Poisson learning: Graph-based semi-supervised learning at very low label rates

Jeff Calder

School of Mathematics
University of Minnesota

PDEs and Graph-Based Learning
Summer School on Random Structures in Optimizations and Applications
University of Minnesota, June 24, 2021

Joint work with:
Brendan Cook (UMN), Dejan Slepčev (CMU), Matthew Thorpe (Manchester)

Research supported by NSF-DMS 1713691, 1944925, and the Alfred P. Sloan Foundation

Outline

1. Introduction
   - Graph-based semi-supervised learning
   - Laplacian regularization
   - Spikes at low label rates
   - Outline of talk

2. Avoiding the spikes (moderate label rates)
   - Random geometric graph
   - Rates of convergence

3. Poisson learning: Embracing the spikes
   - Random walk perspective
   - Poisson learning

4. Experimental results
   - Volume constrained algorithms

5. The continuum perspective

Outline

Quick intro to learning

Fully supervised: Given training data $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, learn a function

$$u : \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m.$$

Quick intro to learning

Fully supervised: Given training data $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, learn a function

$$u : \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m.$$

Semi-supervised learning: Given additional **unlabeled data** $x_{m+1}, \ldots, x_n$ for $n \gg m$, use both the labeled and unlabeled data to learn f.

Quick intro to learning

Fully supervised: Given training data $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, learn a function

$$u : \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m.$$

Semi-supervised learning: Given additional **unlabeled data** $x_{m+1}, \ldots, x_n$ for $n \gg m$, use both the labeled and unlabeled data to learn f.

1. **Inductive learning:** Learn a function

$$u : \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m.$$

Quick intro to learning

Fully supervised: Given training data $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, learn a function

$$u : \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m.$$

Semi-supervised learning: Given additional **unlabeled data** $x_{m+1}, \ldots, x_n$ for $n \gg m$, use both the labeled and unlabeled data to learn f.

1. **Inductive learning:** Learn a function

$$u : \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m.$$

2. **Transductive learning:** Learn a function

$$u : \{x_1, x_2, \ldots, x_n\} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m$$

Quick intro to learning

Fully supervised: Given training data $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, learn a function

$$u : \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m.$$

Semi-supervised learning: Given additional **unlabeled data** $x_{m+1}, \ldots, x_n$ for $n \gg m$, use both the labeled and unlabeled data to learn f.

1. **Inductive learning:** Learn a function

$$u : \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m.$$

2. **Transductive learning:** Learn a function

$$u : \{x_1, x_2, \ldots, x_n\} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, m$$

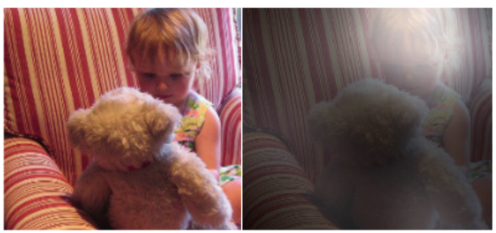
Unsupervised learning: Uses only the unlabeled data $x_1, \ldots, x_n$ (e.g., clustering).

Example: Automated image captioning

A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.

A **stop** sign is on a road with a mountain in the background



A little **girl** sitting on a bed with a teddy bear.

A group of **people** sitting on a boat in the water.

A giraffe standing in a forest with **trees** in the background.

[Yann LeCun, Yoshua Bengio, Geoffrey Hinton. Deep learning. Nature, 2015.]

Graph-based semi-supervised learning

Graph: $\mathcal{G} = (\mathcal{X}, \mathcal{W})$

- $\mathcal{X} = \{x_1, \dots, x_n\}$ are the vertices of the graph
- $\mathcal{W} = (w_{ij})_{i,j=1}^n$ are nonnegative and symmetric ($w_{ij} = w_{ji}$) edge weights.
- $w_{ij} \approx 1$ if x_i, x_j similar, and $w_{ij} \approx 0$ when dissimilar.

Graph-based semi-supervised learning

Graph: $\mathcal{G} = (\mathcal{X}, \mathcal{W})$

- $\mathcal{X} = \{x_1, \dots, x_n\}$ are the vertices of the graph
- $\mathcal{W} = (w_{ij})_{i,j=1}^n$ are nonnegative and symmetric ($w_{ij} = w_{ji}$) edge weights.
- $w_{ij} \approx 1$ if x_i, x_j similar, and $w_{ij} \approx 0$ when dissimilar.

Labels: We assume the first $m \ll n$ vertices are given labels

$$y_1, y_2, \dots, y_m \in \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\} \in \mathbb{R}^k.$$

Graph-based semi-supervised learning

Graph: $\mathcal{G} = (\mathcal{X}, \mathcal{W})$

- $\mathcal{X} = \{x_1, \dots, x_n\}$ are the vertices of the graph
- $\mathcal{W} = (w_{ij})_{i,j=1}^n$ are nonnegative and symmetric ($w_{ij} = w_{ji}$) edge weights.
- $w_{ij} \approx 1$ if x_i, x_j similar, and $w_{ij} \approx 0$ when dissimilar.

Labels: We assume the first $m \ll n$ vertices are given labels

$$y_1, y_2, \dots, y_m \in \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\} \in \mathbb{R}^k.$$

Task: Extend the labels to the rest of the vertices $x_{m+1}, \dots, x_n$.

Graph-based semi-supervised learning

Graph: $\mathcal{G} = (\mathcal{X}, \mathcal{W})$

- $\mathcal{X} = \{x_1, \dots, x_n\}$ are the vertices of the graph
- $\mathcal{W} = (w_{ij})_{i,j=1}^n$ are nonnegative and symmetric ($w_{ij} = w_{ji}$) edge weights.
- $w_{ij} \approx 1$ if x_i, x_j similar, and $w_{ij} \approx 0$ when dissimilar.

Labels: We assume the first $m \ll n$ vertices are given labels

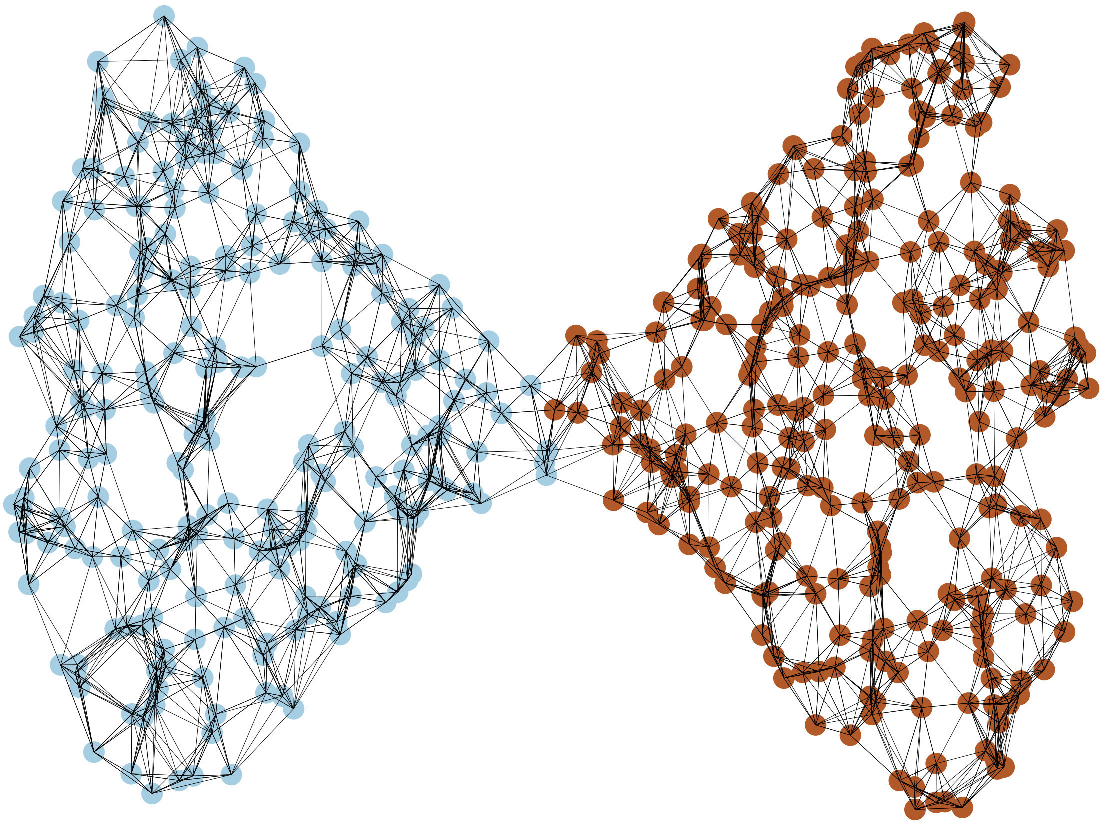
$$y_1, y_2, \dots, y_m \in \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\} \in \mathbb{R}^k.$$

Task: Extend the labels to the rest of the vertices $x_{m+1}, \dots, x_n$.

Semi-supervised: The graph encodes the unlabeled data in an efficient way.

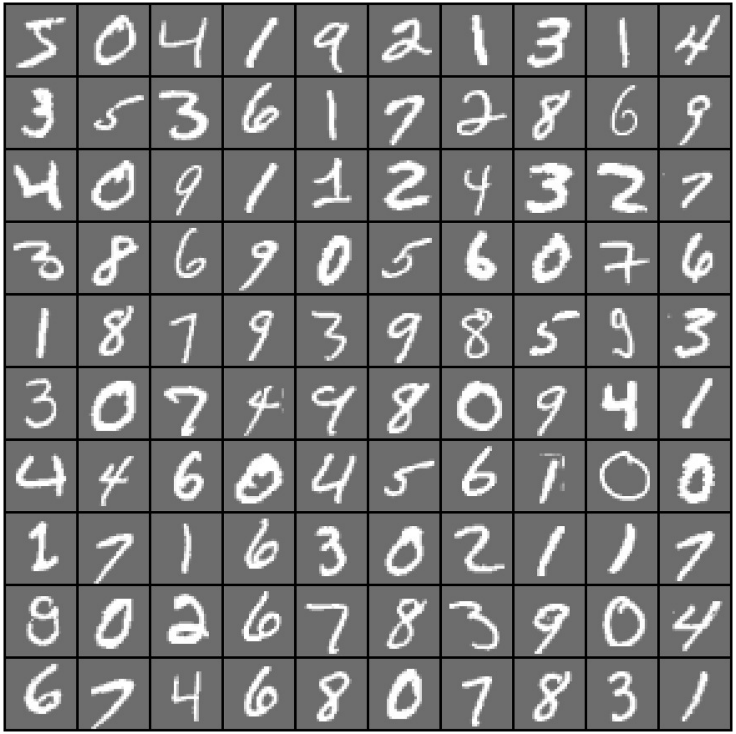
- Goal is to obtain good performance with far fewer labels compared to fully supervised learning.

Example: k-nearest neighbor graph

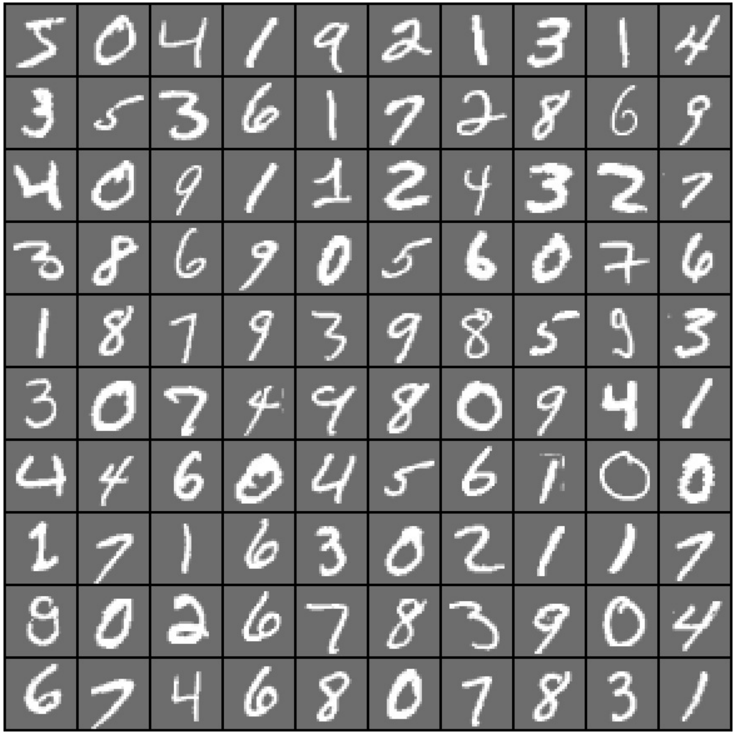


- We connect each point to its k-nearest neighbors ($k = 10$).
- Points are colored by the result of the spectral clustering.

MNIST (70,000 28×28 pixel images of digits 0-9)



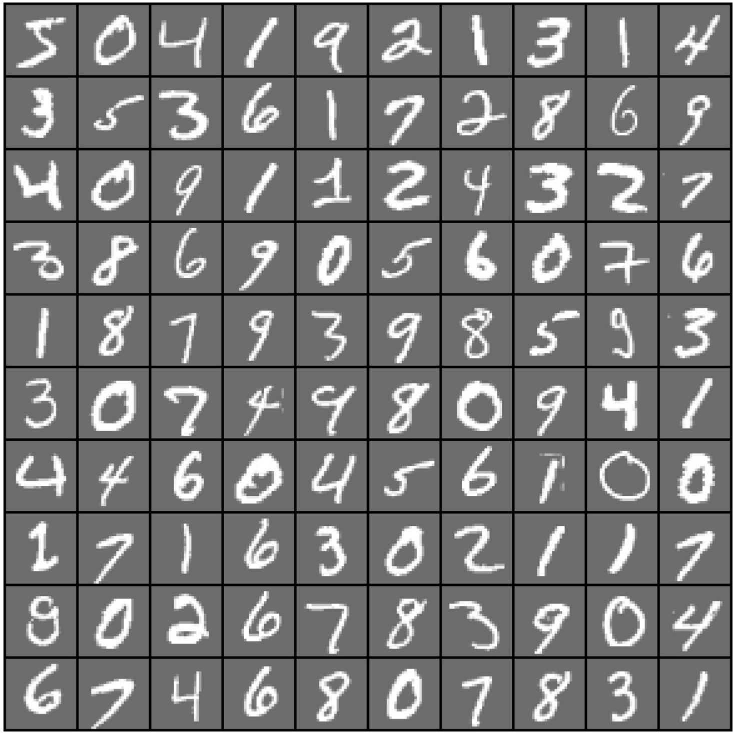
MNIST (70,000 28×28 pixel images of digits 0-9)



- Each image is a datapoint

$$x_i \in \mathbb{R}^{28\times 28} = \mathbb{R}^{784}.$$

MNIST (70,000 28×28 pixel images of digits 0-9)



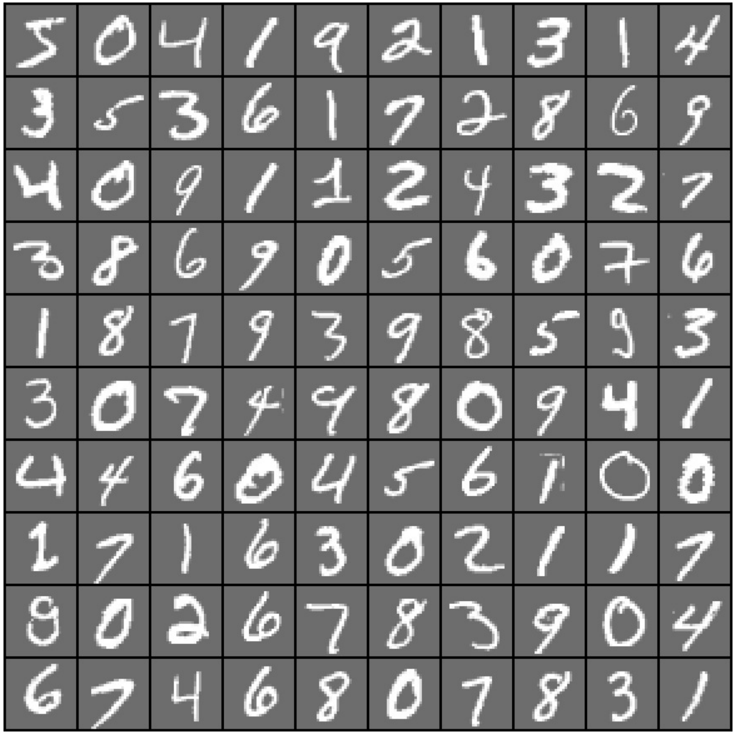
- Each image is a datapoint

$$x_i \in \mathbb{R}^{28\times 28} = \mathbb{R}^{784}.$$

- Geometric weights:

$$w_{ij} = \eta\left(\frac{|x_i - x_j|}{\varepsilon}\right)$$

MNIST (70,000 28×28 pixel images of digits 0-9)



- Each image is a datapoint

$$x_i \in \mathbb{R}^{28\times 28} = \mathbb{R}^{784}.$$

- Geometric weights:

$$w_{ij} = \eta\left(\frac{|x_i - x_j|}{\varepsilon}\right)$$

- k-nearest neighbor graph:

$$w_{ij} = \eta\left(\frac{|x_i - x_j|}{\varepsilon_k(x_i)}\right)$$

Clustering MNIST

`https://divamgupta.com`

Laplacian regularization

Laplacian regularized semi-supervised learning solves the Laplace equation

$$\begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m, \end{cases}$$

where $u : \mathcal{X} \to \mathbb{R}^k$, and $\mathcal{L}$ is the graph Laplacian

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)).$$

Laplacian regularization

Laplacian regularized semi-supervised learning solves the Laplace equation

$$\begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m, \end{cases}$$

where $u : \mathcal{X} \to \mathbb{R}^k$, and $\mathcal{L}$ is the graph Laplacian

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)).$$

The label decision for vertex x_i is determined by the largest component of $u(x_i)$

$$\ell(x_i) = \operatorname*{argmax}_{j \in \{1,\dots,k\}} \{u_j(x)\}.$$

References:

- Original work [Zhu et al., 2003]
- Learning [Zhou et al., 2005, Ando and Zhang, 2007]
- Manifold ranking [He et al., 2006, Zhou et al., 2011, Xu et al., 2011]

Label propagation

The solution of Laplace learning satisfies

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)) = 0 \qquad (m+1 \leq i \leq n).$$

Re-arranging, we see that u satisfies the mean-property

$$u(x_i) = \frac{\sum_{j=1}^{n} w_{ij}\, u(x_j)}{\sum_{j=1}^{n} w_{ij}}.$$

Label propagation

The solution of Laplace learning satisfies

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)) = 0 \qquad (m+1 \leq i \leq n).$$

Re-arranging, we see that u satisfies the mean-property

$$u(x_i) = \frac{\sum_{j=1}^{n} w_{ij}\, u(x_j)}{\sum_{j=1}^{n} w_{ij}}.$$

Label propagation [Zhu 2005] iterates

$$u^{k+1}(x_i) = \frac{\sum_{j=1}^{n} w_{ij}\, u^k(x_j)}{\sum_{j=1}^{n} w_{ij}},$$

d_i

and at convergence is equivalent to Laplace learning.

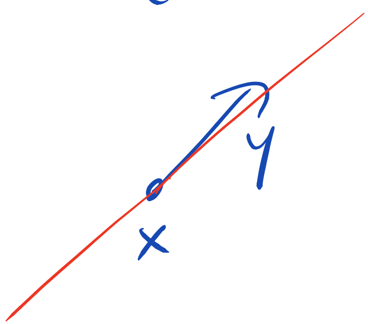
Variational interpretation

Laplace learning is equivalent to the variational problem

$$(*) \quad \min_{u:\mathcal{X}\to\mathbb{R}} \left\{ \sum_{i,j=1}^{n} w_{ij}|u(x_i)-u(x_j)|^2 : u(x_i)=y_i \text{ for } i=1,\dots,m \right\}.$$

Recall: $f:\mathbb{R}^n \to \mathbb{R}$, $\nabla f = (f_{x_1}, f_{x_2}, \dots, f_{x_n})$

$$\frac{d}{dt} f(x+ty) = \nabla f(x+ty)\cdot y \quad \text{Chain Rule}$$



$$\left.\frac{d}{dt}\right|_{t=0} f(x+ty) = \nabla f(x)\cdot y = \langle \nabla f(x), y\rangle$$

Given an inner product, we can define gradients by

$$\forall y, \quad \frac{d}{dt}\Big|_{t=0} f(x+ty) = \langle \nabla f(x), y \rangle$$

Inner product for graph functions

$$u, v : \mathcal{X} \longrightarrow \mathbb{R}^k$$

$$\langle u, v \rangle = \sum_{i=1}^{n} u(x_i) \cdot v(x_i)$$

$$E(u) = \frac{1}{4} \sum_{i,j=1}^{n} w_{ij} \left| u(x_i) - u(x_j) \right|^2$$

Let u be a minimizer of $(*)$.

Let $v : \mathcal{X} \to \mathbb{R}^k$ such that $v(x_i) = 0$

$i = 1, \ldots, m.$

$u(x_i) + t v(x_i) = y_i$

Then

$$E(u) \leq E(u + tv), \quad \forall t \in \mathbb{R}$$

So

$$0 = \frac{d}{dt}\Big|_{t=0} E(u+tv) \overset{?}{=} \langle \nabla E(u), v \rangle$$

$$\frac{d}{dt} E(u+tv) = \frac{1}{4} \sum_{i,j=1}^{n} w_{ij} \frac{d}{dt} \left| u(x_i) - u(x_j) + t(v(x_i) - v(x_j)) \right|^2$$

$$= \frac{2}{4} \sum_{i,j=1}^{n} w_{ij} \left(u(x_i) - u(x_j) + t \left(v(x_i) - v(x_j) \right) \right) \cdot \left(v(x_i) - v(x_j) \right)$$

$t = 0$

$$\left. \frac{d}{dt} \right|_{t=0} E(u + tv) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(u(x_i) - u(x_j) \right) \cdot \left(v(x_i) - v(x_j) \right)$$

$$= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(u(x_i) - u(x_j) \right) \cdot v(x_i)$$

$$- \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(u(x_i) - u(x_j) \right) \cdot v(x_j)$$

swap (i,j)

$$= \sum_{i,j=1}^{n} w_{ij} \left(u(x_i) - u(x_j) \right) \cdot v(x_i)$$

$$(**) = \sum_{i=1}^{n} \Big(\underbrace{\sum_{j=1}^{n} w_{ij} (u(x_i) - u(x_j))}_{Lu(x_i)} \Big) \cdot v(x_i)$$

$$= \langle Lu, v \rangle$$

Hence, $\boxed{\nabla E(u) = Lu.}$ Minimizers of $(*)$ satisfy

$$0 = \frac{d}{dt}\Big|_{t=0} E(u + tv) = \langle Lu, v \rangle$$

for all $v : \mathcal{X} \to \mathbb{R}^K$ s.t. $v(x_i) = 0$
$i = 1, \dots, m$.

Choose

$$V_j(x_i) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

for $j = m+1, m+2, \dots, n$ to get

$$0 = \langle Lu, v_j \rangle = Lu(x_j)$$

So a minimizer u satisfies

Laplace Learning

$$\begin{cases} Lu(x_i) = 0, & i = m+1, \dots, n \\ u(x_i) = y_i, & i = 1, \dots, m. \end{cases}$$

Let's change the inner product to

$$\langle u, v \rangle_d = \sum_{i=1}^n d_i \, u(x_i) \cdot v(x_i)$$

where $d_i = \sum_{j=1}^n w_{ij}$ is the degree.

(**) $\displaystyle \frac{d}{dt}\Big|_{t=0} E(u+tv) = \sum_{i=1}^n Lu(x_i) \cdot v(x_i)$

$$\rangle = \sum_{i=1}^{n} d_i \, d_i^{-1} Lu(x_i) \cdot v(x_i)$$

$$= \langle d^{-1} Lu, v \rangle_d$$

$$\nabla_d E(u) = \underbrace{d^{-1} L}_{\text{Random walk graph Laplacian.}} u$$

Gradient Descent

$$u^{k+1} = u^k - \alpha \nabla_d E(u^k)$$

$$u^{k+1}(x_i) = u^k(x_i) - \alpha d_i^{-1} L u^k(x_i)$$

$$= u^k(x_i) - \alpha d_i^{-1} \sum_{j=1}^n w_{ij} \left(u^k(x_i) - u^k(x_j)\right)$$

$$= u^k(x_i) - \alpha d_i^{-1} \underbrace{\sum_{j=1}^n w_{ij}}_{d_i} u^k(x_i) + \alpha d_i^{-1} \sum_{j=1}^n w_{ij} u^k(x_j)$$

$$= u^k(x_j) - \alpha u^k(x_i) + \alpha d_i^{-1} \sum_{j=1}^n w_{ij} u^k(x_j)$$

Set $\alpha = 1$ to get

$$u^{k+1}(x_i) = d_i^{-1} \sum_{j=1}^n w_{ij} u^k(x_j)$$

Label Propagation

Variational interpretation

Laplace learning is equivalent to the variational problem

$$\min_{u:\mathcal{X}\to\mathbb{R}} \left\{ \sum_{i,j=1}^{n} w_{ij} \left|u(x_i) - u(x_j)\right|^2 \,:\, u(x_i) = y_i \text{ for } i = 1, \dots, m \right\}.$$

Variational interpretation

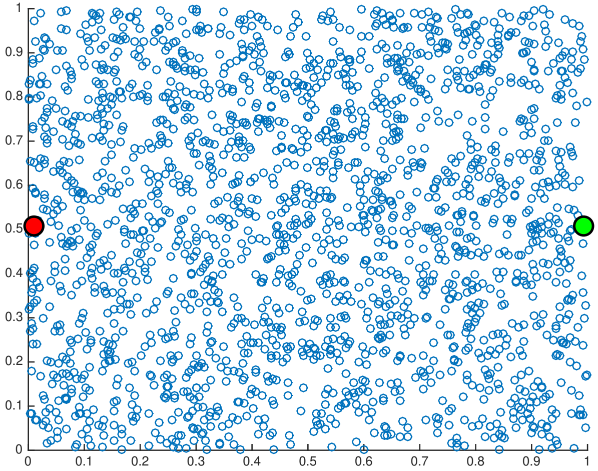
Laplace learning is equivalent to the variational problem

$$\min_{u:\mathcal{X}\to\mathbb{R}} \left\{ \sum_{i,j=1}^{n} w_{ij} |u(x_i) - u(x_j)|^2 \ : \ u(x_i) = y_i \text{ for } i = 1, \dots, m \right\}.$$

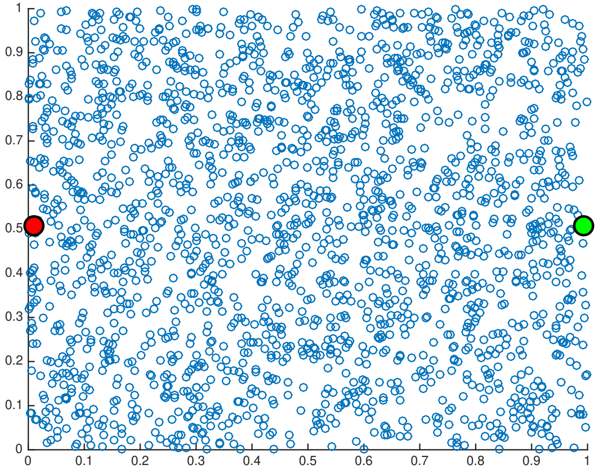
Many soft-constrained versions have been proposed

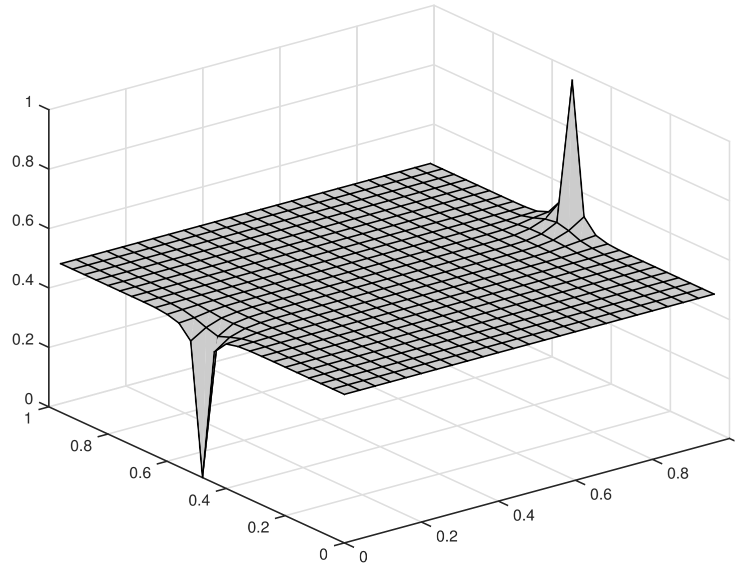
$$\min_{u:\mathcal{X}\to\mathbb{R}} \left\{ \sum_{i,j=1}^{n} w_{ij} |u(x_i) - u(x_j)|^2 + \lambda \sum_{i=1}^{m} \ell(u(x_i), y_i) \right\}.$$

Ill-posed with small amount of labeled data



Ill-posed with small amount of labeled data





- Graph is $n = 10^5$ i.i.d. random variables uniformly drawn from $[0, 1]^2$.
- $w_{xy} = 1$ if $|x - y| < 0.01$ and $w_{xy} = 0$ otherwise.
- Two labels: $y_1 = 0$ at the Red point and $y_2 = 1$ at the Green point.
- Over 95% of labels in $[0.4975, 0.5025]$.

[Nadler et al., 2009, El Alaoui et al., 2016]

Laplace learning on MNIST at low label rates

# Labels per class	**1**	**2**	**3**	**4**	**160**
Laplace Learning	16.1 (6.2)	28.2 (10.3)	42.0 (12.4)	57.8 (12.3)	97.0 (0.1)
Nearest Neighbor	65.4 (5.2)	74.2 (3.3)	77.8 (2.6)	80.7 (2.0)	92.4 (0.2)

- Average accuracy over 100 trials with standard deviation in brackets.
- Nearest neighbor is geodesic graph-nearest neighbor.

Recent work

The low-label rate problem was originally identified in [Nadler 2011].

A lot of recent work has attempted to address this issue with new graph-based classification algorithms at low label rates.

- Higher-order regularization: [Zhou and Belkin, 2011], [Dunlop et al., 2019]
- p-Laplace regularization: [Alaoui et al., 2016], [Calder 2018,2019], [Slepcev & Thorpe 2019]
- Re-weighted Laplacians: [Shi et al., 2017], [Calder & Slepcev, 2019]
- Centered kernel method: [Mai & Couillet, 2018]

Recent work

The low-label rate problem was originally identified in [Nadler 2011].

A lot of recent work has attempted to address this issue with new graph-based classification algorithms at low label rates.

- Higher-order regularization: [Zhou and Belkin, 2011], [Dunlop et al., 2019]
- p-Laplace regularization: [Alaoui et al., 2016], [Calder 2018,2019], [Slepcev & Thorpe 2019]
- Re-weighted Laplacians: [Shi et al., 2017], [Calder & Slepcev, 2019]
- Centered kernel method: [Mai & Couillet, 2018]

While we have lots of new models, the problem with Laplace learning at low label rates was still not well-understood.

Visualization of spikes

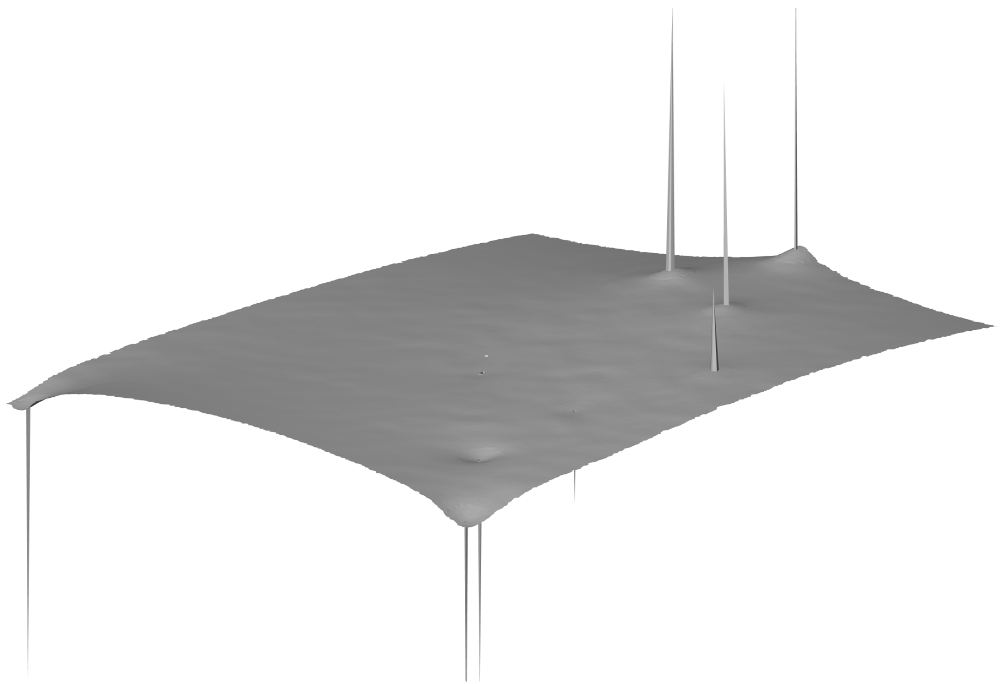


Figure: Demonstration of spikes in Laplacian learning. Label function is $\cos(x_1)$.

Visualization of spikes

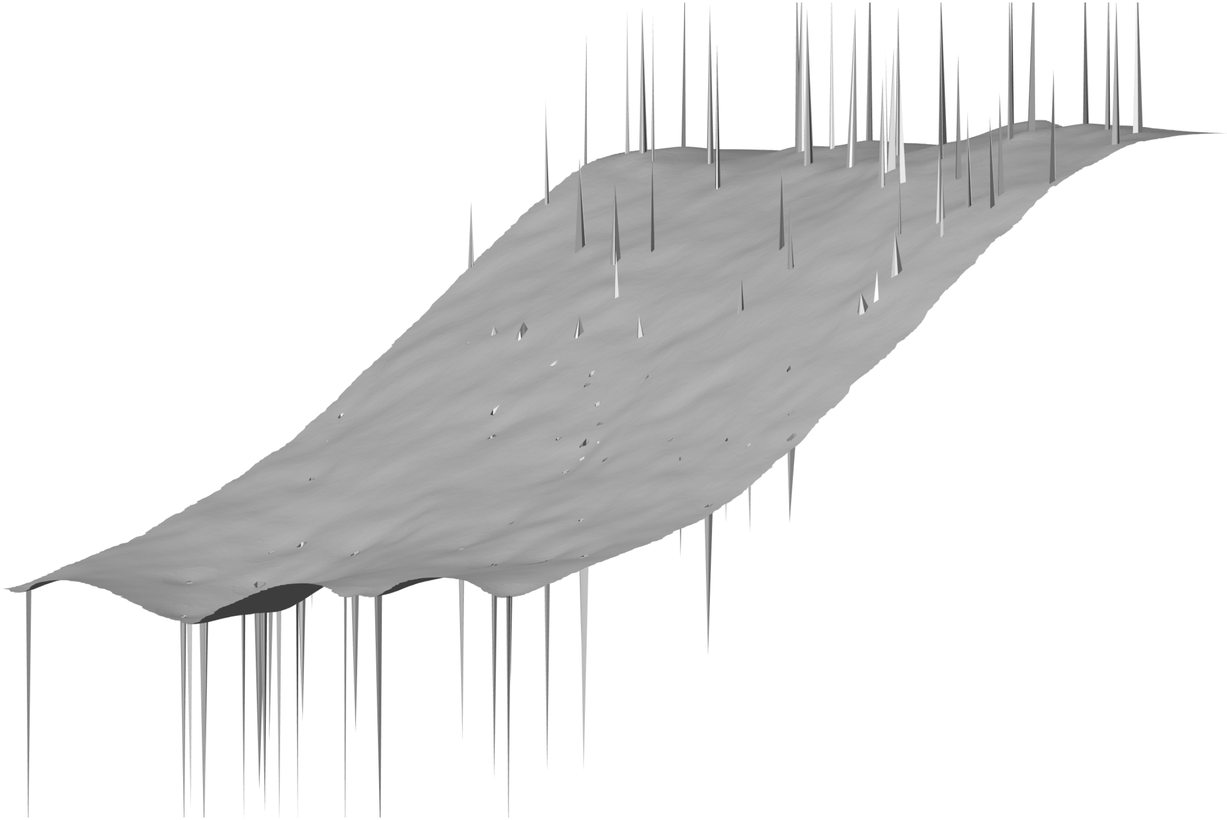


Figure: Demonstration of spikes in Laplacian learning. Label function is $\cos(x_1)$.

Visualization of spikes

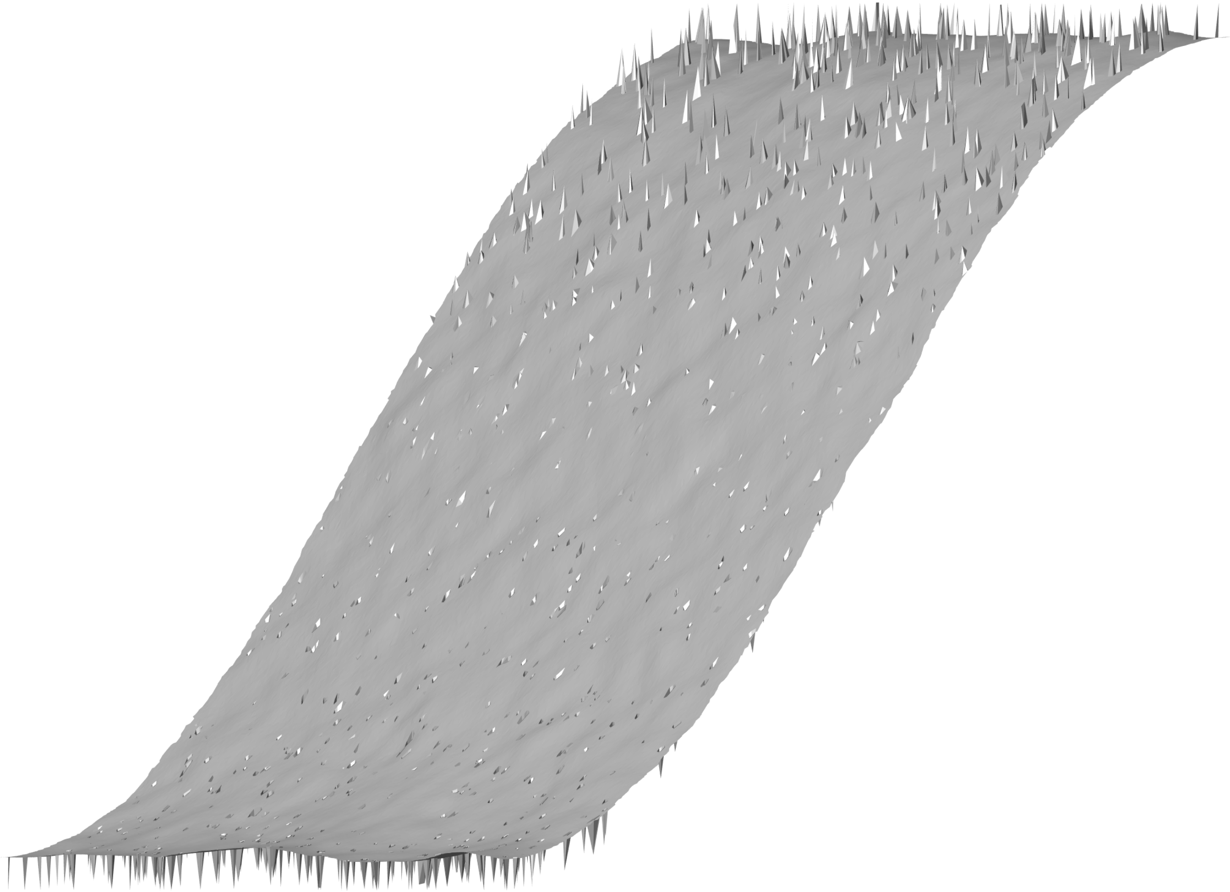
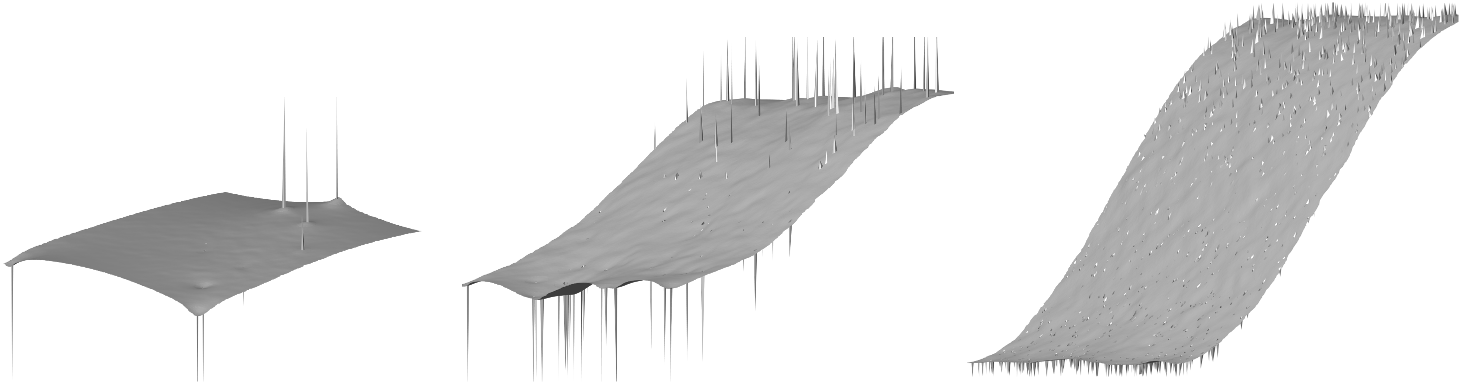


Figure: Demonstration of spikes in Laplacian learning. Label function is $\cos(x_1)$.

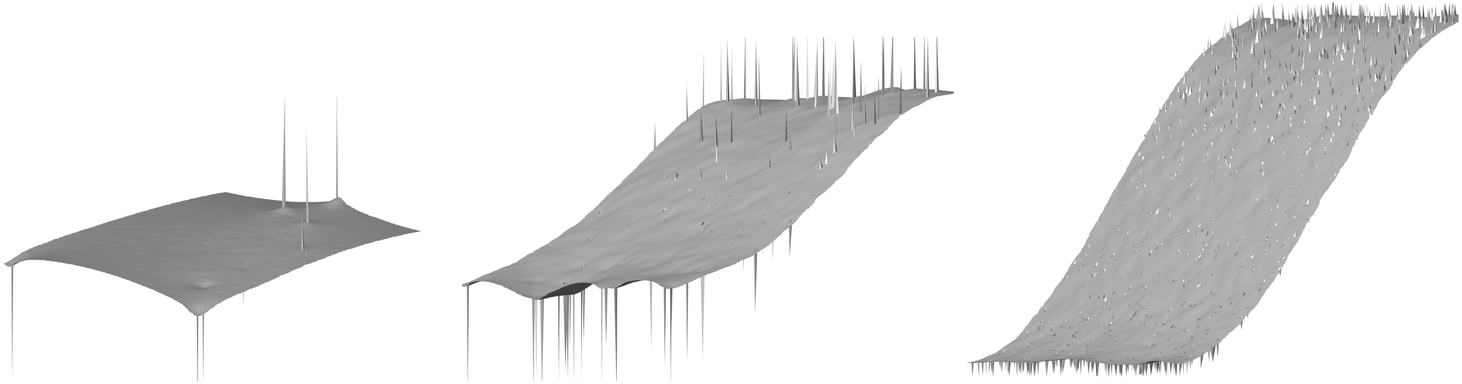
Main directions for talk



Label function is $\cos(x_1)$.

1. **Avoiding spikes:** How many labels do we need to ensure spikes do not form?

Main directions for talk

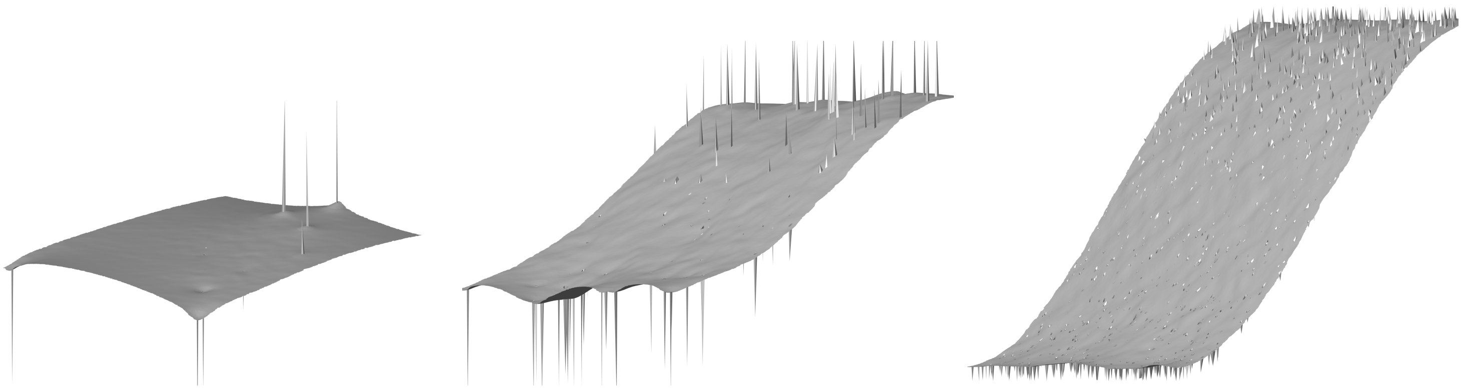


Label function is $\cos(x_1)$.

1. **Avoiding spikes:** How many labels do we need to ensure spikes do not form?

2. **Analyzing the spikes:** Why do we see poor classification results at low label rates?
   - Are the spikes too localized? Do they propagate information?
   - Is a flat scoring function problematic?

Main directions for talk



Label function is $\cos(x_1)$.

1. **Avoiding spikes:** How many labels do we need to ensure spikes do not form?
2. **Analyzing the spikes:** Why do we see poor classification results at low label rates?
   - Are the spikes too localized? Do they propagate information?
   - Is a flat scoring function problematic?
3. **Poisson learning:** Careful analysis will lead to a simple fix and a new algorithm.
   - Spikes can be interpreted as source terms in a Poisson equation.
   - Experiments on MNIST, FashionMNIST, and CIFAR-10

Outline

Random geometric graph

Random Geometric Graph: Assume the vertices of the graph are

$$\mathcal{X} = \{x_1, \dots, x_n\}$$

where $x_1, \dots, x_n$ is a sequence of i.i.d. random variables on $\Omega \subset \mathbb{R}^d$ with positive density ρ, and the weights are given by

$$w_{ij} = \eta\left(\frac{|x_i - x_j|}{\varepsilon}\right), \tag{1}$$

where $\eta : [0, \infty) \to [0, 1]$ is smooth with compact support.

Random geometric graph

Random Geometric Graph: Assume the vertices of the graph are

$$\mathcal{X} = \{x_1, \ldots, x_n\}$$

where $x_1, \ldots, x_n$ is a sequence of i.i.d. random variables on $\Omega \subset \mathbb{R}^d$ with positive density ρ, and the weights are given by

$$w_{ij} = \eta\left(\frac{|x_i - x_j|}{\varepsilon}\right), \tag{1}$$

where $\eta : [0, \infty) \to [0, 1]$ is smooth with compact support. In particular, we assume

$$\begin{cases} \eta(t) \geq 1, & \text{if } 0 \leq t \leq 1 \\ \eta(t) = 0, & \text{if } t > 2 \\ \eta(t) \geq 0, & \text{for all } t \geq 0. \end{cases}$$

Pointwise consistency of graph Laplacian

The graph Laplacian is defined as

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)).$$

Pointwise consistency of graph Laplacian

The graph Laplacian is defined as

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)).$$

In the large data $n \to \infty$ and sparse graph $\varepsilon \to 0$ limit, $\mathcal{L}$ is consistent with

$$\Delta_\rho u = -\rho^{-1}\mathrm{div}(\rho^2 \nabla u).$$

Pointwise consistency of graph Laplacian

The graph Laplacian is defined as

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)).$$

In the large data $n \to \infty$ and sparse graph $\varepsilon \to 0$ limit, $\mathcal{L}$ is consistent with

$$\Delta_\rho u = -\rho^{-1}\mathrm{div}(\rho^2 \nabla u).$$

In particular, it is a standard result [Hein et al., 2007] that

$$\left| \frac{1}{n\varepsilon^{d+2}} \mathcal{L}u(x) - \sigma_\eta \Delta_\rho u(x) \right| \le C(\lambda + \varepsilon)$$

holds for any $u \in C^3(\Omega)$ with probability at least $1 - 2\exp\left(-cn\varepsilon^{d+2}\lambda^2\right)$.

Pointwise consistency of graph Laplacian

The graph Laplacian is defined as

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)).$$

In the large data $n \to \infty$ and sparse graph $\varepsilon \to 0$ limit, $\mathcal{L}$ is consistent with

$$\Delta_\rho u = -\rho^{-1}\mathrm{div}(\rho^2 \nabla u).$$

In particular, it is a standard result [Hein et al., 2007] that

$$\left| \frac{1}{n\varepsilon^{d+2}} \mathcal{L}u(x) - \sigma_\eta \Delta_\rho u(x) \right| \leq C(\lambda + \varepsilon^2)$$

holds for any $u \in C^4(\Omega)$ with probability at least $1 - 2\exp\left(-cn\varepsilon^{d+2}\lambda^2\right)$.

Pointwise consistency of graph Laplacian

The graph Laplacian is defined as

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)).$$

In the large data $n \to \infty$ and sparse graph $\varepsilon \to 0$ limit, $\mathcal{L}$ is consistent with

$$\Delta_\rho u = -\rho^{-1}\mathrm{div}(\rho^2 \nabla u).$$

In particular, it is a standard result [Hein et al., 2007] that

$$\left| \frac{1}{n\varepsilon^{d+2}} \mathcal{L}u(x) - \sigma_\eta \Delta_\rho u(x) \right| \leq C(\lambda + \varepsilon^2)$$

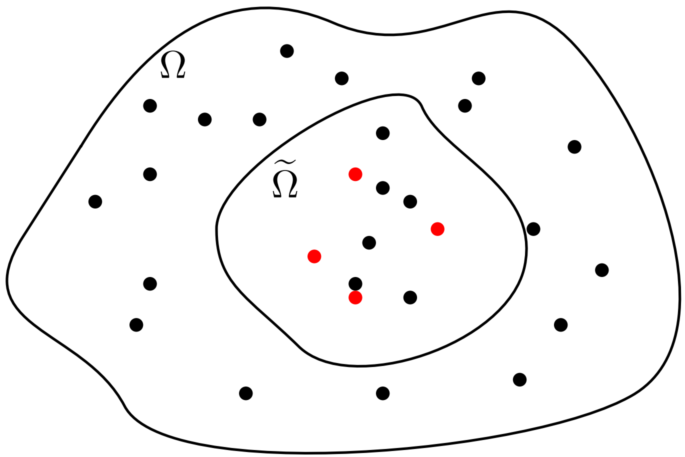
holds for any $u \in C^4(\Omega)$ with probability at least $1 - 2\exp\left(-cn\varepsilon^{d+2}\lambda^2\right)$.

The density ρ acts as an edge detector allowing sharp changes in u between clusters.

- E.g., Image processing equations like Perona-Malik $u_t - \mathrm{div}(\rho(|\nabla u|)\nabla u) = 0$.

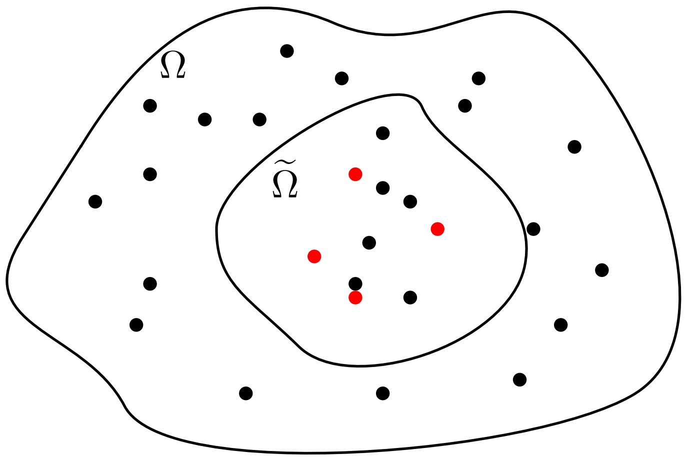
Model for labeled data

Model 1. Let $\beta \in (0,1]$ and $\widetilde{\Omega} \subset\subset \Omega$. Each $x_i \in \widetilde{\Omega}$ is selected as training data independently with probability β. Let $\Gamma =$ training data.



Model for labeled data

Model 1. Let $\beta \in (0,1]$ and $\widetilde{\Omega} \subset\subset \Omega$. Each $x_i \in \widetilde{\Omega}$ is selected as training data independently with probability β. Let Γ = training data.



The Laplacian learning problem is

$$(2) \qquad \begin{cases} \mathcal{L}u_n(x) = 0, & \text{if } x \in \mathcal{X} \setminus \Gamma \\ u_n(x) = g(x), & \text{if } x \in \Gamma, \end{cases}$$

where $g : \Omega \to \mathbb{R}$ is Lipschitz and

$$\mathcal{X} = \{x_1, x_2, \ldots, x_n\}.$$

Main result

The continuum PDE is

$$
(3)\qquad \begin{cases} \text{div}(\rho^2 \nabla u) = 0 & \text{in } \Omega \setminus \widetilde{\Omega} \\ u = g & \text{on } \widetilde{\Omega} \\ \nabla u \cdot \mathbf{n} = 0 & \text{on } \partial\Omega. \end{cases}
$$

Main result

The continuum PDE is

(3) $$\begin{cases} \text{div}(\rho^2 \nabla u) = 0 & \text{in } \Omega \setminus \widetilde{\Omega} \\ u = g & \text{on } \widetilde{\Omega} \\ \nabla u \cdot \mathbf{n} = 0 & \text{on } \partial\Omega. \end{cases}$$

Theorem (C.-Slepcev-Thorpe, 2020)

Let $u_n : \mathcal{X} \to \mathbb{R}$ *be the solution of* (2), *and let* $u \in C^3(\overline{\Omega})$ *be the solution of* (3). *If* $\beta \geq \varepsilon^2$ *and* $\varepsilon \leq \lambda \leq c$ *then*

(4) $$\max_{x \in \mathcal{X}} |u_n(x) - u(x)| \leq C \left(\frac{\varepsilon}{\sqrt{\beta}} \log\left(\frac{\sqrt{\beta}}{\varepsilon} \right) + \lambda \right)$$

holds with probability at least $1 - Cn \exp\left(-cn\varepsilon^{d+2}\lambda^2\right)$.

The negative result

Theorem (C.-Slepcev-Thorpe, 2020)

Assume that $\beta = \beta_n \to 0^+$ *and* $\varepsilon = \varepsilon_n \to 0^+$ *satisfy*

$$\beta_n \ll \varepsilon_n^2, \quad \text{and} \quad n\varepsilon_n^d \gg \log(n). \tag{5}$$

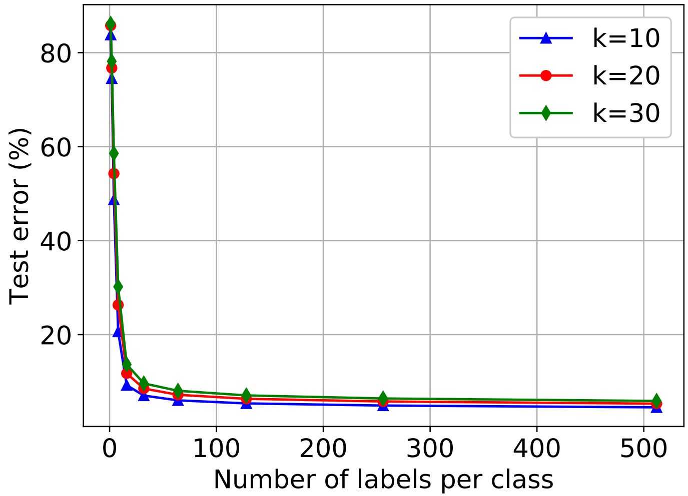
Then, with probability one, the sequence u_n *is pre-compact in* TL^2 *and any convergent subsequence converges to a constant.*

Summary: Laplacian learning propagates labels well when

$$\text{Label rate } = \beta \gg \varepsilon^2.$$

Below this label rate, spikes form and the solution is degenerate.

Error on MNIST



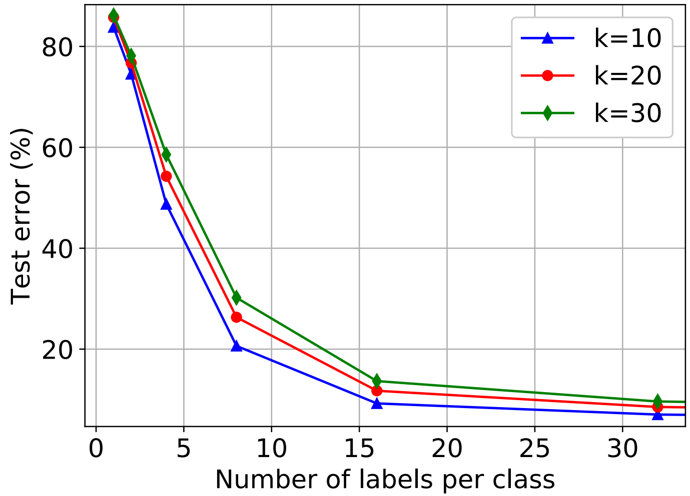


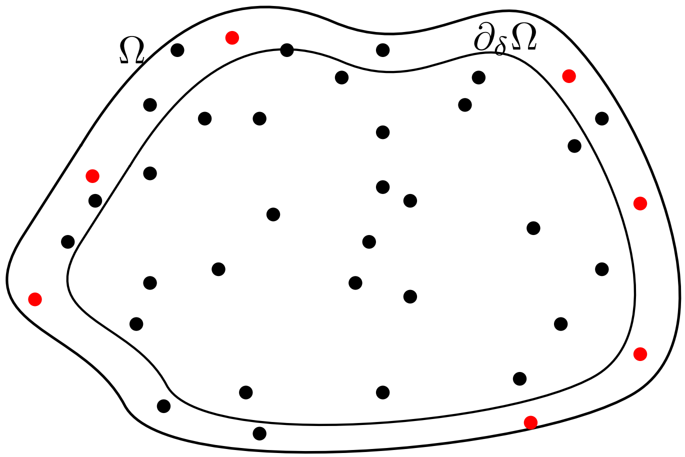
Figure: Error plots for MNIST experiment showing testing error versus number of labels, averaged over 100 trials.

Fits very well to the error rate $\beta^{-1/2}$.

Another model

Model 2. Let $\beta \in (0,1)$, $\delta \in (0,\varepsilon]$. Each $x_i \in \partial_\delta\Omega$ is selected as training data independently with probability β, where

$$\partial_\delta\Omega = \{x \in \Omega : \text{dist}(x, \partial\Omega) < \delta\}.$$



Here, the continuum PDE is

$$\begin{cases} \text{div}(\rho^2 \nabla u) = 0 & \text{in } \Omega \\ u = g & \text{on } \partial\Omega. \end{cases} \tag{6}$$

J. Calder, D. Slepčev, D., and M. Thorpe. **Rates of convergence for Laplacian semi-supervised learning with low label rates.** *arXiv:2006.02765*, 2020.

Outline

1. Introduction
   - Graph-based semi-supervised learning
   - Laplacian regularization
   - Spikes at low label rates
   - Outline of talk
2. Avoiding the spikes (moderate label rates)
   - Random geometric graph
   - Rates of convergence
3. Poisson learning: Embracing the spikes
   - Random walk perspective
   - Poisson learning
4. Experimental results
   - Volume constrained algorithms
5. The continuum perspective

Random walks on random graphs

Let $X_0, X_1, X_2, \ldots$ be a random walk on $\mathcal{X} = \{x_1, \ldots, x_n\}$ with transition probabilities

$$\mathbb{P}(X_k = x_j \mid X_{k-1} = x_i) = \frac{w_{ij}}{d(x_i)}, \quad d(x_i) = \sum_{j=1}^{n} w_{ij}.$$

Random walks on random graphs

Let $X_0, X_1, X_2, \ldots$ be a random walk on $\mathcal{X} = \{x_1, \ldots, x_n\}$ with transition probabilities

$$\mathbb{P}(X_k = x_j \mid X_{k-1} = x_i) = \frac{w_{ij}}{d(x_i)}, \quad d(x_i) = \sum_{j=1}^{n} w_{ij}.$$

For any $u : \mathcal{X} \to \mathbb{R}$ we compute

$$\mathbb{E}[u(X_k) - u(X_{k-1}) \mid X_{k-1}] = \frac{1}{d(X_{k-1})}\mathcal{L}u(X_{k-1}).$$

$$\mathbb{E}\left[u(X_k) - u(X_{k-1}) \mid X_{k-1}\right]$$

$$= \sum_{j=1}^{n} \frac{w_{X_{k-1},j}}{d(X_{k-1})} \left(u(x_j) - u(X_{k-1})\right)$$

$$= \frac{1}{d(X_{k-1})} Lu(X_{k-1})$$

Generator for random walk.

Random walks on random graphs

Let $X_0, X_1, X_2, \ldots$ be a random walk on $\mathcal{X} = \{x_1, \ldots, x_n\}$ with transition probabilities

$$\mathbb{P}(X_k = x_j \mid X_{k-1} = x_i) = \frac{w_{ij}}{d(x_i)}, \quad d(x_i) = \sum_{j=1}^{n} w_{ij}.$$

Random walks on random graphs

Let $X_0, X_1, X_2, \ldots$ be a random walk on $\mathcal{X} = \{x_1, \ldots, x_n\}$ with transition probabilities

$$\mathbb{P}(X_k = x_j \mid X_{k-1} = x_i) = \frac{w_{ij}}{d(x_i)}, \quad d(x_i) = \sum_{j=1}^{n} w_{ij}.$$

For any $u : \mathcal{X} \to \mathbb{R}$ we compute

$$\mathbb{E}[u(X_k) - u(X_{k-1}) \mid X_{k-1}] = \frac{1}{d(X_{k-1})}\mathcal{L}u(X_{k-1}).$$

Random walks on random graphs

Let $X_0, X_1, X_2, \ldots$ be a random walk on $\mathcal{X} = \{x_1, \ldots, x_n\}$ with transition probabilities

$$\mathbb{P}(X_k = x_j \mid X_{k-1} = x_i) = \frac{w_{ij}}{d(x_i)}, \quad d(x_i) = \sum_{j=1}^{n} w_{ij}.$$

For any $u : \mathcal{X} \to \mathbb{R}$ we compute

$$\mathbb{E}[u(X_k) - u(X_{k-1}) \mid X_{k-1}] = \frac{1}{d(X_{k-1})}\mathcal{L}u(X_{k-1}).$$

The random walk Laplacian $\frac{1}{d}\mathcal{L}$ is the generator for the random walk.

Random walks on random graphs

Let $X_0, X_1, X_2, \ldots$ be a random walk on $\mathcal{X} = \{x_1, \ldots, x_n\}$ with transition probabilities

$$\mathbb{P}(X_k = x_j \mid X_{k-1} = x_i) = \frac{w_{ij}}{d(x_i)}, \quad d(x_i) = \sum_{j=1}^{n} w_{ij}.$$

For any $u : \mathcal{X} \to \mathbb{R}$ we compute

$$\mathbb{E}[u(X_k) - u(X_{k-1}) \mid X_{k-1}] = \frac{1}{d(X_{k-1})}\mathcal{L}u(X_{k-1}).$$

The random walk Laplacian $\frac{1}{d}\mathcal{L}$ is the generator for the random walk.

Hence, if $\mathcal{L}u = 0$ on $\mathcal{X}_n$, then

$$E[u(X_k) - u(X_{k-1}) \mid X_{k-1}] = 0$$

so $u(X_k)$ is a martingale.

Random Walk Perspective

Suppose $u : \mathcal{X} \to \mathbb{R}^k$ solves the Laplace learning equation

$$\begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m. \end{cases}$$

Let $X_0, X_1, X_2, \ldots$ be a random walk on $\mathcal{X}$ and define the stopping time

$$\tau = \inf\{k \geq 0 \, : \, X_k \in \{x_1, x_2, \ldots, x_m\}\}.$$

Random Walk Perspective

Suppose $u : \mathcal{X} \to \mathbb{R}^k$ solves the Laplace learning equation

$$\begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m. \end{cases}$$

Let $X_0, X_1, X_2, \ldots$ be a random walk on $\mathcal{X}$ and define the stopping time

$$\tau = \inf\{k \geq 0 \,:\, X_k \in \{x_1, x_2, \ldots, x_m\}\}.$$

Let $i_\tau \leq m$ so that $X_\tau = x_{i_\tau}$. Then (by Doob's optimal stopping theorem)

(7) $$\boxed{u(x) = \mathbb{E}[y_{i_\tau} \mid X_0 = x].}$$

Random Walk Perspective

Suppose $u : \mathcal{X} \to \mathbb{R}^k$ solves the Laplace learning equation

$$\begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m. \end{cases}$$

Let $X_0, X_1, X_2, \dots$ be a random walk on $\mathcal{X}$ and define the stopping time

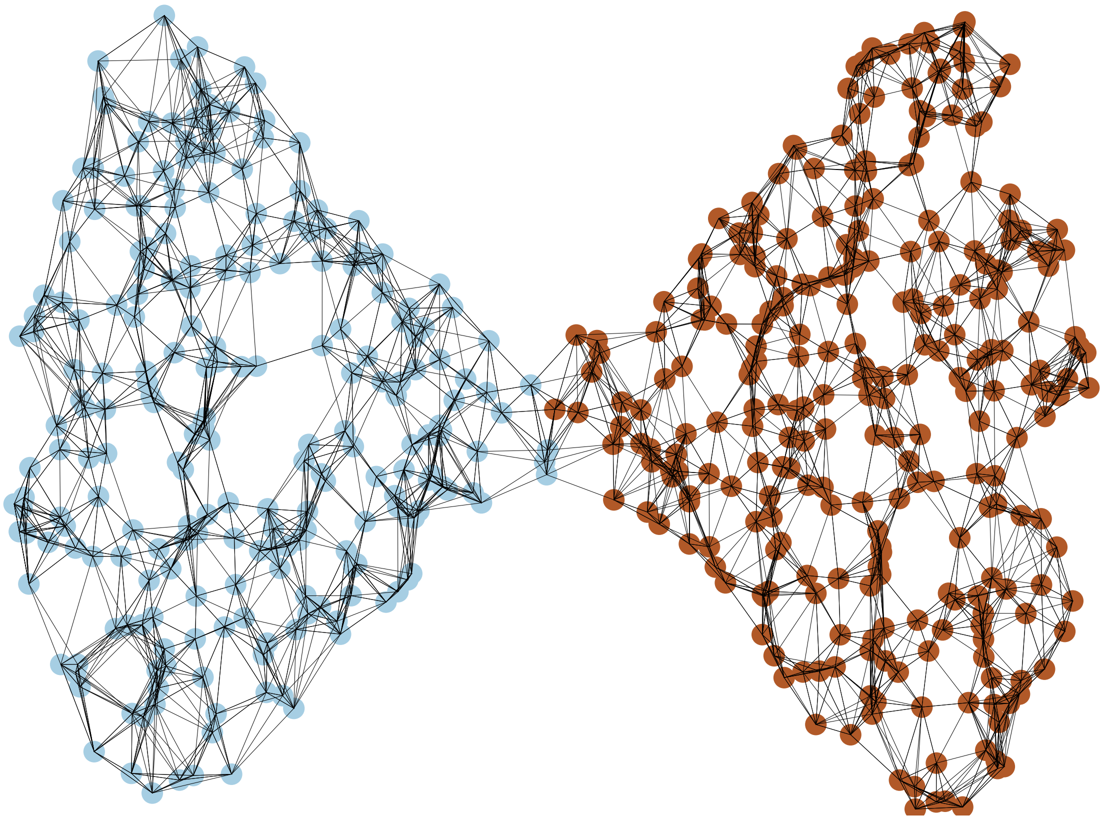
$$\tau = \inf\{k \geq 0 \,:\, X_k \in \{x_1, x_2, \dots, x_m\}\}.$$

Let $i_\tau \leq m$ so that $X_\tau = x_{i_\tau}$. Then (by Doob's optimal stopping theorem)

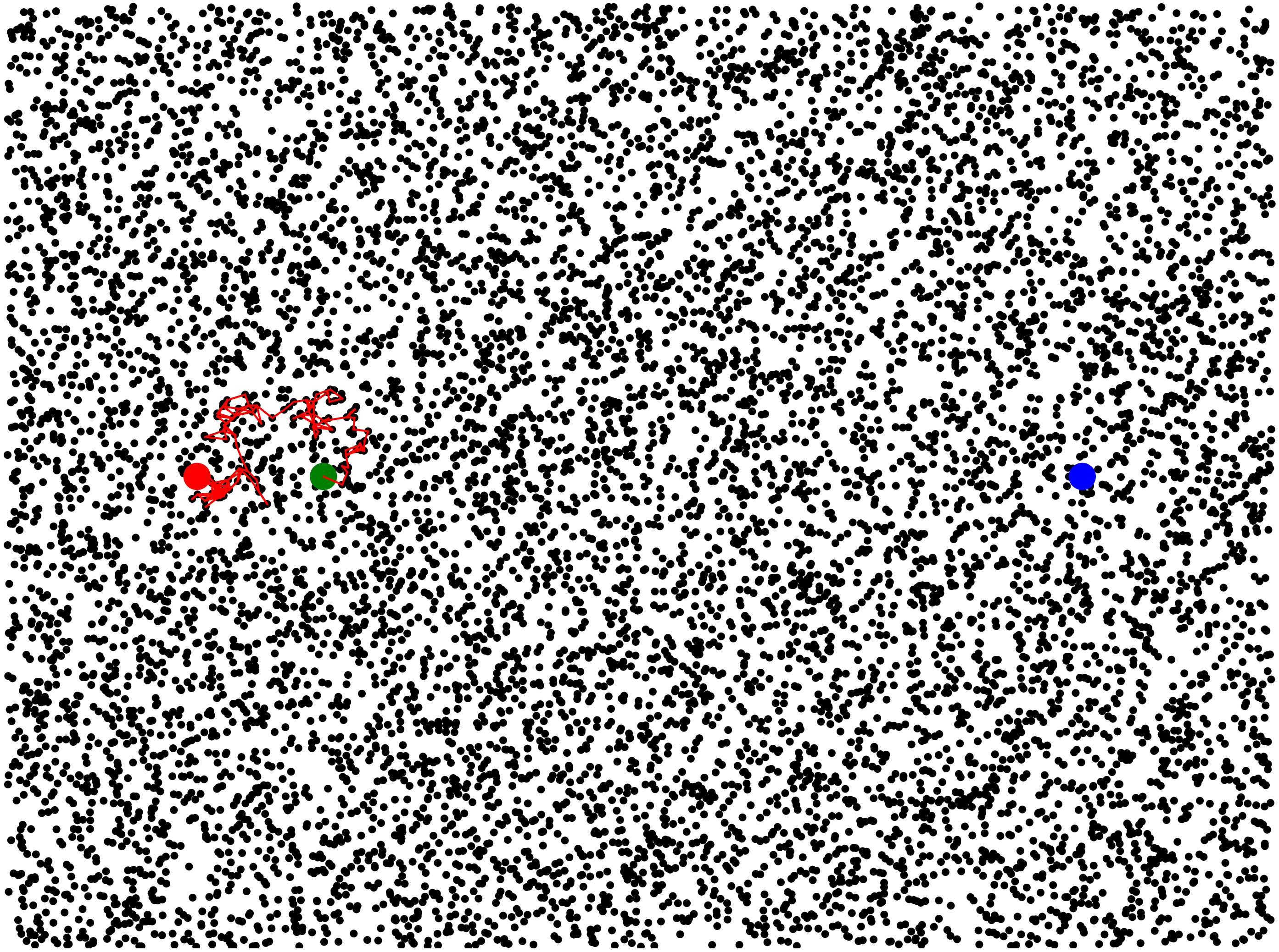
$$\boxed{u(x) = \mathbb{E}[y_{i_\tau} \mid X_0 = x].} \tag{7}$$

This says $u(x)$ is a weighted average of (hopefully) nearby label vectors.

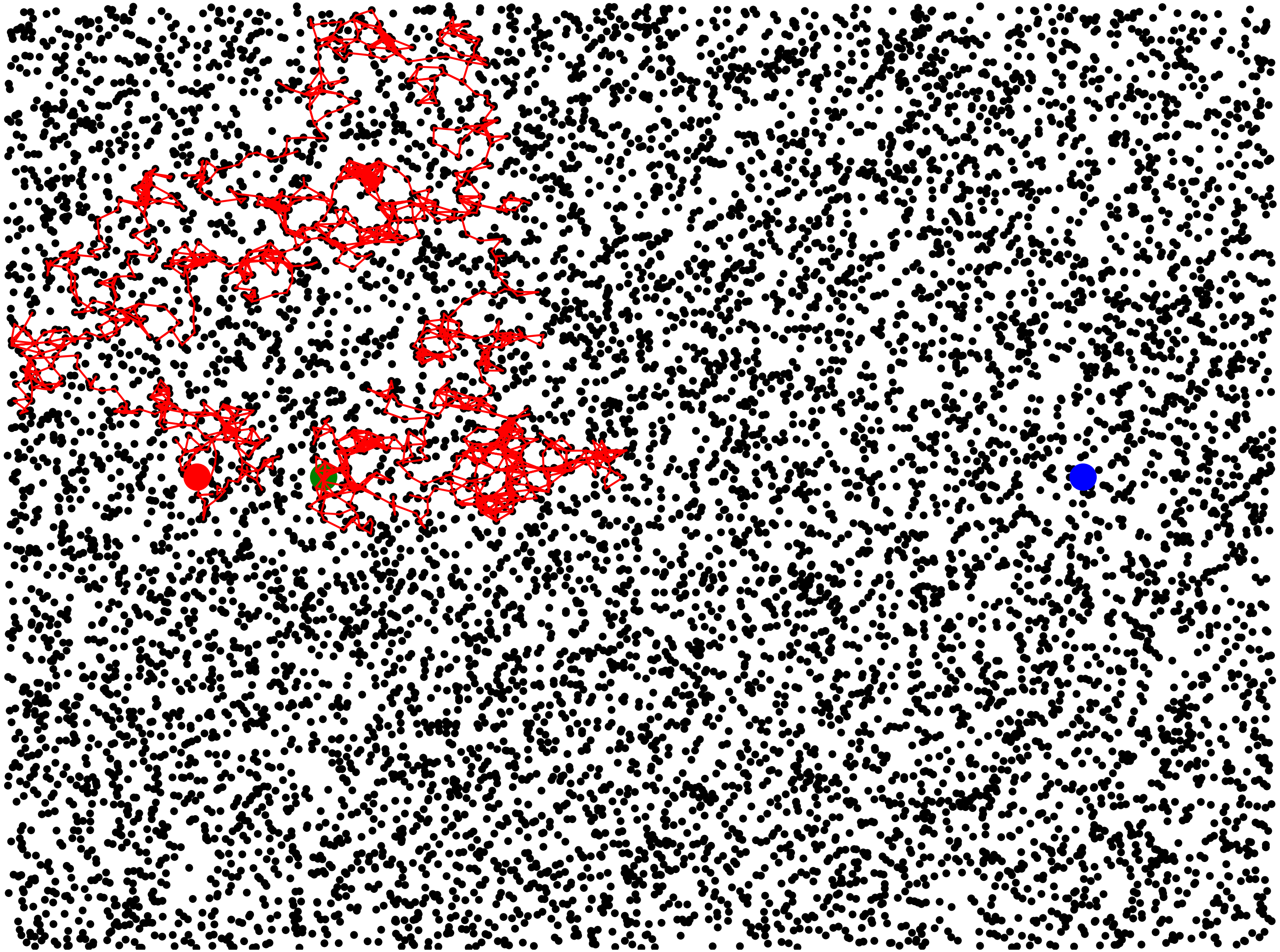
Random Walk Perspective



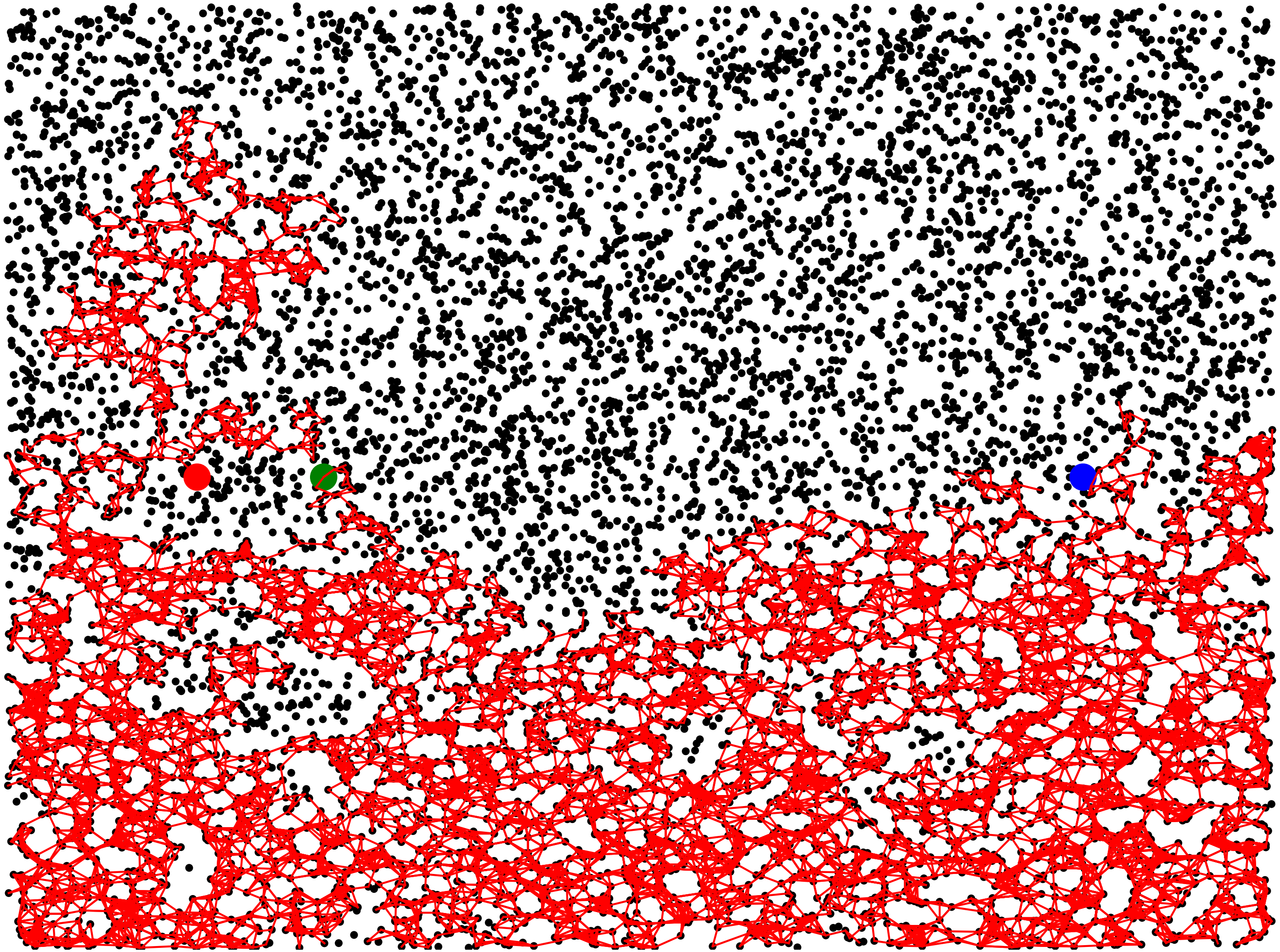
Random walk experiment



Random walk experiment



Random walk experiment



The Random walk perspective

At low label rates, the random walker reaches the mixing time before hitting a label.

- The label eventually hit is largely independent of where the walker starts.

The Random walk perspective

At low label rates, the random walker reaches the mixing time before hitting a label.

- The label eventually hit is largely independent of where the walker starts.

After walking for a long time, the probability distribution of the walker approaches the invariant distribution π given by

$$\pi_i = \frac{d_i}{\sum_{j=1}^n d_j}, \quad d_i = \sum_{j=1}^n w_{ij}.$$

The Random walk perspective

At low label rates, the random walker reaches the mixing time before hitting a label.

- The label eventually hit is largely independent of where the walker starts.

After walking for a long time, the probability distribution of the walker approaches the invariant distribution π given by

$$\pi_i = \frac{d_i}{\sum_{j=1}^n d_j}, \quad d_i = \sum_{j=1}^n w_{ij}.$$

Thus, the solution of Laplace learning is approximately

$$u(x) = \mathbb{E}[y_{i_\tau} \mid X_0 = x] \approx \frac{\sum_{j=1}^n d_j y_j}{\sum_{j=1}^n d_j} =: c \in \mathbb{R}^k.$$

The Random walk perspective

To test this, we consider Shifted Laplace learning, which solves

$$\begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m, \end{cases}$$

and decides on the label by the shifted argmax:

$$\ell(x_i) = \operatorname*{argmax}_{j \in \{1,\dots,k\}} \{u_j(x) - c_j\},$$

where

$$c = \frac{\sum_{j=1}^n d_j\, y_j}{\sum_{j=1}^n d_j}.$$

The Random walk perspective

To test this, we consider Shifted Laplace learning, which solves

$$\begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m, \end{cases}$$

and decides on the label by the shifted argmax:

$$\ell(x_i) = \operatorname*{argmax}_{j \in \{1,\dots,k\}} \{u_j(x) - c_j\},$$

where

$$c = \frac{\sum_{j=1}^n d_j y_j}{\sum_{j=1}^n d_j}.$$

Experiment on MNIST:

# Labels/class	**1**	**2**	**3**	**4**	**5**
Laplace	16.1 (6.2)	28.2 (10)	42.0 (12)	57.8 (12)	69.5 (12)
Shift Laplace	88.3 (5.7)	92.6 (2.4)	94.3 (1.4)	94 (1.5)	95 (0.6)

A related Poisson equation

If the solution to Laplace learning u is roughly constant $u \approx c$, then at labeled nodes $x_1, \ldots, x_m$ we can compute

$$\mathcal{L}u(x_i) = \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j))$$

A related Poisson equation

If the solution to Laplace learning u is roughly constant $u \approx c$, then at labeled nodes $x_1, \dots, x_m$ we can compute

$$\begin{aligned}\mathcal{L}u(x_i) &= \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)) \\ &\approx \sum_{j=1}^{n} w_{ij}(y_i - c) \quad \text{(since } u \approx c\text{)}\end{aligned}$$

A related Poisson equation

If the solution to Laplace learning u is roughly constant $u \approx c$, then at labeled nodes $x_1, \dots, x_m$ we can compute

$$
\begin{aligned}
\mathcal{L}u(x_i) &= \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)) \\
&\approx \sum_{j=1}^{n} w_{ij}(y_i - c) \quad \text{(since } u \approx c\text{)} \\
&= d_i(y_i - c).
\end{aligned}
$$

A related Poisson equation

If the solution to Laplace learning u is roughly constant $u \approx c$, then at labeled nodes $x_1, \ldots, x_m$ we can compute

$$\begin{aligned} \mathcal{L}u(x_i) &= \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)) \\ &\approx \sum_{j=1}^{n} w_{ij}(y_i - c) \quad \text{(since } u \approx c\text{)} \\ &= d_i(y_i - c). \end{aligned}$$

At unlabeled nodes $x_{m+1}, \ldots, x_n$ we have $\mathcal{L}u(x_i) = 0$.

A related Poisson equation

If the solution to Laplace learning u is roughly constant $u \approx c$, then at labeled nodes $x_1, \ldots, x_m$ we can compute

$$\begin{aligned}\mathcal{L}u(x_i) &= \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)) \\ &\approx \sum_{j=1}^{n} w_{ij}(y_i - c) \quad \text{(since } u \approx c\text{)} \\ &= d_i(y_i - c).\end{aligned}$$

At unlabeled nodes $x_{m+1}, \ldots, x_n$ we have $\mathcal{L}u(x_i) = 0$. Thus, u approximately solves

$$\boxed{\mathcal{L}u(x_i) = \sum_{j=1}^{m} d_j(y_j - c)\delta_{ij}, \quad c = \frac{\sum_{j=1}^{n} d_j y_j}{\sum_{j=1}^{n} d_j}.}$$

A related Poisson equation

If the solution to Laplace learning u is roughly constant $u \approx c$, then at labeled nodes $x_1, \ldots, x_m$ we can compute

$$\begin{aligned}\mathcal{L}u(x_i) &= \sum_{j=1}^{n} w_{ij}(u(x_i) - u(x_j)) \\ &\approx \sum_{j=1}^{n} w_{ij}(y_i - c) \quad \text{(since } u \approx c\text{)} \\ &= d_i(y_i - c).\end{aligned}$$

At unlabeled nodes $x_{m+1}, \ldots, x_n$ we have $\mathcal{L}u(x_i) = 0$. Thus, u approximately solves

$$\boxed{\mathcal{L}u(x_i) = \sum_{j=1}^{m} d_j(y_j - c)\delta_{ij}, \quad c = \frac{\sum_{j=1}^{n} d_j y_j}{\sum_{j=1}^{n} d_j}.}$$

Takeaway: At low label rates, there is a connection between hard label constraints, and placing sources and sinks at labels.

Poisson learning

We propose to replace Laplace learning

(8)
$$\begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m, \end{cases}$$

with Poisson learning

$$\mathcal{L}u(x_i) = \sum_{j=1}^{m} (y_j - \overline{y})\delta_{ij} \quad \text{for } i = 1, \dots, n$$

subject to $\sum_{i=1}^{n} d_i u(x_i) = 0$, where $\overline{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$.

Poisson learning

We propose to replace Laplace learning

$$(8) \qquad \begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \leq i \leq n, \\ u(x_i) = y_i, & \text{if } 1 \leq i \leq m, \end{cases}$$

with Poisson learning

$$\mathcal{L}u(x_i) = \sum_{j=1}^{m} (y_j - \overline{y})\delta_{ij} \quad \text{for } i = 1, \dots, n$$

subject to $\sum_{i=1}^{n} d_i u(x_i) = 0$, where $\overline{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$.

In both cases, the label decision is the same:

$$\ell(x_i) = \operatorname*{argmax}_{j \in \{1,\dots,k\}} \{u_j(x)\}.$$

Poisson learning

We propose to replace Laplace learning

$$(9)\qquad \begin{cases} \mathcal{L}u(x_i) = 0, & \text{if } m+1 \le i \le n, \\ u(x_i) = y_i, & \text{if } 1 \le i \le m, \end{cases}$$

with Poisson learning

$$\mathcal{L}u(x_i) = \sum_{j=1}^{m} (y_j - \overline{y})\delta_{ij} \quad \text{for } i = 1, \dots, n$$

subject to $\sum_{i=1}^{n} d_i u(x_i) = 0$, where $\overline{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$.

For Poisson learning, unbalanced class sizes can be incorporated:

$$\ell(x_i) = \operatorname*{argmax}_{j\in\{1,\dots,k\}} \left\{ \frac{p_j}{n_j} u_j(x) \right\},$$

p_j = Fraction of data in class j

n_j = # training examples in class j.

The random walk perspective

Let $X_0^{x_j}, X_1^{x_j}, X_2^{x_j}$ be a random walk on the graph $\mathcal{X}$ starting from $x_j \in \mathcal{X}$, and define

$$u_T(x_i) := \mathbb{E}\left[\sum_{k=0}^{T} \frac{1}{d_i} \sum_{j=1}^{m} (y_j - \overline{y}) \mathbb{1}_{\{X_k^{x_j} = x_i\}}\right], \quad \text{where } \overline{y} = \frac{1}{m}\sum_{j=1}^{m} y_j.$$

The random walk perspective

Let $X_0^{x_j}, X_1^{x_j}, X_2^{x_j}$ be a random walk on the graph $\mathcal{X}$ starting from $x_j \in \mathcal{X}$, and define

$$u_T(x_i) := \mathbb{E}\left[\sum_{k=0}^{T} \frac{1}{d_i} \sum_{j=1}^{m} (y_j - \overline{y}) \mathbb{1}_{\{X_k^{x_j} = x_i\}}\right], \quad \text{where } \overline{y} = \frac{1}{m} \sum_{j=1}^{m} y_j.$$

Theorem (C.-Cook-Thorpe-Slepcev, 2020)

For every $T \geq 0$ we have

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

If the graph G is connected and the Markov chain induced by the random walk is aperiodic, then $u_T \to u$ as $T \to \infty$, where $u : \mathcal{X} \to \mathbb{R}$ is the solution of

$$\mathcal{L}u(x_i) = \sum_{j=1}^{m} (y_j - \overline{y})\delta_{ij} \quad \text{for } i = 1, \ldots, n$$

satisfying $\sum_{i=1}^{n} d_i u(x_i) = 0$.

The variational interpretation

We define the space of weighted mean-zero functions

$$\ell^2_0(\mathcal{X}) = \left\{ u : \mathcal{X} \to \mathbb{R} \,:\, (u)_{\mathcal{X}} = 0 \right\}, \text{ where } (u)_{\mathcal{X}} := \frac{\sum_{i=1}^n d_i u(x_i)}{\sum_{i=1}^n d_i}.$$

The variational interpretation

We define the space of weighted mean-zero functions

$$\ell^2_0(\mathcal{X}) = \left\{ u : \mathcal{X} \to \mathbb{R} \, : \, (u)_{\mathcal{X}} = 0 \right\}, \text{ where } (u)_{\mathcal{X}} := \frac{\sum_{i=1}^n d_i \, u(x_i)}{\sum_{i=1}^n d_i}.$$

Consider the variational problem

(10) $$\min_{u \in \ell^2_0(\mathcal{X})} \left\{ \sum_{i,j=1}^n w_{ij} \left| u(x_i) - u(x_j) \right|^2 - \sum_{j=1}^m (y_j - \overline{y}) \cdot u(x_j) \right\},$$

where $\overline{y} = \frac{1}{m} \sum_{j=1}^m y_j$.

The variational interpretation

We define the space of weighted mean-zero functions

$$\ell_0^2(\mathcal{X}) = \left\{u : \mathcal{X} \to \mathbb{R} \, : \, (u)_{\mathcal{X}} = 0\right\}, \text{ where } (u)_{\mathcal{X}} := \frac{\sum_{i=1}^n d_i u(x_i)}{\sum_{i=1}^n d_i}.$$

Consider the variational problem

$$\min_{u \in \ell_0^2(\mathcal{X})} \left\{ \sum_{i,j=1}^n w_{ij} |u(x_i) - u(x_j)|^2 - \sum_{j=1}^m (y_j - \overline{y}) \cdot u(x_j) \right\}, \tag{10}$$

where $\overline{y} = \frac{1}{m} \sum_{j=1}^m y_j$.

Theorem (C.-Cook-Thorpe-Slepcev, 2020)

Assume the graph is connected. Then there exists a unique solution $u \in \ell_0^2(\mathcal{X})$ of (10), and furthermore, u satisfies the Poisson equation

$$\mathcal{L}u(x_i) = \sum_{j=1}^m (y_j - \overline{y}) \delta_{ij}.$$

Poisson vs Laplace

For Poisson learning we have

$$\min_{u\in\ell^2_0(\mathcal{X})}\left\{\sum_{i,j=1}^n w_{ij}\left|u(x_i)-u(x_j)\right|^2-\sum_{j=1}^m (y_j-c)\cdot u(x_j)\right\}.$$

We compare this with the variational interpretation for Laplace learning is

$$\min_{u\in\ell^2(\mathcal{X})}\left\{\sum_{i,j=1}^n w_{ij}\left|u(x_i)-u(x_j)\right|^2\,:\,u(x_i)=y_i \text{ for } i=1,\dots,m\right\}.$$

Poisson vs Laplace

For Poisson learning we have

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

We compare this with the variational interpretation for Laplace learning is

$$\min_{u\in\ell^2(\mathcal{X})}\left\{\sum_{i,j=1}^n w_{ij}|u(x_i)-u(x_j)|^2\ :\ u(x_i)=y_i \text{ for } i=1,\dots,m\right\}.$$

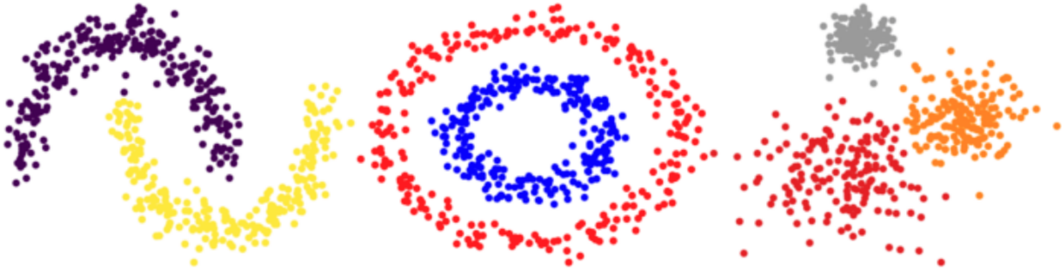
J. Calder, B. Cook, M. Thorpe, and D. Slepčev. **Poisson Learning: Graph based semi-supervised learning at very low label rates.** *International Conference on Machine Learning (ICML)*, 2020.

Outline

GraphLearning Python Package

README.md

Graph-based Clustering and Semi-Supervised Learning



This python package is devoted to efficient implementations of modern graph-based learning algorithms for both semi-supervised learning and clustering. The package implements many popular datasets (currently MNIST, FashionMNIST, cifar-10, and WEBKB) in a way that makes it simple for users to test out new algorithms and rapidly compare against existing methods.

This package reproduces experiments from the paper

Calder, Cook, Thorpe, Slepcev. Poisson Learning: Graph Based Semi-Supervised Learning at Very Low Label Rates., Proceedings of the 37th International Conference on Machine Learning, PMLR 119:1306-1316, 2020.

Installation

Install with

```
pip install graphlearning
```

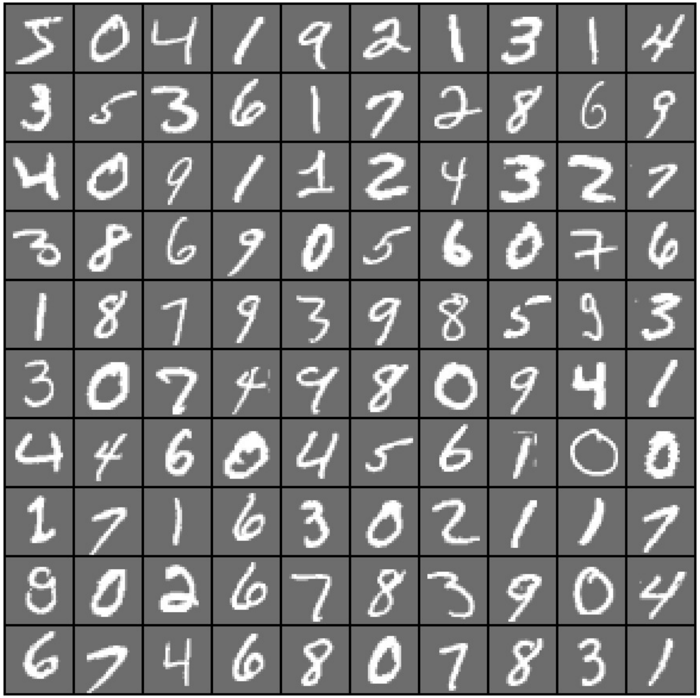
Algorithmic details

Algorithm 1 Poisson Learning

Input: $\mathbf{W}, \mathbf{F}, \mathbf{b}, T$ $\{\mathbf{F} \in \mathbb{R}^{k \times m}$ are label vectors, $\mathbf{b} \in \mathbb{R}^k$ are class sizes.$\}$
Output: $\mathbf{U} \in \mathbb{R}^{n \times k}$
$\mathbf{D} \leftarrow \text{diag}(\mathbf{W}\mathbb{1})$
$\mathbf{L} \leftarrow \mathbf{D} - \mathbf{W}$
$\overline{\mathbf{y}} \leftarrow \frac{1}{m}\mathbf{F}\mathbb{1}$
$\mathbf{B} \leftarrow [\mathbf{F} - \overline{\mathbf{y}}, \textbf{zeros}(k, n - m)]$
$\mathbf{U} \leftarrow \textbf{zeros}(n, k)$
for $i = 1$ **to** T **do**
    $\mathbf{U} \leftarrow \mathbf{U} + \mathbf{D}^{-1}(\mathbf{B}^T - \mathbf{L}\mathbf{U})$
end for
$\mathbf{U} \leftarrow \mathbf{U} \cdot \text{diag}(\mathbf{b}/\overline{\mathbf{y}})$ $\{$Accounts for unbalanced class sizes.$\}$

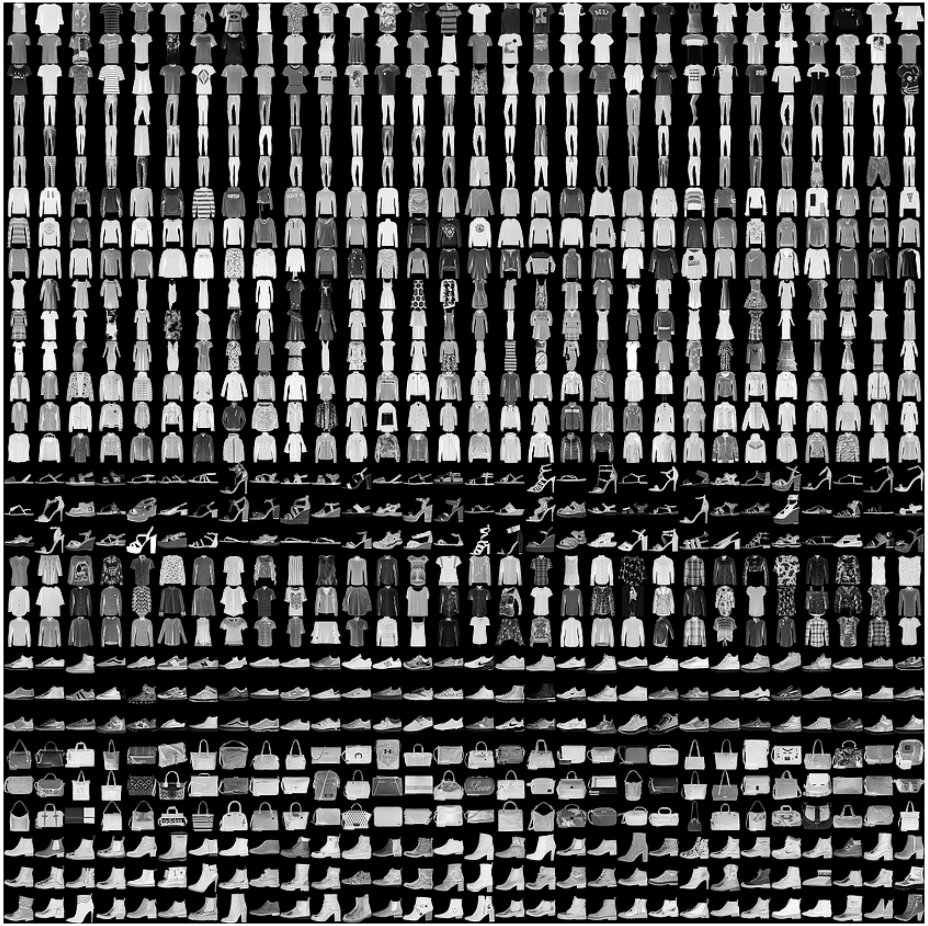
1. We only need about $T = 100$ iterations on MNIST, FashionMNIST, CIFAR-10, to get good results. CPU Time: 8 seconds on CPU, 1 second on GPU.

MNIST (70,000 28×28 pixel images of digits 0-9)



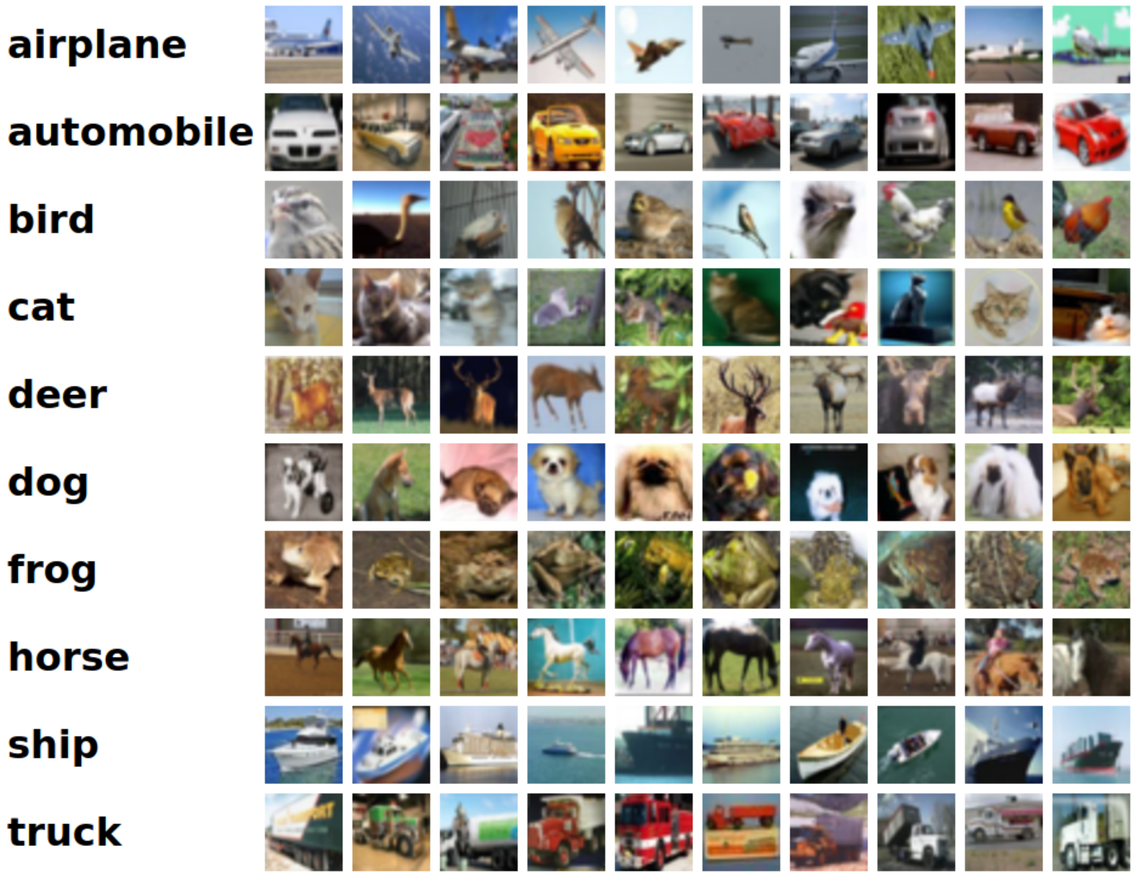
[Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.]

FashionMNIST (70,000 28×28 images of fashion items)



[Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv:1708.07747 (2017).]

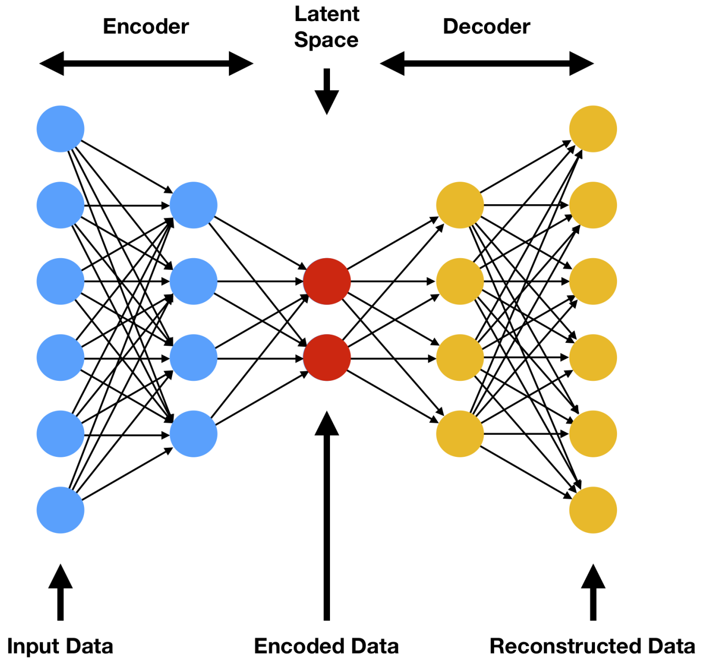
CIFAR-10



[Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009).]

Autoencoders

For each dataset, we build the graph by training autoencoders.



`www.compthree.com`

Autoencoders are "Nonlinear versions of PCA"

Building graphs from autoencoders

For MNIST and FashionMNIST, we use a 4-layer variational autoencoder with 30 latent variables:

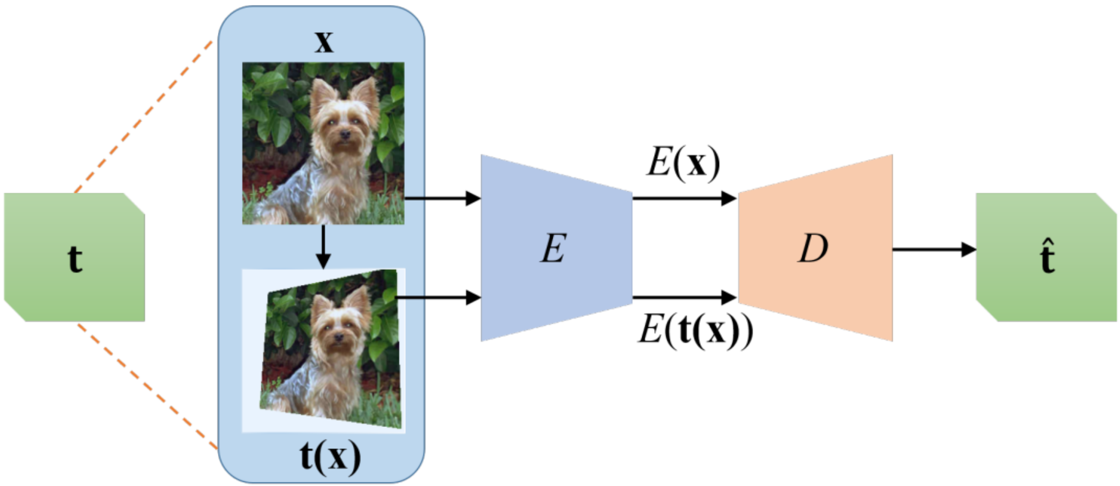
[Kingma and Welling. Auto-encoding variational Bayes. ICML 2014]

Building graphs from autoencoders

For MNIST and FashionMNIST, we use a 4-layer variational autoencoder with 30 latent variables:

[Kingma and Welling. Auto-encoding variational Bayes. ICML 2014]

For CIFAR-10, we use the autoencoding framework from [Zhang et al. AuteEncoding Transformations (AET), CVPR 2019] with 12,288 latent variables.



Building graphs from autoencoders

After training autoencoders, we build a $k=10$ nearest neighbor graphs in the latent space with Gaussian weights

$$w_{ij} = \text{exp}\left(-\frac{4|x_i - x_j|^2}{d_k(x_i)^2}\right),$$

where $d_k(x_i)$ is the distance in the latent space between x_i and its k^{th} nearest neighbor. The weight matrix was then symmetrized by replacing W with $W + W^T$.

Building graphs from autoencoders

After training autoencoders, we build a $k = 10$ nearest neighbor graphs in the latent space with Gaussian weights

$$w_{ij} = \exp\left(-\frac{4|x_i - x_j|^2}{d_k(x_i)^2}\right),$$

where $d_k(x_i)$ is the distance in the latent space between x_i and its k^{th} nearest neighbor. The weight matrix was then symmetrized by replacing W with $W + W^T$.

For CIFAR-10, the latent feature vectors were normalized to unit norm (equivalent to using an angular similarity).

First comparison

We compared against many other graph-based learning algorithms

- Laplace/Label propagation: [Zhu et al., 2003]
- Graph nearest neighbor (using Dijkstra)
- Lazy random walks: [Zhou et al., 2004]
- Mutli-class MBO: [Garcia-Cardona et al., 2014]
- Centered kernel method: [Mai & Couillet, 2018]
- Sparse Label Propagation: [Jung et al., 2016]
- Weighted Nonlocal Laplacian (WNLL): [Shi et al., 2017]
- p-Laplace regularization: [Flores et al. 2019]

MNIST results

Table: Average (standard deviation) classification accuracy over 100 trials.

# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP	16.1 (6.2)	28.2 (10.3)	42.0 (12.4)	57.8 (12.3)	69.5 (12.2)
Nearest Neighbor	65.4 (5.2)	74.2 (3.3)	77.8 (2.6)	80.7 (2.0)	82.1 (2.0)
Random Walk	66.4 (5.3)	76.2 (3.3)	80.0 (2.7)	82.8 (2.3)	84.5 (2.0)
MBO	19.4 (6.2)	29.3 (6.9)	40.2 (7.4)	50.7 (6.0)	59.2 (6.0)
Centered Kernel	19.1 (1.9)	24.2 (2.3)	28.8 (3.4)	32.6 (4.1)	35.6 (4.6)
Sparse Label Prop.	14.0 (5.5)	14.0 (4.0)	14.5 (4.0)	18.0 (5.9)	16.2 (4.2)
WNLL	55.8 (15.2)	82.8 (7.6)	90.5 (3.3)	93.6 (1.5)	94.6 (1.1)
p-Laplace	72.3 (9.1)	86.5 (3.9)	89.7 (1.6)	90.3 (1.6)	91.9 (1.0)
Poisson	**90.2 (4.0)**	**93.6 (1.6)**	**94.5 (1.1)**	**94.9 (0.8)**	**95.3 (0.7)**

FashionMNIST results

Table: Average (standard deviation) classification accuracy over 100 trials.

# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP	18.4 (7.3)	32.5 (8.2)	44.0 (8.6)	52.2 (6.2)	57.9 (6.7)
Nearest Neighbor	46.6 (4.7)	53.5 (3.6)	57.2 (3.0)	59.3 (2.6)	61.1 (2.8)
Random Walk	49.0 (4.4)	55.6 (3.8)	59.4 (3.0)	61.6 (2.5)	63.4 (2.5)
MBO	15.7 (4.1)	20.1 (4.6)	25.7 (4.9)	30.7 (4.9)	34.8 (4.3)
Centered Kernel	11.8 (0.4)	13.1 (0.7)	14.3 (0.8)	15.2 (0.9)	16.3 (1.1)
Sparse Label Prop.	14.1 (3.8)	16.5 (2.0)	13.7 (3.3)	13.8 (3.3)	16.1 (2.5)
WNLL	44.6 (7.1)	59.1 (4.7)	64.7 (3.5)	67.4 (3.3)	70.0 (2.8)
p-Laplace	54.6 (4.0)	57.4 (3.8)	65.4 (2.8)	68.0 (2.9)	68.4 (0.5)
Poisson	**60.8 (4.6)**	**66.1 (3.9)**	**69.6 (2.6)**	**71.2 (2.2)**	**72.4 (2.3)**

Compare to clustering result of 67.2% [McConville et al., 2019]

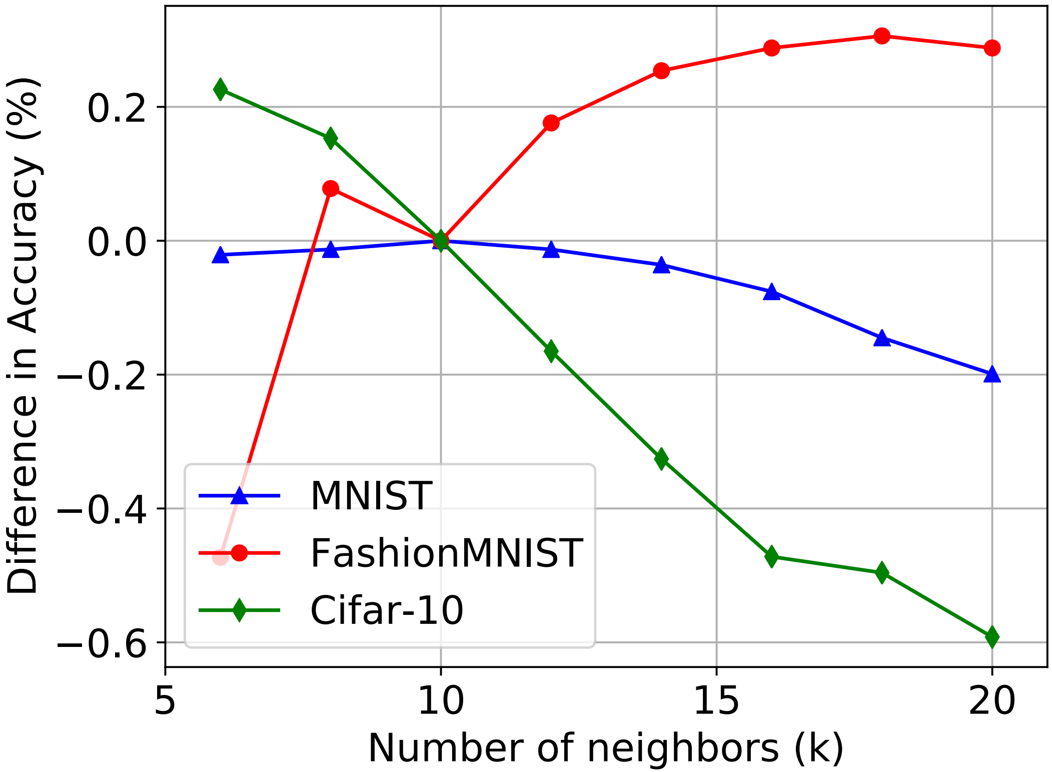
CIFAR-10 results

Table: Average (standard deviation) classification accuracy over 100 trials.

# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP	10.4 (1.3)	11.0 (2.1)	11.6 (2.7)	12.9 (3.9)	14.1 (5.0)
Nearest Neighbor	33.1 (4.3)	37.3 (4.1)	39.7 (3.0)	41.7 (2.8)	43.0 (2.5)
Random Walk	36.4 (4.9)	42.0 (4.4)	45.1 (3.3)	47.5 (2.9)	49.0 (2.6)
MBO	14.2 (4.1)	19.3 (5.2)	24.3 (5.6)	28.5 (5.6)	33.5 (5.7)
Centered Kernel	15.4 (1.6)	16.9 (2.0)	18.8 (2.1)	19.9 (2.0)	21.7 (2.2)
Sparse Label Prop.	11.8 (2.4)	12.3 (2.4)	11.1 (3.3)	14.4 (3.5)	11.0 (2.9)
WNLL	16.6 (5.2)	26.2 (6.8)	33.2 (7.0)	39.0 (6.2)	44.0 (5.5)
p-Laplace	26.0 (6.7)	35.0 (5.4)	42.1 (3.1)	48.1 (2.6)	49.7 (3.8)
Poisson	**40.7 (5.5)**	**46.5 (5.1)**	**49.9 (3.4)**	**52.3 (3.1)**	**53.8 (2.6)**

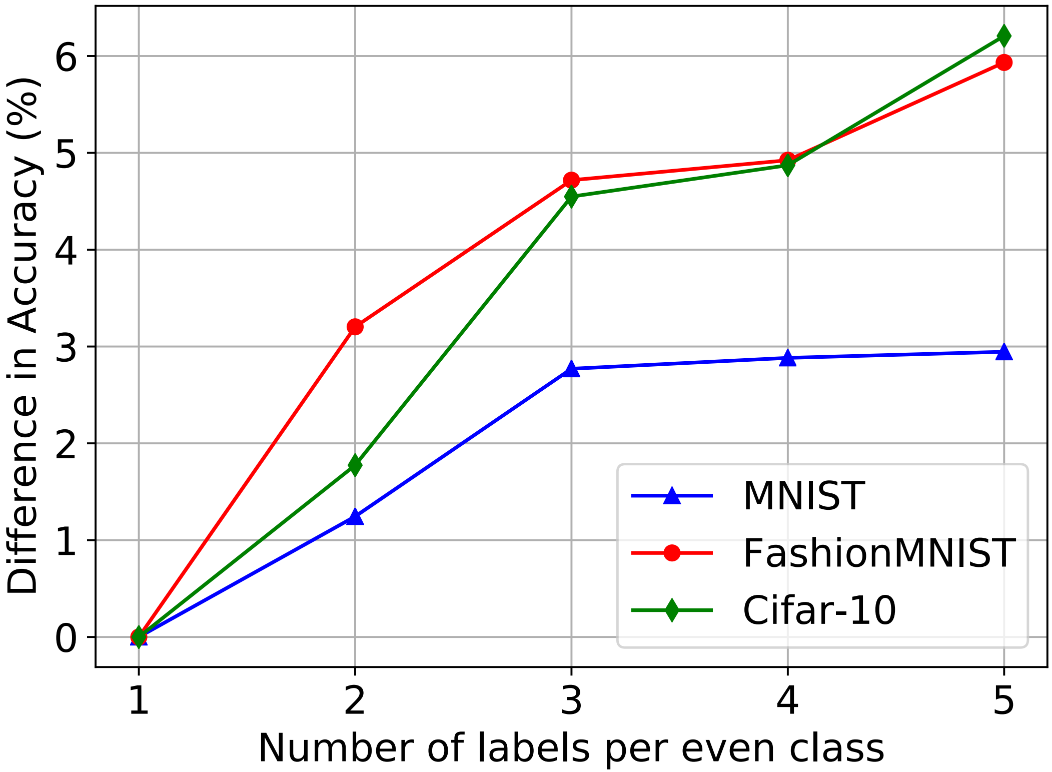
Compare to clustering result of 41.2% [Mukherjee et al., ClusterGAN, CVPR 2019].

Varying number of neighbors k



5 labels per class for all classes.

Unbalanced training data



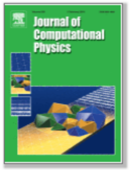
Odd numbered classes got 1 label per class.

Volume constrained semi-supervised learning

ELSEVIER

Journal of Computational Physics

Volume 354, 1 February 2018, Pages 288-310



Auction dynamics: A volume constrained MBO scheme

Matt Jacobs, Ekaterina Merkurjev, Selim Esedoḡlu

Show more

https://doi.org/10.1016/j.jcp.2017.10.036

Classification results can be improved by incorporating prior knowledge of class sizes through volume constraints.

PoissonMBO: Volume constrained Poisson learning

Observation 1: The Poisson learning iteration with a fixed time step

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

is **volume preserving**. That is $(u_{T+1})_{\mathcal{X}} = (u_T)_{\mathcal{X}}$.

PoissonMBO: Volume constrained Poisson learning

Observation 1: The Poisson learning iteration with a fixed time step

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

is **volume preserving**. That is $(u_{T+1})_{\mathcal{X}} = (u_T)_{\mathcal{X}}$.

Observation 2: We can easily perform a volume constrained label decision:

$$\ell(x_i) = \operatorname*{argmax}_{j\in\{1,\dots,k\}} \left\{ s_j\, u_j(x) \right\}.$$

We adjust the weights s_j to grow/shrink each region to achieve the correct class sizes.

- Equivalent to re-weighting the point sources in Poisson learning.

PoissonMBO Algorithm

Algorithm 2 PoissonMBO

Input: $\mathbf{W}, \mathbf{F}, N_{inner}, N_{outer}, \mathbf{b}, \mu, T > 0$
Output: $\mathbf{U} \in \mathbb{R}^{n\times k}$
$\mathbf{U} \leftarrow$ PoissonLearning$(\mathbf{W}, \mathbf{F}, \mathbf{b}, T)$
$\mathrm{d}t \leftarrow 1/\max_{1\leq i\leq n} \mathbf{D}_{ii}$
for $i = 1$ **to** N_{outer} **do**
    for $j = 1$ **to** N_{inner} **do**
        $\mathbf{U} \leftarrow \mathbf{U} - \mathrm{d}t(\mathbf{L}\mathbf{U} - \mu\mathbf{B}^T)$
    end for
    $\mathbf{U} \leftarrow$ VolumeConstrainedLabelProjection$(\mathbf{U}, \mathbf{b})$
end for

Named after the Merriman-Bence-Osher (MBO) scheme for curvature motion, which has been used before in graph-based learning [Garcia, et al., 2014, Jacobs et al., 2018].

MNIST results

Table: Average (standard deviation) classification accuracy over 100 trials.

# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP	16.1 (6.2)	28.2 (10.3)	42.0 (12.4)	57.8 (12.3)	69.5 (12.2)
WNLL	55.8 (15.2)	82.8 (7.6)	90.5 (3.3)	93.6 (1.5)	94.6 (1.1)
p-Laplace	72.3 (9.1)	86.5 (3.9)	89.7 (1.6)	90.3 (1.6)	91.9 (1.0)
VolumeMBO	89.9 (7.3)	95.6 (1.9)	96.2 (1.2)	96.6 (0.6)	96.7 (0.6)
Poisson	90.2 (4.0)	93.6 (1.6)	94.5 (1.1)	94.9 (0.8)	95.3 (0.7)
PoissonMBO	**96.5 (2.6)**	**97.2 (0.1)**	**97.2 (0.1)**	**97.2 (0.1)**	**97.2 (0.1)**
# Labels per class	**10**	**20**	**40**	**80**	**160**
Laplace/LP	91.3 (3.7)	95.8 (0.6)	96.5 (0.2)	96.8 (0.1)	97.0 (0.1)
WNLL	95.6 (0.5)	96.1 (0.3)	96.3 (0.2)	96.4 (0.1)	96.3 (0.1)
p-Laplace	94.0 (0.8)	95.1 (0.4)	95.5 (0.1)	96.0 (0.2)	96.2 (0.1)
VolumeMBO	96.9 (0.2)	97.0 (0.1)	97.1 (0.1)	97.2 (0.1)	**97.3 (0.1)**
Poisson	95.9 (0.4)	96.3 (0.3)	96.6 (0.2)	96.8 (0.1)	96.9 (0.1)
PoissonMBO	**97.2 (0.1)**	**97.2 (0.1)**	**97.2 (0.1)**	**97.2 (0.1)**	97.2 (0.1)

FashionMNIST results

Table: Average (standard deviation) classification accuracy over 100 trials.

# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP	18.4 (7.3)	32.5 (8.2)	44.0 (8.6)	52.2 (6.2)	57.9 (6.7)
WNLL	44.6 (7.1)	59.1 (4.7)	64.7 (3.5)	67.4 (3.3)	70.0 (2.8)
p-Laplace	54.6 (4.0)	57.4 (3.8)	65.4 (2.8)	68.0 (2.9)	68.4 (0.5)
VolumeMBO	54.7 (5.2)	61.7 (4.4)	66.1 (3.3)	68.5 (2.8)	70.1 (2.8)
Poisson	60.8 (4.6)	66.1 (3.9)	69.6 (2.6)	71.2 (2.2)	72.4 (2.3)
PoissonMBO	**62.0 (5.7)**	**67.2 (4.8)**	**70.4 (2.9)**	**72.1 (2.5)**	**73.1 (2.7)**
# Labels per class	**10**	**20**	**40**	**80**	**160**
Laplace/LP	70.6 (3.1)	76.5 (1.4)	79.2 (0.7)	80.9 (0.5)	**82.3 (0.3)**
WNLL	74.4 (1.6)	77.6 (1.1)	79.4 (0.6)	80.6 (0.4)	81.5 (0.3)
p-Laplace	73.0 (0.9)	76.2 (0.8)	78.0 (0.3)	79.7 (0.5)	80.9 (0.3)
VolumeMBO	74.4 (1.5)	77.4 (1.0)	79.5 (0.7)	**81.0 (0.5)**	82.1 (0.3)
Poisson	75.2 (1.5)	77.3 (1.1)	78.8 (0.7)	79.9 (0.6)	80.7 (0.5)
PoissonMBO	**76.1 (1.4)**	**78.2 (1.1)**	**79.5 (0.7)**	80.7 (0.6)	81.6 (0.5)

CIFAR-10 results

Table: Average (standard deviation) classification accuracy over 100 trials.

# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP	10.4 (1.3)	11.0 (2.1)	11.6 (2.7)	12.9 (3.9)	14.1 (5.0)
WNLL	16.6 (5.2)	26.2 (6.8)	33.2 (7.0)	39.0 (6.2)	44.0 (5.5)
p-Laplace	26.0 (6.7)	35.0 (5.4)	42.1 (3.1)	48.1 (2.6)	49.7 (3.8)
VolumeMBO	38.0 (7.2)	46.4 (7.2)	50.1 (5.7)	53.3 (4.4)	55.3 (3.8)
Poisson	40.7 (5.5)	46.5 (5.1)	49.9 (3.4)	52.3 (3.1)	53.8 (2.6)
PoissonMBO	**41.8 (6.5)**	**50.2 (6.0)**	**53.5 (4.4)**	**56.5 (3.5)**	**57.9 (3.2)**
# Labels per class	**10**	**20**	**40**	**80**	**160**
Laplace/LP	21.8 (7.4)	38.6 (8.2)	54.8 (4.4)	62.7 (1.4)	66.6 (0.7)
WNLL	54.0 (2.8)	60.3 (1.6)	64.2 (0.7)	66.6 (0.6)	68.2 (0.4)
p-Laplace	56.4 (1.8)	60.4 (1.2)	63.8 (0.6)	66.3 (0.6)	68.7 (0.3)
VolumeMBO	59.2 (3.2)	61.8 (2.0)	63.6 (1.4)	64.5 (1.3)	65.8 (0.9)
Poisson	58.3 (1.7)	61.5 (1.3)	63.8 (0.8)	65.6 (0.6)	67.3 (0.4)
PoissonMBO	**61.8 (2.2)**	**64.5 (1.6)**	**66.9 (0.8)**	**68.7 (0.6)**	**70.3 (0.4)**

Outline

The continuum perspective

Continuum limits can help explain why Poisson learning works for low label rates.

The continuum perspective

Continuum limits can help explain why Poisson learning works for low label rates.

Manifold assumption: Let $x_1, \dots, x_n$ be a sequence of i.i.d. random variables drawn from a d-dimensional compact, closed, and connected manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$, where $d \ll D$. We assume the random variables have a density $\rho : \mathcal{M} \to \mathbb{R}$ with respect to the volume form $Vol_{\mathcal{M}}$ on the manifold.

The continuum perspective

Continuum limits can help explain why Poisson learning works for low label rates.

Manifold assumption: Let $x_1, \ldots, x_n$ be a sequence of i.i.d. random variables drawn from a d-dimensional compact, closed, and connected manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$, where $d \ll D$. We assume the random variables have a density $\rho : \mathcal{M} \to \mathbb{R}$ with respect to the volume form $Vol_{\mathcal{M}}$ on the manifold.

Fix a finite set of points $\Gamma \subset \mathcal{M}$. The vertices of the random geometric graph are

$$\mathcal{X}_n := \underbrace{\{x_1, \ldots, x_n\}}_{\text{Unlabeled}} \cup \underbrace{\Gamma}_{\text{Labeled}} .$$

We define the edge weights in the graph by

$$w_{xy} = \eta_\varepsilon\left(|x - y|\right),$$

where $\eta : [0, \infty) \to [0, \infty)$ is smooth with compact support, and $\eta_\varepsilon(t) = \frac{1}{\varepsilon^d}\eta\left(\frac{t}{\varepsilon}\right)$.

The continuum perspective

The normalized graph Laplacian is given by

$$\mathcal{L}_{n,\varepsilon} u(x) = \frac{2}{\sigma_\eta n \varepsilon^2} \sum_{y \in \mathcal{X}_n} \eta_\varepsilon(|x-y|)(u(x) - u(y)),$$

where $\sigma_\eta = \int_{\mathbb{R}^d} |z_1|^2 \eta(|z|)\, dz$.

Using the normalized graph Laplacian, the Poisson learning problem is

$$\mathcal{L}_{n,\varepsilon} u_{n,\varepsilon}(x) = n \sum_{y \in \Gamma} (g(y) - c)\delta_{x=y} \quad \text{for } x \in \mathcal{X}_n, \tag{11}$$

where $c = \frac{1}{|\Gamma|} \sum_{x \in \Gamma} g(x)$.

The continuum perspective

The normalized graph Laplacian is given by

$$\mathcal{L}_{n,\varepsilon} u(x) = \frac{2}{\sigma_\eta n \varepsilon^2} \sum_{y \in \mathcal{X}_n} \eta_\varepsilon(|x-y|)(u(x) - u(y)),$$

where $\sigma_\eta = \int_{\mathbb{R}^d} |z_1|^2 \eta(|z|)\, dz$.

Using the normalized graph Laplacian, the Poisson learning problem is

(11) $$\mathcal{L}_{n,\varepsilon} u_{n,\varepsilon}(x) = n \sum_{y \in \Gamma} (g(y) - c)\delta_{x=y} \quad \text{for } x \in \mathcal{X}_n,$$

where $c = \frac{1}{|\Gamma|} \sum_{x \in \Gamma} g(x)$.

Question: What can we say about $u_{n,\varepsilon}$ as $n \to \infty$ and $\varepsilon \to 0$? Is it stable, and does it converge to a well-posed continuum limit?

The continuum perspective

Conjecture

Assume ρ is smooth. Assume that $n \to \infty$ and $\varepsilon = \varepsilon_n \to 0$ so that

$$\lim_{n\to\infty} \frac{n\varepsilon^{d+2}}{\log n} = \infty.$$

Then with probability one

$$\lim_{n\to\infty} \max_{\substack{x\in\mathcal{X}_n \\ \mathrm{dist}(x,\Gamma)>\delta}} |u_{n,\varepsilon}(x) - u(x)| = 0$$

for all $\delta > 0$, where $u \in C^\infty(\mathcal{M} \setminus \Gamma)$ is the solution of the Poisson equation

$$-\,\mathrm{div}\left(\rho^2 \nabla u\right) = \sum_{y\in\Gamma} (g(y) - c)\delta_y \quad \text{on } \mathcal{M}, \tag{12}$$

where $c = \frac{1}{|\Gamma|}\sum_{x\in\Gamma} g(x)$.

Python Notebook: .ipynb

References:

1. J. Calder, D. Slepčev, D., and M. Thorpe. **Rates of convergence for Laplacian semi-supervised learning with low label rates.** *arXiv:2006.02765*, 2020.
2. J. Calder, B. Cook, M. Thorpe, and D. Slepčev. **Poisson Learning: Graph based semi-supervised learning at very low label rates.** *International Conference on Machine Learning (ICML)*, 2020.

Code:

`https://github.com/jwcalder/GraphLearning`