}$$

Here, the minimization is considered over the space of coefficients $a = (a_1, \ldots, a_p)^\top$. The solution to this least squares problem is simply given by,

$$a^* = (U^\top U)^{-1} U^\top y,$$

where $y = (y_1, \ldots, y_m)^\top$ and U is a $m \times p$ matrix with $U_{ij} = e_j(i)$. Finally, the classification on the unlabeled samples can be performed by,

$$y_i = \hat{f}(x_i) = \begin{cases} 1, & \text{if } \sum_{j=1}^{p} a_j e_j(i) \geq 0 \\ -1, & \text{otherwise} \end{cases}; \quad i = m+1, \ldots, n.$$

This can be extended for multiclass problems by considering one-vs-all classifiers compete using $\sum_{j=1}^{p} a_j e_j(i)$ as the confidence measure. In Fig. 1, we illustrate the Laplacian eigenmaps-based SSL algorithm in action.

2.3 Theoretical Foundations

Let $\mathcal{M}$ be an l-dimensional compact manifold isometrically embedded in $\mathbb{R}^d$ and μ be the volume form induced by $\mathcal{M}$. The square integrable functions on $\mathcal{M}$ forms a Hilbert space $\mathcal{L}^2(\mathcal{M})$ with the inner product $\langle f, g \rangle = \int_{\mathcal{M}} fg d\mu$ and norm $\|f\|^2 = \int_{\mathcal{M}} |f|^2 d\mu$ for any $f, g \in \mathcal{L}^2(\mathcal{M})$. With this background, we discuss the following 3 points associated with the Laplace-Beltrami operator Δ which are instrumental for the algorithm described above.

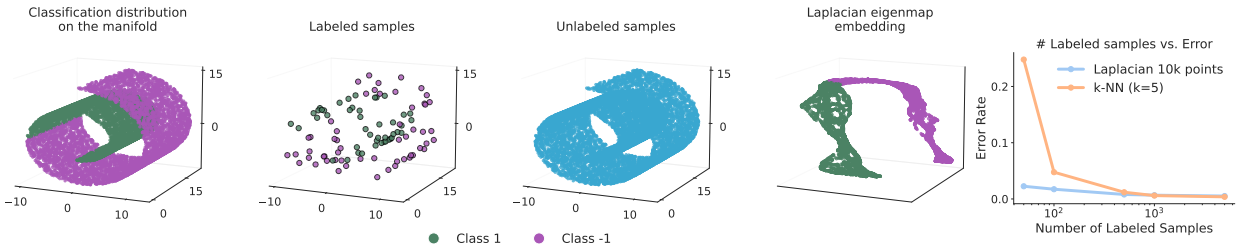


Figure 1: This is a simple illustration of Laplacian eigenmaps-based SSL. Going from left to right, first we have the classification distribution defined on a swiss-roll manifold. Next, we illustrate the a small amount of labeled samples drawn from the classification distribution. In the third panel, we have the large amount of unlabeled samples which could likely provide us with rich information about the manifold underlying the data. The penultimate panel showcases the Laplacian eigenmap embedding in 3-dimensions learnt from the union of labeled and unlabeled samples. It is evident that as little as 3 eigenvectors are able to represent the data in a linearly separable manner. Finally, we illustrate the performance of the Laplacian eigenmaps-based SLL against a k-NN at increasing levels of labeled sample size.

Eigenfunctions of the Laplacian operator provides a natural basis on $\mathcal{L}^2(\mathcal{M})$ The Laplacian operator Δ of $\mathcal{M}$ is self-adjoint positive semi-definite and its eigenfunctions form a natural basis for $\mathcal{L}^2(\mathcal{M})$. If $\mathcal{M}$ is compact, the spectrum of Δ is discrete with the smallest eigenvalue $\lambda_1 = 0$ corresponding to the constant eigenfunction $e_1 = 1/\mu(\mathcal{M})$. Consequently, any function $f \in \mathcal{L}^2(\mathcal{M})$ can be represented as a linear combination of the eigenfunctions $\{e_j\}_{j=1}^{\infty}$ of Δ i.e. $f(x) = \sum_{j=1}^{\infty} a_j e_j(x)$.

The Laplacian operator provides a smoothness measure for functions on $\mathcal{M}$ If $f : \mathcal{M} \mapsto \mathbb{R}$ is a twice differentiable function on $\mathcal{M}$, a natural smoothness measure is

$$S(f) := \int_{\mathcal{M}} \|\nabla f\|^2 d\mu = \int_{\mathcal{M}} f \Delta f d\mu = \langle \Delta f, f \rangle$$

When $S(f)$ is close to 0, we consider f to be smooth. If e_j is a unit eigenfunction of Δ and λ_j is its corresponding eigenvalue, we have $S(e_j) = \langle \Delta e_j, e_j \rangle = \lambda_j$. Thus, smoothness of an eigenfunction is controlled by its eigenvalue. Therefore, for an arbitrary $f = \sum_j a_j e_j \in \mathcal{L}(\mathcal{M}^2)$ we have

$$S(f) = \left\langle \sum_j a_j \nabla e_j, \sum_j a_j e_j \right\rangle = \sum_j \lambda_j a_j^2$$

Hence, approximating a function $f(x) \approx \sum_j^p a_j e_j(x)$ in terms of the first eigenfunctions (corresponding to the p smalles eigenvalues) can control the smoothness of the approximation. This can be thought of as a low-pass filter where the f is approximated by a few smooth eigenfunctions. Following this, the optimal smoothness-preserving approximation can be obtained by minimizing the $\mathcal{L}^2(\mathcal{M})$-norm of the error:

$$a = \underset{a=(a_1,\ldots,a_p)}{\operatorname{argmin}} \int_{\mathcal{M}} \left(f(x) - \sum_{j=1}^{p} a_j e_j(x) \right)^2 d\mu$$

The solution to this minimization problem is given by,

$$a_j = \int_{\mathcal{M}} e_j(x) f(x) d\mu = \langle e_j, f \rangle;\ j = 1, \ldots, p$$

In other words, coefficients a are given by the projection in $\mathcal{L}^2(\mathcal{M})$ onto the span of the first p eigenfunctions of Δ.

However, in practice, we only know the values of f at only $x_1, \ldots, x_m$ sample points. Thus, we resort to solving the discrete version given by (1), where $y_i = f(x_i); i = 1, \ldots, m$.

The graph Laplacian and Laplacian operator Returning to our initial SSL problem, we note that the manifold $\mathcal{M}$ is modeled by a graph G. Therefore, a suitable measure of smoothness for functions defined on graphs is required. The graph-theoretic analogue of the Laplace-Beltrami operator is the graph Laplacian matrix L which can be thought of as an operator on functions defined on graphs. L is also self-adjoint and positive semi-definite and by the spectral theorem any function f on G can be decomposed into a sum of eigenfunctions of L. In terms of smoothness, it is easy to imagine that a f is smooth if it does not change too much between the points in close proximity. This idea is formalized by the smoothness functional $S_G(f) = \sum_{i,j} W_{ij}(f_i - f_j)^2 = fLf^\top$ where $f = (f_1, \ldots, f_n)$ is the function on G. This is analogous to $S(f)$ in the continuous case. It can be further shown that,

$$S_G(f) = fLf^\top = \sum_i^n \lambda_i \langle f, e_i \rangle_G$$

Here, the inner product is the usual Euclidean inner product on the vector space with coordinates indexed by the vertices of G and e_i are normalized eigenvectors of L. Same as before, eigenvectors corresponding to the smallest eigenvalues are more smooth.

2.4 Experimental Results

2.4.1 Real Data

MNIST This dataset contains 28×28 grayscale images for handwritten digits. As a pre-processing step, we normalize each image and compute the first 100 principle components of the set of all images to represent each image as a 100

dimensional feature vector. In order to construct the adjacency matrix, we use 8 nearest neighbors. For each trial, we randomly sample a labeled set and use the rest of the samples as the unlabeled set. If the number of labeled samples is m, we use $p = \min(m/5, 100)$ eigenfunctions. The classification performance is evaluated on the unlabeled set and the errors are averaged over 20 different random seeds. Our results are in agreement with Belkin & Niyogi (2002) and we can conclude that unlabeled data consistently improves the classification accuracy.

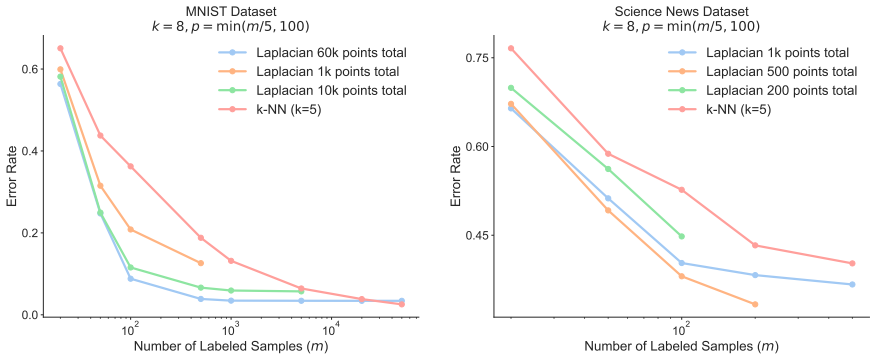


Figure 2: We plot the classification error rate (Y-axis) against the number of labeled samples (X-axis) for MNIST (**left**) and Science News (**right**) datasets for k-NN with 5 nearest neighbors and Laplacian eigenmaps-based SSL across different amounts of total sample sizes.

Science News This is a text classification dataset with 9 different classes. There is a total of 1161 samples with 1153 features each. The pre-processing and experiment trials are performed identically to that of the MNIST datasets. We make the same observation as before, confirming that the unlabeled data can play a role in improving the classification performance.

2.4.2 Algorithmic Analysis

The Laplacian eigenmaps-based SSL algorithm has two main hyperparameters: (1) the number of nearest neighbors k used in the graph construction, and (2) the number of Laplacian eigenfunctions p used in training the linear classifier. We study the effects of these hyperparameters using the MNIST dataset. First, to observe the effect of k, we fix the total number of samples to be $n = 10000$ and set $p = \min(m/5, 100)$ for each labeled sample size m in our experimental trials, and plot the error rates (averaged over 20 replicates) across different values of k (see Fig. 3 middle). We observe that larger values of k lead to a better performance although all $k \geq 4$ results in virtually the same level of error rates. This indicates that when a sufficiently large number of nearest neighbors is considered, the graph can accurately model the underlying manifold of the data.

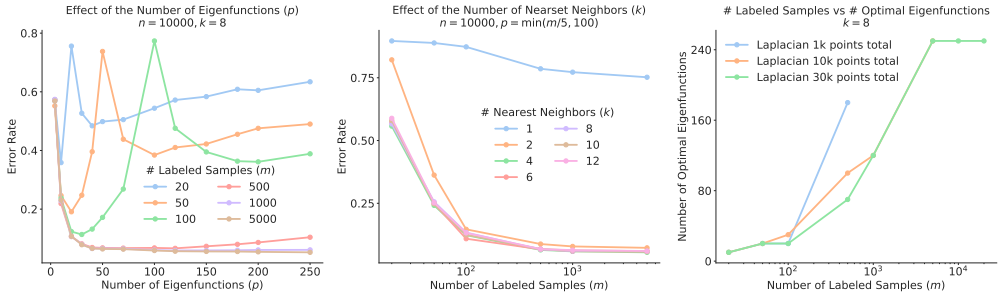


Figure 3: Analysis of the effect of hyperparameters: the number of nearest neighbors k used in the graph construction (**middle**) and the number of Laplacian eigenfunctions p used in training the linear classifier (**left** and **right**).

Next, we study the effect of the number of eigenfunctions p by fixing $k = 8$ and $n = 10000$ and plotting the error (averaged over 20 replicates) against p across different amounts of labeled sample sizes (see Fig. 3 left). This reveals some interesting observations. When we have access to a large amount of labeled samples ($m \geq 1000$), the error rate monotonically decreases and plateaus with p. On the other hand, when m is moderately large ($m = 500$), we notice that the error rate decreases first before increasing beyond a certain threshold of p. This can be explained by the bias-variance tradeoff. Having a small number of eigenfunctions ensures the smoothness condition and reduces the

variance of the classifier by adding some bias. But when p increases the classifier becomes overparameterized leading to a lower bias but a higher variance. While this does not manifest at large labeled sample sizes m, it does so when m drops. Moreover, when m is very small ($m \leq 100$), we observe a double-descent like phenomenon (Belkin et al., 2019) that is generally observed in overparameterized machine learning models. The error drops initially before increasing up to some level of p and then undergoes another descent as p increases.

Furthermore, we find the optimal number of eigenfunctions p^* to use at each labeled sample size m via a simple grid-search and plot it in Fig. 3 (right). There, we notice that p^* scales with m almost linearly. However, as indicated by Fig. 3 (left), it is better not to use large values of p as it would adervsly affect the smoothness condition on which this method depends.

Finally, we empirically analyze the time-complexity of the 2 most critical components of the algorithm: the nearest neighbor graph construction and the Laplacian eigendecomposition. To this end, we plot the computational time of each component against the number of total samples n (see Fig. 4). It can be observed that the computational time roughly scales quadratically with n i.e. the algorithm has a time-complexity of $O(n)$.

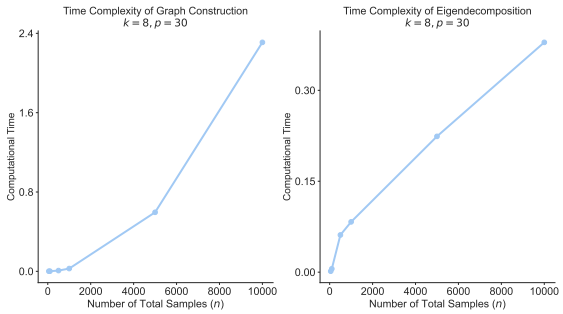


Figure 4: Time complexity of nearest neighbor graph construction (**left**) and Laplacian eigendecomposition (**right**) as the number of total samples n scales up.

2.5 Limitations

The Laplacian eigenmaps-based SSL algorithm assumes that the data lie on a smooth manifold $\mathcal{M}$ and that the classification function f defined on this manifold is also smooth. When these smoothness conditions are not met by $\mathcal{M}$ and/or f, the performance of the algorithm will be sub-optimal. In addition to this, it is susceptible to the following limitations:

Non-uniform distributions on the manifold The algorithm implicitly assumes a uniform probability distribution exists over the manifold from which the samples are drawn. If the sampling was non-uniform, the nearest neighbor graph constructed using the samples would not faithfully represent the manifold geometry.

Inconsistent labeling or label noise The Laplacian eigenmaps-based SSL algorithm assumes that the labeling function of the data is smooth, consistent and reliable. This assumption would not hold true under the presence of label noise and similar inconsistencies. Thus, the algorithm would fail to leverage the available labeled data leading to poor performance.

Challenging real datasets When the feature dimension d is much larger than the number of samples n i.e. $n << d$, the algorithm would not work well as it would fail to capture the underlying geometry of the data. This is why we perform a PCA-based dimensionality reduction for our experiments with MNIST and Science News. The algorithm also requires a sufficient amount of labeled data to successfully learn a classifier based on the manifold information. If the number of labeled samples is very low, it would adversely affect the classification accuracy. Furthermore, since it requires the computation of the Laplacian matrix and its eigenvectors, the method would not be practical for extremely large datasets. Finally, even though manifold structures present in natural datasets is of great interest, in reality, such structures may not inform a lot about the downstream task at hand.

2.6 Theoretical Error Analysis

It is of interest to study the finite sample error bounds and guarantees for the Laplacian Eigenmaps-based SSL algorithm. Zhou & Srebro (2011) provides the following error bound for the algorithm in the limit of infinite unlabeled data ($n \to \infty$),

$$\mathbb{E}_{x\sim\mu}\left(f(x) - \hat{f}(x)\right)^2 \leq \frac{S(f)}{\lambda_{p+1}} + \frac{Cp\log m}{m},$$

where the notations have the usual meaning as before. Based on this error bound, it can be shown that given m labeled samples residing in an intrinsic l dimension, the optimal number of eigenfunction to be used is $p^* = O\left((m/\log m)^{\frac{l}{l+2}}\right)$ which would yield an optimal error of $O\left((m/\log m)^{-\frac{2}{l+2}}\right)$.

3 Poisson Learning

3.1 Description

Another semi-supervised learning technique based on the Graph Laplacian is Poisson Learning, which was introduced in Calder et al. (2020). In Poisson learning, the objective is to solve the following Poisson equation:

$$\mathcal{L}u(x_i) = \sum_{j=1}^{m}(y_j - \bar{y})\delta_{ij} \text{ for } i = 1, ..., n \tag{2}$$

satisfying $\sum_{i=1}^{n} d_i u(x_i) = 0$, where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise, and $\bar{y} = \frac{1}{m}\sum_{j=1}^{m} y_j$ is the average label vector. This is a graph Poisson equation, with the labeled points corresponding to "sources" in the equation to be solved. The author in Calder et al. (2020) posit that this Poisson equation approach is more effective in SSL applications where the rate of labeled samples is very low.

3.1.1 Random Walk Interpretation

The random walk interpretation of Poisson learning is as follows: suppose at time 0 the labeled vertices are released to walk around the graph, defining $x \in X$ and let $X_0^x, X_1^x, ...$ be a random walk on X starting at $X_0^x = x$ with transition probabilities

$$\mathbb{P}(X_k^x = x_j | X_{k-1}^x = x_i) = d_i^{-1} w_{ij} \tag{3}$$

Poisson learning can be interpreted as a random walk in which random walkers are released from the labeled vertices, carrying their label information with them, and their label information then being recorded at each node they visit. Indeed, we can define

$$w_T(x_i) = \mathbb{E}\left[\sum_{k=0}^{T}\sum_{j=1}^{m} y_j \mathbb{1}_{\{X_k^{x_j} = x_i\}}\right] \tag{4}$$

This equation would take the random walk described above, and sum the labels of all the labeled visitors to x_i. The problem is for large T, this would converge to the average of the labels of the labeled vertices. In other words, without further modifications, this approach will result in only a blind average of the labels.

What we are interested in is only the short-time behavior of the random walk. Thus, we subtract off the tail of the distribution (i.e. the average of the labels, $\bar{y}$), and we further normalize by the degree of the vertex, d_i, yielding the following expression:

$$u_T(x_i) = \mathbb{E}\left[\sum_{k=0}^{T}\frac{1}{d_i}\sum_{j=1}^{m}(y_j - \bar{y})\mathbb{1}_{\{X_k^{x_j} = x_i\}}\right] \tag{5}$$

As subsequently demonstrated in Calder et al. (2020), the above quantity converges to the solution of the Poisson equation posed in equation (6) above as $T \to \infty$. This function can be computed iteratively using (per Theorem 2.1 in the paper)

$$u_{T+1}(x_i) = u_T(x_i) + d_i^{-1}\left(\sum_{j=1}^{m}(y_j - \bar{y})\delta_{ij} - \mathcal{L}u_T(x_i)\right) \tag{6}$$

Per Theorem 2.1, the above iterative process yields the unique solution to equation (6) satisfying $\sum_{i=1}^{n} d_i u(x_i) = 0$.

3.1.2 Variational Interpretation

Solving equation (6) can also be viewed as a variational problem. In particular, equation (6) can be solved by minimizing the following variational problem:

$$\min_{u \in \ell_0^2(X)}\left\{\sum_{i,j=1}^{n} w_{ij}|u(x_i) - u(x_j)|^2 - \sum_{j=1}^{m}(y_j - \bar{y}) \cdot u(x_j)\right\} \tag{7}$$

where $\ell_0^2(X)$ refers to the space of mean-zero functions. Per Theorem 2.3 in Calder et al. (2020), the solution to this variational problem is also the solution to equation (6) above.

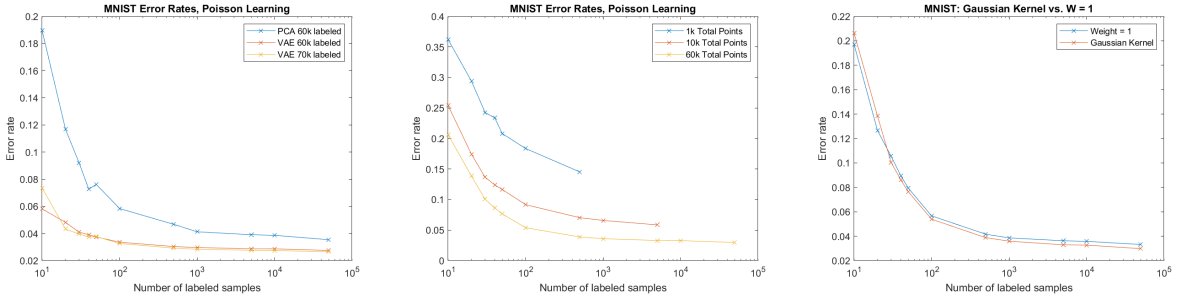


Figure 5: Performance of Poisson Learning on MNIST dataset, varying method of feature extraction (**left**), or number of total samples (**center**), or weight kernel (**right**).

As noted in the paper, this problem differs from some other versions of SSL in that the labeled examples are not imposed as hard constraints on the problem, but rather as an added loss in the expression to be minimized. While, as the authors note, theirs is not the first algorithm to use soft constraints in this kind of problem, they attribute the success of their algorithm to the affine nature of this soft loss term.

3.2 Algorithmic Implementation

As foreshadowed by equation (10) above, Poisson learning can be implemented numerically through an iterative process. The function u is initialized to be a matrix of zeroes, and the Graph Laplacian is computed as described in the earlier sections as $L = D - W$, where W is the symmetric matrix of weights between nodes of the graph, and D is the diagonal matrix such that $D_{ii} = \sum_j W_{ji}$. The weight matrix can use any appropriate heat kernel; in this report, the following Gaussian kernel was used:

$$w_{ij} = \exp(-4|x_i - x_j|^2 / d_K(x_i)^2) \tag{8}$$

where $d_K(x_i)$ is defined to be the distance between x_i and its $k-th$ nearest neighbor. All experiments below used $k = 10$ nearest neighbors in constructing the weight matrix.

The provided labels are encoded as one-hot vectors, from which $\bar{y}$ is subtracted. A $k \times n$ matrix $\mathbf{B}$ is then constructed, such that the first m columns are the one-hot vectors (less $\bar{y}$), and the remaining $n - m$ columns of $\mathbf{B}$ are zeroes.

Next, the following operation is iterated until convergence is reached:

$$\mathbf{U} = \mathbf{U} + \mathbf{D}^{-1}(\mathbf{B}^T - \mathbf{L}\mathbf{U}) \tag{9}$$

After this algorithm converges (100-500 cycles), the unlabeled samples are assigned a class as follows:

$$\ell(x_i) = \arg \max_{j \in \{1,\ldots,k\}} \{u_j(x)\} \tag{10}$$

3.3 Experimental Results

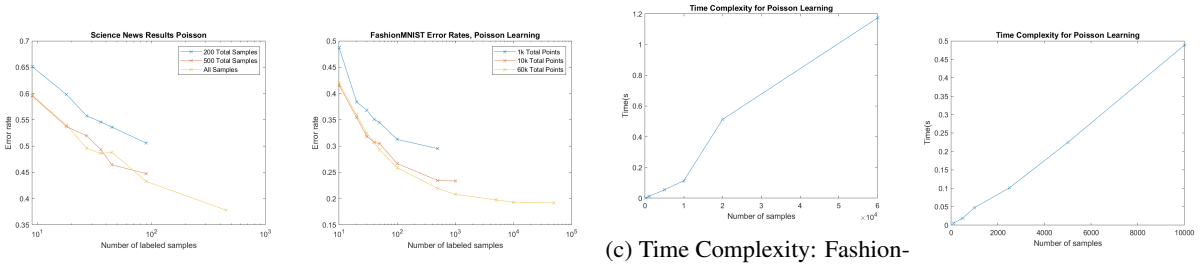
This algorithm, like the others described herein, was tested on the MNIST dataset and the Science News dataset. Additional testing used the FashionMNIST dataset. For all reported results, the algorithm was constructed to ensure an equal number of labels for each class, similar to the method in Calder et al. (2020). Thus, for each trial with m labeled samples, there were $m/|\ell|$ labeled samples per class, where $|\ell|$ is the number of classes.

3.3.1 MNIST Results

In the Calder et al. (2020) paper, the authors first used a variational autoencoder (VAE) to extract meaningful features from the dataset, then ran Poisson learning on the resulting feature vectors. In this report, we tested Poisson learning on this VAE dataset, then also tested it using the first 100 components of a PCA decomposition of the original image vectors. Each experiment was run 20 times and the resulting error rate was averaged to obtain the plot below.

The results of this experiment, showing error rate as a function of the number of labeled samples, are shown in Fig. 5, left panel.

The VAE dataset shows much better performance than using the first 100 PCA components, indicating that some of the success of the Poisson learning algorithm in the Calder et al. (2020) paper can evidently be attributed to the success of



(a) Results on Science News (b) Results on Fashion MNIST (c) Time Complexity: Fashion-MNIST (d) Time Complexity: MNIST

Figure 6: **(a)&(b)**: Results for Poisson Learning on Science News and Fashion MNIST; **(c)&(d)**: Time Complexity for Poisson Learning

their autoencoder at extracting meaningful features. Still, the 0.19 error rate with only 10 labeled samples is much better than the other algorithms tested herein.

Repeating this experiment to vary the number of total samples (rather than varying the method of feature extraction) gives the results in Fig. 5, center panel. Here, we find that the algorithm does not perform as well with fewer samples, failing to outperform the algorithm discussed in the previous section.

3.3.2 Science News/FashionMNIST

The Science News dataset consists of 1,161 articles that have been encoded as vectors, where the entries in each vector correspond to the number of times a certain word appears in the article. To test Poisson learning on this algorithm, we first used PCA to extract the first 100 principal components of the data, then ran the algorithm 20 times each for several different numbers of labeled examples. The results are shown in Fig. 6a.

Interestingly, here the Poisson algorithm does not show an advantage over the other algorithms tested herein. Thus, this casts some doubt on whether, in general, the Poisson learning algorithm proposed in Calder et al. (2020) represents an improvement over existing graph-based SSL algorithms *in all cases*. Perhaps the algorithm works best at both low label rates and high numbers of total samples, as it has empirically shown its greatest edge (in this paper) in the case of the MNIST dataset when a large number of samples but small number of labels are available.

Testing the algorithm on one more dataset, the popular benchmark FashionMNIST dataset, we obtain the results in Fig. 6b. Here, the algorithm performs reasonably well given the increased complexity of the FashionMNIST problem, achieving less than 30% error with just 50 labeled samples, or 5 labels per class.

3.3.3 Weight Kernel

Finally, we tested whether varying the weight kernel would affect the results above. As indicated earlier, the Calder et al. (2020) paper used a Gaussian kernel to construct the weight matrix, while in the prior algorithm from Belkin & Niyogi (2002), the authors used $W_{ij} = 1$ if i is a nearest neighbor of j (or vice versa). We tested this latter approach on our MNIST data with Poisson learning to see if it affected performance. The results are shown in Fig. 5, right panel. The results show that, at least on this dataset, varying the kernel for assembling the weight matrix makes very little difference to the results, although the Gaussian kernel performs slightly better when the relative number of labels is large.

3.3.4 Time Complexity

In order to compare it to the other algorithms above, we also considered the time complexity of running the Poisson learning algorithm. Similar to the results above, we found that the time required to run the algorithm was linear in the total number of samples under consideration. See Fig. 6c & Fig. 6d.

4 Regularization on Graph with Function-Adapted Kernels - Eigenfunction-based

4.1 Motivation and Overview

Szlam et al. (2008) extend the graph-based SSL methods by introducing function-adapted kernels. The idea is that the true function may be not smooth with respect to the original geometry of data, but the geometry can be modified so

that the function is as smooth as possible in the modified geometry. To achieve that, Szlam et al. (2008) propose to incorporate the structure of the function into the construction of weighted graph, by adding a function-related term in local similarity. Szlam et al. (2008) explored both Laplacian Eigenmap-based SSL and diffusion-based SSL. We focus on the former in this section, and study the diffusion-based method in Section 5.

4.2 Algorithmic Description

As decribed earlier, we assume the data lie on a low-dimensional manifold and exploit the manifold structure to perform function approximation and smoothing. For SSL, the function of interest is the the mapping from each data point to its label.

Construction of Weighted Graph To model the manifold structure of the data, we first construct a graph G from the dataset X that contains both labeled and unlabeled points. Naturally, the vertices of G are the data points and the edge weights measure the similarity between data points. To construct the graph, we first find the k-nearest neighbors for each data point using Euclidean distance. For each nearest neighbor, the local similarity between the two data points x and y is computed with a Gaussian heat kernel:

$$W_\sigma(x,y) = h\left(\frac{\rho(x,y)^2}{\sigma}\right), \tag{11}$$

where ρ is a distance metric (Euclidean in this study), $\sigma > 0$ is the "local time" parameter, and h is the Gaussian kernel: $h(a) = exp(-a)$. It is important to constrain the similarity to be *local* by defining it only for the k-nearest neighbors, as local similarities are more reliable than long-range similarities for data sampled from a manifold. The similarities between distant points can be inferred from the weighted graph by, e.g., random walk, which resembles the geodesic distance on manifold.

Theoretical Considerations: Under mild assumptions, the operator $(I - D_\sigma^{-\frac{1}{2}} W_\sigma D_\sigma^{-\frac{1}{2}})/\sigma$ approaches the Laplacian on the manifold asymptotically, where D_σ is a diagonal matrix with diagonal elements equal to the row sums of the weight matrix W_σ. This supports the idea of approximating the manifold structure with a weighted graph, and relates the weights to heat equation on the manifold.

To constrain the similarity to be *local*, the weight matrix W_σ is truncated by keeping only the k nearest neighbors to be nonzero. This will effectively reduce the nonzero entries of the weight matrix from n^2 to kn, where n is the total number of points, which is desirable for fast implementation using sparse matrix. Although this truncation makes W not symmetric, we take the arithmetic average of W and W^T to recover symmetry. The symmetrization will likely increase the number of nonzero entries but its order is still $O(kn)$.

Self-tuning Weight Matrix We use the self-tuning approach proposed in Zelnik-Manor & Perona (2004) to normalize each row of the weight matrix. Specifically, the distances at each point are scaled so that the j-th nearest neighbor has distance 1. Let $\rho_x(z,z') = \rho(z,z')/\rho(x,x_j)$, where x_j is the j-th nearest neighbor to x. The self-tuning weight matrix is defined by

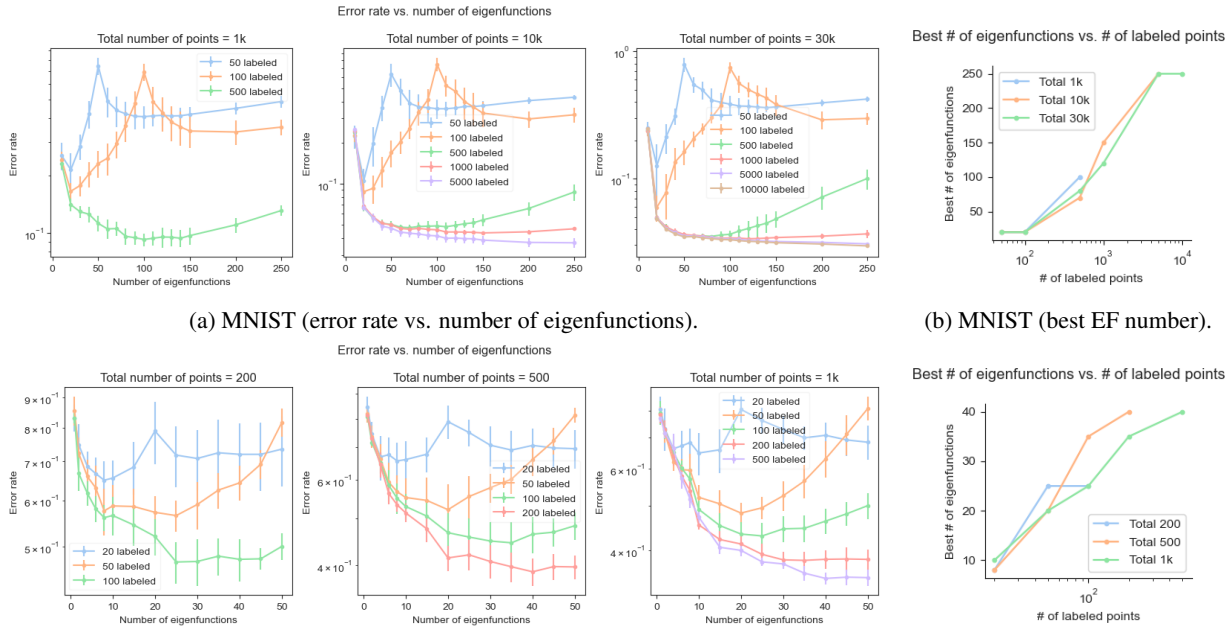
$$W_\sigma(x,y) = h\left(\frac{\rho_x(x,y)\rho_y(x,y)}{\sigma}\right). \tag{12}$$

Here, the geometric mean of ρ_x and ρ_y is used to enforce symmetry. Self-tuning can be useful in normalizing and stabilizing the weight matrix when there are points that are very close to, or far from, all of its nearest neighbors. In the former case, the local similarities will be all close to 1; while in the latter, all close to 0, resulting in an unbalanced weight matrix. Self-tuning will bring the weights at different points to approximately the same level, which makes the weight matrix more balanced and the similarity measure location-dependent.

Function-Adapted Kernels To modify the geometry of data to make the function as smooth as possible, function-adapted kernels are proposed in Szlam et al. (2008). Let $\tilde{f}$ be an estimate for the true function to be approximated. We set

$$W^f(x,y) = \exp\left(-\frac{\|x-y\|^2}{\sigma_1} - \frac{\|\tilde{f}(x)-\tilde{f}(y)\|^2}{\sigma_2}\right) \tag{13}$$

to be the function-adapted weight matrix. Here, $\tilde{f}$ can be a rough estimation of the labels at all data points achieved by some basic SSL classifiers. Following Szlam et al. (2008), we use the predictions from the harmonic classifier at steady state: $\tilde{f} = \frac{g_i^N(x)}{\sum_i |g_i^N(x)|}$, where $g_i^N(x)$ is the (continuous) function value given by the harmonic classifier at point x after N steps of diffusion for class i (in a multi-class classification scenario). N is set as 250 to reach steady state.



(a) MNIST (error rate vs. number of eigenfunctions).

(b) MNIST (best EF number).

(c) Science News.

(d) Science News.

Figure 7: Analysis of the effect of number of eigenfunctions. (a) and (c): error rate at increasing number of eigenfunctions. (b) and (d): best number of eigenfunctions for different labeled data sizes and total data sizes.

Least Squares Regression in Laplacian Eigenfunction Embedding Similar to earlier sections, we use the eigenfunctions of the Laplacian $\mathcal{L}$ to perform smooth function approximation on graphs. We solve the least squares problem that is identical to that defined in Eq. (1).

Notes on Theoretical Foundations. Based on harmonic analysis on graph, the eigenfunctions of the Laplacian (ϕ_i's) form an orthogonal basis for the functions on the graph, and their eigenvalues measure the frequency (oscillation) of ϕ_i. This result resonates beautifully with the Fourier analysis on functions on Euclidean space. Hence, *projecting a function on graph onto its first few terms of its expansion in the eigenfunctions of $\mathcal{L}$ is effectively a smoothing operation* Szlam et al. (2008). Therefore, fitting a partially-observed function on graph by the first few eigenfunctions is effectively a smooth approximation of the function (in the case of SSL, a smooth propagation of the known labels to unlabeled points).

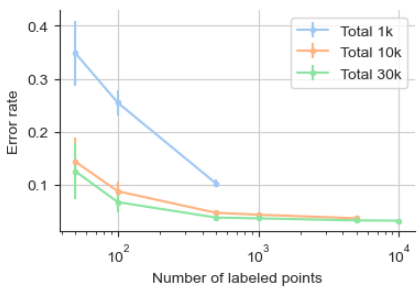
4.3 Experimental Results

4.3.1 Results of Non-adapted Kernel

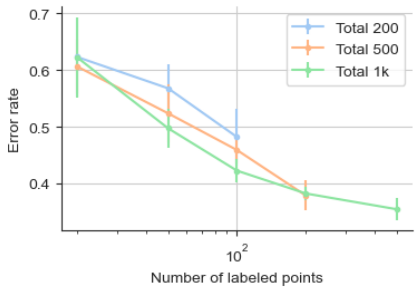
First, we present results of eigenfunction-based SSl with self-tuning weight matrix and non-adpated kernels on the MNIST dataset. Following Szlam et al. (2008), we smooth the images by convolving two times with a 3×3 averaging filter, and then reduce to 50 dimensions by principal component analysis (PCA). We set the number of nearest neighbors $k = 9$, the self-tuning neighbor index $j = 4$ and local time $\sigma = 1$.

Influence of Number of Eigenfunctions First, we study the influence of the number of eigenfunctions on the error rate, with results shown in Fig. 7a. We tested with different numbers of labeled points and total points. For each case, we run the algorithm for 10 repetitions with different labeled/unlabeled splits and report the average and standard deviation.

As shown in Fig. 7a, when the number of eigenfunctions is less than number of labeled points, the error rate first decreases then increases with the number of eigenfunctions. This is because too few eigenfunctions have limited representation power while two many eigenfunctions leads to overfitting, both unfavorable for classification. From the perspective of function smoothing on graph, too few eigenfunctions means the basis is too small and the maximum frequency is too low, leading to over-smoothing. On the other hand, using too many eigenfunctions means allowing too many high-frequency components into the estimation, leading to overfitting. Interestingly, as the number of eigenfunctions surpasses the number of labeled points, the curves starts decreasing again, showing a "double descent"



(a) MNIST.



(b) Science News.

Figure 8: The classification error rate of eigenfunction-based SSL using self-tuning weight matrix with non-adapted kernel on two datasets at different numbers of labeled points and total points.

like phenomenon. This is due to the change in the behavior of least squares solver (numpy.linalg.lstsq herein). When the number of eigenctions is less than the number of labeled points, the least squares system is overdetermined; otherwise, it is underdetermined. In the former case, the solver returns the unique solution, while in the latter case, the solution is not unique and the one with the least $\ell 2$ norm is returned. Therefore, the least squares solver operates in two different regimes, resulting in the "double descent" like phenomenon in the curves.

Best Number of Eigenfunctions vs. Data Size Next, we study how the best number of eigenfunctions (EF) changes with the number of labeled points. As shown in Fig. 7b, the best number of EF scales up with the number of labeled points and hardly changes with the number of total points. This is expected as the the dimension of least squares problem only depends on the labeled points, not the total data size. It is clear that an appropriate number of eigenfunctions is needed to maximize the performance for each number of labeled points.

Error rate vs. Number of Labeled Points Finally, we study how the classification performance changes with the number of labeled points, using the best number of eigenfunctions selected from the previous experiment. The results are obtained on a heldout dataset of 30k images that is not used in parameter selection. As illustrated in Fig. 8a, the error rate decreases with more total points, demonstrating the advantage of estimating the manifold using unlabeled data.

Science News results The corresponding results for Science News data are shown in Fig. 7c, Fig. 7d and Fig. 8b, suggesting similar observations.

4.3.2 Results of Function-Adapted Kernel

We further implemented the function-adapted (FA) kernel and compare it with the non-adapted kernel, with results shown in Fig. 9. Interestingly, we found a **failure case** of FA when the number of labeled points is extremely small (=50), where it performs worse than the non-adapted baseline. This is due to the large error in the initial estimate of the labels $\tilde{f}$ given a small number of labeled points, which cannot provide good information about the desired geometry of the data. With more labeled data (number of labeled ≥ 100), FA can outperform the non-adapted counterpart, and the improvement increases as number of labeled points increases. However, the improvement is not as significant as observed in Szlam et al. (2008), possibly due to insufficient parameter tuning in our experiment - we directly used the previously selected best number of eigenfunctions for non-adapted kernels and fixed the multiplication factor $\beta = 1$.

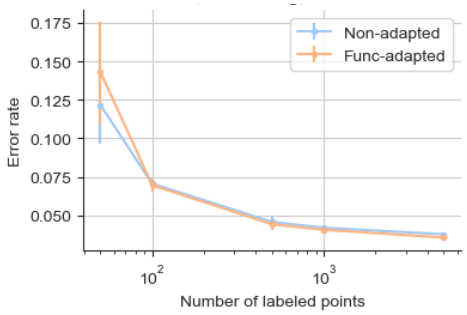


Figure 9: Function-adapted vs. non-adapted kernel with eigenfunction-based SSL and 10k total points on MNIST data.

4.3.3 Time Analysis

Finally, we empirically evaluate the time complexity of the non-adapted method at different amounts of total data. We fix the number of eigenfunctions at 20 and number of labeled points at 100. The results in Fig. 10 confirm again the linear relationship between the run time of the graph construction step (kNN + weight matrix computation), as well as

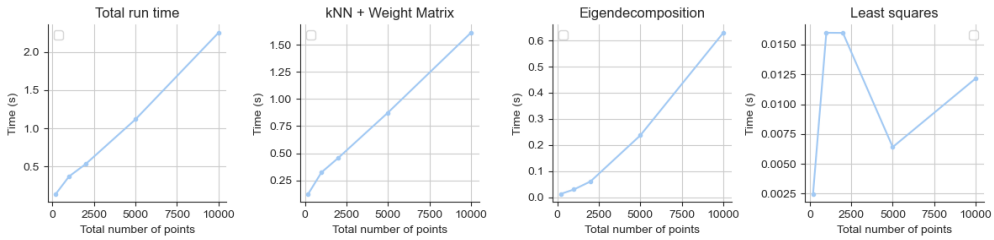


Figure 10: Run time of different components of the non-adapted method.

the eigendecomposition step, to the number of total points. Note that the the run time of least squares regression does not scale with number of total points, as the regression is performed on labeled points only. Since the graph construction step dominates the run time, the total run time also scales linearly with the amount of total data.

5 Diffusion-based Regularization on Graph with Function-Adapted Kernels

5.1 Algorithmic Description and Construction of Weighted Graph

We assume that the data lie on a low-dimensional manifold that can be approximated by a weighted graph, and that the class labels at nodes constitute a (relatively) smooth function f. Under these assumptions, mollifiers (smoothing functions) can be used to estimate the unknown class labels by the propagating the known labels with respect to the underlying geometry of the data. A simple mollifier that has a natural analogy on graphs is the heat diffusion equation – multiplication by the (normalized) weight matrix K constitutes one step in a heat equation. The same mollifier can also be interpreted as a random walk on the graph, with the weight matrix K constraining the walk to only step between similar points. Here, we take K to be the self-tuning weight matrix defined in the previous section, with $h(a) = \exp(-a)$. We also study function-adapted versions of these mollifiers as described in the previous section.

5.1.1 Classifiers

We analyze two very similar classifiers and their function-adapted versions. Both classifiers start with an initial condition f_0 specified by the known labels f and perform N iterations of the heat equation $f_{t+1} = Kf_t$ to iteratively update the estimated class labels through diffusion. The first classifier (referred to as kernel smoothing in Szlam et al. (2008)) introduces the known labels only once at the beginning of the heat equation, while the second classifier (referred to as the harmonic classifier) reintroduces the known labels at every iteration of the heat equation.

5.1.2 Function-Adapted Kernel

The construction of the function-adapted kernel is the same as the previous section. We calculate the predictions from the harmonic classifier at steady state

$$\tilde{f} = \frac{g_i^N(x)}{\sum_i |g_i^N(x)|}, \tag{14}$$

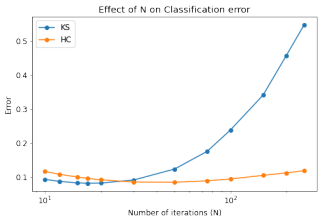
(where $g_i^N(x)$ is the function value at node x after 250 iterations of diffusion), and incorporate these as additional features when constructing the nearest neighbors graph for the final classifier. This has the effect of promoting diffusion within estimated class boundaries and discouraging diffusion across them. An additional parameter β determines the ratio of the average norm of the class estimates to the average norm of the original PCA features.

5.2 Experimental Results

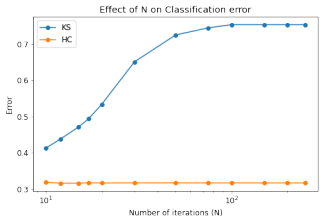
5.2.1 Results of Non-adapted Methods

For the MNIST dataset, we perform the same image pre-processing as in the previous section (two convolutions with a 3x3 filter, 50-dimension PCA). For the Science News dataset, we perform only 50-dimension PCA.

Model parameters were chosen separately for each dataset by averaging classifier performance on training data (50% of total) over 10 different labeled/unlabeled splits. As suggested in Szlam et al. (2008), the number of diffusion iterations (N) was chosen from numbers between 1 and 250, and the number of neighbors in the graph (k) and unit-distance neighbor (j) were chosen from $\{9, 4\}, \{13, 9\}, \{15, 9\}$, or $\{21, 15\}$.



(a) MNIST.



(b) Science News.

Figure 11: The effect of number of diffusion iterations N on the classification error rates of kernel smoothing and harmonic classifiers.

We report model performance on the held-out test data (50% of total), using three numbers of total points: 1k, 10k, and 30k for MNIST, and 200, 500, and 1k for Science News.

Effect of Number of Diffusion Iterations We make the simplifying assumption that there is no significant interaction between N and other parameters, so we only examine the effect of N for models with 30k total points, 100 of which are labeled, where each graph node has 9 neighbors and the 4^{th} neighbor is at unit distance.

For MNIST, we found that $N = 17$ was optimal for kernel smoothing (KS), and $N = 30$ iterations was optimal for the harmonic classifier (HC). As shown in Fig. 11a-11b, kernel smoothing error is lowest for smaller values of N and increases somewhat linearly (x-axis is log-scaled), while harmonic classifier error is quite consistent within each dataset, regardless of N. This is likely because performing many iterations of diffusion oversmooths the label estimates for kernel smoothing, but the harmonic classifier avoids this issue by re-introducing known labels at every iteration of diffusion.

We performed a similar analysis (not shown) to determine that the optimal values of k, j for all combinations of classifier and dataset were $\{9, 4\}$. However, we note that classifier performance was fairly similar for all pairs of k, j that we considered.

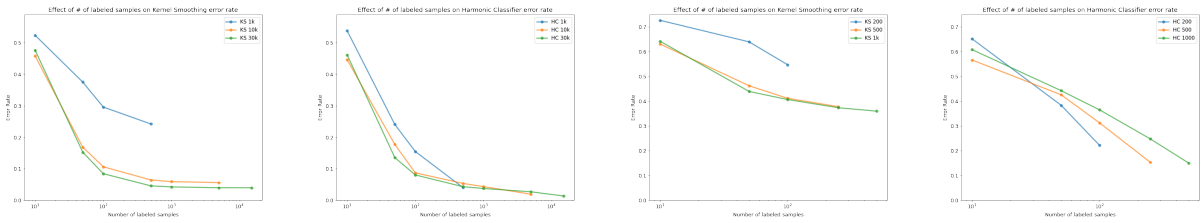
Error rate vs. Number of Labeled and Total Points For each number of total points, we study how the classification performance (on held-out data) changes with the number of labeled points, using the optimal model parameters found for each combination of classifier and dataset. Figs. 12a-12d show our results. For both datasets, the harmonic classifier consistently outperforms the kernel smoothing method by a small amount. Importantly, both kernel smoothing and harmonic classifier have modest performance gains relative to all the eigenfunction classifiers discussed in the previous sections. Additionally, both diffusion classifiers perform much worse on the Science News dataset than MNIST. This contradicts supervised learning intuition since the Science News dataset has less total samples and thus a higher labeled/unlabeled ratio. However, we see that the 1k line in the MNIST figures has similar error to the 1k line in the Science News dataset, so the poor performance on Science News may simply be due to undersampling; 1k points may not be enough to sufficiently approximate the manifold structure for a 9-class problem in 50 dimensions.

Another general pattern is that classification error decreases as the number of total points increases. Since our methods assume that the data lie on a low-dimensional manifold, using more points to construct the graph may improve the graph's approximation of the manifold structure, in turn improving model performance. The results of the harmonic classifier on the Science News dataset (Fig. 12d) defies this trend, but the error in this case is relatively high, so the effects of number of total points may have been masked by other sources of error which decrease as the ratio of labeled to unlabeled samples increases. Finally, as expected, there is a consistent inverse relationship between error and number of labeled samples across all combinations of classifier and dataset.

5.2.2 Results of Function-Adapted Kernel

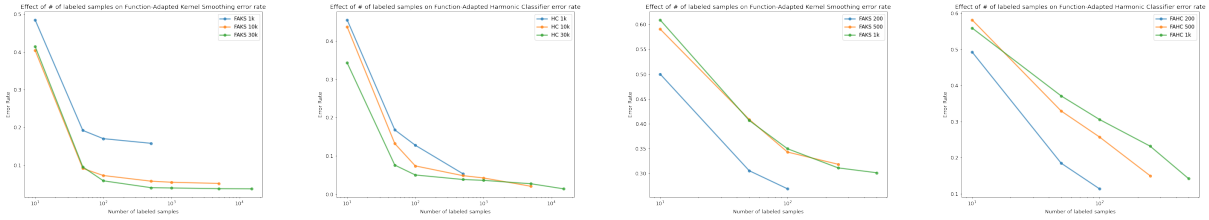
We performed an identical analysis using function-adapted versions of kernel smoothing (FAKS) and harmonic classifier (FAHC). Model parameters were chosen in the same way, with N, k, j chosen from the same set of candidate values as before, and β chosen from $\{1, 2, 4, 8\}$.

Effect of Parameters We first studied the effect of N on the function-adapted classifiers. For MNIST, we found that $N = 30$ was optimal for FAKS and $N = 75$ was optimal for FAHC. The reason for the increase in optimal N compared to the non-function adapted version is not clear, but it could stem from differences in the connectedness of the graph due to the addition of extra features.



(a) MNIST KS. (b) MNIST HC. (c) Science News KS. (d) Science News HC.

Figure 12: The classification error rates of non-adapted KS and HC as number of labeled and total samples are varied.



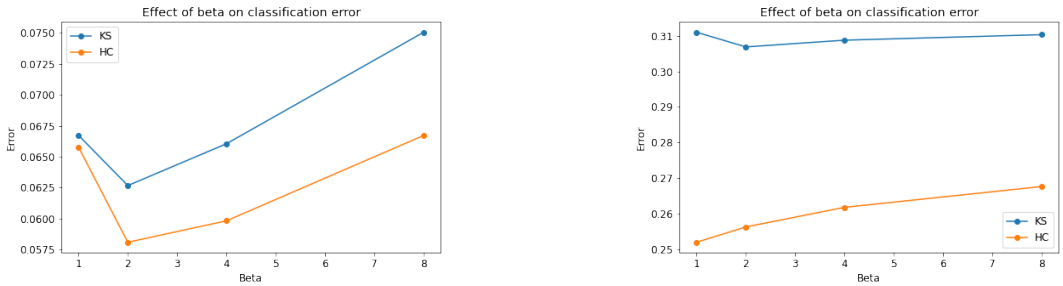
(a) MNIST FAKS. (b) MNIST FAHC. (c) Science News FAKS. (d) Science News FAHC.

Figure 13: The classification error rates of function-adapted KS and HC as number of labeled and total samples are varied.

We performed a similar analysis to determine the optimal β, which controls the relative scale of the original 50 PCA features and the extra class estimate features. For FAKS, $\beta = 2$ was optimal for both datasets. For FAHC, $\beta = 2$ was optimal for MNIST and $\beta = 1$ was optimal for Science News. These relatively low values of β imply that the PCA features and class estimate features are of approximately equal importance. However, we note that the difference in performance between different β values is quite small.

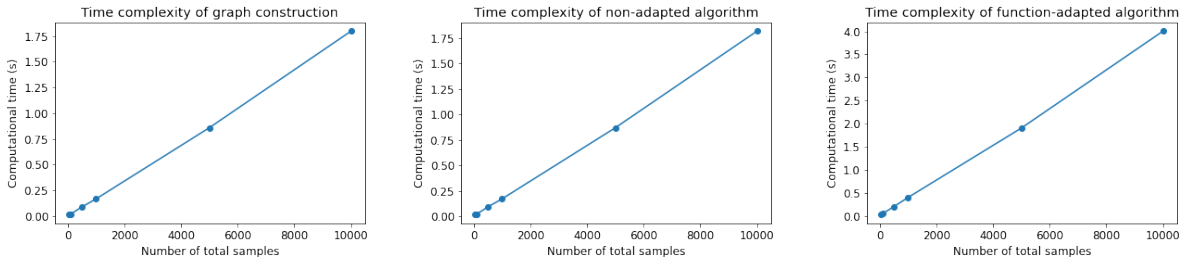
Once again, the optimal k, j for all combinations of classifier and dataset were $\{9, 4\}$, but classifier performance was fairly similar for all pairs of k, j that we considered.

Error vs. Number of Labeled Points Similarly to the non-adapted classifiers, we analyze how the classification performance changes with the number of labeled points, shown in Figs. 13a-13b. Notably, the function-adapted classifiers outperform their non-adapted counterparts at most of the combinations of labeled and total samples. Most other trends from the non-adapted analysis hold for the function-adapted versions: FAHC outperforms FAKS, both classifiers perform poorly on Science News compared to MNIST, and classification error and number of labeled samples are inversely related.



(a) Function-adapted MNIST. (b) Function-adapted Science News.

Figure 14: The effect of β on the classification error rates of function-adapted kernel smoothing and harmonic classifiers.



(a) Graph construction. (b) Non-adapted runtime. (c) Function-adapted runtime.

Figure 15: Runtime complexity for diffusion-based SSL.

5.2.3 Time Complexity

Figs. 15a-15c below show the empirical time complexity of the nearest-neighbor graph construction, non-adapted algorithms, and function-adapted algorithms as the total number of points changes. All three graphs are remarkably linear. This is consistent with the analysis in Szlam et al. (2008), which states that the algorithm runtime should be approximately linear if the nearest-neighbor search is relaxed to an approximate version.

The runtime of the graph construction and entire non-adapted algorithm are almost identical, which means that graph construction dominates the runtime of the non-adapted algorithm. The function-adapted algorithm is more than twice the runtime of the non-adapted version. This is because the function-adapted algorithm involves constructing two graphs: one to calculate steady-state class estimates, and another that incorporates the estimates to produce the final classifier. The remainder of the discrepancy in runtime is likely from the additional matrix multiplications needed to calculate steady-state class estimates – this requires 250 diffusion iterations, while producing a non-adapted classifier requires only $10 - 30$ iterations depending on the algorithm.

6 Discussion

As demonstrated above, graph-based SSL methods provide a computationally inexpensive and powerful method for learning in environments were labeled data is expensive but unlabeled data is readily available. These methods exploit the underlying geometry of the space of labeled and unlabeled data to make predictions with relatively high degrees of accuracy even where few labels are supplied. Still, there is a high degree of variability of performance depending upon the dataset considered. Performance was excellent on MNIST, but much worse on the Science News dataset, which is smaller and has more underlying features. Thus, additional work is needed to optimize graph-based methods for these more challenging SSL problems.

7 Group Member Contributions

- Ashwin De Silva was primarily responsible for the work described in Section 2 of the paper, regarding Laplacian Eigenmaps-based SSL.
- Jeremy Welland was primarily responsible for the work described in Section 3 of the paper, regarding Poisson Learning.
- Zhenghan Fang was primarily responsible for the work described in Section 4 of the paper, regarding Eigenfunction-based Regularization on Graph with Function-Adapted Kernels.
- Sai Koukuntla was primarily responsible for the work described in Section 5 of the paper, regarding Diffusion-based Regularization on Graph with Function Adapted Kernels.
- All group members collaborated on the selection and direction of the project, identifying the techniques and algorithms to be used, and the assembly of the paper and presentation.

References

Mikhail Belkin and Partha Niyogi. Using manifold stucture for partially labeled classification. *Advances in neural information processing systems*, 15, 2002.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Jeff Calder, Brendan Cook, Matthew Thorpe, and Dejan Slepcev. Poisson learning: Graph based semi-supervised learning at very low label rates, 2020.

Arthur D Szlam, Mauro Maggioni, and Ronald R Coifman. Regularization on graphs with function-adapted diffusion processes. *Journal of Machine Learning Research*, 9(8), 2008.

Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. *Advances in neural information processing systems*, 17, 2004.

Xueyuan Zhou and Nathan Srebro. Error analysis of laplacian eigenmaps for semi-supervised learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 901–908. JMLR Workshop and Conference Proceedings, 2011.