Poisson Learning: Graph Based Semi-Supervised Learning At Very Low Label Rates

Jeff Calder [1] Brendan Cook [1] Matthew Thorpe [2] Dejan Slepčev [3]

Abstract

We propose a new framework, called *Poisson learning*, for graph based semi-supervised learning at very low label rates. Poisson learning is motivated by the need to address the degeneracy of Laplacian semi-supervised learning in this regime. The method replaces the assignment of label values at training points with the placement of sources and sinks, and solves the resulting Poisson equation on the graph. The outcomes are provably more stable and informative than those of Laplacian learning. Poisson learning is efficient and simple to implement, and we present numerical experiments showing the method is superior to other recent approaches to semi-supervised learning at low label rates on MNIST, FashionMNIST, and Cifar-10. We also propose a graph-cut enhancement of Poisson learning, called *Poisson MBO*, that gives higher accuracy and can incorporate prior knowledge of relative class sizes.

1. Introduction

Semi-supervised learning uses both labeled and unlabeled data in learning tasks. For problems where very few labels are available, geometric or topological structure in unlabeled data can be used to greatly improve the performance of classification, regression, or clustering algorithms. One of the most widely used methods in graph-based semi-supervised learning is Laplace learning, originally proposed in (Zhu et al., 2003), which seeks a graph harmonic function that extends the labels. Laplace learning, and variants thereof, have been widely applied in semi-supervised learning (Zhou et al., 2005; 2004a;b; Ando & Zhang, 2007) and manifold ranking (He et al., 2004; Yang et al., 2013; Xu et al., 2011), among many other problems.

This paper is concerned with graph-based semi-supervised learning at very low label rates. In this setting, it has been observed that Laplace learning can give very poor classification results (Nadler et al., 2009; El Alaoui et al., 2016). The poor results are often attributed to the fact that the solutions develop localized spikes near the labeled points and are almost constant far from the labeled points. In particular, label values are not propagated well by the Laplacian learning approach. To address this issue, recent work has suggested to consider p-Laplace learning (El Alaoui et al., 2016). Several works have rigorously studied p-Laplace regularization with few labels (Slepčev & Thorpe, 2019; Calder, 2018; 2019), and recent numerical results show that $p > 2$ is superior to Laplace learning at low label rates (Flores et al., 2019). The case of $p = \infty$ is called Lipschitz learning (Kyng et al., 2015), which seeks the absolutely minimal Lipschitz extension of the training data. Other methods to address low label rate problems include higher order Laplacian regularization (Zhou & Belkin, 2011) and spectral cutoffs (Belkin & Niyogi, 2002).

While p-Laplace learning improves upon Laplace learning, the method is more computationally burdensome than Laplace learning, since the optimality conditions are nonlinear. Other recent approaches have aimed to re-weight the graph more heavily near labels, in order to give them wider influence when the labeling rate is very low. One way to re-weight the graph is the Weighted Nonlocal Laplacian (WNLL) (Shi et al., 2017), which amplifies the weights of edges directly connected to labeled nodes. The WNLL achieves better results at moderately low label rates, but still performs poorly at very low label rates (Flores et al., 2019). To address this, (Calder & Slepčev, 2019) proposed the Properly Weighted Laplacian, which re-weights the graph in a way that is well-posed at arbitrarily low label rates.

Much of the recent work on low label rate problems has focused on formulating and implementing new learning approaches that are well-posed with few labels. The exact nature of the degeneracy in Laplace learning, and the question of how the tails of the spikes propagate label information, has not been studied and is still poorly understood.

[1]School of Mathematics, University of Minnesota, Minneapolis, USA. [2]Department of Mathematics, University of Manchester, Manchester, UK. [3]Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, USA.. Correspondence to: Jeff Calder <jwcalder@umn.edu>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

For some problems, the performance of Laplacian learning is good (Zhu et al., 2003), while for other problems it is catastrophic, yielding very poor classification results similar to random guessing (Shi et al., 2017; Flores et al., 2019).

In this paper, we carefully analyze Laplace learning at very low label rates, and we discover that *nearly all* of the degeneracy of Laplace learning is due to a large constant bias in the solution of the Laplace equation that is present only at low label rates. In order to overcome this problem we introduce a new algorithm, we call *Poisson learning*, that gives very good classification performance down to extremely low label rates. We give a random walk interpretation of Poisson learning that shows how the method uses information from the random walkers only *before* they reach the mixing time of the random walk and forget their initial condition. We also propose a graph-cut enhancement of Poisson learning, called *Poisson MBO*, that can incorporate knowledge about class sizes, and further improves the class-label accuracy.

The rest of the paper is organized as follows. In Section 2 we first briefly introduce Poisson learning, and then provide a detailed motivation for the algorithm and a theoretical analysis from both the variational and random walk perspectives. The Poisson MBO approach is presented in Section 2.4. In Section 3 we present the step-by-step algorithms and discuss implementation details for the Poisson and Poisson MBO algorithms. In Section 4 we present numerical experiments with semi-supervised classification on the MNIST, FashionMNIST, and Cifar-10 datasets. The proofs of the results are available in the supplementary materials.

2. Poisson learning

Let $X = \{x_1, x_2, \dots, x_n\}$ denote the vertices of a graph with edge weights $w_{ij} \geq 0$ between x_i and x_j. We assume the graph is symmetric, so $w_{ij} = w_{ji}$. We define the degree $d_i = \sum_{j=1}^n w_{ij}$. For a multi-class classification problem with k classes, we let the standard basis vector $\mathbf{e}_i \in \mathbb{R}^k$ represent the i^{th} class. We assume the first m vertices $x_1, x_2, \dots, x_m$ are given labels $y_1, y_2, \dots, y_m \in \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\}$, where $m < n$. The task of graph-based semi-supervised learning is to extend the labels to the rest of the vertices $x_{m+1}, x_{m+2}, \dots, x_n$.

The well-known Laplace learning algorithm (Zhu et al., 2003) extends the labels by solving the problem

$$\left.\begin{aligned} \mathcal{L}u(x_i) &= 0, && \text{if } m+1 \leq i \leq n, \\ u(x_i) &= y_i, && \text{if } 1 \leq i \leq m, \end{aligned}\right\} \tag{2.1}$$

where $\mathcal{L}$ is the unnormalized graph Laplacian given by

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

Here, $u : X \to \mathbb{R}^k$ and we write the components of u as $u(x_i) = (u_1(x_i), u_2(x_i), \dots, u_k(x_i))$. The label decision for vertex x_i is determined by the largest component of $u(x_i)$

$$\ell(x_i) = \underset{j \in \{1,\dots,k\}}{\arg\max} \{u_j(x)\}. \tag{2.2}$$

We note that Laplace learning is also called *label propagation (LP)* (Zhu et al., 2005), since the Laplace equation (2.1) can be solved by repeatedly replacing $u(x_i)$ with the weighted average of its neighbors, which can be viewed as dynamically propagating labels.

At very low label rates, we propose to replace the problem (2.1) by *Poisson learning*: Let $\overline{y} = \frac{1}{m}\sum_{j=1}^m y_j$ be the average label vector and let $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$. One computes the solution of the Poisson equation

$$\mathcal{L}u(x_i) = \sum_{j=1}^m (y_j - \overline{y})\delta_{ij} \quad \text{for } i = 1, \dots, n \tag{2.3}$$

satisfying $\sum_{i=1}^n d_i u(x_i) = 0$. While the label decision can be taken to be the same as (2.2), it is also simple to account for unbalanced classes or training data with the modified label decision

$$\ell(x_i) = \underset{j \in \{1,\dots,k\}}{\arg\max} \{s_j u_j(x)\}, \tag{2.4}$$

where $s_j = b_j/(\overline{y} \cdot \mathbf{e}_j)$ and b_j is the fraction of data belonging to class j. We explain this label decision in Remark 2.2. The Poisson equation (2.3) can be solved efficiently with a simple iteration given in Algorithm 1.

Technically speaking, in Laplace learning, the labels are imposed as boundary conditions in a Laplace equation, while in Poisson learning, the labels appears as a source term in a graph Poisson equation. In the sections below, we explain why Poisson learning is a good idea for problems with very few labels. In particular, we give random walk and variational interpretations of Poisson learning, and we illustrate how Poisson learning arises as the low label rate limit of Laplace learning.

2.1. Random walk interpretation

We present a random walk interpretation of Poisson learning and compare to the random walk interpretation of Laplace learning to explain its poor performance at low label rates. We note that Laplace learning works very well in practice for semi-supervised learning problems with a moderate amount of labeled data. For example, on the MNIST dataset we obtained around 95% accuracy at 16 labels per class (0.23% label rate). However, at very low label rates the performance is poor. At 1 label per class, we find the average performance is around 16% accuracy. This phenomenon has been observed in other works recently (Nadler et al., 2009; El Alaoui et al., 2016). However, a clear understanding of the issues with Laplace learning at low label rates was lacking. The clearest understanding of this phenomenon comes from the random walk interpretation, and this leads directly to the foundations for Poisson learning.

Let $x \in X$ and let $X_0^x, X_1^x, X_2^x, \dots$ be a random walk on X starting at $X_0^x = x$ with transition probabilities

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

Let u be the solution of the *Laplace* learning problem (2.1). Define the stopping time to be the first time the walk hits a label, that is

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

Let $i_\tau \leq m$ denote the index of the point X_τ^x, so $X_\tau^x = x_{i_\tau}$. Then, by Doob's optimal stopping theorem, we have

$$u(x) = \mathbb{E}[y_{i_\tau}]. \tag{2.5}$$

This gives the standard representation formula for the solution of Laplace learning (2.1). The interpretation is that we release a random walker from x and let it walk until it hits a labeled vertex and then record that label. We average over many random walkers to get the value of $u(x)$.

When there are insufficiently many labels, the stopping time τ is so large that we have passed the *mixing time* of the random walk, and the distribution of X_τ^x is very close to the invariant distribution of the random walk $\pi(x_i) = d_i / \sum_i d_i$. In this case, (2.5) gives that

$$u(x) \approx \frac{\sum_{i=1}^m d_i y_i}{\sum_{j=1}^m d_j} =: \overline{y}_{\mathrm{w}}. \tag{2.6}$$

Of course, the function u is not exactly constant, and instead it is approximately constant with sharp spikes at the labeled vertices; see Figure 1. Previous work has proved rigorously that Laplace learning degenerates in this way at low label rates (Slepčev & Thorpe, 2019; Calder, 2018).

It is important to note that the constant vector $\overline{y}_{\mathrm{w}}$ depends only on the degrees of the *labeled* nodes in the graph, which is very sensitive to local graph structure. When Laplace learning returns a nearly constant label function, it can be catastrophic for classification, since most datapoints are assigned the same label. This explains the 16% accuracy in the MNIST experiment described above.

In contrast, the Poisson equation (2.3) has a random walk interpretation that involves the Green's function for a random walk, and is in some sense dual to Laplace learning. The source term on the right hand side of (2.3) represents random walkers being released from labeled points and exploring the graph, while carrying their label information

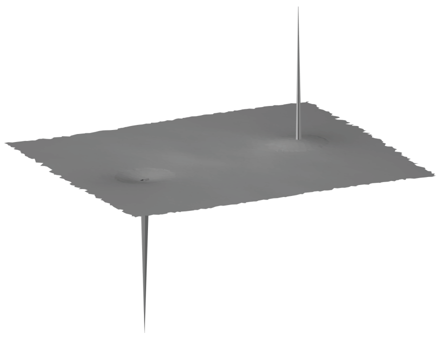


Figure 1. Demonstration of spikes in Laplace learning. The graph consists of $n = 10^4$ independent uniform random variables on $[0,1]^2$ and two points are given labels of 0 and 1. Most values of the solution u of Laplace learning are very close to 0.5.

with them. For $T > 0$ let us define

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

Essentially, each time the random walk starting from x_j, denoted $X_k^{x_j}$, visits x_i we record the label y_j and sum this over all visits from $0 \leq k \leq T$. For short time T, this quantity is meaningful, but since the random walk is recurrent (it is a finite graph), $w_T(x_i) \to \infty$ as $T \to \infty$. If we normalize by $1/T$, we still have the same issue as with Laplace learning, where we are only measuring the invariant distribution. Indeed, we note that

$$w_T(x_i) = \sum_{k=0}^{T} \sum_{j=1}^{m} y_j \mathbb{P}(X_k^{x_j} = x_i)$$

$$\text{and} \quad \lim_{k \to \infty} \mathbb{P}(X_k^{x_j} = x_i) = \frac{d_i}{\sum_{i=1}^n d_i}.$$

Therefore, when k is large we have

$$\sum_{j=1}^{m} y_j \mathbb{P}(X_k^{x_j} = x_i) \approx \frac{d_i}{\sum_{i=1}^n d_i} \sum_{j=1}^{m} y_j,$$

and so the tail of the sum defining $w_T(x_i)$ is recording a blind average of labels.

The discussion above suggests that we should subtract off this average tail behavior from w_T, so that we only record the *short-time* behavior of the random walk, before the mixing time is reached. We also normalize by d_i, which leads us to define

$$u_T(x_i) = \mathbb{E}\left[\sum_{k=0}^{T} \frac{1}{d_i} \sum_{j=1}^{m} (y_j - \overline{y}) \mathbb{1}_{\{X_k^{x_j} = x_i\}}\right], \tag{2.7}$$

where $\overline{y} = \frac{1}{m}\sum_{j=1}^m y_j$. It turns out that as $T \to \infty$, the function $u_T(x_i)$ converges to the solution of (2.3).

Theorem 2.1. *For every $T \geq 0$ we have*

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

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 is the unique solution of the Poisson equation (2.3) *satisfying $\sum_{i=1}^n d_i u(x_i) = 0$.*

Theorem 2.1 gives the foundation for Poisson learning through the random walk perspective, and in fact, it also gives a numerical method for computing the solution (see Algorithm 1).

Remark 2.2. The representation formula (2.7) for the solution of Poisson learning (2.3) shows that the solution u is a *linear* function of the label vectors $y_1, \dots, y_m$. That is, for any $A \in \mathbb{R}^{k\times k}$, the solution $u_A : X \to \mathbb{R}^k$ of

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

satisfying $\sum_{i=1}^n d_i u_A(x_i) = 0$ is exactly $u_A = Au$, where u is the solution of (2.3). This shows that any reweighting of the point sources, by which we mean $y_j \mapsto Ay_j$, is equivalent to reweighting the solution by $u \mapsto Au$.

If we set $A = \mathrm{diag}(s_1, \dots, s_k)$, then Au corresponds to multiplying the point sources for class i by the weight s_i. We can use this reweighting to account for unbalanced classes, or a discrepancy between the balancing of training and testing data, in the following way. Let n_j be the number of training examples from class j, and let b_j denote the true fraction of data in class j. We can choose s_j so that $n_j s_j = b_j$ to ensure that the mass of the point sources for each class, weighted by s_j, is proportional to the true fraction of data in that class. Since n_j is proportional to $\overline{y} \cdot \mathbf{e}_j$, this explains our modified label decision (2.4).

2.2. Variational interpretation

We can also interpret Poisson learning (2.3) as a gradient regularized variational problem. Before proceeding, we briefly review some facts about calculus on graphs. Let $\ell^2(X)$ denote the space of functions $u : X \to \mathbb{R}^k$ equipped with the inner product

$$(u, v)_{\ell^2(X)} = \sum_{i=1}^n u(x_i) \cdot v(x_i).$$

This induces a norm $\|u\|^2_{\ell^2(X)} = (u, u)_{\ell^2(X)}$. We also define the space of mean-zero functions

$$\ell^2_0(X) = \left\{ u \in \ell^2(X) \, : \, \sum_{i=1}^n d_i u(x_i) = 0 \right\}$$

We define a *vector field* on the graph to be an *antisymmetric* function $V : X^2 \to \mathbb{R}^k$ (i.e., $V(x_i, x_j) = -V(x_j, x_i)$). The *gradient* of a function $u \in \ell^2(X)$, denoted for simplicity as ∇u, is defined to be the vector field

$$\nabla u(x_i, x_j) = u(x_j) - u(x_i).$$

The inner product between vector fields V and W is

$$(V, W)_{\ell^2(X^2)} = \frac{1}{2}\sum_{i,j=1}^n w_{ij} V(x_i, x_j) \cdot W(x_i, x_j).$$

and the norm of V is $\|V\|^2_{\ell^2(X^2)} = (V, V)_{\ell^2(X^2)}$.

We now consider the variational problem

$$\min_{u \in \ell^2_0(X)} \left\{ \frac{1}{2}\|\nabla u\|^2_{\ell^2(X^2)} - \sum_{j=1}^m (y_j - \overline{y}) \cdot u(x_j) \right\}. \quad (2.8)$$

The following theorem makes the connection between Poisson learning and the variational problem (2.8).

Theorem 2.3. *Assume G is connected. Then there exists a unique minimizer $u \in \ell^2_0(X)$ of* (2.8)*, and u satisfies the Poisson learning equation* (2.3).

Theorem 2.3 shows that the Poisson learning equation (2.3) arises as the necessary conditions for a gradient regularized variational problem with an affine loss function. We contrast this with the solution of Laplace learning (2.1), which is the minimizer of the variational problem

$$\min_{u \in \ell^2(X)} \left\{ \|\nabla u\|^2_{\ell^2(X^2)} \, : \, u(x_i) = y_i, \, 1 \leq i \leq m \right\}. \quad (2.9)$$

Thus, while both Laplace and Poisson learning are gradient regularized variational problems, the key difference is how each algorithm handles the labeled data; Laplace learning enforces hard label constraints while Poisson learning adds an affine loss function to the energy. Of course, many variants of Laplace learning have been proposed with various types of soft label constraints in place of hard constraints. These variations perform similarly poorly to Laplace learning at low label rates, and the key feature of Poisson learning is the *affine* loss function that can be easily centered.

We note that it would be natural to add a weight $\mu > 0$ to one of the terms in (2.8) to trade-off the importance of the two terms in the variational problem. However, as we show in the supplementary material (see Lemma A.3), this weight would have no effect on the final label decision. We also mention that the variational problem (2.8) has a natural ℓ^p generalization that we also explore in the supplementary material (Section A.2).

2.3. Laplace learning at low label rates

Finally, to further motivate Poisson learning, we connect Poisson learning with the limit of Laplace learning at very low label rates. At low label rates, the solution of Laplace learning (2.1) concentrates around a constant $\overline{y}_{\mathrm{w}}$, for which we gave an interpretation of via random walks in Section 2.1. Near labeled nodes, u has sharp spikes (recall Figure 1) in order to attain the labels. From the variational perspective, the constant function has zero cost, and the cost of spikes is very small, so this configuration is less expensive than continuously attaining the labels $u(x_i) = y_i$.

A natural question concerns whether the spikes in Laplace learning contain useful information, or whether they are too localized and do not propagate information well. To test this, we changed the label decision (2.2) in the MNIST experiment described in Section 2.1 to subtract off the tail constant $\overline{y}_{\mathrm{w}}$ identified in (2.6)

$$\ell(x_i) = \operatorname*{arg\,max}_{j\in\{1,\dots,k\}} \{u_j(x) - \overline{y}_{\mathrm{w}} \cdot \mathbf{e}_j\}.$$

where $\overline{y}_{\mathrm{w}}$ is defined in (2.6). Thus, we are centering the function u at zero. At 1 label per class (10 labeled images total), the accuracy improved from 16% to 85.9%! Hence, the difference $u - \overline{y}_{\mathrm{w}}$ contains enough information to make informed classification decisions, and therefore the spikes contain useful information.

This indicates that much of the poor performance of Laplace learning can be explained by a large shift bias that occurs only at very low label rates. Fixing this seems as simple as applying the appropriate shift before making a decision on a label, but this does not lead to a well-grounded method, since the shifted function $u - \overline{y}_{\mathrm{w}}$ is exactly the graph-harmonic extension of the shifted labels $y_j - \overline{y}_{\mathrm{w}}$. Why should we have to use a harmonic extension of the *wrong labels* in order to achieve a better result? On the other hand, Poisson learning, which we introduced above, provides an intuitive and well-grounded way of fixing the shift bias in Laplace learning.

To see the connection to Poisson learning, let us assume the solution u of the Laplace learning equation (2.1) is nearly equal to the constant vector $\overline{y}_{\mathrm{w}} \in \mathbb{R}^k$ from (2.6) at all unlabeled points $x_{m+1}, \dots, x_n$. For any *labeled* node x_i with $i = 1, \dots, m$ we can compute (assuming $w_{ij} = 0$ for all $j \in \{1, \dots, m\}$) that

$$\begin{aligned}\mathcal{L}u(x_i) &= \sum_{j=1}^n w_{ij}(u(x_i) - u(x_j)) \\ &\approx \sum_{j=m+1}^n w_{ij}(y_i - \overline{y}_{\mathrm{w}}) = d_i(y_i - \overline{y}_{\mathrm{w}}).\end{aligned}$$

Since $\mathcal{L}u(x_i) = 0$ for $i = m+1, \dots, n$, we find that u approximately satisfies the Poisson equation

$$\mathcal{L}u(x_i) = \sum_{j=1}^m d_j(y_j - \overline{y}_{\mathrm{w}})\delta_{ij} \quad \text{for } i = 1, \dots, n. \tag{2.10}$$

This gives a connection, at a heuristic level, been Laplace equations with hard constraints, and Poisson equations with point sources, for problems with very low label rates. We note that since constant functions are in the kernel of $\mathcal{L}$, $u - \overline{y}_{\mathrm{w}}$ also satisfies (2.10). We also note that the labels, and the constant $\overline{y}_{\mathrm{w}}$, are weighted by the degree d_j, which does not appear in our Poisson learning equation (2.3). We have found that both models give good results, but that (2.3) works slightly better, which is likely due to the rigorous foundation of (2.3) via random walks.

2.4. The Poisson MBO algorithm

Poisson learning provides a robust method for propagating label information that is stable at very low label rates. After applying Poisson learning to propagate labels, we propose a graph-cut method to incrementally adjust the decision boundary so as to improve the label accuracy and account for prior knowledge of class sizes. The graph-cut method we propose is to apply several steps of gradient descent on the graph-cut problem

$$\min_{\substack{u:X\to S_k \\ (u)_X = b}} \left\{ \frac{1}{2}\|\nabla u\|^2_{\ell^2(X^2)} - \mu \sum_{j=1}^m (y_j - \overline{y})\cdot u(x_j) \right\}, \tag{2.11}$$

where $S_k = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\}$, $b \in \mathbb{R}^k$ is given, and $(u)_X := \frac{1}{n}\sum_{i=1}^n u(x_i)$. Since we are restricting $u(x_i) \in S_k$, the term $\frac{1}{2}\|\nabla u\|^2_{\ell^2(X^2)}$ is exactly the graph-cut energy of the classification given by u. Likewise, the components of the average $(u)_X$ represent the fraction of points assigned to each class. The constraint $(u)_X = b$ therefore allows us to incorporate prior knowledge about relative class sizes through the vector $b \in \mathbb{R}^k$, which should have positive entries and sum to one. If there exists $u : X \to S_k$ with $(u)_X = b$, then (2.11) admits a solution, which in general may not be unique.

On its own, the graph-cut problem (2.11) can admit many local minimizers that would yield poor classification results. The phenomenon is similar to the degeneracy in Laplace learning at low label rates, since it is very inexpensive to violate any of the label constraints. Our overall plan is to first use Poisson learning to robustly propagate the labels, and then project onto the constraint set for (2.11) and perform several steps of gradient-descent on (2.11) to improve the classification accuracy. While Poisson learning propagates the labels in a robust way, the cut energy is more suitable for locating the exact decision boundary.

To relax the discrete problem (2.11), we approximate the graph-cut energy with the Ginzburg-Landau approximation

$$\min_{\substack{u\in\ell^2(X) \\ (u)_X = b}} \left\{ \mathrm{GL}_\tau(u) - \mu \sum_{j=1}^m (y_j - \overline{y}) \cdot u(x_j) \right\}, \tag{2.12}$$

where

$$\mathrm{GL}_\tau(u) = \frac{1}{2}\|\nabla u\|^2_{\ell^2(X^2)} + \frac{1}{\tau}\sum_{i=1}^n \prod_{j=1}^k |u(x_i) - \mathbf{e}_j|^2.$$

The Ginzburg-Landau approximation allows $u \in \ell^2(X)$ to take on any real values, instead of discrete values $u \in S_k$, making the approximation (2.12) easier to solve computationally. The graph Ginzburg-Landau approximation GL_τ has been used previously for graph-based semi-supervised learning in (Garcia-Cardona et al., 2014), and other works have rigorously studied how GL_τ approximates graph-cut energies in the scalar setting (Van Gennip & Bertozzi, 2012). Here, we extend the results to the vector multi-class setting.

Theorem 2.4. *Assume G is connected. Let $b \in \mathbb{R}^k$ and assume there exists $u : X \to S_k$ with $(u)_X = b$. For each $\tau > 0$ let u_τ be any solution of (2.12). Then, the sequence $(u_\tau)_\tau$ is precompact in $\ell^2(X)$ and any convergent subsequence u_{τ_m} converges to a solution of the graph-cut problem (2.11) as $\tau_m \to 0$. Furthermore, if the solution $u_0 : X \to S_k$ of (2.11) is unique, then $u_\tau \to u_0$ as $\tau \to 0$.*

Theorem 2.4 indicates that we can replace the graph-cut energy (2.11) with the simpler Ginzburg-Landau approximation (2.12). To descend on the energy (2.12), we use a time-spitting scheme that alternates gradient descent on

$$E_1(u) := \frac{1}{2}\|\nabla u\|^2_{\ell^2(X^2)} - \mu\sum_{j=1}^m (y_j - \overline{y}) \cdot u(x_j),$$

$$\text{and } \; E_2(u) := \frac{1}{\tau}\sum_{i=1}^n \prod_{j=1}^k |u(x_i) - \mathbf{e}_j|^2.$$

The first term E_1 is exactly the energy for Poisson learning (2.8), and gradient descent amounts to the iteration

$$u^{t+1}(x_i) = u^t(x_i) - \mathrm{dt}\left(\mathcal{L}u^t(x_i) - \mu\sum_{j=1}^m (y_j - \overline{y})\delta_{ij}\right).$$

We note that $\mathcal{L}u$ and the source term above both have zero mean value. Hence, the gradient descent equation for E_1 is *volume preserving*, i.e., $(u^{t+1})_X = (u^t)_X$. This would not be true for other fidelity terms, such as an ℓ^2 fidelity, and this volume conservation property plays an important role in ensuring the class size constraint $(u)_X = b$ in (2.12).

Gradient descent on the second term E_2, when $\tau > 0$ is small, amounts to projecting each $u(x_i) \in \mathbb{R}^k$ to the closest label vector $\mathbf{e}_j \in S_k$, while preserving the volume constraint $(u)_X = b$. We approximate this by the following procedure: Let $\mathrm{Proj}_{S_k} : \mathbb{R}^k \to S_k$ be the closest point projection, let $s_1, \dots, s_k > 0$ be positive weights, and set

$$u^{t+1}(x_i) = \mathrm{Proj}_{S_k}(\mathrm{diag}(s_1, \dots, s_k)u^t(x_i)), \qquad (2.13)$$

where $\mathrm{diag}(s_1, \dots, s_k)$ is the diagonal matrix with diagonal entries $s_1, \dots, s_k$. We use a simple gradient descent scheme to choose the weights $s_1, \dots, s_k > 0$ so that the volume constraint $(u^{t+1})_X = b$ holds (see Steps 9-14 in Algorithm 2). By Remark 2.2, this procedure can be viewed as reweighting the point sources in the Poisson equation (2.3) so that the volume constraint holds. In particular, increasing or decreasing s_i grows or shrinks the size of class i.

We note that the work of (Jacobs et al., 2018) provides an alternative way to enforce explicit class balance constraints with a volume constrained MBO method based on auction dynamics. Their method uses a graph-cut based approach with a Voronoi-cell based initialization.

3. Poisson learning algorithms

We now present our proposed Poisson learning algorithms. The Python source code and simulation environment for reproducing our results is available online.[1]

We let $\mathbf{W} = (w_{ij})_{i,j=1}^n$ denote our symmetric weight matrix. We treat all vectors as column vectors, and we let $\mathbb{1}$ and $\mathbf{0}$ denote the all-ones and all-zeros column vectors, respectively, of the appropriate size based on context. We assume that the first m data points $x_1, x_2, \dots, x_m$ are given labels $y_1, y_2, \dots, y_m \in \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\}$, where the standard basis vector $\mathbf{e}_i \in \mathbb{R}^k$ represents the i^{th} class. We encode the class labels in a $k \times m$ matrix $\mathbf{F}$, whose j^{th} column is exactly y_j. Let $\mathbf{b} \in \mathbb{R}^k$ be the vector whose i^{th} entry b_i is the fraction of data points belonging to class i. If this information is not available, we set $\mathbf{b} = \frac{1}{k}\mathbb{1}$.

Poisson learning is summarized in Algorithm 1. The label decision for node i is $\ell_i = \arg\max_{1\le j\le k} \mathbf{U}_{ij}$, and the reweighting in Step 11 implements the label decision (2.4). In all our results, we always set $\mathbf{b} = \frac{1}{k}\mathbb{1}$, so Poisson learning does not use prior knowledge of class sizes (the true value for $\mathbf{b}$ is used in PoissonMBO below). The complexity is $O(TE)$, where E is the number of edges in the graph. We note that before the reweighting in Step 11, the Poisson learning algorithm computes exactly the function u_T defined in (2.7). In view of this, there is little to be gained from running the iterations beyond the *mixing time* of the random walk. This can be recorded within the loop in Steps 8-10 by adding the iteration $\mathbf{p}_{t+1} = \mathbf{W}\mathbf{D}^{-1}\mathbf{p}_t$, where the initial value $\mathbf{p}_0$ is the vector with ones in the positions of all labeled vertices, and zeros elsewhere. Up to a constant, $\mathbf{p}_t$ is the probability distribution of a random walker starting from a random labeled node after t steps. Then the *mixing time* stopping condition is to run the iterations until

$$\|\mathbf{p}_t - \mathbf{p}_\infty\|_\infty \le \varepsilon,$$

where $\mathbf{p}_\infty = \mathbf{W}\mathbb{1}/(\mathbb{1}^T\mathbf{W}\mathbb{1})$ is the invariant distribution. We use this stopping condition with $\varepsilon = 1/n$ in all experiments, which usually takes between 100 and 500 iterations.

[1] Source Code: `https://github.com/jwcalder/GraphLearning`

Algorithm 1 PoissonLearning

```
1:  Input: $\mathbf{W}, \mathbf{F}, \mathbf{b}, T$
2:  Output: $\mathbf{U} \in \mathbb{R}^{n\times k}$
3:  $\mathbf{D} \leftarrow \mathrm{diag}(\mathbf{W}\mathbb{1})$
4:  $\mathbf{L} \leftarrow \mathbf{D} - \mathbf{W}$
5:  $\overline{\mathbf{y}} \leftarrow \frac{1}{m}\mathbf{F}\mathbb{1}$
6:  $\mathbf{B} \leftarrow [\mathbf{F} - \overline{\mathbf{y}}, \mathbf{zeros}(k, n-m)]$
7:  $\mathbf{U} \leftarrow \mathbf{zeros}(n, k)$
8:  for $i = 1$ to $T$ do
9:      $\mathbf{U} \leftarrow \mathbf{U} + \mathbf{D}^{-1}(\mathbf{B}^T - \mathbf{L}\mathbf{U})$
10: end for
11: $\mathbf{U} \leftarrow \mathbf{U} \cdot \mathrm{diag}(\mathbf{b}/\overline{\mathbf{y}})$
```

Algorithm 2 PoissonMBO

```
1:  Input: $\mathbf{W}, \mathbf{F}, \mathbf{b}, T, N_{\mathrm{inner}}, N_{\mathrm{outer}}, \mu > 0$
2:  Output: $\mathbf{U} \in \mathbb{R}^{n\times k}$
3:  $\mathbf{U} \leftarrow \mu \cdot \mathrm{PoissonLearning}(\mathbf{W}, \mathbf{F}, \mathbf{b}, T)$
4:  $\mathrm{d}t \leftarrow 1/\max_{1\le i\le n} \mathbf{D}_{ii}$
5:  for $i = 1$ to $N_{\mathrm{outer}}$ do
6:      for $j = 1$ to $N_{\mathrm{inner}}$ do
7:          $\mathbf{U} \leftarrow \mathbf{U} - \mathrm{d}t\,(\mathbf{L}\mathbf{U} - \mu\mathbf{B}^T)$
8:      end for
9:      $\mathbf{s} \leftarrow \mathbf{ones}(1, k)$
10:     for $j = 1$ to $100$ do
11:         $\widehat{\mathbf{b}} \leftarrow \frac{1}{n}\mathbb{1}^T\mathbf{Proj}_{S_k}(\mathbf{U}\cdot\mathrm{diag}(\mathbf{s}))$
12:         $\mathbf{s} \leftarrow \mathbf{max}(\mathbf{min}(\mathbf{s} + \mathrm{d}\tau\,(\mathbf{b} - \widehat{\mathbf{b}}), s_{max}), s_{min})$
13:     end for
14:     $\mathbf{U} \leftarrow \mathbf{Proj}_{S_k}(\mathbf{U}\cdot\mathrm{diag}(\mathbf{s}))$
15: end for
```

The Poisson MBO algorithm is summarized in Algorithm 2. The matrices $\mathbf{D}$, $\mathbf{L}$ and $\mathbf{B}$ are the same as in Poisson learning, and Poisson MBO requires an additional fidelity parameter μ and two parameters N_{inner} and N_{outer}. In all experiments in this paper, we set $\mu = 1$, $N_{\mathrm{inner}} = 40$ and $N_{\mathrm{outer}} = 20$. Steps 9-14 implement the volume constrained projection described in Section 2.4. We set the time step as $\mathrm{d}\tau = 10$ and set the clipping values in Step 12 to $s_{min} = 0.5$ and $s_{max} = 2$. We tested on datasets with balanced classes, and on datasets with very unbalanced classes, one may wish to enlarge the interval $[s_{min}, s_{max}]$.

The additional complexity of PoissonMBO on top of Poisson learning is $O(N_{\mathrm{inner}}N_{\mathrm{outer}}E)$. On large datasets like MNIST, FashionMNIST and Cifar-10, our Poisson learning implementation in Python takes about 8 seconds to run on a standard laptop computer, and about 1 second with GPU acceleration.[2] The additional 20 iterations of PoissonMBO takes about 2 minutes on a laptop and 30 seconds on a GPU. These computational times do not include the time taken to construct the weight matrix.

4. Experimental Results

We tested Poisson learning on three datasets: MNIST (LeCun et al., 1998), FashionMNIST (Xiao et al., 2017) and Cifar-10 (Krizhevsky et al., 2009). FashionMNIST is a drop-in replacement for MNIST consisting of 10 classes of clothing items. To build good quality graphs, we trained autoencoders to extract important features from the data. For MNIST and FashionMNIST, we used variational autoencoders with 3 fully connected layers of sizes (784,400,20) and (784,400,30), respectively, followed by a symmetrically defined decoder. The autoencoder was trained for 100 epochs on each dataset. The autoencoder architecture, loss, and training, are similar to (Kingma & Welling, 2014). For Cifar-10, we used the AutoEncodingTransformations architecture from (Zhang et al., 2019), with all the default parameters from their paper, and we normalized the features to unit-vectors.

We then constructed a graph over the latent feature space by connecting each image to its K-nearest neighbors with Gaussian weights given by

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

where x_i represents the latent variables for image i, and $d_K(x_i)$ is the distance in the latent space between x_i and its K^{th} nearest neighbor. We used $K = 10$ in all experiments. The weight matrix was then symmetrized by replacing W with $W + W^T$. For Poisson learning, we additionally set $w_{ii} = 0$ for all i. Placing zeros on the diagonal does not change the solution the Poisson learning equation (2.3), but it does accelerate convergence of the iteration in Algorithm 1 by allowing the random walk to propagate faster.

We compare against Laplace learning (2.9) (Zhu et al., 2003), lazy random walks (Zhou & Schölkopf, 2004; Zhou et al., 2004a), multiclass MBO (Garcia-Cardona et al., 2014; Bertozzi & Flenner, 2012), weighted nonlocal Laplacian (WNLL) (Shi et al., 2017), volume constrained MBO (Jacobs et al., 2018), Centered Kernel Method (Mai & Couillet, 2018), sparse label propagation (Jung et al., 2016), and p-Laplace learning (Flores et al., 2019). In the volume constrained MBO method we used exact volume constraints and temperature of $T = 0.1$. In the Centered Kernel Method, we chose α to be 5% larger than the spectral norm of the centered weight matrix. For a baseline reference, we also compared against a nearest neighbor classifier that chooses the label of the closest labeled vertex with respect to the graph geodesic distance. In all experiments, we ran 100 trials randomly choosing which data points are labeled, with the exception of the p-Laplace and sparse label propagation methods, which are slower and were run for 10 trials. The same random label permutations were used for all methods.

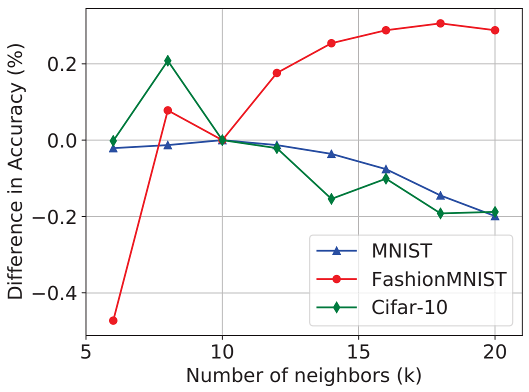
[2]We used an NVIDIA RTX-2070 GPU, and it took 3 seconds to load data to/from the GPU and 1 second to solve Poisson learning.

Table 1. MNIST: Average accuracy scores over 100 trials with standard deviation in brackets.

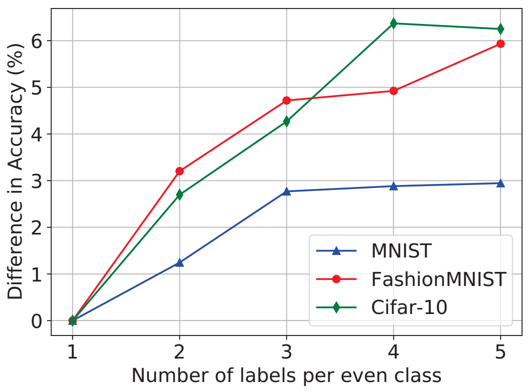
# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP (Zhu et al., 2003)	16.1 (6.2)	28.2 (10.3)	42.0 (12.4)	57.8 (12.3)	69.5 (12.2)
Nearest Neighbor	55.8 (5.1)	65.0 (3.2)	68.9 (3.2)	72.1 (2.8)	74.1 (2.4)
Random Walk (Zhou & Schölkopf, 2004)	66.4 (5.3)	76.2 (3.3)	80.0 (2.7)	82.8 (2.3)	84.5 (2.0)
MBO (Garcia-Cardona et al., 2014)	19.4 (6.2)	29.3 (6.9)	40.2 (7.4)	50.7 (6.0)	59.2 (6.0)
VolumeMBO (Jacobs et al., 2018)	89.9 (7.3)	95.6 (1.9)	96.2 (1.2)	96.6 (0.6)	96.7 (0.6)
WNLL (Shi et al., 2017)	55.8 (15.2)	82.8 (7.6)	90.5 (3.3)	93.6 (1.5)	94.6 (1.1)
Centered Kernel (Mai & Couillet, 2018)	19.1 (1.9)	24.2 (2.3)	28.8 (3.4)	32.6 (4.1)	35.6 (4.6)
Sparse LP (Jung et al., 2016)	14.0 (5.5)	14.0 (4.0)	14.5 (4.0)	18.0 (5.9)	16.2 (4.2)
p-Laplace (Flores et al., 2019)	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)
PoissonMBO	**96.5 (2.6)**	**97.2 (0.1)**	**97.2 (0.1)**	**97.2 (0.1)**	**97.2 (0.1)**

Table 2. FashionMNIST: Average accuracy scores over 100 trials with standard deviation in brackets.

# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP (Zhu et al., 2003)	18.4 (7.3)	32.5 (8.2)	44.0 (8.6)	52.2 (6.2)	57.9 (6.7)
Nearest Neighbor	44.5 (4.2)	50.8 (3.5)	54.6 (3.0)	56.6 (2.5)	58.3 (2.4)
Random Walk (Zhou & Schölkopf, 2004)	49.0 (4.4)	55.6 (3.8)	59.4 (3.0)	61.6 (2.5)	63.4 (2.5)
MBO (Garcia-Cardona et al., 2014)	15.7 (4.1)	20.1 (4.6)	25.7 (4.9)	30.7 (4.9)	34.8 (4.3)
VolumeMBO (Jacobs et al., 2018)	54.7 (5.2)	61.7 (4.4)	66.1 (3.3)	68.5 (2.8)	70.1 (2.8)
WNLL (Shi et al., 2017)	44.6 (7.1)	59.1 (4.7)	64.7 (3.5)	67.4 (3.3)	70.0 (2.8)
Centered Kernel (Mai & Couillet, 2018)	11.8 (0.4)	13.1 (0.7)	14.3 (0.8)	15.2 (0.9)	16.3 (1.1)
Sparse LP (Jung et al., 2016)	14.1 (3.8)	16.5 (2.0)	13.7 (3.3)	13.8 (3.3)	16.1 (2.5)
p-Laplace (Flores et al., 2019)	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)
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 (Zhu et al., 2003)	70.6 (3.1)	76.5 (1.4)	79.2 (0.7)	80.9 (0.5)	**82.3 (0.3)**
Nearest Neighbor	62.9 (1.7)	66.9 (1.1)	70.0 (0.8)	72.5 (0.6)	74.7 (0.4)
Random Walk (Zhou & Schölkopf, 2004)	68.2 (1.6)	72.0 (1.0)	75.0 (0.7)	77.4 (0.5)	79.5 (0.3)
MBO (Garcia-Cardona et al., 2014)	52.7 (4.1)	67.3 (2.0)	75.7 (1.1)	79.6 (0.7)	81.6 (0.4)
VolumeMBO (Jacobs et al., 2018)	74.4 (1.5)	77.4 (1.0)	79.5 (0.7)	**81.0 (0.5)**	82.1 (0.3)
WNLL (Shi et al., 2017)	74.4 (1.6)	77.6 (1.1)	79.4 (0.6)	80.6 (0.4)	81.5 (0.3)
Centered Kernel (Mai & Couillet, 2018)	20.6 (1.5)	27.8 (2.3)	37.9 (2.6)	51.3 (3.3)	64.3 (2.6)
Sparse LP (Jung et al., 2016)	15.2 (2.5)	15.9 (2.0)	14.5 (1.5)	13.8 (1.4)	51.9 (2.1)
p-Laplace (Flores et al., 2019)	73.0 (0.9)	76.2 (0.8)	78.0 (0.3)	79.7 (0.5)	80.9 (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)



(a)



(b)

Figure 2. Accuracy of Poisson Learning for (a) different numbers of neighbors k used to construct the graph and (b) unbalanced training data. In (a) we used 5 labels per class and in (b) we used 1 label per class for the odd numbered classes, and $m = 1, 2, 3, 4, 5$ labels per class for the even numbered classes. Both figures show the difference in accuracy compared to $k = 10$ and balanced training data.

Table 3. Cifar-10: Average accuracy scores over 100 trials with standard deviation in brackets.

# Labels per class	**1**	**2**	**3**	**4**	**5**
Laplace/LP (Zhu et al., 2003)	10.5 (1.3)	12.5 (4.4)	13.1 (3.8)	14.5 (4.7)	18.0 (6.9)
Nearest Neighbor	33.6 (4.4)	37.3 (3.3)	40.3 (3.0)	40.9 (2.7)	42.1 (2.4)
Random Walk (Zhou & Schölkopf, 2004)	37.1 (5.0)	42.1 (3.7)	45.8 (3.4)	47.0 (2.8)	48.8 (2.5)
MBO (Garcia-Cardona et al., 2014)	15.2 (4.1)	20.4 (4.8)	25.9 (4.1)	29.6 (4.3)	34.5 (4.2)
VolumeMBO (Jacobs et al., 2018)	40.3 (8.0)	47.2 (7.1)	52.2 (5.3)	53.3 (4.7)	55.9 (4.0)
WNLL (Shi et al., 2017)	20.8 (6.4)	34.5 (6.2)	42.1 (5.2)	46.1 (4.4)	50.2 (3.5)
Centered Kernel (Mai & Couillet, 2018)	13.8 (1.1)	15.5 (1.2)	17.3 (1.4)	18.8 (1.7)	20.4 (1.6)
Sparse LP (Jung et al., 2016)	10.4 (2.1)	11.1 (1.4)	11.8 (2.1)	12.8 (4.4)	13.6 (3.3)
p-Laplace (Flores et al., 2019)	28.7 (6.6)	39.8 (6.4)	45.7 (2.6)	46.8 (1.7)	50.4 (2.9)
Poisson	41.6 (5.4)	46.9 (4.2)	51.1 (3.4)	52.5 (3.0)	54.5 (3.0)
PoissonMBO	**42.1 (7.0)**	**49.1 (5.3)**	**53.8 (4.4)**	**55.6 (3.7)**	**57.4 (3.4)**
# Labels per class	**10**	**20**	**40**	**80**	**160**
Laplace/LP (Zhu et al., 2003)	31.5 (7.6)	49.7 (4.9)	59.6 (2.2)	63.9 (1.0)	66.7 (0.6)
Nearest Neighbor	45.5 (1.8)	48.7 (1.5)	51.2 (0.8)	53.7 (0.6)	56.1 (0.5)
Random Walk (Zhou & Schölkopf, 2004)	53.5 (1.9)	57.6 (1.1)	60.7 (0.7)	64.0 (0.4)	66.4 (0.3)
MBO (Garcia-Cardona et al., 2014)	47.3 (3.3)	58.6 (2.0)	63.8 (1.1)	66.4 (0.6)	**68.5 (0.4)**
VolumeMBO (Jacobs et al., 2018)	59.5 (2.0)	61.4 (0.9)	62.2 (0.7)	63.3 (0.5)	64.2 (0.4)
WNLL (Shi et al., 2017)	56.7 (2.3)	61.1 (1.1)	63.7 (0.7)	65.4 (0.4)	66.8 (0.3)
Centered Kernel (Mai & Couillet, 2018)	26.5 (2.0)	34.8 (2.1)	43.6 (2.2)	51.8 (2.3)	58.7 (1.5)
Sparse LP (Jung et al., 2016)	16.2 (2.6)	20.3 (1.4)	19.0 (1.1)	19.2 (1.1)	27.1 (1.7)
p-Laplace (Flores et al., 2019)	54.6 (2.2)	60.7 (0.8)	63.9 (0.4)	66.0 (0.5)	67.9 (0.3)
Poisson	58.7 (1.8)	61.9 (1.0)	63.7 (0.7)	65.2 (0.6)	66.5 (0.5)
PoissonMBO	**61.3 (1.8)**	**63.8 (1.0)**	**65.5 (0.6)**	**66.9 (0.5)**	68.3 (0.4)

Tables 1, 2 and 3 show the average accuracy and standard deviation over all 100 trials for various low label rates. We also ran experiments at higher label rates on FashionMNIST and Cifar-10, which are reported in the lower half of their respective tables. We mention that in Tables 1, 2 and 3 the training data is balanced, so $\overline{y} = \frac{1}{10}\mathbb{1}$. Thus, the label decisions (2.4) and (2.2) are equivalent.

We see that in nearly all cases, PoissonMBO outperforms all other methods, with PoissonMBO typically outperforming Poisson learning by a few percentage points. The most drastic improvements are seen at the ultra low label rates, and at the moderate label rates shown in Tables 2 and 3, several other methods perform well. We note that VolumeMBO and PoissonMBO are the only methods that incorporate prior knowledge of class sizes, and are most suitable for direct comparison. Our results can be compared to the clustering results of 67.2% on FashionMNIST (McConville et al., 2019) and 41.2% on Cifar-10 (Ghasedi et al., 2019).

Figure 2(a) shows the accuracy of Poisson learning at 5 labels per class as a function of the number of neighbors K used in constructing the graph, showing that the algorithm is not particularly sensitive to this. Figure 2(b) shows the accuracy of Poisson learning for unbalanced training data. We take 1 label per class for half the classes and $m = 1, 2, 3, 4, 5$ labels per class for the other half. Since the training data is unbalanced, $\overline{y}$ is not a constant vector and the label decision in Step 11 of Algorithm 1 (Poisson Learning) compensates for unbalanced training data. Note that in Figure 2 we plot the difference in accuracy compared to (a) the baseline of $k = 10$ and (b) 1 label per class. In Figure 2 (b), we see an increase in accuracy when only half the classes get additional labels, though the increase is not as large as in Tables 1, 2 and 3 where all classes get additional labels.

5. Conclusion

We proposed a new framework for graph-based semi-supervised learning at very low label rates called *Poisson learning*. The method is efficient and simple to implement. We performed a detailed analysis of Poisson learning, giving random walk and variational interpretations. We also proposed a graph-cut enhancement of Poisson learning, called Poisson MBO, that can give further improvements. We presented numerical results showing that Poisson Learning outperforms all other methods for semi-supervised learning at low label rates on several common datasets.

Acknowledgements: Calder was supported by NSF-DMS grants 1713691,1944925, Cook was supported by a University of Minnesota Grant In Aid award, Thorpe was supported by the European Research Council under the European Union's Horizon 2020 research and innovation programme grant No 777826 (NoMADS) and 647812, and Slepčev was supported by NSF-DMS grant 1814991. The authors thank Matt Jacobs for providing code for volume constrained MBO. The authors also thank the Center for Nonlinear Analysis (CNA) at CMU for its support.

References

Ando, R. K. and Zhang, T. Learning on graph with Laplacian regularization. In *Advances in Neural Information Processing Systems*, pp. 25–32, 2007.

Belkin, M. and Niyogi, P. Using manifold structure for partially labelled classification. In *Advances in Neural Information Processing Systems*, 2002.

Bertozzi, A. L. and Flenner, A. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Modeling & Simulation*, 10(3):1090–1118, 2012.

Calder, J. The game theoretic p-Laplacian and semi-supervised learning with few labels. *Nonlinearity*, 32 (1), 2018.

Calder, J. Consistency of Lipschitz learning with infinite unlabeled data and finite labeled data. *SIAM Journal on Mathematics of Data Science*, 1:780–812, 2019.

Calder, J. and Slepčev, D. Properly-Weighted graph Laplacian for semi-supervised learning. *Applied Mathematics & Optimization*, Dec 2019.

El Alaoui, A., Cheng, X., Ramdas, A., Wainwright, M. J., and Jordan, M. I. Asymptotic behavior of ℓ_p-based Laplacian regularization in semi-supervised learning. In *Conference on Learning Theory*, pp. 879–906, 2016.

Flores, M., Calder, J., and Lerman, G. Algorithms for Lp-based semi-supervised learning on graphs. *arXiv:1901.05031*, 2019.

Garcia-Cardona, C., Merkurjev, E., Bertozzi, A. L., Flenner, A., and Percus, A. G. Multiclass data segmentation using diffuse interface methods on graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36 (8):1600–1613, 2014.

Ghasedi, K., Wang, X., Deng, C., and Huang, H. Balanced self-paced learning for generative adversarial clustering network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4391–4400, 2019.

He, J., Li, M., Zhang, H.-J., Tong, H., and Zhang, C. Manifold-ranking based image retrieval. In *Proceedings of the 12th annual ACM International Conference on Multimedia*, pp. 9–16. ACM, 2004.

Jacobs, M., Merkurjev, E., and Esedoḡlu, S. Auction dynamics: A volume constrained MBO scheme. *Journal of Computational Physics*, 354:288–310, 2018.

Jung, A., Hero III, A. O., Mara, A., and Jahromi, S. Semi-supervised learning via sparse label propagation. *arXiv preprint arXiv:1612.01414*, 2016.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kyng, R., Rao, A., Sachdeva, S., and Spielman, D. A. Algorithms for Lipschitz learning on graphs. In *Conference on Learning Theory*, pp. 1190–1223, 2015.

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

Mai, X. and Couillet, R. Random matrix-inspired improved semi-supervised learning on graphs. In *International Conference on Machine Learning*, 2018.

McConville, R., Santos-Rodriguez, R., Piechocki, R. J., and Craddock, I. N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding. *arXiv preprint arXiv:1908.05968*, 2019.

Nadler, B., Srebro, N., and Zhou, X. Semi-supervised learning with the graph Laplacian: The limit of infinite unlabelled data. *Advances in Neural Information Processing Systems*, 22:1330–1338, 2009.

Shi, Z., Osher, S., and Zhu, W. Weighted nonlocal Laplacian on interpolation from sparse data. *Journal of Scientific Computing*, 73(2-3):1164–1177, 2017.

Slepčev, D. and Thorpe, M. Analysis of p-Laplacian regularization in semisupervised learning. *SIAM Journal on Mathematical Analysis*, 51(3):2085–2120, 2019.

Van Gennip, Y. and Bertozzi, A. L. Gamma-convergence of graph Ginzburg-Landau functionals. *Advances in Differential Equations*, 17(11/12):1115–1180, 2012.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xu, B., Bu, J., Chen, C., Cai, D., He, X., Liu, W., and Luo, J. Efficient manifold ranking for image retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 525–534. ACM, 2011.

Yang, C., Zhang, L., Lu, H., Ruan, X., and Yang, M.-H. Saliency detection via graph-based manifold ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3166–3173, 2013.

Zhang, L., Qi, G.-J., Wang, L., and Luo, J. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pp. 2547–2555, 2019.

Zhou, D. and Schölkopf, B. Learning from labeled and unlabeled data using random walks. In *Joint Pattern Recognition Symposium*, pp. 237–244. Springer, 2004.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pp. 321–328, 2004a.

Zhou, D., Weston, J., Gretton, A., Bousquet, O., and Schölkopf, B. Ranking on data manifolds. In *Advances in Neural Information Processing Systems*, pp. 169–176, 2004b.

Zhou, D., Huang, J., and Schölkopf, B. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 1036–1043. ACM, 2005.

Zhou, X. and Belkin, M. Semi-supervised learning by higher order regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 892–900, 2011.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Zhu, X., Lafferty, J., and Rosenfeld, R. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, 2005.